mcsMQTT

HomeSeer HS3/HS4 Plug-in

HomeSeer HS4 Plug-in

Michael McSharry

March 11, 2024

Version 6.15.1.x

Contents

List of Figures

List of Tables

1 Introduction

mcsMQTT is a Plug-in that understands MQTT and other protocols and conveniently bridges the MQTT environment with the HS3 environment. It can be installed with HS3 on either Windows or Linux. It maintains a local SQLite database to remember message setup, history and the basis for graphing. It also maintains a configuration file for general HS3/user configuration settings. It creates HS3 Devices, Triggers and Actions. It responds to DeviceValue and DeviceString changes from all HS3 devices. It publishes MQTT messages with ability to control Quality Of Service and MQTT broker retention attributes.

The design premise of mcsMQTT is that HS3 is the primary automation node in a typical setup. This allows mcsMQTT to leverage advantages of other protocols such as COAP/RESTful to provide discovery and transaction-oriented messaging while still retaining the advantages of the Quality Of Service that is important for devices that may not always be connected or the connection reliability is poor.

This Plug-in is provided as a service to the community. Others have contributed in various ways with testing, feedback, how-to documentation, encouragement and source code including the SSL implementation by vasrc.

2 Installation

Installation can be performed using the HS Updater facility from the Plugins or Interaces page of HS.

Often pre-release updater packages are available and use the Uploader Override process to intall.

To side-load a plugin the following process is used for HS4
1. Download the zip file that is in the HS4 updater format.
(e.g. http://mcsSprinklers.com/HSPI_mcsMQTT_5_12_0_0.zip)
2. Place the download in the HS4 folder.
3. Unzip the file updater_override.json and put it in the same HS4 folder.
4. With browser navigate the the HS4 plugin menu, Add option. Only HS4 plugin available will be mcsMQTT. Select it. (I do not recall if you need to deselect the current mcsMQTT version before this or not)
5. Remove updater_override.json to restore normal Updater operation.

For HS3 it is paraphrased from the HS3 SDK
1. Download the zip file that is in the HS3 updater
format. (e.g. http://mcsSprinklers.com/mcsMQTT_5_12_0_0.zip)
2. Place your package installation ZIP file into the \Updates3\Zips folder.
3. Unzip updater_override.txt. I believe it goes in the HS3 folder.
3. Now go to the menu Plugins->Manage and click on the Refresh (Update Listing) button so it finds your updater_override.txt file and it should list your package
4. Remove updater_override.txt file to restore normal Updater operation.

In addition, incremental builds are available that contain only changed file. These are installed by unzipping to overwrite the file of the same name. These are typically in \bin\mcsMQTT or \html\mcsMQTT folders.

3 Environment & Architecture

3.1 MQTT Environment

MQTT is a lightweight protocol that came about as a result of the Internet Of Things (IOT) initiatives. Communications use TCP over IP with a publish and subscribe model. The environment is widely supported on both Linux and Windows.

A typical setup has a single MQTT broker who is responsible for keeping track of all subscriptions and assures that published messages are delivered to those that subscribe to the Topic of those messages. MOSQUITTO is the standard broker. Normally it is run on something like RPi, Linux or Windows servers.

mcsMQTT will also provide the function of MQTT Broker if desired. This is the default. This configuration is done on mcsMQTT Page Broker Tab MQTT Broker Name or IP Address text box. When the text box is left empty then no connection to a MQTT Broker will be attempted. If selection is made for use of internal Broker, then the IP will show as 127.0.0.1. It is recommended to use an external MQTT Broker that is running on a network computer with high availability. When the internal MQTT Broker is used, then it will only be running when mcsMQTT is running and if there are other network clients that use MQTT these clients will no longer be able to communicate when mcsMQTT is not running. Any combination of internal and external Brokers can be specified with up to six allowed. Semicolon is used in the setup to separate the IP of each.

mcsMQTT is a MQTT client. By default, it subscribes to all or a selected subset of messages serviced by the MQTT broker and it publishes that which is setup by the user in the form of HS Devices and Event Actions. The mcsMQTT user identifies Topics of interest to HS and places the selected message Payload in HS Devices or uses the Topic as an Event Trigger.

The MQTT broker being used by mcsMQTT is specified in the setup (/MQTT) browser page of the HS Plug-in. This can be seen in Figure 40. A variety of options are available to identify the broker. One is to use the IP address such as “192.168.1.100”. Another is to identify its network name. This is shown in -Figure 40 example as “MQTT” as a dedicated RPi is being used for this purpose. The third option is an internet address such as “MyDomainName.com”. Port 1883 is being used for MQTT communications so the WAN/LAN router will need to be setup to allow port 1883 to be serviced from the WAN wherever the broker is located. In all cases the firewall protection will need to be setup to allow free communication between the broker and all clients.

The MQTT broker can be setup to restrict client access with username/password protection. SSL protocol is typically another broker option. mcsMQTT supports the user of username/password and/or SSL. If the MQTT broker is not using usernames/passwords then the setup of mcsMQTT will leave these fields blank. 4-21-2022 12:30

3.2 mcsMQTT Plug-in Architecture

mcsMQTT is a Plug-in that complies with the HS3 and HS4 API for Devices, Event Triggers, Event Conditions and Event Actions. Multiple browser pages are available from which user options can be specified. The management associated with items that use MQTT protocol is done from the MQTT menu selection. Other protocols and speciality functions such as IR, location tacking and others are done from other browser pages available on the selection menu. The Local page contains setup for

capabilities that are based on communications on the local network. The Cloud page contains setup for capabilities that often connect to the internet to external servers to support the desired functionality.

No mater the source (MQTT, Local or Cloud) the information is decoded into individual endpoints and made available on the MQTT page, Association tab from where the integration with HS can be specified. This integration normally is initiated with use of the "A"ssociate column checkbox and/or "L"ongTerm (InfluxDB, MySQL, SQL Server) or "S"hortTerm to SQLite checkbox. Further customization is then done on the Edit tab or the MQTT Page which can be accessed via hyperlink of the row sequence number or Ref button.

Devices are created and deleted based upon association with MQTT Topics.

- Created devices select their Location and Name properties based upon MQTT Topic with the lowest part of the Topic hierarchy used for Name and the next up used for Location. Alternately, Location properties can be specified on the MQTT Page Client Tab.
- DeviceTypeString is set based upon the receive or transmit nature of the MQTT Topic. Devices created based upon a received MQTT Topic will be of type "MQTT_Receive". "MQTT_Transmit" is used for devices that will publish a MQTT Topic that is not associated with a received Topic.
- The Device MISC properties will be set to SHOW_VALUES and cleared to not perform AUTO_VOICE_COMMAND. These can be edited from the Edit tab.
- The Device Address property is set to the received MQTT topic.
- DeviceVSP are setup depending upon the nature of the Payload with button-orientation for those topics that appear to have two or three discrete statuses (e.g. ON/OFF, OPEN/CLOSED) Payloads and 32-bit integer ranges for numbers and list for enumerated text. These properties are setup when a subscription Topic is associated and then again when a publish Topic is selected. A Device remains status-only until a publish Topic is selected.
- DeviceVGP are used for the buttons with the graphics being the standard images used by HS3.
- The CAPI interface supports buttons, numeric and string types of control. The DeviceValue or DeviceString will store the Numeric and Text Payload, respectively.
- The use of DeviceValue or DeviceString by mcsMQTT will be based upon a combination of the Edit tab Control/Status UI and the radio to select between the two. A Control/Status UI of Text will use DeviceString. All others will use DeviceValue. The radio can override this default.

Information about Topic/Device relationships is maintained in the mcsMQTT database at Data\mcsMQTT\mcsMQTT.db. Information about modes and settings is maintained in Config\mcsMQTT.ini. If scripts are used then they are located in the \scripts subfolder. Backups of the Data, Config, and scripts subfolders as well as the HS folders related to events and devices are made daily if a backup location has been specified. If backups are enabled then it is advised that they be made to a drive different than the one used by HS so if physical damage to main drive occurs the backup will still be available and so the backup will not eat away at available space on the HS drive.

Startup of the plug-in establishes a connection with HS and makes HS aware of the capabilities provided by the plugin. The plug-in then performs a connection to the MQTT Broker or spawns its own MQTT Broker is one has not been specified. It also does its bookkeeping of MQTT topics and HS Devices that have been previously observed. During this time, it will queue any inbound requests to send MQTT messages as well as defer rendering of browser pages until it completes initialization.

mcsMQTT tries to provide a capability that best suits a specific user's needs. This is configured on the Client Tab of the mcsMQTT browser page. In its most basic mode, it will allow a user to manually define a Topic-Device relationship with a set of text boxes and supporting checkboxes or radio buttons for the relationship's characteristics. In its most user-friendly configuration, it will discover all Topics and Devices available and let the relationship be defined by a single checkbox entry. The balance between modes is driven primarily based upon performance capability of the host computer to support the features provided by mcsMQTT in a responsive manner.

The HS Device Management page provides the ability to alter some of these settings after device creation. For example, Location and Name properties can be changed to better suit the user's intent of the MQTT Topic, or a displayed Device's status can be suffixed with "seconds", "F", "C" or whatever is best to put the number in the proper context.

4 Quick Start

This section is intended to show typical MQTT use cases that will get a user going as quickly as possible. Subsequent Sections 5 and 6 provide a more complete reference for sending and receiving MQTT messages, respectively. Other reference subjects follow these.

mcsMQTT supports many special functions where most of the setup is automatic and no subsequent user configuration is needed for integration with HS. For example, messages from WLED, a Shelly device, Sense Energy, Pentair poolController, and others will automatically create HS devices.

The normal case, however, is that the decoded message is shown on the MQTT Page Association tab and the user then uses the "A"ssociate checkbox to create the HS Device and then sometimes the Edit tab to customize it.

There is a middle-ground when HomeAssistant Discovery protocol is used. This protocol consists of messages delivered by a widget that describe what the widget does. For example, a light with brightness control. After receiving this description and the widget then communicates its state the plug-in will create the HS device with on, off and slider controls. Some, but not all, widgets can be configured to use HomeAssistant Discovery protocol.

4.1 Q&A

The Q&A is based upon default settings. While other techniques are often available the objective here is to describe the most common usages. It is assumed that the Association tab filters are setup to not restrict what is displayed in the Association Table or that they have been setup in a manner that does not hinder the intended capability.

4.1.1 How do I get started with MQTT

Not much happens with MQTT until a connection to a MQTT broker is established. The broker can be on the same, different network computer or an internal one built into mcsMQTT. The Broker Tab in the MQTT Broker Operations Table is used to identify the broker IP or network name. Some brokers are setup with username/password protection. If yours is, then these two fields also need to be entered.

Assuming the MQTT broker accepts the connection to mcsMQTT, mcsMQTT will subscribe to all Topics that the broker manages. mcsMQTT will likely receive some messages right away and then receive others as new Topics are published by other widgets (other MQTT clients).

The MQTT Page Association Tab with Topic Filter T1 set to MQTT is good to view to confirm that a broker connection exists. It will also show the last message sent and received as well as statistics about communications overall. Initially mcsMQTT will send a message to announce that it has established a connection to the broker. This will be a message on "Topic xxx/mcsMQTT/LWT" with payload content of "Online", where xxx is the computer's name. The LWT topic stands for Last Will and Testament. As part of the connection to the broker, mcsMQTT tells the broker that its Last Will and Testament is "Offline". If at some time in the future mcsMQTT becomes disconnected from the broker then the broker will deliverer the "Offline" message to all other MQTT clients to let them know that mcsMQTT is no longer communicating.

The Association Tab can be used to observe all the Topics and Payloads delivered by the broker. Note that the Association table is not dynamically built as new Topics are received. The "Show Selected Associations" button needs to be used to build a new table. Payloads in the existing table will be updated as they change from the broker every ten seconds.

4.1.2 How do I view the MQTT Topic Payload in HS Device

The published topics are available on the Associations Tab and will be shown in green rows. Use the "A" column checkbox to create a HS Device. When numeric Payloads are received the HS Device Value will be updated. When non-numeric Payloads are received then either the Device Value will be updated based upon Value Status Pairs that show relationship between text and a number or will be stored in HS Device String.

4.1.3 How do I setup a Command/Response Device so HS can control MQTT item and show its status

The remote item's status via MQTT will be available on the Association Tab. When "A" is used then the status will be available in the HS Device. See Section 4.1.2. The Topic column will provide a text box in which the Topic that HS should publish to control the remote item via MQTT. The specified Topic will be published with the Value as Payload when the HS Device Value changes.

If the Device has been setup using Value Status Pairs (VSP) then either the Status or the Value will be placed in the Payload to be sent. By default, it is the DeviceValue, but can be changed to use the Status or Label by using the substitution variable $$STATUS: or $$LABEL: in the Payload template that is available on the Edit tab.

Note that when the Topic is specified then the HS Device Management UI will create controls to allow the Topic to be published through its UI. If no publish Topic is specified then the device will be a status-only one.

4.1.4 How do I control an existing HS Device with a MQTT Topic

The pink rows in the Association table are the HS Devices that have an interface property that is not "mcsMQTT". These are the non-plugin Devices. When the "A" checkbox is used to map the HS Device to MQTT then two text boxes will appear in the Topic column. One is a command topic. The command Topic being published by the remote item that desires to control the HS Device is entered here. The next time the MQTT Setup Page is drawn this row will show in a blue rather than pink or green color to indicate its hybrid nature. If the remote item is expecting status returned as to the state of the HS Device then the second status Topic text box is used to enter the Topic that will be used to publish status.

4.1.5 I want to subscribe to a Topic, but the Topic has not yet been published through the MQTT Broker

The Edit tab is used add any subscription topic. Enter the new subscription Topic "MQTT Subscribe Topic" row in the green table. Other properties of the subscription can be set at the same time in this table. A common use for this is to subscribe to the special case broker $SYS topics that are otherwise not visible.

4.1.6 The Device Management UI is not showing what I want for MQTT Devices, how do I change it

The UI is determined at the time a status Topic is entered for an "A"ssociated subscription. It is based upon the Payloads that have been received prior to association. If only numbers have been received then the UI will be a number box. If a set of text Payloads are observed then mcsMQTT will setup a selector UI to select among those that have been observed. If it appears that two-state text has been received (e.g. On/Off, Open/Closed, Online/Offline etc.) then a button control will be provided for each of the states.

If these defaults are not really what you want then the Edit tab is used to change them. Start by clicking on the button on the Association tab that has the Ref number or entering the MQTT Subscribe Topic or Device reference at the top of the Edit tab. The remainder of the table will be populated with the properties of this subscription topic. Select the desired radio button on row for HS Device Control/Status UI.

Control/Status UI of Button will show buttons on the HS Device Management page. List type will show a pulldown. Both of these use the VSP associations to manage DeviceValue and Show DeviceStatus.

Control/Status UI of Number or NumberChange will show a numeric text box for control on the HS Device Management page. It uses DeviceValue.

Control/Status UI of Text will show a text entry box for control and the DeviceString.

Control/Status UI of ColorPicker or ColorXy will show a color picker for control and use DeviceValue that will publish a value in #RRGGBB hex format.

Control/Status UI of Slider or HSB will produce a slider control. HSB will also produce a color picker

4.1.7 I see Dim for HS Device Status, how do I remove or change 'Dim'

The HS3 Device Management Page, Value-Status-Pair tab allows editing of prefix and suffix to be added to a numeric status display. Dim is the HS default but can be edited to anything desired including totally removed. A similar capability exists on the HS4 Devices Page, Status/Graphics tab.

4.1.8 Payload numbers contains periods for decimal. I need them to be comma

The Association Tab has a column for Regular Expression processing of incoming Payloads prior to the Payload use within HS. For this particular case the Regular Expression match pattern is escape period (i.e. "\.") and the replace pattern is comma (i.e. ",").

It also has an Expression text box from which expressions can be entered using functions shown in Table 3 and replacement variables in Table 2. The Expression "Replace($$PAYLOAD:,".","","")" will do substitution of period for a comma.

4.1.9 How do I know if I am communicating with MQTT broker

The connection status is shown on the MQTT Page, Client Tab. The Association Tab with Topic Filter T1 set to MQTT provides visibility into the connection status, last received and published messages and timings associated with the subscriptions. Associations can be made to these statistics to facilitate their use in HS Events.

4.1.10 Where do I look when things go wrong

mcsMQTT maintains a text file in \Data\mcsMQTT\mcsMQTT_Debug.txt that contains some general startup trace information and other things of interest such as page rendering times. More information is placed in this file if the General Tab, mcsMQTT Management Section, has the Enable General Debug checkbox checked. This information is geared for developer and not user, but, in general will be needed when support is needed. The file can be accessed directly or uploaded in zip format from the button on the General Tab. For more severe issues, the HS Event Log will contain notifications provided by mcsMQTT.

4.1.11 How do I know if a client has stopped publishing MQTT messages

Event triggers (MQTT Timeout) can be setup from HS Event page to trigger the event when a specified period of time has elapsed without a topic being received. A provision of the MQTT protocol is for Last Will and Testament (LWT) that may or may not be used by the client. If it is, then the LWT Topic for this client can be "A"ssociated as a HS Device and then trigger an event when this Device shows the LWT Payload for that client which is often "Offline". There is also an LWT provision for when a client connects to the broker.

4.1.12 How do I publish a MQTT message when some event has been triggered in HS

An Event Action can be defined for this event that will publish a specific Topic in response to that event. Substitution variables (Table 2) can be used in the Action's Payload that substitute a set of defined items defined within HS, such as DeviceValue, time, etc.

Another approach is to have previously setup a publish Topic within a HS device (See 4.1.3) and then in the Event Action to set this Device to the Value desired to command the MQTT topic to be transmitted.

4.1.13 How do I chart the time history of a topic's payload

Two options exist for collection data in a database for later viewing. One is to store MQTT Topic and Payload in an SQLite database. The other is to store HS DeviceValue in either InfluxDB, SQLite or MS SQL Server database. See 4.1.14 for this second approach. The selection of one vs. the other depends if one is trying to analyzer MQTT message history or history of changes in a specific endpoint's value.

For the MQTT Topic and Payload the setup is done from the MQTT Page History tab. This allows the retention to be specified and the types of MQTT messages will be saved.

A chart can be prepared in one of three methods. The easiest is from Association tab by clicking on the Payload hyperlink. A more customized chart is made from the Chart tab where specific topic and payload parameters are entered and the chart shown at the bottom of the tab. The third is with a HTTP request to the HS Server URL where the parameters are part of the request. See Section 9.

4.1.14 How do I chart the time history of a HS device

When HS DeviceValues are being saved in either SQLite or InfluxDB then the selection of the specific HS Device Reference is done from the MQTT Page Association tab. The "L"ongTerm column and "S"hortTerm are used to select the Device and the target database. SQLite is oriented to short-tem data collection while the network databases are oriented to longer term retention. Short-term storage also has MQTT Page History tab setup to specify the scope of the data collection.

A chart can be prepared in one of three methods. The easiest is from Association tab by clicking on the LastDate hyperlink that will be available for any row where the S(ortTerm) or L(ongTerm) checkbox is checked. A more customized chart is made from the Chart tab where specific topic and payload parameters are entered and the chart shown at the bottom of the tab. The third is with a HTTP request to the HS Server URL where the parameters are part of the request. See Section 9.

4.1.15 How do I change Payload temperature from Centigrade to Fahrenheit

A transformation can be performed on a payload before it is stored into a HS Device. For textual transformation regular expressions can be used. For some textual and all numeric transformation then, numeric expressions can be used that contain functions shown in Table 3 and replacement variables in Table 2.

In this case it will be a numeric expression "$$PAYLOAD: * 1.8 + 32" that is entered on the Expression row of the Edit tab.

4.1.16 My MQTT payload is wattage rate, but I want HS to provide daily wattage use

mcsMQTT can integrate or take the derivative of a MQTT payload input. The integral/accumulation transformation is used if a rate is being provided in the payload. This is done on the Edit tab by checking the "Create a HS Accum Device" checkbox and also using the checkbox to reset the accumulation at midnight. HS Device will show the wattage rate and a second one will show the wattage. Had the sensor been providing the wattage and the desired transformation was to see the wattage rate then the "Create a HS Rate Device" new radio would be used.

When a integral or rate option is selected a pseudo MQTT topic is created with the Topic that is suffixed with "-Accum" or "-Rate". It will show up on the Association Tab from where it can be selected for history data collection.

4.1.17 How do I easily initialize an IOT device with one-tiime configuration messages

The MQTT Page Pub List tab exists to publish a set of messages on demand. The messages that are needed to define the IOT device configuration (e.g. LoRa frequency or irrigation schedule) are place in a text file of type ".pub" located in the \Data\mcsMQTT folder. Each row of the text file will contain a message of format Topic=Payload. The Execute Publication List button is used to send the set of messages.

4.1.18 I have been experimenting and have topics that will never be used again. How do I permanently remove them

The Obsolete function within mcsMQTT is used to remove Topics that have become obsolete. The function can be done on an endpoint-by-endpoint basis on the Association tab or with a Topic template on the General Tab. See 4.1.26

4.1.19 How do I group devices in HS Device Management display

Devices with related information are often desirable to group so always show together in the HS Device Management page. The user can group by whatever organization a user desires on a device-by-device basis. In this case the Edit Tab has “Grouping Parent Ref” where the user can specify a parent device reference and the MQTT device will become the child of this parent.

mcsMQTT will by default group JSON payload items into a parent device of the topic. Since all children are required to have parents in HS4 it is not possible to leave a child without a parent grouping.

If a parent device does not exist then one can be created by entering a negative number for the grouping ref. mcsMQTT will then create a parent device rather than using the number as the identification of the parent.

4.1.20 How do I automatically associate sets of MQTT topics to non-plugin devices

A wildcard capability exists for incoming topics that conform to a pattern and the HS device reference is contained in the topic. For example, a user has all lighting already setup with HS control and wants to also provide a method to control via MQTT with minimum setup within mcsMQTT. On the Client tab in section for Inbound (Subscription) Management is a key “Wildcard Non-Plug-in Control Template” that will accept the pattern of a control Topic where part of the pattern includes any of the substitution patterns described in Table 2. An example is shown in Figure 1 where the topic to control an HS device will start with “test/” and end with “/.cmnd” and the device reference will be in the middle (e.g. test/123/cmnd). When a topic is received that matches this pattern and the HS device exists then mcsMQTT will associate the topic and device automatically. Other wildcard patterns such as “Command/$$COMPUTER:/$$FLOOR:/$$NAME:” could be uses as well to uniquely identify each HS device.

When multiple patterns are desired a comma between each is used. For example, if subscribing to “HS” or “HS3” topics then the template would be specified as “HS/#,HS3/#”.

Normally status feedback when this HS device changes is desired for these devices. The status topic and the payload templates are setup in the Outbound (Publish) Management section which is also shown in Figure 1. Again “$$REF:” is used as the wildcard for the HS device reference in the Topic. Other parts of the Topic can be simple text or other substitution parameters as described in Table 2.

The payload in this status message will normally be either $$LABEL: to indicate to publish the same as what HS shows for a label of a Value Status Pair. It could also be $$STATUS: or $$$VSP: to use the VSP mapping setup on the mcsMQTT Edit tab. This default payload template can be left blank for the default payload of the DeviceValue.

After the command topic is received it is possible to use the Edit tab for the Topic/Device and change any of the default behaviors on a Topic/Device by Topic/Device basis.

It is also possible to create associations for all HS non-plug-in devices immediately and not depend upon receipt of a command topic. In this case both a wildcard subscribe/control topic and a publish default topic template is needed. The radio button next to the subscribe wildcard is changed to the immediate option. If new HS devices are created in the future, then they will be automatically associated with MQTT topics when mcsMQTT is started or when the Enumerate button on the General Tab is used.

If changes are desired in individual device Topics that vary from the default wildcard or template then they can be done from the Edit Tab.

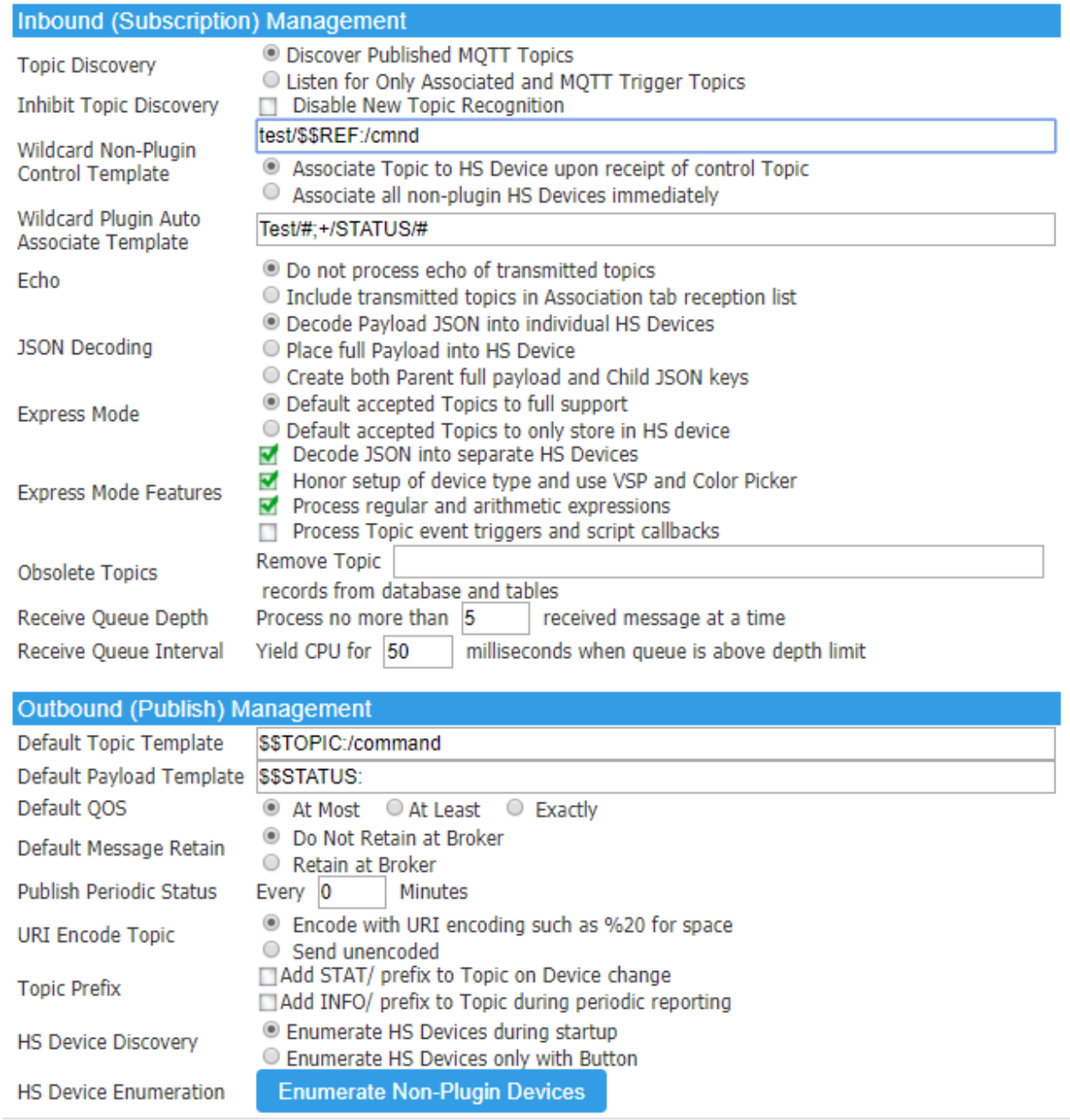


Figure 1 Wildcard Topic Setup

4.1.21 How do I automatically create HS devices based upon MQTT Topics

There are a set of topics that mcsMQTT will recognize and automatically create HS devices. These are:

"shellies/#" for Shelly line of products,
"wled/#" for WLED light control,
"pool" for poolController.js for Pentair,
"owntracks" for OwnTracks and NextTracks Apps,
"voicemonkey" for Echo TTS,
"Broadlink" for Broadlink IR/RF,
"Daikin" for Daikin AC control,
"EcoNet" for Rheem EcoNet
"Yolink" for Yolink line of products,
"sense" for Sense Energy
"espresense" for Espresense room location
"HS" for HS and Plugins
"MQTT" for statistics collected by mcsMQTT

"URL" for IPs setup on Cloud Page URL tab,
URL UDP Port 32101 for Coulisse B.V. Motion-Blinds.com Blinds Control
"URL/api.emporiaenergy.com" for Emporia Vue
"URL/Epson" for Epson projector

No additional effort is needed by the uses when these topics are present. For some the auto create capability is enabled from the Client tab, Inbound Management "Enable Auto Device Creation" checkbox.

mcsMQTT recognizes the HomeAssistant discovery protocol. These are topics starting with "homeassistant/" and ending with "/config". This protocol is used by MQTT clients to disclose the nature of the devices it supports. mcsMQTT will collect this disclosure information and then if any topic matching it is used it will automatically create HS devices for status and control.

mcsMQTT will also use this discovery protocol to disclose any non-plugin devices that have been setup for MQTT. The discovery is published with use of the General tab, Outbound Management, "Publish HomeAssistant Discovery" button. While it will help other clients to understand the nature of what has been setup in HS, it is hampered by a poor functional model of devices within HS so the disclosure may not result in the desired devices being created by other clients.

The above approaches provide the most complete automatic device creation as they handle all the properties such as icons and control UI type appropriate for the device. A more generic, and less rich, mechanism is described in the remainder of this section.

mcsMQTT will collect all topics for viewing in the Association tab and the user is able to create HS devices by using the "A"ssociate checkbox. This opt-in selection can be changed into an opt-out selection by using plug-inTopic wildcard template for the Topics that will automatically have HS devices created. This template is found on the Inboud Management section of the Client tab. If every topic is to

create an HS device then the template is “#”. If it is to create HS devices for all topics with “RESULT” in the second position then the template would be “+/RESULT/#”. The template uses standard MQTT Topic wildcarding. Anything that does match the wildcard template will obey the opt-in rules. Devices that were auto-created and are not desired can be removed with “A”ssociate textbox on the Association tab. Auto-Association only applies to the initial detection of a topic.

When Auto-Association is used then it may be useful to also set the Topic Template in the Outbound Management section of the Client Tab. When this template is not blank then any HS device that is created will have a publish Topic setup and either Buttons or Number text boxes setup with the HS device. For example, Shelly devices suffix the Topic with “/command” to indicate that the topic is commanding the Shelly device. The Topic template in this case would be “$$Topic:/command”. Tasmota devices use “/cmnd” placed either as a prefix or before the last segment of the Topic. If the substitution is “$$TASMOTACMND:” then mcsMQTT will evaluate the Topic and put the “/cmnd” in the proper position. For example, “Switch/Power” becomes “Switch/cmnd/Power”.

If there are multiple sets of Topics that are whitelisted then each should be separated by semicolon. This then servers as an OR function to Auto-Associate different sets of Topics, but not all topics.

Inbound (Subscription) Management

Topic Discovery	◉ Discover All Published MQTT Topics ○ Listen for Only Associated and MQTT Trigger Topics ○ Listen for Only using the Wildcared Template below
Inhibit Topic Discovery	☐ Disable New Topic Recognition
Enable Auto Device Creation	☑ Enable Auto Creation
Wildcard Non-Plugin Control Template	◉ Associate Topic to HS Device upon receipt of control Topic ○ Associate all non-plugin HS Devices immediately
Wildcard Plugin Auto Associate Template	;;
Default HS Device Location	◉ Use Loc2 (Floor) & Loc (Room) based upon MQTT Topic ○ Use Default Loc2 & Loc
Default HS Device MISC	◉ Only update Last Time when value changes ○ Always update Last Time
Default HS Parent Device	◉ Create HS parent device based upon MQTT Topic ○ Use existing MQTT parent device
Echo	◉ Do not process echo of transmitted topics ○ Include transmitted topics in Association tab reception list
Express Mode	◉ Default accepted Topics to full support ○ Default accepted Topics to only store in HS device
Express Mode Features	☑ Decode JSON into separate HS Devices ☑ Honor setup of device type and use VSP and Color Picker ☑ Process regular and arithmetic expressions ☐ Process Topic event triggers and script callbacks
Remove Obsolete Topics	Remove Topic records from database and tables
Obsolete Unassociated on Shutdown	☐ Remove Unassociated records from database and tables on shutdown
Reject Topic Templates	Reject Topics matching template Greenhouse/RESULT
Receive Queue Depth	5 Limit when CPU throttling starts 100 Limit when messages discarded
Receive Queue Interval	Yield CPU for 50 milliseconds when queue is above depth limit

Figure 2 Auto-Association (Opt-out) Setup Example

4.1.22 How do I publish HS device changes without explicit association to MQTT topics

When the objective is to primarily make HS a source of information and to publish all device changes that are occurring in HS then it becomes desirable to do this without explicitly associating each device with a MQTT Topic. The “Publish HS Device Changes” option can be used in conjunction with the default Topic and Payload templates that are available on the Client Tab.

In the example shown in Figure 3 all changes will be published using the floor, room and name properties of the device for the topic and the value, status and ref properties as a JSON payload. The templates can be changed as desired. The option also exists to only publish topics for devices that have not been created by mcsMQTT.

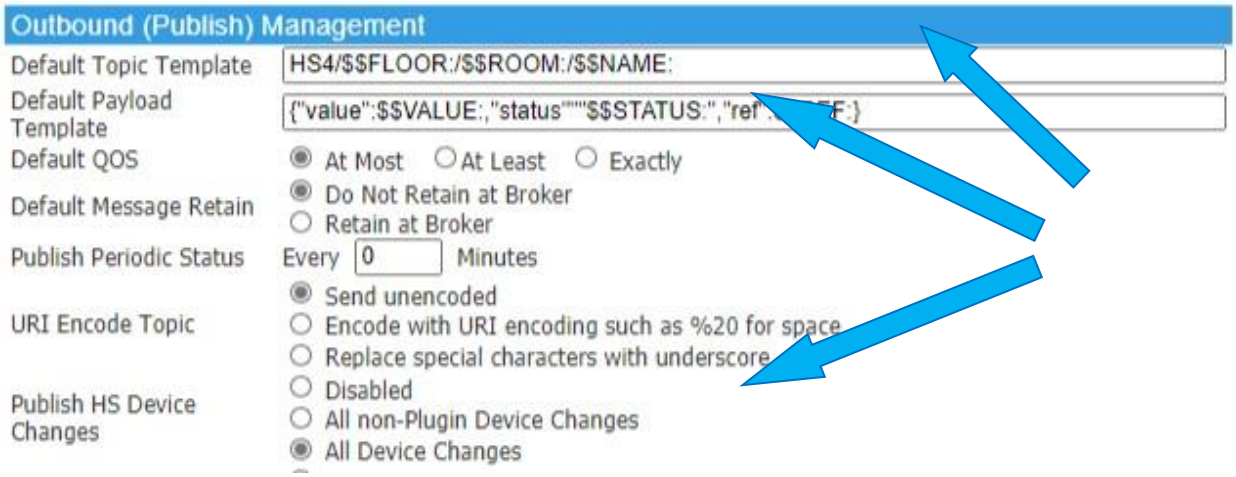


Figure 3 Publish Without Association

4.1.23 How do I setup device with different status and control payloads

The desire is to control a door lock where the MQTT payload for status is LOCKED and UNLOCKED and the HS control buttons show lock=100 and unlock=0.

mcsMQTT has been observing the MQTT status of the lock device and has made a list of LOCKED and UNLOCKED as known payload content of the device. When the subscribed topic is associated and HS device created then the Value Status Pairs will be created such that DeviceValues of 0 and 1 are assigned to UNLOCKED and LOCKED respectively.

If your lock device responds to “lock” and “unlock” payloads then on the Edit Tab leave Payload template blank. If it responds to a number such as 100 and 0 then use “$$VALUE:” for the Payload template and edit as necessary the VSP in mcsMQTT to have 0 and 100 for the two states rather than the default 0 and 1.

To create the “lock” and “unlock” command buttons the HS Device Management page is used. Click on the name of the mcsMQTT device that now contains the LOCKED and UNLOCKED for its status states. Select the Status Graphic tab. Click on the Add New Single Value and enter 0 and “unlock” for the value and status respectively. Select control for the Status-control. Do the same for the “lock” control with a value of 100. Note that the LOCKED and UNLOCKED states should be status for Status-Control so it they are different then change them now.

This should result in 2 buttons or pull-down selections on the GUI. You may need to edit the Value Graphics Pair to get the graphics you want for the two control buttons.

When you click a button the MQTT message will have a payload of 0 or 100 if the payload template is $$VALUE: or “unlock” or “lock” if the payload template is $$LABEL:. When the node reports its status change it will return something like UNLOCKED for which mcsMQTT will store a value of 0 in HS3. HS3 will show UNLOCKED in the status for the device on the browser page. Note that $$VSP: will normally be the equivalent of $$LABEL: and $$STATUS: will use the Edit Tab status field of the VSP for the replacement.

Note that when the desire is to only change the status text from what is received in the payload then the Edit Tab provision for editing VSPs has the advantage of not being later overwritten by mcsMQTT is the device properties are later changed in the Edit Tab. See Section 18.2.3.

4.1.24 How do I conveniently control a colored light

Colored LED lights or light strips come in various forms with some variance in the control capabilities. In some cases, it will accept a Red Green Blue (RGB) which is usually represented in hex format. In other cases, it will accept a JSON payload where the three components are separately encoded. Yet others it will use a different color model such as XY.

mcsMQTT provides four forms of interface which uses the Color Picker provided by HS for means of color selection. Received payloads will be converted from RGB, RGBW, XY or HSB color space into an integer value that will be reflected in the HS color picker device. Publish commands formats will vary based upon the type selected.

mcsMQTT also provides a method to use sliders in the HSB color space rather than color picker and use RGB in the MQTT payload. This is RGBtoHSB option.

RGB type expects #RRGGBB or RRGGBB as the status input and will publish command in #RRGGBB format.

RGBW type expects “`{"color":{"r":RRR,"g":GGG,"b":BBB},”brightness”:bbb}`” as the status input and will publish using the same format. mcsMQTT does the conversion from the HS DeviceValue representing the color picker value.

The ColorXY type is expecting a subscription with a payload containing XY color space and brightness. "color":{"x":<value>,"y":<value>},"brightness":<value>. mcsMQTT translates this into RRGGBB which is what the HS device expects for the Color Picker control. Publish command will use “`{"color":{"r":RRR,"g":GGG,"b":BBB},”brightness”:bbb}`” format.

The HSB type is expecting a subscription with a payload containing a comma-separated-values format of Hue, Saturation, and Brightness “HHH,SSS,BBB”. JSON encoding is not expected for the three components, but the key such as “HsbColor” can be part of a JSON payload.

HSB can also be used to create both color picker and a set of three sliders. This is done when the received payload is in RRGGBB format and one desires to be able to individually conrol H, S, and B as well as have composite color picker control. mcsMQTT will publish as RRGGBB no matter which of the four controls are used.

The use of the Control/Status UI or HSB is likely the most flexible for lighting controllers that can accept HSB commands (e.g. Tasmota). In this case mcsMQTT handles the conversion from the HS RGB Color Picker UI and the HSB commands. It also provides the sliders to individually control Hue, Saturation and Brightness as well as the Color Picker to select specific points in the color space. An example is shown in

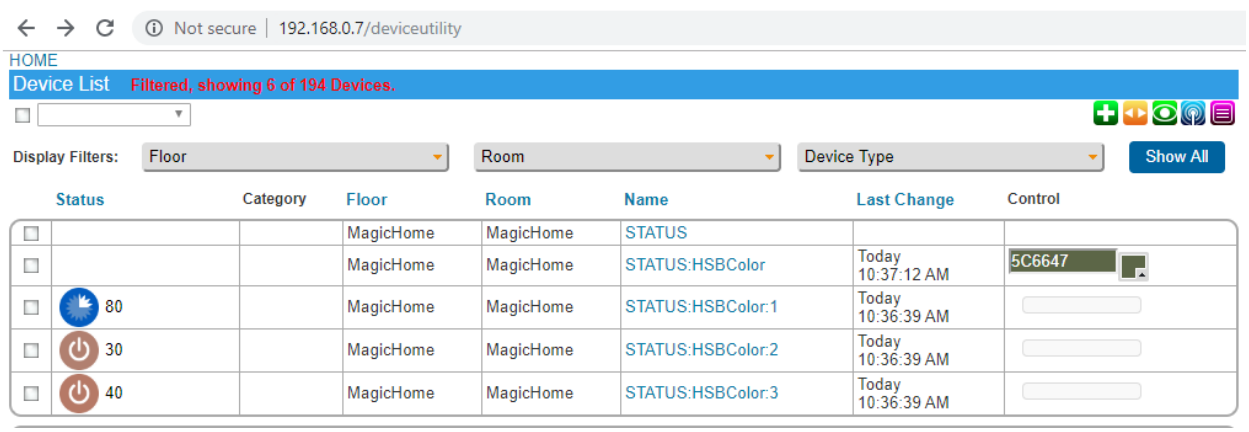


Figure 4 HSB UI in HS

In the example shown in Figure 5 the bulb information is provided in a JSON payload with state, brightness, color rgb, color x, and color y parameters. Assuming that one desires to control all parameters, yet keep the parameters all grouped in HS screens, then the first step is on Client Tab, Inbound (Subscription) Management and select the third JSON Decoding option to create both parent and child keys. This will handle the grouping. Remember to come back when done with the new bulb and change it back to the default first option of decoding into individual devices. If other organization

means are to be applied then the Client Tab setting for JSON payload can remain in it default selection.

Next step is to use “A”ssociate checkbox for the parameters of interest so they map into HS devices. After any of the items are associated then after the subsequent reception of the topic then an additional pseudo-JSON items color:rgb is created. When this item is associated and a publish topic is entered then the Control/Status UI type will be set to ColorXY and a HS Color Picker control made available. Figure 5 shows use of state, color rgb and brightness.

Since these will be controlled parameters with a MQTT publish the publish topic is entered into each of the three text boxes. I n this example it will be the same topic and it will be the same as the base sub topic with a /set suffix. Sometimes the end point is contained in the Topic and sometimes it is a generic topic with the specific parameter identified in the payload.

To support the JSON payload encoding needed for ON/OFF and Brightness control the Payload Template will need to specify the necessary JSON format. Use the Edit tab and change the Publish Payload Template for JSON formatting for the On/Off device. The easiest way to bring up the Edit tab is to click on the Ref button of the state device in the Association tab. In this case the contents of the text box will be {"state":"$$LABEL:"}, {"state":"$$VSP:"}, or {"state":"$$STATUS:"}. The “{"state":" part is what the bulb is expecting to identify the parameter. The closing “}” completes the JSON syntax. The “$$LABEL:”, “$$VSP:” or “$$STATUS:” is a replacement variable telling mcsMQTT to use the HS VSP, the mcsMQTT VSP Key or mcsMQTT VSP Status which will be either “On” or “Off” as setup in the Value Status Pairs (VSP) of the device.

The same process is followed for the brightness device. In this case it will be {"brightness":"$$VALUE"} where $$VALUE: indicates to use the HS Device Value numeric.

The third control is for color selection. Again, use the Edit tab and assure the type to be a ColorXY. This will give the desired Color Picker UI for color selection in HS and will tell mcsMQTT that JSON payload encoding is needed and that XY to RGB translation is needed before storing the color value in HS.

By default, the brightness device will be created with a text box to enter a number in the HS UI. This can be changed from HS Device Management for the Brightness device. Status Graphics tab and change the range to 0 to 255. Also change the type to slider.

Figure 6 shows the result of the above setup with Figure 7 showing various options to make color selections. Note that use of the color picker will result in a RGB color and the brightness which is implicit in the RGB value. There is a parent device 1103 which will contain the last received payload for the bulb. Under it will be the devices to control brightness with a slider, state with on/off buttons, and color with a color picker. When the slider is used only brightness information is updated, but the status published by the light will likely modify its RGB and the color picker control status will also be updated. The same is true with update of brightness device when RGB color is changed.

Filter Association Table by Mqtt Topic and JSON Payload Key | Clear Filters | Rebuild Filters

T1	zigbeeWA	T2	0xb0ce18140001de1b	T3		T4		T5		T6	
J1		J2		J3		J4		J5		J6	

Show Selected Associations

Prev | 0 | of 6 | Next

Association Table for Auto Association of MQTT Topic and HS Device

/\	r	a	ref	TOPIC	payload	h	d	lastdate
0	☐	☐		Sub: zigbeeWA/0xb0ce18140001de1b/set:brightness	194	☐		2018-12-17 14:58:52
1	☐	☑	1104	Dev: |zigbeeWA|0xb0ce18140001de1b:brightness Sub: zigbeeWA/0xb0ce18140001de1b:brightness Pub: the following Topic on Device command zigbeeWA/0xb0ce18140001de1b/set	121	☐	☐	2018-12-17 14:58:53
2	☐	☑	1109	Dev: |zigbeeWA|0xb0ce18140001de1b:Color:RGB Sub: zigbeeWA/0xb0ce18140001de1b:Color:RGB Pub: the following Topic on Device command zigbeeWA/0xb0ce18140001de1b/set	FFFF78	☐	☐	2018-12-17 14:58:53
3	☐	☐		Sub: zigbeeWA/0xb0ce18140001de1b:color:x	0.433	☐		2018-12-17 14:58:53
4	☐	☐		Sub: zigbeeWA/0xb0ce18140001de1b:color:y	0.484	☐		2018-12-17 14:58:53
5	☐	☑	1105	Dev: |zigbeeWA|0xb0ce18140001de1b:state Sub: zigbeeWA/0xb0ce18140001de1b:state Pub: the following Topic on Device command zigbeeWA/0xb0ce18140001de1b/set	ON	☐	☐	2018-12-17 14:58:53

Figure 5 Color Bulb Parameter Mapping

	Ref	Status	Floor	Room	Name	Address	Last Change	Control
☐	1103	{"state":"ON","brightness":230,"color":{"x":0.433,"y":0.484}}		zigbeeWA	zigbeeWA/0xb0ce18140001de1b		Today 2:51:21 PM	
☐	1104	230		zigbeeWA	0xb0ce18140001de1b:brightness		Today 2:51:10 PM	
☐	1105	ON		zigbeeWA	0xb0ce18140001de1b:state		Today 2:51:10 PM	ON OFF
☐	1108	16777121		zigbeeWA	0xb0ce18140001de1b:Color:RGB		Today 2:51:10 PM	FFFFA1

Figure 6 HS Device Setup for Color Control

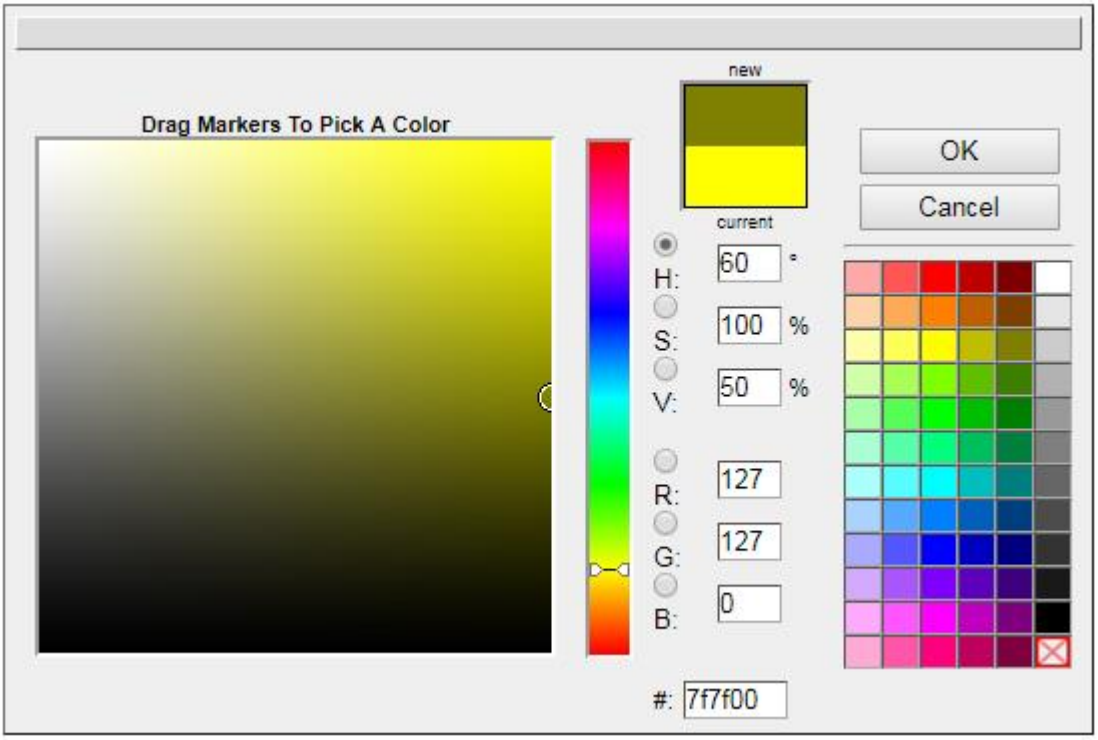


Figure 7 HS Color Picker Control

4.1.25 How do I change the subscribe Topic that is associated with a HS Device

There are two ways to change a subscribed Topic to HS Device relationship. One creates a new HS Device and one preserves the existing HS Device. To illustrate, assume there two Topics visible on the Association tab. One is Energy/Watts and the other is Energy/Amps with the Energy/Amps Topic associated with HS Device 123. The objective is to associate Energy/Watts rather than Energy/Amps.

The first approach is point and click. Uncheck the "a" checkbox on the Energy/Amps row of the Association tab and then check the "a" checkbox on the Energy/Watts row. This will delete Device 123 and HS will assign a new Device such as 124. This is the easiest way when no events or other uses have been made for Device 123.

The second approach is done from the Edit tab. Enter either 123 or Energy/Amps at the top and this will populate the remainder of the page with the properties of this relationship and it will also put a change text box on the second row. In this change row text box enter Energy/Watts. This will preserve Device 123, remove its association with Energy/Amps and assign it to Energy/Watts.

It is not possible to change the reference number of a plug-in Device in this manner because HS assigns reference numbers and all reference numbers used by the plug-in are assigned to other Topics.

It is possible to change the reference number of non-plug-in device. As an example, assume Device 234 is "Bedroom Light" and Device 456 is "Bedroom Plug" and currently Topic Bedroom/Lamp/cmnd Topic is associated with Device 234. On the Edit tab first row enter either 234 or Bedroom/Lamp/cmnd. When the second row appears enter 456. This will change the association of Topic Bedroom/Lamp/cmnd to be Device 456. Device 234 will no longer be associated with any MQTT Topic.

4.1.26 How do I remove Topics that have become obsolete

It is not unusual, especially when just starting or experimenting; that Topics are created that will never be used again. They will appear on the Association tab and just represent clutter.

There are three ways to remove obsolete topics. One is on the Association tab by clicking the "O"bsolete checkbox on any row that is to be removed. Visual feedback is provided by the Obsolete column header changing from "o" to "Delete Marked". After all obsolete Topics have been marked then use this "Delete Marked" button to actually remove them. Note that the "Exclude O & R Columns" checkbox on the top of the Association tab must not be checked to make the "O"bsolete column visible.

When a parent Topic has been marked as obsolete then all JSON items of this Topic will also be marked as obsolete and the item's obsolete checkbox will be disabled. If a parent Topic is not marked as obsolete then individual JSON items can be marked as obsolete and deleted.

The second is on the MQTT Page General Tab Obsolete Topics row. The obsolete Topic is entered and it will be removed from the database. The next time the browser page is refreshed the Topic will no longer be visible.

It is possible to remove a set of Topics by using Wildcard "*", "+" or "#" symbols. The "+" represents anything in the middle of a Topic. The "#" is used to indicate everything to its right. The "*" is similar to "#", but allows partial topic segments to be used. Special care is needed when using wildcard symbols because they can have far reaching effect.

mcsMQTT makes a backup at midnight, if enabled by user General Tab setting, so it is possible to manually copy a prior version to mcsMQTT.db in the \Data\mcsMQTT folder, mcsMQTT in the \Config folder or a script in the \scripts folder and then start mcsMQTT again to get back to that prior version if necessary. This backup includes all files in these three folders so it is also possible to restore information maintained by HS or other plugins using this same technique.

Some examples of using the wildcard symbols follow assuming that the following Topics exist

Test/topic1:a
Test/topic1:b
Test2/topic/power
Test2/topic/power1:a
Test2/topic/power1:b

"Test/topic1:#" will remove the first two Topics.

"Test2/+/power1:#" will remove the last two Topics

"+/topic/#" will remove the last three Topics

"Test*" will remove all five Topics

"Test/top*" will remove the first two Topics

A third way is to automatically remove topics that had not yet been associated with HS device or are not being used for history data retention. It only comes into play at plugin shutdown. The next time the

plugin starts the topics will no longer be present, but can be relearned if they occur again. This is also on the General Tab, Obsolete Unassociated checkbox.

4.1.27 How do I update HS Device with minimum resource utilization

There are times when all that is needed is a simple update of HS devices with MQTT payloads and no need for history, charting and some of the other features available in mcsMQTT. Express mode, which is enabled on a Topic-by-Topic basis, achieves an 80% reduction in CPU utilization that would occur in normal mode.

On the Association tab the "e" column checkbox is used to identify if a Topic has full support or if Express mode support is sufficient.

The Client tab provides a default mode option in the Express mode row. The "E"xpress column checkbox on Association tab will be checked/unchecked when a Topic is associated with the "A"ssociate column checkbox, depending upon the default for Express mode. All unassociated Topics will continue to have full support provided independent of the default mode selected. Express mode can be very austere to minimize CPU utilization, but can also support features available in mcsMQTT if enabled on the Client tab.

Express mode operation is controlled at the Topic level. It is not possible to utilize Express mode for some, but not all JSON payload items. The Association tab "E"xpress column checkboxes are only enabled for selection at the Topic level. The JSON payload items are slaved to whatever is selected at the Topic level.

See Section 17.2 for discussion of Express mode tradeoffs.

4.1.28 How do I associate multiple topics to the same HS Device

It is common that a MQTT node will respond to a command with an acknowledge of the new state as well as provide the state periodically. Each of these two updates will be done using different topics, but represent the same state information.

To model this in mcsMQTT the Ref column text box is used. First "A"ssociate a Topic and create the HS Device. In Figure 8 this was done on row 19 to create the HS Device 1310. 1310 was then entered on row 1 and row 3. Device 1310 will be updated with ON or OFF payload whenever Topic "SpaceHeater/POWER", "SpaceHeater/RESULT:POWER", or "SpaceHeater/STATE:POWER" is received.

Association Table for Auto Association of MQTT Topic and HS Device

/\	e	a	ref	TOPIC	payload	lastdate
0	☐	☐		Sub: SpaceHeater/LWT	Online	2019-03-21 20:28:12
1	☑	☑	1310	Dev: |SpaceHeater|STATE:POWER Sub: SpaceHeater/POWER Pub: the following Topic on Device command SpaceHeater/Command	OFF	2019-03-21 21:38:12
2	☐	☐		SpaceHeater/RESULT		
3	☐	☑	1310	Dev: |SpaceHeater|STATE:POWER Sub: SpaceHeater/RESULT:POWER Pub: the following Topic on Device command SpaceHeater/Command	OFF	2019-03-21 21:29:31
4	☐	☑	1308	SpaceHeater/SENSOR		
5	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:ApparentPower	1496	2019-03-21 20:28:12
6	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:Current	13.229	2019-03-21 20:28:12
7	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:Factor	1.00	2019-03-21 20:28:12
8	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:Period	25	2019-03-21 20:28:12
9	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:Power	1496	2019-03-21 20:28:12
10	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:ReactivePower	0	2019-03-21 20:28:12
11	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:Today	3.861	2019-03-21 20:28:12
12	☐	☑	1309	Dev: Unknown Sub: SpaceHeater/SENSOR:ENERGY:Total Pub: the following Topic on Device command	7.505	2019-03-21 20:27:12
13	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:TotalStartTime	2019-03-12T19:31:11	2019-03-21 20:28:12
14	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:Voltage	113	2019-03-21 20:28:12
15	☐	☐		Sub: SpaceHeater/SENSOR:ENERGY:Yesterday	1.809	2019-03-21 20:28:12
16	☐	☑	1295	Dev: Unknown Sub: SpaceHeater/SENSOR:Time Pub: the following Topic on Device command	2019-03-21T19:27:15	2019-03-21 20:27:12
17	☐	☑	1303	SpaceHeater/STATE		
18	☐	☐		Sub: SpaceHeater/STATE:LoadAvg		2019-03-21 20:28:12
19	☑	☑	1310	Dev: |SpaceHeater|STATE:POWER Sub: SpaceHeater/STATE:POWER Pub: the following Topic on Device command SpaceHeater/Command	OFF	2019-03-21 21:25:12

Figure 8 Association of multiple Topics to same HS Device

4.1.29 How do I use multiple Topics to change status of single HS Device

4.1.29.1 Case 1 Same Status reported on multiple topics

The scenario being described here is the case for a Xiaomi vacuum that uses individual topics to indicate the state of the vacuum such as the following:

vacuum/clean_start with payload that indicates the time it started
vacuum/clean_stop with payload that indicates the time it stopped
vacuum/clean_pause with payload that indicates the time it paused

While these three Topics could be associated with three individual devices it would be more convenient to have a single HS device that provides the Start/Stop/Pause status and let the LastChanged property contain the timestamp.

The setup for this is the same as the setup described in Section 4.1.28 to associate one HS Device to the three topics. In addition, regular expressions on the Edit tab will be used on each of the three subscribed topics to change the timestamp into status text. The “Payload RegEx Match Pattern” will be “.+” (without quotes) and the “Payload RegEx Operation” radio will be to “Replace” for each of the three Topics. This pattern will look for any text so any timestamp will match. The “Payload RegEx Replace Pattern” text box will contain “Stop”, “Start”, “Pause” or whatever status you want reflected in HS and is what is expected by the vacuum in its command topic. Each of the three subscribed Topics will have one of these three.

These same three status items will be itemized in VSP list on the Edit tab for each Topic. This can be done by entering “Stop,Start,Pause” (without quotes) in the VSP text box.

The Control UI radio is then selected to be Button. This will create the HS device with three buttons to control. The HS status will be updated based upon the most recently received of the three Topics from the vacuum.

The Publish Topic is entered in the Edit tab or Association tab such as “vacuum/command”. Note that the status values (Start/Stop/Pause) can be different than the command values. If this is the cases then the edits are made on the HS Device Management page to separate the Status Graphics from “both” to individual entries for “status” and “control”. What must be maintained are the number relationships between mcsMQTT VSP and those on Device Management page.

To avoid duplication of effort in the Edit tab for each of the three Topics it would be best to first setup one of the three Topics and then enter the Ref number of this Topic into the Ref column text box for the other two. The second and third will then be a clone of the first and all that is needed is to use Edit tab on these later two to change the regular expression replace text to one of the other states.

4.1.29.2 Case 2 New status to be derived from multiple topics

Consider the case of a roller blind where two relays are used to control the up/down motion of the blinds. Tasmota firmware is used in this example. The two relay status are reported by Tasmota in Topic xyz and JSON payload something like {..."Power1":ON,"Power2":OFF...}
This will be shown on Association tab as rows
xyz:Power1 ON
xyz:Power2 OFF

The desire is to show in HS as a single device that combines the status of the two relays.
Associate, via checkbox in “A” column the second row (the one that comes later in the JSON payload) to create an associated HS device.
In the Expression field of the Edit Tab combine the two payloads as two catenated strings

```
"$$PAYLOAD:(xyz:Power1):"&"$$PAYLOAD:"
```

This will result in status values of OFFOFF, OFFON, ONOFF and ONON when all combinations are received.

Use the VSP edit to change the HS status to be appropriate. e.g. OFFOFF=0;CLOSED if the status is going to be displayed. If control buttons are also added as described below then it is likely that the edit will be in the HS Devices page to remove the ability to control these four values. That is by deleting them in HS4 and changing their property to Status Only in HS3.

Use the VSP edit to create two additional values that will be used for the control (e.g. ShutterStopClose ShutterStopClose=4;CLOSE)

Use Button or List for the Control/Status UI depending upon buttons vs. pull-down selector for the UI. In the Publish Topic use cmnd/xyz/$$LABEL:

In HS Devices browser page Status/Graphics tab change the first for Value Status Pairs to be status only and the last two to be Control only unless independent control of each relay is also desired.

Tasmota will update the relay status when it receives the ShutterStopClose command and this would then be reflected in the created HS device as one of the four states of the two relays.

4.1.30 How do I create a device that has both slider and On/Off/Last button controls

Consider the case where a MQTT widget provides status and control of a dimmer device where the On/Off is provided on one Topic and the Brightness is on a second Topic. This could also be a single topic with JSON keys for state and brightness. One interface approach is a HS Device with two features. This the the straightforward "A"ssociation of each topic to two HS features that are grouped under the same parent Device.

Another approach to this is with a single HS Feature with controls from both On/Off Topic and Brightness Topic. To accomplish this one of the two Topics is "A"ssociated to create the HS Devcie and Feature. The second topic is linked by using the same Ref number provided by HS for the first association in the Ref column textbox of the second Topic.

If button and slider controls are to be shown on the HS Feature then enter the publish (control) topic for each of these two from either the Association tab row or from the Edit tab. See Figure 9 and Figure 10.

From the Edit tab select the Control/Status UI to be Button and Slider, respectively. mcsMQTT will recognize that a slider and button control are assigned to a single Ref and will merge the controls for each of the two into the Feature. It will also force the Off to be 0, the On to be 101 and the slider to be in the range of 1 to 100. The text for Off and On will attempt to be obtained from previously received payloads for the button. If the text is not as desired then use the Edit tab to setup text for the 0 and 101 values. mcsMQTT will also add a third button "Last" that can be used to restore the brightness to the level before the widget was turned off.

If a On/Off topic is received with an Off status then mcsMQTT will set the Feature's DeviceValue to 0. It will set the DeviceValue to the Brightness value in range of 1 to 100. An On/Off status of On will result in DeviceValue being set to 101.

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload
0	☐	☐	☐		5047	Test/MultiTopic2	
1	☐	☐	☐	☐		Sub: Test/MultiTopic2:A	14
2	☐	☐	☐	☐		Sub: Test/MultiTopic2:B	2050
3	☐	☐	☐	☐		Sub: Test/MultiTopic2:Discrete	1
4	☐	☐	☐	☐		Sub: Test/MultiTopic2:NUMBER	12345
5	☐	☐	☐	✓	5048	Dev: Test|MultiTopic2|MultiTopic2:Slider Sub: Test/MultiTopic2:OnOff Pub: the following Topic on Device command Test/cmnd/OnOff	On
6	☐	☐	☐	☐		Sub: Test/MultiTopic2:POWER1	ON
7	☐	☐	☐	☐		Sub: Test/MultiTopic2:POWER2	ON
8	☐	☐	☐	✓	5048	Dev: Test|MultiTopic2|MultiTopic2:Slider Sub: Test/MultiTopic2:Slider Pub: the following Topic on Device command Test/cmnd/Slider	60

Figure 9 Setup of Multi-Control Feature

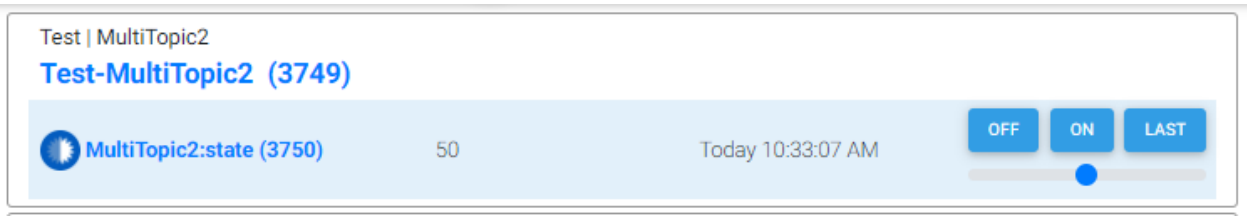


Figure 10 Slider with Buttons Control

When publishing based upon the HS button controls of event action then it will likely need to be formatted in the expected manner. This is done in the Publish Payload Template of the Edit tab. Assuming that the desired Payload for each of the following actions and the previous brightness was 33 is shown below for each of the four control actions:

1. Off button {“state”:”off”}
2. On button {“state”:”on”}
3. Last button {“state”:”on”,”brightness”:33}
4. Slider to 50% {“state”:”on”,”brightness”:50}

The Publish Payload Template can be such as:

```
<<CASE($$VALUE:,'0;101','{"state":"off"};{"state":"on"};{"state":
"on","brightness":$$VALUE:}')>>
```

The << and >> are used to indicate that an inline expression needs to be evaluated

The CASE function provides conditional output based upon the $$VALUE: (HS Command Value). For the value of 0 the payload will be `{"state":"off"}`. For the value of 101 the payload will be `{"state":"on"}`. For all others (i.e. slider values), it will be in the range of 1 to 100 for the slider`{"state":"on","brightness":$$VALUE:}`.

mcsMQTT will intercept the Last button value (255) and change it to the previously received brightness payload.

Note the Publish Payload Template uses both apostrophe and quote. The quote is needed for JSON syntax. The apostrophe or quote can be used to delimit strings in the expression, but in this case since quote is part of the text, the apostrophe is used as the string delimiter.

Note also that the CASE function is expecting each of the case input values and results to be delimited by semicolon and not a comma. This was done to prevent confusion when using numbers that, depending upon region, have decimal represented by either a period or a comma.

In this example there are two Topis (or JSON endpoints) and each will have a Publish Payload Template and a Publish Topic. When responding to a Off or On button, the selected Topic and Template will be the one for which the On and Off buttons are received. The Last button and the slider will use the Topic and Template where the slider was setup.

4.1.31 How do I setup MQTT Associations for an existing ZWave Dimmer

There may be times when the status of a HS dimmer is desired to be published via MQTT. Dimmer devices typically contain both a status and a brightness level and for this example the message will be encoded in JSON with keys for status and brightness so may look something like {“status”:”on”,”brightness”:75}.

When HS reports a change in the Zwave Dimmer via HSEvent it will provide a DeviceValue that will be in the range of 0 to 100. The 0 is interpreted as Off and the other values interpreted as On with a brightness level.

An “A”ssociation is made for the HS Zwave Device on the Association table or the MQTT Page using the “A” column checkbox. This will automatically create a Publish Topic. It is likely that this Publish Topic will be edited to provide a more meaningful name. This can be done on the Association table or on the

Edit tab of the MQTT Page. To have visibility of the Zwave Dimmer on the Association table, the filters on the Association tab need to have checkbox for “Include Non-Plugin HS Devices” and at some prior time the “Enumerate Non-Plugin Devices” button at the bottom of the General tab needs to be clicked.

Since the payload of the MQTT message will be JSON there needs to be a Publish Payload Template that defines the JSON format. This is done with the Association table “Encode Payload per template” textbox or on the Edit tab. HS is providing only one value and the JSON has two keys so a conditional expression is needed for the template. The inline expression is indicated as starting with << and ending with >>.

```
{"status":"<<IF($$VALUE:=0,'Off','On')>>"<<IF($$VALUE:>0,',"brightness
":$$VALUE:','')>>}
```

If the desire is to control the Zwave Dimmer with a MQTT message then a Subscribe/Control Topic needs to be defined at the bottom of the Edit tab. For this example, assume the same status and brightness keys of a JSON-encoded are used to provide the remote control of the dimmer. Assume the Topic is “Zwave/Control/Dimmer”. A status “Off” will command the dimmer OFF. A status of “On” will command the dimmer to the level contained in the brightness key.

After any MQTT JSON message is received on the “Zwave/Control/Dimmer” Topic, the decoded JSON keys will be available for use. These will be identified as “Zwave/Control/Dimmer:status” and “Zwave/Control/Dimmer:brightness”.

An Expression on the Edit tab is used to combined the two JSON keys into a single value to command the HS Device. In this case a conditional IF function is used to look at the status key and if set to “Off” then use “0” otherwise use the value of the brightness key.

```
IFEQ("$$PAYLOAD:(Zwave/Control/Dimmer:status):","Off",0,
$$PAYLOAD:(Zwave/Control/Dimmer:brightness):)
```

The resultant setup is shown in Figure 11.

Ref: 3752	Sub: Zwave/Control/Dimmer	Delete Sub Topic

Edit of Non-Plugin HS Device to Subscribe (Inbound) and Publish (Outbound) on Event Change

Existing HS Reference	3752
Grouping Device Ref	3751
Owning Interface	
HS Event Trigger	Value Change Event ✓ Value Set Event ☐ Log ☐ String Change Event ☐
MQTT Publish (status) Topic	ZWave/Status/Dimmer
MQTT Publish Payload Template	{"status":"<<IF($$VALUE:=0,'Off','On')>>"<<IF($$VALUE:>0,,"brightnes:
MQTT Publish To Sign	Normal Publish ● Publish to Messaging Sign ○
URI Encode Payload	Send unencoded ● Encode with URI encoding such as %20 for space ○ Replace special characters with underscore ○
MQTT Publish QOS	At Most ● At Least ○ Exactly ○
MQTT Publish Retain	Do not retain ● Retain at broker ○
Tag Field	
MQTT Subscribe (control) Topic	Zwave/Control/Dimmer
Payload RegEx Match Pattern	
Payload RegEx Replace Pattern	
Payload RegEx Operation	Replace Match Pattern With Replace Pattern ● Extract Match Pattern ○
Expression	IFEQ("$$PAYLOAD:(Zwave/Control/Dimmer:status):","Off",0, $$PAYL
Store Payload	In HS Device Value ● In HS Device String ○ Report Status on null Payload ○

Figure 11 Zwave Dimmer Control and Status Setup

4.1.32 How do I create a device for blinds or shutters with single feature control buttons

Typical blind and shutter control is provided by two relays where each relay reports its status in individual topics. There will also be a third topic that provides the current position of the blinds or shuters.

When an association is made with Control/Status types having one or more Toggle and one Slider across the topics then mcsMQTT will treat this a a single feature with control and status based upon the topics. The Toggles will be the Up/Down or Open/Close ON-state controls. The Slider is the postion feedback. It will never be shown as a control but its numeric value will be shown when the Toggle states are all 0/OFF. See Figure 12. When one relay is active the status will be the direction of motion (Open/Close). Both relays ON is failure mode.

Figure 12 Topic Association for Shutter - Blinds

The labels for the buttons and the status when a relay is ON will be the VSP status setup on the Edit tab for the ON relay value. The two end positions will be from the OFF relay status. 0 and 101 are used for the VSP values where 101 is significant to support the recognition of the button push from HS. See Figure 13. 0 and 102 are used for the second button. The order of the buttons is the same as the order when the button associations to the reference feature is made. The VSP numbers cannot be edited, but the status values should be edited to get the desired text for the status and the button labels.

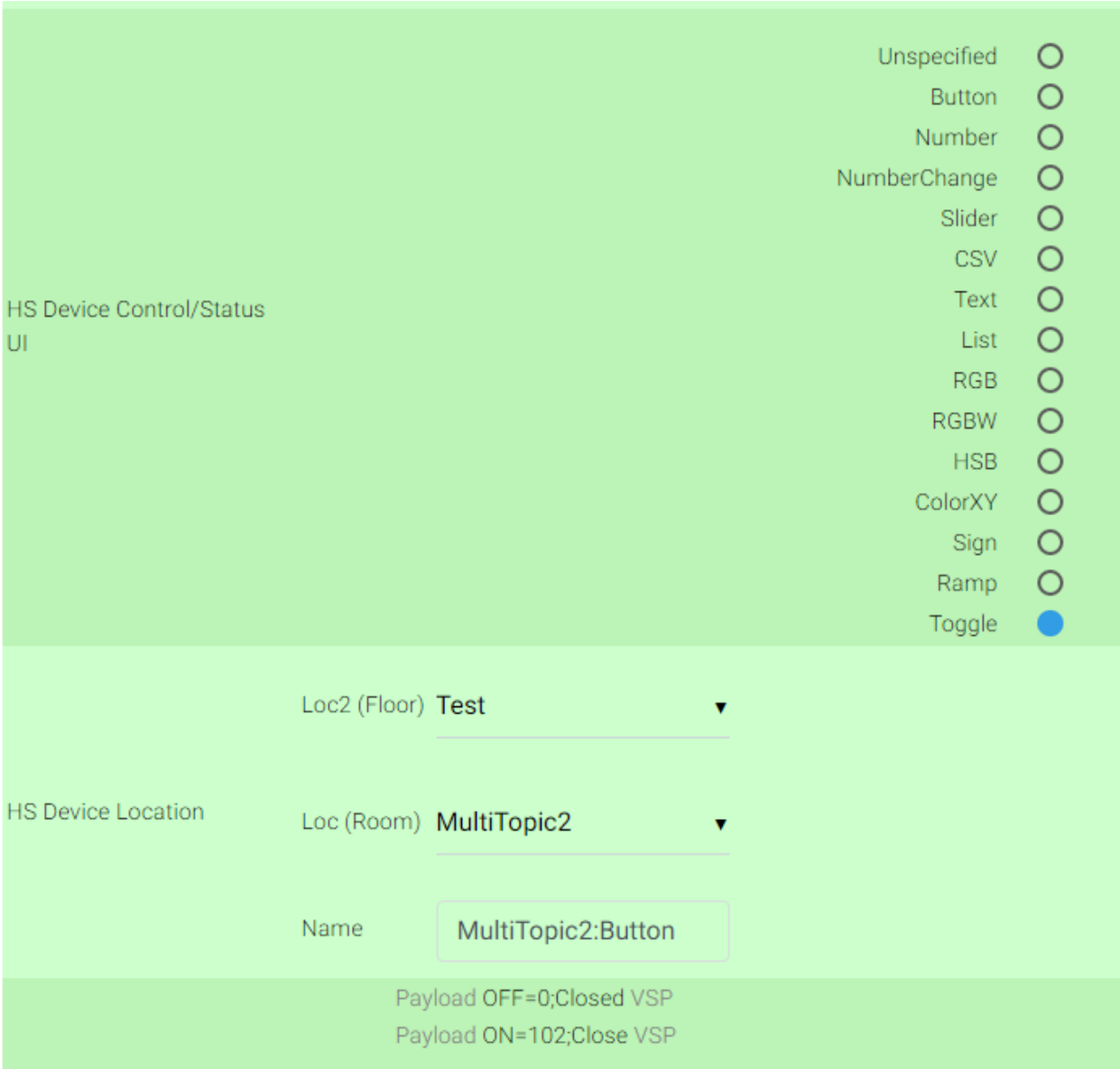


Figure 13 Shades - Blinds VSP Setup

The HS device and feature will be shown as in Figure 14.

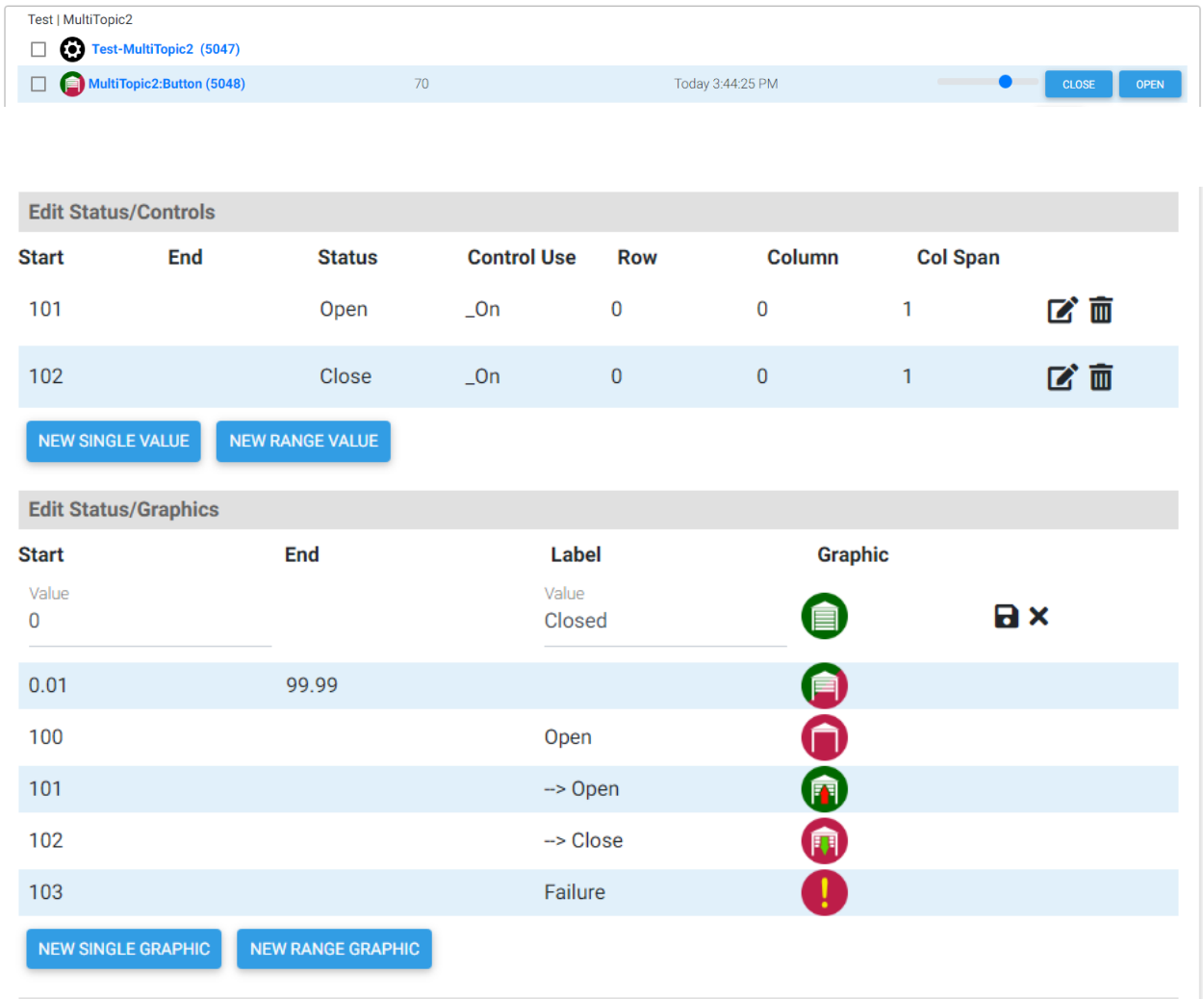


Figure 14 Shutter - Blinds HS Device and Feature

4.1.33 How do I connect to multiple MQTT Brokers

Two techniques are available to use multiple brokers. One is to setup a second broker as a client and use broker bridging. The other is to specify the second to sixth broker on the MQTT Page Broker Tab. A semicolon is used the separate the name or IP of each broker as shown in Figure 15. The port, ID and security will default to the same as the first broker, but each can be individually changed. When editing the credentials for each broker a semicolon is used to separate the credential for each broker, such as shown for port and ID in Figure 15.

Date/Time	Pri	Type/Error	Message/Source
Sep-27 1:01:07 PM		mcsMQTT	Device 2087(SpaceHeater SENSOR:ENERGY:Total) Changed from 136.198 to 136.208
Sep-27 1:01:04 PM		mcsMQTT	Device 2088 set to 0 when SpaceHeater/POWER payload changed from ON to OFF
Sep-27 1:01:04 PM		Device Control	Device: SpaceHeater POWER to OFF (0) by/from: CAPI Control Handler
Sep-27 1:00:54 PM		mcsMQTT	Device 2087(SpaceHeater SENSOR:ENERGY:Total) Changed from 136.197 to 136.198
Sep-27 1:00:54 PM		mcsMQTT	Device 2088 set to 1 when SpaceHeater/POWER payload changed from OFF to ON

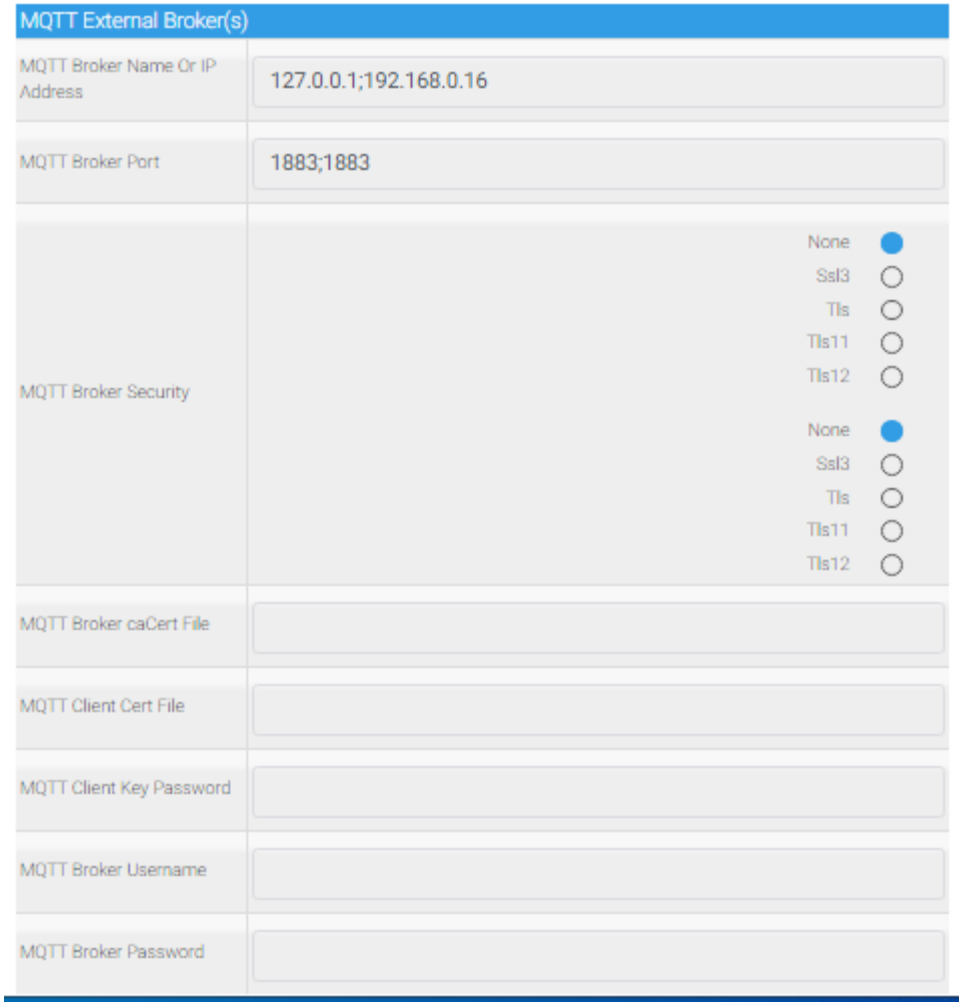


Figure 15 Multiple Broker Setup

All subscriptions will be associated with the broker that delivered the message. If there are multiple brokers setup then the Association tab topic for each parent row will be suffixed with [#] where # is the broker that delivered the message. The example in Figure 16 shows broker 2 as the provider of the message.

Association Table for Auto Association of MQTT Topic and HS Device

Λ	o	r	e	a	ref	TOPIC	payload	h	d	lastdate
0	☐	☐	☐	☐		test/Broker/JSON [2]				
1	☐	☐	☐	☐		Sub: test/Broker/JSON:color:x:DS18B20:Humidity	5.0	☐		2019-09-14 11:16:20
2	☐	☐	☐	☐		Sub: test/Broker/JSON:color:x:DS18B20:Temperature	5001	☐		2019-09-14 11:16:20
3	☐	☐	☐	☐		Sub: test/Broker/JSON:color:x:Switch2	ON	☐		2019-09-14 11:16:20

Figure 16 Broker Identification for Received Topics

The statistics provided on the Association Tab with Topic Filter T1 set to MQTT shows information for all brokers except for the broker status which shows each individually. If HS devices are populated with statistics, then the Online status in DeviceValue is 0/Offline if any broker is offline. By default, the first

broker will be used to publish messages for HS devices that are associated without a subscription. This can be edited on the Edit tab with radio button as shown in Figure 17.

The relative position of a broker in the list is very important as this is what is remembered by mcsMQTT. It will not matter for data that is received from multiple brokers, but when HS is publishing the broker that will be used is based upon the broker index at the time this broker delivered the subscribed topic. For example, consider a setup of “192.168.0.100;127.0.0.1” as the list of brokers. Topic “From100” is received from 192.168.0.100 and associated with a HS Device and has a publish topic “From100/set” setup. 192.168.0.100 is first broker in the list so mcsMQTT will remember to publish From100/set to the first broker in the list. If sometime later the broker list is changed to “192.168.0.200;192.168.0.100” and HS then publishes topic “From100/set”, then it will be sent to broker at 192.168.0.200 because it is the first one in the list.

This makes it important to keep the broker position static and add new brokers to the end of the list. It was done this way so that broker IPs can be changed and no change is needed in the prior setup of the MQTT-based devices.

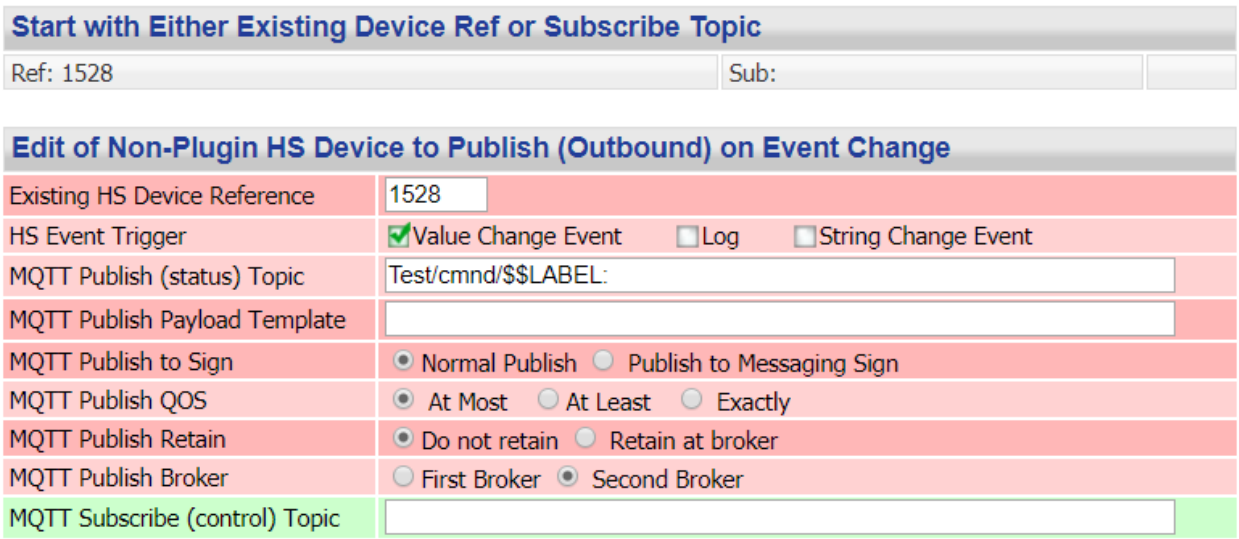


Figure 17 Broker Selection for Non-Plug-in Devices

4.1.34 How to record changes to HS Log

When a device is controlled from the HS UI then a “Device Control” action is put in the HS Log. If a similar logging is desired when the device is updated based upon MQTT payloads then two setup actions are needed. First is on the History tab using the Pub-Sub Message History checkbox to Log changes. The second is on the Association Tab H column to checkbox the specific topics for which logging will be done. When the log entry is made one of two forms will be used. For express mode the from /to payload will be shown. For normal mode the from/to HS values will be shown. See Figure 18.

Date/Time	Pri	Type/Error	Message/Source
Sep-27 1:01:07 PM		mcsMQTT	Device 2087(SpaceHeater SENSOR:ENERGY:Total) Changed from 136.198 to 136.208
Sep-27 1:01:04 PM		mcsMQTT	Device 2088 set to 0 when SpaceHeater/POWER payload changed from ON to OFF
Sep-27 1:01:04 PM		Device Control	Device: SpaceHeater POWER to OFF (0) by/from: CAPI Control Handler
Sep-27 1:00:54 PM		mcsMQTT	Device 2087(SpaceHeater SENSOR:ENERGY:Total) Changed from 136.197 to 136.198
Sep-27 1:00:54 PM		mcsMQTT	Device 2088 set to 1 when SpaceHeater/POWER payload changed from OFF to ON

Figure 18 HS Log of MQTT-based changes

4.1.35 How to view CSV payloads in separate HS Devices

Payloads may contain numeric data as a group or set of values separated by commas. Consider the following payload:

{"HSBColor":" 120 ,100,46","Channel":[0,46,0,47]}

The HSBColor key contains three CSV values and the Chananel key contains four in a group bracket holder. These two items will be stored in HS devices as strings in DeviceString. The Control/Status UI will be of type "text". To create additional HS devices that contain the individual numbers into DeviceValue the Edit tab Contro/Status UI type of "CSV" is selected such as shown in Figure 19. Another option is to select "HSB" which will yield similar results, but also contain the context of controlling a device that is expecting color control.

Settings for Plugin Device	
HS Device Publish Topic	MagicHome/cmnd/HSBColor
HS Device Control/Status UI	○ Unspecified ○ Button ○ Number ○ NumberChange ◉ CSV ○ Text ○ List ○ RGB ○ HSB ○ ColorXY ○ Sign

Figure 19 CSV type Payload Selection

There will be a HS device created for the key and one for each of the CSV components. The default name of each will be the same as the subscribed topic and suffixed with the 1-based position of the number in the CSV list. HS Device Association will be used to keep all the devices remaining together when displayed.

☐	2327	[11,12,13,14]		MagicHome	STATUS:Channel	MagicHome/STATUS	Today 6:49:05 PM	(value) [11,12,13,14] Submit
☐	2326	Off		MagicHome	STATUS	MagicHome/STATUS	Today 6:46:39 PM	
☐	2332	11		MagicHome	STATUS:Channel:1	MagicHome/STATUS	Today 6:49:09 PM	
☐	2333	12		MagicHome	STATUS:Channel:2	MagicHome/STATUS	Today 6:49:09 PM	
☐	2334	13		MagicHome	STATUS:Channel:3	MagicHome/STATUS	Today 6:49:09 PM	
☐	2335	14		MagicHome	STATUS:Channel:4	MagicHome/STATUS	Today 6:49:09 PM	

Figure 20 CSV Type HS Device Creation

If a CSV device is to be controlled then the publish payload topics need to be set on the Association tab or Edit tab. For the CSV case the topic is entered as a set of topics which each using a semicolon (";") as the separator. The controls will be setup as sliders with range between 0 and 100. An example is shown in Figure 21 and Figure 22. In the HSB case the publish topic will be the topic for the color picker control and if the sliders are used for control then the values of the three H, S and B sliders will be combined

and payload will be the same as the color picker payload which is a set of three decimal values separated by commas.

4	☐	☐	☐	☑	2351	Dev: |MagicHome|STATUS:HSBColor Sub: MagicHome/STATUS:HSBColor Pub: the following Topic on Device command MagicHome/cmnd/HsbColor1;MagicHome/cmnd	10 ,10,4	☐	☐	2019-12-09 16:26:01

Figure 21 Setup of CSV Publish CSV topics

☐	2383	6055495		MagicHome	STATUS:HSBColor	MagicHome/STATUS	Today 4:35:33 PM	5C6647
☐	2396	80		MagicHome	STATUS:HSBColor:1	MagicHome/STATUS:HSBColor	Today 4:35:31 PM	
☐	2397	30		MagicHome	STATUS:HSBColor:2	MagicHome/STATUS:HSBColor	Today 4:35:31 PM	
☐	2398	40		MagicHome	STATUS:HSBColor:3	MagicHome/STATUS:HSBColor	Today 4:35:31 PM	

Figure 22 Devices for CSV Control Type with Publish topic

4.1.36 How do I create unique HS devices when the payload content contains device identification

A common situation is that a device will publish on a topic using JSON encoding for the payload and one fo the keys of the payload is the identification of what device characteristic is being reported. Two cases are supported by mcsMQTT. One is where the JSON contains identification information that applies to all other keys in the same payload. The second is where identification applies only to the keys in the JSON group that contains the identification. In the later case the parent of the group is not considered and only treated as a wildcard position holder.

In the first case, for example, consider the following two message payloads that are sent on the “Printer/Status” topic.

{“Id”:”ink”,”empty”:false}
{“Id”:”paper”,”empty”:false}

Normally mcsMQTT will provide two devices when selected since two JSON keys are in the payload. These are:

Printer/Status:Id
Printer/Status:empty

What is actually desired is to have separate “ink” and “paper” status available so the view presented to HS would be as follows. This would allow HS device creation of separate ink and paper status devices.

Printer/Status:Id
Printer/Status:ink:empty
Printer/Status:paper:empty

The Edit tab is used for the actual parent Topic (does not have any of the :JSONkey elements) and in the textbox that is provided for MQTT Subscribe Topic at the top the JSON key “Id” would be entered to elevate “Id” from the Payload to become a member of the Topic.

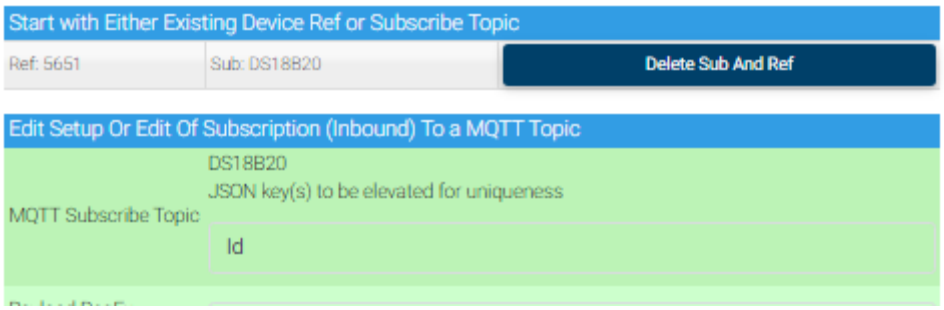
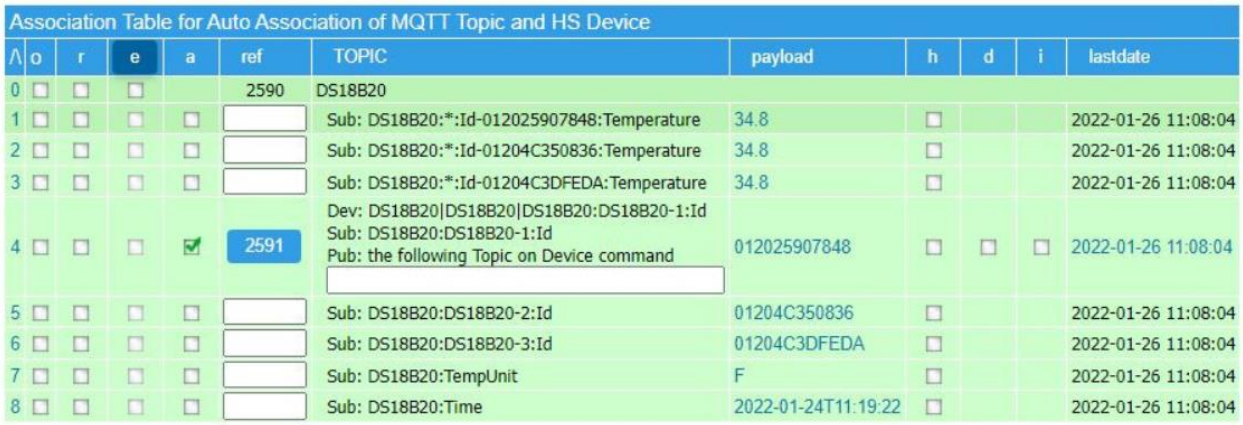


Figure 23 Using Payload Key to Achieve Indepenent Devices

In the second, wildcard, case consider the following JSON payload where the actual temperature sensor identification is the Id part of the message and group name (DS18B20-1/2/3) has no particular significance. The 1/2/3 is usually part of a JSON array and the order of the array elements has no particular significance. To indicate that the 1/2/3 should be ignored (i.e. wildcard) then nomenclature *:Id should be used.

This results in the subscription topic and device name to be shown with ":*:" as shown in Figure 24. The second step is to now "A"ssociate the temperate key rows. From this point forward the DS18B20-1/2/3 group name is ignored and the HS device reference remains aligned with the Id field of the JSON payload. Other JSON keys that are not part of the same gropu as the Id are not affected.

```
sonoff-1234/STATUS = {"StatusSNS":{"Time":"2022-01-27T01:33:31","DS18B20-
1":{"Id":"000003A280FF","Temperature":77.2},"DS18B20-
2":{"Id":"020E92451C31","Temperature":60.2},"DS18B20-
3":{"Id":"021892457E2A","Temperature":58.3},"BME280-
76":{"Temperature":113.4,"Humidity":4.1,"DewPoint":23.6,"Pressure":993.1},"BM
E280-
77":{"Temperature":69.7,"Humidity":25.0,"DewPoint":32.2,"Pressure":993.3},"Pr
essureUnit":"hPa","TempUnit":"F"}}
```



Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
0	☐	☐	☐		2590	DS18B20					
1	☐	☐	☐	☑	2592	Dev: DS18B20|DS18B20|DS18B20:*:Id-012025907848:Temperature Sub: DS18B20:*:Id-012025907848:Temperature Pub: the following Topic on Device command	38.0	☐	☐	☐	2022-01-26 11:20:09
2	☐	☐	☐	☑	2593	Dev: DS18B20|DS18B20|DS18B20:*:Id-01204C350836:Temperature Sub: DS18B20:*:Id-01204C350836:Temperature Pub: the following Topic on Device command	46.5	☐	☐	☐	2022-01-26 11:20:09
3	☐	☐	☐	☑	2594	Dev: DS18B20|DS18B20|DS18B20:*:Id-01204C3DFEDA:Temperature Sub: DS18B20:*:Id-01204C3DFEDA:Temperature Pub: the following Topic on Device command	34.8	☐	☐	☐	2022-01-26 11:20:09
4	☐	☐	☐	☑	2591	Dev: DS18B20|DS18B20|DS18B20:DS18B20-1:Id Sub: DS18B20:DS18B20-1:Id Pub: the following Topic on Device command	012025907848	☐	☐	☐	2022-01-26 11:20:09

Figure 24 Elevate JSON key with Wildcard

In summary, when a Payload with a set of JSON keys exists and one of the JSON keys in the JSON group should be used as part of unique identification then use the Edit Tab with the parent Topic to enter the JSON key that will be elevated. This is done at the top in the MQTT Subscribe Topic textbox. If the JSON key is part of a JSON array where the array position has no significance then prefix the JSON key with "*:" in the MQTT Subscribe Topic textbox. In both cases, from the Association Tabs, "a"ssociate the items to be mapped into HS. Do not associate the row that has the JSON key that was elevated as that would be redundant.

4.1.37 How do I store picture contained in MQTT Payload

MQTT payloads typically contain text, but it is also possible to send binary data such as a jpg image as is done for the Ring doorbell camera as described at [GitHub - tsightler/ring-mqtt: Ring devices to MQTT Bridge]() using a Topic such as:

"ring/afa06d00-91a1-49ed-8698-1653ad64c51f/camera/10082c56ad93/snapshot/image"

If this Topic is selected to be "jpg File" as the Control/Status UI on the Edit Tab as shown in Figure 25 then the received Payload will be stored in file in the subfolder \html\mcsMQTT\File\{floor}\{room}\{name}.jpg where {floor}, {room} and {name} are the HS properties of the "A"ssociated Device. A thumbnail will also be created in the same folder that will show as an icon for the device via the HS VGP mechanism.

HS Device Control/Status UI	○ Unspecified ○ Button ○ Toggle ○ Number ○ NumberChange ○ Slider ○ Ramp ○ CSV ○ Text ○ List ○ RGB ○ RGBW ○ HSB ○ ColorXY ○ Sign ◉ jpg File

Figure 25 Control/Status UI selection for a jpg image

The DeviceValue will be incremented for each time the Topic is received. The DeviceString will contain HTML hyperlink and image tags. The image tag will render a thumbnail of the jpg file and the hyperlink, when clicked, will render the full-size image. An example of HS4 rendering is shown in Figure 26 where two jpg File features with the first one showing it has been received 5 times.

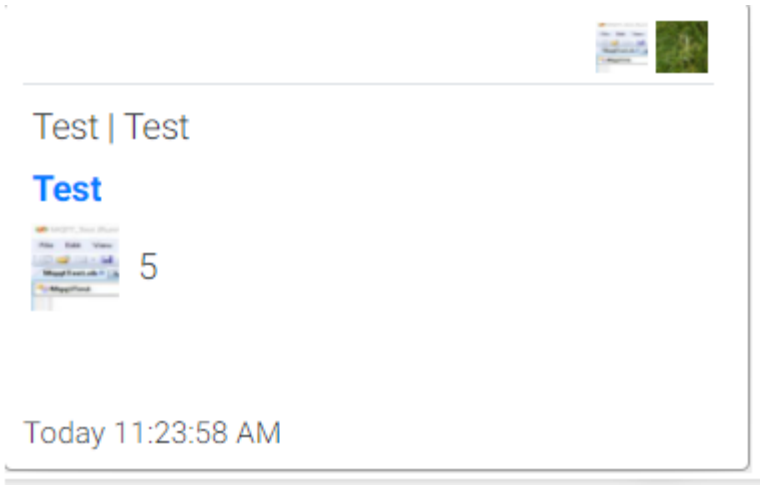


Figure 26 HS4 Device with two jpg File Topics

4.1.38 How do I publish different payload formats depending upon the HS control value

A Publish Template is used to format payload in a message being sent by mcsMQTT. The usual scenario is that the format of the payload is the same and only a data value will change based upon what is being requested by a HS commanded value. Take the scenario, however, where the format of the payload needs to change for each command. For example, HS has three control values UP=0, DOWN=1, ROTATE=2 and the commands that need to be published are shown below with different formats.

UP - {“U”:true, “D”:false}

DOWN - {“U”:false,“D”:true,”R”:”left”}

ROTATE – {“R”:true}

Conditional expressions can be used such as a CASE shown below. Note comma used to separate the three case parameters and semicolon used to separate the items in the second and third parameters that are both strings.

<<CASE$$VALUE:,”0;1;2”,”{“U”:true, “D”:false}; {“U”:false,“D”:true,”R”:”left”}; {“R”:true})>>

Conceptually this will work, however the content of the two strings contain quotes around the U, D and R. There is no way to escape or otherwise indicate the literal quote vs. a delimiter quote in the CASE string parameter. To overcome this the inner quote is changed to a unique character and then part of the expression is to replace this unique character with the quote. An example below where the REPLACE function was added to replace ~ with “.

<<REPLACE(CASE$$VALUE:,”0;1;2”,”{~U~:true, ~D~:false}; {~U~:false,~D~:true,~R~:~left~}; {~R~:true}),”~”,”””)>>

4.1.39 Data is flooding my system, what can I do

Some MQTT clients send data at a very high rate and this could bring HS down to a crawl and experience significant delays.

The best way to address the problem is to reduce the rate at which the client is producing data. This will reduce network traffic to the MQTT Broker and then from the Broker to mcsMQTT/HS. Unfortunately, some clients will not provide this capability.

If data is being received at a high rate and there is no association with a HS device then there is no need to subscribe to that Topic. The Topic Discovery row on the Inbound Management Section of MQTT Page, Client Tab provides a radio control to select between the default listen to everyting to selecting specific topics. The middle radio is usually the easiest, and recommended selection, for any system that is running without need to discover new widgets. Note that it is the Topic and not just a JSON item within a Payload that is being selected in this case. If any data in the Payload is associated with a HS Device Feature then the Topic will be included and all JSON items in that Payload will be processed.

The MQTT Page, General Tab provides a Reject textbox to specify one or more Topic patterns that should be ignored upon receipt without further processing. The wildcards of + and # per MQTT standard syntax can be used to specify groups of topics that should be rejected.

When the data is of interest and associated with HS Device Feature then two techniques can be applied to reduce the CPU burden. One is to use the Express Mode ("e" column checkbox on Association Table). This will minimize the times the plugin will use the HS object. The HS Object is the largest CPU user when Device Features are being updated. MQTT Page, Client Tab, Inbound Management Section has Express mode tuning options that limit the capabilities that are supported on Topics selected for this mode of operation.

Another solution is to only process a subset of the data being received. Telemetry type data is typically only marginally changing on each sample so there will be minimal loss of fidelity when data is subsampled. For example, data may be received 4 times per second, but any HS event action based upon this data will only be needed every minute. This means that only 1 of the 240 samples are actually needed to achieve the objectives of the event action. Subsampling can be defined on a topic-by-topic basis on the Edit Tab of any parent Topic (i.e. Associaiton Table Sub without a colon). In this example a value of 60 would result in one sample processed every 60 seconds and the others discarded.

The second method to specify subsampling is on the MQTT Page, Client Tab, Inbound Management Section with a Subsample Template and Interval. The interval is the sample effect as on the Edit Tab, but the Template can have wildcard (e.g. Widget/HighRateData/#) to specify a range of one or more Topics.

4.1.40 Can I use the ZigbeePlus MQTT Broker with mcsMQTT

By default, the HS ZigbeePlus plugin will create a MQTT Broker on interface 127.0.0.1:1883 using the MQTT 3.1.1 protocol. The default configuration for mcsMQTT is also to create a MQTT Broker on 127.0.0.1:1883. mcsMQTT will first confirm that port 1883 is available for use and if so then not create the MQTT Broker on 127.0.0.1. If mcsMQTT starts first, then itr is likely that ZigbeePlus will have a problem creating the MQTT Broker on 127.0.0.1.

Both mcsMQTT and ZigbeePlus have options to use other MQTT Brokers. For example, both can be configured to use Mosquitto running on some computer that is network-visible.

For those that are already using ZigbeePlus and desire to keep using the MQTT Broker that it created, then mcsMQTT MQTT Page, Broker Tab should be configured to not use its Internal MQTT Broker with the radio setting and the External MQTT Broker should be set to the IP of the HS computer. This will not be 127.0.0.1, but something like 192.168.0.100.

ZigbeePlus and mcsMQTT both use the same MQTT library so functionality should be very similar. A MQTT Broker like Mosquitto has a much wider user base so should be a more mature implementation than those used by the two HS plugins. External Brokers also have the advantage that they continue to function when HS or the HS Plugin is not running.

The mcsMQTT implementation supports both the version 3 and 5 protocols, contains persistent retain message support, and provides statistics of the clients that are connected such as shown in Figure 203. If one has very basic needs for MQTT communications then it should not make much difference which MQTT Broker is used. For others, the mcsMQTT or Mosquitto MQTT Brokers may be a preferred choice.

4.2 Default Settings for mcsMQTT

mcsMQTT is setup from Plug-in button – mcsMQTT option. Multiple pages are available. For MQTT operation the MQTT Page is selected. Multiple tabs are available with Client, Broker and General being those of interest for the setup. The other tabs are used for later interaction.

The default configuration is a MQTT Broker that is hosted by mcsMQTT and mcsMQTT as the client connecting to this MQTT Broker. These can be altered on the MQTT Page, Broker and Client tabs. A common configuration is to use an external Broker rather than the internal one. Mosquitto is a popular MQTT Broker.

4.3 Automatic Setup of Device to Topic Relationship

The default operation of mcsMQTT is to firewall all MQTT Topics and allow the user to selectively associate each with a HS device. This is the "opt-in" approach. The Association tab shows all the Topics that have been detected.

This default operation can be changed where the Association tab firewall becomes an "opt-out" or hybrid of "opt-in" and "opt-out". The Client tab, Incomming (Subscription) Management section has a line for "Wildcard Plugin Auto Associate Template". When this text box is left blank the default "opt-in" behavior will be used. When one or more Topic templates are entered then the Topics that match the template with have HS device automatically created. Other topics will be firewalled at the Association tab for user selection. The Topic template follows MQTT topic wildcard standards where "+" designates current position in hierarchy and "#" is used to indicate everyting at this and postions to the right. To Auto-Associate everything the template would simply by "#".

After some Topics have been published by the Broker the next step is the Associations Tab. This tab can display the universe of information related to received Topics and potential Topics that can be published by HS. Because the amount of information is large a subset is normally selected for display. Three tables are used to narrow down information of interest. Initially let us consider only the Topics that have been forwarded by the Broker. The first table will have one checkbox, "Include MQTT Topics" checked. The two tables will have no selections made in the pull-downs. If there a limited number of Topics published the second pull-down table can be left unchanged. If there are many then select one or two columns of the pull-down to be more specific in the Topic of interest.

The "Show Selected Associations" button at the bottom will then be used to display all the Topics forwarded by the Broker. If there have been hundreds then it will take many seconds to render the table. If only a handful then it will be a second or two.

The table rows will be colored green to indicate that this is an inbound Topic and potential mcsMQTT device in HS. The "A"ssociate column will have no checkboxes and all the Ref column entries will be blank text boxes. The Topic and Payload of the messages will be shown. If the Payload has been encoded using JSON then multiple rows will be shown with each JSON item on a row.

<u>Map Topic to mcsMQTT HS Device</u>

If one of these Topics is of interest and the desire is to show the Payload in HS3 Device then the "A"ssociate checkbox is checked. This will create a status-only device in HS. The device will be updated each time the Topic is received.

When “A”ssociate is checked then an additional text box is provided in the Topic column. This text box is used to Publish a MQTT Topic that will contain the HS Device Value or String each time the Device changes. The selection of Value or String for the Payload is based upon the receive Topic’s Payload. If it is a number then DeviceValue will be Published otherwise DeviceString will be Published. The exception is for a receive Payload of ON, OFF, OPEN, CLOSED, FALSE, TRUE, DISARMED, ARMED, INACTIVE, ACTIVE 0 or 1. In these six cases the HS control presented on the UI will be a button with DeviceValue of 0 and 1 and Labels that correspond to the text Payload or On/Off for the two numbers. Selection of Label (Status) or Number (Value) is made in the Publish Payload Template on the Edit tab with $$VALUE: for Value. $$LABEL:, $$VSP:, or $$STATUS: will result in the label being published.

When an association is made the update of the HS Device, action triggers on this Topic, History of the Topic is collected and if the payload is JSON-encoded then mcsMQTT will be decoded it into its individual components. If the additional features beyond the update of the HS Device are not needed then the “E”xpress checkbox can be used to identify this Topic being one that will be processed in Express mode. Express mode CPU consumption is about 20% of full support mode. The default is full support, but can be toggled on the Client tab Express mode setting. See Section 17.2 for a discussion of Express mode tradeoffs.

<u>Map Topic to an existing non-Plug-in Device</u>

Assume the Topic of interest is a command that is desired to control an existing HS Device. In this case rather than using the “A”ssociate to create a new mcsMQTT Device the Ref column text box will be used to enter the Device Reference of the existing HS Device. The Topic column will show an additional text box and the “A”ssociate checkbox will be checked. If a Topic is entered into this new text box then this Topic will be published upon HS Device change.

The same general process can be performed by viewing the non-Plug-in devices and using the “A”ssociate checkbox to establish a relationship. In this case the “Include non-Plug-in HS Devices” checkbox would be checked to make available the list of HS devices in the Association table.

<u>Change an existing Topic to HS Device association</u>

If an association between a Topic and HS Device needs to be changed to a different Device or different Topic then the Edit tab can be used to make this change. Alternately the existing association can be removed by unchecking the “A”ssociate checkbox on the Association tab to remove the association. This is followed by using the “A”ssociate checkbox on the desired Topic. This alternate method may be quicker point and click method but it will result in the original Device being deleted if it is a mcsMQTT plug-in device. If there were events or other uses of the original Device reference then these will be broken.

When using the Edit tab to accomplish the change the original Device reference number is maintained. Start by entering the Ref number or the Topic at the top of the Edit tab. In the second row a change text box will appear. In the text box enter the Topic that should be associated with the Ref. If the Ref is a non-plug-in Device then it can be changed to another non-plug-in Device to associate the Topic with a difference Device. It is also possible to change the association of the Topic from a plug-in Device to a non-plug-in Device by changing the Ref, but the reverse it is not possible because HS assigns reference numbers and all that have been assigned by HS are already associated with other Topics.

4.4 Manual Setup of Device to Topic Relationship

If you know the Topic “myHome/myTemperature” will be Published by a temperature sensor then this Topic can be specified in the Subscription (Inbound) table of the Edit tab. A HS status-only device will be created and its DeviceValue will be updated when a numeric Payload is Published on this Topic.

If the Topic is “myHome/myLight” then the same process can be followed. In this case the “HS Device Publish Topic” in the green table can be used to control the light. Let us say the control Topic is “cmnd/myHome/myLight”. The Button radio selected to generate On/Off controls in the HS UI. The HS Device will be controlled by the button on the UI, or other means; mcsMQTT will Publish Topic “cmnd/myHome/myLight” with Payload of “On” or “Off”; myLight will respond to the command and Publish “myHome/myTemperature” with a Payload of “1” or “0” to reflect its state. Note that the specific values and text are customizable via the HS Device Management Page Value-Status-Pairs.

If you have an existing HS device and you want it to respond to commands from MQTT Topics or want it to be commanded by MQTT Topics then the pink Publish (Outbound) table is used. Separate text boxes are provided for each case.

4.5 Leveraging Multiple mcsMQTT Browser Pages

mcsMQTT allows multiple browser windows to be opened and each is maintained independently. One page can be used to display a large Association table and another page could be used with the Edit Tab to edit associations. In essence one becomes a reference lookup and the other used to perform the edits.

Note that each browser page will continue to be updated with new Payloads, statistics etc. as long as each window is open. When the plug-in is restarted those browsers windows become stale and cannot be used to interact with a new instance of mcsMQTT. The URL needs to be refreshed to redraw the page or a new page opened.

5 Sending MQTT Messages

Sending a MQTT message can be done via Device changes, Event actions, or script. Two paradigms exist with respect to publishing MQTT Topics. One is to support a HS Topic that reflects the status of an HS device. This is described in subsequent paragraphs of this section. An example is publishing the HS Uptime since last restart.

The second is part of bidirectional communication with another publisher where HS publishes a command to affect the state of an external node. This is described in Section 0. An example is a light that publishes its ON/OFF status and HS is used to control the ON/OFF state of this light.

Any HS Device that has been "A"ssociated can publish a status or command Topic. The Topics that are published are entered into the Topic text box on the Association Tab. The published Payload will depend upon another configuration described below.

In the case where a non-Plug-in Device is "A"ssociated the Payload will default to the DeviceValue and occur when the DeviceValue changes. See Figure 27. This can be changed on the Edit tab immediately or later by entering the HS Device Reference number. The trigger can be selected as a change in DeviceString or DeviceValue (or both). The Payload will be based upon the publish template and can contain substitution variables. See Figure 27. Substitution variables are described in Section 5.2.

The special symbols +, / and # are reserved per MQTT protocol and cannot be used in Topics. If these symbols are part of the HS device name, for example, then they should be URI-encoded so as to not violate the standard. The space character is not recommended and by default it should be URI-encoded as well.

To specify URI-encoding the substitution variable should be suffixed with "_URI_ENCODE". A second option is to use the URI Encoding radio for the message that is available on Edit tab. This will result in either URI encoding or replacement of all special characters with an underscore character. This will apply to all characters in the message.

Another option exists for space where substitution variable is suffixed with "_UNDERSCORE". For example, "$$NAME_UNDERSCORE:" as part of the publish Topic will result in any spaces in the HS Device Name to be replaced with "_".

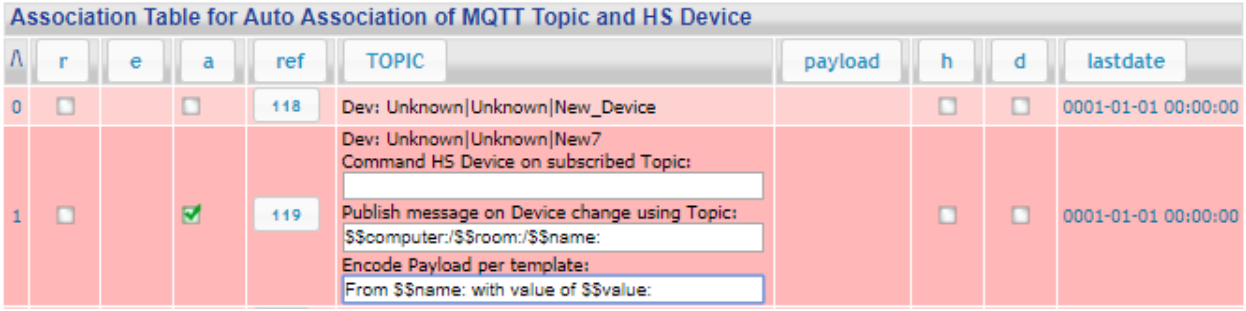


Figure 27 Non-Plug-in Device Association

Figure 28 shows a non-plug Edit popup. Figure 29 is a popup that appears when the "Configure Sign Parameters" hyperlink is clicked. It shows that the message going to the sign will appear in Row 1, and

will show forever (Duration = 65535) or until another Log message is published or the Duration field of message topic is later set to 0. The text color of the messages will be hex FFFFFF (white).

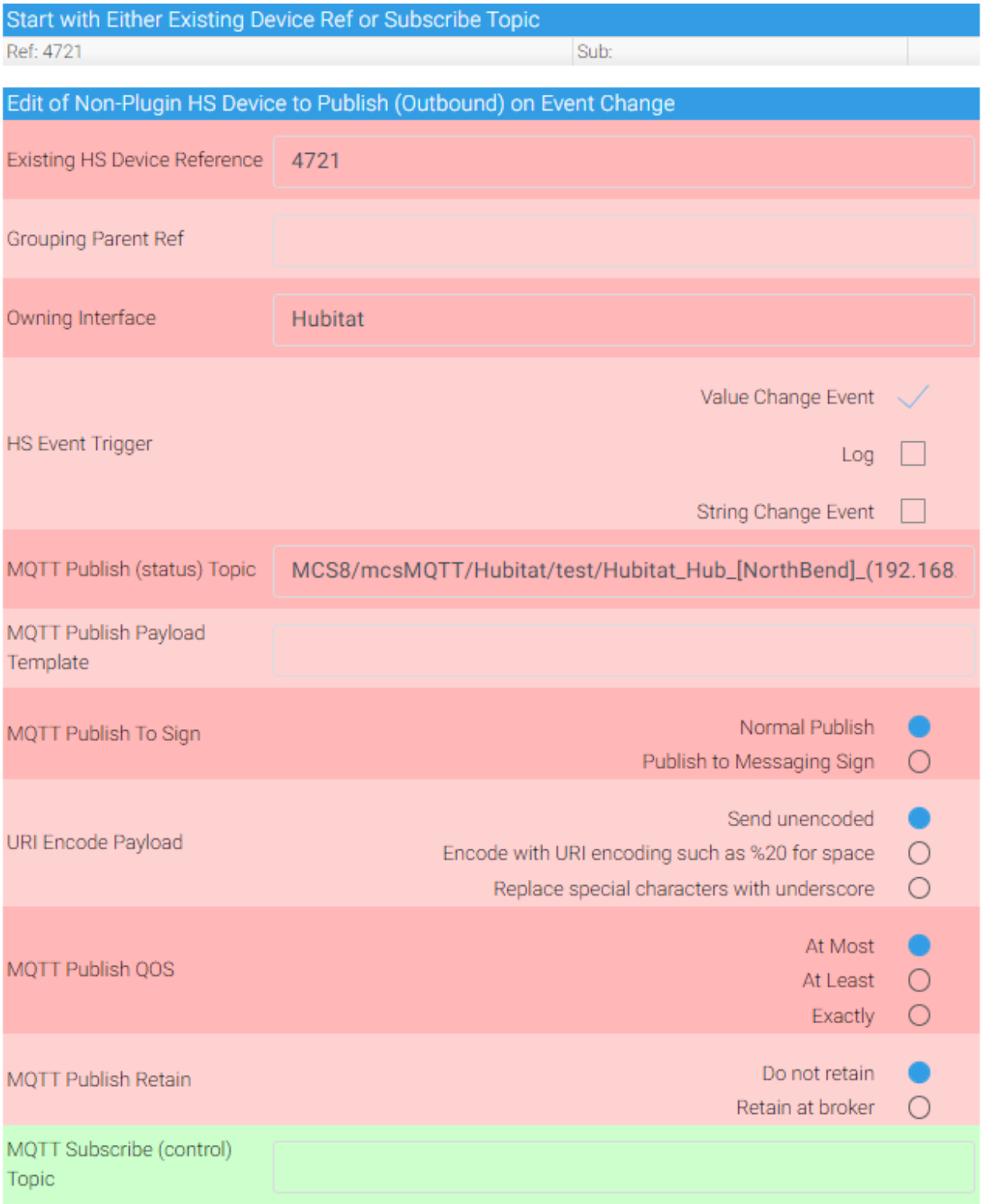


Figure 28 Non-Plug-in Device Manual Setup

Messaging Sign Text Properties	
Sign Display Row	1
Message Duration (minutes)	1
Text Color RRGGBB	007F7F
Default Text	C:\HomeSeer\Data\MyPlugin\Chart.jpg
Image Processing for Sign	
JPEG Image Scaling %	200

Figure 29 Messaging Sign Properties

If the desire is to control this HS Device from MQTT subscription then the Edit tab or Associations Tab can be used to enter the subscription Topic in the text box provided. In Figure 27 and Figure 28 these boxes are currently blank so there will be no means to control Device 10 via MQTT, but Device 10 status will be reported via MQTT as the DeviceValue changes.

In the case where HS is to control via sending a MQTT Topic to an external entity that has published status the Association Tab is used to select the published Topic with the "A"ssociate checkbox. This will bring up a publish Topic text box where the command Topic is specified. See the last row of Figure 30 where Device 212 was created. In this example the status Payload that is on the Subscription (Sub:) Topic GarageDoor/Door is "CLOSED". Because this is one of the special case Payloads mcsMQTT will create a two-state button with labels CLOSED and OPEN and assign DeviceValues of 0 and 1 to these. The radio under the command Topic shows the label (i.e. CLOSED or OPEN) will be published when the DeviceValue changes between 0 and 1. Had the radio shown Value rather than Label then the published Payload would be 0 or 1.

Had the Payload been something like the "Online" shown in the second row of this figure, then the UI presented would not be a button, but a text box where text could be entered and this text would then be the Payload delivered. In this case publishing occurs on a DeviceString change.

The third and last case is where the Payload is numeric such as for Device 248. For this one the UI will again be a text box for entry of a number and the Topic will be published on a DeviceValue change with the DeviceValue in the Payload.

Association Table for Auto Association of MQTT Topic and HS Device

	R	A	Ref	Topic	Payload	H	D	LastDate
3	☐	☐		Sub: GarageDoor/INFO1:Module	Sonoff Basic	☐		2018-04-16 22:56:45
4	☐	☐		Sub: GarageDoor/INFO1:Version	5.9.1	☐		2018-04-16 22:56:45
5	☐	☐		Sub: GarageDoor/INFO1:FallbackTopic	GarageDoor	☐		2018-04-16 22:56:45
6	☐	☐		Sub: GarageDoor/INFO1:GroupTopic	sonoffs	☐		2018-04-16 22:56:45
7	☐	☐		Sub: GarageDoor/INFO2:WebServerMode	Admin	☐		2018-04-16 22:56:45
8	☐	☐		Sub: GarageDoor/INFO2:Hostname	GarageDoor	☐		2018-04-16 22:56:45
9	☐	☐		Sub: GarageDoor/INFO2:IPAddress	192.168.0.70	☐		2018-04-16 22:56:45
10	☐	☐		Sub: GarageDoor/INFO3:RestartReason	Software/System restart	☐		2018-04-16 22:56:45
11	☐	☐		Sub: GarageDoor/POWER	Off	☐		2018-05-12 15:03:03
12	☐	☐		Sub: GarageDoor/RESULT:POWER	On	☐		2018-05-07 19:00:25
13	☐	☐		Sub: GarageDoor/cmnd/Door	OPEN	☐		2018-05-12 15:03:02
14	☐	☐		Sub: GarageDoor/STATE:Time	2018-05-20T00:44:24	☐		2018-05-19 16:44:24
15	☐	☐		Sub: GarageDoor/STATE:Uptime	32 32	☐		2018-05-19 16:44:24
16	☐	☐		Sub: GarageDoor/STATE:Vcc	3.192	☐		2018-05-19 16:44:24
17	☐	☐		Sub: GarageDoor/STATE:Wifi:AP	1	☐		2018-05-19 16:44:24
18	☐	☐		Sub: GarageDoor/STATE:Wifi:SSId	U	☐		2018-05-19 16:44:24
19	☐	☐		Sub: GarageDoor/STATE:Wifi:IPAddress	192.168.0.70	☐		2018-05-19 16:44:24
20	☐	☑	207	Dev: |GarageDoor|LWT Sub: GarageDoor/LWT Pub: the following Topic on Device command	Online	☐	☐	2018-05-19 20:00:43
21	☐	☑	248	Dev: |GarageDoor|STATE:Wifi:RSSI Sub: GarageDoor/STATE:Wifi:RSSI Pub: the following Topic on Device command	68	☐	☐	2018-05-19 16:44:24
22	☐	☑	212	Dev: |GarageDoor|Door Sub: GarageDoor/Door Pub: the following Topic on Device command GarageDoor/cmnd/Door	CLOSED	☐	☐	2018-05-19 16:44:24

Figure 30 Plug-in Device Subscription Association

When a subscribed Topic has been selected with the “A”ssociate checkbox the Edit tab Subscription table is populated with the same information as the row of the Association Tab table. This allows further editing if the default behavior is not as desired. It can also be later edited by entering the “GarageDoor/Door” subscription Topic as shown in Figure 31.

The Client tab has default setting for the publish Topic in the Outbound Management section. When this text box is not blank then the Pub text box will be automatically completed with this default. Since all publish topics will be different the default publish Topic is normally defined using substitution variables. See Section 5.2. A reasonable Topic template for Tasmota devices is “$$TASMOTACMND:”. For Shelly device it is “$$TOPIC:/command”. If your location has primarily one family of devices that share a common command topic structure then using the template will be beneficial.

Edit Setup or Edit of Subscription (Inbound) to a MQTT Topic	
MQTT Subscribe Topic	GarageDoor/Door
Payload RegEx Match Pattern	
Payload RegEx Replace Pattern	
Payload RegEx Operation	◉ Replace Match Pattern with Replace Pattern ○ Extract Match Pattern
Low Pass Filter	Filter sensitivity of 1 (range is 0.00 to 1.00 (most sensitive))
Expression	
Add Rate Device	☐ Create a HS Rate Device with rate sensitivity of 0.75 (Range 0.00 to 1.00) ○ Per Second ○ Per Minute ◉ Per Hour
Add Accum Device	☐ Create a HS Accum Device ○ No Reset ○ Accumulation Since Midnight ◉ Delta Since Midnight
Settings for Plugin Device	
HS Device Publish Topic	GarageDoor/cmnd/Door
HS Device Control/Status UI	○ Unspecified ○ Button ○ Number ◉ Text ○ List ○ ColorPicker ○ ColorXY
HS Device VSP List	0 OPEN 1 INDETERMINATE 2 CLOSED Clear existing VSP
HS Device MISC Properties	☐ NO_STATUS_DISPLAY ☑ NO_GRAPHICS_DISPLAY ☐ AUTO_VOICE_COMMAND ☐ SET_DOES_NOT_CHANGE_LAST_CHANGE ☑ SHOW_VALUES ☐ STATUS_ONLY
Grouping Parent Ref	Create New Parent Device
Publish Payload Template	
Publish QOS	○ At Most ○ At Least ◉ Exactly
Publish Retain Flag	◉ Do not retain ○ Retain at broker
Settings for Non-Plugin Device	
Control non-Plugin HS Device	
Note additional customization of buttton text, display graphics, button/number relationships etc. is done via HS Device Management Page by clicking on the Device Name link and using the Status Graphic and Configuration tabs	

Figure 31 Plug-in Device Publish Setup

5.1 Send MQTT via HS Device Change

Any HS Device that has a publish Topic defined can have a MQTT message published whenever the Device changes Value or String. If the HS Device is owned by mcsMQTT (i.e. created by mcsMQTT by associating it with a subscription Topic), then the either Value or String change can cause the Topic to be published. The decision on which is controlled by the HS Device Control UI as shown in Figure 31. It is

also dependent on the publish payload template also shown in Figure 31. Table 2 contains a list of substitution variables. Those applicable to commands from HS controlling plugin devices are described Table 1.

For non-plugin devices where the publish action is initiated by the HSEvent callback the same substitutions are used. If no substitution variable is used then no replacement will occur in this case. In addition, $$PREVIOUS: and $$PUBLISHED: variables are also available for the previous HS DeviceValue and last published payload respectively.

Table 1 Plugin Device Control Publish Template Options

Template Replacement Variable	**Text to Send in the Payload**
none	Use Edit tab VSP key based upon the number provided by HS. The key will be the subscribed topic payload value. For example the VSP entry of “true=1;On” will use “true” if there is no replacement variable or it is $$VSP: in the template.
$$VSP:	Same as none. This is the default
$$LABEL:	Use the text provided by HS for control. This will be the HS status value that is normally shown on the button label. It is normally the same as the VSP key, but not always for the case when the control is a HS selector pulldown.
$$STATUS:	Use Edit tab VSP status based upon the number provided by HS. The status will be the last text on the VSP line. For example the VSP entry of “true=1;On” will use “On” is $$STATUS: is used in the template.
$$VALUE:	Use the number provided by HS for the control

If set to Button or Number then the Value change of the HS Device will result in publishing the Topic with the new DeviceValue in the Payload. If it is Button then either the numeric Value will be put in the Payload or the button label as specified by the radio in the same figure.

The Topic will be published with the new DeviceString as the Payload If the selection is Text. The String change of the HS Device will be the event to publish the Topic.

If no publish Topic is setup such as Device 207 in Figure 30, then any change of this Device will not result in any MQTT published Topic.

A UI is provided on the Device Management page to allow manual entry of the Payload that is desired to be published. The same UI can exist on custom pages. It is also possible to publish on changes of the Device Value or String. This can be from another Plug-in, script, or event action. An example is provided that has Device 138 associated with a MQTT published Topic and running this event will cause the Topic to be published. See Figure 32.

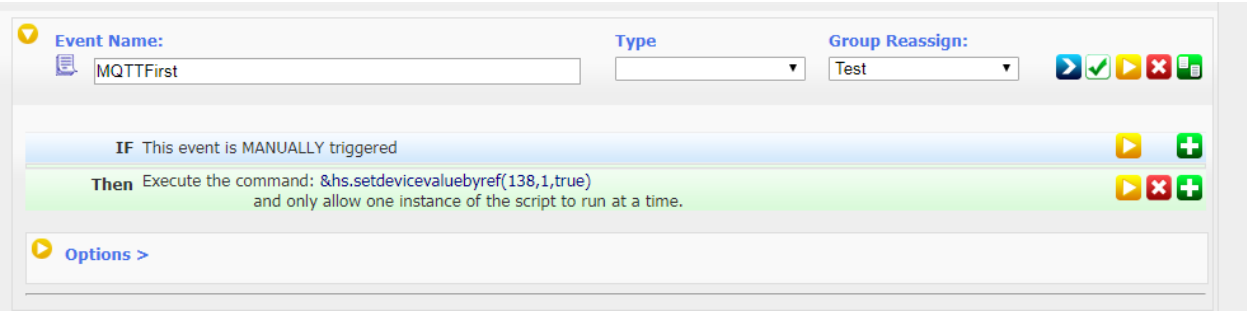


Figure 32 Example of Controlling a mcsMQTT Device

5.2 Send User-Customized Topics and Payloads

The publish behaviors described in Section 5.1 describe the default behaviors of the plug-in with static user-defined Topics and unformatted and fixed Payloads. Both of these can be customized. Figure 27 actually shows a Topic and Payload that uses the customization via set of defined replacement variables. These variables have the syntax of $$xxx: where xxx is the name of the replacement variable. The set of replacement variables support is shown in Table 2.

Plug-in defaults for both published Topic and Payload follow the plug-in convention of ComputerName/mcsMQTT/Loc2 /Loc/Name. The payload is the unformatted DeviceValue, DeviceString, or enumerated Value-Status-Pairs. Use of these defaults is achieved by leaving the Outbound (Publish) Management section entries for both of these blank. See Figure 40.

The actual Topic used when publishing is specified by the user as described previously. The default will show up when a HS device has been associated for MQTT publications. This can then be edited to whatever is needed on a Device-by-Device basis. To minimize the user entry burden, the default Topic template can be customized with the entry on the Client Tab. What is shown now in Figure 40 is a customization for "$$computer:/$$room:/$$name:". Every time a HS Device is "A"ssociated this template will populate the status Topic text box. The published messages will look something "MyPC/Room1/Try1" for the example shown in Figure 27. Note that "/" are used in the Topic template as part of standard MQTT Topic formats. Any text in the template that is not a substitution variable will become part of the Topic.

The $$TasmotaCmnd: substitution is a somewhat special case variant of the $$Topic: substitution. It will take the received Topic and insert "cmnd" to turn it into a publish command. If the topic starts with tele/ or stat/ then it will replace these with cmnd/, otherwise it will insert "cmnd/" before the last leg of the topic such as Topic "Light/Power" will turn into "Light/cmnd/Power".

Table 2 Substitution Variable List

Topic-Oriented Substitution Variables	
$$ADDRESS:	HS Device Address property
$$CODE:	HS Device Code property

$$COMPUTER:	Network name of the host computer
$$FLOOR:	HS Device Floor which is also the Location2 property
$$INSTANCE:	Instance number of the plug-in Interface property
$$INTERFACE:	HS Device Interface (Plug-in) property
$$NAME:	HS Device Feature Name
$$PARENTNAME:	HS Device Name
$$REF:	HS Device Feature Reference Number
$$PARENTREF:	HS Device Reference Number
$$ROOM:	HS Device Room which is also the Location property
$$TYPE:	HS Device Type String property
$$TASMOTACMND:	Topic to publish command to Tasmota device
Payload-Oriented Substitution Variables	
$$DATE:	Current date in short format
$$DEVICETYPE:	Device Type (String) property
$$TAG:	Free form text field on Edit tab of an associated device
$$DEVICESUBTYPE:	DeviceType SubtypeDescription property
$$PAYLOAD: $$PAYLOAD:(Topic): $$PAYLOAD:(JSONKey):	Last received MQTT Payload Last received MQTT Payload of any Topic (e.g. My/Topic:item:subitem) Last received MQTT Payload of this Topic (e.g. item:subitem)
$$PAYLOAD_EUROPE: $$PAYLOAD_EUROPE:(JSONKey): :	Same as $$PAYLOAD except numeric results with comma are converted to numeric results with decimal.
$$STATUS:	Status label as defined for status (3rd parameter) for VSP on mcsMQTT Edit Tab
$$CONTROL:	HS feature control label (2nd parameter) from Edit tab VSP
$$LABEL:	HS CAPI label (available only when using CAPI for control)
$$VSP:	Button label as defined for key (1st parameter) for VSP on mcsMQTT Edit Tab

$$STRING:	HS Device String
$$TIME:	Current time in short format
$$YEAR:	Local Current Year number
$$MONTH:	Local Current Month number
$$DAY:	Local Current Day of Month number
$$HOUR:	Local Current Hour number
$$MINUTE:	Local Current Minute number
$$SECOND:	Local Current Second number
$$UTCYEAR:	UTC Current Year number
$$UTCMONTH:	UTC Current Month number
$$UTCDAY:	UTC Current Day of Month number
$$UTCHOUR:	UTC Current Hour number
$$UTCMINUTE:	UTC Current Minute number
$$UTCSECOND:	UTC Current Second number
$$LASTCHANGE:	HS DeviceLastChange property using default date/time display format
$$TOPIC:	Last received MQTT Topic
$$VALUE:	HS Device Value
$$VALUE_EUROPE:	HS Device Value with comma replaced with period. When numeric expressions are used the fractional part of a number needs to be separated by a period.
$$PREVIOUS:	The previous (i.e. current before change) DeviceValue that exists at the time a DeviceChange event is occurring. It is typically used in conjunction with the inline expression function “IfChange” such as <<IFCHANGE($$VALUE:,$$PREVIOUS,”5%”)>>
$$PUBLISHED:	The published payload that was last sent. It is typically used in conjunction with the inline expression function “IfChange” such as <<IFCHANGE($$VALUE:,$$PUBLISHED,”5%”)>>
$$CAPIVALUE:	Last Value change request sent through CAPI interface used by mcsMQTT SetIOMulti

$$CAPISTRING:	Last String change request sent through CAPI interface used by mcsMQTT SetIOMulti
$$HSEVENT:	Last parameter delivered by HS on HSEvent callback. This will typically be the new DeviceValue or DeviceString
HS Device-Oriented Substitution Variables	
$$DVA:(address):	Use device value. Identify device using address.
$$DVC:(device code):	Use device value. Identify device using device code. (HS3 only)
$$DVR:(reference):	Use device value. Identify device using device reference.
$$DTA:(address):	Use device string. Identify device using address.
$$DTC:(device code):	Use device string. Identify device using device code. (HS3 only)
$$DTR:(reference):	Use device string. Identify device using device reference.
$$DSA:(address):	Use device status. Identify device using address.
$$DSC:(device code):	Use device status. Identify device using device code. (HS3 only)
$$DSR:(reference):	Use device status. Identify device using device reference.
Date and Time Substitution Variables	
$$DATEL:	Date in long format (e.g. Tuesday 10 September 2019)
$$TIMEL:	Time with seconds (e.g. 2:00:13 PM)
$$DATE:	Date in short format (e.g. 9/10/2019)
$$TIME:	Time in short format (e.g. 4:52 PM)
$$UNIX:	Time in UNIX format (seconds since 1970)
Other Substitution Variables	
$$DEG:	Degree Symbol (Decimal 176)
$$WANIP:	WAN-facing IP address, updated every 10 minutes when used
$$SECRETKEY:	User entry on the URL tab or Cloud page for UDP communication used for token compuations. Substitution is in context of the URL Topic for which it was entered
$$JWT:(key):.	JSON Web Token (JWT) contains information that may be

	needed in endpoints of the URL endoints. For example, if "user_id" is needed in the endpoint or the URL (or in the data payload) then it can be references such as: URL/https://myServer/user/$$JWT:(user_id):/status

The Payload format can be customized as well. In this case if a template is entered for the payload, then the DeviceValue, DeviceString or VSP Status that would be published in the no-template case will no longer be part of the Payload unless $$VALUE: or $$STRING: or $$STATUS: is used as part of the template. As a further example the following could be specified for a JSON-encoded Payload of three items that reflect the Status of the HS Device that had just changed.

{Name:$$NAME:, State:$$STATUS:, TimeStamp:$$DATE: $$TIME:}

Topics are always published using URL encoding to assure MQTT reserved characters are not included in the Topic. URL encoding replaces all character codes except for letters, numbers, and the following punctuation characters:

- - (minus sign)
- _ (underscore)
- . (period)
- ! (exclamation point)
- * (asterisk)
- ' (apostrophe)
- (and) (opening and closing parentheses)

Payload is not URL encoded unless explicitly requested in the publish template. This is done by using substitution variables in the publish template and including "_URI_ENCODE" in the substitution variable name. Alternately a replacement of spaces with underscores is specified by including "_UNDERSCORE" in the substitution variable name.

For example, consider the case where a HS Device String has content of "A'B C".

If the publish template has $$STRING: then A'B C would be published.
If the publish template has $$STRING_URI_ENCODE:" then A%27B%20C would be published.

If the publish template has $$STRING_UNDERSCORE:" then A'B_C would be published.

The "_URI_ENCODE" and "_UNDERSCORE" suffix can be used with any of the substitution variables.

It is possible to use expressions in the expression text box for inbound messages. They can also be used in event action payload or device payload template by encasing the expression in "<<" and ">>". Nesting expressions to two levels is supported.

Examples:

$$DVA:(PI_RELAY-R1):	Substitute using DeviceValue from device @ address “PI_RELAY-R1”
$$DTC:(S33):	Substitute using DeviceString from device @ code S33
$$DSR:(123):	Substitute using DeviceStatus from device @ reference 123
<<$$DVR:(123):*2.55>>	Scale DeviceValue of device @ reference 123 by 2.55

5.3 Send MQTT via Event Action

The second mechanism to send an MQTT message is with HS Event Actions. Figure 33 shows both the setup of an Action (MQTTTrigger2) and how an Action will be shown that has been setup (MQTTTrigger). The user entry, as shown by MQTTTrigger2, is of the format “Topic=Payload”. The first “=” is used to separate the two components of the message. The QOS will always be the default setup on the Client Tab for any MQTT message sent as an Event Action.

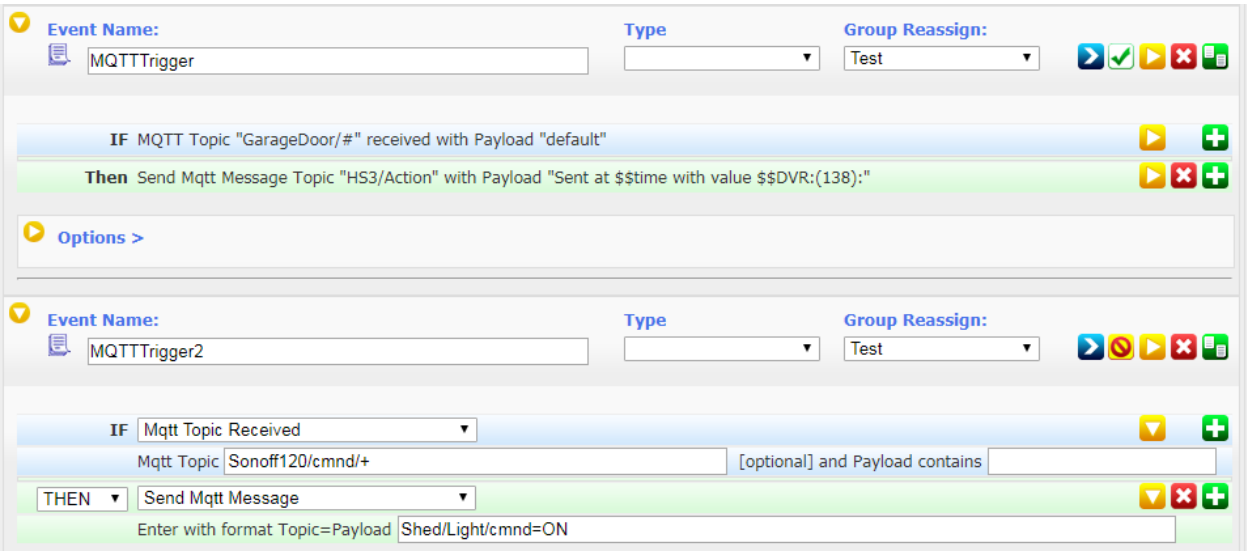


Figure 33 MQTT Event Action

The MQTT Payload of Event Actions support substitution variables. An example use is show in Figure 34 where the “$$time:” and “$$DVR:(ref):” variables are used. In this example the Payload will include the current time and the DeviceValue of Device with reference 138. The Value, String and Status of a device are available for substitution. The reference, address and device code are the available methods to identify a device. A six-character mnemonic is used to specify each of the nine combinations. These are shown in Table 2.

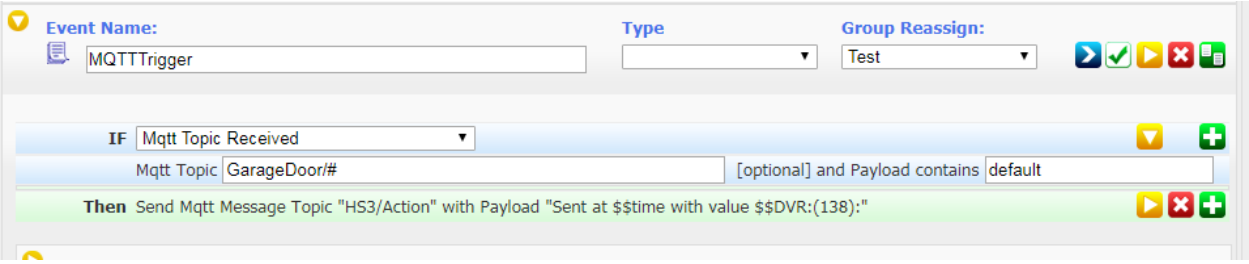


Figure 34 Substitution Variables in Event Action

It is also possible to send a set of MQTT messages from the HS Event Action. This is done by sending a publication list that contains the message list. See Sections 5.7 and 18.8 for description and use of publication lists. When the action is setup the list of created publication lists will be available from the selector and if one has been previously selected for this event, it will show as the default. See Figure 35.

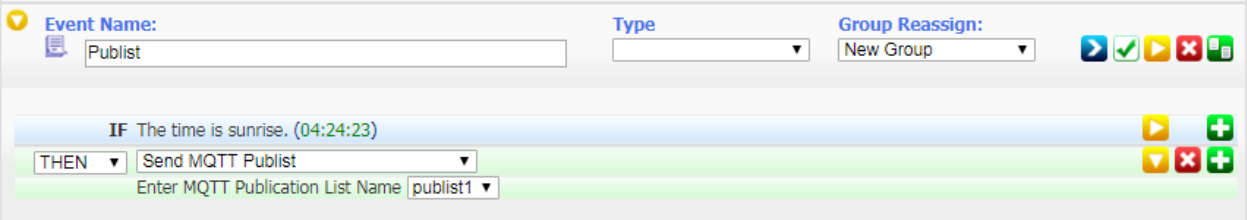


Figure 35 Send MQTT Publication List Event Action

A similar capability exists for requesting WLED playlist where the playlist name rather than the publist name is specified.

Voice Monkey is able to do text to speech on Echo devices and images/videos on Echo Show devices. The event action supports both capabilities with the default being TTS as shown in Figure 36. Other options with Voice Monkey use JSON to specified the desired set of parameters. These are

- notification – if set to “true” then the Echo device LED ring will be illuminated and “play notification” command used to hear the announcement and clear the LED
- image – URL of an Echo Show compatible image
- video – URL of a mjpeg file for display on Echo Sho
- websiteurl – URL of text from a web site

An example for picture with notification would be:

kitchen-show={“image”:http://myShapshotServer/last.jpg,”notification”:true}

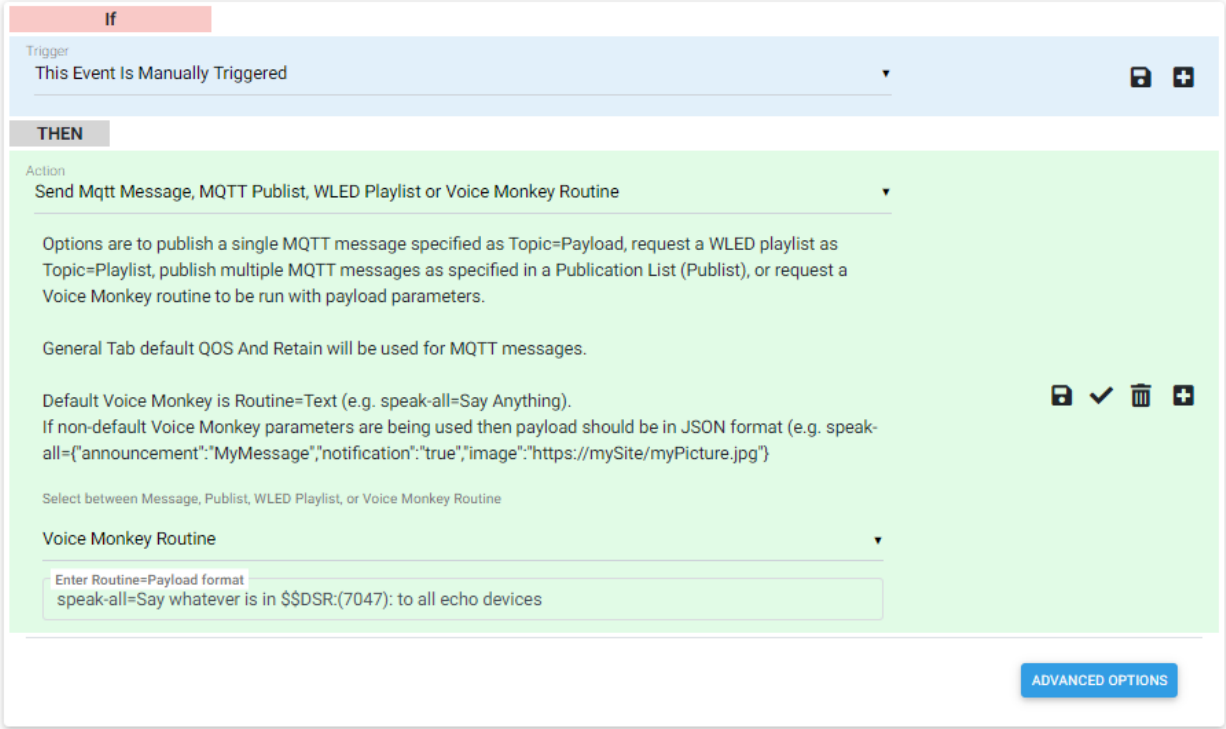


Figure 36 Play Voice Monkey Routine

Note that Voice Monkey can also be commanded via MQTT received messages rather than HS Event actions. In this case the MQTT Topic format is “voicemonkey/routine” and payload similar to that used in the event action text box. In this mode of use, mcsMQTT becomes the server to bridge MQTT messages to Voice Monkey API.

5.4 Send MQTT via Script

The third mechanism to send MQTT message is via scripting. In this case the HS scripting command “PluginFunction” will be used to get the mcsMQTT object and the mcsMQTT function “SendMQTTMessage” will be used to send the message.

The prototype for “SendMQTTMessage” is

```
Public Function SendMqttMessage(ByVal sArray() of String) As Integer
```

A positive sequential number is returned with success and a -1 on failure. The array contains four entries:

1. Topic
2. Payload
3. QOS
4. Retain
5. Optional Broker index (0.. number of brokers – 1)

The QOS parameter has the following enumerated values

"AT_MOST_ONCE"
"AT_LEAST_ONCE"
"EXACTLY_ONCE"

The Retain parameter contains the following values

"FALSE" – Do not ask broker to retain the message
"TRUE" - Ask broker to retain the message for transmission to new subscribers

An example is below where a positive value return as an incrementing number and a negative value is returned if there is some form of failure.

```
Sub Main(parm as object)

        Dim iResult as integer
        Dim sTopic as string = "Sonoff/Bedroom/Light/cmnd"
        Dim sPayload as string= "OFF"
        Dim sQOS as string = "EXACTLY_ONCE"
        Dim sRetain as string = "FALSE"
        Dim sArray() as string = {sTopic,sPayload,sQOS,sRetain}

        iResult = hs.PluginFunction("mcsMQTT","","SendMqttMessage",sArray)

        if (iResult < 0) then
                hs.Writelog("mcsMQTT","MQTT Message Failure for " & sTopic)
        end if
end Sub
```

A second form of the scripting send function is to send to the Messaging Sign. Its prototype and array parameters are:

```
Public Function SendSignMessage(ByVal sArray As String()) As Integer
    'return -1 on failure
    '0-Topic
    '1-Text
    '2-Row
    '3-Duration
    '4-Color
    '5-QOS
    '6-Retain
```

Failure is indicated as a -1 returned value. Its use is similar to the basic send function but includes the details of information going to the sign in additional parameters.

5.5 Send Status on MQTT Request

Any HS device that has a Publish Topic defined can be queried to report its current status from a MQTT message with the same Topic as the Publish Topic and an empty Payload. The current status that will be either the DeviceValue or DeviceString depending upon which HS Event Trigger has been definded for the device on the Edit tab. See Figure 28.

In this example of a non-plugin device where the status is reported on Topic Test/Status and it will report the DeviceValue. This status will be published every time the DeviceValue changes as well as when the Topic Test/Status is received with a blank Payload.

The status query can also be requested by using the Subscribe (control) topic. In this second option non-plugin devices that are status only will provide no response because no control topic will have been defined.

When Publish (stauts) topic is used then the broker will likely return the same message to the sending client. There is no guarantee what order the broker returns to the sending client the topic sent by the sender vs. the topic sent by mcsMQTT. If mcsMQTT is first then the "final" status will have a blank payload. It could also result in an event in the sending or other clients as the status response transitions between null and current status.

5.6 Sending Periodic Status

All non-Plug-in HS devices that have been “A”ssociated will have their current DeviceValue or DeviceString published on the interval setup on the Client Tab. This serves to assure subscribers are refreshed following unexpected mcsMQTT disconnects when the QOS will not otherwise provide this capability.

5.7 Sending Configuration / Setup Messages

A set of Topic/Payload messages can be defined that contain the configuration of an IOT device. The list is a text file located in the folder \Data\mcsMQTT and of file type “.pub”. Each line of the file will have a message defined as Topic=Payload. The first four lines of the file will contain the substitution variable definitions.

The Pub List tab is used to start the transmission of the messages. This tab can also be used to create or edit existing publication list files. The file of interest is selected from a pull-down. If a new file is to be created then the text box to create a new file is used to specify the file name.

Provisions exist for substitution variables $$1: through $$4: to allow a single .pub file to contain a template which is instantiated based upon the values entered for the substitution variables. Substitution variables do not need to be used, but their position in the .pub file does need to exist even if the substitution value is blank.

Edit provisions via browser also exist. This can be used to create of modify a .pub file.

In the Figure 37 example a file of name LoRa.pub exists to define the frequency of three LoRa units. The frequency is defined as $$1: to be 915. Three messages are setup to transmit this frequency to three IOT devices. The Execute Publication List is used to initiate the publication.

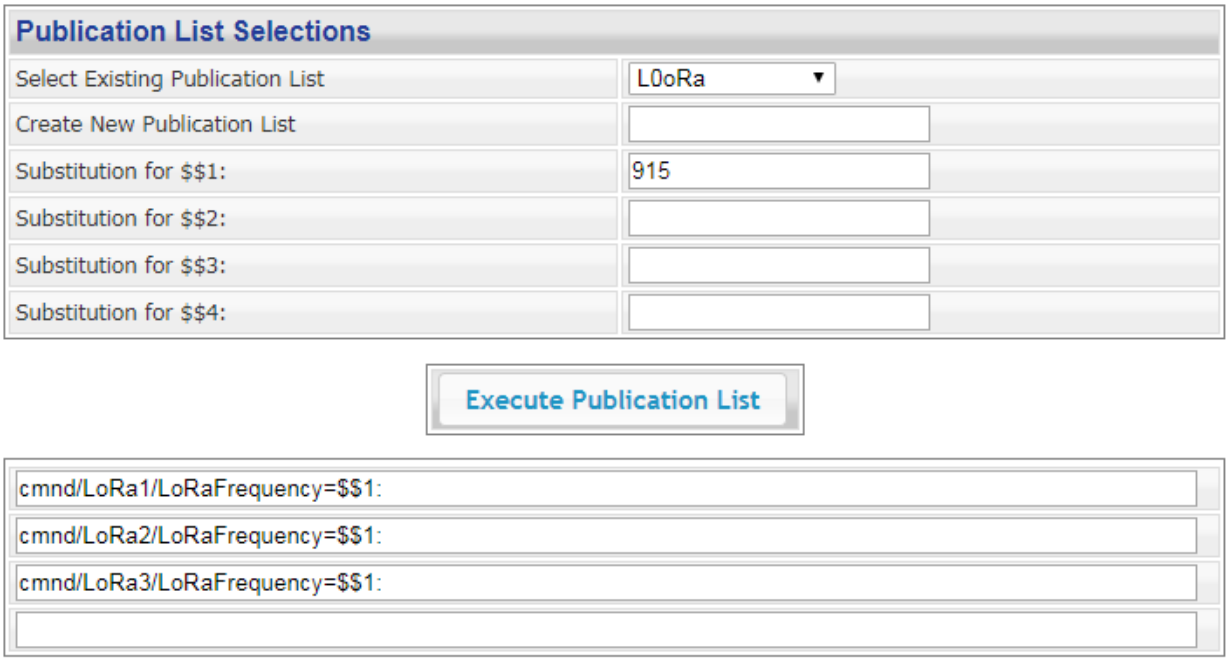


Figure 37 Publication List to Setup a Lora Frequency

5.8 Sending Messages to LED Messaging Sign

The Messaging Sign whose API is described in Section 21.19 will display text and images on a matrix of color LEDs. Provisions have been made in mcsMQTT to facilitate use of this sign with HS.

On the Edit tab for plug-in devices a Control/Status UI of type Sign exists to indicate that published information will be formatted for this sign. A similar provision exists for non-Plug-in devices to publish to the sign.

The sign has on-board provisions to store twelve messages for display. The first four are stored in flash so will persist a power cycle of the sign. The other eight will not persist a restart. Messages are updated via MQTT topics. The twelve are identified by the topic PTEXT for persistent and TEXT for volatile. Each is prefixed with /CMND/ and suffixed with index 1 through 4 or 1 through 8. (e.g. LedSign/cmnd/Text8" to indicate the 8th volatile text message.

When a Sign type selected and the "A"ssociate checkbox used to create to HS device then buttons are created to correspond to each display row on the sign. See Figure 38. The status will indicate the row a message is being displayed on the sign. The button will cause a message to be sent to the specified row of the sign or to remove the message from the sign's buffer. The color and duration parameters from the setup of the sign are also sent.

Figure 38 Control/Status UI setup in Device Management to Support Sign Type

The user can use the buttons on the UI or it can store into DeviceValue the 0, 1 etc. to affect the desired result. The sign's feedback will indicate which row the message is showing. If the HS DeviceString is null then this status will appear on the Device Management page. The row status is also available in the DeviceValue.

The color and other parameters for the sign are entered from mcsMQTT from the Edit Tab on the hyperlink "Configure Sign Parameters". These parameters include the duration in minutes that a message will be retain for display, the row on which the message will be shown and the color of the message. The color can also be embedded in the text of the message using "[RRGGBB]" notation to change the color of the text for the subsequent characters. A default text string that can be used for text or image path can also be entered and will be used if publish Template and HS DeviceString are null.

MQTT has a limit of 128 characters for a payload. It the message exceeds this then mcsMQTT will split it into multiple transmissions and the sign will reassemble them so it appears that a long message has been received. The limits of the sign are 80 characters for non-volatile (PTEXT1 through 4) and 320 characters for the others (TEXT1 through TEXT8). mcsMQTT will enforce this and truncate extra characters so a user need not be concerned about message length for a successful transmission.

The sign also has provisions for display of images. If the payload of a message going to the sign is a file path to a jpg image, then mcsMQTT will transmit the binary for the jpg per the API defined in Section 21.19.2. The image can be scaled by a percentage of the capability of the sign's resolution. By default, this resolution is 16 x 40 pixels. 10,000 bytes are reserved in the sign to hold the compressed jpg image. If mcsMQTT recognizes that the image specified with scaling will not fit then it will scale it progressively smaller in 10% increments until it will fit prior to transmission.

Experience has shown that good results occur when the image is 100% to 200% of the screen size. There will be sufficient content at any point in time as the image pans to be able to recognize the image. Lower scaling may result in poor recognition because of the resolution of the screen. Higher scaling will result in difficulty in recognizing the image content in the viewport provided by the sign.

Provisions have been made to extract information from the HS log to be sent to the screen. This is described in Section 5.8.1.6 as well as more in the introduction to Section 6. Otherwise, the text of the message or the file path of the image are generally contained in the DeviceString of the device that has been assigned the Sign as its Control/Status type. All HS log entries sent to the sign are done on topic TEXT8.

Provisions also exist to monitor a file name or file folder for an update to a jpg image and then send the updated image to the sign. This is described in Section 5.8.1.8. All images are sent using the IMAGE topic.

When using the Sign, a strategy should be developed for how each row is to be used. For example, the first row is for messages that need user attention and the second row is for messages that are just informational.

The Sign will automatically scroll through each of its twelve message buffers and show any that have a non-zero duration. This will be done in a round-robin manner with a three-character space between each message.

The Sign Duration parameter should be selected so that stale messages are removed automatically and set to 65535 for those that need user attention so they can be manually acknowledged. Use of PTEXT rather than TEXT as the Topic is also appropriate for important messages that need to persist power cycles or restarts of the Sign.

5.8.1 Messaging Sign Use Cases

The following discussion provides concepts of how mcsMQTT can support use of the Messaging Sign. In all cases HS devices will exist with the Control/Status UI having been selected as the type. For plug-in devices this is the radio on the Edit page. For non-Plug-in devices this is also a radio on the Edit page label "Publish To Sign" as shown on Figure 28.

5.8.1.1 Send Text to Sign when HS DeviceString Changes

When the DeviceString of any HS device receives a change in its content then the text of the string will be stripped of any HTML encoding and sent to the sign. The duration that the text will continue to be displayed, color information and the row on the sign and on the sign are setup from the "Configure Sign Parameters" hyperlink on the Edit tab. See Figure 28 and Figure 29. The topic used is taken from the Publish topic that is setup on either the Edit tab or the Association tab.

If the Sign's corresponding status topic has been associated to a HS device then the Sign will acknowledge receipt of the text by echoing the row number on which the text was displayed. For example, if published to topic LEDSign/cmnd/TEXT2 then the Sign will publish acknowledge status on LEDSign/TEXT2. If the duration countdown expires then the Sign will again publish LEDSign/TEXT2 but in this case the payload will be 0 to indicate that the text is no longer being displayed on the sign.

5.8.1.2 Send Augmented Text to Sign when HS DeviceString Changes

This scenario is the same as the first scenario but additional text is added to the DeviceString content to provide context of the text. An example is Caller ID being available from HS and it is to be sent to the screen when a call is received. The CID will be contained in a non-Plug-in DeviceString.

In this case the Publish Template from the Edit page is used to provide a prefix "CID:" along with substitution variable $$STRING:. The Publish Template text box would look like "CID:$$STRING:" mcsMQTT would then pull the DeviceString text and append it to the "CID:" prefix before sending the Sign. The duration parameter on the Sign Parameters popup (Figure 29) in this case would likely be relatively short such a one minute since the CID info only has immediate use. After one minute the Sign will remove the CID message.

5.8.1.3 Send Static Text to Sign

Consider a scenario where the text to be sent is predefined such as "Amber Alert" and an HS DeviceValue changes when the "Amber Alert" is occurring. The HS DeviceString will be null and the Publish template will be null. The Default Text in the Sign properties popup (Figure 29) is set to "Amber Alert".

For discussion let us use the LedSign/cmnd/TEXT2 as the Publish Topic and Sign Duration of 65535 to indicate that the message will continue to be shown until manually removed. The plug-in device "A"ssociated with Topic LedSign/TEXT2 will have a "Off" button that can be used to do the removal. Alternately the DeviceValue of the plug-in device can be set to 0. Either of these actions cause a

LedSign/cmnd/TEXT2 Topic to be sent with a payload containing a JSON content of Duration being 0 that supersedes the 65535 minutes originally sent.

5.8.1.4 Send Text to Sign via Script

In this scenario consider an irrigation Plug-in such as mcsSprinklers where the predicted date of the next irrigation cycle is desired. In this case the mcsSprinklers Plug-in script function needs to be called to get the predicted date and then from the script one would use the mcsMQTT Plug-in function SendSignMessage to send a MQTT message per the JSON format expected by the sign. See Section 5.4 for information on the scripting within mcsMQTT.

In this scenario the script formats the text as desired before passing it along with the parameters for row, duration and color used in the SendSignMessage Plug-in function. There is no dependence on any other HS device for this operation to function. Status feedback from the Sign does continue to be provided in the HS device "A"ssociated with the corresponding TEXT or PTEXT topic.

5.8.1.5 Send Augmented Text to Sign when DeviceValue Changes

In this scenario consider a HS DeviceValue that contains the daily energy use that is updated every minute. The implementation of this scenario is the same the one described in Section 5.8.1.2 except the substitution variable is $$VALUE: rather than $$STRING: to indicate that mcsMQTT should pull the value from DeviceValue rather than DeviceString.

Each time the DeviceValue changes an updated message will be sent to the sign to replace the one that was previously viewed. In this case the Sign Duration parameter could be set to something like 2 minutes so the message will be removed from the sign if it becomes stale.

5.8.1.6 Send all HS Log Entries to Sign Generated by Specific Plug-in

On the Publist/Sign tab Messaging Sign Setup section is a text box where a regular expression can be entered to specify a filter for HS Log entries. If, for example, the plug-in of interest is mcsSprinklers then it will place the text "mcsSprinklers" in the Log's type/error column. This is what would be used in the regular expression to indicate any occurrence of this text.

At this same location is a radio to enable use of the HS log as source of messages to the Sign.

The TEXT8 topic will be used for HS Log publications to the Sign. The Publish Topic will be setup to be something like "LedSign/cmnd/TEXT8". The Sign Parameters popup ((Figure 29) contains a radio to select HS Log entries to be sent to Sign.

The other Sign parameters should also be setup as desired. In the HS Log case the Color parameter is ignored and the color used by the HS Log will also be used by the Sign.

5.8.1.7 Send Jpeg Image to Sign

A Topic similar to "LedSign/cmnd/IMAGE" is used to deliver an image to the Sign. The image will supersede any text that is currently being displayed. A subsequent text message to screen will remove the image from the Sign.

The jpg file name can be contained in DeviceString or Sign Properties popup Default Text property. When a DeviceString changes, mcsMQTT will look at the last four characters for ".jpg" and send the file content rather than sending the text. If the file does not exist then nothing will be sent.

All the methods described in the text-oriented scenarios can be used for images but the Publish Topic should use the IMAGE rather than the TEXT or PTEXT keywords

5.8.1.8 Automatically Send Updated Jpeg Charts to Sign

On the Publist/Sign tab Messaging Sign Setup section is a text box where a file name or a folder name can be entered. mcsMQTT will monitor this location and every ten seconds when the last change date changes for a .jpg file then it will scale the image and send it to the Sign. The remaining setup is the same as used from HS Log monitoring described in Section 5.8.1.6 except file monitor radio is selected and the only other Sign parameter of interest is the image scaling percentage.

If a folder is being monitored then mcsMQTT will select the most recent file if multiple files change in the last ten seconds. Charts that have fine detail will perform poorly with this use due to the resolution of the Sign. Things like Area and Column charts will have more success than Line or Scatter charts.

5.9 Monitoring Ability to Send and Receive via Broker

mcsMQTT monitors the Broker connection on ten second intervals. If a connection is lost it will attempt reconnection. This will be followed by resubscription. This background activity should be transparent to the mcsMQTT user except for those who are using the Broker Connection Trigger. Events with this trigger can be setup for either disconnect or connect transitions as shown in Figure 39.

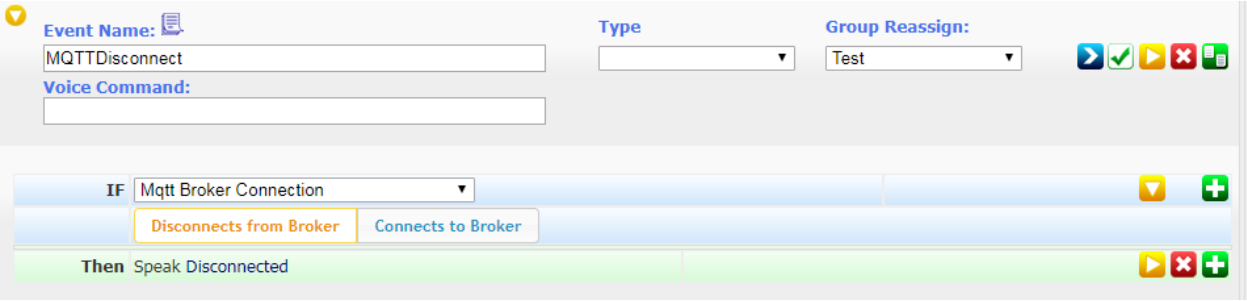


Figure 39 MQTT Broker Connection Event Trigger

6 Receiving MQTT Topics

Telemetry, sensors and other devices that provide current information via MQTT can be captured in HS Devices. Triggers can also be raised by either the reception of a Topic or after a Topic has not been received for a specified time.

The default mode of operation of mcsMQTT is that it is listening to all topics available from the broker. This is referred to as Discovery and can be deselected from the Client Tab Inbound (Subscription) Section of Figure 40. This mode is good when initially building the HS/MQTT relationships or when new MQTT topics are being introduced to the environment. When a stable environment has been achieved the mode can be changed to only "Listen for only Associated messages and those messages that are used as MQTT event triggers. This will reduce the volume of traffic being sent between the broker and mcsMQTT to be only those topics which will result in a change in HS devices. This also means that the Payload shown in Figure 30 will only be updated for the "A"ssociated devices.

The default is also that published messages are not shown on the Association tab rows. When the echo feature is enable on the Client Tab Inbound (Subscription) Section of Figure 40 any published message will be visible on the Association tab as a subscribed topic if Discovery is enabled. If Discovery is not enabled then the Topic can be made visible from the Manual page by entering the specific subscription.

Topics received that match the subscription template option will be available for management on the Association tab unless rejected by other settings. They can be hidden from view on the Association tab by using the "R"eject column checkbox. In this case the messages are still recorded in history, but not visible on the Association tab. The Client tab provides a Reject Topic Template where a set of wildcarded topics can be entered to firewall reception of these topis. In this case they are not recorded in the history.

When an Association of MQTT Topic is made to create a HS plug-in device the device will be placed in a Location2 (Floor) and Location (Room) as determined by the MQTT topic or can be at fixed locations if a default location is setup on the Client tab. This location can be edited from the Edit tab or can be edited in HS Device Management. Rather than the location being dependent on the MQTT Topic it can also be specified so that all new mcsMQTT devices go into the same Location2 (Floor) and Locataion (Room). The Client tab Inbound section provides the ability to make this choice and select the locations to be used.

The Payload items are stored in HS Features (HS4) or HS Child Devices (HS3). The Feature or Child Device will be organized under HS Devices (HS4) or HS Parent Devices (HS3). The Device into which the Features and Child Devices are grouped is normally created using the MQTT Topic hierarchy. Given a Topic of "Test/Me" there will be a Device Test and a Feature Me created when an Association is made.

Two options exist to modify this Device/Feature relationship within HS. One is on the Client Tab setting for "Default HS Parent Device". When this option is used any new association made will use a static Parent Device. This approach minimizes the number of Parent Devices that are created. Once a Feature has been created the parent to which is grouped can be changed from the Edit Tab, "Grouping Parent Ref" text box.

Inbound (Subscription) Management	
Topic Discovery	Discover All Published MQTT Topics ○ Listen for Only Associated and MQTT Trigger Topics ● Listen for Only using the Wildcared Template below ○
Inhibit Topic Discovery	Disable New Topic Recognition ☐
Subsample Topic Templates	
Subsample Min Secs	0
Reject Topic Templates	
Enable Auto Device Creation	Auto-create Recognized Topics (e.g. wled, shellies) ✓ Auto-create from HomeAssistant or Tasmota Discovery Protocol ✓
Wildcard Non-Plugin Control Template	Associate Topic to HS Device upon receipt of control Topic ● Associate all non-plugin HS Devices immediately ○
Wildcard Plugin Auto Associate Template	
Default HS Feature Location	Use Loc2 (Floor) & Loc (Room) based upon MQTT Topic ● Use Default Loc2 & Loc ○
Default HS Feature Name	Use last segment of Topic ● Use full Topic ○
Default HS Feature MISC	Only update Last Time when value changes ● Always update Last Time ○
Default HS Parent Device	Create HS parent device based upon MQTT Topic ● Use existing MQTT parent device ○
Echo	Do not process echo of transmitted topics ● Include transmitted topics in Association tab reception list ○

Express Mode	Default accepted Topics to full support ● Default accepted Topics to only store in HS device ○
Express Mode Features	Decode JSON into separate HS Devices ✓ Honor setup of device type and use VSP and Color Picker ✓ Process regular and arithmetic expressions ✓ Process Topic event triggers and script callbacks ✓
Obsolete Unassociated on Shutdown	Remove Unassociated records from database and tables on shutdown ✓
Remove Obsolete Topics	
Remove Retained at Broker	
Max Received Queue Limit	500
Max Received Processed Together	5
Pause ms for Full Queue	50

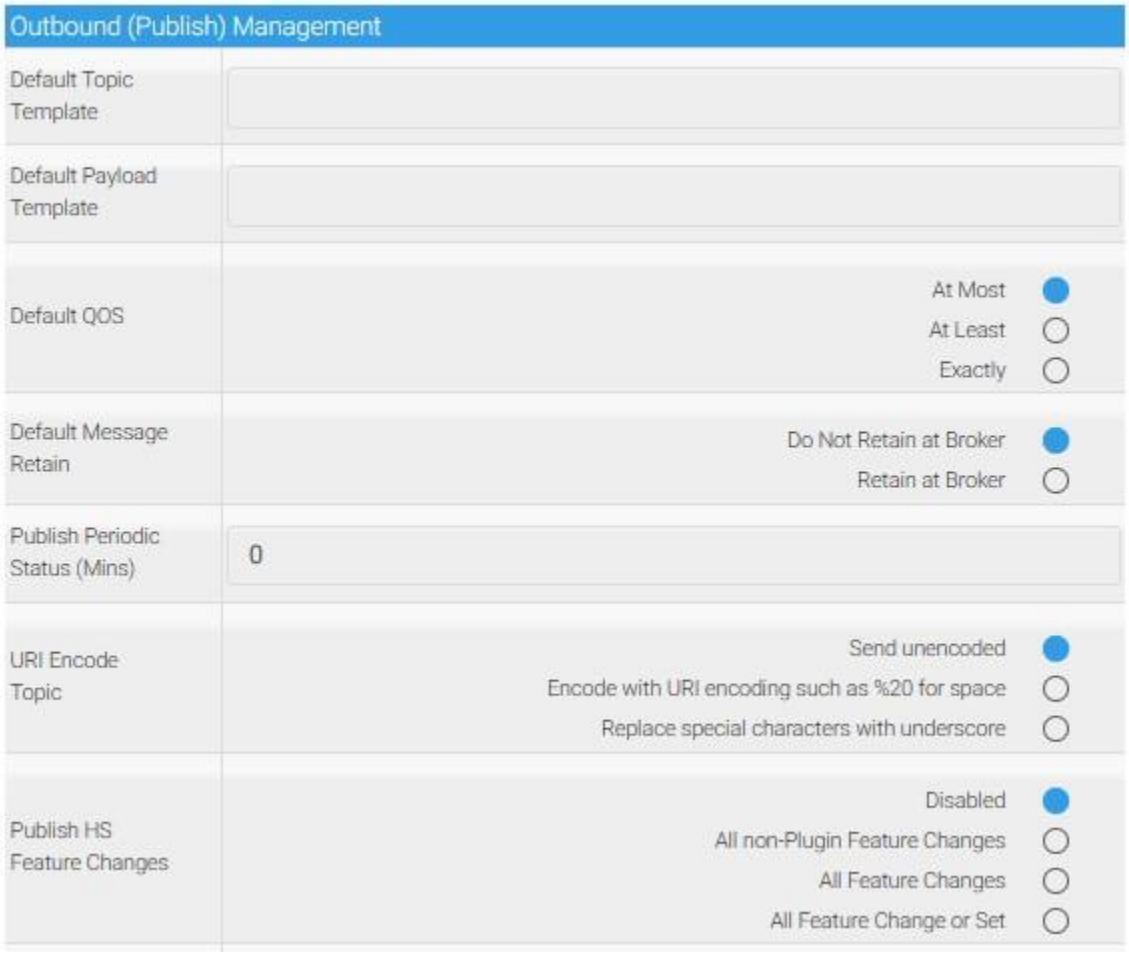


Figure 40 Client Tab Inbound and Outbound Setup Options

6.1 Receive MQTT Payload in HS Device

Figure 27 through Figure 31 are used to setup the mapping between a MQTT Topic and HS Device. The received Topics will be listed on the Association Tab and a HS device created for each where the "A"ssociated checkbox is used.

The received list will be formed from all the Topics that were delivered by the MQTT Broker with subscription to "#". In addition, special Topics can be specified in the Edit tab. This provision is to handle the special cases not covered by "#".

The MQTT Payload can be either raw text or it can be text formatted using JSON. mcsMQTT will parse the JSON and create a mapping for each JSON element into an HS Device. A ":" is used in the Topic name to indicate that part of the Topic is based upon the JSON Payload. In Figure 30 all rows were formed from one MQTT Topic with Payload encoded with JSON keys. The Topic "GarageDoor/Uptime" is shown as having JSON formatted Payload content for "Time" and for "Uptime". The ":" is used to identify that everything to the right of the ":" is part of the JSON Payload.

Take for example the following MQTT message. In this case there are two levels of JSON formatting. As one example a row will be shown for "Sonoff120/STATE:Wifi:AP" because the "Wifi" value is further decomposed into multiple components using JSON.

Topic: Sonoff120/STATE
Payload: {"Time":"2017-12-02T03:38:30", "Uptime":0, "Vcc":3.007, "POWER":"OFF", "Door":"CLOSED", "Wifi":{"AP":1, "SSId":"default", "RSSI":100, "APMac":"40:16:7E:A2:EE:68"}}

JSON data format consists of an optional group specification starting with "[" and ending with "]". Inside the group (if it exists) are sets of key-value pairs between a starting "{" and an ending "}". The key-value pair is separated with a ":". If the value is text then it is encased in quotes. If it is numeric then it is not encased in quotes. The key, since it is text, should always be in quote, but mcsMQTT is tolerant of keys that do not have quotes.

The last Payload for the Topic is shown and if "A"ssociated then it will also be placed in the mapped HS Device. If numeric it will be put in the DeviceValue otherwise in the DeviceString. Provisions also exist to setup Value-Status Pairs (VSP) so that DeviceValue rather than DeviceString is updated. This is desirable because HS has a richer set of features for Values than Strings.

It is common for an IOT device to respond to a command with some form of acknowledge and also for the device to periodically report its state. There two activities will be published on different topics but the represent the same information. Section 4.1.28 describes how multiple topics can report status into a single HS device.

mcsMQTT will observe the Payload history of a Topic and select VSPs, Numbers or Text as the most appropriate type of Payload mapping to HS Device. The selection can be changed on the Edit tab by selecting a radio for the HS Device Control/Status UI. VSPs are used for Button and List. HS Devices created with Button type will take a snapshot of the received text values to setup the VSP for the HS Device. They can be manually edited later. Devices of type List will continue to automatically update the VSPs as new text is received in the payload. These also can be manuallyed edited. Types of valuesNumber and NumberChange are used for numeric Payload. Type NumberChange will only update the HS DeviceValue if it has changed. Text is used for random strings. Color Picker is also available for special case of asking HS to show a color picker control for the Device. Three forms exist for the Color Picker. The ColorXY is used if the end point is an XY color space. HSBColor for control using Hue, Saturation and Brightness. The other is used if #RGB is the payload format. Button is used for two state VSPs and List used for longer ones. Special cases exist for Payloads of ON, OFF, OPEN, CLOSED, OFFLINE, ONLINE, FALSE, TRUE, DISARMED, ARMED, INACTIVE, ACTIVE where the 0 and 1 values are fixed (e.g. OFF=0, ON=1). For other cases the values are selected based upon order observed in the Payloads.

6.2 Payload Transformations

In some situations there may be only a part of the payload that is of interest or some textual transformation is desired on the payload before further process and mapping into HS. Regular Expressions can be used to achieve this objective. The result of the RegEx will change the visible Payload in the Association table so this the case where the Payload may not match what is actually received from the MQTT Broker.

The Regualar Expression parameters are available on the Edit tab.

Regular expressions are for textual transformation such as extracting key date from a long string or changing a character into another (e.g. comma to decimal or date M/D/Y to Y/M/D). The pattern box is

used to specify the match pattern. If the pattern is left blank then the Payload will be placed in the Device and no regular expression activity will be performed on the Payload.

The match radio is to specify the regular expression mode. When unchecked it is set to replacement mode. When checked it is in extract mode. Replacement mode uses to the pattern to find matches in the Payload and change those matches to what is contained in the replacement text box. Extract mode looks for text that matches the pattern and uses it as what is placed in the HS Device. The second text box in this case is used to identify which of multiple matches are to be used. If blank then the first will be used. If no matches are found then null is placed in the HS Device. If the number of matches specified is not found then the last one will be used. A large number in the second text box could be entered to extract the last instance of the match in the Payload.

The pattern text box is used to for substitution for the match criteria when in replacement mode. The second text box is used to specify the match instance number when in extract mode. If the pattern is valid the result of the regular expression applied to the Payload will be placed in the Device. If the replace box is left blank then the match text will be replaced with null text.

Syntax reference for regular expression supported by mcsMQTT can be found at https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference. rNote that this reference is for the PERL/.NET syntax. Quite often online RegEx expression analyzers use Java or other conventions. While the concepts are similar there is some variance in the symbols and their use.

A few simple examples below.

1. Change period to comma in a number “1.23”

   Pattern: \. (look for a “.”, it needs to be escaped with \)
   Extract Checkbox: Unchecked
   Replace: ,

   Result: 1,23

   Note: \. is escaped “.” Because “.” is reserved. Every instance of “.” Will be changed to “,” so could fail if mining large text string

2. Remove suffix “ seconds” suffix from a Payload of “1234 seconds”

   Pattern: seconds
   Extract Checkbox: Unchecked
   Replace:
   Result: 1234

   Note: replace suffix of “ seconds” with null. Note alternate method using general rather than specific suffix in next example

3. Extract numeric part of Payload of “1234 seconds”

   Pattern: \D+
   Extract Checkbox: Unchecked
   Replace:
   Result: 1234

   Note: pattern is consecutive set of non-digits

4. Transform date from YYYYMMDD to MMDDYYYY format in "2018/10/01T00:24:18"

   Pattern: (\d+)-(\d+)-(\d+)(T)(.+)
   Extract Checkbox: Unchecked
   Replace: $2/$3/$1 $5
   Result: 01/10/2018 00:24:18

   Note: use of “()” provides sequenced substitution groups that are stored in $1, $2 etc. The first group will capture the year. The second will be the month. The third will be the day. Between these groups a “-“ is needed. The “T” is then needed and defined as group $4. $5 will contain the remainder to string. The replace transforms the order, changes “T” to “ “ and “-“ to “/”.

5. Extract last segment of IP “192.168.1.100”

   Pattern: \d+
   Extract Checkbox: Checked
   Replace: 4
   Result: 100

   Note: four groups of digits are detected because the period delimits the digit group. The Replace text indicates that the 4th match is to be extracted.

6. Change IP address (192.168.5.127) to a hyperlink (courtesy of bartz, HS board)

Pattern: (.+)
Extract Checkbox: Unchecked
Replace: $1
Result: 192.168.5.127

7. Extract a value for a JSON key such as “Action” in {“status”:”online”,”Action”:”stop”,”IP”:”192.168.0.1”}. Note JSON decoding is normally done by mcsMQTT, but for cases where nexted JSON is used then this could be useful. Assume the JSON key is Action:

Pattern: ((.*)Action:(.*)(,.*))
Extract Checkbox: Unchecked
Replace: $2
Result: “stop”

Note: The first group is everything before the text Action: The second group is everything after it until group 3. The third group is then the remainder after the comma. The second group is the Action value of interest so the $2 is the replacement.

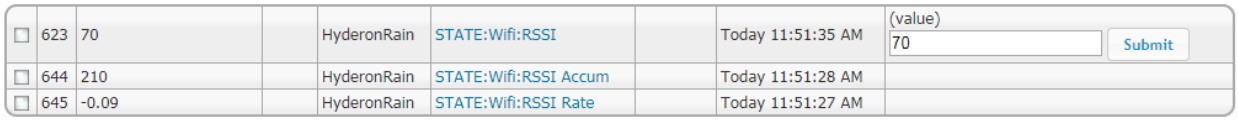
6.3 Payload Numeric Transformations

Four forms of numeric transformation are available and are invoked for “A”ssociated Topics. If no transformation is specified on the Edit tab, then the raw Payload will be placed in the HS Device.

A low pass filter can be used to reduce the noise of a Payload input. The filter sensitivity is set near 0.0 to make the make the value very stable thus reflecting only longer-term trends. Sensitivity near 1.0 will have very little smoothing effect.

An arithmetic expression can be used for simple conversion of units to complex functions. The expression can be specified using a combination of math operators (e.g. +), functions (e.g. round, sin, int), and replacement variables (e.g. $$PAYLOAD:). See Table 2 for lists of replacement variables and Table 3 for available expression functions. As an example, the following will converts Payload in inches to centimeters with one digit precision: “(round($$PAYLOAD: * 2.54,1)”

An additional HS Device can be created to capture either the derivate (rate) or integral (accumulation) of the Payload input. When the Device is created via checkbox selection it is associated with the base device. This will result in HS always keeping these grouped together such as shown in Figure 41.



☐	623	70		HyderonRain	STATE:Wifi:RSSI		Today 11:51:35 AM	(value) 70 Submit
☐	644	210		HyderonRain	STATE:Wifi:RSSI Accum		Today 11:51:28 AM	
☐	645	-0.09		HyderonRain	STATE:Wifi:RSSI Rate		Today 11:51:27 AM	

Figure 41 Base, Rate, and Accumulation Device Associations

A rate device contains a sensitivity factor. A sensitivity of 1.0 will result in the rate determined only by the last two payload inputs received. As sensitivity approaching 0.0 will result in a very slow changing rate. The time units can also be selected between per second, per minute or per hour.

An accum device contains a provision to reset the accumulation at midnight. When enabled the accumulation device value will be set to 0 at midnight. The accumulation device will behave in one of two ways when reset at midnight. One is to treat the midnight value as the starting point and the accum device will reflect the change since the midnight value. The other is to set the midnight value to zero and the accum device will accumulate the values that are received throughout the day.

Status formatting for numeric values is left up to the user using the Status-Graphics edit capability of the HS3 Device Management or HS4 Devices browser page. Figure 42 is an example of a user edit to change the Status format displayed to “XX Minutes”.

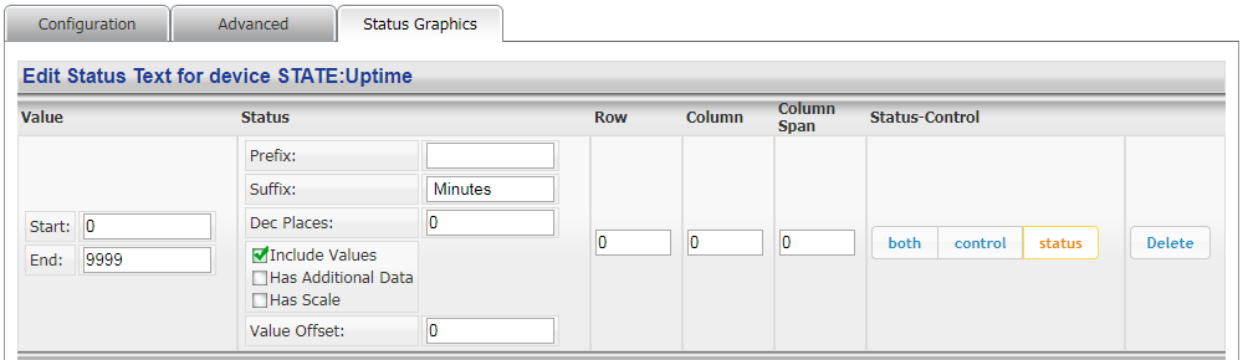


Figure 42 Edit of Status for Numeric Devices

6.4 Payload OtherTransformations

Expressions can also be used where the result is text that will be stored in DeviceString. Date and currency formatting are examples. There are also extensions to math functions that can be used for numeric transformations that will be stored in DeviceValue. If the result of a transformation is to be stored in DeviceValue then the type will typically be “Number”, but could also be “Button” or “List”. If it is stored in DeviceString then it will be “Text”.

An example of a transformation from a Unix/Epoch date/time format into a local time format would be the expression “Local_DateTime($$PAYLOAD: ,”d MMMM YYYY h:m:s tt”)” where the payload may be something like “1554758969”. This would result in “8 April 2019 9:29:29 PM” being stored in the HS DeviceString.

The prototypes of functions supported are listed in Table 3.

Of special note is “IfChange” as it will inhibit the publish operation if a null string value is returned. It takes three parameters. The new value, the previous value, and the hysteris threshold. If the value’s magnitude difference from the previous value is more than the threshold then the function will return the new value, otherwise it will return a null string. The threshold can be specified as a number or as a numeric string suffixed with “%”. Its typical use will be in the publish payload template for non plugin devices where the desire is to inhibit publishing MQTT messages when only a small change occurs in the HS device value. The following three examples for the publish payload template will inhibit where device value changes by less than 5% and by less than 20. In the first two cases the delta is from the prior HS DeviceValue and the third is the case when it is the delta of the Payload from the last time the Topic was published.

<<IfChange($$VALUE:,$$PREVIOUS:,”5%”)>>

<<IfChange($$VALUE:$$PREVIOUS:,20)>>

<<IfChange($$VALUE:$$PUBLISHED:,20)>>

Table 3 Expression Functions

Math Functions
Public Function Sin(ByVal v As Double) As Double
Public Function Cos(ByVal v As Double) As Double
Public Function Tan(ByVal v As Double) As Double
Public Function ArcSin(ByVal v As Double) As Double
Public Function ArcCos(ByVal v As Double) As Double
Public Function ArcTan(ByVal v As Double) As Double
Public Function Sqrt(ByVal v As Double) As Double
Public Function Power(ByVal v As Double, ByVal e As Double) As Double
Public Function Limit(ByVal v As Double, ByVal eMin As Double, eMax as Double) As Double
Public Function Mod(ByVal x As Double, ByVal y As Double) As Double
Public Function Min(ByVal v1 As Double, ByVal v2 As Double, _ Optional ByVal v3 As Double = Double.MaxValue, _ Optional ByVal v4 As Double = Double.MaxValue, _ Optional ByVal v5 As Double = Double.MaxValue) As Double
Public Function Max(ByVal v1 As Double, ByVal v2 As Double, _ Optional ByVal v3 As Double = Double.MinValue, _ Optional ByVal v4 As Double = Double.MinValue, _ Optional ByVal v5 As Double = Double.MinValue) As Double
Public Function Abs(ByVal val As Double) As Double
Public Function Floor(ByVal value As Object) As Integer
Public Function Ceiling(ByVal value As Object) As Integer
Public Function Int(ByVal value As Object) As Integer
Public Function Trunc(ByVal value As Double, ByVal prec As Integer = 0) As Integer
Public Function Dec(ByVal value As Object) As Double
Public Function FromHex(ByVal value as Object) as Double (e.g. “0A” → 10.0)
Public Function Round(ByVal value As Object, ByVal prec As Integer = 0) As Double
Public Function Exp(ByVal base As Double, ByVal pexp As Double) As Double
Public Function Rnd() As Double

Conditional Functions
Public Function [If](ByVal cond As Boolean, ByVal TrueValue As Object, ByVal FalseValue As Object) As Object
Public Function IfEQ(ByVal Parm1 As String, ByVal Parm2 As String, ByVal TrueValue As String, ByVal FalseValue As String) As String
Public Function IfGT(ByVal Parm1 As String, ByVal Parm2 As String, ByVal TrueValue As String, ByVal FalseValue As String) As String
Public Function IfLT(ByVal Parm1 As String, ByVal Parm2 As String, ByVal TrueValue As String, ByVal FalseValue As String) As String
Public Function IfChange(ByVal value as Object, _ ByVal previousValue as String, _ ByVal threshold as String) as String If value and previousValue are numeric then return null if value from previousValue is under threshold. Threshold can be number or percent if % is suffix. It not numeric then compare of two string with equality returning null string.
Public Function IfDelta(ByVal value as Object, _ ByVal previousValue as String, _ ByVal threshold as String) as String If value and previousValue are numeric then return previousValue if value from previousValue is under threshold otherwise return value. Threshold can be number or percent if % suffix is used on threshold. Example IFDELTA($$PAYLOAD:,$$DVR:(4038):,"10%") Example IFDELTA($$PAYLOAD:,$$DVR:(4038):,5
Public Function Case(ByVal variable As String, ByVal value As String, ByVal out As String) As String Variable is item being evaluated Value is string of semicolon-separated values Out is string of semicolon-separated result for each value. If out contains more items than value then last out will be the else result. Example CASE($$PAYLOAD:,”10”;20;30”,”0;100;255;-1” will return 0, 100,255 or -1 depending upon the payload being 10, 20, 30 or something else

String Functions
Public Function Trim(ByVal str As String) As String
Public Function LeftTrim(ByVal str As String) As String
Public Function RightTrim(ByVal str As String) As String
Public Function PadLeft(ByVal str As String, ByVal wantedlen As Integer, ByVal addedchar As String = " ") As String
Function Replace(ByVal base As String, ByVal search As String, ByVal repl As String) As String
Public Function Substr(ByVal s As String, ByVal from As Integer, ByVal len As Integer = Integer.MaxValue) As String
Public Function Len(ByVal str As String) As Integer
Public Function Lower(ByVal value As String) As String
Public Function Upper(ByVal value As String) As String
Public Function WCase(ByVal value As String) As String
Public Function Format(ByVal value As Object, ByVal style As String) As String
Public Function Char(ByVal c As Integer) As String
Public Function ChCR() As String
Public Function ChLF() As String
Public Function ChCRLF() As String
Public Function Split(ByVal s As String, Optional ByVal delimiter As String = ",") As String()
Public Function SRound(ByVal value As Object, ByVal prec As Integer = 0) As String
Public Function Hex2(ByVal value As Object) As String (15 -> “0F”)
Public Function Hex4(ByVal value As Object) As String (15 -> “000F”)
Public Function ToHex(ByVal value As Object) As String (15 -> “F”)
Public Function FromHexString(ByVal value As String) As String (e.g. “41 42” → “AB”)
Public Function RGB(ByVal value As Object) As String (e.g. 255 → “0000FF”)
Date Functions
Public Function Now() As DateTime date and time in local format
Public Function Today() As String date in local format
Public Function Time() as String time in local format
Public Function [Date](ByVal year As Integer, ByVal month As Integer, ByVal day As Integer) As DateTime
Public Function Year(ByVal d As DateTime) As Integer
Public Function Month(ByVal d As DateTime) As Integer
Public Function Day(ByVal d As DateTime) As Integer
Public Function WeekDay(ByVal d As DateTime) As Integer
Public Function NameOfDay(ByVal d As DateTime) As String
Public Function Format_DateTime(ByVal d As Object, ByVal fmt As String) As String (d is number then local Unix/Epoch time. d is date then local datetime. Otherwise now. See https://learn.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings for fmt)
Public Function Local_DateTime(ByVal d As Object, ByVal fmt As String) As String (d is number then GMT Unix/Epoch time. d is date then GMT datetime. Otherwise now. See https://learn.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings for fmt)
Public Function Unix_time(ByVal n As Object) As String will be obsoleted with use of Format_DateTime and Local_DateTime can perform the same function
Public Function Unix_date(ByVal n As Object) As String will be obsoleted with use of Format_DateTime and Local_DateTime can perform the same function
Public Function Long_date(ByVal d As DateTime) As String will be obsoleted with use of Format_DateTime and Local_DateTime can perform the same function
Public Function Long_time(ByVal d As DateTime) As String will be obsoleted with use of Format_DateTime and Local_DateTime can perform the same function

Public Function Format_date(ByVal d As DateTime, ByVal fmt As String = Nothing) As String will be obsoleted with use of Format_DateTime and Local_DateTime can perform the same function
Public Function DateAdd(ByVal Interval As String, ByVal Delta As Integer, ByVal d As DateTime) As DateTime where Interval is "dayofyear","weekofyear","year","quarter","month","week","day","hour","minute","second"
Misc Functions
Public Function Money(ByVal d As Object) As String
Public Function [If](ByVal cond As Boolean, ByVal TrueValue As Object, ByVal FalseValue As Object) As Object
Public Function Entry(ByVal n As Integer, ByVal s As String, ByVal delim As String = ",") As String
Public Function Index(ByVal s As String, ByVal search As String, ByVal delim As String = ",") As Integer
Public Function Inlist(ByVal search As String, ByVal list As String, ByVal delim As String = ",") As Boolean
Function DbNull() As System.DBNull
Public Function FileExists(ByVal f As String) As Boolean
Public Function FileInfo(ByVal f As String) As IO.FileInfo
Public Function BasicAuth(ByVal username As String, ByVal password As String) As String
Public Function AES128(ByVal svalue As String, ByVal sKey As String) As String
Public Function MD5Hash(ByVal input As String) As String (e.g. MD5HASH("username:pw"))

Operators

operator_plus +
operator_minus -
operator_mul *
operator_div /
operator_percent %
open_parenthesis (
comma ,
dot .
close_parenthesis)
operator_ne <>
operator_gt <=
operator_ge >=
operator_eq =
operator_le <=
operator_lt <
operator_and and
operator_or or
operator_not not
operator_concat &

any word starting with a letter or value_identifier

value_true true
value_false false

any number starting 0-9 or .

value_number

any string starting ' or "

value_string
open_bracket [
close_bracket]

6.5 Payload Storage

The Payload of a MQTT message will usually be stored in the “A”ssociated HS device in either the DeviceValue or DeviceString property of the device. The selected property will depend upon the setup of the Control/Status UI and Store Payload lines on the Edit tab for the Device/Topic as highlighted in Figure 44.

Anytime Control/Status UI is selected to be “Text” or “Sign”, or Store Payload is selected to be “Device String” then the Paylaod will be stored in the Device String of the HS Device.

If Control/Status UI is selected to be “jpg File” the Device String will contain HTML for thumbnail image of the Payload stored in a file at HS subfolder \html\mcsMQTT\File\{floor}\{room}\{name}.jpg and the Device Value will increment for each time the Topic is received.

When Control/Status UI is selected to be List or Button then a VSP relationship will be setup that uses the text in the Payload to map into a number and then mcsMQTT will update DeviceValue with the number.

When Control/Status UI is selected to be Number or NumberChange then the Payload is expected to be numeric, or has become numeric after using regular expression or expression capability of the Edit tab. The number will then be stored in DeviceValue.

When Control/Status UI is selected to be RGB, RGBW or HSB then the Payload is interpreted as a color space definition and mcsMQTT will convert it into a 24-bit number and stored in DeviceValue

When Control/Status UI is selected to be Slider then the Payload is expected to represent a percentage in the range of 0 to 100. It may be necessary to use numeric expressions on the Edit tab to transform the payload into a number of this range. The result will be stored in DeviceValue.

When Control/Status UI is selected to be CSV then the Payload is assumed to be a comma-separated set of numbers. A HS device will exist for each number and the DeviceValue will be updated.

When Control/Status UI is selected to be Ramp then then a slider control and a rate number box is presented for this topic. The slider will range from 0 to 100 (percent). The rate will be the number of seconds for the slider value to go from 0 to 100. The example shown in Figure 43 will send commands at a rate of (100 seconds / 100% = 1 second rate) with each command being an increase or decrease of 1% value. The commands will stop when the target level is achieved.

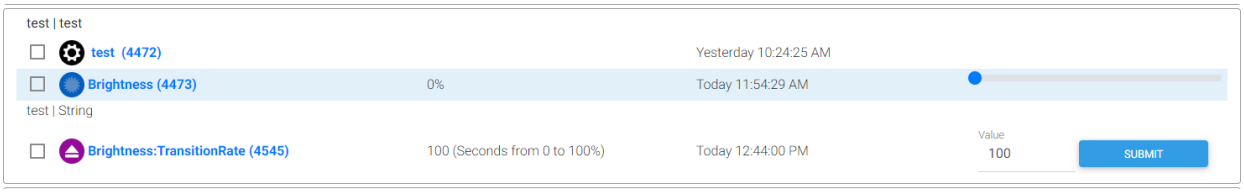


Figure 43 Transition Rate Ramp Contorl/Status UI

Start with Either Existing Device Ref or Subscribe Topic

Ref: 8848 Sub: owntracks/mcsSolutions/M1:lat [Delete Sub and Ref]

Edit Setup or Edit of Subscription (Inbound) to a MQTT Topic

MQTT Subscribe Topic	owntracks/mcsSolutions/M1:lat ☐ Check to treat JSON key value as topic
Payload RegEx Match Pattern	
Payload RegEx Replace Pattern	
Payload RegEx Operation	◉ Replace Match Pattern with Replace Pattern ○ Extract Match Pattern
Low Pass Filter	Filter sensitivity of 1 (range is 0.00 to 1.00 (most sensitive))
Expression	
Add Rate Device	☐ Create a HS Rate Device with rate sensitivity of 0.75 (Range 0.00 to 1.00) ○ Per Second ○ Per Minute ◉ Per Hour
Add Accum Device	☐ Create a HS Accum Device ○ No Reset ○ Accumulation Since Midnight ◉ Delta Since Midnight
Store Payload	◉ In HS Device Value ○ In HS Device String ○ Report Status on null Payload

Settings for Plugin Device

HS Device Publish Topic	
HS Device Control/Status UI	○ Unspecified ◉ Button ○ Toggle ○ Number ○ NumberChange ○ Slider ○ Ramp ○ CSV ○ Text ○ List ○ RGB ○ RGBW ○ HSB ○ ColorXY ○ Sign ○ jpg File
HS Device Location	Loc2 (Floor) owntracks Loc (Room) mcsSolutions Name M1:lat
HS Device VSP List	Max number of VSP 12 Payload 47.5268014=0;47th Parallel VSP Add/Edit [Clear existing VSP]
HS Device MISC Properties	☐ NO_STATUS_DISPLAY ☐ NO_GRAPHICS_DISPLAY ☐ AUTO_VOICE_COMMAND ☐ SET_DOES_NOT_CHANGE_LAST_CHANGE ☑ SHOW_VALUES ☐ STATUS_ONLY
Grouping Parent Ref	8847 [Create New Parent Device]
Publish Payload Template	
URI Encode Payload	◉ Send unencoded ○ Encode with URI encoding such as %20 for space ○ Replace special characters with underscore
MQTT Publish to Sign	◉ Normal Publish ○ Publish to Messaging Sign
Publish QOS	◉ At Most ○ At Least ○ Exactly
Publish Retain Flag	◉ Do not retain ○ Retain at broker
HS Energy Database	◉ Do not save ○ Save as Watts

Settings for Non-Plugin Device

Control non-Plugin HS Device	HS Device Reference Number

Note additional customization of buttton text, display graphics, button/number relationships etc. is done via HS Device Management Page by clicking on the Device Name link and using the Status Graphic and Configuration tabs

Figure 44 Override to Store String rather than Value

6.6 Controlling HS Device via MQTT Topic

Lists of MQTT Topics that have been received are available for mapping into HS Devices on the Associations Tab. The "A"ssociated checkbox or existing Device Reference is used to establish this relationship. For mcsMQTT devices, Numeric changes in Payload are reflected in the HS DeviceValue. Non-numeric Payload is reflected in DeviceString. The special case of Payload text of case-insensitive ON, OFF, TRUE, FALSE, OPEN and CLOSED create a Device-Status pair with DeviceValues of 0 and 1 or OFF/CLOSED and ON/OPEN, respectively.

The setup described previously in Section for associating Topics and Devices did not include the case where an existing HS Device is to be controlled by a MQTT Topic. This setup is done in one of two ways. The first is done on the Association Tab by entering a non-Plug-in Device Reference for a subscribed Topic rather than using the "A"ssociate checkbox. That will associate the subscription as if it is a command to control the existing HS Device. It can also be done on the Edit tab in either the Publish or Subscribe tables. When the Association table is built with this hybrid type setup the row color will be blue rather than pink or green. Figure 45 provides an example where non-Plug-in Device 72 is being commanded by "RadarMotion2/SENSOR:Time"Topic and reporting status on "Dell-PC/mcsMQTT/Unknown/Unknown/New4" Topic. The DeviceValue will be in the Payload as the default if no substitutions are provided in the Payload template.

Substitutions in the publish template are indicated by use of expressions encased in "<<" and ">>". For example, if the desire is to publish a 8 bit brightness value and the HS brightness is maintained as a percentage. Furthermore, a JSON payload rather than just a raw number are to be published. A template of '{"Brightness":<<$$VALUE:*255/100>>}' would be used.

The second method to associate MQTT topic with existing HS device is with use of the Subscription Wildcard template available on the Client Tab. The HS Device identification will be included in the received Topic and specified as a set of substitution parameters described in Table 2 and in the template. This approach has the minimum setup for the user, but the incoming topics must follow a standard pattern that complies with the wildcard. Note that when the wildcard is used the Outbound (Publish) Management is usually also setup so changes in the HS device are reported via MQTT. See Figure 1 and Section 4.1.20 for an example.

Further auto-association can be done such that all existing non-plug-in HS devices are associated with a publish topic and subscribe topic. This option is selected with the radio selection at the same location as the subscribe wildcard on the Client Tab. Note that when the immediate selection is made the associations are made at time of selection, upon use of the Enumerate button on General Tab or when the plug-in starts.

Association Table for Auto Association of MQTT Topic and HS Device

Λ	r	e	a	REF	topic	payload	h	d	lastdate
0	☐	☐	☑	72	Dev: Unknown|Unknown|New4 Command HS Device on subscribed Topic: RadarMotion2/SENSOR:Time Publish message on Device change using Topic: Dell-PC/mcsMQTT/Unknown/Unknown/New4 Encode Payload per template:		☐	☐	0001-01-01 00:00:00

Figure 45 mcsMQTT Devices Mapping to Specific MQTT Topics

When MQTT messages of this type are received the CAPI interface is used to control the HS Device. This is in contrast with mcsMQTT devices that are changed by SetDeviceValueByRef and SetDeviceString methods. The Plug-in will search through the CAPI interface and try to find a match with the Payload received and then forward that CAPI control item to HS to have the HS Device controlled. The owning Plug-in is responsible for updating Device String and Value as appropriate. Normally this update will result in a HS Event callback indicating that a device status has changed. If a publish topic is setup for this subscribe topic then the status change will be published unless the publish topic and subscribe topic are the same.

6.7 MQTT Receive Event Triggers

Three forms of MQTT event triggers are available. One detects a Topic update. This is the "MQTT Topic Received" event trigger. This is shown as the two lower events in Figure 46. The trigger occurs on receipt of the Topic. If can optionally be specified further by a specific text somewhere in the Payload. See Figure 34 where the MQTT event trigger is being setup. This is similar to event triggers that are from changes in a mapped HS Device where a change in Payload is needed to trigger an event. In this case the Payload text will be anywhere in the Payload while in the Device change trigger it is setup per HS trigger specification rules.

The Payload text box has three options: simple text, regular expression, change.

Simple text is a match of the text anywhere in the payload. The conditional operators && and || can be used to specify multiple conditions such as "id&&45".

Regular expression is indicatated in one of three formats. REGEX(expression), <<REGEX(expression)>>, or <<expression>>. Match is found when the regular expression indicates one or more matches

Keyword 'change' is entered for the Payload field of a MQTT Receive trigger then the trigger will fire whenever the received Payload for the specified Topic changes. This could be used to trigger on changing textual Payloads which are not possible with the basic HS Event trigger based upon device changes.

The receive trigger can be used as an event condition (AND IF / OR IF) .

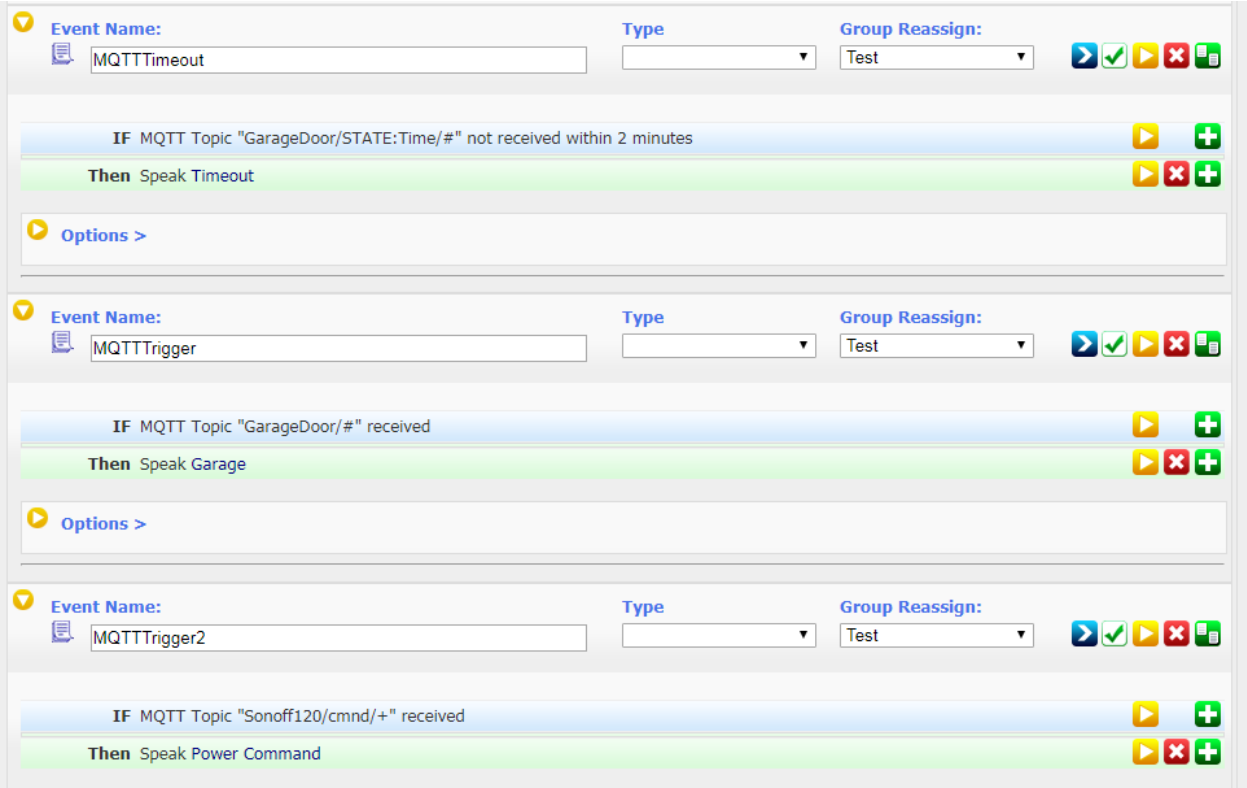


Figure 46 MQTT Receive Event Triggers

The "MQTT Topic Timeout" trigger is used to determine that an expected periodic update has not been received. The first event of Figure 46 shows this setup. The timeout duration is in minutes and only one trigger will be produced after the timeout period. If the Topic is subsequently received then the timer countdown is reset to enable triggering again.

6.8 Topic Wildcards

The "/", "+" and "#" symbols have special meaning in MQTT Topics. The hierarchical structure of a MQTT Topic is indicated with "/". Generally, the order in the hierarchy is most general to most specific much like the HS Locaation2/Locaation/Name hierarchy.

When specifying a receive trigger condition the "+" and "#" can be used as wildcards. The "+" indicates that anything at the specified level of the hierarchy is acceptable. The "#" is used to indicate a match for current and all subsequent levels of the hierarchy. Examples of these are used in Figure 46.

6.9 HomeAsistant Discovery

The home automation application Home Assistant has specified a protocol https://www.home-assistant.io/docs/mqtt/discovery/ by which devices advertise their characterists to facility inclusing in the Home Assistant application. Tasmota, among other devices, has support for this definition. In the Tasmota case the discovery is advertised in the topic "homseer/x/y/config" where x is the type being advertised and y is the ID of the device. mcsMQTT will recognize this topic and automatically create HS devices for the primary status and control aspects of the device. It will also make available other end points in the Association tab for manual association with HS devices.

6.1 Tasmota Discovery

Tasmota firmware has defined a discovery protocol specific to Tasmota devices to assist the client in creating devices for the IO setup in Tasmota. There are two Tasmota/discovery messages when the Tasmota SetOption19 0 is used (in Tasmota versions 12 and later). Prior versions of Tasmota also supported the HomeAssistant Discovery.

The first is tasmota/discovery/+/config where basic IO deivces are defined such as switches, relays and lights. The second is Tasmota/discovery/+/sensors where sensor reporting is provided as part of the topic. mcsMQTT will create HS Devices and Features for each sensor automatically if Automatic Device Creation has been enabled on the MQTT Page, Client Tab. It will also create all simple relay and switch configuraitons. When a Shutter has been announced it will create the normal shutter configuration. When a Light has been announced it will create a dimmer control and a color picker control if Tasmota also reports a Color capability.

Figure 47 illustrated the Last Will and Testiment disclosure in HS Feature 1738. The simple relay in 1740, The color light in 1741 through 1743, the switch input in 1742, the shutter in 1745 through 1748.

Once a Feature has been created it can be removed from HS by using the MQTT Page, Association Tab row by unchecking the “A”ssociate checkbox. Customizations of automatically created devices can be done form the HS Devices Page.

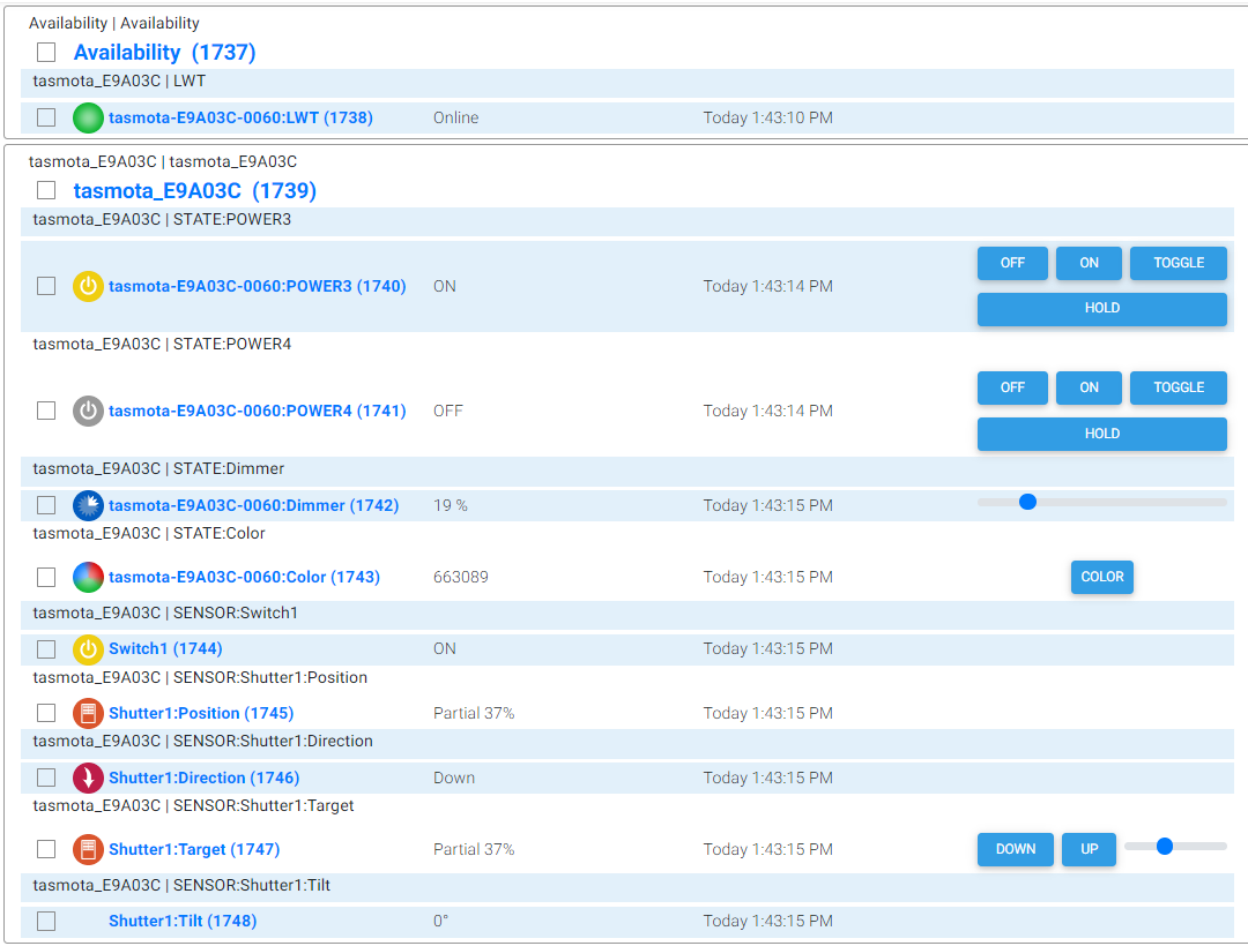


Figure 47 Tasmota Discovery Device Creation Examples

6.2 Homie Discovery

Provisions exist for discovery per Homie definition https://homieiot.github.io/. While this definition is more generic and has potential for more wide-spread adoption there are no devices that have been integrated with mcsMQTT using Homie.

6.3 Scripting Callback

If a user desires to handle the raw MQTT received payload in a script then a callback can be used to receive MQTT topics with their payload. This is setup with a call to the plug-in to register the script. The RegisterTopicReceivedScript callback expects an array of three parameters. The first is the filename that will be located in the HS scripts folder. The second is the name of the procedure in this file. The third is the Topic that will result in a callback. Multiple Topics can be requested with multiple calls to this function. MQTT Topic wildcards can also be used.

hs.PluginFunction("mcsMQTT", "",
"RegisterTopicReceivedScript",{"MyScriptFileName.vb", "MyFunctionName",
"MyMQTTTopic"})

As an example, the following two scripts are used in the file TestCallback.vb. The first (Main) was invoked by event with script action. The second (TheCallback) was invoked by mcsMQTT when it received the MQTT Topic “test/topic”.

```
sub Main(parm as object)
     hs.WriteLog("TestCallback", "Registering Callback for Script")
     Dim callbackParameters = New String {"TestCallback", "TheCallback", "test/topic"}

     hs.PluginFunction(
        "mcsMQTT",
        "",
        "RegisterTopicReceivedScript",
        callbackParameters
     )
End Sub

sub TheCallback(parm as object)
     hs.Writelog("TestCallback", "Received Topic " & parm(0) & " with Payload " & parm(1))
End Sub
```

6.4 Scripting Receive

If a user desires to use capabilities built into mcsMQTT for other devices or data then they can simulate the publication of a MQTT messages with the scripting function “ReceiveMqttMessage”. The parameters consist of an array of two strings where the first is the Topic and the Second is the Payload. A sample usage is below. In a real use the Topic and Payload parameters would be based on user data rather that the static text in the example.

```
sub Main(parm as object)
     hs.WriteLog("TestReceive", "Simulating MQTT Message")
     Dim sTopic as string = “Test/Topic”
     Dim sPayload as string = “TestPayload”
     hs.PluginFunction(
        "mcsMQTT",
        "",
        "ReceiveMqttMessage",
        {sTopic,sPayload}
     )
End Sub
```

7 Display Filtering/Sorting and Scripting Automation

7.1 Display Filtering and Sorting

The top of Association Tab contains a set of checkboxes, filter pull-downs and column buttons that can be used to affect the presentation of rows on the Association Table. Each press of the button will act as a request to sort the table by this column. Subsequent press toggles ascending vs. descending.

The general approach to showing the Association table is to setup the filters as desired and then click the "Show Selected Associations" button. This is different than most other interactions with mcsMQTT where immediate feedback is provided on each mouse click. It is done this way because of the time it takes to generate the Association table and no desire to wait while intermediate results are shown as each filter is selected.

In general, the filtering constrains the number of rows that show in the Association table. The exception is the "Displayed Checkbox Columns" filter affects the columns being shown on the table.

Overriding filters are done with checkbox. The "Show All Associated Only" has precedence over the Include checkboxes. The subsequent filter pull-down tables are updated to show the subset that match the checkbox selections when these checkboxes are clicked.

The top filter pull-down is oriented to HS Devices. The lower pull-down and the text textbox filter is oriented to MQTT Topics and JSON keys within these Topics. A combination of all can be used. The more filters selected the fewer number of rows will be rendered.

Provisions exist for six levels in each the Topic hierarchy and JSON Payload items. The pull-down for each of these twelve will show the different names that have been seen at each level of the hierarchy. If a particular name is selected then only those rows are shown on the display. Multiple levels of filters can be used.

The "Filter By Text" queries the mcsMQTT database (mcsMQTT.db) looking at all the text-oriented fields as well as the HS Device Ref. This includes things like MQTT Payloads, Topics, VSP entries, Expressions, Templates and others. The textbox can contain simple text or it can contain a regular expression that specifies a textual sequence. Simple text is entered as the sequence of characters that must exist in one of the database fields. Regular expresions are applied to the same fields and are specififed in either of three ways. << regular expression >>, << REGEX(regular expression)>> or REGEX(regular expression). This is the same syntax used for HS Event triggers that are looking for text in the payload of a MQTT message.

Figure 48 Topic Filter Setup

There may be times where you want the Plug-in to ignore new Topics. It may be a situation where a new device is being debugged and immature messages are being published. The “Disable New Topic Recognition” checkbox on General Tab (Figure 40) Inbound (Subscription) table is used for this purpose.

Going even further the “Disconnect from MQTT Broker” checkbox on the Client Tab is used to total disconnect mcsMQTT from the MQTT environment. In this case no MQTT messages will be sent or received with mcsMQTT.

Presentation sort order is selected by the association table header buttons as shown in Figure 45. Ascending vs. descending status is shown in the first column. The column with upper case text is the sort field.

7.2 Scripting Automation

Scripting interface consists of sending MQTT messages, setting up callbacks for received MQTT messages, editing of the MQTT topic properties, process management support, database support, and other access to mcsMQTT features. The first two of these are described in Section 5.4 and Section 6.3. The third is described here with the four functions EditPropertyByRef, EditPropertyByTopic, ClearVSP and AddVSP.

7.2.1 Edit of mcsMQTT Properties

```
Public Function EditPropertyByRef(ByVal sRef As String, ByVal sProperty As String, ByVal
sValue As String) As String

Public Function EditPropertyByTopic(ByVal sSource As String, ByVal sProperty As String,
ByVal sValue As String) As String

Public Function ClearVSP(ByVal sRef As String) As String

Public Function AddVSP(ByVal sRef As String, ByVal sPayload As String, ByVal sValue As
String, ByVal sStatus As String) As String
```

The returned value from these functions is normally null. When an error occurs the error message is returned. All parameters are handled as strings.

A property is updated by providing a handle to either the HS feature reference number or the MQTT subscribed topic. Two methods are provided for convenience and produce the same result. The propery being modified is from the following set. While the input parameter is a string its contents will need to comply with the data type (Boolean, Integer, or String) used by mcsMQTT.

```
Select Case sProperty
         Case "Source"
             oMQTT.Source = sValue
         Case "Topic"
             oMQTT.Topic = sValue
         Case "Template"
             oMQTT.Template = sValue
         Case "Pattern"
             oMQTT.Regex(0) = sValue
         Case "Replace"
             oMQTT.Regex(1) = sValue
         Case "Match"
             oMQTT.Regex(2) = sValue
         Case "Reject"
             oMQTT.Reject = CType(sValue, Boolean)
         Case "Express"
             oMQTT.Express = CType(sValue, Boolean)
         Case "Elevate"
             oMQTT.Elevate = CType(sValue, Boolean)
         Case "ElevateKeys"
             oMQTT.ElevateKeys = sValue
         Case "URIEncode"
             oMQTT.URIEncode = CType(sValue, Integer)
         Case "Accept"
             oMQTT.Accept = CType(sValue, Boolean)
         Case "RetainFlag"
             oMQTT.Retain = CType(sValue, Boolean)
         Case "PluginDevice"
```

```
    oMQTT.PluginDevice = CType(sValue, Integer)
         Enum RecordType
             NonPlugin = 0
             Child = 1
             Parent = 2
         End Enum
Case "Subscribe"
    oMQTT.Subscribe = CType(sValue, Boolean)
Case "Chart"
    oMQTT.Chart = CType(sValue, Boolean)
Case "History="
    oMQTT.History = CType(sValue, Boolean)
Case "StorePayload"
    oMQTT.StorePayload = CType(sValue, Integer)
         Enum RecordType
             InDeviceValue = 0
             InDeviceString = 1
         End Enum

Case "Ref"
    oMQTT.Ref = CType(sValue, Integer)
Case "ChangeType"
    oMQTT.ChangeType = CType(sValue, Integer)
         Enum EventChangeType
             IgnoreChange = 0
             ValueChange = 1
             StringChange = 2
             AnyChange = 3
             LogChange = 4
             ValueAndStringChange = 5
             ValueAndLogChange = 6
             StringAndLogChange = 7
         End Enum
Case "Misc"
    oMQTT.Misc = CType(sValue, Integer)
            per HomeSeerAPI.Enums.dvMISC (HS3)
            per HomeSeer.PluginSdk.Devices.EMiscFlag (HS4)
Case "QOS"
    oMQTT.QOS = CType(sValue, Byte)
            Enum QOS
                 AT_MOST_ONCE = 0
                 AT_LEAST_ONCE = 1
                 EXACTLY_ONCE = 2
            End Enum
Case "StatusType"
    oMQTT.StatusType = CType(sValue, Integer)
         Enum StatusTypes
             StatusOnly = 0
             Button = 1
             Number = 2
             Text = 3
             List = 4
             ColorPicker = 5
             ColorXY = 6
             NumberChange = 7
             Sign = 8
             CSV = 9
             HSB = 10
```

```
                Slider = 11
                RGBW = 12
                Ramp = 13
                Toggle = 14
                jpgFile = 15
            End Enum
        Case "Broker"
            oMQTT.Broker = CType(sValue, Integer)
                    Broker index 0, 1, or 2
        Case "Reflist"
            oMQTT.RefList = sValue
        Case "Energy"
            oMQTT.Energy = CType(sValue, Integer)
        Case "VgpMax"
             oMQTT.VgpMax = CType(sValue, Integer)
        Case “RateDevice”
            oTransform.RateDevice = CType(sValue, Integer)
        Case “RateSensitivity”
            oTransform.RateSensitivity = CType(sValue, Integer)
        Case “RateInterval”
            oTransform.RateInterval = CType(sValue, Integer)
            Enum RateIntervals
                PerSecond = 0
                PerMinute = 1
                PerHour = 2
            End Enum
        Case “AccumDevice”
            oTransform.AccumDevice = CType(sValue, Integer)
        Case “AccumReset”
            oTransform.AccumReset = CType(sValue, Integer)
            Enum AccumTypes
                NoReset = 0
                ResetTotal = 1
                ResetDelta = 2
            End Enum
        Case “AccumMidnight”
            oTransform.AccumMidnight = CType(sValue, Double)
        Case “Expression”
            oTransform.Expression = sValue
        Case “FilterSensitivity”
```

This scripting facility provides great power over the configuration of mcsMQTT topics it comes with corresponding danger to really mess things up. Use with caution and have appropriate backups with specific focus on the \Data\mcsMQTT\mcsMQTT.db file where the knowledgebase of the properties exists.

The following test script illustrates the use of these scripting methods.

```
Sub Main(parm as object)
       'update publish Topic property using feature reference number
       Dim sRef as string = "5656"
       Dim sProperty as string = "Topic"
       Dim sResult as string = ""
       Dim sValue as string = "Sonoff/Bedroom/Light/cmnd"
       Dim sArrayRef() as string = {sRef,sProperty,sValue}
       sResult = hs.PluginFunction("mcsMQTT","","EditPropertyByRef",sArrayRef)
```

```
if sResult <> "" then
      hs.Writelog("mcsMQTT","MQTT SetProperty Message Failure " & sResult)
end if

'update MISC property using subscription topic
Dim sTopic as string = "pool/state/temps/bodies/1/pool/setPoint:setPoint"
sProperty  = "Misc"
sValue = "4096"
Dim sArrayTopic() as string = {sTopic,sProperty,sValue}
sResult = hs.PluginFunction("mcsMQTT","","EditPropertyByTopic",sArrayTopic)
if sResult <> "" then
      hs.Writelog("mcsMQTT","MQTT SetProperty Message Failure " & sResult)
end if

'update Expression property using subscription topic
sTopic = "Beacon/DD.0D.30.46.3D.2E"
sProperty  = "Expression"
sValue = "{""Key"":""$$VALUE:+1""}"
Dim sArrayExp() as string = {sTopic,sProperty,sValue}
sResult = hs.PluginFunction("mcsMQTT","","EditPropertyByTopic",sArrayExp)
if sResult <> "" then
      hs.Writelog("mcsMQTT","MQTT SetProperty Message Failure " & sResult)
end if

'define new VSP values and statuses
Dim sArrayClear() as string = {sRef}
sResult = hs.PluginFunction("mcsMQTT","","ClearVSP",sArrayClear)
if sResult <> "" then
      hs.Writelog("mcsMQTT","MQTT ClearVSP Message Failure " & sResult)
end if

Dim sPayload as String = "CLOSE"
Dim sNumber as String = "0"
Dim sStatus as String = "CLOSED"
Dim sArrayClose() as string = {sRef,sPayload,sNumber,sStatus}
sResult = hs.PluginFunction("mcsMQTT","","AddVSP",sArrayClose)

sPayload = "OPEN"
sNumber = "1"
sStatus = "OPENED"
Dim sArrayOpen() as string = {sRef,sPayload,sNumber,sStatus}
sResult = hs.PluginFunction("mcsMQTT","","AddVSP",sArrayOpen)

'Error Examples
sTopic = "Beacon/DD.0D.30.46.3D.2E"
sProperty = "BadProperty"
sValue = "dontcare"
Dim sArrayTypo1() as string = {sTopic,sProperty,sValue}
sResult = hs.PluginFunction("mcsMQTT","","EditPropertyByTopic",sArrayTypo1)
if sResult <> "" then
      hs.Writelog("mcsMQTT","MQTT SetProperty Message Failure " & sResult)
end if

sTopic = "Beacon/BadTopic"
sProperty = "dontcare"
sValue = "dontcare"
Dim sArrayTypo2() as string = {sTopic,sProperty,sValue}
sResult = hs.PluginFunction("mcsMQTT","","EditPropertyByTopic",sArrayTypo2)
```

```
        if sResult <> "" then
                hs.Writelog("mcsMQTT","MQTT SetProperty Message Failure " & sResult)
        end if
end Sub
```

Time	From	Type	Message
10/22/2020 10:42:40 AM	Script	mcsMQTT	MQTT SetProperty Message Failure Beacon/BadTopic not recognized as a subscribed topic
10/22/2020 10:42:40 AM	Script	mcsMQTT	MQTT SetProperty Message Failure BadProperty is not recognized
10/22/2020 10:42:40 AM	HomeSeer	Event	Running script in background (EditProperty): EditProperty.vb
10/22/2020 10:42:40 AM	HomeSeer	Event	Event Trigger "UnNamed EditProperty"

7.2.2 Process Management Scripting Helpers

Assistance for process management is provided with the two methods “Shutdown” and “ProcessId” as parameters to the “PluginFunction” method. The following invocation will result in mcsMQTT going through an orderly shutdown. This will normally be followed by HS automatically restarting mcsMQTT since the process has disappeared from HS.

```
hs.PluginFunction("mcsMQTT", "", "Shutdown", {""})
```

The immediate script command from an event for HS4 prefixes with &n so

```
&nhs.PluginFunction("mcsMQTT", "", "Shutdown", {""})
```

To get the process Id for mcsMQTT the following call is made. This can be useful if a need exists to monitor or shutdown the mcsMQTT process using shell commands. It will return the integer value for process Id or 0 if mcsMQTT process has not yet been identified.

```
processId = hs.PluginFunction("mcsMQTT", "", "ProcessId", {""})
```

The plugin mcsMonitor is included in the Updater package, but not normally installed. If HSPI_mcsMonitor.exe is copied to the HS folder, and enabled, it will monitor the HS log for one of the following two conditions by default. These can be changed and others added from the Config page of mcsMonitor as shown in Figure 49. If either is met then it will use the above two methods for an ordely shutdown of mcsMQTT which will then be followed by it’s restart by HS. This work-around is provided for the case where the bellow conditions have occurred and root cause has not yet been identified.

```
"Plugin mcsMQTT is not responding but it is still running, not restarting yet."

"Dropping event callbacks due to full queue"
```

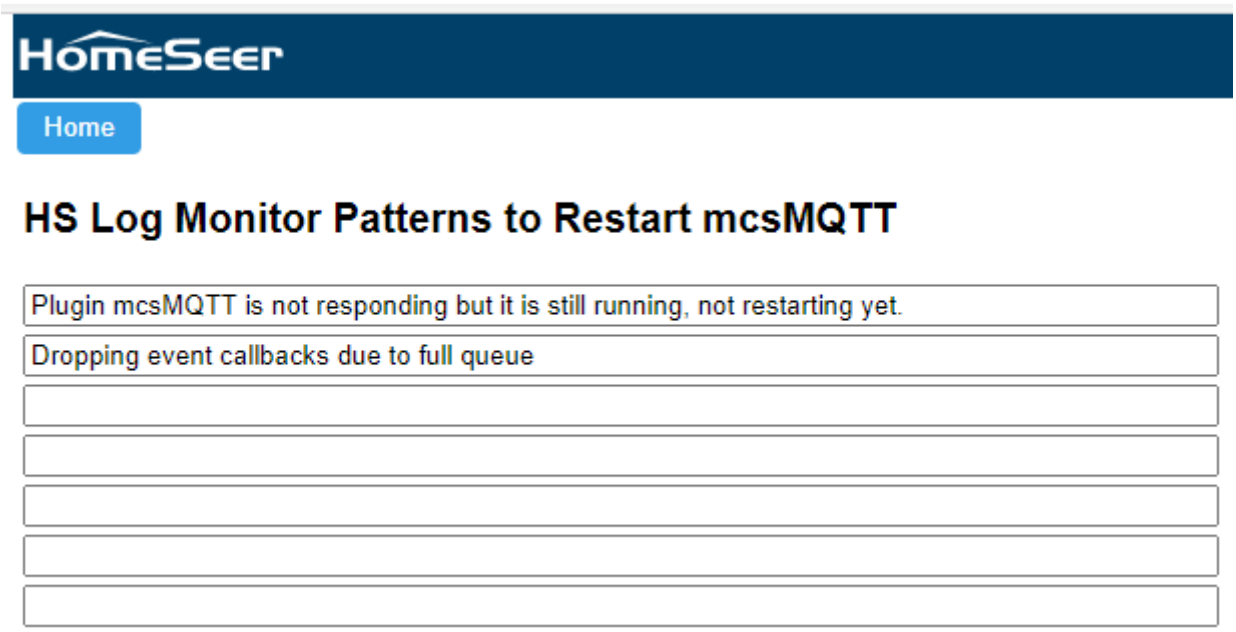


Figure 49 mcsMonitor Config

A preferred method for monitoring mcsMQTT is to setup an event where there is an expectation of some periodic event such as the value of a MQTT device being updated. When this trigger occurs then execute the action of restarting mcsMQTT plugin. This restart capability of mcsMQTT, and all HS plugins, is available when the Association Table Topics starting with HS/ have been associated with HS Devices.

In lieu of running a plugin to perform monitoring it is also possible to run the same code in a script that would normally be run as part of the HS startup script. Some modifications are required to put the code in the scripting context, but central logic does not need to change.

```
Function InitHW() As String
        Dim sResult As String
        Try
            sResult = InitialInit()
            With PeriodicStatusTimer
                .Stop()
                .Interval = 10000
                .AutoReset = True
                .Start()
            End With
            callback.RegisterEventCB(HomeSeerAPI.Enums.HSEvent.LOG, PLUGIN_NAME, "")
            oHSEventCollection = New System.Collections.Queue
            oHSEVentThread = New System.Threading.Thread(AddressOf DoHsEvent)
            oHSEVentThread.Start()
            RegisterConfigPage(CONFIG_PAGE)

            logList = New Generic.Dictionary(Of String, String)
            Dim section As String = hs.GetINISection(GENERAL_GROUP, INI_FILE)
            If section <> "" Then
                Dim items() As String = Split(section, Chr(0))
                gBusy = True
                For k As Integer = 0 To items.Length - 1
                    Dim arrKVP() As String = items(k).Split("=")
                    If Not logList.ContainsKey(arrKVP(0)) Then
                        logList.Add(arrKVP(0), arrKVP(1))
                    End If
                Next
                gBusy = False
            End If

            If logList.Count = 0 Then
                logList.Add("R0", "Plugin mcsMQTT is not responding but it is still
running, not restarting yet.")
                logList.Add("R1", "Dropping event callbacks due to full queue")
            End If

            For Each sLog As String In logList.Values
                Console.WriteLine("Monitoring HS Log for " & sLog)
            Next

            Return ""

        Catch ex As Exception
            hs.WriteLog(PLUGIN_NAME, "InitHW " & ex.Message)
            Return "InitHW " & ex.Message
        End Try

    End Function

    Private Function InitialInit() As String
        Try
            If InStr(My.Application.Info.DirectoryPath, "\") = 0 Then
```

```
                slash = "/"
                gLinux = True
            Else
                slash = "\"
                gLinux = False
            End If

            If hs Is Nothing Then
                MsgBox("Unable to access HS interface")
            Else
                hs.WriteLog(PLUGIN_NAME, "Version " &
My.Application.Info.Version.Major & "." & My.Application.Info.Version.Minor & "." &
My.Application.Info.Version.Revision & " Registered with Homeseer")
            End If
            Return ""

        Catch ex As Exception
            hs.WriteLog(PLUGIN_NAME, "InitialInit " & ex.Message)
            Return ex.Message
        End Try
    End Function

    Public Sub HSEvent(ByVal EventType As Enums.HSEvent, ByVal parms() As Object)
Implements HomeSeerAPI.IPlugInAPI.HSEvent
        Try
            SyncLock oHSEventCollection.SyncRoot
                oHSEventCollection.Enqueue(parms(3)) 'log message
                oEventQueueWaiting.Set()
            End SyncLock

        Catch ex As Exception
            hs.WriteLog(PLUGIN_NAME, "HSEvent " & ex.Message)
        End Try
    End Sub
    Public Sub DoHsEvent()
        Do While Not gShutdown
            Try
                Dim bRetry As Boolean = True
                Dim iQueueSize As Integer
                Do While bRetry AndAlso Not gShutdown
                    Try
                        SyncLock oHSEventCollection.SyncRoot
                            iQueueSize = oHSEventCollection.Count
                        End SyncLock

                        Do While iQueueSize > 0 AndAlso Not gShutdown
                            Try
                                Dim sLog As String = oHSEventCollection.Dequeue()
                                For Each sMatch As String In logList.Values
                                    If InStr(sLog, sMatch, vbTextCompare) > 0 Then
                                        RestartIt()
                                        oHSEventCollection.Clear()
                                        Exit For
                                    End If
                                Next
                                SyncLock oHSEventCollection.SyncRoot
                                    iQueueSize = oHSEventCollection.Count
                                End SyncLock

                            Catch ex As Exception
                                oHSEventCollection.Clear()
                                iQueueSize = 0
                            End Try
```

```
                        Loop
                        bRetry = False
                    Catch ex As InvalidOperationException
                    Catch ex As Exception
                        hs.WriteLogEx(PLUGIN_NAME, "DoHSEvent Thread", ex.Message)
                    End Try
                Loop
                If gShutdown Then
                    Console.WriteLine("HSEvent Shutdown")
                End If

            Catch ex As Exception
                hs.WriteLogEx(PLUGIN_NAME, "DoHSEvent", ex.Message)
            End Try

            If Not gShutdown Then
                oEventQueueWaiting.WaitOne()
            End If
        Loop

    End Sub

    Private Sub RestartIt()

        If gProcessId <> 0 Then
            hs.WriteLog(PLUGIN_NAME, "Stopping mcsMQTT process " &
gProcessId.ToString())

            Try
                Dim oProcess As Process = Process.GetProcessById(gProcessId)
                If oProcess IsNot Nothing Then
                    'pid = oProcess.Id
                    Console.WriteLine("Monitor requesting mcsMQTT Shutdown")
                    Try
                        hs.PluginFunction("mcsMQTT", "", "Shutdown", {""}) 'let plugin
do orderly shutdown.  Expect process to disappear at this point
                    Catch ex As Exception
                        Console.WriteLine("mcsMQTT PluginFunction call error " &
ex.Message)
                    End Try

                    Try
                        oProcess = Process.GetProcessById(gProcessId)
                        If oProcess IsNot Nothing Then 'oProcess.Id = pid Then
                            Console.WriteLine("Killing mcsMQTT Process")
                            Try
                                oProcess.Kill()
                                oProcess.WaitForExit(100000)
                            Catch ex As Exception
                                Console.WriteLine("mcsMQTT Process Kill Error " &
ex.Message)
                            End Try
                            'Exit For
                        End If

                    Catch ex As Exception
                        'process has succefully stopped
                    End Try
                    'Next
                    Console.WriteLine("mcsMQTT Process Abort Completed")
                End If

            Catch ex As Exception
```

```
                Console.WriteLine("Process " & gProcessId.ToString & " is not running,
unable to stop.")
            End Try
        End If
        'setup for next monitoring
        gProcessId = 0
        PeriodicStatusTimer.Start()

    End Sub

    Private Sub PeriodicStatusTimer_Elapsed(ByVal Sender As System.Object, ByVal e As
System.Timers.ElapsedEventArgs) Handles PeriodicStatusTimer.Elapsed
        Try
            gProcessId = hs.PluginFunction("mcsMQTT", "", "ProcessId", {""})
            If gProcessId <> 0 Then
                PeriodicStatusTimer.Stop()
                hs.WriteLog(PLUGIN_NAME, "mcsMQTT is process " & gProcessId.ToString)
                Console.WriteLine("mcsMQTT is process " & gProcessId.ToString)
            End If
        Catch ex As Exception
        End Try
    End Sub
```

7.2.3 Custom Database Scripting

7.2.3.1 Expand Database

mcsMQTT integrates the UI with database functions for message history and device history. A custom SQLite database table can be created and data store in the database via scripting methods. The scripting methods of interest to support this are bolded in the sample script below

CreateCustomDatabase, SaveToCustomDatabase and **Replacement** will normally be used with the custom database.

CreateCustomDatabase accepts an array of column names (fields) that are to be created in the MQTT_Custom table of the MQTTHistory.db database. mcsMQTT prepends the fields Sequence and LastDate. This method can be call once or each time data is being written. It is not possible to alter the set of columns after they have been created. Options are to manually edit the structure or to delete the table and let a new one be created.

SaveToCustomDatabase accepts the same size array of values to store. mcsMQTT populates the Sequence and LastDate fileds automatically.

Replacement is a means to get access to data using replacement variables. These variables are defined in Table 2.

The script below is design to be run on receipt a MQTT topic. The expected payload is CSV. A set of fields are parsed from the CSV data and the values are stored to the database.

```
Sub Main(parm as object)

  Const ENTER_LOCATION as String = "MainGate"
  Const EXIT_LOCATION as String = "ExitGate"
  Const NANNY_CODE as String = "1234"
  Const URL_TOPIC as String = "192.168.1.248:8992.TCP"
  Const KEYPAD_GLOBAL as String = "KeypadExitTime"

      'parms contains the received message payload which is expected to be
CSV string such as
      '    2022-03-25T09:10:39, Confirmation=202, GateExit, Function=99,
Code=1234
      Dim payloadOfInterest = "$$PAYLOAD:(" & URL_TOPIC & "):"
      Dim payload as string = hs.PluginFunction("mcsMQTT", "",
"Replacement",{" & payloadOfInterest & "})

      'parse and validate the data
      Dim arrCSV() as String = payload.Split(",")
      if arrCSV.Length < 2 then
          exit sub
      end if

      'setup the database fields and put the parsed data values in the
appropriate fields
      Dim arrFields() as string =
{"Confirmation","Location","Function","Code","Button","Minutes"}
      Dim arrValues() as string = {"","","","","","",""}
      for iValue as integer = 1 to arrCSV.Length - 1
          Dim arrKeyValue() as string = arrCSV(iValue).Split("=")
```

```
          'the location field does not have a key, but only a value such as
"MainGate"
          if arrKeyValue.Length > 1 then
              for iField as integer = 0 to arrFields.Length - 1
                  if arrKeyValue(0).Trim = arrFields(iField) then
                      arrValues(iField) = arrKeyValue(1).Trim
                      exit for
                  end if
              next
          Else
              arrValues(1) = arrCSV(iValue).Trim
          End if
      next

      'Use global variable to remember when Nanny entered
      Dim enterTime as object = hs.GetVar(KEYPAD_GLOBAL)
      if enterTime is Nothing then
          hs.CreateVar(KEYPAD_GLOBAL)
          enterTime = now
      end if

      if arrValues(1) = ENTER_LOCATION then
          arrValues(arrValues.Length-1) = 0
          enterTime = now
          hs.SaveVar(KEYPAD_GLOBAL,enterTime)
      end if

      'Minutes database field is computed.  0 for all records except Nanny
Exit
      arrValues(arrValues.Length-1) = 0
      if arrValues(1) = EXIT_LOCATION andalso arrValues(3) = NANNY_CODE then
          if isDate(enterTime) then
              arrValues(arrValues.Length-1) = DateDiff(DateInterval.Minute,
enterTime, Now)
          end if
      end if

      'put the data in the database
      hs.PluginFunction("mcsMQTT", "", "CreateCustomDatabase",arrFields)
      hs.PluginFunction("mcsMQTT", "", "SaveToCustomDatabase",arrValues)
End Sub
```

7.2.3.2 Retrieve Data

The scripting method ReadDatabase is available to request values for a specified duration from the Long-Term database. The input parameter is an array with the first element being the Ref number as a string. The optional second and third parameters provide the date range being retrieved. Data from the last 30 seconds is retrieved if no range is specified.

InfluxDB contains an implicit date field while the other databases use LastDate as the field name. If data was recorded with UTC date, then the returned data will also be UTC. InfluxDB data is returned in ordered pairs of [date,value]. If multiple values are returned then the pairs will be comma-separated. JSON format is used for others {“LastDate”:”yyyy-mm-dd hh-mm-ss”}, {“LastDate”:”yyyy-mm-dd hh-mm-ss”} etc.

```
    Dim arrQuery() as string = {iRef.ToString()} ',optional startDate,
startTime as 2nd and 3rd parameters - default last 30 seconds

    Dim sResult as String =
hs.PluginFunction("mcsMQTT","","ReadDatabase",arrQuery) 'read value recorded
in last 30 seconds
```

7.2.3.3 Write Data

The scripting method WriteDatabase is available to store values in the existing database. The parameter array contains three string entries such as WriteDatabase({“123”,”first_kitchen_lamp,mytag”,”100”}) where “first_kitchen_lamp” is the Loc2_Loc1_Name to be stored in “device” database field, “mytag” is a tag field, “100” is stored in the “value” field. “123” is the Ref of the device which is used to know what format is expected in the “device” field.

'(0) = iRef
'(1) = FieldName,Tag ,Tag is optional
'(2) = value

7.2.3.4 Execute SQL Command

The scripting method ExecuteDatabaseCommand is available to deliver SQL commands to the database. The parameter array contains one item which is the SQL command. For example, the following will remove all records for the device field that contans the name first_kitchen_lamp.

```
Function Main(parm as object) as String

  Dim arrQuery() as string = {"DELETE FROM mcsmqtt WHERE device=' first_kitchen_lamp’'"}

  Dim sResult as String = hs.PluginFunction("mcsMQTT","","ExecuteDatabaseCommand",arrQuery)

End Function
```

7.2.4 PluginFunction Reference Methods

The full list of of methods that are contained with mcsMQTT PluginFunction scripting call is listed below. Most are previously described in the context of where they may be used.

Return SendMqttMessage(parms)
Return ReceiveMqttMessage(parms)
Return RegisterTopicReceivedScript(parms)
Return SendVoiceMonkey(parms)
Return CreateCustomDatabase(parms)
Return SaveToCustomDatabase(parms)
Return Replacement(parms)
Return ReadDatabase(parms)
Return WriteDatabase(parms)
Return ExecuteDatabaseCommand(parms)
Return ProcessId(parms)
Return Shutdown(parms)
Return Name(parms)
Return PagePreLoad(parms)
Return EditPropertyByRef(parms)
Return EditPropertyByTopic(parms)
Return ClearVSP(parms)
Return AddVSP(parms)

8 History

The History feature of mcsMQTT provides a means to record all or a subset of HS DeviceValues and/or MQTT Topics and Payload that pass through mcsMQTT.

Two types ofdatabase repositories are available. Long term or high-volume data can be strored in InfluxDB, mySQL or MS SQL Server. Shorter term data for use in analysis is stored in SQLite.

The overall data collection parameters are available on the top of the History tab shown in Figure 50. The Association page "H"istory, "L"ongTerm, and "S"hortTerm columns provide retention selection on an item-by-item basis.

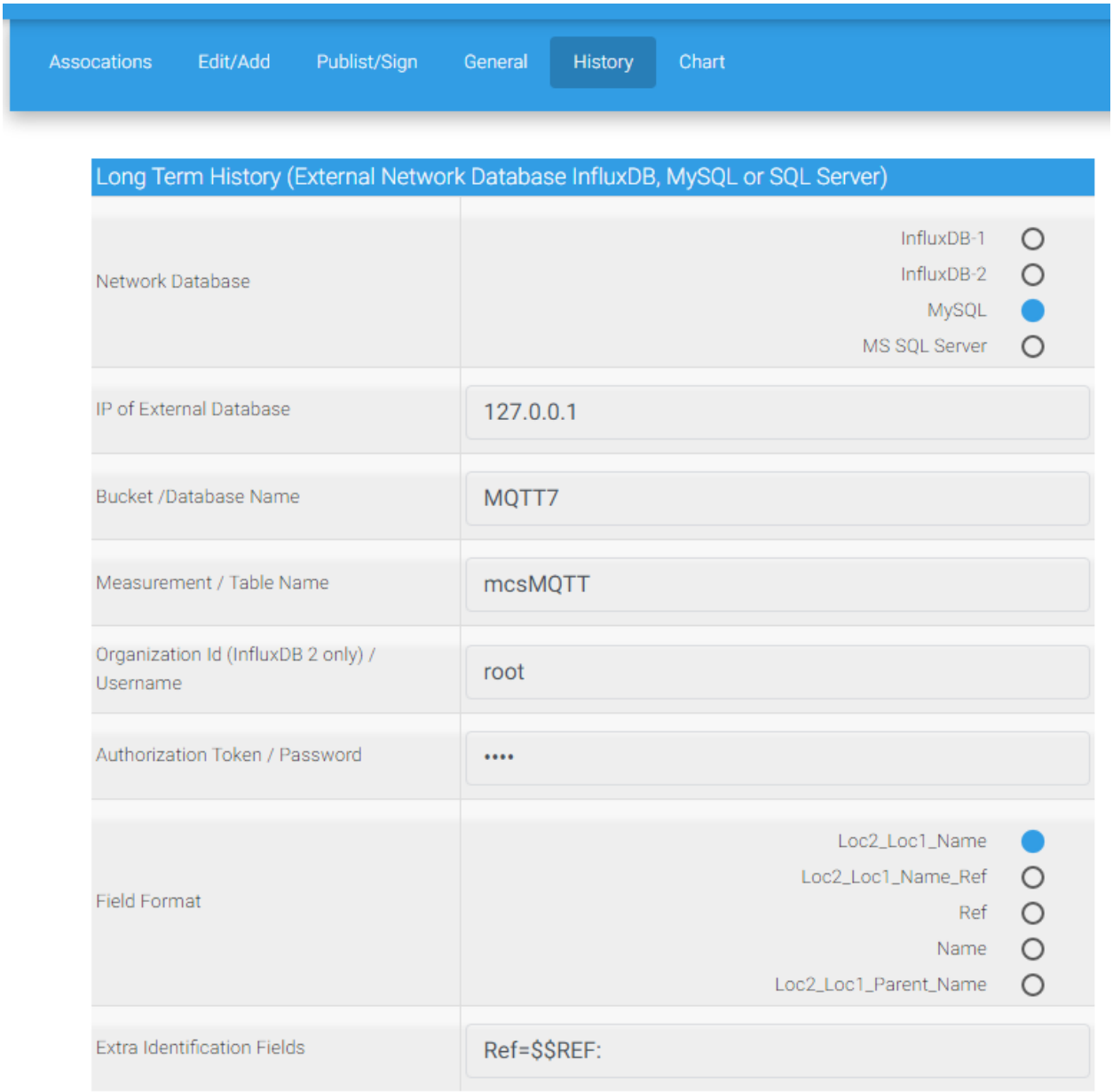


Figure 50 Hitory Data Collection Setup

8.1 Long Term Storage in Network Database (InfluxDB, mySQL, SQL Server)

One of the four available long-term network-connected is selected for mcsMQTT to use. This selection is based primarily on a user's environment and which database they are most comfortable. mcsMQTT will create that database (or buckets) it will use. It will also create the schema used in the database. These database and schema parameters are specified at the top of the History tab.

The long-term data storage is enabled by providing an IP address of the databse which is the first row on the History Tab shown in Figure 50. If an IP (or network name) is not entered then long term storage will not be enabled.

Different authentication methods are used based upon the database type selected. Influx 1.8, mySQL, and SQL Sever can be used without entry of username and password if the user has setup their providers to no require authentication. In the MS Server case authentication is required, but can be setup as integrated with Windows login credentials. In this case username and password fields are left blank in mcsMQTT. Influx DB2 requires a authentication token and requires an Organization Id. When using InfluxDB 2.0 the token is generated by the InfluxDB UI tools. It will be a long string of characters. When using InfluxDB 1.8 the token is entered as username:password and the username field in mcsMQTT is left blank.

Data storage will include a timestamp, device identification, and DeviceValue. Device identication can be one of several formats which are selected by a radio button. The field format radio should be selected to make the record unique. The Floor, Room and Name options are under user control. The Ref option uses a number assigned by HS so if a device is deleted and then recreated in the future it will get a new Ref number and could be more difficulat to identify in the database.

If a user desires to have additional data recorded then they specify the additional fields in the additional identification fields text box. The additional identification fields can be used to help with query of the data for Graphana or other uses. The data is entered as a set of key-value pairs which each pair separated by comma. Replacement variables will often be used and expressions are also allowed. An example of adding three fields (Categroy, Derived, and Location) to the measurement is shown below. The value of each being determined by replacement variables.

```
Category=$$TAG:,Derived=<<(ROUND($$DVR:(123):+$$VALUE:)/2,1)>>,Location
=$$FLOOR:_$$ROOM:_$$NAME:
```

The recording of a record in the database will be performed each time an identified HS Device (Feature) has been set by user, HS or any plugin action. The recording conditions are specified globally on the History tab as shown in Figure 51 and individually on the Association tab with use of "L" checkbox shown in Figure 52.

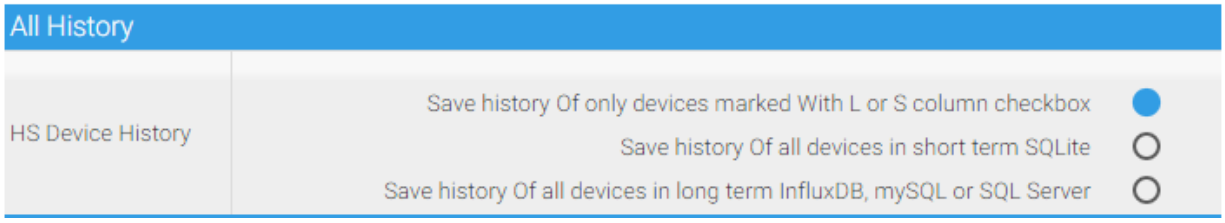


Figure 51 Database Storage Global Settings

The selection of the Devices (Features) to be used is done on the Association Tab using the checkbox in the "L"ongTerm column. This checkbox will only be available on children Devices (Features) available in HS. An example is shown in Figure 52.

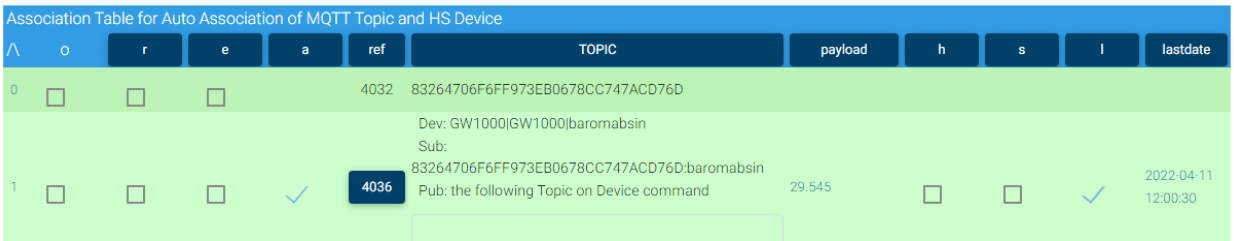


Figure 52 Selection of HS Devices/Features for Recording in Long Term Database

Use of the data recorded in Long Term Database will typically be for use with external charting and analysis tools, also can be viewed from within mcsMQTT on the History tab and Chart tab of the MQTT Page. On-demand chart requests are also available. Grafana is a popular graphic package that is compatible with InfluxDB and mySQL. mcsMQTT has no built-in support of charts using Grafana, but the data recorded by mcsMQTT can be used with Grafana. mcsMQTT uses .NET tools for charting.

The internet is a good source for site installation of a network database. For those who want to use RPi for InfluxDB this purpose the following is provided as a good source to walk one through it. It will install a version 1.8 of InfluxDB on a standard 32 bit RPi image.

https://pimylifeup.com/raspberry-pi-influxdb/

It will contain a sequence of commands that will be entred to accomplish the installation. They are copied here for ease of reference

```
sudo apt update
sudo apt upgrade
wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
echo "deb https://repos.influxdata.com/debian buster stable" | sudo tee
/etc/apt/sources.list.d/influxdb.list
sudo apt update
sudo apt install influxdb
sudo systemctl unmask influxdb
sudo systemctl enable influxdb
sudo systemctl start influxdb
```

```
influx
CREATE DATABASE pimylifeuptemperature
USE pimylifeuptemperature
INSERT temperature,location=living_room value=20
INSERT temperature,location=living_room value=10
INSERT temperature,location=bedroom value=34
INSERT temperature,location=bedroom value=23
SELECT * FROM temperature
SELECT value FROM temperature WHERE location='bedroom'

CREATE USER admin WITH PASSWORD 'xxx' WITH ALL PRIVILEGES
sudo nano /etc/influxdb/influxdb.conf
 auth-enabled = true
 pprof-enabled = true
 pprof-auth-enabled = true
 ping-auth-enabled = true
sudo systemctl restart influxdb
influx -username admin -password xxx
```

InfluxDB version 2 is emerging with downloads available for 64-bit images at https://portal.influxdata.com/downloads/ . When version 2 is used there is a requirement that authentication and organizations be identified. This is setup during the initial run after the install. This setup will also include the creation of a bucket for the data. This in contrast with version 1.8 where mcsMQTT is able to create the bucket if not already setup.

Some other differences are that the authentication is optional with 1.8 and if included the Authentican Token is username:password. For version 2 a token needs to be generated within InfluxDB (https://docs.influxdata.com/influxdb/cloud/security/tokens/create-token/) and then used in the mcsMQTT setup. Version 2 also requires that an organization id be specified. Like the bucket this is handled during the initial setup. The org id for version 2 needs to be transcripted into the mcsMQTT setup. Note this is the org Id and not the org Name. The Id is needed when creating the database bucket. When using InfluxDB versions before 2, the organization Id needs to be blank so mcsMQTT will know which database version is being used.

MySQL and MS SQL Server have free community editions that are well documented and included tools for viewing the schema and data.

8.2 Short Term Storage in SQLite

Data stored in SQLite database is intended for near term analysis. At the start of each day mcsMQTT removes records from the database that are older than the retention period setup by the user. There are no specific retention limits for the SQLite data, but large datasets will tend to have higher CPU utilization and greater potential for corruption. The SQLite database is located at data\mcsMQTT\mcsMQTTHistory.db.

Use of SQLite for history data is enabled from the History Tab by selecting a non-zero value for the number of days of history that will be retained. If this field is blank then no short-term data will be collected.

The selection of data for retention in the SQLite database is the same as for the LongTerm data with global settings on the History tab and individual device settings on the Association Tab in the "S"hortTerm column checkbox.

8.1 Viewing History Data

Access to the History data is from the History Tab and Chart Tab. It will present a checkbox and pull-down selectors as shown in Figure 53 to select from the total History in the database. In this figure's example all data between April 1, 2018 and April 2, 2018 that start with Topic GarageDoor is being selected.

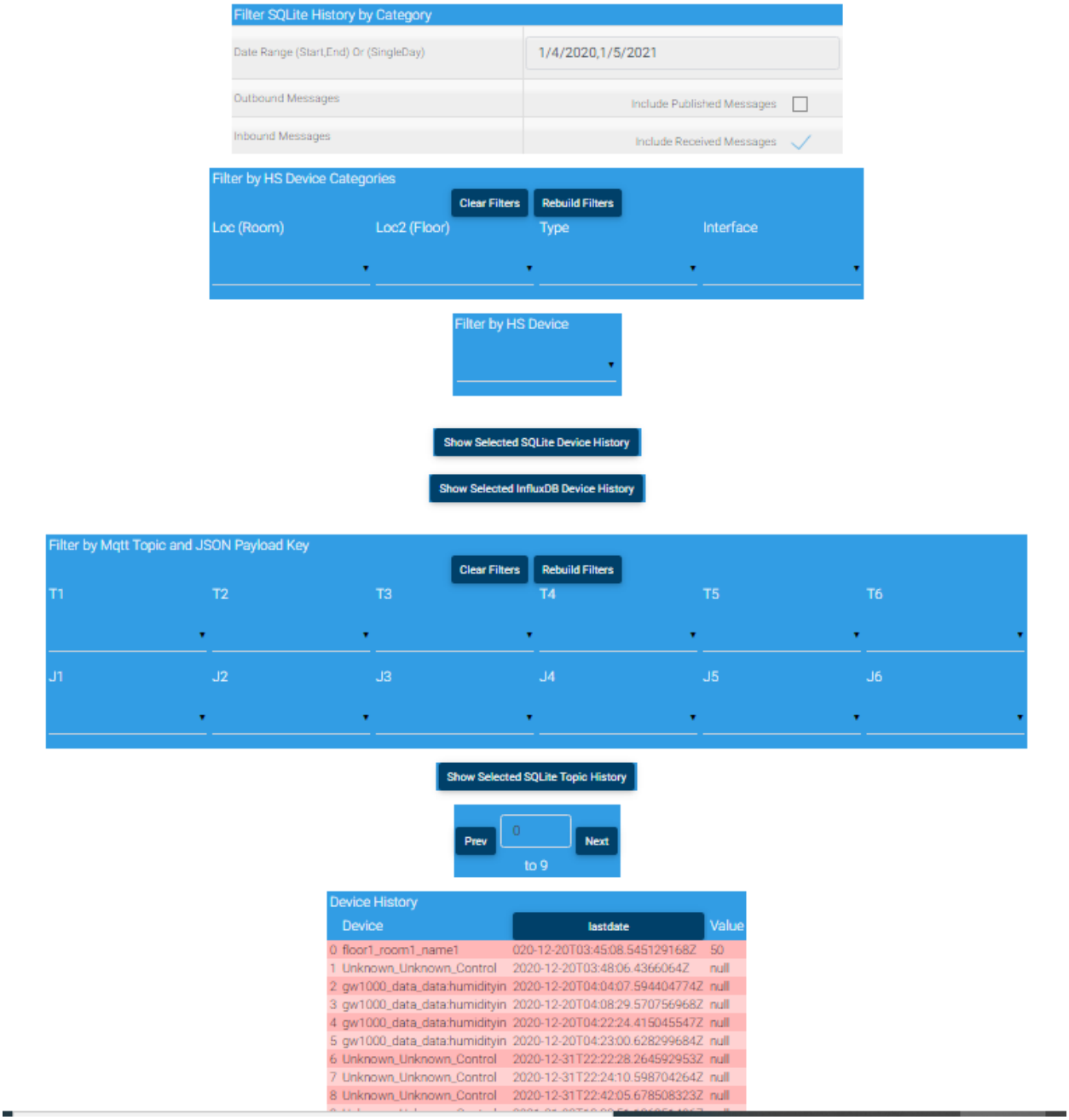


Figure 53 History Filter Selection and Device Display

History data can be viewed from either the ShortTerm or LongTerm database in a tabular format on the History Tab and graphical format on the Chart Tab. This section describes the tabular format.

ShortTerm is able to collect either MQTT messages and payloads or HS Device Values. LongTerm is able to only collect data resulting from HS Device Value changes. Three “Show” buttons are provided to view MQTT Topic history, HS Device History from SQLite and HS Device History from LongTerm database. It

also provides filters to contrain the amount of data that presented so it will be easier to locate in the table that is rendered. These are shown in Figure 53.

When the “Show Selected …” button is clicked the database is queried and the first 20 rows of the selected history is shown in the table. An example is in Figure 54. At the top of the table are four buttons that when clicked will sort the data in that column in ascending of descending order. The first column P/S shows the Subscription Topics and Publish Topics.

Note that the full Topics are displayed vs. the individual JSON items had JSON decoding been selected.

Scrolling buttons are provided to advance through the history data if more than 20 rows are presented based upon the filters selected. A text box entry is also presented to allow 20 records to be selected starting at a particular position. In general, it will be more convenient to use the filters to show a limited set of data rather than using the scrolling/windowing provisions.

Message History

	P/S	LastDate	Topic	Payload
0	S	2018-04-08 18:04:09	FilteredWater/INFO2	{"WebServerMode":"Admin", "Hostname":"FilteredWater", "IPAddress":"192.168.0.4"}
1	S	2018-04-08 17:36:28	FilteredWater/INFO2	{"WebServerMode":"Admin", "Hostname":"FilteredWater", "IPAddress":"192.168.0.4"}
2	S	2018-04-08 17:29:33	FilteredWater/INFO2	{"WebServerMode":"Admin", "Hostname":"FilteredWater", "IPAddress":"192.168.0.4"}
3	S	2018-04-08 17:10:05	FilteredWater/INFO2	{"WebServerMode":"Admin", "Hostname":"FilteredWater", "IPAddress":"192.168.0.4"}
4	S	2018-04-08 16:55:50	FilteredWater/INFO2	{"WebServerMode":"Admin", "Hostname":"FilteredWater", "IPAddress":"192.168.0.4"}
5	S	2018-04-08 16:32:30	FilteredWater/INFO2	{"WebServerMode":"Admin", "Hostname":"FilteredWater", "IPAddress":"192.168.0.4"}
6	S	2018-04-08 18:02:14	FilteredWater/UPTIME	{"Time":"2018-04-08T17:02:00", "Uptime":1}
7	S	2018-04-08 17:22:10	FilteredWater/STATE	{"Time":"2018-04-08T16:22:18", "Uptime":384 367, "Vcc":3.105, "Wifi":{"AP":2, "SSId":"Anthem", "RSSI":74, "IPAddress":"192.168.0.4", "APMac":"E0:3F:49:9D:B9:68"}}
8	S	2018-04-08 17:08:21	FilteredWater/STATE	{"Time":"2018-04-08T16:08:29", "Uptime":384 367, "Vcc":3.105, "Wifi":{"AP":2, "SSId":"Anthem", "RSSI":72, "IPAddress":"192.168.0.4", "APMac":"E0:3F:49:9D:B9:68"}}
9	S	2018-04-08 17:01:52	FilteredWater/UPTIME	{"Time":"2018-04-08T16:02:00", "Uptime":1}

Figure 54 MQTT Topic History Display

9 Charts

The Chart tab of mcsMQTT provides a means to graphically observe the time history of a particular item that had been collected in the History database. See Section 8 for setup associated with collection of history data.

Time history can be collected for MQTT subscribe and publish message and it can be collected to HS Device Value changes. The MQTT messages are identified by Topic. The HS Devices are identified by Device Reference. One of the two can be selected for each of up to two vertical axes.

The History tab selects the global conditions for collection of data to be charted. The Association tab identifies specific items to be available for charting. The Topic selectors on the Chart tab will provide the list of potentially available items. The selectors can be all topics with numeric data or can be filtered with the Topic-based selectors at the top of the page and the "Topic Selector" checkbox to include only associated topics which is located just above the selectors. Text filters can also be used in a similar manner.

The chart setup is specified by five selections shown in Figure 55. The Date range can be a single day or span of days. The start time of the earliest date and end time of the latest date can be specified if partial days are to be used to get finer resolution of the displayed data.

Lines can be selected for either left or right Y axis. Left axis lines are solid and right axis lines are dashed. Labels and colors are used to identify lines on the chart. If minimum and/or maximum values for the Y axis is specified then they will be used, otherwise auto scaling will be done that provides about 10% top and bottom margins.

The selection of the items for the lines is from a pair of pull-downs. If there are a large number of items saved in the History then it may be beneficial to use the checkbox that includes only associated items (i.e. mapped to HS Devices) to reduce the length of the pull-down selectors.

Once the chart lines are constructed the set can be saved with a chart name and then can be later restored with the load selector. The selector will also include "All". When selected then all defined charts will be shown in round robin sequence with ten second dwell times.

The Topics available are synchronized with the selections made for Associations and History. If, for example, the non-Plug-in are not checked for inclusion in the History data, then the Chart Topic Selector will have no non-Plug-in items.

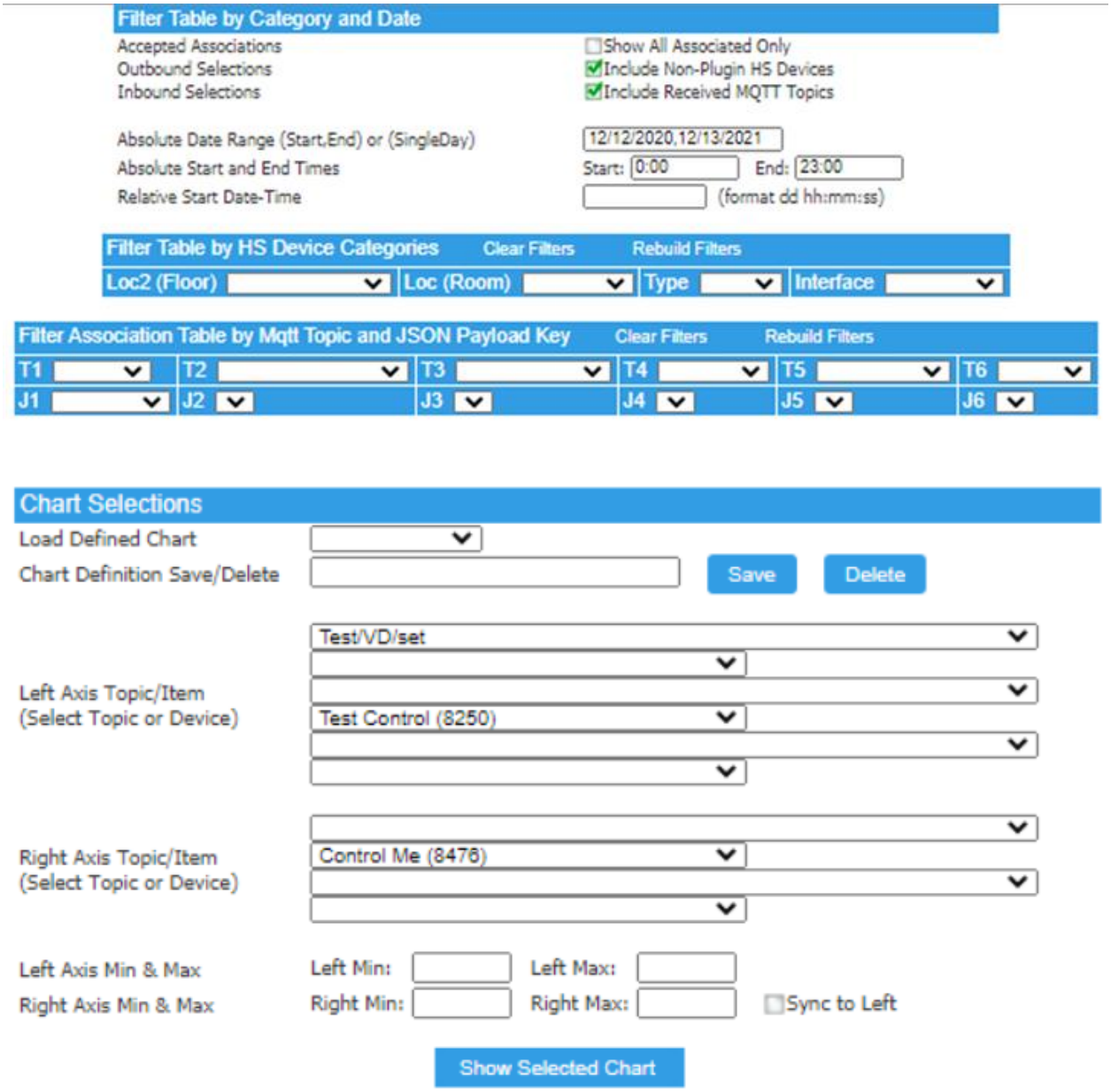


Figure 55 Chart Setup

Charting parameters can be setup and given a name. The name can then later be entered to restore the parameters. These controls are at the top and bottom of Figure 55.

When the "Show Selected Chart" button is used the requested chart will be shown below the button such as is shown in Figure 57. In this example three lines are drawn. If there is only one line on each Y axis then the axis and line color (white on left and yellow on right) can be used to identify the item being charted. If more than one exists on either Y axis then a legend will be shown in the upper right of the chart as shown in Figure 56. The legend item will start with the axis code of 1 or 2 and then the Device

Ref of the signal. The end of the legend will be the current value of the item. The Y axis lables will contains both the name and the ref of the item.

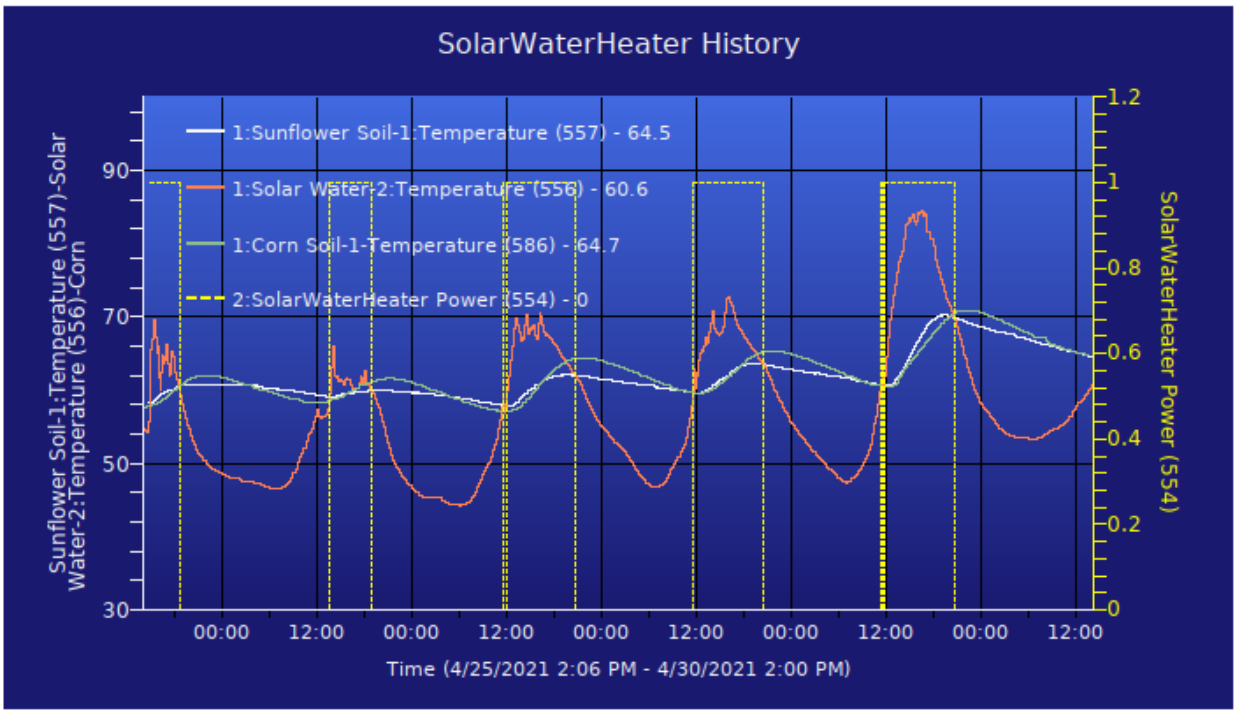


Figure 56 Chart with Line Legend

When a topic has VSP relationships the VSP legend will be included below the chart such as in Figure 57. The left axis is from Payloads that contain “OPEN”, “CLOSED” and “INDETERMINATE” text values. If these have been setup in mcsMQTT as Value-Status-Pairs then the same Status-Value relationship will be used, otherwise they will be assigned dynamically. The pairs legend is shown below the chart. The white line (GarageDoor/Door) shows at 9:56 going from CLOSED (0) to INDETERMINATE (1) and then shortly thereafter going to OPEN (2). About one minute later the door is shown to close while transitioning through the indeterminate state.

On the right Y axis an RSSI value is shown through four values ranging from 74 to 80 during the selected time period. The line and the axis labels are in yellow. Since the data is numeric there is no VSP legend generated below the chart.

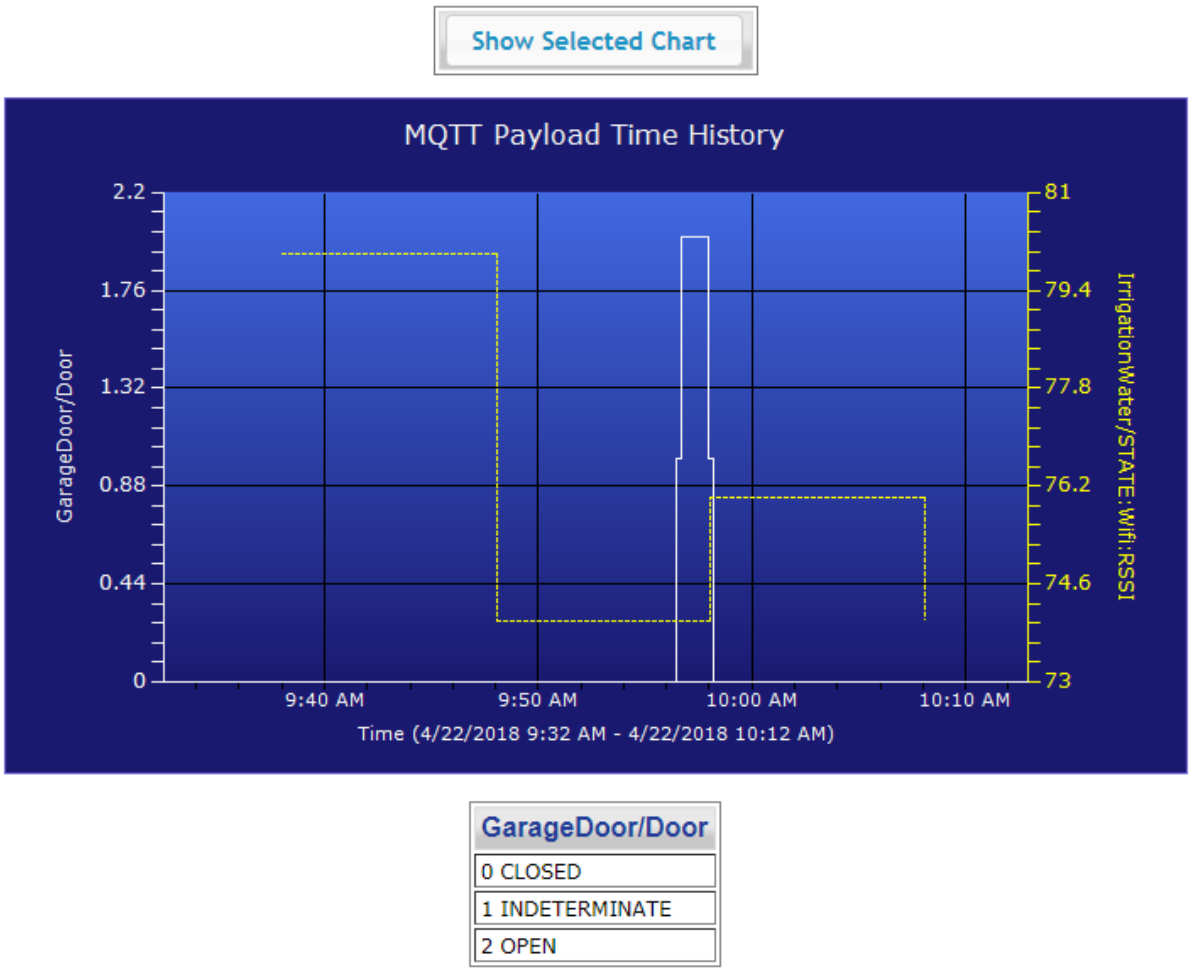


Figure 57 Chart Display with VSP Legend

MQTT-sourced charts can also be produced by clicking on the Payload value of a row in the Association tab. HS Device-sourced charts can be produced by clicking on the LastDate value of a row in the Association tab if the hyperlink is available. It becomes available when the "S"hortTerm column checkbox has been selected.

The chart will be shown in a new browser window in the upper left of the screen. In this case the chart will contain only the left axis and the time span will include all data for the item in the history database. This can be customized by making a chart definition and saved under the same name as the Topic for MQTT-sourced or Ref number for Device-sourced. An example is Figure 210. In this case the parameters will be used to define what is on the chart, the scaling and the time span.

A third method is provided to generate charts through automation rather than interactively. HTTP get requests are made to the HS server of the form

```
http://<IP>/MQTT?< parameters>
```

where <IP> is the computer name or IP address of the HS server. It may include the port if something other than port 80 is used. <parameters> are from the items below

Identification

Chart=name – name is chart definition file name
Payload=name – name is chart definition file name
Y1=name –name is topic on left Y axis

Formatting

File=string – location to place created .png file. Either full path of just filename if being placed in the \html\mcsMQTT subfolder of HS
Y1Min=number - optional left Y axis minimum
Y1Max=number - optional left Y axis maximum
Y2=name - optional, topic name on right Y axis
Y2Min=number - optional right Y axis minimum
Y2Max=number - optional right Y axis maximum
StartDate=date - optional start date
EndDate=date - optional end date
Duration=timespan - optional in format of dd hh:mm:ss (end date is set at current date/time)

Duration has priority over StartDate and EndDate
If no date is given then period is current day
If no Y min or max is given then auto scaling is done

Three identification options exist. Either the word "Chart" or "Payload" is required. If it is used with a name value then that name will be used as the saved chart definition name and all formatting parameters will default to the named definition values. If it is used without the name value then the Y1 value is required to identify what will be put on the chart.

"Chart" will return the filename of the chart with the file deposited in the HS server html\mcsMQTT folder. "Payload" will return an HTML formatted page that uses an <img> tag to load the graphic file.

Formatting parameters can be used to supersede the definition file or if a definition file is not specified then they can be set to provide whatever chart customization is desired. If a parameter is not used then defaults will be selected.

A few examples are shown below for the HS3 plugin.

```
http://192.168.0.14/MQTT?Chart&Y1=HyderonRain/STATE:Wifi:RSSI&StartDate=1-1-
2018&EndDate=5-1-2018&Y1Min=0&Y1Max=100

http://192.168.0.14/MQTT?Chart&Y1=HyderonRain/STATE:Wifi:RSSI&Y1Min=0&Y1Max=100&Durati
on= 10 20:30:40

http://192.168.0.14/MQTT?Chart=RSSI-Garage

http://192.168.0.14/MQTT?Payload=RSSI-Garage
```

For the HS4 plugin the URL changes to use /mcsMQTT/MQTT.html rather than /MQTT and each parameter is separated by "?" rather than "&" as shown below.

```
http://192.168.0.14/
mcsMQTT/MQTT.html?Chart?Y1=HyderonRain/STATE:Wifi:RSSI?StartDate=1-1-2018?EndDate=5-
1-2018?Y1Min=0?Y1Max=100

http://192.168.0.14/
mcsMQTT/MQTT.html?Chart?Y1=HyderonRain/STATE:Wifi:RSSI?Y1Min=0?Y1Max=100?Duration=10
20:30:40

http://192.168.0.14/ mcsMQTT/MQTT.html?Chart=RSSI-Garage

http://192.168.0.14/ mcsMQTT/MQTT.html?Payload=RSSI-Garage
```

9.1 Charts with HS Touch

HSTouch provides a control that can be used as the container for an image file such as one generated by mcsMQTT. There may be multiple ways to setup HS Touch. The following discussion illustrates a specific way as shown below. In this case the image control is placed on the canvas. The ImageURLNormal propery is set to the HS URL of a png file that will be created by mcsMQTT. The IsVideo property is set to true to facilitate HSTouch monitoring for a change in the file.

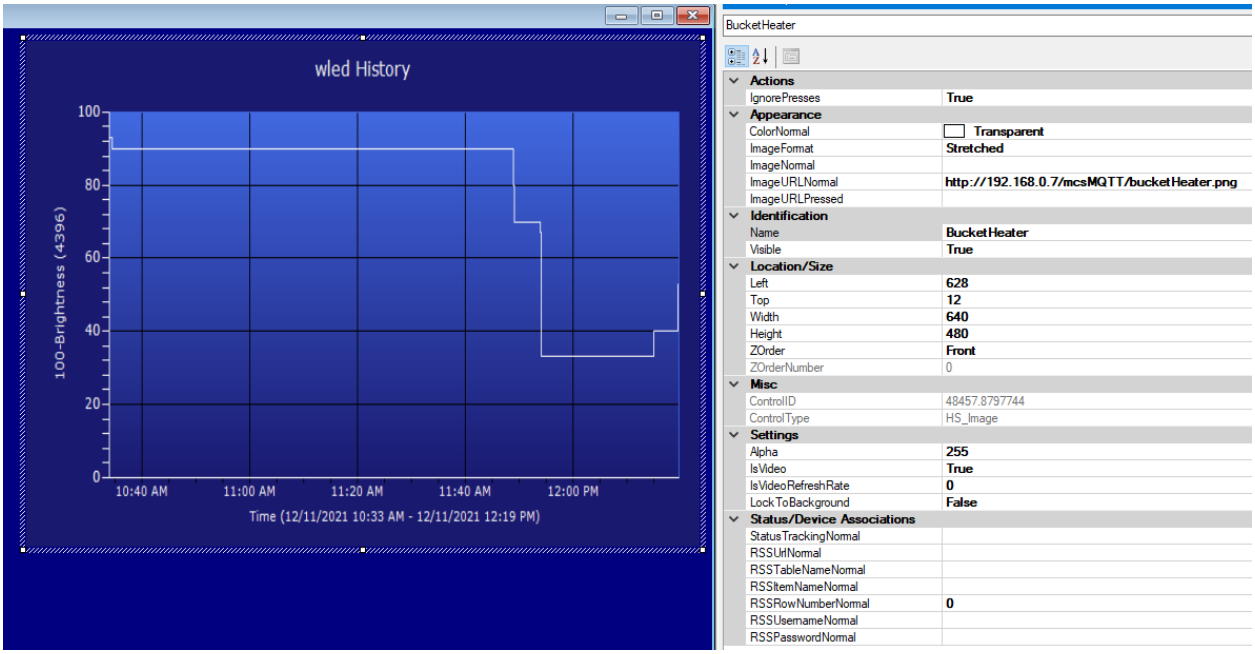


Figure 58 HS Touch Setup to Show Chart

An event is setup in HS for periodic execution at the update rate desired for the chart. In this example it is set to 15 seconds. The event action is an immediate script that uses the hs.GetURL method that requests mcsMQTT to rebuild the chart from the chart setup called “wled” and to place the png in file bucketHeater.png. When the event is run the file will be rewritten every 15 seconds and HSTouch will recognize the update and refresh the image control on the screen. Note the syntax for File parameter in hs.GetURL uses “?” to separate it from the Chart parameter. For HS3 it would be “&”.

Figure 59 Event to Refresh Chart Every 15 Seconds

When mcsMQTT generates the png file it will actually create two files. One will be “plotXX.png” and the second will be the name given in the File parameters if this parameter exists. The XX will increment for each new chart generated. The plotXX.png is appropriate for clients that are looking for a new file typically embedded in HTML img tags such is produced when using the Payload rather than the Chart parameter for the chart being generated.

10 mcsMQTT Self Signed Certificate Support

This section, as well as the SSL implementation within mcsMQTT has been provided by **vasrc** from Homeseer Message Board.

There are two parts. The first part is a general overview of the SSL process as it relates to IOT/MQTT. The second part is an instruction manual on how to create, install and configure SSL communications using the mcsMQTT plug-in on HS.

10.1 Part I. SSL/TLS Communications

10.1.1 Why encrypt your IOT/MQTT Network

Since the MQTT protocol isn't currently encrypted natively, all that's needed is the Broker username and password to access your IOT network and create havoc. If you don't use SSL/TLS encryption, all of the usernames and password may be sent "in the clear" (you can MD5 hash passwords on some devices).

So overall, it's just a good thing to do since MQTT is such a basic protocol (invented back in 1999 by IBM, before there were a lot of issues with security) and has no real internal security provisions.

10.1.2 SSL Communications Overview

In this document, we use SSL to represent both TLS and SSL. SSL certificates are also commonly called public key, digital or identity certificates.

To understand why we use SSL certificates, it's helpful to understand how they work. This is a high-level description of the process as it relates to IOT.

Some of the key elements when communicating with SSL are:

- An SSL certificate is an electronic document used to prove the ownership of a public key. The certificate includes information about the key, information about the identity of its owner (called the subject), and the digital signature of an entity that has verified the certificate's contents (called the issuer). If the signature is valid, and the software examining the certificate trusts the issuer, then it can use that key to communicate securely with the certificate's subject. In IOT networks a certificate's subject is typically a computer or other device. In a typical IOT public-key infrastructure (PKI) scheme, the certificate issuer's certificate authority (CA) is usually self-signed vs signed by a company that charges customers (root certificate).
- Certification authority (CA) is a third party that is trusted by both the SSL client (Device) and the SSL server (Broker). Its role is to provide the SSL client (Device) and the SSL server (Broker) a means to authenticate that each of their certificate(s) were issued by a trusted source. CA certificates can be both Root authorized or Self-Signed.

- When establishing SSL-based encrypted communication channels, the authentication of the devices communicating is optional. This means the ownership of the certificates may or may not be checked to confirm who the users (devices) are, but rather they are used to provide encryption services to the data being sent and received.
- Due to the diverse number of MQTT capable devices available, the type of SSL communication will vary. Some of the options are:
    - Unidirectional Authentication: only the client (Device) will verify the SSL server's (Broker) certificate. For end devices this is typically the most common SSL method used currently.
    - Bidirectional Authentication: both the SSL client (Device) and the SSL server (Broker) will mutually verify each other's certificates. This is typically done on more "intelligent" clients such as a Windows or PI based system that have more resources or an MQTT utility such as MQTT_Spy.

10.1.3 Root Signed and Self-Signed certificates

- A **root certificate** is a public key certificate that identifies a root certificate authority (CA). While root certificates are technically self-signed they are not the same as the certificates we call usually call self-signed. A root certificates primary difference is that their signers have been universally accepted as being valid. Lists of these valid signers are typically included in Browsers and applications so they can be used to confirm that SSL certificate (usually purchased) that's being used is valid. *https://en.wikipedia.org/wiki/Root_certificate*

- A **self-signed certificate** is very similar to a root certificate with the primary difference being it is only considered valid by the user/vendor that created it and is not universally accepted like root certificates are. Therefore, they have to be verified internally. A self-signed certificate is very common when running an internal DIY IOT network as an external Root signed certificate isn't always easy to validate on smaller IOT devices due to the lack of a Trusted Root store on the device.
- Both types of certificates are used to sign the SSL certificates that are used to provide the authentication and encryption functions. The primary difference of these SSL certificates is whether they are signed internally (self-signed) or externally (Root).

We use SSL communications for two primary reasons:

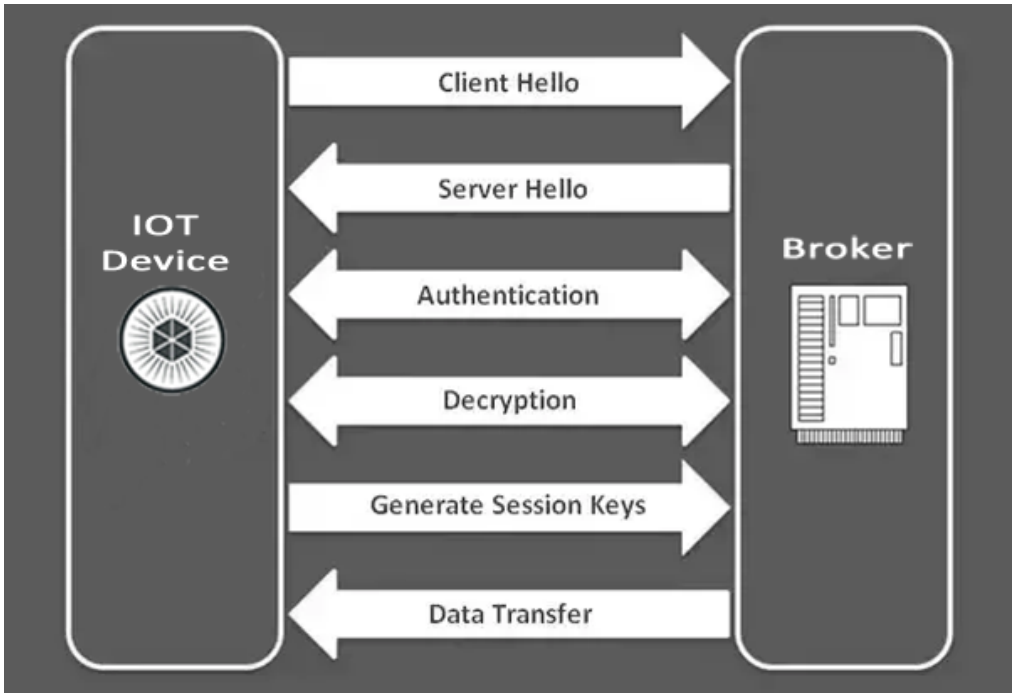
1. Data Encryption. To provide communication security and prevent session hijacking over a network by encrypting the data being between the device and Broker.
2. Device Authentication. To ensure both the Devices and Brokers in our network are who they say they are. This step is called authentication and is not always done in a DIY IOT network.

For data encryption it's necessary to confirm that the SSL certificates and their associated keys match. This level of authentication is to ensure that the data encryption is secure. This is hardcoded into the SSL libraries and is always done. If this level of authentication fails, the network connection will fail.

Device authentication is used to ensure that the SSL certificates represent their owner's identity correctly. This is done using HTTPS all the time so you know that the website you think you're looking at is really that website. This level of authorization requires a CA that has been signed by a Root Authority and is confirmed by checking for a Root certificate that is stored on your computer (usually by the Browser vendor). In a DIY IOT network, this isn't always feasible as the Devices can be limited in memory or pre-programmed to operate in a specific manner already. Therefore, having an SSL certificate that has been signed by a Root Authority won't always be helpful as you'll need to install the Intermediate Certificates
(Certificates that point at the Root Certificates) the same as you would have to install your own Self-Signed certificate. Therefore, many of the IOT devices run with this level of authentication disabled and inherently trust the CA certificate it is assigned (the same as it would an Intermediate/Root certificate).

To provide the minimal (unidirectional) encrypted SSL communications on MQTT devices, two SSL certificates must be installed on the MQTT broker as well as one on each device. Typically, in an internal IOT network, this would be a self-signed CA certificate on both the Broker/server and client/device along with a Broker/server certificate on the Broker, signed by the self-signed certificate. The CA certificate can be reused on all IOT devices. The Broker/server certificate can only be used on the Broker. These certificates provide adequate protection from sniffing of the TCP packets being sent between the devices and the Broker (many times using WiFi). Although most Brokers and some devices do support client certificates (not currently supported on this Plug-in) client certificates are not as commonly used, primarily as on most Brokers they force the user to only use devices that support client authentication. Client authentication currently is used more for commercial systems than home systems.

Here is a simple overview of how a secure SSL session is accomplished



(The Client certificate process is not shown in this diagram)

SSL establishes a secure connection between the device and the Broker and the process of encrypting information and authentication. The technical steps are:

Client Hello:
A Device initially tries to establish connection to the Broker using Client Hello and ask it for identification details like SSL version number, cipher setting.

Server Hello:
After Client Hello, the Broker responds to the Device with its public key along with a copy of its SSL certificate including SSL version, cipher settings, session specific information called Server Hello.

Authentication:
A Device authenticates the details of the Broker certificate (more on Authentication later), generates a pre-master secret, encrypts it with the Brokers certificate public key and send it back to the Broker.

Decryption:
The Broker then decrypts the pre-master secret with its private key. Both the Device and the Broker generate Master secret with agreed ciphers.

Generate Session Keys:
After generating master secret, both the Device and the Broker generate session keys for encryption and decryption used in information exchange.

Data Transfer:
Both the Device and the Broker will now exchange encrypted information.

10.1.4 SSL Options that NEED CLOSURE

Not all devices are SSL capable. It's usually possible to create a second listening port on the Broker for non-secure devices. If you're concerned with network security (i.e. your IOT network isn't isolated on a VLAN or separate Subnet), you should read up on Access Control lists for the Broker. Typically, port 1883 is used for non-secure devices and port 8883 is for SSL type devices. Any port can be selected for either though.

To implement this, both the Broker as well as the Device need to support SSL type communications. Most newer devices should have this capability. In this example, I'll also provide an example of how to configure one of the more popular MQTT brokers, Mosquitto.

10.2 Installing SSL support on the mcsMQTT Plug-in

mcsMQTT provides an automated mechanism to create self-signed certificates, a certificate and key for the Broker and a certificate and key for the mcsMQTT client. This automated functionality is used when a Broker security of None is not selected and certificate files are not setup on the MQTT Page, Broker Tab. In this case a set of certificates and keys are generated in the \data\mcsMQTT subfolder with naming convention derived from the Broker IP that has been setup on the same Page for the Broker certificate, the same for the CA certificate with a "ca" prefix, and the computer name where mcsMQTT is running for the client. The CA certificate, Broker certificate and the Broker key files still need to be manually copied to the computer where the MQTT Broker is running.

If a user desires to use the same capability manually from a command window, then navigate to the \Bin\mcsMQTT folder and enter command

CAandClientCert.exe "FilePath, BrokerName/IP,ClientName/IP"

Where BrokerName/IP is replaced by either the computer name (or fully qualified domain name that can be resolved via DNS) of the MQTT Broker, and ClientName/IP is the same for computer where the client is running. If ClientName/IP is blank then no client certificate or key wil be produced. The FilePath is some location on the computer to where the generated certificates will be placed. For example,

CAandClientCert.exe "C:\Temp, 192.168.0.16, "

CAandClientCert.exe " c:\Program Files (x86)\HomeSeer HS4\Data\mcsMQTT, MQTTBroker,HSRPi "

The example above is for a Windows use. CAandClientCert.exe is a .NET 5.0 application so should be able to be used on Linux with a prefix of "mono ". This evaluation on Linux has not beed performed.

What is significant in the names provided is that the same name needs to be used when trying to connect. For example, if HSRPi resolves to 192.168.1.100 and HSRPi is used in the certificate geneate name, then connectiong via MQTT needs to be to HSRPi and not 192.168.1.100.

Rather than using the automated process or CAandClientCert.exe tool, other 3rd party tools can be used to generate the certifiates and keys. The description of using OpenSSL (Windows or Linux) is provided in the following paragraphs.

10.2.1 SSL/TLS Certificate creation:

The first step in setting up SSL communications between the mcsMQTT plug-in and your Broker is to create the necessary SSL certificates. You'll typically need two. A Self signed CA certificate that you can use to sign your Broker "server" certificate, and the Broker certificate itself. If you decide to you also want Client certification, you'll need to create another certificate for each Device/Client as well.

10.2.2 Software/Tools:

CAandClientCert.exe utility is described above. A more interactive mechanism using openssl is shown below. Openssl can be install on either Windows or Linux.

This example uses openssl to create the certificates. If you're on a Windows platform you'll need to install the Openssl application on your platform:
https://slproweb.com/products/Win32OpenSSL.html

If you are on some variant of Linux, it will either be installed already or you can use the associated install app for the flavor of Linux you're using (apt-get, yum, etc).

There are other automated methods to create certificates as well:
https://github.com/owntracks/tools/blob/master/TLS/generate-CA.sh

10.2.3 Certificate Creation

10.2.3.1 Certificate Naming

In the following instructions, you can use any file naming convention you want for the names of the SSL certificates, but it's best to name them something that explains where/what they are used for (i.e. MQTT_Broker.xxx, MQTT_CA.xxx, MQTT_garage.xxx, etc).

10.2.3.2 Select the directories used to save the certificates

Since SSL certificates can be created on many different devices, they may not always end up on the device you're using them on. For this reason, it's a best practice to create all your SSL Certificates on one computer both for security as well as for backup rather than directly on the device (which isn't always possible for smaller devices). You'll need to use some of these Certificates again (particularly the CA certificate) for new devices and/or reconfigurations so it's convenient to have them all in one location. This directory can be located anywhere that's convenient to you (and hopefully backed up regularly). Linux typically stores their certificates at /etc/ssl/private for the key and /etc/ssl/certs for the certificates. On Windows OpenSSL is typically C:\OpenSSL-Win32\bin\PEM\

Since the SSL Certificates may be on another machine, they will typically need to be copied to their proper storage location on the device that is using them. This location is usually device dependent and is almost always located on the device that's using the certificates (eg Broker, Device). If the vendor/application doesn't define a location, you'll need to determine where you want to store them. For the Mosquitto Broker, on Linux there's usually a directory for the mosquito broker already created for you at /etc/mosquito. On windows it's typically located at, C:\Program Files (x86)\mosquitto.

In this example, while I'll use standard directories, it's perfectly fine to use any file structure if you understand the openssl process. I'll also assume that the user is creating their SSL certificates on one machine and has created/confirmed appropriate storage directories are on that machine. For example, on Linux *keyDir/* would be /etc/ssl/private and *certDir/* would be /etc/ssl/certs. On Windows, since openSSL usually stores both the key and certificate in the same directory both *keyDir/* and *certDir/* would be the same: C:\openSSL-Win32\PEM.

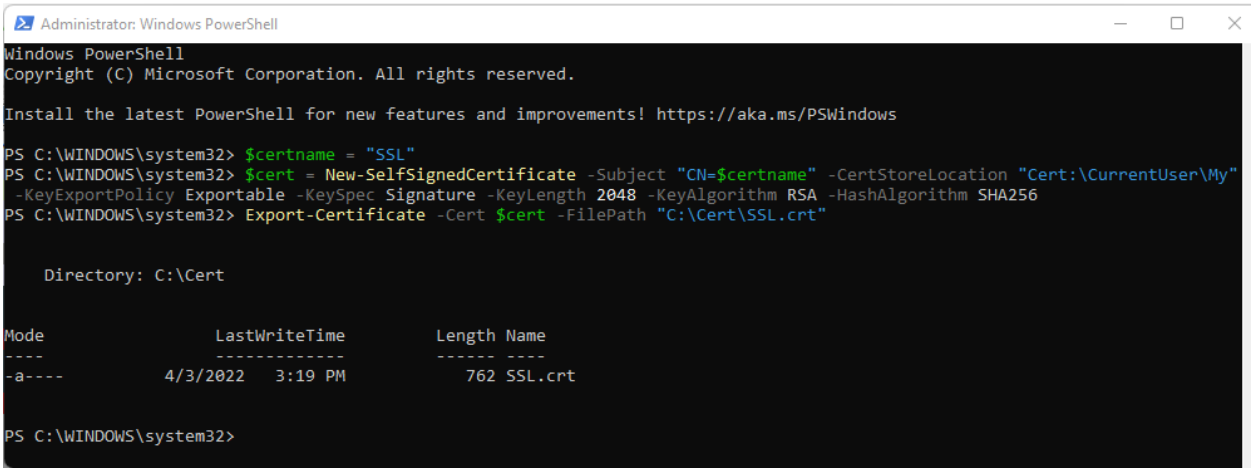
10.2.3.3 *Create the Key for the CA (Certificate Authority) certificate:*

From Windows elevated powershell the following can be done. Otherwise use the openssl process that follows. In the Powershell example the certificate name is SSL. Change is to the appropriate naming convention here and in subsequent certificate creation steps if Windows Powershell is being used.

PS C:\WINDOWS\system32> $certname = "SSL"

PS C:\WINDOWS\system32> $cert = New-SelfSignedCertificate -Subject "CN=$certname" -CertStoreLocation "Cert:\CurrentUser\My" -KeyExportPolicy Exportable -KeySpec Signature -KeyLength 2048 -KeyAlgorithm RSA -HashAlgorithm SHA256

PS C:\WINDOWS\system32> Export-Certificate -Cert $cert -FilePath "C:\Cert\SSL.crt"



openssl genrsa -des3 -out keyDir/MQTT_CA.key 2048

This creates your IOT networks CA certificate Key which is used to create the final certificate. The "-des3" requires you to provide a PassPhrase to protect the key and is optional. Remove it from the command line if you don't want to have to remember it (You'll need to use it when you create the new Broker or Client certificates).

10.2.3.4 Create the CSR file from the previous CA Key for creating the final CA certificate:

openssl req -new -key keyDir/MQTT_CA.key -sha256 -out certDir/MQTT_CA.csr

-sha512 is acceptable if the latest SSL libraries are installed.

This command requires you to input several variables. Most are informational and/or not used for a self-signed cert. The Common Name is the only one you need to consider.
Country Name (2 letter Code) [xx]: i.e. US
State or Province Name (full name) [Berkshire]: i.e. Washington
Locality Name (e.g., city) [Newbury]: i.e. Seattle
Organization Name (e.g., company) [My Company Ltd]: i.e. MQTT Ltd
Organizational Unit Name (e.g., section) []: i.e. IT (Optional)
***Common Name** (e.g., your name or your server's hostname) []: i.e. IOTnetwork
This should be something like local host, your name or network name. It should not be the same as what you use on the broker certificate later.
Email Address []: i.e. not required. You can leave blank
Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []: leave blank
An optional company name []: leave blank

10.2.3.5 Create the final CA certificate:

openssl x509 -req -days 3650 -in certDir/MQTT_CA.csr -signkey keyDir/MQTT_CA.key -out certDir/MQTT_CA.crt

If you entered a passphrase for your CA key, you'll need to enter it when requested. This certificate will be good for 10 yrs (3650). You can select a smaller period if you like.

10.2.3.6 Create Broker certificate Key (NO passphrase)

openssl genrsa -out keyDir/MQTT_Broker.key 2048

10.2.3.7 X509 Create Broker certificate request:

openssl req -new -out certDir/MQTT_Broker.csr -key keyDir/MQTT_Broker.key

Again, you'll be asked the following questions. None of the entries are all that important EXCEPT the Common Name (CN)
Country Name (2 letter Code) [xx]: i.e. US
State or Province Name (full name) [Berkshire]: i.e. Washington
Locality Name (e.g., city) [Newbury]: i.e. Seattle
Organization Name (e.g., company) [My Company Ltd]: i.e. MQTT Ltd

Organizational Unit Name (e.g., section) []: i.e. IT
* Common Name (e.g., your name or your server's hostname) []: i.e. IOTmachine
This needs to be the Name of the PC running your Broker. It's the name that's attached to the IP address of the machine, i.e. MyPC, MQTTBroker.local or just the IP address.
Email Address []: i.e. not required. You can leave blank
Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []: leave blank
An optional company name []: leave blank

10.2.3.8 Create Broker certificate

openssl x509 -req -days 3650 -in certDir/MQTT_Broker.csr -CA certDir/MQTT_CA.crt -CAkey keyDir/MQTT_CA.key -CAcreateserial -out certDir/MQTT_Broker.crt

If you entered a passphrase for your CA key, you'll need to enter it when requested.

10.2.3.9 Create Client certificate Key (If needed. No passphrase unless client device requires it (MQTTspy, mcsMQTT plug-in)

openssl genrsa -out keyDir/MQTT_nameOfClient.key 2048

10.2.3.10 X509 Create Client certificate request:

openssl req -new -out certDir/MQTT_nameOfClient.csr -key keyDir/MQTT_nameOfClient.key

Again, you'll be asked the following questions. None of the entries are all that important EXCEPT the Common Name (CN)
Country Name (2 letter Code) [xx]: i.e. US
State or Province Name (full name) [Berkshire]: i.e. Washington
Locality Name (e.g., city) [Newbury]: i.e. Seattle
Organization Name (e.g., company) [My Company Ltd]: i.e. MQTT Ltd
Organizational Unit Name (e.g., section) []: i.e. IT
* Common Name (e.g., your name or your server's hostname) []: i.e. IOTmachine or IP address
This needs to be the Name of the device or PC the client app is running on. It's the name that's attached to the IP address of the machine/device, i.e. MyPC, garageControl, or just the IP address.
Email Address []: i.e. not required. You can leave blank
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: leave blank
An optional company name []: leave blank

10.2.3.11 Create Client certificate:

openssl x509 -req -days 3650 -in certDir/MQTT_ nameOfClient.csr -CA certDir/MQTT_CA.crt -CAkey keyDir/MQTT_CA.key -CAcreateserial -out certDir/MQTT_ nameOfClient.crt

If you entered a passphrase for your CA key, you'll need to enter it when requested.

10.2.3.12 Copy the SSL certificates to the correct location (if necessary)

If the SSL certificates were created on a different machine than the Broker, you'll need to copy them from that machine to the Broker machine.
Copy the CA (MQTT_CA.crt), Broker Key (MQTT_Broker.key) and SSL certificates (MQTT_Broker.crt) to the Broker:
For Mosquitto on Linux it's typically: /etc/mosquito/ssl/private for the Broker Key and /etc/mosquito/ssl/certs for the CA and Broker certificates.
For Mosquitto on Windows typically they all go to the same directory: C:\Program Files (x86)\mosquitto\PEM\

10.2.3.13 Create Client composite Certificate

openssl pkcs12 -export -out certDir/MQTT_nameOfClient.pfx -inkey keyDir/MQTT_nameOfClient.key -in certDir/MQTT_nameOfClient.crt -certfile certDir/MQTT_CA.crt

10.2.3.14 Copy certificates to Client Devices

For devices/clients you'll need to copy the CA (MQTT_CA.crt) to the individual devices appropriate location. If you run with Client Authorization and depending on the type of device, you'll either need to copy the Client (MQTT_name OfClient.crt) and Key (MQTT_nameOfClient.key) or paragraph 10.2.3.13 MQTT_nameOfClient.pfx to the devices appropriate location as well. The mcsMQTT plug-in requires the PFX file from paragraph 10.2.3.13.

10.2.3.15 Summary for ease of copy/past

openssl genrsa -out C:\Cert\MQTT_CA.key 2048
openssl req -new -key C:\Cert\MQTT_CA.key -sha256 -out C:\Cert\MQTT_CA.csr
openssl x509 -req -days 3650 -in C:\Cert\MQTT_CA.csr -signkey C:\Cert\MQTT_CA.key -out C:\Cert\MQTT_CA.crt
openssl genrsa -out C:\Cert\MQTT_Broker.key 2048
openssl req -new -out C:\Cert\MQTT_Broker.csr -key C:\Cert\MQTT_Broker.key

```
openssl x509 -req -days 3650 -in C:\Cert\MQTT_Broker.csr -CA C:\Cert\MQTT_CA.crt -
CAkey C:\Cert\MQTT_CA.key -CAcreateserial -out C:\Cert\MQTT_Broker.crt

openssl genrsa -out C:\Cert\MQTT_Client.key 2048
openssl req -new -out C:\Cert\MQTT_Client.csr -key C:\Cert\MQTT_Client.key
openssl x509 -req -days 3650 -in C:\Cert\MQTT_Client.csr -CA C:\Cert\MQTT_CA.crt -
CAkey C:\Cert\MQTT_CA.key -CAcreateserial -out C:\Cert\MQTT_Client.crt
openssl pkcs12 -export -out C:\Cert\MQTT_Client.pfx -inkey C:\Cert\MQTT_Client.key -
in C:\Cert\MQTT_Client.crt -certfile C:\Cert\MQTT_CA.crt
```

10.3 Mosquitto Broker configuration for SSL

This is for a Linux based broker. The configuration file is the same for Windows based broker

To add SSL/TLS support to your Mosquitto broker, open the mosquitto.conf file with a text editor. If you are already running non-secure using port 1883 (or similar), they you'll need to decide if you want to add another listening port (so you can listen on both secure and non-secure ports) or just listen only on the new secure port. I'd recommend you try adding a second port initially and then remove the non-secure one later if appropriate. Assuming you have your primary (1883) already running,

1. Search in the moquitto.conf file for "Extra Listeners". It may not be present depending upon the install version of Mosquitto. It is possible to have both insecure communications on port 1883 and secure communications on port 8883. The "listener" line can be added multiple times to address specific needs.
2. Remove "#" from "#listener" to enable this listener or just add a section starting with "listener"
3. Scroll down (Be sure you're below the Extra Listener section (or added listener) as all of these entries exist for the normal listener at the top of the Configuration file as well) until you find the line with #cafile. You'll need to add the location of the MQTT_CA.crt file. Be sure to remove the Comment (#) marker.
   For Linux it would be something like:
   cafile /etc/mosquitto/ssl/MQTT_CA.crt (this is completely dependent on
   For Windows:
   cafile C:\OpenSSL-W32\bin\PEM\MQTT_CA.crt
4. Scroll down to #certfile and add the location of the MQTT_Broker.crt file such as:
   certfile /etc/mosquitto/ssl/certs/MQTT_Broker.crt
   certfile C:\OpenSSL-W32\bin\PEM\MQTT_Broker.crt
   There is no specific path locations. Just be consistent to not confuse yourself.

5. Scroll down to #keyfile and add the location of the MQTT_Broker.key file
   keyfile /etc/mosquitto/ssl/private/MQTT_Broker.key
   keyfile C:\OpenSSL-W32\bin\PEM\MQTT_Broker.key

6. Restart the mosquitto broker to accept the changes. You should now be able to access the broker using SSL/TLS communications.

Version 1 and 2 of Mosquitto have different defaults so the transition from 1 to 2 is a breaking change for most. Another observation is that with version 2, I have only had success with encrypted communications when both a username/password for Mosquitto and the certificates

are setup. In version 1, I did not have a need to provide username and password. The mosquito.conf file that has worked for me for insecure communications on 1883 and secure communications on 8883 is shown below. The **mosquitto_passwd** utility that comes with the **client tools** when using Steve's guide at [How to Install The Mosquitto MQTT Broker- Windows and Linux (steves-internet-guide.com)](#) is one reference to setup username/password security with Mosquitto.

The example below uses the auto-generated files for a MQTT Broker IP at 192.168.0.16 that has been specified as 192.168.0.16 in the mcsMQTT or utility tool and had been copied from the HS computer to the Linux MQTT Broker on 192.168.0.16. Note that .NET Framework, which HS uses, does not support TLS1.3 while Mosquitto does.

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example
per_listener_settings true
pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

listener 1883
allow_anonymous true

listener 8883
require_certificate false
password_file /etc/mosquitto/passwd
certfile /etc/mosquitto/certs/192-168-0-16.crt
keyfile /etc/mosquitto/certs/192-168-0-16.key
cafile /etc/mosquitto/ca_certificates/ca192-168-0-16.crt

tls_version tlsv1.2

include_dir /etc/mosquitto/conf.d
```

Here are also some links to "prettier" versions of some of the instructions by others:

Mosquitto Broker Configurations:
http://www.steves-internet-guide.com/mossquitto-conf-file/

SSL/TLS certificate generation:
https://mcuoneclipse.com/2017/04/14/enable-secure-communication-with-tls-and-the-mosquitto-brokerLocal

Not all IP-interfaced devices use MQTT protocol. A selective, and growing, set of devices have been interfaced to HS via mcsMQTT that use TCP as the control/status protocol. mcsMQTT handles the device-level API and presents to the user a pseudo MQTT presentation.

11 Local

11.1 IP Relay

The IP Relay tab provides setup for an eight-channel relay and eight channel opto-isolated input interfaced via TCP connection. The hardware is often referred to as Web-Relay or Relay-Net.

An 8 Channel Relay and Input device which is available from many vendors such as https://www.ebay.com/itm/173093192868?ul_noapp=true has been interfaced. It is shown in Figure 60. The default login is user admin and password 12345678 for IP 192.168.1.166 to configure the device. From a browser with URL at this IP the System Setttings can be modified to change the IP, if desired. Another setting exists to bind the input to the relay. This will be changed to not bind unless the inputs are going to be used to also directly control the relay.

Versions 5.7 and 5.8 of the board are available. 5.7 from the link above and 5.8 from 8 Channel Relay Network IP Relay Web Relay Dual Control Ethernet RJ45 Relay | eBay (Aug 9 delivery). The V5.7 has known issue of network connectiveiy through routers and switches that support 1G bandwidth. The same issue may or may not exit with the V5.8. The version is etched into the edge of the board and the V5.8 is dated 2019. The input connector block for the V5.8 has 3 pin block for RS-485. The V5.7 in the same position has labels of COM 5V GND as well as a second COM between the 8 inputs.

My experience with multiple suppliers on Ebay that provide USA stock are not able to assure shipment of V5.7 or V5.8 boards. It seems both versions are sold under the same product number with the "bin" having both versions. **I find the Dingtian IOT relay described in Section 24 to be a much better product, at a lower cost ($15 for 8 channels of input and output) and a DIN-rail case available at just a few dollars more.**

Documentation, focused on the V5.8, is available at http://mcsSprinklers.com/IOZone IP Relay.zip

Figure 60 8 Channel Relay / 8 Channel Digital Input IP Network Module

Setup for this device is contained on the Local page that is available from the HS Menu for Plug-in mcsMQTT. From this page a table is presented where the module's IP address and TCP port are entered

such as shown in Figure 61. The polling interval entry is used if any of the module's digital inputs are being used. If not used then the polling interval text box should be left blank. The input for polling is in milliseconds, but reasonable values of 10000 vs. unreasonable 100 should be selected when the inputs are being interfaced.

Two protocols are supported. HTTP is selected for Port 80. Both V5.7 and V5.8 use the same HTTP protocol. TCP is selected for Port 1234. TCP for V5.7 has been implemented.

Multiple modules are accommodated with a row provided for each module.

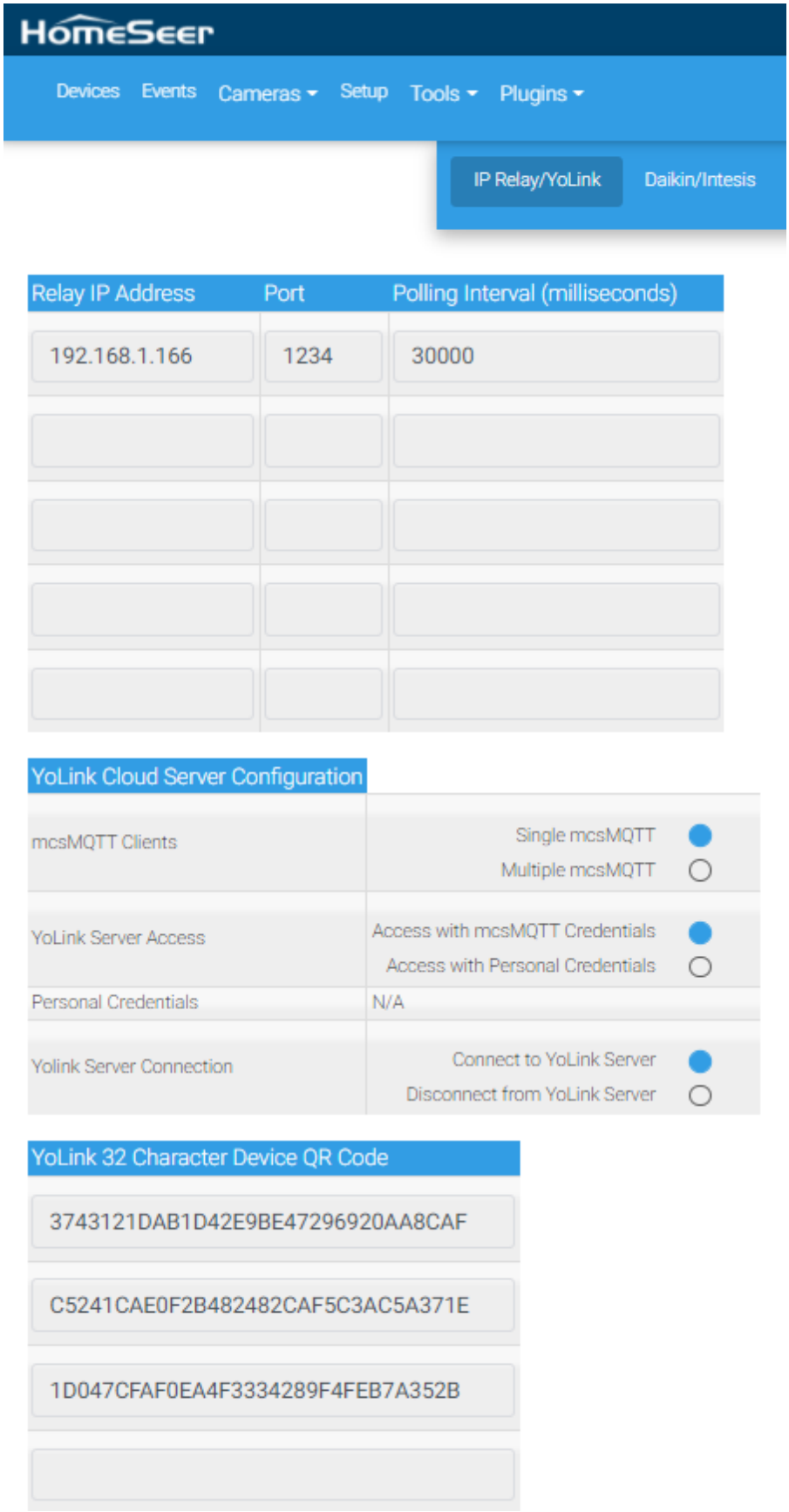

HomeSeer
Devices
Events
Cameras
Setup
Tools
Plugins
IP Relay/YoLink
Daikin/Intesis
Relay IP Address
Port
Polling Interval (milliseconds)
192.168.1.166
1234
30000
YoLink Cloud Server Configuration
mcsMQTT Clients
Single mcsMQTT
Multiple mcsMQTT
YoLink Server Access
Access with mcsMQTT Credentials
Access with Personal Credentials
Personal Credentials
N/A
Yolink Server Connection
Connect to YoLink Server
Disconnect from YoLink Server
YoLink 32 Character Device QR Code
3743121DAB1D42E9BE47296920AA8CAF
C5241CAE0F2B482482CAF5C3AC5A371E
1D047CFAF0EA4F3334289F4FEB7A352B

Figure 61 Local Page Setup for 8 Channel Relay/Input and YoLlink Devices

After the module has been identified a set of pseudo-Topics are created and visible on the MQTT Setup Page, Association Tab. This table is shown in Figure 62. Note the page has been filtered in the T1 pulldown to show only the 192.168.0.166 pseudo-Topic items.

Filter Association Table by Mqtt Topic and JSON Payload Key Clear Filters Rebuild Filters

T1	T2	T3	T4	T5	T6
192.168.0.166					

J1	J2	J3	J4	J5	J6

Show Selected Associations

Prev | 0 of 16 | Next

Association Table for Auto Association of MQTT Topic and HS Device

Λ	e	a	ref	topic	payload	h	d	lastdate
0	☐	☐		Sub: 192.168.0.166/INPUT1	Off	☐		2019-12-21 19:27:25
1	☐	☐		Sub: 192.168.0.166/INPUT2	Off	☐		2019-12-21 19:27:25
2	☐	☐		Sub: 192.168.0.166/INPUT3	Off	☐		2019-12-21 19:27:25
3	☐	☐		Sub: 192.168.0.166/INPUT4	Off	☐		2019-12-21 19:27:26
4	☐	☐		Sub: 192.168.0.166/INPUT5	Off	☐		2019-12-21 19:27:26
5	☐	☐		Sub: 192.168.0.166/INPUT6	Off	☐		2019-12-21 19:27:26
6	☐	☐		Sub: 192.168.0.166/INPUT7	Off	☐		2019-12-21 19:27:26
7	☐	☑	694	Dev: |192.168.0.166|INPUT8 Sub: 192.168.0.166/INPUT8 Pub: the following Topic on Device command	Off	☐	☐	2019-12-21 19:27:26
8	☐	☑	693	Dev: |192.168.0.166|RELAY1 Sub: 192.168.0.166/RELAY1 Pub: the following Topic on Device command 192.168.0.166/RELAY1	On	☐	☐	2019-12-21 19:27:24
9	☐	☑	695	Dev: |192.168.0.166|RELAY2 Sub: 192.168.0.166/RELAY2 Pub: the following Topic on Device command 192.168.0.166/RELAY2	Off	☐	☐	2019-12-21 19:27:24
10	☐	☐		Sub: 192.168.0.166/RELAY3	Off	☐		2019-12-21 19:27:24
11	☐	☐		Sub: 192.168.0.166/RELAY4	Off	☐		2019-12-21 19:27:25
12	☐	☐		Sub: 192.168.0.166/RELAY5	Off	☐		2019-12-21 19:27:25
13	☐	☐		Sub: 192.168.0.166/RELAY6	Off	☐		2019-12-21 19:27:25
14	☐	☐		Sub: 192.168.0.166/RELAY7	Off	☐		2019-12-21 19:27:25
15	☐	☐		Sub: 192.168.0.166/RELAY8	Off	☐		2019-12-21 19:27:25

Figure 62 Local Page Psuedo-Topic for 8 Channel Relay/Input Module

The device is as a Topic by it's IP address. Its Relay and Input points are identified by RELAY1..8 and INPUT1..8. When the "A"ssociate column checkbox is checked then HS device is created. In Figure 62 these are 693, 694 and 695 where 694 is the input and the others are relays.

The HS devices contain buttons for relays and status only for the inputs such is shown in Figure 63 for HS4. Similar functionality is visible in HS3 from the Device Managment page.

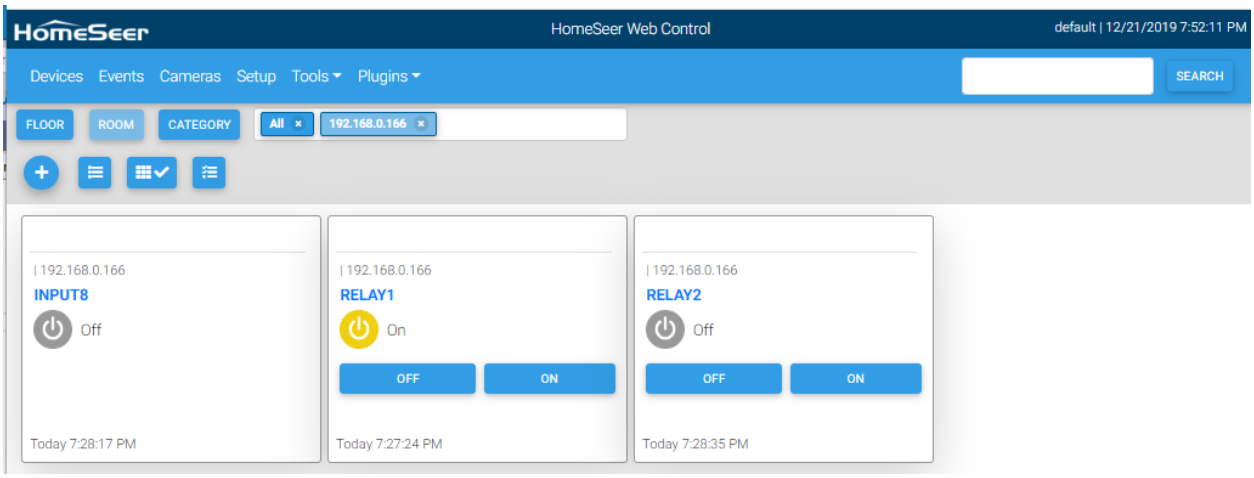


Figure 63 HS4 Devices View for Relay/Input Module

All features of mcsMQTT and HS are available to the created devices and the pseudo-Topics such as History, Charting, and Events etc.

11.2 Local HVAC

Venstar, AirTouch, Midea, Daikin or Intesis Gateway provides a means to obtain status of a large number of parameters and a means to control the primary functionality of the units with a local IP connection. The IP address of any unit that is to be interfaced can be entered on the HVAC tab of the Local page as shown in Figure 64.

Most of the thermostat parameters will automatically be associated with HS Device and Features upon detection of data being received from the thermostat such as shown in Figure 65.

The Association tab will also provide the values of many other parameters. These parameters can be selected with the "A"ssociate checkbox on the Association tab of the MQTT page (See Figure 66) to have HS devices created and then the device updated on each polling interval.

The ":Power" device is "A"ssociated by default. It is used for the On/Off control of the unit. The Homeseer API for a thermostat has the Off control as part of the ":Mode" device and selection of any mode will result in an implicit On power state. To use the Homeseer Thermostat API the "A"ssociate checkbox should be unchecked on the Association Tab. This will remove the ":Power" device from HS. To explicitly control On/Off with a separate device then it should be checked.

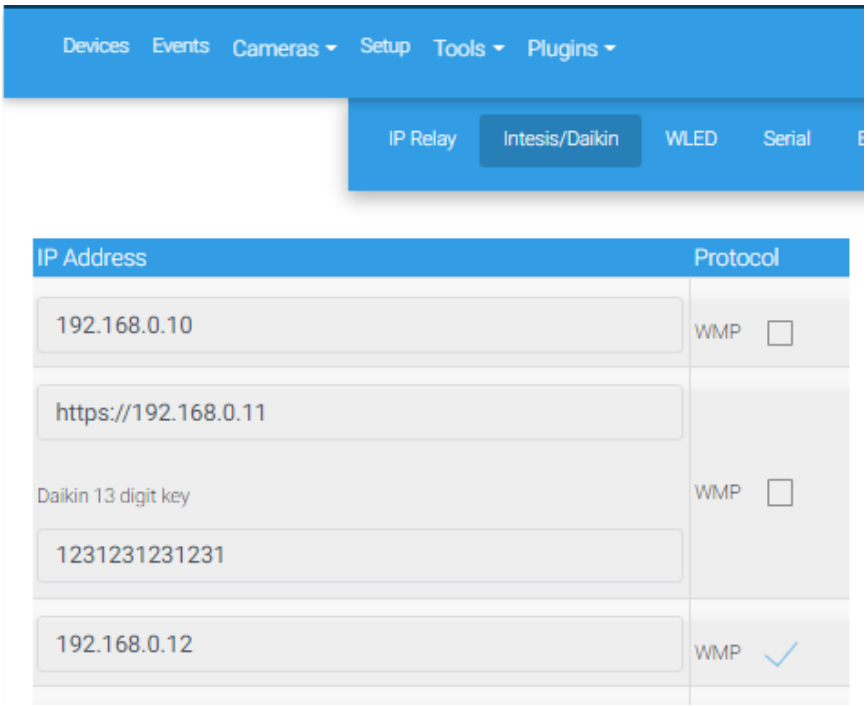


Figure 64 Daikin/Intesis/Venstar Unit IP Address Entry

Display Filters: Floor | Room | Device Type | Show All

	Ref	Status	Category	Floor	Room	Name	Last Change	Control
☐	3038			Daikin	192.168.0.127	Daikin-192.168.0.127	Today 3:09:10 PM	
☐	3039	Heat		Daikin	192.168.0.127	Daikin@192.168.0.127-Mode	Today 3:09:10 PM	Auto Dry Cool Heat Fan
☐	3040	Off		Daikin	192.168.0.127	Daikin@192.168.0.127-Power	Today 3:09:10 PM	Off On
☐	3041	12		Daikin	192.168.0.127	Daikin@192.168.0.127 - TempSetpoint	Today 3:09:25 PM	(value) 12 Submit
☐	3042	0		Daikin	192.168.0.127	Daikin@192.168.0.127 - HumSetpoint	Today 3:09:26 PM	(value) 0 Submit
☐	3043	Auto		Daikin	192.168.0.127	Daikin@192.168.0.127-Fan	Today 3:09:26 PM	Auto F1 F2 F3 F4 F5
☐	3044	None		Daikin	192.168.0.127	Daikin@192.168.0.127 - FanDirection	Today 3:09:26 PM	None Vertical Horizontal HorAndVert
☐	3045	0		Daikin	192.168.0.127	Daikin@192.168.0.127 - OutsideTemp	Today 3:09:26 PM	
☐	3046	0		Daikin	192.168.0.127	Daikin@192.168.0.127 - InsideTemp	Today 3:09:26 PM	

Figure 65 Default Daikin/Intesis HS Devices

Filter Association Table by Mqtt Topic and JSON Payload Key Clear Filters Rebuild Filters

T1 Daikin	T2	T3	T4	T5	T6
J1	J2	J3	J4	J5	J6

Show Selected Associations

Prev 0 of 45 Next

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	d	lastdate
0	☐	☐	☐	☑	3038	Daikin/192.168.0.127				
1	☐	☐	☐	☐		Sub: Daikin/192.168.0.127: dfd4	0	☐		2020-02-27 15:24:55
2	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:adv		☐		2020-02-27 15:24:55
3	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:alert	255	☐		2020-02-27 15:24:55
4	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:b_f_dir	0	☐		2020-02-27 15:24:55
5	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:b_f_rate	A	☐		2020-02-27 15:24:55
6	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:b_mode	4	☐		2020-02-27 15:24:55
7	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:b_shum	0	☐		2020-02-27 15:24:55
8	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:b_stemp	12.0	☐		2020-02-27 15:24:55
9	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfd1	0	☐		2020-02-27 15:24:55
10	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfd2	0	☐		2020-02-27 15:24:55
11	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfd3	0	☐		2020-02-27 15:24:55
12	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfd5	0	☐		2020-02-27 15:24:55
13	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfd6	0	☐		2020-02-27 15:24:55
14	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfd7	0	☐		2020-02-27 15:24:55
15	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfdh	0	☐		2020-02-27 15:24:55
16	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfr 3	5	☐		2020-02-27 15:24:55
17	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfr1	5	☐		2020-02-27 15:24:55
18	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfr2	5	☐		2020-02-27 15:24:55
19	☐	☐	☐	☐		Sub: Daikin/192.168.0.127:dfr4	A	☐		2020-02-27 15:24:55

Figure 66 Daikin Additional Parameters

11.2.1 Intesis

The Intesis interface uses WMP protocol. It has the capability to interface Daikin as well as many other air conditioning units. When selecting Intesis/WMP the IP of the Intesis interface is specified as well as the WMP checkbox.

The entered IP address will be polled at 15 second intervals to get status updates for units interfaced via Intesis.

11.2.2 Daikin

Original Daikin models with Wifi are specified with the IP and without the WMP checkbox selected. For later model Daikin BRP072C42 and likely other most recently introduced units add additional security to the communications. To specify this later model use “https://” prefix in the IP as shown in example of Figure 64. This consists of using SSL (https) and including a https header for key “X-Daikin-uuid”. The

key's value is generated by mcsMQTT at the time the user enters the Daikin's 13-digit key from the sticker on the unit. mcsMQTT will register the header with the unit thus allowing only communications only with mcsMQTT. The reverse engineering of this security change is described at https://github.com/ael-code/daikin-control/issues/27.

The entered IP address will be polled at 15 second intervals to get status updates for Daikin and Intesis. The Venstar polling entry (in milliseconds) allows the user to poll at a desired rate or set to 0 to disconnect from the thermostats. Upon the initial status update a set of HS devices will be created as shown in Figure 65.

11.2.3 Venstar

Venstar integration setup is found on the Local Page, HVAC tab. At statup or upon use of the Discover button all Venstar thermostats be found and HS Device and Features setup. SSDP is used for the discovery.

By default, the pollilng rate is 0 so it needs to be changed before any data will be returned from the thermostat.

If a thermostat is not found or if the type is not correctly identified then they can be explicitly specified as shown in Figure 67.

In the Venstar case there are four endpoints available. The /Info, /Sensor, and /Alerts returned data will result in HS Device & Features being automatically created. The /Runtime endpoint will populate the Association Table and if desired to be be viewed as HS Features then the “a” checkbox on the Association Table is used.

Venstar control from HS Devices, Events or Script is managed to assure the requested value conforms to the Venstar constraints. These constraints consist of a minimum delta for setpoints and the relationship between heat and cool setpoints when in Auto mode.

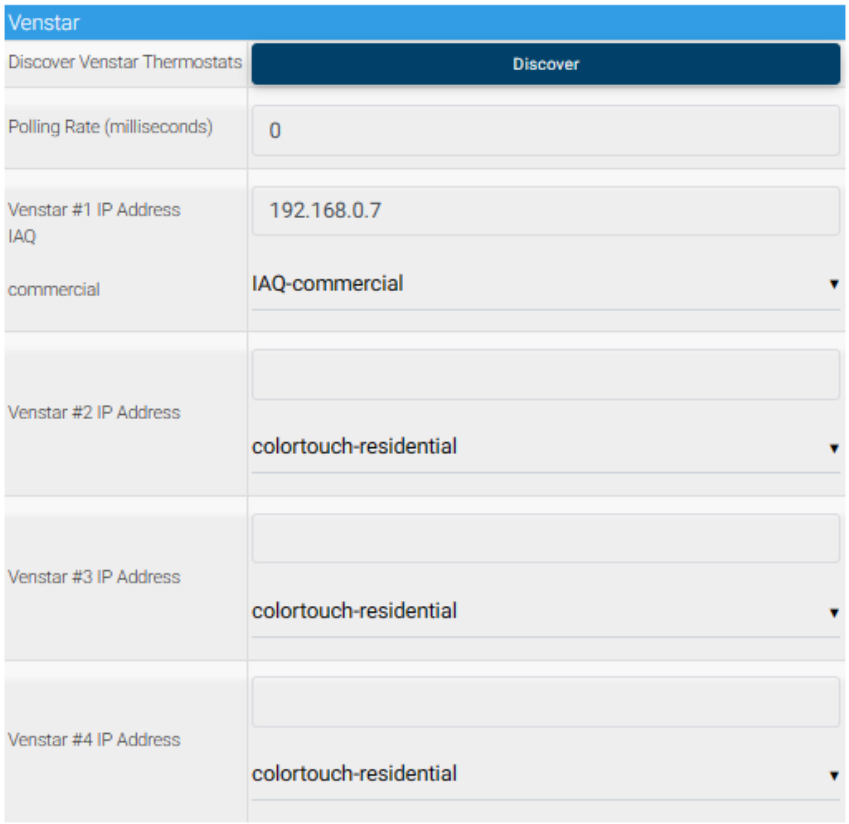


Figure 67 Venstar Integration Setup and HS Devices

11.2.4 Midea

Midea provides several appliances that have local as well as internet control. mcsMQTT implementation is based upon local control. It uses the library provided at GitHub - nbogojevic/midea-beautiful-air: Python client for accessing Midea air conditioners and dehumidifiers (Midea, Comfee, Inventor EVO) via local network. The link identifies the models of Air Conditioner and Dehumidifer that are know to be supported using the V2 and V3 protocols.

The plugin has only been tested with the Air Conditioner, but the hooks are in place should somebody have a Dehumidifier and desires to support the integration. The current implementation is fully functional, but the library seems to not support fan speed and cound control, but only provides status for these.

The library needs to be installed per the reference instructions wich are:

```
pip install --upgrade midea-beautiful-air
```

This assumes pip and Python are available on the computer where the plugin is installed.

The mcsMQTT setup is found on the Local Page, HVAC Tab of the plugin as shown in Figure 68. Required parameters are the Midea account credentials and the rate at which HS status will be refreshed.

The library is accessed with application midea-beautiful-air-cli.exe. It should be on the search path following its installation, but the path can be explicitly provided if the login account is different than the install account.

Discovery of all appliances will be attempted on the LAN where HS is installed. If the appliance is on a different LAN, then it's IP can be explicitly specified. Discovery will be attempted at startup or when the setup parameters are modified. It can also be explicity attempted with the Discover button.

Midea	
Discover Midea Thermostats	Discover
Polling Rate (milliseconds)	60000
Account Email	me@gmail.com
Account Password	•••••••••
Folder Path of midea-beautiful-air-cli.exe	C:\Users\mcs1\AppData\Local\Programs\Python\Python39\Scripts\
Thermostat IP	192.168.33.45

Midea | Laurie-downstairsAC

Midea-192.168.207.104 (9320)

192.168.207.104:running (9321)	Running	Today 10:07:42 AM	OFF RUNNING
192.168.207.104:mode (9322)	Cool	Today 10:07:42 AM	AUTO COOL DRY HEAT FAN
192.168.207.104:target (9323)	74°F	Today 10:07:42 AM	74°F
192.168.207.104:fan (9324)	Auto	Today 10:07:42 AM	AUTO
192.168.207.104:sleep (9325)	Normal	Today 10:07:42 AM	NORMAL SLEEP
192.168.207.104:eco (9326)	ECO	Today 10:07:42 AM	NORMAL ECO
192.168.207.104:purify (9327)	Normal	Today 10:07:42 AM	NORMAL PURIFY
192.168.207.104:indoor (9328)	72°F	Today 10:07:42 AM	
192.168.207.104:outdoor (9329)	82°F	Today 10:07:42 AM	

Figure 68 Midea Integration Setup and HS Devices

11.2.5 Polyaire AirTouch

The AirTouch 2 protocol V1.1 is a local control/status implementation using TCP port 9200 which looks to be a modernization of a serial interface. It provides the ability to get various statuses and control of power, mode, setpoint and fan. The interface is to a hub which is likely the serial to ethernet connection and supports up to for Air Conditioning units per hub.

mcsMQTT provides the interface to one or more hubs. The setup is contained on the Local Page, HVAC Tab as shown in Figure 69. It has provisions for user entry of the polling rate to refresh data and the IP where the AirTouch Hub is located on the local network.

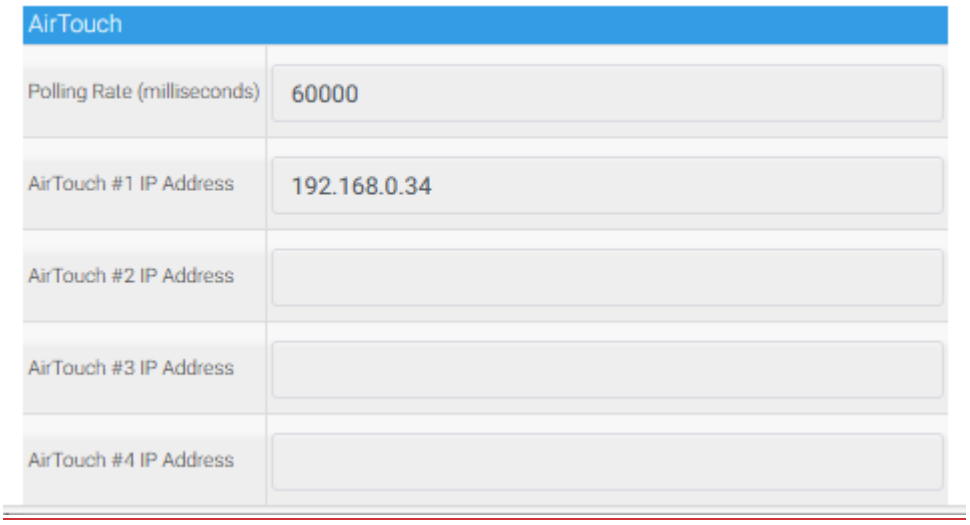


Figure 69 AirTouch Setup

Two queries are used. One is to get the names of the units. The other is to get the status. The name query is used at startup and continues to be used until names delivered. The status query is run at the polling rate and when data is returned it will create the HS Device and Features. A Device will exist for each AC Unit and Features will be created for its properties. Figure 70 provides a view of the HS UI.

Controls are buttons for state-type Features and selector for the setpoint. The unit operates using the Celcius scale and mcsMQTT converts to Farenheit if that is what has been setup in the HS regional settings. Celcius setpoints are specified to the nearest 0.5 degree. Farenheit ones are to the nearest degree.

With the serial communication orientation, a CRC is used. mcsMQTT generates the CRC for packets it sends. Provisions exists to validate the CRC for the data from the AirTouch hub, but have not been enabled pending integration testing.

Figure 70 AirTouch HS Device and Features

11.3 WLED

WLED is firmware typically loaded in ESP8266 and ESP32 that is used to provide a high degree of control for color LEDs. The project is described at https://github.com/Aircoookie/WLED. The main string(s) of LED support a MQTT protocol. A string that is divided into segments is controlled by REST/JSON protocol. JSON is also used for playlist. mcsMQTT uses both protocols to provide full WLED integration.

WLED MQTT Topics all start with "wled". mcsMQTT recognizes these topics and will create a primary set of HS device for control and status. Advanced controls are also available with WLED and these can be enabled by associating the Topic to HS device on the Association tab.

No setup is needed for WLED support for the primary strings. Segments and playlists use HTTP so an IP address and the size of the segment is needed so mcsMQTT can provide the communication interface with HS. This setup is on the Local page, WLED tab as shown in Figure 71.

In this case two WLED devices have been recognized by mcsMQTT based upon the MQTT traffic with the topic starting with wled/. One of these two has been identified to have only the primary string and no segments with an IP at 192.168.0.143.

If any of these are to be controlled as segments then those parameters should be entered. The IP needs to be set if either segments or playlists are going to be used. If no segments or playlists are used for a given WLED string then no additional setup for that string is needed.

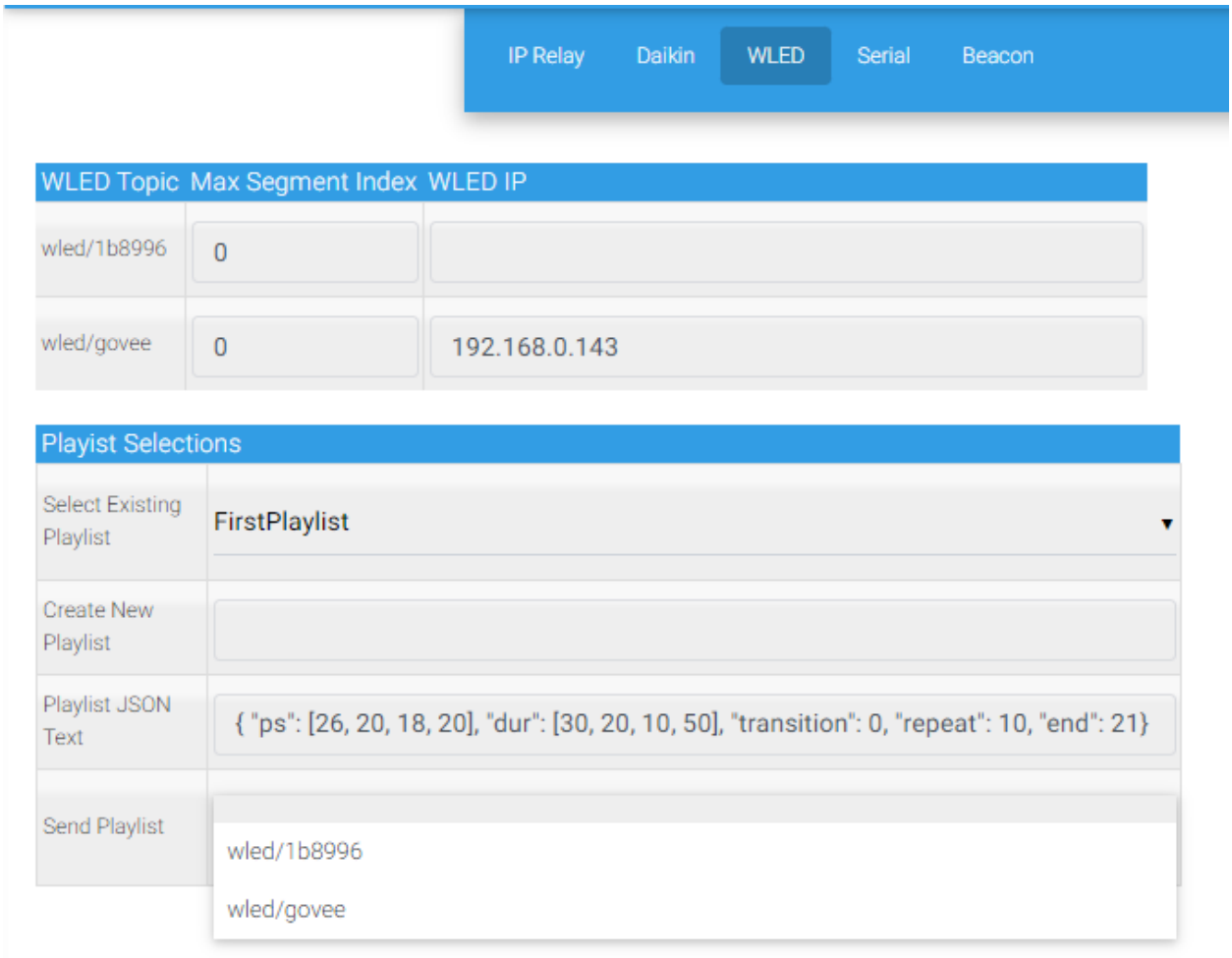


Figure 71 WLED Segment and Playlist Setup

A playlist has been defined called FirstPlaylist that contains the JSON format expected by the WLED playlist. If the JSON text is modified then the focused playlist will be updated. If a new playlist is to be setup then the Create New Playlist text box is used to give it a name.

The playlists are stored as files in the \data\mcsMQTT folder with a file type of ".wled". If the JSON text of a playlist is cleared then the playlist file will be removed.

To test the playlist the Send Playlist selector can be used to select the WLED unit to send what is showing in the Playlist JSON Text textbox.

The playlist can also be requested from the HS Device with the same Send Playlist control. Event actions to send the WLED playlist are also available.

The Device and Features setup for a WLED device, including the playlist device, is shown in Figure 72. When the playlist is commanded the HS playlist defice is updated based upon the command. There is no actual acknowledgement from the WLED device.

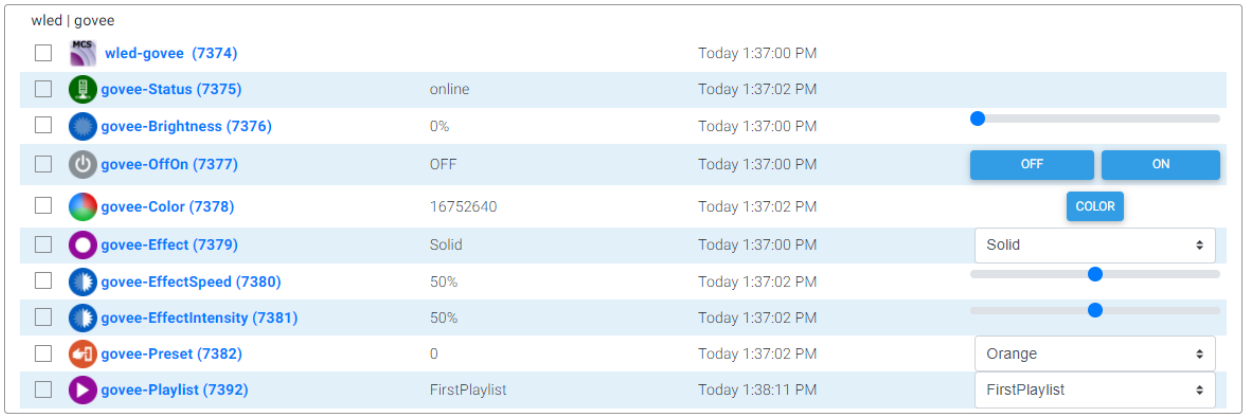


Figure 72 WLED Device and Features

The Preset device is created with a control of a Selector with values between 0 and 16. While this will work to set each of the potential 16 presets in WLED it will be desirable to provide names rather than numbers in the control. This is done from the Edit tab of the Preset Top using the VSP definitions. Figure 73 provides an example where the first four have been renamed to Unknown, Orange, Rainbow and Mood. WLED uses values starting at 1. It is likely the names given to the Preset in WLED will be transcribed here to avoid name confusion. If there are presets that are not setup in WLED then they can be removed from the VSP table so they do not show up in the HS Feature control.

Figure 73 Name WLED Presets

11.4 Serial (IP Serial and COM Serial)

Serial communication is normally provided via a bidirectional set of wires using a Universal Asychronous Receiver Transmitter (UART) that is contained within the computer or provided by a USB to Serial adapter. The technology to perform serial communication over Ethernet provides a similar capability but is not as ubiquitous as the UART. mcsMQTT supports both of these mechanisms.

The setup is done on the Local page, Serial tab as shown in Figure 74. In this example the first row identifies a serial connection to a Lantronix IP/Serial device. The IP/Serial device baud rate needs to be setup manually as no provisions exists to communicate meta data over the Ethernet link.

The second row identifies port 1 at 4800 baud. The other serial parameters are set to No Parity, eight bits, one stop bit (N,8,1). Port 1 will be COM1 for Windows or /dev/ttyUSB1 for Linux. In the Linux case mcsMQTT will search for other /dev/ttyUSB ports if USB1 is not available. Linux switches ports, especially when error conditions are detected so mcsMQTT tries to be tolerant of this. If multiple serial ports are in use then this could be problematic on Linux.

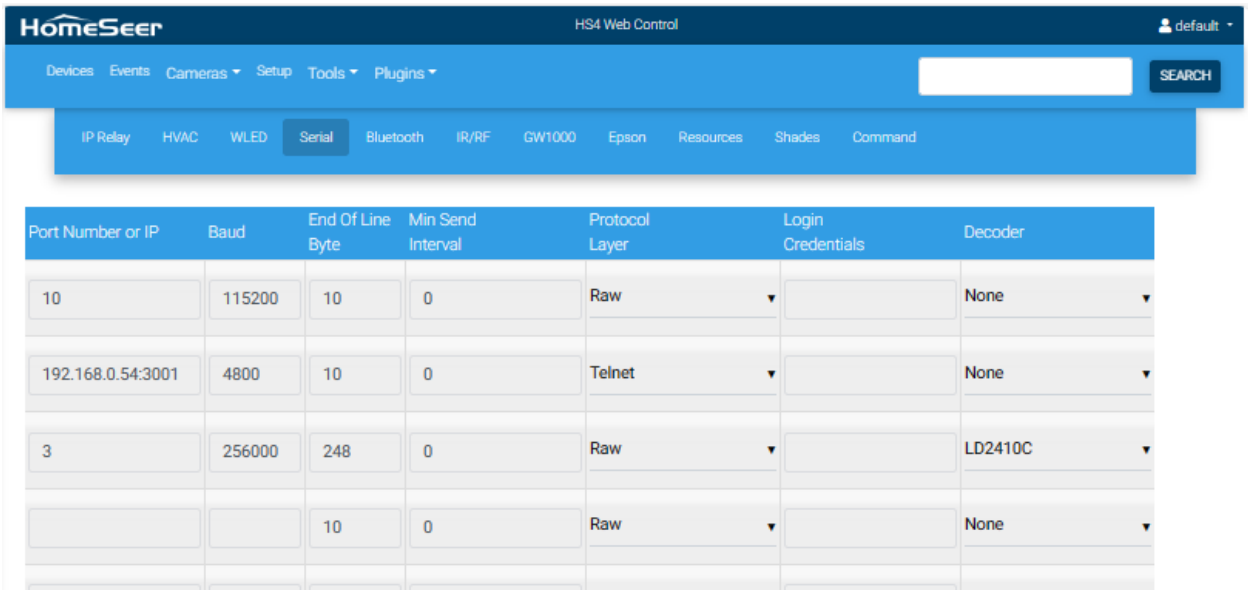


Figure 74 Serial Communications Setup

The serial data received is treated as a MQTT payload. The payload is the data that is received between End Of Line characters. This defaults to the Line Feed (LF)/Chr(10) byte. It can be change in the setup for each serial port. If this field is left blank then a message end is defined as three second pause in the reception of serial data.

Provisions have also been made on the transmit side to limit the rate of message transmission. This is not normally needed, but if a limited link budget exists then it could be useful.

Data is transmitted via the DeviceString of the HS device that was created for each serial port such as shown in Figure 75. Three controls are provided. The Submit button will send the DeviceString to the serial port. The Close and Open buttons will close and open the serial port. If strings are going to be sent through other mechanisms the CAPI is used to interface with mcsMQTT.

☐	649		Serial		Serial	Serial	MQTT_Receive	Yesterday 8:47:14 PM	
☐	650	":"Mail","P":{"door":0,"link":10}y	Serial	192.168.0.49:3001	192.168.0.49:3001	Serial/192.168.0.49:3001	MQTT_Receive	Today 1:34:54 PM	":"Mail","P":{"door":0,"linl Submit Close Open

Figure 75 Serial Port HS Device

Serial messages are managed as MQTT messages with the topic “Serial/xxx” where xxx is the port address that was setup. Other features of mcsMQTT such as JSON decoding, History retention, Charting, etc are available for the serial messages.

In addition, decoding of known byte streams is available. The decoded data will be presented in JSON format and available in the Association Table for subsequent use.

In the case of the HEX decoder, the received bytes are expanded into an ASCII hex format (e.g. byte 31 is converted to bytes with string representation of “1F “). With HEX decoding, the data being sent is also encoded into bytes from a ASCII hex input. (e.g. “1F 20” is sent as two bytes 31 and 32).

The Jacuzzi decoder will also expand the serial bit pattern into a JSON set of messages. These appear in the Association Table on the MQTT Page. If Auto-Create is enabled on the MQTT Page, Client Tab then the HS Device and Features will be automatically created such as shown in Figure 76.

Jacuzzi | Jacuzzi

Jacuzzi - 127.0.0.1_9200 (231)

Jacuzzi:StatusUpdate:SetTemperature (232)	102	Today 11:53:09 AM	102
Jacuzzi:StatusUpdate:HeatMode (233)	Auto	Today 11:53:07 AM	NEXT
Jacuzzi:StatusUpdate:Pump1 (234)	Off	Today 11:53:08 AM	TOGGLE
Jacuzzi:StatusUpdate:Pump2 (235)	Off	Today 11:53:08 AM	TOGGLE
Jacuzzi:StatusUpdate:Pump3 (236)	Off	Today 11:53:08 AM	TOGGLE
Jacuzzi:StatusUpdate:Pump4 (237)	Off	Today 11:53:08 AM	TOGGLE
Jacuzzi:StatusUpdate:Light1 (238)	Next	Today 11:53:08 AM	NEXT
Jacuzzi:StatusUpdate:Light2 (239)	Next	Today 11:53:08 AM	NEXT
Jacuzzi:Light:Status (240)		Today 11:53:08 AM	
Jacuzzi:Light:RGB (241)	0	Today 11:53:08 AM	COLOR
Jacuzzi:Light:Brightness (242)	0	Today 11:53:08 AM	
Jacuzzi:Light:Speed (243)	Static	Today 11:53:08 AM	
Jacuzzi:StatusUpdate:PrimaryFilterMode (244)	Holiday	Today 11:53:08 AM	HOLIDAY LIGHT HEAVY
Jacuzzi:StatusUpdate:FilterMode (245)	Light	Today 11:53:09 AM	HOLIDAY LIGHT HEAVY
Jacuzzi:StatusUpdate:ExpireClearRay (246)	10	Today 11:53:10 AM	10 SUBMIT
Jacuzzi:StatusUpdate:ExpireFilter (247)	148	Today 11:53:10 AM	148 SUBMIT
Jacuzzi:StatusUpdate:ExpireWater (248)	148	Today 11:53:10 AM	148 SUBMIT
Jacuzzi:StatusUpdate:LockAccess (249)	Locked	Today 11:53:10 AM	LOCK UNLOCK
Jacuzzi:StatusUpdate:LockAccessory (250)	Locked	Today 11:53:10 AM	LOCK UNLOCK
Jacuzzi:StatusUpdate:LockMaintenance (251)	Locked	Today 11:53:10 AM	LOCK UNLOCK
Jacuzzi:StatusUpdate:Id (252)	Query	Today 11:53:09 AM	QUERY
Jacuzzi:StatusUpdate:Panel (253)	Config	Today 11:53:09 AM	Config
Jacuzzi:StatusUpdate:SetTime (254)	Set Time	Today 11:53:09 AM	SET TIME
Jacuzzi:StatusUpdate:DegreeC (255)	C	Today 11:53:09 AM	C F
Jacuzzi:StatusUpdate:FilterStart (256)	0	Today 11:53:09 AM	
Jacuzzi:StatusUpdate:ErrorCode (257)		Today 11:53:09 AM	
Jacuzzi:StatusUpdate:WaterTemperature (258)	101	Today 11:53:09 AM	
Jacuzzi:StatusUpdate:ControlPanelTemperature (259)	91	Today 11:53:10 AM	
Jacuzzi:StatusUpdate:OtherUV (260)	0	Today 11:53:09 AM	
Jacuzzi:StatusUpdate:OtherBlower (261)	0	Today 11:53:09 AM	
Jacuzzi:StatusUpdate:OtherPrimary (262)	0	Today 11:53:09 AM	
Jacuzzi:StatusUpdate:OtherSecondary (263)	0	Today 11:53:09 AM	
Jacuzzi:StatusUpdate:FlowSensor1 (264)	0	Today 11:53:09 AM	
Jacuzzi:StatusUpdate:FlowSensor2 (265)	0	Today 11:53:09 AM	

Figure 76 Jacuzzi Auto Device Creation

11.5 Broadlink / BestCon RM Pro and Mini

Broadlink produces a family of reasonably priced devices that are capable to transmitting IR and RF, learning IR and RF codes, and has an online repository of many codes of commercial products. The device is designed to operate with an App via cloud. It can be taken out of the cloud for local operation. Much reverse engineering has been done with a leading contribution by with a python implementation available at https://github.com/mjg59/python-broadlink.

The mcsMQTT integration started with the .NET rewrite of the early python code available at https://github.com/kemalincekara/Broadlink.NET that includes several models through the RM2. With a combination of Wireshark and other information gleaned from various sources the capability expanced to include models through RM4 Pro. The RM4 Pro and RM4C Mini have been tested in this integration, but nothing has been intentionally done to prevent earlier models' operation.

The integration with HS provides the ability to connect the broadlink device to the local network, discover the broadlink devices this has been paired with the network, learn IR and RF codes from a remote, import IR codes from a Pronto hex format, and transmit IR and RF. The RF capability of the Broadlink is robust for carrier frequencies locused on 433 MHz and 310 MHz, but falls off rapidly for other frequencies. Fans are often controlled by 350 MHz or 305 MHz and the Broadlink will not be able to be used in these applications. Bond is a supplier for a wider RF frequency range.

11.5.1 Putting Broadlink Unit on Local Network

Joining the network is a one-time activity to teach Broadlink what network to use. It is done with Broadlink device setting up an open Access Point on a SSID it advertises. A computer or smartphone with WiFi capability connects to the Broadlink AP and then sends the desired network credentials. Broadlink device then drops its AP and connects to the desired network. It will reconnect to the network with each power cycle.

A utility is provided for the case where the HS computer does not have WiFi, but some Windows/Linux computer on the network does. A smartphone can also be used following the standard App for getting connected to the cloud, but stopping without actually completing the process. The danger is not stopping and the device is now cloud rather than local control. In this case one needs to start over with the long press of the reset button.

Each Broadlink device will communicate with one IP. This is normally the cloud server when using the Broadlink App. To take it out of the cloud it needs to setup to use the IP where mcsMQTT/HS is running. This is done by putting the device in setup mode where it is producing an AP with SSID "BroadlinkProv" or "Broadlink_WiFi_Device" and then providing it a packet that identifies the local network credentials.

The device has two setup modes. One is smart and one is AP. The AP is the one that is to be used. Smart is indicated by a rapid flashing LED. The AP is by an intermittently flashing LED. Hold the Broadlink setup button with a paperclip until it fashes rapidly. Relase it. Hold again for five seconds, but not 10 seconds, until it blinks slowly. This setup mode selection may change among Broadlink models so follow the instructons that came with the unit to put it in AP Setup mode.

The packet with network credentials is provided by mcsMQTT with the "Join Network" button. It can also be provided by running the utility "BroadlinkJoinNetwork.exe" (described below) or can be done

with smartphone, but one needs to abort the smartphone process before joining the cloud. Only one of these three options needs to be used.

This is setup from the Local page, IR/RF tab of mcsMQTT. The local network SSID, password and security mode information are provided followed by the button to join the network. See Figure 77.

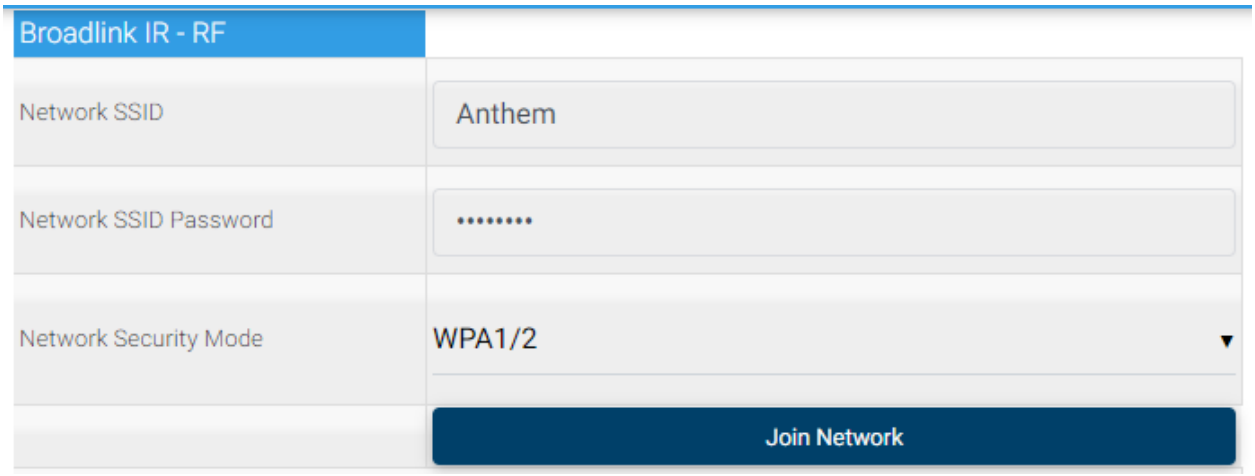


Figure 77 Take Broadlink Device out of Cloud

Once the Broadlink deivce is in AP mode then use the Join Network button to allow it to join the local network. It's IP will be obtained by DCHP so look at DHCP server/router to get it or use other means to sniff IP addresses. It may take multiple clicks of the Join Network button. Success is indicated by the BroadlinkProv SSID is no longer present and a DHCP server has given the Broadlink device an IP address.

The WiFi network credentials are not retained, but only used in conjunction with the Join Network button. This means that they need to be reentered if a second Broadlink device is going to be added.

As an alternate to joining the network a console application is also included in the bin\mcsMQTT that performs the same function as the Local IR/RF page. This allows the target computer to establish the network connection with the Broadlink device independent of HomeSeer. Note that when running "BroadlinkJoinNetwork.exe" the file "broadlink.dll" also needs to be present at the location where "BroadlinkJoinNetwork.exe" is running. Both of these files are available in the Updater zip or in \bin\mcsMQTT folder.

It is used with a command line containing parameters of "WiFi SSID, WiFi Password, Security Mode Number". From Windows it is run from a console with path currently at \bin\mcsMQTT using the command like

BroadlinkJoinNetwork "MySSID,MyPassword,4"

A similar Terminal window command is used on Linux

mono BroadlinkJoinNetwork.exe "MySSID,MyPassword,4"

It will provide a feedback direction to get the Broadlink device into mode to allow the WiFi network to be used. This is 10 seconds hold followed by 5 to 9 second hold of the reset button. From the factory

the Broadlink device will already be in this mode and have the Broadlink* SSID being advertised. In this case there is not need to use the reset button.

If there are issues with the command syntax there will be feedback when running the application. It may not take the first time so repeated execution may be necessary. The “BroadlinkProv” or “Broadlink_WiFi_Device” SSID will drop off as it connects to the local network. The blue lilght on unit will stop blinking when the WiFi to the local network has been setup.

11.5.2 Use of Broadlink Unit Overview

It may be possible to discover the IP address and this will be done each time mcsMQTT starts, but there are times that the Broadlink device is silent to this request. The known IP address can/should be entered in the text box provided. Refresh the page if entering multiple IPs. The status of the discovered units and the IP entry is shown in Figure 78. If units are not found then the name (e.g. RM4 Pro) will not be shown.

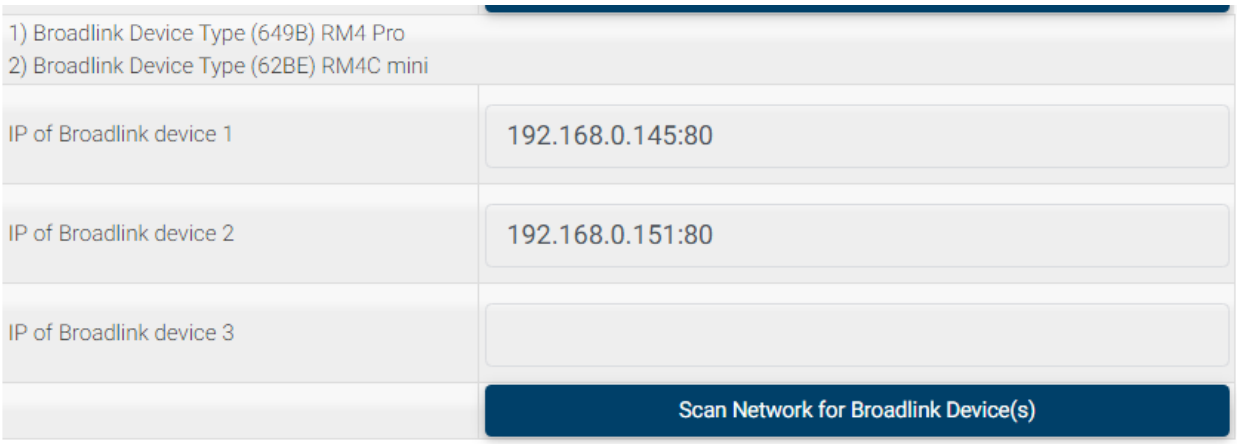


Figure 78 Broadlink Unit Discovery and IP Address

Two Device/Feature models are supported for Broadlink IR/RF. One is a Device for IR and/or RF for a Broadlink Unit. Features of the Device are the appliances that will be controlled by the specific Broadlink Unit. The DeviceValues and Status pairs will contain the set of codes for the appliance. This s a consise representation of the control information

The second is oriented for voice control where each appliance a Broadlink Unit will control will be a Device. Features will be created for each IR/RF code used to control the appliance.

A user can select which model best suits their need. They can be changed back and forth without loosing the appliance code information, but swapping will result in new HS Device Ref numbers. Examples of these two representations are shown in Figure 79 and Figure 80. The selection control is shown in Figure 81.

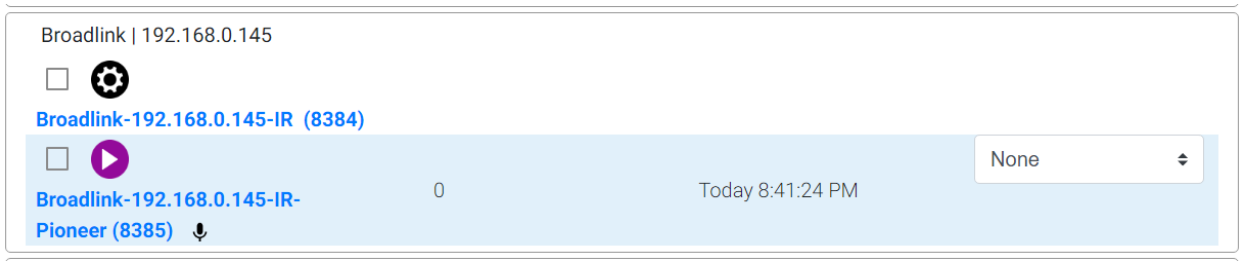


Figure 79 Broadlink VSP Oriented Representation

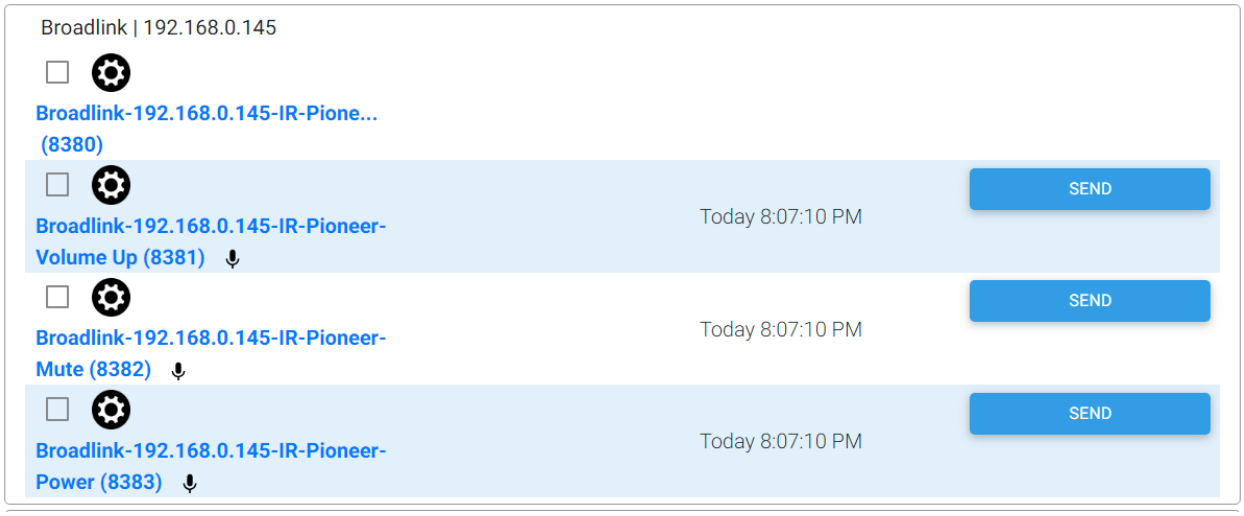


Figure 80 Broadlink Feature-Oriented Representation

At startup, entry of an IP or use of the "Scan for Broadlink Device" button a broadcast query will be made to confirm the Broadlink Unit is online and to get its authorization and encryption credentials. It will also be done if a HS request to send a code is made and prior attempts have not succeeded.

To start a learning process a Broadlink Unit needs to be selected. It will be the one being controlled when learning from a remote. If auto-assignment is selected it will also be the one for which a HS Device will be created. Both the Broadlink Unit selection and auto vs. manual assignment selection are shown in Figure 80.

Learning also requires that an appliance and the code function be identified. These are two text boxes shown in Figure 80. When the learning function is complete the code function text box will be cleared. The appliance text box will remain to allow subsequent code learning.

Rather than learning an appliance code function from a remote, a Pronto hex code can be entered. There are two ways to do this. One is to cut/paste a Pronto code for another source. In this case the appliance name and code function need to be entered prior to entering the Pronto hex in the text box.

The second approach is to import a file that contains the Pronto codes for a given appliance. In this case the filename rather than the individual Pronto code is entered. The contents of the file are assumed to be alternating lines of the code name and the hex code in Pronto format. Blank lines are ignored. For example, a file may have the following contents. The code names are imported with spaces replaced by underscores to avoid syntax and parsing issues.

Toggle power
0000 006C 0022 0002 015B 00AD 0016 0016 0016 0016 0016 0016 0016 0041 0016 0016 …

Toggle display
0000 006C 0022 0002 015B 00AD 0016 0016 0016 0016 0016 0016 0016 0041 0016 0016 …

Buttons are also available to cancel a learning that has started and to play back the learned or imported code. Cancel will be used when the Broadlink Unit is not able to recognize the remote or can be used to abort the process.

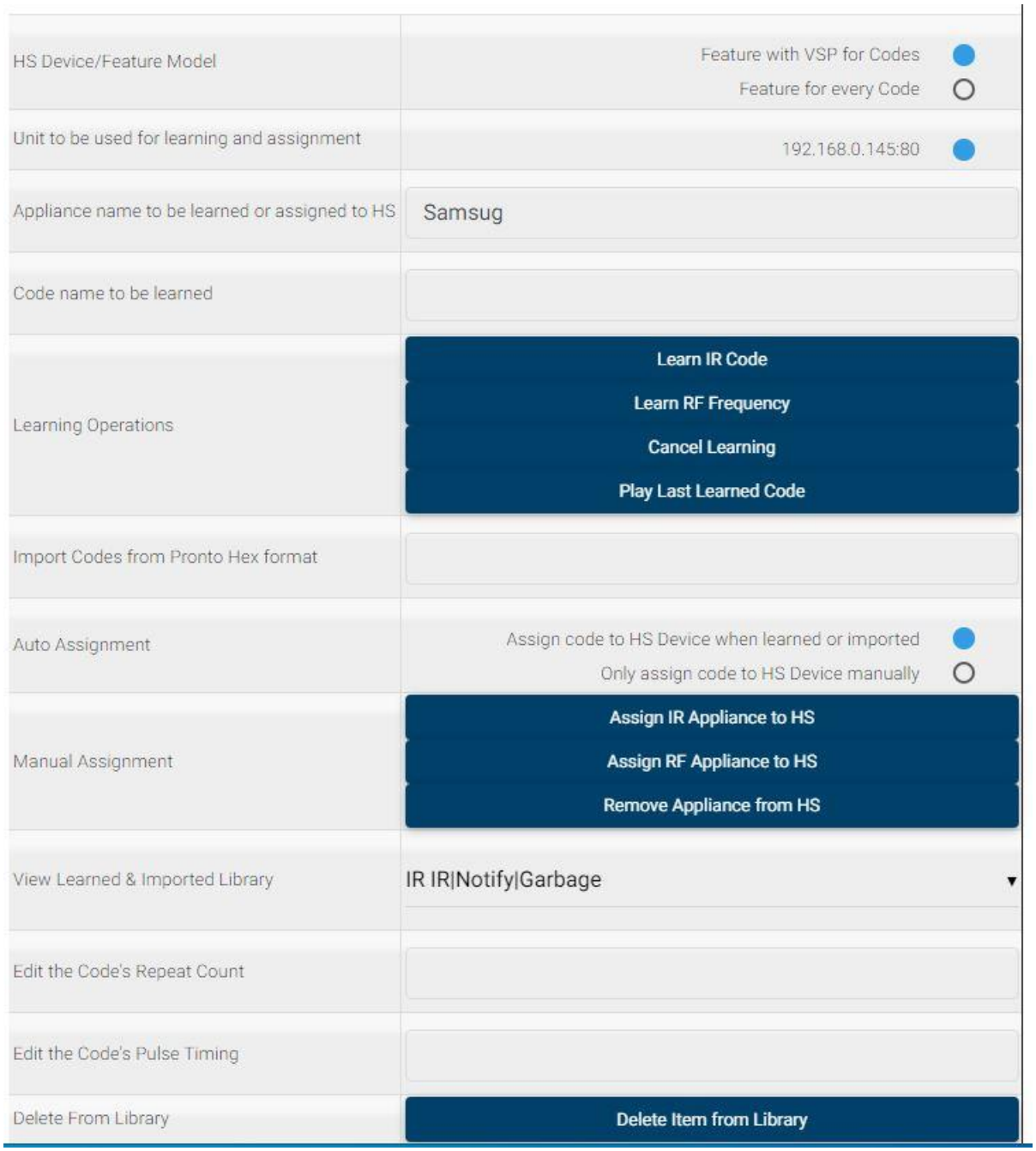


Figure 81 Broadlink Learning and Verification

11.5.3 Learning and Importing Detail

Learning can be done of IR and RF code from an existing remote device. They can also be imported from codes available on internet sites such as http:// irdb.tk/find/ that will be available in pronto hex format. RemoteCentral is another location that a large user-contributed database of pronto codes, but it is somewhat more dated. An example of the import format for Samsung TV numbers 0 and 1 is shown in Figure 82. To use them enter the appliance name and code name into the mcsMQTT textboxes, copy from the site the hex sequence text and paste into the Import Codes from Pronto Hex format textbox.

Known IR codes for Samsung TVs

Note that not every Samsung TV necessarily supports all functions on this page.
Possibly only the top-of-the-line models have all the functions.

What to do with these codes? Program them into your programmable remote control, or build something great using Arduino, or...

Protocol information | Pronto Hex | UEI Hex | Raw | Widget

0

```
0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016
0016 0016 0016 0016 0016 0016 0016 0041 0016 0016 0016 0016 0016 0016 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0041 0016 0041 0016 0041 0016 0016
0016 0041 0016 0041 0016 0041 0016 06FB
```

1

```
0000 006C 0000 0022 00AD 00AD 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016
0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0041 0016 0041 0016 0016 0016 0041 0016 0041
0016 0041 0016 0041 0016 0041 0016 06FB
```

Figure 82 Sample Pronto IR Code

The IR and RF codes are stored in \Config\BROADLINK.ini. It is possible to manually add entries to this file if there are other sources from which the codes can be imported. The “Assign IR Appliance to HS”, “Assign RF Appliance to HS”, and “Remove Appliance from HS buttons” are used to manage the subset of the Appliances in BROADLINK.ini (i.e. library) that are visible to HS. For example, the “View Learned & Imported Library” selector can be used to view what appliances are available in the library. The appliance name then can be put in the “Appliance name to be learned to assigned to HS” textbox. The “Assign IR Appliance to HS” button then used to create a HS control for this appliance and this control will include all the IR codes from the library for this appliance.

Learnning is performed by entering the name of the code to be learned and then using either the Learn IR or Lear RF Frquency button. The Broadlink device will illuminate its LED to indicate it is in learning mode. The remote should be place in front for IR and nearby for RF and button pressed. The Broadlink device will turn off the LED when the learning operation has completed. mcsMQTT will provide feedback as well on the Local Page. The Cancel button is available should the Broadlink device not be able to learn the code.

The IR learning is a simple process that completes very quickly. The RF learning is a two-step process. The first is where the Broadlink devices detects the carrier frequency. The second is where it detects the encoding on this frequency. Frequencies near 433 MHz and 315 MHz are supported by the Broadlink. It appears the electronics are optimized for 433 MHz and the signal strength for 315 is not as strong when transmitting 433.

The RF learning starts with the Frequency detection. When found the LED will no longer be illuminated, mcsMQTT will show in red feedback that this phase is complete and the RF Freq button will be replaced by a RF Code button. Click on the mcsMQTT RF Code button to start learning of the code at the detected frequency. Prompt will be present for short press of the remote for the code to be learned. Usually only one press is needed and the LED will no longer be illuminated to show the success. mcsMQTT will also provide the success feedback. If the process fails, then use the Cancel button on mcsMQTT and start over.

For Pronto hex then code can be obtained interactively from the web site and the code pasted into the pronto hex text box after a name has been entered for the code. mcsMQTT will convert the code to the Broadlink format and stored just as if it had been learned with the physical remote.

11.5.4 Appliance Code Library

All learned and imported codes are maintained in the file \Config\BROADLINK.ini. One section is for IR and one for RF. Each row will contain the Appliance|Code name and the base64 Broadlink format for playback.

This file is augmented as new codes are learned and imported. The library of codes can be made available to HS using the manual assignment buttons on Figure 81. Prior to manual assignment the Broadlink Unit selection radio needs to be set and the Appliance name text box needs to be completed.

This same assignment operation can be done immediately after a new code is learned. This is the auto-assignment option shown in Figure 81. If the same Appliance is to be controlled by two Broadlink Units then the second one will need to selected with the radio and the manual assignment button used.

A dropdown selector is available to view the Appliance|Code names contained in the library. There is no control action with a selection. However, after this selection the ability to edit the pulse and repeat properties as well as deleting the item from the library becomes enabled.

The "Delete Code from Library" button the item will be permanently removed from Config\Broadlink.ini file. The textboxes for pulse and repeat will augment the library code with instructions for modifying the pulse timing or repeating the code. See Section 11.5.5 for more information on these properties. The event JSON, when used, will have priority over properties defined here in the library.

11.5.5 Broadlink MQTT and Event Interface

The Broadlink information will also be available via the MQTT Page, Association and Edit tabs. The Topic structure is "Broadlink/IPxxxx/YY/Appliance/Code" or "Broadlink/IPxxxx/YY/Appliance" depending upon the HS Device model of Feature vs. VSP, respectively. The IPxxxx will be the IP of the Broadlink device. YY will be IR or RF. Appliance and Code are the names assigned at time of learning.

The Association Tab will show which topics have been associated with HS devices and will also show a "/set" suffix added for the publish topic. When HS commands an IR or RF code there will be a MQTT message sent. If a MQTT message is received that matches the topic then the code will be sent through the Broadlink Unit.

When used in HS Events the control action will be the Broadlink IR or RF feature. There are no triggers available for when an IR or RF code has been learned or sent beyond that provided by the DeviceValue change of the feature.

Generally, when an IR or RF code is requested to be sent the code will be the one that was learned. It is also possible to modify the IR code as part of an Event Action. In this case a JSON payload is used to specify the tweaking that is to be performed. See Figure 83.

The JSON payload consists of any of three keys. If a key is not in the payload, then the code will be used unchanged from the learned library.

“code” is used when the VSP device model is used to specify which code is being sent. It can also be used in the Feature model and should be the same as the code using the Topic.

“repeat” is the number of times the code IR should be sent. A value of 0 is zero repeats. 1 is for one repeat which means the code is sent twice.

“pulse” is the change in the number of pulses that represents each On/Off burst. This can be used to tweak the timing of the IR signal for sensitive equipment. A value of -1 will reduce the number of pulses at each transition by one. A value of 3 will increase the number of pulses by 3.

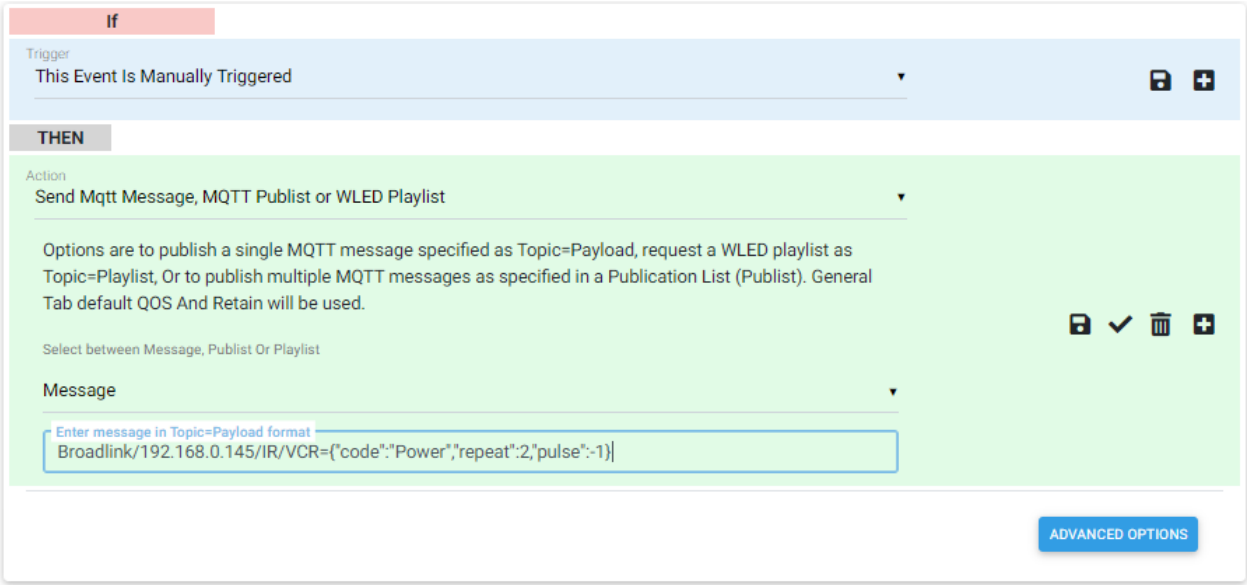


Figure 83 Broadlink Control Event Action

11.5.6 Broadlink Sensors

Broadlink makes available a USB power cable (Model HTS2) that contains temperature and humidity sensors in the cable. The RM4C Mini does not support this cable, but others do. The earlier RM versions only support temperature. If it is being used then mcsMQTT will create the HS devices as shown in

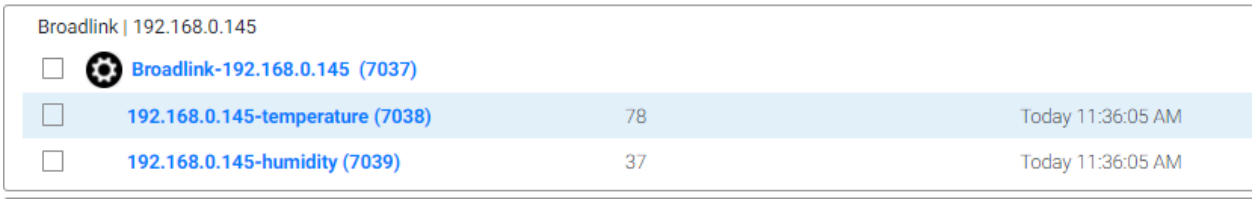


Figure 84 HS Devices for Broadlink Sensors

The device delivers temperature in Celcius. If Farenheit is desired then use the Edit tab or popup Expression text box to do the conversion. For example enter $$PAYLOAD:*1.8+32.

11.6 Bluetooth

11.6.1 Sensors and Actuators

There are a multitude of low powered devices that transmit locally via Bluetooth. Most are sensors such as temperature, humidity, door/window, motion, etc. There are some that can be controlled such as the Switchbot Bot or Curtains.

Bluetooth is similar to Zigbee with respect to battery usage and range, but generally a hair slower in responding. This difference will usually not be noticed. Because of the limited range, whole-house coverage cannot be achieved with a single Bluetooth receiver. Fortunately, the ESP32 has built-in Bluetooth in most of its models that can act as a bridge between Bluetooth and WiFi or wired ethernet.

Firmware for the ESP32 comes in various varieties such as with ESPHome and Arduino projects. OpenMQTTGateway is the choice made for support with mcsMQTT due to it extensive library of devices that it will decode and active support to handle new ones as they become available. The firmware is available at links from OpenMQTTGateway v1.7.0 .

mcsMQTT will accept all Bluetooth devices discovered by OpenMQTTGateway and place their properties in the Association Table that is available on the MQTT Page, Association Tab. For the popular ones such as Shelly and Switchbot it will auto-create HS Device and Features. The others of interest can be associated with HS by clicking the “a” column checkbox on the Association Table.

In the case of Shelly Bluetoth devices, it is possible to use the ESP32 contained within the powered Shelly Gen2 and Gen3 devices to act as gatetways. mcsMQTT will automatically detect the Shelly Gen2/3 devices and configure them to support the gateway function using the same MQTT message format that is used for OpenMQTTGateway. It will use the MQTT Topic ShellyBLU/x/BTtoMQTT/y where x is the Id of the Shelly Gen2/3 device and y is the MAC address of the transmitting Bluetooth device.

mcsMQTT is made aware of OpenMQTTGateway (OMG) from the Local Page, Bluetooth Tab as shown in Figure 85.

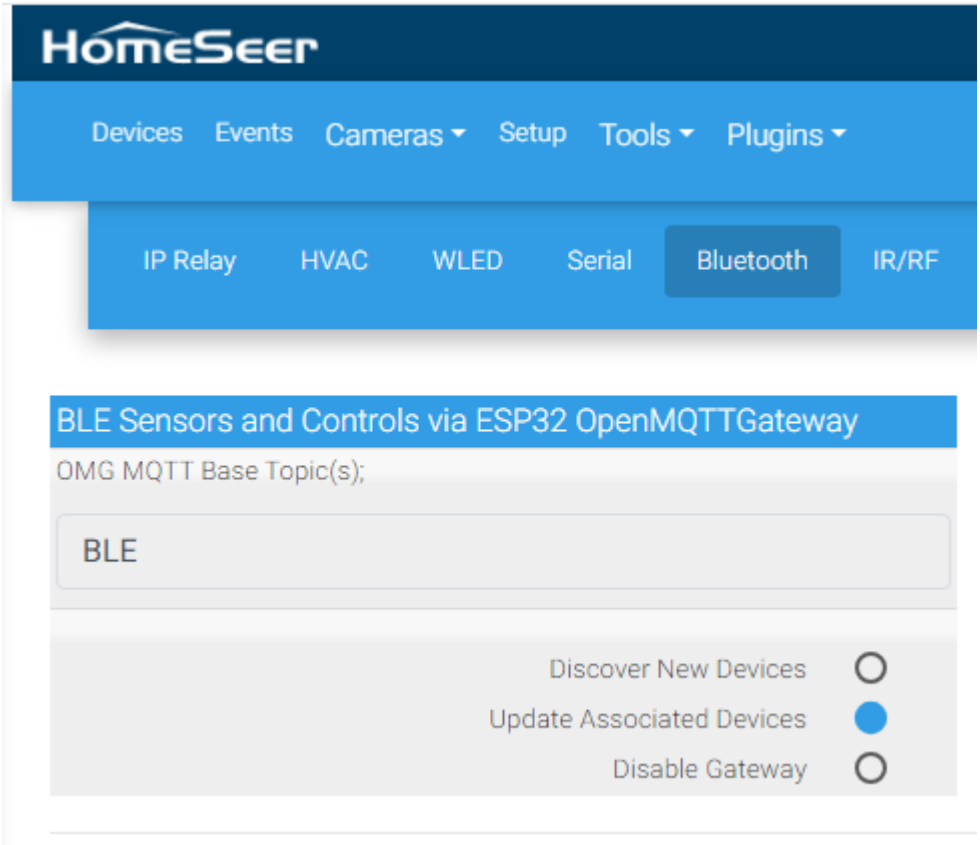


Figure 85 Bluetooth Sensor and Actuator Setup

OMG has its own Web Server for configuration and it includes specification of the base Topic that will be used. It defaults to “home/” but can be changed if desired. Whatever is used is identified to mcsMQTT by the Base Topic entry shown on Figure 85. If more than two ESP32 devices are being used for OMG then each can be specififed using semicolon to separate each. mcsMQTT does not expect the trailing “/” in its setup as is used in OMG.

This Topic is used to know that a message may need special handling to look for Shelly, Switchbot and others for auto-creation. It is also used in conjunction with the three radio button positions under the Topic. The default is Discover and operates like the same on the MQTT Page Client Tab for MQTT Topic discovery. This can be chatty as some BLE devices tend to advertise their presence every few seconds.

The plugin also configures OMG to send only messages for things that look like sensor/actuator type things and excludes the beacon type reports that are normally used for tracking. This significantly lowers the traffic.

When the radio is in the middle position, mcsMQTT will create a whitelist of BLE devices that OMG should forward updates. This really quiets it way down. The whitelist is formed by the set of BLE devices that have been associated with HS. In the Shelly BLU DW case, the message is only received when the sensor changes.

Another message filtering approach to reduce the transmissions from OMG is to use its blacklist. Any Topic in the Association Table that from OMG can be checked in the “r” column to mark it for rejection. The set of rejected Topics will be delivered to OMG as the blacklist.

In summary, Association Table “a” is used for whitelist and “r” is used for blacklist of the OMG-delivered Topics.

The bottom position will disable all updates from OMG.

The received data in the Association Table will look similar to Figure 86. This figure shows where some of the properties of the Shelly DW have been automatically associated with HS Devices, where some are not associated, but are available for viewing or association, and other device Emporia, PROV_437034v was not associated. The HS View of the Shelly DW Device is shown in Figure 87.

Association Table for Auto Association of MQTT Topic and HS Device

Λ	o	r	e	a	ref	TOPIC	payload	h	s	l	lastdate
0	☐	☐	☐		4957	BLE/OMG_ESP32_BLE/BTtoMQTT					
1	☐	☐	☐			BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C					
2	☐	☐	☐	☑	4958	Dev: shellies|shellies|040D84E8528C-batt Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:batt Pub: the following Topic on Device command		☐	☐	☐	2024-02-13 19:27:54
3	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:brand	Shelly	☐			2024-02-13 19:27:54
4	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:id	04:0D:84:E8:52:8C	☐			2024-02-13 19:27:54
5	☐	☐	☐	☑	4959	Dev: shellies|shellies|040D84E8528C-lux Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:lux Pub: the following Topic on Device command		☐	☐	☐	2024-02-13 19:27:54
6	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:mac	04:0D:84:E8:52:8C	☐			2024-02-13 19:27:55
7	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:model	ShellyBLU Door/Window	☐			2024-02-13 19:27:54
8	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:model_id	SBDW-002C	☐			2024-02-13 19:27:54
9	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:name	SBDW-002C	☐			2024-02-13 19:27:54
10	☐	☐	☐	☑	4961	Dev: shellies|shellies|040D84E8528C-open Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:open Pub: the following Topic on Device command		☐	☐	☐	2024-02-13 19:27:54
11	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:packet	63	☐			2024-02-13 19:27:55
12	☐	☐	☐	☑	4960	Dev: shellies|shellies|040D84E8528C-rot Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:rot Pub: the following Topic on Device command		☐	☐	☐	2024-02-13 19:27:54
13	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:rssi	-35	☐			2024-02-13 19:27:54
14	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/040D84E8528C:type	CTMO	☐			2024-02-13 19:27:55
15	☐	☐	☐			BLE/OMG_ESP32_BLE/BTtoMQTT/34865D437036					
16	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/34865D437036:id	34:86:5D:43:70:36	☐			2024-02-13 19:27:51
17	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/34865D437036:name	PROV_437034v	☐			2024-02-13 19:27:53
18	☐	☐	☐	☐		Sub: BLE/OMG_ESP32_BLE/BTtoMQTT/34865D437036:rssi	-81	☐			2024-02-13 19:27:51
19	☐	☐	☐			BLE/OMG_ESP32_BLE/BTtoMQTT/B09122F762BF					

Figure 86 Bluetooth Devices in Association Table

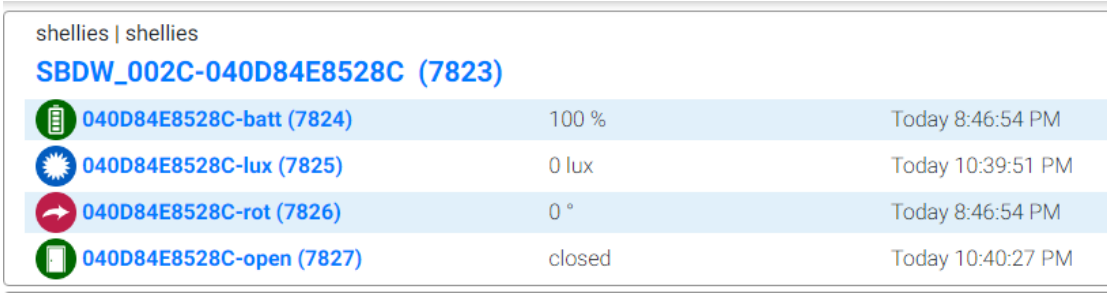


Figure 87 Shelly Bluetooth Door Window Sensor

11.6.2 Beacon

Beacon monitoring using BLE is available on Windows and Linux platforms. Evaluation on Linux has only been done with RPi usng its built-in Bluetooth capability. Windows 10 is required for the Windows platform using a USB Bluetooth receiver.

On Windows 10 platform that has a Bluetooth Low Energy interface it is possible to monitor the presence of BLE beacons with HS4 version of mcsMQTT. The products used for this evaluation include the following as well as other BLE beacons.

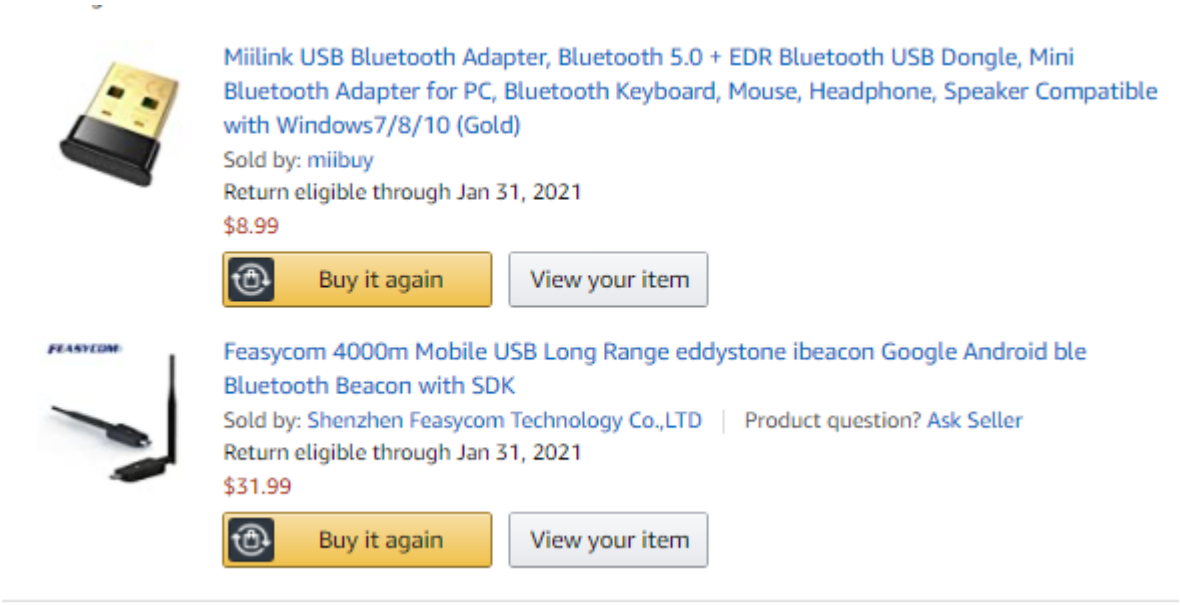


Figure 88 Bluetooth Beacon and Interface

Beacon discovery is controlled by the setting on the Beacon tab of the HS4 Local page as shown in Figure 89 or on the HS3 BLE page. The top radio disables the beacon scan on the local computer and does not process any MQTT Beacon/ topics from remote computers. The second radio continues to update the status of previously discovered beacons. The third radio enables discovery of new beacons. The fourth

radio disables the local computer BLE processing but will update existing beacons from remote computetrs. The last radio allows new beacons to be discovered from remote computers.

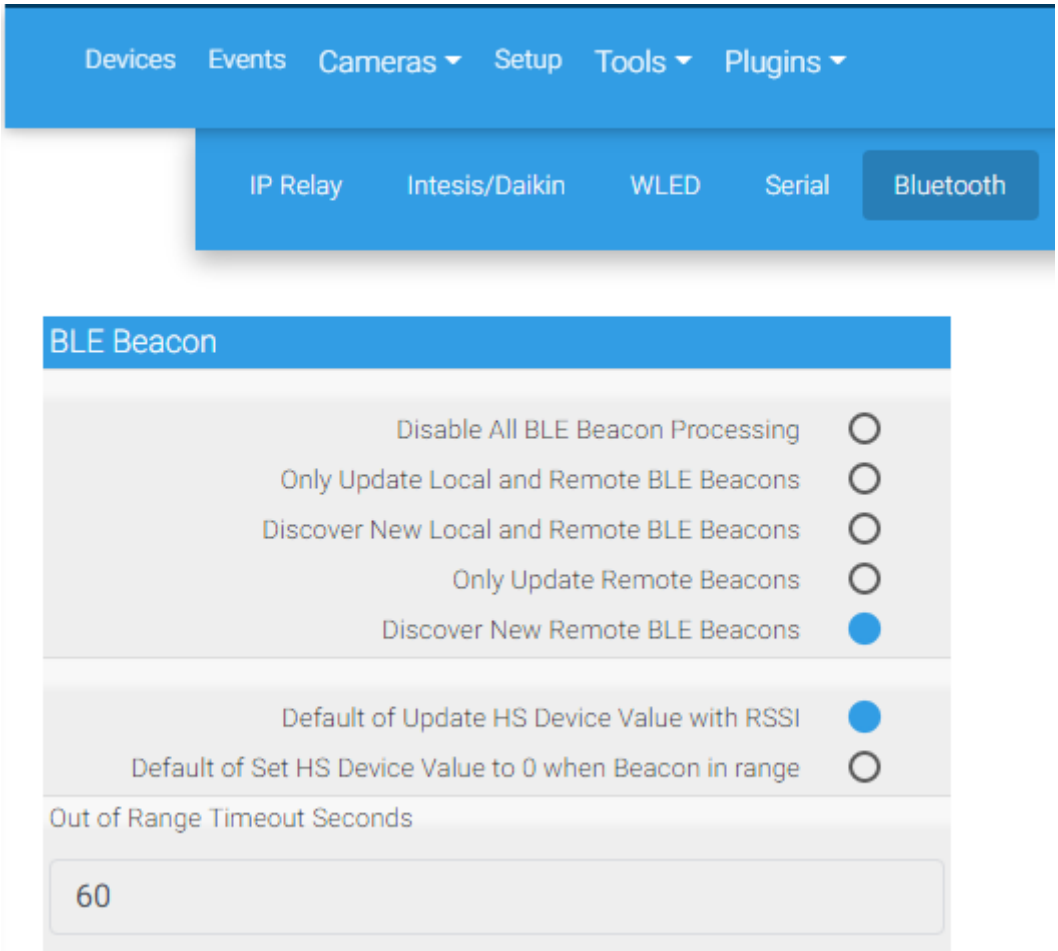


Figure 89 Beacon Tab Settings

Discovered beacons appear in the Association table. They can be easily identified by using the Topic filter of Beacon in the T1 position. A beacon for which status updates are desired in HS will be associated by using the “A”ssociate column checkbox. The bottom row of Figure 90 illustrates this and the HS device appearance is a shown in Figure 91.

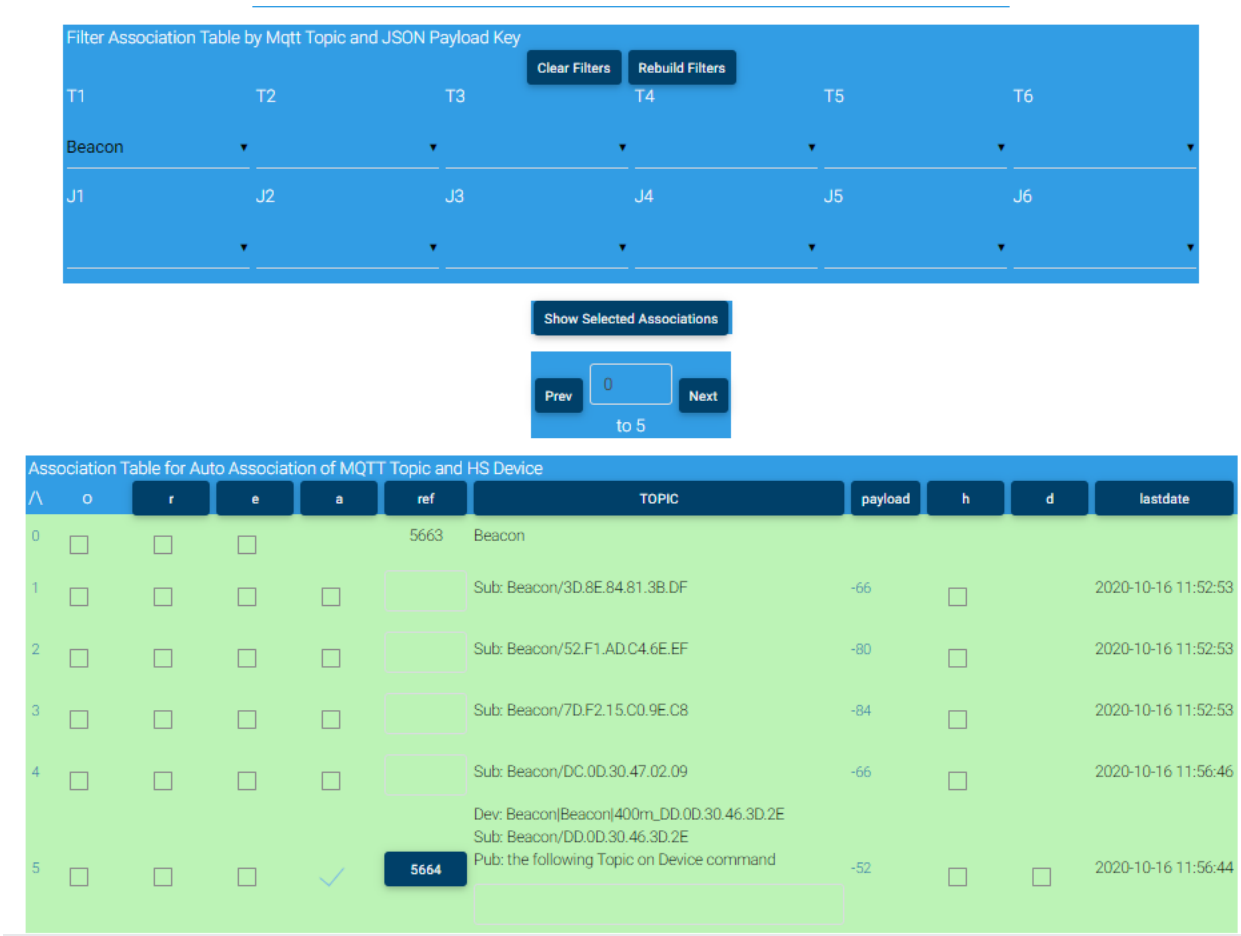


Figure 90 BLE Beacon in Association Table

Figure 91 Beacon Status in HS

When a beacon advertisement is in range of the Bluetooth interface the HS device will be updated with the signal strength (negative of RSSI) shown in the DeviceValue and green icon. The signal strength is smoothed with a low pass filter so its value will not be jumping around due to sampling differences.

When out of range for 60 seconds the icon will change to red and the DeviceValue will be changed to -1.

In a typical situation there will be only a limited number of beacons that are of interest. Once these have been discovered then the Beacon tab entry changed to the second or fourth position. It can be changed to allow new discovery later if a different beacon is to be monitored.

The Association table will contain beacons that not of any interest. To reduce clutter the “O”bsolete column checkbox can be used to remove them.

Beacons that are associated with HS devices can be customized on the Edit tab to change the location and name if desired as shown in Figure 93. Edit tab is most easy viewed by clicking the Ref column button of the device that is being edited from the Association table.

The global beacon timeout and RSSI parameters setup on the Local page, Bluetooth tab shown in Figure 89 can be altered on a beacon-by-beacon basis on the Edit tab. When a beacon is accepted for HS device association these parameters will default to what has been setup on the Local page, but later an be changed on the Edit tab or the MQTT page.

Two applications are made available for Linux that can be run on local or remote computers. BLEMQTT is for HS3 positions identification support. BLEMQTTHS4 is for HS3 or HS4 to support Home-Away logic. In the HS3 case the mode of operation of the plugin for BLE is selected on the BLE Page as shown in Figure 92.

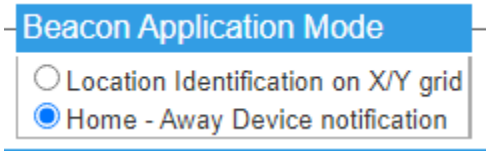


Figure 92 HS3 BLE Application Mode Selection

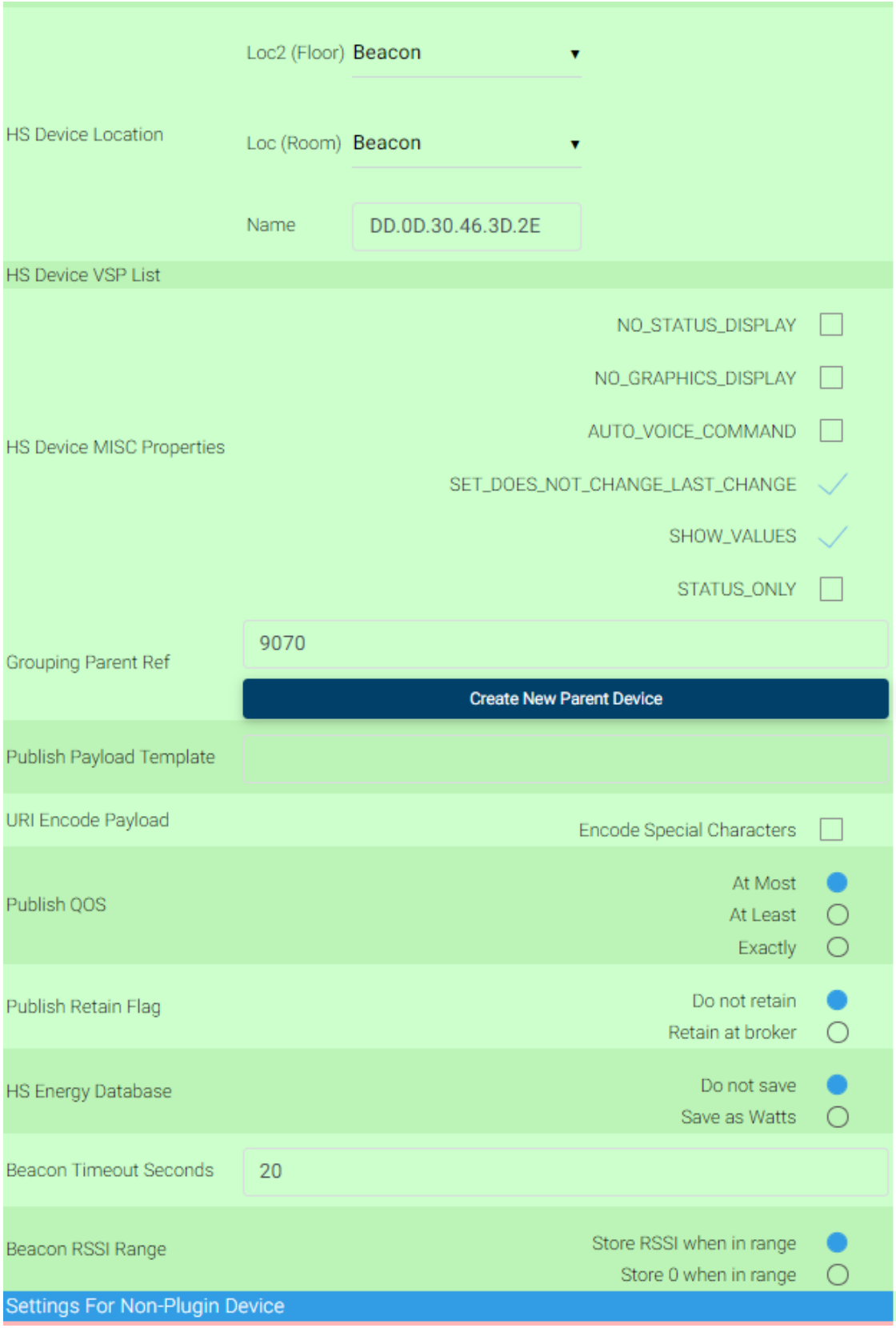


Figure 93 Beacon Parameter Edit

11.6.3 Espresense

Espresense is a application that is installed in an ESP32 that listens for Bluetooth beacon advertisements and can interrogate them for more detailed information. What it attemts to do is overcome the randomization of the advertised MAC address that is done on smarthones, but fingerprinting the reporting device using behaviorial and other information that it can ascertain. This means it may be possible to provide tracking of presence of a smartphone as well as other Bluetooth devices that provide a periodic advertisement.

The Espresense integration is a middleground integration for Bluetooth between what is available with HS3 where a specific (X,Y) coordindate is determined based upon a beacon's signal strength relative to a set of ESP32 or RPI listening stations. The other extreme is the HS4 integration where the objective of Bluetooth tracking is to determine if a beacon is within or out of range to provide a simple home vs. away status. The Espresense integration isolates a Bluetooth device to the nearest ESP32 station where the assumption is that ESP32 stations are located in various rooms so isolation provides the room location of the Bluetooth device.

Theintegration provided by mcsMQTT is a variant of the integration of the HomeAssistant integration described at https://espresense.com/. This link will also be the starting place to understand the hardware and simple approach to loading the application in an ESP32.

mcsMQTT accepts the MQTT messages delivered on the espresence/devices/+/+ topic where first + identifies the Bluetooth device (fingerprint) and the second + identifies the ESP32 room identification. The JSON payload on these topics includes distance between the Bluetooth device and the ESP32 room as well as other related information.

mcsMQTT will create a HS Device Feature for each user-selected Bluetooth device and report which room this device is located. A room location is setup by the user as a distance radius from the ESP32. It is also characterized by a dwell time after which the Bluetooth device is assumed to have exited the room if no distance update has been received for this time. The radius and dwell parameters are setup on the Rooms Table shown in Figure 95. This table can also be used to delete rooms that have become obsolete and are no longer being reported by Espresense. If a Room is still being reported then it cannot be removed from the table.

There are two special case "virtual" rooms called Home and Away. A Bluetooth device is assigned the Home room when it is still reporting, but no longer within the radius of any of the ESP32 rooms. Away room is assigned when no reports from any ESP32 room has been received for 60 seconds. A typical HS Devices page view for a couple Bluetooth beacons is shown in Figure 94. In this example the first Bluetooth device "eddy…." Is showing Home status with icon reporting no specific room. The second "apple…" shows it is in room family. Family is the name setup in the ESP32 configuration of the room name. The dates shown are the times they entered the room.

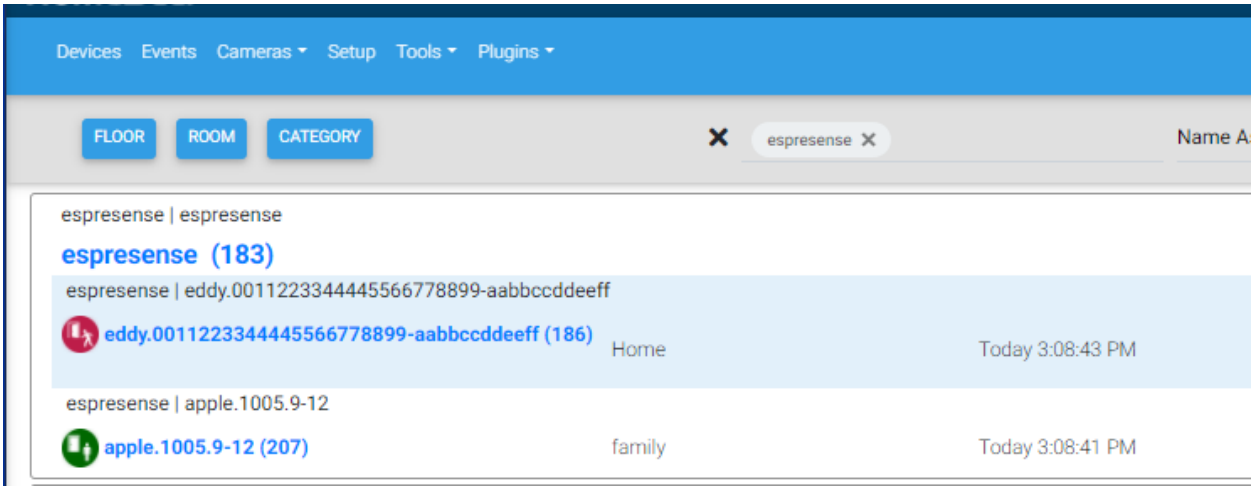


Figure 94 Espresense Status Reporting in HS Device Features

The setup of the room and Bluetooth device parameters is done on the Local Page, Bluetooth tab on HS4 or on the BLE Page of HS3. Two tables contain configuration parameters. One for each ESP32/Room and one for each Bluetooth device. See Figure 95.

The first table shows “Away”, “Home” and two ESP32 rooms. The room “family” has a radius of 30 feet and the room “den” is 5 feet. A Bluetooth device needs to much closer to the den ESP32 than it needs to be to the family ESP32 for it to be considered in the den. For both rooms the dwell time is set at 10 seconds so if the distance is not updated in 10 seconds the room will be assigned elsewhere. The assignment priority is:

1 Within Radius of nearest room with room still reporting within dwell seconds
2 Within radius of a room with bigger radius and room still reporting it
3 Outside the radius of currently assigned room with transition occurring in dwell seconds
4 Outside the radius of all rooms and some room still reporting it (Home)
5 No room reporting it for 60 seconds (Away)

Selecting a large radius will allow overlap and could eliminate all “Home” assignments. Selecting small radius will result in being the room when very close to the ESP32 which could result in a large number of “Home” assignments.

Selecting a large dwell time will delay the reporting of a Bluetooth device leaving a room unless it gets within the radius of another room. Selecting a small dwell time could result in excessive transitions out the room due to intermittent reporting from the ESP32.

The Bluetooth devices table allows creating of HS Device Features to track a Bluetooth device. It also provides the ability to record the changes of room assignement in a database from which charts or history log can be later viewed. The final parameter is the TxPower gain of the Bluetooth device. This is not an exact calibration, but does provide a means to account for variation in the Tx Power of different Bluetooth devices. Typically, battery powered devices transmit at lower power levels than those that use mains power.

Delete	Espresense Room	Radius (feet)	Exit Dwell (seconds)
☐	Away	0	0
☐	Home	0	0
☐	den	7	30
☐	family	7	30

Associate	HS Ref	Espresense Devices	Tx Gain	ShortTerm	LongTerm
☐	-1	msft.cdp.0902	1	☐	☐
✓	2169	iBeacon.426c7565-4368-6172-6d42-6561636f6e73-3838-4949	1	☐	☐
☐	-1	c47c8d6c8a43	1	☐	☐
✓	2168	apple.ipad6-11	1	☐	☐
☐	-1	apple.ipad5-1	1	☐	☐
☐	-1	apple.ipad14-1	1	☐	☐
☐	-1	eddy.0011223344445566778899-aabbccddeeff	1	☐	☐

Figure 95 Espresense Configuration in mcsMQTT

Espresense publishes distance information in MQTT Topics espresense/rooms and espresense/devices Topics. This information is redundant between the two. mcsMQTT uses that which is published in espresense/devices. The ESP32 setup should exclude the espresense/rooms. The telemetry topic contains ESP32 parameters that is not used in room identification, but could be of interest. Figure 97 shows a suggested setup for Preferences and Calibration.

The matrix of distance of each Bluetooth device from each ESP32 room as well the time since that distance was published in available for analysis as shown in Figure 96. A button is provided to refresh the matrix of data. Distance and time in black is the assigned room for the device. The distance units will be either f=feet or m=meters. The time units will be up to three digits with units of s=seconds,

m=minutes, h=hours, d=days. Devices that have not been associated with HS will be shown with gray names. Those that are associated will be black.

Two checkbox controls are provided. The first puts the MQTT data received for the rooms and devices in the Association table of the MQTT Page, Association tab. This raw data can then be used in HS or used for analysis with charts.

The second checkbox is used to inhibit addition of newly discovered devices. Each time mcsMQTT starts it will remove the unassociated espresence devices. This checkbox provides this control to remove them immediately and to prevent new devices from being included in the room asssignements.

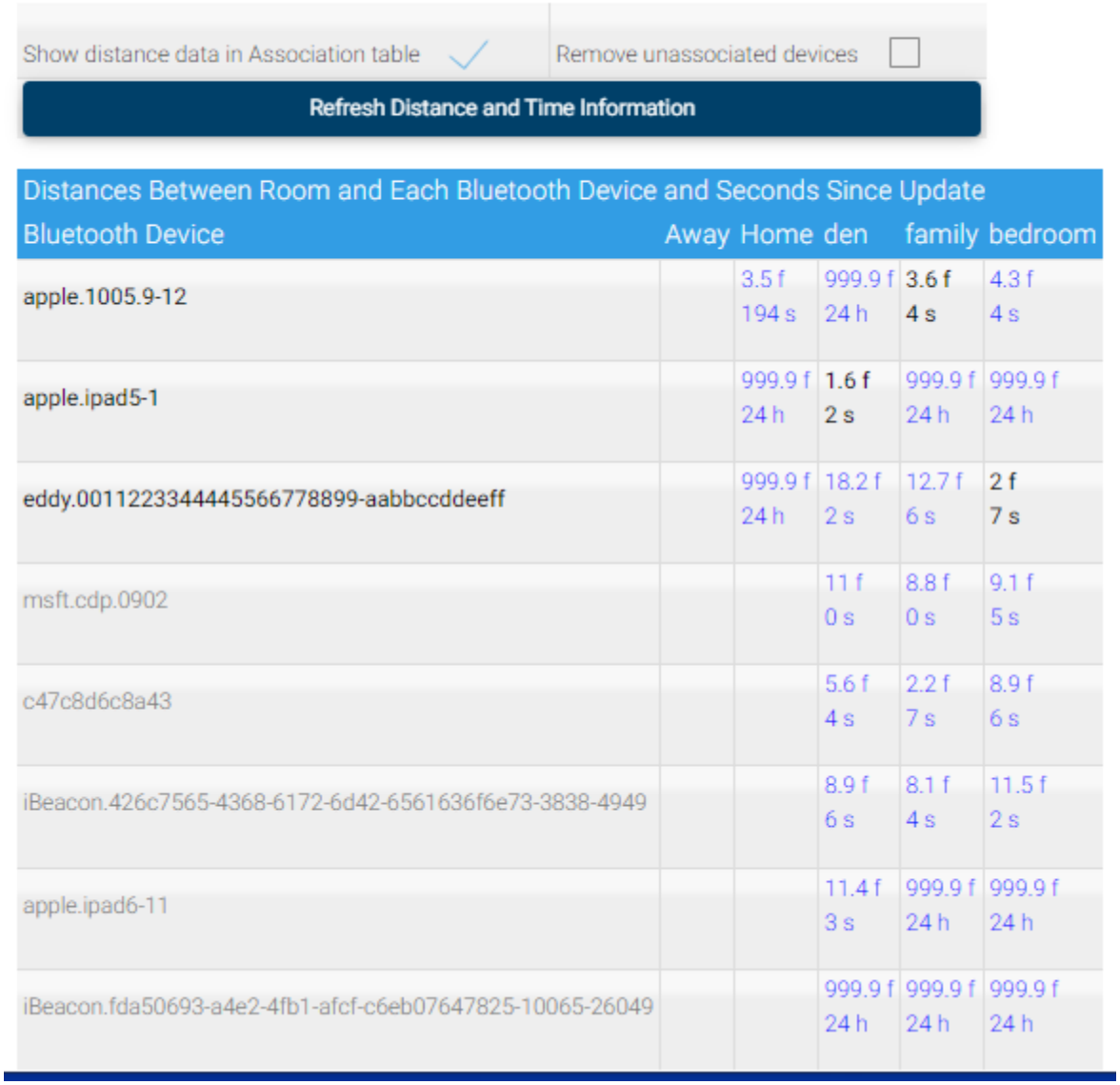


Figure 96 Espresense Distance - Time Matrix

All distance information reported by Espresence is available on the MQTT Page Association Tab if the checkbox is enabled to show it. In general, it will not be of much interest unless doing investigative

analysis. The downside to using the checkbox is the CPU burden it could incur when raw data is being received at a high rate. It will be under the espresence/# Topic such as shown Figure 98. An example of viewing time history of Bluetooth device and distances is shown in Figure 99. This was a case for ESP32 stations being unpowered and then repowered for testing. To support this chart data was collected in the internal Sqlite database. Other than the thee MQTT Topic checkboxes the others are at user discretion.

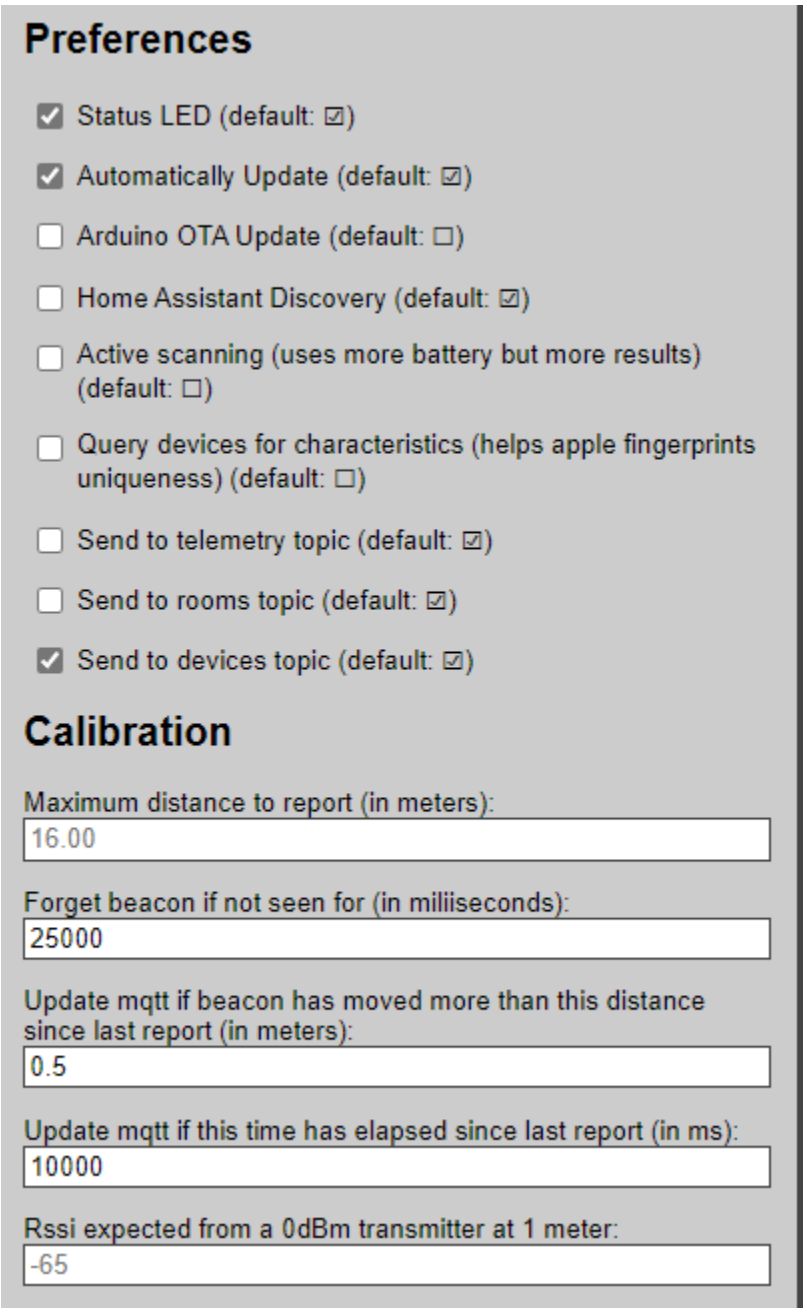


Figure 97 Suggested Espresense ESP32 Setup

Calibration is setup to balance the capability provided by mcsMQTT and Espresense. The top two settings are for what MQTT traffic is produced as a Bluetooth device is at long distance from the ESP32. The two lower settings are for MQTT traffic that is published when near the ESP32. Each pair of settings are for distance threshold and for time threshold.

The maximum distance setting is not critical as the maximum time will typically occur before the distance threshold has been exceeded for most battery-operated Bluetooth devices.

The maximum time setting works in concert with the time setting for a close device and the 60 second monitoring interval used by mcsMQTT. When an ESP32 has not published a distance update for 60 seconds then mcsMQTT will move the device out of a previously assigned room. The total time to recognize a device is no longer present in a room then becomes the max time setup for ESP32 plus 60 seconds.

When a device is close to the ESP32, the ESP32 will unconditionally publish distance at the close time interval which is shown Figure 97 at suggested 10,000 millisecond rate. When the device has gone out of range of the ESP32 then it will no longer be published at this rate. mcsMQTT will then detect it 60 seconds later.

This unconditional time interval for in-range devices can be made shorted to reduce latency at the expense of CPU burden to handle the more frequent raw data reporting.

The final setting is the amount a device needs to move before a MQTT message is produced. This is in addition to the unconditional periodic reporting. Since the raw data is being used to select the closest room for a Bluetooth device, the selection of the minimum distance interval should be made in context of how far apart ESP32 stations are located. If they are far apart then a larger minimum can be used since it will take some time for the device to move. If they are close then a smaller distance should be used to the point where the variation in RSSI meansuments result in apparent motion when none has actually occurred.

Clear Filters | Rebuild Filters

T1	T2	T3	T4	T5	T6
espresense					

J1	J2	J3	J4	J5	J6
distance					

Show Selected Associations

Prev 0 Next

to 13

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
0	☐	☐	☐	☐		Sub: espresense/devices/apple.1005.9-12/den:distance	1.16	☐			2022-01-23 22:18:10
1	☐	☐	☐	☐		Sub: espresense/devices/apple.1005.9-12/family:distance	1.6	☐			2022-01-19 22:21:27
2	☐	☐	☐	☐		Sub: espresense/devices/c47c8d6c8a43/den:distance	1.37	☐			2022-01-23 22:18:19
3	☐	☐	☐	☐		Sub: espresense/devices/c47c8d6c8a43/family:distance	3.06	☐			2022-01-19 22:21:26
4	☐	☐	☐	☐		Sub: espresense/devices/dc0d30470209/family:distance	1.93	☐			2022-01-20 16:58:32
5	☐	☐	☐	✓	2521	Dev: espresense|devices|den:distance Sub: espresense/devices/eddy.001122334445566778899-aabbccddeeff/den:distance Pub: the following Topic on Device command	2.24	☐	✓	☐	2022-01-23 22:18:17

Figure 98 Espresense JSON Data in Association Tab

Show Selected Chart

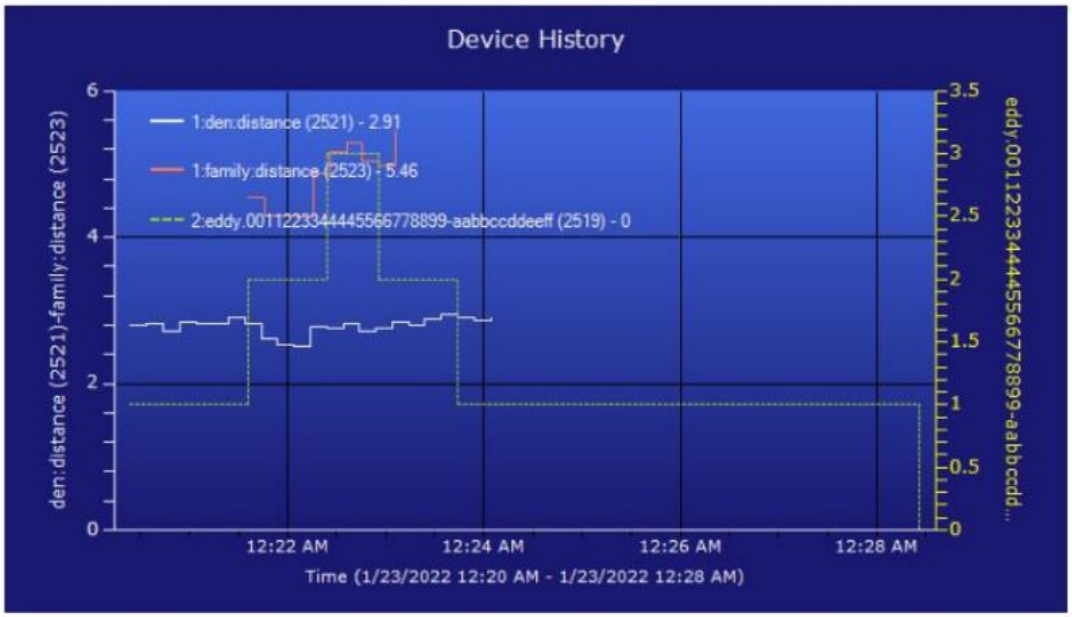


Figure 99 Espresence Room vs. Distance Visualization

11.7 GW1000

GW1000 is a USB-powered device that converts RF protocol to WiFi for Ecowitt and Ambient sensors. The integration of this device with mcsMQTT is based upon the work and source code provided by Homeseer member jim@beersman.com**.**

These sensors use different RF Frequency depending upon the region where they are being used so when obtaining the GW1000 assure that the correct frequency for your region. One source is Amazon at https://www.amazon.com/ECOWITT-Gateway-Temperature-Humidity-Pressure/dp/B07JLRFG24/ref=sr_1_2_sspa?crid=26R3CDJOTDDY1&dchild=1&keywords=gw1000+wifi+gateway&qid=1630618449&sprefix=gw1000%2Caps%2C646&sr=8-2-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUExMVBNMUI1TVlLWTRXJmVuY3J5cHRlZElkPUEwMDA0ODA0R1gyUUhJWlFNR0pSJmVuY3J5cHRlZEFkSWQ9QTA3MDEwNTczOVhYT1lESlVRTDNWJndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ==

Figure 100 GW1000 RF-WiFi Gateway

The GW1000 hardware is setup from the WS View App for Android or Apple. Other than this setup there is no further need for the App, but it can still be used if desired to view sensor data. Data remain local and does not depend upon the cloud.

To perform the setup in WS View the following steps are performed

1. Connect to local WiFi network using the WiFi button on the GW1000. See instruction manual that came with the GW1000 if necessary.

2. Device List. Publish to Weather Service. In this case it will be the "Customized" service that is being provided by mcsMQTT. See Figure 101 for context. Start on the Live Data view. Weather Services. Clike "Next" or "More" in upper right until the "Customized" option becomes available. About four clicks. The two fields that need to be edited are the Server IP and Port. These should match the NIC address and port that have been (or will be) setup on the mcsMQTT Local Page, GW1000 Tab. It cannot be 127.0.0.1, but needs to be the actual V4 IP address being used by HS. As an option, the update interval can be changed from the default. The Save button and the Finish button are then used to program the GW1000 to send data updates.

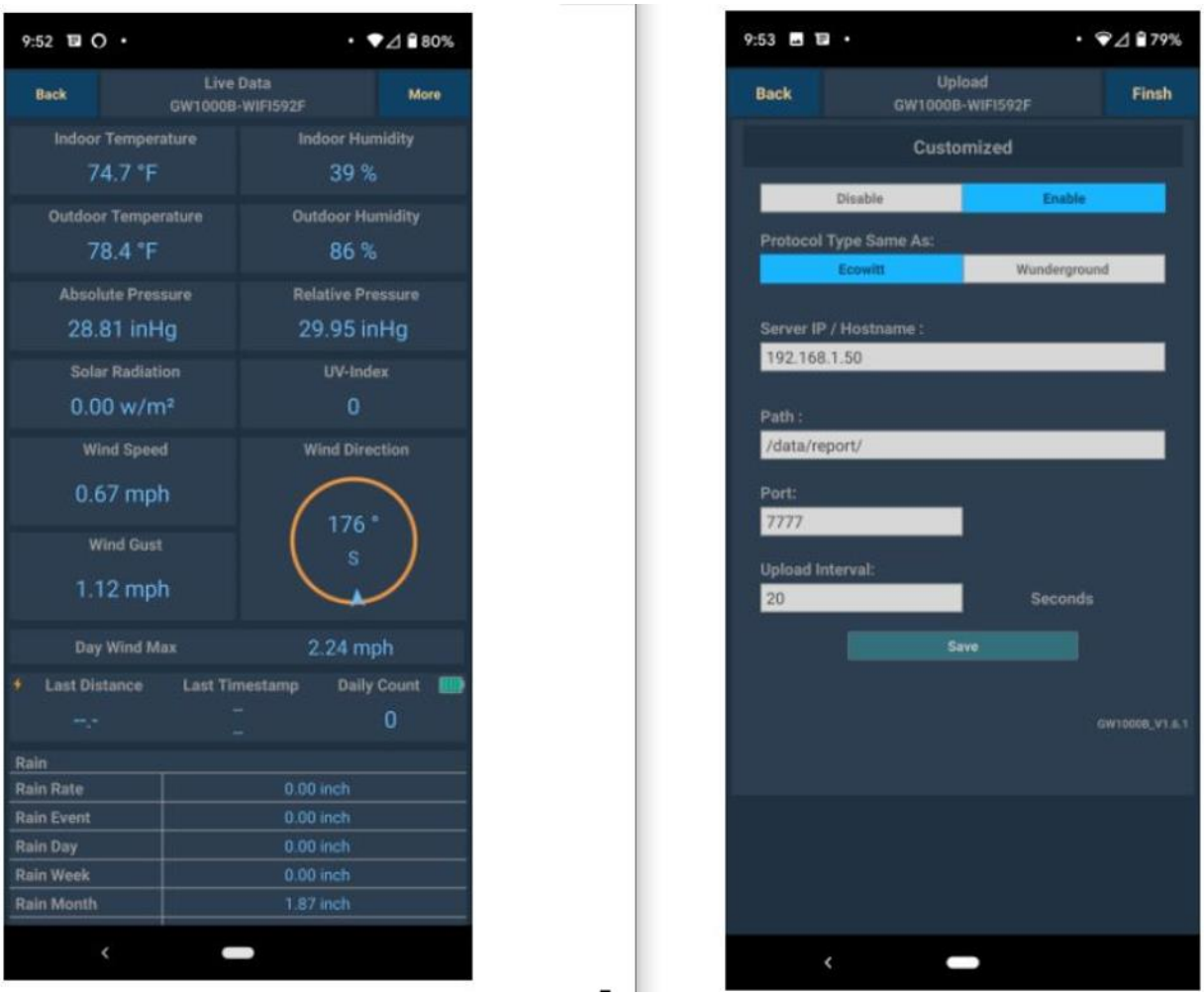


Figure 101 WS View Setup for GS1000

The corresponding setup with mcsMQTT is on the Local Page, GW1000 Tab. See Figure 102. The defaults are 127.0.0.1 to indicate single NIC and port 8080. If those are acceptable then the only setting needed to the radio to connect to GW1000.

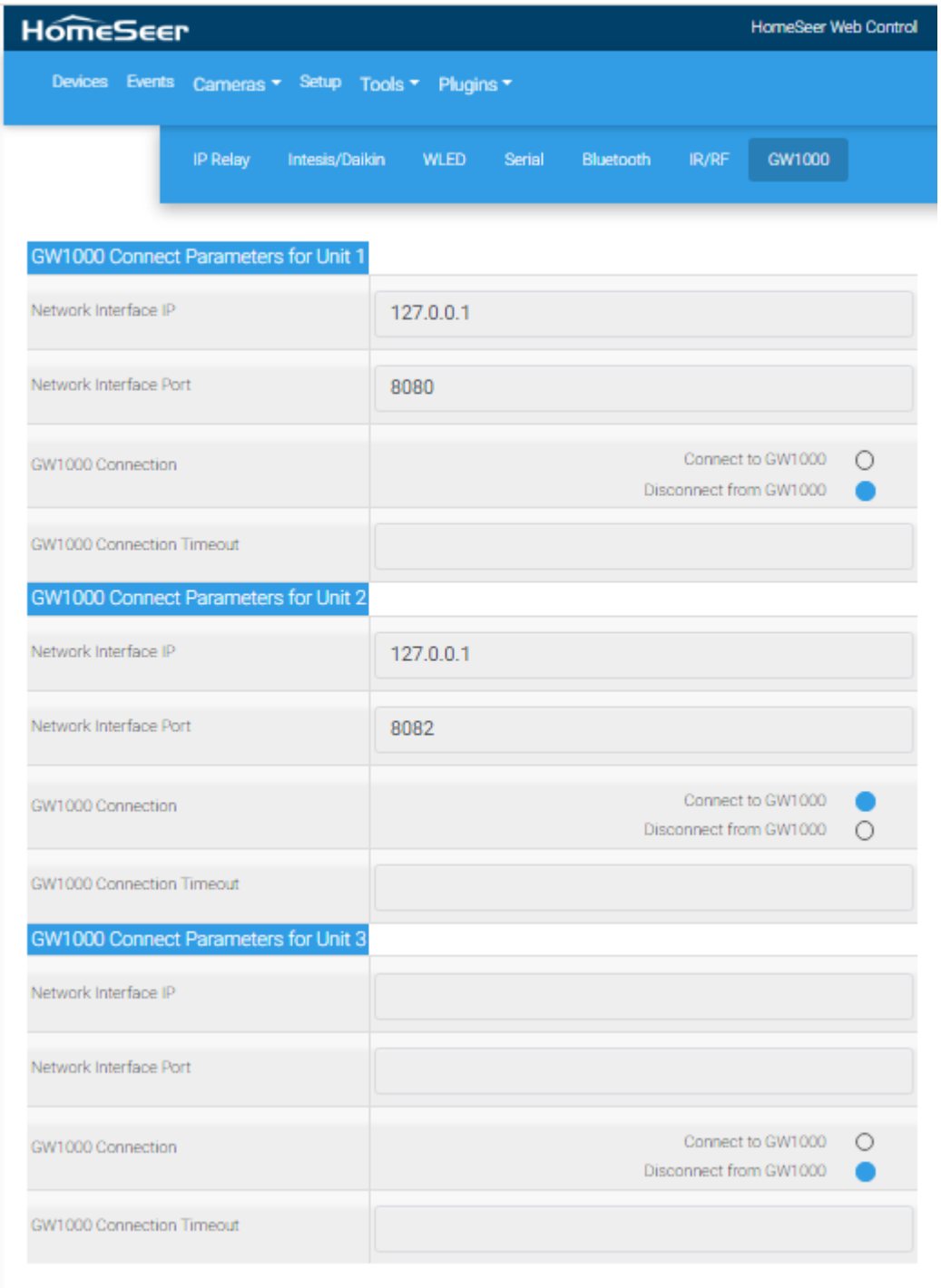


Figure 102 GW1000 mcsMQTT Setup

When both are setup then the GW1000 will be providing updates at the interval that was setup, mcsMQTT will populate the Association Tab, it will create HS devices and features for each sensor reading and will update these as new data has been received. An example is shown in Figure 103.

The connection to GW1000 can be enabled and disabled with radio connection. It is also possible to have mcsMQTT monitor the connection and if no data received for the specified timeout then mcsMQTT will establish an new connection.

The Device name being used, and the pseudo-MQTT Topic is the passkey that is contained in the GW1000 data stream. If other stations are providing data, then it becomes easy to distinguish one from the other.

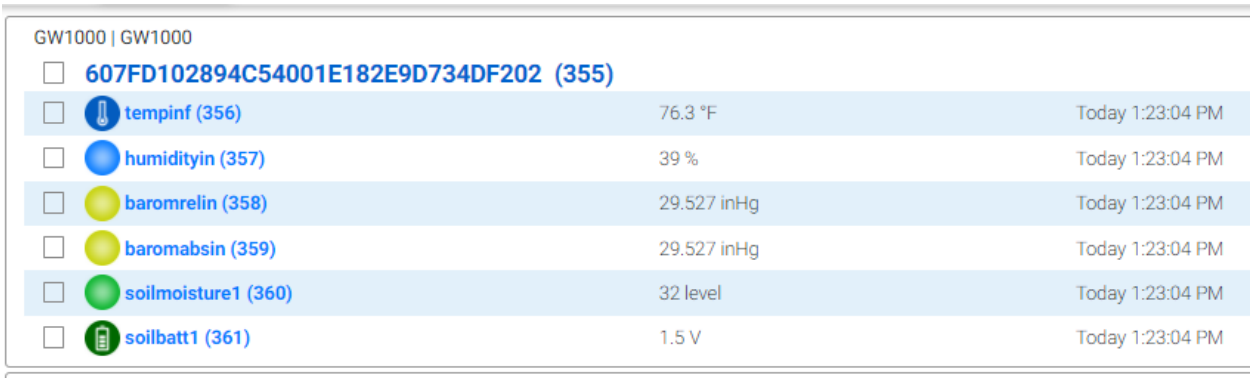


Figure 103 GW1000 Ecowitt/Ambient Sensors Viewed by HS

11.8 Epson Projector ESC/VP.net

Epson produces a family of projectors that are network connected and an API https://files.support.epson.com/pdf/pl600p/pl600pcm.pdf that can be used to provide control and status. Some detailed integration information is also available at https://www.epson.eu/en_EU/faq/KA-01...ents?loc=en-us

mcsMQTT provides an integration to the projector with HS3 and HS4. The setup starts with the Local Page, Epson Tab where the network address of the projector(s) is(are) entered along with the polling interval that should be used. See Figure 104. Status information from the projector is only provided upon request. This request is issued by the plugin following a command that changes a parameter and on the polling interval.

Epson does report events that the projector has been externally commanded such as from a remote. mcsMQTT initates a polling cycle when the IMEVENT message is received.

The connection to the projector is monitored. If it is lost for any reason then the plugin will attempt to reestablish it. Once establish again it will send the required ESP/V21.net message to initate communication on the new connection.

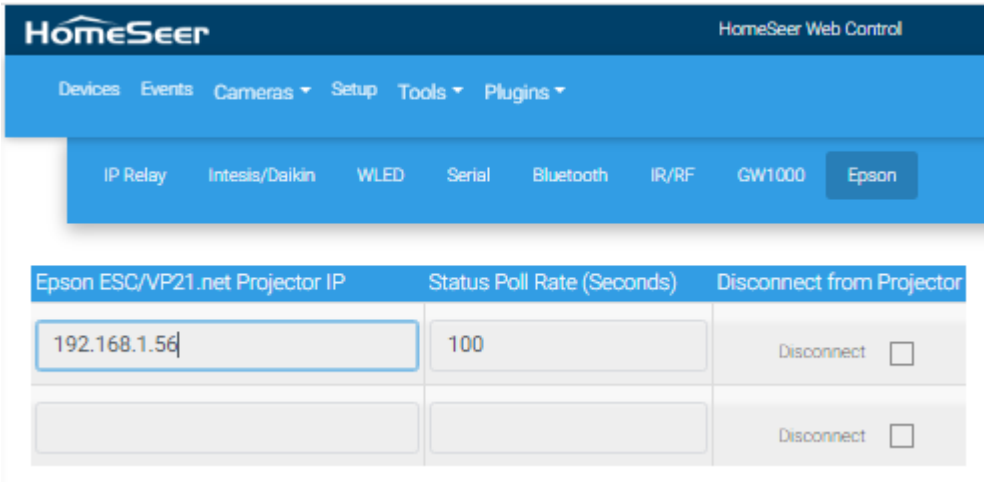


Figure 104 Epson Projector Setup

When the Epson network address is entered a default HS Device and set of Features will be added such as shown in Figure 105. User control from the HS Devices page or from HS Events can be used to command the projector. Status will be updated for each Feature based upon the response received from the projector.

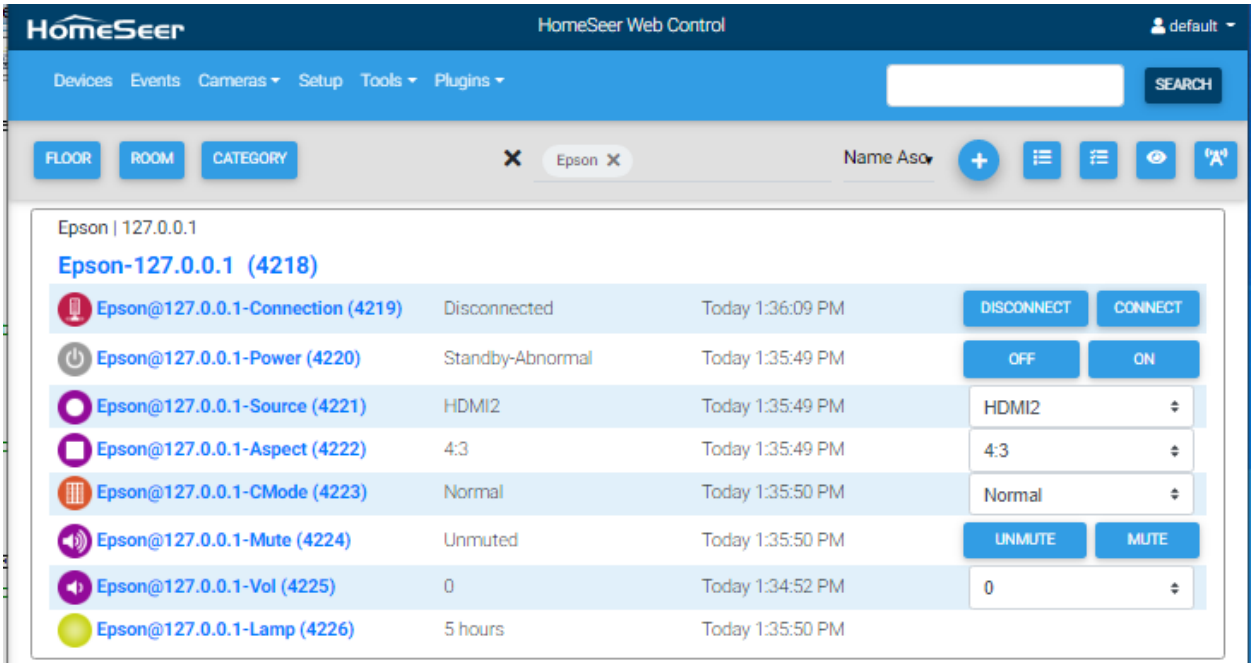


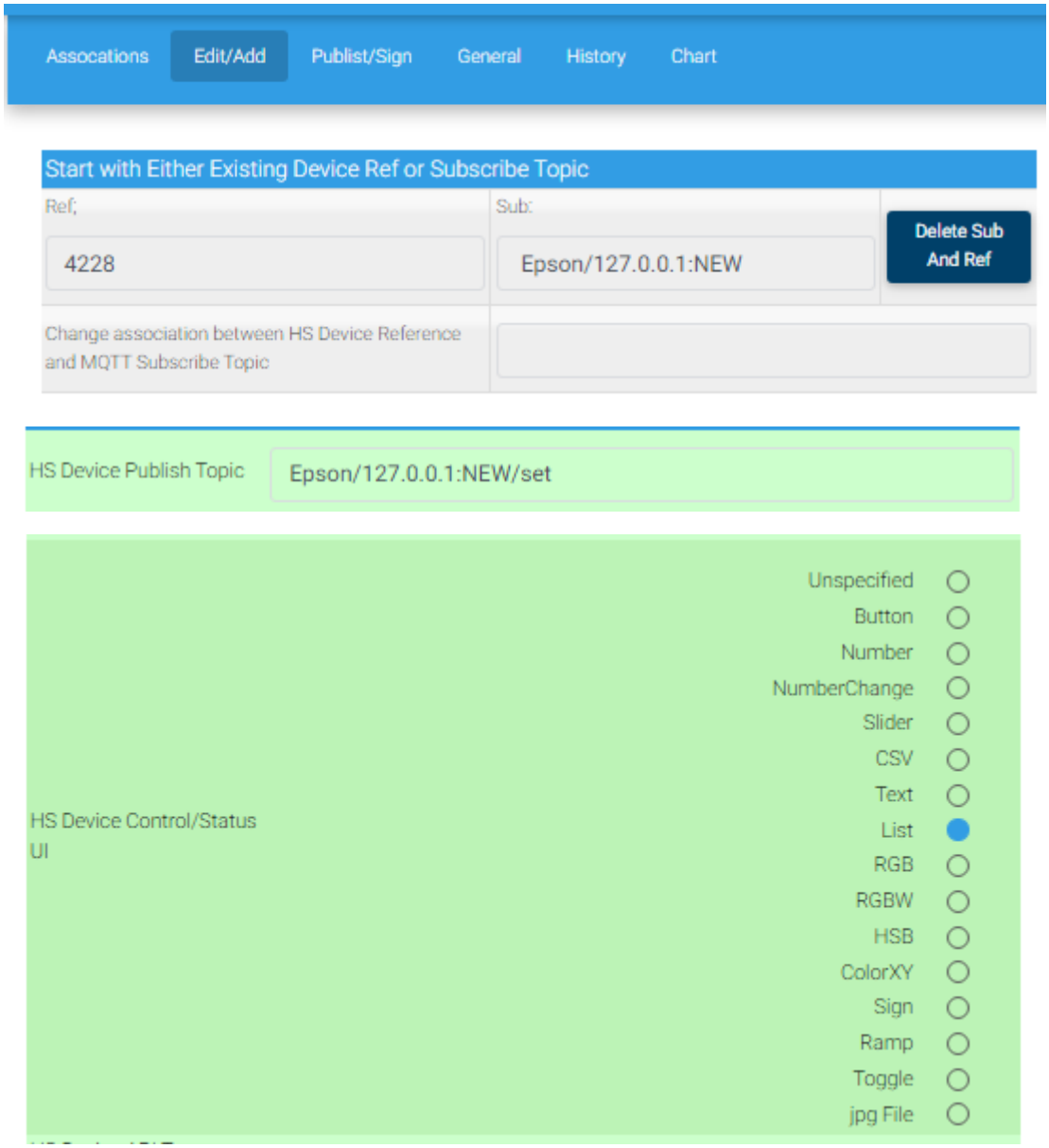
Figure 105 Epson Project Default HS Device and Features

While it is likely that the default set of projector properties will be sufficient for any user, the API does provide other information that could be of interest. To include another property from the projector, then start with the Edit Tab or the MQTT Page and create a Sub Topic that looks like the other Epson Topics, but ends with the API parameter that is being added. In Figure 106 is an example of a API parameter "NEW" which is not a real parameter, but is being used for illustration. A real property example is "AUTOKEYSTONE".

Once the Epson/127.0.0.1:NEW is entered, mcsMQTT will create a HS Device Feature (4228) and provide a table where other specifications can be entered. Since NEW is a parameter that can be controlled from HS, the Publish topic text box needs to be completed as shown in Figure 106. The Control/Status UI needs to be completed so the plugin knows the nature of the data that is being communicated.

If is is a hex digit-pair as is most common for the projector, then the type will be List and the subsequent VSP defined on the Edit Tab. Look at predefined Aspect topic in the Edit tab for an example. If the property is expecting something like ON/OFF as is used for PWR then it should be a Button. If it is an integer such as is returned for Lamp Hours, then it is a Number.

The Grouping Ref should be reviewed and changed to assure this new Feature is grouped under the Epson Device.


Assocations
Edit/Add
Publist/Sign
General
History
Chart
Start with Either Existing Device Ref or Subscribe Topic
Ref;
4228
Sub:
Epson/127.0.0.1:NEW
Delete Sub And Ref
Change association between HS Device Reference and MQTT Subscribe Topic
HS Device Publish Topic
Epson/127.0.0.1:NEW/set
HS Device Control/Status UI
Unspecified
Button
Number
NumberChange
Slider
CSV
Text
List
RGB
RGBW
HSB
ColorXY
Sign
Ramp
Toggle
jpg File

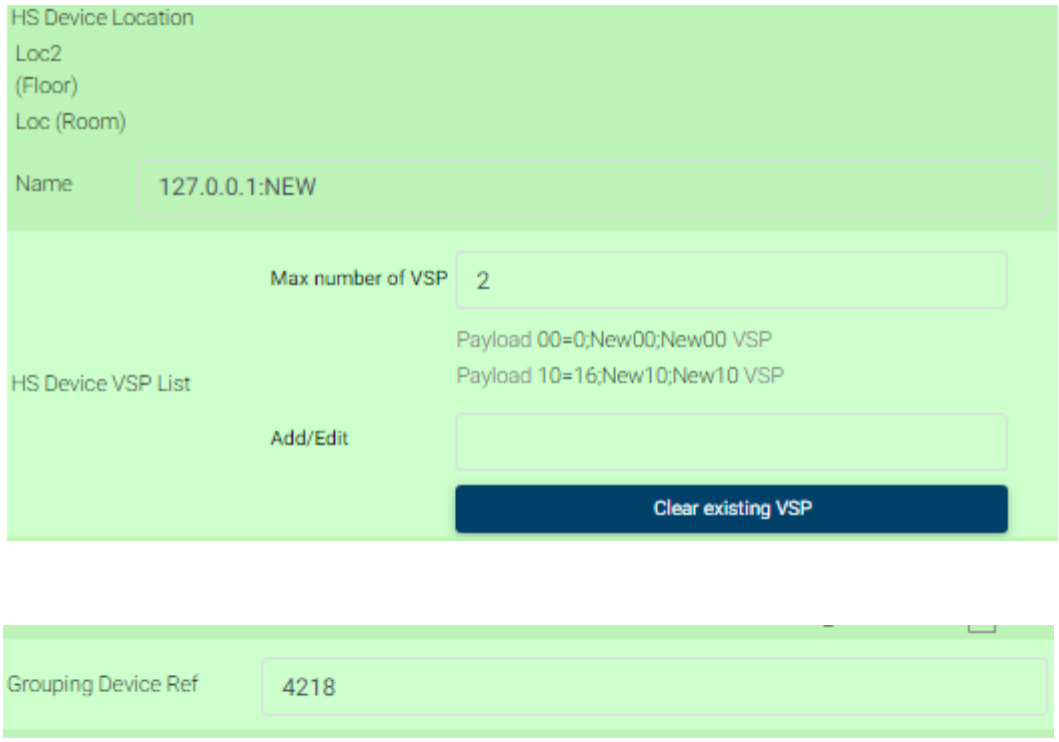


Figure 106 Augmenting Epson Command/Device List

The result of this is a new Feature such as shown in Figure 107. From this point the user is expected to rename the Feature and add the appropriate icon. This is done from t he HS Devices page in HS4 or DeviceUtility page is HS3.

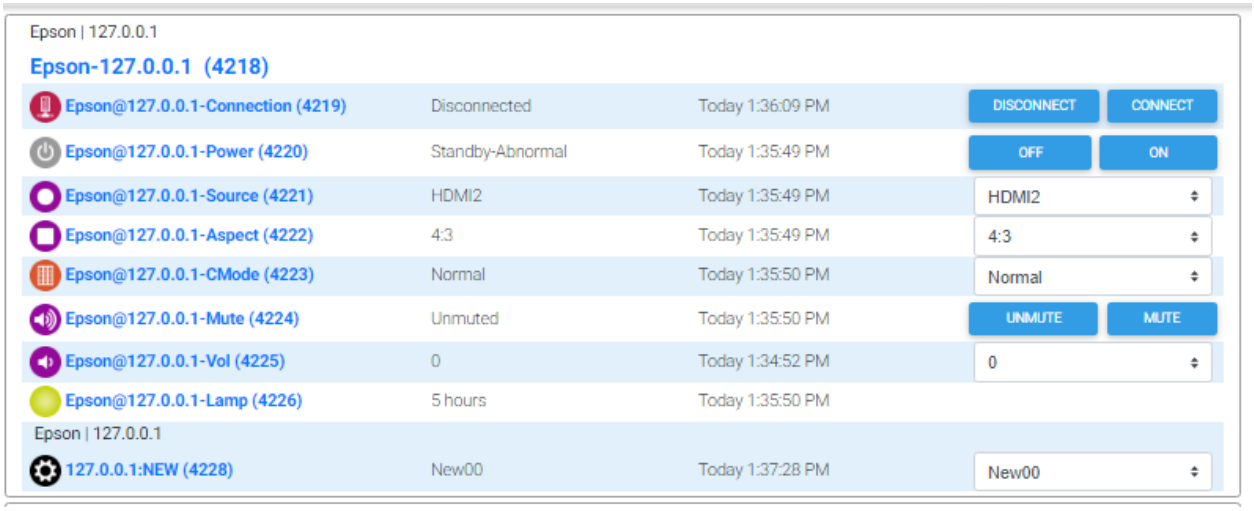


Figure 107 HS Epson Device Features after Augmentation

11.9 HS and Plugin Monitoring with Enable, Disable and Restart Controls

mcsMQTT accumulates the CPU utilization and other performance measures over each 60 second time interval and reports it in the pseudo-topic of HS. HS itself, any plugin, and any process name can be selected for "A"ssociation with a HS Device Feature. Once associated the DeviceValue will show the resource utilization such as percent CPU use over the prior minute. HS and plugin CPU monitors will have control available to Enable, Disable, or Restart HS or the plugin. Plugins that are not running will be shown as "Stopped" in HS with as DeviceValue of -1.

The selection of the desired measures is done from the Local Page, Resources Tab as shown in Figure 108. If other executable processors are to be monitored then the name of the process is also entered on the same Tab. Use of Windows Task Manager or Linux Top may be helpful to correctly identify the name of the process.

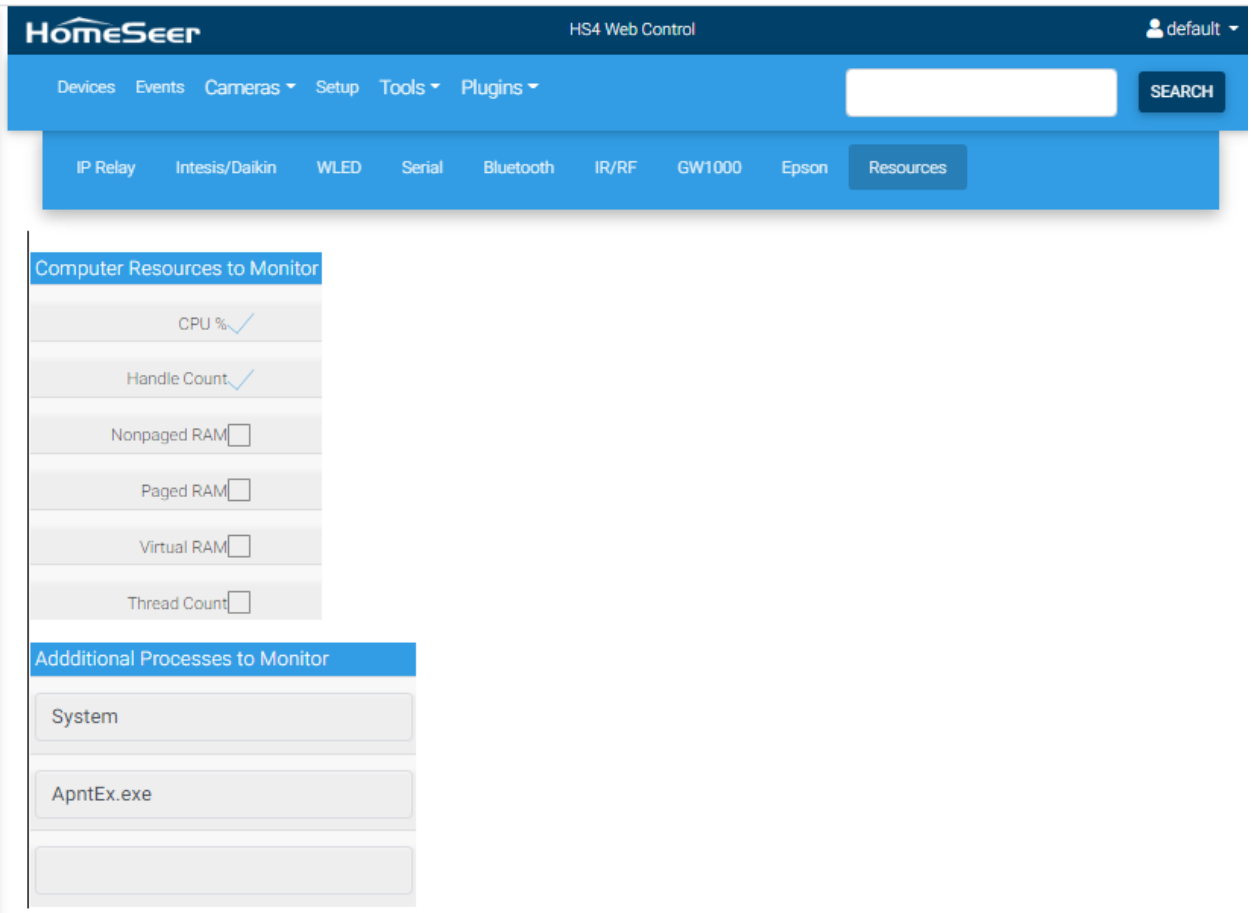


Figure 108 Computer Resource Monitor Selection

Figure 109 HS Pseudo-Topic for Monitor and Control of HS and Plugins

A sample of the HS pseudo-topics is shown in Figure 109. A sample of the associated Device and Features is shown in Figure 110. Most plugins have controls for Disable, Enable, Restart. HS and mcsMQTT do not have the Enable control because they need to be running for these controls to be used.

The HS/Plugin control devices will show the CPU use measurement. The other performance measures, if enabled on the Local Page Resource Tab and "A"ssociated on the MQTT Page Association Tab wil not have any controls presented in HS. They can also be selected for History data collection.

The Green vs. Yellow graphic reflects the status of the process in which a plugin or HS runs. A plugin that is not running will show as Yellow. In the case of HS3 plugins that have multiple instances there will

be one process shared by all instances of the plugin. It will be shown in HS with a Device Featue for each instance so each instance can be individually controlled, but only one process will be running. The process will remain active as long as at least one of the instances has been enabled.

The percent utilization is also a reflecton of the process. One that is not running will be shown in the status as “Stopped”. For those that are running the DeviceValue will contain the percentage being used over the last 60 second period.

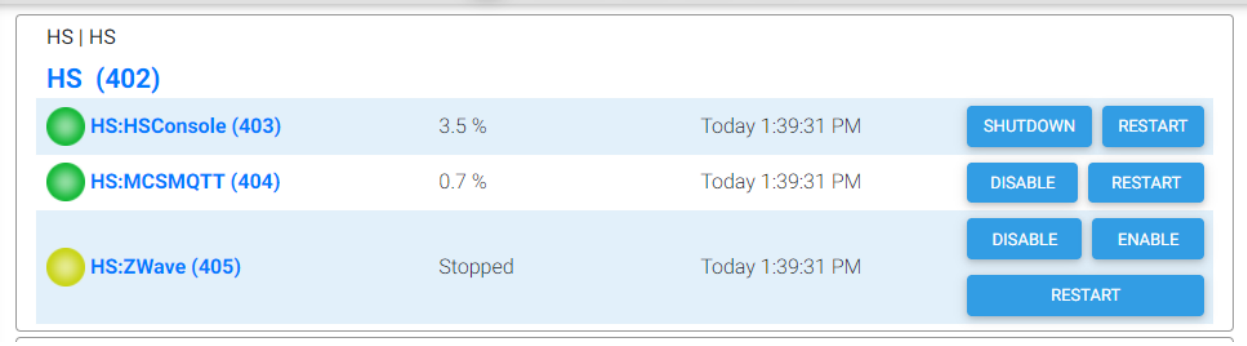


Figure 110 HS Device Features to Monitor and Control HS and Plugins

The operation of the Enable and Disable controls is similar to the controls available on the HS Interfaces or Plugins menu. They actually invoke the same operation as the controls on these pages. For the Disable it is an orderly shutdown of the plugin.

The Restart control is implemented as a Disable control, monitor for the plugin/HS process to disappear then invoke the enable action. In the case of mcsMQTT plugin a restart is achieve by internally doing an orderly shutdown and then killing mcsMQTT process. HS will observe mcsMQTT has disappeared and restart it. In the case of HS, the shutdown is done through the HS API. The HS restart is performed by a shutdown, waiting for the HS process to disappear, and then launch HS.

mcsMQTT identifies the plugins by using the HSPI_ filenames in the HS folder. It then attempts to correlate the filenames with the plugin names which are usually similar. For example, HSPI_SAMPLE.exe has a plugin name of “Sample Plugin”. The filename is needed for CPU utilization monitoring. The plugin name is needed for the Controls.

Unlike something like Windows Task Manager or Linux Top that attempt to show instantaneous CPU utilization, the collection and reporting interval for mcsMQTT is 60 seconds. The intention is not to capture short term spikes, but to capture steady state utilization.

The DeviceValue from any of these Device Features can be used as Event Triggers. The Controls can be used in Event Actions. For example, and event can be setup if a plugin is using more than 25% of CPU then Restart the plugin.

In the Linux case, HS provides tools to shutdown HS or to restart the computer. It does not provide a tool to restart HS. The Restart control for HS provided by mcsMQTT can fill this gap.

For those that keep historical trends then these Device Features can be slected for database storage in mcsMQTT or other providers and then charting for CPU use can be produced.

11.10 Hunter Douglas PowerView Gen2 & Gen3

Hunter Douglas provides a family of approximately twenty shades that are networked interfaced through a hub. The current hub firmware version is Gen3. Earlier versions are not supported by mcsMQTT.

The URL Path for Powerview for access to shades and room information will be either “home” or “api”, depending upon the equipment generation available. This is a setting on the Local Page, Shades Tab for PowerView. The local Hub IP address and the update interval to refresh HS status is setup at the same place as shown in Figure 111.

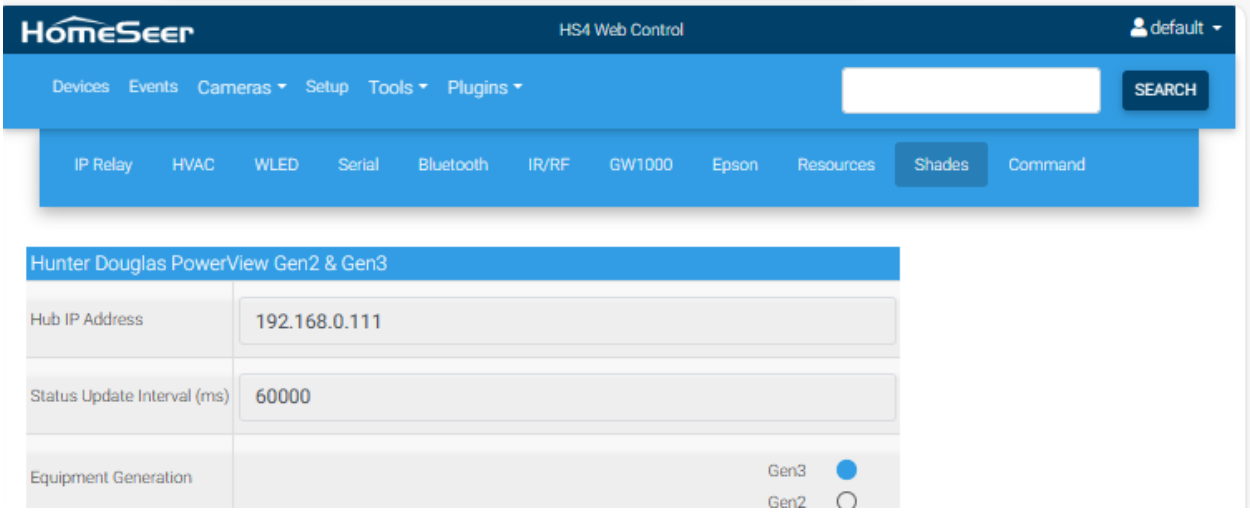


Figure 111 Hunter Douglas PowerView Setup

mcsMQTT provides the capability to moving shades to a specific position and reporting status of the position. Some models have a tilt capability which is also controllable. Some models have a secondary shade that can also be controlled. mcsMQTT will use the model information provided and create the appropriate set of Features.

A battery status will also be created. If line power rather than battery power is being used then the battery Feature can be removed by using the “a” column checkbox on the MQTT Page Association Tab to remove the association with HS. A Device is created for each room setup in the Hunter Dougles setup.

Figure 112 provides an illustration of the various Features setup within a room Device for the Gen3 equipment. Figure 113 shows the Room and Sceen-Group Features for Gen2.

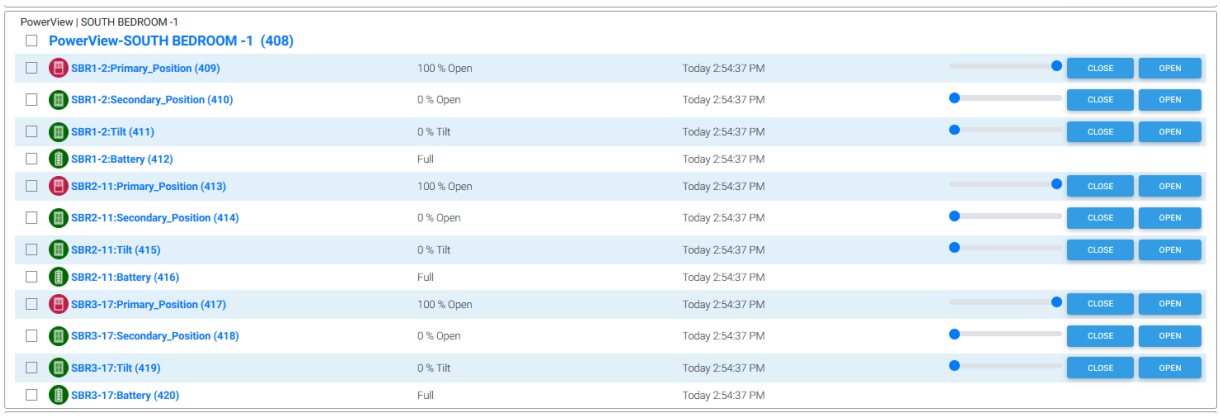


Figure 112 Hunter Douglas PowerView Gen3 HS Features

Figure 113 Hunter Douglas PowerView Gen2 HS Features

The Local Page, Shades Tab contains the setup for the Hunter Douglas Gen3. The IP is the network IP address of the hub. The polling rate is the periodic interval when status of shades position is queried. The status is also queries at plugin startup. HS Devices are created when both the IP and polling setting are setup.

11.11 Command Terminal

The Command Tab provides a mechanism to periodically run executable programs that provide a command line output. mcsMQTT will recognize the format of the output and convert it to JSON from where a MQTT pseudo-topic is created using the JSON payload.

In general, it is up to the user to selected from the MQTT Page, Association Tab the pieces of information that they desire to see in HS using the “a” checkbox.

At this time APC battery backup monitor apcaccess.exe is the application that has been implemented. As interest exists others can be added to the plugin.

The Updater install will include APCMQTT.exe in the bin\mcsMQTT subfolder. This application is intended to be run remotely such as in a RPi where the APC monitor has been connected. It will publish MQTT messages on the Topic apcaccess/IP/status where IP is the IP of the host computer. APCMQTT.exe is a native .NET/Windows application or a Mono/Linux application that should be started at boot of the host computer. For Linux this is normally systemctl, but other options are available. Examples of use of systemctl exist in this document. For Windows there are also multiple options to start an application at boot time.

The apcassess application will output to the console data of a format similar to below. APCMQTT will format this in JSON and deliver it as a MQTT message. mcsMQTT will run this application every five minutes.

```
APC      : 001,048,1088
DATE     : Fri Dec 03 16:49:24 EST 1999
HOSTNAME : daughter
RELEASE  : 3.7.2
CABLE    : APC Cable 940-0024C
MODEL    : APC Smart-UPS 600
UPSMODE  : Stand Alone
UPSNAME  : SU600
LINEV    : 122.1 Volts
MAXLINEV : 123.3 Volts
MINLINEV : 122.1 Volts
LINEFREQ : 60.0 Hz
OUTPUTV  : 122.1 Volts
LOADPCT  :  32.7 Percent Load Capacity
BATTV    : 26.6 Volts
BCHARGE  : 095.0 Percent
MBATTCHG : 15 Percent
TIMELEFT :  19.0 Minutes
MINTIMEL : 3 Minutes
SENSE    : Medium
DWAKE    : 000 Seconds
DSHUTD   : 020 Seconds
LOTRANS  : 106.0 Volts
HITRANS  : 129.0 Volts
RETPCT   : 010.0 Percent
STATFLAG : 0x08 Status Flag
STATUS   : ONLINE
ITEMP    : 34.6 C Internal
ALARMDEL : Low Battery
LASTXFER : Unacceptable Utility Voltage Change
```

```
SELFTEST : NO
STESTI   : 336
DLOWBATT : 05 Minutes
DIPSW    : 0x00 Dip Switch
REG1     : N/A
REG2     : N/A
REG3     : 0x00 Register 3
MANDATE  : 03/30/95
SERIALNO : 13035861
BATTDATE : 05/05/98
NOMOUTV  : 115.0
NOMBATTV :  24.0
HUMIDITY : N/A
AMBTEMP  : N/A
EXTBATTS : N/A
BADBATTS : N/A
FIRMWARE : N/A
APCMODEL : 6TD
END APC  : Fri Dec 03 16:49:25 EST 1999
```

APCMQTT.exe is expecting command line parameters of path to apcassess that is installed on the host computer and the IP of the MQTT Broker that it should be using to facilitate the communications. If the Broker is expecting a username and password then add it as the last parameter in format username:password. The following are examples of the two cases:

mono /etc/apcupsd/APCMQTT.exe apcaccess 192.168.0.100
mono /etc/apcupsd/APCMQTT.exe apcaccess 192.168.0.100 un:pw

The apcaccess/status Topic will be received and available in mcsMQTT MQTT Page, Association Tab from where selected items can be associated with HS Device and Features. Note that the raw data includes formatting so it may be advantageous to remove the formatting so the data can be stored in DeviceValue rather than DeviceString. Regular Expressions or Expressions on the Edit Tab of MQTT Page can be defined to remove the formatting. For example, look at Section 6.2 case #2 to remove the suffix. In these cases, the Control/Status UI should be selected to be a number rather than text.

12 Cloud

12.1 URL

12.1.1 Overview

The URL tab provices access to internet sites and local widgets using HTTP/REST GET, HTTP/REST POST, UDP, WebSocket/Webhook and TCP protocols. The returned data will be analyzed for XML or HTTP Querystring formats and automatically converted to JSON. JSON is further decoded and others are handled as text on the MQTT Page Association Tab. The setup options available are the URL to the site being accessed, the protocol to use, polling rate, and if appropriate any changes to the protocol headers. An example is shown in Figure 114.

The GET, POST and URL protocols support periodic polling of the site to obtain data updates. They can also be used to send data from HS Device page, HS Event Actions or scripting. The UDP, WebSocket and TCPIn setup a persistant socket connection so that event data can be reported through mcsMQTT to HS.

Received data is analyzed to assess if it is XML and if so, is converted to JSON for subsequent use. If it is POST data using HTTP QueryString format (i.e., key value pairs separated with “&”) it is also converted to JSON. Non-JSON data that does not fit these categories is treated as raw text.

If HTML data is retuned from the URL it will not be visible on the HS browser pages because the HTML changes the rendering of an existing HTML page, but it does exist so extraction or other operations are possible.

-

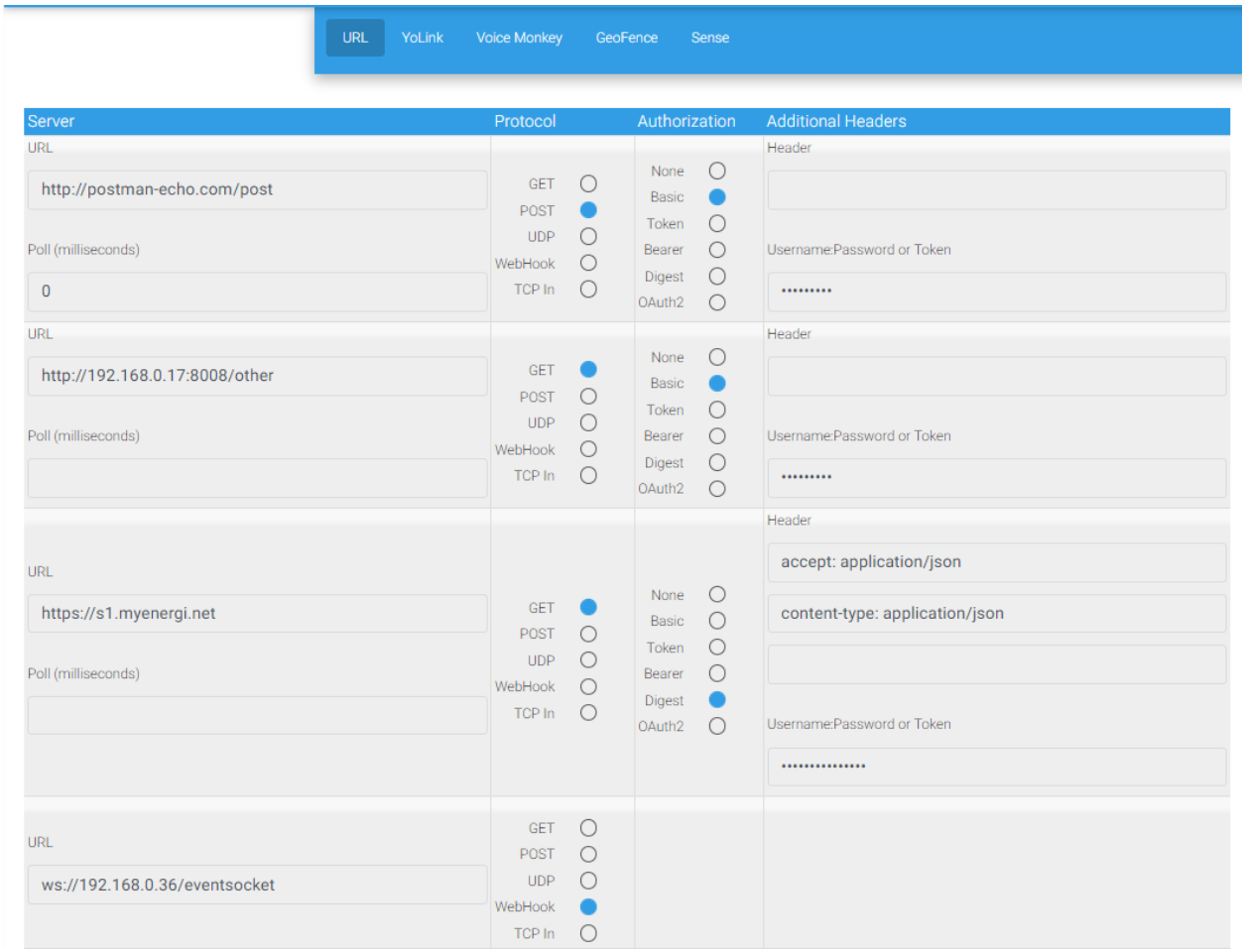


Figure 114 Sites for polling JSON data via HTTP

GET protocol uses REST where the data being sent from mcsMQTT is placed in the URL as the QueryString. POST protocol also uses REST with data being sent placed in the message body. UDP places data being sent from HS in Datagrams. WS negotiates a persistent communication socket with the URL setup. This is a listen-only protocol. TCPin is similar to WS, but the URL in this case is the NIC IP and socket port that will be opened for listening. The external widget would need to be setup to use the same if it desires to send data. The TCPIn URL can be defined as some port on 127.0.0.1 and in this case mcsMQTT will pick the NIC being used. This implies that a single NIC exists on the computer to assure know communication.

When the GET, POST or URL protocol is used then HS device is created through which the polling can be stopped or started and text can be commaned to be sent. See Figure 115. When sending data through a device the text box and Submit button are used. When sending data via Event Action the Send MQTT Message Event Action is selected and the user entry for the action will be Topic=Text. The Topic will start with URL/ and is visible in the Association tab as rows 12 through 16 of Figure 116 Pub text box. It is of the format URL/URL:Control/Send. For example

URL/mcsSprinklers.com-8080:Control/Send=Text to Send

In this example the port is separated from the address with dash rather than a colon so that confusion does not exist with JSON key (colon delimit)

The GET and POST headers use default of the following. KeepAlive and Timeout are not actual headers being transmit, but can be used to affect the connection persistence. Additional headers can be entered in the last column of Figure 114 or the default headers can be changed by entering the header key and value in the format key:value.

- `Content-Type:application/x-www-form-urlencoded,`
- `Accept:*/*,`
- `User-Agent:mcsMQTT,`
- `Accept-Encoding:identity;q=1.0,*;q=0`
- `KeepAlive:True,`
- `Timeout:5000`

.

Various authentication methods are employed across the web. mcsMQTT supports the most common ones. The Authorization column provides a radio for selection of the method employed by the server at the URL. mcsMQTT will either negotiate the authentication or it will include in the header the authorization information for the Token and Bearer techniques.

REST GET and POST protocols usually have the URL prefixed with case-insensitive HTTP:// or HTTPS://. UDP and TCPIn protocol does not have a prefix. Websocket/Webhook uses WS:// or WWS"// prefix. A site that is expecting direct TCP commands can be setup with the GET protocol using "TCP://" prefix.

Data that is returned is expected to be in JSON format and decoded into individual keys such as shown in Figure 116 . If XML it is converted to JSON. If not JSON then the entire message is treated as text or number. The "A"ssociate checkbox on the Association tab can be used to map the returned data into HS devices in the same manner that is used for associating MQTT data. If a transformation is needed such as KPH to MPH values then the Edit tab Expression textbox can be used.

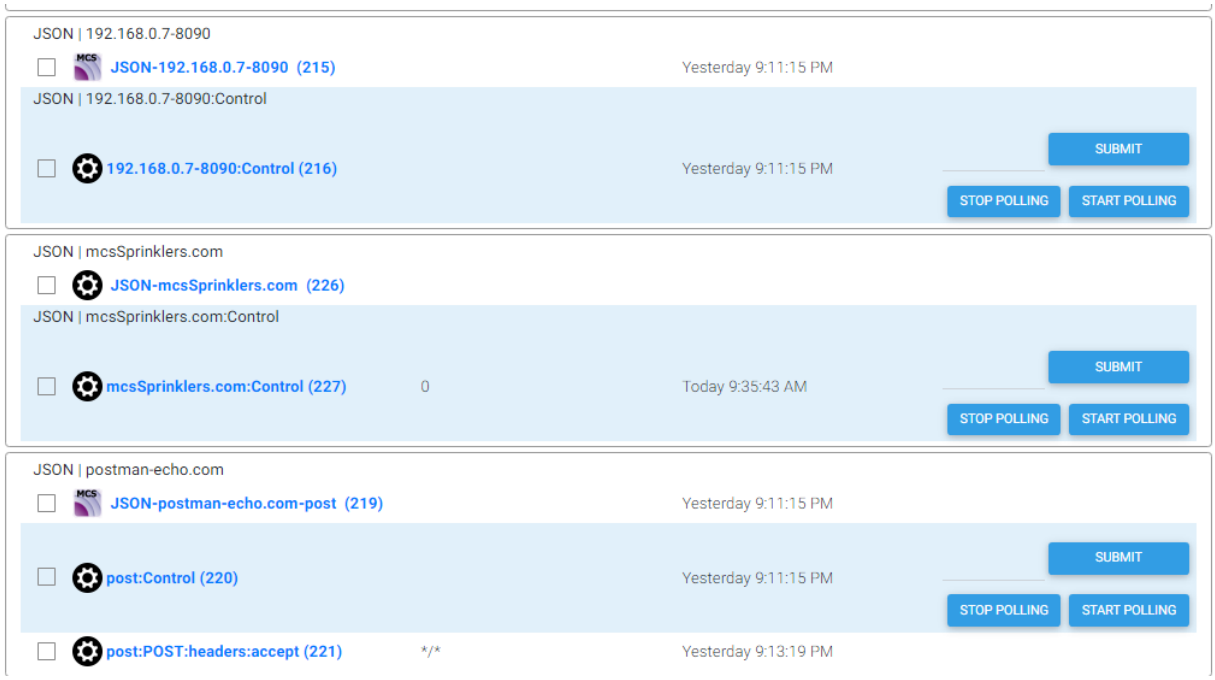


Figure 115 REST and UDP HS Device Interface

9	☐	☐	☐	☐		Sub: URL/127.0.0.1-8080:TCP:dateutc	2021-09-02 20:42:25
10	☐	☐	☐	☐		Sub: URL/127.0.0.1-8080:TCP:stationtype	GW1000B_V1.6.8
11	☐	☐	☐			URL/127.0.0.1-8080:TCP	
12	☐	☐	☐	✓	270	Dev: URL|127.0.0.1-8080:Control|127.0.0.1-8080:Control Sub: URL/127.0.0.1-8080:Control Pub: the following Topic on Device command URL/127.0.0.1-8080:Control/Send	
13	☐	☐	☐		269	URL/127.0.0.1-8080	
14	☐	☐	☐	✓	268	Dev: URL|postman-echo.com|post:Control Sub: URL/postman-echo.com/post:Control Pub: the following Topic on Device command URL/postman-echo.com/post:Control/Send	
15	☐	☐	☐		267	URL/postman-echo.com/post	
16	☐	☐	☐	✓	266	Dev: URL|mcsSprinklers.com:Control|mcsSprinklers.com:Control Sub: URL/mcsSprinklers.com:Control Pub: the following Topic on Device command URL/mcsSprinklers.com:Control/Send	

Figure 116 JSON Topics in Association Tab

12.1.2 URL Base and Endpoints

A server may have multiple endpoints that all share the same authentication methods. In this scenario there should be one URL IP setup on the URL tab and that will be the IP of the base URL. For example, if a server supports commands and status where the command URL is http://myServer/command and the status URL is http://myServer/status then http://myServer would be the base IP used to to establish the link between the server and a HS device.

The HS Device will show a text box for parameters to be entered to route to this server. If the parameter starts with "/" or with "?" then they will be appended to the IP to form the full URL that will be sent to the server. For other cases, except the publist, the parameters will be considered to be the data send as part of a POST request. GET requests have all their parameters on the querystring that will start with "?".

The publist needs to be used for the case where both the endpoint of the URL and the posted data need to be sent. The publist is a text file located in the \data\mcsMQTT subfolder that contains both the full URL and the data to be posted. It can be generated from the MQTT Page, Publist Tab or it can be done with any text editor. Each line of the publist file is a text list that uses "=" to separate two parts of the

line. The first part can be a local replacement variable name, a MQTT Topic, or a URL. The replacement variable is a convenience so often used text can be defined once in the file and then applied multiple times. A line that defines a replacement variable has the syntax of $$xxxx:=yyy where xxxx and yyy can be any text strings. The Publist Tab editor of the MQTT Page has provisions of xxxx being 1, 2, 3 or 4; but any text can be used for xxxx when making the file with a text editior. The definition of a local replacement variable is an option that is not required.

In this context of definig multiple URL endpoints, the left side of the "=" will be the URL if not a replacement variable definition. Using the same example of http://myServer as above a publist to send a command to turn device 123 on and 456 off could be:

URL/http://mcsServer/command={"device":"123","state":"on"}
URL/http://mcsServer/command={"device":"456","state":"off"}

The use of the publist is designated with the parameter in the HS Device control text box ending with ".pub" such as "myServer.pub" where the file will reside at subfolder \data\mcsMQTT\myServer.pub.

The publist is also available as an event action. A publist can send using HTTP or MQTT protocols. The user of HTTP is indicated with the line starting with "URL/", "GET/", "POST/", "PUT". Other starting text will result in a MQTT publish. "URL/" will use the method setup on the URL tab of the Cloud Page. "GET/", PUT" and "POST/" will use the GET, PUT and POST methods respectively. For these HTTP methods the specified URL must have the base URL defined on the Cloud Page, URL Tab. Using the same example as above, if the POST method was to be explicitly requested rather than using the method setup on the URL tab then the following would be used:

POST/http://mcsServer/command={"device":"123","state":"on"}
POST/http://mcsServer/command={"device":"456","state":"off"}

If the data in the publist contains a "=", other than the one used to separate the Topic from the Paylaod, then it should be escapted with a backslash such as "\=". It will be sent without the backslash.

A third method to use the publist is for the case where periodic polling has been enabled on the Cloud Page, URL tab for the server. If the endpoint entered ends with ".pub" then mcsMQTT will get the URL information from the specified publist file.

In addition to the publist, a single endpoint can be controlled from a HS Device Feature that has been associated with data received from the URL. For example, data received from xyz.com using the GET method will show up in the Association table as URL/xyz.com.GET:xxx where xxx is JSON key. When URL/xyz.com.GET:xxx is associated to HS then the Publish Topic can be defined to send data to that endpoint in a manner similar that is done with publist. For this example, "PUT/http://xyz.com?id=1234" could be entered for the Publish Topic and Payload Template setup with JSON format such as {"move":$$VALUE:}. This will send a PUT message to http://xyz.com with query parameter of id=1234 and body with JSON move key and value received from HS.

To send to a URL from a non-plugin device that has been associated the pub topic will contain the URL that is setup on the URL Tab of the Cloud Page which will be typically a POST method. The publish payload template will contain the post data and can contain replacement variables. For example, if URL

https://someserver.com was setup as the URL and message to be sent is JSON with ref number of HS Device and its value, the following would be used

Publish Topic – URL/https://someserver.com or URL/someserver.com

Publish Payload Template – {“ref”:$$REF:,”value”:$$VALUE:}

12.1.3 oAuth2 Authentication

When the authentication method being used by the server is oAuth2 then a second URL is involved in the authentication process. This URL as well as the data payload being sent for authentication is entered on the two text boxes provided on the row where oAuth2 authtication is selected.

Typically, the authentication URL will contain /oauth such as https://api.flumewater.com/oauth/token. The data payload is JSON and will contain the grant type, client and secret tokens and other information required by the server. An example is:

{"grant_type": "password","client_id": "ABC","client_secret": "DEF","username": "myEmail@emailserver.com","password": "xxx"}

A successful authentication will result in a token that can be used in subsequent access and information about expiration and method to renew the token. This information will be stored in mcsMQTT.ini in an encrypted format for use when later needed to access the base URL.

The user does not need to be concerned by the token except in cases where a failure is reported and visible on the MQTT Page, Association Tab related to the abilty to access the server. If access failure then the authentication credentials need to be reviewed.

The token is typically formatted in a standard manner called JSON Web Token (JWT) and will contain information that may be needed in subsequent endpoints of the URL. The API for the server will document the endpoint requirements and the key that can be obtained from the JWT token to satisfy those. They are retrieved by mcsMQTT using replacement variable $$JWT:(key):. For example, if “user_id” is needed in the endpoint or the URL (or in the data payload) then can be references such as:

URL/https://myServer/user/$$JWT:(user_id):/status

While it should not be needed it is also possible to get access to the JWT token itself using $$JWT:. The $$JWT replacement variable is only valid in the context of Cloud Page URLs that have been setup for oAuth2 authentication method. If used elsewhere then an empty string will be returned. If the key is not in the JWT token then an empty string will also be returned during the replacement.

mcsMQTT will monitor the expiration date of the authentication token on a daily basis and send a request that the token be refreshed on the day before it expires. If for some reason mcsMQTT is not able to refresh it then a new token can be requested when needed.

WebSocket and Webhook (TCP Listener)

Websocket and Webhook protocols open socket listeners on the HS computer at a specified port. When a packet is received, it is converted to a MQTT pseudo-topic and payload that can be viewed on the Association table and associated with HS Device Feature.

If there is special processing that is to be done or if a response to the received message is needed on the same socket on which it was received then a script can be specified to provide the response text as shown in Figure 117. If this processing is not needed then the script name does not need to be entered.

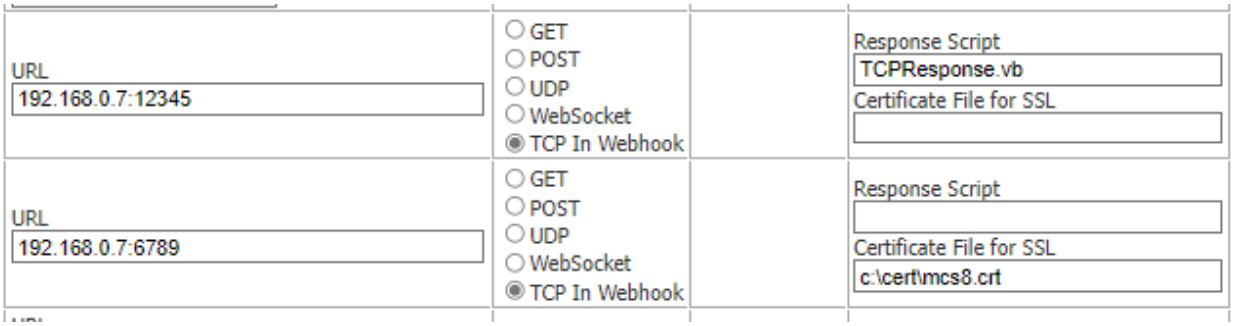


Figure 117 Listening Sockets Script Option

The script will be a Function with procedure name "Main". It should expect two parameters. The first is the URL that provided the packet of data and the second is the data in UTF8 string format that was received.

The script will return the text that should be returned on the socket or a null string if no response is desired.

As an example the following script will return the URL and received message if the message is "ECHO" otherwise it will return a null string. mcsMQTT will return the text from the function in the first case on the socket. It will return nothing in the second.

```
Function Main(parm as object)
     Dim URL as String = parm(0)
     Dim Data as String = parm(1)
     if Data = "ECHO" then
         Return URL & "=" & Data
     else
         Return ""
     end if
End Function
```

If the TCP Listener needs to support a secure SSL connection then a certificate needs to specified as shown in Figure 117. If an unencrypted communication is being used then the SSL textbox should be left blank. Generation of certificates is described in Section 10.2.3.

12.2 Voice Monkey

Voice Monkey provides a means to communicate with Echo devices from HS. There is a setup to be performed that involves:

1. registering with Voice Monkey,
2. enabling Voice Monkey skill in Alexa App, and
3. entering the skill's access token and secret token on the mcsMQTT Cloud Page.

After setup then HS event acitons are used to speak the Text, show Video or show HTML page.

Voice Monkey setup of access tokens is done on the same Tab as the Serial setup as shown in Figure 120. The tokens become available on the dashboard when a free Voice Monkey account is requested via Voice Monkey - Sign In (https:// app.voicemonkey.io).

This account provides ability to link the Echo Voice Monkey skill with a user's Amazon account. This skill provides the ability to do Text-To-Speech to Echo devices as well as pictures, videos and HTML pages to Echo Show devices.

The theory of operation is that a mapping is performed using Echo Routines between an Echo Device and a corresponding Voice Monkey Routine. When the Voice Monkey Routine is run the Text-To-Speech, video, image, or HTML page is delivered to the mapped Echo Device. The Voice Monkey Routine is run when the Routine Id is used in mcsMQTT event action or MQTT message.

The Voice Monkey Routine to Echo Device mapping is done using the Alexa App run on IOS or Android. Prior to this the Voice Monkey account, via a browser, is used to define the Routine names. The Routine names can be defined all at once or can be done at anytime before the Routine name is used in the mapping operation in the Alexa App.

Routines have a name and have an Id. The name is entered by the user in the format "Speak EchoDevice" in both Voice Monkey and Alexa App. The Id is the same as the lower-case name, but with a hyphen rather than a space (e.g., speak-kitchen). The Id is used in the mcsMQTT event action and MQTT topic.

The process using Voice Monkey and browser will result in a set of Monkeys which are the same as the Routine names as show in Figure 118.

Then naming convention is the same as is being used by Jon00 in his Voice Monkey integration with HS. What is important is that whatever convention is used there must be a relationship setup between name being used, the Echo Device, and the Routine name. Using the outlined naming convention assures the relationships will exist.

The Voice Monkey name is referred to as a Monkey. In the API Version 2 format it is "device". It is used to form the Routine Id that is used by mcsMQTT in Event Actions and MQTT Topics and becomes the trigger to cause Voice Monkey to route a payload to the Echo Device. One Name/Monkey/Device defined in Voice Monkey can be used multiple times in Alexa App Routines so that multiple Echo Devices can receive the same payload with a single Event Action or MQTT Topic.

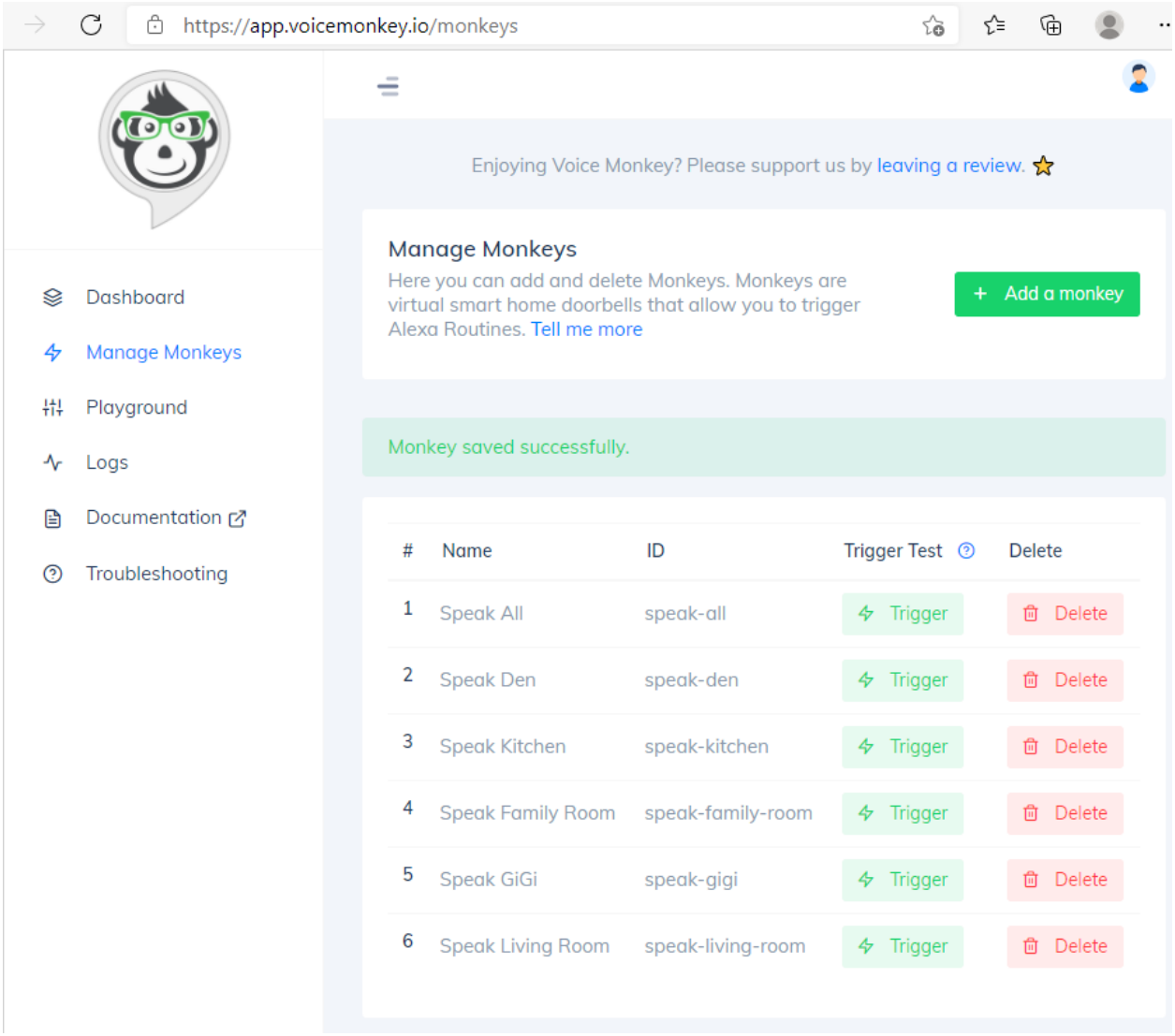


Figure 118 Voice Monkey Routines Seteup

In the Alexa App a mapping is done for each Echo Device with the result shown in Figure 119.

1. Start with Routines (from more menu in lower right)
2. Use + icon in upper right to add a new Routine
3. Enter into “Enter Routine Name” textbox “Speak <EchoDevice>” where <EchoDevice> is the name given in Alexa to the Echo Device. When done use the “Next” prompt in upper right.
4. Click “When this happens”, on new screen select “Smart Home”. On next page there will be a listing of the Routines that were setup previously in Voice Monkey browser. Select the Routine name that is the same as the EchoDevice being mapped. The screen will echo “When ‘Speak <EchoDevice> is pressed’”. Select Save.

5. Click “Add Action”. Select Skills. From Your Skills select “Voice Monkey – Routine Triggers & Text To Speech”. Screen will update with “Alexa will open Voice Monkey Voice Monkey – Routine Triggers & Text To Speech. Select Next in upper right.
6. On bottom of screen “From “ click on “Choose Device”. Select the <EchoDevice> that is being mapped. Click Save in upper right.

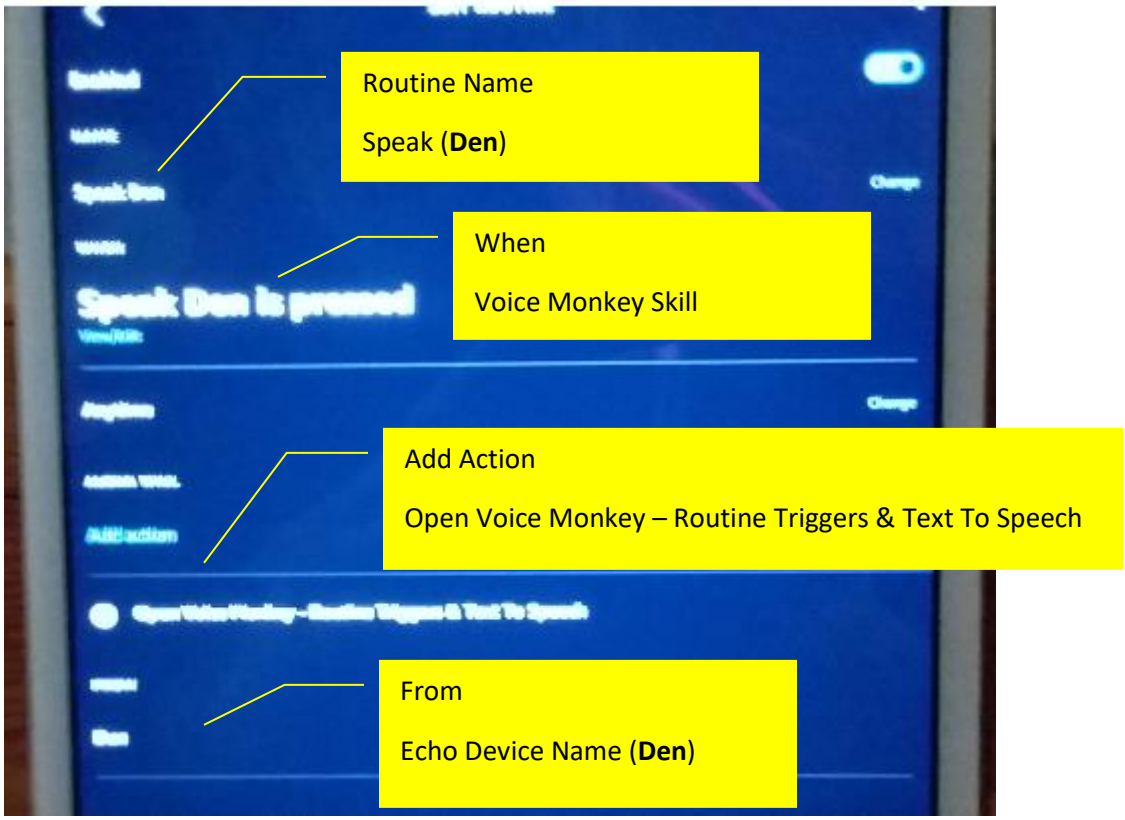


Figure 119 Alexa App Edit Routine for Voice Monkey

There is a special case Routine name “Speak All” that is used as the “When” part of the Routine definition. There can be multiple Routine names that use “Speak All”. This has the effect of sending the same text, video, image or HTML to multiple Echo Devices with a single Event Action or MQTT Topic.

There are two benefits. One is to reduce the size and labor of defining HS Event Actions where the announcement is to be sent to all Echo Devices. The second is that there is a limit of 30 requests to Voice Monkey per minute. If one has 30 Echo Devices, for example, then only one message can be sent per minute. For 15 then it would be two, etc.

mcsMQTT maintains a queue of Voice Monkey requests that run in a separate thread. It manages the queue so that no more than 25 requests to Voice Monkey are made in a rolling 55 second window. If there is heavy request rate then it is possible that there will be a delay. If the throttling was not done then Voice Monkey would enforce a 15-minute lockout if the 30 per minute was exceeded.

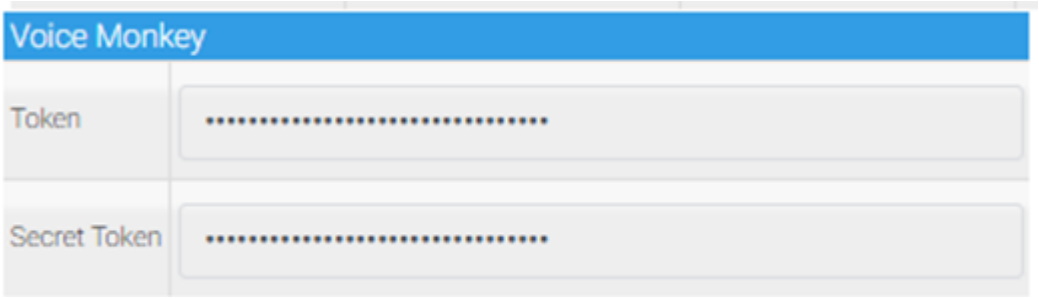


Figure 120 Voice Money Token Setup

The Voice Monkey event action defaults to Text. To show Video of HTML pages then JSON is used to specify the details. Section 5.3 contains more details on the use of the event action.

When using Voice Monkey via MQTT message, the full capability of the Voice Monkey V2 Announcement API is exposed in the MQTT Payload. Refer to https://voicemonkey.io/docs for a complete description of the announcement optioins. The MQTT Topic is "voicemonkey". The options consist of `image", video", "video_repeat", "voice", "no_bg","media_width, "media_height", "media_scaling", "media_align", and "media_radius with mandadory keys device and text` In the JSON payload.

12.3 Yolink

YoLink has make available an integration API to interact with its cloud server much like their smartphone apps is able to provide the UI for their devices. There is no API provided or hacked at this time to interact directly with the YoLink hub. Only access via the YoLink server is available.

YoLink provides the API to support partner product integration or individual user intergration. mcsMQTT was submitted and approved to have a partner account access. This is the mechanism by which mcsMQTT integrates with HS. As a partner, via mcsSolutions, mcsMQTT has visibility of the state of any device identified for mcsMQTT management. It does not have visibility into a user's account. In the aggregate it has visibility into all the devices being integrated via mcsMQTT. This is typical of Internet-based servers where data is collected, but the data is not associated with any individual.

The user has two options to identify the YoLink devices that mcsMQTT will integrate. One is to use the list of devices that were setup on their account. This is done with the "Get All Devices …" button. The other is to identify a YoLink device with its 32-character QR code. A smartphone QR scanner can provide this. It is entered on the YoLink QR code table. mcsMQTT will inform the YoLink cloud server of this device being integrated via mcsMQTT. The YoLink cloud server will acknowledge and, in the future, will provide notification of any state changes fo the device. See Figure 121. In this figure it shows the first four where the QR code was entered. The last row shows a device unique id which is signified by encasing it in "[" and "]".

The YoLink hub devices are typically not returned in the list of devices in the YoLink account. If HS monitoring of the hub is desired then it will be entered with the QR code.

Note also that there are two hubs specified in this setup. This is the scenario where one HS instance can monitor YoLink devices at two residences or otherwise widely separated locations. In this scenario the radio for multiple mcsMQTT should be selected on both mcsMQTT instances and the same QR devices or account list set of devices should also be setup on both.

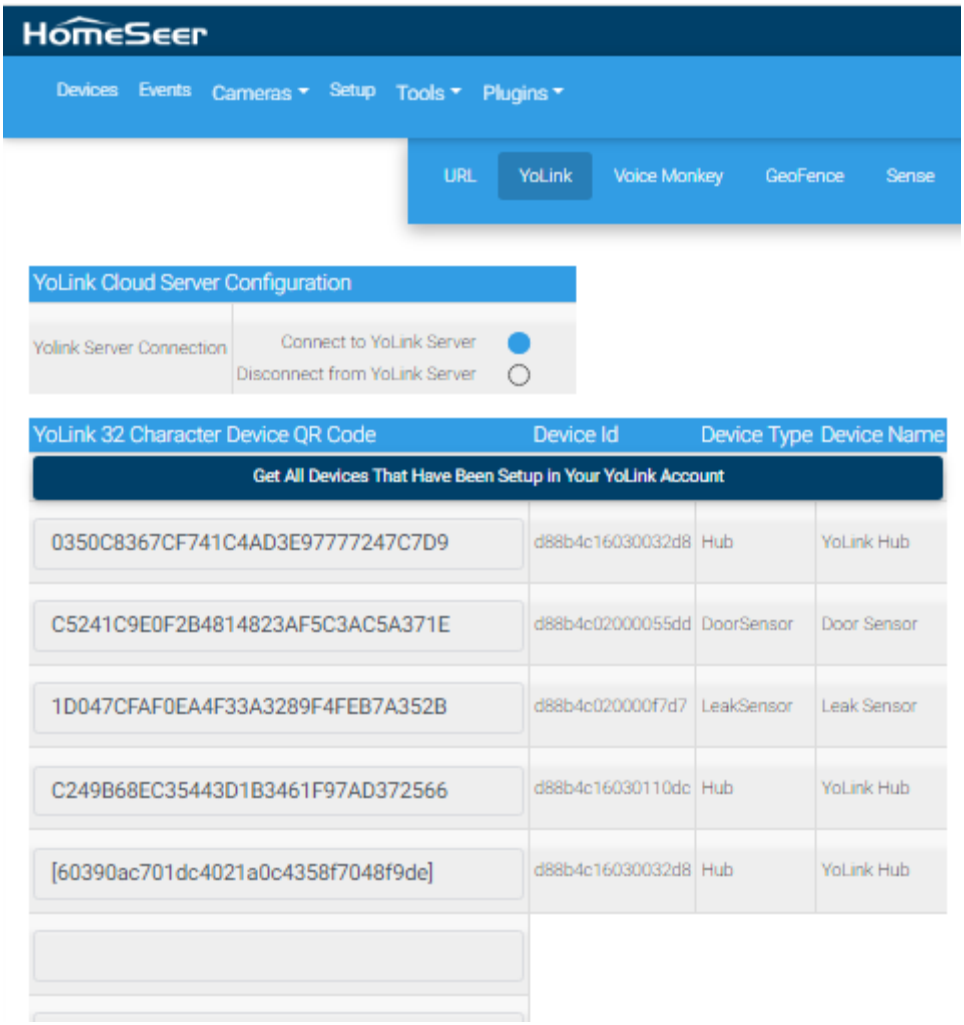


Figure 121 YoLink Device QR Code Entry

When getting YoLink devices from a user's account an oAuth2 process is invoked where the user logs into the YoLink site and grants access to the devices in their account to the plugin. This screen is presented when the "Get all Devices…" button is clicked and shown in Figure 122. Since this browser page is being serverd by the YoLink server, mcsMQTT has no visibility into the user's account other than the listing of devices that are on the account that are subsequenty delivered by the server once access approval is granted. This is the same as when the QR codes for the devices are individually added. The account that these devices were setup is not visible to mcsMQTT.

The oAuth2 process directs the browser to the YoLink server and then once approval has been granted the YoLink server redirects the browser to an externally-visible internet site. The assumption being made in the plugin implementation is that the redirection will be back to the same IP as where it started. This means that use of the "Get all Devices…" button can only be done from a browser that is running on the same computer as HS.

The YoLink server does not provide the identification of the YoLink Hub when the list of user devices is delivered. If monitoring of its status from HS is desired, then the QR code of the Hub needs to be explicitly specified in the QR table.

Data in the table is a 32 character code. When manually entered from the QR code from the YoLink device then it shows exactly as entered. Codes that are obtained from access grant to the user account are shown encased in [and]. These are not the QR codes, but the unique ID's of the deivce which has a one to one relationship with the QR code.

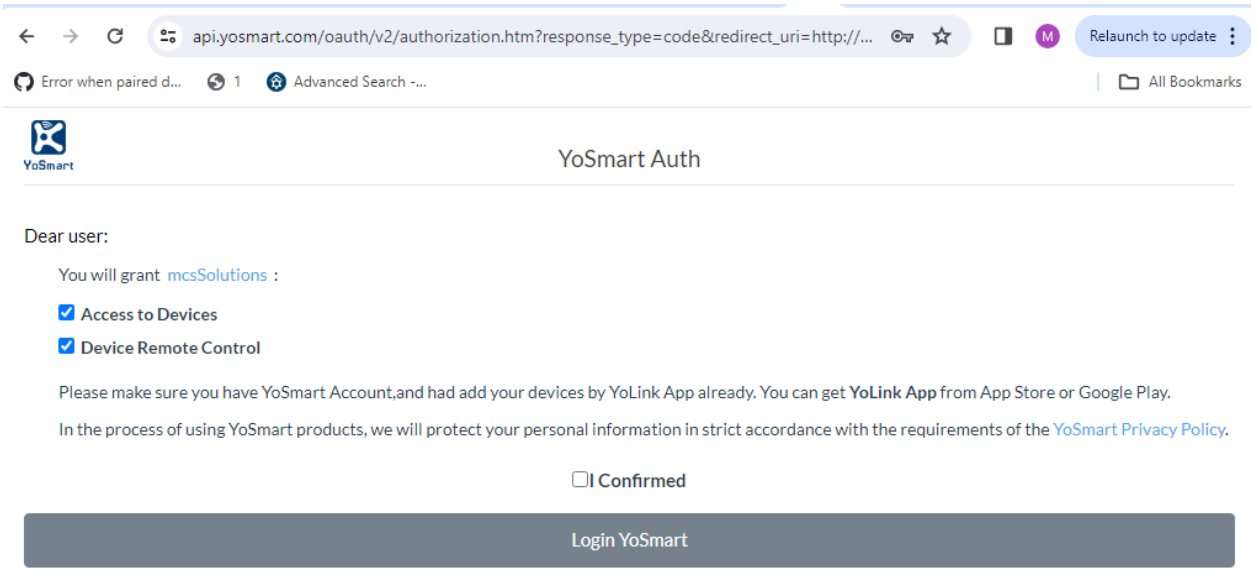


Figure 122 YoLink oAuth2 Authoriztion Screen

The information that is received from the YoLink cloud server is available on the mcsMQTT MQTT Page, Association tab such as is shown in Figure 124. mcsMQTT will automatically associate the likely features of interest such as the state or battery level if the MQTT Page, Client Tab setting has been enabled for auto-creation of known integrations. Figure 123 provides a typical example. The user can manually associate any of the items shown in the Association table if additional visibility in HS Device Features is desired. This is done using the "A"ssociate column checkbox. This will be on the /report topic.

Manual association also applies to new widgets that YoLink sells and have not yet been recognized by mcsMQTT. In this case the user can manually associate the widget for HS monitor and control and not need to wait for a future plugin update.

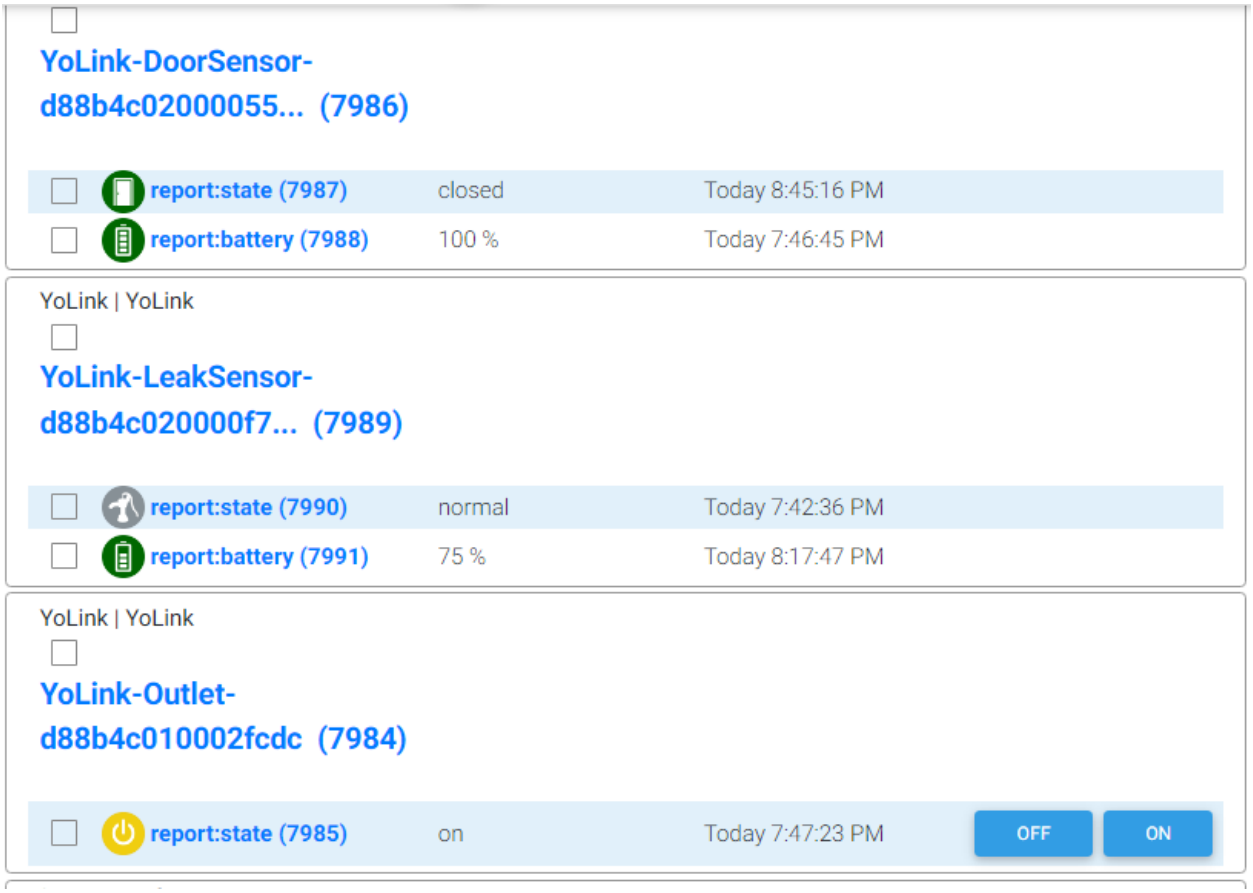


Figure 123 Auto-created YoLink Device and Features

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
0	☐	☐	☐		9601	YoLink/d88b4c010002fcdc/report					
1	☐	☐	☐	☐		Sub: YoLink/d88b4c010002fcdc/report:data:delay:ch	1	☐			2021-07-25 13:46:36
2	☐	☐	☐	☐		Sub: YoLink/d88b4c010002fcdc/report:data:delay:off	0	☐			2021-07-25 13:46:36
3	☐	☐	☐	☐		Sub: YoLink/d88b4c010002fcdc/report:data:delay:on	0	☐			2021-07-25 13:46:36
4	☐	☐	☐	☐		Sub: YoLink/d88b4c010002fcdc/report:data:loraInfo:gatewayId	d88b4c16030032d8	☐			2021-07-25 13:17:06
5	☐	☐	☐	☐		Sub: YoLink/d88b4c010002fcdc/report:data:loraInfo:gateways	1	☐			2021-07-25 13:17:06
6	☐	☐	☐	☐		Sub: YoLink/d88b4c010002fcdc/report:data:loraInfo:signal	-48	☐			2021-07-25 13:17:06
7	☐	☐	☐	☐		Sub: YoLink/d88b4c010002fcdc/report:data:power	0	☐			2021-07-25 13:46:36
8	☐	☐	☐	☑	9602	Dev: YoLink|d88b4c010002fcdc|report:data:state Sub: YoLink/d88b4c010002fcdc/report:data:state Pub: the following Topic on Device command YoLink/d88b4c010002fcdc/Outlet	open	☐	☐	☐	2021-07-25 14:16:36

Figure 124 YoLink Report Data

12.4 Geofence

The Geofence settings are used in conjunction with the smartphone tracking that is provided by the OwnTracks or NextTracks App. The setup and use of Owntracks for purpose of setting up a geofence is described in Section 15.

The location setup in HS Settings is always available as the center of a geofence. As many other locations can be setup as desired. For example, multiple work locations, a friend's location, etc. can be defined such as shown in Figure 125.

URL | YoLink | Voice Monkey | Geofence | Sense

Geofence Location Name	Latitude	Longitude	Boundary (ft)
HS	47.49323	-121.7831	500
Shop	47.49325	-121.7837	200
Property	47.49323	-121.7831	1000

Figure 125 Geofence Locations Setup

When position updates from Owntracks is received mcsMQTT will calculate the distance from each geofence location and determine if the position is inside or outside the geofence.

The geofence boundry that is setup will determine the parameters for the here-away logic. A phone moving toward a location that is identified as a geofence location will show here status if within 80% of boundary distance. A phone that is moving away will show away status when it exceeds 120% of the boundary distance. The hysteresis minimizes status toggling as the boundary is being traversed.

The geofence distance and the distance displayed with the Owntracks MQTT topics will be based upon the HS setting for use of English vs. Metric units. It will be either feet or meters.

12.5 Sense Energy

Sense Energy provides a device that measures electricity utilization and then based upon patterns it tries to determine specific appliances that are using the energy at any given point in time. The data that is collected is sent to the Sense cloud server for collection and analysis. There is not local access to the data that has been published.

When a Sense device is obtained there is a process to setup an account that consists of an email and password. To acess the data via their App or via mcsMQTT the login credentials are needed. The API for accessing the Sense cloud server account is not published but has been reversed engineered and implementation provided for powershell, nodejs and Python. mcsMQTT is a .NET implementatioin based upon information that is on the internet from other implementations.

There are three main capabilities provided. The primary is to make available the real time energy consumption of each device identified by Sense. This can be seen on the Association Table of Figure 128 and HS Devices of Figure 127. The second is to provide a graphical display of the energy consumed by each device over day, week, month and year timespans. Clicking on the MQTT Page, Association Tab, Payload column value for the sense/realtime/… topic of interest will produce the popup chart, but only if the additional download has been requested on the Sense setup of Figure 126. The third is to provide access for HS visibility to other data related to status of the Sense algorithm to identify devices, a timeline of events for specific devices turning off and on with chart display shown in Figure 129, and the accounting of device properties maintained by Sense Energy.

mcsMQTT will setup a websocket to receive realtime data. Fast polling places greater burden on the Sense server and HS/mcsMQTT. Default polling rate is every minute (60000 milliseconds). The polling is for realtime data. If the user selects Devices, Status or Timeline then it will be polled at the rate specified by user on the Cloud Page Sense tab as shown in Figure 126, but no faster than every minute

The realtime data feed provides energy utilization of each Sense-identified device and the total. For each of these a HS Device Feature will be created and updated as shown in Figure 127.

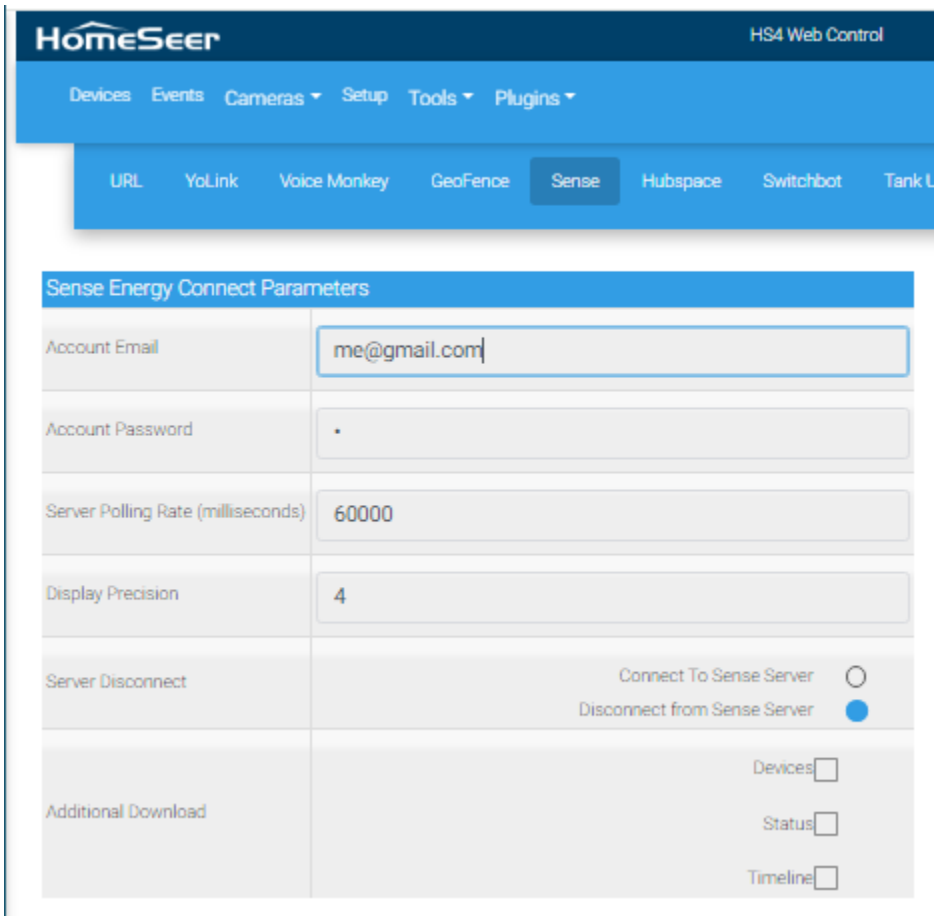


Figure 126 Sense Energy Setup

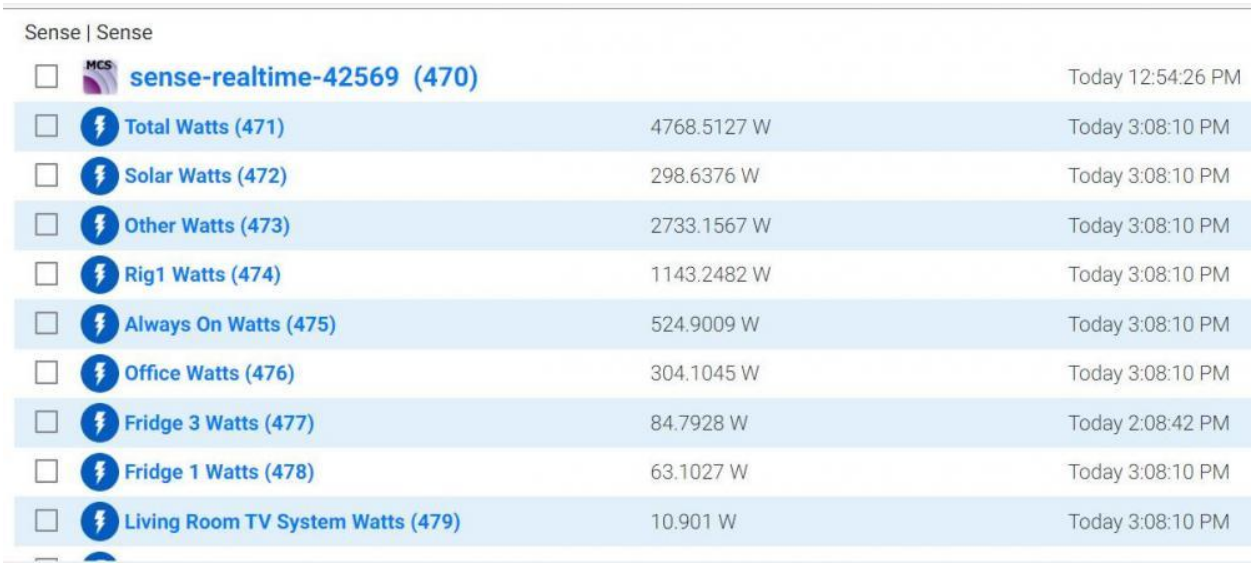


Figure 127 Sense Energy HS Devices and Features

6	☐	☐	☐	✓	473	Dev: Sense|Sense|Other Watts Sub: sense/realtime/42569:Other Watts Pub: the following Topic on Device command	
7	☐	☐	☐	☐		Sub: sense/realtime/42569:payload:device_data_checksum	9730466D784E3CCBBDA81F50A2E0DBB60B35F9C5
8	☐	☐	☐	☐		Sub: sense/realtime/42569:payload:features	MONITOR_RESET,DEVICE_DELETION,DEVICE_STATE_UPDA
9	☐	☐	☐	☐		Sub: sense/realtime/42569:payload:monitor_overview_checksum	78F923234ED82CBE4CCA7273C68FEDCC5A5C34FC
10	☐	☐	☐	☐		Sub: sense/realtime/42569:payload:online	true
11	☐	☐	☐	☐		Sub: sense/realtime/42569:payload:partner_checksum	FE8BB93DCBA62E8CE8A062D724DBC8A69309BD57
12	☐	☐	☐	☐		Sub: sense/realtime/42569:payload:pending_events:goal:guid	0

Figure 128 Sense Energy Topic Endpoints

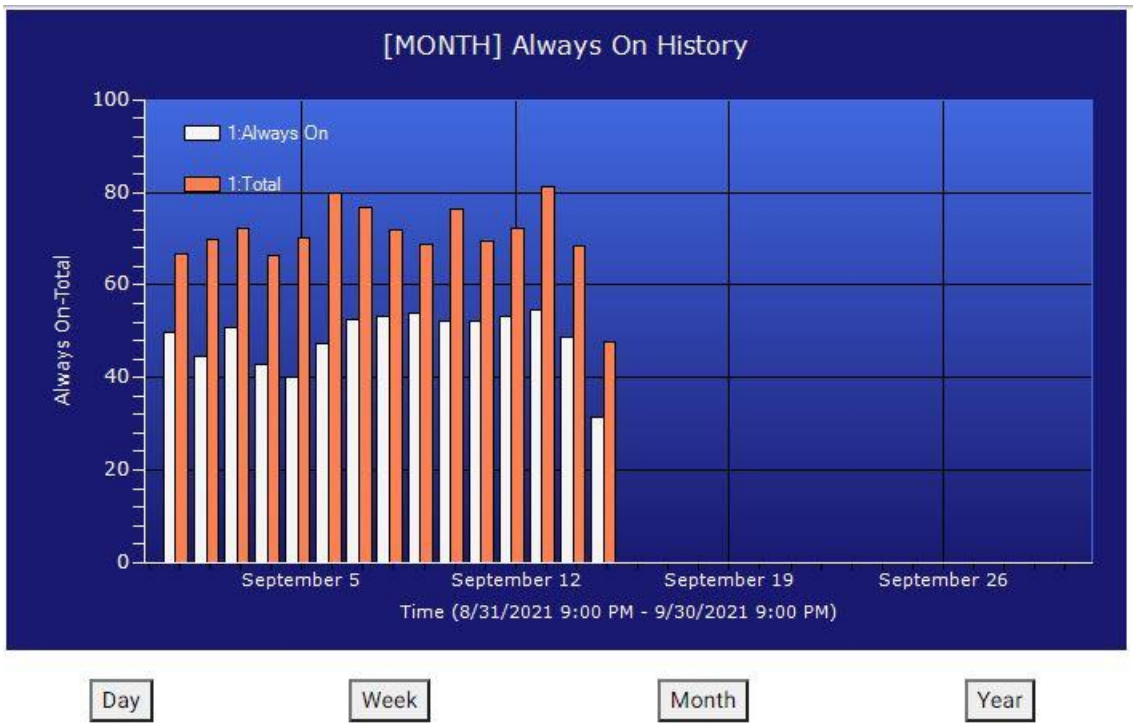


Figure 129 Sense Energy Trend Chart

12.6 Hubspace

Hubspace is a brand that shares a common smartphone App for outlets, locks, lights, fans and other products distributed through Home Depot. A snapshot from the HD marketing page is shown in Figure 130.

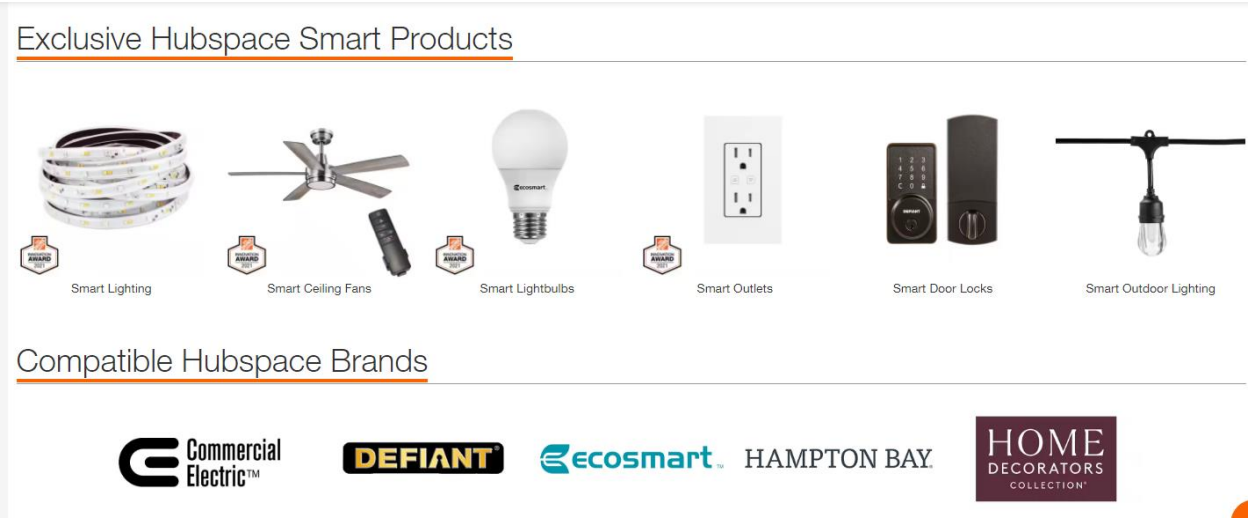


Figure 130 Hubspace Products

The products are setup by first creating a Hubspace account and using the QR codes provided with each instance of the product that is being installed. Bluetooth is used for initial setup of the product to gain access to the WiFi network and then WiFi is used to communicate with the Hubspace cloud server.

A Python library is available with instructions and repository at https://github.com/jan-leila/hubspace-py and https://pypi.org/project/paho-mqtt/#installation It is most easily installed from the command line / terminal window using PIP with the command

```
pip install hubspace
pip install paho-mqtt
```

If Python3 and PIP are not yet installed on the Homeseer computer they first need to be installed. Use Google for guidance on installing them on the OS that is hosting Homeseer. After installation, the folder where python.exe needs to be identified as it varies. There will also be a \Scripts subfolder that normally a subfolder of where python.exe is located. HubspaceRequest.py will be copied to the \Scripts subfolder.

During installation of Python, the PATH environment variable is normally updated with path to python and the scripts subfolder. It will likely be necessary to have these definitions in PATH.

The integration of this library with Homeseer is done with the Python script HubspaceRequest.py that is available in the mcsMQTT download package and originally installed at subfolder \bin\mcsMQTT. It will not be run from this location, but is moved to the \Scripts subfolder of python install.

The mcsMQTT setup for Hubspace integration is on the Cloud Page, Hubspace Tab as in Figure 131

Note the full path to python.exe and the Scripts folder in the setup. Also needed are the email and password to the account that was setup with Hubspace. Data is polled for status updates to handle the local control being synced with HS. Provision is also provided to disconnect the connection with the cloud server. This disconnection will also result in the Python script HubspaceRequest.py being terminated. HubspacaceRequest.py is managed by mcsMQTT and run when the setup is complete and not disconnected.

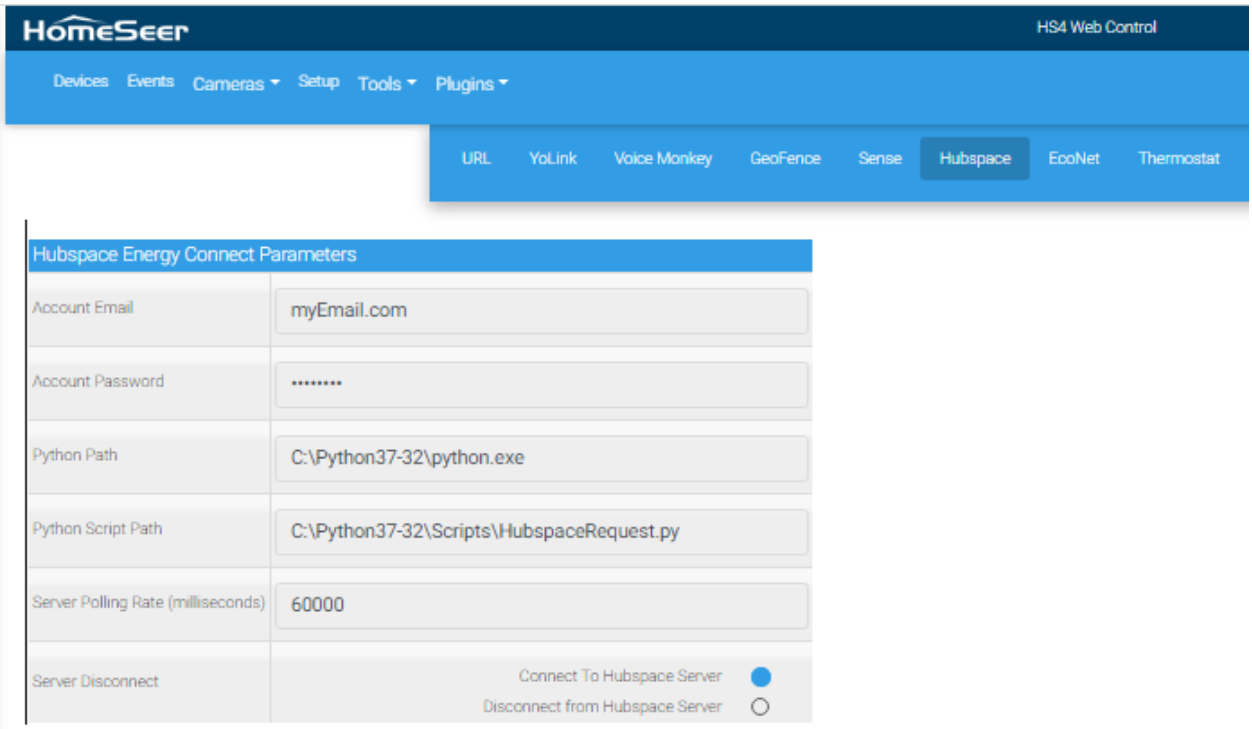


Figure 131 Hubspace Integration Setup

After everything is put in place it would be good to run HubspaceRequest.py from console/terminal window such as below after navigating to the Python \Scripts folder. In this example, 192.168.0.5 is the MQTT Broker address. If no MQTT Broker has been setup then use 127.0.0.1 here and on the mcsMQTT MQTT Page, Broker Tab. myEmail.com will be your Hubspace account email. myPW can be anything as this test will not have success with connection to the cloud server. The test is to assure all the Python components are installed correctly. If no error message on the console/terminal then likely the install is good.

```
python hubspaceRequest.py myEmail.com myPW 192.168.0.5
```

At this time the integration supports the outlet and lightbulb. Others can be added in the future with assistance from users of the other equipment. The mcsMQTT integration structure is setup to add additioinal products with minimum difficulty.

mcsMQTT will use two endpoints with the Hubspace server. One provides Meta data that describes the characteristics of the product that exist on the account/server. The other provides the Attributes of the products which includes the current state.

The data of interest is visible on the MQTT Page, Association Tab. This includes the Hubspace product attributes and message feedback in communicaitons between mcsMQTT and HubspaceRequest.py. The Meta data is not shown in this table. A slice of this is shown in Figure 132. Beyond the setup of HS Device Features other features of mcsMQTT, such as History data and charting, can be selected from this page.

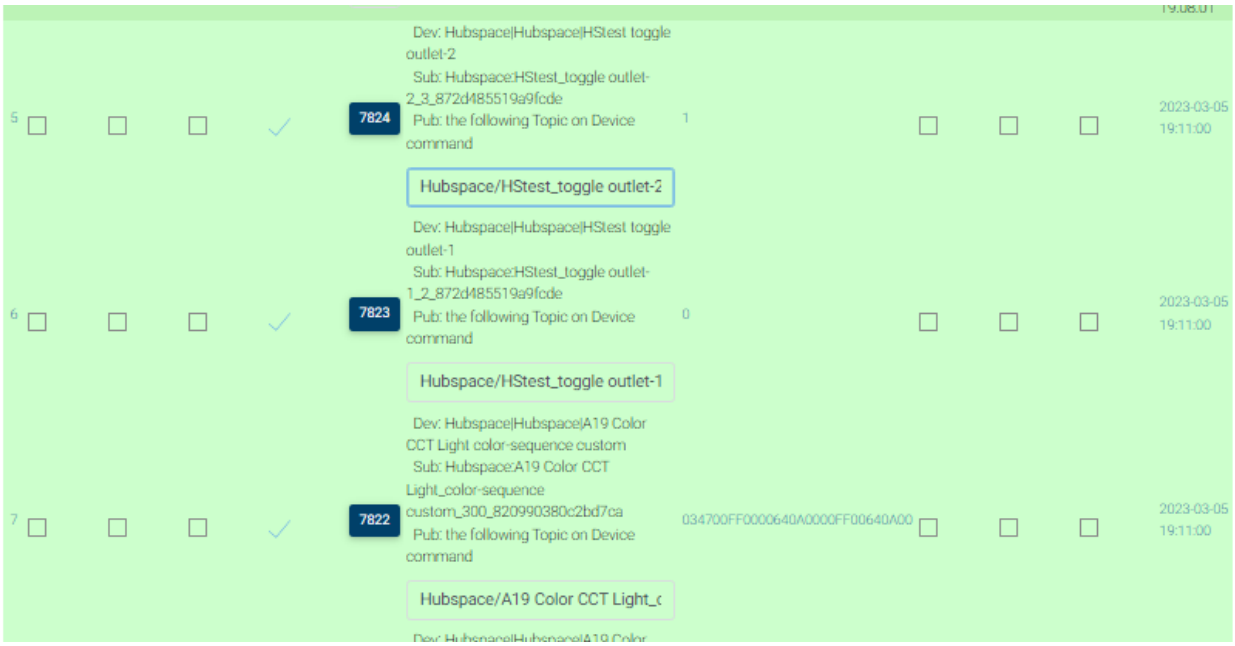


Figure 132 Hubspace Product Data as MQTT pseudo-Topics

The outlet is presented by the Hubspace server as two independent plugs. There is no single command to control both simultaneously, but the HS linking capability can be used if the desire is to have both plugs controlled together.

A feature is provided by mcsMQTT to allow an outlet to be always ON. At each polling interval mcsMQTT will confirm it is ON and if not, it will command it to the ON state. The On/Off controls will still be visible on the HS Devices page, but can be removed if desired using the Status/Graphics Tab of the HS Devices Page.

To indicate that an outlet plug is be always ON, the PubTopic on the Association Tab will be changed so that is indicates it is “on” rather than the actionId that is normally at the end. Using the examble of HS Feature 7823 shown in Figure 132, the Pub textbox will be changed

From: Hubspace/HStest_toggle outlet-2_3_872d485519a9fcde/3

To: Hubspace/HStest_toggle outlet-2_3_872d485519a9fcde/on

HS Device and Features will be automatically created based upon the Meta data with example of outlet and light in Figure 133. The Feature names are taken from the Meta data. They can be changed by the user from the default names if desired. Renaming will not affect operation. Similarly, Floor, Room and Device/Feature grouping can all be changed using the Devices page tools provided by HS.

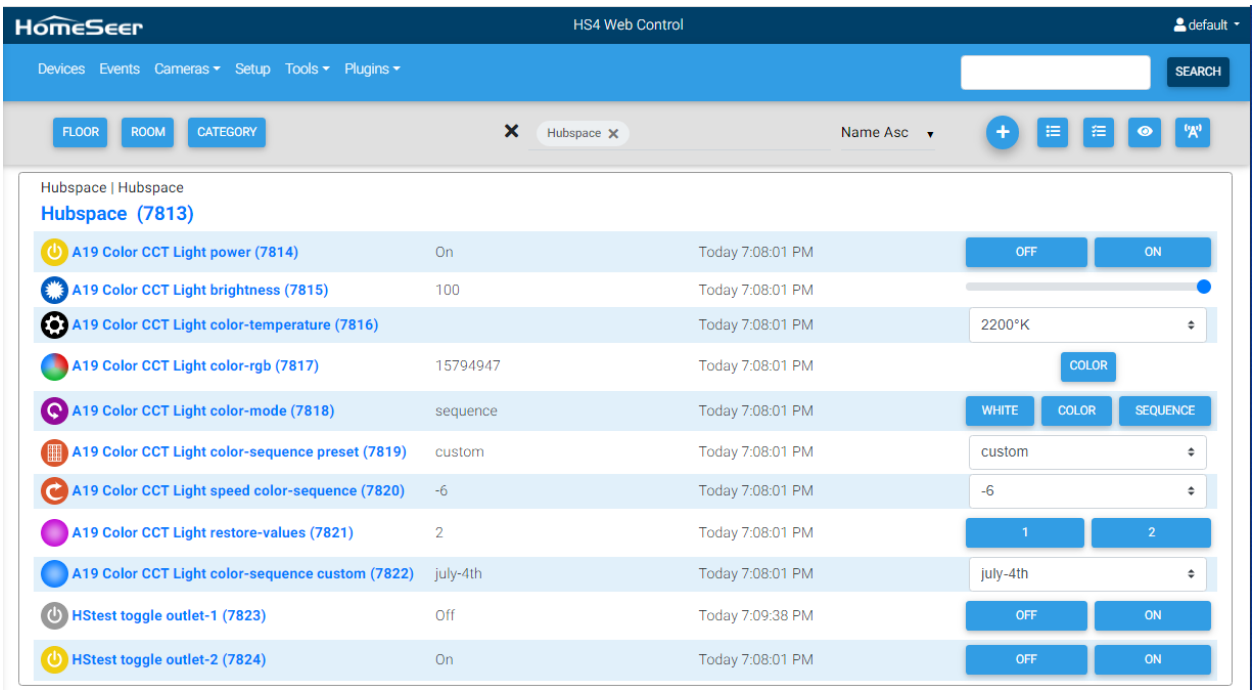


Figure 133 Hubspace Auto-Created HS Device and Features

12.7 Switchbot

12.7.1 Introduction

Switchbot provides a line of interfaces that are managed via their Cloud server. This includes Bot, Curtain, Meter, Lock, Keypad, Keypad Touch, Motion Sensor, Contact Sensor, Ceiling Light Pro, Plug Mini (US), Plug Mini (JP), Plug, Strip Light, Color Bulb, Indoor Cam, Pan/Tilt Cam, Robot Vacuum Cleaner S1, Robot Vacuum Cleaner S1 Plus, Blind Tilt and likely more in the future.

For those who want to use the Switchbot smartplugs and lights locally without dependence on a Cloud server, a mechanism exists with Over-The-Air change of the firmware. Section 21.13.2 describes this process. Once changed, the integration described in this section via the Cloud server will no longer be used.

12.7.2 Setup

To get started with Switchbot, one first needs to install the Switchbot App on their mobile device to get access to the token and secret keys. This process is described in [GitHub - OpenWonderLabs/SwitchBotAPI: SwitchBot Open API Documents]() for the Version 1.1 integration performed by mcsMQTT. These keys are obtained from the App using the steps:

> Go to Profile > Preference b) Tap App Version 10 times. Developer Options will show up c) Tap Developer Options d) Tap Get Token

The tokens are then entred on the mcsMQTT Cloud Page, Switchbot Tab as shown in Figure 134. Two access methods to the cloud are available. mcsMQTT will poll status updates at the rate specified by the user in the setup. If the user has a means to accept data pushed by the Switchbot Cloud server then the WAN-visible URL is entered in the setup. This URL will be used by mcsMQTT to setup a webhook listener and inform the Switchbot server of its presense.

A setup provision also exists to disconnect from the Switchbot server and reconnect while still maintain all previous setup.

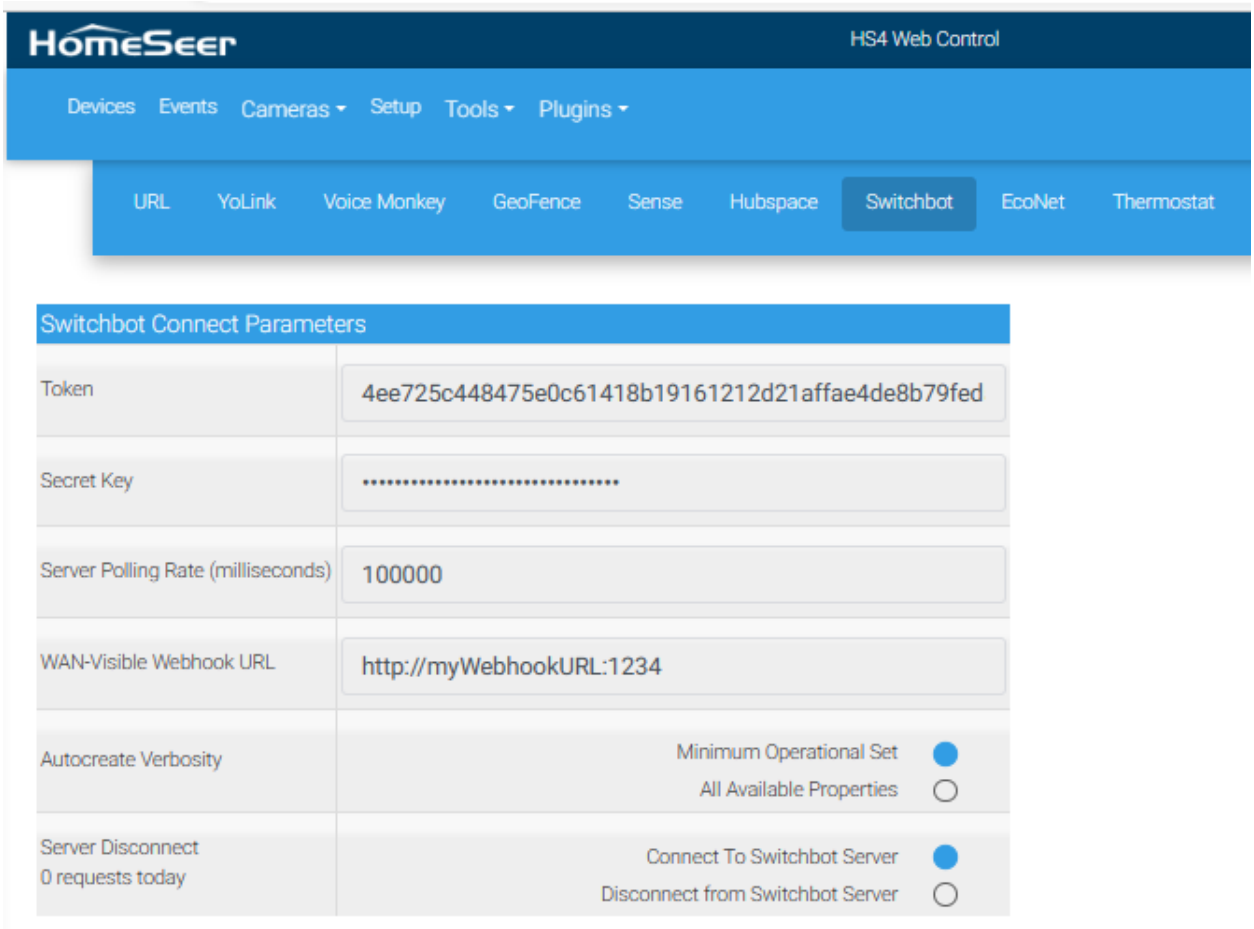


Figure 134 Switchbot SetupFigure 134

mcsMQTT will ask the Switchbox server for the list of devices that have been setup on the user account. It will then create HS Device and Features such as shown in Figure 135. The number of Features per Device will depend upon the Autocreate Verbosity setting in Figure 134. This setting can be toggled and will apply to all Switchbot Devices.

12.7.3 Switchbot Devices

When a Switchbot item is controlled from HS via Devices, Event, or Script/CAPI, mcsMQTT will deliver the request with a retry if necessary. The Switchbot response will include and updated status and HS Feature wil be updated based upon this status. mcsMQTT will then poll to get status from Switchbot server to confirm. The HS Feature status will be updated again. No update occurs unless the response status indicates a success status (100). If the Webhook has been setup the state change will also be reported via the Webhook and Feature status updated.

The API does not document the status reponse content for each Switchbot device. The only device available for initial development that can be controlled is the Bot. Other devices that users may have will need the debug output to include status reporting for that device. As a note, the US Smartplug was converted to Tasmota, but still shows in the Switchbot account. It, however, could not connect to the Hub so could not be controlled by the Cloud integration (or smartphone App).

Status updates are polled and optionally delivered via Webhook. Polling limit of the Switchbot server is 1000 requests per day. mcsMQTT enforces a limit of around 900 with the polling interval minimum of 100000. The status of the number of requests and if the daily limit has been exceeded is shown in the Server Disconnect label such as is shown in Figure 134. While the user enters a desired polling rate that would cover the interval for a single device, mcsMQTT will increase the interval by the number of non-Hub devices that the user has their account. This means that if a user has 10 non-Hub devices the polling interval will be increased by a factor of 10 so the daily limit is not exceeded.

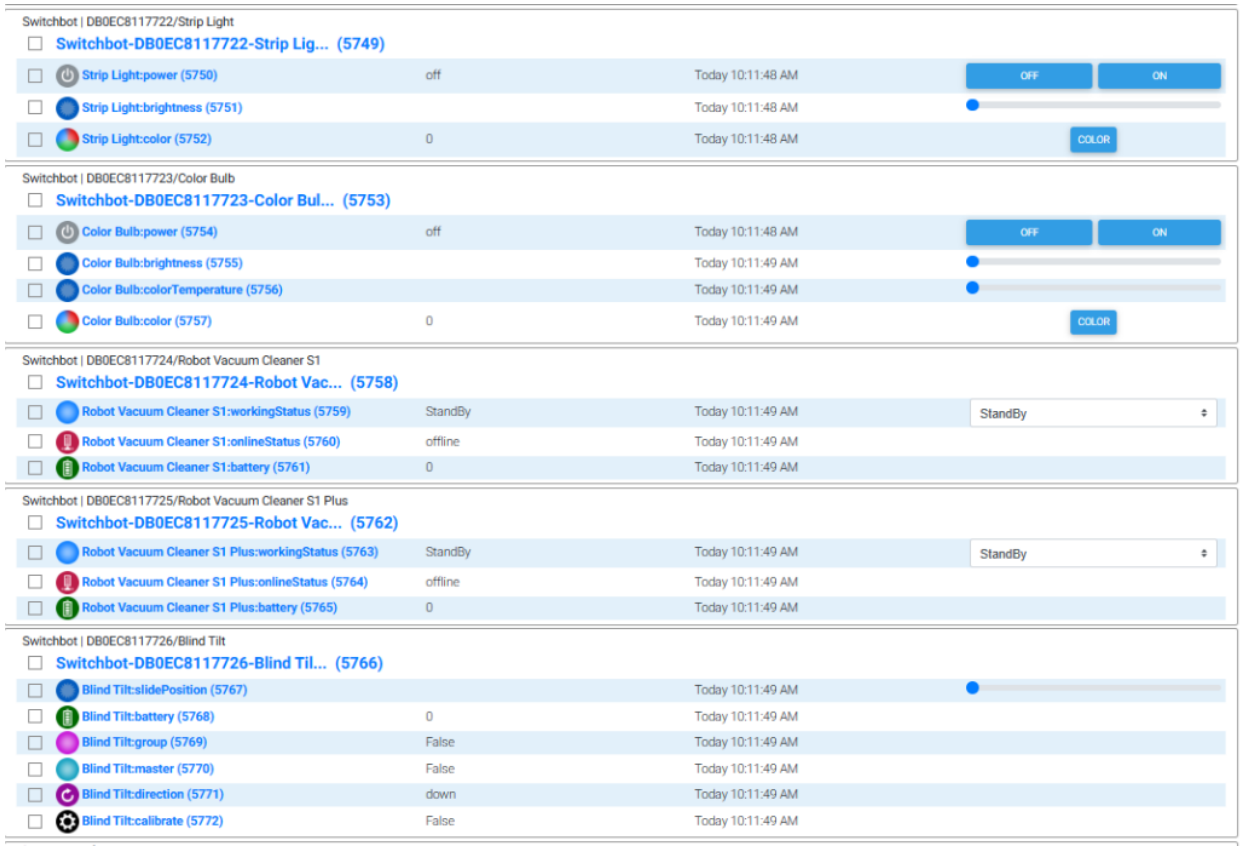


Figure 135 Sample Switchbot Devices and Features

User tweaking of the setup is available on the MQTT Page, Association Tab and Edit Tab such as shown in Figure 136. This will allow HS-properties edit via Edit tab and show the mechanism by which the devices can be accessed via MQTT Topic/Payload orientation.

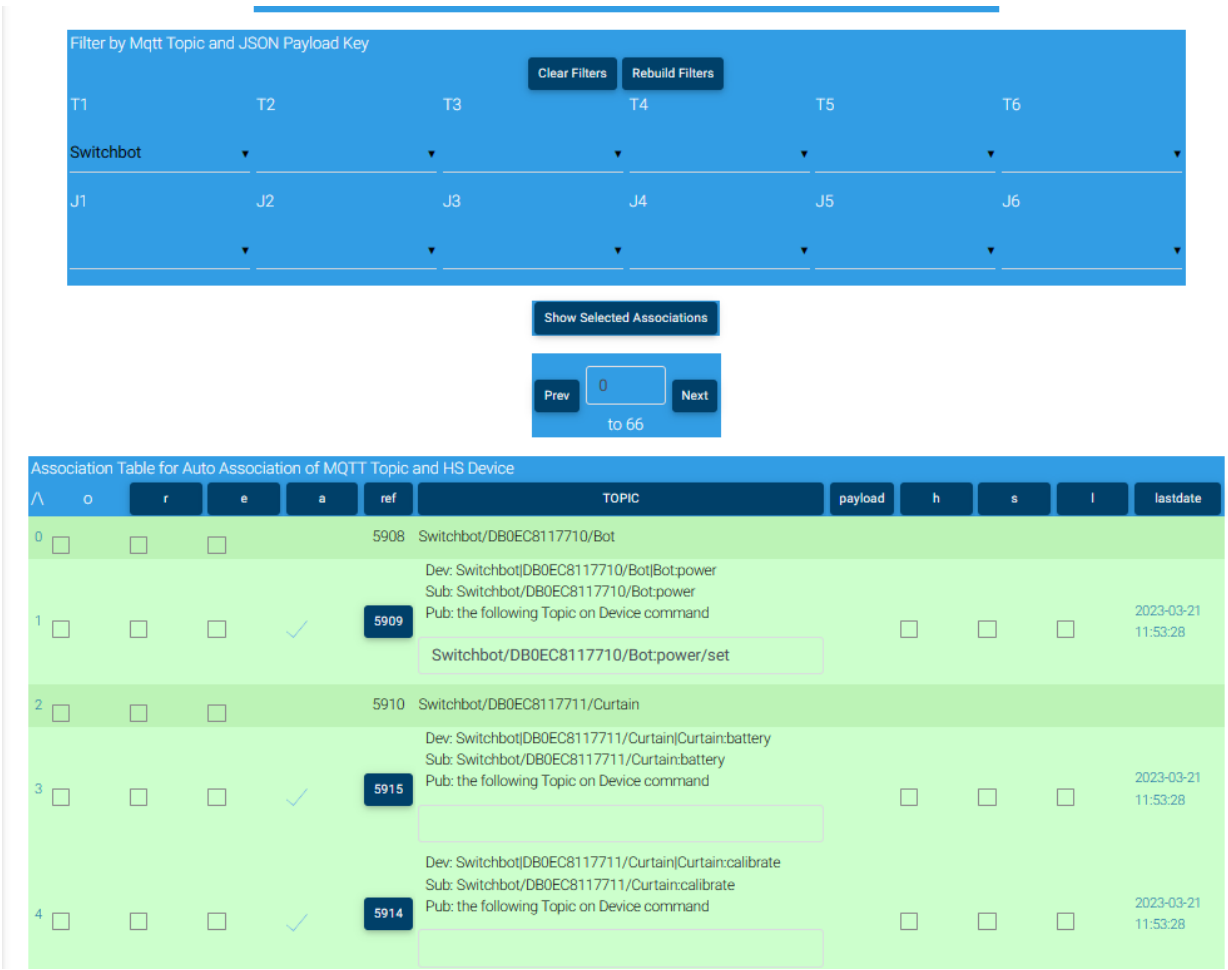


Figure 136 Switchbot Pseudo-Topics

12.7.4 Switchbot Infra-Red

Switchbot hub supports learning of IR code for custom devices and from a library of standard devices such as TV, DVR, Fan, etc. Each appliance that has been learned will be reflected in HS as a Device and Feature with a dropdown selector control. Control is also possible through standard event action and scripting methods. See Figure 137.

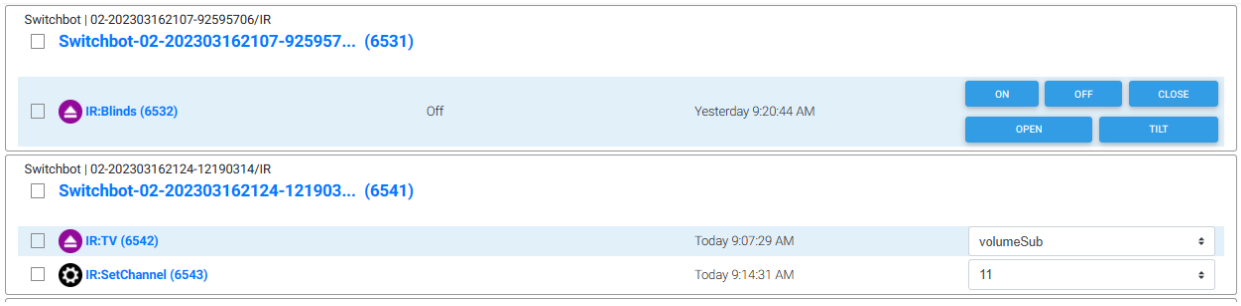


Figure 137 Switchbot IR Appliance Devices

For the standard IR appliances known by Switchbot, mcsMQTT will populate the selector control with the available IR-code names. These are the only ones that the Switchbot hub will recognize and respond by blasting an IR pulse sequence.

For the custom IR appliances, the user needs to define the IR-code names that were setup on their account via the Switchbot App. The names are maintained as Value Status Pairs (VSP). They can be entered from the HS Devices Page, click on the IR Feature, Status/Graphics Tab, New Single Value, Edit icon, enter Label column Status such as “Close” in Figure 138.

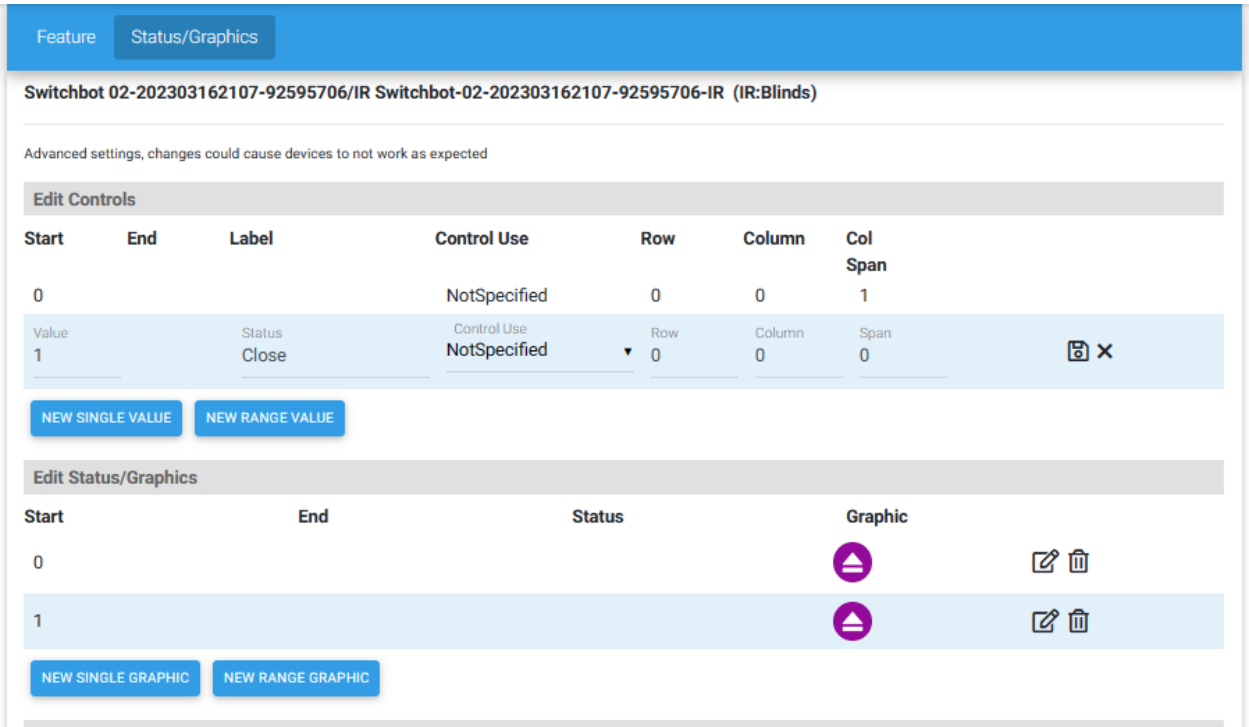


Figure 138 Switchbot Custom IR Code Definition

This can also be done on the MQTT Page, Edit Tab/Popup for the selected IR Topic as shown in Figure 139. The syntax in the Add/Edit textbox is Name=number;Name for a single entry or to put all entries in at once use comma between each name such as “-,On,Off,Close,Open,Tilt” to define six IR Codes. Before doing this make certain the Max Number of VSP texbox will account for all codes being entered.

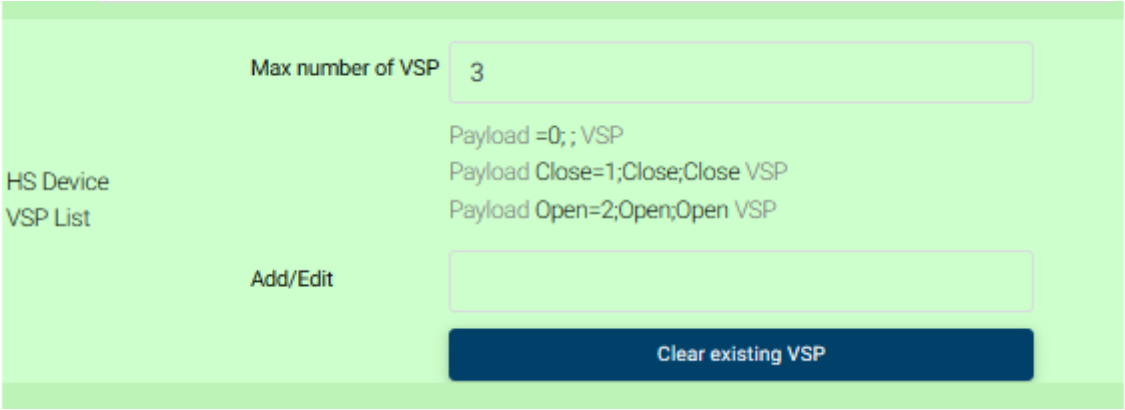
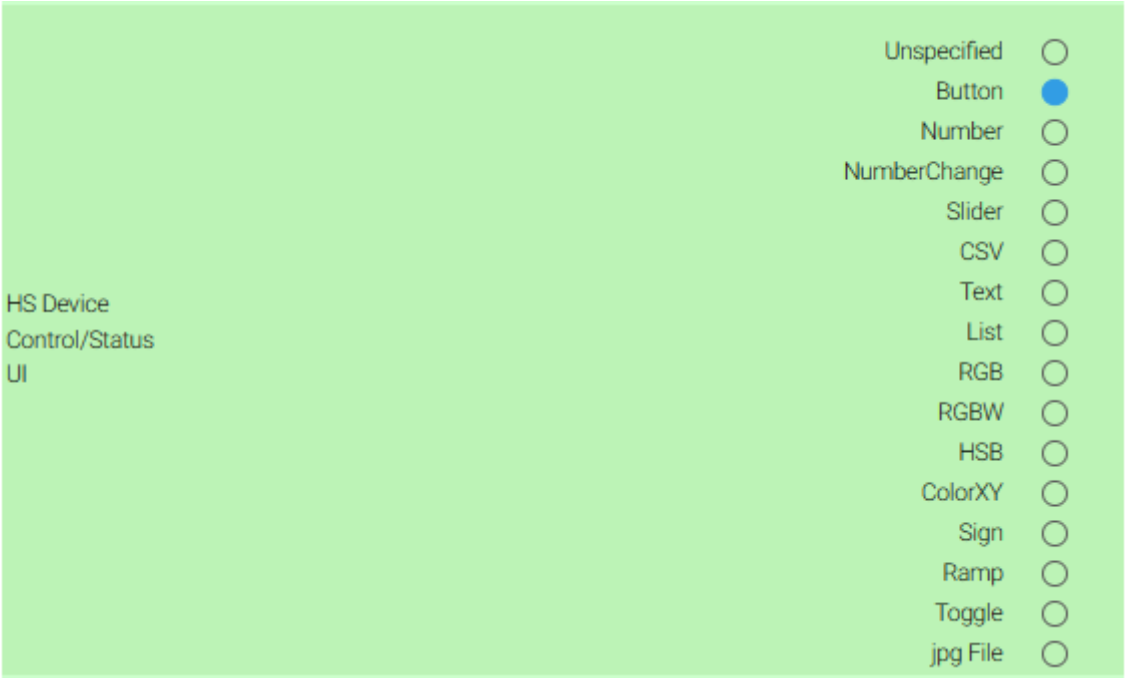


Figure 139 Switchbot Custom IR Codes Setup on Edit Tab

For both custom and standard IR Appliances it is possible to change between using Buttons or a Selector for the HS Devices control. The default is Selector. The Selector is identified on the Edit Tab with a HS Device Control/Status UI of List. To change the list to a set of Buttons on the HS Devices Page for control then select the Button radio option.



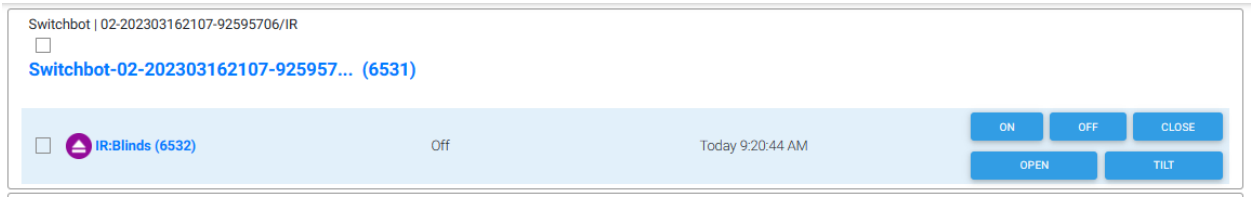


Figure 140 Switchbot IR Control via Buttons

In the case of the TV, IPTV, and Set Top Box the IR control contains two Features. One for discrete control and one for channel number selection. By default, the channel number selection is shown to the user as a number text box into which the desire channel is entered.

This default can be changed to a list selector where the list of favorite channels is available. It could also be done with individual button for each channel favorite. This change is done on the Edit Tab with the Control/Status UI for Button vs. List and the VSP text box where the list of channels is entered such as "4,5,11,13,121,118" without quotes. This process is similar to what is shown in Figure 139 and Figure 140. Note that this needs to be done via Edit Tab and cannot be done directly in HS Devices Status/Graphics. The icons, however, will be done on the Status/Graphics page if desired.

12.8 Rheem EcoNet

Rheem EcoNet is a cloud integration of Rheem user equipment. Access to the cloud server is via a username and password. This provides ability to login and obtain credentials for subsequent access to the EcoNet server. This is not a public API, but one that has been reverse engineered so it is possible that it could change in the future without notice. The mcsMQTT integration is based upon the work contained in GitHub https://github.com/w1ll1am23/pyeconet/find/master) .

The mcsMQTT Cloud Page, EcoNet tab provides the ability to enter username and password as well as disconnect when desired. This is shown in Figure 141. This login provides the ability to obtain a user token and an account id that are used in subsequent communications. These secret credentials are not made visible by mcsMQTT except when the option for the additional download is selected via checkbox. For normal updates from the EcoNet server the additional downloads are not needed. The additional data may be useful to get specific identification information if trying to later publish setpoints, modes, or other control from HS to the equipment.

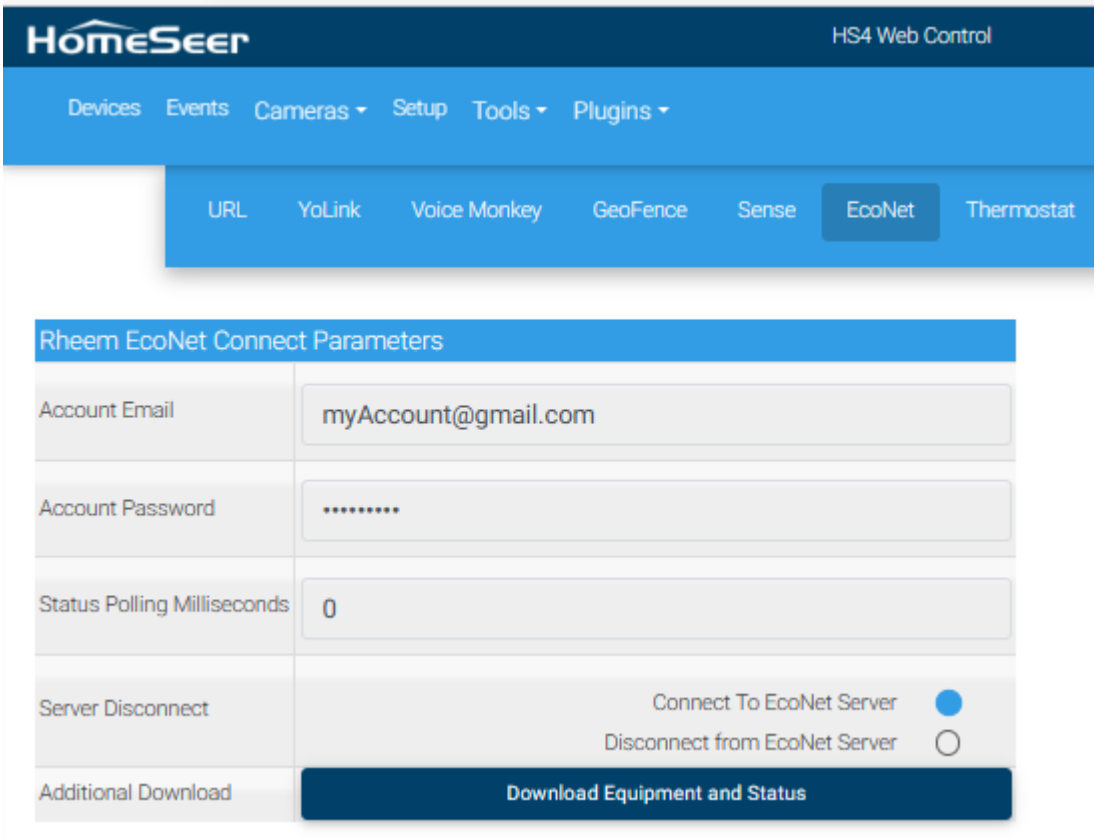


Figure 141 Rheem EcoNet Setup

The observation is that reports from EcoNet server for normal status are infrequent with an update rate of perhaps every couple of hours. It was also observed that after the @CONNECTED message was delivered with a "false" status that no subsequent messages were received. The assumption in this case is the local equipment has gone offline and not visible to the EcoNet server.

EcoNet server provides data from two sources. Pulled data includes the list equipment, it's properties and status. This pull request is done on startup and when a setup change is made. Pushed data is sent

encrypted as status changes. mcsMQTT consolidates the two sources into a single set of messages used to create and then update HS Device Features. A manual pull of data can be done with the button shown in Figure 141. While it should not be necessary, it is also possible to schedule periodic pulling of the data as shown on the same figure where a 0 value is now visible. Perhaps every hour (3600000 milliseconds) or longer would be reasonable intervals if desired.

During development there were cases where connection was lost with the push/MQTT communication channel and graceful reconnection was not always possible. To overcome the difficulty the EcoNet communication with the EcoNet Cloud Server was moved to a separate process (EcoNet.exe) and the MQTT on the LAN used to exchange information with this EcoNet process. The Topic used is EcoNet/message with JSON payload. The data from EcoNet server is communicated on the LAN using the EcoNet/Location Topic. Commands to change the equipment use the topic /user/+/device/desired where + is the location identifier obtained via the login process.

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	s	l	lastdate
0	☐	☐	☐			EcoNet					
1	☐	☐	☐		4364	EcoNet/Location					
2	☐	☐	☐	✓	4365	Dev: EcoNet|Location|Location:1017:ElectricWaterHeater:ACTIVE Sub: EcoNet/Location:1017:ElectricWaterHeater:ACTIVE Pub: the following Topic on Device command	true	☐	☐	☐	2023-02-14 10:54:28
3	☐	☐	☐	✓	4366	Dev: EcoNet|Location|Location:1017:ElectricWaterHeater:AWAY Sub: EcoNet/Location:1017:ElectricWaterHeater:AWAY Pub: the following Topic on Device command	false	☐	☐	☐	2023-02-14 10:54:28
4	☐	☐	☐	✓	4367	Dev: EcoNet|Location|Location:1017:ElectricWaterHeater:CONNECTED Sub: EcoNet/Location:1017:ElectricWaterHeater:CONNECTED Pub: the following Topic on Device command	true	☐	☐	☐	2023-02-14 10:54:28
5	☐	☐	☐	✓	4368	Dev: EcoNet|Location|Location:1017:ElectricWaterHeater:DRACTIVE:value Sub: EcoNet/Location:1017:ElectricWaterHeater:DRACTIVE:value Pub: the following Topic on Device command	null	☐	☐	☐	2023-02-14 10:54:28
6	☐	☐	☐	✓	4369	Dev: EcoNet|Location|Location:1017:ElectricWaterHeater:ENABLED:value Sub: EcoNet/Location:1017:ElectricWaterHeater:ENABLED:value Pub: the following Topic on Device command user/30612480086952676/device/desired	1	☐	☐	☐	2023-02-14 10:54:28

Figure 142 EcoNet MQTT Report Snapshot on Association Tab

HS Device and Features are created based upon the Location Equipuipment information provided by EcoNet Server. The only equipment available during test was Water Heater so that is all that is auto-created. Updates to mcsMQTT can be made if access to other equipment is made possible.

For the Water Heater the HA user interface is shown in Figure 143.

During evaluation the WiFi SIGNAL update occurred as the signal strength changed by 1 unit at a rate of once or twice per hour. This will be dependent upon local WiFi characteristics. As a control is issued,

such as change of Setpoint, the status is immediately updated and will usually include multiple properties such as ScheduleResume being shown with “Resume” to reflect that the schedule has been superceded and Resume control is now active to restore operation to the schedule.

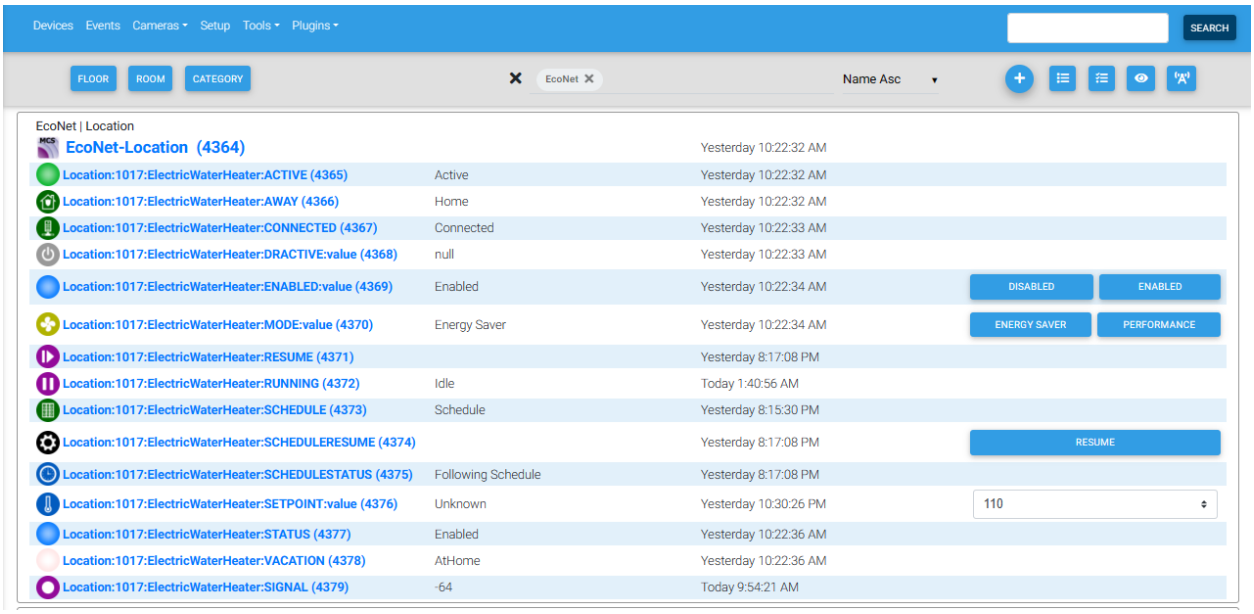


Figure 143 EcoNet Water Heater Device and Features

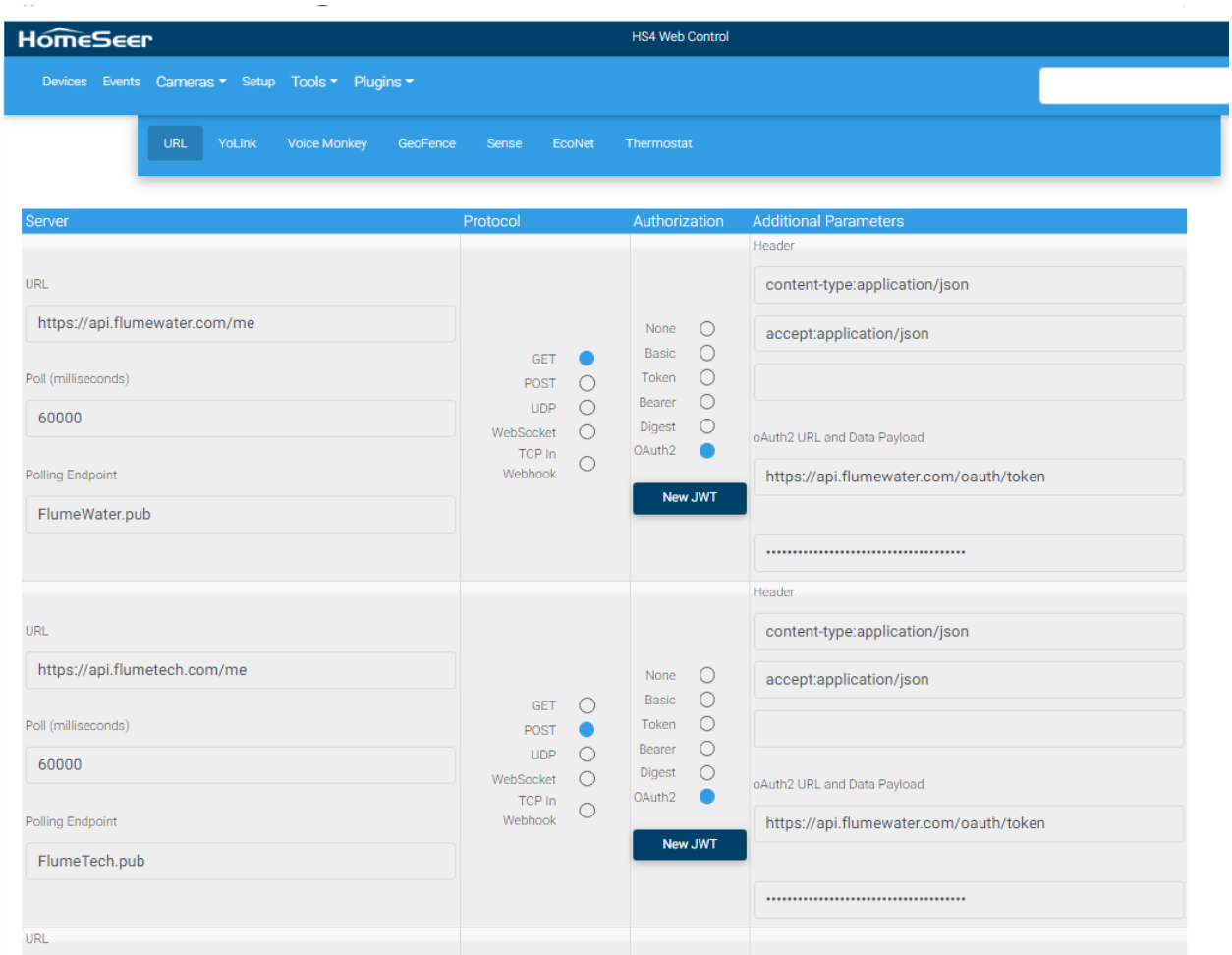
12.9 Flume Water

Flume provides a Personal API that is well documented at [Flume Personal API | Flume Help Center (flumewater.com)](). This enables a user to interact with the Flume Cloud server to get information similar to that available via their smartphone App. Interaction with the API can be setup from the mcsMQTT Cloud Page, URL Tab.

There are two Servers provided by Flume. One for general information. The other for water usage queries. Two IP's are setup on the URL tab to handle both servers. oAuth2 authentication is used. The oAuth2 server for Flume is https://api.flumewater.com/oauth/token. The oAuth2 payload looks like:

{"grant_type": "password","client_id": "XXXXX","client_secret": "YYYYY","username": "myemail@gmail.com","password": "mypassword"}

Email and password are the ones used for the Flume account and App. The XXXXX and YYYYY are obtained from the Get Tokens link at [Introduction (readme.io)]. User login to this site is needed.



Note that endpoints that will be used for water usage are using POST while those for general information are using GET.

Polling was setup at one minute and five-minute intervals with the endpoints contained in the \data\mcsMQTT\FlumeTech.pub and \data\mcsMQTT\FlumeWater.pub. If different endpoints should

be polled at different rates, then additional .pub files should be used and HS Events setup to Send MQTT Publish message. If only one endpoint is to be sent then it can be put directly in the MQTT Send Message action text box rather than creating a .pub file.

12.9.1 Water Use Queries

Once the tokens have been obtained and the oAuth2 authentication has been setup then one needs to get the identification of the installed equipment. This is done with Fetch User's Devices from the same page where the token was obtained. Alternately device identification it can be obtained by entering "/devices?user=false&location=false" in the Submit text box on the HS Devices page for the device created for the flumewater URL.

The returned payload is JSON and the key of interest is "id" such as "id": "ZZZZZ". If done from the HS Device page then it will be shown on the Association tab.

The device identification (ZZZZZ value) is used in the water utilization queries. These queries are most easily setup and managed using a publist which is a structured file located in the HS subfolder \data\mcsMQTT with a filename ending in ".pub". See Section 12.1.2 for more information on the publist use in this context.

It is possible to make a single request with multiple queries or multiple requests of a single query each. The former is preferred to reduce overhead. The example below are three queries. The first gets the water usage for the past minute. Th second for the current day and the third gets the usage for the current month. Flume provide the options available for queries at [Querying Samples (readme.io)](). Note that the URL contains a replacement variable
`$$PAYLOAD:(URL/api.flumewater.com/me.GET:data:*:type-1:device_id):`
which could be changed to manually enter the device_id rather than picking it up from the payload from an earlier /devices query from flumewater.

An example query is shown below where the start of the period being requested is at the first second of the bucket's interval.

```
URL/https://api.flumetech.com/me/devices/$$PAYLOAD:(URL/api.flumewater
.com/me.GET:data:*:type-
1:device_id):/query={"queries":[{"request_id":"Now","bucket":"MIN","op
eration":"MAX","since_datetime":"$$YEAR:-$$MONTH:-$$DAY:
$$HOUR::$$MINUTE::00"},{"request_id":"Today","bucket":"DAY","operation
":"MAX","since_datetime":"$$YEAR:-$$MONTH:-$$DAY:
00:00:00"},{"request_id":"ThisMonth","bucket":"MON","operation":"MAX",
"since_datetime":"$$YEAR:-$$MONTH:-01 00:00:00"}]}
```

Another approach is to query from the start of the desired interval such as for the "Now" query where the start is 60 seconds in the past. In this case inline expressions are used to compute 60 seconds in the past and to format the date in the desired format.

```
{"request_id":"Now","bucket":"MIN","operation":"MAX","since_
datetime":"<<Format_DateTime("<<DateAdd("second",-60,Now)>>","yyyy-MM-
dd HH:mm:ss")>>"}
```

The returned data will be in the Association table. Note the use of “request_id” parameters in the above queries. They will be part of the structure in the Association table to identify the utilization of interest. This was done for Devices 126 and 128 in the example below.

Filter Association Table by Mqtt Topic and JSON Payload Key Clear Filters Rebuild Filters

T1	T2	T3	T4	T5	T6
URL	api.flumetech.com				

J1	J2	J3	J4	J5	J6

Show Selected Associations

Prev 0 of 34 Next

Association Table for Auto Association of MQTT Topic and HS Device

Λ	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
0					19	URL/api.flumetech.com/me					
1						Sub: URL/api.flumetech.com/me.GET	The remote server returned an error: (401) Unautho rized.				2022-02-09 15:10:47
2					125	URL/api.flumetech.com/me.POST					
3						Sub: URL/api.flumetech.com/me.POST:code	602				2022-02-10 13:00:37
4						Sub: URL/api.flumetech.com/me.POST:count	0				2022-02-10 13:00:37
5						Sub: URL/api.flumetech.com/me.POST:data:01:ThisMonth:datetime	2022-02-01 00:00:00				2022-02-10 13:00:37
6				✓	128	Dev: URL|api.flumetech.com|me.POST:data:01:ThisMonth:value Sub: URL/api.flumetech.com/me.POST:data:01:ThisMonth:value Pub: the following Topic on Device command	8478.82961472				2022-02-10 13:00:37
7						Sub: URL/api.flumetech.com/me.POST:data:01:Today:01:datetime	2022-02-10 00:00:00				2022-02-10 13:00:37
8						Sub: URL/api.flumetech.com/me.POST:data:01:Today:01:value	1254.38205204				2022-02-10 13:00:37
9						Sub: URL/api.flumetech.com/me.POST:data:01:Today:datetime	2022-02-10 00:00:00				2022-02-10 13:00:37
10				✓	126	Dev: URL|api.flumetech.com|me.POST:data:01:Today:value Sub: URL/api.flumetech.com/me.POST:data:01:Today:value Pub: the following Topic on Device command	1254.38205204				2022-02-10 13:00:37

12.9.2 Notifications

Notifications are obtained with the GET query to the /notifications endpoint. Flume returns a list of notifications with the most recent first and older ones later. What is of interest is only the most recent so the query is setup to return only one notification. The JSON returned payload contains a “type” parameter that can be USAGE_ALERT, BUDGET, GENERAL, HEARTBEAT or BATTERY. The remainder of the keys in the JSON payload then apply to this notification type. To provide unique HS devices for each notification type this “type” key can be elevated for uniqueness after selecting type as an Associated topic. This is done on the Edit tab as shown in Figure 144 below. Flume provides the notifications in a JSON array. It does not matter which position in the array the notification is delilvered so the elevated key can be prefixed with “*:” to make the position a don’t-care.

Edit Setup Or Edit Of Subscription (Inbound) To a MQTT Topic	
MQTT Subscribe Topic	URL/api.flumetech.com/me.POST JSON key(s) to be elevated for uniqueness *:type

Figure 144 Elevate Flume JSON Key 'type' to Provide Uniqueness of Notifications

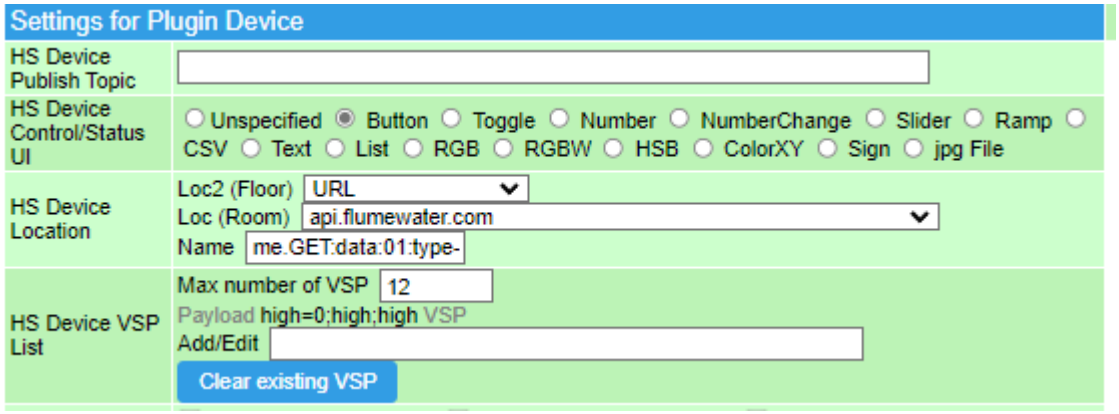
For usage alerts there is a message and the time the message was created and when last seen. It is possible to create a HS device for each of the keys or as shown below the payloads can be combined using the Expression textbox on the Edit tab for the Associated topic. For example the expression below concatenates three strings to be stored in the HS DeviceString. Note the use of quotes in the expression to assure each element that is being combined with "&" is a string.

```
"$$PAYLOAD:"&" created at "&"$$PAYLOAD:(
URL/api.flumewater.com/me.GET:data:*:type-1:created_datetime):"
```

Association Table for Auto Association of MQTT Topic and HS Device

Λ	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
12	☐	☐	☐	☑	123	Dev: URL|api.flumewater.com|me.GET:data:01:type Sub: URL/api.flumewater.com/me.GET:data:01:type Pub: the following Topic on Device command	1	☐	☐	☐	2022-02-10 11:06:58
13	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:address	3029 Gulf of Mexico Dr	☐			2022-02-10 11:06:56
14	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:address_2		☐			2022-02-10 11:06:56
15	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:away_mode	false	☐			2022-02-10 11:06:56
16	☐	☐	☐	☑	124	Dev: URL|api.flumewater.com|me.GET:data:01:type-1:battery_level Sub: URL/api.flumewater.com/me.GET:data:01:type-1:battery_level Pub: the following Topic on Device command	high	☐	☐	☐	2022-02-10 11:06:39
17	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:bridge_id	6803870226318230084	☐			2022-02-10 11:06:39
18	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:building_type	SINGLE_FAMILY_HOME	☐			2022-02-10 11:06:56
19	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:city	Longboat Key	☐			2022-02-10 11:06:56
20	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:connected	true	☐			2022-02-10 11:06:39
21	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:country	United States	☐			2022-02-10 11:06:56
22	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:created_datetime	2022-02-10T07:15:00.000Z	☐			2022-02-10 11:06:58
23	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:device_id	6886483410961583712	☐			2022-02-10 11:06:58
24	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:id	5694328	☐			2022-02-10 11:06:58
25	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:installation	DONE	☐			2022-02-10 11:06:56
26	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:insurer_id	2	☐			2022-02-10 11:06:56
27	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:last_seen	2022-02-10T18:58:09.000Z	☐			2022-02-10 11:06:39
28	☐	☐	☐	☐		Sub: URL/api.flumewater.com/me.GET:data:01:type-1:location_id	72613	☐			2022-02-10 11:06:39
29	☐	☐	☐	☑	122	Dev: Unknown Sub: URL/api.flumewater.com/me.GET:data:01:type-1:message Pub: the following Topic on Device command	High Flow Alert triggered at LBK. Water has been r unning for 15 minutes averaging 9.92 gallons every minute.	☐	☐	☐	2022-02-10 11:06:58

This figure also shows the Association of the battery where the status shows “high”. By default, this is considered text by mcsMQTT so will have the text stored in DeviceString. To be able to trigger on the status VSP can be used to map the text into a number to be stored in DeviceValue. The radio for Control/Status UI is changed to Button (or List). “high” is the only state currently reported. When other states are reported they will be added to the VSP and unique values assigned.



The above selections resulted in the following HS Devices page. It contains a submit control to manually request data from an endpoint from each the flumetech and flumewater servers. These same controls contain two buttons to manually control polling of the endpoints that are contained in the FlumeWater.pub and FlumeTech.pub publication list files.

URL | api.flumetech.com

URL-api.flumetech.com-me (142)

me:Control (143)	0	Today 9:31:47 AM	SUBMIT STOP POLLING START POLLING

URL | api.flumetech.com

URL-api.flumetech.com-me.POST (153)

me.POST:data:01:Now:01:value (154)	1.8245178	Today 11:07:31 AM
me.POST:data:01:Today:02:value (155)	1138.46435448	Today 11:08:35 AM
me.POST:data:01:ThisMonth:03:value (156)	9646.95541572	Today 11:08:45 AM

URL | api.flumewater.com

URL-api.flumewater.com-me (146)

me:Control (147)	0	Today 9:31:56 AM	SUBMIT STOP POLLING START POLLING

URL | api.flumewater.com

URL-api.flumewater.com-me.GET (149)

me.GET:data:01:type (150)	2	Today 10:07:36 AM
me.GET:data:01:type-2:battery_level (151)	high	Today 10:03:28 AM
me.GET:data:01:type-1:message (152)	High Flow Alert triggered at LBK. Water has been running for 15 minutes averaging 10.27 gallons every minute. created at 2022-02-11T07:16:00.000Z	Today 10:07:36 AM

12.10 Emporia Energy Vue

Emporia Energy provides a cloud server from which real time energy usage can be obtained for each circuit that has been installed for monitoring. Their site is at https://www.emporiaenergy.com/.

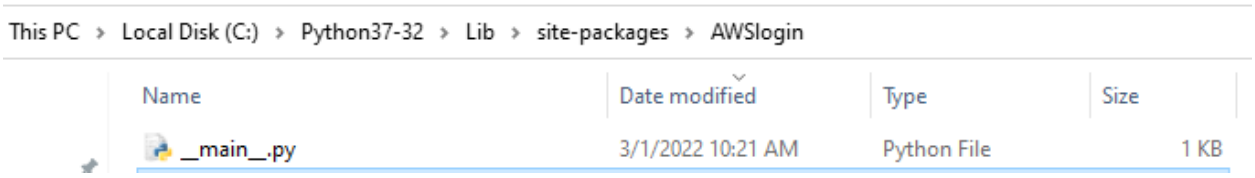
Figure 145 Emporia Hub and Circuit Clamps

Integration of their server with HS via mcsMQTT is started on the Cloud Page, URL Tab by entering the Emporia cloud server URL https://api.emporiaenergy.com, the endpoint query "/AppAPI?apiMethod=getDeviceListUsages&deviceGids={123456}&instant=$$UTCYEAR:-$$UTCMONTH:-$$UTCDAY:T$$UTCHOUR::$$UTCMINUTE::$$UTCSECOND:.0Z&scale=1MIN&energyUnit=KilowattHours" and a polling rate as shown in Figure 146. The protocol is GET and the authorization is oAuth2. The red text for deviceGids parameter is unique for each user. Getting this value will be discussed later in this section.

The API for Emporia Vue has been reversed engineered and the company has indicated that an official one will be published in 2022. Emporia uses the AWS servers. The login to these servers is with AWS's flavor of oAuth2. mcsMQTT utilizes an existing Python implementation of the cognito login to accomplish the authentication. This is expected to change to a native .NET implementation when the official API is published. There is no need to enter anything in the Cloud Page, URL Tab "oAuth2 URL and Data Payload" textboxes as this is currently handled with the Python glue.

The Python implementation is available at https://pypi.org/project/pyemvue/. It is installed from a command line with "pip install pyemvue". In addition, a Python module was developed as glue logic

between mcsMQTT and pyemvue login. The AWSlogin folder and __main__.py file in this folder is installed in the same location as pyemvue. The location will depend upon the computer. For a clean install on Windows the path is "C:\Python37-32\Lib\site-packages". AWSLogin that contains both the folder and file is available at http://mcsSprinklers.com/AWSLogin.zip. Unzip while maintaining the folder structure as shown in example below.



Also available at http://mcsSprinklers.com/AWSKeys.zip is template file for the Emporia account login email and password. This file is placed in the \data\mcsMQTT folder and edited to provide the account login information. This file will be updated by AWSLogin with a set of tokens after a successful login has been accomplished. It will continue to be updated as tokens expire and new tokens need to be obtained. This will occur typically on an hourly basis. No user action is need after then initial edit of the template.

mcsMQTT will execute AWSlogin.py when it receives a 401 unauthorized result. The expected path of Python is C:\Python37-32\python.exe. If this not the path on your computer then add a line under the [General] section of \config\mcsMQTT.ini that looks like below with the red text containing the path on your computer.

PythonPath= C:\Python37-32\python.exe

On a later RPi/Linux install additional steps were needed as the required cryptography libraries had to be locally built from source because of Linux hardware variations. The following steps were used. Note the third line below captures the interactive inputs for rust configuration.

```
sudo apt-get install python3-pip python-dev

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

2, arm-unknown-linux-gnueabihf , nightly, complete, Y, 1

sudo apt-get install libssl-dev

pip install pyemvue
```

The second issue on the Linux install is that during pyemvue install the python path environment was not updated to include the path to this and dependent modules. To overcome this a bash script was used that first set the environment variable and then executed the AWSlogin module. The script is shown below. I called the script AWSlogin.sh and placed it in the /Data/mcsMQTT folder.

```
#!/bin/bash

export PYTHONPATH=/home/mcs/.local/lib/python3.9/site-packages

/usr/bin/python3 $1 $2 $3
```

This script can be put anywhere and does need to be made executable (chmod AWSlogin.sh 755). Something like /Data/mcsMQTT or /scripts seems like a reasonable place.

It needs to be edited to put the path where AWSlogin module was placed in the PYTHONPATH line. Note that the PYTHONPATH environment is being explicitly stated here for whatever user is associated with the shell execution. In my case it was not the logged-in user that was running HomeSeer. It would be safer to add to existing PYTHONPATH rather than replacing, but I did not evaluate this approach. I found Google reference to do this with "export PYTHONPATH=/home/mcs/.local/lib/python3.9/site-packages:$PYTHONPATH.

It needs to be edited to put the path to python if not /usr/local/python3.

/Config/mcsMQTT.ini needs the PythonPath= xxx wherer xxx is the path to where this script was placed. For example
Code:

```
PythonPath="/usr/local/HomeSeer/Data/mcsMQTT/AWSlogin.sh"
```

Figure 146 Emporia Vue mcsMQTT Setup

The polling endpoint setup has a scale parameter which can any of the following `[1S, 1MIN, 1H, 1D, 1W, 1MON, 1Y].` 1MIN is a reasonable value. It means that Emporia will provide the KwH measurement for data collected over the past one minute. In the setup of Figure 146 the pollilng rate is shown as every 10 seconds. This means the data will be refreshed every 10 seconds for the amount of energy used over the past 60 seconds.

mcsMQTT converts the KwH data received into Watts in the HS Device Features. It does this by using the Expression text box of the Edit tab to use the appropriate conversion based upon the scale parameter being used. This is all done automatically by mcsMQTT with no user action needed. If the user changes the scale parameter, then mcsMQTT will change the conversion expression. For example, a KwH reading that was taken with scale=1S will have an expression 3600*1000 to handle the

conversion from hours to seconds and from killowatts to watts. For scale=1MIN the conversion is 60*1000, etc.

mcsMQTT automatically creates HS Device Features for the ":usage" JSON keys of the data from the Emporia server. It gives the Feature the value of the ":name" JSON key. This is shown in Figure 147.

Emporia JSON payload is provided as arrays. This results in multiple rows on the mcsMQTT Association table of the MQTT Page with similar data. It is possible to "A"ssociate other rows into HS Device Features, such as ":percentage". If this is done then take care to associate only one of the multiple rows that could provide the same data.

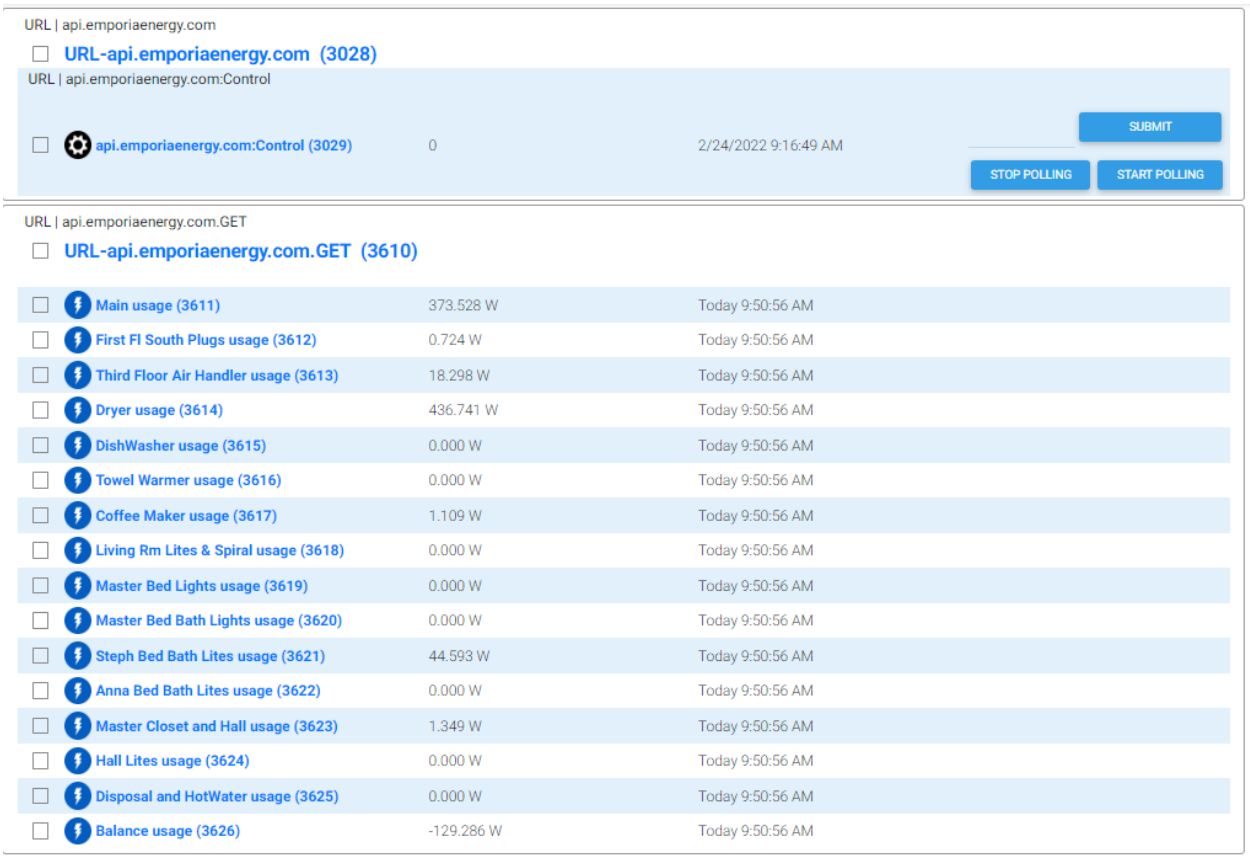


Figure 147 Emporia Usage as HS Device Features

At the top of in Figure 147 is the polling control buttons and interactive endpoint text box. The buttons are used to start and stop the polling. These will not normally be used. The text box is used to send one-time requests to the Emporia server.

The specific need for this text box is to obtain the deviceGids value that is needed to poll the energy data that was setup on the Cloud Page, URL Tab. Enter (without quotes) "`/customers/devices`". This should result in a request being sent. If accepted then the MQTT Page, Association Tab will show

many rows of data about the Emporia devices. The payload column will show the deviceGid for each device such as:

Sub: URL/api.emporiaenergy.com.GET:channels:02:deviceGid 123456

This number will be used in endpoint text box of the Cloud Page, URL tab for the Emporia row.

There is no longer any need for the data in the Association Table that was just populated to get the deviceGid. To clean this up go the MQTT Page, General Tab, Obsolete row text box and enter (without quotes) "URL/api.emporiaenergy.com.GET:#"

The step-by-step setup of the pieces described above is itemized below.

1. Install PyEmVue from command line with "pip install pyemvue" in command/terminal window. Note possible variation between Windows and Linux described above to have success with the install.
2. Download AWSlogin and place in Python packages folder
3. Download AWSKeys, unzip into \data\mcsMQTT folder and edit for login credentials
4. Edit \config\mcsMQTT.ini if the path to python.exe is not C:\Python37-32\python.exe. Note again a variation in Winows vs. Linux where Linux may require use of an additional bash script and change to haow PythonPath is specified.
5. On Cloud Page, URL tab enter URL https://api.emporiaenergy.com, GET radio for protocol, oAuth2 as authorization
6. On HS Devices page select URL for Room and locate the Feature api.emporiaenergy.com:Control. In the text box enter `/customers/devices`
7. On MQTT Page, Association Tab, look for the deviceGid in Payload column and write it down. Using the J3 filter for deviceGid on the Topic filter may make it easier to find the six-digit deviceGid in the Payload column.
8. Optionally, on MQTT Page, General Tab, Obsolete Row textbox enter `URL/api.emporiaenergy.com.GET:#` to remove the data that was just obtained for the deviceGid
9. On Cloud Page, URL tab enter the following after updating the deviceGids number and scale if desired /AppAPI?apiMethod=getDeviceListUsages&deviceGids={123456}&instant=$$UTCYEAR:-$$UTCMONTH:-$$UTCDAY:T$$UTCHOUR::$$UTCMINUTE::$$UTCSECOND:.0Z&scale=1MIN&energyUnit=KilowattHours
10. On Cloud Page, URL tab enter polling rate that should be consistent with the scale parameter. For example, if scale=1D then no need to poll every second, but perhaps 1 hour would be appropriate.

If an Emporia outlet is being used and the desire is to also control this outlet from HS then an “A”ssociation will need to be made on the Topic that reports the outlet state and some changes made on the Edit tab.

The Topic is provided in the same URL endpoint that is used to determine the Gid and will end with “outletOn” such as row 1 in Figure 148. Once “A”ssociated with the “A” checkbox then the HS Ref will be assigned and the Edit tab accessed by clicking on it. There needs to be a publish topic defined so the HS buttons will be created and the VSP modified for case sensitivity and to change True/False from Emporia to On/Off in HS. These edits are shown in Figure 150.

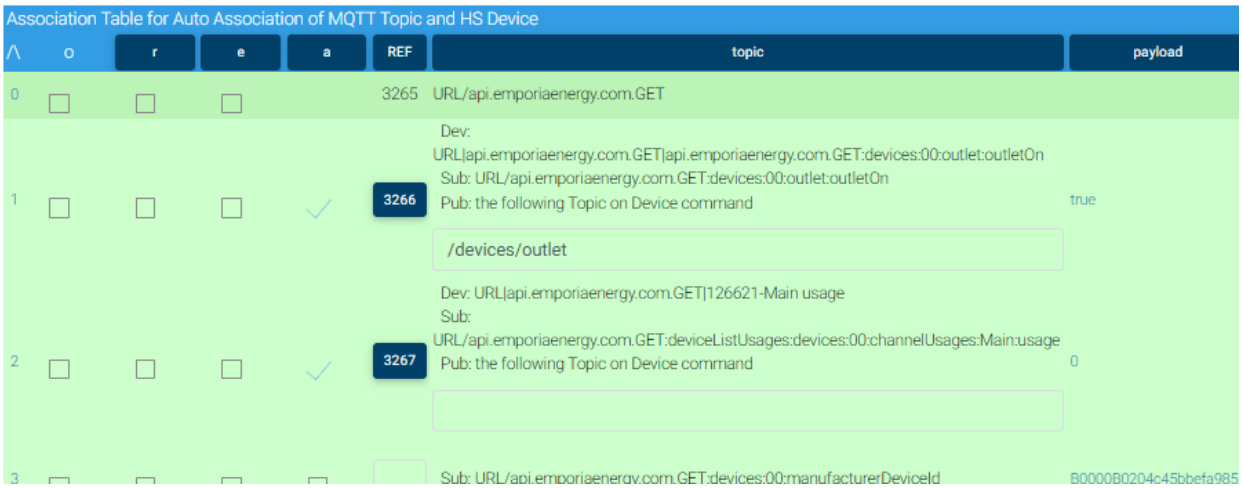


Figure 148 Emporia Outlet Topic

The URL endpoint on the Cloud Page URL tab also needs to be changed from polling the single endpoint to get usages to also poll the devices endpoint to get the outlet status. This is done by using the name of the publist file for the endpoint rather than the explicit single endpoint. See Figure 149.

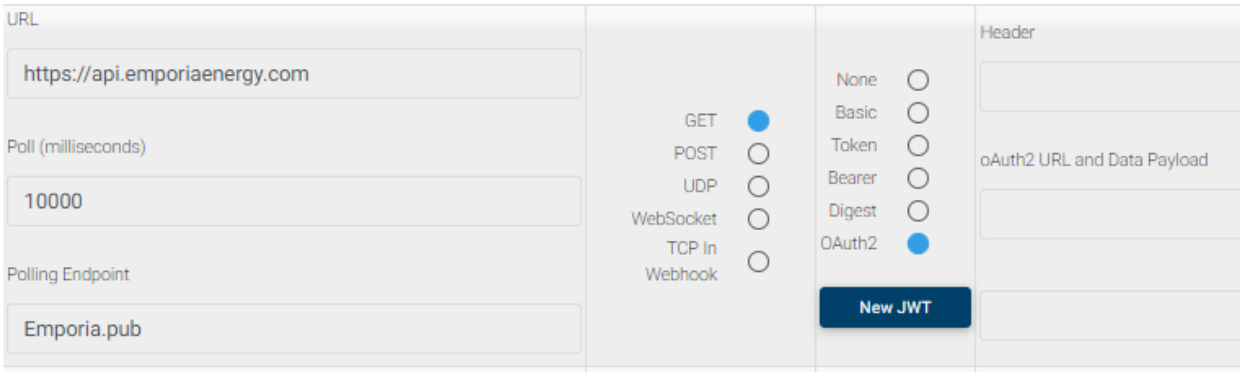


Figure 149 Emporia Outlet Publist Endpoint

An example \data\mcsMQTT\Emoria.pub file contents is shown below where Gid shown in red will be user dependent. Note multiple Gids are entered with "+" separator such as 123456+654321. Note that the Emporia querystring contains "=" so it needs to be escaped with "\=" because mcsMQTT uses the "=" to separate the URL querystring from the data that is sent with POST and PUT methods.

```
$$1:=URL/https://api.emporiaenergy.com
$$2:
$$3:
$$4:
$$1:/AppAPI?apiMethod\=getDeviceListUsages&deviceGids\={/AppAPI?apiMet
hod\=getDeviceListUsages&deviceGids\={126621}&instant\=$$UTCYEAR:-
$$UTCMONTH:-
$$UTCDAY:T$$UTCHOUR::$$UTCMINUTE::$$UTCSECOND:.0Z&scale\=1MIN&energyUn
it\=KilowattHours}&instant\=$$UTCYEAR:-$$UTCMONTH:-
$$UTCDAY:T$$UTCHOUR::$$UTCMINUTE::$$UTCSECOND:.0Z&scale\=1MIN&energyUn
it\=KilowattHours=
$$1:/customers/devices=
```

Settings for Plugin Device

HS Device Publish Topic devices/outlet

HS Device Control/Status UI

- Unspecified
- Button
- Number
- NumberChange
- Slider
- CSV
- Text
- List
- RGB
- RGBW
- HSB
- ColorXY
- Sign
- Ramp
- Toggle
- jpg File

HS Device API Type and SubType

Generic

Battery

HS Device Location

Loc2 (Floor) URL

Loc (Room) api.empo..y.com.GET

Name api.emporiaenergy.com.GET:devices:03:outlet:outletOn

HS Device VSP List

Max number of VSP 12

Payload false=0;Off;Off VSP

Payload true=1;On;On VSP

Add/Edit

Clear existing VSP

Figure 150 Emporia Outlet Edits

12.11 Coulisse B.V. Motion-Blinds.com Blinds Control

WiFi controlled motion blinds can be obtained from https://motionblinds.com/ with a WiFi hub as described at https:/d.otto.de/files/65ff764c-b2c7-538f-b555-4751394351b5.pdf.

Several other brands are known to work as well including the llist below from a HomeAssistant integration. Other information included herein was also derived from the integration described at Motion Blinds - Home Assistant (home-assistant.io).

> AMP Motorization, Bliss Automation - Alta Window Fashions, Bloc Blinds, Brel Home, 3 Day Blinds, Dooya, Gaviota, Havana Shade, Hurrican, Shutters Wholesale, Inspired Shades, iSmartWindow, Martec, Motion Blinds, Raven Rock MFG, Smart Blinds, Smart Home, Uprise Smart, Shades

An API reference is avaialbe on post #10 at Possible to controll blinds from "motion-blinds.com" / Coulisse B.V. ? - HomeSeer Message Board. Communications are on UDP port 32101 from the WiFi hub and on 32100 back to the hub. Multicast IP 238.0.0.18 is used by the WiFi hub.

Integration with HS is setup from the mcsMQTT Cloud Page, URL Tab by entering the IP of the WiFi hub using port 32101 such as shown in Figure 151.

Figure 151 Setup for Coulisse B.V. Blinds

The WiFi hub will be sending a “heartbeat” message on 238.0.0.18:32101 every 30 or 60 seconds. mcsMQTT will be listening for this message. If a heartbeat is not heard for a minute, then mcsMQTT will simulate the heartbeat. This is to deal with the installations that do not reaceive the multicast UDP messages.

When received, the plugin will send a request on the URL IP that was setup to ask for the list of devices being supported by the WiFi hub. From this list mcsMQTT will create a HS Device and set of Features for each such as shown in Figure 152.

There are three types of blinds supported by the WiFi bridge. Standard, Top-Down/Bottup Up, and Double Roller. While the structure is in place to support all types, at this time the Standard blinds are the only of the three that are implemented. Users with other types are needed to complete the integration testing.

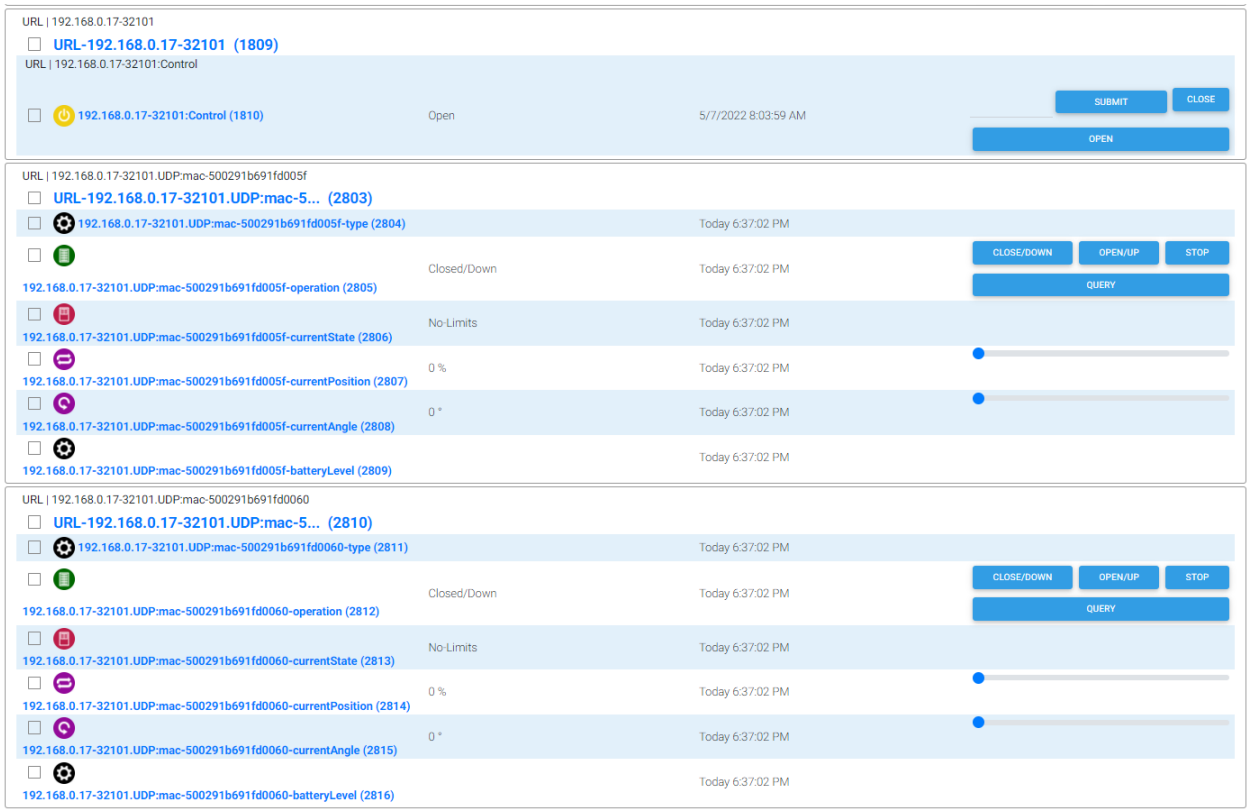


Figure 152 Coulisse B.V. Blinds HS Device and Features

To be able to interact with the WiFi hub an access token is needed. The token is a 16-bit AES-128 encryption of the token provided in the heartbeat message using an encryption key that can be obtained from the smartphone App for the blinds. Obtaining this key is described below.

> The Motion Blinds API uses a 16 character key that can be retrieved from the official “Motion Blinds” app for IOS or Android.
>
> Open the app, click the 3 dots in the top right corner, go to “settings”, go to “Motion APP About”, Please quickly tap this “Motion APP About” page 5 times, a popup will appear that gives you the key.
>
> Please note that “-” characters need to be included in the key when providing it to Home Assistant. The key needs to be similar to 12ab345c-d67e-8f

The secrey encryption key is entered on the mcsMQTT Cloud Page, URL Tab as shown in Figure 151.

The payloads used to communicate use JSON format. The decoded data received can be see on the MQTT Page, Association Tab. There is more information available than is automatically used by mcsMQTT to create the HS Device and Features. This data can also be “A”ssociated with HS from the “A” column checkbox of the Association Tab. Similarly, the “A” checkbox can be unchecked for Features that are not of interest within HS. See Figure 153

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	s	l	lastdate
0	☐	☐	☐		2143	URL/192.168.0.17-32101.UDP					
1	☐	☐	☐	☐		Sub: URL/192.168.0.17-32101.UDP:data:currentState	1	☐			2022-04-30 19:43:40
2	☐	☐	☐	☐		Sub: URL/192.168.0.17-32101.UDP:data:numberOfDevices	3	☐			2022-04-30 19:43:40
3	☐	☐	☐	☐		Sub: URL/192.168.0.17-32101.UDP:data:RSSI	-21	☐			2022-04-30 19:43:40
4	☐	☐	☐	☐		Sub: URL/192.168.0.17-32101.UDP:deviceType	02000001	☐			2022-04-30 19:43:40
5	☐	☐	☐	☐		Sub: URL/192.168.0.17-32101.UDP:mac		☐			2022-04-30 19:43:13
6	☐	☐	☐	✓	2149	Dev: URL|192.168.0.17-32101.UDP:mac-500291b691fd|192.168.0.17-32101.UDP:mac-500291b691fd-batteryLevel Sub: URL/192.168.0.17-32101.UDP:mac-500291b691fd:batteryLevel Pub: the following Topic on Device command		☐	☐	☐	2022-04-30 19:43:13
7	☐	☐	☐	✓	2148	Dev: URL|192.168.0.17-32101.UDP:mac-500291b691fd|192.168.0.17-32101.UDP:mac-500291b691fd-currentAngle Sub: URL/192.168.0.17-32101.UDP:mac-500291b691fd:currentAngle Pub: the following Topic on Device command		☐	☐	☐	2022-04-30 19:43:13
						192.168.0.17:32100					

Figure 153 Coulisse B.V. Association Table

mcsMQTT send data to the WiFi hub using port 32100. This is the Publish Topic on the Association Tab. It uses the Payload Template to format the JSON such as below:

```
{"msgType":
"WriteDevice","mac":"500291b691fd","deviceType":"$$PAYLOAD:(URL/192.16
8.0.17-32101.UDP:mac-500291b691fd:deviceType):","AccessToken":
"<<AES128("$$PAYLOAD:(URL/192.168.0.17-
32101.UDP:mac:token):","$$SECRETKEY:")>>","msgID":"<<$$UNIX:>>","data"
:{"targetPosition":$$CAPIVALUE:}}
```

Replacement variables are used to dynamically complete the payload. deviceType is obtained from the rececieved JSON of the deviceType JSON key.

```
"deviceType":"$$PAYLOAD:(URL/192.168.0.17-32101.UDP:mac-
500291b691fd:deviceType):",
```

AccessToken gets the token received in heartbeat message, the SecretKey entered on the URL tab, and the AES128 encryption function.

```
"AccessToken": "<<AES128("$$PAYLOAD:(URL/192.168.0.17-
32101.UDP:mac:token):","$$SECRETKEY:")>>",
```

msgID needs to change with each transmission so the Epoch time which changes every second is used to get a changing value.

```
"msgID":"<<$$UNIX:>>",
```

Data contains the command to be executed which is from the HS CAPI Contorl that was the event to send the message.

```
"data":{"targetPosition":$$CAPIVALUE:}}
```

mcsMQTT makes a request back to the WiFi hub each time it receives a heartbeat (or simulated heartbeat) message. It will make a request for one device supported by the hub and after all device statuses have been requested it will ask again for the list of devices. This methodology will allow new devices to be added to the hub and for status to be refreshed periodically without any user action necessary.

The WiFi hub used “mac” as the key to uniquely identify each device it is supporting. The “mac” key has been setup as an Elevated JSON key so it looks like part of the Topic thus allowing the data to be separately viewed for each device reported by the WiFi hub.

The battery status is reported as a voltage. mcsMQTT assumes a two-cell battery pack is being used which has a full change reading of 8.4V. Three cells have 12.6V. Four cells have 16.8V. If two-cell is not being used then the mcsMQTT Edit Tab, expression text box should be edited to correctly capture the range of the battery.

12.12 Thermostats

12.12.1 NuHeat Thermostat

NuHeat provides a WiFi-enabled thermostat used to control floor heating. The device is managed via a Cloud connection and this connection is used to integrate with HomeSeer.

Figure 154 NuHeat Thermostat

The setup of each NuHeat thermostat is done from the Cloud Page, NuHeat tab as shown in Figure 155. If multiple thermostats are used then a semicolon is used to separate the Id of each. This Id is available from the NuHeat site account login.

Polling can be setup to keep HS in sync with the NuHeat server and the thermostat. If not setup then a status update in HS will only occur 20 seconds after a HS control action is taken such setting a temporary setpoint.

Normal operation of the thermosat is to run off of its internal schedule. This schedule can be interrupted by selecting a SetPoint Temperature. This will be the new temperature for heating control for the next hour. The ScheduleMode will reflect TempHold status.

The Schedule Mode can be changed to indefinite hold of the SetPoint Temperature with the Hold button. The Run button on Schedule Mode is used to restore the normal internal schedule of the thermostat.

Status is shown for the equipment being in heating vs. quiescient operation. Status is also shown for the temperature sensed by the thermostat.

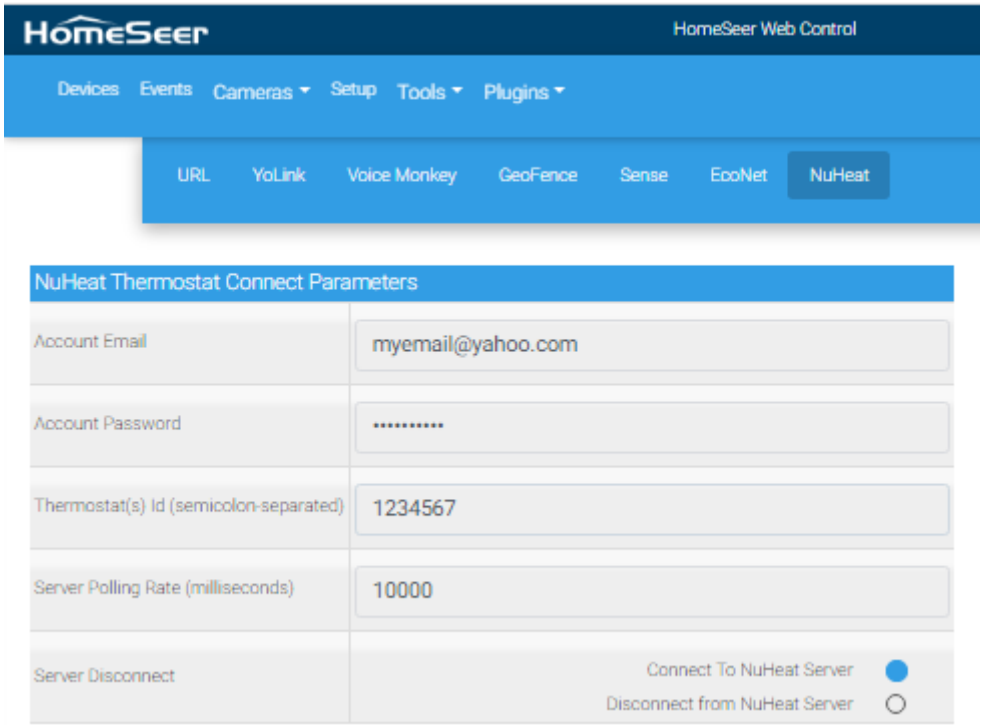


Figure 155 NuHeat Thermostat Setup

The following information is provided by the NuHeat cloud server. Each item is visible in the Association table of the MQTT Page. Those in blue are automatically mapped into HS Device and Features as shown in Figure 156.

SerialNumber:1234567	MaxTemp:7000
Room:NUHEAT	MinTemp:500
GroupName:	ErrorCode:0
GroupId:-1	Confirmed:true
GroupAwayMode:false	Email:myemail@yahoo.com
Temperature:2050	TZOffset:-06:00
SetPointTemp:2000	Assigned:true
ScheduleMode:1	FloorArea:75
OperatingMode:1	KwCharge:0
HoldSetPointDateTime:2022-06-13T02:00:00+00:00	WPerSquareUnit:12
Online:true	SWVersion:201
Heating:false	HasBeenAssigned:true

	DistributorId:1

Figure 156 NuHeat Thermostat Device(s) and Features

12.12.2 Nexia / Trane / American Standard Thermostat

Figure 157 Trane Thermostat

Nexia thermostats are branded as Nexia, Trane, or American Standard. They are integrated via a cloud server. Two servers are available. One used to support a desktop/browser access and one to support a mobile App. There is overlapping information. mcsMQTT toggles between each server to get the full set of data of interest. The integration is done by emulating the operation of the desktop and App.

The cloud server login requirements are username and password, the house id of the thermostat(s) and the brand. Polling rate is also sepecified in the setup as shown in Figure 158.

Nexia/ASAir/Trane Thermostat Connect Parameters	
Account Email	somebody@yahoo.com
Account Password	•••••••••
House(s) Id (semicolon-separated)	1234567
Brand	nexia ○ asair ○ trane ●
Server Polling Rate (milliseconds)	60000
Server Disconnect	Connect to trane Server ● Disconnect from trane Server ○

Figure 158 Nexia / Trane / American Standard Thermostat Setup

There is much information available from the servers. A subset is selected for mapping into HS Devices and Features. A thermostat device is created and one or more zone devices are created. In the case of a single zone, some zone information is contained in the thermostat device based upon how the cloud server reports. A combination of controllable and status only features are created for each device as shown in Figure 159.

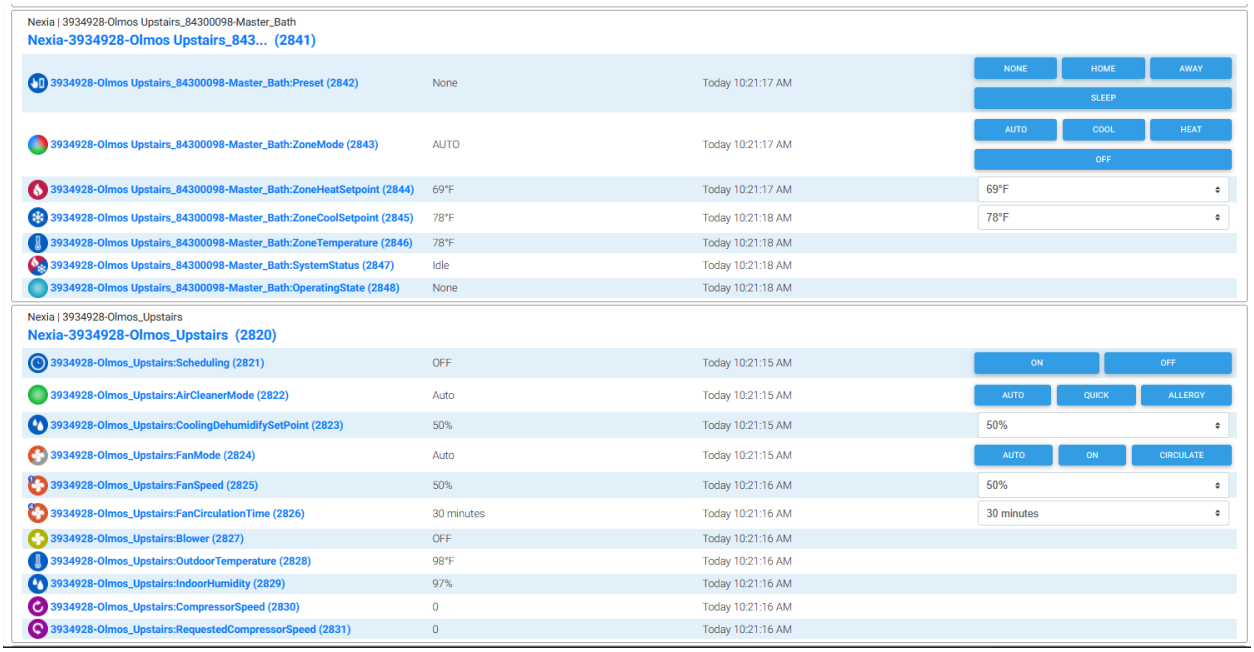


Figure 159 Nexia HS Devices

The are two cloud logins that are used. One intended for mobile applications (first URL) and one for the desktop applications (second URL). The mobile application includes configuration information so is used for most of the setup of HS Devices and Features. In the case where login does not give access to the mobile site, the standard site is used, but controls created will not be functional. The second site just contains data of current state. Some information is available on only one of the two sources. Some is available on both. The first URL is polled at 20% of the polling rate and the second at 80%. This means mode, scheduling, etc. is only updated every fifth polling interval and the others are update on four of five polling intervals. The specific features and source are shown below.

Blower - Only 2nd URL	ZoneMode
CompressorSpeed - Only 2nd URL	Scheduling
RequestedCompressorSpeed - Only 2nd URL	OperatingState - Only 1st URL
Emergency Heat	SystemStatus - Only 1st URL
Aircleaner	HeatingSetpoint
DehumiditySetPoint	CoolingSetpoint
FanMode	Temperature (if available)
FanSpeed	Humidity (if available)
FanCirculationTime	Preset (if available)
OutdoorTemperature	
IndoorHumidity	

12.12.3 Carrier Infinity / Bryant Evolution / Ion

The Carrier family of advanced HVAC equipment provides a System Access Module (SAM) that provides a bridge between the Carrier Internet server and the RS-485 serial communications used between the thermostat and the HVAC equipment. mcsMQTT has implemented a means to use the cloud server to get status and control the thermostat. There was also a reverse-engineering of of the RS-485 called Infinitude that is not supported by the plugin.

The cloud API provided by Carrier has been encapsulated using Python as is available as a library module under the name carrier-api. A Python, **version 3**, install on the same computer as mcsMQTT is needed to run this library. The Pyton install varies based upon the OS.

A Python library is available with instructions and repository at https://pypi.org/project/carrier-api/ . The Python MQTT and HTTP library is also needed. They are most easily installed from the command line / terminal window using PIP with the command

```
pip install carrier-api

pip install paho-mqtt

pip install aiohttp
```

When both Python2 and Python3 are in the environment then

```
python3 -m pip install carrier-api

python3 -m pip install paho-mqtt

python3 -m pip install aiohttp
```

If Python3 and PIP are not yet installed on the Homeseer computer they first need to be installed. Version 3.10 and 3.11 are known to work with the Carrier-API library. Use Google for guidance on installing them on the OS that is hosting Homeseer. After installation, the folder where python.exe needs to be identified as it varies. There will also be a \Scripts subfolder that normally is a subfolder of where python.exe is located on Windows.

The integration of this library with Homeseer is done with the Python script CarrierRequest.py that is available in the mcsMQTT download package and originally installed at subfolder \bin\mcsMQTT. It will not be run from this location, but is moved to the \Scripts subfolder of the python install or an alternate location on Linux.

During installation of Python, the PYTHONPATH environment variable is normally updated with path to wherer the Pyython libraries are installed. PYTHONPATH will be discussed later.

The mcsMQTT setup for Carrier integration is on the Cloud Page, Thermostats Tab. See Figure 160.

Note the full path to python.exe and the Scripts folder in the setup. Also needed are the email and password to the account that was setup with Carrier.

Data is polled for status updates to handle the local control being synced with HS. Provision is also provided to disconnect the connection with the cloud server. This disconnection will also result in the Python script CarrierRequest.py being terminated. CarrierRequest.py is managed by mcsMQTT and run when the setup is complete and not disconnected.

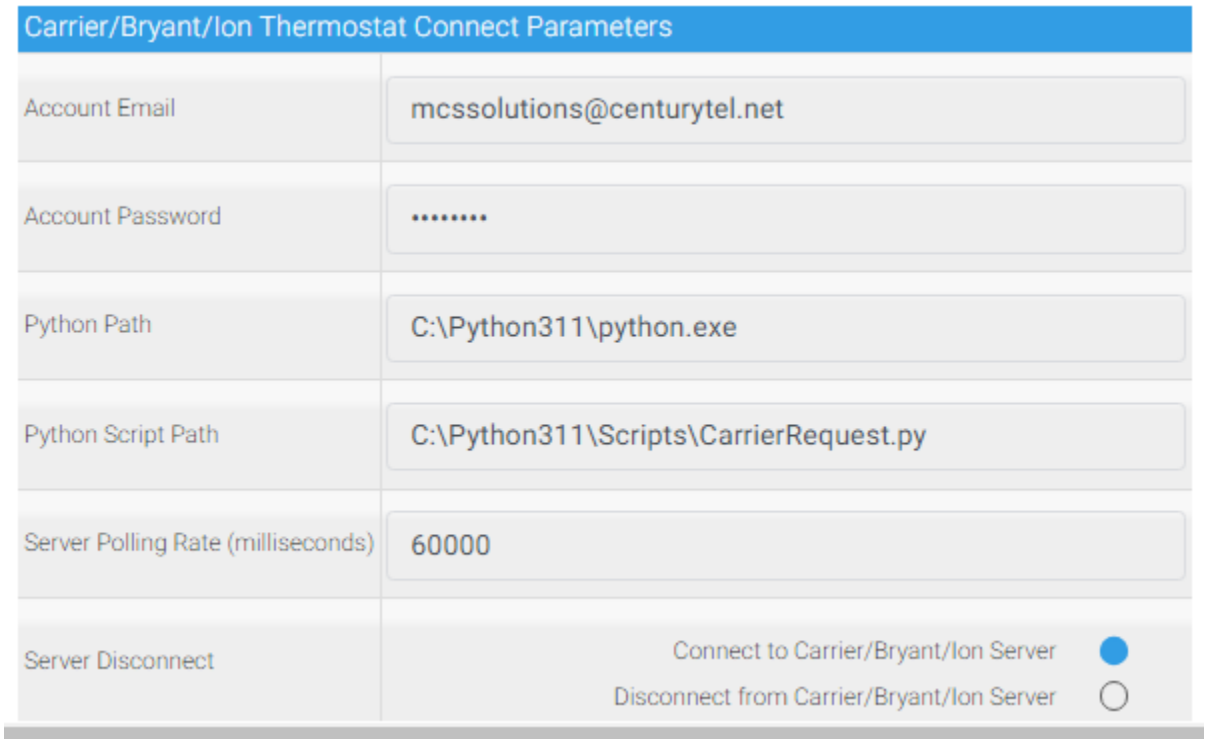


Figure 160 Carrier Thermostat Setup

mcsMQTT will launch CarrierPython.py using the default shell account. If the Python libraries were installed with a different account, which is the normal Linux situation, then Python cannot be executed directly because PYTHONPATH has not bee defined for the shell account. In this situation then it needs to be launched via a script that also contains the definition of the PYTHONPATH.

A sample script for Linux will contain something like below when the path to the site-packages Python version being used and the path to where the CarrierRequest.py was placed. It also requires that mcsMQTT be told to run the script rather than Python. The setup is the same as shown for Omnilogic below with the name Omnilogic changed to Carrier.

```
#!/bin/bash

export PYTHONPATH=/home/mcs/.local/lib/python3.11/site-packages

/usr/bin/python3 /home/mcs/Scripts/CarrierRequest.py $1 $2 $3 $4 $5 $6
```

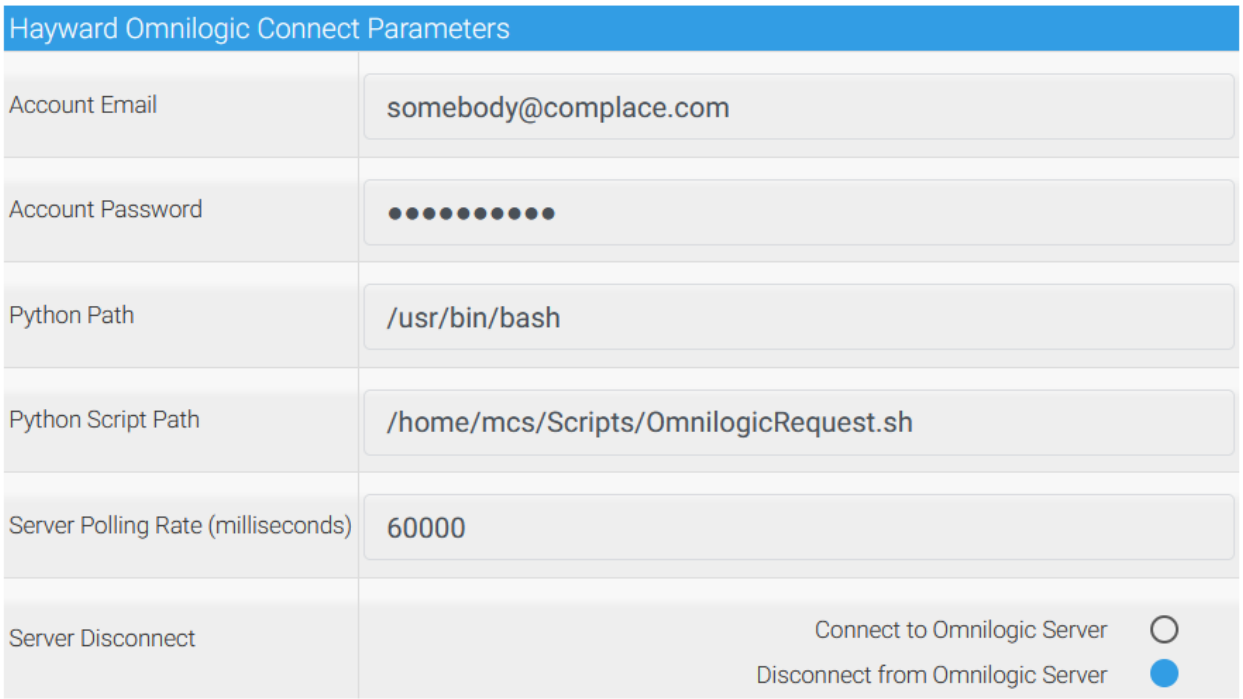


Figure 161 Launching Python via a Bash Script

In summary, the setup consists of installing Python and two libraries and then filling in the setup info on the MQTT Cloud Page, Thermostats Tab. A successful install will result in HS Device and set of Features being created such as shown in Figure 162.

A failed install has multiple places to look for clues. mcsMQTT Debug.txt is enabled from the MQTT Page, General Tab. The Python \Scripts folder will have CarrierDebug.txt. MQTT messages on Topic Carrier/Response will also exist that may contain Python execution traceback information.

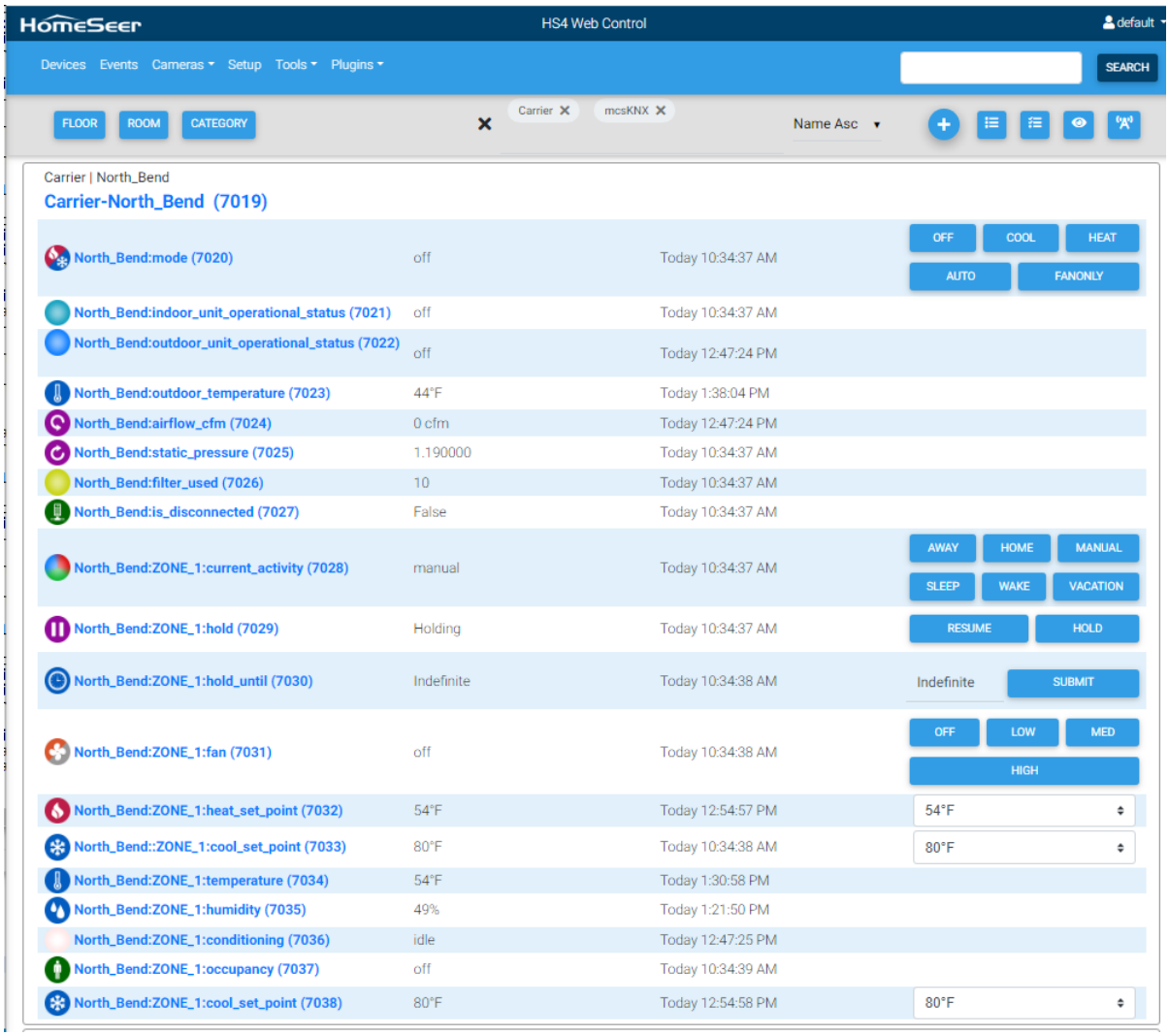


Figure 162 Carrier Thermostat HS Device and Features

12.13 Tank Utility

Tank Utility is instrumentation to monitor Liquid Propane (LP) tanks. The commercial product consists of the sensor and a smartphone app. It can be purchased on Amazon at https://www.amazon.com/Generac-7009-Tank-Monitor-White/dp/B09WZGK4FL . There is a published API at http://apidocs.tankutility.com/ which is what the integration with mcsMQTT is based. The integration is based upon periodically polling the data available from the Tank Utility server and making the data available in a convient presentation within HS. The data is uploaded from the sensor to Cloud server once per day. mcsMQTT synchronizes to this time and then polls at 10 minute intervals 24 hours later to get the next reading.

Multiple tanks can be integrated. For each tank the following JSON payload is available. Those items in yellow highlight are used for the HS integration. If other items are desired then an update can be done.

```
device_id:00270037323237310347303 5
short_device_id:ABCDEFGH
name:Tank 1
address:123 MyStreet,My Town, My State, My Country
account_id:
fuel_type:propane
fuel_dealer_id:
connection_type:wifi
product_id:WIFI-G
product_name:Generac Tank Monitor
supplier_id:-supplier123
status:deployed
capacit :25
orientation:horizontal
consumption_types:cooking,fireplace
consumption_type{11}
   backup_heating:false
   bulk_storage:false
   commercial_industrial:false
   cooking:true
   fireplace:true
   generator:false
   heating:false
   hot_water:false
   laundry_dryer:false
   pool:false
   retail_fill_up:false
battery_warn:false
battery_crit:false
battery_level:good
average_consumptio:0.10714285714285714
```

```
        estimated_fill_da:2024-03-23T21:08:01.600Z
        fixed_transmission_time:-1
        reading_interval:21600
        transmission_interval:86400
        threshold_1:-1
        threshold_2:-1
        change :-
        lastReading{12}
            tan :56.47379
            temperature:58.782
            time:1680590098000
            time_iso:2023-04-04T06:34:58.000Z
            sw_rev:12.005
            event_code:0
            fixed_transmission_time:-1
            reading_interval:21600
            transmission_interval:86400
            threshold_1:-1
            threshold_2:-1
            change_of_value:-1

        telemetry[5]
            0{13}
                attempt_no:0
                chn:1
                cipher:229
                ecn:3
                http_status_code:200
                modem_ram:51032
                module_temp:0
                module_voltage:0
                rssi:-87
                ssid:MySSID
                time_to_conn:17.395
                tlm_time:1680590120
                type:wifi
device {29
e      }
```


The setup includes account username/email and password. This is available on the Cloud Page, Tank Utility Tab such as Figure 163. An option also exists to disconnect/reconnect to the Tank Utility server.

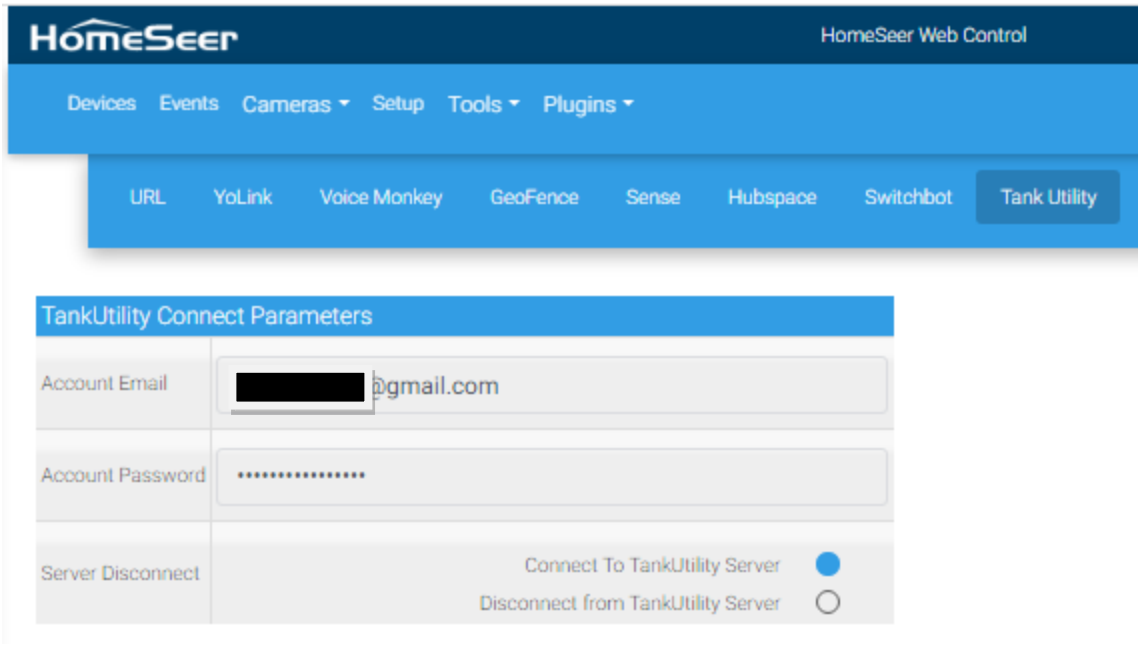


Figure 163 Tank Utility Setup

One HS Device is created for each tank with the properties of interest shown as Features. See Figure 166. This information is also available on the MQTT Page, Association Tab. Customizations can be done from this location such as selecting a measurement for historical recording and subsequent charting. The example in Figure 164 shows where the tank level measurement has been selected for "s"hort term storage. The History Tab of the MQTT is used to define the when the data is removed from short term strorage. Perpahs a month or a year. It can also be stored in external database for "l"ong term storage.

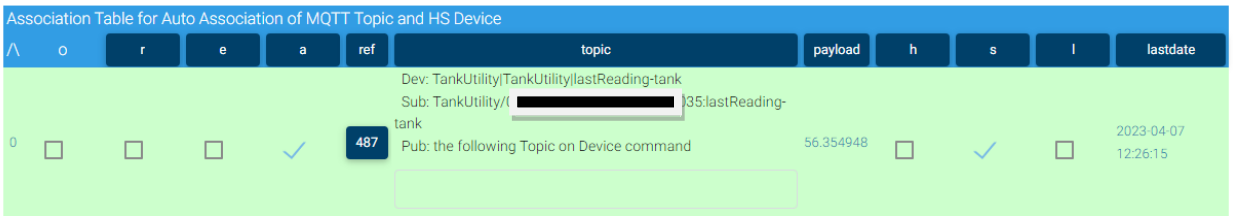


Figure 164 Tank Utility Association Table Entry

Selecting a chart is done from the MQTT Page, Chart Tab where the measurements to appear on the same chart are selected and the period of time of the data being drawn. An example is shown in Figure 165.

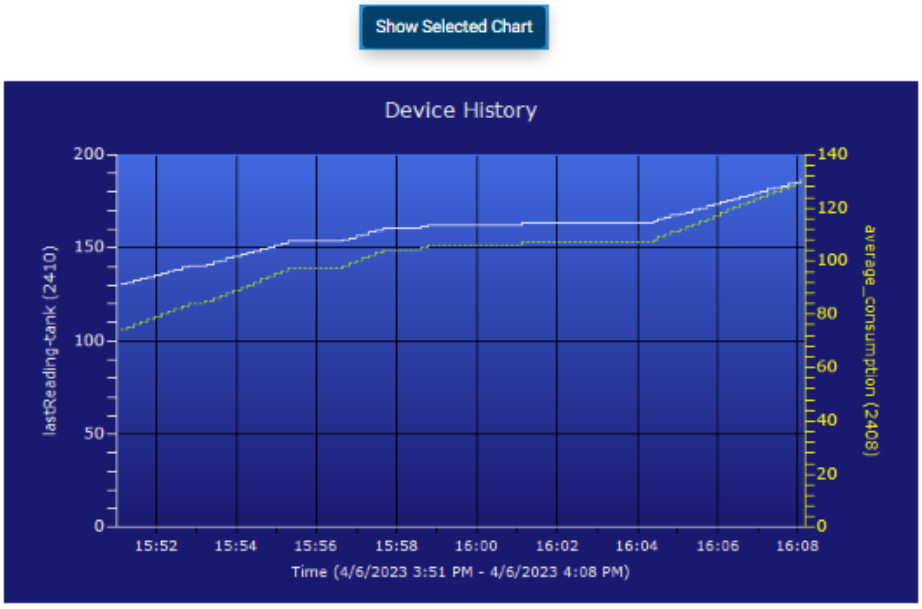


Figure 165 Tank Utility Measurement Chart

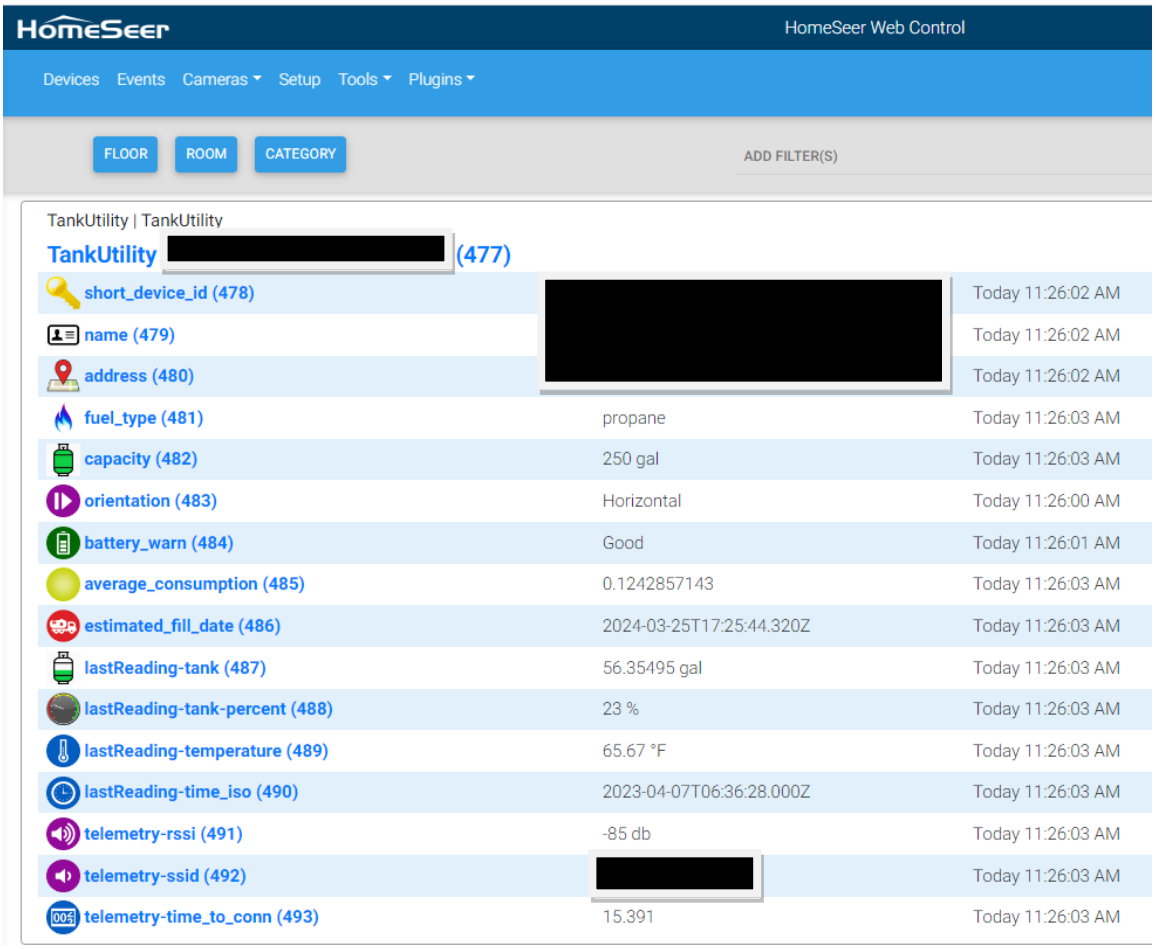


Figure 166 Tank Utility HS Device and Features

12.14 Abode Security

Abode provides a security panel at a reasonable price that utilizes modern technologies such as WiFi and Zigbee for sensors and actuators and a wired or wireless connection to the internet. All activity is managed through their cloud server with primary UI being a smartphone. They provide a monitoring service, but is not required for normal operation.

Abode does not provide a public API. Data availalble has been reversed engineered. Much of the mcsMQTT development has levereged the Hubitat Groovy https://github.com/jorhett/hubitat-abode/wiki and Home Assistant Python https://github.com/MisterWil/abodepy implementations. Both of these developments are stale with Abode having added new devices and are not contained in these references. What is included in mcsMQTT is based upon devices in possision or with cooperation of other HS users. In general all status information is available in HS Device and Features. Most controls are also available, but not all due to the lack of information on the expected control parameters.

The integration to HS has a minimal setup of email and password as shown in Figure 167. The email and password are the same as used when installing the security system via smartphone.

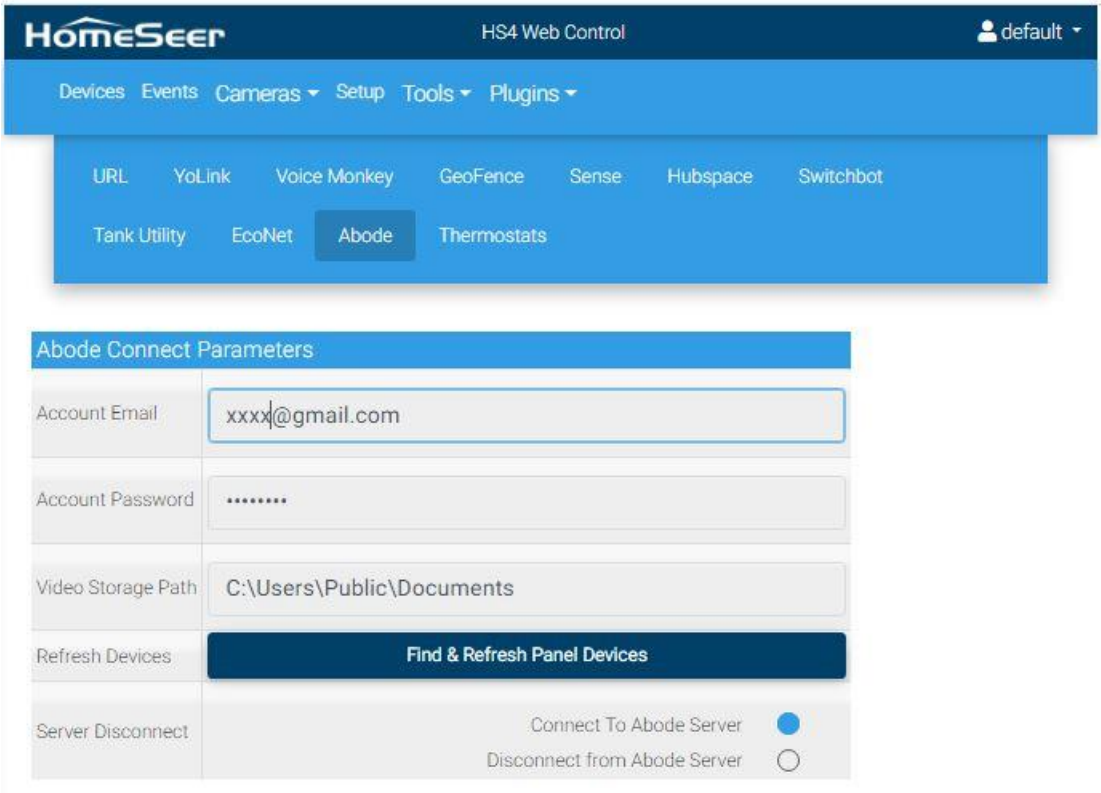


Figure 167 Abode Setup Parameters

On startup mcsMQTT will request information about the panel and devices that have been configurerd on the account. HS Devcies and Features will be created such as shown in Figure 168.

The plugin will establish a WebSocket with the Abode server. This conduit is the primary mechanism to receive status update of the panel and devices. In some cases the event reported will contain new status information. In other cases the event report will include only the identification of the device that has had some type of status change (e.g. color change in light bulb). In these cases mcsMQTT will request a status update of the device from with the HS Feature status is updated.

The primary interface with the panel is the mode information. The panel supports two areas that are independently managed. Devices will be setup to be in one of the two areas via the smartphone App. When mcsMQTT, fob, or smartphone app commands a mode change, the event will be reported, but the event report will not provide which area has had a mode change. The change is available immediately when requested for devcie changes, but panel changes are not immediately availalble. mcsMQTT waits 60 seconds to request the panel status update which will include the mode status of both areas.

Two timers that countdown every second during arming and prior to alarm that mimics the timers contained in the Abode panel. They are only infomational.

The plugin includes a Connection Featue that has three states. Normal, Inactive, and Failure. It is managed as part of the Pong response from Abode server on the WebSocket which occurs every 25 seconds. If it has not been received for 120 seconds then the Normal state will transition to the Inactive state and an attempt made to restart the connection. If the conection is not restored in 20 seconds

then the status will migrate to the Failed state. This monitoring is in addition to monitoring that is done behind the scenes with auto recovery attempts done automatically.

Panel mode management is focused on the Area 1 Mode. Abode does not provide event notification that can be used to distinguish Area 1 from Area 2. The HS Area 1 Mode Feature is updated based upon event reporting for the panel or event reporting for the Fob. Fob status will show the last event state reported for the fob which could be different than the last event reported for the panel. Variance will typically occur if the Fob is not used to change the mode.

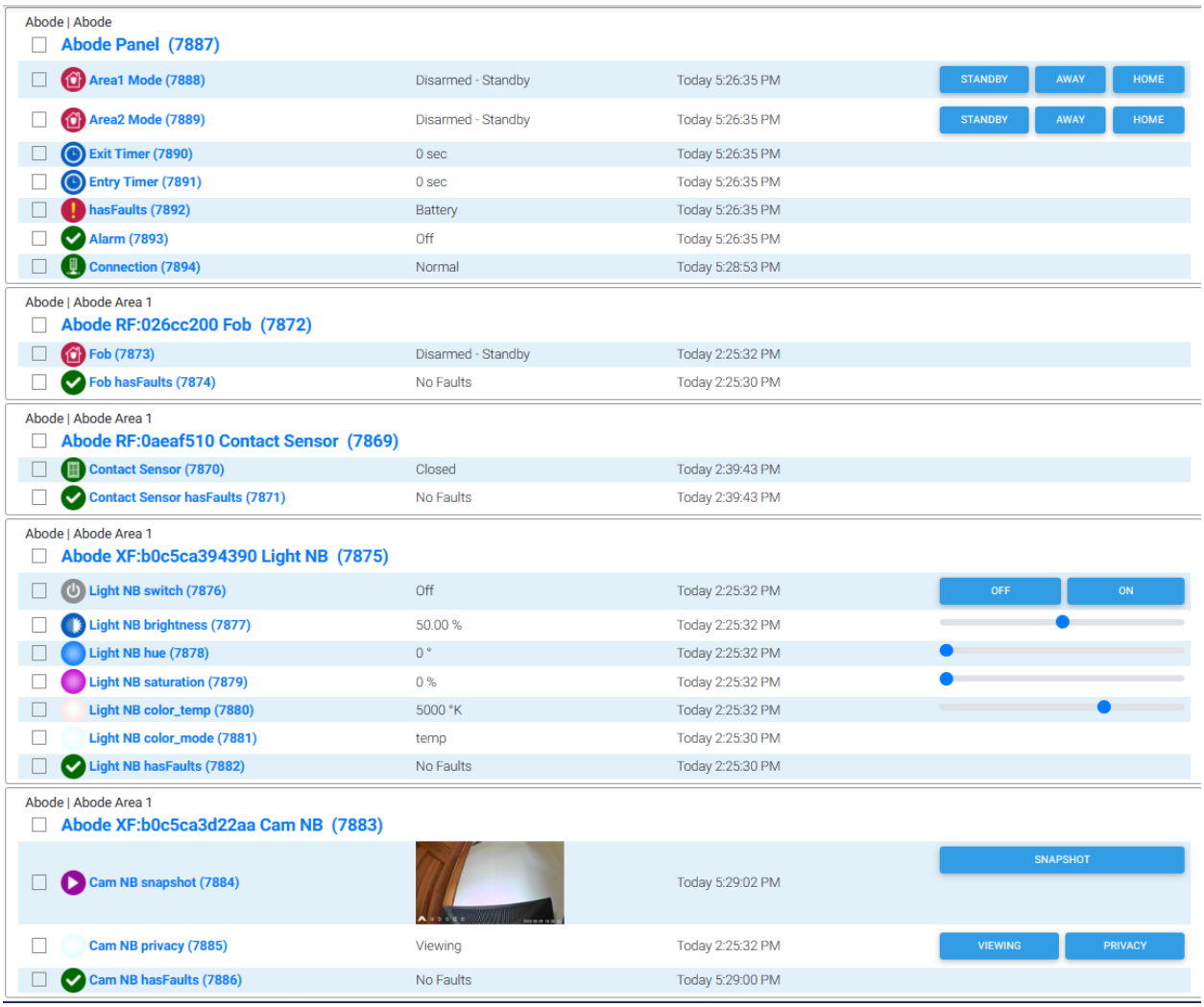


Figure 168 Abode HS Devices and Features

Camera support consists of two controls. One to turn the camera on or off for privacy. The other is to capture the current image being seen by the camera. Images will also be captured when the alarm panel snaps a picture when the alarm triggers. Each image will be saved in a HS subfolder \html\mcsMQTT\xxx.jpg where xxx is the ID of the camera that took the snapshot. A second copy if made in same folder with fixed filename of Abode.jpg. mcsMQTT will also create a thumbnail of the image and place it in the DeviceString of the camera's snapshot Feature. The DeviceValue of this

Feature will increment with each image download. Clicking on this image will bring up the full-size image. In addition, an archive of the image will be copied to the user-specified video folder with a subfolder created for each camera if that location is specified on the Cloud Page, Abode Tab setup.

Note that a snapshot will still take the image even when in privacy mode.

The color light device has multiple controls. All are functional except the switch and brightness controls. The Abode returns code 600 – Panel Error when trying to control these from the plugin.

12.15 Irrigation

12.15.1 Orbit B-Hyve Irrigation

Orbit provides consumer-grade smart irrigation controllers with their B-Hyve line. It operates using WiFi to provide setup and operation via smartphone. Once setup it will operate locally, but if dependent upon weather data when setup for smart or ET-based irrigation the control will become stale.

mcsMQTT integrates the run-time information from the controller and provides common user controls such as selecting among the programs that have been setup or running a station asychronously. Plugin setup is on the Cloud Page, Orbit Tab as shown in Figure 169. Setup uses the login to the user's Orbit account and provides a mechanism to disconnect and reconnect to the cloud server as well as asychronously request all data from the Orbit Server.

Two communication channels are used. One is HTTPS that is used at startup and polled every five minutes. Primary use is to get the equipment definitions that are used to setup the HS Devices and Features. The polling will update status, but the primary status update is with the WebSocket channel that reports real time events from the controller and is used to deliver asynchronous commands to it.

The implementation is based upon data gathered by others through a reverse engineeing process. The control flow for the plugin is based upon the Hubitat Groovy code GitHub - dcmeglio/hubitat-orbitbhyve: Provides integration with a Orbit™ Bhyve Timer and SmartThings. The control endpoints were primarily gathered from Python implementaiton that looks to also be used by Home Assistant GitHub - sebr/pybhyve: Python library for interacting with the Orbit BHyve API . In the end the integration was completed using a man-in-the-middle exploit to decode the traffic from the Orbit BHyve App and using this to reverse-engineer the functionality provided in the plugin.

The plugin gathers information from multiple URL endpoints that include Devices, Timelines, and Landscapes. A History endpoint is also available that is not used. Devices describe the Stations. Timelines relate to the Programs. Landscapes related to smart irrigation.

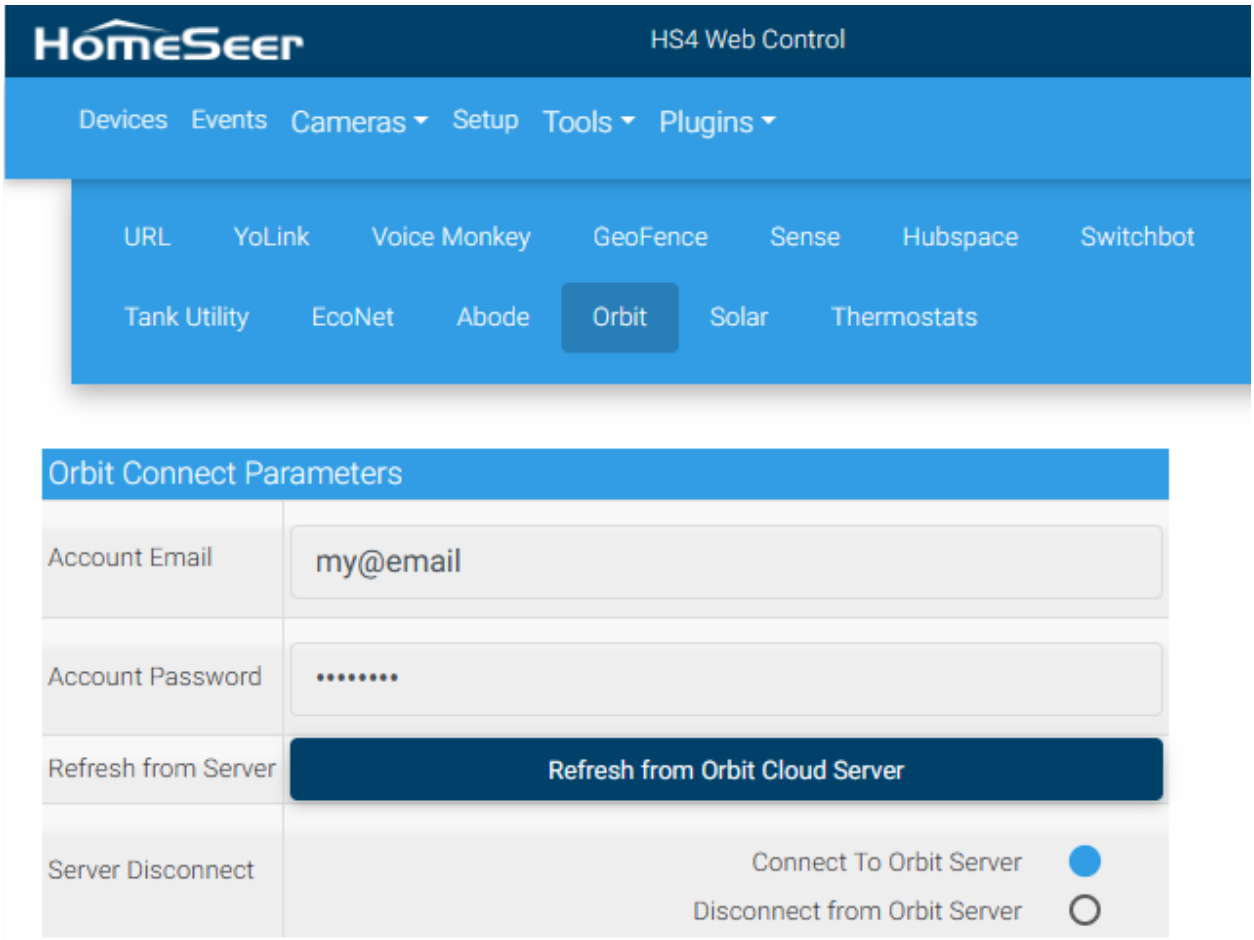


Figure 169 Orbit B-Hyve Account Setup

A HS Device is create for the Panel and one for each Station. From the Panel Device it is possible to select amoung the programs that were setup on the smartphone, to change the operation between Off and Auto and to pause or delay the program. Status is also provided for the next start time, the battery status, panel connection status and rain delay that may be active. Date-Time information is presented in the Status and DeviceValue is populated with the Unix local time.

Each station has a control to run the station for a specified amount of time and to set the soil moistue level when smart watering is active. Status information for the Station is watering status, last start and end times, rain sensor status and the timed vs. smart control. Orbit only provides the current water level in response to a request to change the level. It does provide any other polling or event-based reporting to reflect the current moisture level so the HS status will not change from the level at which it was manually set.

Status updates from Orbit server are obtained on an event basis via a WebSocket connection. This connection is actively monitored and attempts wil be made every 25 seconds if the connection is lost. Polling will occur every hour as a precaution and as a means to monitor the connection in a closed-loop manner.

A Monitor HS Feature is provided that shows the number of minutes since the last data from Orbit Server has been received. During quiescent times this value is expected to reach 60 minutes. A control on this Feature is provided to restart the plugin's support of the Orbit integration.

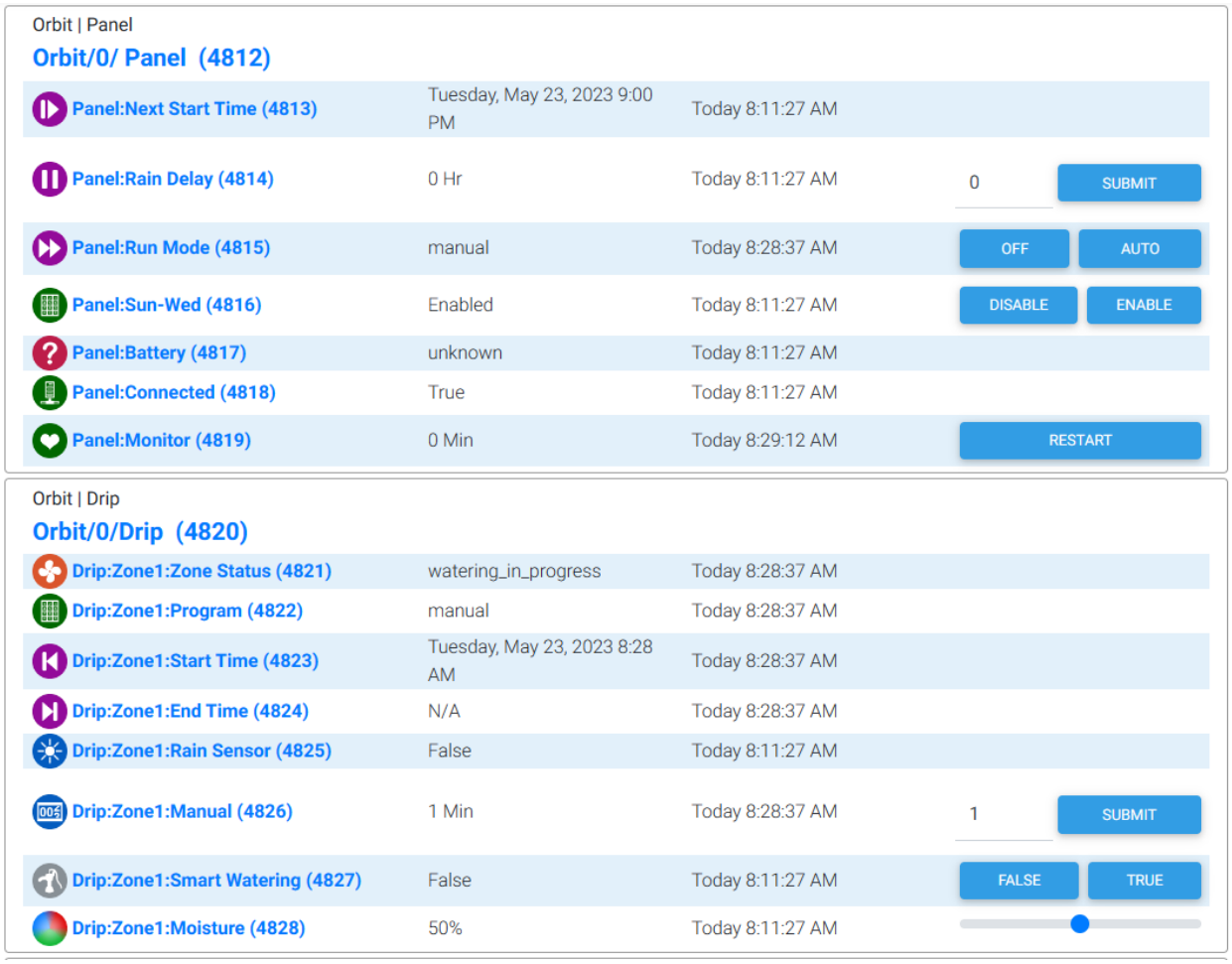


Figure 170 Orbit B-Hyve HS Devices and Features

12.15.2 Hunter Hydrawise Irrigation

The integration with the Hunter Hydrawise unit is performed with the second generation (V2) methodology of oAuth2 for authentication and GraphQL for query of the Hunter Hydrawise server. This API provides access to the controller, sensors, and zone relays. Much information is available from the Hunter Hydrawise server, of which a subset is selected for automatic creation of HS Devices and Features. A Device is created for each controller and Features are created for status information for the controller, sensors and zones.

The controller Summary Feature can be used to control all the zones per the schedule that has been setup. No provisions exist in mcsMQTT to manage schedules. This is done with the Hydrawise App/Account.

Controller overall control provisions apply to the current schedule with ability to Start, Stop, Suspend, and Resume the schedule. See Figure 171. When suspending the schedule, the HS Feature Suspend.Hours is used to specify the number of hours that the suspension will be enforced by the controller. By default, this is one hour, but can be change by HS Event or HS Devices Page user entry. The entry can contain fraction hours such as 1.25 to provice a pause of 75 minutes. Setting Suspend.Hours has no immediate effect. Suspend operation is performed by the “Suspend All” or “Suspend Buttons”.

The status is reported in Device Value and Device String. The Device String is what will be visible on the HS Devices Page as the status. The controller Device Value will be zero when no zone is running. It will be one when any zone is running. The Devices Page status icon will also provide a visual indication of the stopped vs. running status.

Similar controls and status exist for each zone. A run time text box is also included for the zones. The number entered via HS Device Page or Event action is the number of minutes to manually run the zone, independent of what is programmed in the schedule. Again, fractional values can be used such as 10.5 to run the zone for 10 minutes and 30 seconds. The submit texbox will show the time entered and will be updated each minute to show the time remaining. It wil be 0 when not running.

The status reporting is from the Hunter Hydrawise server and stored in the HS Device String. With standard HS Events the Device String cannot be used as Event triggers, but the EasyTrigger plugin has such capability if needed.

All Features will be updated based upon the current information on the Hunter Hydrawise server every 60 seconds when no zone is active and 20 seconds when at least one is active. The water flow information is updated based upon the raw water flow sensor to show real-time water use. Upon completion of a zone the water use is synchronized with the Hydrawise server using the completion event data from the server.

Zone status is shown in the Zone Feature’s Device String and will be visible on the HS Devices Page. This will be the current status summary from the Hunter Server and will include the “suspend until” information if is is available from the server.

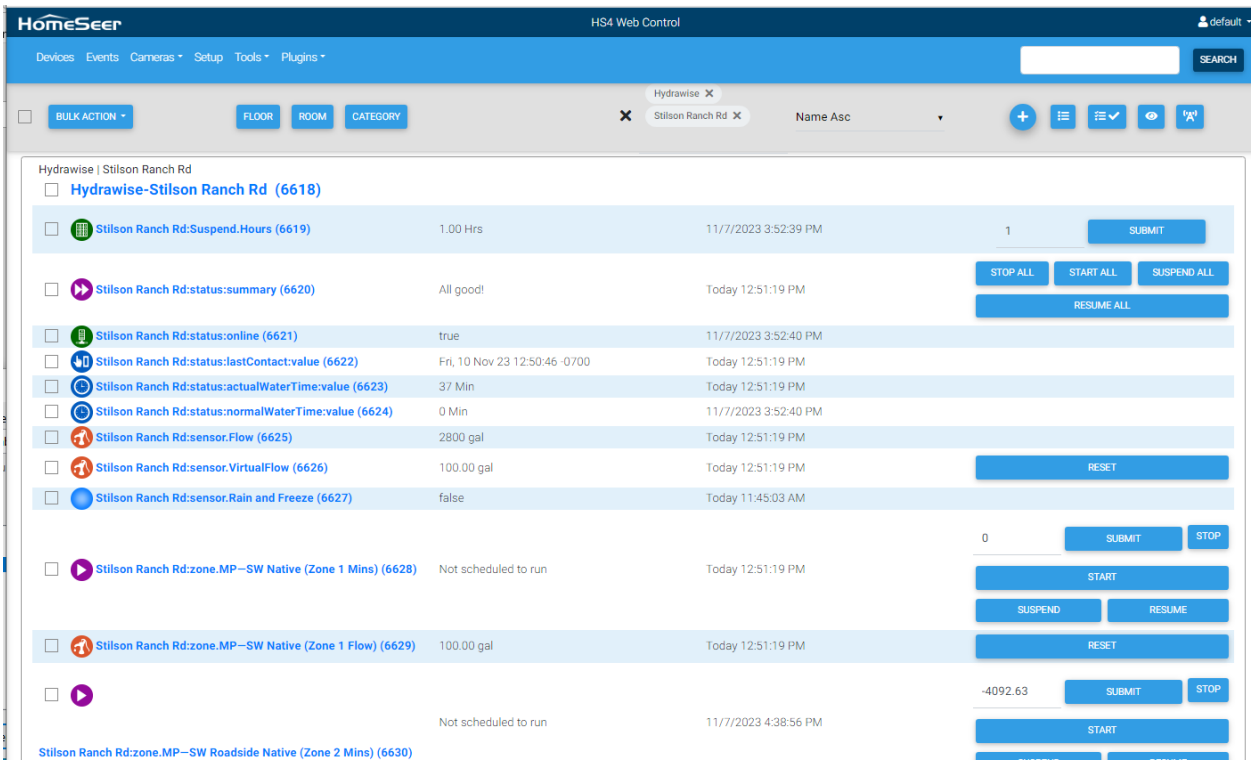


Figure 171 Hunter Hydrawise Default HS Devices and Features

User setup is on the Cloud Page, Irrigation Tab, Hunter Hydrawise Section as shown in Figure 172. Required is the username (email address) and password that was setup when registering the Hydrawise device. One or more Hydrawise accounts are supported.

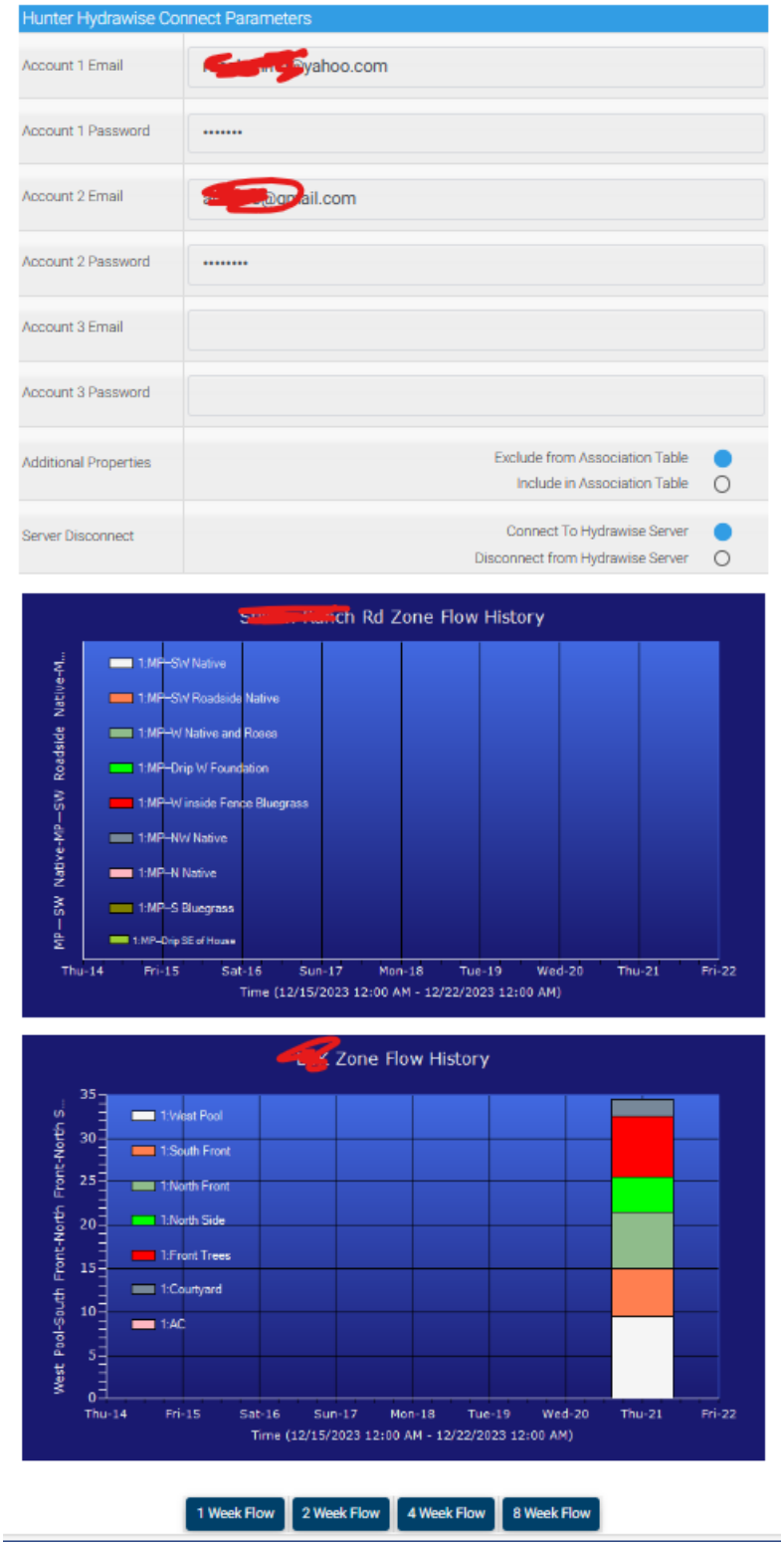


Figure 172 Hunter Hydrawise Setup

There is considerable information available from the Hunter Hydrawise server. By default, it is not exposed. If a user wants to associate additional information into HS Device Features, then it can be exposesd with the setup selection. Creation of additional HS Device Features is done from the MQTT Page, Association Tab with the suggested filter of “Hydrawise” as the T1 filter. Figure 172 is and example of the additional information in the table. Note that for this one controller there are 633 items in the table.

Hydrawise provides a flow sensor with the running count of the volume of water that has been used. The plugin computes each zone flow from this sensor when a zone is active. A reset button is provided that can also be controlled by event to reset the virtual flow sensor value back to zero. It is automatically set to zero at the start of a run and at midnight.

The historical flow data will be stored in the SQLite short term database. The data can be viewed graphically with a stacked column chart in one for four time periods. The period is selected by a button push and the chart will appear above the buttons. These charts can also be requested via a http request to the HS IP such as http://192.168.0.100/mcsMQTT/Popup.html?Payload=2681&Days=2 for HS3 and http://192.168.0.100/mcsMQTT/Popup.html?Payload=2681?Days=2 forHS4 where 2681 is the Ref of the Hydrawise controller Virtual Flow Feature and Days=2 is the number of days of stacked columns. If Days parameter is omitted then it will default to the prior chart request number of days.

There is also user selection to disconnect from the Hunter Hydrawise server. This applies to all accounts. When disconnected there will be no communication for control or status updates.

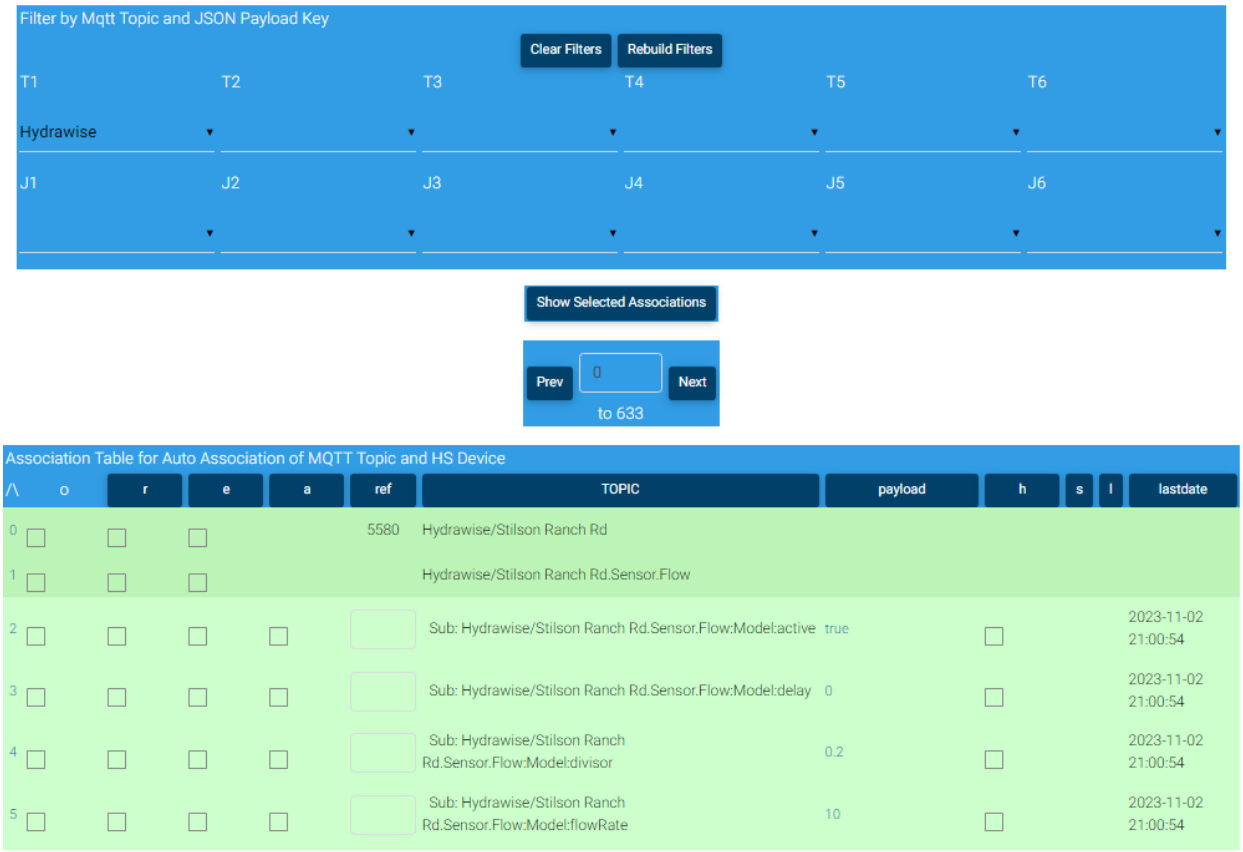


Figure 173 Hydrawise Additional Information

12.16 Solar Panel Integration

Solar integration collects data from locally installed panels using Solar_Assistant and pulls data for forecast energy generation from Solcast for the next 24 hours with eight grouping for the day. The forecast is nominally updated each hour. Local panel generation data is pushed over a MQTT channel.

This integration is in cooperation with HS user Daveyboy who initiated the discussion and provided most of the ideas and data for this integration.

Setup is from the Cloud Page, Solar Tab. The forecast from Solcast requires the user's resource Id to be entered. Solar Assistant MQTT server IP is needed for the local data. A radio control is also provided to connect/disconnect from both servers.

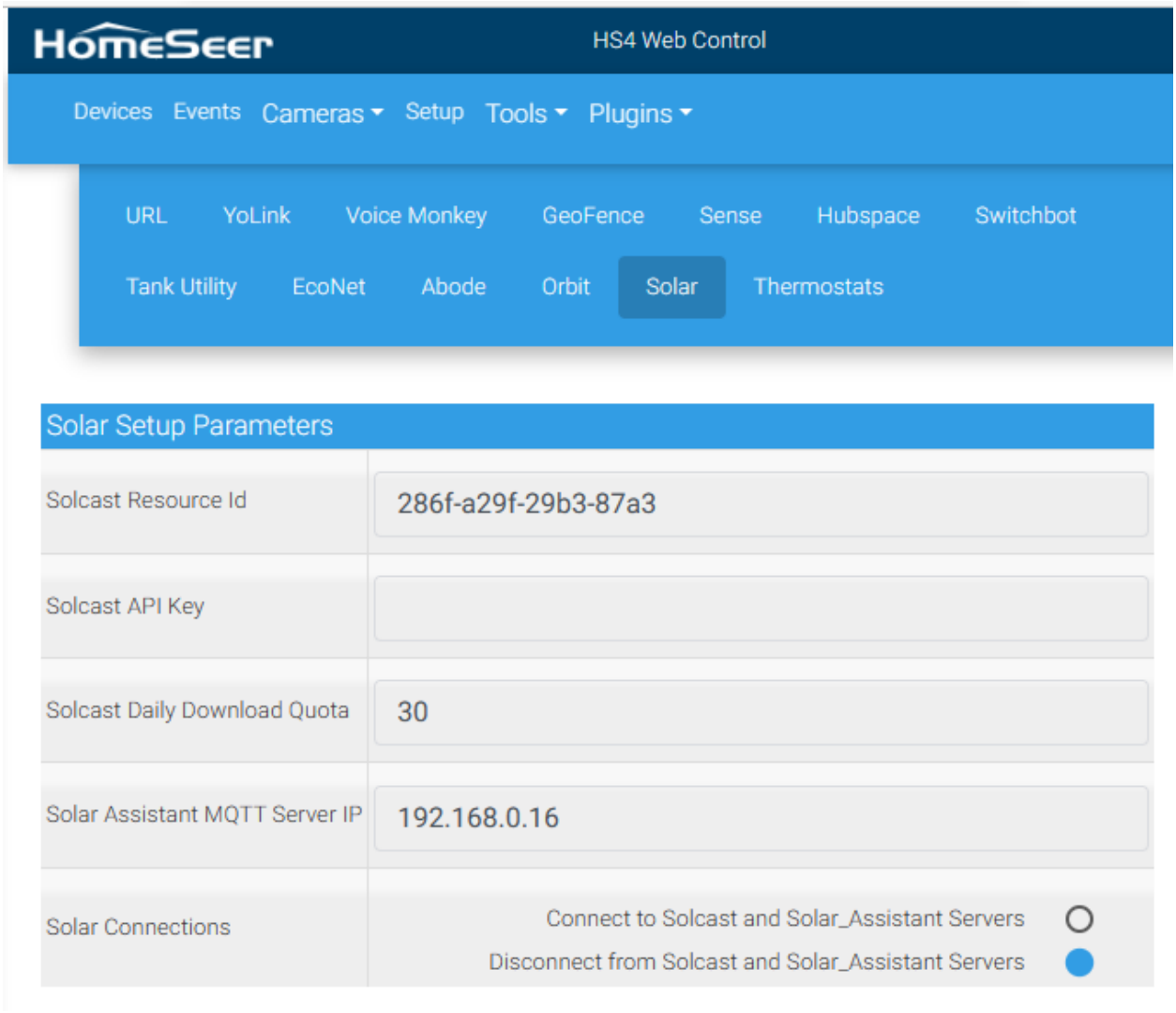


Figure 174 Solar Panel Integration Setup

12.16.1 Solcast

HS Device and Features are created at startup when the Solcast resource id has been entered. Updates occur each hour. The data from Solcast is assumed to be in UTC time so it converted to local time for grouping ins HS Features. This grouping is defined as:

Early morning	6 am through 9 am
Late morning	9 am through 12 pm
Early afternoon	12 pm through 3 pm
Late afternoon	3 pm through 6 pm
Early evening	6 pm through 9 pm
Late evening	9 pm through 0 am
Early night	0 am through 3 am
Late night	3 am through 6 am

The forecast data is delivered in 30-minute intervals. Each 30-minute interval within the defined timer periods is summed to produce the value recorded in the HS Feature.

Solcast is a subscription service with different rate plans available. The plan will determine the number of daily downloads that will be available. The quota is a user setup parameter as shown in Figure 174.

Since data is being quantized into eight time-slots per day there is not a need for frequent downloads. Downloads more frequent than every 30 minutes will result in little new information since that is the interval quantized by Solcast. Download every hour (24/day) seems to be a good balance.

Each forecast provides 48 hours of data. mcsMQTT uses only data for the next 24 hours. The three-hour bucket it selects is based upon the local time of day for the forecast. This means that if the download occurs at 10 AM, the data in the 6 AM to 9 AM bucket will be tomorrow's forecast while data in the 12 PM to 3 PM bucket will be today's forecast.

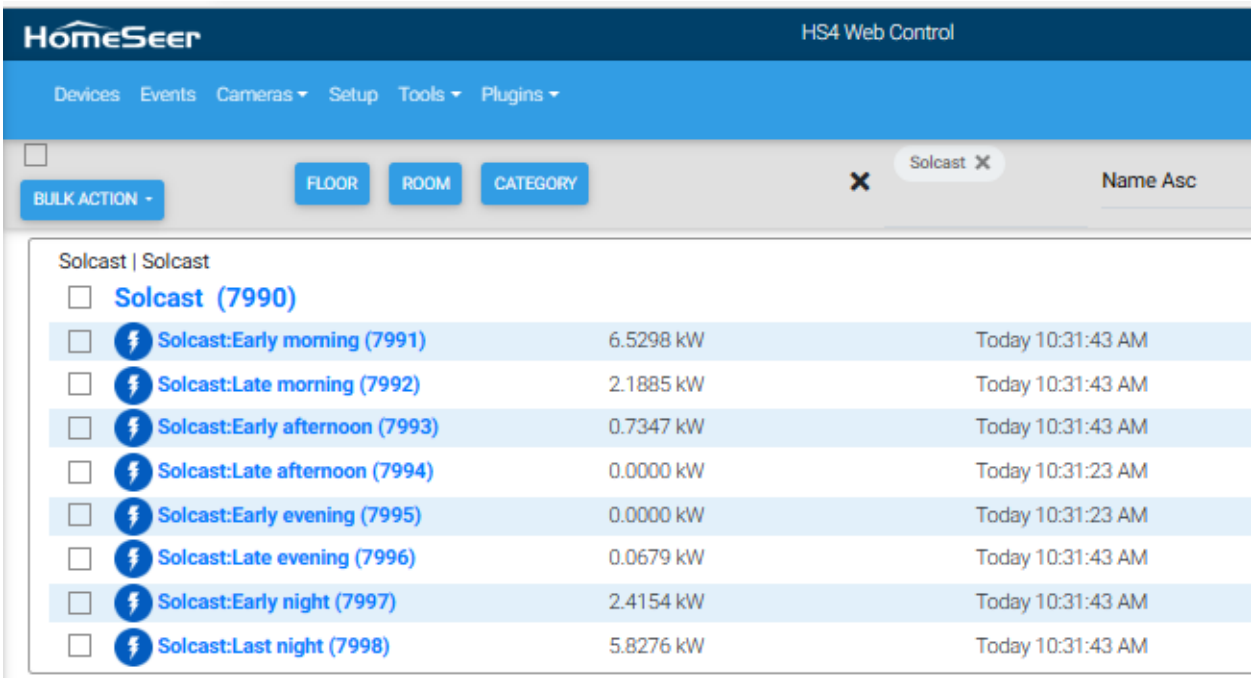


Figure 175 Solcast HS Device and Features

12.16.2 Solar Assistant

SolarAssistant is software used to locally monitor and control your solar PV install. It is designed to run on a Raspberry Pi that is plugged into the solar inverter and optionally a battery BMS. The application can be accessed from a web browser or the Android/iPhone app via local network or the internet. It is available at Online Shop | SolarAssistant (solar-assistant.io) as a RPI image for around $55. Also needed are a Pi, a SD card and a special cable to link from the Pi to your Inverter.

The table below provides the status of the SolarAssistant support of various inverters at the time of this writing (May, 2023).

Supported Inverters	Not-yet Supported Inverters
Axpert	ABB
Growatt	APEX/MLT
EG4	Atess
Deye	Fronius
Kodak	Epever (Expected late 2023)
SunSynk	Renology
MPP Solar	Schneider
Sol-Ark	Sofar
Mecer	Solar edge
RCT	Solax
Must Power	Solis (Expected mid 2023)
SRNE	Victron
InfiniSolar	

Easun Power	
Megarevo	
Luxpower	

The killer app with SolarAssistant is that it's all local, manufacturer independent and you get some really cool charts too - no need to rely on a cloud connection, the manufacturer's website, data being bounced through China etc.

mcsMQTT provides the mechanism to leverage this capability with integration of the data into HS. You can view and make actions within Homeseer on *live* data direct from your PV install. Battery charge levels, PV levels, house load, grid load. You can even change configuration parameters on the Inverter to preemptively take advantage of tomorrow's weather conditions. Turn the hot tub on/off if there is excess PV, turn the water heater on/off if the batteries are getting low, kill the power to the tumble dryer etc - there are so many use cases that this integration enables.

12.16.2.1 SolarAssistant Installation

Full SolarAssistant configuration is available at https://solar-assistant.io/help/gett...prepare-device. After you've got your SolarAssistant up and running you'll need the local IP address from their Configuration Page:

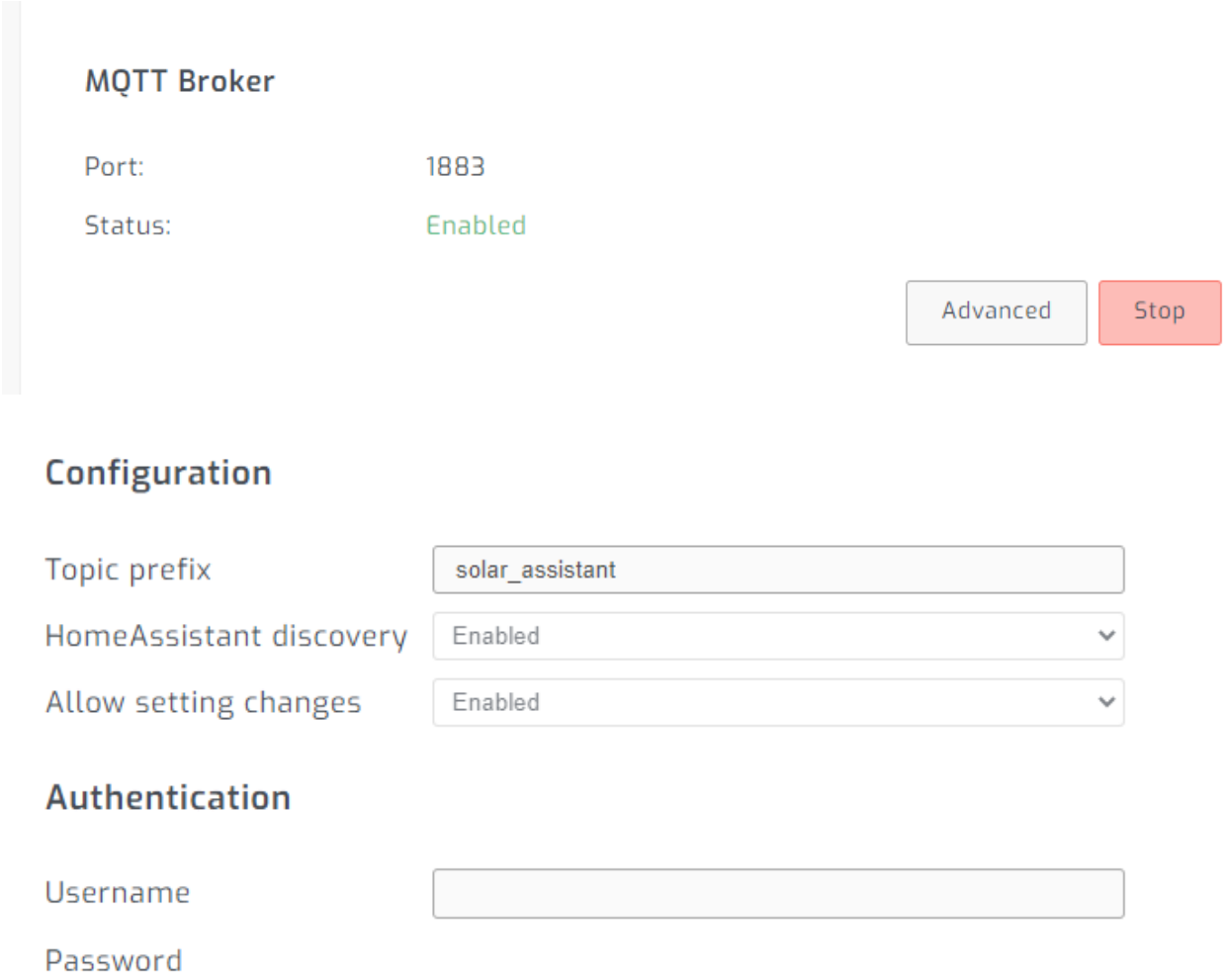
Network status

Internet	84.66.213.15	Up
eth0	10.0.0.5	Down
wlan0	192.168.1.106	Up

And then enter this into the Solar Tab, under the mcsMQTT, Cloud page in HomeSeer:

Solar Assistant MQTT Server IP	192.168.1.106
Solar Connections	Connect to Solcast and Solar_Assistant Servers Disconnect from Solcast and Solar_Assistant Servers

Back in SolarAssistant make these changes from the Configuration Screen:



Note: No authentication is required for this local network connection.

With the SolarAssistant setup to Enable HomeAssistant Discovery, wIthin a few minutes there will be new devices automatically created in Homeseer. Is HomeAssistant Discovery is not enabled then the data from SolarAssistant will only appear in the Association Table of the MQTT Page of mcsMQTT. From this table, the “a” column checkbox is used to mark those pieces of data for which HS Device and Features will be created.

The created devices will be placed in a HS Room based upon the MQTT being used by SolarAssistant. If HomeAssistant Discovery is Enabled then the following is the expected view from HS.

HS totals room:

387 W	Today 16:44:02	solar_assistant/total/battery_power/state
36 %	Today 16:34:11	solar_assistant/total/battery_state_of_charge/state
28 °C	Today 16:09:06	solar_assistant/total/battery_temperature/state

HS inverter_1 room:

There will also be a bunch of new devices created in the HS inverter_1 room (around 80) - these are unique to each install and to each inverter. You will probably discover that this information mostly will not be useful to you but a few certainly will be (e.g. battery temperature, PV load etc), however, we cannot give guidance here which ones you may wish to remove. Removing Features is done from the MQTT Page, Association Table by removing the “a” column checkbox from the table’s row. This tells mcsMQTT to not associate the SolarAssistant specific piece of data to a HS Feature. Note, if the Feature is removed using the HS Devices Page, then a new Feature will be created if HomeAssistant Discovery has been enabled.

Communicating to the Inverter is possible from HS if the Association Table or Edit Tab of the mcsMQTT MQTT Page is provided a Pub(lish) Topic. This will be the same as the Sub(scribe) Topic, but end with /set rather than /status.

Our Disclaimer: You must be extremely careful in sending data to control your inverter from HS as this could render your Inverter inoperative. Only change what you would normally change via the main Inverter console. You will have received warnings and disclaimers from SolarAssistant when you signed up to their service so you will be aware of the risks.

13 Pool

13.1 Hayward Omnilogic

Hayward provides a Omnilogic control for pool heating, chlorination, lighting and other features.

The integration with Homeseer is based upon the Github Python API implementation at https://github.com/djtimca/omnilogic-api. A Python, **version 3**, install on the same computer as mcsMQTT is needed to run this code. The Pyton install varies based upon the OS.

The Python MQTT and HTTP library is also needed. They are most easily installed from the command line / terminal window using PIP with the command

```
pip install paho-mqtt

pip install aiohttp
```

When both Python2 and Python3 are in the environment then

```
python3 -m pip install paho-mqtt

python3 -m pip install aiohttp
```

If Python3 and PIP are not yet installed on the Homeseer computer they first need to be installed. Use Google for guidance on installing them on the OS that is hosting Homeseer. After installation, the folder where python.exe needs to be identified as it varies. There will also be a \Scripts subfolder that normally is a subfolder of where python.exe is located for Windows installs.

The integration of this library with Homeseer is done with the Python script OmnilogicRequest.py that is available in the mcsMQTT download package and originally installed at subfolder \bin\mcsMQTT. It will not be run from this location, but is moved to the \Scripts (or alternate) subfolder of the python install.

During installation of Python, the PATH environment variable is normally updated with path to python and the scripts subfolder. It may be necessary to have these definitions in PATH.

The mcsMQTT setup for Omnilogic integration is on the Cloud Page, Pool Tab. See Figure 176.

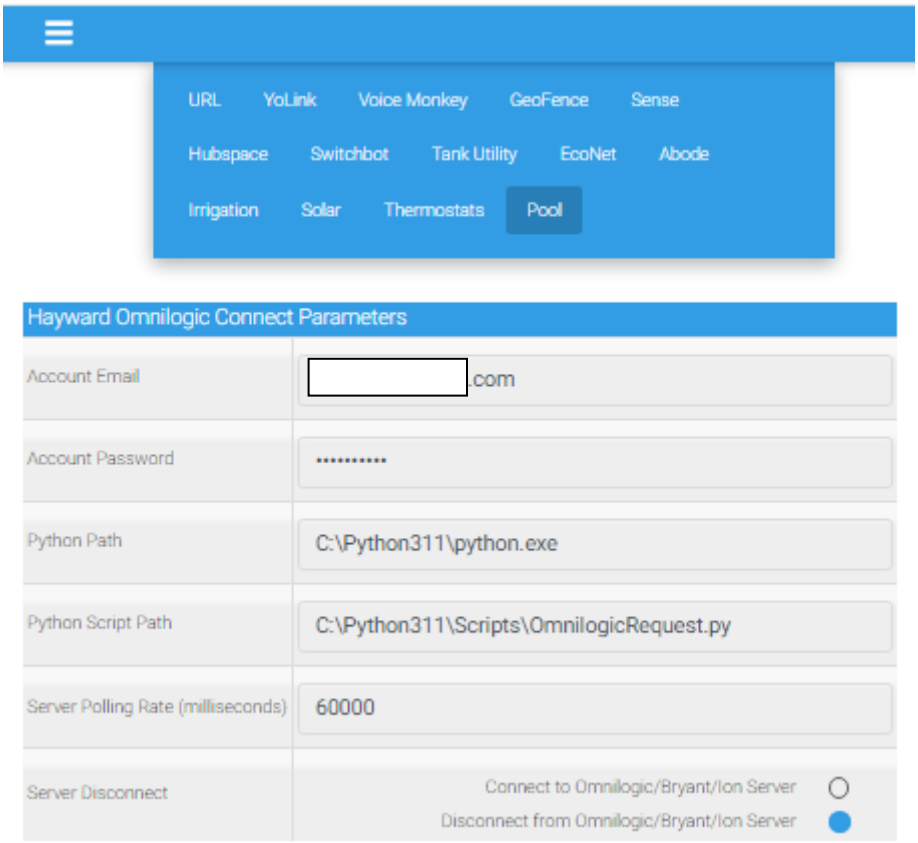


Figure 176 Hayward Omnilogic Pool Integration Setup

Note the full path to python.exe and the Scripts folder in the setup. Also needed are the email and password to the account that was setup with Hayward Omnilogic.

For Linux users, see the setup for Carrier Integrationin Section 12.12.3 for the approach to deal with the environment variable PYTHONPATH that is not defined when running Python from the shell account.

Data is polled for status updates to handle the local control being synced with HS. Provision is also provided to disconnect the connection with the cloud server. This disconnection will also result in the Python script OmnilogicRequest.py being terminated. OmnilogicRequest.py is managed by mcsMQTT and run when the setup is complete and not disconnected.

mcsMQTT will automatically create an HS Device for each pool and set of features such as in shown in

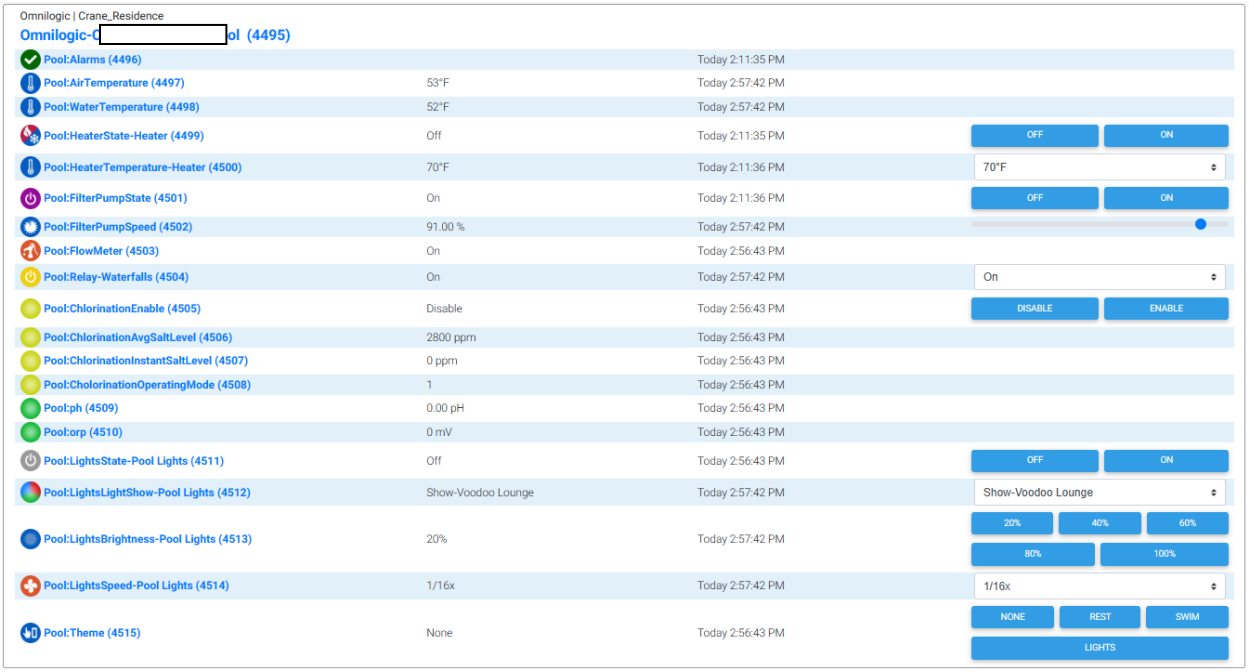


Figure 177 Hayward Pool Omniologic HS Device and Features

14 Interactive

The Interactive page provides a means to interrogate varibles, execute expressions and send messages similar to the MQTT Send Message event action.

Any expression that is supported by mcsMQTT can be used. This includes all the HS replacement variables using syntax $$X: or $$X:(Y):. The first are replacements without parameters such as TIME. e.g. $$TIME:. The second is for those that have a parameter such as DTR. e.g. $$DVR: (123): [no space before ( ]. If the result is a string rather than a number then encase in quotes. This will be the case for TIME so the first example should be "$$TIME:". Another example is "$$DTR: (123):".

Expression and functions can be used such as SIN($$DVR: (123): + MOD(5,2) to use it as a HS-oriented calculator.

When sending messages and an expression is desired the use the << and >> symbols to encase the expression. This is shown in Figure 178.

Two forms of interactively working with the HS object are provided. One is a command line format that can be an expression. It will be much like the immediate script command event action provided by HS. The other is to run an existing script command located in the \scripts subfolder of HS and interactively include input parameters to the script. The full syntax in the textbox is script filename, functionname, parameter. If the script has functionname of “Main” and no parameters being used then simply enter the name of the script file. Examples are:

Test.vb,Main,1	pass a number
Test.vb,ArrayMain,{1,2,3}	pass array of integers as the parameter
Test.vb,StringMain,”1,2,3”	pass a string
Test.vb	no parameters passed and function name is Main

```
Function Main(Parm As Object) as Object
    hs.Writelog("Script","FunctionTest")
    If isNumeric(Parm) Then
        Return "Number " & (11 + Parm).ToString
    Elseif isArray(Parm) Then
        Return "Array " & Parm.Length.ToString
    Else
        Return "String " & Parm
    End if
End Function
```

Figure 178 shows use of replacement variable DVR to get access to the DeviceValue as well as using the hs.DeviceValue method for the same purpose. Both use the same expression of summing the two devices values. Result & Feedback boxes show the result of both interactive methods. If a script does not return a result or an error occurs running the script, the feedback will be “null”.

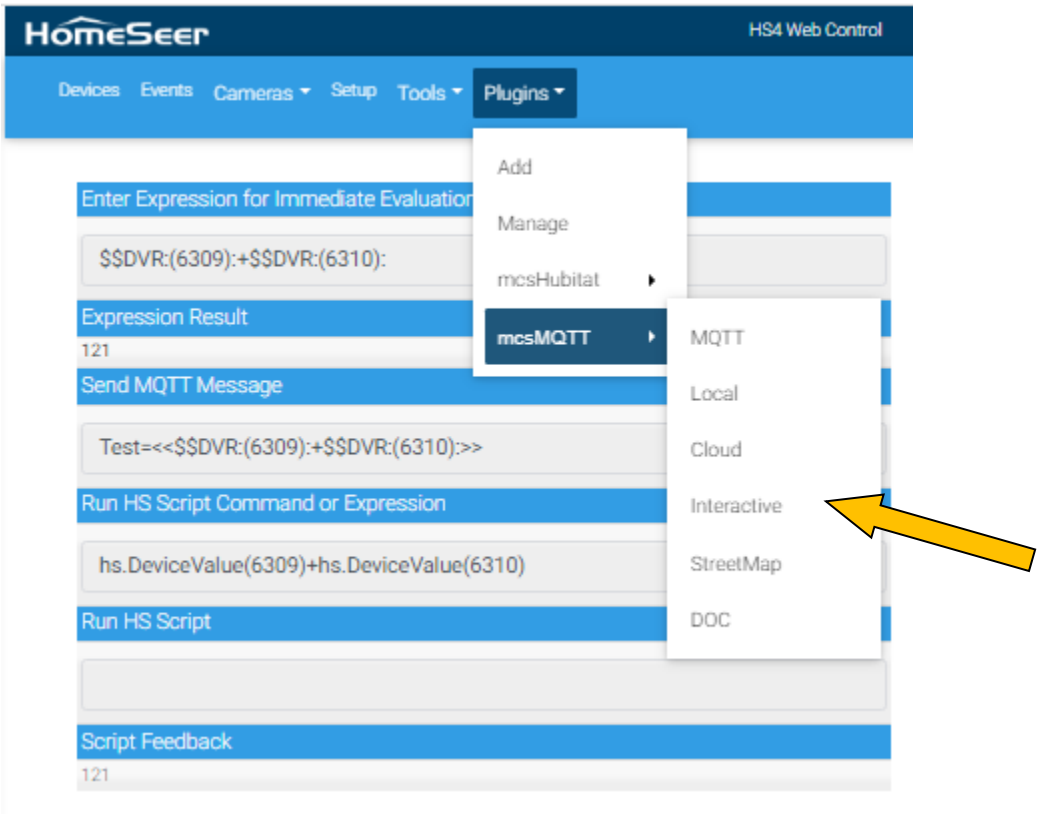


Figure 178 Interactive Page

15 StreetMap (HS4 Only)

Two capabilities are provided by mcsMQTT that utilize the data feed from OwnTracks or NextTracks that provides location information for smartphones. The first is to augment the data feed with Distance and Here-Away indication. The HS location is always available so the Here-Away normally can be used for Home vs. Away indication. Additional locations that identify the center of another geofence can be added on the Cloud Page, Geofence tab.

The second feature is available on HS4 only. It provides a browser page with a street map view that identifies the smartphone locations on the map.

15.1 OwnTracks Setup

The OwnTracks page provides a streetmap view of Android phone OwnTracks tracking App. OwntTracks is available from the Android Play Store. Once installed it is setup in preferences to point to MQTT WAN address of the MQTT broker being used by mcsMQTT. mcsMQTT supports multiple brokers so it is possible to setup a WAN broker and and LAN broker.

The OwnTrack App in your smartphone will send position updates to a MQTT broker. You setup the IP/URL of the MQTT broker, a username and a password in the smartphone App. When using mcsMQTT the easiest MQTT broker setup is the internal broker that shares the same IP/URL as Homeseer. Anything can be entered for the username and password. There is no setup to be done on the mcsMQTT side other than was exists as the defaults. If you want you can enter the same username and password on the MQTT page, Broker Tab, Broker Username and Broker Password text boxes to improve security, but it is not required to achieve functionality.

The smartphone needs a way to get through your firewall on port 1883 so you need to make this happen with your router. The IP/URL that you use for the MQTT Broker needs to be visible on the WAN. This means you cannot enter something like 192.168.1.100, but use a DNS name such as myHome.com. If you do not have your own domain then you need to use a service such as no-ip.com or if you have a web camera, they often provide the DNS service. ASUS routers also provide this service.

The smartphone will push position changes to the MQTT broker, which in this example is mcsMQTT. mcsMQTT will show what the smartphone has provided on the mcsMQTT MQTT Page, Association tab. Anything on that page can be mapped into HS devices with the "A"ssociate checkbox. Lat and Lon are typically what is of interest.

In summary
1. Setup a DNS service and firewall so your Homeseer computer can be accessed from the WAN on port 1883. As an alternate use a public WAN MQTT Broker. I think HiveMQ is one, but I have no particular experience with this one or other MQTT Broker in the cloud.
2. Install mcsMQTT plugin for HS4
3. Install OwnTracks on smarthone and configure it to use the DNS name of Homeseer computer and any username and password
4. Repeat step 3 for your second smartphone
5. Navigate from HS Menu to mcsMQTT MQTT Page, Association tab and observe table Lat and Lon rows and click "A" column checkbox on each to create HS device and features.

6. If you want to observe a street view map of the location of your smartphones then navigate to mcsMQTT plugin, StreetMap page.

OwnTracks has multiple modes of reporting so that battery drain is minimized. The highest resolution is for movement. The lowest resolution is for major changes. When the movement threshold has been reached (based upon mode being used) a MQTT message is delivered with new Lat and Lon coordinates. These will be visible on the Association Tab of the MQTT page such as shown in Figure 179.

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
0	☐	☐	☐		7693	owntracks/mcsSolutions/M1					
1	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:_type	location	☐			2021-05-27 11:16:49
2	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:acc	33	☐			2021-05-27 11:16:49
3	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:alt	117	☐			2021-05-27 11:16:49
4	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:batt	80	☐			2021-05-27 11:16:49
5	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:bs	1	☐			2021-05-27 11:16:49
6	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:conn	w	☐			2021-05-27 11:16:49
7	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:created_at	1622008166	☐			2021-05-27 11:16:49
8	☐	☐	☐	✓	7695	Dev: owntracks|mcsSolutions|M1:lat Sub: owntracks/mcsSolutions/M1:lat Pub: the following Topic on Device command	47.5266756	☐	☐	☐	2021-05-27 11:16:49
9	☐	☐	☐	✓	7694	Dev: owntracks|mcsSolutions|M1:lon Sub: owntracks/mcsSolutions/M1:lon Pub: the following Topic on Device command	-121.7514224	☐	☐	☐	2021-05-27 11:16:49
10	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:t	p	☐			2021-05-27 11:16:49
11	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:tid	M1	☐			2021-05-27 11:16:49
12	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:tst	1622001353	☐			2021-05-27 11:16:49
13	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:vac	3	☐			2021-05-27 11:16:49
14	☐	☐	☐	☐		Sub: owntracks/mcsSolutions/M1:vel	0	☐			2021-05-27 11:16:49

Figure 179 Owntracks MQTT Report for Android Phone

15.2 Street Map Browser Page (HS4 Only)

mcsMQTT knows that a street map visibility is desired when the ":lat" JSON item is associated wth a HS device. Normally one also Associates the ":lon" item so the postion coordinates are available in HS Devices.

To view the streetmap, use the StreetMap page from the mcsMQTT plugin menu. When this page is viewed and a new MQTT message is received with a change in Lat or Lon then the page will be refreshed to show the new position.

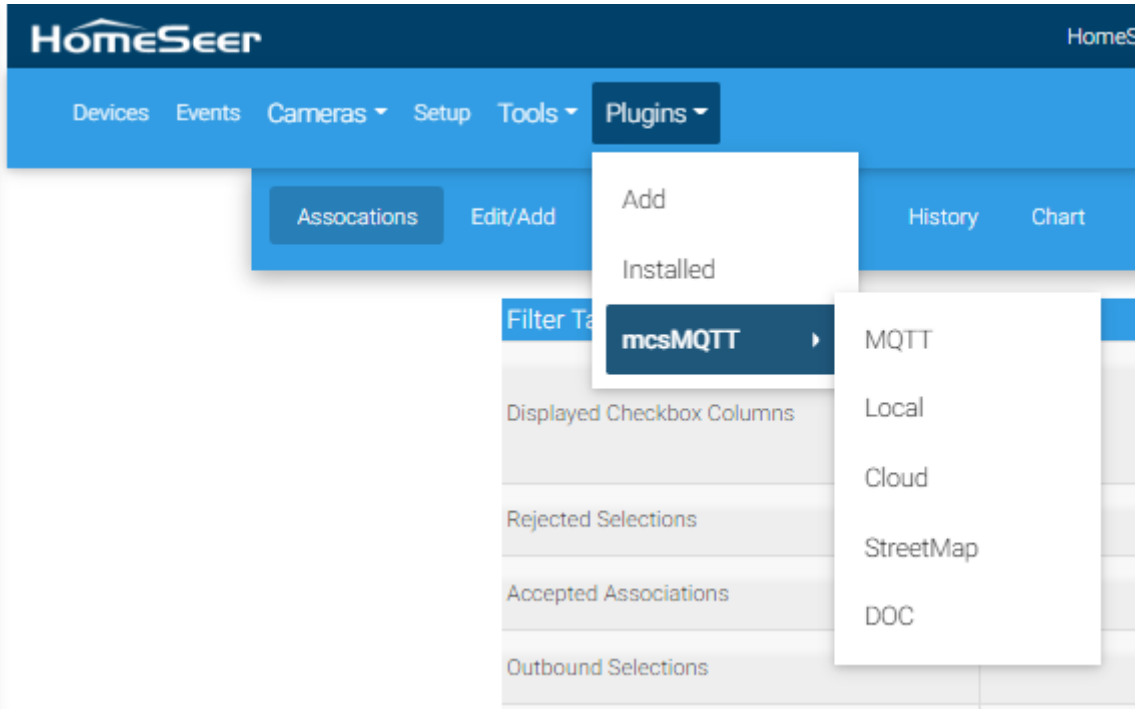


Figure 180 mcsMQTT Plugin Browser Page Options

Figure 181 shows an example of display for two Android phones. The "@" symbol starts at the reported position. The Id setup in OwnTracks App (and is used as the last element of the MQTT Topic) is also drawn to identify the specific phone.

The Map is scaled so that all phones are visible. If they are close then the map will span a few hundred feet. It is rescaled as the separation between phones changes.

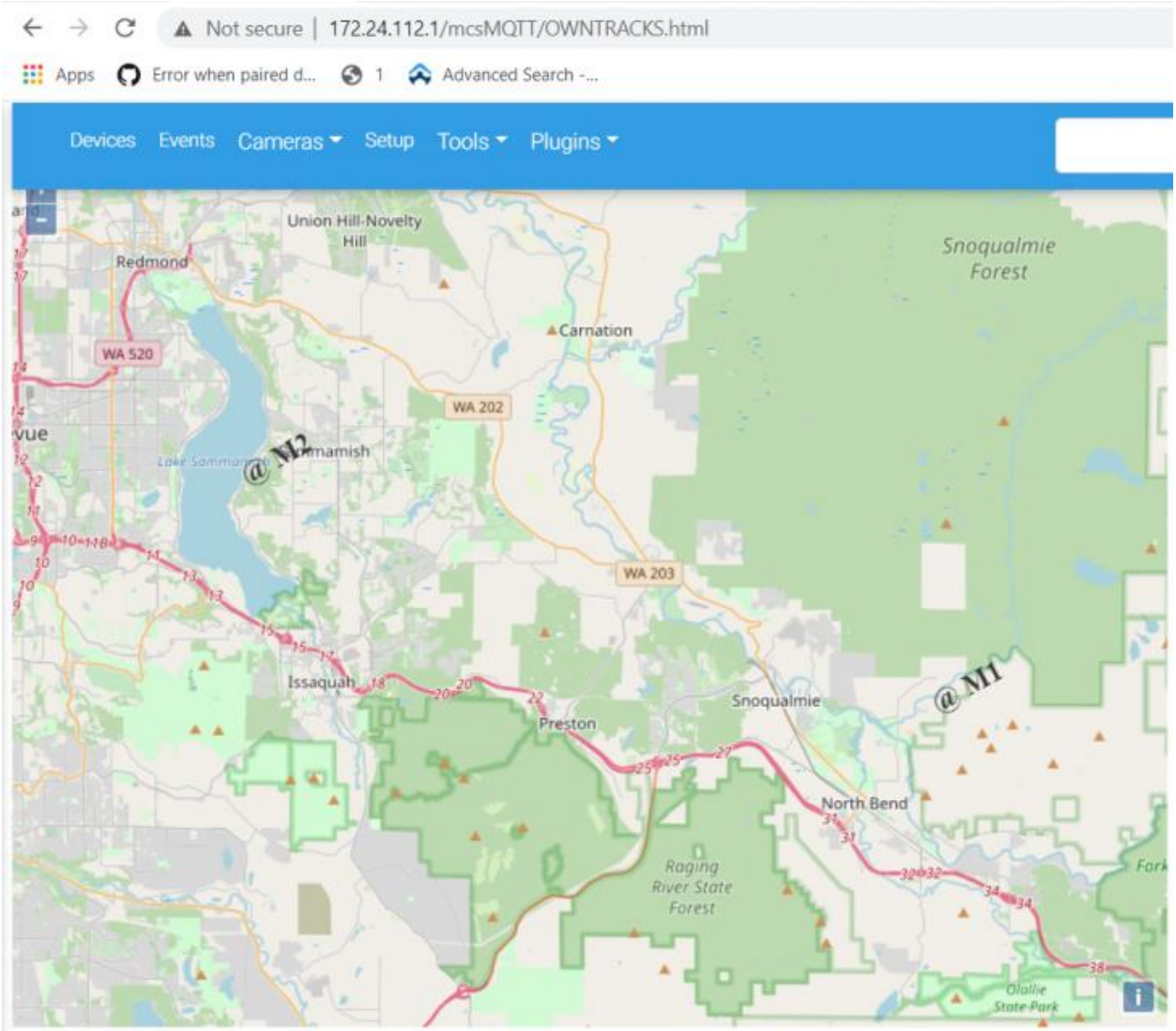


Figure 181 OwnTracks Display Page

15.3 Geofence Here-Away Tracking

mcsMQTT will calculate the distance from a smartphone to the center of a geofence locations and then use these to assess if the smartphone is inside or outside the geofence boundry. The information will be included in the owntracks topic such as shown in Figure 182. In this example there are two geofence locations, named HS and Home, setup on the Cloud Page, Geofence tab. Each will have a JSON key of Distance and HereAway.

Filter Association Table by Mqtt Topic and JSON Payload Key Clear Filters Rebuild Filters

T1	owntracks	T2		T3	M2	T4		T5		T6	
J1		J2		J3		J4		J5		J6	

Show Selected Associations

Prev | 0 | of 8 | Next

Association Table for Auto Association of MQTT Topic and HS Device

Λ	o	r	e	a	ref	topic	payload	h	d	i	lastdate
0					174	owntracks/mcsSolutions/M2					
1						Sub: owntracks/mcsSolutions/M2:Home:Distance	159339				2021-08-23 15:18:11
2				✓	175	Dev: owntracks|mcsSolutions|M2:Home:HereAway Sub: owntracks/mcsSolutions/M2:Home:HereAway Pub: the following Topic on Device command	Away				2021-08-23 15:18:11
3						Sub: owntracks/mcsSolutions/M2:HS:Distance	246				2021-08-23 15:18:11
4				✓	176	Dev: owntracks|mcsSolutions|M2:HS:HereAway Sub: owntracks/mcsSolutions/M2:HS:HereAway Pub: the following Topic on Device command	Here				2021-08-23 15:18:11
5						Sub: owntracks/mcsSolutions/M2:lat	47.49323				2021-08-23 15:18:11
6						Sub: owntracks/mcsSolutions/M2:lon	-121.782				2021-08-23 15:18:11
7						Sub: owntracks/mcsSolutions/M2:vel	1				2021-08-23 15:18:11

Figure 182 Geofence Topics

Any or all of these can be associated with HS Devices. In this example the two HereAway devices are created with HS status reflected as shown in Figure 183.

owntracks | mcsSolutions

owntracks-mcsSolutions-M2 (174)		Today 3:18:54 PM
M2:Home:HereAway (175)	Away	Today 3:18:54 PM
M2:HS:HereAway (176)	Here	Today 3:18:56 PM

Figure 183 Geofence Presence Devices

16 Bluetooth Low Energy (BLE) Page (HS3 Only)

HS3 supports two modes of BLE operation. One is for purpose of determining Home vs. Away for a beacon. This is the same capability provided in HS4 on the Local Page, Bluetooth tab. HS3 plugin also supports a more extensive implementation that uses multiple BLE receivers for purpose of placing a beacon on a X/Y grid. Onley one of the two modes can be used at the same time. The following paragraphs describe the location identification mode of operation. See the Local Page, Bluetooth tab for Home-Away mode of operation.

The mcsMQTT support for BLE is intended to provide location information of devices that contain BLE capability. It can be used for presence detection of location identification. The BLE scanner(s) being supported is described in Section 21.16.

BLE is often used in smartphones, fitness bands, smartwatches, audio equipment and tracking beacons. The BLE protocol provides a means for one BLE device to request that other BLE devices advertise their MAC address. In the Section 21.16 implementation ESP32 devices will periodically make this request and the ESP32 will then listen for the MAC addresses that are advertised. After signal processing by the ESP32, information on changes in advertised locations are published via MQTT. mcsMQTT will provide the visualization and HS interface for this information.

The BLE MAC address is a crucial element for the tracking beacon and the ability of devices such as fitness bands to recognize what or who is reporting. For smartphones the MAC address presents a security concern so the MAC address being advertised is usually randomized to protect the identification of the smartphone at any given location.

While discrimination of specific smartphones is not possible, it is possible to determine if the MAC address provided is from a valid supplier and if not then assume it is a randomly generated one and is associated with some smartphone. The ESP32 can be configured to group all MAC addresses without a recognized supplier into a single MAC so that presence can be reported. Location information, however, will not be valid for the case of multiple smartphones being present.

16.1 BLE Page Description

The BLE page is selected from the HS menu under Plugins\mcsMQTT. It provides three tabs. One for the setup and viewing of data from ESP32 microcontrollers that report beacon position information. See Section 21.16 for the ESP32 side of the project which describes the algorithms employed and the MQTT interface specification. The remaining two tabs are used for viewing bubble charts that show beacon locations. One is for current location of all beacons and the movement of a beacon over the last 24 hours. These charts can also be viewed as popups or created on-demand via HTTP requests.

The architecture of the system consists of multiple ESP32 each running the BLE scanning function. The ESP32 will exchange information about the beacon measurements each is making. They will each calculate the distance a beacon is from itself based upon the RSSI measurement. It will communicate this distance to all scanners. Each scanner will calculate the beacon location based upon the distance radius from each scanner. Since all scanners have the same information the location calculation done by each scanner will have the same X,Y result. There are timing differences that will cause slight variance in each calculation. For final location reporting a master scanner is selected. All scanners will report the

same beacon location as computed by the master scanner. Should the master scanner go offline then a new master will be determined.

In addition to (X,Y) coordinates on a 100 ft by 100 ft grid, the scanners also report a Figure Of Merit (FOM) and a Zone. The FOM is an assessment of the quality of the location determination and will range from 0 up to 49.

> A FOM of 0 will indicate that the beacon is no longer in range of any scanner.
>
> A FOM of 11 indicates that a beacon is in view of exactly one scanner. The beacon (X,Y) will be the location of the detecting scanner.
>
> A FOM of 22 to 39 indicates that at least two scanners view the beacon, but the range radius is too small to have overlapping radius of three scanners. The last digit of the range is the FOM is the number of scanners that view the beacon. In these cases, a weighted average algorithm is used to assess a position somewhere in the middle of the beacons that detected the beacon.
>
> A FOM above 33 indicates that trilateration can be done to compute the (X,Y) location of the beacon. Again, the last digit is the number of scanners that can detect the beacon. The location will be determined by a set of three scanners that have overlapping distance radius from the beacon.

The Zone is a single number computed as 100 * X + Y coordinate of the beacon. It is intended to be available for use in detecting a beacon that has moved. Hysteresis is applied to the Zone value so that it will not change with small changes in X or Y. This makes it a good event trigger.

16.2 Getting Started with BLE

There are two major components to the BLE scanning system. One is the MQTT infrastructure that supports the mcsMQTT Plug-in. Sections 2 and 3.1 contain information to setup this piece.

The second is the ESP32 that will need to have the firmware installed that was developed to perform the BLE scan function. Source that was developed based upon a branch of Tasmota using Platform IO and VS Code. This environment provides for compilation and downloading over USB serial. The files of interest are at http://mcsSprinklers.com/ESP32_BLE_BinaryAndLoader.zip (1 MB) binary application, bootloader and partition table http://mcsSprinklers.com/BLEScanner.zip (75 MB) ESP32 source from PlatformIO folders. Updates to the original uploads are at http://mcsSprinklers.com/ESP32BLEScanner.zip for binary and http://mcsSprinklers.com/BLEScannerSource.zip for source.

The firmware can also be installed from a precompiled binary using the tools provided by Espressif at https://www.espressif.com/en/products/hardware/esp32/resources . Select Tools and “Flash Download Tools”. Unzip to a folder, run the .exe and select ESP32 button. A panel shown in Figure 184 will appear. On the top row navigate to the binary to be downloaded. The partition used for the firmware is 0x10000 so enter that at the end of the first row. The same zip file contains two additional .bin files. One is the bootloader and the other is the partition table. Add these in subsequent rows with addresses 0x1000 and 0x8000 respectively. Use checkbox on the three rows. Select the COM port at the bottom of the panel. Click first the ERASE button to clean any prior data from the flash. Click START button. Download will occur with progress bar at the bottom and info about the chip populated in the text boxes on the panel. It takes a few minutes. The button shown as IDLE in green will change to

FINISH when done. Only the first file is needed so uncheck the row 2 and row 3 checkboxes if firmware is loaded later. If the ERASE button is used again then all three are needed to be downloaded.

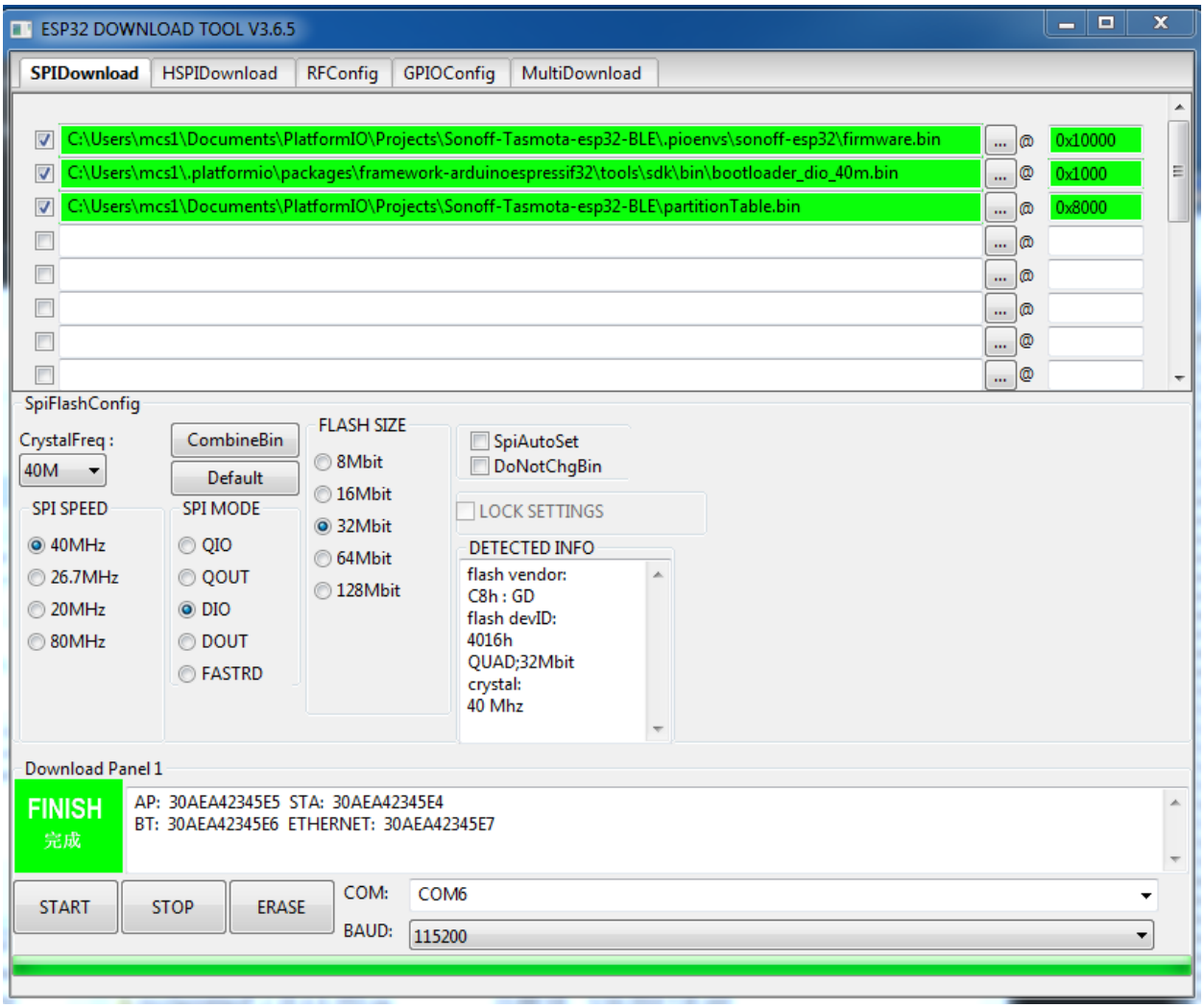


Figure 184 Espressif Flash Download Tool

Start up a terminal program such as Termite and connect to the same COM port as used for the flashing. Termite is at https://www.compuphase.com/software_termite.htm.

Cycle power on ESP32 or use its reset button on the circuit card. A startup sequence will show such as Figure 185. There are three items of particular interest. The first is the group topic BLEScanners shown at the end of the Project line. The second is the MQTT topic that shown as BLEScan/0. The IP will be needed to access ESP32/Tasmota via browser.

If the IP is not shown then the SSID and password are needed. At the bottom of Termite page is a text box to enter data to be sent over the serial connection. The commands that can be used are at https://github.com/arendst/Sonoff-Tasmota/wiki/Commands. In particular the SSID and PASSWORD commands. I use the backlog command to consolidate these two into a single line. This is shown below

where ???? will be replaced by your specific WiFi network credentials. The ";" in backlog command is used as a command separator.

backlog SSID ????; PASSWORD ????

The ESP32 will restart and the startup will be displayed on the serial terminal again. Note the IP address assigned by DHCP server. One can also go to their router or other mechanism to identify the IP address. Use a browser with URL at this IP to complete the setup.

```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:1100
load:0x40078000,len:9232
load:0x40080400,len:6412
entry 0x400806a8

00:00:00 APP: Set Baudrate to 115200

00:00:00 Project BLEScan BLE Scan (Topic BLEScan/0, Fallback DVES_00000000/0, GroupTopic BLEScanners) Version 5.12.0c-STAGED
00:00:00 WIF: Connecting to AP1 U in mode 11
00:00:06 WIF: Connected
00:00:06 DNS: Initialized
00:00:06 HTP: Web server active on BLEScan/0-3632.local with IP address 192.168.0.253
00:00:06 MQT: Attempting connection...
00:00:06 MQT: Connected
00:00:06 MQT: BLEScan/0/LWT = Online (retained)
00:00:06 MQT: BLEScan/0/cmnd/POWER =
00:00:06 MQT: BLEScan/0/INFO1 = {"Module":"WEMOS","Version":"5.12.0c","FallbackTopic":"DVES_00000000/0","GroupTopic":"BLEScanners", "UTimers":"8"}
00:00:06 MQT: BLEScan/0/INFO2 = {"WebServerMode":"Admin","Hostname":"BLEScan/0-3632","IPAddress":"192.168.0.253"}
00:00:06 MQT: BLEScan/0/INFO3 = {"RestartReason":"1"}
00:00:06 MQT: BLEScan/0/CONFIG = {"Location":{"X":0,"Y":0},"Filter":{"RSSI":"0","XY":0,"Zone":0,"Dropout":4},"Management":{"ReceiverGain":100,"ScanInterval":60,"Scan(
00:00:06 MQT: BLEScan/0/STATE = {"Time":"1970-01-01T00:00:06","Uptime":"0T00:00:18","Wifi":{"AP":1,"SSId":"U","RSSI":82,"APMac":"78:8A:20:84:48:1D"}, "UTimers":8}
```

Figure 185 ESP32 Startup Log

It may also be possible to establish the connection using WPS, but I have no experience with this approach.

Select the Tasmota Console button which will provide an interface similar to the serial one. The scanner ID is a number between 1 and 10 needs to be specified. This is done with command like:

ScannerLocation 1,10,20

Where 1 is the ID and 10,20 are the X,Y coordinates on the 100x100 grid that the ESP32 will be placed.

The ScannerLocation command can be entered on the terminal program (e.g. Termite) rather than at the Tasmota Console. No particular advantage of using one vs. the other, but the next step is to configure Tasmota so a browser connection is needed.

16.2.1 Tasmota Configuration

The BLE Scanning operation is affected by the setup of the following Tasmota pages:

Configuration : ConfigureModule - No Effect on BLE scanning. Other modules functions should be able to be selected to extend the capability of the ESP32 to more than BLE scanning.

Configuration : ConfigureTimers – No Effect.

Configuration : ConfigureWifi – Needs to be setup but no direct implications to BLE scanning.

Configuration : Configure MQTT – Needs to be setup but no direct implications to BLE scanning.

Configuration : Configure Logging – When Serial or Web log level is set to “3 Debug” then the scanning report of which beacons were found in the last scan will be shown on the Console. Telemetry period defines the interval when the scanner CONFIG and beacon Characteristics messages will be published. They will also be published on a change.

Configuration: ConfigureOther : The friendly name does not affect the BLE scanning operation, but it is useful to set it so access to a scanner can easily identify which scanner is being accessed. In my case I use the scanner ID so a friendly name may be something like “BLE Scan 3”.

Information – No Effect.

FirmwareUpgrade – Convient way to update the firmware vs. using the serial connection. I have had mixed results with sometimes it works, sometimes I get a timeout and sometimes it tells me that the image is too big.

Console – Used to interact with a single scanner. The additional commands described in Table 7 can be used via the Console.

Restart – No Effect.

16.3 Tips

MQTT topics are formed from the friendly name of the beacons. This means that all scanners need to have the friendly names set for a proper exchange of information and collection of data by mcsMQTT.

It is easier to collect beacon names and then remove the ones that are not of interest rather than trying to catch a particular beacon. The general sequence is to have the BeaconDisable set false (0), collect beacons for several minutes until you are certain that all have been seen and the beacon info exchanged among scanners. Set BeaconDiable true (1) to prevent other beacons from being recognized. Look at the beacons that have been observed by each scanner on the Beacon Location table. Click the R(emove) checkbox followed by the column header button to send the BeaconRemove command to the scanners. What will remain is just the beacons of interest.

RSSI measurements have fluctuation so cannot be considered to be very accurate for distance measurement. Use of the Kalman filter will tend to improve the accuracy, but still movement of the beacon to a different distance may not always result in much of a RSSI difference.

If all scanners and beacons are placed at the same location the variance in RSSI measurements can be observed. If one scanner appears to be out of whack vs. the others then the ScannerGain can be adjusted for it. Use the Distance measurement on the Beacon Locations table to observe the effect of changing the gain.

Location identification is done by different algorithms depending upon the quality of the measurements being received. A good location will be identified when three scanners have intersecting distances from the beacon. If, for example, all beacons distances make it appear the beacon is close to each scanner

then the distance radius will not overlap. This means that when tuning the beacon distances, one should error in making the beacons appear further rather than being closer. For example, if the RSSI at 1 meter fluctuates around 60 and 70 then select a calibration reading closer to 60.

Don't worry too much about maximizing the availability of the trilateration algorithm. The Figure of Merit in the 20's uses the weighted average or sometimes called triangulation algorithm. It will produce reasonable results.

All BLE data should be tagged for Express mode processing unless there are special other needs for the data. There can be considerable MQTT traffic and Express mode minimizes the processing time to handle each received message.

16.4 Setup Tab

The setup tab contains tables for configuration of the ESP32, their status and the viewing of the beacon data. A table also exists to manage the viewing within the setup tab and a pair of buttons for facilitate actions.

When setting beacon parameters from the mcsMQTT BLE Setup tab the entered values will be sent to all ESP32 scanners. If updates are desired to only one scanner then the commands should be entered on the Tasmota Console page per the new command definitions in Table 7.

16.4.1 Page Viewing Options

The page viewing options are shown in Figure 186.

The beacon current location chart is shown on the second tab on the BLE page. It can be generated on demand using the button provided on that tab. It can also be generated and updated automatically as long as the BLE page is open. The "Auto Update of Beacon Location Graphic" checkbox enables the auto update. The negative side to auto update is the CPU resources used, but this should be minor unless considerable beacon motion is occurring or filter parameters are setup such that noisy behavior exists in the location determination.

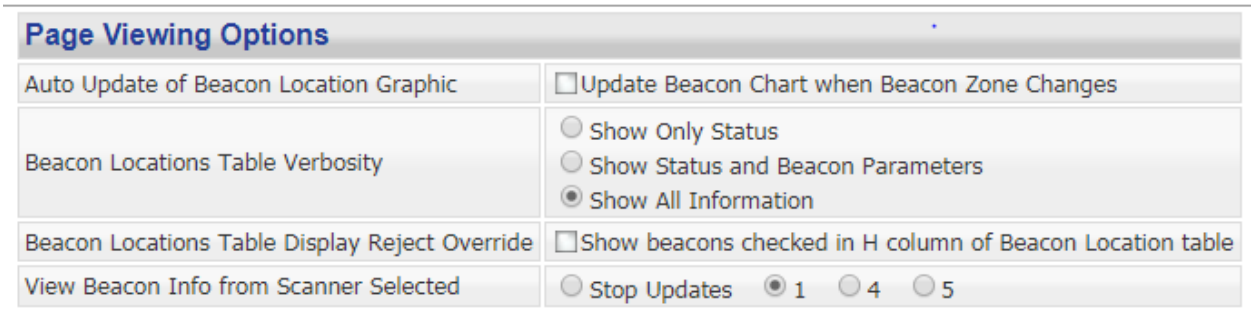


Figure 186 BLE Setup Tab Viewing Options

The beacon table is organized with one row for each beacon and a number of columns that contain information about the beacon. In its most austere format, the columns will show location information. Additional columns can be shown to include the text boxes to setup the transmit power calibration of each beacon. The most verbose format includes the RSSI information to assist with setting up the

Kalman filter parameters. Once created, the Beacon Location table, will generally not be updated as new data is received from the scanners. The “Refresh Beacon Locations Table” can be used to refresh the table with the most current reporting from the scanners.

The first column of the Beacon Location table is a (H)ide checkbox. Using this checkbox will exclude the beacon from the display. This is a declutter function. All hidden beacons can be restored for display by using the “Override” checkbox. Unless there are many beacons this function likely will not be used.

The Beacon table and the Parameters table is populated with data from one scanner. The scanner’s data that is to be used is selected with the “Selected Scanner” radio. As the radio selection is changed the Beacon Location and Parameters tables are automatically updated. The Parameters table is updated in real time. Selecting the “Stop Updates” option will prevent the table entries from being updated as one is trying to make changes to the parameters for subsequent publishing. After the desired changes are made then a specific scanner should be selected so the data in the tables is a reflection of what is being provided by the ESP32.

16.4.2 Configuration Parameters Table

The Configuration Parameters table is shown in Figure 187. Its primary function is to view the settings that exists within the ESP32 and provide a mechanism to change them. The data values in this table are updated based upon the MQTT CONFIG topic that is published periodically or upon a change of a parameter. The specific ESP32 scanner being viewed is selected from the Viewing Options table as described in Section 16.4.1.

It should always be the case that all ESP32 scanners are using the same configuration parameters. All the configuration parameters, including those in the Beacon Location table, can be publish to all online ESP32 scanners using the “Publish all Scanner and Beacon Parameters” button on the Setup tab.

Normally this button is not needed unless a new ESP32 is being brought online and you want to set it to have the same configuration as the others. For ESP32 that have already been setup then and just individual parameter is to be changed then when the change is made in the Configuration Parameters table then it will be published to all ESP32 scanners.

Configuration Parameters	
BLE Scanner Group Topic (e.g. BLEScanners)	BLEScanners Change only used within mcsMQTT
BLE Scanner Root Topic (e.g. BLEScan)	BLEScan Change only used within mcsMQTT
BLE Scanner Scan Interval (seconds)	60 Change will be published to ALL scanners
BLE Scanner Scan Duration (seconds)	30 Change will be published to ALL scanners
BLE Scanner Reporting Frequency	(•) Publish on every scan ( ) Publish on change in beacon X or Y ( ) Publish on change in zone Change will be published to ALL scanners
Beacon RSSI Measurement Error	7 Change will be published to ALL scanners
Beacon XY Measurement Error	0 Change will be published to ALL scanners
Beacon Zone Hysteresis	5 Change will be published to ALL scanners
Beacon Dropout Count	4 Change will be published to ALL scanners
Beacon New Discovery Disable	( ) Retain new beacons found in scan (•) Ignore new beacons found in scan Change will be published to ALL scanners
Multiple Beacon Removal	REMOVEALL REMOVE UNNAMED Change will be published to ALL scanners
Master Scanner Selection	(•) Auto-Select ( ) 1 ( ) 4 ( ) 5 Change will be published to ALL scanners
MQTT Retain Usage	(•) Do not use retain flag when publishing configuration parameters ( ) Use retain flag so MQTT broker will retransmit each time a subscribed client reconnects Change will be published to ALL scanners

Figure 187 BLE Configuration Parameters Table

The top two rows of the Configuration Parameters table are not transmitted to the ESP32 scanners. They are used to tell mcsMQTT the Group and Individual topics recognized by the ESP32 scanners. While these can be changed from the defaults of BLEScanners and BLEScan, there is no need. It is necessary that what is reported by the ESP32 is the same as what is setup in mcsMQTT so it there is a change then make the change in both places. The ESP32 will suffix the base topic with the scanner ID for the individual scanner topics. Tasmota is configured with the same topics used here. The ESP32 will suffix the individual topic with the ID of scanner. When one looks at the topic published by the ESP32 they will see something like BLEScan/1 where 1 is the ID of this scanner.

The Tasmota console or a serial terminal such as Termite is used to define the ID of each scanner individually. The command is “ScannerLocation ID,X,Y” where ID is a number between 1 and 10, X is the grid X coordinate and Y the Y coordinate where this ESP32 scanner is located. These coordinates are on a 100 ft by 100 ft grid.

Most of the remaining parameters are just a user interface to setup the ESP32 parameters per the protocol defined in Section 21.16.5.

The Scan Interval and Scan Duration parameter define how often a scan is performed. 60 second interval and 30 second duration are the values used in the prototype written to evaluate the function. Longer durations allow infrequently reporting beacons to be more easily detected. Shorter durations improve the responsiveness of the location determination.

Reporting frequency is to manage the MQTT traffic based upon what is useful. Every scan will result in a new calculation iteration. Any change in distance between the scanner and beacon will be sent following the scan. This is used by the other scanners to maintain coordination of all distance information being collected. Other information about the beacons and the calculations will be sent periodically at the log rate which is typically every five minutes. They can also be sent more often based upon a change in (X,Y) or change in Zone. This extended information is intended for other clients such as mcsMQTT when one is trying to view more deeply into the execution of the ESP32 scanner.

When assessing the location algorithms within each ESP32 scanner it may be helpful to select a specific scanner to be a master. Only the master's (X,Y) and Zone are reported via MQTT. After the investigation is complete the master should be restored to auto-select to allow the ESP32 to determine the healthiest scanner to be master.

The next four rows of the Configuration Parameters table set the filtering parameters to reduce noise in the location determination. See Section 21.16 for a better understanding of the contribution of each parameter.

Beacon New Discovery as well as the Blacklist the Remove provisions in the Beacon Locations table are used to manage spurious or obsolete beacons. Beacons that are not of interest should be removed to avoid the MQTT traffic that is of no interest. Once all beacons of interest have been identified then the New Discovery should be disabled.

The removal of individual beacons can be done from the Beacon Locations table using the (R)emove column checkboxes. They can also be done in group using the one of the two Multiple Button Removal buttons in the Configuration Parameters table. The REMOVEALL button clears out all the beacons info from all scanners. The REMOVE UNNAMED removes all that have not yet been given a friendly name and continue to use their address for topic identification. This second button will likely be the easiest was to remove beacons that are not of interest. The process would be to first disable new discovery, then to remove all the unnamed ones. What should remain are those that have been named so should be just the ones of interest remaining.

The last parameter in this table is the MQTT retain flag. When mcsMQTT starts it will use the last parameters that existed at the time of its prior shutdown. These may or may not be correct and will not be updated until the periodic message is published from each ESP32. This is typically every five minutes. When retain is enabled then the MQTT broker will hold the parameters published by each ESP32. When mcsMQTT or any other subscribed client goes online the broker will provide the messages immediately. Since each ESP32 also subscribes to data published by the other ESP32 scanners the broker will be resending every time the ESP32 temporarily goes offline and then back online. One a message has been retained by the broker it is a manual process to remove it.

16.4.3 Beacons Locations Table

The Beacons Locations Table serves the purpose of viewing and changing individual beacon parameters. The table is shown in one of three formats depending upon the viewing option selected as described in Section 16.4.1. Figure 188 shows the most verbose version.

Hide | Remove | Blacklist | UnBlacklist						Beacon Locations with Last 24 Hour Data											Distance (ft)	
H	R	B	U	Address	Vendor	Name	TxPower @1m	TxPower @10m	RSSI	Filt RSSI	Zone	FOM	1	2	3	4	5	(X,Y)
☐	☐	☐	☐	54:43:9f:d0:3f:23	Unknown	54-43-9f-d0-3f-23	60	100	-85	-85.00	3709	11	14					(37,9)
☐	☐	☐	☐	5c:4e:ba:c9:9a:02	Unknown	5c-4e-ba-c9-9a-02	60	100	-71	-71.00	3709	11	8					(37,9)
☐	☐	☐	☐	5e:a6:37:04:4b:bb	Unknown	5e-a6-37-04-4b-bb	60	100	-1	-1.00	-101	0	-1	-1	-1	-1	-1	(-1,-1)
☐	☐	☐	☐	60:14:1f:07:0e:b5	Unknown	60-14-1f-07-0e-b5	60	100	-1	-1.00	-101	0	-1	-1	-1	-1	-1	(-1,-1)
☐	☐	☐	☐	b0:91:22:f7:62:bf	Texas Instruments	BlueCharm	60	100	-78	-78.00	3918	24	9	37	-1	35	37	(39,18)
☐	☐	☐	☐	dc:0d:30:47:02:09	Shenzhen Feasycom	Feasycom	60	100	-70	-70.00	3717	24	6	26	-1	41	13	(37,17)

Note that any change in Beacon Locations table except H column will be published to ALL BLE scanners

Figure 188 Beacon Location Table

Checkbox and textbox entries will result in the MQTT message being published to all scanners for the updated (or refresh of existing) value. The leftmost H(ide) column checkbox only affects the displayed page and does not affect the ESP32 scanners.

The R(emove) column will remove the beacon on the selected row to be removed from its flash memory. The B(lacklist) column will mark the beacon as being inhibited from publishing status. The (U)nblacklist column will undo the B(lacklist) column.

The Address column is the unique identifier for each beacon that is produced by the beacon during a scan. The address is hyperlinked to a popup chart of this beacon's location history over the past 24 hours.

The Vendor column will contain the results of a MAC database lookup. The database is in the cloud. Three situations can occur with this. The vendor allocated the MAC address will be identified. The second is that it is not found. In this case "Unknown" will be displayed. The third is too many accesses have been made to the database in the allocated period. "Try Later" will be shown in this case. It will be attempted again the next time the table is generated.

The Name column is used to give the beacon a friendly name. This name will become part of the Topic that is publish for this beacon. An unchanged name will show as the MAC address with the colons replaced by dashes.

The TxPower@1m and TxPower@10m entries serve as the calibration of the beacons transmit power levels. The first is the RSSI when the beacon is 1 meter from the scanner. The second is the RSSI when the beacon is 10 meters from the scanner. Some beacons will not have sufficient power to reach 10 meters so an estimate such as 100 (or -100) can be used. These can be adjusted as one learns the behavior of the scanner/beacon relationship. The objective is to get reasonable distance measurements from the RSSI reading. When making the measurements the RSSI filter should be turned off or set to a small value as described in Section 16.4.2 so the distance reading will be sufficiently responsive to the beacon position.

The next two columns show the RSSI reading and its value as it is filtered through the Kalman filter. In the sample shown it can be seen that the RSSI and filtered RSSI are the same which is an indication that the RSSI filter is turned off.

The Zone is a single number that can be used as a trigger. Small variances in the (X,Y) location will not result in the Zone value change, but larger ones that exceed the zone hysteresis threshold (described in

Section 16.4.2) will result in a change of the Zone. Note that the Zone is the value determined by the master scanner. The other scanners just relay the master's value.

The Figure Of Merit was previously described in the introductory Section 16. It provides a feedback as the algorithm used and the number of scanners that are able to hear the beacon. Higher values generally are better figures of merit.

X and Y are the calculated coordinates of the beacon. Like the Zone it is the value determined by the master. Other scanners serve as a relay. This allows HS to see the same value no matter which scanner happens to be the master at any given time.

The last set of columns are the distance measurements from all scanners for a given beacon. In the example scanners 2 and 3 had not reported data in the last 24 hours so no information was available for them. Distances are reported as -1 if no scanner has the beacon in range. In this example only scanner 5 had seen the first beacon in the last 24 hours and it no longer has it in range.

The address is a hyperlink that when clicked will pop-up the graphic that shows the 24 hour history of the beacon. This is the same graphic that can be view on the 24 hour tab.

16.4.4 Scanner Location Table

The Scanner Locations table provides visibility into the scanners online status and the provisions to setup its X,Y location as well as a factor to represent the receiver's antenna gain. Figure 189 provides an example showing three scanners.

The red highlight is an indication that the scanner is offline. In this case it is scanner 1. Because of this offline status the other ESP32 have agreed to use Scanner 4 as the master for reporting the X,Y and Zone.

The scanner location can be setup when the scanner Id is defined via serial or the Tasmota Console. Alternately is can setup on the Scanner Locations table by entering the X comma and Y coordinates for the desired scanner.

The receiver gain is nominally 100 percent. The gain can be reduced to make beacons appear closer. This may be necessary if the scanner is partially blocked for RF. To make them appear at a greater distance then the gain is increased. Greater distance has advantages in assuring that multiple beacons will have overlapping radius thus improving the quality of location identification.

The scanner ID is a hyperlink that when clicked will popup the graphic that shows the current locations of all the beacons that are currently in range of at least one scanner. This is the same graphic that can be view on the Current tab.

Scanner Locations				
Scanner ID	LWT	XY Location	Receiver Gain	Selected for Master Scanner
1	Offline	37,9	100	Scanner 1
4	Online	58,5	100	Scanner 4
5	Online	20,30	120	Scanner 4
Note that any X.Y change in Scanner Locations table will publish the change to the ONE changed scanner				

Figure 189 Scanners Locations Table

16.5 Location Tab

The Location Tab contains the graphic of the beacon locations at the time the BLE page was produced. A button exists at the top to manually regenerate the graphic. It is also possible to regenerate it automatically when the Zone value changes. This is setup as shown in 16.4.1.

Figure 190 provides an example of the location of two beacons with respect to five scanners. The beacons are circles and the scanners are squares. The diameter of the beacon reflects the number of scanners that are able to hear the beacon. Larger size implies more scanners. In this example there were two scanners that were hearing BlueCharm beacon and four were able to hear Fessycom beacon. Beacons with a Figure Of Merit of zero are excluded from the graphic.

The scanners are labeled with their Id number. The beacon is labeled with its friendly name. There are two other ways to produce this chart. One is to produce a popup by clicking on one of the scanner Id number hyperlinks in the Scanner table of the Setup tab. A second is to create the chart on demand via HTTP request such as:

```
http://192.168.0.14/BLE?Beacon=current
```

In this case the filename in the HS \HTML\mcsMQTT folder that contains the graphic will be returned from the http request.

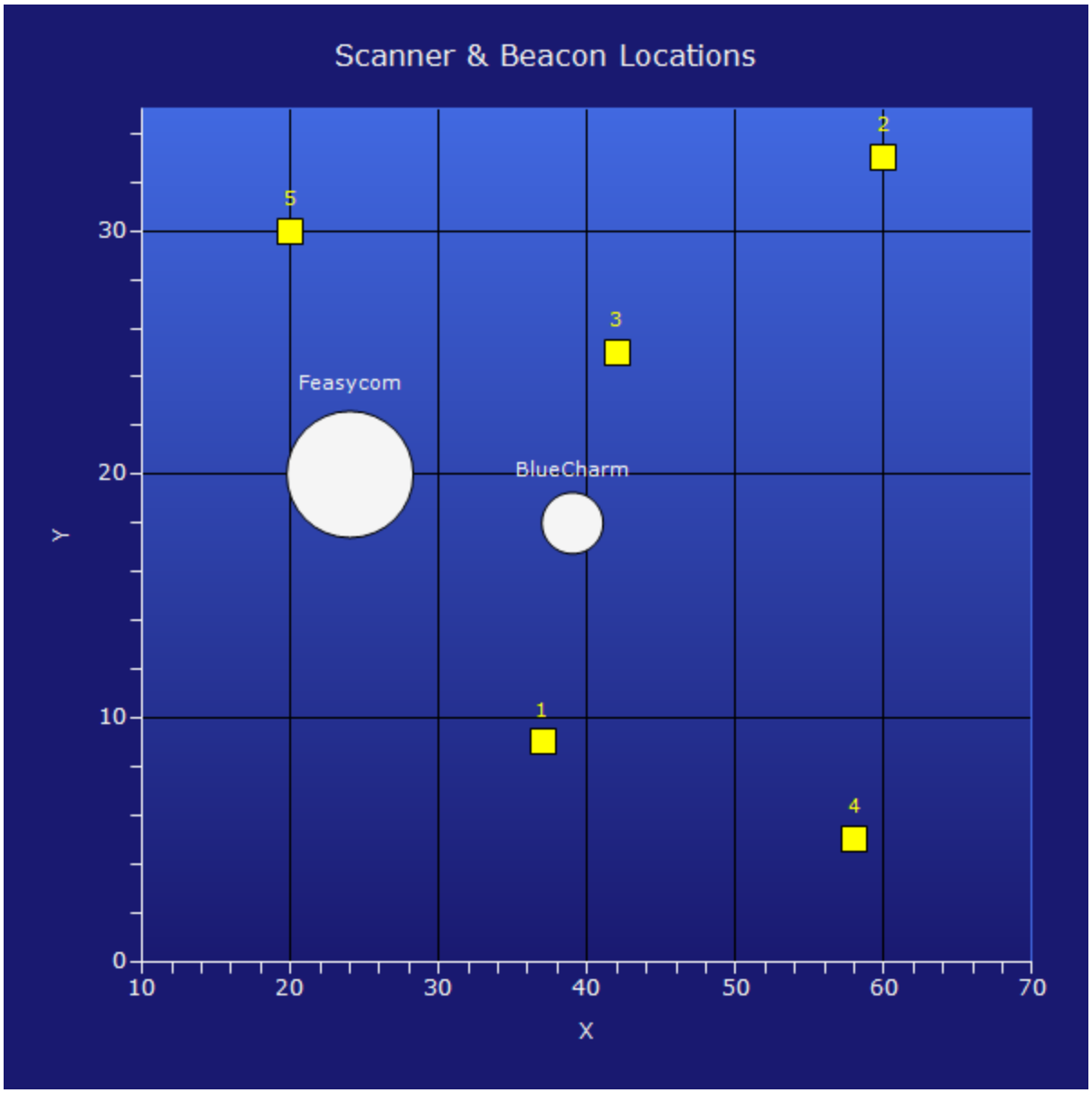


Figure 190 Beacon Current Locations Graphic

16.1 Distance Tab

The Distance Tab contains the graphic of the beacon distance from each of the scanners at the time the BLE page was produced. A button exists at the top to manually regenerate the graphic. It is also possible to regenerate it automatically when the Zone value changes. This is setup as shown in 16.4.1.

Figure 191 provides an example of the location of two beacons with respect to three scanners. The beacons are circles and the scanners are squares. The diameter of the beacon reflects the distance from

the scanner for the scanners that are able to hear the beacon. In this example there were two scanners that were hearing BlueCharm (green) beacon and three were able to hear Fessycom (bisque) beacon. While not shown on the graphic, the Figure Of Merit (FOM) was 22 and 23 for BlueCharm and Fessycom respectively. The first "2" indicates that triangulation algorithm was used to identify location. The second digit reflects the number of scanners able to hear the beacon. In the Fessycom case there were three, but as the graphic shows that the distance circles do not intersect so triangulation rather than trilateration was used to determine location.

The scanners are labeled with their Id number. The beacon is labeled with its friendly name and color used to distinguish the beacons. There is one other ways to produce this chart is with a chart on demand via HTTP such as:

```
http://192.168.0.14/BLE?Beacon=distance
```

In this case the filename in the HS \HTML\mcsMQTT folder that contains the graphic will be returned from the http request.

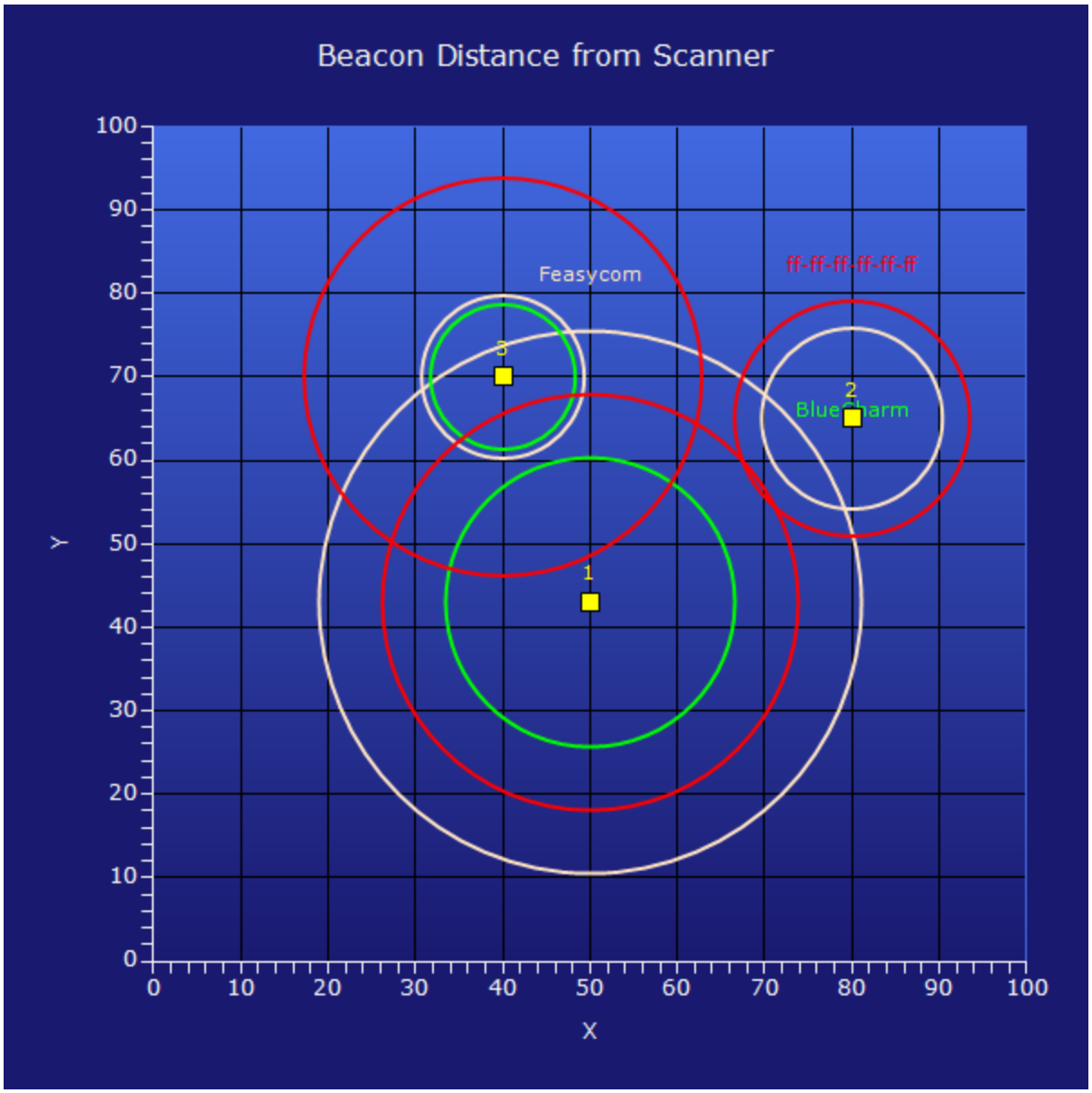


Figure 191 Beacon Distance from Scanner Graphic

16.2 24 Hour Tab

The 24 Hour Tab contains a selector to choose among the beacon to show their location history over the past 24 hours. To collect history data to support this graphic the History tab on the MQTT Setup page needs to enable history data collection for at least one day.

No graphic is shown at time of BLE page generation. Once a selection is mode the graphic will appear below the selector.

Figure 192 provides an example of the BlueCharm beacon with respect to five scanners. The beacon positions are circles and the scanners are squares. The cases where the beacon overlays the scanner position, such as the upper left scanner in this figure, is an indication that this scanner was the only one that was hearing the beacon at that time.

The scanners are labeled with their Id number. The beacon is identified in the chart title. There are two other ways to produce this chart. One is to produce a popup by clicking on one of the beacon address hyperlinks in the Beacon Locations table of the Setup tab. A second is to create the chart on demand via HTTP request such as:

```
http://192.168.0.14/BLE?Chart=BlueCharm
```

In this case the filename in the HS \HTML\mcsMQTT folder that contains the graphic will be returned from the http request.

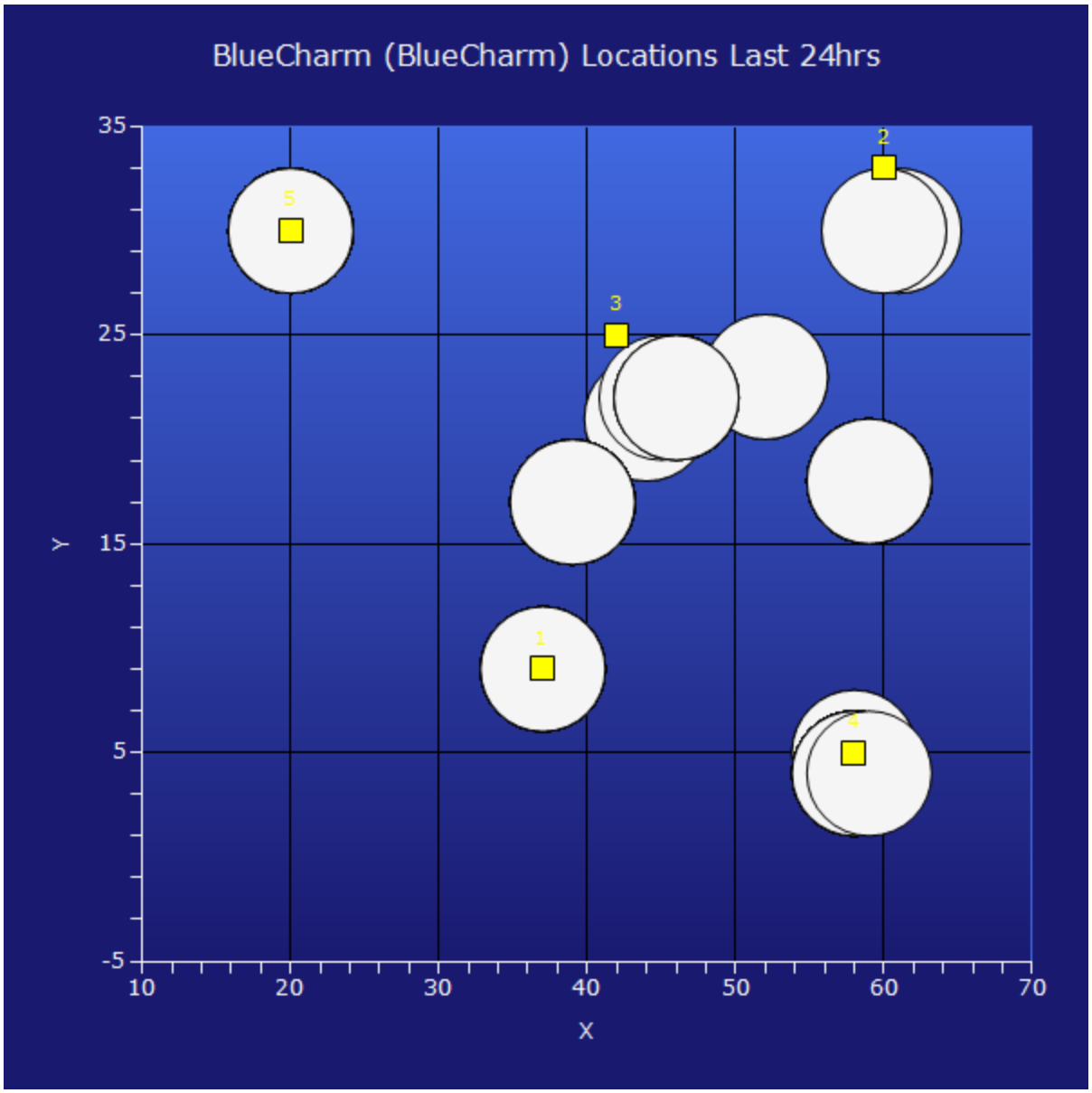


Figure 192 Beacon 24 Hours History Graphic

17 Performance Considerations

mcsMQTT has the potential to manage a large set of data or it can be used in a very modest environment. In the later situation it is likely that maximum user conveniences are made available. In the former compromise may be needed to have a sufficiently responsive and CPU-considerate operation.

17.1 HS Event Callbacks

mcsMQTT subscribes to HS callbacks for Device Value change or Device String change. If there are many active HS devices then many callbacks could be generated and each evaluated to determine if a MQTT Topic needs to be sent. mcsMQTT determines if a user has selected any Publish Value/Status and if so, registers for the Value callback. It does the same for String. If at least one Device is mapped to Value and String Topic publication then all changes in HS devices will be routed through mcsMQTT.

The callbacks are used to trigger publication of MQTT messages and to collect data for charting. If all HS devices are selected the database will tend to grow large more quickly. The History tab selection for all vs. only specific devices selected with Association tab D column checkbox will affect the database size and CPU supporting this capability.

mcsMQTT maintains a queue of HS Event callbacks. In the callback it first assesses if the Device has a publish Topic associated with it. If it does then it puts it in the Queue for later processing in an independent thread. The Queue is processed until empty and then the thread is blocked until another item is put in the Queue.

17.2 Express Mode

Express mode, selected on a Topic-by-Topic basis on the Association tab provides a means to update HS devices based upon MQTT payload with a minimum of processing overhead. A CPU reduction of better than 80% can be achieved for Topics identified for Express mode.

Express mode excludes the following:

- ability to store history of received messages with implications that charts cannot be made
- ability to use low pass filter on Device values, make rate devices or accumulate devices
- ability to store JSON content with parent and children devices grouped
- ability to map color x/y to RGB
- ability to collect VSP relationships after Topic selected as Express
- ability to refresh Association tab GUI payload and last change columns as Topics received

Express mode can be customized on the Client tab to include or exclude the following. There is little CPU penalty for supporting these features, but if a feature is not used then some CPU savings can be achieved by deselecting them on the Client Tab.

- decode JSON into separate HS devices
- use VSP text and color picker RGB to DeviceValue mapping

- use regular and arithmetic expressions
- trigger HS events and script callbacks based upon received Topic

Table 4 shows a benchmark that compares Express mode vs. Full mode CPU utilization to process a message with content shown for the Test Case. A RPi model 3 was used for the test. A test program generated a MQTT message at 500 millisecond intervals. The time to process the message was observed in the Statistics window and recorded. In general, the inclusion of additional features supported in Express mode has a small effect on the CPU burden, but in situations of high MQTT traffic these may become material.

During the analysis the most illuminating discovery is that almost all of the time for processing messages was within HS following the calls to SetDeviceValue or SetDeviceString. When removing these calls, the Express mode times were under 1 millisecond.

During Full support mode calls are made to get the HS device object (dv = hs.GetDeviceByRef) which consumed 12 milliseconds and then each call that used the hs object as a parameter (e.g. dv.Interface(hs)) used a similar time so working with the HS device object became expensive . In version 3.5.5.0 the use of the hs parameter was eliminated in favor of Nothing for the real time processing. The assurance that DeviceString was empty for non-text devices was removed from the real time processing and done during initialization. These steps resulted in a significant reduction in the CPU utilization for Full support mode.

Table 4 Receive Message Benchmark for Full vs. Express Modes

Test Case	V 3.5.1 Full Support	V3.5.5 Full Support	V3.5.5 Express Mode
Incrementing Number	200 ms	32 ms	6 ms
VSP Toggle On / Off	200 ms	32 ms	6 ms
JSON with 5 keys and 1 associated with HS Device	310 ms	33 ms	6 ms
JSON with all 5 keys associated with HS Device (Number, Text, VSP)	510 ms	142 ms	32 ms

Express Mode does not support Parent/Child grouping. If any item of a JSON group is selected for Express Mode then all items of that group will be forced to Express Mode as well. The opposite occurs for deselection of any item that was previously selected for Express Mode.

17.3 Subscription Topics

By default, mcsMQTT subscribes to all Topics serviced by the connected Broker. This is done by providing the wildcard # as the template for subscription along with any special Topics that have been manually entered which contain Broker statistics. When mcsMQTT starts, it will connect to the Broker and the Broker will respond with a burst of messages that it is responsible to deliver to newly subscribed clients. The Retain=true flag per MQTT protocol used by the various clients will increase the size of the initial connection message quantity. As a minimum the Broker will deliver the LWT status of all clients.

mcsMQTT maintains a receive Queue to deal with the flooding at initial connection and in general control the peak CPU utilization during message bursts. Threre user settings are available on the Client Tab Inbound (Subscription) Section to tune operation of the receive Queue. It is likely that no tuning is needed, but it is available. mcsMQTT will process a set of messages as a group and then yield the CPU for a user-specified period and then resume the process until the Queue is empty. When the number of messages received exceed the size of the queue then they are discarded and not included in the statistics.

Visibility to help tune the parameters is provided at the bottom of the MQTT/Statistics Topics shown in Figure 193. A current Queue depth near 0 will provide the lowest latency between message receipt and update of HS Device. The max size will usually occur upon initial connection. The average processing time measures the time it takes to update HS devices and evaluate trigger conditions for each received message. The average receive milliseconds is the interval between each incoming MQTT message on the average. These averages will be dynamic at startup and become stable as a large number of messages are processed.

Filter Association Table by Mqtt Topic and JSON Payload Key Clear Filters Rebuild Filters

T1 MQTT | T2 | T3 | T4 | T5 | T6

J1 | J2 | J3 | J4 | J5 | J6

Show Selected Associations

Prev 0 of 21 Next

Association Table for Auto Association of MQTT Topic and HS Device

Λ	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
0	☐	☐	☐	☐		Sub: MQTT/Statistics/1A-Broker Connection Status	192.168.0.16:Online @ 2021-04-04 10:28:58 for 0 Days 0 Hours 6 Mins 2 Secs				2021-04-04 10:35:01
1	☐	☐	☐	☐		Sub: MQTT/Statistics/1A-Broker Connection Status(2)	192.168.0.30:Online @ 2021-04-04 10:28:58 for 0 Days 0 Hours 6 Mins 2 Secs				2021-04-04 10:35:01
2	☐	☐	☐	☐		Sub: MQTT/Statistics/2A-Publish Count	2				2021-04-04 10:35:01
3	☐	☐	☐	☐		Sub: MQTT/Statistics/2B-Publish Count Today	2				2021-04-04 10:35:01
4	☐	☐	☐	☐		Sub: MQTT/Statistics/2C-Receive Associated Count	11				2021-04-04 10:35:01
5	☐	☐	☐	☐		Sub: MQTT/Statistics/3A-Receive Associated Count Today	11				2021-04-04 10:35:01
6	☐	☐	☐	☐		Sub: MQTT/Statistics/3B-Receive not-Associated Count	193				2021-04-04 10:35:01
7	☐	☐	☐	☐		Sub: MQTT/Statistics/3C-Receive not-Associated Count Today	193				2021-04-04 10:35:01
8	☐	☐	☐	☐		Sub: MQTT/Statistics/4A-Last Published Topic	MCS8/mcsMQTT/LWT				2021-04-04 10:35:01
9	☐	☐	☐	☐		Sub: MQTT/Statistics/4B-Last Published Payload	Online				2021-04-04 10:35:01
10	☐	☐	☐	☐		Sub: MQTT/Statistics/4C-Last Received Associated Topic	homeassistant/switch/CFB4C6_RL_1/config				2021-04-04 10:35:01
11	☐	☐	☐	☐		Sub: MQTT/Statistics/4D-Last Received Associated Payload	{"name":"Subaru","stat_t":"Subaru/STATE","avty_t":"Subaru/LWT","pl_avail":"Online","pl_not_avail":"Offline","cmd_t":"Subaru/cmnd/POWER","val_tpl":"{{value_json.POWER}}","pl_off":"OFF","pl_on":"ON","uniq_id":"CFB4C6_RL_1","dev":{"ids":["CFB4C6"]}}				2021-04-04 10:36:11
12	☐	☐	☐	☐		Sub: MQTT/Statistics/4E-Last Received Topic	Subaru/HASS_STATE				2021-04-04 10:36:11
13	☐	☐	☐	☐		Sub: MQTT/Statistics/4F-Last Received Payload	{"Version":"8.4.0.3(tasmota)","BuildDateTime":"2020-09-25T12:19:37","Module or Template":"Generic","RestartReason":"Power On","Uptime":"40T20:20:20","Hostname":"Subaru","IPAddress":"192.168.0.139","RSSI":"72","Signal (dBm)":"-64","WiFi LinkCount":3,"WiFi Downtime":"0T00:00:45","MqttCount":4,"LoadAvg":45}				2021-04-04 10:36:11
14	☐	☐	☐	☐		Sub: MQTT/Statistics/4G-Last Received Timestamp	2021-04-04 10:36:02				2021-04-04 10:36:11
15	☐	☐	☐	☐		Sub: MQTT/Statistics/5A-Receive Queue Depth	0				2021-04-04 10:36:11
16	☐	☐	☐	☐		Sub: MQTT/Statistics/5B-Max Receive Queue Depth	86				2021-04-04 10:36:11
17	☐	☐	☐	☐		Sub: MQTT/Statistics/5C-AverageReceiveTime	42				2021-04-04 10:36:11
18	☐	☐	☐	☐		Sub: MQTT/Statistics/5C-AverageReceiveTime(2)	95				2021-04-04 10:36:11
19	☐	☐	☐	☐		Sub: MQTT/Statistics/5D-AverageReceiveInterval	36000				2021-04-04 10:36:11
20	☐	☐	☐	☐		Sub: MQTT/Statistics/5D-AverageReceiveInterval(2)	1982				2021-04-04 10:36:11

Figure 193 MQTT Statistics

It is also possible to have the statistics reflect in HS Devices for other uses. This option is selected with the "A"ssociate checkbox on the desired statistic. These devices will be grouped in a parent/child association with the parent being "MQTT Statistics". See Figure 194 as an example of selected statistics mapped into HS.

Events triggered by a change in connection status can be done either using the mcsMQTT connection trigger (See Figure 39) or DeviceValue change of the statistics online/offline Device.

The last statistic is the CPU utilization over the past 60 second interval and represented as a percentage. It is possible to setup an event that monitors the CPU use of the plugin. It is the same as the statistic preported as part of the HS pseudo-topic.

MQTT | Statistics

MQTT-Statistics (5597)		Today 10:44:34 AM
1A-Broker Connection Status(2) (5598)	192.168.0.30:Online @ 2021-04-04 10:28:58 for 0 Days 0 Hours 17 Mins 38 Secs	Today 10:44:34 AM
2A-Publish Count (5599)	0	Today 10:44:52 AM
2C-Receive Associated Count (5600)	0	Today 10:44:57 AM
3A-Receive Associated Count Today (5601)	0	Today 10:45:38 AM
4A-Last Published Topic (5602)	Device Created	Today 10:45:48 AM
4E-Last Received Topic (5603)	IrrigationWater/SENSOR	Today 10:46:36 AM
5A-Receive Queue Depth (5604)	0	Today 10:46:03 AM
5B-Max Receive Queue Depth (5605)	0	Today 10:46:04 AM
5C-AverageReceiveTime (5606)	40	Today 10:46:27 AM
5D-AverageReceiveInterval (5607)	75571	Today 10:46:37 AM

Figure 194 MQTT Statistics in HS Devices

17.4 Plug-in Startup

During startup the Plug-in performs a two-pass initialization. The first pass covers those things that need to be setup to continue communication with HS. When this is complete a new thread is spawned to cover those things that are needed to completely restore the plug-in to its state at time of last shutdown.

For setups that have a large number of HS devices it takes some time to enumerate all of them to make them available for viewing on the Association Tab. In most setups the HS device does not change that often and then even when they do it may not be of interest to being a Topic that will be published via MQTT. The Client Tab (Figure 40) Outbound (Publish) Section contains a radio to enumerate devices at startup or only manually by button press.

The received topics and those setup for association are maintained in the \data\mcsMQTT.db database. During startup the database contents are validated and transferred to RAM cache for better real time performance. This process was benchmarked at approximately 30 milliseconds per topic. If there are a significant number of topics being managed then this can take a significant amount of time before mcsMQTT is totally ready to interact with HS. Two approaches are availabled to deal with environments that produce a large number of topics.

The Client tab, Inbound Management section, Topic Discovery row allows the user to subscribed to all topics being managed by the broker (the default) or to only receive a subset. If the subscription is limited then there will be less for mcsMQTT to manage and performance will improve.

The General Tab, Obsolete Unassociated row has a checkbox that will remove all topics at shutdown that had not yet been associated with HS device or used for history data retention. This allows new topics to be recognized during a session, but if they have not been associated with HS device they will be removed from the database as shutdown. This means that the next startup will not have the records in the database and the startup will be faster.

The plugin will monitor for excessive accumulation of records from received MQTT Topics. The user has the ability to set this threshold on the General Tab, Receive Topic Warning Threshold textbox. The default value is 10000 records which was selected based upon a benchmark of initialization times of 15 records per second on a 10-year-old Windows laptop (i.e. 12 minutes) and the size of the Device table beign used by HS. Each day the plugin will evaluate the size of the Association Table and provide a warning in HS Log if it exceeds this threshold. It will also automatically remove the oldest unassociated records to bring the record count to 75% of the warning setting when the warning setting is exceeded or remove all unassociated records if there are more that 1000 unassociated.

During normal operation and much larger record count can be managed without a noticeable performance penalty. This is a reflection of the general design philosophy of cacheing in RAM those things that are part of repetitive operation. This tradeoff comes at the expense of startup time to build the cache and validate the integrity of the data. The use of the the General Tab, Obsolete Unassociated row checkbox is the recommended means to keep the Association Table size in check, but it is only activated following shutdown/restart of the plugin.

The timing for startup and page rendering can be viewed in the debug file \data\mcsMQTT\mcsMQTT_Debug.txt. This will allow for assessing the benefit/cost of various settings for any given environment.

17.5 Browser Page Rendering

When the mcsMQTT page is requested by either direct URL or from the Plug-in button on HS Browser page, the is generated with multiple tabs. In general, the Association and History Tab content will only be the selection filters with tables of data for each only produced when manually requested by button push. The exception is for the Associations table that will be initially built if it is small. Small is defined by the user setting in the mcsMQTT Management Section of the Client Tab.

All updatable and user-entry fields on the Browser page are encoded for update via AJAX protocol. On user action or otherwise two second intervals changes will be updated on the page. The two-second update is initiated by a timer in the Browser that does a post back to HS Server which forwards it to mcsMQTT to send any updates needed via AJAX.

Independent Browser pages are managed by mcsMQTT. Within a Browser page the fields between the Association Tab and Edit tab are synchronized as changes are made on either. This synchronization does not occur across Browser pages. A Browser page that remained open when the plug-in started or otherwise has expired will no longer be updated by mcsMQTT, nor will user entry on the expired page be recognized.

18 Reference Tool Tips

18.1 MQTT Page Association Tab

The Association tab is where most HS-MQTT activity is initiated. This tab contains a table that can be quite large that will show all received MQTT topics and all HS devices. A user then needs to only click the "A"ssociate column checkbox to map the topic to the HS device. If the communication is to be bidirectional then a publish topic is also entered by the user in the textbox provided. Note that the user may want to user the Edit popup or tab to specify the format of the MQTT payload that is being published if it is not a simple numeric value.

Because the table can be quite large there are filtering provisions provided in the form of checkboxes, textbox, and pull-down selectors. Use of these limits the size of the table. Once the filters are setup the show button is used to display the selected topics and devices.

Three filter tables are provided. The first is a textbox into which the user can specify text that must exits in the MQTT record. The record has multiple fields such as the subscribe Topic, the publish Topic, the Payload, the VSP entries when mapped to HS devices and others. The text of interest is specified as either simple text or as a regular expression. Regular expressions use the syntax REGEX(expression), <<REGEX(expression)>>, or <<expression>>. Simple text is the pattern that must exit in one of the MQTT record fields.

The second is oriented to HS devices and allow selection of one or more of the Device properties. The second is oriented to subscription topics. Each segment of the topic (e.g Home/Printer/Status) is listed in the pull-down corresponding to its position in the hierarchy. In addition if JSON keys are in the payload then these can be selected as well (e.g. Wifi:{MAC:1234565789012,…} will include Wifi and MAC in the two JSON selectors.)

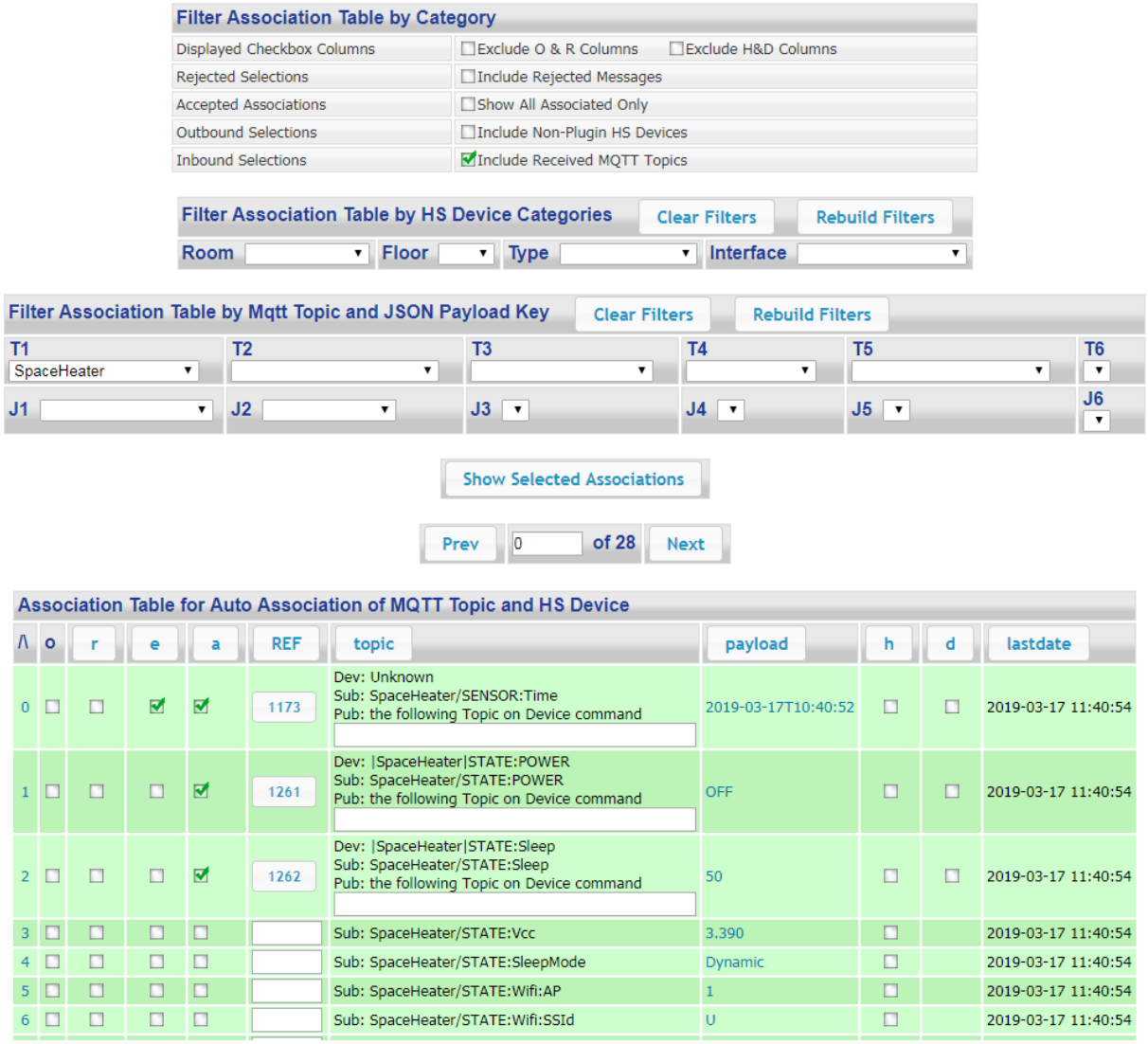


Figure 195 Association Tab

18.1.1 Filters to Restrict Number of Displayed Rows in Association Table

18.1.1.1 Accepted Associations

"Exclude 'O'bsolete and 'R'eject columns in Association table display for selection of obsolete non-Displayed rows. When obsolete row is checked it will be removed.

"Exclude 'H'istory and 'D'evice columns in Association table display for selection of History filters for subscribed and published topics"

18.1.1.2 Rejected Selections

"Checked overrides the 'R'ejected checkbox on each topic to allow them to be shown in Association Table"

18.1.1.3 Accepted Associations

"Checked hides all devices that have not yet been 'A'ssociated from page"

18.1.1.4 Outbound Selections

"Check to include HS Devices in Association Table to select for publication"

18.1.1.5 Inbound Selections

"Check to include received MQTT Topics in Association Table to select for HS device creation"

18.1.1.6 Filter By Text

"Enter text pattern that must exist in some field of the MQTT records for records that should be included in the Association Table. Simple text can used or regular expression can be used with syntax <<xxx>>, <<REGEX(xxx)>> or REGEX(xxx)."

18.1.1.7 Filter Association Table by HS Device Categories

"Press to clear all device pull-down filters"

"Press to reconstruct all device pull-down filters"

18.1.1.8 Filter Association Table by Mqtt Topic and JSON Payload Key

"Press to clear all Topic/JSON pull-down filters"

"Press to reconstruct all Topic/JSON pull-down filters"

"Filter on Topic Segment # "

"Filter on JSON Segment Key # "

18.1.2 Associations Build/Display Control

Up to twenty rows of topics will be shown in the Association table. Previous, Next and Row locator controls are provided to select the starting row of these twenty.

At the top of each column there are also sort buttons that affect the order of the topics presented.

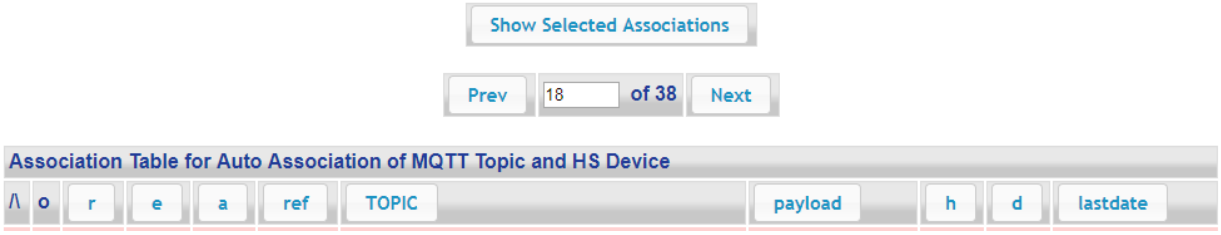


Figure 196 Association Tab Build/Display Control

18.1.2.1 Show Selected Associations

"Press to show MQTT message and HS Device relationships based upon filters that have been setup"

18.1.2.2 Prev/Next

"Association table is shown 20 rows at a time. Enter the starting row"

"Click to display previous 20 rows in Association Table"

"Click to display next 20 rows in Association Table"

18.1.3 Association Table Header

"Sort on Obsolete"

"Sort on Reject"

```
"Sort on Express"
"Sort on Associate"
"Sort on Device Reference"
"Sort on Topic"
"Sort on Payload"
"Sort on History"
“Sort on Chart”
"Sort on Long Term"
"Sort on Last Date"
```

User entry is allowed in most of the columns of the Association table. Some of the entry options are hidden until another entry is made on the same row to make these additional inputs options relevant.

The rows are color-coded. If green the topic is a subscription of inbound traffic. If pink it is HS Device that will publish when it changes. If blue it is an existing HS Device that will be commanded by the subscribed topic. The most used rows will likely be the green ones that will enable HS to have access to MQTT topics published elsewhere.

The leftmost columns of the table contain three checkboxes. The most-often used with be the “A”ssociate checkbox to establish an association between a MQTT topic and a HS Device. For the green rows the “A”ssociate will create a HS Device and place the payload in the Device. Each update to the payload will be reflected in the Device. It will also open publish text box that can be used to specify a topic to be published when the HS Device has been commanded. Note that the UI for the command will be based upon the payload received. If it is a number then a text box will be presented for number entry. If it is something like On or Off then a two-state button will be presented. Other options are available and these can be edited later from the Edit tab.

Following “A”ssociate on the pink rows the user needs to specify the publish topic. The Client Tab default topic will be initially populated, but it can be changed. Since pink rows are existing HS Devices there will be no new HS Device created.

The other two checkboxes “H”istory and “R”eject are used for filtering specification on a topic-by-topic basis. Those tagged for “H”istory will be included in the History database. Those tagged for “R”eject will not normally be shown later in the Association Table unless overwritten by a Show Reject checkbox above the Association table.

The Ref column is the HS Device Reference number. It is a unique number of each HS Device. If the topic/Device has been “A”ssociated then the Ref is shown as a button. When clicked the Edit tab will be populated with the properties of this Ref Device. If a text box is shown then an existing HS Device Ref number can be entered to be associated with the green row topic and this topic will then be used to command the existing HS Device.

Inbound payloads may not be in the desired format when used within HS. Regular expressions can be used on the payloads before being stored in the Device Value or Device String. The first part of the regular expression is the match pattern. The second part is a replacement pattern. A checkbox is used

to specify if the match is to be extracted or if it is to be replaced by the replacement pattern. Simple manipulations are date format or command vs. period for decimal.

Association Table for Auto Association of MQTT Topic and HS Device

Λ	o	r	e	a	ref	TOPIC	payload	h	d	lastdate
18	☐	☐	☐	☑	71	Dev: Unknown Command HS Device on subscribed Topic: test/Dev27 Publish message on Device change using Topic: Dell-PC/mcsMQTT/Unknown/Unknown/NonPluginBu Encode Payload per template:		☐	☐	0001-01-01 00:00:00
19	☐	☐	☐	☑	80	Dev: Unknown Sub: test/JSON:color:x Pub: the following Topic on Device command	5040	☐	☐	2019-02-15 09:06:12
20	☐	☐	☐	☑	1095	Dev: Test|Unknown|RefTest Command HS Device on subscribed Topic: Test/RefTest Publish message on Device change using Topic: MCS7/mcsMQTT/Test/Unknown/RefTest Encode Payload per template:	Test Payload	☐	☐	2018-11-27 11:16:36
21	☐	☐	☑	☑	1167	Dev: |Test|Timon2 Sub: Test/Timon2 Pub: the following Topic on Device command MCS7/mcsMQTT//Sprinklers/Area_3_Status	5071	☐	☐	2019-02-21 13:10:10
22	☐	☐	☑	☑	244	Dev: Unknown Sub: Test/Timon3:DS18B20:Humidity Pub: the following Topic on Device command	5.0	☐	☐	2019-02-21 13:03:49

Figure 197 Association Table

18.1.3.1 O(bsolete)

"Check to remove obsolete topic from Association table and database"

"Click to delete all rows marked as obsolete"

18.1.3.2 R(eject)

"Check to remove topic row from page unless overridden by show rejected checkbox"

18.1.3.3 A(ccept)

"Check to associate new HS Device with MQTT Topic. If Topic is to be associated with existing HS Device then use the Ref box to the right to enter the non-Plug-in Device Reference."

18.1.3.4 E(xpress)

"Check to assign this Topic to Express mode where only update of HS Device will occur. No other support such as event actions or history will occur in Express mode."

18.1.3.5 H(istory)

"Check to include topic in history recording"

18.1.3.6 S(hort term)

"Check to include device in SQLite recording"

18.1.3.7 L(ong term)

"Check to include device in InfluxDB, mySQL or MS SQL Server recording"

18.1.3.8 Ref(erence Number)

"Enter an existing HS Feature to associate this subscribed topic. This can be the Ref of a non-plugin device or it can be the Ref of another mcsMQTT device to associate multiple MQTT topics to the same HS device. If new HS device is to be created then check the 'A' box"

"Click to populate Edit tab with this Ref and properties"

18.1.3.9 Pub: the following Topic on Device command

"Enter Topic published when HS commands Device"

18.1.3.10 Command HS Device on subscribed Topic:

"Received message Topic to command HS Device to change. Received Payload will be the command"

18.1.3.11 Publish message on Device change using Topic:

"Substitutions are $$NAME:,$$ROOM;$$FLOOR;$$COMPUTER:,$$INTERFACE:,$$INSTANCE:,$$TYPE:,$$ADDRESS:,$$CODE:,$$REF:, $$TASMOTACMND: and those in the payload list"

18.1.3.12 Encode Payload per template:

"Substitutions are $$VALUE:,$$VALUE_EUROPE:,$$PREVIOUS:,$$PUBLISHED:,$$STRING:,$$STATUS:,$$LABEL:,$$VSP:,$$PAYLOAD:,$$TOPIC:,$$DATE:,$$TIME: and those in status Topic list"

18.2 MQTT Page Edit Tab

The Edit tab is the place where edits are made to existing MQTT-HS associations or where subscriptions are manually defined. Manual entry is for case of the broker topics that start with "$SYS" as these are not normally subscribed by mcsMQTT, if Topic Discovery is turned off on Client Tab, or if a message has not yet been published but a HS relationship setup when it does get published.

Three tables exist on this tab. Normally one starts with the first one where they identify the Device or the subscription topic that they desire to observe, edit or delete. Note that use of the Ref button on the Association tab row will automatically fill the tables on the Edit tab and will bring up a new window that will contain the Edit table associated with the device.

The second table is used to edit non-mcsMQTT HS Devices that are associated with MQTT topics. The orientation is for publishing something when the HS Device changes as reflected in the pink coloration.

The third table is used to edit mcsMQTT HS Devices that were previously defined based upon a topic subscription. The orientation is for changes in subscribed topic payloads as reflected in the green coloration.

Changes in the tables will be retained in the mcsMQTT.db database. The Delete button will remove the records from the database.

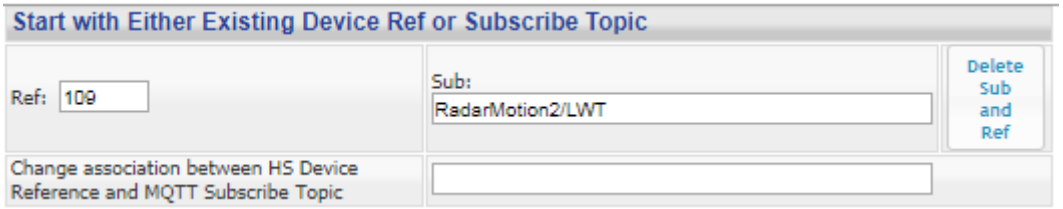


Figure 198Edit Tab Device Identification

18.2.1 Start Reference

18.2.1.1 Ref:

```
"Enter existing HS unique reference number (DeviceRef)"
```

18.2.1.2 Sub:

```
"Enter subscription topic.  Use : for JSON keys."
```

18.2.1.3 Delete Sub and Ref

```
"Press to remove the subscription topic from database and referenced device from
HS"
```

18.2.1.4 Change association between HS Device Reference and MQTT Subscribe Topic

```
"Change the association between subscribe topic <…> and <…>. Enter either a new
subscribe Topic or a new non-Plug-in Device Reference."
```

18.2.2 Publish (Outbound)

The key to non-mcsMQTT Devices is the HS Device Reference number. This number cannot be edited, but is used to identify the Device being edited. This is the same as the Ref in the first table on this tab.

When this Device changes in Value or String content, or appears in the Log then mcsMQTT will publish. By default, the value change event will be used, but string change or log can be selected. When Value Change is selected then publish will only occur if the DeviceValue changes. If Log is selected then it will occur on any update of the Device and the publish will be the DeviceValue. If multiple are selected then multiple messages will be published. The publish topic can be static text (e.g., "MyHS/AtticSensor/Temperature/Centigrade") that follows MQTT topic protocol. The topic can also be specified using a combination of static text and replacement variables. An example may be "$$COMPUTER:/$$ROOM:/$$NAME/Centigrade". When substitution variables are used then the substitution occurs before the topic is published so if the computer network name, Device Room property and Device Name property are MyHS, AtticSensor, and Temperature respectively then the first example will be published in both cases.

The Quality Of Service (QOS) and Retain flag defaults are setup on the Client Tab, but change changed for particular messages. Normally a common QOS and Retain policy is employed and no editing is needed.

It is possible that the existing non-mcsMQTT HS Device will be controlled by a MQTT topic. In this case the subscription topic used for this control is entered for the subscribe topic. The payload in this topic will need to comply with the CAPI (Control Application Program Interface) definition that exists when the non-mcsMQTT Device was created. Normally text such as "On" and "Off" will be payloads for these cases, but it is totally dependent on the HS Device expectations.

Two HS Device property fields are provided to facilitate editing of properties that are not normally available via a browser. One is the grouping parent ref which is the parent Device under which Features are organized. In essence these are the Association and Relatioship properties.

If a Parent Device is being edited then the grouping text box can be used to list all the Child Features that are to be grouped under this Parent Device. If a Child Feature is being edited then the grouping text box is used to assign this Child Feature to a different Parent Device.

The second is the Interface property. Changing this to the name of the plugin will change the ownership of the device. This mechanism can be used for mcsMQTT to take ownership of a Virtual Device or a Device created by another plugin.

Edit of Non-Plugin HS Device to Subscribe (Inbound) and Publish (Outbound) on Event Change

Existing HS Device Reference	4427
Grouping Parent Ref	4426
Owning Interface	
HS Event Trigger	Value Change Event ✓ Log ☐ String Change Event ☐
MQTT Publish (status) Topic	Test/VD
MQTT Publish Payload Template	
MQTT Publish To Sign	Normal Publish ● Publish to Messaging Sign ○
URI Encode Payload	Send unencoded ● Encode with URI encoding such as %20 for space ○ Replace special characters with underscore ○
MQTT Publish QOS	At Most ● At Least ○ Exactly ○
MQTT Publish Retain	Do not retain ● Retain at broker ○
MQTT Subscribe (control) Topic	Test/cmnd/VD
Payload RegEx Match Pattern	
Payload RegEx Replace Pattern	
Payload RegEx Operation	Replace Match Pattern With Replace Pattern ● Extract Match Pattern ○
Expression	

Figure 199 Edit Tab Publish

18.2.2.1 Existing HS Device Reference

"Enter existing HS unique reference number (DeviceRef) that is not a mcsMQTT Device"

18.2.2.2 Grouping Parent Ref

""Enter grouping parent or comma-separated child references, negative value To create New parent device, Or blank to remove from grouping."

18.2.2.3 Owning Interface

"Name of the plugin that has ownership. This is the interface property of the device.")

18.2.2.4 HS Event Trigger

"Check to publish on DeviceValue change"
"Check to publish on DeviceString change"
"Check to use log event. This is used if Device update without a value change is needed."

18.2.2.5 MQTT Publish (status) Topic

"Use semicolon to separate CSV topics. Substitutions are $$NAME:,$$ROOM:,$$FLOOR:,$$COMPUTER:,$$INTERFACE:,$$INSTANCE:,$$TYPE:,$$ADDRESS:,$$CODE:,$$REF: and those in the payload list"

18.2.2.6 MQTT Publish Payload Template

"Substitutions are $$VALUE:,$$VALUE_EUROPE:,$$PREVIOUS:,$$PUBLISHED:,$$STRING:,$$STATUS:,$$PAYLOAD:,$$PAYLOAD_EUROPE,$$TOPIC:,$$DATE:,$$TIME:,$$LASTDATE:,$$DEVICETYPE:,$$DEVICESUBTYPE: and those in status Topic list"

18.2.2.7 URI Encode Payload

"Use URI encoding convention to encode special characters such as space, colon, etc."

18.2.2.8 MQTT Publish to Sign

"Publish to Messaging Sign uses JSON encoding and API specific for sign. The plug-in will format to this API when this radio is selected."

18.2.2.9 MQTT Publish QOS

"Select among the three levels of Quality Of Service"

18.2.2.10 MQTT Publish Retain

"Flag to send to broker to retain or not retain messages per MQTT protocol"

18.2.2.11 MQTT Publish Broker

"Select the broker to which the message will be published"

18.2.2.12 MQTT Subscribe (control) Topic

"Enter MQTT Topic that when received will command HS Device change"

18.2.3 Subscription (Inbound)

Subscriptions are keyed to the inbound MQTT topic. The first row of the third table and the first Edit tab table will have the same topic populated. The association with a HS Device is normally done on the Association tab.

Regular expression editing and publish topic editing is the same as on the Association tab row. The QOS and Retain policies are the same as with other outbound publication.

The status and shown in HS Devices is based upon the subscribed topic's payload. If the payload is a number, then that number will be shown for status. From HS Device Management a prefix or suffix can be added if desired. HS normally appends a "Dim" suffix for most numbers.

For textual payload three options exist. For common two-state text of Off/On, Closed/Open, False/True, Offline/Online, Inactive/Active, and Disarmed/Armed a button and graphic will be setup in Value-Status-Pairs (VSP) of the HS Device. For other payloads that have been received with a single text string then a button will also be setup with the second state being "Not xxx" where xxx is the received text. For other payloads the VSP will not be used and the payload text will be placed in the Device String.

All of these default/automatic decisions can be edited by selecting a different Control/Status UI radio button. The List type will create multiple value-status pairs. The Color Picker and ColorXY is used only for published topics where a color selector is the desired control UI.

Different Payload states are observed and collected as Value-Status-Pairs until the Topic has been identified as being a number via the Control/Status UI selection or until the max VSP text box entry value has been reached. Up to twelve states are collected by default.

The VSPs that have been observed in payloads can be edited by adding additional pairs or the value-status relationship of the pair can be edited with the VSP box. There are four fields available. The first is the text received in the MQTT Payload. The second is a number to uniquely associate the Payload text to a value to be used in HS Device. The third and fourth are optional and if excluded will take on the Payload text. The third is the text to be put on HS Device/Feature control. The fourth is the text to show in HS for the status.

Four options exist to edit the VSP.

The first establisheds a Payload text to unique value relationship such as "label=number". The label is the payload text received via MQTT. Number is the numeric pair assigned to the label.

The second does the same as the first, but also allows user to specify a different control text to be shown on the control button or selector of the HS Device/Feature. This is done in the format "label=number;control". An example is "87WRTX=3;KitchenDoor" where 87WRTX is the MQTT payload and KitchenDoor is what HS shows for the control. Figure 200 is another example where the received Payload is upper case, but the Control will have mixed case spelling.

The third is an extension of the second where a different status text is used in HS than the text received in the Payload. The format is label=number;control;status". An example is "On=1;Open;Opened" where "Open" shows on the control, "Opened" is shown for status when the received MQTT Payload is "On".

The fourth removes a label where the syntax is to just enter the label. For example, "On" to remove the VSP for the received Payload of "On"

The last redefines the entire VSP list with assignment of increasing numbers starting at zero. The syntax in this case is a set of comma-separated labels such as "off, on, unknown" to assign off=0, on=1, unknown=2. The typical edit format that maps the Payload text "OFF" to the HS Device VSP of "Off" for value 0 is shown in Figure 200.

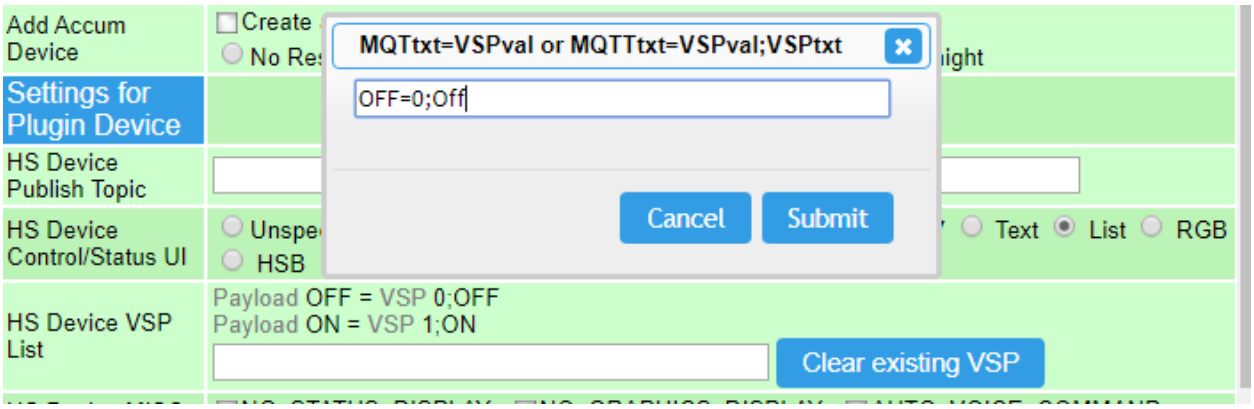


Figure 200 Typical VSP Edit Syntax

HS Devices have a MISC property that affects how the control and status of the device operates. Normally the radio button selections for control/status will set these to provide the desired operation. If a change is needed then they can be made here.

Normally subscription associations create a mcsMQTT HS Device. This can be modified to associate the subscription with a non-mcsMQTT HS Device and if it is done then the mcsMQTT device will be deleted. This relationship between subscription and non-mcsMQTT device is the same as can be done from the second table on this tab.

HS maintains a database to record energy utilization. It will accept the Watt consumption or production over sensed periods of time. To utilize this database the HS Energy Database row of the Edit tab is used. The preferred source is a total energy sensor source. The plug-in will handle a daily energy sensor as well, but energy consumed between the last sample of prior day and midnight may introduce minor variance, depending upon the frequency of the sensor's sampling and reporting.

The Control/Status UI selection determines how mcsMQTT processes the payload and how information is made viewable. Text is stored in DeviceString. Button, Toggle and List will create Value Status Pairs (VSP) where the status is rendered as text while DeviceValue is used to identify each of the text enumerations. List can grow as new payload text is received. Button text is static once associated with HS Device Feature.

Various forms of color space processing are provided. These are Color Picker, RGB, RGBW, HSB and color XY. The general strategy is to convert from the client's color space to the HS RRGGBB color space with the numeric value stored in DeviceValue. ColorXY is for a client that uses XY color space. Zigbee often uses it. JSON is normally used. RGBW is expecting comma-separated decimal values for R, G, B and W. HSB will typically be JSON values for each parameter. Because of all the different client's representation of color space there are many special-case situaitons.

Numbers are shown as text boxes or as sliders with Number, Number Change, and Slider selections.

Jpg File will store binary data from payload. If base64 encoded, then it will be converted to raw binary before storage into the jpg file.

CSV is expecting a comma separated list of numbers and each will be handled as separate HS Features if associated.

Sign contains the formatting for the LED Matrix Sign described in Section 21.19 .

Ramp is a slider to which a change filter is applied to control the rate at which a DeviceValue will change.

Settings For Plugin Device

HS Device Publish Topic	
HS Device Control/Status UI	Unspecified ○ Button ○ Number ● NumberChange ○ Slider ○ CSV ○ Text ○ List ○ RGB ○ RGBW ○ HSB ○ ColorXY ○ Sign ○ Ramp ○ Toggle ○ jpg File ○
HS Device API Type and SubType	Generic Battery
HS Device Location	Loc2 (Floor) DS18B20 Loc (Room) DS18B20 Name DS18B20:DS18B20-1:
HS Device VSP List	

HS Device MISC Properties	NO_STATUS_DISPLAY ☐ NO_GRAPHICS_DISPLAY ☐ AUTO_VOICE_COMMAND ☐ SET_DOES_NOT_CHANGE_LAST_CHANGE ☐ SHOW_VALUES ✓ STATUS_ONLY ☐ IS_LIGHT ☐ IS_DIMMABLE ☐
Grouping Device Ref	450
Publish Payload Template	
URI Encode Payload	Send unencoded ● Encode with URI encoding such as %20 for space ○ Replace special characters with underscore ○
Publish QOS	At Most ● At Least ○ Exactly ○
Publish Retain Flag	Do not retain ● Retain at broker ○
HS Energy Database	Do not save ● Save as Watts ○
Tag Field	

Settings For Non-Plugin Device

Control non-Plugin HS Device	HS Device Reference Number

Note additional customization Of buttton text, display graphics, button/number relationships etc. Is done via HS Device Management Page by clicking On the Device Name link And Using the Status Graphic And Configuration tabs

Figure 201 Edit Tab Subscribe

18.2.3.1 MQTT Subscribe Topic

"Elevate to treat as part of topic to establish uniqueness of message. The value contained in the payload will be added as a level to topic and all members of the JSON group will be children of it."

18.2.3.2 Payload RegEx Match Pattern

"Enter regular expression match pattern to be used on received payload"

18.2.3.3 Payload RegEx Replace Pattern

"Enter regular expression replace pattern to be used on received MQTT Payload"

18.2.3.4 Payload RegEx Operation

"Extract or Replace of Regular Expression on received MQTT Payload before storing in HS device"

18.2.3.5 Subsample Min Seconds

"Subsample to reduce the processing burden of high-rate data. Number specified is the minimum number of seconds since last message before accepting a new one from the same topic"

18.2.3.6 Low Pass Filter

"Smooth payload value using low pass filter with sensitivity ranging from 0.00 for payload ignored to 1.00 for no smoothing"

18.2.3.7 Expression

"Numerically transform payload (e.g. $$PAYLOAD: * 2.54 to convert centimeter to inches). Use replacement variables, math operators and functions."

18.2.3.8 Add Rate or Extra Device

"Select between no rate/extra device, creation of a new device, or use an existing device."

“Use and existing HS device to allow a different expression to be used when updating this additioinal HS device.”

"Set rate sensitivity with value between 0.00 for no rate change to 1.00 for change based only on last two samples"

“Select units for rate of change”

18.2.3.9 Add Accum Device

"A device that sums the payload can be created independent of the payload device"

"Select between no reset, accumulate totals since midnight, or accumulate change since midnight"

18.2.3.10 Payload Store

"Normally only Control/Status of Text will store payload in DeviceString. Others in DeviceValue. This can be overridden for non-Text types with this setting so DeviceValue is not updated, but DeviceString will contain the Payload. For the case of null payloads, the null value can be stored in the DeviceString or it can be used to trigger sending current Device Status."

18.2.3.11 HS Device Publish Topic

"Substitutions are $$NAME:,$$ROOM:,$$FLOOR:,$$COMPUTER:,$$INTERFACE:,$$INSTANCE:,$$TYPE:,$$ADDRESS:,$$CODE:,$$REF: and those in the pub payload list"

18.2.3.12 HS Device Control/Status UI

"Identify how UI control will be shown on HS Device. Button and List are for VSP mappings that update DeviceValue and show DeviceStatus text. Number, NumberChange and CSV are for DeviceValue updates. RGB, HSB and ColorXY are for color wheel that use DeviceValue. Text is for DeviceString updates. Sign is for data send to messaging sign. Jpg File is to store binary data from Payload into file of type .jpg."

18.2.3.13 Device Type API

"Select the Device API type for the Device or Feature"

"Select the Device API subtype for the Device or Feature"

18.2.3.14 HS Device Location

"Loc2 (Floor) where this HS Device placed.”

"Loc (Room) where this HS Device placed.”

“Name to be used for the HS device."

18.2.3.15 HS Device VSP List

"Value status pairs (VSPs) are the values, controls and statuses visible on the device's Status Graphics tab in the Device Management page of HS. The syntax for entry is PayloadText=Number;Control;Status where PayloadText comes from MQTT Message and Number, Control and Status are what is used in HS. PayloadText will be used for HS Control and Status unless other text is provided.
Five edit formats are supported.
1: text=number (e.g. On=1) syntax to specify or overwrite a specific entry.
2. text=number;control (e.g. Closed=0;Close).
3. text=number;control;status (e.g. off=0;Close;Closed)
4: text alone (On) to delete it.
5: comma-separated list (e.g. Off,On) to define full VSP list numbered starting at 0."

"Click to clear previousaly defined VSP definitions"

18.2.3.16 HS Device MISC Properties

"Check to set MISC property"

18.2.3.17 Grouping Parent Ref

"Enter grouping parent reference, negative value to create new parent device, or blank to remove from grouping"
"Click to create a new mcsMQTT device that will serve as a parent for this and possibly other devices in group"

18.2.3.18 Publish Payload Template

"$$label:, $$status:, $$vsp: (default if blank) for text and $$value: for number are typical for CAPI, view Publish Topic & Payload for other substitutions"

18.2.3.19 Publish QOS

"Select among the three levels of Quality Of Service"

18.2.3.20 Publish Retain Flag

"Flag to send to broker to retain or not retain messages per MQTT protocol"

18.2.3.21 HS Energy Database

"Enter energy sensor that provides readings in Watts. Plug-in will compute the delta usage since last sensor report to store in HS Energy database."

18.2.3.22 Tag Field

"Free form input field to record notes or othre information that can then be later retrieved using $$TAG: replacement variable"

18.2.3.23 Control non-Plug-in HS Device

"If associating subscription topic with an existing non-Plug-in device then enter the existing Device Ref otherwise leave blank"

18.3 MQTT Page Client Tab

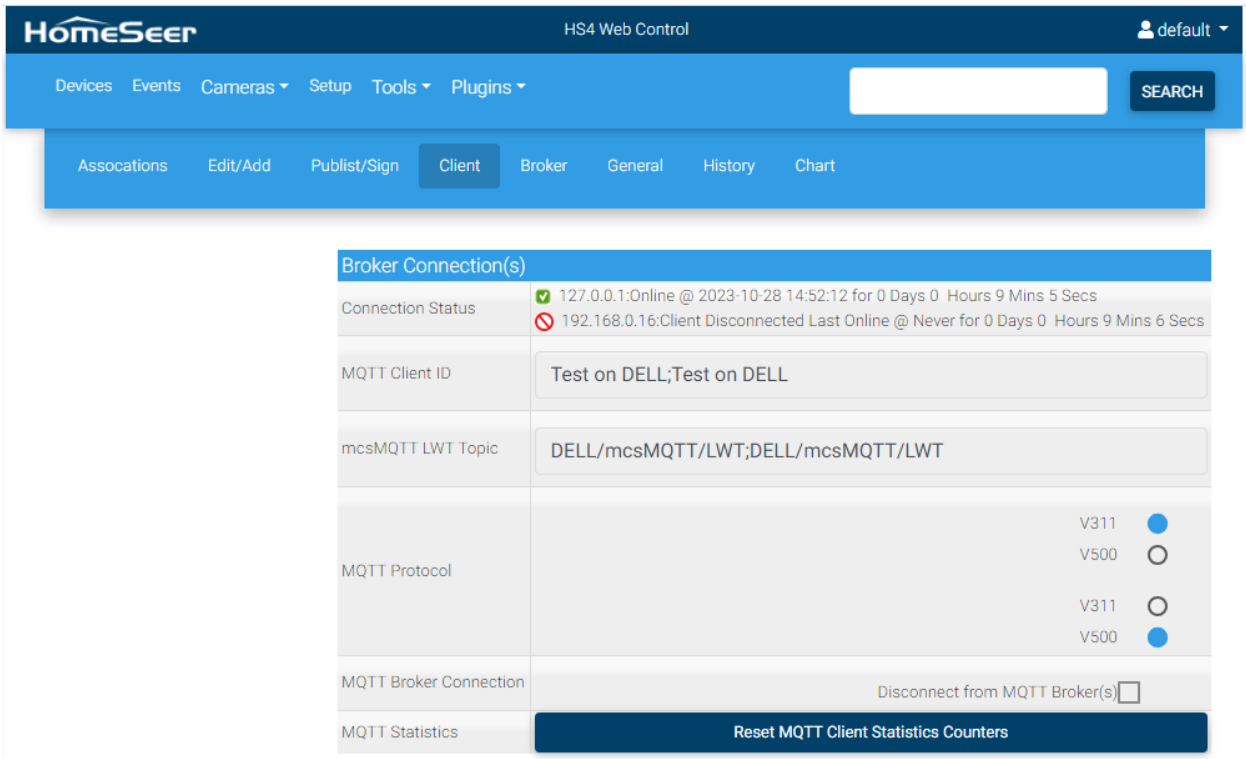


Figure 202 MQTT Broker Connection Settings

The dominant MQTT protocol in use is version 3.1.1 while the updated standard 5.0 has been available for several years. Clients using 5.0 can interact with clients that uses 3.1.1, but legacy clients will not be able to communicate with clients using 5.0.

The Broker can accept an anonymous connection or it can be with a username/password protection. Additional security is with encryption with certificates identified used to decode the encryption. mcsMQTT is a client and provides a default ID of mcsMQTT on <hostname>, but this can be edited if desired.

18.3.1.1 MQTT Client ID

"mcsMQTT Client ID to uniquely identify mcsMQTT as the source of message. Separate multple with semicolon."

18.3.1.2 mcsMQTT LWT Topic

"Last Will and Testament topic given to broker by mcsMQTT. Broker publishes the topic when mcsMQTT appears to have disconnected. Payload will be 'Offline'."

18.3.1.3 MQTT Protocol

“Select MQTT communication protocol version to be used by mcsMQTT for Broker.”

18.3.1.4 MQTT Broker Connection

"Check to disconnect from MQTT Broker and no longer subscribe"

18.3.1.5 Reset MQTT Statistics

"Use button to reinitialize receive statistics to facilitate specific time measurements"

18.3.2 Inbound (Subscription) Management

MQTT has bidirectional communication. Messages published by others can be subscribed by mcsMQTT and is considered inbound from mcsMQTT's perspective. Normally mcsMQTT will listen to all messages delivered by the broker. This makes it easier for user selection of the topics of interest. If this feature is not desired for performance or simplicity reasons then the discovery of published topic can be turned off. Similarly, if a stable configuration exists and there is no desire to receive any new topics then new topic discovery can be turned off.

When only specific topics are desired then the topic discovery template is defined using a comma between each pattern is used. For example, if subscribing to "HS" or "HS3" topics then the template would be specified as "HS/#,HS3/#".

Normally mcsMQTT will not subscribe to topics that it publishes. If there is a need to receive published messages then it can be turned on.

mcsMQTT will treat payloads in one of three manners. One is to place it into a HS Device String or Device Value depending upon it being numeric. If the payload is JSON-encoded then it can expand to topic to include the JSON keys and the Device Value or Device Sting will be populated with only that one JSON element value. The third option is to create a parent HS Device and then children devices with the decoded JSON items in each of the children.

IOT devices many be removed from one's location or there may be some experimenting that resulted in Topics being discovered that will never be used again. To remove the clutter of obsolete Topics a provision exists to remove them from the database. This can be done on individual Topics or on a group identified with wildcard characters. See Section 4.1.26. Also note the obsolete unassociated on shutdown on the General Tab that will remove all unassocited topics that accumulated during the session.

Another approach to excluding topics is with the Reject mechanism. This can be done on a topic-by-topic basis on the Association tab or can be done based upon the reject Topic template on the Client tab. The template to prevent discovery is processed at the time of message reception and considers only the Topic. Its use is not as desirable as changing discovery to only update associated messages, but will consume fewer resources than the individual reject checkboxes.

In the singular case and reject templates will only hide the Topics from the Association table. the messages are processed, but just not visible on the Association tab. The Reject done using a Reject Topic template will firewall the message and no record of the message will be retained.

There are some MQTT clients that send messages with the retained flag set. This means that every time mcsMQTT starts it will receive these retained messages from the Broker. If these Topics are no longer of interest, then they can be removed from the MQTT Broker's database using a wildcard template (i.e., with +, * and #) or individually specified. When using this capability, the message will be removed from the MQTT Broker database andif it has not bee associated with a HS Device then also removed from the mcsMQTT database. If the Topic is again sent through the Broker, then it will be restored to the databases.

On lower power processors it may be desirable to manage the processing of inbound MQTT traffic. This is especially significant following startup when the broker may have a large number of topics that it will burst to mcsMQTT. Received messages are placed in a queue and the queue is worked-off based upon the depth and delay parameters. Normally no special provisions are needed to manage the queue.

18.3.2.1 Topic Discovery

"Discover listens for all (#) topics otherwise subscribe to only Associated topics. Separate each topic by comma. + and # wildcard symbols are honored.

18.3.2.2 Inhibit Topic Discovery

"Prevent new topics from being discovered"

18.3.2.3 Subsample Topic Templates

"Enter set of semicolon-separated topic templates to be subsampled with interval specified below. MQTT wildcard symbols +, * and # can be used to identify Topic range. For example, 'test/topic/#' will reject all Topics that start with 'test/topic'. Reject template applies only to Topic and does not consider anything in the Payload."

18.3.2.4 Subsample Min Seconds

"Subsample to reduce the processing burden of high-rate data. Applies only to topics that match the subsample template above. Number specified is the minimum number of seconds since last message before accepting a new one from the same topic."

18.3.2.5 Enable Auto Device Creation

"Specific topics including shellies/#, wled/#, and other devices will automatically create HS devices when received."

"Specific topics advertised by Tasmota.Discovery or Homeassistant/../config topic will automatically create HS devices when received."

18.3.2.6 Wildcard Plugin Auto Associate Template

"Enter topic template for auto-association of plugin devices. When not blank and a topic that matches template is received then HS device(s) will be automatically created with publish topic set to the Default Topic Template and Default Payload Template (e.g. +/Tasmota/# for any topic that is Tasmota in the second position)."

18.3.2.7 Wildcard Non-Plug-in Control Template

"Enter template for auto-association of non-plug-in devices with topics using $$REF: for device reference (e.g. HS/$$REF:/cmnd)"

"Must define an subscribe wildcard template and publish topic template before auto association can be done"

18.3.2.8 Default HS Device Location

" If default is selected then all new associations will be placed in the same HS Loc2 (Floor) and Loc (Room), otherwise the location will be based upon the MQTT Topic."

"Loc2 (Floor) name where all new Associations will be placed"

"Loc (Room) name where all new Associations will be placed"

18.3.2.9 Default Name Property

"Selection affects the update of the Last Change property. Options are to always change it when any value Is written to a device or to only change it if the written value changes."

18.3.2.10 Default Misc Property

"Selection affects the update of the Last Change property. Options are to always change it when any value is written to a device or to only change it if the written value changes."

18.3.2.11 Default HS Parent Device

"If default is selected then parent HS device will be based upon the MQTT topic hiearchy, otherwise the newly created feature will be the child of the specified device reference number."

"Device reference number that will be parent device of newly created feature"

18.3.2.12 Echo

"Identify if need to have transmitted topics to be wrapped around and seen as potential subscription topics"

18.3.2.13 Express Mode

"This setting defines the default setting for the Association tab 'E'xpress column checkbox. Full support or Express. When a Topic is associated then the 'E' checkbox will be set to this default option."

18.3.2.14 Express Mode Features

"If Payload is JSON-encoded then decode into separate HS devices rather than storing payload in DeviceString"

"If Edit tab has been definded payload to be color picker, button, or list then convert payload to DevcieValue rather than storing in DeviceString"

"Regular and arithmetic expressions will be included to transform received payloads before storing in HS Device"

"Ability to trigger an event or generate a script callback based upon a received Topic"

18.3.2.15 Reject Topics

"Enter set of semicolon-separated topic templates to be rejected from subscription. MQTT wildcard symbols + and # can be used to identify Topic range. For example, 'test/topic/#' will reject all Topics that start with 'test/topic'. Reject template applies only to Topic and does not consider anything in the Payload."

18.3.2.16 Receive Queue Depth

"Enter maximum number of received messages to be processed immediately"

"Enter maximum depth of the receive queue"

18.3.2.17 Receive Queue Interval

"Enter number of milliseconds to wait to process received messages when queue is above 'queue depth' threshold"

18.3.3 Outbound (Publish) Management

mcsMQTT will publish topics when HS Device(s) change, as a result of an Event action or from a script. The topic to be sent is user-specified on a case-by-case basis, but if there is a standard format that is being used for publish topics then this format can be specified in a template with substitution variables. These variables will then be filled out at the time the topic is published. The substitution variables available are $$NAME:, $$ROOM, $$FLOOR, $$COMPUTER:, $$INTERFACE:, $$INSTANCE:, $$TYPE:, $$ADDRESS:, $$CODE:, $$REF:, $$VALUE:, $$VALUE_EUROPE:, $$STRING:, $$STATUS:,$$LABEL:, $$VSP:,$$PAYLOAD:,$$TOPIC:, $$DATE:, and $$TIME:.

The same substitution variables are available for published payloads. If not specified then the payload will be Device Value if numeric, Device String if non-numeric or a CAPI value or label if commanding a mcsMQTT Device.

Expressions can be used in the template by encasing the expression in "<<" and ">>". Nesting expressions to two levels is supported. A typical JSON template with an expression is '{"Brightness":<<ROUND($$VALUE*255/100,0)>>}' where the expression produces an integer in range of 0 to 255 based upon an input in the range 0 to 100.

The templates here are setup as the defaults. Individual topics can later be specified that vary from the templates. Templates do not need to be defined, but only exists for those who have a large number of topics to publish and desire to standardize on the structure.

MQTT protocol provides for Quality Of Service and message retention on a topic by topic basis. Just as payloads can have a default template, the same is true for these attributes. They can later be changed on a topic-by-topic basis.

Normally MQTT messages are published only when something changes. They can be setup to periodically publish to assure that subscribed clients contain current information. Note there is a redundancy between QOS/Retain and the periodic status reporting.

For users that have a large number of Devices and infrequent addition to the set then it may be desirable to reduce CPU burden and not enumerate all the devices each time mcsMQTT starts. A button is provided for this alternate mode of enumeration.

Provisions exist to support LED Messaging Sign described in Section 21.19. If information from the HS Log is to shown on the screen, then the Sign Log radio is set to enable it. It is likely that a subset of everything going to the sign is all that is desired. The Regular Expression that is applied to the Log type field is used to select the desired subset.

18.3.3.1 Default Topic Template

```
"Blank for computer/plug-in/location/name or formatted text including substitution
variables"
```

18.3.3.2 Default Payload Template

```
"Blank for Device Value/String or formatted text including substitution variables"
```

18.3.3.3 Default QOS

```
"Select among the three levels of Quality Of Service"
```

18.3.3.4 Default Message Retain

"Flag to send to broker to retain or not retain messages per MQTT protocol"

18.3.3.5 Publish Periodic Status

"Enter number of minutes between periodic status reporting. 0 to publish only on status change."

18.3.3.6 URI Encode Topic

"Spaces are not recommended in topics, but are legal. This option will send the space as ' ' or '%20'. URI encoding applies to all special characters and not just space."

18.3.3.7 Publish HS Device Changes

"Normally MQTT messages are published based upon explicit associaitons between a HS Device and a MQTT publish topic as shown in Associaiton tab. This option removes the need for explicit associations. It allows all Device changes to be published or only non-plugin Device changes. The default topic and payload templates are used to form the Topic and Payload in this case."

18.4 MQTT Page Broker Tab

Each MQTT environment needs a MQTT Broker and this is normally the internal Broker built into mcsMQTT or an external one such as Mosquitto. The Broker can be identified by network name or IP address. The internal Broker is always 127.0.0.1. It normally uses port 1883 or 8883 if secure connection is established.

If the internal Broker is used, then two options are available as to how it should handle the retained messages. At shutdown it can save the retained message information and then restore it at startup. This is the standard method such as is used by Mosquitto Broker. It can also discard them and start fresh to accumulate retained message information on next start. This is the implementation used by Home Assistant.

Note that the MQTT Page, Client Tab, Inbound Management Section provides a means to remove retained messages from any Broker on a wildcard or topic-by-topic basis.

Statistics maintained by the internal Broker are visible for all MQTT clients that are using this Broker such as shown in Figure 203. The client is the reference so the “Send” columns reflect the MQTT messages published by the client. The “Time” column reflects the last time the message transaction occcurrerd. The current day is assumed unless an explicit date is shown in the table. Clients that are no longer connected to this Broker drop off of the table

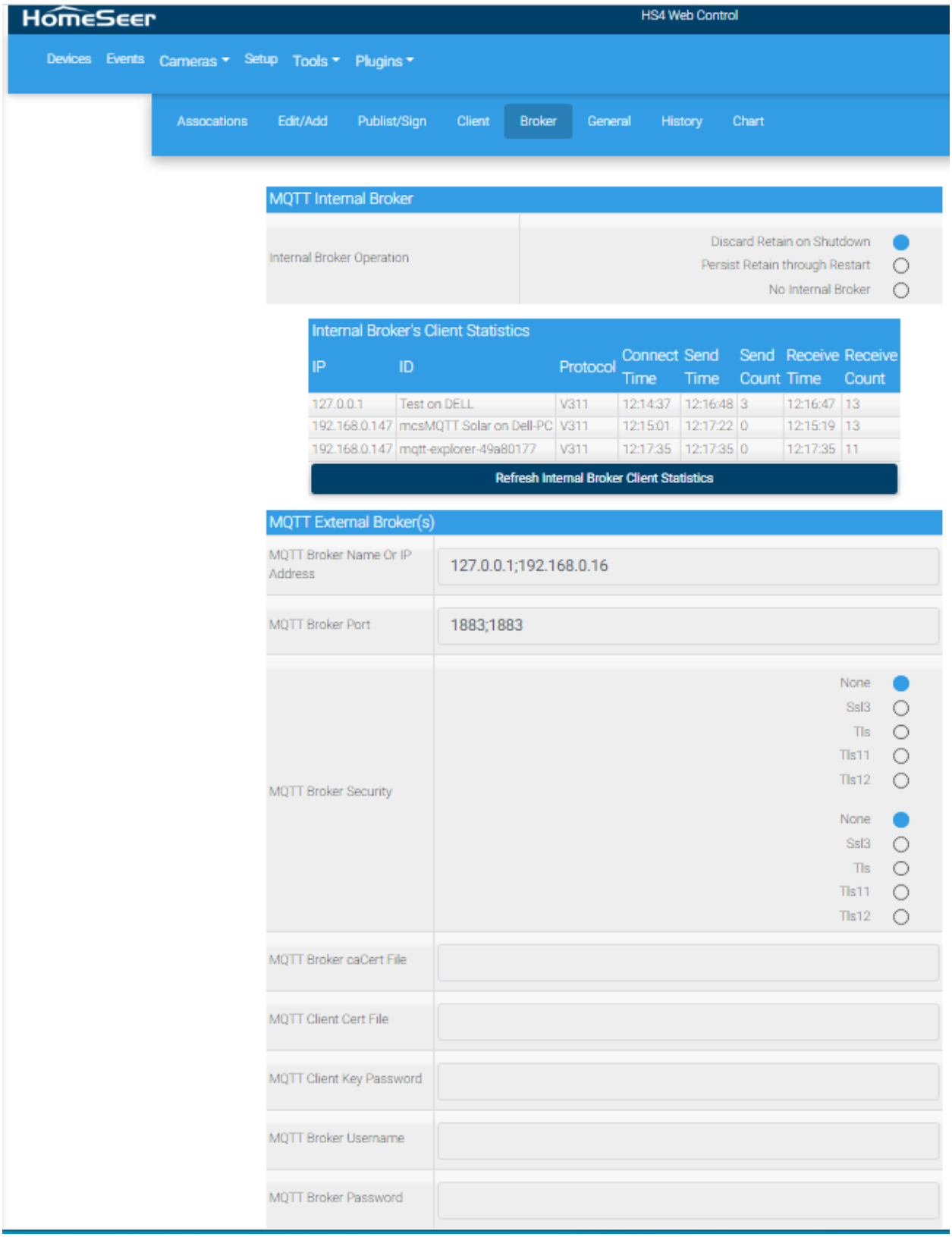


Figure 203 MQTT Broker Tab

18.4.1.1 MQTT Internal Broker Operation

"When using the internal MQTT Broker the option exists to save retained messages on shutdown and then restore them on restart. This is standard operation for MQTT Broker. The other option is to only persist retain messages for those that are received since a restart."

"Use button to get current Internal Broker client list and statistics."

18.4.1.2 MQTT External Broker Name or IP Address

"MQTT Broker (e.g. Mosquitto) network name or IP address. Separate multiple brokers IP address with semicolon. If left blank then mcsMQTT will spawn its own internal MQTT Broker unless “No Internal Broker” is selected."

18.4.1.3 External MQTT Broker Port

"Optional if port other than 1883 is being used by Broker. Separate multple ports with semicolon."

18.4.1.4 External MQTT Broker Security

"Select the communication security used by the Broker"

18.4.1.5 External MQTT Broker caCert File

"Enter full filename path for caCert used by each Broker. Use semicolon to separate broker’s file paths"

18.4.1.6 External MQTT Client Cert File

"Enter full filename path for Client Certificate. Use semicolon to separate broker’s file paths"

18.4.1.7 External MQTT Client Key Password

“Enter password for Client Key Certificate"

18.4.1.8 External MQTT Broker Username

"Optional username expected by MQTT broker is this security implemented by broker. Use semicolon to separate brokers usernames."

18.4.1.9 External MQTT Broker Password

"Optional password expected by MQTT broker is this security implemented by broker. Use semicolon to separate brokers passwords."

18.5 MQTT Page General Tab

The General tab contains settings that are generally one-time edits to configure mcsMQTT to a user's environment. General configuration information is retained in \Config\mcsMQTT.ini. In the mcsMQTT Management header the version of the plug-in that is running is visible.

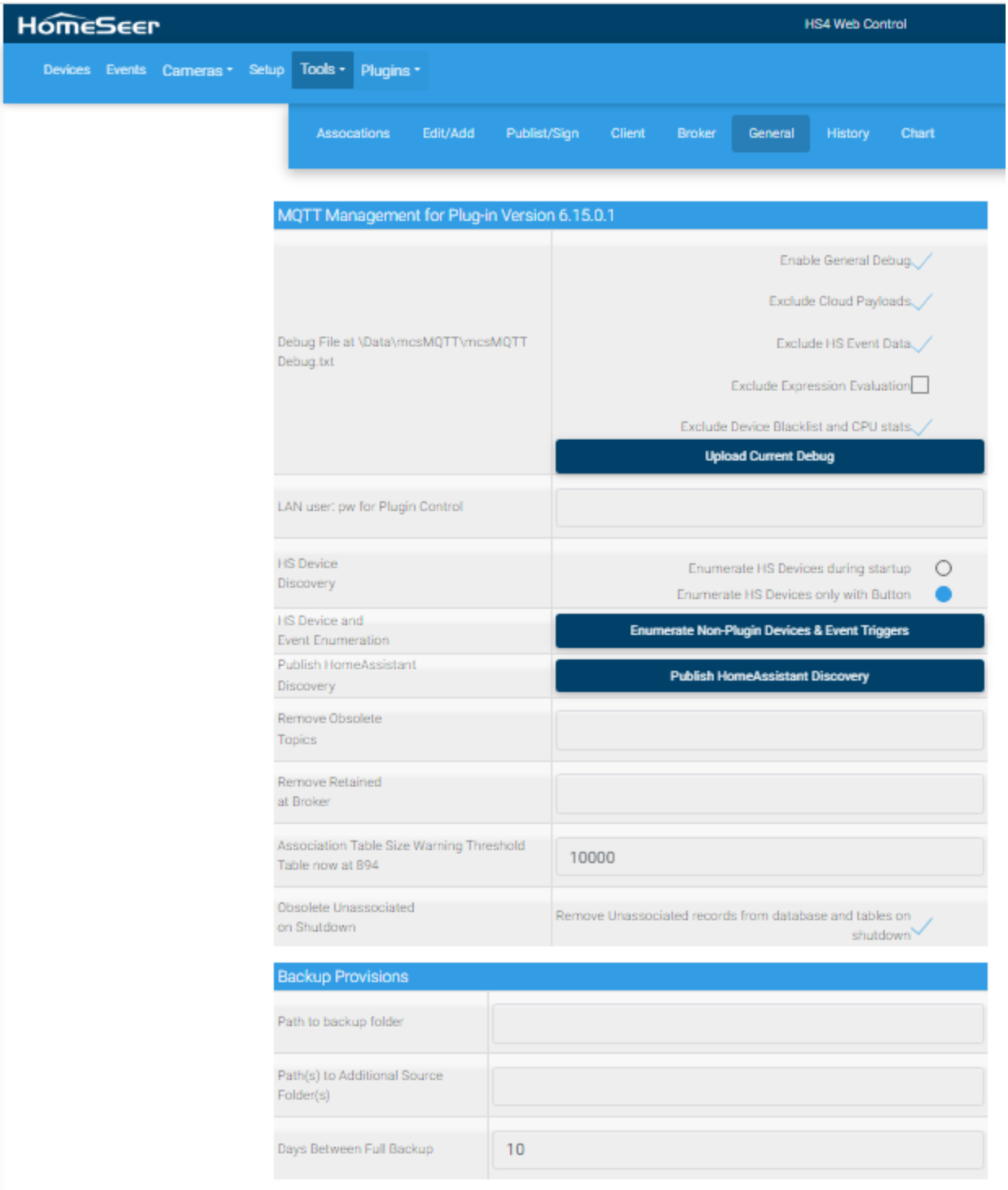


Figure 204 mcsMQTT General Settings

The configuration information that relates to specific MQTT topics is maintained in an SQLite database at \Data\mcsMQTT\mcsMQTT.db. The database is backed up every time the mcsMQTT starts which will normally be each time HomeSeer starts or when an update is installed.

There are three locations where mcsMQTT provides user feedback. One is on the browser with alert boxes when a user entry is not valid. The second is the HS Log when significant errors are detected. The third is in file \Data\mcsMQTT\mcsMQTT_Debug.txt. This third location will always contain some information related to the startup and timing. Additional debug information is included when the General Debug checkbox is used. Some of the debug data is voluminous or chatty so could result in a large debug file. This data can be selectively excluded with checkboxes. In general, they should be checked unless debugging topics specific to their content.

Access to the debug file can be achieved by navigating to the folder where it is stored or it can be uploaded to the Browser's standard upload folder (typically …\Downloads on Windows) in a compressed zip format.

mcsMQTT provides a backup facility to protect its integrity and has extended it to allow a user to do a daily backup of any folder they desire. A backup can be done with a full backup every day or it can be done as a delta backup from the prior day. In either case a full backup is done on the number of days interval specified by the use.

If no backup folder is specified then no backup is one with the assumption that the user has other means to create a backup of the user data. If a backup folder is specified the mcsMQTT will backup in a compressed zip format all files that are necessary to restore user data so mcsMQTT can function from the backup. This includes all the \config, \data for HS and mcsMQTT, and \script subfolders of HS. It does not include other paths such as those used by external databases.

It is also possible to backup other folders using this mcsMQTT feature if the path to the folder is specified. Semicolons are used if multiple additioinal backup paths are desired. Note that if an external database is to be backed up then the facility provided by MySQL, MS SQL Server, or InfluxDB should be used to assure the integrity of backup is maintained.

Backups can be in either of two formats. In both cases a ".zip" file is created for the top-level folder that is being backed-up. In one case all the files are included. In the other case only the files that have changed are backed up. It is a tradeoff between ease of restoration vs. space required to perform a backup. A full backup is done on the interval specified by the user. The incremental backup is done otherwise. To restore an incremental backup, one needs to copy all files changes from the backup since the last full backup. If backup storage space and CPU utilization to perform a backup are not limiting factors then a daily full backup can be done and restoration is simply to restore from the day of last known desired operation.

mcsMQTT collects CPU utilization information for HS and pluigns and these can be associated with HS devices. These devices will also have controls to stop, start and restart HS or plugin. To perform the plugin control, it is necessary to get access to the HS web server. The default setup for HS is to not require login on access to HS from LAN. If a user changes this to require a login, then mcsMQTT will also need to know the login credentials for stop, start, restart control of the plugins. This is provided in as the password for the user 'default' or in fomat 'username:password' for other users.

18.5.1.1 LAN user:pw PW for Plugin Control

"This setting is only needed if LAN access to HS has been modified on HS Setup page to require password on Local login. Access to HS server is needed to perform individual plugin start, stop, restart control. If only password is entered then assumed user account is 'default', otherwise enter both username and password with colon seperator (e.g. name:pw). Enter the password for this user login."

18.5.1.2 HS Device Discovery

"Enumeration makes HS Devices visible on Association table for ease of associating HS Devices with MQTT Topics for publishing"

18.5.1.3 HS Device Enumeration

"Use button to refresh HS devices listed in Association table"

18.5.1.4 Publish HomeAssistant Discovery

"Use button to publish associated non-plugin devices discovery information using HomeAssistant discovery protocol"

18.5.1.5 Obsolete Topics

"Enter a topic to be removed from mcsMQTT database. If Topic is again received it will be restored to database. MQTT wildcard symbols + and # can be used to identify Topic range, but be careful. For example, 'test/topic/#' will remove all Topics that start with 'test/topic'. To remove a Topic with JSON payload then use # at end of Topic."

18.5.1.6 Remove Retained at Broker

"Remove a set of topics from MQTT Broker database that may be retained and if not associated also remove from the mcsMQTT database. If Topic is again received by the broker with the retain flag it will be restored to its database. MQTT wildcard symbols + and # can be used to identify Topic range, but be careful. For example, 'test/topic/#' will remove all Topics that start with 'test/topic'. Individual JSON payload keys cannot be removed. All payload keys in the Topic will be removed."

18.5.1.7 Obsolete Unassociated Topics On Shutdown

"Remove topics that have not been associated to improve startup performance. Where there are a large number of records collected during a session that are of no interest then statup performance will suffer."

18.5.1.8 Receive Topic Warning Threshold

"Enter the max number of Association Table records that will be accumulated before warning received. Oldest unassociated records will be removed to reduce size to 150% of max value.

18.5.1.9 Debug File at \Data\mcsMQTT\mcsMQTT.txt

"Check to generate debug output in \data\mcsMQTT_debug.txt"

"Check to inhibit the data download from Cloud servers being included in the debug"

"Check to inhibit the data showing event callbacks (e.g. devcie value changes) being included in the debug"

"Check to inhibit expression evaluation data in the debug"

"Check to inhibit notification of a HS device being blacklisted from MQTT event evaluation"

"Use button to upload the debug file in zip format. It will go into standard downloads folder.

18.5.1.10 Configuration Backup

"Config, Data (HomeSeer and mcsMQTT), and script HS subfolders will be copied at midnight to a newly created folder in the specified path. Backup can be used to restore to a prior setup.

18.5.1.11 Path(s) to Additional Source Folder(s)

"Full path of other folders that should be included in backup. If multiple then separate each with semicolon."

18.5.1.12 Days Between Full Backup

“Backup will occcur daily and save files that have changed since the prior day. An unconditional backup will occur on the interval of days specified. If one desires a full backup every week then enter 7. If a full backup every day, then enter 1"

18.6 MQTT Page Sign Popup

The Messaging Sign popup illustrated in Figure 205 is shown when the “Configure Sign Parameters” hyperlink is clicked. This hyperlink is located on the Edit Tab/Popup when the Sign has been selected for use.

The Message Sign is a LED Pixel matrix described in Section 21.19.

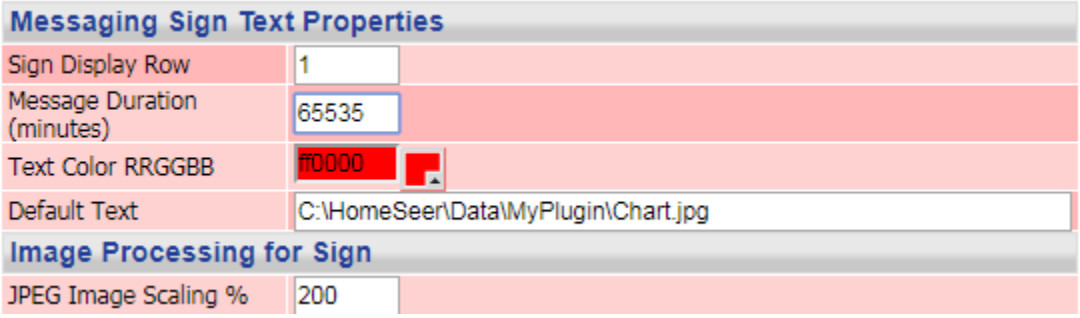


Figure 205 Sign Properties Configuration Popup

18.6.1 Sign Display Row

"Enter row (top=1) of sign where the message will be shown"

18.6.2 Message Duration (minutes)

"Duration of 0 is to remove from sign. Duration of 65535 is until manually removed. Others are number of minutes to show message."

18.6.3 Text Color RRGGBB

"Select color that is default for message. Color can be changed on a character-by-character basis using [RRGGBB] encoding in the message text."

18.6.4 Default Text Payload

"If DeviceString is null then the text entered here will be used for payload when topic is published. This included both Text and Image."

18.6.5 JPEG Image Scaling %

"The JPEG image will be scaled to the percentage of screen size entered. Values between 100 and 200 are reasonable. If the image is too large to fit the sign then automatic reduction will be done."

18.7 MQTT Page Edit Popup

The Edit popup is an alternate means to edit the properties of associations. The popup window is displayed when the Ref button or row sequence number on an Association Tab row is clicked. One table will be shown depending upon if the row is associated with a Plug-in Device or a Non-Plug-in Device.

Start with Either Existing Device Ref or Subscribe Topic

Ref: 212	Sub: GarageDoor/Door	Delete Sub and Ref

Edit Setup or Edit of Subscription (Inbound) to a MQTT Topic

MQTT Subscribe Topic	GarageDoor/Door
Payload RegEx Match Pattern	
Payload RegEx Replace Pattern	
Payload RegEx Operation	◉ Replace Match Pattern with Replace Pattern ○ Extract Match Pattern
Low Pass Filter	Filter sensitivity of 1 (range is 0.00 to 1.00 (most sensitive))
Expression	
Add Rate Device	☐ Create a HS Rate Device with rate sensitivity of 0.75 (Range 0.00 to 1.00) ○ Per Second ○ Per Minute ◉ Per Hour
Add Accum Device	☐ Create a HS Accum Device ○ No Reset ○ Accumulation Since Midnight ◉ Delta Since Midnight
Settings for Plugin Device	
HS Device Publish Topic	GarageDoor/cmnd/Door
HS Device Control/Status UI	○ Unspecified ◉ Button ○ Number ○ Text ○ List ○ ColorPicker
HS Device VSP List	0 CLOSED 1 INDETERMINATE 2 OPEN
HS Device MISC Properties	☐ NO_STATUS_DISPLAY ☐ NO_GRAPHICS_DISPLAY ☐ AUTO_VOICE_COMMAND ☐ SET_DOES_NOT_CHANGE_LAST_CHANGE ☑ SHOW_VALUES ☐ STATUS_ONLY
Publish Payload Template	
Publish QOS	○ At Most ○ At Least ◉ Exactly
Publish Retain Flag	◉ Do not retain ○ Retain at broker
Settings for Non-Plugin Device	
Control non-Plugin HS Device	

Note additional customization of buttton text, display graphics, button/number relationships etc. is done via HS Device Management Page by clicking on the Device Name link and using the Status Graphic and Configuration tabs

Figure 206 Edit Popup Window

18.8 MQTT Page PubList/Sign Tab

The publication lists are a set of Topics and Payloads that are contained in text files of type ".pub" located in folder \Data\mcsMQTT. These files can be manually created and edited outside of mcsMQTT or interactively with the browser on this tab. When using the MQTT Page, Publish/Sign Tab editor, the first four rows are the substitution variables $$1:, $$2:, $$3: and $$4:; and the remaining rows are the Topic=Payload assignments.

If the contents of publist entry contains a "=" then it needs to be escaped with "\=" because mcsMQTT uses the "=" to separate the URL querystring from the data that is sent with POST and PUT methods. An example is shown in Section 12.10.

18.8.1 Publication List

The set of ".pub" files that exist can be selected from the pull-down, lines edited if desired, and then the group published via the Execute Publication List button. Prior to publication substitution variables are applied if they are setup.

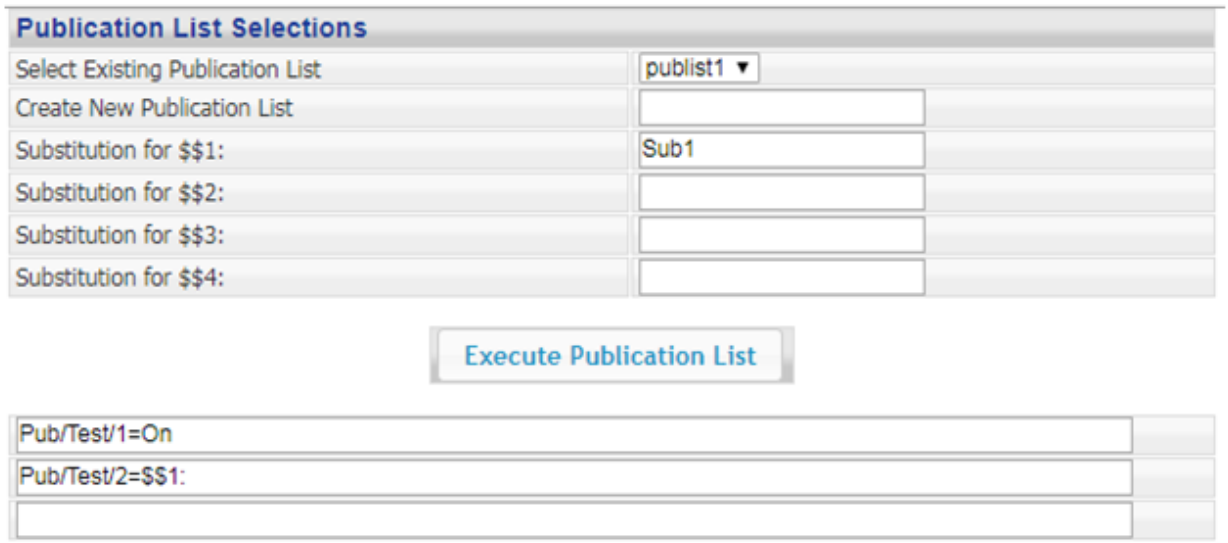


Figure 207 Publication List and Sign Setup Tab (Publist)

18.8.1.1 Select Existing Publication List

"Select file that contains the list of messages to be published"

18.8.1.2 Create New Publication List

"Publication list file contains a list of topics and payloads that can be published on command"

18.8.1.3 Substitution for $$1:, $$2:, $$3:, $$4:

"Topics and payloads can have substitution variables that are replaced prior to publication"

18.8.1.4 Execute Publication List

"Press to publish list of messages"

18.8.1.5 List

"Each row contains one message in format Topic=Payload and can contain $$1:, $$2:, $$3: or $$4: substitution variables"

18.8.2 Sign Use Setup

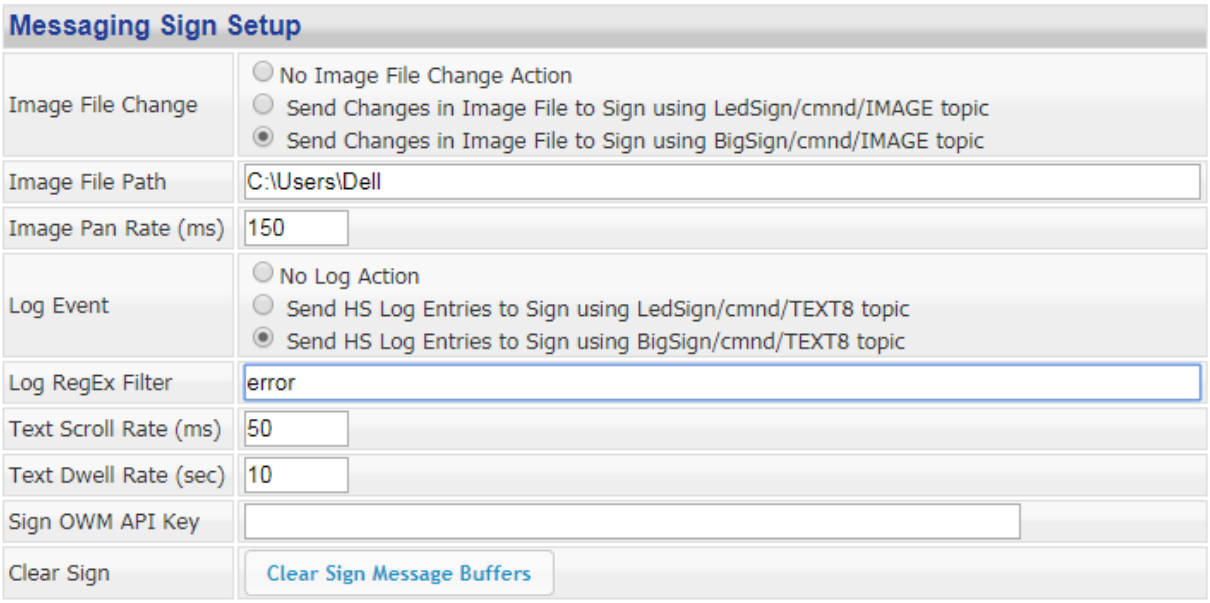


Figure 208 Publication List and Sign Setup Tab (Sign Setup)

18.8.2.1 Image File Change

"jpg images can be sent to the LED Messaging Sign when an identified file path has changed. The file path being monitored is setup on the Client tab."

18.8.2.2 Image File Path

"jpg file or folder in which it is located that will be monitored for change in last modified date and then sent to LED Messaging Sign as a jpg image."

18.8.2.3 Image Pan Rate

"Panning is performed by sign by shifting all pixels up/down/left/right one pixel at a constant rate. The entry is the number of milliseconds to wait between each shift. Resolution is 50 milliseconds."

18.8.2.4 Log Event

"HS Log can be sent to messaging sign using the publish topic for this device. They will retain the color of message in log. Regular Expression filter on Client tab filters the type field of the message that will be sent to sign."

18.8.2.5 Log RegEx Filter

"Regular Expression to select the subset of HS log entries that will be sent to messaging sign."

18.8.2.6 Text Scroll Rate

"Scrolling is performed by sign by shifting all characters of a row left one pixel at a constant rate. The entry is the number of milliseconds to wait between each shift. Resolution is 50 milliseconds."

18.8.2.7 Text Dwell Rate

"When multiple messages are to be shown on a Sign's row each will be shown in round-robin sequence. The duration in seconds that a message will stay on sign before showing the dwell time."

18.8.2.8 OWM API Key

"Open Weather Map free API key is used by Sign to show five-day forecast when no other information is being shown on last row of sign. If not provided then no forecast will be downloaded and shown by Sign."

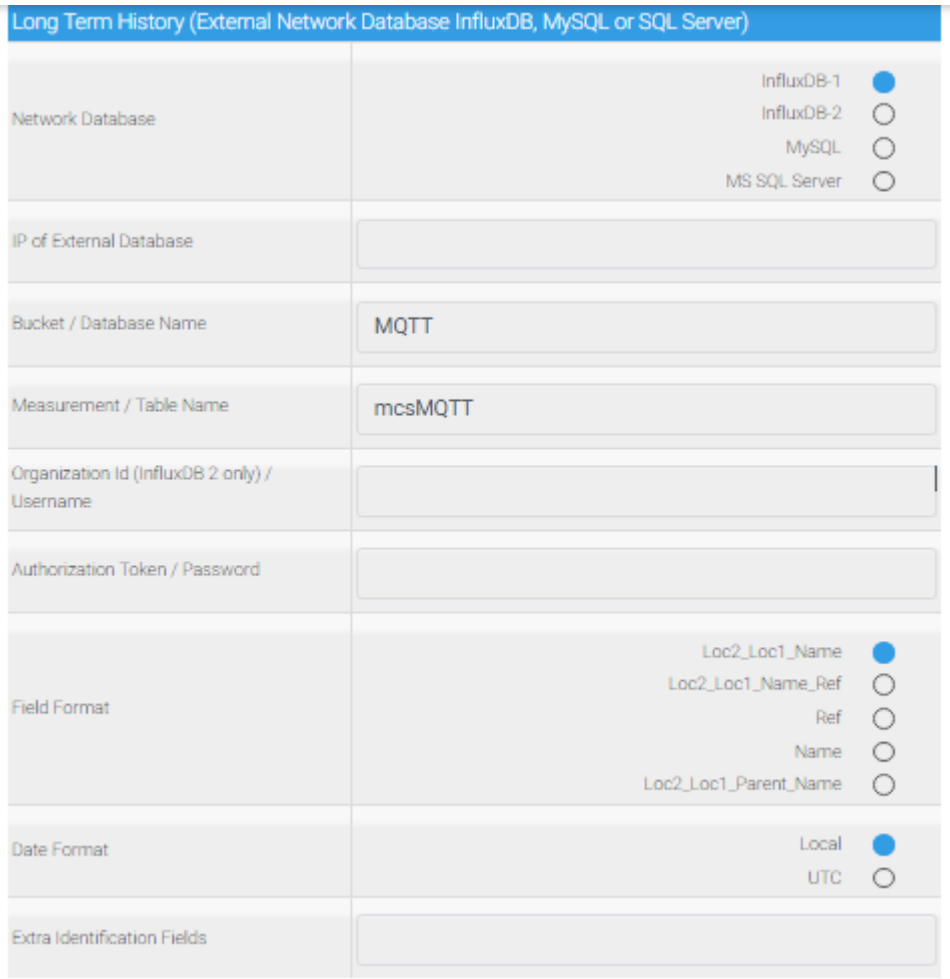
18.8.2.9 Clear Sign

"Button is used to clear all twelve message buffers by setting their duration to 0 in the Sign."

18.9 MQTT Page History Tab

History Tab contains data collection criteria in either the external long term or SQLite short term History databases. It also contains ability to query these databases based upon filter criteria.

18.9.1 Long Term History (InfluxDB, mySQL and MS SQL Server)



18.9.1.1 Network Database

"Select the database to which long term data will be placed."

18.9.1.2 IP of External Database

"Network address of the Database server. Default InfluxDB port is 8086. If a different port is being used then enter it with IP:Port format"

18.9.1.3 Bucket/Database Name

"Name of the bucket or database repository. Default is MQTT."

18.9.1.4 Measurement/Table Name

"Name of the measurement or table repository. Default is mcsMQTT."

18.9.1.5 Organization Id (InfluxDB 2 only) / Username

"Name of the InfluxDB 2.x authorized organization for this repository or Username for MySQL or SQL Server. Leave it blank for InfluxDB Version 1.x."

18.9.1.6 Authorization Token

"Token used to authenticate access to InfluxDB. For Influx 2.x token is obtained for InfluxDB UI. For Influx 1.x format is username:password if database has been setup with username & password. For MySQL and SQl Server use account password if username/passord are required."

18.9.1.7 Field Format

"Select the way the fields will be identified in the measurement or table. The companion field will be obtained from the HS Device Value."

18.9.1.8 Date Format

" Select between local time and universal time for the LastDate field of data being stored."

18.9.1.9 Extra Identifiation Fields

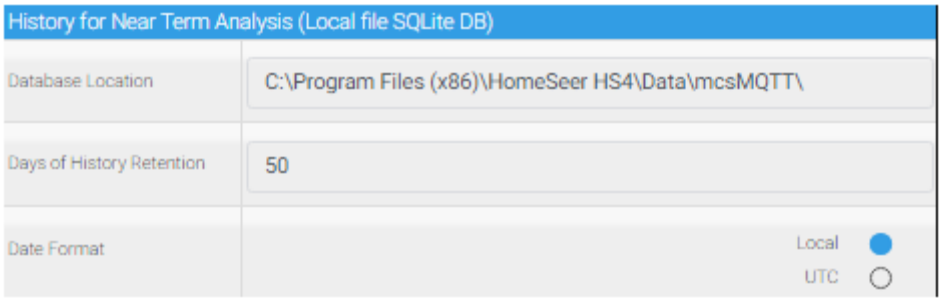
" Additional fields can be specified to store additional data in each record to supplement the Device Value. Specify using string of comma-separated key=value pairs where key is the field name and value is data to be stored. Replacement variables and possibly expressions are used for the field values. e.g. Category=$$TAG:,Derived=<<(ROUND($$DVR:(123):+$$VALUE:)/2,1)>>,Location=$$FLOOR:_$$ROOM:_$$NAME:”

18.9.2 History for Near Term Analysis (SQLite)

Filters are used to narrow down the slice of interest. The first is a date or range of dates. The second is publish vs. subscribe topics. The third is based upon the topic and payload content.

Up to twenty rows of selected history is displayed at a time. Previous/Next buttons and a starting row locator are available to select the start of the twenty.

Sort buttons at the top of each column in the History table are available to sort ascending of descending on the particular column.



All History	
Pub-Sub Message History	Retain history Of published messages ☐ Retain history Of Accepted subscribed messages ☐ Retain history Of Not-Accepted subscribed messages ☐ Retain history Of topics marked With H checkbox On Association tab ✓
HS Device History	Save history Of only devices marked With L or S column checkbox ○ Save history Of all devices in short term SQLite ○ Save history Of all devices in long term InfluxDB, mySQL or SQL Server ●
HS Log	HS Log of changes in topics marked with H checkbox on Association tab ☐

Filter History by Category

Date Range (Start,End) or (SingleDay)	4/1/2018,4/22/2018
Outbound Messages	☑ Include Published Messages
Inbound Messages	☑ Include Received Messages

Filter Message History by Mqtt Topic and JSON Payload Key

T1 IrrigationWater ▾ T2 ▾ T3 ▾ T4 ▾ T5 ▾ T6 ▾

J1 ▾ J2 ▾

Show Selected History

Prev 0 of 238 Next

Message History

	P/S	LastDate	Topic	Payload
0	S	2018-04-21 16:04:11	IrrigationWater/LWT	Online
1	S	2018-04-21 16:07:56	IrrigationWater/STATE	{"Time":"2018-04-22T00:07:34", "Uptime":167 167, "Vcc":3.156, "Wifi":{"AP":1, "SSId":"U", "RSSI":78, "IPAddress":"192.168.0.71", "APMac":"78:8A:20:84:48:1D"}}
2	S	2018-04-21 16:07:56	IrrigationWater/Gallons	167
3	S	2018-04-21 16:17:56	IrrigationWater/STATE	{"Time":"2018-04-22T00:17:34", "Uptime":167 167, "Vcc":3.156, "Wifi":{"AP":1, "SSId":"U", "RSSI":76, "IPAddress":"192.168.0.71", "APMac":"78:8A:20:84:48:1D"}}
4	S	2018-04-21 16:17:56	IrrigationWater/Gallons	167
5	S	2018-04-21 16:19:10	IrrigationWater/LWT	Online
6	S	2018-04-21 16:27:56	IrrigationWater/STATE	{"Time":"2018-04-22T00:27:34", "Uptime":167 167, "Vcc":3.157, "Wifi":{"AP":1, "SSId":"U", "RSSI":74, "IPAddress":"192.168.0.71", "APMac":"78:8A:20:84:48:1D"}}
7	S	2018-04-21 16:27:56	IrrigationWater/Gallons	167
8	S	2018-04-21 16:37:56	IrrigationWater/STATE	{"Time":"2018-04-22T00:37:34", "Uptime":167 167, "Vcc":3.158, "Wifi":{"AP":1, "SSId":"U", "RSSI":80, "IPAddress":"192.168.0.71", "APMac":"78:8A:20:84:48:1D"}}
9	S	2018-04-21 16:37:56	IrrigationWater/Gallons	167

Filter Table by Category and Date

Accepted Associations	☐ Show All Associated Only
Outbound Selections	☑ Include Non-Plugin HS Devices
Inbound Selections	☑ Include Received MQTT Topics
Date Range (Start,End) or (SingleDay)	1/1/2019,1/12/2021

Filter Table by HS Device Categories Clear Filters Rebuild Filters

Loc2 (Floor) ▾ Loc (Room) ▾ Type ▾ Interface ▾

Filter by HS Device

Control Me (8476) ▾

Show Selected SQLite Device History

Show Selected InfluxDB Device History

Filter Association Table by Mqtt Topic and JSON Payload Key Clear Filters Rebuild Filters

T1 ▾ T2 ▾ T3 ▾ T4 ▾ T5 ▾ T6 ▾

J1 ▾ J2 ▾ J3 ▾ J4 ▾ J5 ▾ J6 ▾

Show Selected SQLite History

Prev 0 of Null Next

Device History

	Device	lastdate	Value
0	Control Me	021-01-10T06:20:41.655524503Z	null
1	Control Me	2021-01-10T20:32:39.249829594Z	null

Figure 209 History Tab Provisions for SQLite

The History tab and Chart tab are able to look at MQTT traffic in either tabular or visually in time history. Since much MQTT traffic can exist and the database will tend to become large there are parameters that can be used to restrict the growth of the database.

18.9.3 Near Term History

18.9.3.1 Database Location

```
"SQLite History database is located by default at HS subfolder \data\mcsMQTT.
Alternate locations can be specfied."
```

18.9.3.2 Pub-Sub Message History Retention

```
"History can be viewed via chart and message history can be viewed from History
tab, use 0 if history is not needed"
```

18.9.3.3 Date Format

```
" Select between local time and universal time for the LastDate field of data
being stored."
```

18.9.4 All History

18.9.4.1 Pub-Sub Message History

```
"Check to include all published messages in the history database"

"Check to include all associated messages in the history database"

"Check to include all non-Associated messages in the history database"

"Check to include all History checked messages in the history database"
```

18.9.4.2 HS Device History

```
"Select between saving all devices or saving only those explicitly identified on
Association tab S(hort Term) column"
```

18.9.4.3 HS Log

```
" HS Log of changes in topics marked with H checkbox on Association tab"
```

18.9.5 Filters History by Category, Topic and Payload

18.9.5.1 Date Range

```
"Pick one or two dates that will mark the start and end dates of displayed messages"
```

18.9.5.2 Outbound Messages

```
"Check to include messages published by HS"
```

18.9.5.3 Inbound Messages

```
"Check to include messages received via subscription"
```

18.9.5.4 Filter Message History by Mqtt Topic and JSON Payload Key

```
"Press to clear all Topic/JSON pull-down filters"

"Filter on Topic Segment # "
```

"Filter on JSON Segment Key # "

18.9.5.5 HS Device Selector

"Select Device to be viewed"

18.9.6 History Table Build/Display Control

18.9.6.1 Show Selected SQLite Device History

"Press to show message history from SQLite based upon device and filters that have been setup"

18.9.6.2 Show Selected InfluxDB, MySQL or SQL Server Device History

"Press to show message history from external database based upon device and topic filters that have been setup"

18.9.6.3 Show Selected SQLite Topic History

"Press to show message history based upon topic filters that have been setup"

18.9.6.4 Prev/Next

"History table is shown 20 rows at a time. Enter the starting row"

"Click to display previous 20 rows in History Table"

"Click to display next 20 rows in History Table"

18.9.7 History Table Header

"Sort on Publish or Subscribe"

"Sort on Last Date"

"Sort on Topic"

"Sort on Payload"

18.10 MQTT Page Chart Tab

The Chart tab is used to visualize the time history of one or two items that have been retained in the History. If numeric data is available then it will be shown directly. If textual information is available, it will be mapped into text-number pairs with the number shown in the chart and the text in the legend table below the chart.

Date and Time selectors are used to identify the specific period of interest. This allows ranges down to minutes. The range can be specified with specific dates or as a time span ending in current time.

The primary Y axis is on the left and the optional secondary Y axis can be used to draw a second item. Pull-down selector will contain all the items that have been associated with HS Devices from the History database. This can be expanded to all items via checkbox.

The Y axes on the chart can be auto scaled or specific minimum and maximum can be specified.

A full definition of the chart can be saved by name and later loaded.

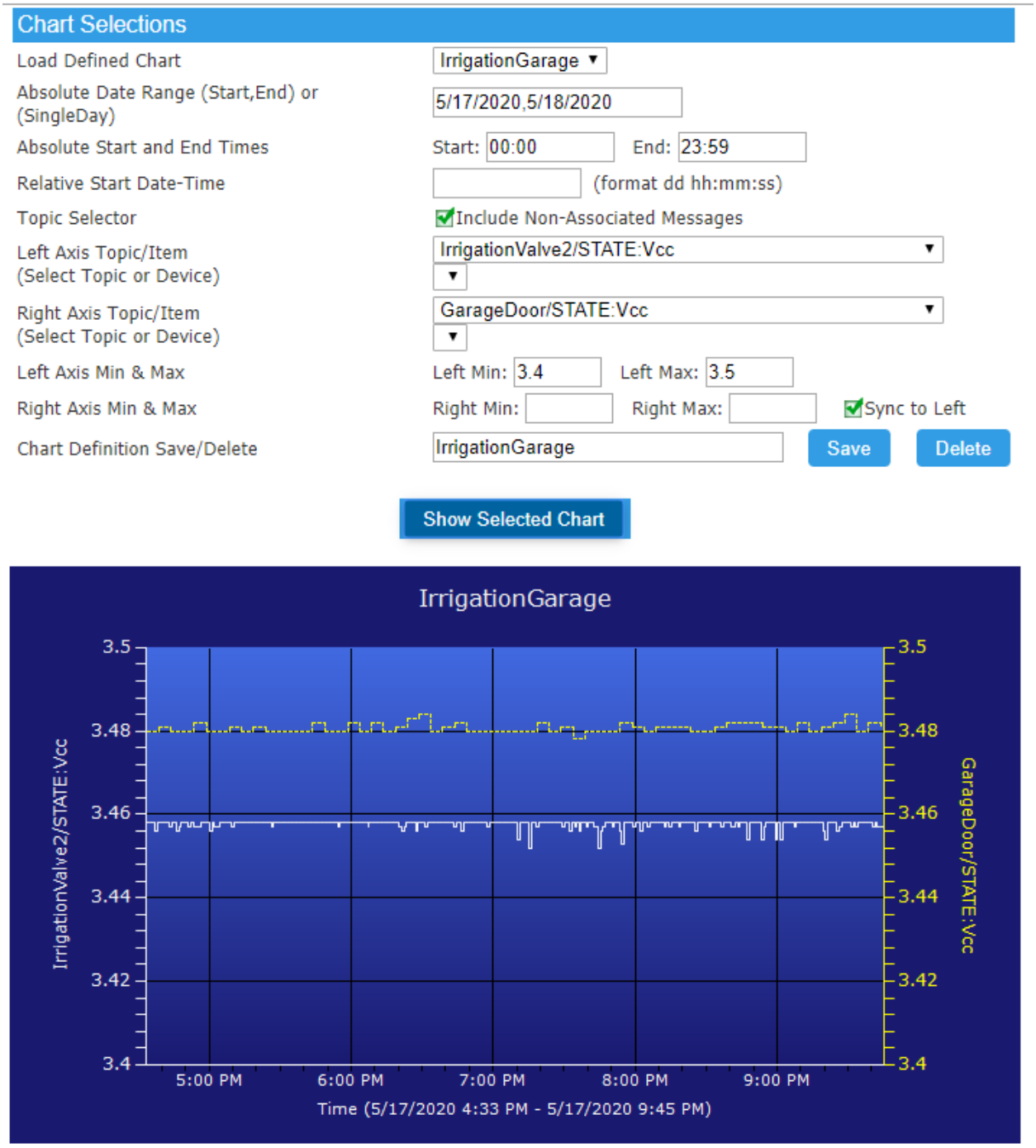


Figure 210 Chart Tab

18.10.1 Chart Definition Load / Save

18.10.1.1 Load Defined Chart

"Select previously saved chart setup to be used"

18.10.1.2 Chart File Format

"Select png vs. jpg file format for chart."

18.10.1.3 SaveChart Definition

"Name of chart that can later be loaded with defined setup parameters"

"Press to save setup parameters for this chart"

"Press to delete the setup parameters for this chart"

18.10.2 Date/Time Range Selection

18.10.2.1 Absolute Date Range (Start,End) or (SingleDay)

"Pick one or two dates that will mark the start and end dates of displayed messages"

18.10.2.2 Absolute Start and End Times

"Pick time of the chart start date"

"Pick time of the chart end date"

18.10.2.3 Relative Start Date-Time

"Number of dd hh:mm:ss in past to start chart. Blank to use absolute start and end date-time."

18.10.3 Chart Selections

18.10.3.1 Load Defined Chart

"Select previously saved chart setup to be used. If All is selected then all defined charts will be shown for ten seconds in round robin sequence."

18.10.3.2 Chart Definition Save / Delete

"Name of chart that can later be loaded with defined setup parameters"

"Save", "Press to save setup parameters for this chart"

"Delete", "Press to delete the setup parameters for this chart"

18.10.4 Topic/Item Selection

18.10.4.1 Include Non-Associated Messages

"Check to include Topics that have not been associated in Topic selectors for charting"

18.10.4.2 Left Axis Topic/Item

"Select Topic to be charted"

"Select Device to be charted"

18.10.4.3 Right Axis Topic/Item

"Select Topic to be charted"

"Select Device to be charted"

18.10.5 Chart Y Axis Scaling

18.10.5.1 Left Axis Min & Max

"Minimum value to be used on Left Y axis. Blank is autoscale."

"Maximum value to be used on Left Y axis. Blank is autoscale."

18.10.5.2 Right Axis Min & Max

"Minimum value to be used on Right Y axis. Blank is autoscale."

"Maximum value to be used on Right Y axis. Blank is autoscale."

"Check to force right Y axis to be scaled same as left scaled axis."

18.10.6 Chart Build/Display Control

18.10.6.1 Show Selected Chart

" Press to show Chart of topics or devices selected."

18.11 BLE Setup Page (HS3)

18.11.1 Page Viewing Options

18.11.1.1 Auto Update of Beacon Location Graphic

"Beacon chart can be rebuilt on demand with pushbutton or automatically when a beacon Zone has changed. A Zone is a single value formed with X and Y locations.

18.11.1.2 Beacon Locations Table Verbosity

"The beacon table will normally be only status, but when programming the beacon parameters more information is needed. When tuning the Kalman filters the RSSI data can also be viewed."

18.11.1.3 Beacon Locations Table Display Hide Override

"All beacons reported by the scanners are shown in the Beacon Locations table excepted those checkboxed in the H column. They can be restored for display by using this checkbox."

18.11.1.4 View Beacon Info from Scanner Selected

"All scanners are expected to report the same beacon information. The scanner selected as Master is the default one for which the Beacon Location data is shown. To view other then select the desired scanner."

18.11.2 Beacon Locations with Last 24 Hours Data

18.11.2.1 H(ide)

"Check to hide beacon row from page unless overridden by show all beacons checkbox."

18.11.2.2 R(emove)

"Click to mark for Remove. Publish to ALL BLE scanners will occur when column header button is pushed."

"Click to publish message to scanners to remove all rows marked"

18.11.2.3 B(lacklist)

"Click to mark for Blacklist. Publish to ALL BLE scanners will occur when column header button is pushed."

"Click to publish message to scanners to blacklist all rows marked"

18.11.2.4 U(nblacklist)

"Click to mark for Unblacklist. Publish to ALL BLE scanners will occur when column header button is pushed."

"Click to publish message to scanners to unblacklist all rows marked"

18.11.2.5 Address

18.11.2.6 Vendor

18.11.2.7 Name

"The name is used as part of the topic and is used as the label on the beacon chart. As names are entered they are published to all BLE scanners.

18.11.2.8 TxPower@1m

"The power is the RSSI measured at 1 meter from the scanner. It is used to improve the distance measures that affords better X, Y location determination."

18.11.2.9 TxPower@10m

"The power is the RSSI measured at 10 meters from the scanner. It is used to improve the distance measures that affords better X, Y location determination."

18.11.2.10RSSI

18.11.2.11Filt RSSI

18.11.2.12Zone

18.11.2.13FOM

18.11.2.14<Scanner Number>

18.11.2.15X

18.11.2.16Y

18.11.3 Scanner Locations

18.11.3.1 Scanner ID

18.11.3.2 LWT

18.11.3.3 XY Location

"Scanner locations can be changed, but only after the scanner ID is first setup manually (via Tasmota Console) on an individual ESP32 scanner. Use format X,Y for the two numeric coordinates in range 0 to 100"

18.11.3.4 Receiver Gain %

"The scanner gain is a percentage of nominal gain to account for the receiver antenna gain and for block effects due to scanner location. Gains lower than 100 will result in the distance to beacon calculation to be increased by the specified percentage. Above 100 have the opposite effect."

18.11.3.5 Selected for Master Scanner

18.11.4 Configuration Parameters

18.11.4.1 BLE Scanner Group Topic (e.g. BLEScanners)

"A group topic is used to publish the same message to all BLE scanners. This assures that all will have the same parameters when calculating beacon position."

18.11.4.2 BLE Scanner Root Topic (e.g. BLEScan)

"The BLE scanners use a topic format of 'root/#/name' to publish beacon information. This text box entry is the 'root' part of this topic. It will be shared among all BLE scanners."

18.11.4.3 BLE Scanner Scan Interval (seconds)

"A scan will be started periodically at the number of seconds entered for the scan interval. There is a corresponding entry for scan duration which is the actual time when listening for beacons. The difference in the two is when the antenna will be dedicated to Wifi. During scanning the antenna will be multiplexed between the bluetooth and Wifi. Default is 60 seconds."

18.11.4.4 BLE Scanner Scan Duration (seconds)

"During the scan duration (seconds) the ESP32 will be listening for announcements made by the beacons. Note related setting for Scan Interval. Default is 30 second duration."

18.11.4.5 BLE Scanner Reporting Frequency

"The scanners can publish beacon position after every scan or publish only with a change in X,Y or Zone"

18.11.4.6 Beacon RSSI Measurement Error

"When the beacon RSSI is considered to have no error then a value of 0 is entered and no filtering will be done. Otherwise, a Kalman filter will be used to account for RSSI variation in measurement. A larger number reflects greater variably in the quality of the measurement. A value under 10 is reasonable. Max is limited to 50 by scanner."

18.11.4.7 Beacon XY Measurement Error

"The XY calculation is based upon the RSSI (filtered or unfiltered) from all scanners. As beacons drop out from the scanner's range the calculation will be affected by the smaller sample size. Filtering this effect will reduce variability in the X,Y location reporting. It also used as the Zone window hysteresis. The Zone is the single value reporting of the X & Y parameters. A value under 5 is reasonable. If value of 0 is used then no filtering will be done on XY which also results in a noisy Zone report."

18.11.4.8 Beacon Zone Hysteresis

"The Zone is a single number that represents the X and Y coordinates using formula X*100+Y. The Zone value only changes when the specified hysteresis has been reached since the last Zone change."

18.11.4.9 Beacon Dropout Count

"A scan is performed every minute or other interval specified. Beacons may or may not respond to a scan. After a number of consecutive scans without a response the beacon should be considered out of range of the scanner and this scanner's information no longer considered in the XY location determination. Values above 2 are reasonable."

18.11.4.10 Beacon New Discovery Disable

"Beacons tend to appear over time and clutter the system with bluetooth devices that have no interest. After the known beacons have been discovered then new discovery should be disable in a typical setup."

18.11.4.11 Multiple Beacon Removal

"Click to remove all beacons from all scanners. This includes the names of the beacons."

"Click to remove all beacons that have not been given names from all scanners."

18.11.4.12 Master Scanner Selection

"When 0 is selected then the ESP32 will find the lowest numbered scanner that is online to be the master. If a specific master is selected then it will always be master even when not online. Selecting a specific one may be useful for analysis of system behavior, but for normal use the selection should be 0 to provide the most robust reporting."

18.11.4.13 MQTT Retain Usage

"The retain flag is part of the MQTT protocol. A publish message that has been flagged as retain will be saved by the MQTT broker and it will be resent to any subscribed client that has just come online. Retained messages tend to be hard to remove and result in behavior that may be hard to understand. They are useful to assure that mcsMQTT is initialized immediately when it starts. Otherwise, it needs to wait for up to the log reporting interval which is typically five minutes before confidence will exist in the Beacon Selection table."

18.12 Local Page

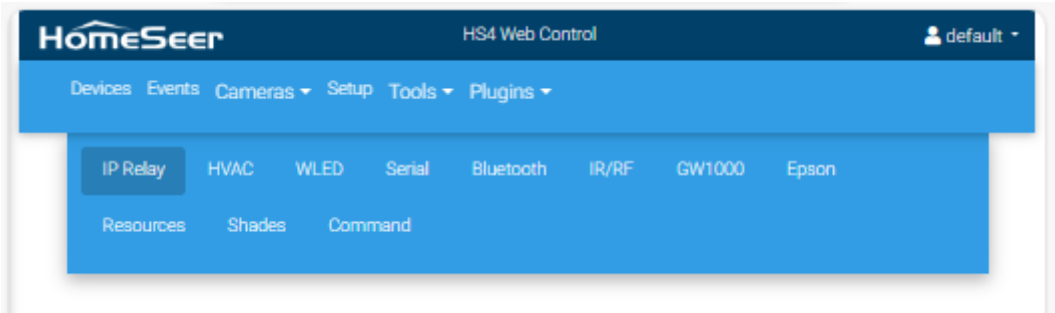


Figure 211 Local Page Tabs

18.12.1 IP 8 Channel Relay/Input

18.12.1.1 IP Address

"Used for TCP access. Needs to be on same subnet as HS."

18.12.1.2 Port

"Used for TCP access. Default is 1234."

18.12.1.3 IP Address

"Used to poll the inputs of relay/input module. Leave blank if inputs are not going to be used."

18.12.2 Local HVAC (Intesis/Daikin/Venstar/Midea/AirTouch)

18.12.2.1 Intesis/Daikin IP Address

"Used for IP access to Daikin or Intesis unit. Needs to be on same subnet as HS."

18.12.2.2 Daikin 13 Digit Key

"13-digit key from sticker on the back of the Daikin Wifi unit."

18.12.2.3 Intesis vs. Daikin Protocol

"Protocol for some units is REST and for others such as Intesis it is WMP."

18.12.2.4 Venstar Discover

"Use SSDP to try to find Venstar Thermostat on the network."

18.12.2.5 Venstar Polling Rate

"Polling rate in milliseconds for status updates. Use 0 to disable contact with thermostat."

18.12.2.6 Venstar IP

"IP address of the Venstar thermostat."

“Select model of thermostat that is being setup."

18.12.2.7 Discover Midea Thermostats

“Use UDP to try to find Midea Thermostat on the network.”

18.12.2.8 Polling Rate (milliseconds)

“Polling rate in milliseconds for status updates. Use 0 to disable contact with thermostat.”

18.12.2.9 Account Email

"Account Email associated with the appliance(s)."

18.12.2.10 Account Password

"Account password associated with the appliance(s)."

18.12.2.11 Folder Path of midea-beautiful-air-cli.exe

"Full path of folder in which Python executable midea-beautiful-air-cli.exe is located. e.g. C:\Users\me\AppData\Local\Programs\Python\Python310\Scripts\"

18.12.2.12 Thermostat IP

"Specific IP of thermostat for the case where it is not visible with UDP discovery on the HS LAN. Not needed otherwise."

18.12.2.13 AirTouch Polling Rate

"Polling rate in milliseconds for status updates. Use 0 to disable contact with thermostat."

18.12.2.14 AirTouch IP

"IP address of the AirTouch thermostat hub."

18.12.3 WLED

18.12.3.1 Max Segment Index

"If WLED controller is setup with multiple segments then enter then maximum index for this controller."

18.12.3.2 IP Address

"Segements use HTTP JSON so the IP address is needed."

18.12.3.3 Select Playlist

"Select file that contains the WLED playlist"

18.12.3.4 New Playlist

"Playlist file contains a WLED playlist that can be published on command"

18.12.3.5 Playlist JSON Content

"Example...{""ps"":[26,20,18,20],""dur"":[30,20,10,50],""transition"":0,""repeat"":10,""end"":21}. This example applies preset ID 26 For 3 seconds, then preset 20 for 2 seconds,

then preset 18 for 1 second, lastly preset 20 again for 5 seconds. This repeats 10 times, then preset 21 Is applied."

18.12.3.6 Execute Playlist

"Select WLED Topic that will show the playlist"

18.12.4 Serial

18.12.4.1 Serial Port

"Exclude the prefix. The plugin will prefix with COM for Windows and /dev/ttyUSB for Linux. Alternately enter the full IP and port such as 192.168.0.2:1234 for a raw network connection."

18.12.4.2 Serial Baud

"e.g. 9600. Other properties will be N,8,1. If IP is used for the port then the baud entry is ignored. It must be set manually in the IP/Serial device."

18.12.4.3 Serial End Of Line

"Byte that Is used To terminate a line Or messages. Default Is 10 (Line Feed). Blank indicates to use 3 second without additional bytes."

18.12.4.4 Serial Transmit Rate

"Minimum number of milliseconds between messages sent on the serial port. Used to limit low bandwidth links."

18.12.4.5 Serial Protocol Layer

"Select protocol layer on serial transport."

18.12.4.6 Serial Decoder

"Select decoder to convert the serial stream to JSON"

18.12.5 Bluetooth for Sensor and Actuators

18.12.5.1 OpenMQTTGateway Base Topic

"Enter base MQTT Topic of each ESP32 OpenMQTTGateway unit. Use semicolon to seperate. e.g. home/OMG_ESP32_BLE;BLE/OMG_heltec_ble or more generic as BLE."

18.12.5.2 OpenMQTTGateway Traffic Control

"Discovery will enable reception of all BLE devices. Update will set whitelist of devices that have been associated with HS on MQTT Page, Association Tab. Disable will ignore all data from the Gateway(s). Note the MQTT Page, Association Tab, Reject column can be used to selectively restrict Gateway from sending updates for that device."

18.12.6 Bluetooth Beacon for Home-Away

Figure 212 BLE Beacon Presence Detection for HS4

The Windows implementation of the BLE Becon presence detection uses the Bluetooth capabilitites built into Windows. It may be a Bluetooth device integrated into the motherboard or may be a USB dongle. The Linux implementation uses the Bluetooth build into RPi and should also work with a USB dongle, but this second configuration has not been tested. Install instructions of the Linux BLEMQTT application that communicates with the plugin via MQTT is provided in Section 21.16.6.

When using Bluetooth on the local computer then mode radio is selected among the top three options. The first turns off the process that is gathering beacon advertisements. The second enables the process, but will ignore any beacon advertisements that have not been associated with HS Devices. The third collects all beacon advertisements and makes them visible on the MQTT Page, Association tab from where association to HS Device can be made. It is recommended that this discovery mode only run for a short period until a known beacon has been identified. Smartphones and similar randomize their Bluetooth MAC so a large number of beacons appear to exist while they are all phantoms.

When doing beacon data collection using BLEMQTT on a remote computer then the top and lower two mode radios are used. The functionality is the same as with local, but management of running BLEMQTT is not handled by the plugin and is a user responsibility.

Starting BLE Beacon detection on Windows or Linux is automatic when running the plugin on the same computer as HS4. No remote Windows capability is provided.

For remote RPi installations where one or more RPi are not on the HS4 computer then BLEMQTT is started with two methods to provide configuration parameters. One is on the command line. The other is in the file CommandLine.ini. Use of CommandLine.ini provides additional security when username and password are needed.

Systemctl will normally be used to manage the execution of BLEMQTT on the remote computer. No matter what mechanism is used, the expected command line syntax is shown below. Required is the command BLEMQTT and the broker-ip as its parameter. The other parts of the syntax are dependent upon the user's configuration. Sudo if not logged in as an administrator. Username and password if the

MQTT broker has login credentials. If using the mcsMQTT Internal MQTT Broker then the broker-ip is the actual IP (e.g. 192.168.0.100) of the HS computer. It is not localhost or 127.0.0.1.

```
sudo ./BLEMQTT broker-ip username password
```

The plugin will attempt to start BLEMQTT on the local computer when Local Page Bluetooth tab enables BLE scanning so either make a dummy executable of the same name or remove it from the HS computer if you do not want to run BLEMQTT on the HS4 computer.

When the plugin exits or when the Local Page Bluetooth tab radio is set to disable scanning then the plugin will send MQTT message (Beacon/MQTTEXIT 1) that will shut down BLEMQTT on all computers that use the same MQTT broker. If you use systemctl to launch it then it will be able to recognize it is shut down and start it again.

mcsMQTT will process all received Beacon/xx.xx.xx.xx.xx.xx topics and treat them as beacons.

Two options exist for data that is stored in the HS4 Beacon Feature. One is the RSSI of the beacon to get an idea of distance. The other is a simple on/off binary for ease of presence change trigger. Presence or RSSI will continue to be reported until the specified timeout has occurred since the last advertisement by the beacon. Shorter timeouts have greater potential for false detection. Longer timouts will increase the latency in reporting that a beacon is no longer present. A becon detection will be reported immediately.

18.12.6.1 Beacon Enable

"BLE Beacon operations to enable, to only update status of previously discovered beacons, or to collect New beacon advertisements. Can be selected for Windows Local and Remote or Remote only."

18.12.6.2 HS Device Value Contents

"HS Device Value is set to -1 when Beacon is out of range. When Beacon is in range then the HS Device Value will hold either 0 or the negative of filtered RSSI."

18.12.6.3 Beacon Timeout

"Enter number of seconds of no advertisement before declaring beacon no longer in range."

18.12.7 Bluetooth using Espresense for Room Localizaation

18.12.7.1 Delete

"Check to delete room from table. If Espresense continues to report the room it cannot be deleted."

18.12.7.2 Room Radius

"Enter the distance from the ESP32 station for which Espresense Devices will be considered to be in the station's room."

18.12.7.3 Room Dwell Exit Seconds

"Time is seconds to consider a Espresense Device to have left a room before changing status to 'Home'. If Espresense Device enters another room this exit dwell time is ignored."

18.12.7.4 Associate Bluetooth Device

"Check to collect room assignment changes in network long term database for later viewing or charting"

18.12.7.5 Tx Power Gain

"Multiplier to be applied to distance measurement from Espresense Device to account for variation in Tx power of the Device. Value > 1.0 will increase the distance that is being reported by Espresense, which implies a higher Tx Power for the Device than a nominal bluetooth beacon."

18.12.7.6 Short Term Recording

"Check to collect room assignment changes in SQLite database for later viewing or charting"

18.12.7.7 Long Term Recording

"Check to collect room assignment changes in network external database for later viewing or charting"

18.12.7.8 Show Distance Data in Association Table

"Check to put raw distance data on MQTT Page, Association Tab"

18.12.7.9 Remove Unassociated Devices

"Check to remove espresense devices that have not been associated with HS and inhibit adding devices to table."

18.12.7.10Room – Device Matrix Refresh

"Use button to refresh espresence bluetooth distance and time data"

18.12.7.11Association Table Data

"Check to put raw distance data on MQTT Page, Association Tab"

18.12.8 Broadlink IR/RF

18.12.8.1 Network SSID

Enter SSID of AP to which Broadlink will join.

18.12.8.2 Network SSID Password

"Enter passwork credentials of local nettwork AP."

18.12.8.3 Network Security Mode

"Select the security mode used by network AP."

18.12.8.4 Join Network

“Long press the reset button until the blue LED is blinking quickly. Long press again until blue LED is blinking slowly. Manually connect to the WiFi SSID named BroadlinkProv. Assure correct SSID, password and security mode parameters have been entered."

18.12.8.5 Broadlink IP

"IP of Broadlink device"

"IP Address of the Broadlink device. Needed when scan does not detect it. There must be an IP before other Broadlink operations can be done."

18.12.8.6 Scan for Broadlink Device

"Scan for Broadlink Device", "Click to discover Broadlink device on network. This will allow authorization data to be collected so further communications are possible."

18.12.8.7 HS Device/Feature Model

"Select how IR/RF codes will be represented in HS Device. VSP is more consise. Feature per code for voice control."

18.12.8.8 Unit for Learning and Assignment

"Select the broadlink device that will be used for learning and assignment support."

18.12.8.9 Code Name to be Learned

"Name that will be assigned to to next learned IR or RF code. Set this box before using a Learn button."

18.12.8.10 Learn IR

"Learn IR Code", "Enter the name of the code to be learned in textbox above. Press this learn button, Broadlink device LED will turn red to indicate learning. Point remote at red led and press button on remote to learn the code. Red led will turn off."

18.12.8.11 Learn RF

"Learn RF Code", "Enter the name of the code to be learned in textbox above. Press this learn button, Broadlink device LED will turn red to indicate learning. Point remote at red led and do long press on remote followed by multiple short presses on remote to learn the code. Red led will turn off."

18.12.8.12 Cancel Learning

"Cancel Learning", "Learning is cancelled automatically when code has been detected and red led goes out. To abort this process, use the Cancel button."

18.12.8.13 Play Code

"To confirm proper control of learned code it can be played at this time. It can also be played later via HS Device."

18.12.8.14 Import Codes

"Copy pronto hex code from a database such as http://irdb.tk/find/ and paste into text box after the name of the code has been entered above. Alternately, enter a full filename and import a set of pronto codes for the current Appliance. In this case the file format is assumed to be a line with the code name follwed by the Pronto hex code on the subsequent line. Blank lines are ignored."

18.12.8.15 Auto Assignment

"Assignment of an Appliance to a HS Device can be done at time of learning or can be done manually later."

18.12.8.16 Manual IR Assignment to HS

"Assigning an IR Appliance will create a HS Feature or add a VSP entry depending upon the HS Feature model selected. All learned codes for that Appliance will be available for HS control."

18.12.8.17 Manual RF Assignment to HS

"Assigning an RF Appliance will create a HS Feature or add a VSP entry depending upon the HS Feature model selected. All learned codes for that Appliance will be available for HS control."

18.12.8.18 Remove IR/RF Appliance from HS

"Remove Assigned Appliance Codes from HS Device / Feature."

18.12.8.19 View IR/RF Library

"Contains listing of IR and RF learned and imported library"

18.12.8.20 Edit the Code's Repeat Count

"Numeric change in number to times to repeat the learned/imported code. A value of 0 is zero repeats. 1 is for one repeat which means the code is sent twice."

18.12.8.21 Edit the Code's Pulse Timing

"Numeric change in pulse count to be applied to the learned/imported code. Pulse is the change in the number of pulses that represents each On/Off burst. This can be used to tweak the timing of the IR signal for sensitive equipment. A value of -1 will reduce the number of pulses at each transition by one. A value of 3 will increase the number of pulses by 3."

18.12.8.22 Delete from Library

"The selected item from the library can be permanently deleted. This will remove the item from Broadlink.ini library."

18.12.9 GW1000

18.12.9.1 IP of HS NIC

"Enter network interface IP. If only one NIC is present then 127.0.0.1 can be used. This is the one that has a route to GW1000. On the App WS View a specific IP is needed (not 127.0.0.1) on the Customized page of the WS View App after clicking on the device in the App from Device List menu and selecting 'more' option until Customized is viewed."

18.12.9.2 Port of HS NIC

"Enter network Interface Port. Default port Is 8080, but can be anytyhing that matches the port setup on the App WS View."

18.12.9.3 Disconnect from GW1000

"When setup, mcsMQTT will be listening for any upates from GW1000 unless disabled."

18.12.9.4 GW1000 Connection Timeout

"Enter number of minutes without data received from GW1000 before resetting listener port. Leave blank to not monitor timeout."

18.12.10 Epson

18.12.10.1 Epson IP

"Used for TCP access to Epson ESC/VP21.net projector. Default port is 3629. Use :xxxx suffix if other port being used."

18.12.10.2 Epson Poll

"Interval to request status updates from Epson ESC/VP21.net projector."

18.12.10.3 Epson Disconnect

"Disconnect from Epson projector."

18.12.11 Resources

18.12.11.1 Resource Selection

"Check to include collection of data for this resource."

"Name of additional (Non-HS/Non-Plugin) processes to monitor."

18.12.12 Hunter Douglas PowerView Gen3

18.12.12.1 Hub IP Address

"IP address of PowerView hub. e.g. 192.168.100.200"

18.12.12.2 Status Update Interval (ms)

"Status update query interval (milliseconds)."

18.12.12.3 URL Path

"Select between Gen3 and Gen2 for the equipment being used.

18.12.13 Command Terminal

18.12.13.1apsaccess Path

"Full path to apsaccess executable. e.g. C:\Program Files\APS\apsaccess.exe"

18.12.13.2Execution Interval (ms)

"Execution interval (milliseconds)."

18.13 Cloud Page

18.13.1 URL

18.13.1.1 URL

"Used as the address that will be polled. HTTP protocols normally start with 'http://' or 'https://'. UDP protocols and TCP normally start without protocol prefix. WebHook protocol normally start with 'ws://’"

18.13.1.2 Polling Interval

"Enter Input polling interval in milliseconds if periodic requests to this URL is desired. If this URL is setup to be used only from Device change requests from HS then set to 0."

18.13.1.3 Polling Endpoint

"Endpoint to add as suffix to URL IP when polling for updates. Alternately if text ends in .pub then it is a publication list file in \data\mcsMQTT subfolder that can containe one or more endpoints."

18.13.1.4 Protocol

"Select protocol to use when communicating to the server. GET is command/response and uses URL querystring to send data. POST is command/response and uses body to send data. UDP uses datagram and can be listen-only. WS listens as a WebSocket request. TCP In listens on local NIC port (127.0.0.1 can be used if only 1 NIC)."

18.13.1.5 Authorization

"Select authorization approach. This will be included in the Headers as requred with the Authentication key.”

(oAuth2 Only) "Manually generate a new oAuth2 token. This is not normally needed as tokens are monitored for expiration and new tokens generated before they expire.

18.13.1.6 Additional Parameters

"Default headers are Content-Type:application/x-www-form-urlencoded, Accept:*/*, User-Agent:mcsMQTT, KeepAlive:True, Timeout:5000, Accept-Encoding:identity;q=1.0,*;q=0. Enter additional headers or redefine any default with syntax key:value"

"Secret Key for UDP; Username and password separated by colon for Basic Authentication (.e.g. myuser:mypass); Token when Token or Bearer authentication is selected."

"URL to be used when authentication is needed per oAuth2 authencation method. A token will be returned with a call to this URL from which subsequent access to the primary URL can be made using this token for authentication."

"JSON text with the parameters necessary to perform authentication. Included will be the client token, client secret, grant type, and typically username and password. An example is
{""grant_type"":""password"","client_id"":""12345"","client_secret"":""ABCDE"","username"":""xxx@gmail.com"","password"":""xxxxx""}"

18.13.1 YoLink

18.13.1.1 mcsMQTT Clients

"Select single unless multiple instances of mcsMQTT will be accessing the same YoLink device"

18.13.1.2 YoLink Server Connection

"Enable or disable communication with the YoLink Server."

18.13.1.3 Account Access Button

"Using this button will present an authhorization screen to allow mcsSolutions (mcsMQTT) to collect the list of YoLink devices that have been setup in your account. This can be done in lieu of listing the QR codes that the plugin is being asked to manage."

18.13.1.4 YoLink 32 Character QR Code

"32-character code obtained by scanning QR Code on YoLink device"

18.13.2 Voice Monkey

18.13.2.1 Voice Monkey Token

"Enter token from voicemonkey.io dashboard"

18.13.2.2 Voice Monkey Secret

"Enter secret from voicemonkey.io dashboard"

18.13.3 Geofence

18.13.3.1 Geofence location name

"Enter name to assigne to this waypoint location. Distance and Here-Away subtopics will be created under this name."

18.13.3.2 Latitude

"Enter latitude of this location."

18.13.3.3 Longitude

"Enter longitude of this location."

18.13.3.4 Distance Boundary

"Enter distance from location that establishes geofence boundary. 20% hysteresis margin is used when determining here vs. away."

18.13.4 Sense Energy

18.13.4.1 Account Email

"Enter email used to setup account with Sense Energy."

18.13.4.2 Account Password

"Enter password used to setup account with Sense Energy"

18.13.4.3 Server Polling Rate (milliseconds)

"Enter number of digits of precions to show for wattage.”

18.13.4.4 Display Precision

"Enter rate in milliseconds at which the Sense server will be polled for data. Min And max are 10000 and 3600000 which is each minute and each hour

18.13.4.5 Server Disconnect

"When setup, mcsMQTT will be requesting upates from Sense server at the specified polling rate unless disabled."

18.13.4.6 Additional Download

"Enable download of this data from Sense server"

18.13.5 Hubspace

18.13.5.1 Account Email

"Enter email used to setup account with Hubspace.

18.13.5.2 Account Password

"Enter password used to setup account with Hubspace."

18.13.5.3 Python Path

"Enter path where python.exe is located including the name of executable python such as C:\Python37-32\python.exe"

18.13.5.4 Python Script Path

"Enter path where HubspaceRequest.py was placed such as C:\Python37-32\Scripts\HubspaceRequest.py"

18.13.5.5 Status Poling Milliseconds

"Enter rate in milliseconds at which the Hubspace server will be polled For data. Each minute is 60000. Each hour is 3600000. etc."

18.13.5.6 Server Disconnect

"When setup, mcsMQTT will be requesting upates from Hubspace server at the specified polling rate unless disabled."

18.13.6 Switchbot

18.13.6.1 Token

"Enter token from Switchbot App. Go to Profile > Preference b) Tap App Version 10 times. Developer Options will show up c) Tap Developer Options d) Tap Get Token"

18.13.6.2 Secret Key

"Enter secret from Switchbot App. Secret is obtained the same way as token."

18.13.6.3 Server Poilling Rate (milliseconds)

"Enter rate in milliseconds at which the Switchbot server will be polled for data. Minimum interval is 100000 due to Switchbot server limit of 1000 requests per day."

18.13.6.4 WAN-Visible Webhook URL

"Enter address to which Switchbot server will push event updates to HS. This URL needs to be routable from the WAN to LAN on which HS resides."

18.13.6.5 Server Disconnect

"When setup, mcsMQTT will be requesting upates from Switchbot server at the specified polling rate unless disabled."

18.13.7 Tank Utility Connect Parameters

18.13.7.1 Account Email

"Enter email used to setup account with Tank Utility."

18.13.7.2 Account Password

"Enter password used to setup account with Tank Utility."

18.13.7.3 Server Disconnect

"When setup, mcsMQTT will be requesting upates from Tank Utility server at the specified polling rate unless disabled."

18.13.8 Abode

18.13.8.1 Account Email

"Enter email used to setup account with Abode."

18.13.8.2 Account Password

"Enter password used to setup account with Abode."

18.13.8.3 Video Storage Path

"Enter path to folder where Cam caputures will be saved."

18.13.8.4 Refresh Devices

"Press to find any newly added panel devices and refresh state of existing devices."

18.13.8.5 Server Disconnect

"When fully setup, mcsMQTT will be accepting upates and providing control with Abode server. When disabled no contact with the Abode server will exist."

18.13.9 Orbit B-Hyve

18.13.9.1 Account Email

"Enter email used to setup account with Orbit."

18.13.9.2 Account Password

"Enter password used to setup account with Orbit."

18.13.9.3 Refresh from Server

"Request from Orbit Server the Devices, Timelines and Landscape information. Most current status is included in these responses."

18.13.9.4 Server Disconnect

"When fully setup, mcsMQTT will be accepting upates and providing control with Abode server. When disabled no contact with the server will exist."

18.13.10 Hunter Hydrawise

18.13.10.1Account Email

"Enter email used to setup account with Hydrawise."

18.13.10.2Account Password

"Enter password used to setup account with Hydrawise."

18.13.10.3 Additional Properties

"Additional information is available from Hydrawise server that is not included in HS Devices that are automatically created. This selection will put these in the Association Table for user association to HS as desired."

18.13.10.4 Virtual Flow Sensor

"If enabled a HS Device Feature will be created and mimic the change in the sesnor reported by Hydrawise. In addition, it can be reset to 0 on manual Devcie Control or HS Event; or automatic action within the plugin with a cycle starts."

18.13.10.5 Virtual Zone Flow Sensor

"If enabled a second Device Feature will be created for each zone. It will contain the flow sensor change reported by Hydrawise when the zone is active.the Hydrawise server will exist. In addition, it can be reset to 0 on manual Devcie Control or HS Event; or automatic action within the plugin with a cycle starts."

18.13.10.6 Auto reset Virtual Flow Sensors

"If enabled, then all virtual flow sensors will be reset to 0 when a cycle starts. It is on a zone-by-zone basis for the Virtul Zone Flow sensors and the virtual controller flow sensor when a controller cycle starts."

18.13.10.7 Server Disconnect

"When fully setup, mcsMQTT will be accepting upates and providing control with Hydrawise server. When disabled no contact with the Hydrawise server will exist."

18.13.10.8 Historical Flow Chart

"Show chart for past one week zone flow"

"Show chart for past two weeks zone flow"

"Show chart for past four weeks zone flow"

"Show chart for past eight weeks zone flow"

18.13.11 Solar

18.13.11.1 Solcast Resource Id

"Enter resource Id from Solcast account”

18.13.11.2 Account Password

"Enter IP of Solar_Assistant MQTT Server”

18.13.11.3 Server Disconnect

"Enable or disable communication with the Solar Energy Servers.”

18.13.12 Rheem EcoNet

18.13.12.1 Account Email

"Enter email used to setup account with Rheem EcoNet."

18.13.12.2 Account Password

"Enter password used to setup account with Rheem EcoNet."

18.13.12.3 Server Disconnect

"When setup, mcsMQTT will be connected to the EcoNet server ready to accept status updates unless disabled."

18.13.12.4Status Poling Milliseconds

"Time interval in milliseconds to request status. Status is pushed when a change occurs due to equipment or HomeSeer control. Status is not pushed from control performed by Rheem EcoNet App."

18.13.12.5Additional Download

"Get login and equipment information from EcoNet server. HS Device and Features will be created if not already created for most of the equipment properties. Others will only be visible in the Association table."

18.13.13 Thermostats NuHeat

18.13.13.1Account Email

"Enter email used to setup account with NuHeat."

18.13.13.2Account Password

"Enter password used to setup account with NuHeat."

18.13.13.3House(s) Id

"Enter numeric Id of House for thermostats. Use semicolon to separate multiple thermostats. Id can be obtained from NuHeat account browser page or App."

18.13.13.4Server Pollling Rate

"Enter rate in milliseconds at which the NuHeat server will be polled for data. Min And max are 10000 and 3600000 which is each minute and each hour."

18.13.13.5Server Disconnect

"When setup, mcsMQTT will be requesting upates from NuHeat server at the specified polling rate unless disabled."

18.13.14 Thermostats Nexia/Trane/American Standard

18.13.14.1Account Email

"Enter email used to setup account with Nexia."

18.13.14.2Account Password

"Enter password used to setup account with Nexia."

18.13.14.3House(s) Id

"Enter numeric Id of House for thermostats. Use semicolon to separate multiple thermostats. Id can be obtained from Nexia account browser page or App."

18.13.14.4Brand

"Access to the cloud server depends upon the brand of thermostat. Select the appropriate brand."

18.13.14.5Server Pollling Rate

"Enter rate in milliseconds at which the Nexia server will be polled for data. Min And max are 10000 and 3600000 which is each minute and each hour."

18.13.14.6Server Disconnect

"When setup, mcsMQTT will be requesting upates from server at the specified polling rate unless disabled."

18.13.15 Thermostats Carrier/Bryant/Ion

18.13.15.1 Account Email

"Enter email used to setup account with Carrier/Bryant/Ion."

18.13.15.2 Account Password

"Enter password used to setup account with Carrier/Bryant/Ion."

18.13.15.3 Python Path

"Enter path where python.exe is located including the name of executable python such as C:\Python37-32\python.exe"

18.13.15.4 Python Script Path

"Enter path where CarrierRequest.py was placed such as C:\Python37-32\Scripts\CarrierRequest.py

18.13.15.5 Server Pollling Rate

"Enter rate in milliseconds at which the Carrier/Bryant/Ion server will be polled for data. Min And max are 10000 and 3600000 which is each minute and each hour."

18.13.15.6 Server Disconnect

"When setup, mcsMQTT will be requesting upates from server at the specified polling rate unless disabled."

18.13.16 Pool

18.13.16.1Account Email

"Enter email used to setup account with Hayward Omnilogic."

18.13.16.2 *Account Password*

"Enter password used to setup account with Hayward Omnilogic"

18.13.16.3 *Python Path*

"Enter path where python.exe is located including the name of executable python such as C:\Python37-32\python.exe"

18.13.16.4 *Python Script Path*

"Enter path where OmnilogicRequest.py was placed such as C:\Python37-32\Scripts\OmnilogicRequest.py"

18.13.16.5 *Server Polling Rate (milliseconds)*

"Enter rate in milliseconds at which the Omnilogic/Bryant/Ion server will be polled for data. Min And max are 10000 and 3600000 which is each minute and each hour."

18.13.16.6 *"Server Disconnect"*

"Connect to Omnilogic Server", "Disconnect from Omnilogic Server"

18.14 Interactive Page

18.14.1 Expression

"Replacment variables are typically used to identify desired property such as $$DVR:(123): to show DeviceValue of feature 123. Expression can also be used such as $DVR:(123): + $DVR:(456):. If text rather than number will be evaluated then encase the expression in quote. e.g. ""$$DTR(123):"". Full set of replace variabled and expression operators and functions are in mcsMQTT.pdf tables."

18.14.2 Send MQTT Message

“Send MQTT Message using format Topic=Payload. Both Topic and Payload can use substitution variables and expressions.”

18.14.3 Run HS Script Command or Expression

“Execute a script command such as hs.DeviceValue(123)-hs.DeviceValue(124). Script uses vb.net syntax"

18.14.4 Run HS Script

"Execute script. Syntax filename,funcname,parms e.g. MyScript.vb,Main,""abc"" or MyScript.vb,AnArray,{1,2,3} or MyScript.vb"

19 Zigbee2MQTT

Zigbee devices are designed for low power operation and communicate over a limited distance. They typically communicate with a hub that is connected to internet via IP and then cloud logic used to respond to the Zigbee communication. Smartthings, Echo, Phillips, Osram are typical examples of hub providers. There is also a Raspberry Pi based hub Raspbee or Conbee that has interface logic local and does not use the internet. A means is still needed to interface between the RPi and whatever user control/status interface is employed by the used, such as a HS plug-in.

What is described here is another non-Cloud interface mechanism that employs MQTT protocol so is able to utilize mcsMQTT or any other MQTT client to interact with Zigbee devices. The cost of the hardware interface is under $10 and the software is open source community supported.

A good accounting of which interface mechanism is able to handle which zigbee devices is at https://zigbee.blakadder.com\

Zigbee2MQTT is a node.js software application that utilizes a RF USB dongle to decode Zigbee protocol and provides the decoded result via MQTT. It also operates in the reverse to accept MQTT commands and control Zigbee devices. A picture of the covered and uncovered USB dongle is shown in Figure 213.

Figure 213 Zigbee USB Dongle

The Wiki for this is at https://www.zigbee2mqtt.io/. The "getting started" step of obtaining the USB dongle and flashing it is what I have done and will send them to anyone who wants one. Anybody can also buy the programming tools and USB stick and program it themselves. While there are some domestic suppliers, most hardware is available from China.

The Running Zigbee2mqtt step (https://www.zigbee2mqtt.io/getting_started/running_zigbee2mqtt.html) describes the process of setting up this node.js application on one's computer. It can be done on any modest Linux system. A step by step guide for setup on Windows is at https://forums.homeseer.com/forum/lighting-primary-technology-plug-ins/lighting-primary-technology-discussion/mcsmqtt-michael-mcsharry/1264779-zigbee2mqtt-on-windows

The Wiki also documents devices that known to work with the USB dongle. It also describes how to add new devices that come to market. Support by the developer and other users, via the github forum, is also very good at this time.

The convention that I have observed is that JSON payload is used for MQTT communication in both directions. A Zigbee device will have one Topic. The parameters of the topic are encoded in the JSON payload. When commanding the device, such as turning a light on or off the same publish topic is used as the subscribed on with a "/set" suffix.

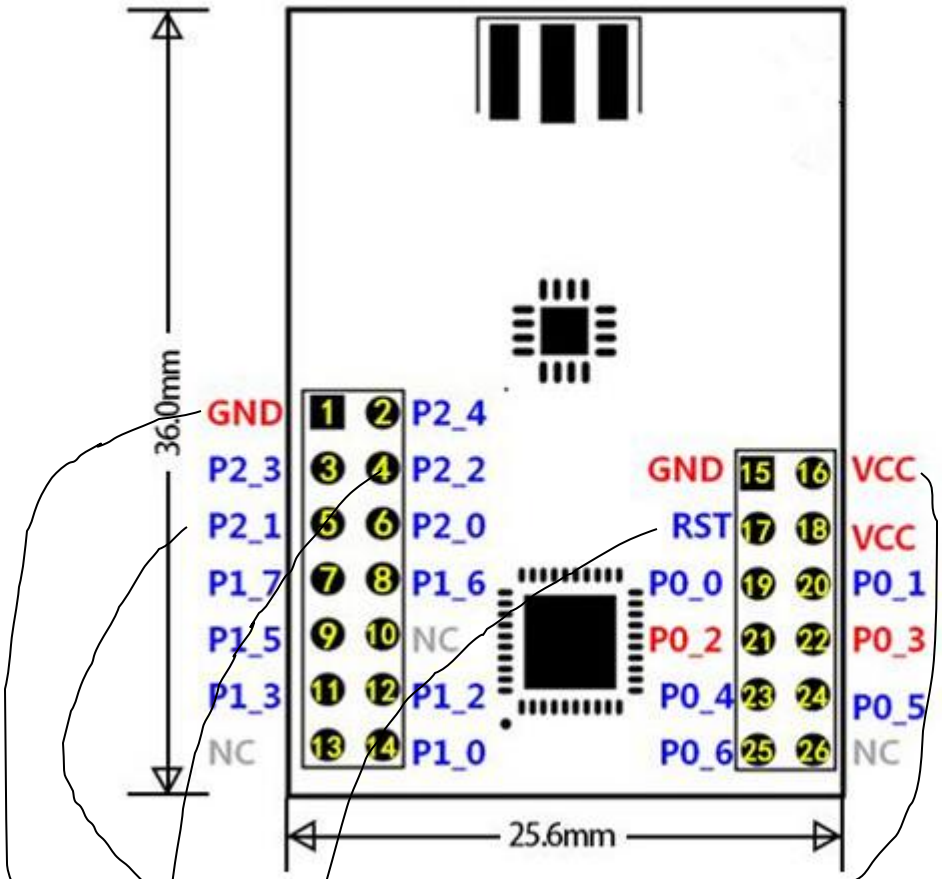
19.1 Zigbee2mqtt Firmware

Firmware is now available in three flavors. The "default" is what is being used to flash the CC2531. There is also a "max stability" and a "max devices". The repository for the latest firmware for the coordinator is at https://github.com/Koenkk/Z-Stack-firmware/tree/master/coordinator. It is also possible to flash the device to be a router rather than a coordinator. The router acts as a repeater and has no interface to the interfacing computer. Router firmware is at https://github.com/Koenkk/Z-Stack-firmware/tree/master/router.

There are three types of modules that can be flashed. The simplest is CC2531 which provides a USB interface to the computer and an antenna etched into its circuit card. Flashing this is per the Wiki https://www.zigbee2mqtt.io/getting_started/flashing_the_cc2531.html.

A CC2530+CC2591 can be used to improve range with an external antenna. An example is https://www.aliexpress.com/item/ZigBee-Wireless-Module-CC2530-CC2591-PA-Module/1831284083.html?spm=a2g0s.9042311.0.0.b9b74c4dxZ8wmN It does not have a USB interface so something like a FTDI TTL/Serial adapter would need to be connected TX to P0_3, RX to P0_2, 3.3V to VCC, Gnd to Gnd. This is shown in Figure 217.

To flash this device the CCDebugger is used with Dupont wires to make the connections between the header pins in the module to the cable from the CCDebugger. Five wires need to be connected (P2_2, RST, P2_1 VCC and Ground). See Figure 214, Figure 215 and Figure 216. Note the reference cutout slot of Figure 215 is from the CCDebugger and the cable has a mating protrusion so be careful to connect pins in the correct orientation.



引脚序号	引脚名称
1	GND
2	P2_4
3	P2_3
4	P2_2
5	P2_1
6	P2_0
7	P1_7
8	P1_6
9	P1_5
10	NC
11	P1_3
12	P1_2
13	NC
14	P1_0
15	GND
16	VCC
17	RST
18	VCC
19	P0_0
20	P0_1
21	P0_2\U_RX
22	P0_3\U_TX
23	P0_4
24	P0_5
25	P0_6
26	NC

Figure 214 CC2530+CC2591 Header Pins

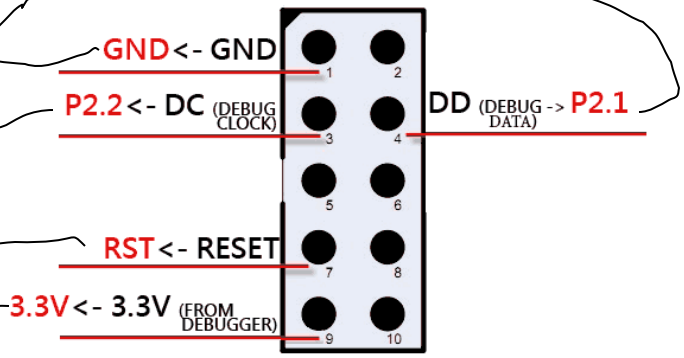


Figure 215 CCDebugger Cable Pinout

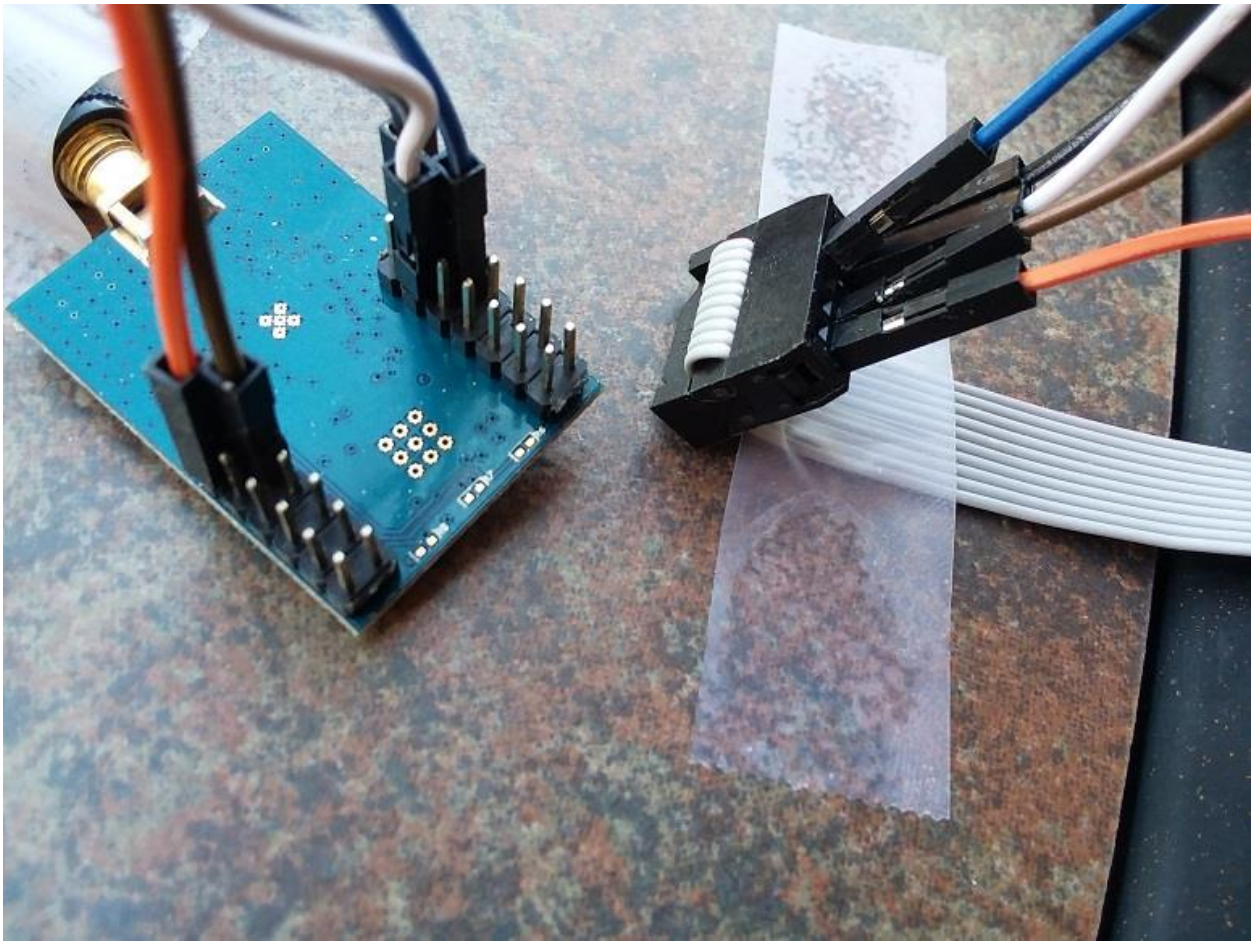


Figure 216 CCDebugger to CC2530+CC2591 Flashing Wiring

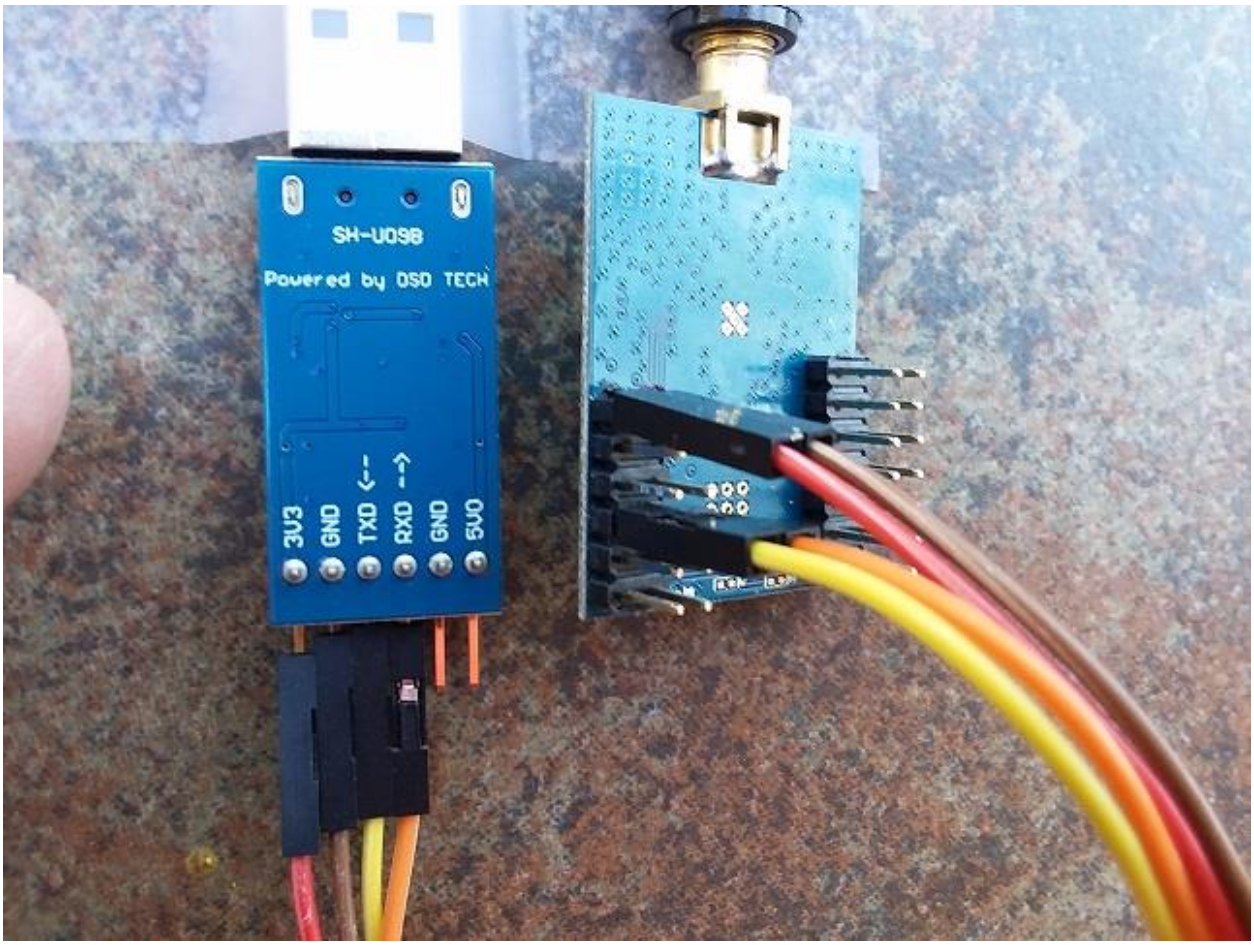


Figure 217 CC2530+CC2951 Wired to USB/Serial

Modules also can be obtained that have the CC2530+CC2951+USB. An example is at https://www.aliexpress.com/item/RF-TO-USB-CC2530-CC2591-RF-switch-USB-transparent-serial-data-transmission-equipment/1996354384.html?spm=a2g0s.9042311.0.0.27424c4d2np0bN. They are the most expensive and also require soldering with jumpers between three pins and ground to perform the flash as shown in Figure 218. Versions of firmware are available in the repository for each variant.

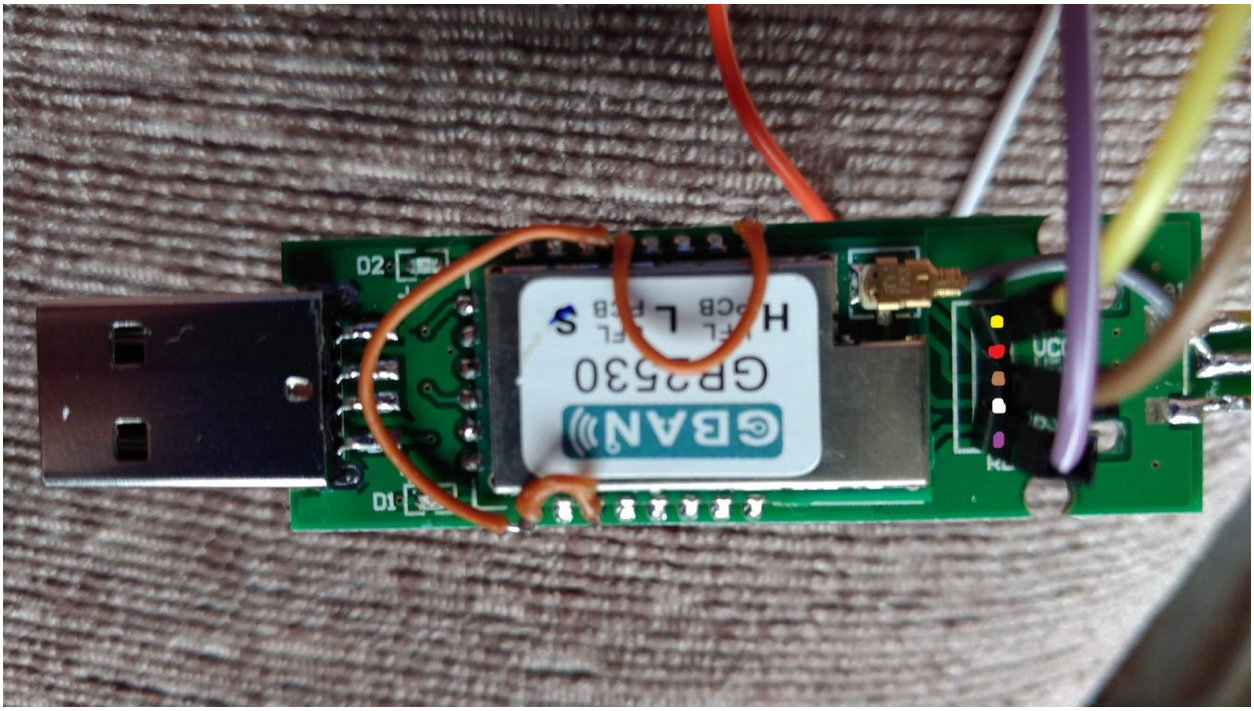


Figure 218 RF TO USB (CC2530 CC2591) Wiring

19.2 Zigbee2MQTT on Windows

The segment below was originally published on the Homeseer Message Board https://forums.homeseer.com/forum/lighting-primary-technology-plug-ins/lighting-primary-technology-discussion/mcsmqtt-michael-mcsharry/1264779-zigbee2mqtt-on-windows and placed here for easier reference.

I have been communicating with ptvo on https://github.com/Koenkk/zigbee2mqt...ment-444345329 for a Windows port of zigbee2mqtt. I have had success. A few pieces of information are needed to make the port.

1. Need to install driver for CC2531. It is available at http://www.ti.com/general/docs/lit/g...8&fileType=zip . Unzip to someplace on the computer. Plug the USB dongle into available USB port. It should show up as an "Other Device" TI CC2531 USB CDC. Right click on this device and select option to install driver. Browse to "\driver" subfolder of the unzip. Device should now show up under Ports (COM & LPT) as a COMXX where XX was 15 in my case.

Restart the computer to remove access dependencies.

Note that I was using a bluetooth mouse on W7 and when this driver was installed it made the mouse non-operational. I had to uninstall/remove the driver for the CC2531 before the mouse was operational.

2. Install node.js from https://nodejs.org/en/download/. I used the Windows Installer (.msi).

3. Create folder to place zigbee2MQTT. I used C:\opt\zigbee2mqtt, but likely can be anywhere. The

remaining instructions use this location.

4. Open command window as administrator (Windows search for "cmd", right click, run as administrator)

5. Clone zigbee2MQTT from git repository from command window. (git clone https://github.com/Koenkk/zigbee2mqtt.git C;\opt\zigbee2mqtt). If git is not yet installed on Windows then it can be from https://git-scm.com/download/win . It likely is possible to just download the zip and expand it into a Windows folder, but I did not try this approach.

6. Edit the C:\opt\zigbee2mqtt\data\configuration.yaml. One line to provide the MQTT broker IP address. One line for the USB Dongle port (e.g. COM15). You can also change the base topic from zigbee2mqtt if you desire. I have a different topic for each computer where I have the USB Dongle installed.

7. Navigate to the install folder (CD C:\opt\zigbee2mqtt) in Command Window

7a. (added after initial post). Install zigbee2mqtt dependencies. From command prompt run "npm install"

8. Run zigbee2mqtt from Command Window (npm start). Feedback will be in the Command Window. You should also observe the MQTT LWT message being online on your MQTT client, such as mcsMQTT. If it does not work then it is possible that there is feedback telling you reset the USB dongle using the button nearest the USB connector. With the case I provided installed it is possible with a small non-metalic probde angled to the button. One can also drill a hole in the top of the case above the button. If the case is removed then the plastic latch may break. I provided a flexible top should the case latch break. One could also use tape as another alternative.

9. To run on windows startup I followed the following process. While it may not be the most elegant it does work:
9a. Install PM2 from Command Window (npm install pm2 -g)
9b. Create batch file that will be used to start the js application. I called it z.bat with one line contents or "pm2 start c:\opt\zigbee2mqtt\index.js"
9c. Create shortcut to the bat file (right click, create shortcut)
9d. Copy/Paste shortcut into Windows startup folder. In my case it was at C:\Users\Dell\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup for user Dell
9e. Restart computer
9f. Observe the node process in Windows Task Manager or observe the LWT MQTT message on a MQTT client.

19.3 Zigbee Sniffer

Wireshark is used when lower level of information is needed such may be the case for a new device that does not follow the pattern of others. The capture device for Wireshark is most conveniently a CC2531 that has been flashed with sniffer firmware and whsniff to read the interface and pipe data to Wireshark. A how-to is available at https://www.zigbee2mqtt.io/how_tos/how_to_sniff_zigbee_traffic.html. It includes the download links for the firmware, whsniff and Wireshark with an orientation to an Ubuntu install.

In my case I separated the Wireshark operation on Windows from the sniffer operation on a RPi. An SSH connection was used to pipe the collected data from whsniff on RPi to Wireshark on Windows. Plink.exe, which is part of the PuTTY download is used to establish the SSH connection. From the Windows command prompt one of the following two commands are used. The first can be used if a login with superuser privileges is possible. The second is a login using authentication key rather than password.

"C:\Program Files\PuTTY\plink.exe" -ssh root@192.168.0.200 -pw root_password "/opt/whsniff-1.1/whsniff -c 11" | "C:\Program Files\Wireshark\Wireshark.exe" -k -i -

"C:\Program Files\PuTTY\plink.exe" -i "C:\Users\Dell\Desktop\id_rsa.ppk" pi@192.168.0.200 "sudo /opt/whsniff-1.1/whsniff -c 11" | "C:\Program Files\Wireshark\Wireshark.exe" -k -i –

To setup authentication a public key needs to be installed on the RPI in file ~/.ssh/authorized_keys. I used WINSCP to create the file and paste the text of the public key that was generated on the Windows computer. Note that Linux path "~/" is "/home/pi/" for user "pi".

The private key is stored in a file on the computer running Wireshark. In the above command line, it is at "C:\Users\Dell\Desktop\id_rsa.ppk", but can be any name and located anywhere on the computer. The private/public key pair can be produced using PuTTYgen which is also part of the PuTTY download. No passphrase should be included for this use. A walkthrough of using PuTTYgen is available at https://www.ssh.com/ssh/putty/windows/puttygen.

19.4 New Zigbee Devices

In general, as new devices are discovered they are added to the repository so new downloads of Zigbee2MQTT will have the information to discover them. A list that was originally compiled on March 2019 was posted at https://forums.homeseer.com/forum/homeseer-products-services/general-discussion-area/1294291-list-of-zigbee-non-cloud-supported-devices that compares three non-cloud interfaces including Zigbee2MQTT.

I have also been working the characterization of the zigbee remote RGBGenie and for the most part have the remote interfaced through Zigbee2MQTT. The discussion is at https://github.com/Koenkk/zigbee2mqtt/issues/642#issuecomment-477231522 I am not certain if it will be added to the main repository or not. The delta for this device consists of edits to devices.js and fromZigbee.js

devices.js

```
// RGBGenie

{
  zigbeeModel: ['ZGRC-KEY-013'],
  model: 'ZGRC-KEY-013',
  vendor: 'RGBgenie',
  description: '3 Zone remote and dimmer',
  supports: 'onoff dim scene control',
  fromZigbee: [fz.generic_battery,
             fz.ZGRC013_brightness_onoff, fz.ZGRC013_brightness, fz.ZGRC013_brightness_stop,
             fz.ZGRC013_cmdOn, fz.ZGRC013_cmdOff,
```

```
            fz.ZGRC013_scene,
           ],
  toZigbee: [],
  configure: (ieeeAddr, shepherd, coordinator, callback) => {
    const device = shepherd.find(ieeeAddr, 1);
    const cfg = {direction: 0, attrId: 0, dataType: 16, minRepIntval: 0, maxRepIntval: 1000,
repChange: 0};
    const actions = [
        (cb) => device.bind('genOnOff', coordinator, cb),
        (cb) => device.foundation('genOnOff', 'configReport', [cfg], foundationCfg, cb),
    ];

    execute(device, actions, callback);
  },
},
```

fromZigbee.js

```
ZGRC013_cmdOn: {
    cid: 'genOnOff',
    type: 'cmdOn',
    convert: (model, msg, publish, options) => {
        const button = msg.endpoints[0].epId;
        if (button) {
            return {click: `${button}_on`}
        }
    },
},
ZGRC013_cmdOff: {
    cid: 'genOnOff',
    type: 'cmdOff',
    convert: (model, msg, publish, options) => {
        const button = msg.endpoints[0].epId;
        if (button) {
            return {click: `${button}_off`}
        }
    },
},
ZGRC013_brightness: {
    cid: 'genLevelCtrl',
    type: 'cmdMove',
    convert: (model, msg, publish, options) => {
        const button = msg.endpoints[0].epId;
        const direction = msg.data.data.movemode == 0 ? 'up' : 'down';
        if (button) {
            return {click: `${button}_${direction}`}
        }
    },
},
ZGRC013_brightness_onoff: {
    cid: 'genLevelCtrl',
    type: 'cmdMoveWithOnOff',
    convert: (model, msg, publish, options) => {
        const button = msg.endpoints[0].epId;
        const direction = msg.data.data.movemode == 0 ? 'up' : 'down';
        if (button) {
            return {click: `${button}_${direction}`}
        }
```

```
    },
},
ZGRC013_brightness_stop: {
    cid: 'genLevelCtrl',
    type: 'cmdStopWithOnOff',
    convert: (model, msg, publish, options) => {
        const button = msg.endpoints[0].epId;
        if (button) {
            return {click: `${button}_stop`}
        }
    },
},
ZGRC013_scene: {
    cid: 'genScenes',
    type: 'cmdRecall',
    convert: (model, msg, publish, options) => {
        return {click: `scene_${msg.data.data.groupid}_${msg.data.data.sceneid}`};
    },
},
```

20 KNX-MQTT-Bridge

KNX-MQTT-Bridge is a node.js application that runs on Windows or Linux that contains the protocol translation between KNX and MQTT. It can be used as a simple way to integrate a KNX environment with Homeseer.

The install and usage instructions are at https://www.npmjs.com/package/knx-mqtt-bridge

On my Windows install the package was placed at C:\Users\Dell\AppData\Roaming\npm\node_modules\knx-mqtt-bridge. At this location was a sample config.yaml that I edited for my setup with the only changed needed was the IP address of the MQTT broker as shown in red below. The default is 'localhost'. This will be the IP of Homeseer unless an external MQTT broker is running. I also removed the MQTT broker login credentials as shown in blue since I do not use them with my broker. Likely did not need to do this as the broker would have ignored the credentials.

```
# One of 'error', 'warn', 'info', 'verbose', debug', 'silly'
loglevel: 'silly'
# One of value-only, full
#   value-only - converts any known group addresses to its value
#   full - a json object containing value. Additionally also name and unit type for known group addresses.
#messageType: value-only
messageType: full
# Ignore unknown group addresses
ignoreUnknownGroupAddresses: false
knx:
  # ETS exported group addresses
  etsExport: 'knx.xml'
  # Configuration passed to the KNX library
  options:
mqtt:
  # URL to MQTT broker
  url: 'mqtt://192.168.0.17'
  # Configuration passed to the MQTT library
  #options:
    #username: 'root'
    #password: 'root'
  # Prefix to mqtt topic
  topicPrefix: 'knx'
  # Set retain flag on messages
  retain: false
```

The KNX configuration of endpoints and routing in the KNX network is the file ets.xml stored in the same location as config.yaml. While it is not required for KNX-MQTT-Bridge, it is required for the mcsMQTT integration because this file contains the encoding used for each of the datapoints. The encodings are described at http://www.sti.uniurb.it/romanell/Domotica_e_Edifici_Intelligenti/110504-Lez10a-KNX-Datapoint%20Types%20v1.5.00%20AS.pdf but an end-user should never need to get into this level of

information. There are many different encodings used in KNX and ETS5 is the only place where this knowledge is maintained. This is very similar to UPStart in the the UPB networks.

For my testing I used a dummy file that I modified from a file I googled that contains the following. Of particular need is the group address to where communications will occur shown in red and the datapoint encoding type shown in blue.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<GroupAddress-Export xmlns="http://knx.org/xml/ga-export/01">
 <GroupRange Name="Beleuchtung" RangeStart="2048" RangeEnd="4095">
  <GroupRange Name="Untergeschoss" RangeStart="2048" RangeEnd="2303">
   <GroupAddress Name="Tasmota Light Status 1/1/0" Address="2/2/2" DPTs="DPST-1-1" />
   <GroupAddress Name="Tasmota Light Button 1/1/0" Address="2/2/3" DPTs="DPST-1-1" />
   <GroupAddress Name="Tasmota Light Control 1/1/0" Address="2/2/1" DPTs="DPST-1-1" />
   <GroupAddress Name="Tasmota Light Temperature 1/1/0" Address="2/2/4" Description="1/1/0"
DPTs="DPST-9-1" />
   <GroupAddress Name="Tasmota Light Temperature Reply 1/1/0" Address="2/2/5"
Description="1/1/0" DPTs="DPST-9-1" />
  </GroupRange>
  <GroupRange Name="Erdgeschoss" RangeStart="2304" RangeEnd="2559">
   <GroupAddress Name="Flur Halogendeckenspots " Address="1/1/0" Description="auch Garderobe
(wird mitgeschaltet)" DPTs="DPST-1-1" />
  </GroupRange>
 </GroupRange>
</GroupAddress-Export>
```

To run KNX-MQTT-Bridge I opened a command window, and used the following two commands. The first navigates to the install location. The second starts it.

```
CD C:\Users\Dell\AppData\Roaming\npm\node_modules\knx-mqtt-bridge
npm start
```

For auto-start on login I believe a similar approach as was disclosed for Zigbee2MQTT can be used. This is step 9 shown in 19.2. I believe the Linux install of KNX-MQTT-Bridge does auto start, but I did not go back and confirm this.

Again for my testing I use a ESP8266 that was programmed with Tasmota firmware that had KNX support enabled in the source. I set it up as AiLight plus a DS18B20 temperature sensor. More information on the KNX Tasmota firmware support is at https://github.com/arendst/Tasmota/wiki/KNX-features.

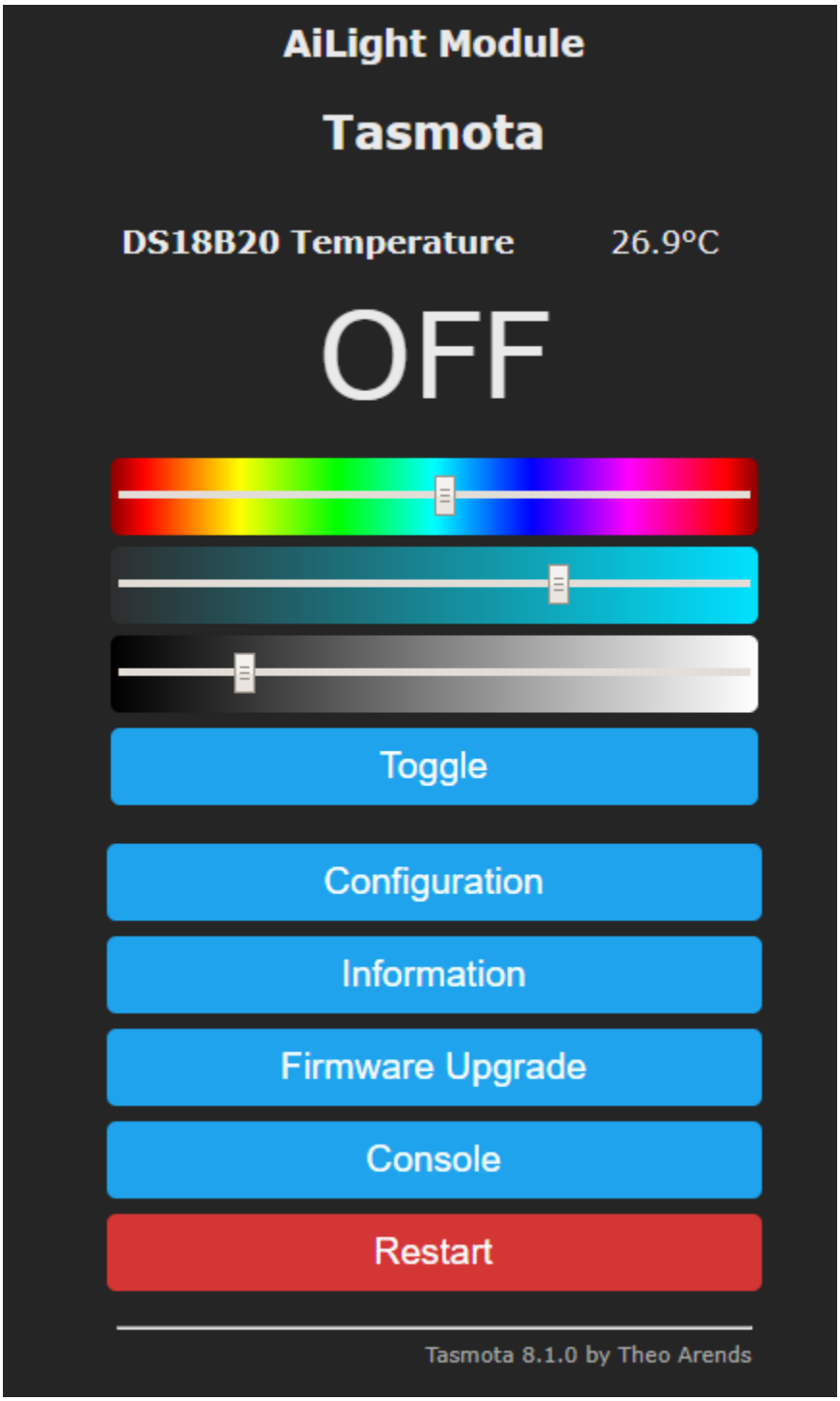


Figure 219 Tasmota Device for KNX Integration

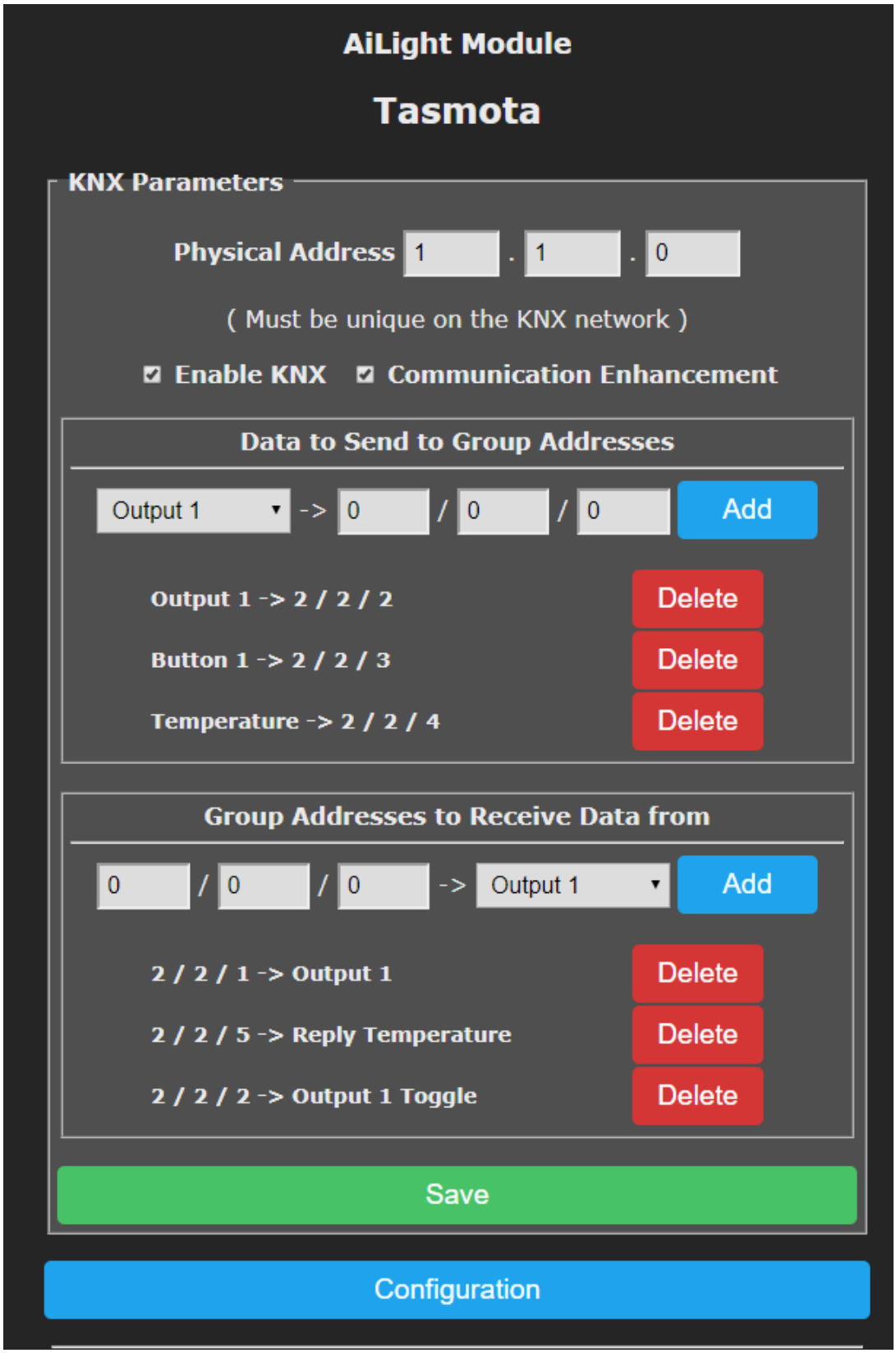


Figure 220 Tasmota KNX Test Configuration

The Tasmota device sends telemetry for the temperature sensor and ON/OFF changes as the power toggle button is used. This then becomes visible in mcsMQTT. Figure 221 and Figure 222 show this after the On/Off message was Associated to HS device 594.

Filter Association Table by Mqtt Topic and JSON Payload Key Clear Filters Rebuild Filters

T1	T2	T3	T4	T5	T6
knx					

J1	J2	J3	J4	J5	J6

Show Selected Associations

Prev 0 of 4 Next

Association Table for Auto Association of MQTT Topic and HS Device

/\	O	r	e	a	ref	TOPIC	payload	h	d	lastdate
0	☐	☐	☐	☑	593	knx/2/2/2				
1	☐	☐	☐	☑	594	Dev: knx|2|Tasmota_Light_Status_1-1-0 Sub: knx/2/2/2/Tasmota Light Status 1-1-0 Pub: the following Topic on Device command knx/2/2/2/write	true	☐	☐	2020-02-29 14:43:42
2	☐	☐	☐	☐		knx/2/2/4				
3	☐	☐	☐	☐		Sub: knx/2/2/4/Tasmota Light Temperature 1-1-0 °C	25.5	☐		2020-02-29 14:45:03

Figure 221 Association Tab for KNX Test Messages

Display Filters: Floor Room Device Type Show All

	Ref	Status	Category	Floor	Room	Name	Last Change	Control
☐	593			knx	2	knx-2-2-2	Today 12:43:10 PM	
☐	594	false		knx	2	Tasmota Light Status 1-1-0	Today 3:12:01 PM	false true

Figure 222 KNX Device in HS

When the HS true button for device 594 is pressed then a MQTT message is sent on Topic knx/2/2/2/write by mcsMQTT which is routed by the MQTT Broker to KNX-MQTT-Bridge. The bridge translates this into a KNX protocol and provides feedback in the console window. This is the white portion of Figure 223. The black portion of this figure shows the feedback as the Tasmota device provides an update of the temperature sensor reading over the KNX network and it forwards this via MQTT where it is received by mcsMQTT.

```
silly: Received MQTT message on topic knx/2/2/2/write with value true
verbose: Parsed MQTT message into gad 2/2/2 with command write, value true and dpt undefined
silly: onKnxEvent GroupValue_Write, 2/2/2, true
verbose: 2020-02-29 22:43:42 **** KNX EVENT: GroupValue_Write, dst: 2/2/2, value: "{\"value\":\"true\",\"name\":\"Tasmot
a Light Status 1/1/0\",\"unit\":\"\"}"
silly: onKnxEvent GroupValue_Write, 2/2/4, 25.5
verbose: 2020-02-29 22:45:03 **** KNX EVENT: GroupValue_Write, dst: 2/2/4, value: "{\"value\":\"25.5\",\"name\":\"Tasmot
a Light Temperature 1/1/0\",\"unit\":\"°C\"}"
```

Figure 223 KNX-MQTT-Bridge Terminal Window Feedback

The KNX network has a routing node similar to the MQTT Broker or xAP hub. This is normally a dedicated piece of hardware/firmware. For my testing I used software emulation that I installed on a RPi/Buster. It is available at https://github.com/knxd/knxd. Those which already have KNX network installed will have this router already installed.

With this test setup the Toggle button on Tasmota could be used to change the relay state and this updated state was reflected in HS device. A change of the HS device would change the relay and Tasmota status to complete the bidirectional communication.

21 Applications

21.1 Applications with Tasmota

The Sonoff Basic from ITEAD is one instance of an ESP8266 application. Multiple software packages are available including Tasmota that has been used in this case. ITEAD makes several other devices that Tasmota can be easily installed. Other manufactures also produce ESP8266-based devices. The dominant firmware in these devices is Tuya, also known as Smart Life. The firmware on these canb e changed to Tasmota using Tuya Convert application on RPi. Development boards such as Wemos D1 Mini or NodeMCU provide a more generalized microcontroller interface application but do not have the backaging or power supply built in.

21.2 Sonoff Basic (Original Version) Firmware Upload

There are two means to upload software to the Sonoff. Initially it is done via 3.3V level serial with USB/Serial adapter. Make certain it is at 3.3V levels and not 5V or +/-12V. There are several tutorials on the web for this process. Essentially hold the black button down (Ground GPIO0) and plug in the adapter then release the button. Using the Atom/VSCode/Arduino environment upload the compiled project. I have been using Atom based upon the tutorial provided at https://www.youtube.com/watch?v=n4MDRm2yAJg.

After the binary has been flashed then it needs to be configured. The easiest way, if not already done be editing user_config.h before building the binary, is to use a serial connection and a program like Termite (https://www.compuphase.com/software/termite-3.4.zip) to provide commands for configuration. The baud rate should be set to 115200. The list of available commands is at https://github.com/arendst/Sonoff-Tasmota/wiki/Commands with the most important ones for the initial install being SSId <ssid> and Password <pwd> to establish a Wifi connection. After this is done then a browser can be used to configure the module to establish the MQTT connection and other basic setup.

Another approach to initial configuration is by quickly pushing the Sonoff button 4 times and looking for a new SSID available on your computer's Wifi connection. The Sonoff will be serving HTTP from 192.168.4.1. Connect to this new Wifi SSID and put 192.168.4.1 in browser URL to get the config page. Setup the desired WiFi SSID and password that is used for your network. After it is saved it will restart and try to connect to this SSID. It may require a power cycle if auto-reset is not successful.

Once the browser connection has been established the Console of the Tasmota page can be used to enter commands just like was done with serial connection. It is also possible to batch together a set of commands using ";" separator. This way a text file can be saved and pasted into command line.

After the initial upload then it can be done more easily and quickly with Wifi connection from the browser that is running the installed Tasmota software on the Sonoff. This is especially true for firmware updates that can be done via Wifi rather than a serial connection.

The firmware, which is the same Tasmota-variant firmware described for devices in this manual is at http://mcsSprinklers.com/mcsTasmota.zip. This same firmware can be used for all other applications of the Sonoff devices in this manual as of June 2018.

The source, based upon Tasmota 5.9.1 can also be made available for anyone interested. There is also a firmware based upon Tasmota 6.0.0a that is available under mcsTasmota6 http://mcsSprinklers.com/mcsTasmota6.zip. This contains only the addition of the irrigation control logic.

These zip files contain two binary (.bin) files. One is a small image designed to only serve as a bootstrap to load the other binary. Sonoff units typically contain 1024K flash. To do OTA the running program needs to leave enough of the flash space to hold the new binary. Tasmota6 is over 512K so the minimal must first be installed and then when running the minimal image the desired one can be installed.

There appears to be a configuration layout change between 5.x and 6.x Tasmota. What I have found is that going from 5.x to 6.x the minimal image for 5.x should first be flashed. This will retain the WiFi settings in the minimal so it will run without any additional configuration. Once 5.x minimal is installed then the full 6.x binary can be installed. The settings then need to be edited. It may be possible to backup and restore settings, but I have not attempted this.

The failure mode I ran into when trying to toggle between 5.x and 6.x is that the SSID will be blank so WiFi connection lost. In my case I flashed via Serial the new image when this happened, but it should be possible to use Termite.

21.3 WiFi Garage Door Control

21.3.1 Original GDO Tasmota 5.9.1

Modification to version Tasmota 3.9.1 was made to support a Wifi Garage Door interface. In this case the Garage Door is characterized by a pushbutton to toggle the door from the open to close position and two closed-contact sensors to become active when the door is in the open and closed positions. The Sonoff relay is used for the pushbutton to model this and two of the discrete inputs used for the two door position sensors. The Tasmota software was updated to recognize the status of the two inputs to be OPEN, CLOSED, INDETERMINATE and OPEN&CLOSE, to specify the wait time expected for the door to get from open to closed, and a four time retry when the desired state is not being achieved.

The user control to open or close the garage door is with HTTP, MQTT or Echo/Alexa. The desired position is commanded (ON=Open, OFF=Closed) and the Sonoff will use the relay to command the pushbutton as necessary to achieve the desired state. If the door is already at the desired state when a new command is received then no pushbutton action is commanded.

The status of each event is provided with MQTT including when the pushbutton transitions and when the door position reaches a new state. If the desired state is not reached after four retries then a FAIL Topic will be reported. OPEN&CLOSED state is invalid and indicate a hardware malfunction. INDETERMINATE are expected when the door is moving, but if it remains after the timeout period then it is also an indication that some service is needed such as removing obstructions or again a hardware failure.

Normally the Sonoff Basic relay is used to control power line load. A cut and jumper are used to disconnect the line power from the relay and route the output connector to the two relay outputs so it can be used as a dry contact. The red oval in Figure 224 shows where a hacksaw was used to cut the board to sever the two thick soldered runs and then a small wire soldered between the two cuts. When making the cut the saw blade can be guided by the edge of black relay next to the resistor next to the relay. Make certain to cut deep enough to where the ½ watt resistor is soldered on the board top side to assure the relay connection to the main voltage is fully severed. Use ohm/volt meter to confirm no mains power is connected to the relay.

The yellow circle shows where an additional input can be wired by soldering directly onto the corner pin of the ESP8266. This is GPIO4. It is not necessary as other inputs available on header pins can be used. At the time of construction, it was not clear if the one normally used for serial port Tx could be used for this purpose and still retain the debug environment. It turned out to be available. Soldering directly onto the small pin is difficult and needs very low gauge wire to properly transfer heat for a good solder connection. Note also that the 10K pull-up resistor already connected to GPIOI14 via surface-mount is not done for GPIO4 so somewhere a 10K resistor needs to be connected between the GPIO4 wire and 3.3V to act as a pull-up to have a positive inactive signal input.

All wires were routed using Cat3/5 to a short pigtail with RJ11 connector. A mating RJ11 is used on this install. This provides for an easy disconnect should it be necessary for later maintenance. In the figure the orange GPIO14 and two white (orange/white & green/white) ground wires can be seen. The GPIO4 was merged onto the green. The blue and third white (blue/white) is connected to the two output pins which are now connected to the two relay outputs. All wires were then routed to the top of the board

where they exit the case under the strain relief. Note also the two dabs of hot glue on the small GPIO4 wire to provide strain relief on the fragile direct solder connection.

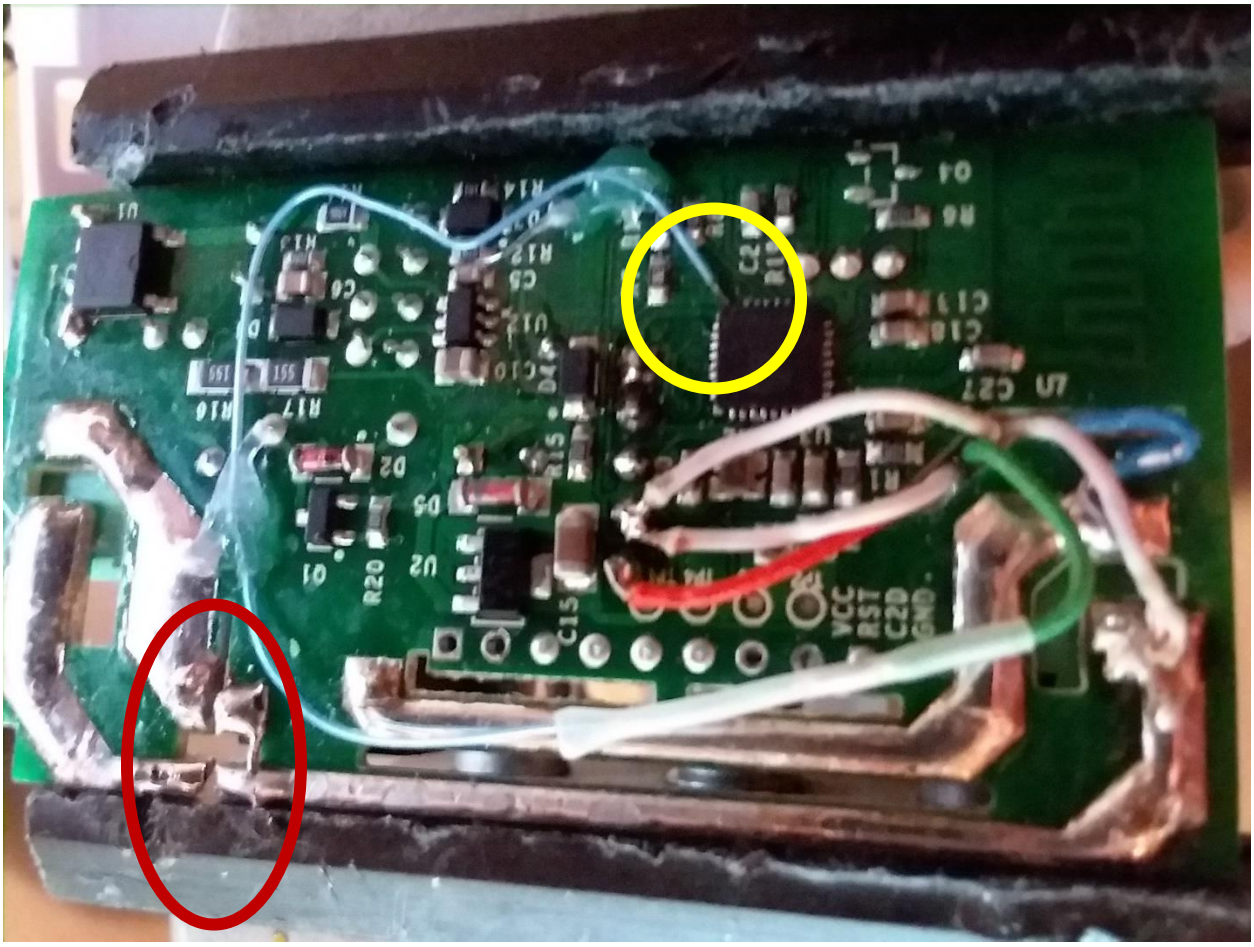


Figure 224 Sonoff Hardware Modification

A browser interface is used to setup the environment parameters. Figure 225 shows the Tasmota software configured as a Sonoff Basic and Switch 1 and Switch 2 are used for the CLOSED and OPEN status inputs. Note in this figure it shows GPIO3 as the OPEN status input, but with the hardware configuration shown in Figure 224 the GPIO4 rather than GPIO3 would be selected.

The input discrete name will be the last part of the Topic when the status of the two discrete inputs change. The relay pulse duration is the time the simulated pushbutton is depressed to activate the garage door motor. The max time is the length of time it takes for the door to move from open to closed plus some margin.

Under Figure 225 the circuit board top side documents the pin order where the connections are made with the built-in pushbutton included to provide orientation reference. Note that this figure was captured with first generation modification of Tasmota. Current mcsTasmota firmware will use 56 DoorOpn and 57 DoorClsd as the two sensor input selections.

The MQTT setup is shown in Figure 226. The MQTT broker is the IP or name for Host. The first part of the MQTT Topic is the Topic entry. Figure 227 is the Other Configuration page where MQTT is enabled, and the Echo/Alex information is entered.

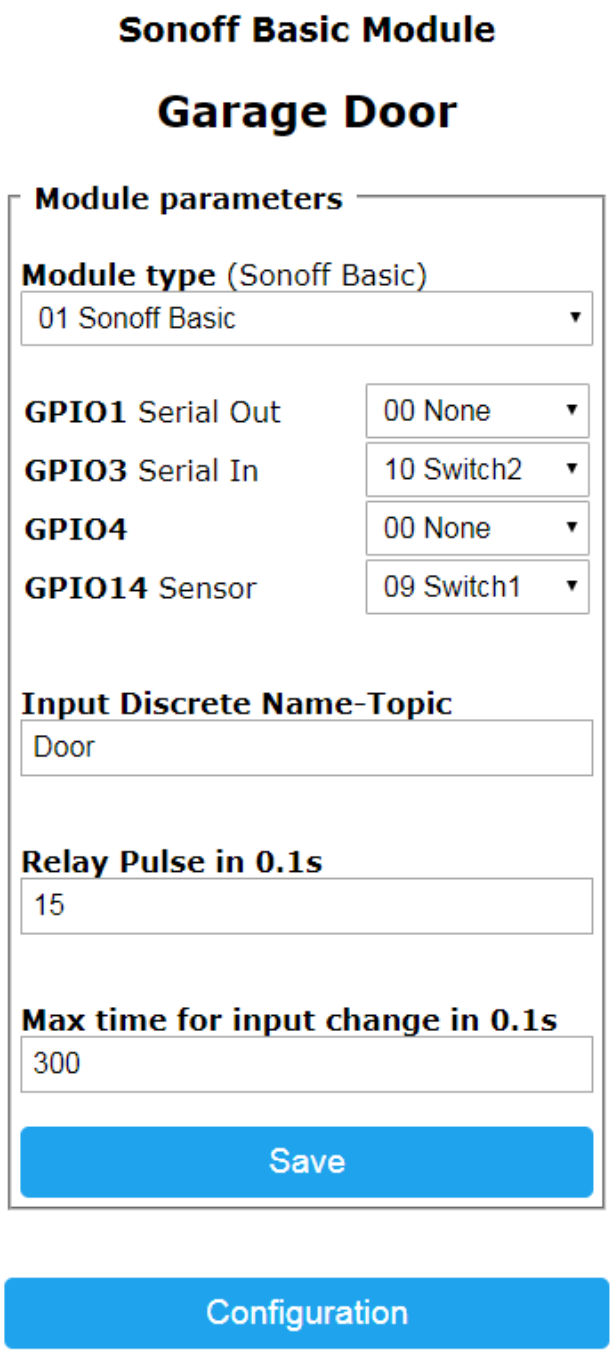


Figure 225 Module Configuration

On Board Interface Pins & Controls

GPIO-0 Button (used for boot control and manual pushbutton input)
Vcc 3.3V (for development testing power)
Door Open Input [Switch 2] (Door confirmed open when input at ground)
Serial TX (for serial monitor debug)
Ground
Door Closed Input [Switch 1] (Door confirmed closed when input at ground)

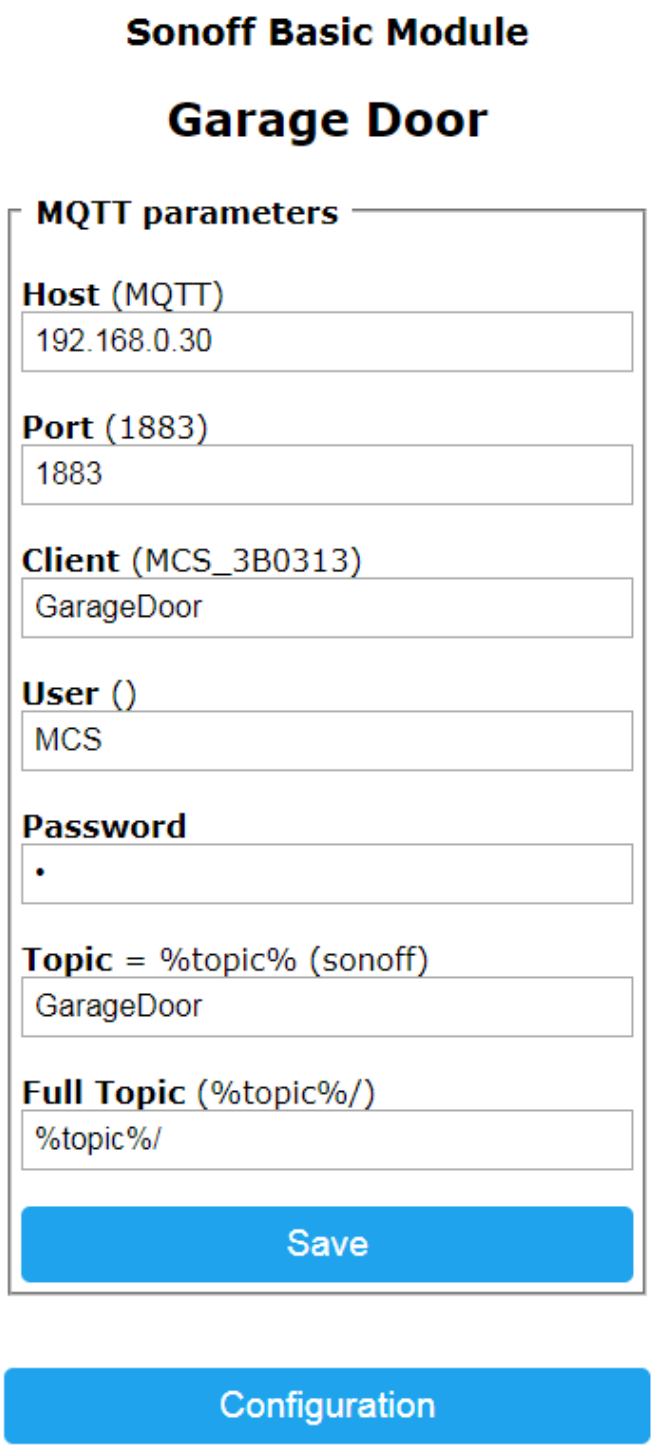


Figure 226 MQTT Configuration

```
MQTT Control
GarageDoor/cmnd/Power 1 – to open garage door (ON payload can also be used)
GarageDoor/cmnd/Power 0 – to close garage door (OFF payload can also be used)
```

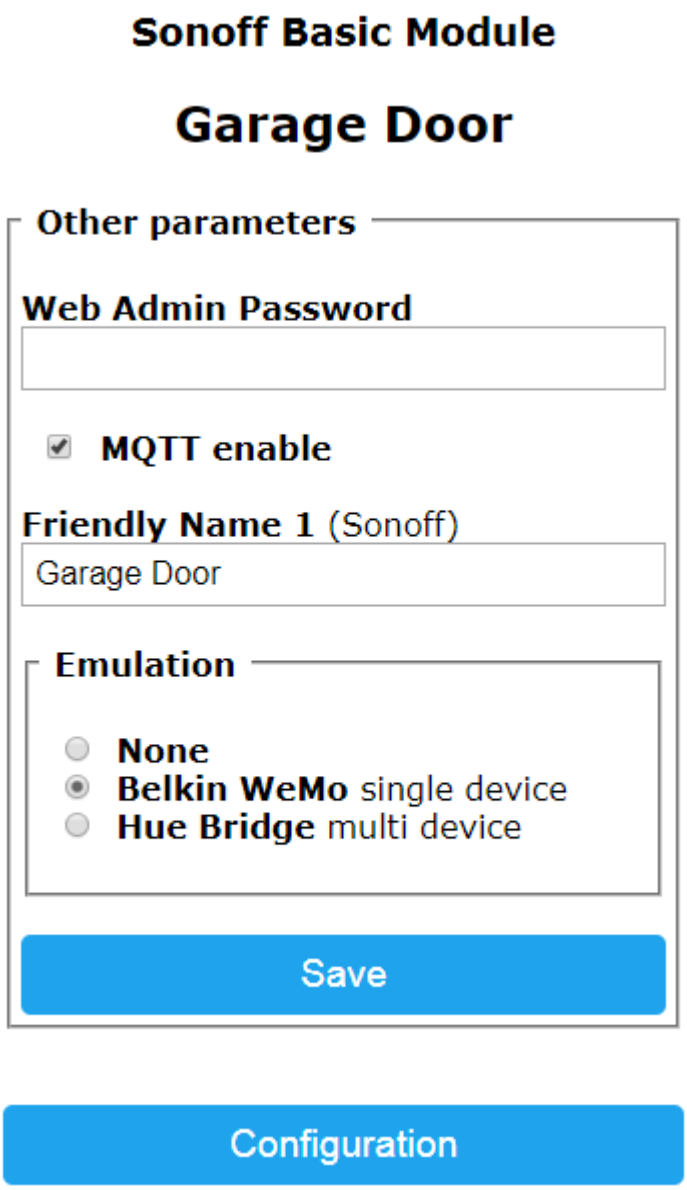


Figure 227 Other / Echo Configuration

Friendly name is the Echo Alexa name

```
Echo Control (native)
Alexa garage door ON (to open garage door)
Alexa garage door OFF (to close garage door)

Echo Control (routine in Alexa App to recognize Open and Close and control the garage
door device)
Alexa garage door Open (to open garage door)
Alexa garage door Close (to close garage door)

When putting the modified Sonoff into service for Garage Door control the relay and two
discrete inputs were soldered and routed to a RJ11 connector to provide ease of install
```

and later maintenance. The relay output connector on the Sonoff was no longer used. Figure 228 shows the installation provisions that were made. For the RJ11 the following was used, but any layout that has same wiring through the connector can be done.

RJ11 Pin	Wire Color	Function	Wire to Garage Door
1	Orange/White	GPIO14-Gnd	Black - Common
2	Orange	GPIO14	Red – Door Closed Contact
3	Green	GPIO4	Red – Door Open Contact
4	Green/White	GPIO4-Gnd	Black – Common
5	Blue	Relay	Yellow – Common
6	Blue/White	Relay	Green - Pushbutton

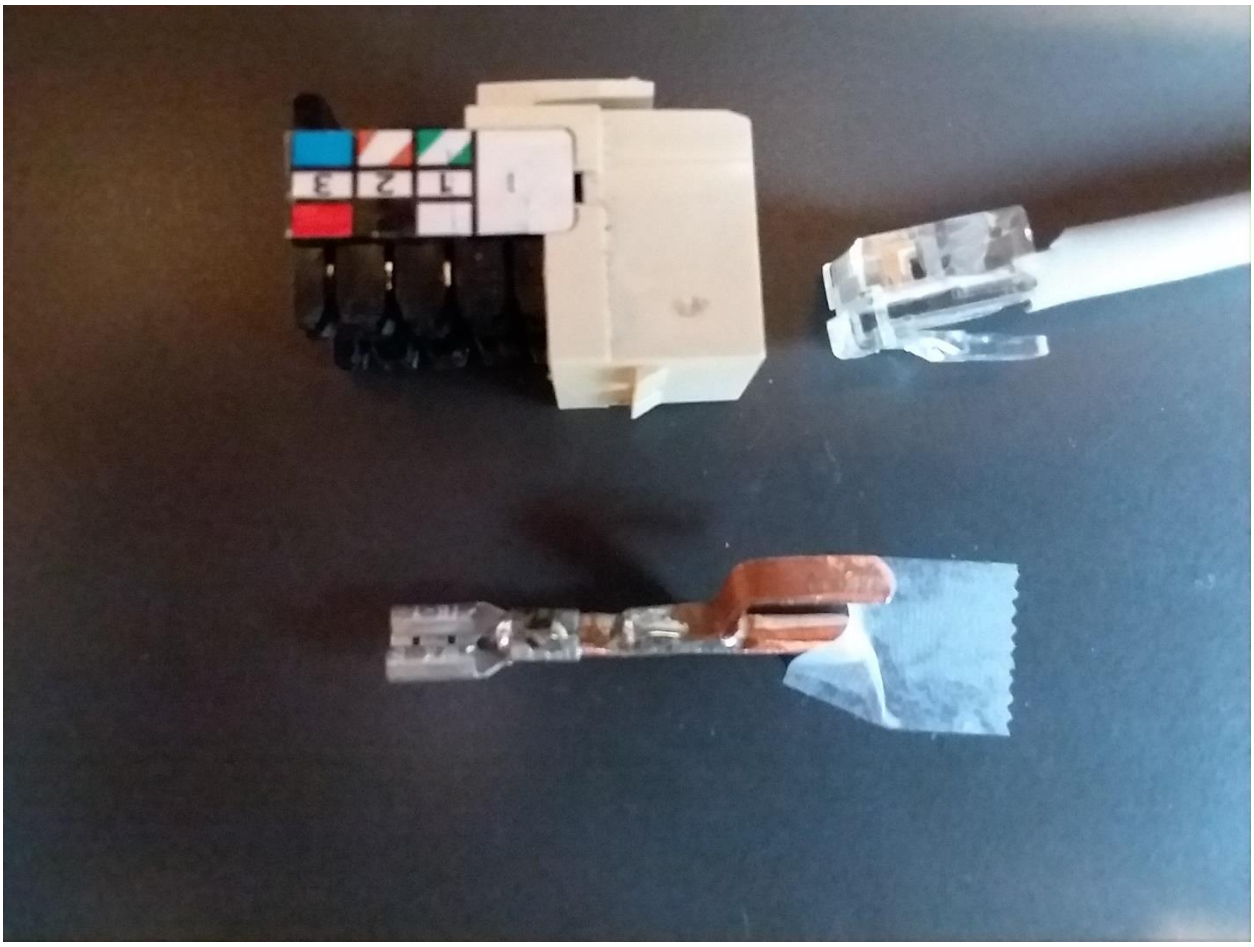


Figure 228 Garage Door Installation Connections

Also shown in Figure 228 is an adapter that was made from a copper 1/2 inch plumbing pipe to provide slip-on contacts to the two Door-Open and Door-Closed switches on the Garage Door. Cat 5 was then used between the pushdown connection on each pins 2 and 3 to a spade connector similar to the left side of the fabricated adapter. There may be commercial versions of this adapter, but I did not find them locally so just cut two short length of the copper pipe, bend one piece to an S, soldered and filed edges smooth.

21.3.2 Updated GDO Tasmota 8.4.0.3

The garage door control has been ported to Tasmota version 8.4.0.3 using features built into Tasmota rather than a hack of the Tasmota code. The same functionality exists as in the original version, but has been modernized for long term maintenance. There is no change in the hardware interface. Binary for this version is at http://mcsSprinklers.com/mcsGDO_8403.zip.

The following console/MQTT configuration options are utilized

Pulsetime1 <#> -- duration of the pushbutton press in 0.1 seconds (default 10 if not set)

Pulsetime2 <#> -- time allowance for door to open or close in seconds+100 (e.g. 130 is 30 seconds)

Poweronstate 0 -- forced by firmware to 0 so pushbutton is off on power up

Switchmode1 2 (invert) - set if open switch status is ground when active

Switchmode2 2 (invert) - set if closed switch status is ground when active

SetOption114 1 -- to enable the GDO functionality

SetOption115 1 -- to prohibit Echo control of "Open Door" or "Door On", Close/Off control not affected

The two Pulsetime options replace the hacked browser interface for setting up the timing for the door control. Switchmode parameters are only needed if the switch input is at ground potential to indicate an active state of opened or closed.

SetOption114 must be 1 to enable the GDO logic. If left at 0 then the relay control will follow normal Tasmota logic.

SetOption115 affects the WeMo emulation. If set to 1 then Echo can only be used to close the door. Any open attempts with Echo will be ignored. If left at 0 then both directions of control via Echo are enabled.

The new SetOptions will not be found in the online Tasmota command list. No changes made to the other commands other than forcing poweronstate to off and power control is not a relay control, but abstracted to represent a desired position of the door.

power ON/1 command is used to open the door
power OFF/0 command is used to close the door
power TOGGLE/2 command will command door to opposite position. If current position cannot be sensed then it will act like OFF command.

The actual control of the pushbutton can be tracked with the RESULT or POWER topics

Status is reported in the SENSOR topic. It will have JSON entries for the two switches and a 4-state composite for the door position such as below
.....MQT: GarageDoor/SENSOR = {"Time":"2020-08-29T21:52:47","Switch1":"ON","Switch2":"OFF","Door":"Closed"}

STATE topic is unchanged with info about the device.

The GDO logic is also designed to work with only one switch to detect open postion. Use the switchmode command to assign the correct polarity for this switch. It is actually a door closed switch then reverse the polarity from what would be the case if is a door opoen switch. Do no define only Switch2, but define Switch1 for the case of a single switch sensor.

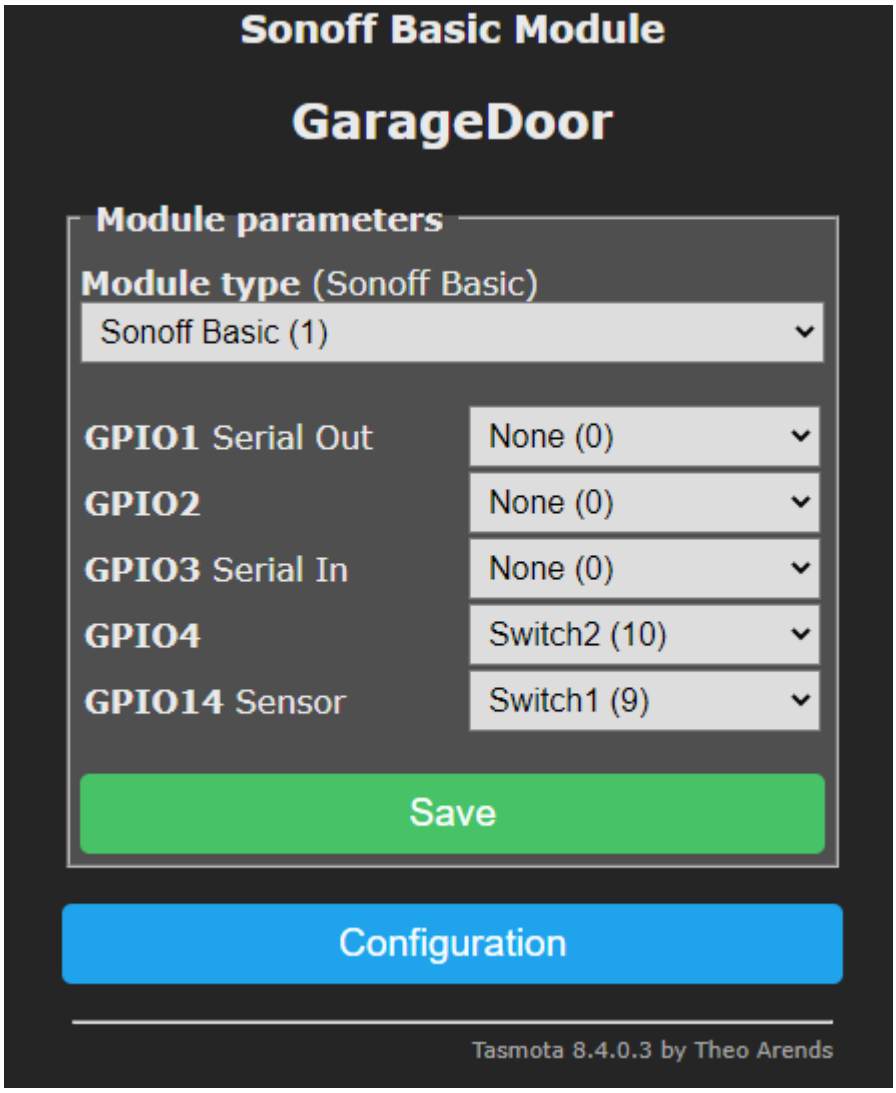


Figure 229 GDO GPIO Setup

21.4 Pulse Counter

I have been using DS2423 1-wire counters to count pulses for measuring things such as water meter. The stock Tasmota firmware was used with a Wemos D1 Mini to satisfy this need. This installation actually used one input as a counter and one input as a switch. The counter was for gallons of water use. The switch was for indication if fireplace fan turned on or off. Configuration is show in Figure 230.

Low pass filter signal conditioning on the GPIO12 switch input was provided by 15K resistor attached to 3.3V and 2200 uF capacitor to ground. This filtering was also done using Tasmota rule for both inputs. Without this conditioning the wire run between the fireplace fan control and the Wemos D1 Mini would pick up extraneous noise spike inputs.

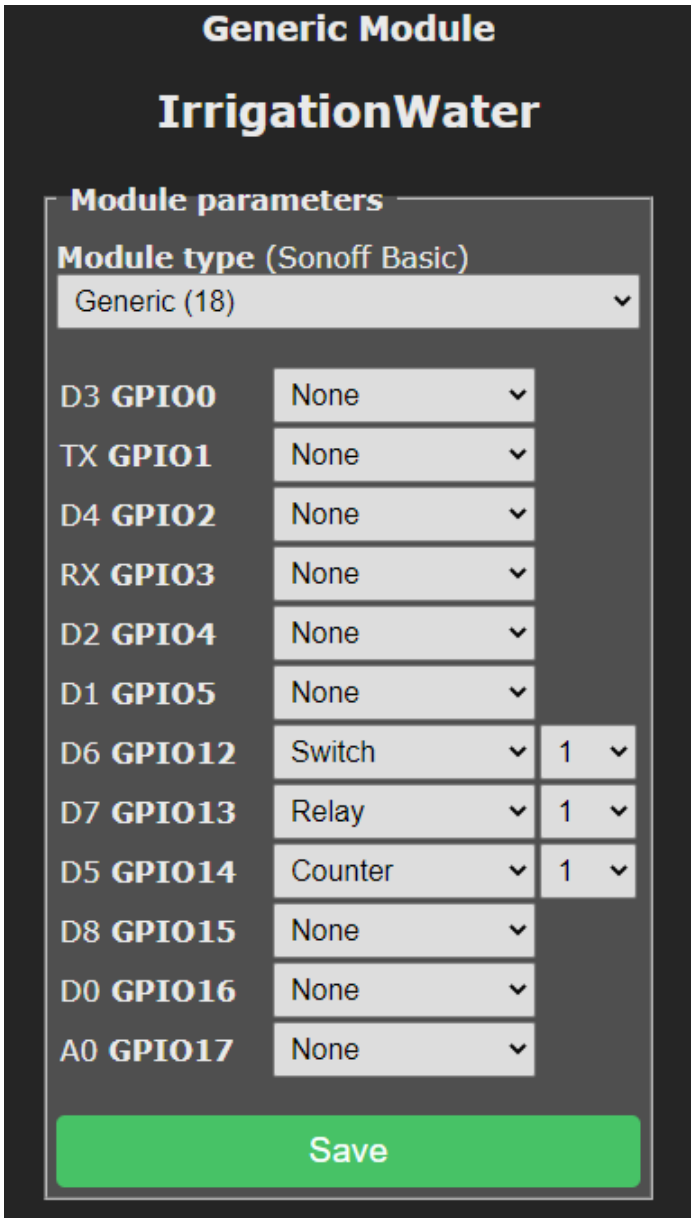


Figure 230 Counter Module Configuration

Rules were used to augment the counter and the notification inputs. At boot the counter was configured and debounce set for inputs. At midnight reset the counter. On change of the fan switch input publish MQTT to IRSend device to bump up or down the TV volume to account for fan noise.

Rule1

Rule1 ON system#boot DO backlog CounterType1 0; CounterDebounce 500; SwitchDebounce 1000 ENDON ON Time#Minute=001 DO counter1 0 ENDON ON Power1#State=1 DO publish IRSend/cmnd/Power2 0 ENDON ON Power1#State=0 DO publish IRSend/cmnd/Power2 1 ENDON

21.5 Low Volume Water Flow

A low cost ($10) water flow meter is available from Digiten that is interfaced to with a 5V supply and a square wave output toggling between 0 and 5V as water flows through it. Two challenges exist to use Sonoff Basic to interface this sensor. One is that 5V is needed to support the sensor and the other is that the input provided by the senor will reach 5V. The Sonoff working voltage is 3.3V.

The first issue is resolved is by picking off the voltage from the Sonoff power supply before it is regulated. The Digiten has a wide voltage range so anything available between 4 and 12V will suffice. It turns out that 5V is available with the pickoff point shown on Figure 231 Sonoff 5V pickoff. Note board in this figure has been modified to support dry contact relay rather than mains voltage across the relay, but it does not matter for this application.

A voltage divider with 4.7K ohm resistors was used to address the sensor input voltage.

This sensor has three wires and the nomenclature is printed on the device. The red is the 5V. Black is common ground. Yellow is the pulse output which floats up to the 5V and is pulled down to ground. The 4.7K resistors are wired in series between the 5V and ground thus producing 2.5V at the point between the two resistors. The yellow wire as well as the GPIO14 from the pin header of the Sonoff is connected to this central point between the two resistors. When the Digiten output floats high the voltage remains near 2.5v. When the Digiten output is pulled to ground this central point is also pulled to ground. This means that the Sonoff GPIO14 will see either 0 or 2.5V which are acceptable inputs for the ESP8266.

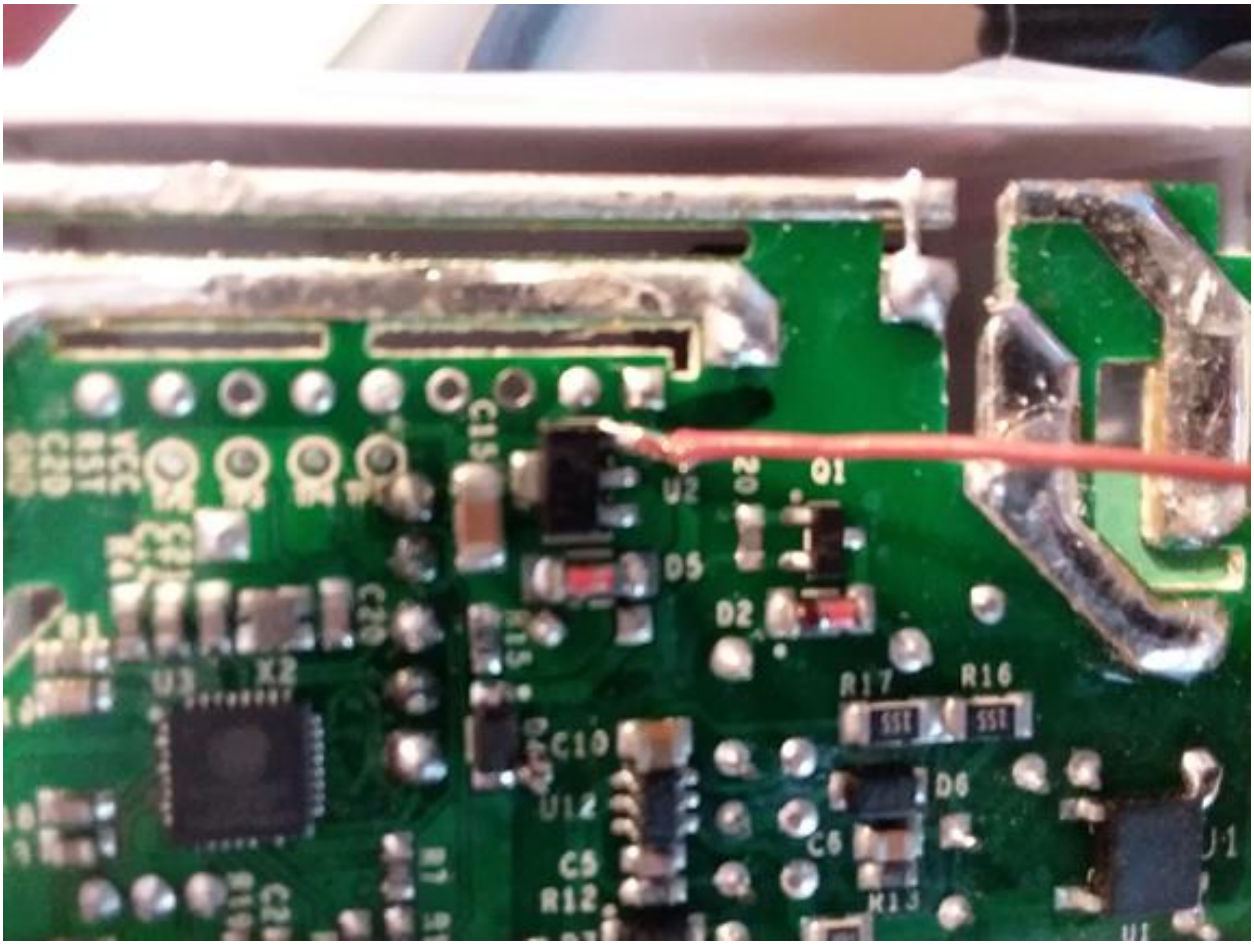


Figure 231 Sonoff 5V pickoff

The configuration of the Tasmota firmware is the same as the configuration used for the pulse counter described in Section 21.4 except Figure 230. This water usage counter has a different gallons/pulse relationship. In this case I elected to use Topic of Ounces. Each pulse (count) represents 0.87 ounces of water flow. This will need to be calibrated for each user's situation by running a know amount of water through the sensor and determining how many counts were recorded.

Expect about 5% accuracy depending upon flow rates. I selected the model FL-S402B that has a working range of 0.3 to 10 liters/minute (10 to 338 ounces/minute) (0.17 to 5 ounces/second) which is in the range of the osmosis system I am monitoring.

Figure 232 shows the configuration. Note the main Tasmota Sonoff page will show the counts before any scaling performed. The MQTT topic will be transmitted with the scaled value.

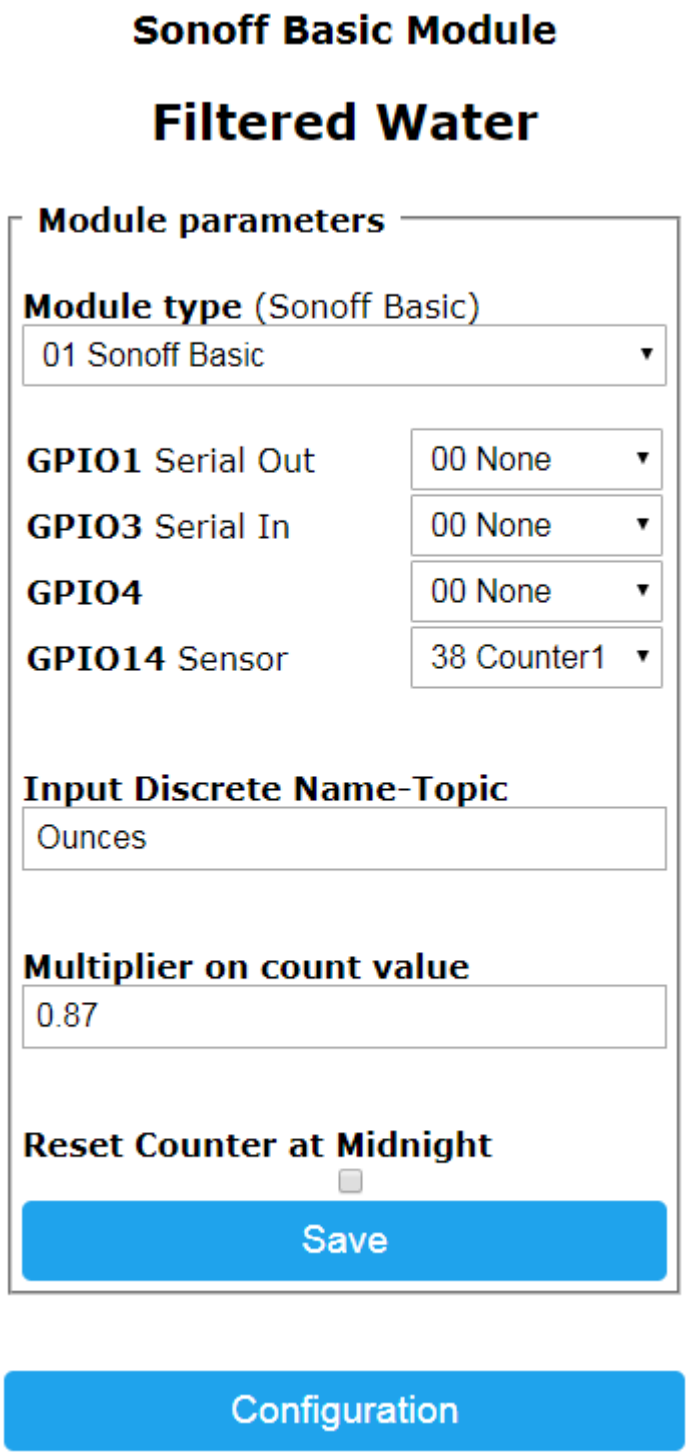


Figure 232 Filtered Water Flow Calibration Seteup

21.6 Multiple Light Control on Single Switch

One wall toggle switch is wired to three ceiling lights each illuminating a different part of the room. The desire is that each light have an independent control, but it is not possible to rewire with three independent toggle switches. To deal with the problem a Sonoff 4CH Pro has been deployed with standard Tasmota firmware.

Access to the light cans is available to allow separation of the line power to each light. The power line that was originally daisy-chained between each light can is routed to the power input of the 4CH. The light power wire at each can was removed from the daisy-chain and separate wire added and installed in each of inputs 1, 2 and 3 of the 4CH. The common return also was routed to the 4CH to complete the power circuit.

The master toggle switch providing power remains in the ON position for normal operation. To assure that the toggle switch works as before the automation addition it is necessary to have the 4CH turn each light on when initially powered. This means that toggle switch provides the OFF control by removing power and Sonoff 4CH provides ON control by Echo or MQTT. The default Tasmota configuration is that relay return to the state they were that last time power existed to the 4CH. This can be changed to always initially turn relays on by changing the user.config prior to initially flashing or by sending MQTT message. I used the MQTT approach by publishing one time:

```
"FamilyLights/cmnd/PowerOnState" with payload of “1”
```

The same firmware used for other Sonoff devices was installed and configured to be a 4CH. It was configured to be a 4CH and MQTT parameters setup to provide the desired control as shown in Figure 234 and Figure 235. Control is also available via Amazon Echo using the Phillips HUE emulation within Tasmota.

Upon testing it was discovered that Echo control was not successful. Google research identified that the issue was with Amazon with lack of HTTP protocol adherence. A workaround was found at the URL and content of the screen shot below. In essence it adds an OR condition when parsing the HTTP header.

Secure | https://github.com/arendst/Sonoff-Tasmota/issues/1639#issuecomment-360218536

OFreddy commented on Jan 24 • edited

With this **workarround** in parsing.cpp in the ESP8266WebServer of the esp8266 library (Arduino, V2.4.0, line 193) it works fine for me:

```
if(!isEncoded||(0==_currentArgCount)){ // @20180124OF01: Workarround for Alexa Bug
```

```
179     if (!isForm){
180       size_t plainLength;
181       char* plainBuf = readBytesWithTimeout(client, contentLength, plainLength, HTTP_MAX_POST_WAIT);
182       if (plainLength < contentLength) {
183         free(plainBuf);
184         return false;
185       }
186       if (contentLength > 0) {
187         if(isEncoded){
188           //url encoded form
189           if (searchStr != "") searchStr += '&';
190           searchStr += plainBuf;
191         }
192         _parseArguments(searchStr);
193         if(!isEncoded||(0==_currentArgCount)){ // @20180124OF01: Workarround for Alexa Bug
194           //plain post json or other data
195           RequestArgument& arg = _currentArgs[_currentArgCount++];
196           arg.key = "plain";
197           arg.value = String(plainBuf);
198         }
```

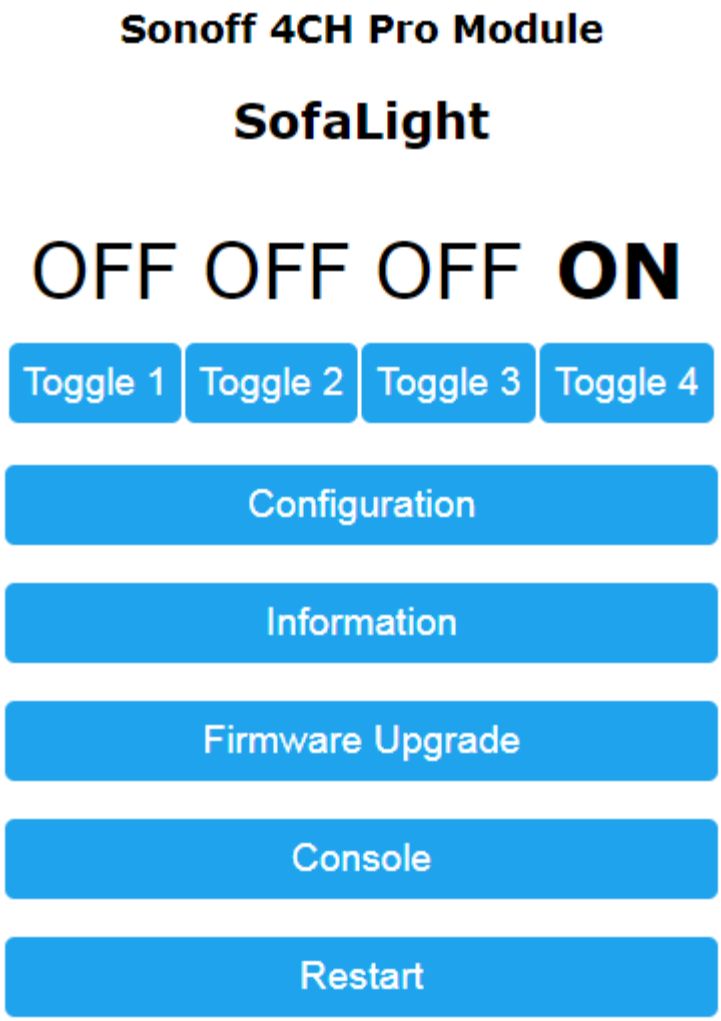


Figure 233 Sonoff 4CH Module Tasmota Main Page

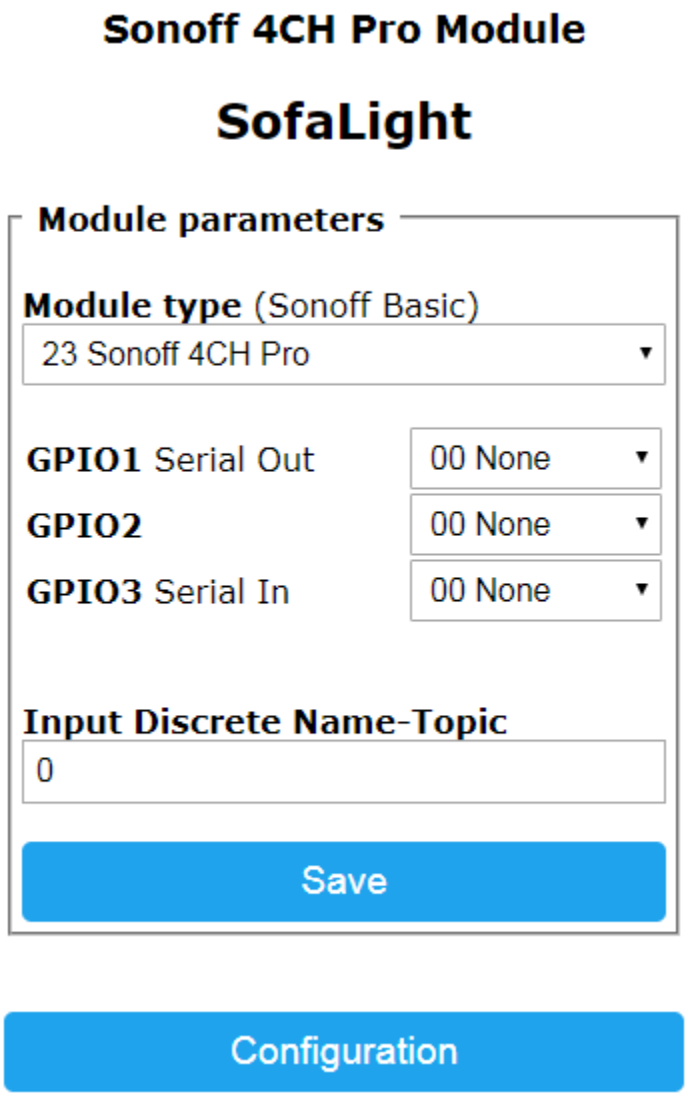


Figure 234 Sonoff 4CH Tasmota Configuration

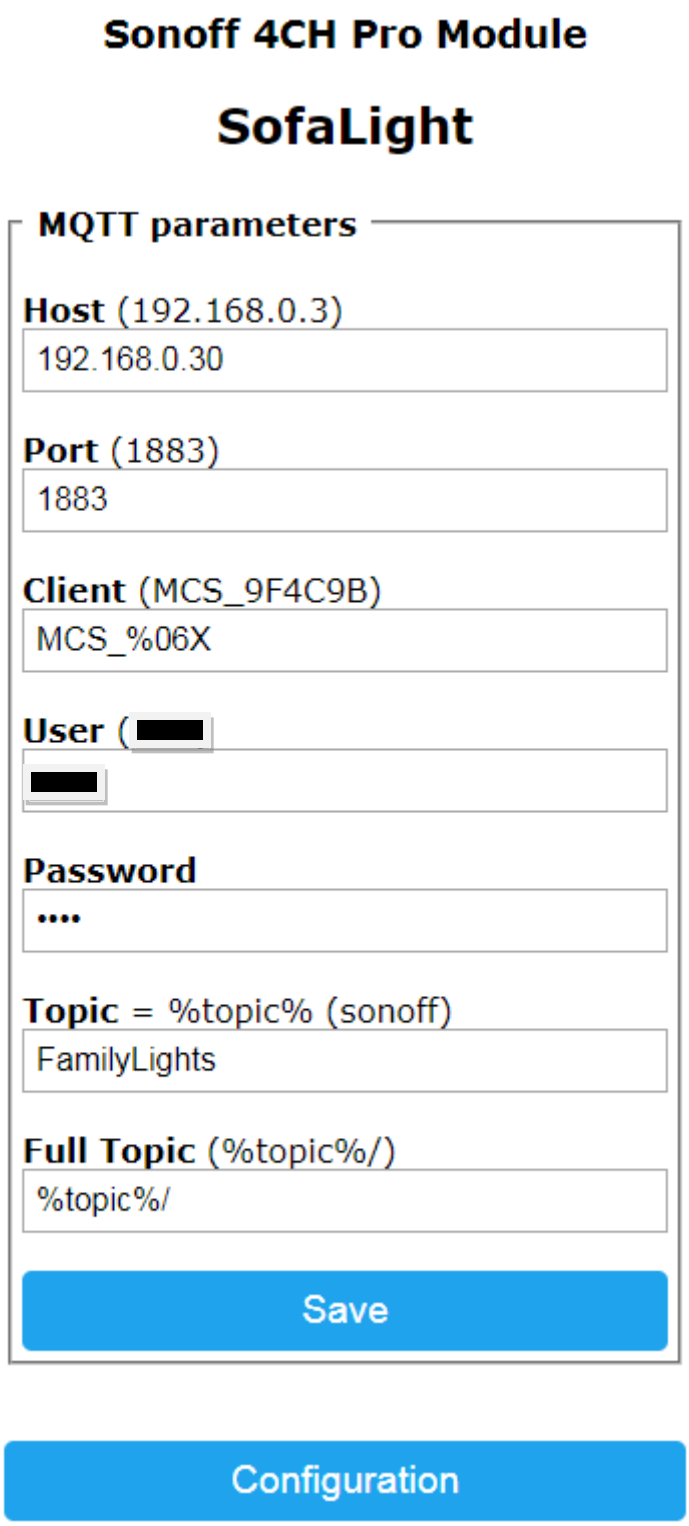


Figure 235 Sonoff 4CH MQTT Configuration

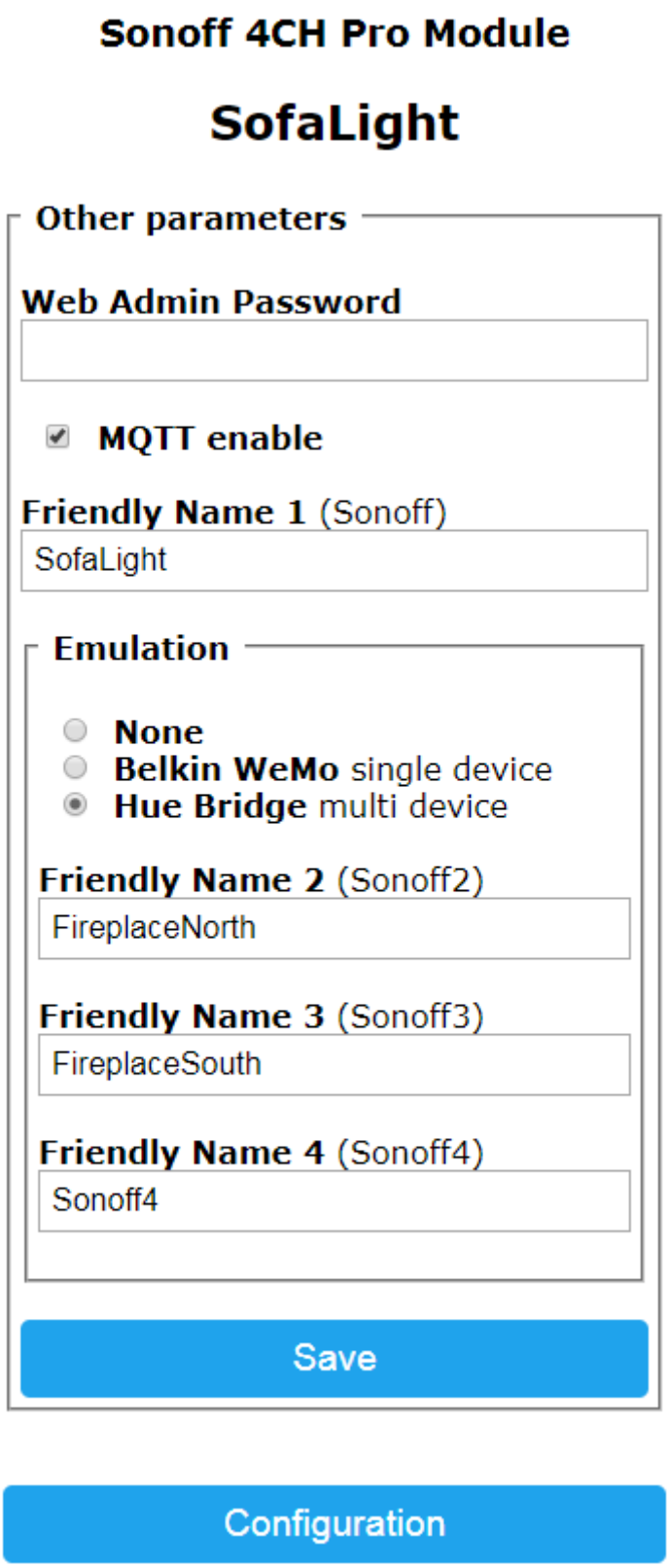


Figure 236 HUE Emulation Setup

21.7 Failback Irrigation Controller

Irrigation control can be performed with a timer, with a smart timer or with an automation system scheduler that controls relatively dumb relay-like valve controllers. The Sonoff devices with relay(s) can be used to control irrigation on a schedule and to be slaved to a more intelligent irrigation control system.

The concept of operation is that the Sonoff unit will respond to commands to turn a relay (valve) ON or OFF. The Sonoff will monitor the ON and OFF intervals and if either an ON or OFF command is not received in a programmed period of time then the Sonoff will perform the ON and OFF actions per the programmed interval.

The Sonoff 4CH Pro makes for a good four channel irrigation control system. Multiples of these can be used to expand beyond four valves. Many of the Sonoff units such as the Basic and the 4CH (not Pro) have internal connections so the relay is used to control mains voltage. If these are used then hardware modification is needed to isolate the relay from mains so 24VAC can be used with the relays. The Sonoff Basic relay isolation modification was described in Section 21.3.

The board of the 4CH Pro R2 is shown in Figure 237. In this case a 5VDC connection was made to a wall-wart in the upper right. During flashing an external power supply is desired to assure adequate current is available. In the lower left is the header that was soldered in place from which Gnd, Tx and Rx are connected to the USB/Serial adapter used for flashing and serial monitoring. The board also shows two LED being illuminated on the left. The red is in the Relay 1 position. The blue will illuminate when any of the four relays are ON. On the right of the board are four pushbuttons that can be used to manually control each relay. The 4CH Pro also has a RF connection to remotely control the valves. This is useful for field testing of the irrigation system. The RF use has not been attempted, but it may be necessary to pair the RF buttons with the unit prior to flashing Tasmota. Internet search should provide more information in this area.

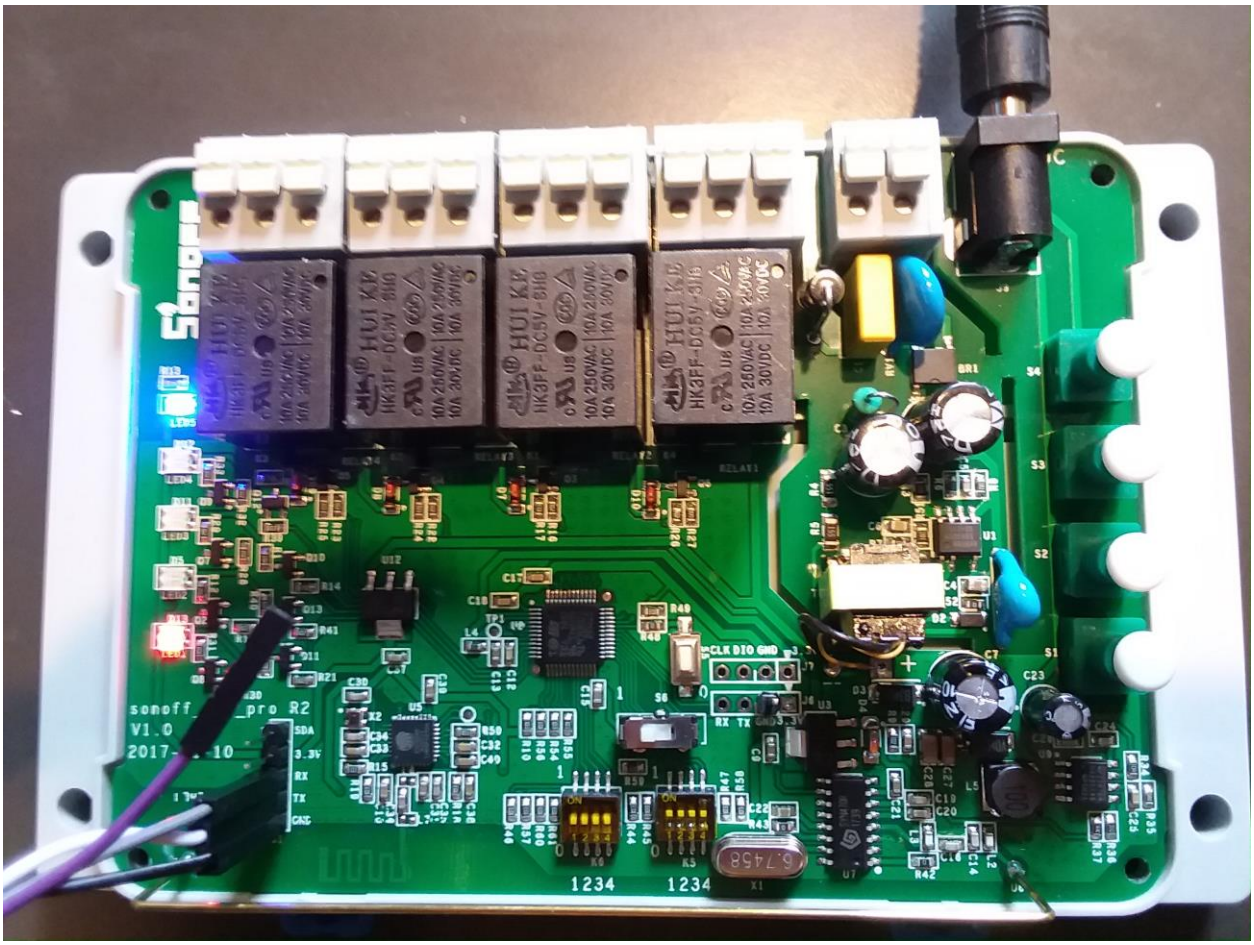


Figure 237 Sonoff 4Ch Pro R2 Circuit Board

The failback parameters and value control are communicated via MQTT. Normal valve control using a HS device is setup as shown in Figure 238. The devices can then be assigned to events or included in an irrigation schedule such as with mcsSprinklers.

Association Table for Auto Association of MQTT Topic and HS Device

	H	R	A	LastDate	Ref	Topic	Payload
0	☐	☐	☐	2018-05-16 13:12:23		Sub: IrrigationValve/LWT	Online
1	☐	☐	☑	2018-05-16 13:15:27	258	Dev: |IrrigationValve|POWER1 Sub: IrrigationValve/POWER1 Pub: the following Topic on Device command IrrigationValve/cmnd/POWER1	OFF
2	☐	☐	☑	2018-05-16 13:18:27	259	Dev: |IrrigationValve|POWER2 Sub: IrrigationValve/POWER2 Pub: the following Topic on Device command IrrigationValve/cmnd/POWER2	OFF
3	☐	☐	☑	2018-05-16 13:21:27	260	Dev: |IrrigationValve|POWER3 Sub: IrrigationValve/POWER3 Pub: the following Topic on Device command IrrigationValve/cmnd/POWER3	OFF
4	☐	☐	☑	2018-05-16 12:06:36	261	Dev: |IrrigationValve|POWER4 Sub: IrrigationValve/POWER4 Pub: the following Topic on Device command IrrigationValve/cmnd/POWER4	OFF

Figure 238 HS Device Setup for Irrigation Control

The failback, or monitoring parameters are also delivered via MQTT. The set of topics for monitoring are shown in the Publication List of Figure 240 and Figure 239. They consist of:

IrrigationOnMinutes (with numeric suffix for relay number if for specific relay on module)
IrrigationOffMinutes (with numeric suffix for relay number if more than one on module)
IrrigationNextMinutes (with numeric suffix for relay number if more than one on module)
IrrigationStartHour (with numeric suffix for relay number if more than one on module)
IrrigationMode (payload can be OFF/ON/AUTO/BLINK or 0/1/2/3)

The irrigation feature of mcsTasmota 5.9.1e and later is enabled by sending a topic or IrrigationMode. To remove the irrigation feature from the module the device needs to be reflashed with the memory erased so settings (including irrigation mode) are reset.

The Figure 240 failback schedule has been setup to run each valve for 30 minutes after a period of three days (3*24*60= 4320 minutes) without the valve having been turned ON. It also specifies that no valve will be turned ON before 6 AM. The last topic is to enable (or disable) the irrigation failback monitoring.

It was specified to run in IrrigationMode of AUTO/2 . This means that there will be a master irrigation controller that will be providing IrrigationNextMinutes topics with payloads that indicate the number of minutes in the future that the next irrigation cycle is expected to run. When the expected future time arrives one of four things will occur.

1. The normal situation is that the master controller will publish a cmnd/POWER=ON message and the relay valve will be turned ON.
2. The second is that the master controller has revised the number of minutes in the future and sent a subsequent IrrigationNextMinutes message.

3. The third is that the master controller failed and did **not** send the cmnd/POWER=ON messages. In this case the IrrigationOffMinutes value may become (see condition 4) the determining time when the relay is turned on by the firmware.

4. The last is the same as the third, but in this case the IrrigationOffMinutes has elapsed before the IrrigationNextMinutes expired and in this case the firmware will turn the relay ON when the IrrigationNextMinutes expires.

The firmware will operate without consideration for the IrrigationNextMinutes messages when IrrigationMode is set to ON/1. In essence there is no master irrigation controller available. The relay valve ON and OFF intervals are determined by IrrigationOffMinutes and IrrigationOnMinutes messages.

When IrrigationMode is set to BLINK/3 then it disables irrigation scheduling and reporting in the STATE message. BLINK is not intuitive, but is the equivalent command in Tasmota that gives a value of 3.

When IrrigationMode is set to OFF/0 then it disables scheduling, but still reports irrigation status in the STATE message.

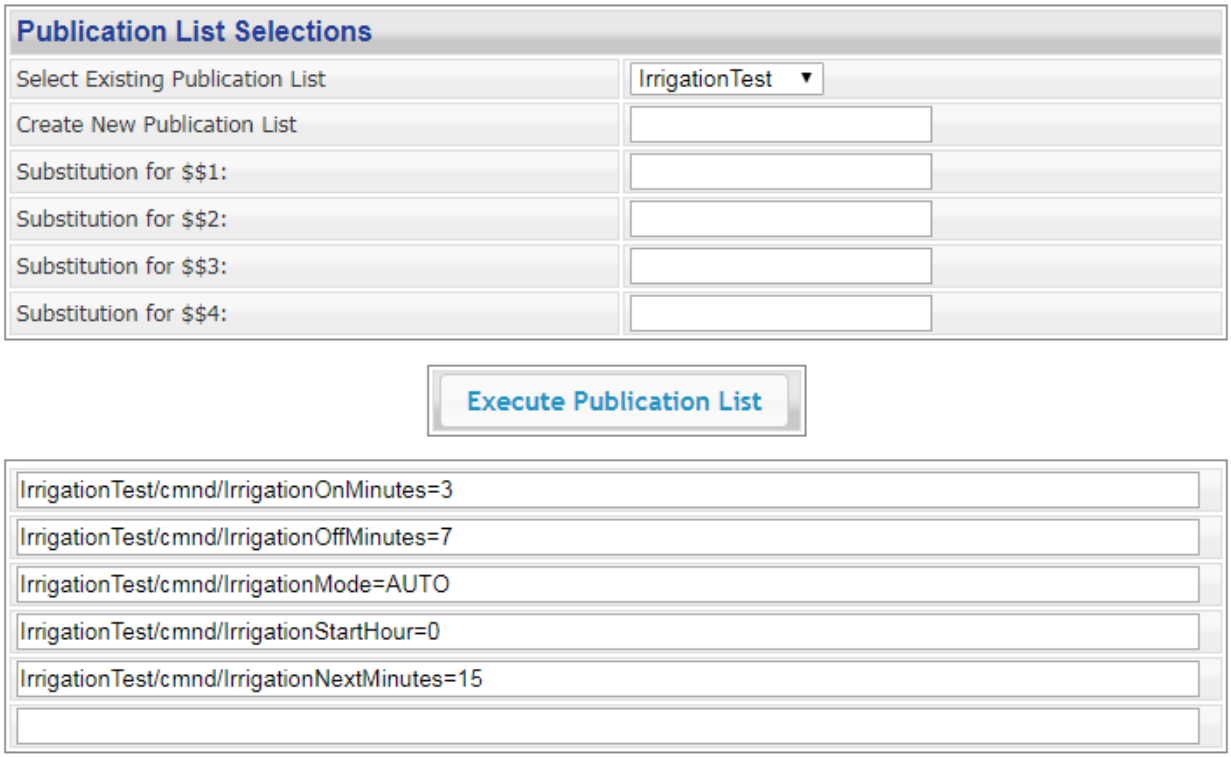


Figure 239 Sonoff Basic Setup Test for Irrigation Control

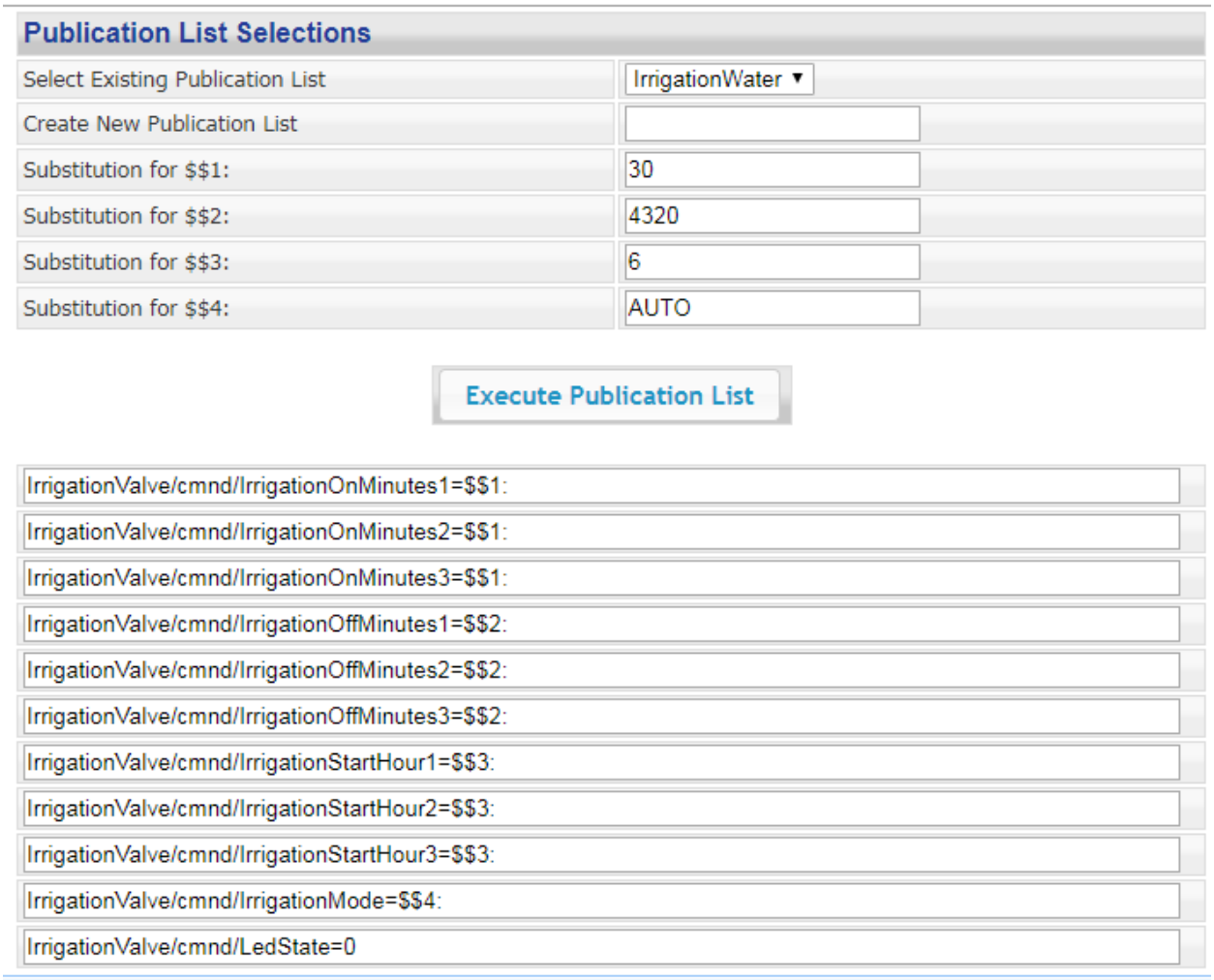


Figure 240 Irrigation Monitoring Topics

If no ON and OFF command is received before the Next and Off topics then the Sonoff will perform an irrigation schedule starting at 6 AM every 3rd day and run the valves in sequence for 30 minutes each. Each of these relays/values can be setup with different on, off, and start parameter values even though the sample shown uses the same values for each of the four.

Inputs such as rain sensors, water use counters or external enables can be wired if desired and values available via MQTT. They are not used as part of the failback monitoring for irrigation. On the 4CH Pro the GPIO2 input is available on the same header as the TX and RX and is labeled as SDA. GPIO1 and GPIO3 are also available after the flashing is complete and use of the serial connection is no longer needed.

In Figure 242 GPIO2 has been selected as Switch1. This switch has been implemented in the firmware as a local override of the Irrigation Mode such that when the input is grounded the irrigation mode will be considered OFF. Note that the Sonoff will not boot when GPIO2 is grounded at power-on so if this switch is used it needs to be set to enable irrigation when powered up. When the switch is toggled a MQTT topic such as “IrrigationEnable/cmnd/POWER1=ON” is published. The topic is based upon the

input topic entered in e 51. At this time and at the telemetry rate (default 5 mins) a state message will be published such as


```
IrrigationValve/STATE={"Time":"2018-05-31T11:09:59", "Uptime":0 0, "Vcc":3.458, "POWER1":"OFF",
"POWER2":"OFF", "POWER3":"OFF", "POWER4":"ON", "Irrigation":{  "IrrigationMode":"AUTO" ,
"Time1":"14", "Time2":"8", "Time3":"10", "Time4":"15"} , "Wifi":{"AP":1, "SSId":"U", "RSSI":78,
"IPAddress":"192.168.0.29", "APMac":"78:8A:20:84:48:1D"}}
12:10:22 PM Received
```


The “Irrigation:” section of the STATE message contains two parts. One is the irrigation mode. It is the composite of the mode requested via MQTT Topic and the Override switch input if it is used. The second are the expected number of minutes until each relay will turn ON based upon the current fallback parameters. 9999 is used for the case of the irrigation mode being disabled.

At the same rate a sensor message will be published such as:


```
IrrigationTest/SENSOR={"Time":"2018-05-31T11:05:22", "Switch1":"ON"}
```


The IrrigationOffMinutes, IrrigationOnMinutes, IrrigationNextMinutes and IrrigationStartHour commands can be used with or without suffix. If no suffix is provided then it applies to all relays in the module. If a suffix of “2”, for example, is provided then it applies only to the 2nd relay. It is possible to send a programming schedule where all relays are given the same value, then a subsequent command given to change a specific relay’s value.

The LED operation is controlled by the Tasmota LedMode parameter. On the Sonoff 4CH it is the blue “WiFi” labeled on. On the Sonoff Basic it is the red one. Normally it will be ON when any relay is ON. This is LedState=1. For irrigation control this LED has been implemented as being the ON when irrigation mode is AUTO or ON. The LedState configuration needs to be set to 0 via the Browser Console page (LedState 0) or MQTT message such as shown in Figure 240. For units such as the Sonoff Basic it likely will be desirable to leave LedState=1 so the one LED will be used to reflect the state of the relay and not the state of the irrigation mode.

Figure 241 shows the addition of a rocker switch on the Sonoff 4CH Pro cover and wired to the header with SDA (GPIO2) and GND labels. This switch corresponds to the Figure 242 GPIO2 setting. When the switch is in the closed position the irrigation logic is overridden. When the switch contact goes open or no Figure 242 user input setting specified then the irrigation mode will be fully controlled by MQTT.

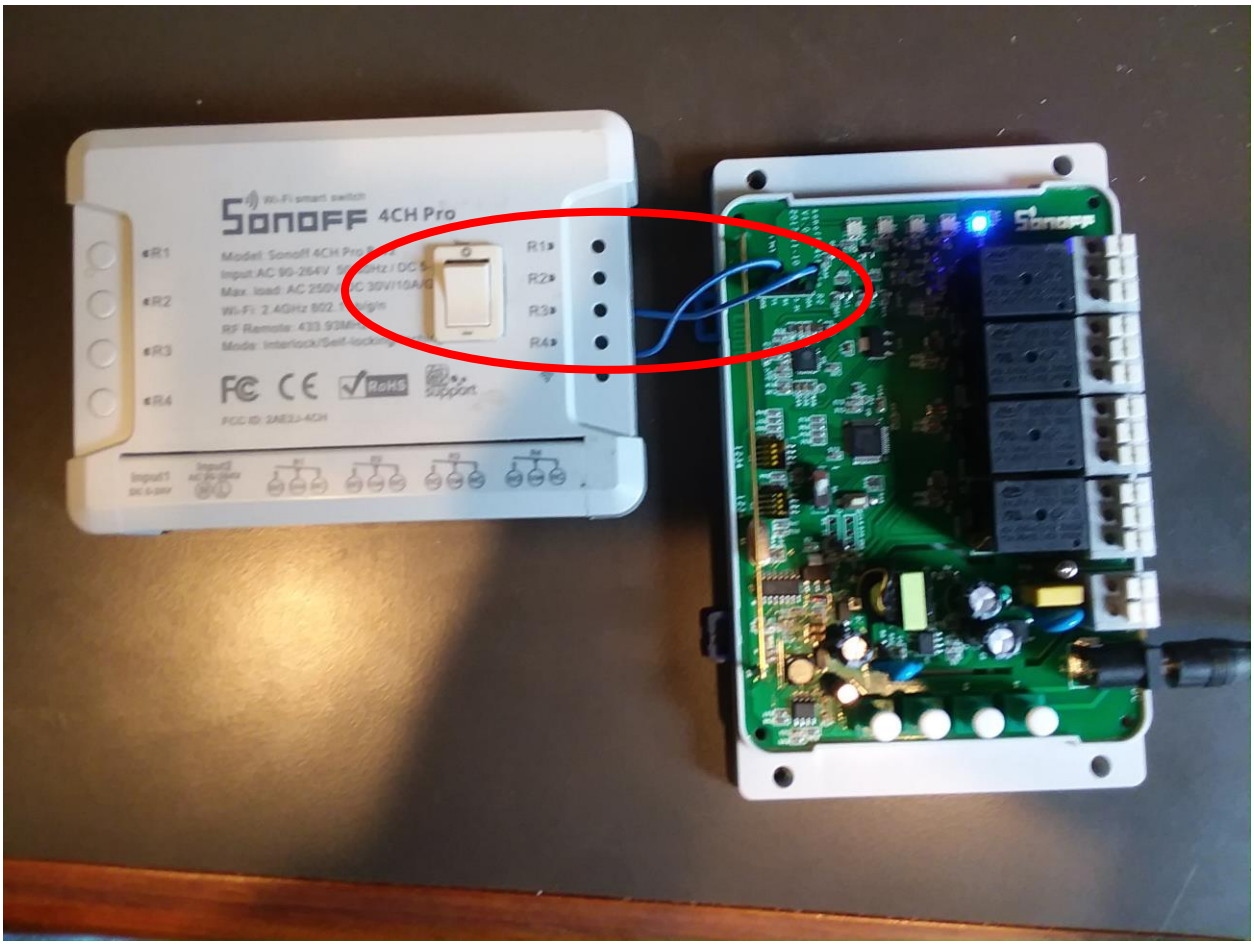


Figure 241 Switch Install for Enable Override

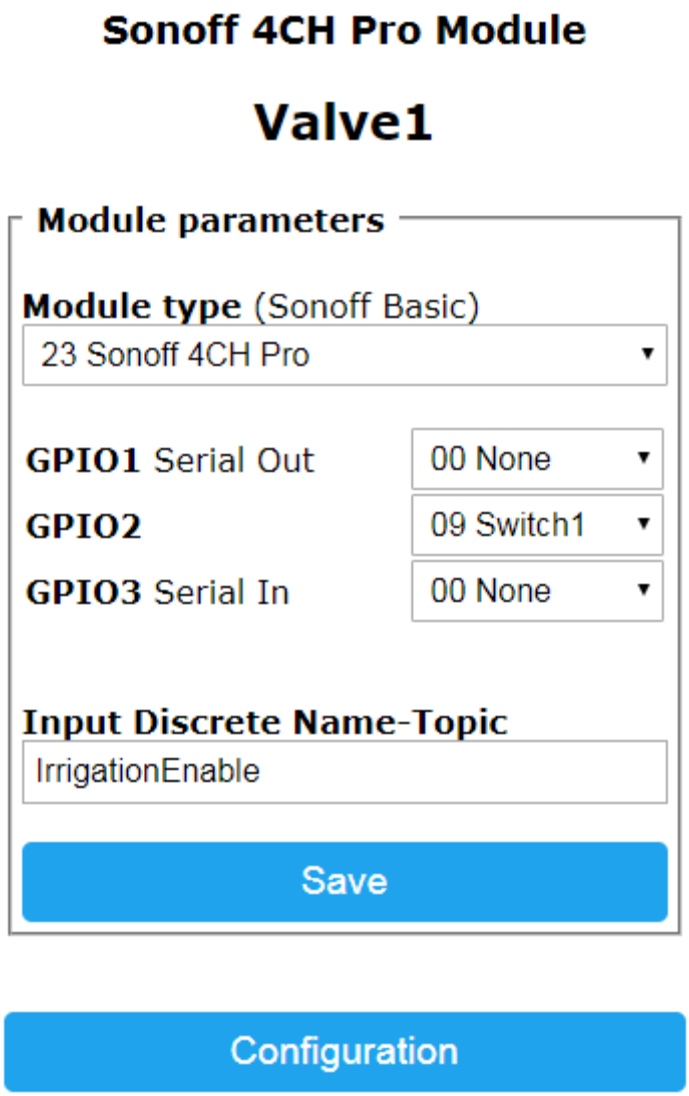


Figure 242 Sonoff 4CH Pro Module Setup

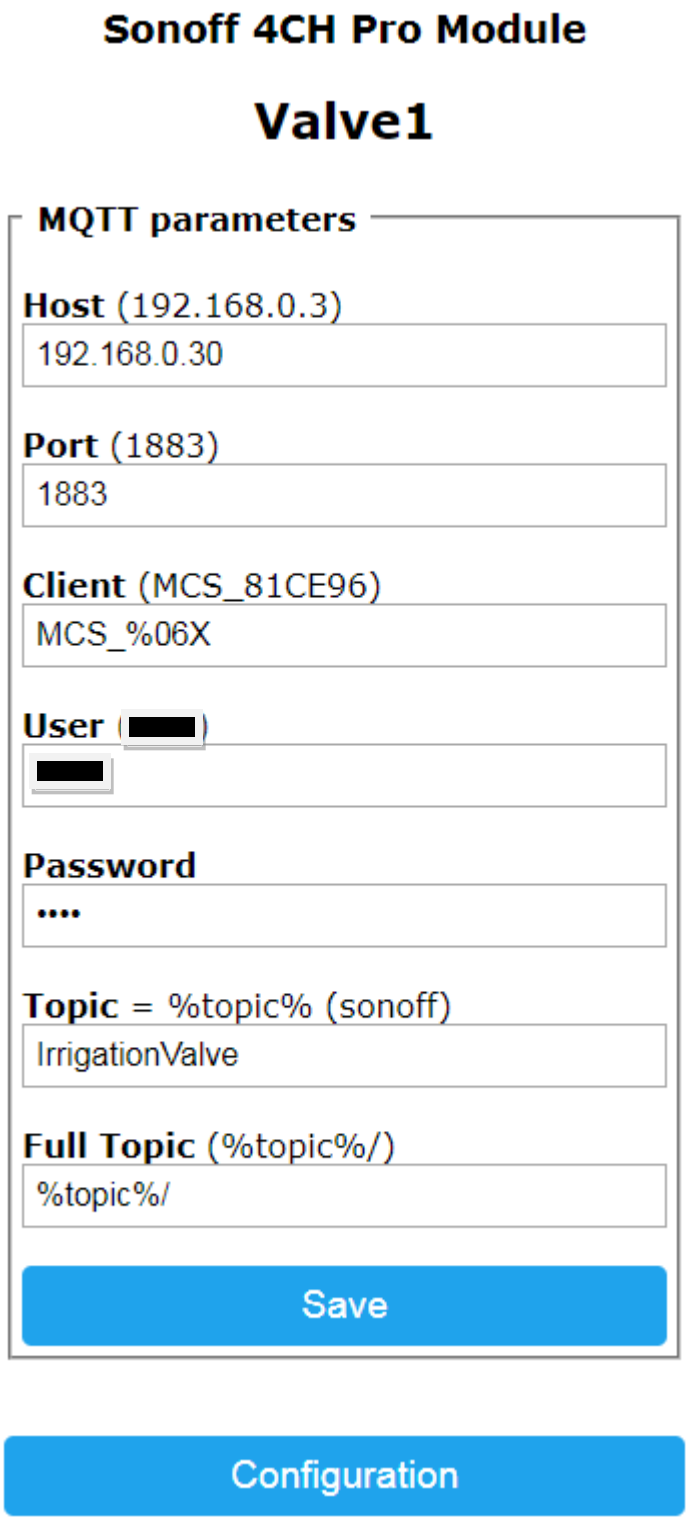


Figure 243 Sonoff 4CH Pro MQTT Setup

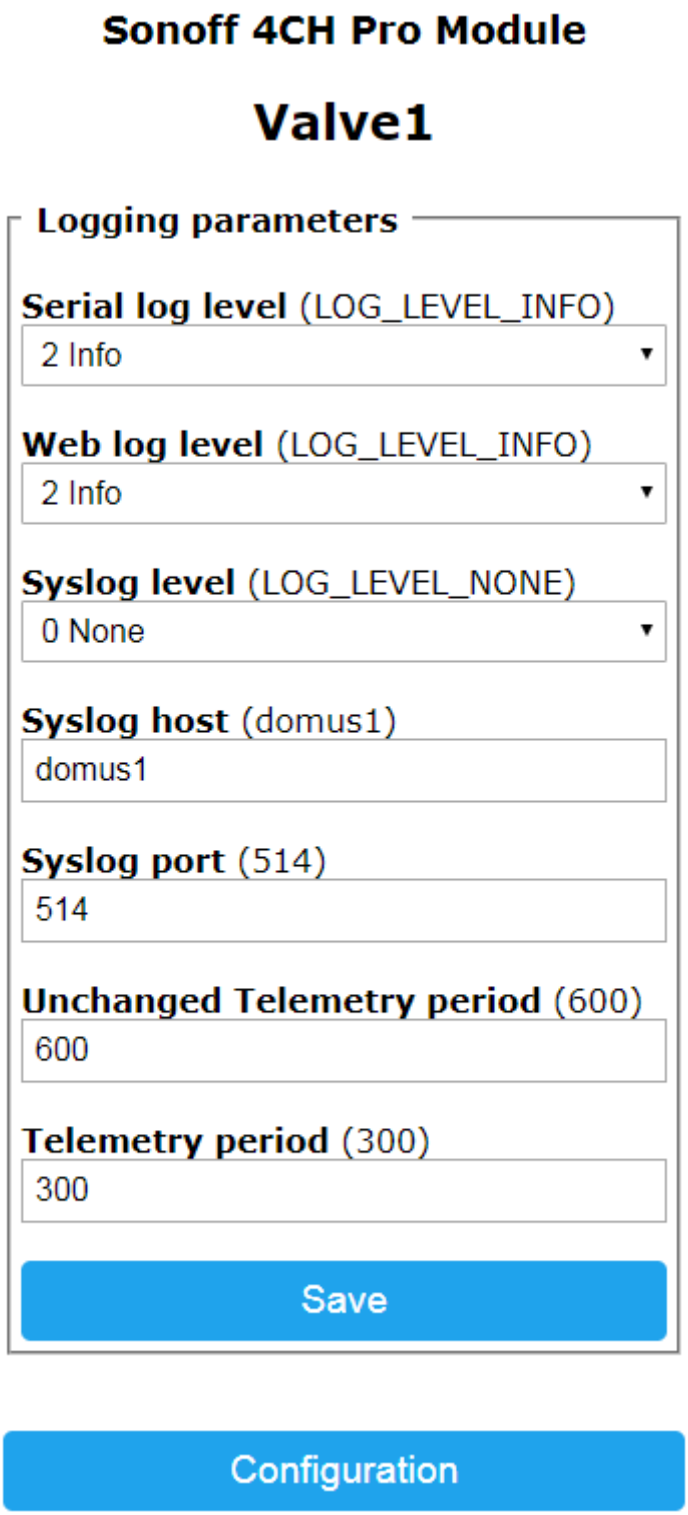


Figure 244Sonoff 4Ch Pro Telemetry / Logging Setup

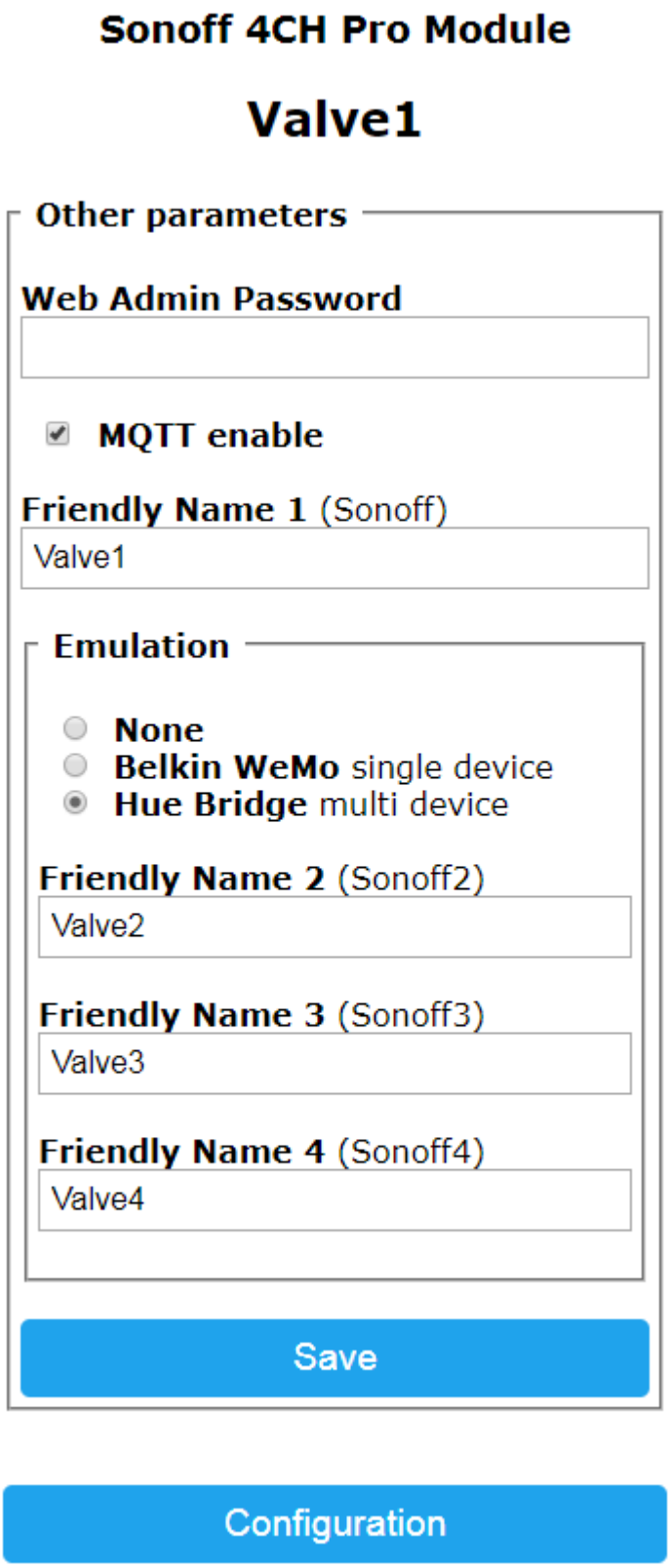


Figure 245 Sonoff 4Ch Pro Other Setup

21.8 Doppler Radar Motion Sensor

21.8.1 Warehouse Motion Light Switch

Passive Infra Red (PIR) sensors are commonly used to detect motion. They work in many situations and do poorly in others. Specific issues are they are directional so mounting orientation is important. They are also sensitive to sunlight/shadows, flying bugs, wind blowing.

A different technology using microwave frequencies are also very effective for objects that reflect microwaves. Water is reflected and people are made up largely of water. They are omni-directional so motion from any direction will be detected. This provides greater freedom in mounting the sensor, especially when trying to detect motion anywhere in a given area.

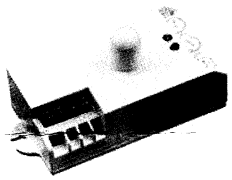
These devices can be purchased for around $1. I elected to use one that is nicely packaged and has adjustments for range. It also has notification pulse duration and Lux sensitivity for night-only use. For these additional settings I selected minimum 10 second pulse, and no light limitations. The link, picture and datasheet are captured below.

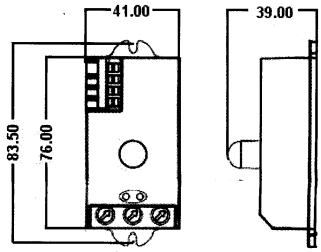
https://www.aliexpress.com/snapshot/0.html?spm=a2g0s.9042647.6.2.7d614c4dMCAYVu&orderId=9481903052658 4&productId=32794241378

Microwave sensor instruction

Mode No: SK-807K-DC

1. Appearance and size



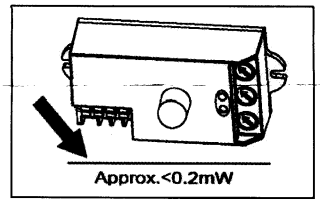


2. Technical specifications

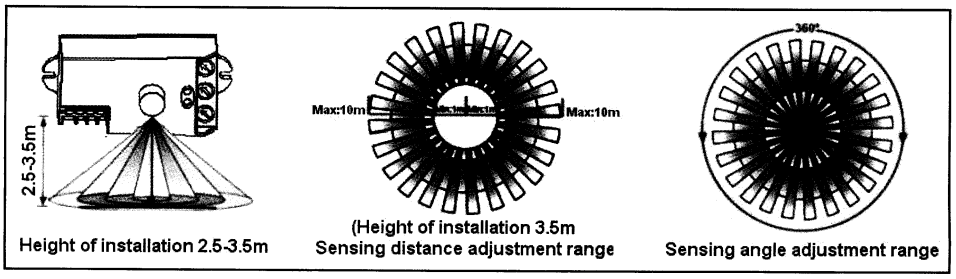
Power supply: 12-24VDC
HF system: 5.8GHz +/-75MHz CW radar, ISM band
Max load: 7A
Transmission power: <0.2mW
Power consumption: approx.0.5W
Reach: 1-10m (radii.), adjustable

Installation sit: ceiling mounting
Time setting: 10sec to 30min
Detection angle: 360°
Light control: 5~2000LUX
Operating temperature: -35℃~+80℃
Product size(L*W*H) : 90*41*39mm

NOTE: The high-frequency output of this sensor is <0.2Mw-that is just one 5000th of the transmission power of a mobile phone or the output of a microwave oven.



3. Induction range

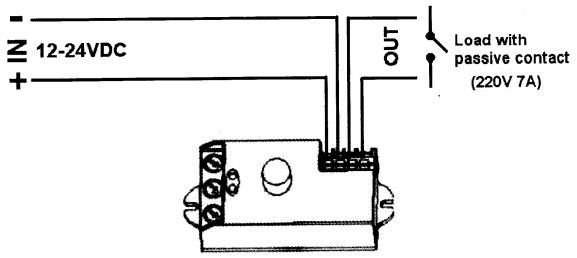


4. Utilizing field and introduction

SK-807K-DC is an active motion detector with adjustable sensitivity, daylight detection and time setting. It's based Doppler principle and RADAR technology, adopting rod antenna & high performance micro-chip. It is easy to install, provide high probability of detection, low nuisance alarms and resistance to rain, fog, wind, dust, falling snow and temperature extremes. The product detect range of 360°and its working frequency is 5.8G.The advantage of this product is stable working state (stable working temperature: -35°C~+80°C), SK-807K-DC adopts a microwave sensor (high-frequency output<0.2mW),so that it is safe and performs better than infrared sensor. It can be installed inside of product that is made of glass and plastic because that these materials make little effect to microwave.

Connect the product as shows below; you can change a common light to an automatic light.

Figure 246 Microwave Radar Sensor Data Sheet (1/2)



5. Settings

Detection range setting (Sensitivity)

Reach is the term used to describe the radius of the circular detection zone produced on the ground. After mounting the sensor light at a height of 2.5m, turn the reach control completely in anti-clockwise direction to select minimum reach (approx.1 m radius), and turn the reach control completely in a clockwise direction to select the maximum reach (approx.10m radius).The LED indicator will flash when the reach control is rotated. It flashes 1 to 10 times, representing 1m to 10m for the radius of the detection zone.

NOTE: The above detection distance is measured using a person who is between 1.6m~1.7m tall with an average build, moving at a speed of 1.0~1.5m/sec. if any of these variables are changed, the detection distance will also resultantly change.

Time setting

The light can be set to stay ON for any period of time between approx. 10sec (dial turned fully anti-clockwise) and a maximum of 30min(dial turned fully clockwise). Any movement detected during the "on" time will reset the timer. The LED indicator will flash when adjusting the time setting dial. The number of flashes means the following: 1 flash = 10sec, 2 flashes= 1 min, 3 flashes= 2 min, 4 flashes= 5 min,5 flashes=8 min, 6 flashes=10 min, 7 flashes=15 min, 8 flashes=20 min,9 flashes=25 min,10 flashes=30 min.

NOTE: After the light switches off, it takes approx. 1sec before it is able to start detecting movement again. The light will only switch on in response to movement once this period has elapsed.

Light-control setting

The chosen light response threshold can be infinitely from approx. 5-2000LUX. Turn it fully anti-clockwise to select dusk to dawn operation at about 5 LUX. Turn it fully clockwise to select daylight operation at about 2000LUX. The knob must be turned fully clockwise when adjusting the detection zone and performing the walk test in daylight.

6. Troubleshooting

malfunction	Cause	Remedy
The Load does not work	Wrong light control setting selected	Adjust setting
	Load faulty	Change load
	Mains switch off	Switch on
The load work always	Continuous movement in the detection zone	Check zone setting
The load work without any identifiable movement	The sensor not mounted for detecting movement reliably	Reinforcement install accessories
	Movement occurred, but not been identified by the sensor (movement behind wall, movement of a small object in immediate lamp vicinity etc.)	Check the induction space settings
The load will not work despite movement	Rapid movements are being suppressed to minimize malfunctioning or the detection zone you have set is too small	Check the induction settings

Tips

Please under the guidance of professionals personage to set related parameters of the sensors, do not secretly dismantling the product. We reserve the right to make technical change without prior notice.

Figure 247 Microwave Radar Sensor Data Sheet (2/2)

For me what was inconvenient was the 12VDC to 24VDC voltage range. The technology is low power so battery use may be possible, but this unit is rated at 500mw so will not work for very long using batteries.

The interface is a switch that closes for the selected pulse duration. This means anything that can sense a switch closure can be used to provide the interface to the HA system. Something like a door-window sensor would work well. Since I needed to provide non-battery power anyway, I elected to use a microcontroller and WiFi with the Sonoff Basic being the item that hangs out in my junk box. My hookup is shown in Figure 248.

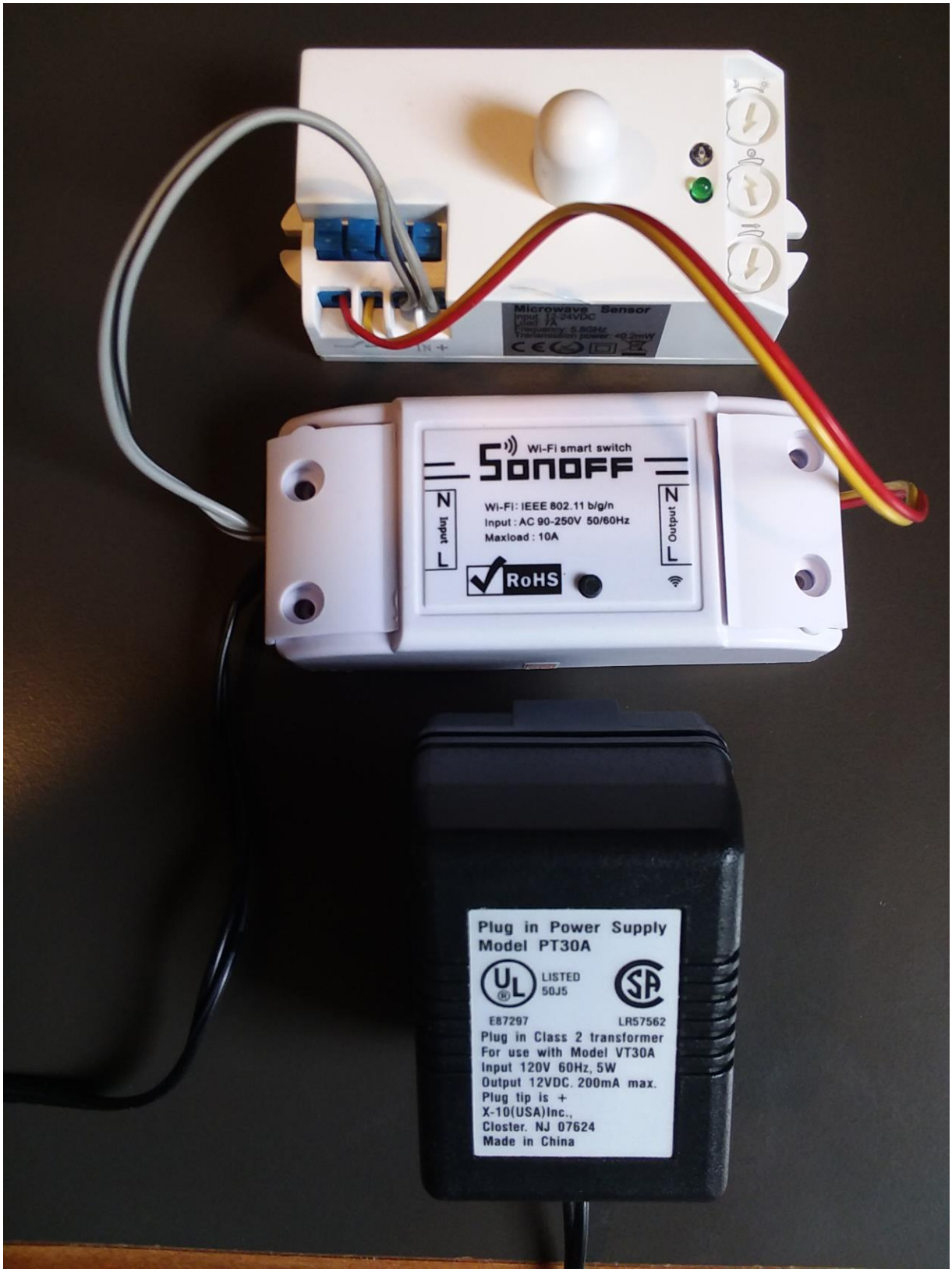


Figure 248 Sonoff Interface to Radar Motion Sensor

To apply the Sonoff Basic to this application there are some circuit board cuts and jumpers needed. The stock unit is expecting 120/240VAC power and the radar needs 12VDC so needed to bypass the Sonoff power supply and run 12VDC directly into the 3.3VDC voltage regulator in the Sonoff. The Sonoff IO is designed as an output but this use has it as an input. While wire can be run directly from the Sonoff circuit board, I elected to use the screw terminals that are normally used for the output.

The output connector is shown on the left in Figure 249. Just to the lower right of the connector the top trace was cut with a utility knife. This same connector pin needed further cut on the back side as shown with the red ellipse in Figure 250.

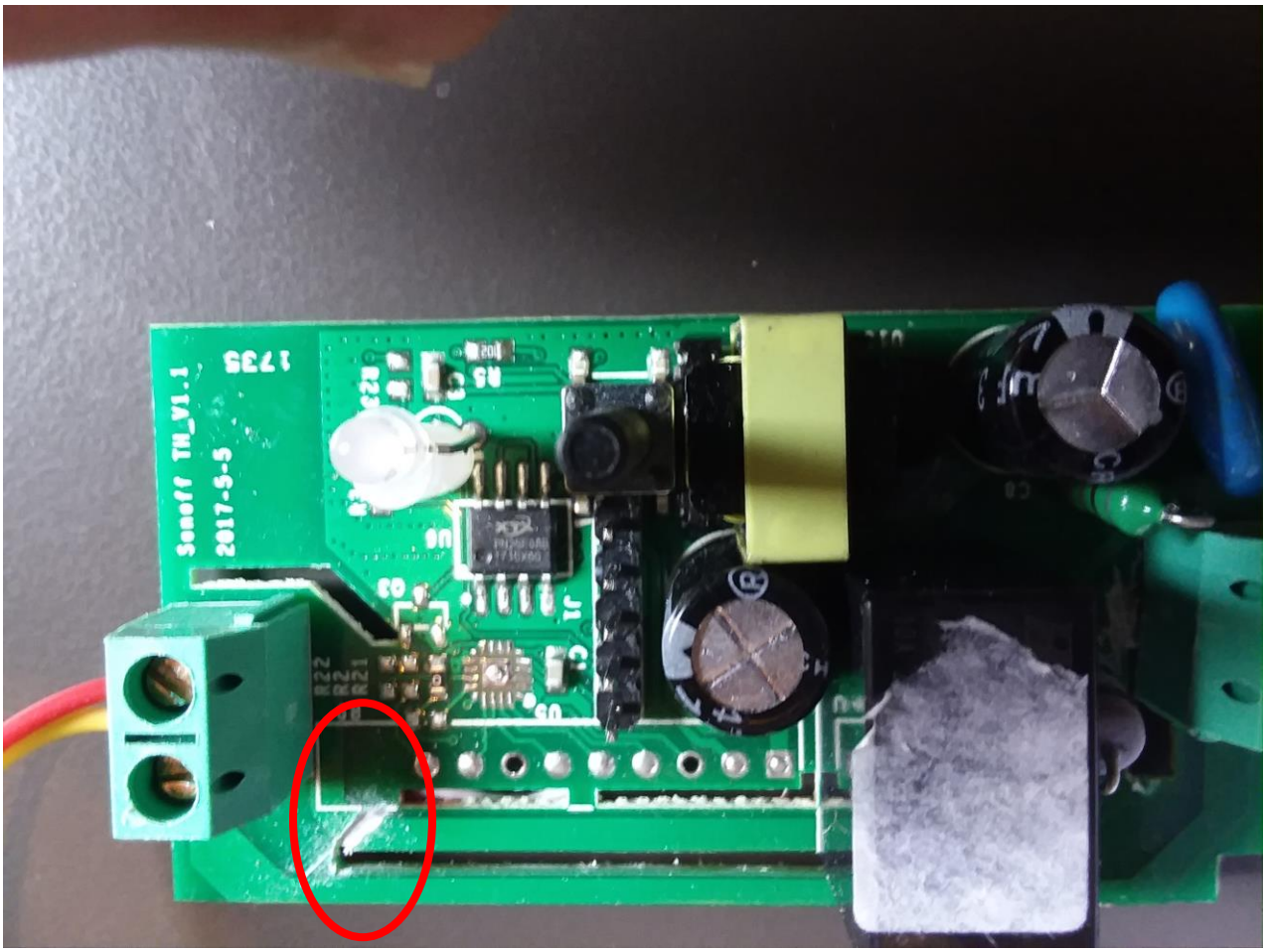


Figure 249 Sonoff Basic Circuit Board Top

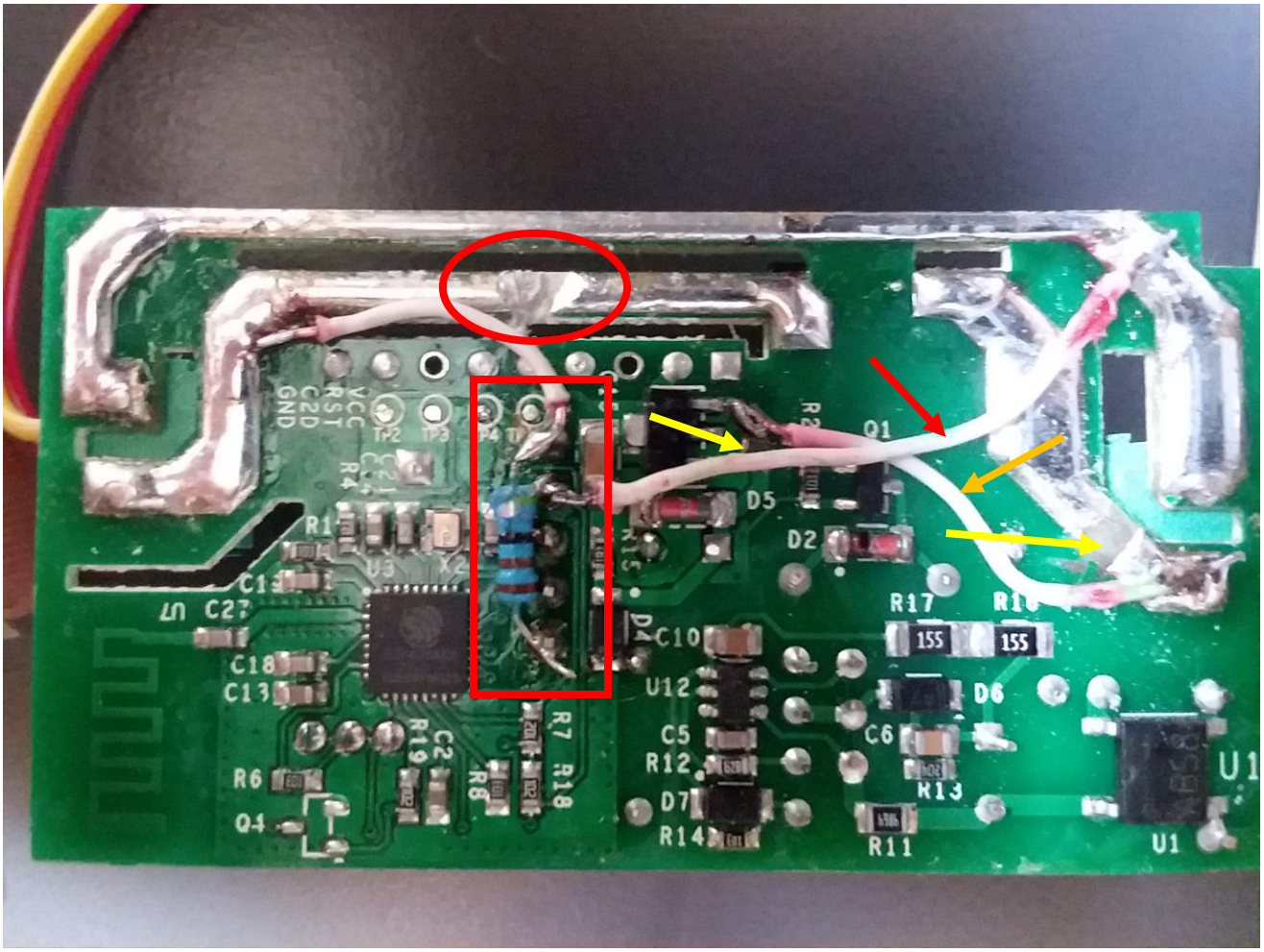


Figure 250 Sonoff Basic Circuit Board Back

The other side of the output connector is already wired to the Neutral side of the input connector. This will be used to form the ground connection when the Radar Sensor switch closes.

I did not recall is the GPIO 14 had a pull-up resistor or not so I added one. A row of 5 pins are available in the center of the circuit board where the resistor was added as shown with a red rectangle. The bottom one is 3.3V and the top one is GPIO14. Next to the top is Ground. A wire was soldered from Ground to the Neutral input/output connections as shown with the red arrow.

To bypass the Sonoff power supply the trace on Line input connector and on the input pin of the voltage regularo was cut with a utility knife. These are shown with a yellow arrow. A wire was soldered on each side of the cut connections to provide 12VDC input pin connection to the voltage regulator input. This wire is shown as orange arrow.

With the above alterations the input connector Line is 12VDC and Neutral is ground. The output connector Line is switch input for GPIO14 and Neutral is ground.

Any version of Tasmota firmware can be loaded and configured to be a Switch input as shown in Figure 251. WiFi and MQTT parameters also need to be set as shown in Figure 252 and Figure 253. Once configured the Tasmota switch mode should be set to 2. This will result in the switch status to be the

same as radar sensor output. If this is not done then each motion detection pulse will toggle the switch status. This is most easily done from the Sonoff Tasmota Console browser page with command “Switch Mode 2”.

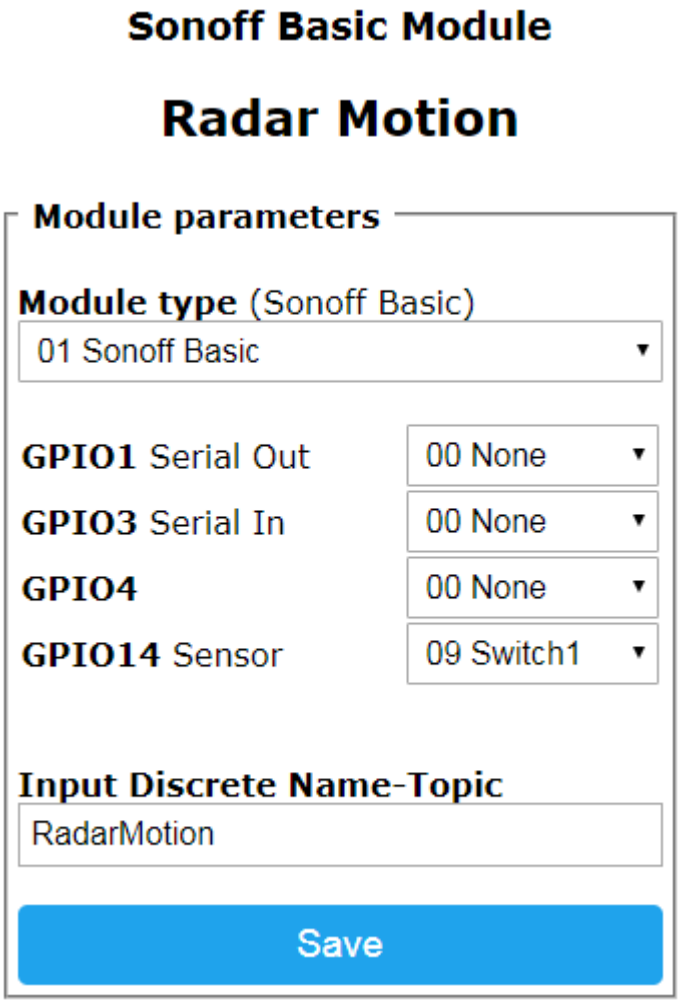


Figure 251 Radar Sensor Module Configuration

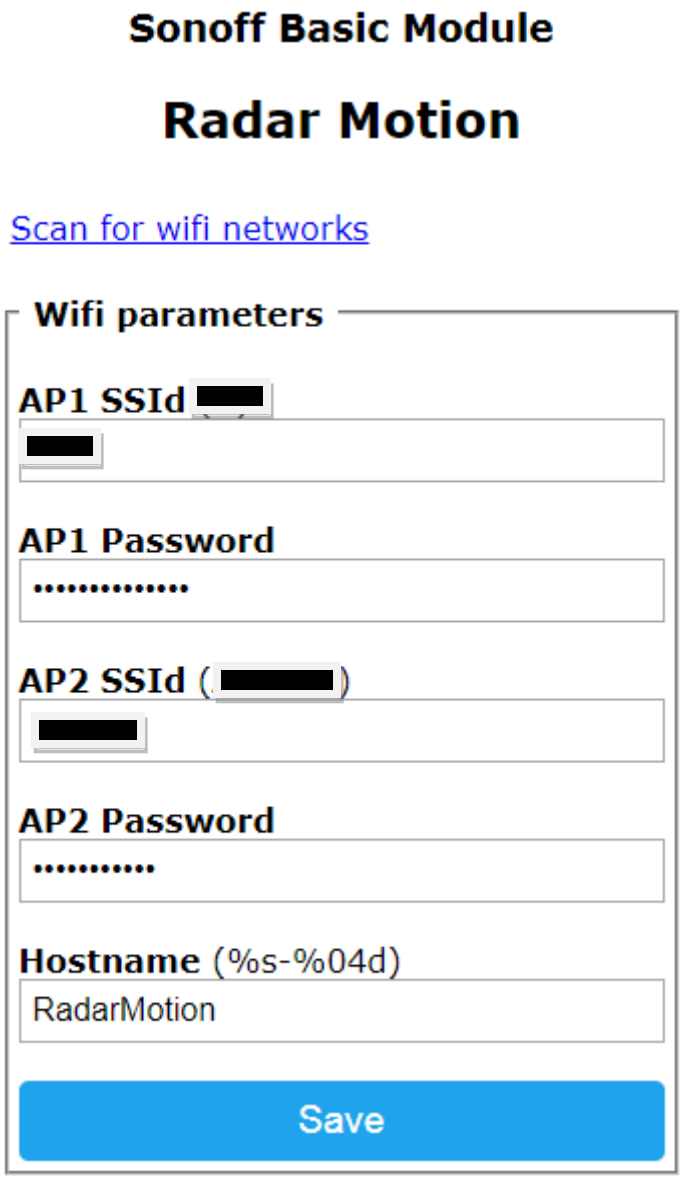


Figure 252 Radar Sensor WiFi Parameters

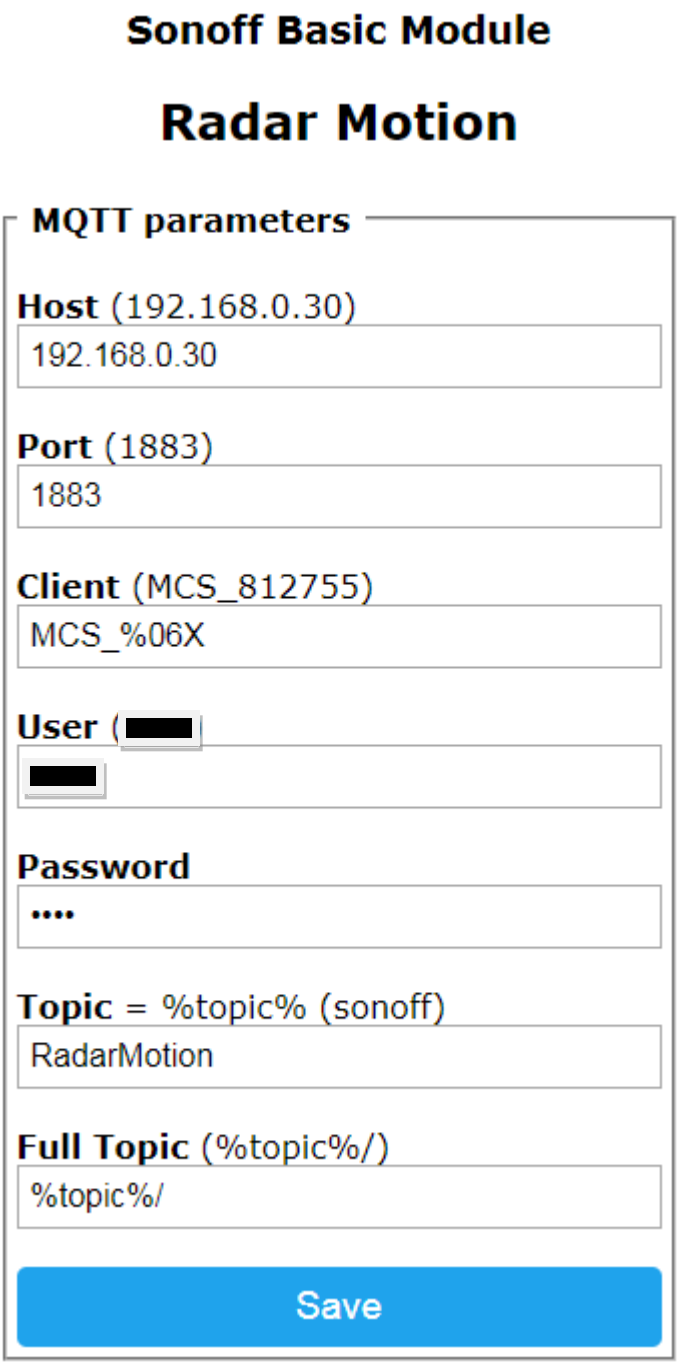


Figure 253 Radar Sensor MQTT Parameters

21.8.2 RCWL-0516 (Automotive Proximity)

The RCWL-0516 is a similar microwave motion detection circuit card. It accepts 4VDC to 28VDC input and produces a 3.3VDC output. No sensitivity adjustments available other than changing components. Schematic is shown in Figure 254. Placement of the sensor is specified as minimum of 1 centimeter from metal objects and this likely includes the circuit traces etc of the Sonoff.

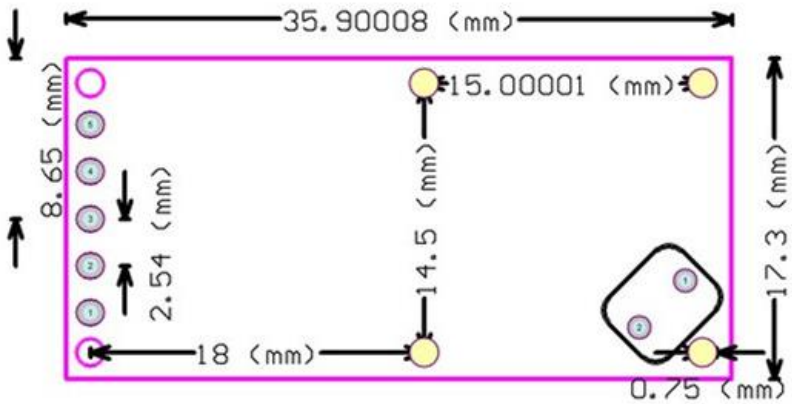
This mini doppler radar sensor module is equipped with supporting DC 4-28V wide voltage This is a mini radar motion sensor module is equipped with RCWL-9196 chip based on the doppler microwave induction technology. It will automatically comtinuously output the high level TTL signel when there is motion. 360 degree no blind angle detection and maximum 7m sensoring distance with adjustable delay time and sensitivity. It is perfect for DIY microwave motion sensor light switch, human sensor toys, smart security devices, etc.

Notice:

1.The right ahead of induction side should't place any metal to block

2.The front/back side should reserve more than 1cm space

3.Module and installation carrier



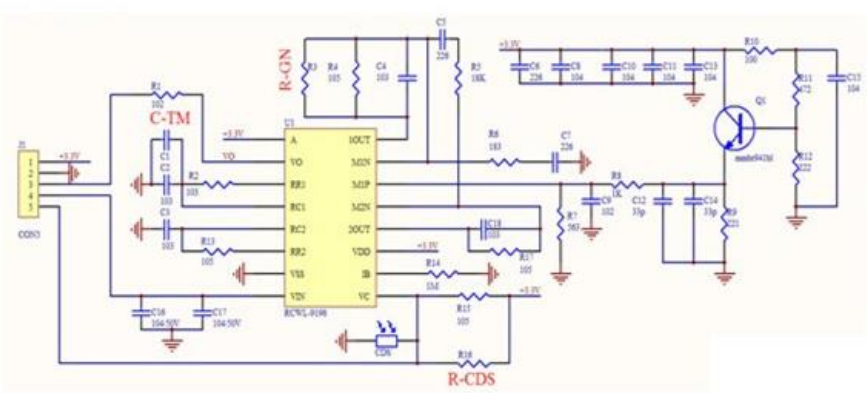


Figure 254 RCWL Schematic

The Sonoff Basic interface is done by tapping off of the 5VDC regulator input on the Sonoff and using GPIO14 as the input for the RCWL-0516 output. A 1K resistor was placed in series with GPIO14 input. As near as possible to the RCWL-0516 a 3300 microfarad capacitor was placed between the power input and ground to stabilize the voltage to the sensor. The wiring is shown in Figure 255.

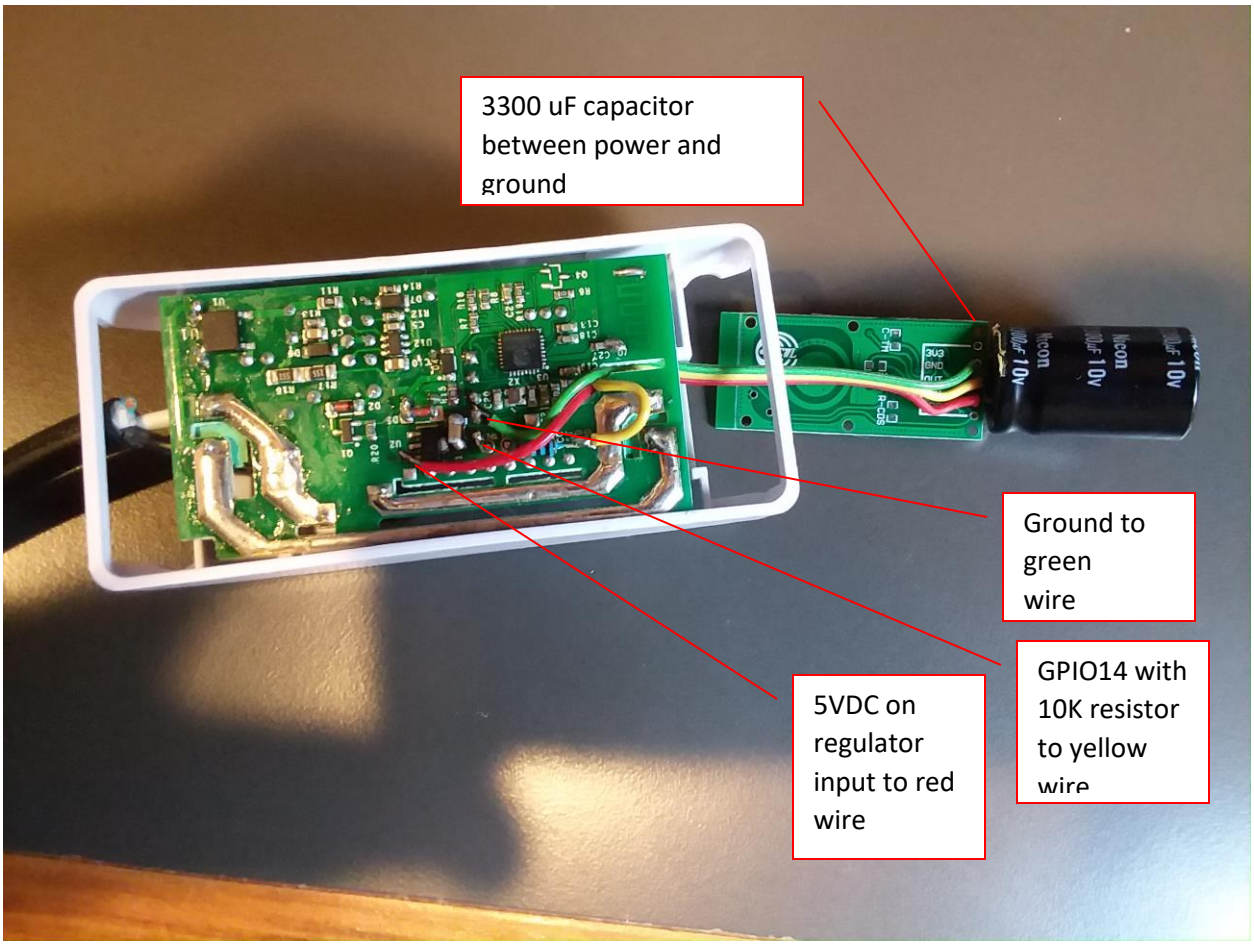


Figure 255 RCWL-0516 Interface Wiring

The Tasmota setup is the same as with the other radar sensor with one exception. The sensor output is high when motion occurs so the switchmode should be set to 1 in the Tasmota console page. This will result in the MQTT status to be the same as the sensor status.

21.8.3 HLK-LD2410C Human Presence

The HLK-LD2410C is a small module designed for microcontroller integration for purpose of human presence detection. It contains a Bluetooth interface that is paired with a smartphone App (search for HLK Radar in Google or Apply stores) and contains a serial and discrete interface for automation integration. It is well documented in the manufacture's documents at HLK-LD2410C - Google Drive that includes both interface and installation guidance.

It can be obtained at reasonable price locally via Amazon https://www.amazon.com/dp/B0C8H7ZBRS and even cheaper via AliExpress. This is a directional device with a 60-degree field of view and detection range up to 5 meters. It can be tweaked via Bluetooth (or serial) to constrain the detection range to a more limited distance.

It is possible to interface this device with a TTL-level UART or with a microcontroller such as ESP8266 or ESP32. In the UART case, the mcsMQTT Local Page, Serial Tab is used to specify the COM port, baud rate, End Of Line byte and LD2410C serial decoding. A hex F8 (248) is a repeating character in the serial header so made for a convenient way to break up the continuous serial stream from the sensor.

Using the serial interface does not include the discrete output for binary present/not-present, but the serial data can be used with more resolution to include stationary vs. moving as well. If the UART does not support 256000 baud, then it can be changed on the Bluetooth App to a compatible baud rate.

Later evaluation of the data shows that the discrete is an important output that cannot be easily derived from the other data so it makes the use of the serial port UART a more limited choice.

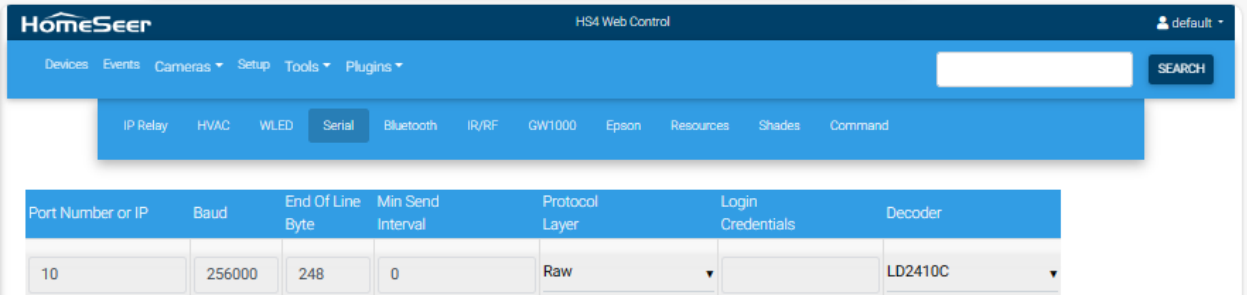


Figure 256 LD2410C Interface via UART

It is also possible to interface this module via Bluetooth in a manner similar to what is done with the HLK Radar App. The documentation indicates that the Bluetooth protocol is the same as the serial protocol.

I did query the advertisement with the following returned to have three services

The first of my units advertised on HLK-LD2410_EF43 with the following parameters

...Service Count 3 for BluetoothLE#BluetoothLE00:e0:45:da:ed:03-62:8d:ee:1f:ef:43
....Characteristics Count 1 for service 1 00001800-0000-1000-8000-00805f9b34fb
...ReadCharacteristic 1 for Service 1 characteristic 00002a00-0000-1000-8000-00805f9b34fb:48-4C-4B-2D-4C-44-32-34-31-30-5F-45-46-34-33 HLK-LD2410_EF43
...WriteableCharacteristic 1 for Service 1 characteristic 00002a00-0000-1000-8000-00805f9b34fb
...Characteristics Count 2 for service 2 0000fff0-0000-1000-8000-00805f9b34fb
...Characteristics Count 2 for service 3 0000ae00-0000-1000-8000-00805f9b34fb

The manufacturer manual documented two UUID shown below si it appears the second service is the primary one. I did not pursue the Bluetooth interface any further.

Feature UUID	Operation authority	Function definition
0000fff1-0000-1000-8000-00805f9b34fb	Read/Notify	Module send, APP receive
0000fff2-0000-1000-8000-00805f9b34fb	Write Without Response	APP send, module receive

For the performance evaluation it was evaluated by coupling the device with an ESP8266 using both the serial and the discrete outputs. A D1 Mini was used for this. The prototype is shown in Figure 257 with the wiring connections shown in table below. Use of D1 Mini D2, D5 and D6 are not critical so if not available on other microcontroller than others available can be used. Just need to align the Tasmota module setup with the pins being used.

HLK-LD2410C Pin	D1 Mini Pin
TX	D6

RX	D5
Out	D2
Gnd	Gnd
Vcc	5V

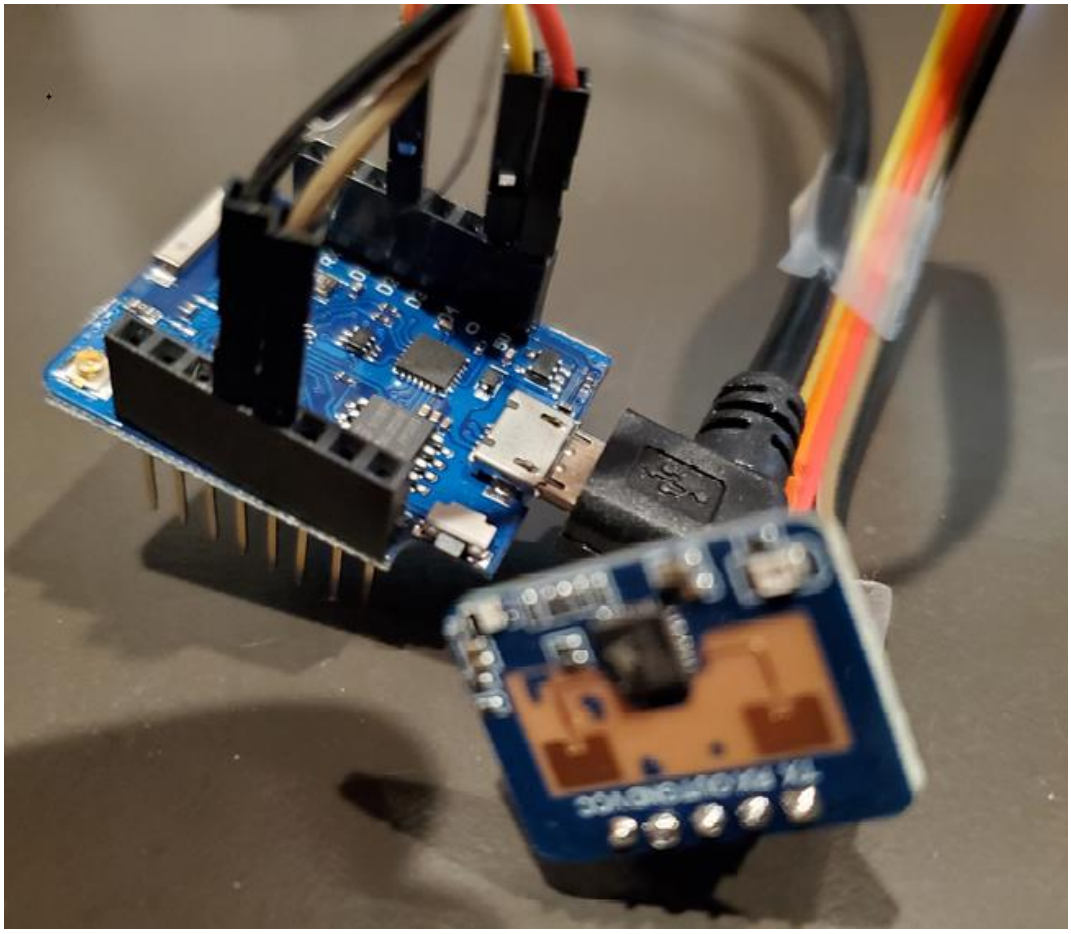


Figure 257 LD2410C Evaluation Prototype

Stock Tasmota firmware was installed using the Tasmota Web Installer, with the Tasmota (English) firmware version selected, in the ESP8266. An ESP32 could also be used, but the additional horsepower is not needed.

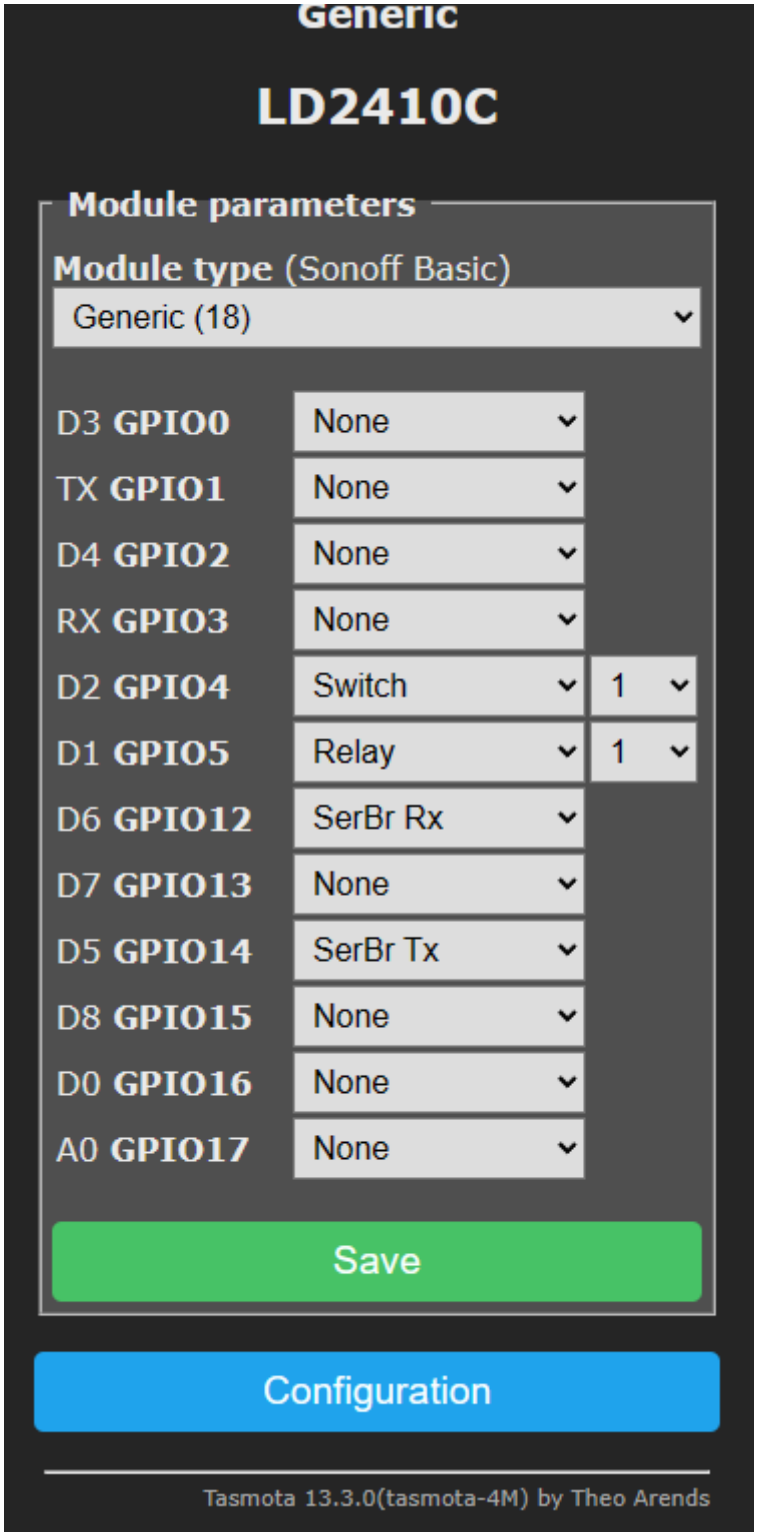
The Tasmota configuration is shown in Figure 258. The Serial connection used the D6 and D5 pins of the ESP8266 with the RX and TX swapped when connecting to the radar module. The radar module discrete (center pin) output was connected to D2. A virtual relay was configured on D1. This was done to simply the MQTT output all being available in the x/RESULT topic.

The Tasmota Other Tab was used to give it friendly names of LD2410C and the MQTT Tab was used to assign the Topic and MQTT Broker address. In my case I prefer the Full Topic starting with the topic

name rather a prefix so I removed the prefix in this setting. **I also used LD2410C as the Topic as this is the Topic that mcsMQTT recognizes to perform the serial decoding**. If multiple LD2410C are being used then the Topic should include additional identification such as LD2410C/Kitchen so each can be uniquely identified.

From the Tasmota browser Page, Console Tab, the following was entered to have the serial data reported as hex and to match the 256000 baud of the radar module

```
sbaudrate 255900
serialdelimiter 254
```


Generic
LD2410C
Module parameters
Module type (Sonoff Basic)
Generic (18)
D3 GPIO0
None
TX GPIO1
None
D4 GPIO2
None
RX GPIO3
None
D2 GPIO4
Switch
1
D1 GPIO5
Relay
1
D6 GPIO12
SerBr Rx
D7 GPIO13
None
D5 GPIO14
SerBr Tx
D8 GPIO15
None
D0 GPIO16
None
A0 GPIO17
None
Save
Configuration
Tasmota 13.3.0(tasmota-4M) by Theo Arends

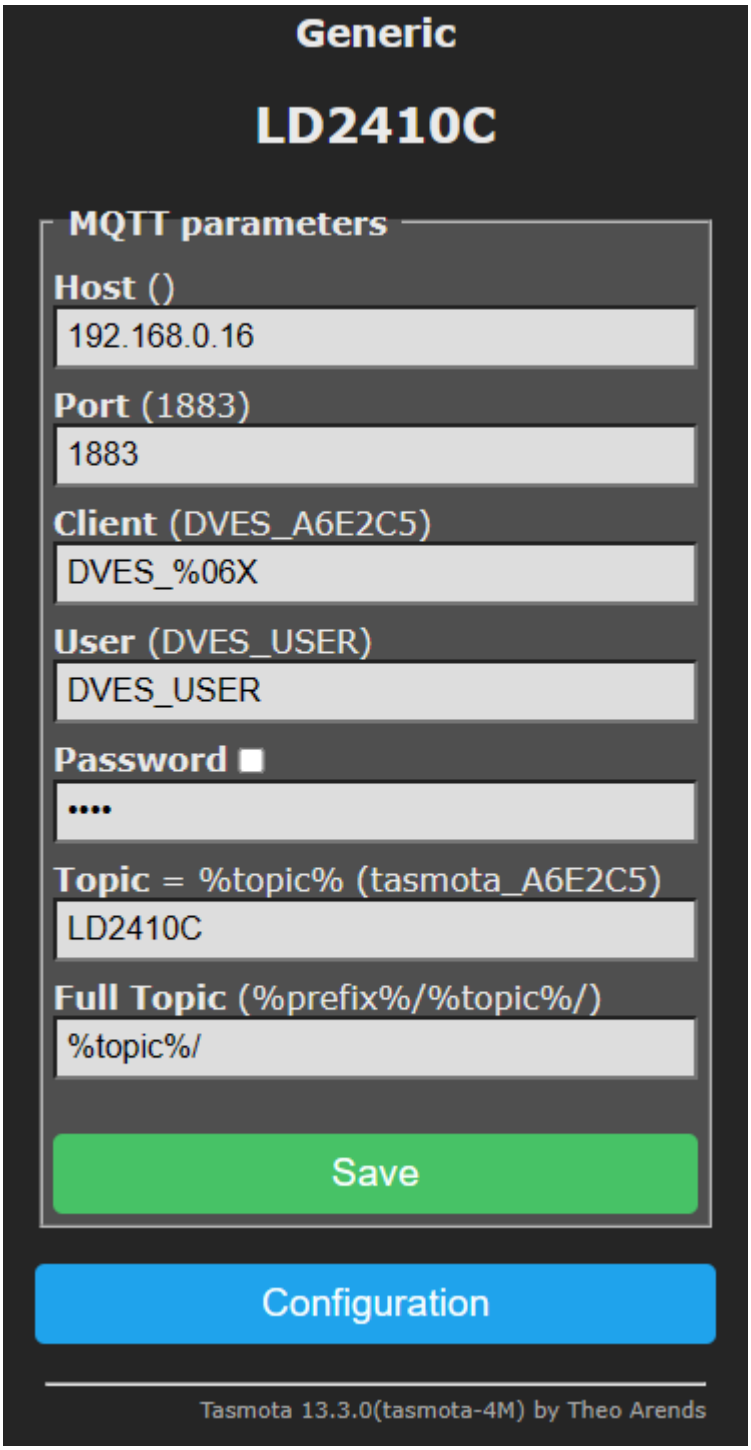


Figure 258 LD2410C Tasmota Module Setup

The data shown on the MQTT Page, Association Tab (filtered for LD2410C/RESULT Topic) is shown in Figure 259. The “a” column checkbox was used to create the HS Device Features and the “s” column checkbox was used to collect data for charting analysis.

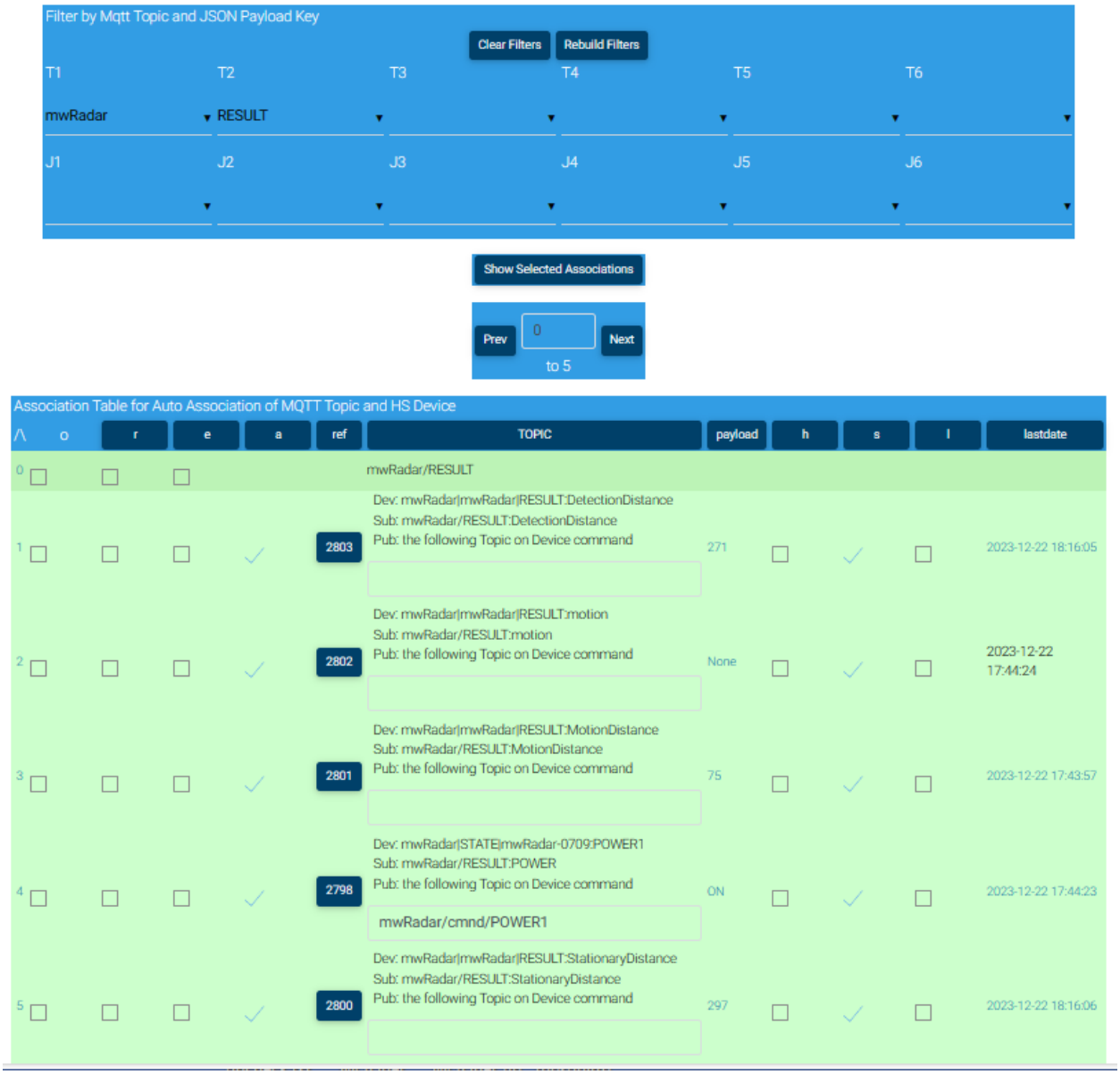


Figure 259 LD2410C Reported RESULT Data

The data made available from the unit are three different distance measurements, the discrete state, and a motion state consisting of of one the four states

"None",
"Motion",
"Stationary",
"Motion+Stationary"

These four states are mapped in the VSP entries in the created HS Feature. They also show up in the Bluetooth HLK Radar App in its status summary display.

For the three-minute test caputed in Figure 260, it can be seen at 5:50 PM the distance reports that one was walking into the room where the sensor was located. Overall, it was a good correlation between distance and actual distance.

About 40 seconds later the discete output goes high and remains high for about two minues. At 5:43 motion out of the field of view (but still in the room) was detected. As walking out of the room, and the field of view was reentered, the distance showed the change until after 5:43 when the room was exited.

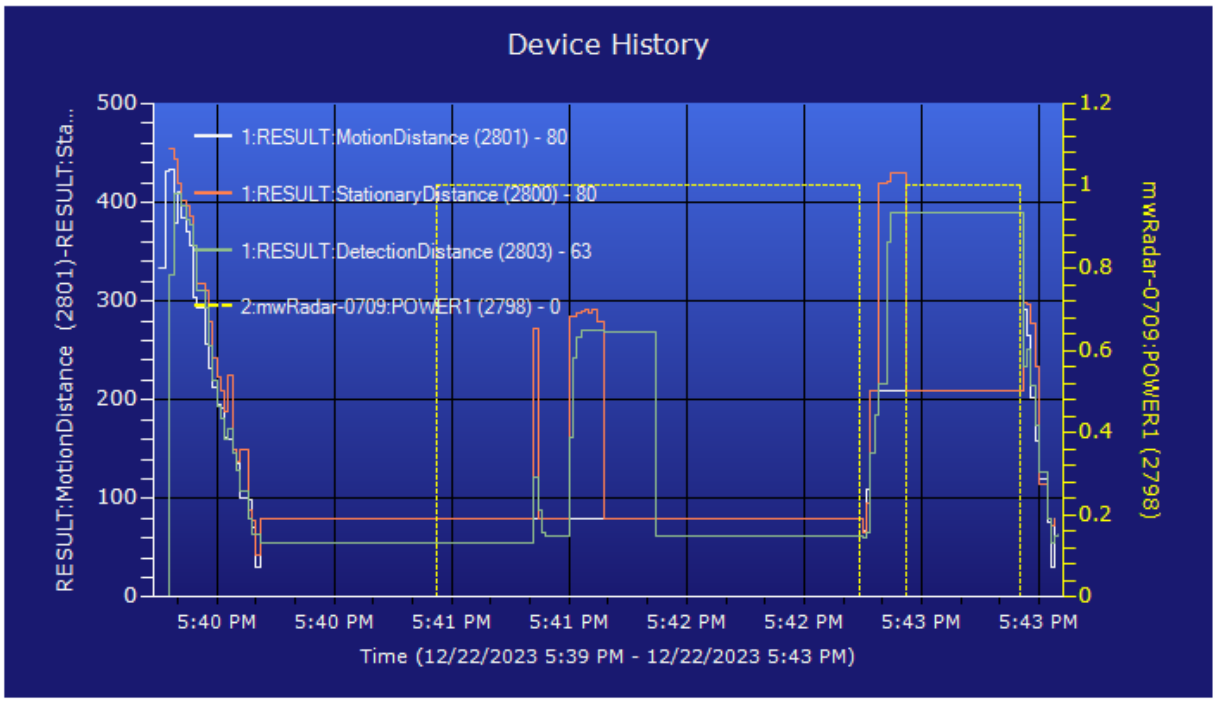


Figure 260 LD2410C Data Analysis

The corresponding HS Device and Features are shown in Figure 261. In this snapshot it showed a stationary person in the field of view and three different distance measurements. A better understanding of each distance measurement is needed to support potential automation trigger logic. Note also that the POWER discrete is OFF when presense is detected. The more natural state would be ON. This can be changed in the Tasmota configuration using the configuration command on the console to invert the logc as:

“switchmode 2”

Another approach in on the mcsMQTT Edit page for POWER and define the VSP to be something like

“ON=0,Unoccupied,Occupied”
“OFF=1,Occupied,Unoccupied”

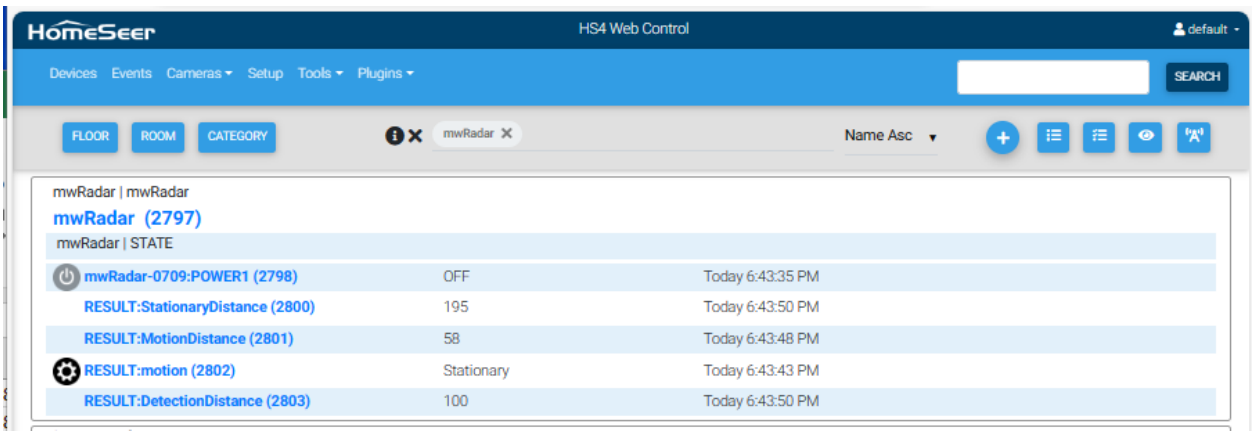


Figure 261 LD2410C HS Device and Features

Overall, this module is easy to setup and appears to provide reasonable and responsive results for presence detection. It is directional so its setup has the same considerations of a camera setup to assure the field of view in the area of interest.

While not tested much, it appears that the beam does penetrate door and walls using the default factory configuration, but obviously only in the forward-facing direction. This has the advantage of a more controlled location response vs. the omni-directional sensors.

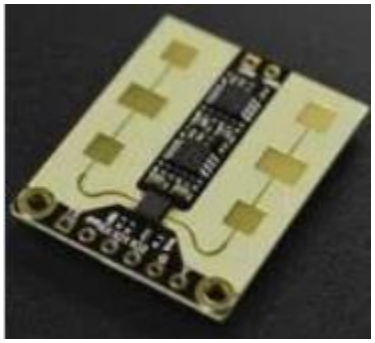
Contrast this with the other radar sensors evaluated, where the field of view was 360 degrees and penetration through wall occurred. In the latter case they worked very well for a ceiling mounted sensor that was setup to turn lights on when an area of the home was being approached. The lights would actually turn on as one was opening the door rather than a second after the doorway was entered.

For this sensor it looks to have a delay of up to a second from when motion was first detected, and the discrete output toggled. If one needs faster response, then the distance measurements are available.

I did not see any obvious way to assess being within a “zone”. I think anywhere along the detection arc will report the same distance so right vs. left of the beam center is unknown. This is a variance from the Aqura FP1/FP2 sensors that advertise zone reporting. I have never actually tried zone detection with the Aqura.

21.8.4 DF Robot SEN0395 mm Wave Radar Detection Sensor

The SEN0395 is available from Amazon [Amazon.com: DFROBOT mmWave Radar - Human Presence Detection Sensor (9 Meters) : Electronics]() . It is a high-quality construction at a higher price-point of $34. It uses a longer range (9 meter) and narrower field of view (100x40 degrees) so represents the most precise of the units evaluated from an installation perspective.



The SEN0395 is another sensor that has both serial and discrete interface capability. At this time the serial interface contains the same binary information available from the discrete interface. There are provisions in the serial protocol for additional information so in the future it may provide data such as is available from the LD2410C.

It has the ability in the serial protocol to identify four distance-based zones. It is not clear how these get reported as there is only a single binary output. Contrast this with the LD2410C that reports distance on a continuous scale.

The prototype was constructed using dupont wires between the sensor and the UART. The 5V power was selected from the UART and the signal lines selected to be 3.3V for TX and RX. Ground was the fourth wire. It can be seen in Figure 262.

Initial evaluation was using Termite to access the serial port UART. This confirmed a message stream that looks like the following repeated every second. When out of range the “1” changes to “0”. This means the only info of interest is the number.

```
$JYBSS,1, , , *.
```

The serial protocol is documented at [mmWave Radar Sensor Arduino-Human Presence Detection Wiki - DFRobot]() . I tried using it to stop the sensor and to configure the detection area. It was done using copy/paste from the document to Termite input window. The commands had no effect. There was not a respone of “Done” or “Error” as described in the manual. I did not pursue the control via serial any further.

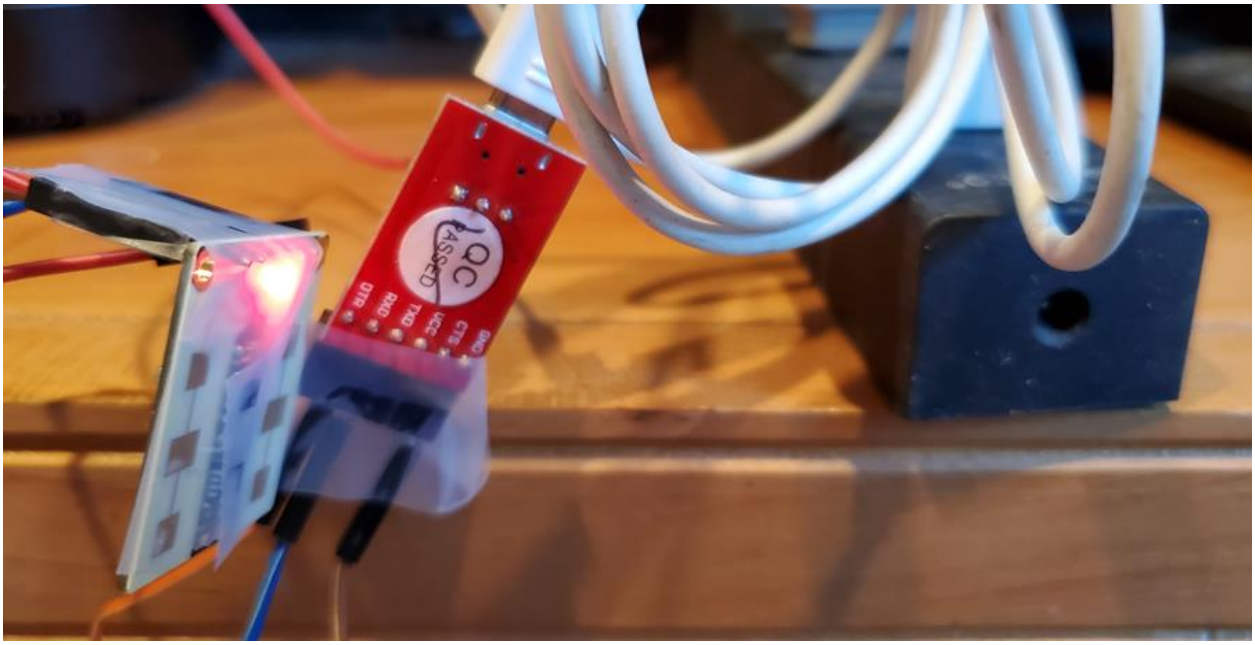


Figure 262 SEN0395 Prototype Wiring

Next step was to interface to mcsMQTT and HS. The UART continued to be used, but the port now being setup to be used by mcsMQTT. mcsMQTT Local Page, Serial Tab was setup using the default 115200 baud and LF character as shown in Figure 263.

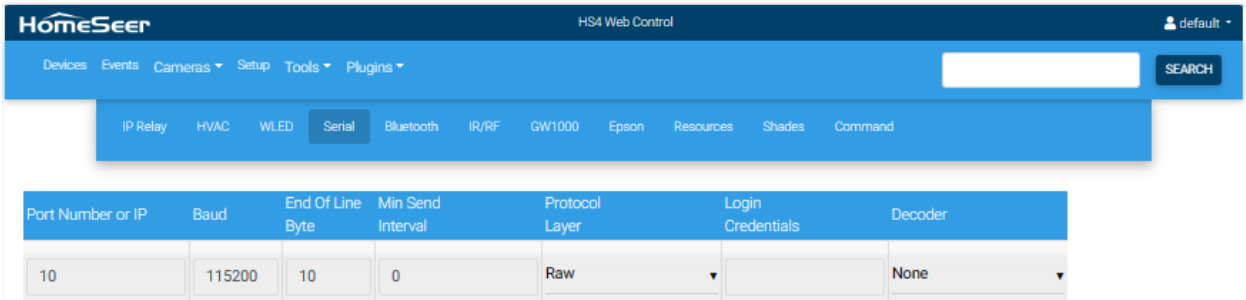


Figure 263 SEN0395 Interface Setup

Data was being received and mapped into the HS Feature as DeviceString. This was edited to extract the number from the $JYBSS,1, , , *. Text using regular expression and then to setup VSP to report Occupied vs. Unoccupied for the 0/1 values. Figure 264 show the use of the Edit Tab to accomplish this. Figure 265 shows the view from HS with COM10 providing data with DeviceValue of 0 and Status of Unoccupied.

Figure 264 SEN0395 MQTT Edit Tab Setup

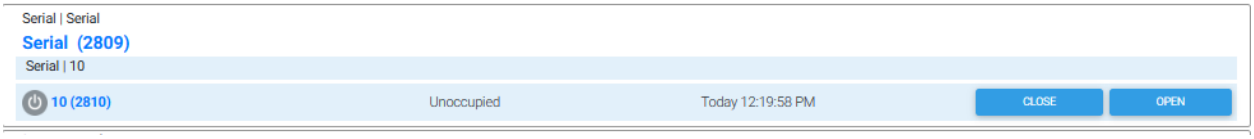


Figure 265 SEN0395 HS Serial Feature

The only data available was the binary status and this was a reliable measure of occupancy. The wiki documentation indicated that the default detection and departure times are 2.5 and 10 seconds around the beam detection. This provides for a stable occupancy status, but lacks usefulness if trying to also use it for low-latency detection.

21.9 InfraRed Motion Direction Sensor

An infra-red LED emitter and receiver pair form an invisible beam where blockage of the beam can be detected in a matter of milliseconds. When multiple receivers are placed side-by-side the beam breaks will occur in sequence based upon the direction of motion.

Standard 5mm LEDs typically have a current capacity of 20 ma. These will produce a sufficiently bright beam to be detected up to about a foot. Larger distances can be detected with increased current, but this requires the emitter to be pulsed to avoid destroying the LED or LED rated at higher current or efficiency. Long distances, such as used with entertainment equipment and garage door safety beams will typically modulate the IR emitter at a high frequency and demodulate at the receiver and may use lenses for focus the beam.

In this implementation a steady current will be applied to the emitter and three side-by-side receivers will be used for motion direction detection. This can be used for pet doors or similar small passages of under 12 inches.

The simplest circuit is a resistor and LED for the emitter and each receiver. The emitter resistor serves as current limit to protect the current handling capacity of the LEDs when conducting. The receiver resistor also protects the LED, but also serves as a pullup resistor on the GPIO inputs to assure a high logic level when the receive LEDs are not conducting (emitter beam is not present). Value is not critical with the tradeoff being switching speed vs. current draw. The circuit is shown in Figure 266.

LEDs are polarity sensitive. The LED cathode goes to most negative side. It is identified as having the shorter lead and shaved/flat side of the LED.

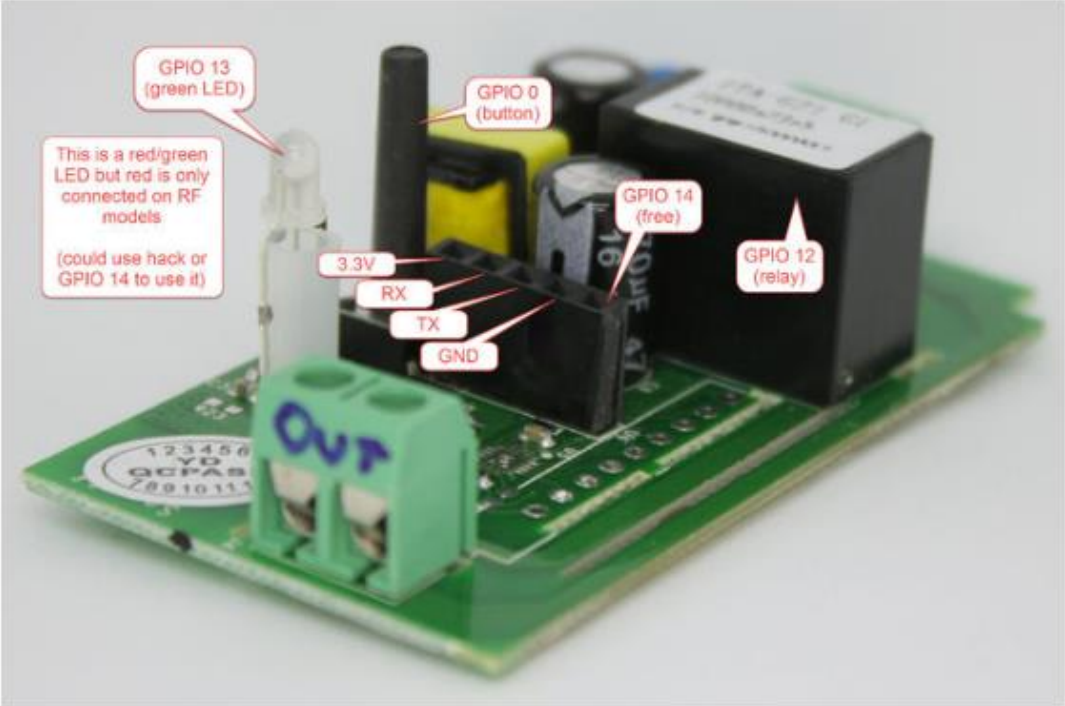
The circuit shows a 180-ohm resistor for the emitter. Depending upon the LED this value will likely be in the 150 to 220 range to give adequate emitter intensity. Consider a 1.2 V drop across the LED then a 180-ohm resistor will yield a current of 3.8/180 = 21 ma. LED emitters for this application are more desirable if their field of view is small so the energy will achieve the most directional/longest beam. The LEDs I used were obtained from Amazon Cylewet 30Pcs 5mm 940nm LEDs Infrared Emitter and IR Receiver Diode for Arduino (Pack of 30) CYT1057 and contained no data sheets. This one looked to give a good field of view for a 12” receiver distance which each receiver having minimal separation between the next.

Experimentation showed that the ESP8266 would not boot into the Tasmota application if one (or more) of the three GPIO inputs were at ground potential when power applied. Since the state of the emitter/receiver is not known, in the general case, at time of power application it is possible that the input will be held low and startup will not complete.

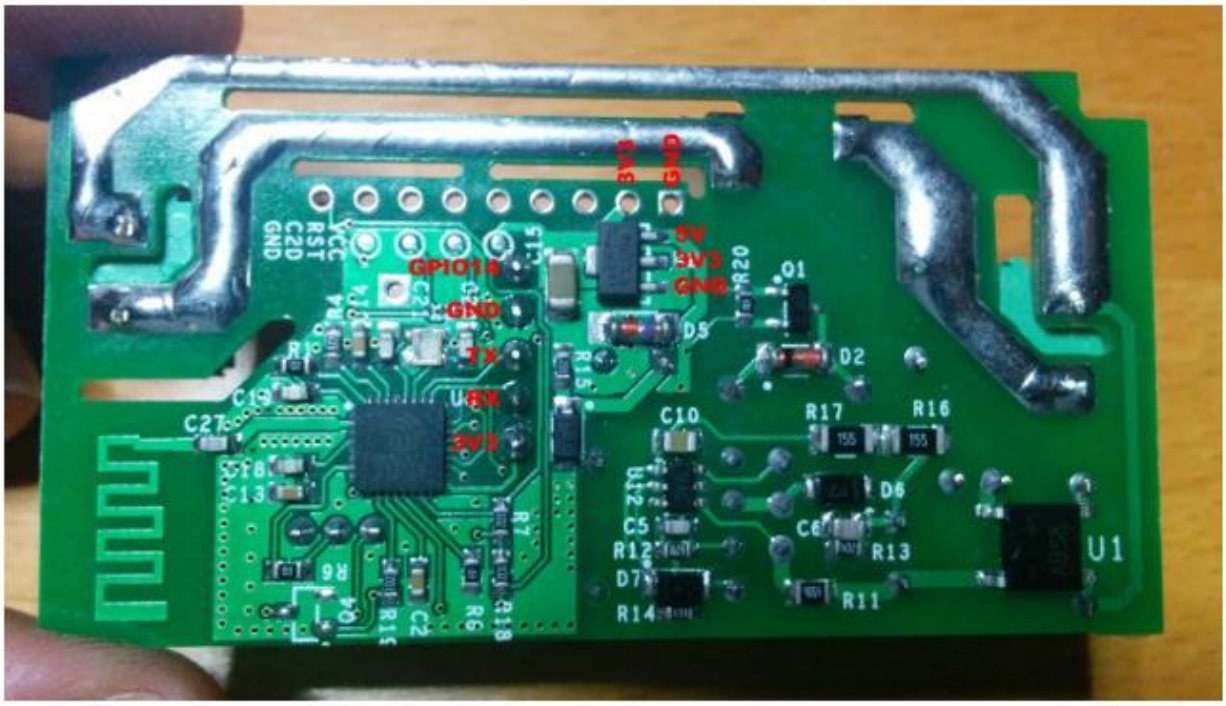
To solve the problem the Sonoff relay is used to switch the IR LED ground connection. At application of power the GPIO12 controlling the relay will always be quiescent so the relay will be open. There will be no ground to complete the receiver IR LED so the three GPIO inputs will all be pulled up to allow the boot to complete. During Tasmota initialization the relay will be set based upon the PowerOnState value (0=relay open/off, 1=relay closed/on, 3=relay at Sonoff power-down state). This parameter will normally be set based upon how the motion direction detector fits into the automation application.

A mod was needed to isolate the relay. See red circle of Figure 224 in the garage door section for board cut that is needed. Rather than using the relay as dry-contact control as was done for garage door, it is being used here to provide switched ground. This means the two relay output pads are wired to ground and switched ground. The switch ground is connected to the cathode (negative) side of the three IR receiver LEDs. It is also brought out to one of the pins of the output screw terminal block. The other pin of the block is connected 5 VDC input of the Sonoff regulator to allow the Sonoff provide controlled power to the emitter LED. (See Figure 231 for 5 VDC location on Sonoff). This allows the emitter to be run from the Sonoff rather than needing an independent wall-wart.

GPIO1, GPIO3 and GPIO14 are available in the center of the Sonoff circuit card. See Figure 267. Header pins are added for flashing and used again to connect the receive IRs. Dupont wires are used to connect the header pin to the resistors for each receive LED. GPIO1 and GPIO3 are used for the serial connection for flashing and debugging. For this application they will be used as discrete inputs so Tasmota needs to be informed that serial communication will no longer be enabled. This is done from browser Console page with “SerialLogging Off”. If flashing is done via serial, then flashing should precede this step.



- GPIO 03 - RX PIN
- GPIO 01 - TX PIN
- GPIO 04 - Second image (must solder wire to pin on ESP chip)
- GPIO 14 - Below GND PIN



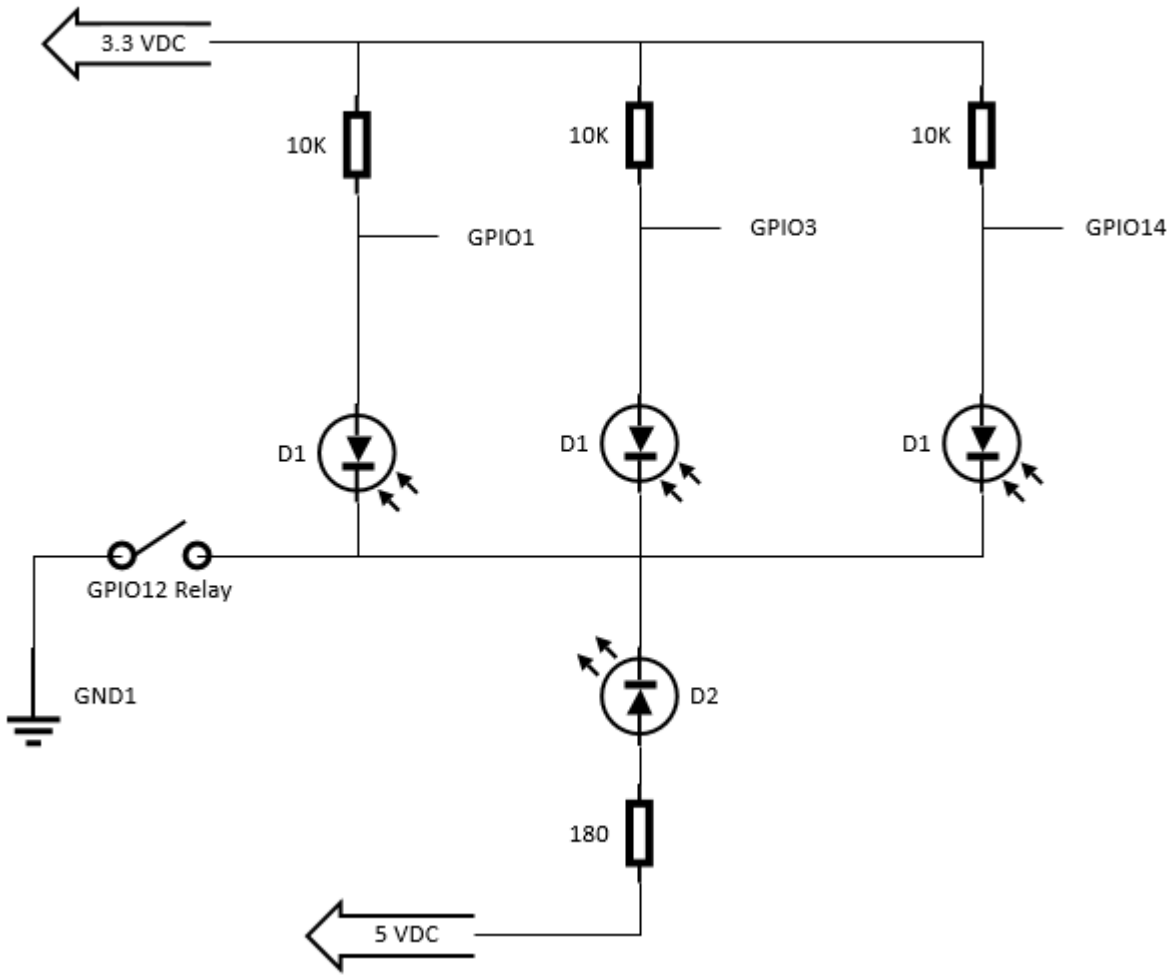


Figure 266 IR Emitter and Receiver Circuit

The Sonoff case (Figure 269) was used to mount the three IR receiver LEDs. Initially they were spaced across the case. Experimentation showed that the single emitter did not have sufficient field of view to span this distance. Putting all three close yielded good results.

The pull-up resistors were solder on the Sonoff circuit card, but could just as easily been soldered on the LED mount in the case. Take care to not run the resistors or connecting wires over the antenna trace on the Sonoff to minimize disturbing the WiFi.

For the emitter a 5V wall-wart that has a barrel connector was initially used. A mate to this connector with screw terminals was used to mount the LED. The 180-ohm resistor was soldered in series with the longer lead anode of the LED. See Figure 270. The alternate emitter source voltage is available from the Sonoff for a more self-contained unit. For this case the 5VDC and switched ground were wired to the output screw terminals of the Sonoff.

This emitter (20 ma) provided a beam distance of about one foot. A higher-powered emitter such as the IR333-A which is available through Digi-key or overseas has a 100 ma capacity. For this implementation I used a 56 ohm resistor to mate with the measured 5.4V of the wall-wart. Distance in this case increased to about three feet and beam pointing became very sensitive to correctly align the emitter and receiver. To further extend the distance and reduce somewhat the sensitivity to beam alignment the implementation shown in Figure 271 was used. Three 100 ma emitters were mounted in parallel. This increased the distance to four feet. Going beyond this the beams because too unstable and would produce false motion reports. Of note is the orientation of the three LEDs made little difference to the range, but found a perpendicular alignment between emitter and receiver seems to make alignment easier.

Pictures of the install are shown in Figure 268 through Figure 271. Note that the IR emitter shown in Figure 270 and Figure 271 is not visible to human eye but is visible to camera.

Figure 267 Sonoff GPIO Pin / Header location

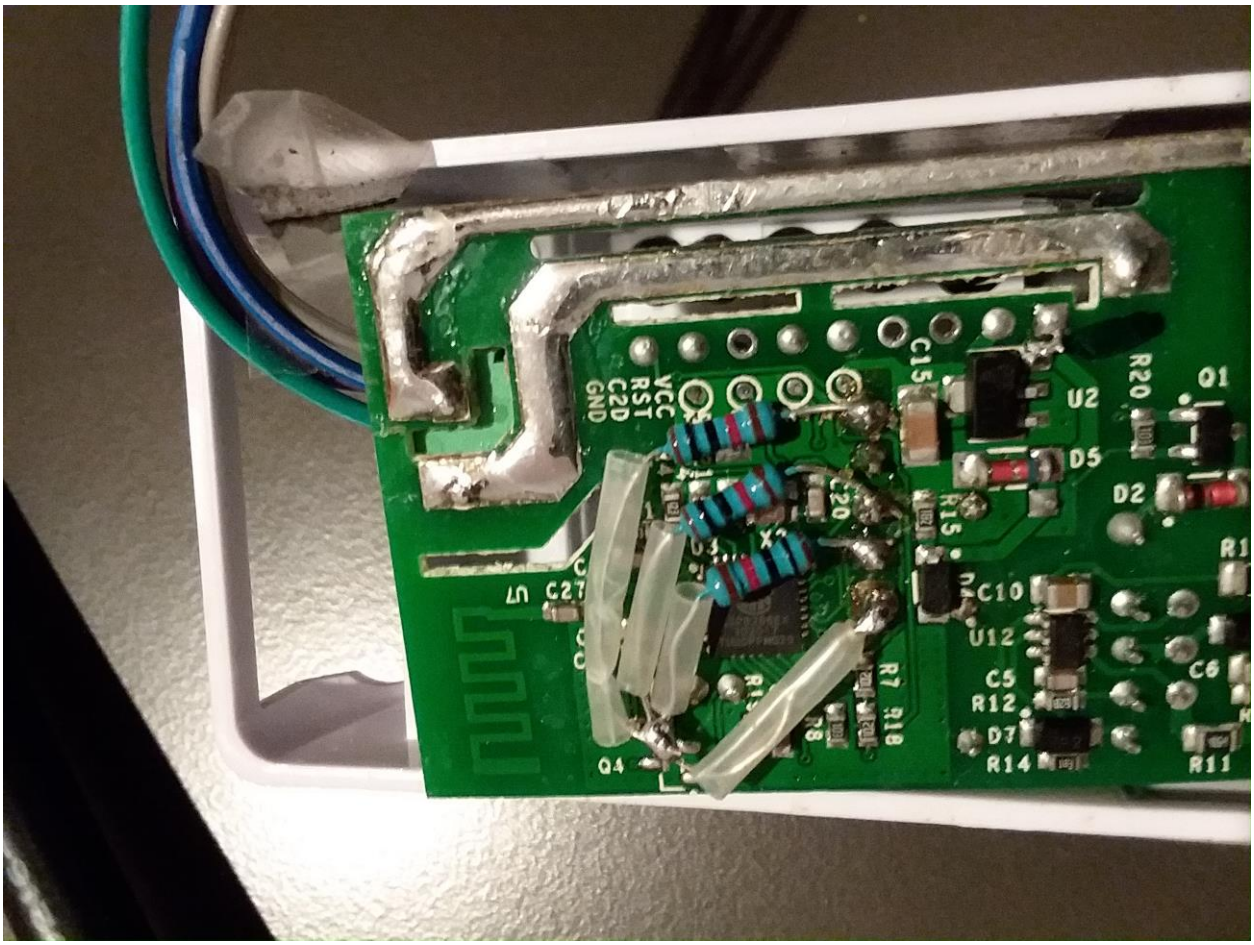


Figure 268 GPIO Input Pull-up for three Receivers

Figure 269 Three IR Receivers Mounted in Sonoff Case

Figure 270 IR LED Emitter

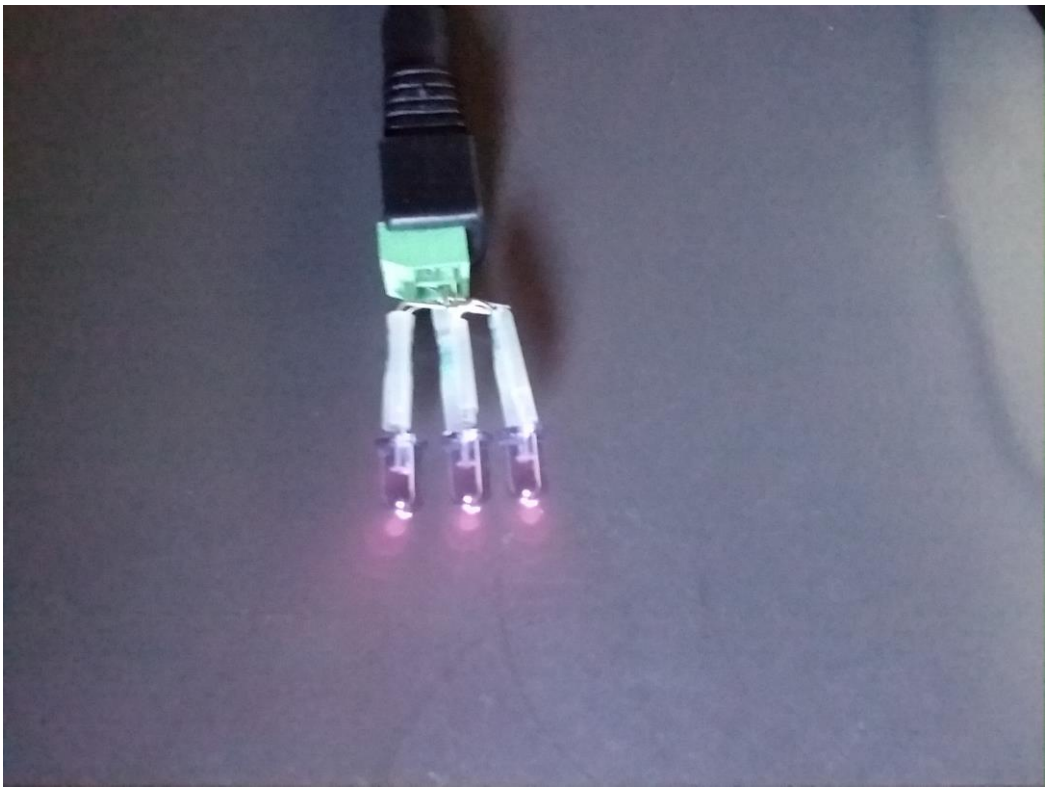


Figure 271 High Power Triplex Emitter

mcsTasmota was updated to 5.9.13h to include three additional input options of MotionL, MotionM, MotionR for Left, Mid and Right position of the receiver LEDs. When Motion types are selected then the inputs are assumed to be normally low until a beam break occurs and then the input becomes high. The Tasmota option SwitchMode has been repurposed when Motion is a selected input. The default 0 will cause “Moving Left” or “Moving Right” to be reported as motion across beams are detected. When set to 1 then there will also be a “No Motion” report after all beams have no blockage. This could be useful when a positive indication is needed that the event is over. When SwitchMode is set to 2 then the additional “Confimed” is appended to the Left or Right messages. It is produced when the middle beam has been broken after one of the other two. In this case a normal report will be “Moving Right”, Moving Right Confirmed” and “No Motion”.

The /STATE topic has also been augmented for motion with an additional “Moving” key. It will take on values between -2 and 3 to reflect the internal state of the motion logic per the following:

-2 Moving Left Confirmed (MotionL ON and then MotionM ON)
-1 Moving Left (MotionL ON)
0 Not Moving (MotionL OFF, MotionR OFF, MotionM OFF)
1 Moving Right (MotionR ON)
2 Moving Right Confirmed (MotionR ON and then MotionM ON)
3 No motion report (MotionR ON, MotionL ON, MotionM ON)

State 3 reflects a fast motion case that can occur because polling [0.05 second SwitchHandler procedure] rather than interrupt driven [CounterUpdate procedure] implementation was used for beam break detection. If real-world application shows polling to be inadequate then the interrupt approach can be done.

The transition from State 3 to state -2 or 2 will reflect the exiting rather than entering state for motion direction. For example, if MotionR beam is the first to be turned OFF then the state transition will be to -2 and the report will be Moving Left Confirmed (or Moving Left if switchmode is not 2)

A debug level log is provided to assess the beam break and resultant state transition such as shown below. The line containing “Beam” starts with the number of interrupts that have been received from beam break state changes since the last time the actions were performed. The beam values are: 0=no beam break, 1= right beam broken, 2= left beam broken, 3= both beams broken. The state values are the same as described above, but biased by 3. This means 1= moving left confirmed, 2 = moving left, 3= no motion, 4 = moving right, 5 = moving right confirmed. Action is taken in the confirmed states.

```
15:43:22 MQT: GarageLight/RESULT = {"POWER":"ON"}
15:43:22 MQT: GarageLight/POWER = ON
15:43:22 MQT: GarageLight/Moving = Moving Left
15:43:22 47 Beam 00000111111111111111111131311113333333333333333 State
333332222222222222222221111111111111111111111
15:43:22 CFG: Saved to flash at F8, Count 29, Bytes 3584
15:43:23 MQT: GarageLight/Moving = Not Moving
15:43:23 20 Beam 33333322222222220000 State 11111111111111113333
```

```
15:43:27 MQT: GarageLight/RESULT = {"POWER":"OFF"}
15:43:27 MQT: GarageLight/POWER = OFF
15:43:27 MQT: GarageLight/Moving = Moving Right
15:43:27 71 Beam
000000000000000000000202000022202202222222202022222222222222222333333333 State
333333333333333333334343333444344344444443434444444444444444444555555555
15:43:27 MQT: GarageLight/Moving = Not Moving
15:43:27 9 Beam 222002000 State 555334333
```

mcsTasmota 5.9.13h does not support multiple DS18B20 sensors to keep the image size below 500K. This allows a one-step OTA flash. mcsTasmota 5.9.13.g supports multiple 1-wire sensors.

It is also possible to configure only MotionL and MotionR. In this case the Confirmed states can only be entered for exit rather than enter beam path case.

The setup of mcsTasmota is the same as with other applications described in this document with the exception of the Module configuration as shown in Figure 272.

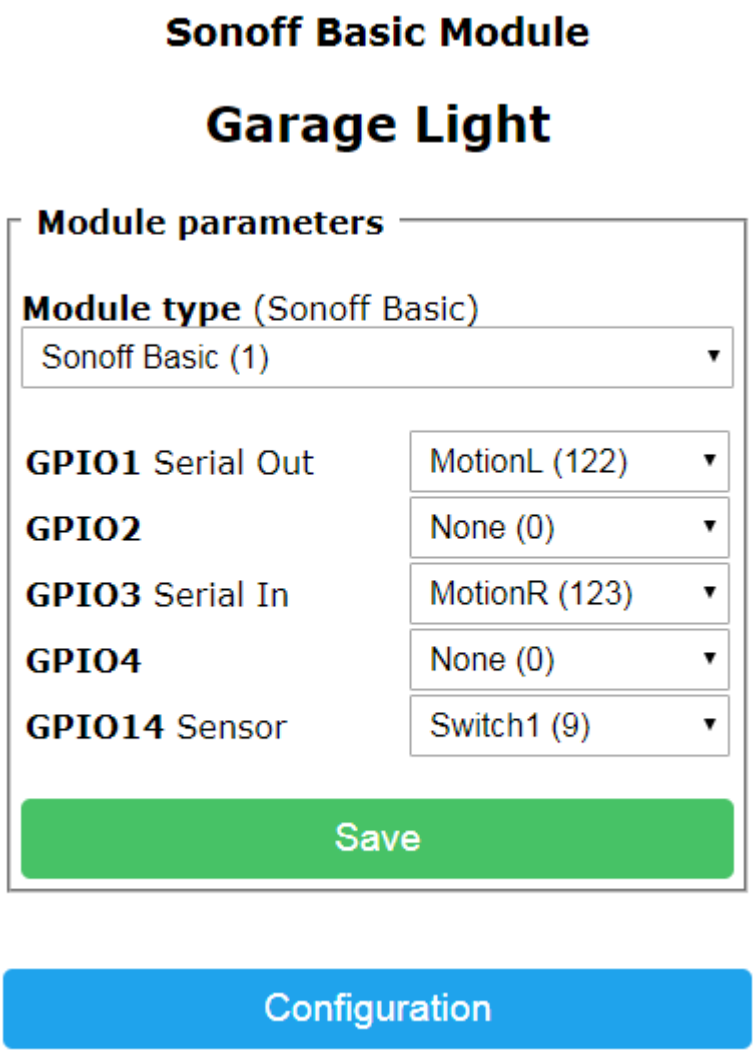


Figure 272 Module Setup for Motion Direction

21.9.1 Motion Direction Version 2

Use of IR emitter for up to about 2 feet yielded a good result. When trying to span 30 inches the operation was not reliable. Just not enough IR light reaches the receivers even with the higher power ones and use of 3 LEDs. This technique for the typical doorway or hallway will need modulation and demodulation of the IR beam. That adds complexity to the electronics so unless prepackaged it is beyond the simple hack.

The easy way to achieve longer distances is with use of coherent light (laser) for the emitter and a photo-resistor (Light Dependent Resistor - LDR) becomes an easy replacement for the IR LED receiver. The laser is readily available in red light which has the advantage of making it much easier to align the light beam over the LDR. The beam now is visible so that has advantages and disadvantages operationally. One sees the laser beam across their body as they pass through the doorway.

The laser is a little more expensive than the IR LED emitter at $6 from Amazon
https://www.amazon.com/gp/product/B0764LS98H/ref=od_aui_detailpages00?ie=UTF8&psc=1

The LDR selected in one with 10K to 50 ohm range also from Amazon for $6 for more that I will ever use
https://www.amazon.com/gp/product/B00H4ZSGXC/ref=od_aui_detailpages00?ie=UTF8&psc=1

Both these parts are direct replacements for the IR LEDs in the circuit so the electrical redesign and construction was trivial. Unsolder old parts and solder the new ones. One change that was made is use of 4.7K rather than 10K resistors for the LDR pull-up resistors. False beam breaks were being seen with the less aggressive resistors.

The mechanical change was a little more difficult because more care was needed on the emitter side to allow fine adjustment of the laser beam. I mounted two laser diodes in a short 2x2 and threaded short screws in two axis of each. Little pieces of shim were used to bias the laser so it was in range of the screw adjustments. Over the 30" span the width of the laser beam came pretty close to the 0.2" width of the LDR. Power for the two lasers was a 5V wall-wart. This mounting is shown in Figure 273.

Figure 273 Laser mounting with adjustment screws

The receiver side of the circuit remained inside of the Sonoff and a cover printed that exposed the LDR, provided a window for the Sonoff LED and button, and a couple mounting holes for a screw into the wall where the doorway was installed. This is shown in Figure 274

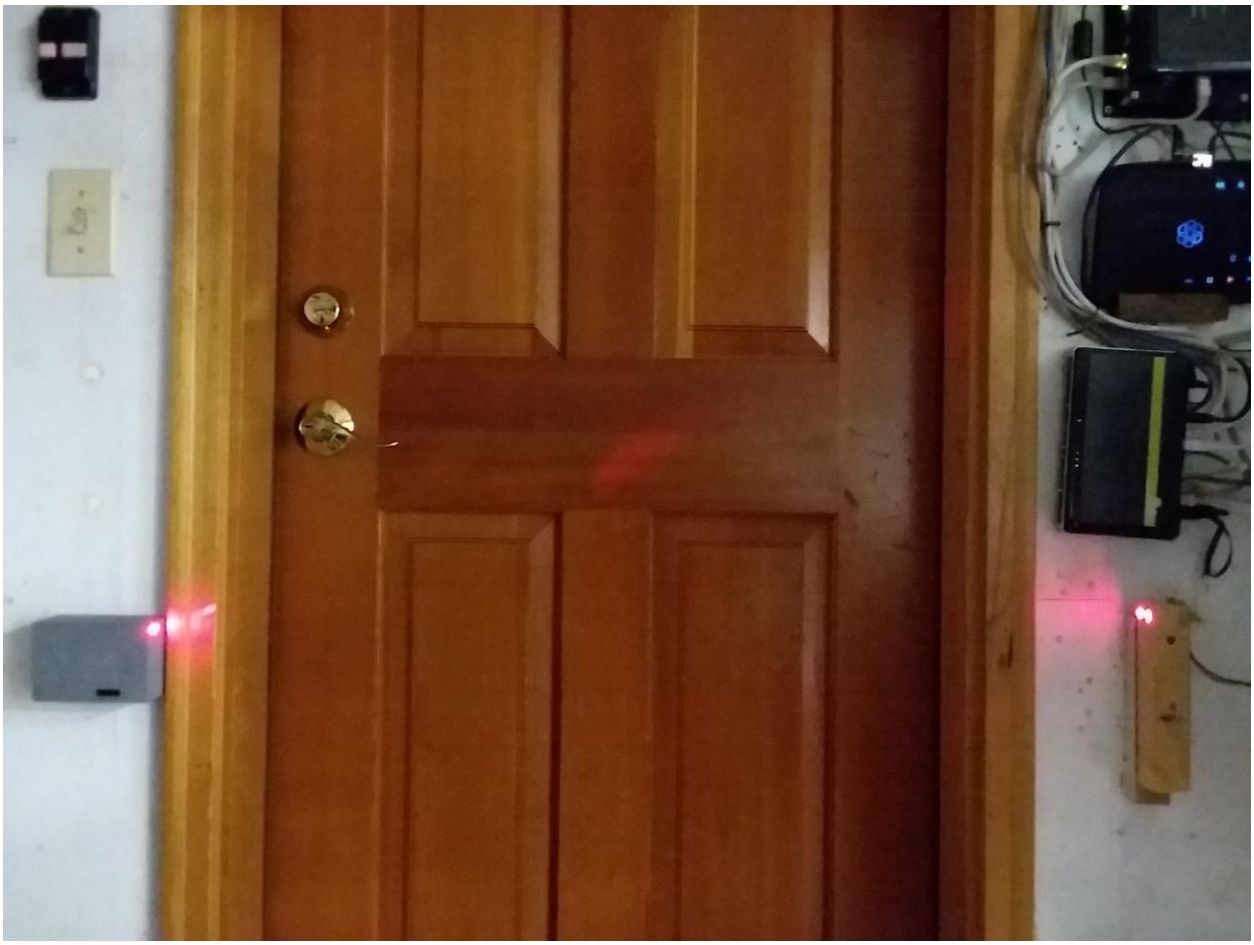


Figure 274 Laser beam break doorway mounting

Wiring between the previously installed light switch and Sonoff was fished through the wall. This consisted of Power, Common, Light, Switch post 1 and Switch post 2.

The Tasmota configuration is shown in Figure 272. The two LDR are on GPIO1 and GPIO3. Most users will not use GPIO4 because it requires solder directly to the ESP8266. The mechanical switch provides a ground connection to GPIO14. Console operation is done to set PulseTime to 1900. This means that when motion is evaluated to be Right (Into Garage) the light will turn on for 1800 seconds (30 minutes). When motion is evaluated to be Left (Into House) there will be a three second delay and the light will turn off. This three second delay is to provide time for the inside light to be turned on while the garage door is still open and light being provided by the garage light. If Switch1 (GPIO14) is in the ON/Ground position then the light will remain ON until the switch is turned OFF. This means that automated control only occurs when the wall switch is OFF. Since logic 0 is ON the console is used to set SwitchMode to 2. PoweronState is set to 3 to persist the light state through power cycle.

A change was needed in the mcsTasmota firmware 5.9.13i to handle the situation where one enters the garage and forgets to close the door and then steps back into the house to grab and shut the door. This activity appears to the firmware as motion into the house so turns OFF the light. The 5.9.13j change augments the state logic to handle motion starting when both beams are blocked with the end result being that the final motion back into the garage will return the light to the ON state. Firmware installed

is mcsTasmota 5.19.13j initially and then 5.9.13k which simplifies the motion state machine to only allow transitions which reset upon both beams no longer broken. Change of direction between beams was problematic. It was later updated to 6.4.1.11 to capture the Arduino core 2.5.0 and then later to mcsTasmota641MotionDirection.bin to streamline the interrupt service processing to avert fatal exceptions.

21.10 Mouse Trap Notification

There are many ways to sense if a mouse is in a mouse trap. To evaluate the effectiveness a mouse trap was obtained from Amazon.com. They come two per order so two traps were configured with mouse presence detection. This approach demonstrates two different detection mechanisms while using a Sonoff Basic to provide a report via WiFi.

ASprint Humane Mouse Trap, Live Trap Catch and Release to Get Rid of Mice, No Kill No Mess, Safe for Kids and Pets, Best Mouse Control -2 Pack
Sold by: ASprint
Return eligible through Nov 7, 2018
$11.77

The completed project is shown in Figure 275 with annotations for the mounting of the sensing components. The wire connections inside the Sonoff are similar to those shown in Section 21.9. Connection points are 5 VDC for IR emitter, GPIO14 for 10K pull-up and IR receiver anode, GPIO3 for 10K pull-up and reed switch. The main voltage was isolated from the relay contacts as was done in garage door application shown in Figure 224, but the bridge wire was not added to connect the relay to the output pins. One side of the relay was connected to the ground circuit pad next to GPIO14. The other side of the relay was connected to the return side of the two IR LEDs and reed switch.

The physical modifications included use of small file under each of the traps to remove a little material to allow the wires to pass from the Sonoff; removal of material from the center base of the Sonoff to make room for the wires that are routed through the base; two holes drilled in one of the traps to mount the two IR LEDs. Hot glue was used to attach the Sonoff base to one of the mouse traps, secure IR emitter, attach magnet on trap door and to secure the bottom of the two traps to a piece of hardboard to allow both to be moved as a single duplex unit.

Figure 275 Wifi Equipped Duplex Mouse Trap

Two discrete inputs of the Sonoff are used to detect presence in each of two mouse traps. One case is an IR beam break and the other case is a reed switch and magnet. The circuit shown in Figure 276 is very similar to the Motion Direction circuit described in Section 21.9. I

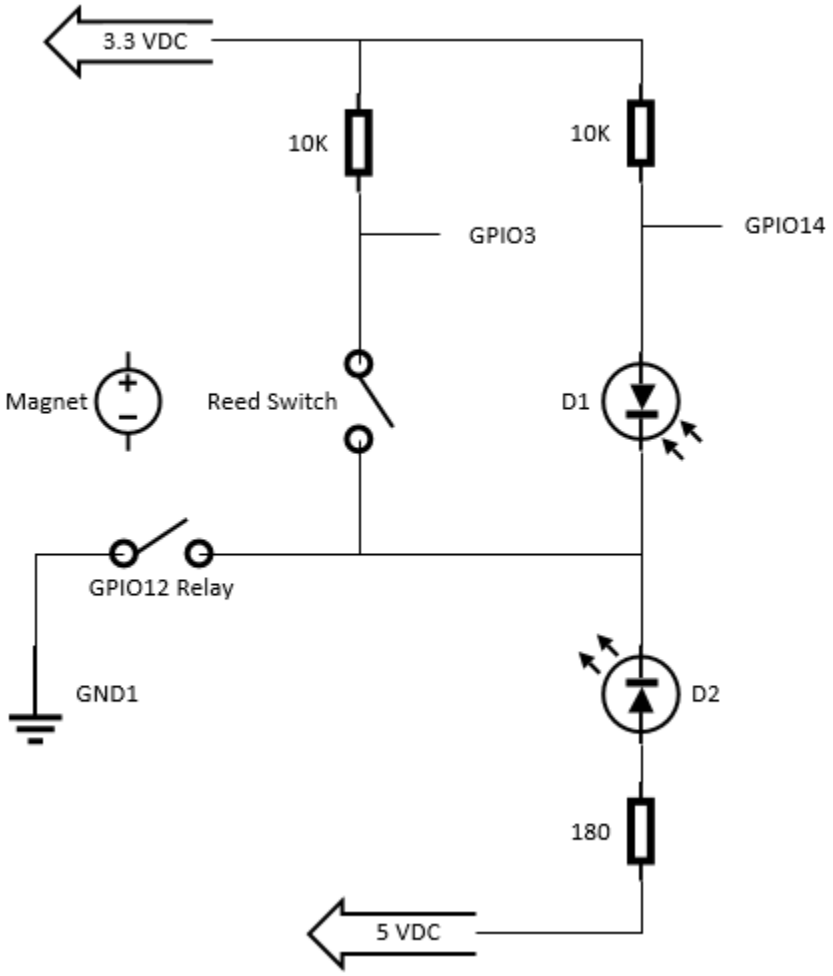


Figure 276 Mouse Presence Detection Circuit

The Tasmota setup, using 5.9.13h of mcsTasmota was done much like the setup for motion direction application described in Section 21.9. This was chosen so that a single topic can be used to report presence of mouse in either trap. The left trap will report "Moving Left"; the right trap will report "Moving Right" depending upon which trap the first mouse enters. The module setup is shown in Figure 277.

From the browser console the following settings were used:

"PowerOnState 1" to enable the sensors after boot,

"SerialLog 0" to allow use of GPIO3 as discrete input.

"SwitchMode1 0" and "SwitchMode2 0" so only event reported when mouse detected.

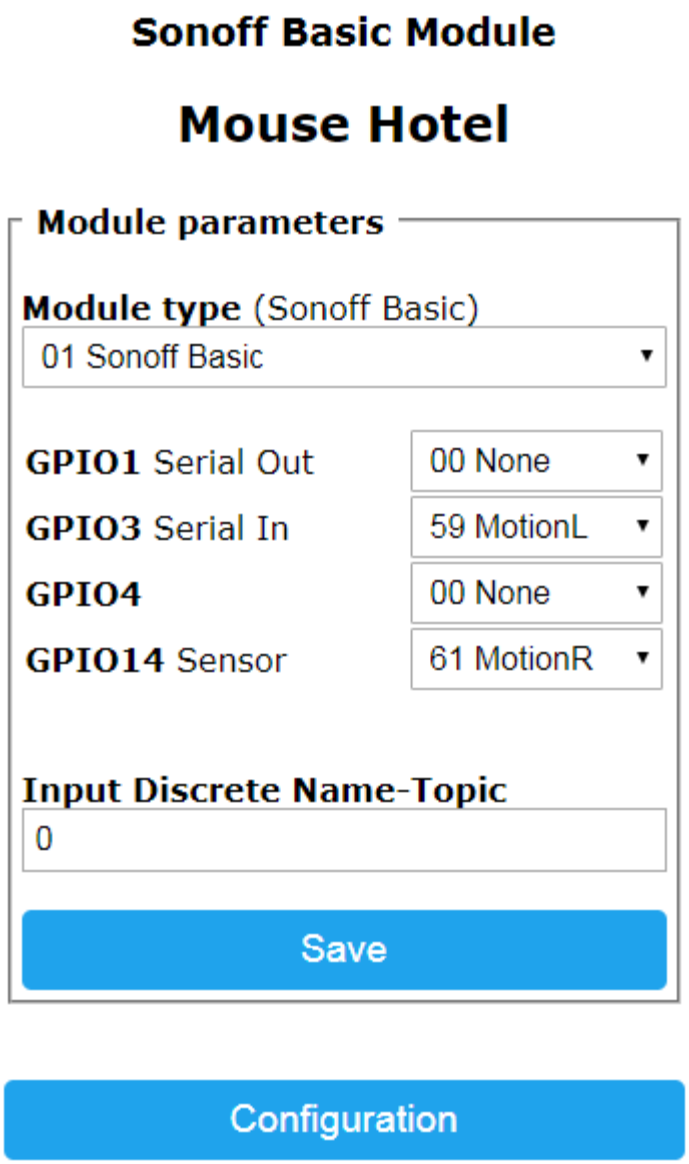


Figure 277 Mouse Detection Module Setup

21.10.1 Mouse Hotel Version 2

Lessons from the original design based upon in-service use is that mounting of the electronics on the mouse traps is a bad idea because following capture of a mouse the trap needs to be removed from service, the mouse released and then the trap washed to remove the trace of mouse odor. There is also no practical reason that separate sensors are needed to identify which trap has the mouse. All that is needed is that notification is that at least one trap has a mouse in residence.

The semi-clear plastic of the trap allows transmission of IR. This means that a single emitter/receiver pair can be used and no hole is needed in the trap to allow the beam to penetrate. The mounting of the

emitter/receiver can be in a cradle outside the traps. Wiring needed for these consists of emitter power, receiver GPIO input, and a common ground. A cheap audio earplug uses a 1/8 " 3-conductor connector so makes a easy choice as the wiring between the trap cradle and the Sonoff. The earplugs are cut off and wire soldered to the Sonoff at the same points where the prior wiring was routed. The 1/8" female connector mounted on the cradle to make wiring separation easy in the future, but this was not essential. Figure 278 shows the cradle and the wiring. The two traps are loose within the cradle so are easy to remove to service the mouse.

The Tasmota configuration has only MotionR or MotionL, but not both. The IR sensor is connected to GPIO14 now that only one input is needed. PowerOnState still set to 1 so relay is engaged to power the IR emitter. SwitchMode 0 so only Left motion reported, hence providing a latch function or to 1 to have both moving and not moving events reported.

Figure 278 Mouse Hotel Revision 2

21.11 Notification Frame

A common problem is some form of dashboard that is a central place where events that need attention can be displayed. It is possible to rely on smartphone notifications if one is tethered to their phone. It is also possible to use a tablet or similar display with HSTouch, Imperihome other UI and then construct a dashboard.

The approach described here is to repurpose an old digital picture frame and use NeoPixels (Individual RGB LED Strip) for the notification. The mechanical parts of the frame were retained and the electrical parts replaced by a Wemos D1-Mini running Tasmota. The IR sensor, buttons, power connector was retained. A DS18B20 sensor and the ambient light (LDR) sensor were added. Figure 279 provides the visual of the project.

Figure 279 Notification Frame

Two strips of 8 LEDs were glued to right and left edge of the picture frame. The original picture frame screen was removed and replaced with black hardboard onto which a glue stick was used to attach the printed labels for each LED. This label can be easily updated by printing and attaching a new sheet.

The organization of the frame is with eight LEDs on the left for events that need immediate attention. They will take on Green for OK, Red for Failure, Yellow for update not received. The right column is for

activities that eventually need attention. They use the same color scheme as the left column except the OK state is blue rather than green.

Two buttons on the side of the frame are connected to the Wemos ESP8266. One will reset one column of LEDs to the OK state. The other will do the other column of LEDs. IR can also be used to reset individual LEDs. Tasmota Rules are used to map an IR button and frame button to a LED(s).

rule1	
on IrReceived#Data=FF9867 do LED16 001000 endon	Red
on IrReceived#Data=FFD827 do LED15 001000 endon	Green
on IrReceived#Data=FF8877 do LED14 001000 endon	Dk blue
on IrReceived#Data=FFA857 do LED13 001000 endon	W
on IrReceived#Data=FFE817 do LED12 001000 endon	Dk orange
on IrReceived#Data=FF48B7 do LED11 001000 endon	Lt green
on IrReceived#Data=FF6897 do LED10 001000 endon	Blue
on IrReceived#Data=FFB24D do LED9 001000 endon	Flash
on IrReceived#Data=FF02FD do LED8 000010 endon	Orange
on IrReceived#Data=FF32CD do LED7 000010 endon	Lt blue
on IrReceived#Data=FF20DF do LED6 000010 endon	Dk purple
on IrReceived#Data=FFD0FF do LED5 000010 endon	Strobe
on IrReceived#Data=FF38C7 do LED4 000010 endon	Peach
on IrReceived#Data=FF28D7 do LED3 000010 endon	aqua
on IrReceived#Data=FFF00F do LED2 000010 endon	Purple
on IrReceived#Data=FF30CF do LED1 000010 endon	fade

Other IR Remote buttons not used

FF906F	Up
FFB847	Down
FFF807	Off
FFB04F	On
FF38C7	Yellow
FF28D7	Dk teal
FFF00F	Pink
FF30CF	smooth

The frame button part of the rule is:
on button1#state do backlog LED16 001000; LED15 001000; LED14 001000; LED13 001000; LED12 001000; LED11 001000; LED10 001000; LED9 001000 endon
on button2#state do backlog LED8 000010; LED7 000010; LED6 000010; LED5 000010; LED4 000010; LED3 000010; LED2 000010; LED1 000010 endon

A rule needs to be on a single line, but no more than 512 characters. This means the rule needed to be split up into three due to the character limit. In format used by Tasmota Console or MQTT Topic are the line to enable the rule and the one line rule:

rule1 on IrReceived#Data=FF9867 do LED16 001000 endon on IrReceived#Data=FFD827 do LED15 001000 endon on IrReceived#Data=FF8877 do LED14 001000 endon on IrReceived#Data=FFA857 do LED13 001000 endon on IrReceived#Data=FFE817 do LED12 001000 endon on IrReceived#Data=FF48B7 do LED11 001000 endon on IrReceived#Data=FF6897 do LED10 001000 endon on IrReceived#Data=FFB24D do LED9 001000 endon

rule2 on IrReceived#Data=FF02FD do LED8 000010 endon on IrReceived#Data=FF32CD do LED7 000010 endon on IrReceived#Data=FF20DF do LED6 000010 endon on IrReceived#Data=FFD0FF do LED5 000010 endon on IrReceived#Data=FF38C7 do LED4 000010 endon on IrReceived#Data=FF28D7 do LED3 000010 endon on IrReceived#Data=FFF00F do LED2 000010 endon on IrReceived#Data=FF30CF do LED1 000010 endon

rule3 on button1#state do backlog LED16 001000; LED15 001000; LED14 001000; LED13 001000; LED12 001000; LED11 001000; LED10 001000; LED9 001000 endon on button2#state do backlog LED8 000010; LED7 000010; LED6 000010; LED5 000010; LED4 000010; LED3 000010; LED2 000010; LED1 000010 endon

rule1 1
rule2 1
rule3 1

The Wemos A0 pin is connected to the LDR with a 100K resistor added to bias A0 toward 3.3V. The intention was to change the intensity of the LED between day and night viewing. The LEDs during daytime are set to 16% and setting below this level does not help much with intensity. It does more color bias at such low levels. Because of this the LEDs remain at the same brightness day and night.

The DS18B20 was mounted in the back of the frame at the bottom. This is the best location to be isolated from any heat that may build up with the LEDs or Wemos. Temperature and A0 light intensity are reported via MQTT SENSOR topic at 300 second intervals. STATE topic is also delivered.

15:27:16 MQT: Notify/STATE = {"Time":"2019-01-21T15:27:16","Uptime":"0T00:05:18","SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"POWER":"OFF","Dimmer":10,"Color":"191919","HSBColor":"0,0,10","Channel":[9,9,9],"Scheme":0,"Width":1,"Fade":"OFF","Speed":1,"LedTable":"OFF","Wifi":{"AP":2,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":11,"RSSI":80}}

15:27:16 MQT: Notify/SENSOR = {"Time":"2019-01-21T15:27:16","ANALOG":{"A0":452},"DS18x20":{"DS1":{"Type":"DS18B20","Address":"28E36E2C000000BF","Temperature":80.7}},"TempUnit":"F"}

The LED color is normally controlled from other locations such as HS. Tasmota is expecting a Topic of Notify/cmnd/LED1 RRGGBB for the case of the first LED (upper right). The 16th LED is on upper left.

Something like HS or Node Red can serve the logic function to group various low level status into the single notification status as shown on a specific LED. Tasmota Rules can be used when there is a direct mapping of the Tasmota device with a notification LED.

Examples of both are provided below.

Water leak detection is done by Zigbee Aqara SJCGQ11LM that is received via MQTT Topic zigbee/0x00158d0002334682:water_leak and mapped into HS Device Water|zigbee|Anthem_Laundry_4682:water_leak. Nine similar sensors are located at various locations within the house. When any of these nine report true then the HS event looking for this sends MQTT Topic "notify/cmnd/LED16 100000" to cause the upper left LED to turn red. Other events are looking for the LWT going offline for each sensor for a period of greater than one hour. If any of these trigger then the MQTT Topic "notify/cmnd/LED16 101000" will be sent to turn the LED yellow.

Washing machine power use is being monitored by Sonoff S31 power plug that has Tasmota firmware installed. A typical cycle was monitored and graph viewed shown in Figure 280. From this it could be seen that one could be assured a load of wash was completed when power use was a low level (e.g. 100 watts) for 10 minutes. This information allowed a Tasmota rule to be formed:

```
Rule2 0
Rule2 on ENERGY#POWER>100 do backlog publish notify/cmnd/LED1 001000; rule2 0 endon

Rule1 on ENERGY#POWER>100 do ruletimer1 600 endon on rules#timer=1 do backlog  publish
notify/cmnd/LED1 100000; rule2 1 endon
Rule1 1
```

Rule2 is used to reset the notification LED to green when a load has started. Rule1 is used to detect when the machine has stopped for 10 minutes to set the LED to red. Normally Rule1 is always active, Rule2 is inactive and only becomes active after Rule1 has triggered and the LED set to red state.

At the end of the laundry day the Washer LED notification will show red as there was no subsequent running of the washer. Resetting it green could be done manually via IR or button on the frame or further automation added that will sense when the washer lid has been raised or perhaps at midnight if one is not concerned for forgetting the washing the entire day. The Zigbee Aqara tilt sensor DJT11LM would be a good choice for the sensing the lid as it is small and no wires needed. Just a new CR2032 coin cell would need to be replaced every year.

Another approach to using Tasmota rules is to use Events in HS to orchestrate the control of the notify LED for the washing machine.

Load Defined Chart	▼
Absolute Date Range (Start,End) or (SingleDay)	1/12/2019
Absolute Start and End Times	Start: 11:20 End: 12:05
Relative Start Date-Time	(format dd hh:mm:ss)
Topic Selector	Include Non-Accepted Messages
Left Axis Topic/Item (Select Topic or Device)	WashingMachine/SENSOR:ENERGY:Power ▼ ▼
Right Axis Topic/Item (Select Topic or Device)	▼ ▼
Left Axis Min & Max	Left Min: Left Max:
Right Axis Min & Max	Right Min: Right Max:
Save Chart Definition	Save

Show Selected Chart

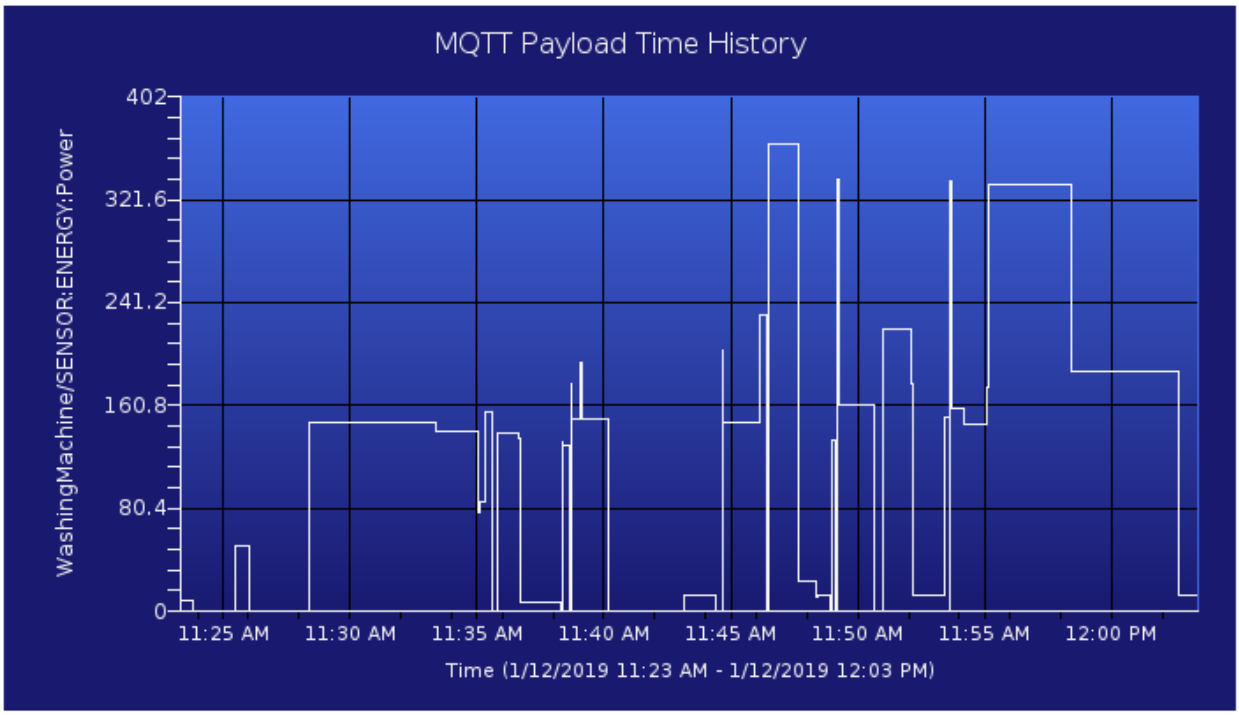


Figure 280 Washing Machine Power Use

The physical modification to the old picture frame are partially shown in Figure 281. Non-conductive tape was placed over the frame's original circuit board. Only the physical jack for power supply was used from this circuit board. One can see the black and white wires coming from the bottom of the circuit board to the pins on the Wemos. Wires are also visible from the pushbuttons that are located at the top of the frame's case. The wires going to the right of the case is for the addition of the DS18B20 temperature sensor. From the left side are the wires for the IR sensor, LDR and the two LED strips on top on bottom of the front part of the frame. All wiring was stitched onto the breakout pins of the Wemos. The original frame connectors were used or the IR sensor and panel switches. Otherwise dupont wires were used with the male and female connection protected from separation with some electrical tape.

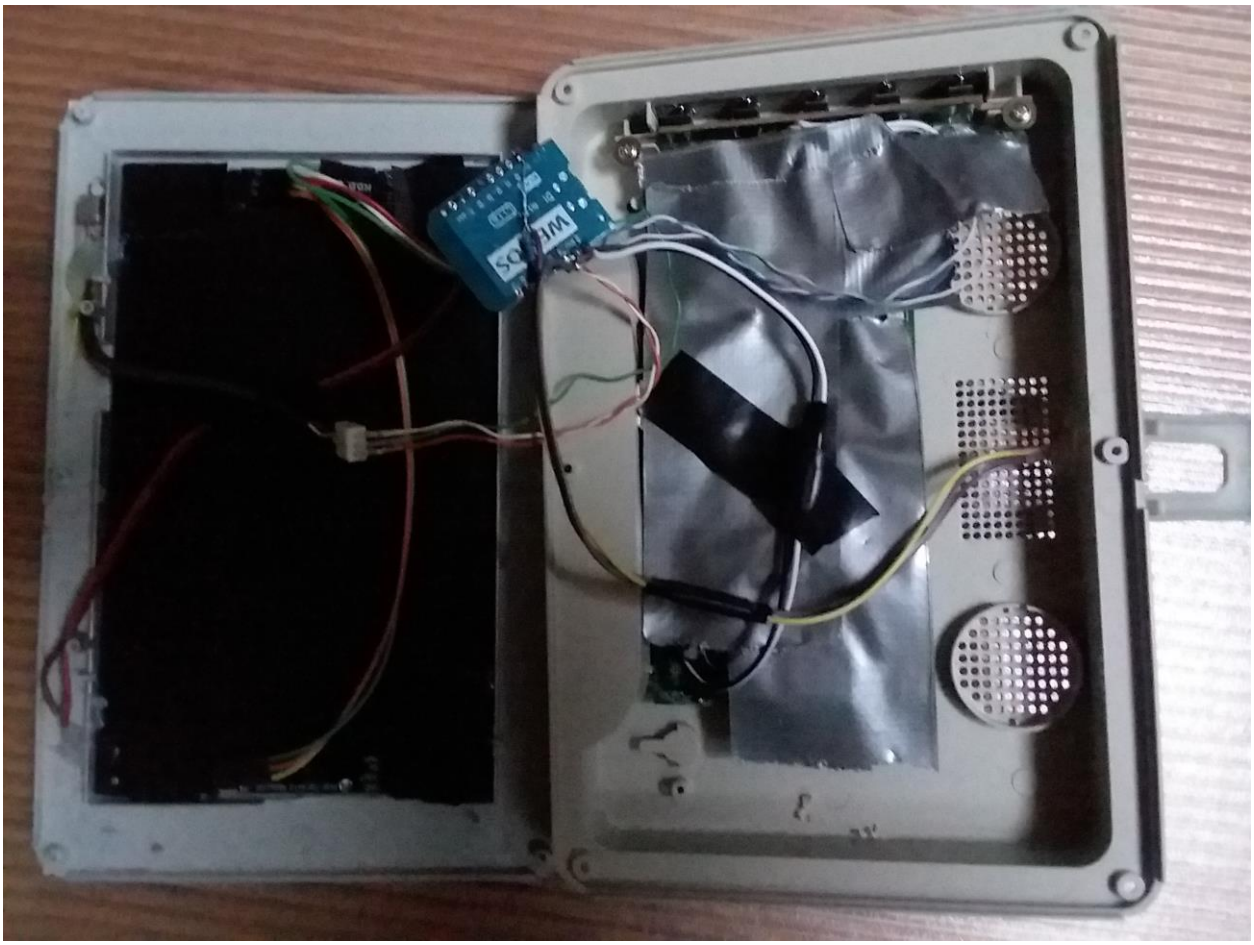


Figure 281 Notification Frame Internal Wiring

The firmware installed with mcsTasmota 6.4.1.6 which is a derivative of the 6.4.1.5 Tasmota distribution. The change that was made was a provision to restore the LED colors upon a power cycle. Stock Tasmota had no provisions for this with the WS2812. Power up state was always all pixels off. To activate this feature the "poweronstate 3 " is used in Console of MQTT message.

The configuration setup of this configuration is captured in Figure 282 and Figure 283.

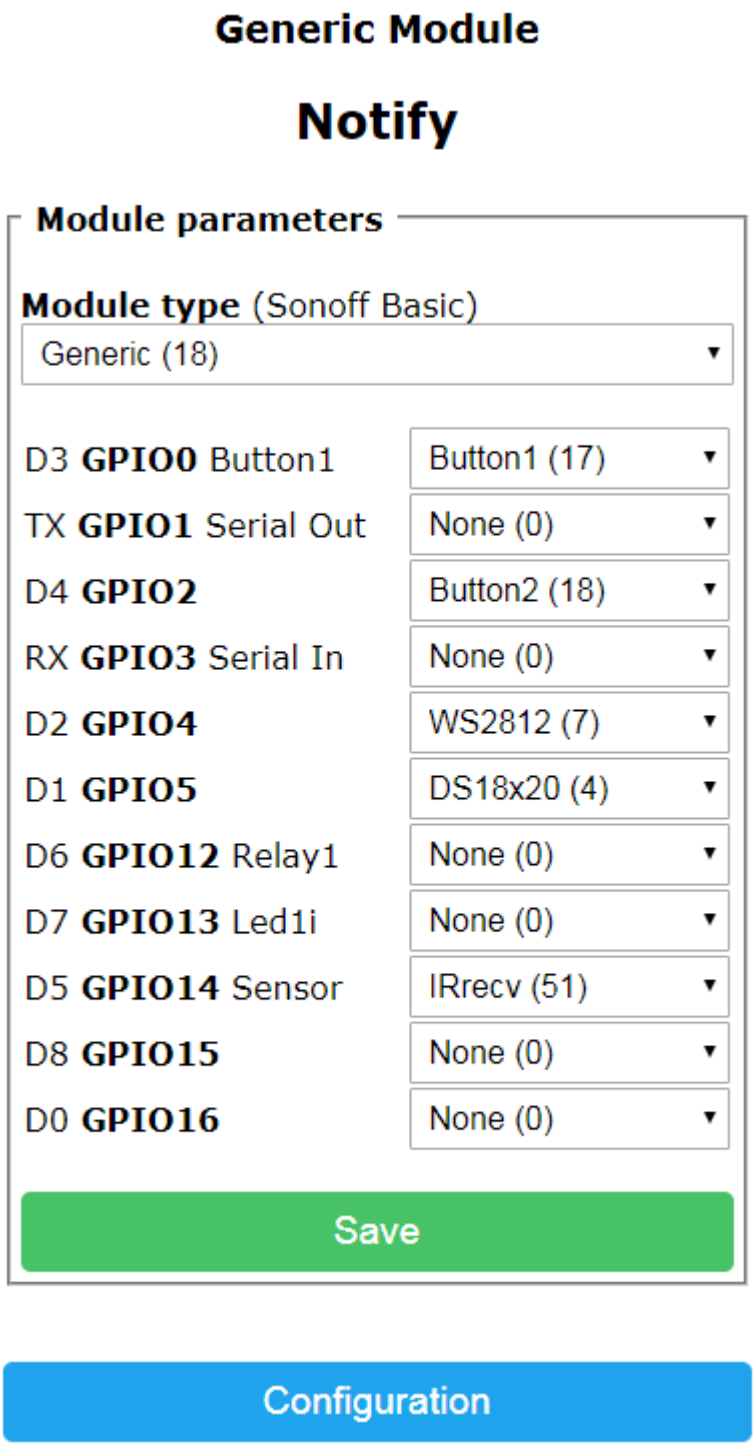


Figure 282 Notify Module Configuration

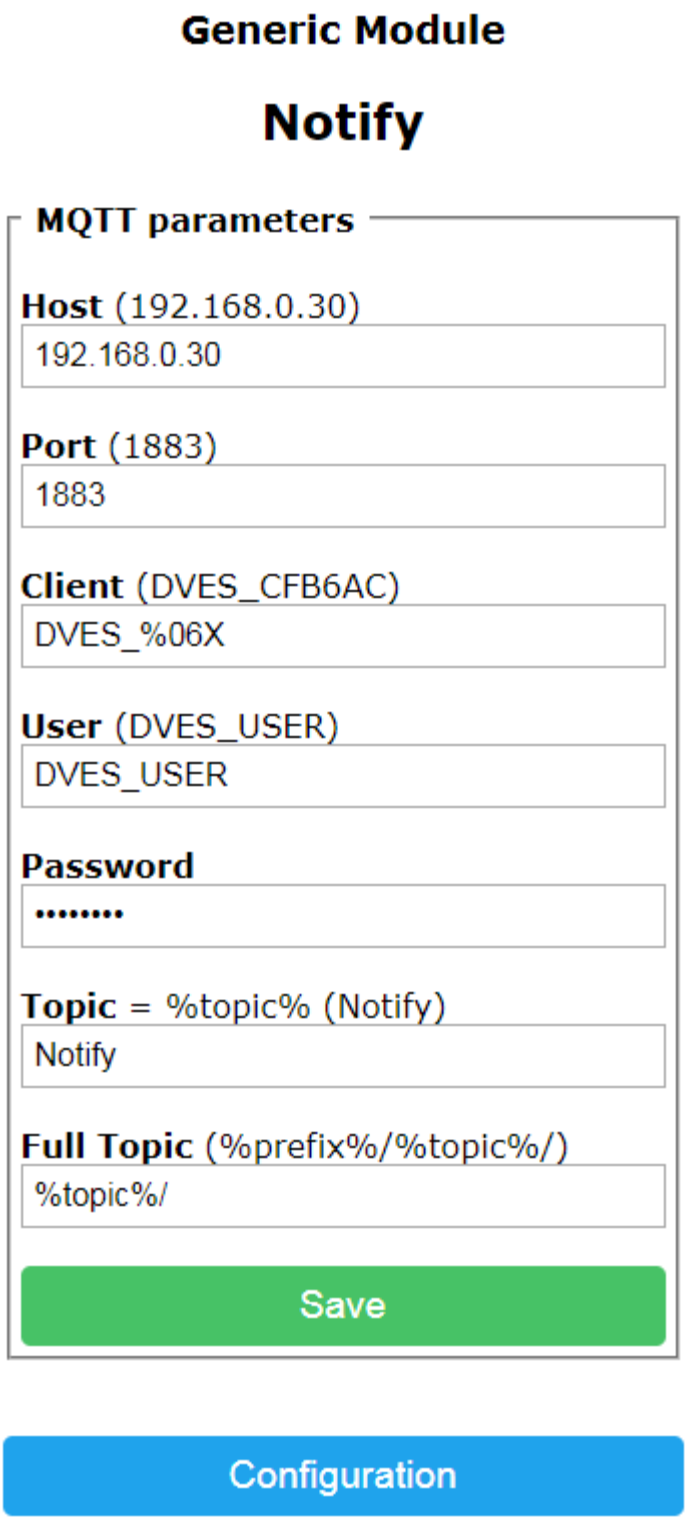


Figure 283 Notify MQTT Configuration

21.12 CID Robocall Blocker

Robocallers are a nuisance during the day, but especially obnoxious when they come during morning or evening hours when one is sleeping. One solution is turn off the ringer when one goes to bed, but this prevents emergency calls from family members from being recognized.

To deal with this problem a modified Sonoff Basic was employed to recognize CID from family members and allow these calls while blocking all other calls during the sleeping hours. Tasmota firmware 6.4.1.5 enhanced with CID recognition was used in the Sonoff.

The CID components consist of a NetCallerID that reports CID information via RS232

Figure 284 NetCallerID

and RS232 to TTL voltage level translator

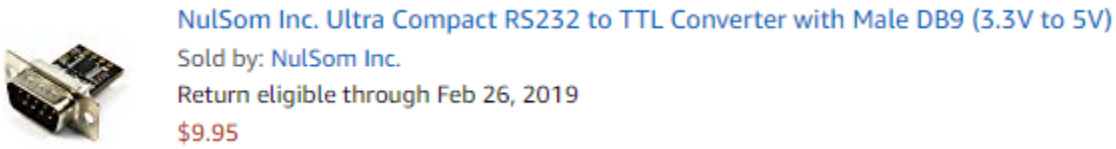


Figure 285 RS232 to TTL Translator

On the translator the RTS and DTR DB-9 pins 7 and 8 were wired to 3.3V just in case the NetCallerId used RS232 flow control.

The Sonoff pickoff points for the serial interface to RS232 are on the four-pin header shown in Section 21.9. Dupont wires were used to make the connections. The Sonoff case was modified with a cutout in the top to allow the four wires to penetrate. The Sonoff circuit board was modified per the red circled area of Figure 224 to convert the relay from switching mains power to completing a dry contact connection.

Two design options were available at the bedroom phone. One was to open the phone and run wires to/from the ringer. This has the advantage of never inhibiting outbound calling, but disadvantage of a wiring mess and difficulty of opening the phone.

The second is the approach is to modify the cable with the RJ12 connectors that plug into the phone so that one of the two wires goes through the Sonoff relay. This is the approach that was taken. An Amazon Spot exists next to the phone so that if an outbound call is desired during nighttime then “Alexa turn on phone” is used to close the relay for a call of up to 10 (pulsetime) minutes.

A simple enclosure was printed to hold all the pieces. Inside the box is the NetCallerID and its wallwart that is plugged into the end of an extension cord. Additional wires soldered to the extension cord to provide power to the Sonoff. The RS232 voltage translator was wrapped in electrical tape to protect the circuit and assure the Dupont wire connections do not come off the header. A hold in the box allowed the power extension cord, telco connection wire and phone connection wire to penetrate the box.

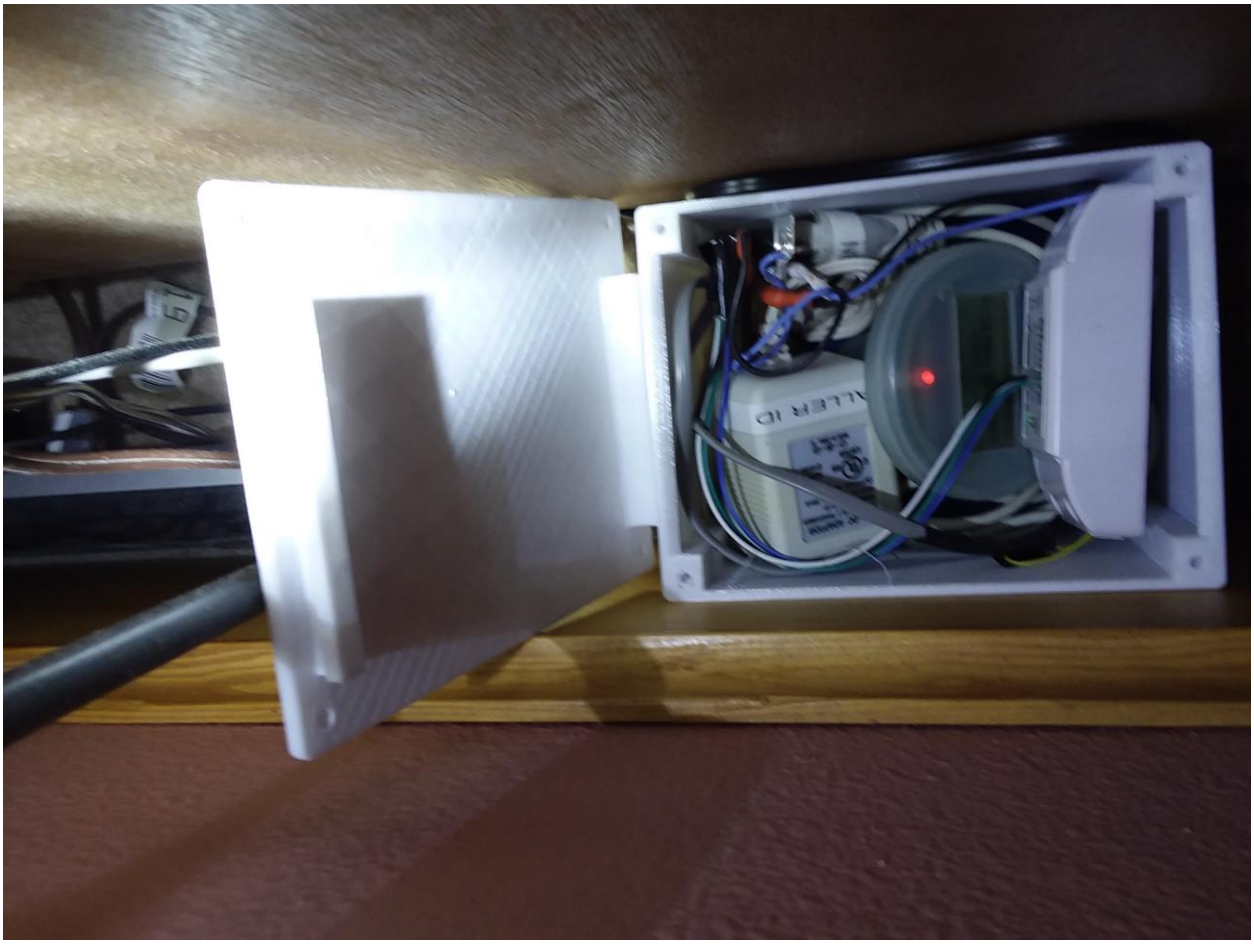


Figure 286 Sonoff CID Enclosure

The Tasmota firmware required an update to support a CID whitelist. This is mcsTasmota version 6.4.1.7 (http://mcsSprinklers.com/mcsTasmota.zip). It was later updated to provide direct publish of CID information to LED Messaging sign in mcsTasmota641Sign.bin. Provisions exist for up to 20 10-digits numbers in the whitelist. A number in the whitelist that is received will result in the relay being closed

for 10 (pulsetime) minutes to allow the ring and then the audio connection. Note pulsetime is a run-time configurable setting.

This modification consists of two elements. One is to enter the numbers that form the whitelist. This is with the Console (or MQTT/HTTP) CID# 1234567890 where # is between 1 and 20 and 1234567890 is the ten-digit telephone number. The Tasmota backlog command can be used to enter the entire whitelist with one command. The mcsMQTT publist is another way.

The second is to decode the serial stream from NetCallerID which consists of "NMBR1234567890." Where 1234567890 is the incoming CID and period is the termination. Any ten-digit number that does not match one in the whitelist will result in the relay being open. Other data from the NetCallerID is ignored.

Tasmota forwards via MQTT Topic/RESULT when serial data is received using "SerialRecieved" as JSON key. If it contains "NMBR" then additional CID information is provided. The Power status on the CID is 0/1 to reflect the state of the relay or equivalently the number being in the whitelist.

{"SerialReceived":"###DATE02010920...NMBR4251234567...NAMEAnn+++",
"CID":{"Number":"4251234567","POWER":"0","Name":"Ann"}}

The remainder of the logic to achieve the desire modes of control was done using the following Tasmota Rules:

```
rule1 on system#boot do backlog serialdelimiter 13; baudrate 4800; serialsend AT#CID=1;
pulsetime 600; poweronstate 4 endon

on Time#Minute=1320 do poweronstate 0 power off endon on Time#Minute=600 do
poweronstate 4  endon

on Time#Initialized do backlog event before10pm=%time%; event after10am=%time% endon

on event#before10am<600 do poweronstate 0 power off endon

on event#after10pm>1320 do poweronstate 0 power off endon
```

On power-up the serial connection is assured to match the expectations of NetCallerID with baudrate of 4800, <cr> as the line termination, and "AT#CID=1" as the initialization string to the NetCallerID to have it report the CID via serial. At this time the pulsetime is set to 10 minutes which is the duration of the relay being closed after engagement. Poweronstate is set to 4 to initially close the relay and not allow it to be changed. Poweronstate is later modified based upon the time of day. At 10 PM (1320 minutes) poweronstate is set to zero to allow it to be changed and then relay turned off. At this point the phone is no longer connected to telco. At 10 AM poweronstate is set back to turn relay on and keep it on.

The remaining three lines in rule1 deal with the situation of a power cycle at some time. It will set the poweronstate to the desired value based upon time of day when the power cycle happened.

The new commands recognized by Tasmota to support the CID are CID, CIDTopic and CIDRow as shown in Table 5.

Table 5 CID Tasmota Commands

Command	**Description**
CID# number	Add number to whitelist # is index in range 1 to 20 Number is 10-digit telephone number e.g. Phone/cmnd/CID12 12345467890
CIDTopic topic	Define topic to send CID info to LED Messaging Sign e.g. Phone/cmnd/CIDTopic LedSign/cmnd/TEXT8
CIDRow row	Define row of LED Messaging Sign on which to show CID number and name Row in range of 1 to 10 e.g. Phone/cmnd/CIDRow 2

Other Tasmota setup is standard as shown in subsequent pictures. Note that the module type definition does not matter.

Even though the defined rule is executed at system boot, it appears that Tasmota does not allow the serial use to be defined at that time and accepts it only from console or perhaps MQTT. The following should be used in the console to have the serial port used by the NetCallerID rather than the standard user interface.

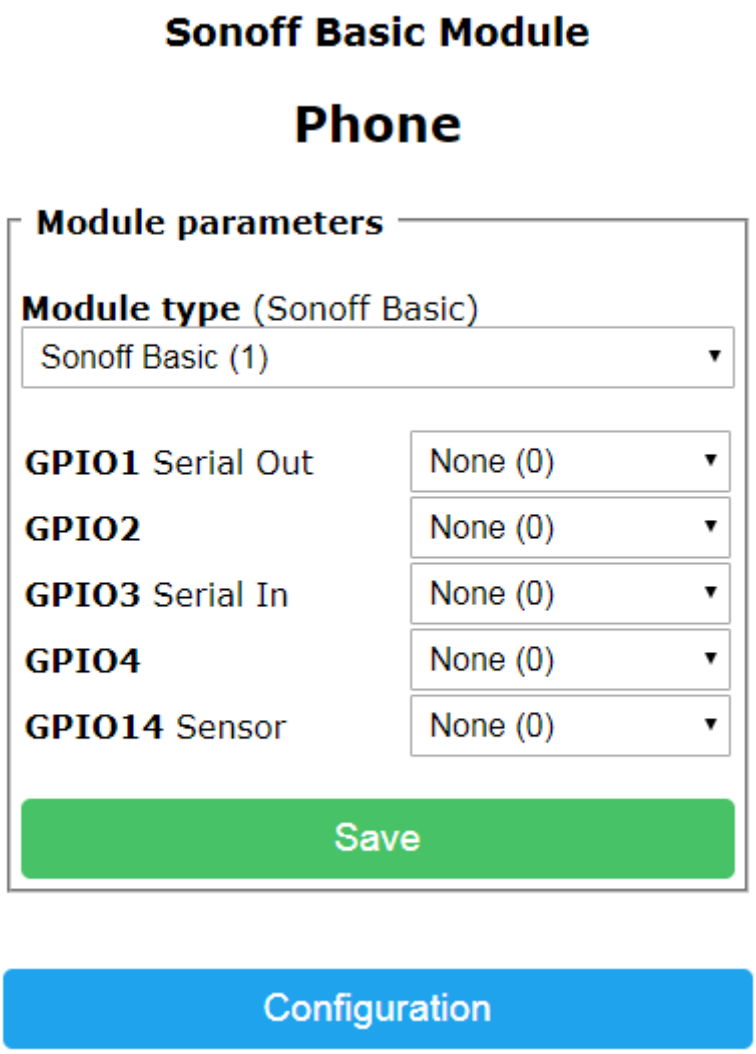


Figure 287 Sonoff CID Module Configuration

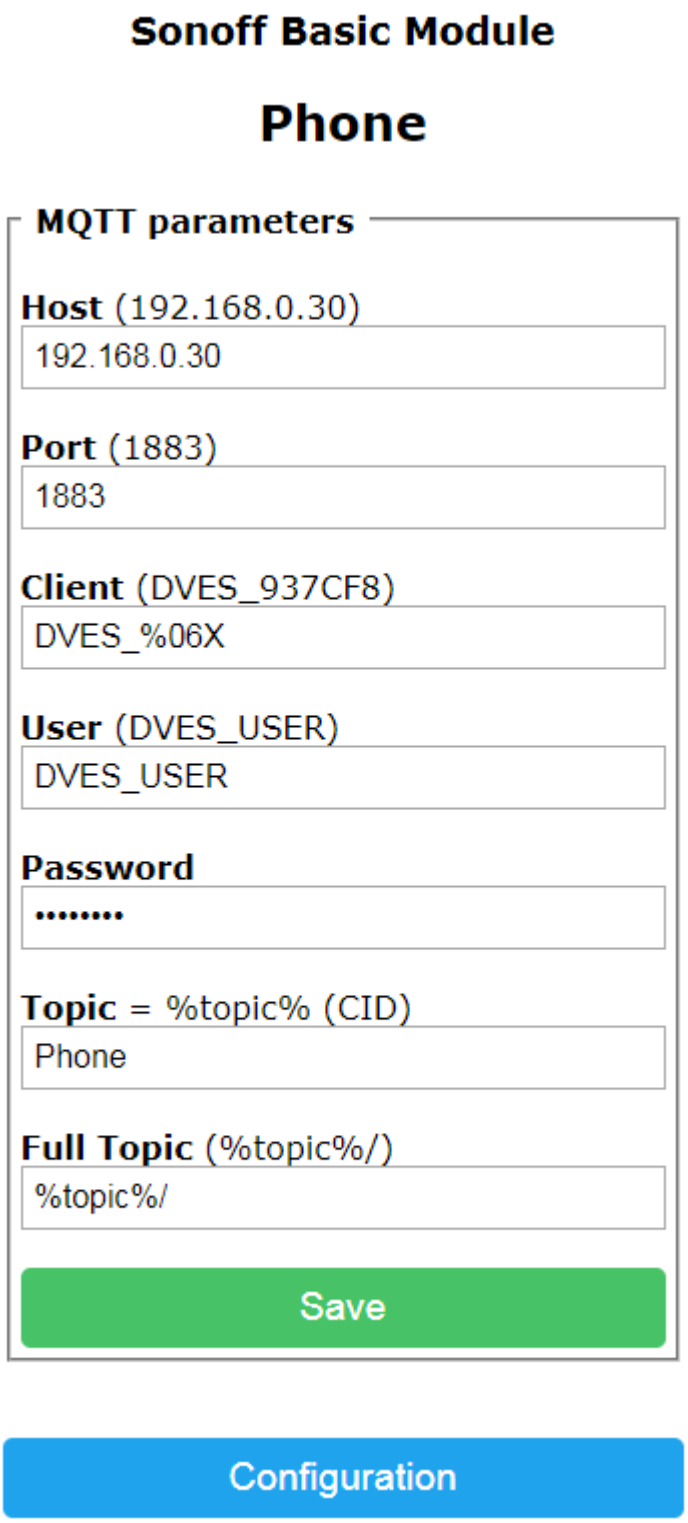


Figure 288 Sonoff CID MQTT Configuration

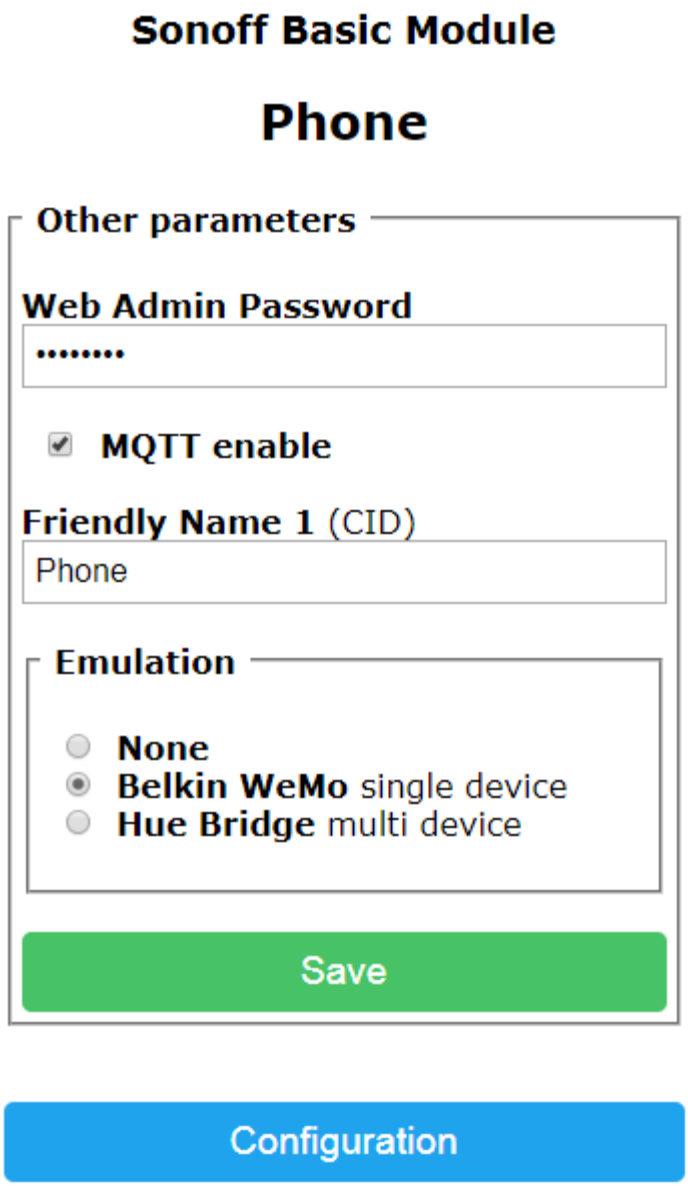


Figure 289 Sonoff CID Other Configuration

21.13 Reflash with Tasmota or Other Favorite Firmware

21.13.1 Tuya Version 1

The Tuya devices can be flashed with Tasmota using the procedure described on YouTube https://www.youtube.com/watch?v=O5GYh470m5k. I have extracted the shorthand of the step-by-step:

Parts List
-=-=-=-=-=
Raspberry Pi 3 - https://amzn.to/2SfpDQM
32gb Micro SD Card - https://amzn.to/2MwYVNY

Software and Github Links
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Etcher - https://www.balena.io/etcher/
Raspbian Stretch Lite - https://www.raspberrypi.org/downloads...
Putty (SSH) - https://www.chiark.greenend.org.uk/~s...
Tuya Convert Github - https://github.com/ct-Open-Source/tuy...

Procedure
-=-
Using Etcher, flash downloaded Raspbian Stretch Lite on SD
Using Window/Linux file system add file “ssh” to the boot partition of SD
Install SD on RPi
Use PuTTY to connect to RPi via SSH. Get the IP from router or other utility.
Login using pi raspberry for user/password
Run command “sudo raspi-config”, select advanced options then first one to expand onto entire SD
Reboot, reconnect via PuTTY
Run command “sudo apt-get update”
Run command “sudo apt-get dist-upgrade”
Run command “sudo apt-get install network-manager”
Run command “sudo apt install git”
Run command “git clone https://github.com/ct-Open-Source/tuya-convert”
Run command “cd tuya-convert”
Run command “./install_prereq.sh”
Run command “./start_flash.sh”
Follow the instructions on PuTTY window

1. Use a smartphone or other computer with WiFi to connect to the following network
   WIFI: vtrust-flash
   PASS: flashmeifyoucan
2. Insert Tuya device and hold button to put it in pairing mode (rapid LED flash)

3. Press <enter> to continue process. Be patient as dots appear on PuTTY window. The firmware download will scroll by and be followed with prompt for one of three actions
4. Run command "curl http://10.42.42.42/flash3" which is option 3

Go to smartphone of other Wifi device and connect to Sonoff network
Use browser to get to http://192.168.4.1
Enter SSD and password for normal WiFi network where this device will eventually be run. Be careful as recovery from a mistake here will need to reference https://github.com/arendst/Sonoff-Tasmota/wiki/Button-usage to get Tasmota to again setup access to 192.168.4.1. Even this may not be possible because the device button's GPIO pin may not be what default Tasmota is expecting. There are two SSID entries that are allowed. It is a good idea to setup a second one that is very simple so another router can be configured with this network if necessary.

The remainder of the operation is standard Tasmota configuration setup for the device.

Subsequent flashing of Tuya devices is done by repeating the steps above highlighted in red font. Each takes a few minutes and is actually faster than the flashing of Sonoff devices using SonOTA.exe.

It is possible in step 4 above to select Option4 to flash a custom image rather than the vanilla Sonoff one. If this is done make certain that the SSID is not set in the image or the SSID is set one to which you are able connect using WPA-compliant passwords (i.e. between 8 and 32 characters).

21.13.2 Tuya Version 2

Later versions of Tuya firmware are not able to be flashed with the original Tuya Convert. An update to Tuya Convert is available with video and links available at https://www.youtube.com/watch?v=dyUyewiKpRA and later at Tuya Convert 2.3 Update | Flash Tuya Smartlife Devices | No Soldering! | Remove the cloud | Custom Firmware (digiblur.com) . The process is similar as Version 1 as described in the prior section. Shorthand version is described below.

SD card for RPi is flashed with Raspian Buster Lite that can be obtained from https://www.youtube.com/redirect?event=video_description&v=dyUyewiKpRA&q=https%3A%2F%2Fwww.raspberrypi.org%2Fdownloads%2Fraspbian%2F&redir_token=HY-nuxnkaCPaxCurlGjY4WZT4lF8MTU3MzM0MDQxNkAxNTczMjU0MDE2

Once flashed the following SSH or Console commands are used:

```
sudo raspi-config
```

```
sudo apt install git
```

```
git clone
https://www.youtube.com/redirect?event=video_description&v=dyUyewiKpRA&q=https%3A%2F%2Fgithub.com%2Fct-Open-Source%2Ftuya-convert&redir_token=HY-nuxnkaCPaxCurlGjY4WZT4lF8MTU3MzM0MDQxNkAxNTczMjU0MDE2
```

```
cd tuya-convert
```

```
./install_prereq.sh
```

```
./start_flash.sh
```

Use a smartphone or other computer with WiFi to connect to the following network

WIFI: vtrust-flash
PASS: flashmeifyoucan

21.13.3 WS-1 Smart Plug

In the case of the Tuya WS-1 mini smart plug the module was configured as Generic and the specific IO pins mapped to the functions of the plug as shown in Figure 291. For this module the Console was used to assure the plug is restored to same output after a power cycle “poweronstate 3”.

Figure 290 Tuya WS-1 smart plug

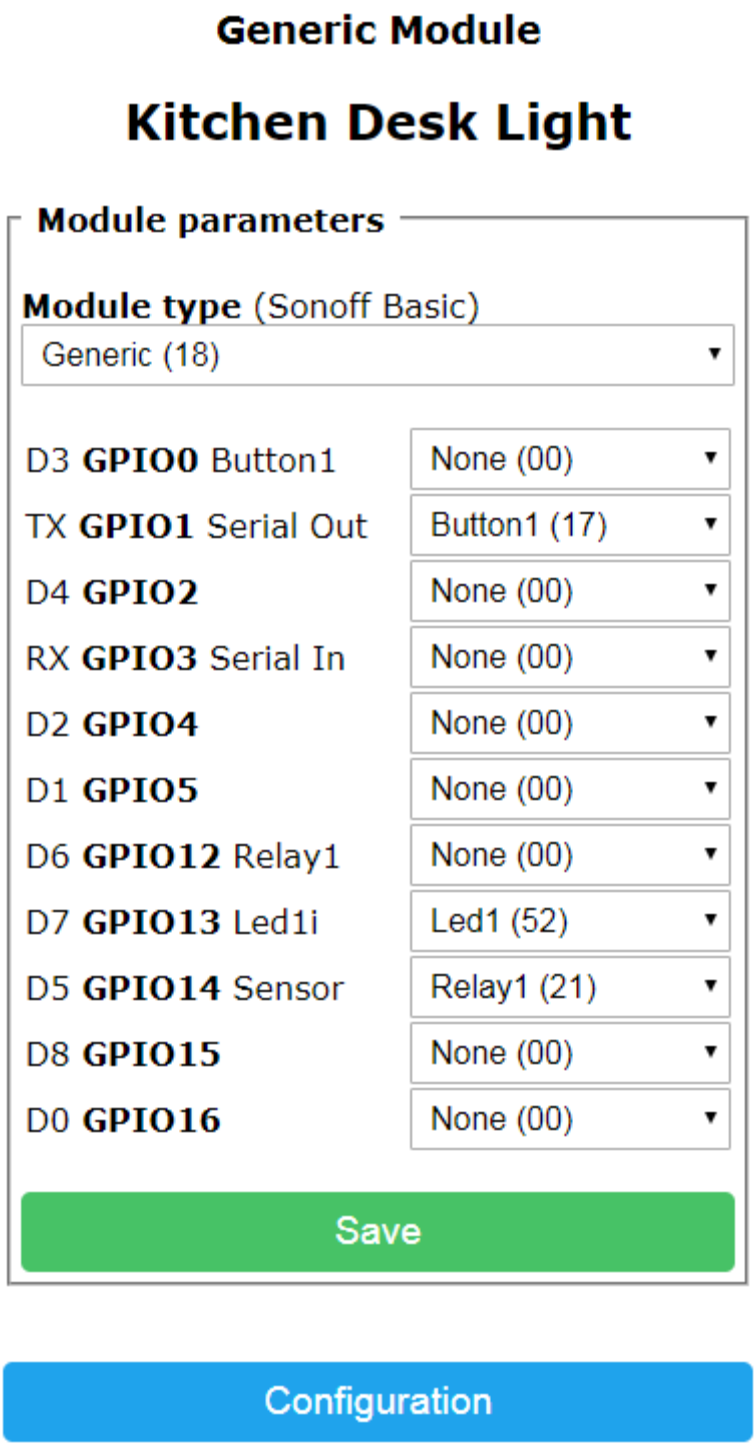


Figure 291 Tuya WS-1 Smart Plug Module Configuration

21.13.4 Luntak US101/US/102/US103/X6 WiFi Plug

Another round smart plug branded Luntak with model number US102/US102/US103/X6 is available from Amazon https://www.amazon.com/gp/product/B07CWQQY1Y/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1 sold for a package of 4 for $27 ($6.75/plug) on lightning deal. It is shown in Figure 292. These are 10A plugs with local and remote control. Three GPIO are used Relay-GPIO15, Button-GPIO13 and indicator LED (inverted)-GPIO2. The Tasmota configuration after flashing is shown in Figure 293.

Figure 292 Luntak WiFi Smart Plug

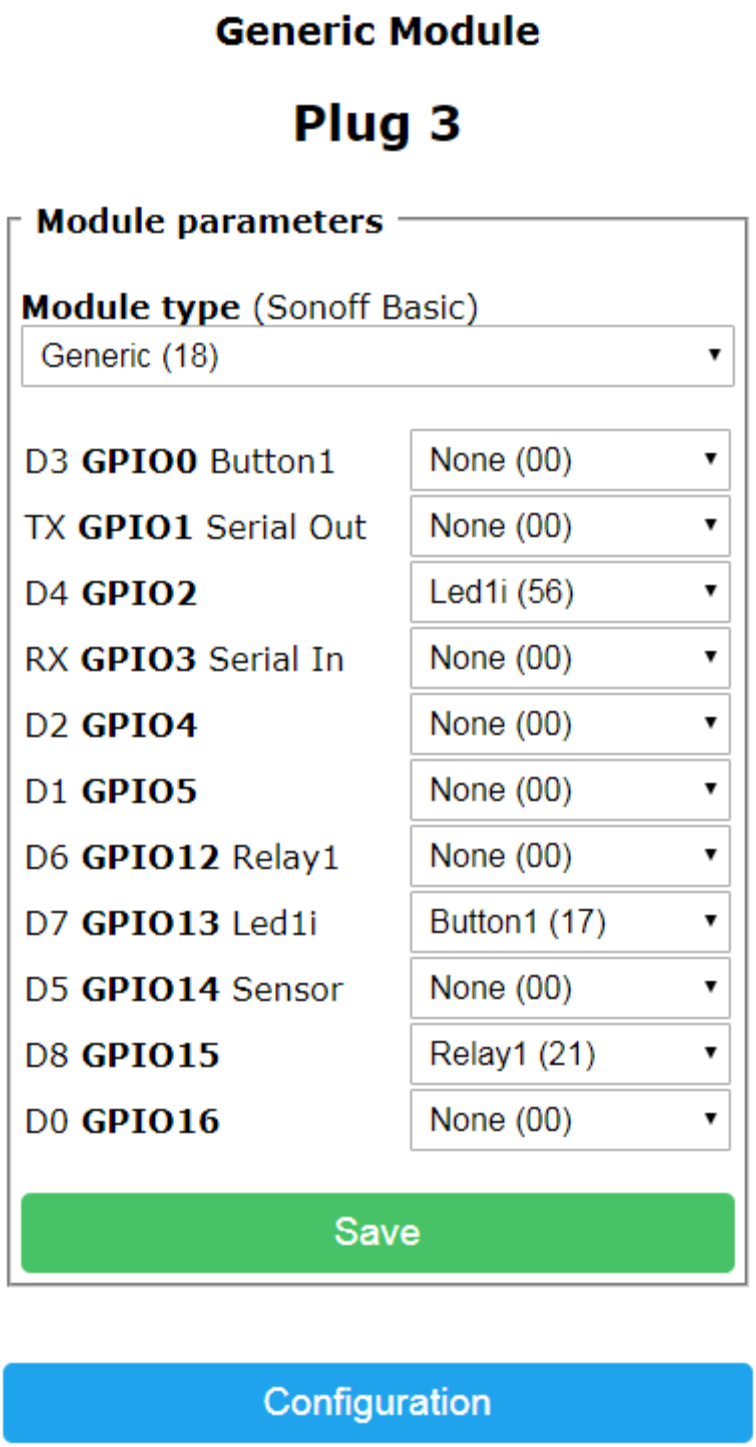


Figure 293 Luntak US101/US102/US103/X6 Configuration

21.13.5 EVA LOGIK Smartplug

Contributed by taylormia from Homeseer Message Board.
https://forums.homeseer.com/forum/lighting-primary-technology-plug-ins/lighting-primary-technology-discussion/mcsmqtt-michael-mcsharry/1299446-cheap-tuya-smart-plug.

The EVA LOGIK smart plug https://www.amazon.com/Socket-Outlet.../dp/B06XZ3J66L and Figure 294 can be flashed with Tasmota using tuya-convert https://github.com/ct-Open-Source/tuya-convert.

Figure 294 EVA LOGIK

The Tasmota template for this device is:

```
{"NAME":"EVA LOGIK Plug","GPIO":[255,17,255,255,255,255,255,255,255,52,21,255,255],"FLAG":0,"BASE":18}
```

The GPIO assignments based on the Generic (18) module are shown in Figure 295.

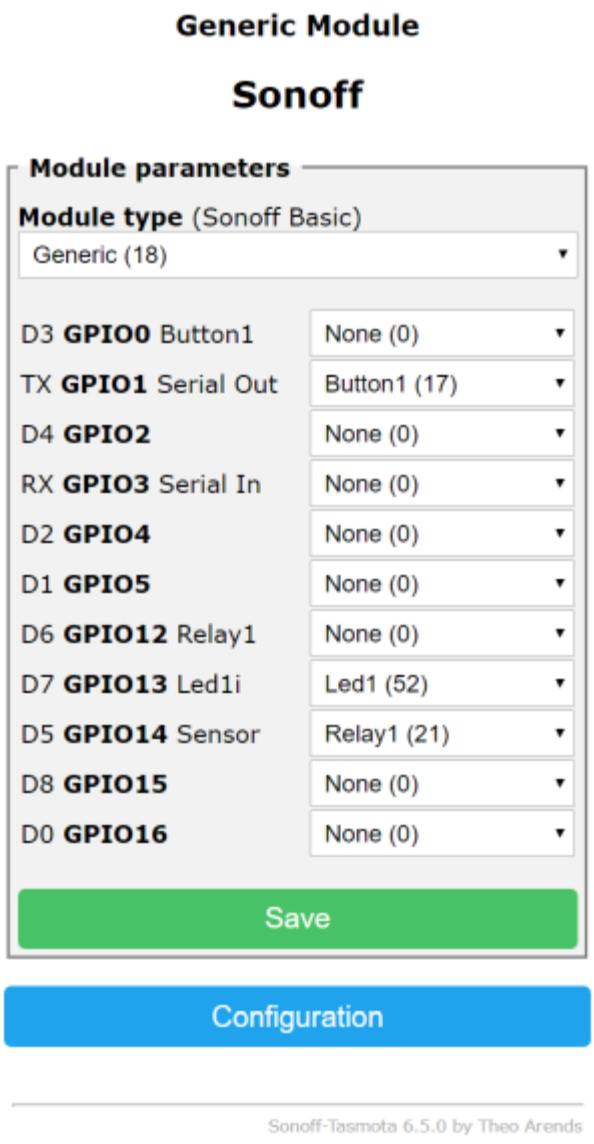


Figure 295 EVA LOGIK Smartplug GPIO Usage

21.13.6 WS212 WiFi Dual Plug with Energy Monitoring

The second device that was reflashed from Tuya to Tasmota was the WS212 dual plug with energy monitoring which is available from Amazon https://www.amazon.com/gp/product/B07G147QHX/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&th=1 for $16. It is similar in size to the Sonoff S31 and has the advantage of dual rather than single plug. Its downsides are 10A vs. 16A for the S31 and no easy way to open up the device.

Figure 296 WS212 dual plug with energy monitoring

This device did not have a standard configuration within Tasmota or Espurna. A new characterization for WS212 was added to mcsTasmota 6.4.1.9 which is shown below. It is similar to the Blitzwolf plug but uses GPIO12 for Relay #1 and GPIO 3 for the BL0937 SEL pin.

```
{ "WS212 Energy",   // WS212 dual plug witn single button with energy monitoring (ESP8286 - BL0937 or HJL-01 Energy Monitoring)
            // NX-SP201 dual plug with dual button and energy monitoring
            // https://www.amazon.com/gp/product/B07G147QHX/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&th=1
            // https://www.amazon.com/gp/product/B07F6X4KX3/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1
            // if LED1 & 2 user defined then the following color scheme can be achieved
            // unused - blue off, violet on
            // LED_INV - clear off, red on
            // LED - blue off, red on
  GPIO_USER, //GPIO_LED2_INV,   // GPIO00 Blue Led plug 2(1 = On, 0 = Off)
  0, //GPIO_USER,       // GPIO01 Serial RXD and Optional sensor
  GPIO_USER, //GPIO_LED1_INV,   // GPIO02 Blue Led plug 1(1 = On, 0 = Off)
  GPIO_NRG_SEL_INV,  //BL0937 or HJL-01 Sel output (0 = Voltage)  GPIO_USER,       // GPIO03 Serial TXD and Optional sensor
  GPIO_KEY1,       // GPIO4 Button 1 only on NX-SP201
  GPIO_HJL_CF,     // GPIO05 BL0937 or HJL-01 CF power
            // GPIO06 (SD_CLK   Flash)
            // GPIO07 (SD_DATA0 Flash QIO/DIO/DOUT)
            // GPIO08 (SD_DATA1 Flash QIO/DIO/DOUT)
  0,            // GPIO09 (SD_DATA2 Flash QIO or ESP8285)
  0,            // GPIO10 (SD_DATA3 Flash QIO or ESP8285)
            // GPIO11 (SD_CMD   Flash)
  GPIO_REL1,       // GPIO12 Relay (0 = Off, 1 = On)  GPIO_NRG_SEL_INV, // GPIO12 BL0937 or HJL-01 Sel output (0 = Voltage)
  GPIO_KEY2,       // GPIO13 Button 2 GPIO_KEY1,
```

```
  GPIO_NRG_CF1,    // GPIO14 BL0937 or HJL-01 CF1 current / voltage
  GPIO_REL2,       // GPIO15 Relay (0 = Off, 1 = On)
  0, 0
}
```

Each of the two plugs can be independently turned on and off. Energy monitoring is for the total utilized by both plugs. MQTT reporting for STATE and SENSOR are shown below:

Energy/STATE = {"Time":"1970-01-01T00:30:14","Uptime":"0T00:30:14","Vcc":3.456,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"POWER1":"OFF","POWER2":"ON","Wifi":{"AP":1,"SSId":"Anthem","BSSId":"E0:3F:49:9D:B9:68","Channel":8,"RSSI":66}}

Energy/SENSOR = {"Time":"1970-01-01T00:30:14","ENERGY":{"TotalStartTime":"2019-02-23T22:32:45","Total":0.002,"Yesterday":0.000,"Today":0.002,"Period":0,"Power":1,"ApparentPower":7,"ReactivePower":7,"Factor":0.18,"Voltage":128,"Current":0.056}}

Note that Tasmota Rules can be used for local control based upon energy use or can be used to report other MQTT events based upon energy or other parameters such as was illustrated in Section 21.11.

Table 6 WS212 Dual plug with energy monitoring Configuration

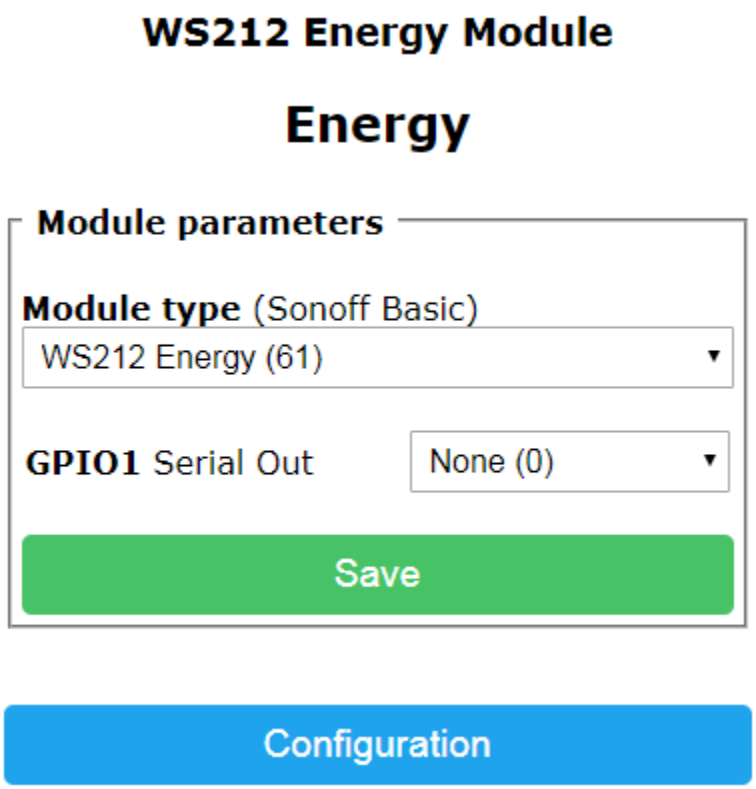


Figure 297 Tuya WS212 Configuration

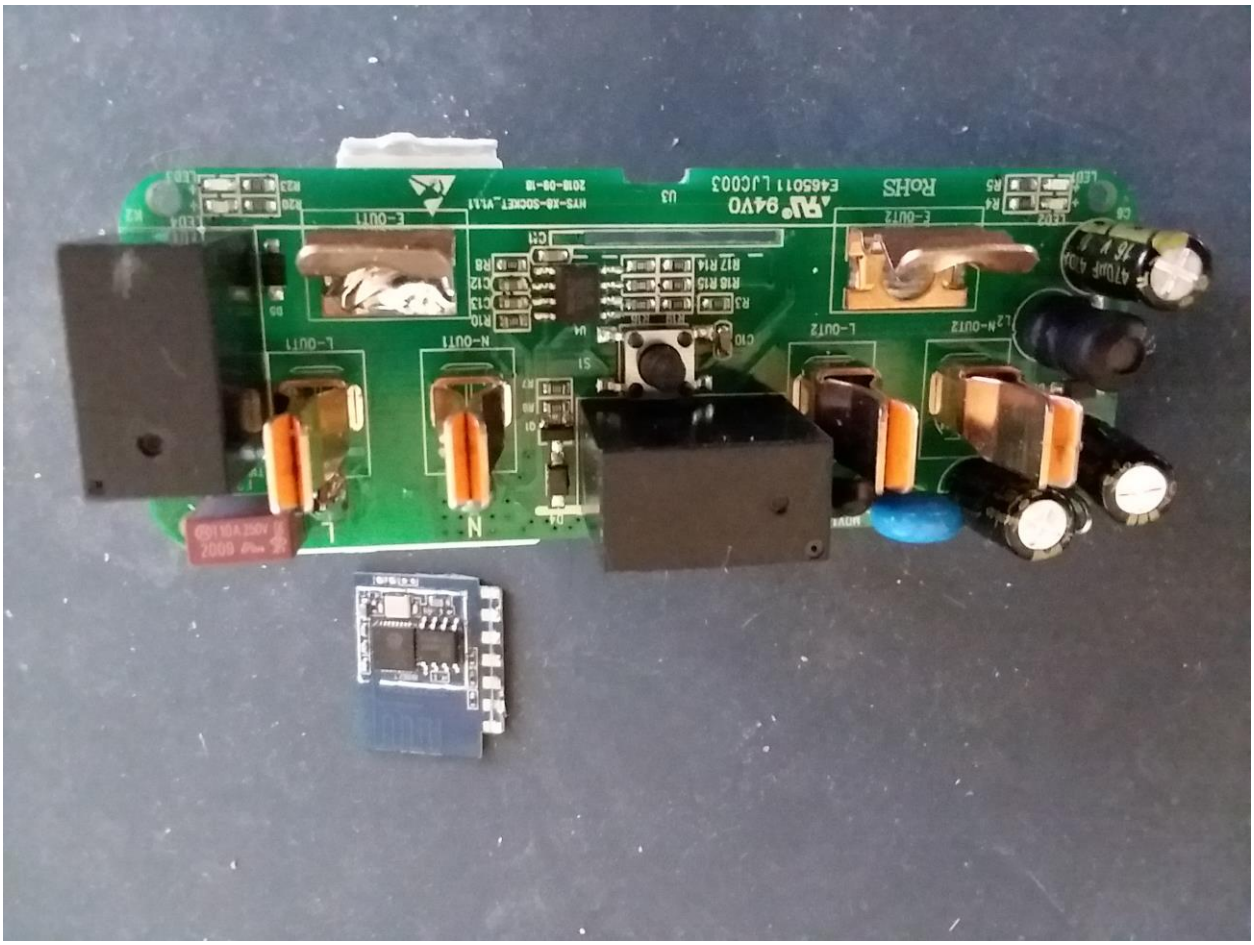


Figure 298 WS212 Circuit Cards

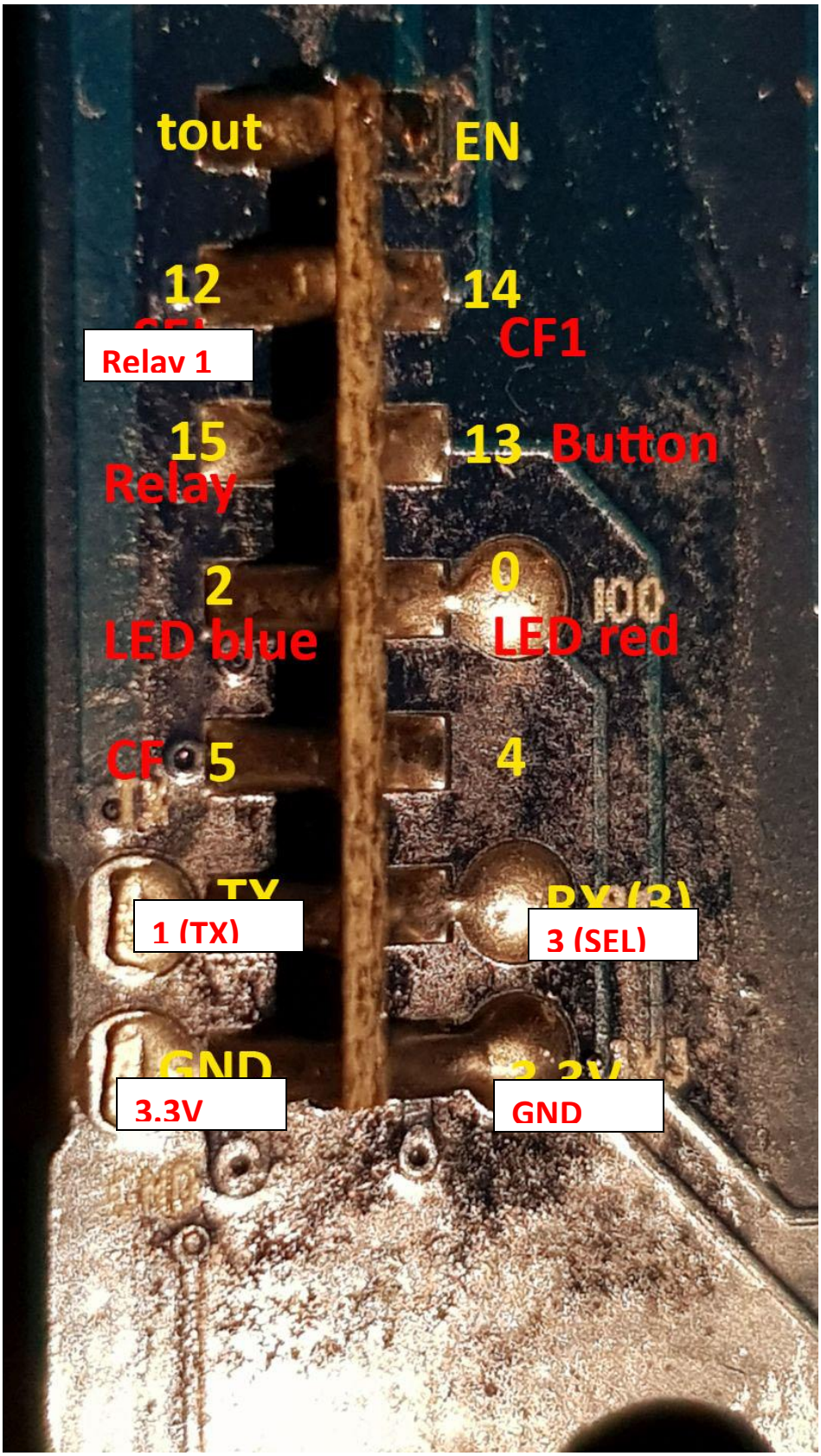


Figure 299 WS212 Circuit Interface Pinout

21.13.7 NX-SP201 Slitinto Dual Energy Monitoring Plug

A second energy monitoring plug was obtained from Amazon https://www.amazon.com/gp/product/B07F6X4KX3/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1. It is shown in Figure 300.

Figure 300 Slitinto NS-SP01 Dual Energy Plug

It is the same height as the WS212 so will only occupy one plug in a standard wall outlet. It is wider with a total span of about 4.5 inches. Electrically it is superior with a rated load of 15A. It has two buttons to locally control each socket individually. The illumination on each button is from two LEDs. One is red and it is slaved to the socket's relay position. The other is blue which was setup in the WS212 Tasmota configuration as user configurable, however only LED options are valid per the following

GPIO0 Led2 gives blue indication when plug 2 is off and red when it is on
GPIO0 Led2_Inv gives red when on and clear when plug is off
GPIO0 None gives blue when off and violet when on
GPIO2 has the same options for plug position 1.

The energy monitoring connections and relay connections are the same internally as the WS212. The configuration I used for the Den Heater is shown in Figure 301 using mcsTasmota641A.

This plug was configured with console “poweronstate 3” to persist on/off state after power cycles. No rules were added. Primary use is tracking energy use of the Den Heater.

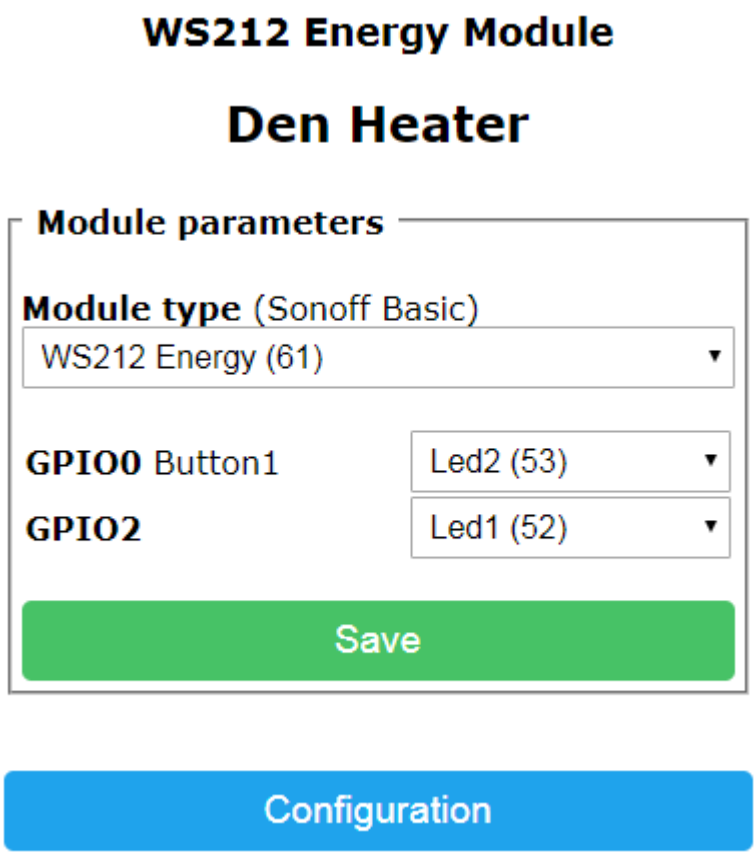


Figure 301 Slitinto NX-SP201 Dual Energy Plug Configuration

21.13.8 BN-LINK BNC-60/U133TJ Energy Monitor Plug (BL0937)

The BN-LINK is a value-priced plug that supports 15 Amps and contains energy monitoring using the BL0937 chip. Available at https://www.amazon.com/gp/product/B07CX5KLXN/ref=ppx_od_dt_b_asin_title_s00?ie=UTF8&psc=1 as well as other mass outlets such as https://www.walmart.com/ip/BN-LINK-2-Pack-Smart-Wi-Fi-Plug-Outlet-with-Energy-Monitoring-and-Timer-Function-No-Hub-Required-Works-with-app/933347753. The Amazon offering lightning deal was $7.25/plug in pack of 4.

It's dimensions allow two plugs to be installed in a standard duplex outlet with an overhang on the right side of the outlet. It is pictured in Figure 302.

Figure 302 BN-LINK Smart Plug with Energy Monitoring

The device has a single button, single relay and a pair of status LEDs that show through the button. Red and Blue colors are used for the two LEDs. Construction appears solid when the top is popped off as shown in Figure 303. The relay is GOLDEN GJ-1A-5L rated at 15A/125V, 12A/277V. Based upon size of the digital board it appears to ESP8285.

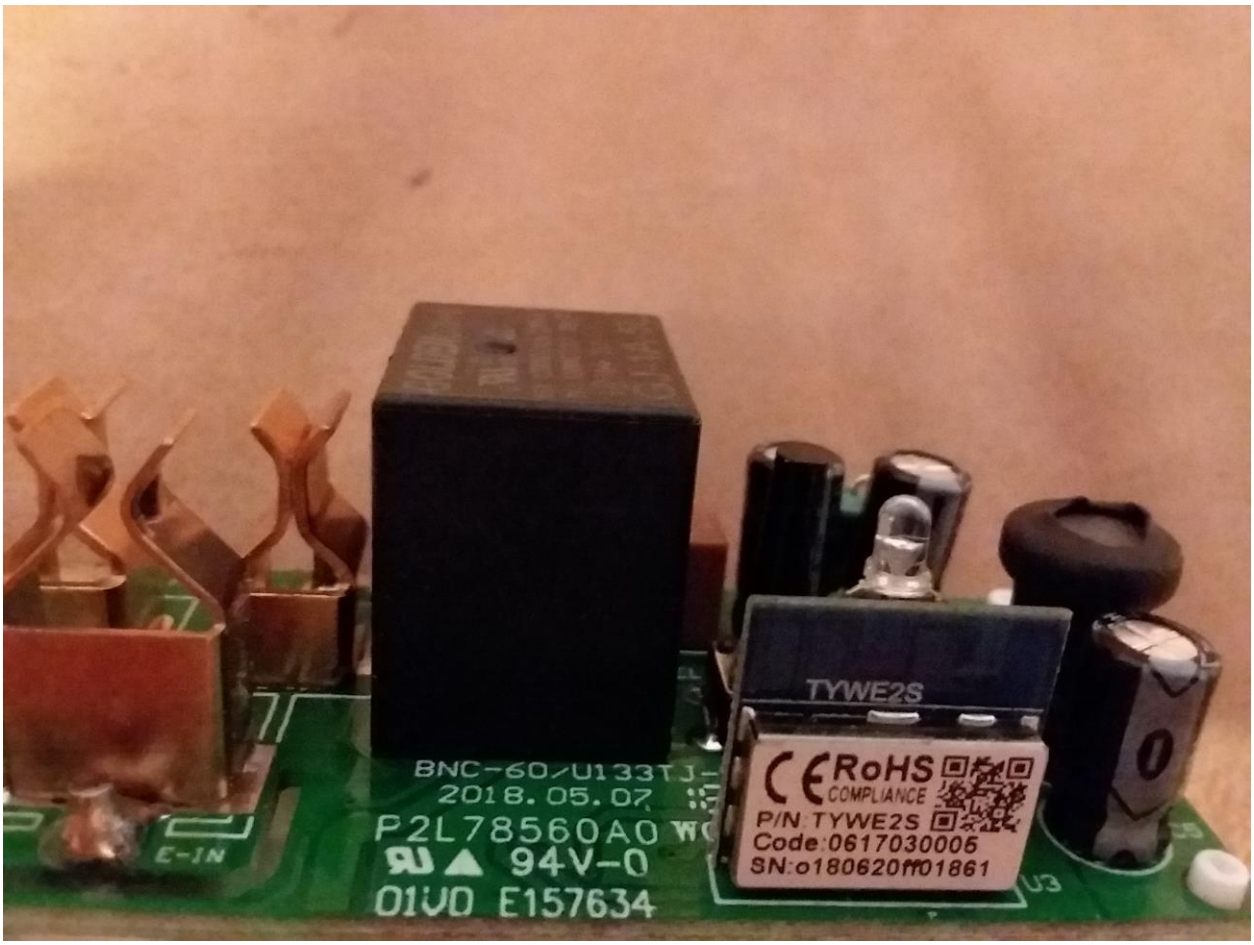


Figure 303 BN-LINK Component Side Circuit Board

The BL0897 was mounted on the bottom of the main circuit board. It was not visible until a cutout was made through the bottom of the case. This is shown in Figure 304 . Epoxy was used to replace the bottom cutout. Since the ESP was covered with a metal shield it was not possible to trace pins. The back of the digital card did have labeled pads of the following GPIO in order from left to right: 5, 0, 4, 13, 2, 3, 12, 14, RST. Tracing from the BL0937 pins (Figure 305 U4) to the pins on the digital card (left bottom in figure) allowed the power monitoring configuration to be determined.

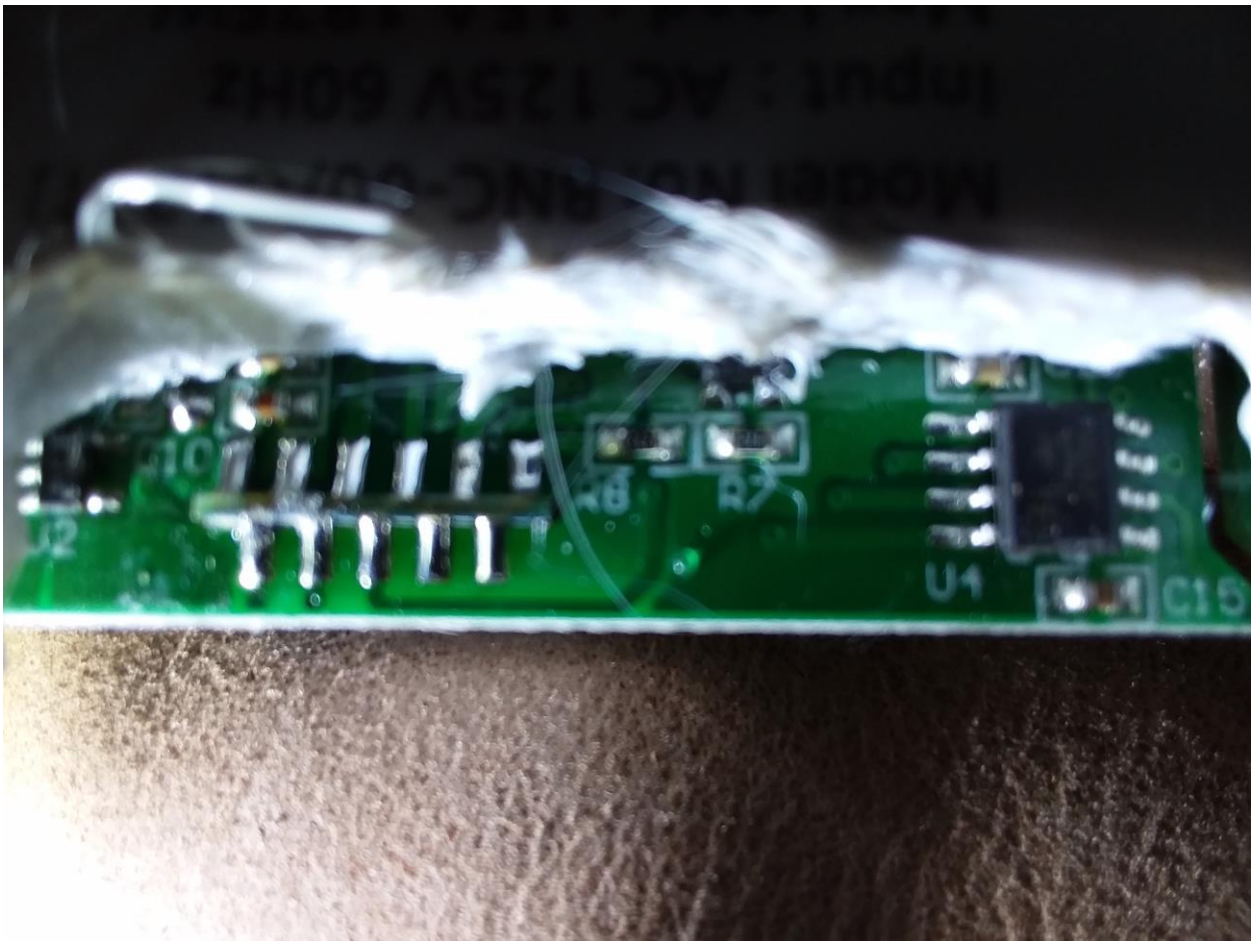


Figure 304 BN-LINK Main Circuit Bottom

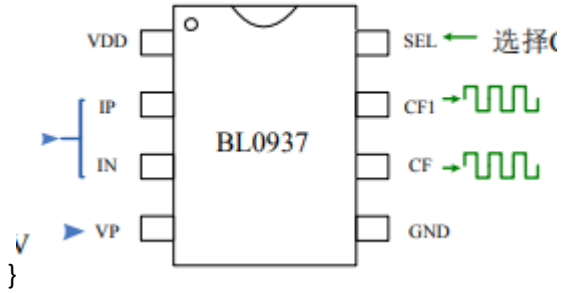


Figure 305 BL0937 Pinout

The LED, Relay and Button pins were determined by experimentation from the Generic Tasmota template. The configuration defined is shown below:

```
{ "BN-LINK Energy",   // BN-LINK Energy Monitoring model BNC-60/U133TJ (BL0937)
 /https://www.amazon.com/gp/product/B07CX5KLXN/ref=ppx_od_dt_b_asin_title_s00?ie=UTF8&psc=1
    0,                          // GPIO00
    GPIO_LED2_INV,              // GPIO01 Red Led (1 = On, 0 = Off)
    0,                          // GPIO02
    GPIO_KEY1,                  // GPIO03 Button
    GPIO_HJL_CF,                // GPIO4 BL0937 power
    GPIO_NRG_CF1,               // GPIO05  BL0937 current / voltage
                                // GPIO06 (SD_CLK   Flash)
                                // GPIO07 (SD_DATA0 Flash QIO/DIO/DOUT)
                                // GPIO08 (SD_DATA1 Flash QIO/DIO/DOUT)
    0,                          // GPIO09 (SD_DATA2 Flash QIO or ESP8285)
    0,                          // GPIO10 (SD_DATA3 Flash QIO or ESP8285)
                                // GPIO11 (SD_CMD   Flash)
    GPIO_NRG_SEL_INV,           // GPIO12 BL0937 Sel output (0 = Voltage)
    GPIO_LED1_INV,              // GPIO13 Blue Led (1 = On, 0 = Off)
    GPIO_REL1,                  // GPIO14  Relay
    0,                          // GPIO15
    0, 0
```

The binary that includes the BN-LINK option is at http://mcsSprinklers.com/mcsTasmota.zip with file mcsTasmota641B.bin. The corresponding source is at http://mcsSprinklers.com/mcsTasmota_Source_641B.zip

The Tasmota configuration is shown in Figure 306. Reporting on the main page is shown in Figure 307.

With the other energy plugs that have been converted to Tasmota the voltage appeared reasonable out of the box so no calibration was done. With this one the voltage was showing 143 vs. the digital multi-meter of 120 so calibration is needed in this case. A Kill-A-Watt or other known standard can be used to assist in the calibration. The Tasmota Console commands are below as needed. Full Tasmota options can be found at https://github.com/arendst/Sonoff-Tasmota/wiki/Commands.

CurrentSet	`<value> = calibrate current to target value in mA`
VoltageSet	`<value> = calibrate voltage to a target value in V`
PowerSet	`<value> = calibrate power to a target value in W`
FrequencySet	`<value> = calibrate frequency to a target value in Hz`

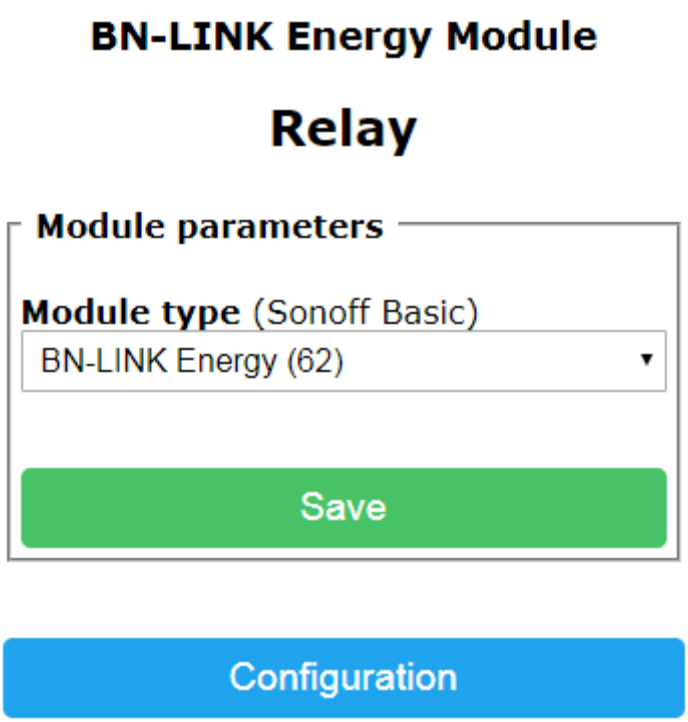


Figure 306 BN-LINK Configuration

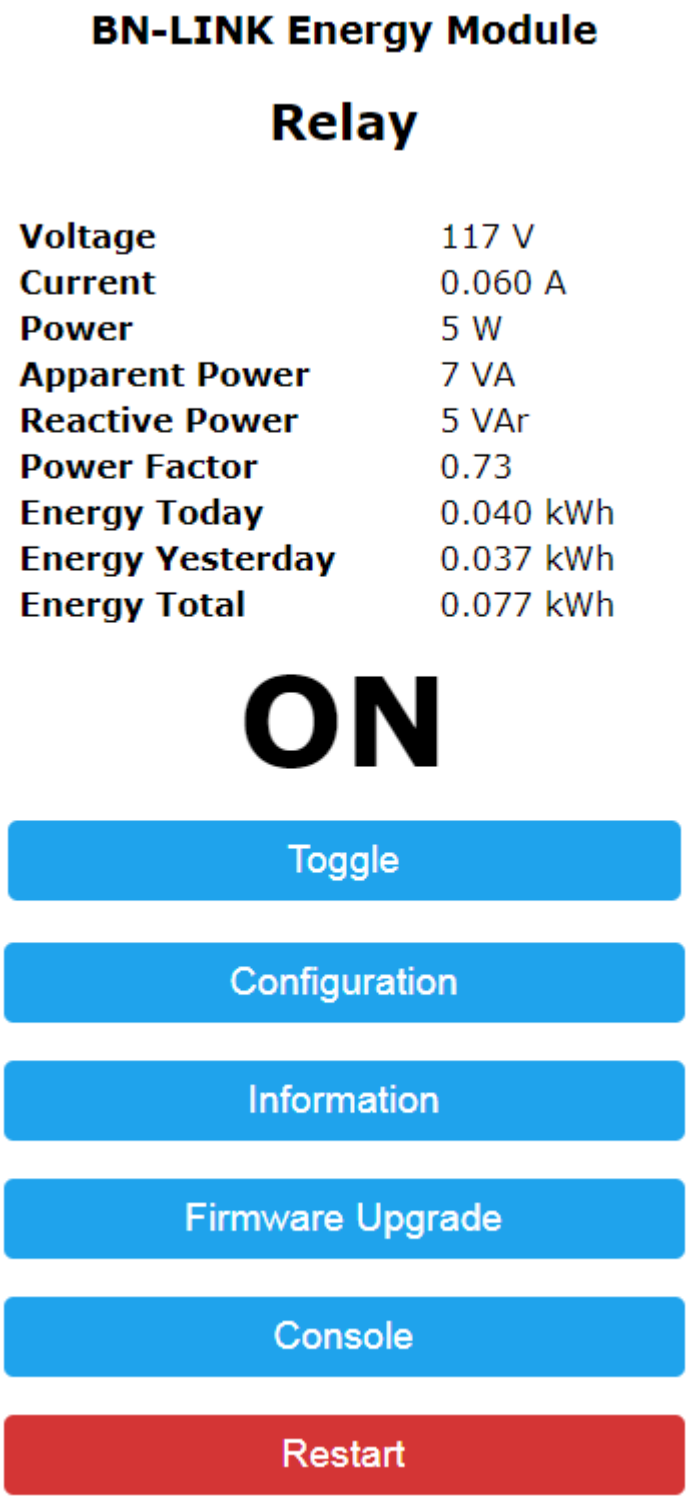


Figure 307 BN-LINK Status Display

21.13.9 Wheswell USB Power / Mains Power Wifi Power Strip with Surge Protection

The Wheswell ZLD-44USA-W Wifi Power Strip is available from Amazon for $27 https://www.amazon.com/gp/product/B0796R56J8/ref=ppx_yo_dt_b_asin_title_o02_s01?ie=UTF8&psc=1 and lightly cheaper if obtained on a deal. It is a well built unit and decent power handling capacity of 1875 watts total; 10 amps per plug. USB power is 6 amps total or 2.4 amps/plug. The unit has LED feedback for each of four power plugs and the bank of USB plugs. Each can be individually controlled via WiFi. There also exists a button with LED feedback. It is shown in Figure 308.

Figure 308 Wheswell Model ZLD-44USA-W WiFi Power Strip

The Tasmota configuration that I setup is shown in Figure 311. Relay 1 (which is only a LED) and Button 1 are used to act as master control for the strip. Rules are used to slave the other five relays to their previous On state or all Off state depending upon Relay 1 state. Relay 6 is for the USB and Relay 2, 3, 4 and 5 for the four mains plugs.

The Alexa HUE emulation will only support four Relays so this setup has overall control and control of three of the mains plugs via Alexa. The other mains plug and the USB can be controlled via HTTP or MQTT, but not Alexa directly.

As I was identifying the GPIO configuration I had setup something that prevented the unit from booting. This gave me reason to open it up so it could be reflashed via serial connection. The 3.3v, Gnd, Rx and Tx pads were available at the edge of the circuit card and the GPIO was a pad in the middle of the card. All were labeled so not much effort to do the serial programming. The top of the digital card has the etched WiFi antenna. This can be seen in Figure 309. Figure 310 shows the bottom with the full back of the mains circuit card and the solder connects/breakout for the serial programming.

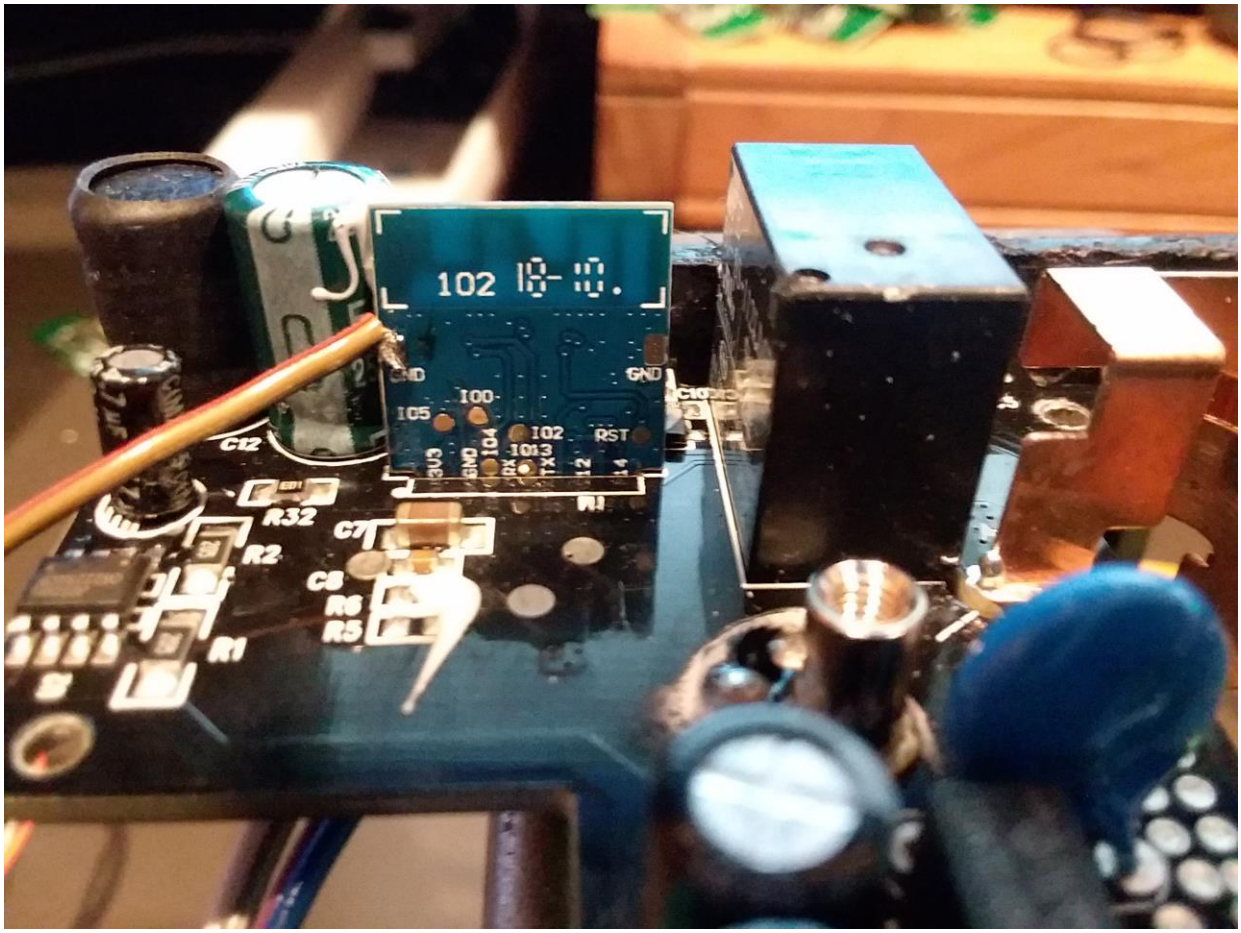


Figure 309 Wheswell Power Strip Digital Card (back)

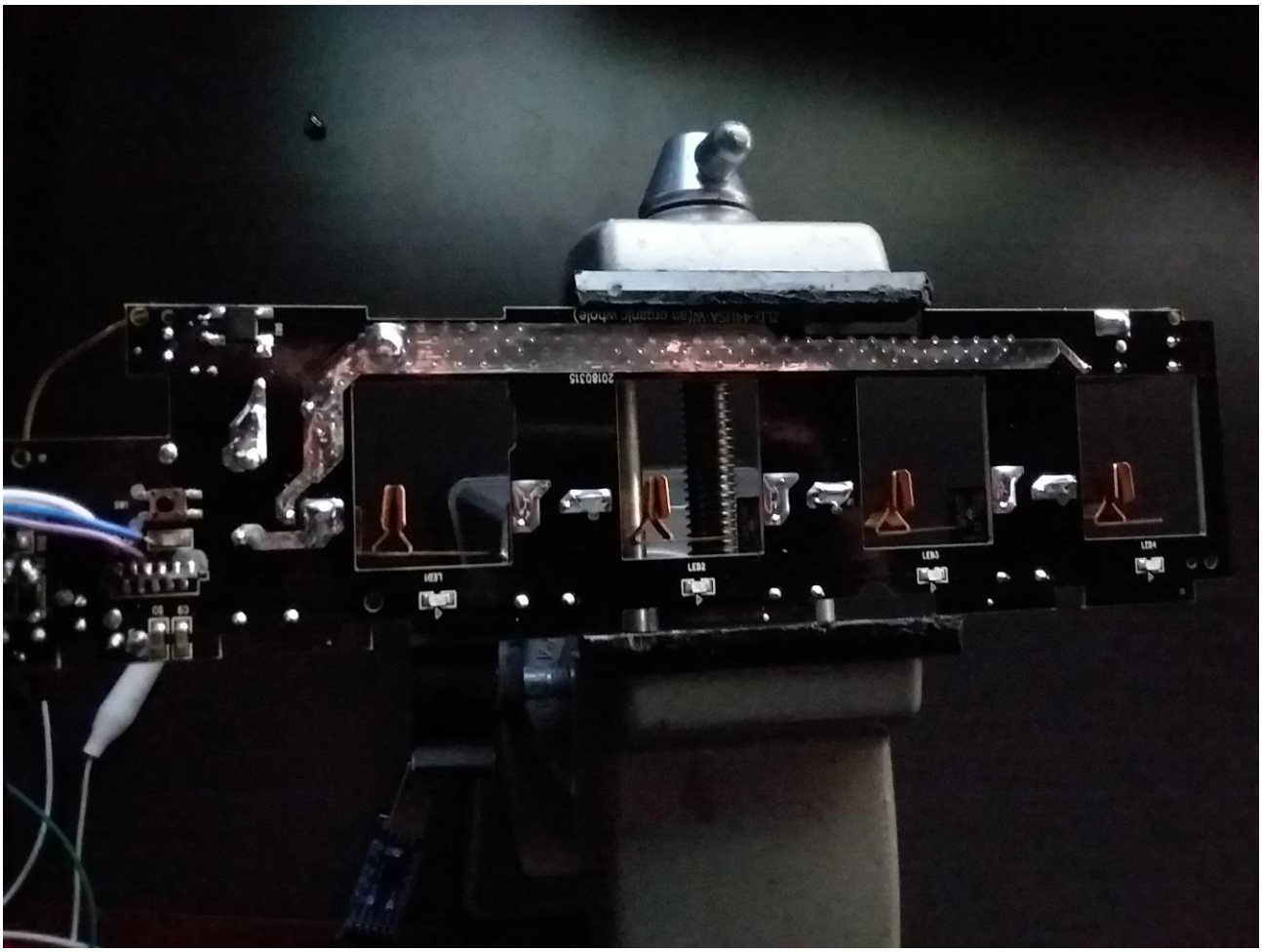


Figure 310 Wheswell Mains circuit card with digital serial breakout

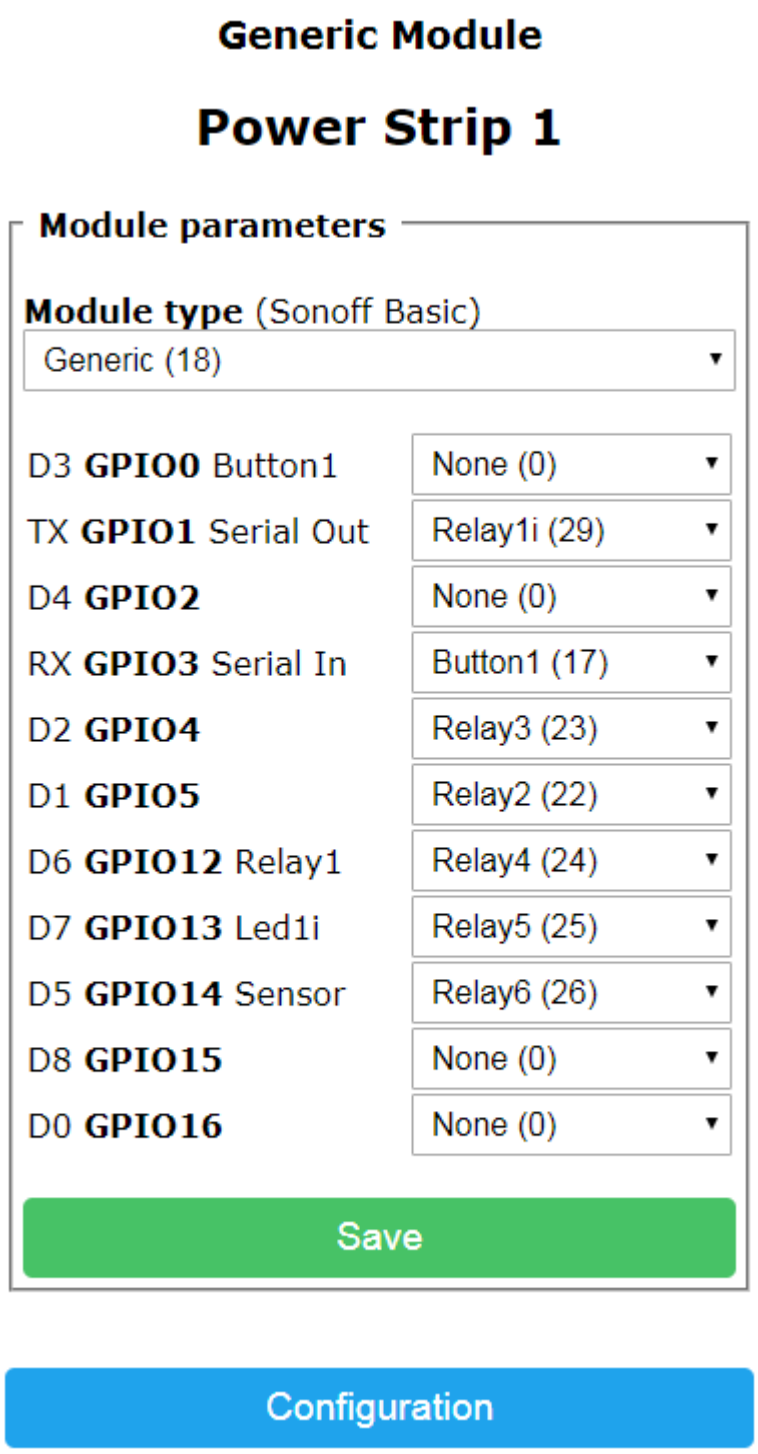


Figure 311 Tuya Wheswell Power Strip Configuration

21.13.1 JINVOO Water Shutoff Valve

The water shutoff valve under the JINVOO name is available at Amazon https://www.amazon.com/gp/product/B07F36MVVT/ref=pe_2640190_232586610_pd_te_s_mr_im?_encoding=UTF8&pd_rd_i=B07F36MVVT&pd_rd_r=0Z5SS3KR5QJZSZ5RNKPH&pd_rd_w=LR4Kg&pd_rd_wg=wYgrA I found it as a lightning deal for $28. Even at regular price it is a far more cost effective water shutoff than the HS-WV100+, FortrerzZ, WaterCop which requires pluming changes and even similar products that do not require plumbing modifications such as Guardian, or Dome. A good comparison is at https://www.diycontrols.com/t-automatic_water_shut_off_valves.aspx.

The unit has easy access with four screws, but the ESP8266 access is very hard and no exposed pins were obvious for Rx and Tx and GPIO0. Fortunately, TuyaConvert version 2 does work for this device. Its GPIO configuration is shown in Figure 313. This image is from a HomeAssistant thread https://community.home-assistant.io/t/has-anyone-implemented-the-jinvoo-wifi-smart-valve-sm-aw713/101542 but it was found to not be the same as a similar branded unit I obtained. The configuration for my unit is shown in Figure 312.

This device does have a bistable relay. When power is lost the valve remains in the position that it was when power was lost. Tasmota should be configured with powreonstate of 3 so that when power is restored the valve will not be commanded to another position.

Initial attempt for the GPIO configuration used the information from a Home Assistant site shown in Figure 313. This failed so via experimentation the following was determined. This resulted in a configuration shown in Figure 312.

GPIO4 Blue LED
GPIO5 Red LED
GPIO12 Relay
GPIO13 Button

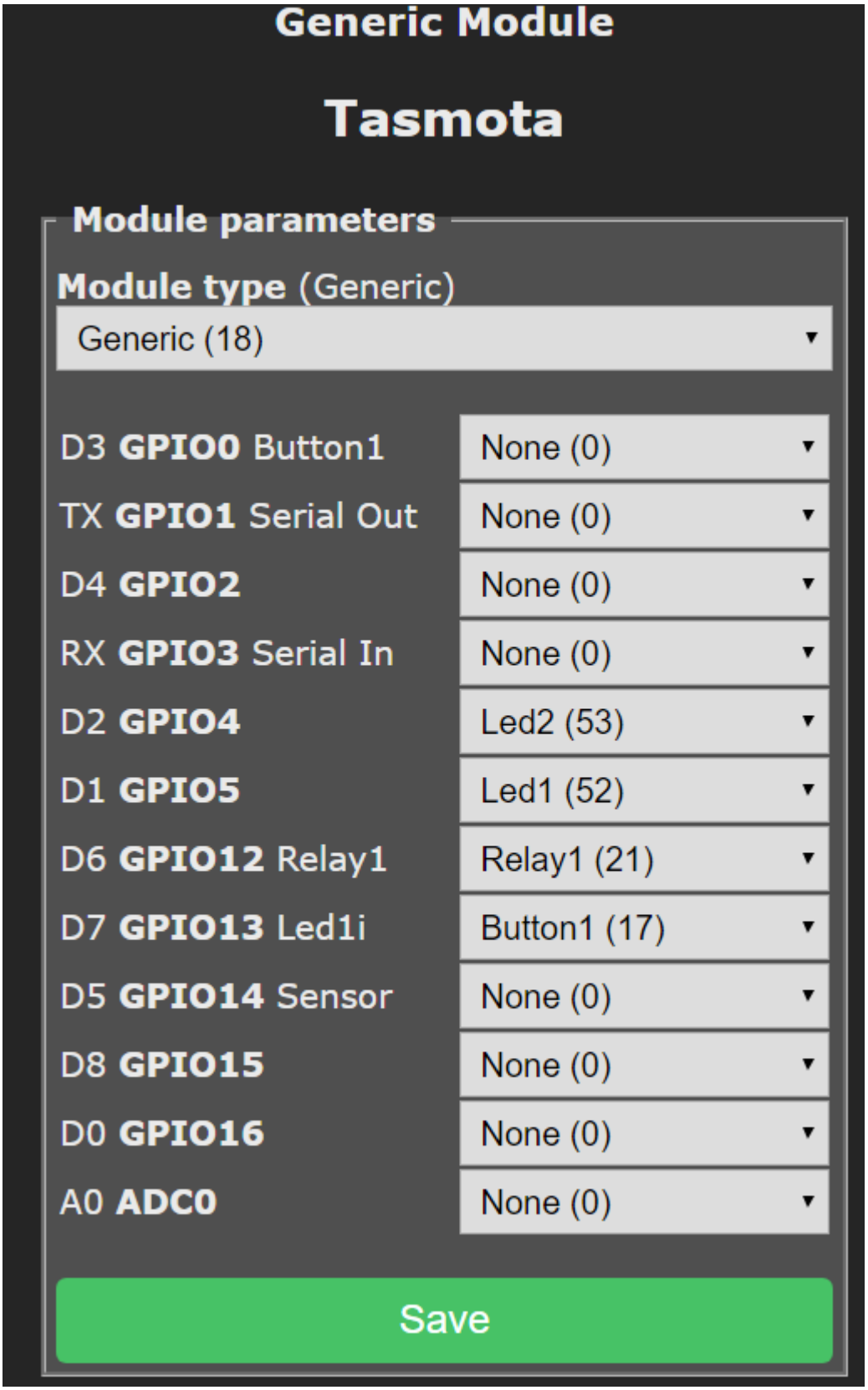


Figure 312 JINVOO Water Valve GPIO Configuration

Based on	Generic (18)
GPIO0	None (0)
GPIO1	None (0)
GPIO2	None (0)
GPIO3	None (0)
GPIO4	Relay1 (21)
GPIO5	Led1 (52)
GPIO9	None (0)
GPIO10	None (0)
GPIO12	Button1 (17)
GPIO13	Led2 (53)
GPIO14	None (0)
GPIO15	None (0)
GPIO16	None (0)

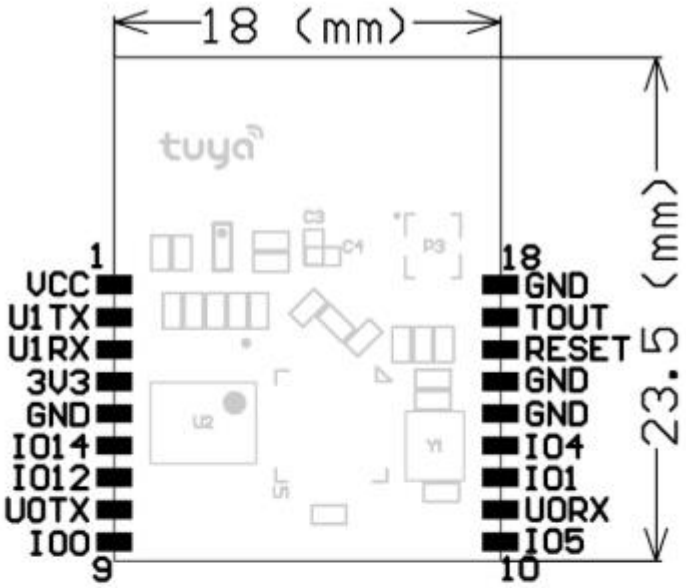
Options

☑ **ADC0 input**

Template info: {"NAME":"Jinvoo Valve","GPIO":[0,0,0,0,21,52,0,0,17,53,0,0,0],"FLAG":1,"BASE":18}

Figure 313 Other Water Shutoff Valve GPIO Setup

TWE1S Module is used by many devices that host Tuya firmware. Many times they are accessible when the package is opened. This will allow access to flash the device using serial connection. This and other modules are described in https://docs.tuya.com/docDetail?code=K8uhkbx8bzbiw.



2.1 Pin Definition

PIN NO.	NAME	TYPE	DESCRIPTION
1	VCC	S	UART1 power (3.3V)
2	U1TX	I/O	UART1_TXD
3	U1RX	I/O	UART1_RXD
4	3V3	S	Supply voltage (3.3V)
5	GND	S	Ground
6	IO14	I/O	GPIO_14
7	IO12	I/O	GPIO_12
8	U0TX	I/O	UART0_TXD(used to print module's internal information)
9	IO0	I/O	GPIO_0(processing during initials, caution when used)
10	IO5	I/O	GPIO_5
11	U0RX	I/O	UART0_RXD(used to print module's internal information)
12	IO1	I/O	GPIO_1(status is uncertain during initials)
13	IO4	I/O	GPIO_4
14	GND	S	Ground
15	GND	S	Ground
16	RESET	I/O	External reset singal(negative level effects)
17	TOUT	AI	ADC terminal
18	GND	S	Ground

Figure 314 TWE1S Module Pinout

21.13.2 Switchbot Mini Plug

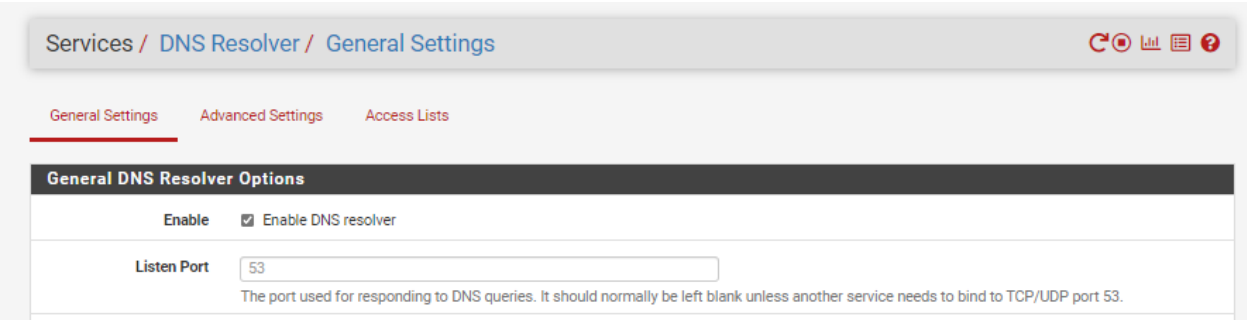
Switchbot mini plugs contain power monitoring and utilize an ESP32-C3.

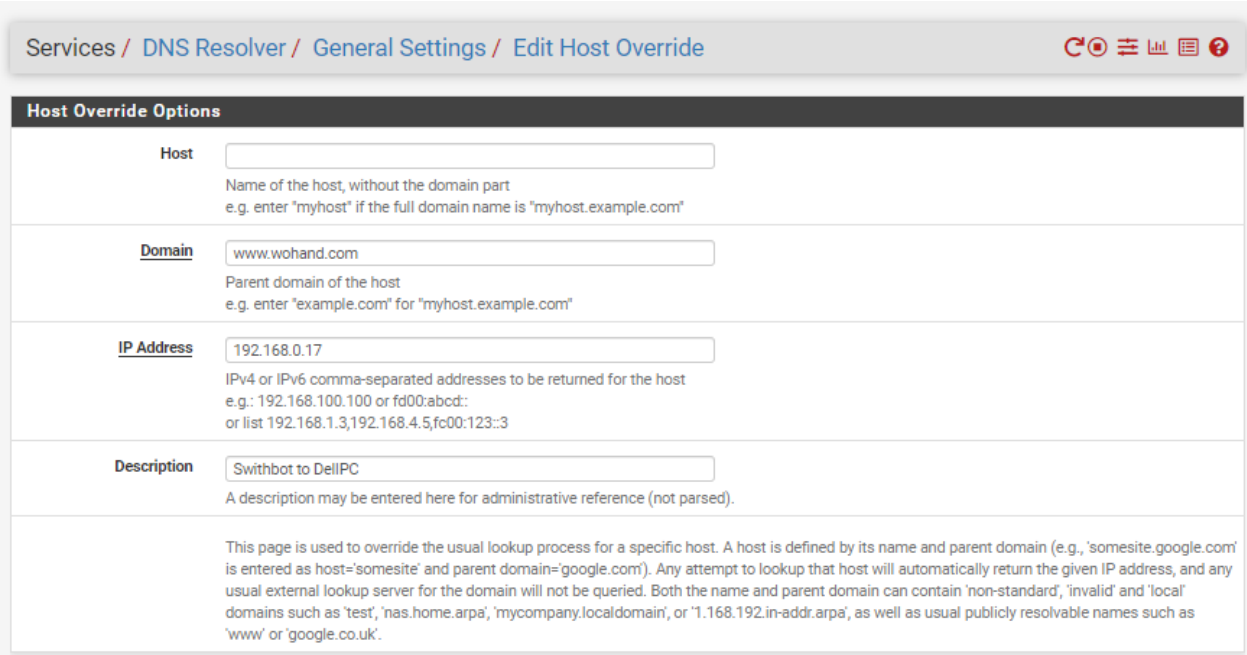
Amazon sells them at https://www.amazon.com/dp/B09YV2L3MN?psc=1&ref=ppx_yo2ov_dt_b_product_details at the time I purchased a box of 4 they well selling for $25.50 or a little over $6 per plug. A good value for both 15A (resistive) and power monitoring.

They can be opened by breaking plastic seal on the bottom and flashed via GPIO pins, but an OTA method was developed per GitHub - kendallgoto/switchbota: Replaces the factory firmware on the SwitchBot Plug Mini via OTA, enabling the use of Tasmota without disassembling the unit. A good video showing the firmware install process is at SwitchBot ESP32-C3 Plug/Bulb to TASMOTA ESPHome | NO Solder Upgrade - YouTube .

The methodology employed is to setup a local host DNS override on the domain www.wohand.com. This is the domain the switchbot plug uses to communicate with the switchbot App. A little server application is then run at the new local target of the www.wohand.com domain. When a firmware update request is made this server will field the request and install Tasmota rather than what may be available on the wohand server.

The first step is to setup the local host overrided. It is typically on the router. The setup of some popular routers is contained in Confusion on how to OTA · Issue #3 · kendallgoto/switchbota · GitHub . I used pfSense and setup DNS Resolver service for the host override with host field blank and domain field www.wohand.com. The local IP was for my laptop.





I installed the nodejs server application on this laptop; opened a command window; entered “npm i” to get all the reference libraries; entered “node index.js”. This starts the server listening on port 80. I did not have anything else running on port 80 on this laptop at this time. I installed four plugs and did not need to restart the server application during the process.

The second step is to get the switchbot plug onto the WiFi network. The switchbot App from Google Play Store is used for this. The plug communicates via both WiFi and Bluetooth. Since wohand.com cloud is no longer available the plug needs to be manually added. Select the mini plug from the icon menu at the bottom. It will be with Bluetooth connection to the ESP32. The on/off switch is held for a couple seconds to start its LED flashing. The App will then prompt for the SSID and WiFi password. It will also give you chance to give the plug a name. It will be later useful to know the BLE MAC of the plug so navigate the App for Device Info and notate the BLE MAC. This is the end of the use of App for this plug.

Bluetooth is used to make the request for a firmware update. Google play store has “nRF Connect” to accomplish this. When the nRF Connect is started it will scan the Bluetooth devices and list them. Click on the MAC for the plug. The Client tab will be used to interact with the plug. Select Unknown Service / Unknown Characteristic. Where it shows WRITE, WRITE NO RESPONSE there will be an uparrow that allows uploading data. Click the uparrow, select BYTE ARRAY and enter “570F0A010C”. This will result in the console window where the local server is running to echo that it is installing a WoPlugUS_V12.bin file. Give it about 40 seconds and then it will be ready to install Tasmota. Start this in a similar manner with nRF Connect with the BYTE ARRAY “570F0B”. The server’s console will show it is installing payload.bin. The image below shows the server handling each of the four plugs that were done.

```
c:\>cd switchbota-main

c:\switchbota-main>cd server

c:\switchbota-main\server>npm i
npm WARN switchbota-server@1.0.0 No repository field.

audited 51 packages in 3.692s
found 0 vulnerabilities


c:\switchbota-main\server>node index.js
Server listening on port 80
^C
c:\switchbota-main\server>node index.js
Server listening on port 80
::ffff:192.168.0.224 - /version/wocaotech/firmware/WoPlugUS/WoPlugUS_V12.bin
::ffff:192.168.0.224 - /payload.bin
::ffff:192.168.0.49 - /version/wocaotech/release.json
::ffff:192.168.0.225 - /version/wocaotech/firmware/WoPlugUS/WoPlugUS_V12.bin
::ffff:192.168.0.225 - /payload.bin
::ffff:192.168.0.226 - /version/wocaotech/firmware/WoPlugUS/WoPlugUS_V12.bin
::ffff:192.168.0.226 - /payload.bin
::ffff:192.168.0.227 - /version/wocaotech/firmware/WoPlugUS/WoPlugUS_V12.bin
::ffff:192.168.0.227 - /payload.bin
```

After a few minutes Tasmota will be running on the plug. It will open a WiFi SSID with name starting with Tasmota… Connect to this SSID and navigate to 192.168.4.1 with a browser. I used same laptop running the server for this. Tasmota will allow you to enter your network SSID and password and then restart again.

The plugin needs to be configured for the GPIO pins being used. This is most easily done using the template

```
{"NAME":"W1901400","GPIO":[0,0,32,0,0,0,224,320,321,0,0,0,0,0,0,0,0,0,2720,2656,2624,
0],"FLAG":0,"BASE":1}
```

It is entered on the Configuration page, Configure Other link. The Activate checkbox also needs to be clicked. At this same time the name to be used for the plug can be setup. Other configuration that can be done is the MQTT Broker, MQTT Topic on the MQTT Setup link. A second WiFi SSID can also be added on the WiFi page.

The last step is the calibration of the energy management functions per [Power Monitoring Calibration - Tasmota] where the powerset, voltageset and currentset commands are used at the console.

21.14 Closet Door Light Control and Monitor

An after-the-fact light was installed in the main coat closet. The switch used for control of the light was mounted in the door frame and mains power routed through the switch. While this works well for shining light in the closet when the door is open, it has the downside of continuing to shine the light if the door is left ajar. To address this issue a Sonoff Basic was used in conjunction with the door jamb switch and a RCWL-0516 microwave radar motion sensor. When the door is closed the Sonoff makes certain the light is OFF. When the door is open the motion sensor retriggers a 100 second timer to keep the light on. If no motion for 100 seconds then the light turns out. If sometime later somebody approaches the closet the light will turn on again unless the door has been closed.

A 33,000 microfarad capacitor was placed between the power terminals of the motion sensor to minimize interference of this sensor with the power to the ESP8266 of the Sonoff Basic.

The RCWL-05016 is powered from 5VDC so the pickoff was at the voltage regulator of the Sonoff. This is shown in Figure 255. The sensor output to GPIO14 which is available at a header. Ground to the sensor is also available at the Sonoff header.

The door open contact sensor was connected at the Sonoff header to GPIO1 (TX). If this switch was not already mounted in the door jamb a basic window/door sensor as shown in Figure 315 would have been used to sense door open vs. closed. This is a reed switch and magnet to provide a dry contact signal.

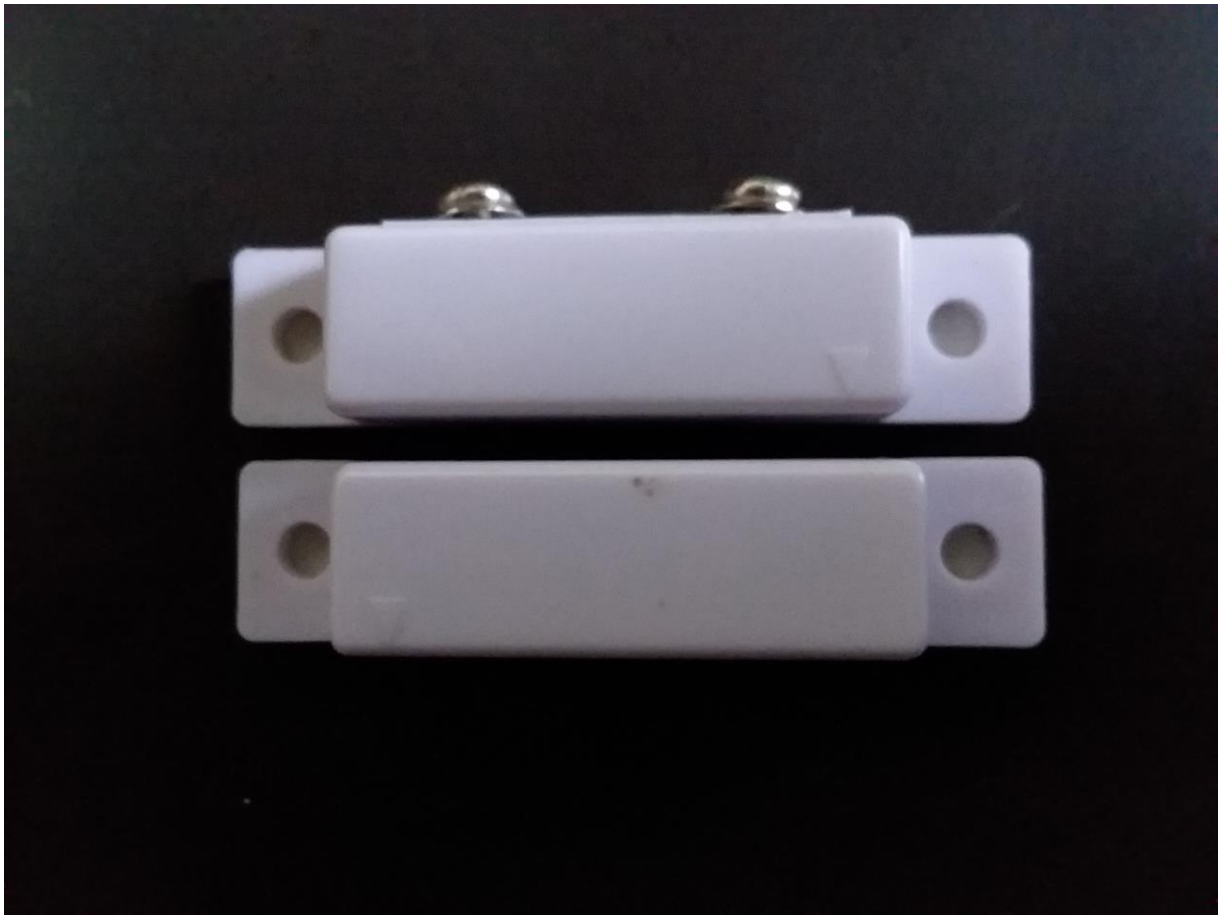


Figure 315 Generic Door-Window Sensor

Any version of Tasmota that supports Rules can be used. Rules were introduced in version 5.9. The module configuration is shown in Figure 316. The rules employed are shown below. Poweronstate assures the light is at the same state should a power cycle occur. Swithmode specify that the status follow the polarity of the door open sensor and the reverse-polarity of the radar sensor. Rule1 is for when the door is closed and Rule2 for when it is open. Rule1 turns the light on when the door sensor opens and enables Rule2. Rule2 keeps the light on as long as there is motion from the radar sensor and when the door closes it enables Rule1.

```
poweronstate 3
switchmode2 2
switchmode3 1
rule1  on system#boot do var1 0 endon on Switch2#State do power1 %var1% endon on Switch3#State=1 do backlog power1 1; var1 1; rule2 1; rule1 0 endon
rule2 on Rules#Timer=1 do power1 0 endon on Switch2#State do backlog RuleTimer1 100; power1 1 endon  on Switch3#State=0 do backlog var1 0; power1 0; rule1 1; rule2 0 endon
```

An addition could be made to Rule2 that sends MQTT message to tell the automation system that somebody should close the closet door. Its effectiveness will depend upon who is listening to the automation system (i.e. HS).

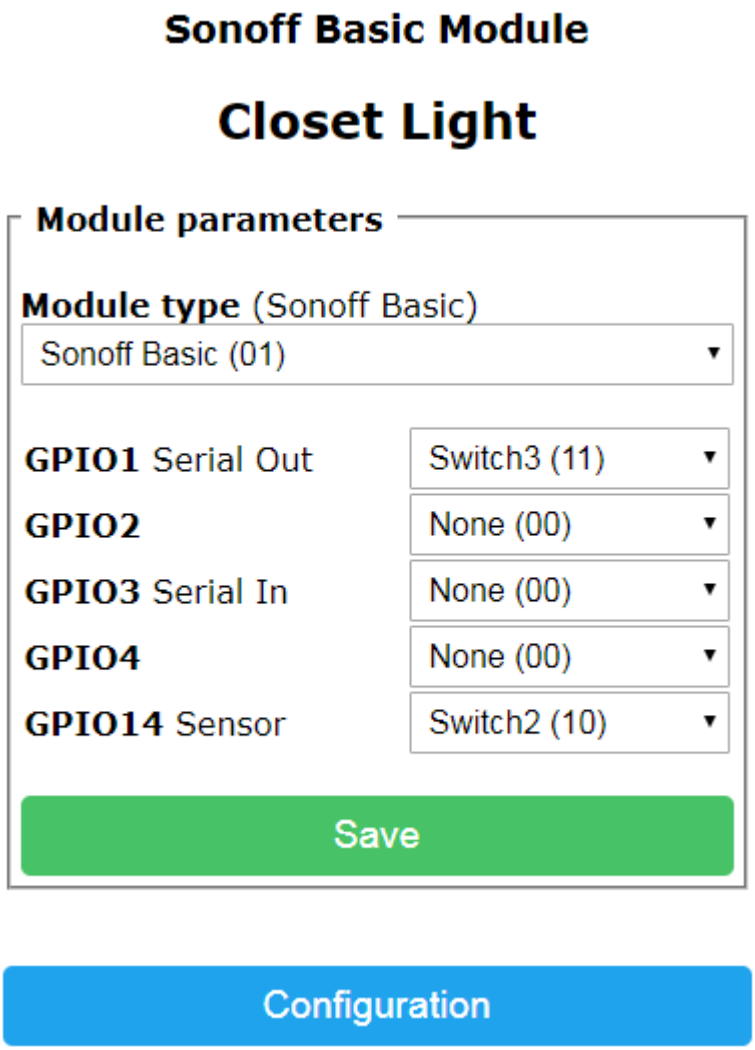


Figure 316 Closet Light/Monitor Configuration

A case was printed to house the radar sensor and Sonoff Basic. The case on the Sonoff Basic was not used since the printed case protects the device. This is shown in Figure 317 and Figure 318. Corners of the case have provisions for screws on two corners to mount into the back of the closet wall. Cutouts were included for heat control and access to the manual on/off button (GPIO0) of the Sonoff.

The wiring between power and the light is done in a j-box that contains a power receptacle. One side of the receptacle is standard power wiring and used to power the Sonoff. The other side was decoupled and contains the output of the Sonoff to control the light. The light is plugged into the lower receptacle; The Sonoff into the upper one. The wires from the case, save the power plug, were routed to inside the

j-box. Knurl plugs used to make the connection with the previously installed door jamb switch wires. The plug receptacle screw connection was used for the Sonoff output where the light plug is inserted. Plastic cable management channels where used to hide wiring from the Sonoff to the j-box. This could have been done behind the wallboard as well since the wiring runs were only one foot and placement not critical. The back of the closet, however, did not warrant too much effort since everything is behind the hanging coats.

Figure 317 Closed Light Contol Case Top

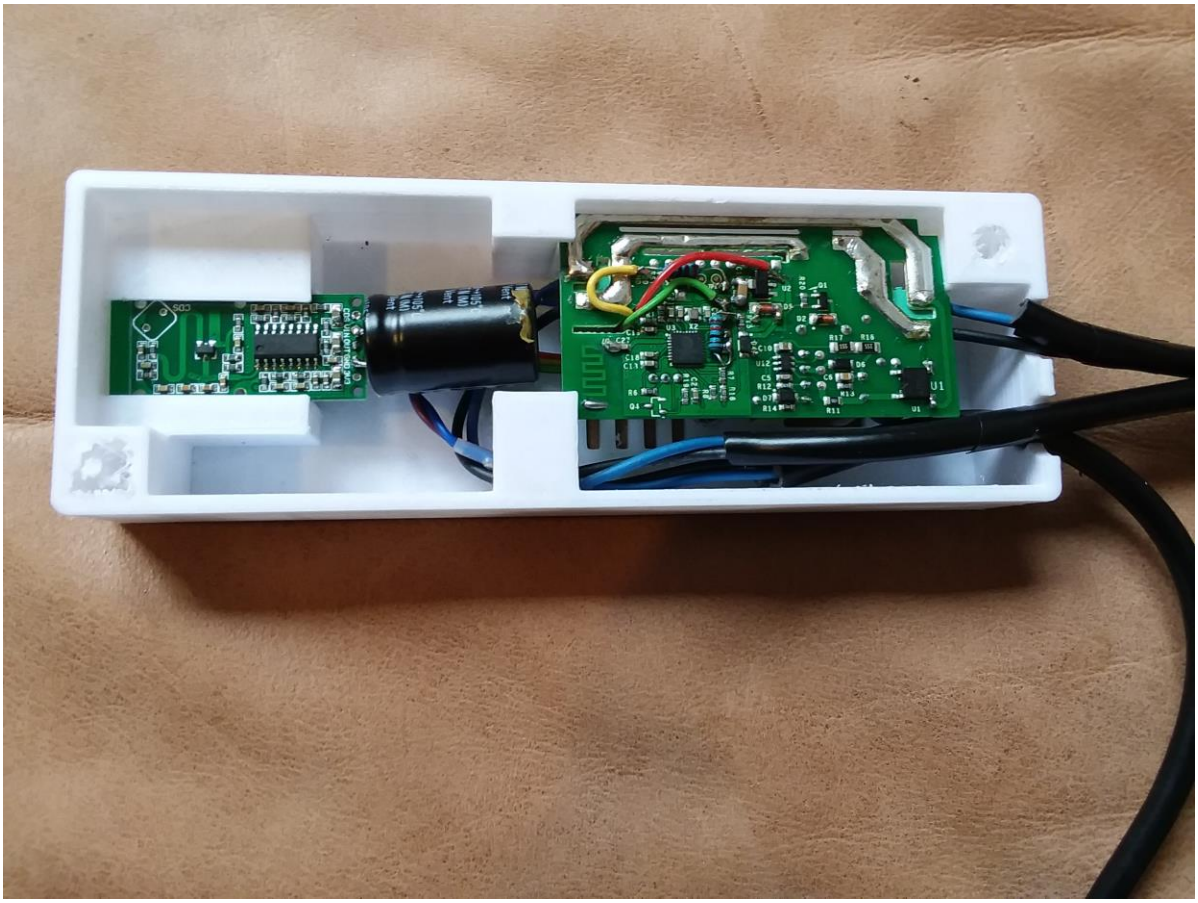


Figure 318 Closet Light Control Internals

In the case where power to the light is from the ceiling and a switch such as a pull string is used the same approach can be used. Likely an extension to the light fixture j-box would be installed using a plastic extension. If a metal one is used then the Sonoff and Radar sensor would need to be mounted outside the j-box since they are both RF devices. The Sonoff would receive full time power and the light would receive the Sonoff output.

The placement of the radar sensor is not critical since it is Omni-directional with range beyond the size of a closet. The door open/close sensor can be any dry-contact type device. I did explore the use of trigboard which is a WiFi device that sleeps until awaken by the sensor change of state. This would eliminate the wire from door sensor to the Sonoff. The problem I had with this device is that it could only awaken on a positive or a negative edge, but not both. It would work to turn the light on, but sensing the door closed could only be done on a hourly basis when the trigboard wakes up to report battery level.

21.15 Fake TV

A Fake TV is an item that simulates the light that is produced while watching a typical television. The simulated light patterns are produced by Neopixels. The core product is described in https://learn.adafruit.com/fake-tv-light-for-engineers/overview where a Arduino Uno is used to produce the light transitions.

I had a Uno that had been sitting around for years so this became a good use. In addition to the light control I also wanted to have remote control to enable and disable as well as automatically control the time of day when the simulated TV is on.

One approach was to augment the sketch loaded in the Uno. The concern is the effect that management software would have on the timing of the Neopixel control. Rather than changing software a more straightforward approach seemed to be changing the hardware to add a Sonoff Basic with Tasmota to perform the management.

The Uno and the Neopixels work off of 5V and the Neopixels current draw likely is beyond the capability of the 5V power supply built into the Sonoff. A wallwart of 0.5A was used for the project to power the Sonoff, Uno and Neopixels. The Sonoff power supply was no longer needed so a cut was made on the power resistor next to the relay to isolate the power supply. The 5VDC was provided directly from the wallwart to the input side of the 3.3V regulator. Figure 255 shows the connection point for the 5VDC.

The Sonoff relay will now be switching 5VDC power to the Uno. The walwart is wired to the power input of the Sonoff. The negative side of the wallwart input is the ground so it was soldered on the circuit card so the neutral input screw terminal was wired to the digital grounds of the circuit card. The relay now switches 5VDC rather than mains since the input is 5VDC rather than mains level.

As a bonus a temperature probe DS18B20 was also wired to the Sonoff GPIO14 input header. A 4.7K resistor was used to pull up this pin to 3.3v which was also on the header. It was not needed for the lighting control, but just became a convenient place to install a temperature sensor for the area of the room where the TV is located.

Standard Tasmota firmware with a compile that supports DS18B20 is used. Tasmota timers from the web GUI were setup for the daily timing of when the simulated TV is on and off. Remote control and temperature reporting are via MQTT topics. Figure 320 shows the configuration to control the relay and provide the temperature.

A case was printed with the Uno resting on the bottom and the Sonoff cradled above it. Dupont wires were used from the Uno header pins to the Neopixels and for the power wires from the Sonoff relay output. This is shown in Figure 319. The unit was installed in an alcove above the regular TV to maximize the appearance that the light was coming from the real TV.

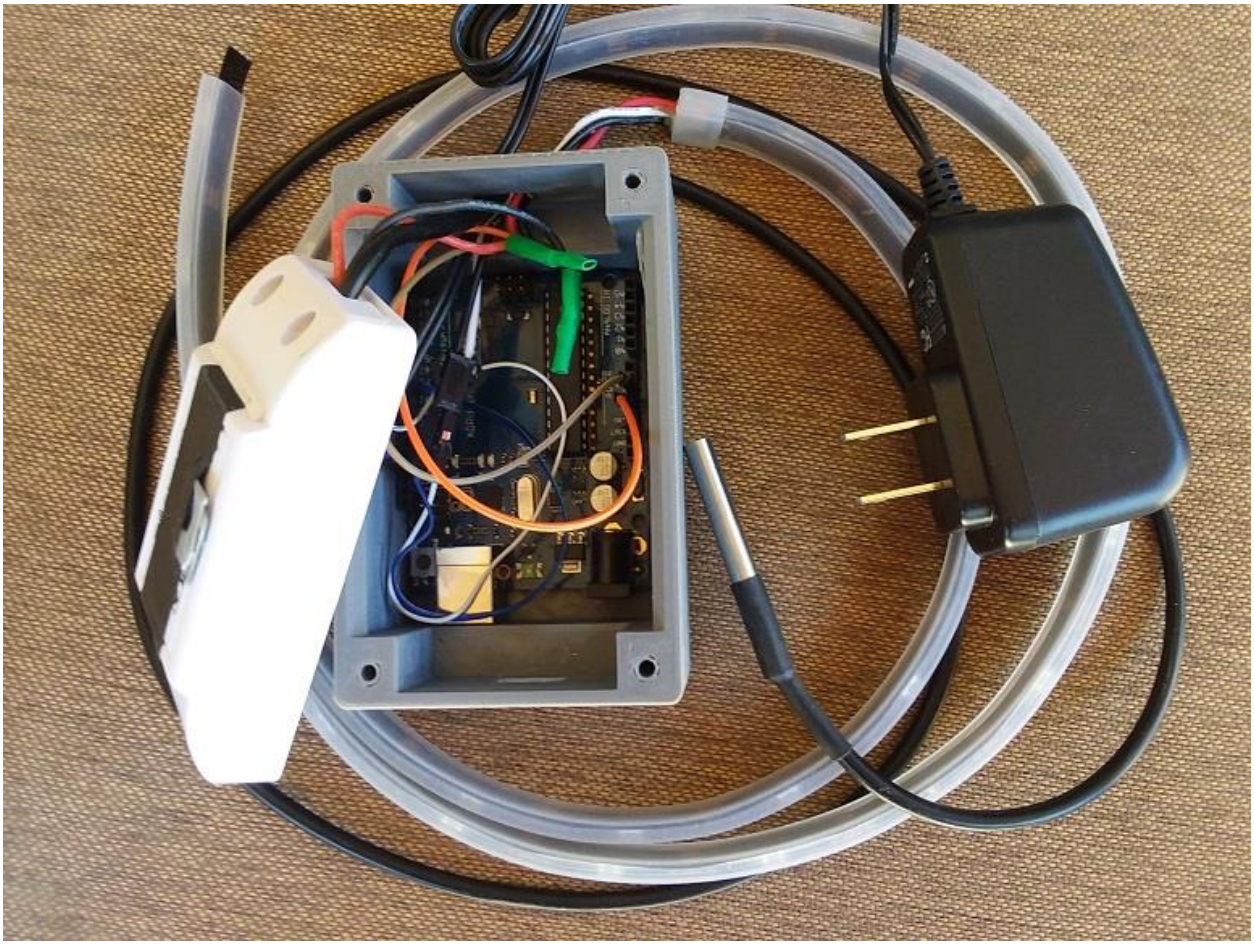


Figure 319 Fake TV Physical Construction

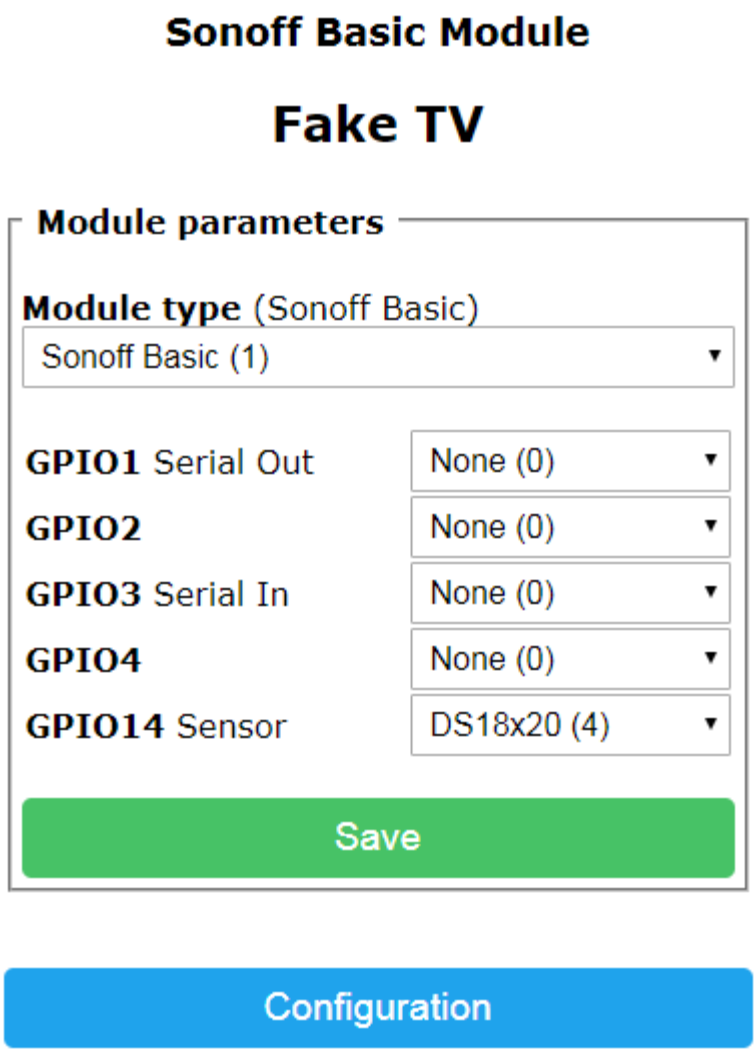


Figure 320 Fake TV Configuration

21.16 Bluetooth Low Energy Scanner

21.16.1 ESP32 Beacon Tracking

The prototype described in Section 21.16.7 demonstrated the ability to find BLE beacons using ESP32. This section describes the development and use of multiple ESP32 as scanners to perform triangulation to assess each beacon X/Y location. Use of one ESP32 has the ability to detect presence of a beacon. Use of two ESP32 has the ability to identify the relative location of beacons between the two ESP32. Use of more than two has the ability to identify the beacon position on a 100 x 100 X/Y grid.

The firmware for the ESP32 is a port of Tasmota for ESP8266. The starting point is located at https://github.com/btsimonh/Sonoff-Tasmota/tree/esp32-dev-firsttest which is based upon Tasmota 5.12.0c. Issues exist with this baseline with respect to keeping time of day, but this functionality is not needed for the beacon scanning functionality. New source and binary are located at http://mcsSprinklers.com/BLEScannerSource.zip. Binary only is also contained in http://mcssprinklers.com/ESP32BLEScanner.zip.

Initial flashing of the ESP32 must be done using serial connection. Subsequent flashing can be done Over The Air (OTA). The partition of the 4Mb flash is shown below. It is defined to maximize the size of program space while still allowing OTA programming. The BLE libraries are large and account for most of the delta from the basic ESP32 Tasmota and the one with the scanning functionality. Approximately 1.8Mb of the partitioned 2.0Mb is being used. Static RAM space grew by 1.5K and is now at 3K. Total RAM available for static and dynamic is 4K. Growth is primarily for the beacon address and friendly names. Provisions are made for 10 ESP32 scanners and 50 beacons.

```
otadata,    data, ota,     0xe000,  0x2000,
app0,       app,  ota_0,   0x10000, 0x1F0000,
app1,       app,  ota_1,   0x200000,0x1F0000,
eeprom,     data, 0x99,    0x3F0000,0x1000,
spiffs,     data, spiffs,  0x3F1000,0xF000,
```

New MQTT/Console commands have been added as shown in Table 7. The first must be entered at the console as this forms part of the MQTT topic. The others can be done via MQTT publish or the Tasmota Console.

21.16.2 Beacon Location Algorithm

The process for positioning a beacon is done in a two-step process. The first is distance and the second is triangulation. Distance is based upon the RSSI delivered in the Bluetooth data. Distance is measured from a defined location of scanner. The “ScannerLocation” topic is used to setup each scanner’s position.

Different beacons transmit with different power levels so the RSSI needs to be normalized. The calibration is done by taking two measurements. One with the beacon located 1 meter from the scanner. The other when it is located 10 meters away. There is much jitter and drift with the RSSI measurements due to external disturbances. It may even be the case that the 10-meter reading will not be available and it that case extrapolate from one that can be made at a shorter distance. I have used -100 as a ballpark in this case. Picking values is not critical because the intent is providing an

approximate position and not an exact location of a beacon. The two reading are communicated to the scanners via the "Beacon1Power" and "Beacon10Power" topics.

The RSSI and distance have a logarithmic relationship. If one was to use RSSI directly as a distance measurement then distances for highly negative RSSI would appear to much shorter than they are.

Inspiration for the equation used by mcsMQTT was obtained from https://iotandelectronics.wordpress.com/2016/10/07/how-to-calculate-distance-from-the-rssi-value-of-the-ble-beacon/ which was also referenced sites at Estimote.com, Kontakt.com and quora. The equation presented there is:

Distance = 10^((RSSI @ 1 meter) – (RSSI))/(10*Environment)

The equation used is:

Distance = 10^((RSSI @ 1 meter) – (RSSI)) / (RSSI @ 1 meter – RSSI @ 10 meter)

Converting multiple distance vectors to a X and Y coordinate depends upon the number of scanners that are able to see the beacon and calculate the distance from its location. If there is only one distance available then it could be anywhere on a circle (or sphere) with the center at the scanner's location. mcsMQTT will report the scanner X,Y in this case.

If there are two distances then mcsMQTT will report a X,Y on a line between the location of the two scanners. It will use the weighted contribution of each distance measurement where the weight goes to the scanner location that has the shorted distance identified.

As an example, assume two scanners with one at (0,100) and another at (50,40). The beacon distance computed for the scanners are 10 and 20 respectively. The inverse of the distance is used for the weighting so 0.1 and 0.05. The beacon is then located at:

X = 0 * (0.1/(0.1+0.05)) + 50 *(0.05/(0.1+0.05)) = 17

Y = 100 * (0.1/(0.1+0.05)) + 40*(0.05/(0.1+0.05)) = 79

Three and more scanners that can see the beacon will provide the best location determinate using a trilateration algorithm such is used to identify the epicenter of an earthquake based upon the seismometer readings at different locations. The distance vector computed by the scanner is the radius of a circle defined at $(x-x_n)^2 + (y-y_n)^2 = r_n^2$. Solving for pairs of these equations allows the second order terms to dropout thus making the solution a simple linear set of equations. The mcsMQTT implementation is modeled closely after the description provided at https://everything2.com/title/Triangulate. When more than three scanners can see a beacon then the same algorithm is used to obtain multiple results and then average the X and Y of each result.

21.16.3 BLE Scanning & Reporting

Two scanning parameters can be tweaked. "ScannerInterval" is the period or interval for each scan. "ScannerDuration" is the duration of each scan. The default is a 30 second scan every minute. The time when scanning is not performed is used for two purposes. One is to allow the antenna to be dedicated to WiFi. During scanning the antenna is multiplexed with priority given to WiFi. The other use is for

analysis on the results of the scan. It appears that analysis is well under one second so a minimum of one second is assured between the duration and the interval of the scan.

Shorter intervals should result in a more responsive recognition of a beacon that has moved, but short duration scans may not provide sufficient time to catch a beacon's announcement and this may make the beacon appear to be out of range. A 10 to 15 second scan duration appears to be sufficient for the beacons that I have been using.

With several scanners and many beacons, the amount of MQTT traffic can be significant if status is reported on every scan. The exchange of RSSI measurements among scanners is needed so triangulation can be performed to locate a beacon. However, for a beacon that is not moving there is no need to report the information from the latest scan if there is no material change in the data. The "BeaconReportingMode" topic is used to select the events upon which a beacon location report is made.

One of three reporting modes can be selected. One is report on every scan. This is useful when initially looking at the scanner's information. The second is to only report when the computed X or Y coordinate of a beacon changes. If no filtering is done then it is likely that most scans will result in a slightly different set of coordinates. When one or both Kalman filters are used the coordinates will be more stable, but also be dependent upon the measurement error defined for the filter. The third reporting mode is based upon a change in the calculated zone. The zone has hysteresis parameter so coordinate calculations that only bounce around within this hysteresis window will all look to be the same and no reporting will be done. The downside of a large zone hysteresis is that the other scanners will not be getting updates of changes in the RSSI measured by the other scanners and this could degrade into no scanner reporting. To overcome this downside to some degree all scanners will provide an update once every five minutes.

21.16.4 Signal Processing Filters

Three types of filters are used to improve the location determination. The Kalman filter operates on the principle of making a prediction of what the next measurement should be and then assess the error in the prediction vs. the actual measurement. At the same time the measurement is also known to have variability (i.e. error). This measurement variability is a user parameter. When the measurement is considered to have little error, it will be trusted more and the output of the filter will be responsive to change in the latest measurement. If there is high uncertainty of individual measurements, yet they all locus on the real value then the average of past values will be used for the filter output.

The beacons that I have been using change RSSI by about 30 when moving from the scanner to out of range. If we consider a 20% uncertainty of any given measurement then the measurement error used for RSSI filter will be 6. The same measurement error value is used for all beacons seen by a scanner and since all scanners are intended to be identical this RSSI filter values is used throughout the system. Experimentation for each user is needed to find a good balance of responsiveness vs. stability. The topic to set the RSSI error is "BeaconRssiError".

The second type of filter is a timeout. It is used for beacons where the advertisement is temporarily lost. It is also used to handover the master role from a scanner that has stopped reporting. The dropped beacon timeout is a user parameter. The non-reporting scanner is determined from the LWT message from the MQTT broker.

If we look at a scenario where four scanners are all seeing a given beacon. The beacon location will be calculated by the average of two three-point triangulations. If one scanner loses the beacon then the location is based upon the results of a single three-point calculation. This will change the reported beacon location somewhat. If a second scanner loses the beacon then triangulation is no longer possible. The calculation reverts to a weighted average based upon the remaining two reading and this result will try to position the beacon somewhere on a line between the two scanners. If there is only once scanner that sees the beacon then the location will be calculated as being at the same location as the scanner.

Because of the above scenario there needs to be tolerance to short term losses of beacon by a scanner. When it is lost the scanner will not calculate a new set of data but will continue to use the same data that was last calculated when it had visibility of the beacon. I have observed often two or three scans without a beacon reporting even though the beacon was close to the scanner. I have not done analysis of a large history of data to determine if three is typical or some other value should be expected tolerance level for dropouts. The downside of selecting a large dropout count tolerance is that a beacon that has moved will not be first sensed until after the timeout and then not reported until the effect works its way through the Kalman filter. The topic "BeaconDropout" is used to set the count of consecutive scans without beacon presence before the beacon is no longer reported as present from the scanner.

This is where the second Kalman filter plays a role. It will tend to smooth the X and Y location calculation and reduce the effect of scanners that drop out of the calculation. The XY Kalman filter will allow smaller dropout count filters since the effect of the dropout will not immediately propagate through the Kalman filter. The XY Kalman filter error term is set using the topic "BeaconXYError".

The third type of filter is hysteresis that is use for the Zone reporting. The Zone is a single value that is calculated as 100 * X + Y. The magnitude of hysteresis filter "BeaconZone" will determine how much either X or Y needs to change from the prior reported value of Zone before a new Zone will be reported. For example, starting with X=10, Y=20 and BeaconZone=5, the first report will be with Zone=1020. If in the next three scans the X changes to 11, 9, 14 and Y changes to 22, 23, 24 then Zone will continue to be reported as 1020. If on the fourth subsequent scan the X=13, Y=26 then Zone will be reported as 1326 and this will be the new center of the hysteresis window.

If any of the Kalman, dropout or hysteresis filters are to be turned off then publish the topic with payload set to 0.

21.16.5 Beacon Management

The ESP32 retains through power cycles information about the beacons it has already observed. Since the scanners share information to assure they are all reporting the same a beacon that is seen by one scanner is captured by the others when the first reports the beacon status. Non-volatile memory has been reserved for 50 beacons.

A beacon is identified by its MAC address. A scanner is identified by an instance number. The topics are formed from a base which is by default "BLEScan", followed by the scanner number and then followed by the topic of interest. In the case of beacon reporting the topic of interest is the friendly name of the beacon. The name can be up to 17 characters. Until a name is defined the name will be a derivative of the MAC address. A typical beacon topic will then be BLEScan/1/MyBeacon.

Since most commands apply to all scanners it is desirable that one topic sent can be understood by all. This is done using the Group Topic. By default, it is “BLEScanners”. If one desires to give the ‘MyBeacon’ name to the beacon with address ‘4f:9a:d1:63:7f:aa’ to all scanners then the topic would be BLEScanners/cmnd/BeaconName with payload of 4f:9a:d1:63:7f:aa,MyBeacon. It could of also be done with multiple commands with topics such as BLEScan/1/cmnd/BeaconName and BLEScan/2/cmnd/BeaconName using the same payload in each.

Scanning seems to collect beacons that are either not real or at least not recognizable. Rather than accumulating a bunch of these, the “BeaconDisable” topic can be used to ignore any additional beacons from being added to the ESP32 memory. This would be done after all recognizable beacons have been discovered.

To stop the reporting of a don’t-care beacon the “BeaconBlacklist” topic can be used. This will not free up memory in the ESP32 for new beacons, but it will reduce the MQTT traffic.

To actually reclaim the memory the “BeaconRemove” topic is used. One payload option is to use ‘all’ and that will erase memory of all previously scanned beacons. It will also erase the names of these beacons. Individual beacons can also be removed. After a beacon has been removed it is still possible for it to announce itself and reclaim the memory if BeaconDisable has not been set.

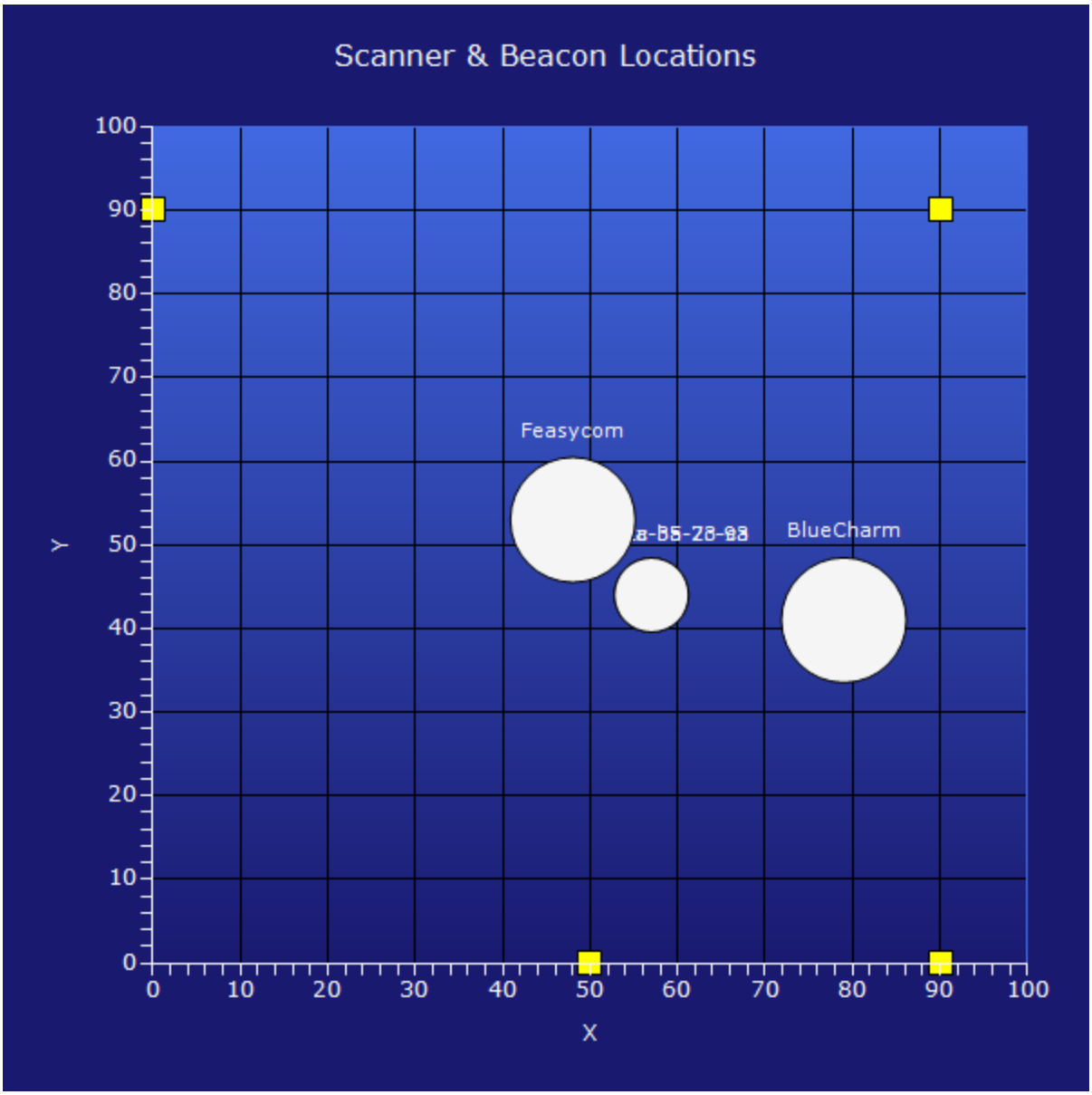


Figure 321 Beacons shown with good scanner separation

Table 7 BLE Scanner Tasmota Commands

Command	Description
ScannerLocation <id>,<X>,<Y> Used to identify the position of the ESP32 performing the scan on a 100 x 100 grid. The <id> is used to augment the topic to uniquely identify the messages being published and commands being given. This should be entered from console because the <id> is part of the MQTT topic. On initial load of ESP32 the <id> will default to 0 so this command can be sent via MQTT such as below to define id=2 at xy of 90,90 Topic: BLEScan/0/cmnd/ScannerLocation Payload: 2,90,90 After sending it the ESP32 needs to be reset to take on the topic definition	<id> is the index number of the ESP32 that performs the scan. Numbers in range 1 to 10 are supported <X> is the integer position of the ESP32 on a 100 point line in range of 0 to 99 <Y> is the integer position of the ESP32 on a 100 point orthogonal line in range of 0 to 99
ScannerInterval <#>,<#> Scanning is performed periodically with the period specified by the first parameter. During this period the ESP32 will be listening for beacons for the duration of the second parameter. The balance of time is used to do the analysis of the beacon RRSi data and dedicate the antenna for WiFi use.	<#> are integer number of seconds between the start of each scan and the duration of each scan respectively. Small values are problematic due to both lack of beacons that will be discovered in a short scan and the interactions that result in use of ESP32 resources. Most evaluation has been done with a 50% duration, but significantly higher duty cycles should be able to be achieved.
ScannerRetain <0|1> MQTT protocol retain flag that can be applied to the messages published by the ESP32 for configuration of the scanner and of the beacons. Retain flag is never used on the beacon location reporting.	<0> turns off the retain flag <1> turns on the retain flag
ScannerMaster <#> The master scanner is considered to have the true position of all beacons. All scanners will publish the X, Y and Zone information computed by the master scanner. Normal operation is <0> to allow the ESP32's to	<0> for auto-select <1..10> for specific scanner

determine the best scanner to serve the master role. It will determine it as the scanner with the lowest ID of those which are online.	
ScannerGain <#> The scanner gain accounts for the strength of the receiver antenna as well as any receiver sensitivity due to nearby obstructions. A scanner with a higher gain antenna will result in less negative RSSI readings and thus apparent shorter distance to the beacon. To account for this a gain of 110% will increase the distance measured for every beacon by 10%	<#> is a percentage with a default value of 100 and range of 0 to 255. No % suffix is used. Higher gain values will increase calculated distances to the beacons. Lower gain values will make the beacons appear closer by reducing the distance calculation.
BeaconTxPower <address>,<#1>,<#10> RSSI reading at 1 meter and 10 meter between beacon and scanner. Beacon TxPower establishes the base power of each beacon and a second point on the log curve that is used to calibrate the beacon RSSI to distance transformation.	<#> is in range 10 to 150. Default 60,100. It is separated from <address> with a comma character. Used in distance calculation: Distance = 10^((P-RSSI)*C) P = RSSI @ 1 meter C = 1/(P-RSSI @ 10 meter)
BeaconName <address>,<name> Each beacon will identify itself with a 17 character <address> using MAC-notation. This< address> is what is used by the ESP32 firmware to uniquely identify a beacon. The <name> is the friendly name used in the MQTT topics to identify the beacon. A typical topic is of the form "BLEScan/1/MyBeacon" where 1 is from the ScannerLocation command and MyBeacon is <name>.	<name> can be up to 17 characters. It is separated from <address> with a comma character. The <address> will be observed after scanning starts and MQTT topics are published. Until a <name> is assigned the topic will use the beacon address with hyphen replacing the colon.
BeaconGroup <0/1>	<0> Do not group <1> Group all beacons that do not have a manufacture code recognized in the MAC address into single beacon with MAC address of FF:FF:FF:FF:FF:FF Smartphones typically randomize their BLE MAC address to protect privacy. While it is not possible

	to distinguish smartphones when multiple are present, it will be possible to detect if any smartphone is present.
BeaconRssiError <#> Kalman filter can be employed on the RSSI signal from a beacon to remove jitter from the RSSI reading.	<0> = do not use Kalman filter (default) <1..50> = use Kalman filter with RSSI measurent error of the specified value. A Kalman filter is used to account for RSSI variation in measurement. A larger number reflects greater variably in the quality of the measurement. A value under 5 is reasonable.
BeaconXYError <#> Kalman filter can be employed on the triangulated X and Y position calculations of a beacon to remove jitter from the location reporting.	<0> = do not use Kalman filter and provide no hysteresis on Zone report(default) <1..50> to apply a Kalman filter with the entered error. As beacons drop out from the scanner's range the calculation will be affected by the smaller sample size. Filtering this effect will reduce variability in the(X, Y) location reporting. It also used as the Zone window hysteresis. The Zone is the single value reporting of the X & Y parameters. A value under 10 is reasonable.
BeaconZone <#> Amount of hysteresis that is allowed before a Zone is changed. The Zone is the single value representation of X and Y using expression 100*X+Y. The hysteresis is the change in either X or Y that will change the reported Zone.	<#> range 1 to 50. Default 5
BeaconReportingMode <0|1|2> Reporting mode of the beacon locations. The lower the mode selected, the chattier the reports will be.	<0> = report on every scan [1 minute] (Default) <1> = report on change of X or Y <2> = report on change of Zone
BeaconDropout <#> Number of consecutive scans without detecting beacon before last received RSSI is excluded from beacon X/Y calculation	<#> range is 1 to 50, (default 3) Smaller values will make quicker reporting of a beacon that has gone out of range at the expense of jitter in the location X,Y reporting. Larger values will stabilize location reporting for a non-moving beacon. Values above 2 are reasonable.

BeaconDisable <0/1>	<0> = add newly discovered beacons (default) <1> = do not allow new beacons to be added
BeaconRemove <address> | <all>	Erase memory of all discovered beacons or memory of a single beacon
BeaconBlacklist <address>[,0]	Mark a beacon for exclusion in reporting position. This is different than BeaconRemove because if a removed beacon is seen again then it will be added back and continue being reported. If a previously blacklisted beacon is to be added back then the optional [,0] can be appended to the command to indicate that it should no longer be blacklisted.

The setup of beacon names is conveniently handled with Publist tab of mcsMQTT. The example in Figure 322 shows two beacons being defined to each of the ESP32 scanners. Note that the group topic "BLEScanners" is used to cover all the ESP32 doing the BLE scan function. The Tasmota command GroupTopic is used (e.g. GroupTopic xyz from MQTT or Console) to change this group name.

In this publist I also included the filtering parameters so every ESP32 will be operating off of the same criteria.

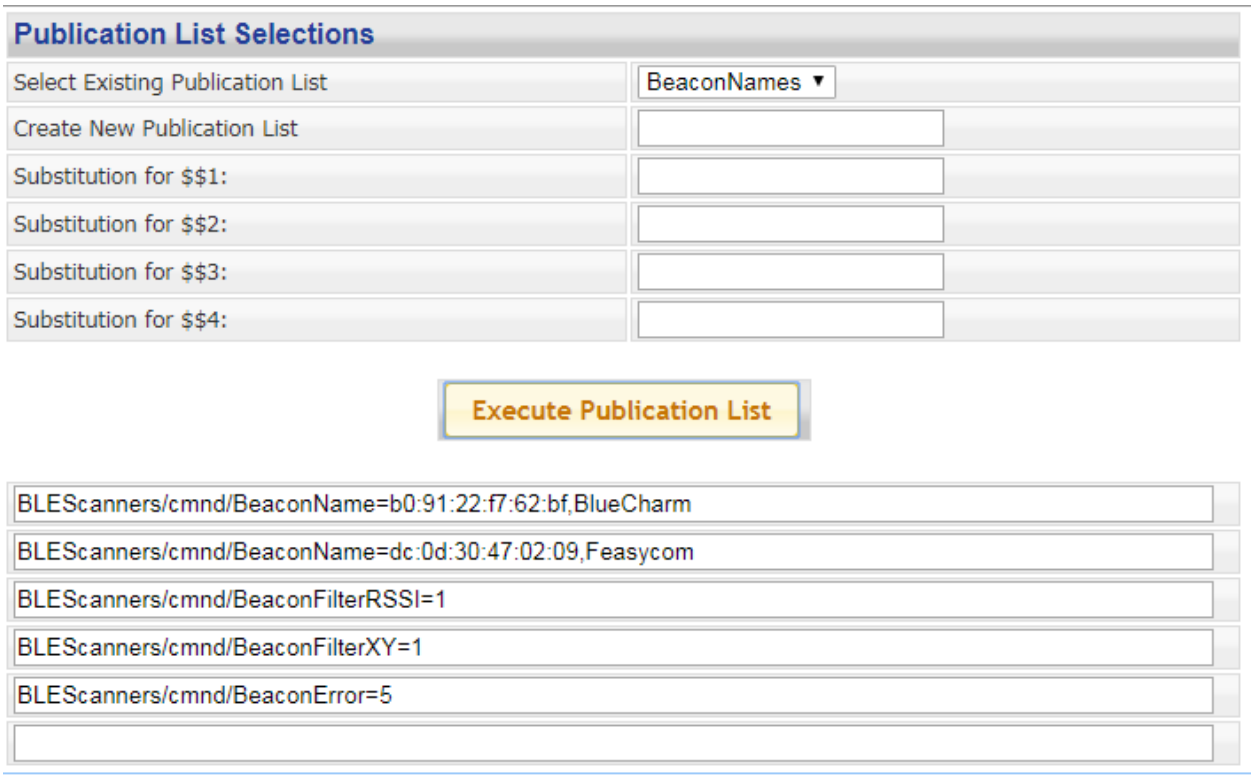


Figure 322 Publist to define beacon friendly names

Other than acknowledgement of commands, there is a single published topic per beacon using JSON encoding in the format shown below:

```
Topic: BLEScan/1/BlueCharm
Payload:
{"Scan":{"ScannerX":30,"ScannerY":60,"Address":"b0:91:22:f7:62:bf","RawRSSI":38,"FilteredRSSI":
38.48,"Present":1,"Master":1},"Location":{"Distance:5,"BeaconFOM":3,"BeaconX":10,"BeaconY":90,"Zo
ne":1090}}


Topic: BLEScan/1/4a-63-b4-eb-d3-fa
Payload:{"ScannerX":30,"ScannerY":60,"Address":"4a:63:b4:eb:d3:fa","RawRSSI":90,"FilteredRSSI":
90.04,"Present":1,"BeaconFOM":1,"BeaconX":30,"BeaconY":60,"Zone":3060}
```

Payload Key	**Description**
Scan:ScannerX	X axis position of reporting scanner. Each scanner will report slightly different values based upon variance in timing, but if filtered will tend to converge over time.
Scan:ScannerY	Y axis position of reporting scanner. Each scanner will report slightly different values based upon variance in timing, but if filtered will tend to

	converge over time.
Scan:Address	Beacon address
Scan:RawRSSI	Last RSSI seen by ESP32 for this beacon
Scan:FilteredRSSI	RSSI after filtering
Scan:Present	1 = detected in at least one of the recent scans. BeaconDropout sets the magnitude of number of scans that are considered recent 0 = not detected in most recent scans
Location:Scanner	Identification of the scanner that was selected to be the one as the master. Masters are selected based upon having the lowest scanner ID for those scanners that are online based upon LWT status. When a new master is selected it will initialize the X and Y coordinates of every beacon to the coordinates last reported by the previous master.
Location:Distance	Distance in feet calculated based upon RSSI and calibration measurements at 1 and 10 meters. Distance will be set to -1 if reporting scanner no longer has the beacon in range
Location:BeaconFOM	Figure Of Merit which is the count of the number of scanners that see beacon in its scan. It is affected by the BeaconDropout filter. When it goes to zero the reported beacon measurements are set to -1 to indicate that the X , Y and Zone values are no longer valid.
Location:BeaconX	Calculated X coordinate based upon calculated distance. It is affected by the filters being employed. It is set to -1 when no scanner has the beacon in range.
Location:BeaconY	Calculated Y coordinate based upon calculated distance. It is affected by the filters being employed. It is set to -1 when no scanner has the beacon in range.
Location:Zone	Location as represented as X*100+Y. Zone will remain constant until X or Y exceeds BeaconZone hysteresis. This is intended to be used as a stable singe-value marker when determining if a beacon has changed locations. It is set to -1 when no

	scanner has the beacon in range.

21.16.6 BLE on Raspberry Pi

There are two configurations for BLE support on Linux. One is associated with the HS3 mcsMQTT plugin and uses the the same API developed for the ESP32. This implementation provides for position isolation using multiple scanners to triangulate. The second is associated with the HS4 mcsMQTT plugin. It is a simplified version that provides presence detection.

The RPi 3/4 has a built-in Bluetooth capability. Other devices with a Bluetooth dongle should also work in a similar manner, but these have not been attempted.

The RPi has an advantage over ESP32 in that it does not share hardware resources between WiFi and Bluetooth. This allows for scan intervals that are not limited by the hardware, but only by the Bluetooth specification for response time to the advertise packet. Continuous one-second scan can be achieved, but more practical timing is likely more appropriate to conserve beacon battery and RPi computational resources.

Scan interval and duration can be as short at one second. Limits are 1 to 1000 seconds for both. This effectively provides continuous scanning. The ESP32 is not able to achieve these fast rates due to sharing of 2.4GHz antenna for both Wifi and Bluetooth. Minimum five-second intervals are enforced on the ESP32.

Settings are stored in file BLEScan.ini on the RPi rather than in flash used for ESP32. Settings are the same as for ESP32 as shown in Table 7 with the addition of those in Table 8. The keys in BLEScan.ini are all upper case.

21.16.6.1 Installation and Setup on RPi

The source and object code for HS3 version are at http://mcsSprinklers.com/BLEMQTT.zip . The source and object for the HS4 version are in the Updater zip files associated with each version.

One file is needed called BLEMQTT. It can run from any folder on the RPi. This description assumes it is placed at /usr/local/BLEMQTT for HS3 version and will be in a HomeSeer subfolder \bin\mcsMQTT. BLEMQTT should be made executable from command line with one of the two following or other methods such as with WINSCP properties.

`chmod +x usr/local/BLEMQTT/BLEMQTT` (HS3)

`chmod +x usr/local/HomeSeer/bin/mcsMQTT/BLEMQTT` (HS4)

Additional libraries will likely need to be installed is there are errors when starting BLEMQTT. In particular the ones shown in gray boxes of Section 21.16.6.2 will be needed.

The RPi application is BLEMQTT for HS3 that can be run from the command line or as a service under systemctl. For HS4, mcsMQTT manages the execution of BLEMQTT. Testing was done using DietPi (Debian Stretch & Buster), Raspian (Debian Buster), but any OS compatible with the RPi-3 or RPI-4 should provide the same result.

For HS4 the BLEMQTT is a stripped-down version that does not use .ini support. Two versions are available with the only difference being the level of detail that is written to the file BLEMQTT.Trace. BLEMQTT_TRACE contains additional detail.

The HS4 version is started from the command line from the BLEMQTT folder as:

sudo ./BLEMQTT brokerIp brokerusername brokerpassword

If no brokerusername or brokerpassword is provided then the connection is attempted without login credentials.

If there are errors reported about libraries not already installed then look at the next section for getting these libraries.

The BLEMQTT.service file that is located in /etc/systemd/system contains:

```
[Unit]
Description=BLE Beacon tracking via MQTT
After=network.target

[Service]
ExecStart=/usr/local/BLEMQTT/autostart_BLEMQTT
Restart=on-failure
TimeoutStopSec=90

[Install]
WantedBy=multi-user.target
```

With BLEMQTT installed in /usr/local/BLEMQTT the autostart_BLEMQTT fie contains:

```
#!/bin/sh
export LANG=en_US.UTF-8
cd usr/local/BLEMQTT
sleep 10s
/usr/local/BLEMQTT/BLEMQTT
```

autostart_BLEMQTT and BLEMQTT need to be executable (e.g. chmod +x `usr/local/BLEMQTT/BLEMQTT`). The service also needs to be enabled with command line "systemctl enable BLEMQTT.service". Manual stop, start and status can be done with "systemctl stop BLEMQTT.service", "systemctl start BLEMQTT.service" and "systemctl status BLEMQTT.service" respectively.

Before enabling the autostartup it is best to run manually from the command prompt (cd /usr/local/BLEMQTT followed by sudo ./BLEMQTT localhost). This will provide feedback in the console to confirm things are working as expected. It can be manually stopped from the command line using Ctl-C which will be recognized by BLEMQTT as an exit signal.

The HS3 version initial start will create the BLEMQTT.ini file as the BLEMQTT which then can be edited to setup the configuration. The only items that need to be done manually are the SCANNERID to be a unique number between 1 and 10 where 1 is best for the first scanner on the network. The MQTT parameters should also be defined so mcsMQTT interface is established. MQTTBROKER, MQTTTOPIC, MQTTGROUP and MQTTPERIOD should be set. An example is:

```
[BEACON]
42:00:07:B2:C6:C7=60,60,60,60,60
B0:91:22:F7:62:BF=BlueCharm,60,100,0,0
DC:0D:30:47:02:09=Feasycom,60,100,0,0
45:00:E7:41:9B:60=60,60,60,60,60
[CONFIG]
```

```
SCANNERID=1
SCANNERGAIN=100
SCANNERX=30,30,20,0,0,0,0,0,0,0,0
SCANNERY=3,25,30,0,0,0,0,0,0,0,0
BEACONSCANDURATION=2
BEACONRSSIERROR=2
BEACONXYERROR=0
BEACONDISABLE=1
BEACONDROPOUT=4
BEACONZONE=5
BEACONREPORTINGMODE=0
BEACONSCANINTERVAL=5
BEACONRETAIN=0
BEACONMASTER=0
MQTTBROKER=tcp://localhost:1883
MQTTTOPIC=BLEScan
MQTTGROUP=BLEScanners
MQTTPERIOD=60
```

The other parameters are most easily setup from mcsMQTT BLE page, but they can also be done by direct edit of BLEMQTT.ini. The API for these is defined in Table 7 with the RPI-specific extensions in Table 8. For HS4 version there is no need for any user action with the API.

Table 8 BLE Scanner RPi Tasmota Command Extension

Command	Description
MQTTBROKER <address> MQTT Broker address. While it is not possible to deliver this setting via MQTT it should be done by manual edit of BLEScan.ini. This is a bootstrapping issue.	<address> is in the form "tcp://localhost:1883" where localhost is the network name or IP address and 1883 is the port . Default is "tcp://localhost:1883"
MQTTTOPIC <topic> Base topic for MQTT publication and subscription Actual topic used in MQTT messages will have appended the Id of the scanner as well as the command	<topic> is the base topic for the scanner. Default is “BLEScan”
MQTTGROUP <topic> Group topic for MQTT publication that will be recognized by all scanners	<topic> is the base topic for the group of scanners. Default is “BLEScanners”
MQTTPERIOD <#> Define the telemetry period for scanner and beacon state information	<#> is a number of seconds. Default is 300. Limits are 1 and 1000.
MQTTEXIT <#> Action to be taken when broker disconnects. It can be internal attempt to reconnect or systemctl to restart BLEMQTT to reconnect	<#> = 0 if broker disconnect uses internal auto reconnect method. The MQTT library does not appear to reconnect automatically (or produce disconnect callbacks) <#> = 1 if disconnect causes BLEMQTT to exit and allows systemctl to restart BLEMQTT to establish new broker connection
MQTTLOG <#> Log file is BLEMQTT.trace. The detail of information going to the log is controlled by this parameter.	<#> = integer to represent the level of detail in the log file. The options are: TRACE_MAXIMUM 1 TRACE_MEDIUM 2 TRACE_MINIMUM 3 TRACE_PROTOCOL 4 LOG_ERROR 5 LOG_SEVERE 6 LOG_FATAL 7 The lower numbers produce more detail in the log

21.16.6.2 Installing dependencies and Compiling from Source

Three source libraries are used. One for Bluetooth, one for MQTT, one for Ini in the HS3 version and then the main BLESCAN.c program. MQTT requires the openssl library which may or may not already be installed on the Linux distribution.

The steps in the following three grayed boxes are needed unless the dependent libraries are alredy installed.

openssl is installed with:

```
sudo apt-get install libssl-dev
```

MQTT is obtained from https://www.eclipse.org/paho/clients/c/. Any folder can be used when cloning the source. The actual file being used is libpaho-mqtt3a.so. The build instructions are contained on the reference page which are:

```
sudo apt-get install git
sudo git clone https://github.com/eclipse/paho.mqtt.c.git
cd paho.mqtt.c
sudo make
sudo make install
```

The async library is the one that was implemented and its object was created as `/usr/local/lib/libpaho-mqtt3a.so`

The bluetooth library is obtained using apt-get from command line:

```
sudo apt-get install libbluetooth-dev
```

The main program BLEMQTT.c is available at http://mcsSprinklers.com/BLEMQTT.zip for HS3 or Updater package for HS4. For HS4 two binary executable are produced. BLEMQTT_TRACE contains a log of the beacons detected in the created BLEMQTT.Trace file. This can be run to confirm beacons are being detected. Normally BLEMQTT is run with the created file only containing execution errors.

If one desires to build the BLEMQTT executable from source the following line is used after navigating to the folder where BLEMQTT.c was placed.

```
sudo gcc BLEMQTT.c -lbluetooth -lm ini.o /usr/local/lib/libpaho-mqtt3a.so -o BLEMQTT
```

A dependent libarary that may be required is installed with:

```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

Ini is only used for the BLEMQTT that performs translateration with support in HS3 mcsMQTT. It is not used for the HS4 implementaion. Ini was obtained from https://github.com/benhoyt/inih. The required file ini.o is in BLEMQTT.zip and placed in the same /usr/local/BLEMQTT/ folder. Minor modification was made and the modified source ini.c/ini.h is included in the same zip as BLEMQTT.c described later. The files were copied to RPi from the Windows download into the BLEMQTT folder. The ini.o library was

created from RPi command line `gcc -c -fPIC ini.c -o ini.o` after navigating to BLEMQTT folder with "cd /usr/local/BLEMQTT".

21.16.6.3 Filtering

For H4 a Low Pass filter was used. For HS3, filtering was evaluated for Kalman and LowPass filters. Results were similar with the LowPass having greater user control of the filter response and a more intuitive cause-effect relationship of filter constant and the response. Figure 323 shows the Low Pass filter with time constants (RSSIError) of 1 and 10 and a steady vs. moving beacon. This data shows there is random noise about a measurement as well as a periodic sawtooth of about two seconds. Figure 324 shows the evaluation using an RSSIError value of 20. The heavier filtering provides a stable steady state RSSI and about 3 minute response time to a beacon that moved 20 ft. Another observation is that there is a greater fluctuation in RSSI for beacons at a larger distance than at a shorter distance.

If the raw RSSI is stable then a low filter time constant will give the best results. If there is more need for position stability and response time is not critical then higher values will work better. The limits are 1 and 50.

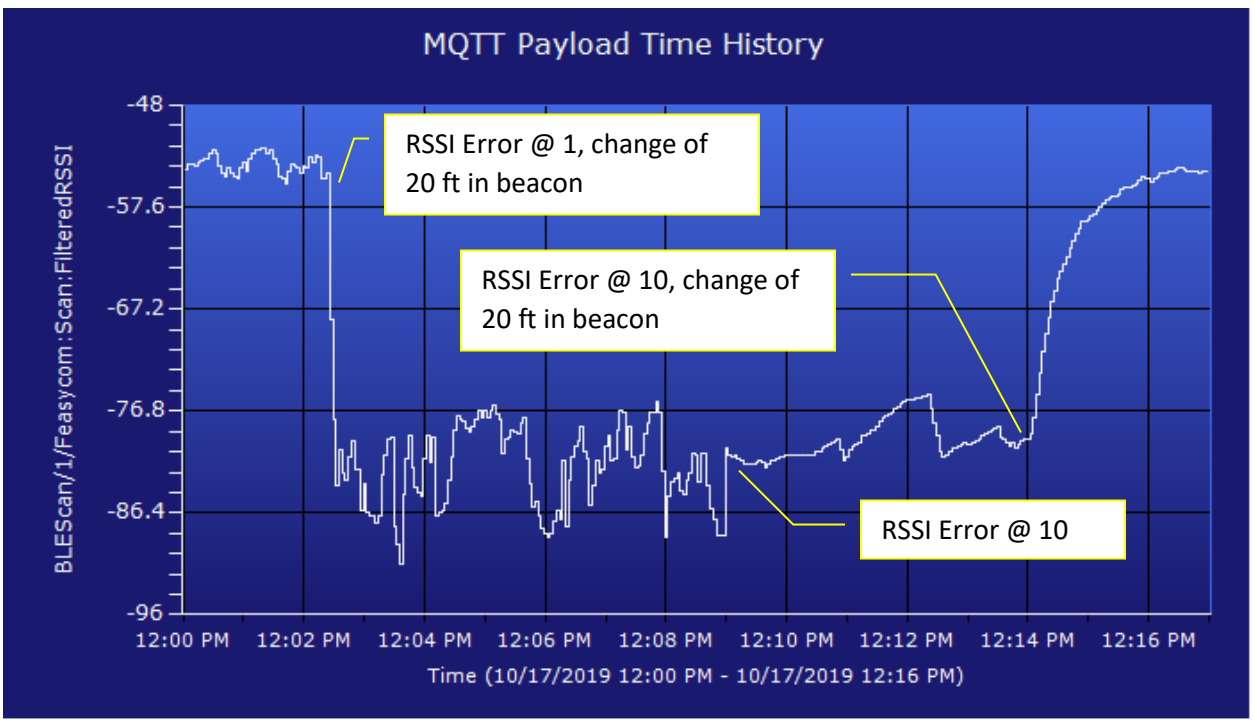


Figure 323 Low Pass Filter Evaluation

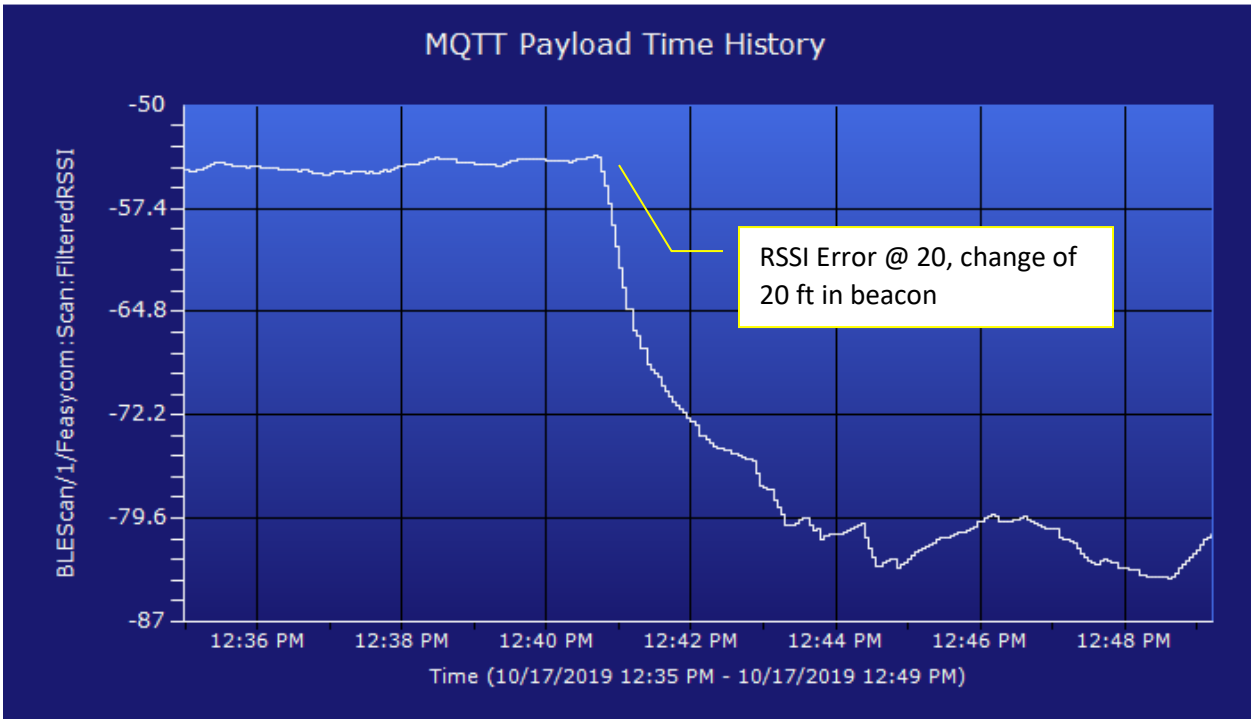


Figure 324 Low Pass Filter Response with RSSIError 20

21.16.7 Prototype

The ESP32 contains a Bluetooth capability that in this project is used to scan the available devices. The project is based upon the work done in https://www.instructables.com/id/Nano-ESP32-BLE-Scanner/. Reference was also made of the YouTube video by Andreas Spiess at https://www.youtube.com/watch?v=k_D_Qu0cgu8 for making use of the two cores of the ESP32.

One core is used to perform the Bluetooth scan. The other core is used to manage WiFi/MQTT and evaluate the results of the Bluetooth scan. The scan is a 30 second activity and repeated at approximately 60 second intervals. Semaphore is used to coordinate the scan and the reporting of the scan results.

The report consists of two payload formats. The first reports a payload of Inactive or Active for the topic that is specified as BLEscan/address where address is the device Bluetooth address with colons replaced by hyphens. The second is BLEscan/address/INFO with JSON payload of everything reported during the scan. The JSON keys will consist of one or more the following: Address, Name, Appearance, ManufacturerData, ServiceUUID and TxPower.

The log below shows three hours of reporting where device is detected, no longer detected and then detected again. The actual devices are unknown and no activity was performed to intentionally enable of disable a Bluetooth-capable device.

When an Android smartphone or an Amazon Echo were powered-off and powered-on there was no specific correlation with the Bluetooth scan. Until a better understanding of what the scan is actually doing and device dependencies are recognized then an application of the BLE scanner will not be possible.

In any case is still provides a basis for further work using ESP32 where the technology of Bluetooth scan, MQTT and use of multiple cores is demonstrated.

The sketch is provided at the end of this section. Setup of Arduino IDE for ESP32 is available several places including https://www.youtube.com/watch?v=DgaKlh081tU. This sketch compiles to 1.5Mbyte and SPDIFF is not used. Most ESP32 development boards have 4 Mbyte. This means that it will fit in a generic development board and still have OTA capability after initially flashed. This option is setup under Tools/Partition Scheme in the Arduino IDE.

3:20:20 PM Received BLEscan/5d-49-bc-fb-08-f9={"Address":"5d:49:bc:fb:08:f9","RSSI":-63,"ManufacturerData":"4c001005171cf14dc3"}
3:22:20 PM Received BLEscan/53-c6-e0-13-ee-23=Active
3:22:21 PM Received BLEscan/5f-4a-35-96-16-7a=Active
3:22:21 PM Received BLEscan/5f-4a-35-96-16-7a/INFO ={"Address":"5f:4a:35:96:16:7a","RSSI":-58,"ManufacturerData":"4c001005171c725483"}
3:23:21 PM Received BLEscan/4e-7c-03-13-bd-1c=Inactive
3:23:21 PM Received BLEscan/5d-49-bc-fb-08-f9=Inactive
3:26:21 PM Received BLEscan/4d-c0-97-28-68-3b=Active
3:26:21 PM Received BLEscan/4d-c0-97-28-68-3b/INFO ={"Address":"4d:c0:97:28:68:3b","RSSI":-96,"ManufacturerData":"4c0010050b1c62c94b"}
3:27:21 PM Received BLEscan/4d-c0-97-28-68-3b=Inactive
3:30:20 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-89,"ManufacturerData":"4c0010050318c75348"}
3:30:21 PM Received BLEscan/5f-4a-35-96-16-7a/INFO ={"Address":"5f:4a:35:96:16:7a","RSSI":-63,"ManufacturerData":"4c001005171c725483"}
3:37:21 PM Received BLEscan/52-a5-7d-88-19-74=Active
3:37:21 PM Received BLEscan/52-a5-7d-88-19-74/INFO ={"Address":"52:a5:7d:88:19:74","RSSI":-71,"ManufacturerData":"4c001005171cb30b17"}
3:38:21 PM Received BLEscan/5f-4a-35-96-16-7a=Inactive
3:40:21 PM Received BLEscan/52-a5-7d-88-19-74/INFO ={"Address":"52:a5:7d:88:19:74","RSSI":-61,"ManufacturerData":"4c001005171cb30b17"}
3:40:21 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-78,"ManufacturerData":"4c0010050318c75348"}
3:50:21 PM Received BLEscan/52-a5-7d-88-19-74/INFO ={"Address":"52:a5:7d:88:19:74","RSSI":-69,"ManufacturerData":"4c001005171cb30b17"}
3:50:22 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-71,"ManufacturerData":"4c0010050318c75348"}
3:52:20 PM Received BLEscan/56-fe-df-19-84-53=Active
3:52:21 PM Received BLEscan/56-fe-df-19-84-53/INFO ={"Address":"56:fe:df:19:84:53","RSSI":-69,"ManufacturerData":"4c001005171cb30b17"}
3:53:21 PM Received BLEscan/52-a5-7d-88-19-74=Inactive
4:00:21 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-77,"ManufacturerData":"4c0010050318c75348"}
4:00:22 PM Received BLEscan/56-fe-df-19-84-53/INFO ={"Address":"56:fe:df:19:84:53","RSSI":-61,"ManufacturerData":"4c001005171c7435a5"}
4:07:21 PM Received BLEscan/58-71-b0-72-bb-f8=Active
4:07:21 PM Received BLEscan/58-71-b0-72-bb-f8/INFO ={"Address":"58:71:b0:72:bb:f8","RSSI":-69,"ManufacturerData":"4c001005171c0b7e34"}
4:08:21 PM Received BLEscan/56-fe-df-19-84-53=Inactive
4:10:21 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-71,"ManufacturerData":"4c0010050318c75348"}
4:10:21 PM Received BLEscan/58-71-b0-72-bb-f8/INFO ={"Address":"58:71:b0:72:bb:f8","RSSI":-61,"ManufacturerData":"4c001005171c0b7e34"}
4:20:21 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-77,"ManufacturerData":"4c0010050318c75348"}
4:20:21 PM Received BLEscan/58-71-b0-72-bb-f8/INFO ={"Address":"58:71:b0:72:bb:f8","RSSI":-61,"ManufacturerData":"4c001005171c0b7e34"}
4:22:23 PM Received BLEscan/75-8f-e5-87-41-7b=Active
4:22:23 PM Received BLEscan/75-8f-e5-87-41-7b/INFO ={"Address":"75:8f:e5:87:41:7b","RSSI":-61,"ManufacturerData":"4c001005171c0b7e34"}
4:23:21 PM Received BLEscan/58-71-b0-72-bb-f8=Inactive
4:30:22 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-77,"ManufacturerData":"4c0010050318c75348"}
4:30:22 PM Received BLEscan/75-8f-e5-87-41-7b/INFO ={"Address":"75:8f:e5:87:41:7b","RSSI":-69,"ManufacturerData":"4c001005171cfe6dcb"}
4:37:21 PM Received BLEscan/6f-e5-82-e4-f1-12=Active

4:37:21 PM Received BLEscan/6f-e5-82-e4-f1-12/INFO ={"Address":"6f:e5:82:e4:f1:12","RSSI":-62,"ManufacturerData":"4c001005171c516cfa"}
4:38:21 PM Received BLEscan/75-8f-e5-87-41-7b=Inactive
4:40:21 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-70,"ManufacturerData":"4c0010050318c75348"}
4:40:22 PM Received BLEscan/6f-e5-82-e4-f1-12/INFO ={"Address":"6f:e5:82:e4:f1:12","RSSI":-61,"ManufacturerData":"4c001005171c516cfa"}
4:50:22 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-71,"ManufacturerData":"4c0010050318c75348"}
4:50:22 PM Received BLEscan/6f-e5-82-e4-f1-12/INFO ={"Address":"6f:e5:82:e4:f1:12","RSSI":-61,"ManufacturerData":"4c001005171c516cfa"}
4:52:21 PM Received BLEscan/40-19-dc-53-26-0c=Active
4:52:21 PM Received BLEscan/40-19-dc-53-26-0c/INFO ={"Address":"40:19:dc:53:26:0c","RSSI":-62,"ManufacturerData":"4c001005171ce12d8f"}
4:53:21 PM Received BLEscan/6f-e5-82-e4-f1-12=Inactive
5:00:23 PM Received BLEscan/40-19-dc-53-26-0c/INFO ={"Address":"40:19:dc:53:26:0c","RSSI":-70,"ManufacturerData":"4c001005171ce12d8f"}
5:00:23 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-70,"ManufacturerData":"4c0010050318c75348"}
5:07:22 PM Received BLEscan/53-9d-f9-14-4f-aa=Active
5:07:22 PM Received BLEscan/53-9d-f9-14-4f-aa/INFO ={"Address":"53:9d:f9:14:4f:aa","RSSI":-62,"ManufacturerData":"4c001005171ccc1b74"}
5:08:22 PM Received BLEscan/40-19-dc-53-26-0c=Inactive
5:10:22 PM Received BLEscan/53-9d-f9-14-4f-aa/INFO ={"Address":"53:9d:f9:14:4f:aa","RSSI":-69,"ManufacturerData":"4c001005171ccc1b74"}
5:10:22 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-71,"ManufacturerData":"4c0010050318c75348"}
5:20:22 PM Received BLEscan/53-9d-f9-14-4f-aa/INFO ={"Address":"53:9d:f9:14:4f:aa","RSSI":-69,"ManufacturerData":"4c001005171ccc1b74"}
5:20:22 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-70,"ManufacturerData":"4c0010050318c75348"}
5:22:22 PM Received BLEscan/67-e1-bd-c2-3d-33=Active
5:22:22 PM Received BLEscan/67-e1-bd-c2-3d-33/INFO ={"Address":"67:e1:bd:c2:3d:33","RSSI":-61,"ManufacturerData":"4c001005171cae1a3b"}
5:23:21 PM Received BLEscan/53-9d-f9-14-4f-aa=Inactive
5:30:21 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-70,"ManufacturerData":"4c0010050318c75348"}
5:30:22 PM Received BLEscan/67-e1-bd-c2-3d-33/INFO ={"Address":"67:e1:bd:c2:3d:33","RSSI":-61,"ManufacturerData":"4c001005171cae1a3b"}
5:37:22 PM Received BLEscan/4b-70-9d-4d-e9-5f=Active
5:37:22 PM Received BLEscan/4b-70-9d-4d-e9-5f/INFO ={"Address":"4b:70:9d:4d:e9:5f","RSSI":-62,"ManufacturerData":"4c001005171c97560c"}
5:38:21 PM Received BLEscan/67-e1-bd-c2-3d-33=Inactive
5:40:23 PM Received BLEscan/4b-70-9d-4d-e9-5f/INFO ={"Address":"4b:70:9d:4d:e9:5f","RSSI":-61,"ManufacturerData":"4c001005171c97560c"}
5:40:23 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-77,"ManufacturerData":"4c0010050318c75348"}
5:50:21 PM Received BLEscan/4b-70-9d-4d-e9-5f/INFO ={"Address":"4b:70:9d:4d:e9:5f","RSSI":-61,"ManufacturerData":"4c001005171c97560c"}
5:50:21 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-71,"ManufacturerData":"4c0010050318c75348"}
5:52:23 PM Received BLEscan/52-e0-87-14-a3-41=Active
5:52:24 PM Received BLEscan/52-e0-87-14-a3-41/INFO ={"Address":"52:e0:87:14:a3:41","RSSI":-69,"ManufacturerData":"4c001005171c21e7b7"}
5:53:21 PM Received BLEscan/4b-70-9d-4d-e9-5f=Inactive
6:00:21 PM Received BLEscan/52-e0-87-14-a3-41/INFO ={"Address":"52:e0:87:14:a3:41","RSSI":-68,"ManufacturerData":"4c001005171c21e7b7"}
6:00:21 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-70,"ManufacturerData":"4c0010050318c75348"}
6:00:21 PM Received BLEscan/40-16-3b-f0-70-df=Inactive
6:07:22 PM Received BLEscan/58-30-63-9c-aa-6f=Active
6:07:22 PM Received BLEscan/58-30-63-9c-aa-6f/INFO ={"Address":"58:30:63:9c:aa:6f","RSSI":-64,"ManufacturerData":"4c001005171cf2d5dc"}

6:08:22 PM Received BLEscan/52-e0-87-14-a3-41=Inactive
6:10:22 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-82,"ManufacturerData":"4c0010050318c75348"}
6:10:22 PM Received BLEscan/58-30-63-9c-aa-6f/INFO ={"Address":"58:30:63:9c:aa:6f","RSSI":-67,"ManufacturerData":"4c001005171cf2d5dc"}
6:20:22 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-74,"ManufacturerData":"4c0010050318c75348"}
6:20:22 PM Received BLEscan/58-30-63-9c-aa-6f/INFO ={"Address":"58:30:63:9c:aa:6f","RSSI":-62,"ManufacturerData":"4c001005171cf2d5dc"}
6:22:22 PM Received BLEscan/60-be-49-b5-9a-79=Active
6:22:22 PM Received BLEscan/60-be-49-b5-9a-79/INFO ={"Address":"60:be:49:b5:9a:79","RSSI":-65,"ManufacturerData":"4c001005171c56b008"}
6:23:22 PM Received BLEscan/58-30-63-9c-aa-6f=Inactive
6:30:22 PM Received BLEscan/53-c6-e0-13-ee-23/INFO ={"Address":"53:c6:e0:13:ee:23","RSSI":-83,"ManufacturerData":"4c0010050318c75348"}
6:30:22 PM Received BLEscan/60-be-49-b5-9a-79/INFO ={"Address":"60:be:49:b5:9a:79","RSSI":-62,"ManufacturerData":"4c001005171c56b008"}

```
/*
  Based on Neil Kolban example for IDF: https://github.com/nkolban/esp32-
snippets/blob/master/cpp_utils/tests/BLE%20Tests/SampleScan.cpp
  Ported to Arduino ESP32 by Evandro Copercini
*/
const char* ssid="Yours";
const char* password="Yours";
const char* broker="192.168.0.3";
const char* outTopic="BLEscan/3/";
const long RESCAN_TIME = 60000;  //milliseconds polling interval

#define SCAN_TIME  30 // seconds
#define STATUS_TIME 10 // polling loops per status report

// comment the follow line to disable serial message
#define SERIAL_PRINT

#include <Arduino.h>

// for BLE
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>

// for MQTT over WiFi
#include <WiFi.h>
#include <PubSubClient.h>

// for multiple core usage
TaskHandle_t BLEtask, WiFiTask;
SemaphoreHandle_t scanDone;

// data shared by two cores
BLEScanResults foundDevices;  // passed from BLE to WiFi
bool newScan = false; // assure only one wifi loop per scan

// data for Wifi/MQTT/Reporting
WiFiClient espClient;
PubSubClient client(espClient);
int deviceIndex = 0; // pointer into deviceList
char deviceList[50][18]; //list of discovered addresses
int deviceRSSI[50];      //list of RSSI of each address
bool deviceFound[50]; // marker for each pass to know if a device has left
```

```
bool deviceInactive[50]; // flag to indicate that a device inactive has been reported
bool statusReport = false;  // flag when to produce periodic JSON status of connected devices
long lastTime = 0;
int statusCount = 0;
int count = 0;

#ifdef SERIAL_PRINT
bool bBLEcoreReported = false;
bool bWiFiCoreReported = false;
#endif

void setup()
{
#ifdef SERIAL_PRINT
  Serial.begin(115200);
  Serial.println("ESP32 BLE Scanner");
#endif

  BLEDevice::init("");

  setupWiFi();
  client.setServer(broker,1883);

  // init arrays used to remember BLE devices observed
  for (int i = 0; i < 50; i++) {
    deviceInactive[i] = false;
    deviceFound[i] = false;
    deviceList[i][17]= '\0';
  }

  // mutex so reporting waits for scan to be done
  scanDone = xSemaphoreCreateMutex();

  // define the two tasks
  xTaskCreatePinnedToCore(
    &codeForBLE,
    "BLE",      // task name
    10000,      // stack size
    NULL,       // parameter
    1,          // priority
    &BLEtask,   // handle
    0);         // core

  delay(10000); // start-up BLE.  It will grab mutex
```

```
  xTaskCreatePinnedToCore(
    &codeForWiFi,
    "WiFi",      // task name
    10000,      // stack size
    NULL,       // parameter
    1,          // priority
    &WiFiTask,  // handle
    1);         // core
}


///////////// BLE ////////////////////////////////////////////

void codeForBLE(void * parameter) {
  for(;;) {
#ifdef SERIAL_PRINT
    if (!bBLEcoreReported) {
      Serial.print("BLE running on Core ");
      Serial.println(xPortGetCoreID());
      bBLEcoreReported = true;
    }
#endif
    //grab mutex when it becomes available
    xSemaphoreTake(scanDone,portMAX_DELAY);
#ifdef SERIAL_PRINT
    Serial.println("BLE start");
#endif

    // scan Bluetooth devices
    Scan();
    newScan = true;
    // release mutex
    xSemaphoreGive(scanDone);
#ifdef SERIAL_PRINT
    Serial.println("BLE done");
#endif
    delay(SCAN_TIME*1000);
  }
}

class MyAdvertisedDeviceCallbacks : public BLEAdvertisedDeviceCallbacks
{
    void onResult(BLEAdvertisedDevice advertisedDevice)
```

```
    {
#ifdef SERIAL_PRINT
      Serial.printf("Advertised Device: %s \n", advertisedDevice.toString().c_str());
#endif
    }
};

void Scan() {

  BLEScan *pBLEScan = BLEDevice::getScan(); //create new scan
  pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
  pBLEScan->setActiveScan(true); //active scan uses more power, but get results faster
  pBLEScan->setInterval(0x50);
  pBLEScan->setWindow(0x30);

#ifdef SERIAL_PRINT
  Serial.printf("Start BLE scan for %d seconds...\n", SCAN_TIME);
#endif
 //look for devices and  pass list of found devices to wifi/reporting task
  foundDevices = pBLEScan->start(SCAN_TIME);
}



///////////// WIFI ////////////////////////////////////////////

void codeForWiFi(void * parameter) {
  for(;;) {
#ifdef SERIAL_PRINT
    if (!bWiFiCoreReported) {
      Serial.print("WiFi running on Core ");
      Serial.println(xPortGetCoreID());
      bWiFiCoreReported = true;
    }
#endif
    xSemaphoreTake(scanDone,portMAX_DELAY);
#ifdef SERIAL_PRINT
    Serial.println("WiFi start");
#endif
    loopWiFi();
    newScan = false;
    xSemaphoreGive(scanDone);
#ifdef SERIAL_PRINT
    Serial.println("WiFi done");
```

```
#endif
  //keep polling around 60 second interval
   long currentTime = millis();
   long deltaTime = RESCAN_TIME - (currentTime - lastTime);
   lastTime = currentTime;
   if (deltaTime > 0) {
     delay(deltaTime);
   }
   else {
     delay(100);
   }
   // don't do eval unless new scan done
   while (!newScan) {
     delay(1000);
   }
  }
}

void setupWiFi(){
  delay(100);
  WiFi.begin(ssid,password);
  while(WiFi.status() != WL_CONNECTED){
    delay(100);
#ifdef SERIAL_PRINT
    Serial.print(".");
#endif
  }
#ifdef SERIAL_PRINT
  Serial.print("\nConnected to SSID ");
  Serial.println(ssid);
#endif
}

void reconnect(){
  while(!client.connected()){
    if(client.connect("BLEscan","","")){
#ifdef SERIAL_PRINT
      Serial.print("\nConnected to Broker ");
      Serial.println(broker);
#endif
    } else {
#ifdef SERIAL_PRINT
       Serial.print("\nTrying again to Broker ");
       Serial.println(broker);
```

```
#endif
      delay(5000);
    }
  }
}

// Assess if any new devices found and confirm existing devices still exist
void EvaluateActive() {
  count = foundDevices.getCount();
#ifdef SERIAL_PRINT
      Serial.print("EvaluateActive ");
      Serial.println(count);
#endif

  for (int i = 0; i < count; i++) {
    BLEAdvertisedDevice d = foundDevices.getDevice(i);
    std::string sAdd = d.getAddress().toString();
    char addr[18];
    char sTopic[40];
    char source[18];
    std::copy(sAdd.begin(), sAdd.end(), addr);
    addr[17] = '\0';
    strcpy(source,addr);
    addr[2]='-';
    addr[5]='-';
    addr[8]='-';
    addr[11]='-';
    addr[14]='-';
    bool bFound = false;
    bool bInActive = false;
    int Zone = -d.getRSSI()/5;
    int foundIndex;
    bool bMoved = false;
    for (int j = 0; j < deviceIndex; j++) {
      if( strcmp(deviceList[j], source) == 0) {
        bFound = true;
        foundIndex = j;
        deviceFound[j] = true;
        bInActive = deviceInactive[j];
        deviceInactive[j] = false;
        int RSSI = deviceRSSI[j];
        deviceRSSI[j]= d.getRSSI();
        bMoved = (abs(RSSI - deviceRSSI[j])> 10);
        break;
```

```
      }
    }
    if (!bFound || bInActive || bMoved) {
      if (!bFound){
        strcpy(deviceList[deviceIndex],source);
        deviceRSSI[deviceIndex] = d.getRSSI();
        deviceFound[deviceIndex] = true;
        deviceInactive[deviceIndex] = false;
        foundIndex = deviceIndex;
        deviceIndex++;
      }
      snprintf(sTopic,40,"%s%s",outTopic,addr);
#ifdef SERIAL_PRINT
      Serial.print(foundIndex);
      Serial.print(" Publish ");
      Serial.print(sTopic);
      Serial.print("=");
      Serial.println("Active");
#endif
      if (bMoved) {
        client.publish(sTopic,"Moved");
      }
      else {
        client.publish(sTopic,"Present");
      }

      if (statusReport || !bFound){
        char status[300];

snprintf(status,300,"{“Address”:”%s”,”RSSI”:%ld,”Zone”:%ld,”Status”:”Present”",deviceList[foundIndex],
deviceRSSI[foundIndex],Zone);

        if (d.haveName()){
          snprintf(status,300,"%s,”Name”:”%s”",status,d.getName());
        }

        if (d.haveAppearance()){
          snprintf(status,300,"%s,”Appearance”:”%ld”",status,d.getAppearance());
        }

        /* manf data causes inability to publish payload
        if (d.haveManufacturerData()){
          std::string md = d.getManufacturerData();
          uint8_t* mdp = (uint8_t*)d.getManufacturerData().data();
```

```
        char *pHex = BLEUtils::buildHexData(nullptr, mdp, md.length());
        snprintf(status,300,"%s,”ManufacturerData”:”%s”",status,pHex);
        free(pHex);
      }
      */

      if (d.haveServiceUUID()) {
        snprintf(status,300,"%s,”ServiceUUID”:”%s”",status,d.getServiceUUID().toString());
      }

      if (d.haveTXPower()){
        snprintf(status,300,"%s,”TxPower”:”%ld”",status,d.getTXPower());
      }

      snprintf(status,300,"%s}",status);
      snprintf(sTopic,40,"%s%s%s",outTopic,addr,"/INFO");
      client.publish(sTopic,status);
#ifdef SERIAL_PRINT
      Serial.print(foundIndex);
      Serial.print(" Publish ");
      Serial.print(sTopic);
      Serial.print("=");
      Serial.println(status);
#endif
    }
  }
 }
}

// Assess if any devices dropped off scan
void EvaluateInActive() {
 #ifdef SERIAL_PRINT
    Serial.print("EvaluateInActive ");
    Serial.println(deviceIndex);
#endif
 for (int i = 0; i < deviceIndex; i++){
  char addr[18];
  strcpy(addr,deviceList[i]);
  addr[2]='-';
  addr[5]='-';
  addr[8]='-';
  addr[11]='-';
  addr[14]='-';
  if (!deviceFound[i]) {
```

```
      if (!deviceInactive[i]) {
        char sTopic[40];
        snprintf(sTopic,40,"%s%s",outTopic,addr);
#ifdef SERIAL_PRINT
        Serial.print(i);
        Serial.print(" Publish ");
        Serial.print(sTopic);
        Serial.print("=");
        Serial.println("Abscent");
#endif
        client.publish(sTopic,"Abscent");
        deviceInactive[i] = true;
      }
    }
    if (deviceInactive[i]) {
      // provide info for devices previously discovered, but now missing
      // show last known zone and RSSI and abscent status
      if (statusReport) {
        char status[200];
        int Zone = -deviceRSSI[i]/5;

snprintf(status,200,"{“Address”:”%s”,”RSSI”:%ld,”Zone”:%ld,”Status”:”Abscent”}",deviceList[i],deviceRSS
I[i],Zone);
        char sTopic[40];
        snprintf(sTopic,40,"%s%s%s",outTopic,addr,"/INFO");
        client.publish(sTopic,status);
#ifdef SERIAL_PRINT
        Serial.print(i);
        Serial.print(" Publish ");
        Serial.print(sTopic);
        Serial.print("=");
        Serial.println(status);
#endif
      }
    }
  }
}


// assure connected to broker, evaluate new and missing devices, assess if periodic INFO report needed
void loopWiFi()
{
  // eval if time to report periodic status
  statusCount++;
```

```
  if (statusCount >= STATUS_TIME) {
    statusCount = 0;
    statusReport = true;
  }
  else {
    statusReport = false;
  }

  if (!client.connected()){
    reconnect();
  }
  client.loop();

  EvaluateActive();
  EvaluateInActive();

  //reset flag to be able to detect when a device is no longer present
  // flag will be set if device still esists in scan
  for (int i = 0; i < deviceIndex; i++){
      deviceFound[i] = false;
  }
}

// main loop required per IDE, but does nothing
void loop() {
  delay(10);
}
```

21.17 RFID

21.17.1 CheaperRFID

CheaperRFID is an active RFID transmitter and receiver pair that surfaced over ten years ago. The RF transmitter sends a four-character identification approximately every two seconds. The lack of a transmission indicates that the transmitter is no longer in range. A 9600 baud serial interface is used to transfer the reception information to a host processor. A space character is used to delimit transmissions. A model 9315 receiver was later released that added two additional bytes of information for signal strength.

A Wemos or other ESP8266 makes for a good host and MQTT used to provide ON/OFF presence indication. A picture of the CheaperRFID receiver with case and Wemos D1 mini is shown in Figure 325.

The connection between the ESP8266 and CheaperRFID is at the TX, RX and Gnd on DB9 connector pins 2,3 and 5 and the D1 mini on TX, RX and Gnd pins. It was not obvious to me if the CheaperRFID supported the RS-232 levels since it was powered with 12VDC. I used level converter to be safe but it may not have been necessary. Power to the D1-mini can be picked off the 12VDC used by the CheaperRFID and a regulator such as 7805 to provide 5VDC power to the module.

Figure 325 CheaperRFID Receiver with Wemos D1 Mini

The firmware update of Tasmota 6.4.1 is at [http://mcsSprinklers.com/mcsTasmota.zip file mcsTasmota641C.bin](http://mcsSprinklers.com/mcsTasmota.zip). It is 483K so should be able to be flashed and later OTA update done in a single step.

The Tasmota configuration required is for the Wifi credentials and MQTT information. No Module configuration is needed as no user-defined pins are used. From the console the following two commands are needed to establish the serial pins to be used by CheaperRFID rather than as an alternate console. These can be done by MQTT, Browser Console page, or serial port.

Bauderate 9600

Serialsend anything

The firmware report payload of ON or OFF based upon a transmitter reporting within a user-specified time. The topics are only transmitted in the change of ON (present) and OFF (not-present) states.

Provisions for 20 transmitters was made. These are retained in flash memory once discovered. All 20 locations can be cleared with the MQTT command **RFIDClear** (e.g. MQTT message RFID/RFIDClear topic with don't care payload). This can be done at the Browser console as well.

The default timeout period for a transmitter to be reported as OFF (not-present) is 10 seconds. This can be changed with the MQTT/Console command **RFIDtimeout** (e.g. Console command RFIDtimeout 20 to change timeout to 20 seconds).

Periodic state information is provided in the RFID group of the STATE message. For example, the message below with bolded content reflects three transmitter IDs (llog, 1234, and send) with timeout values of 0. The 0 indicates there are no longer present. These are just test IDs. They conform to the 4 characters used by CheaperRFID, but the number/text combination will be different.

10:16:47 MQT: RFID/STATE = {"Time":"2019-08-10T10:16:47","Uptime":"0T00:16:15","Vcc":3.048,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":6,"RSSI":92}}

10:16:47 MQT: RFID/SENSOR = {"Time":"2019-08-10T10:16:47",**RFID:{"EFGH":1},"RBTN3":1}}**

10:20:02 MQT: RFID/SENSOR = {"Time":"2019-08-10T10:20:02",**RFID:{"EFGH":0},"RBTN3":1}}**

21.17.2 Cheapest RFID

21.17.2.1 Conclusion

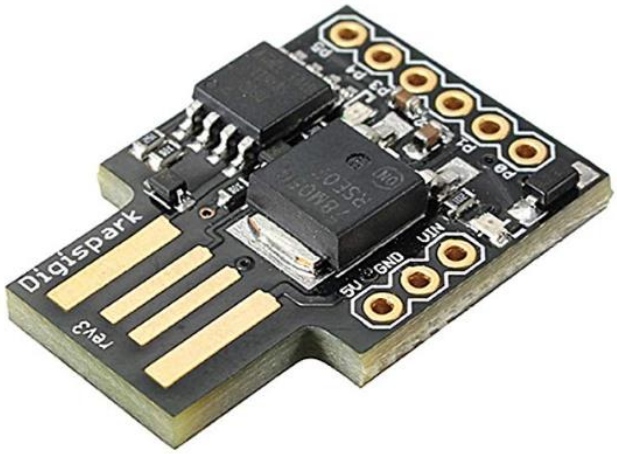
I was successful in getting working configurations of cheap 433 MHz transmitters and receivers with range of up to 150 ft. Total cost around $5 excluding power source. I evaluated both 5V and 12V transmitter power and I was surprised that the range was not improved that much with the higher voltage, but it was somewhat more consistent at reception with the higher voltage. The approach using generic transmitter and receiver provided a little better range than the approach using the RF from remote encoder/decoder, but at up to 50ft the remotes also worked.

I am not confident that then next set of generic 433 RF parts will perform the same as t he set I was using. To get a working configuration with the generic components I needed to use a transmitter from one source and a receiver from another. The paired units from the same source did not work together.

21.17.2.2 Analysis and Construction

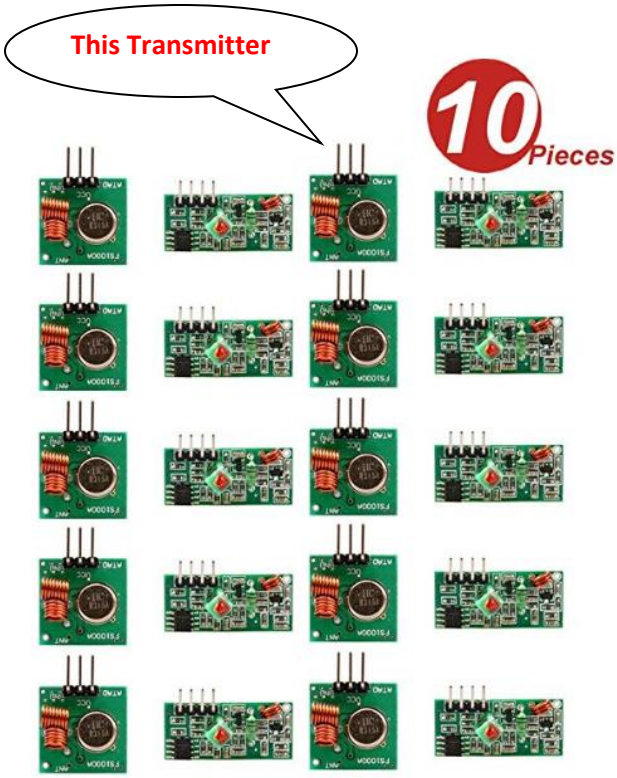
Two approaches have been used to modernize the Active RFID presence solution. One uses a generic 433Mhz transmitter along with an ATTiny85 microcontroller and a paired receiver. The ATTiny85 was obtained from Amazon at a cost of $2 each and used to provide the timing and pattern for the transmitter.

https://www.amazon.com/gp/product/B01MDUHSWO/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&psc=1



Three RF products were evaluated. The first from Amazon at $1.30/pair worked well as a transmitter with good range. The receiver was ineffective. I also tried to adjust the tuning coil and a variety of antenna, but none of the three I applied had acceptable result. To assist with the analysis to isolate transmit vs. receive issues a SDR https://www.amazon.com/NooElec-NESDR-SMArTee-Bundle-R820T2-Based/dp/B079C4S2BT/ref=sr_1_6?keywords=nooelec&qid=1565541880&s=hi&sr=1-6-catcorr was used with rtl_433 software from https://github.com/merbanan/rtl_433. This made it easy to see that the SDR was able to reliably receive the transmissions that were decoded to the same as the transmissions, thus isolating the problem to the 433 Mhz generic receiver from WINGONEER, assuming that the tuning coil adjustment is too course to get the transmitter frequency. The two transmitters I used were sending on 434.03 MHz and 434.06 MHz vs. the advertised frequency of 433.92.

https://www.amazon.com/gp/product/B07B9SD6LV/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1



WINGONEER 10Pcs 433MHz RF Wireless Transmitter and Receiver Module Kit for Arduino Raspberry Pi

by WINGONEER®

Be the first to review this item

Price: $12.99 ✓prime & FREE Returns

- 10 x Transmitter module (433MHz)
- 10 x Receiver module(433Mhz)
- To increase the reception range, consider using a copper wire to the XY FST solder / plug at the contact point ANT.
- These modules are used for projects when you need to communicate wirelessly between 2 devices.
- It is possible to increase the range by adding a small length of cable which the device will use as an aerial.

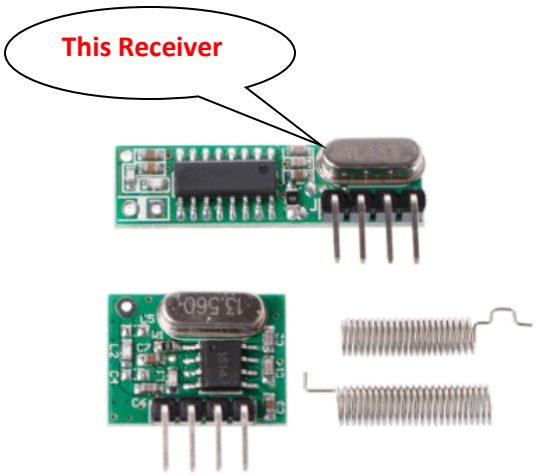
New (1) from $12.99 ✓prime FREE Shipping

Report incorrect product information.

fire tv **Shop on Fire TV, now with delivery tracking**
"Alexa, where's my stuff?" Learn more about FireTV

The second was obtained from Aliexpress at $1.30/pair. I continued to use the transmitter in the original evaluation and the receiver worked well. When I tried to use the transmitter from this second source I was unable to get even as much as a 433 MHz carrier in two different units so while the receiver was good, the transmitter was not functional in my testing. These units are WL101 Receiver and WL102 transmitter.

https://www.aliexpress.com/item/32840951211.html?spm=a2g0s.9042311.0.0.12c44c4dwg3h1k



1 Set 433Mhz RF Superheterodyne Receiver Transmitter Module Kit With Antenna Fo r Arduino/ARM/MCU

★★★★★ 4.9 386 Reviews 925 orders

US $0.98 ~~US $1.22~~ -20%

US $6.00 off on US $70.00 Get coupons

Quantity:

− 1 + 426 Sets available

Shipping: US $0.32 to United States via SunYou Economic Air Mail

Estimated Delivery on 09/24

Buy Now | Add to Cart | ♡ 713

The third was also obtained from Ebay for $2.30. For those in a rush then Amazon for just the receiver at a higher cost of $6. It is advertised to have the added feature or providing RSSI signal strength, but I was unable to get any data on how (or if) the RSSI is interfaced. I was also unable to get it to recognize

the transmitter from either of the two above sources. I was also unable to get it to receive the data from any of the transmitters I tried.

https://www.ebay.com/itm/RXB6-433Mhz-Superheterodyne-Wireless-Receiver-Module-for-Arduino-ARM-AVR-/201535806115

https://www.amazon.com/gp/product/B01NCQVFYE/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

Cylewet RXB6 433Mhz RF Superheterodyne Wireless Receiver Module for Arduino CYT1002

by Cylewet

2 customer reviews

Price: $5.99 prime & FREE Returns

Get $125 off: Pay $0.00 upon approval for the **Amazon Business Prime Card**. Terms apply.

- High Receiving Sensitivity: Up to -116dBm
- Good selectivity, capable of suppressing stray radiation
- It has a RSSI signal strength and cansimulatevoltage output
- Capable of suppressing local oscillator radiation, allowing a number of receiving modules to work together
- Temp Range: -40-85 degree centigrade, can work even in a harsh environment and temperature

New (1) from $5.99 prime FREE Shipping

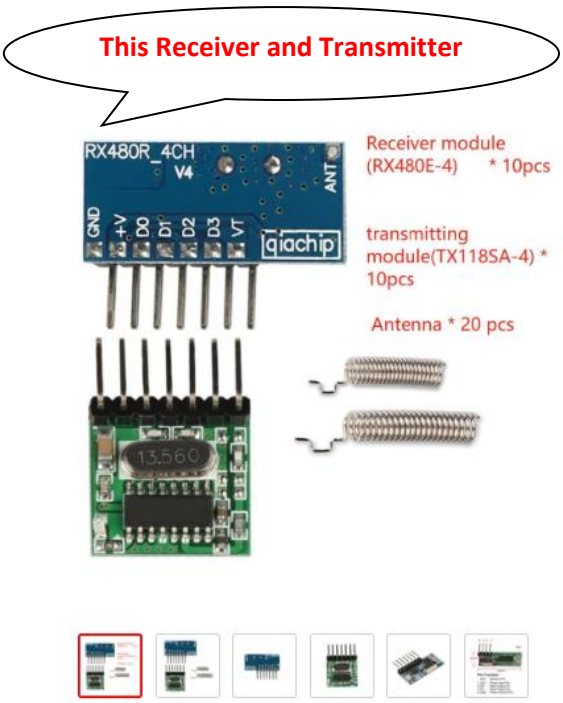
Report incorrect product information.

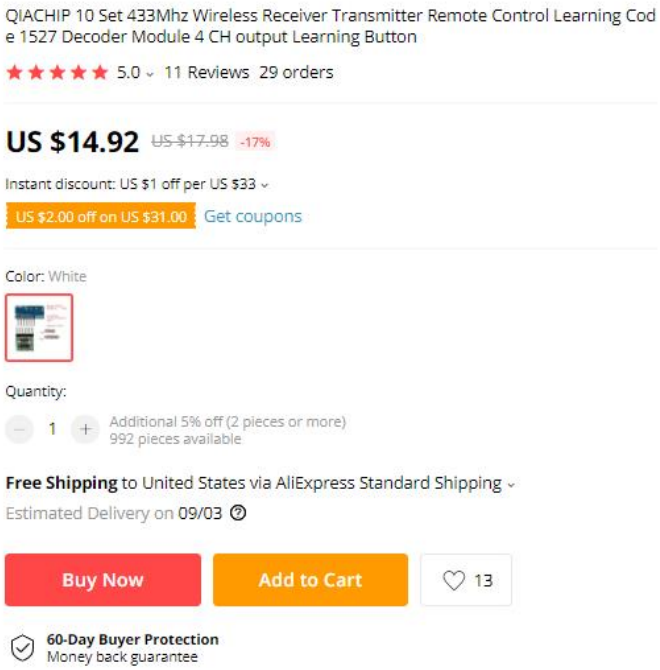
A good discussion on different low-cost RF products is at http://www.rflink.nl/blog2/wiring. It included improvements to reduce digital noise and add a ground plane for the antenna. These improvements made no difference in my case. This author's recommendation is the unit from Aurel Wireless that has an Italy source, but is not easily available in USA.

The second approach is also a 433 Mhz solution, but used the transmitter/receiver pair designed for use in four-button remotes. The ATTiny85 continued to be used to provide the timing duty cycle. A discrete timer such as 555 was also considered, but component count was higher and randomness to avoid collision could not be implemented easily with the discrete approach.

The transmitter/receiver pair used was $1.50/pair. These also provided good operation and like the generic 433 Mhz transmitter the voltage could be increased to 12 or 24 VDC to increase range. There was sufficient range to determine occupancy anywhere in or near the house using 5 VDC for the transmitter.

https://www.aliexpress.com/item/32998698554.html?spm=a2g0s.9042311.0.0.12c44c4dwg3h1k





The ATTiny85 and all the transmitters had about the same ¾ inch footprint so I used 2-sided tape to mount the two boards to each other and stitched wired the power, ground and data signal needed for the communications. See Figure 326 and Figure 327 for the construction.

If a higher voltage is to be provided to the transmitter than the 5 VDC used for the ATTiny85 then the power wiring will not be direct, but likely include a 7805 regulator to step down the higher voltage such as shown in Figure 328. Only one RF transmitter shown in the diagram will be connected for any given implementation. If 3 V to 5 V is used then all the 12V connections and 5V connections are combined and the 7805 is not used.

If the 7805 is to be added then it can also be piggybacked with 2-sided tape and short wiring connections stitch soldered. I did confirm that the 5 VDC provided by the ATTiny85 to be sufficient to control the waveform when the transmitter is connected to 12 VDC.

If a 5VDC rather than 12VDC source is to be used then the 7805 would not be used and 5VDC from the ATTiny85 wired directly to the transmitter. In this configuration it is possible to provide power from the USB connector on the ATTiny85 or it can be provided with direct wiring from a battery or other source.

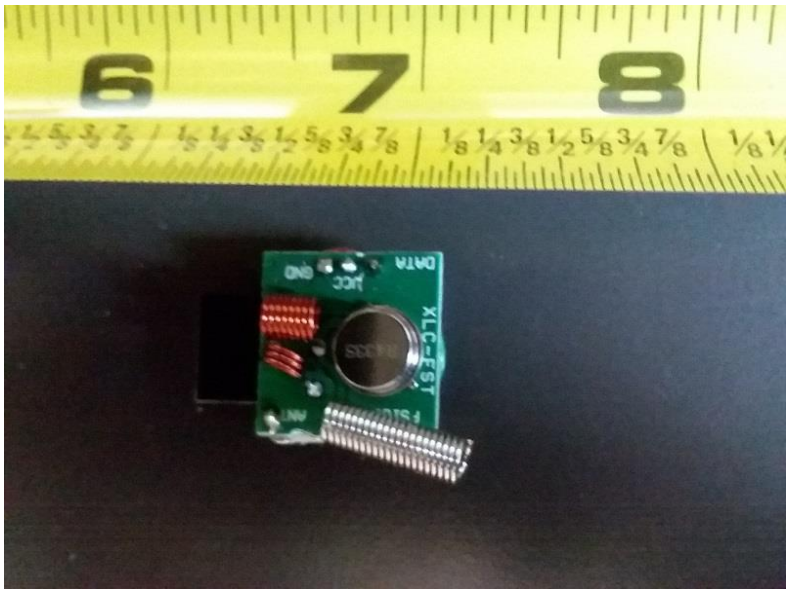


Figure 326 Generic 433 MHz RF Transmitter

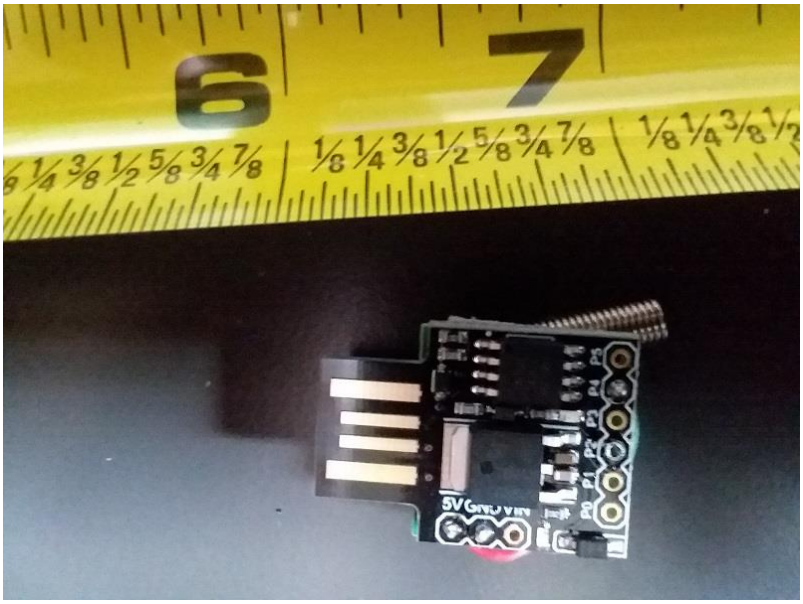


Figure 327 ATTiny85 Piggyback to RF Transmitter

The receiver was paired to Wemos D1 mini. I had previously purchased these, but they are similar to Amazon https://www.amazon.com/IZOKEE-NodeMcu-Internet-Development-Compatible/dp/B076F52NQD/ref=sr_1_2?crid=3GPEKA9BX2YQE&keywords=lolin+d1+mini&qid=1565466096&s=gateway&sprefix=lolin%2Caps%2C296&sr=8-2 at $3.60 each. Two GPIO pins were used for input. One from the generic 433 Mhz receiver and one from the RF Remote receiver. These are shown in wiring connections of Figure 329.

The Wemos D1 mini GPIO inputs expect voltages in the range of 0 to 3.3V while the RF Receiver provides a 0 to 5V range. A level shifter was used to mate these two levels. It could also have been done with a resistor voltage divider more easily.

The design supports both types of RF receivers being used together. Normally only one type of RF receiver is used.

The generic receiver can support many transmitters, each with unique RFID codes. The RF Remote can support 4 transmitters and each requires a separate GPIO input on the D1 Mini.

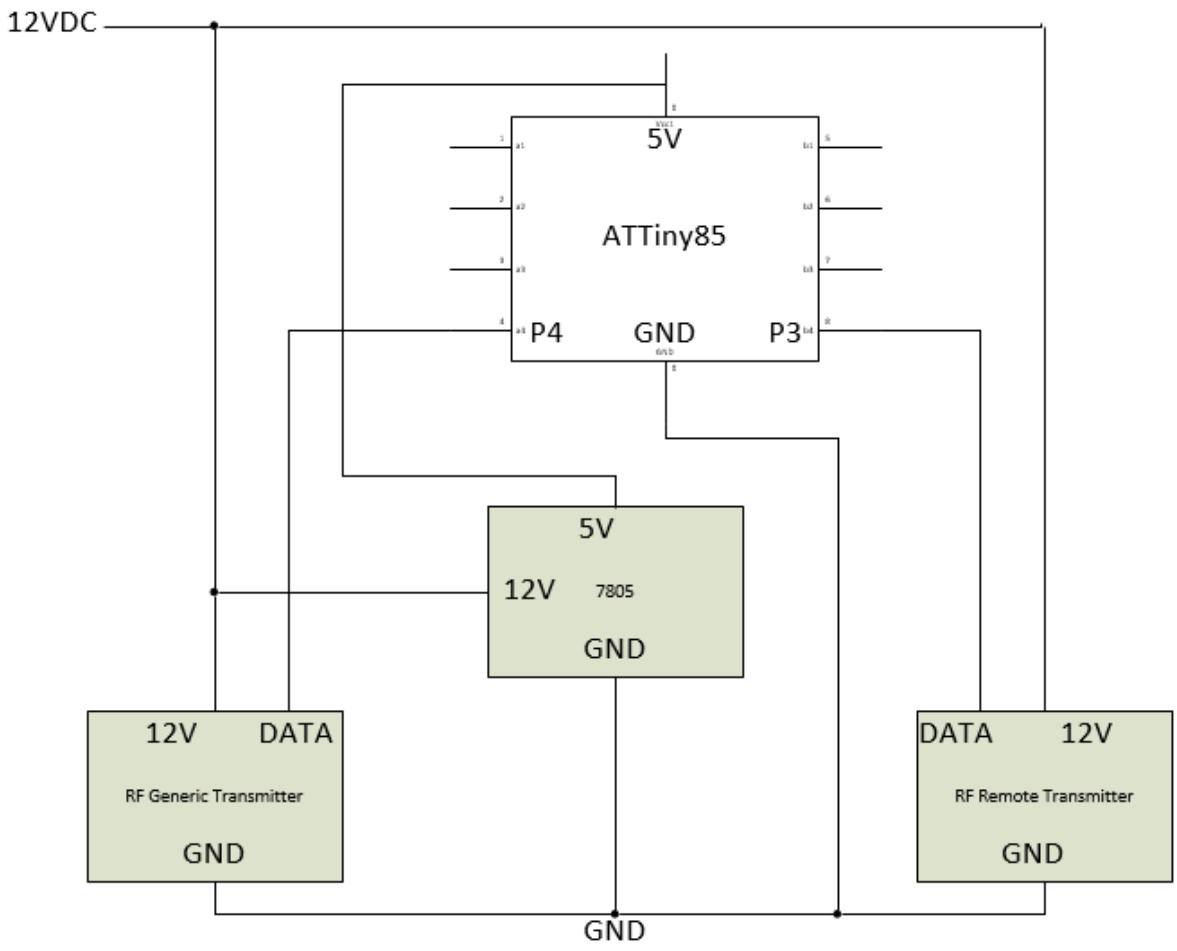


Figure 328 RFID Transmitter Connections

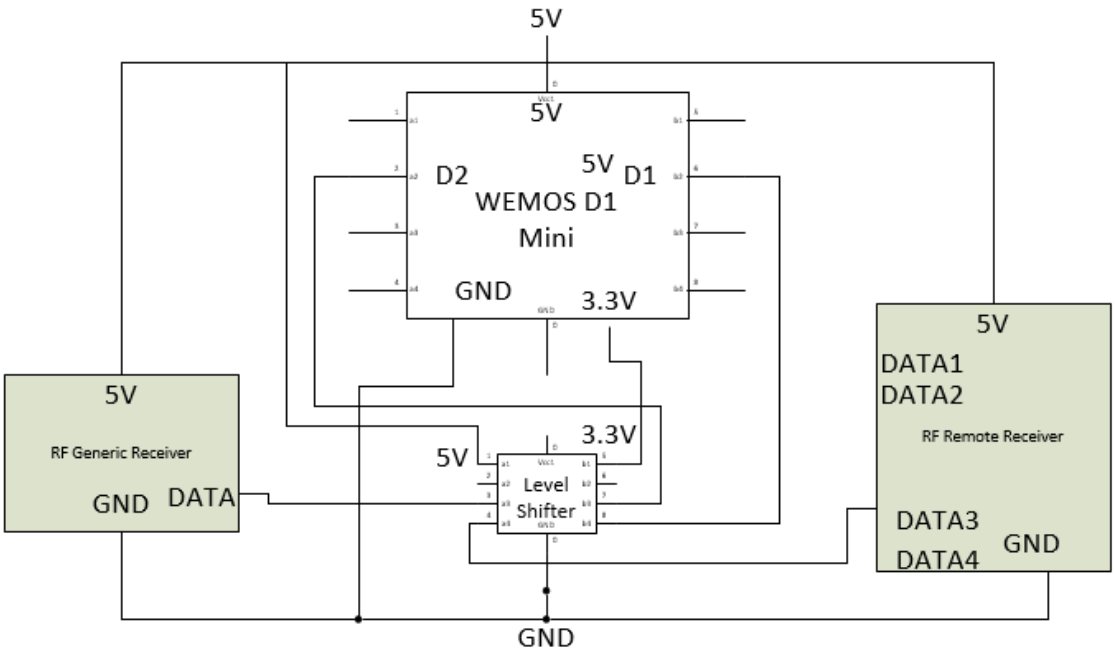


Figure 329 RFID Receiver Connections

Tasmota 6.4.1 was augmented to support both of the approaches and both can be used together if desired. The MQTT reporting is contained in the SENSOR subject using JSON key "RFID". The tag ID and a 0 vs. 1 payload are provided. The 1 indicates presence. For the RF Remote the convention of "RBTN" followed by number from 1 to 4 is used to identify the transmitter. For the generic, the tag id is encoded by ATTiny85 as part of its transmission.

```
15:23:31 MQT: RFID/SENSOR = {"Time":"2019-07-27T15:23:31",RFID:{"ABCD":0,"EFGH":0}}
15:23:31 MQT: RFID/SENSOR = {"Time":"2019-07-27T15:23:31",RFID:{"ABCD":1,"EFGH":0}}
```

10:16:47 MQT: RFID/SENSOR = {"Time":"2019-08-10T10:16:47",RFID:{"EFGH":1,"RBTN3":1}}
10:20:02 MQT: RFID/SENSOR = {"Time":"2019-08-10T10:20:02",RFID:{"EFGH":0,"RBTN3":1}}

The Tasmota configuration showing use of both approaches is presented in Figure 330 with the RFID type being for the generic receiver and the RFIDRemote1, RFIDRemote2, RFIDRemote3, and RFIDRemote4 for the remote receiver. The binary can be found in http://mcsSprinklers.com/mcsTasmota.zip mcsTasmota641CheapestRFID.bin

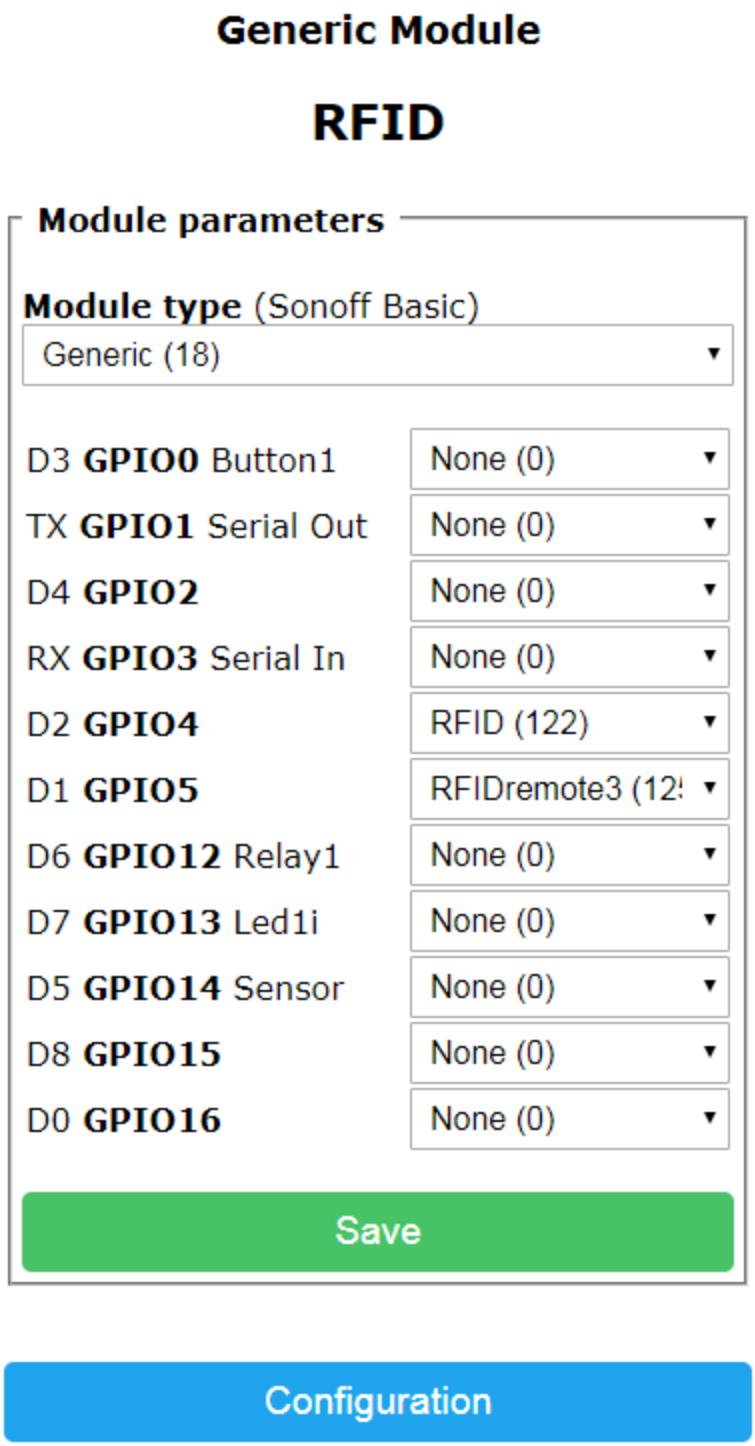


Figure 330 Configuration for Two RFID Receiver Options

The ATTiny85 obtained was installed on a development board that is a clone of the Digispark product. This makes programming the device much easier. Setup instructions for the Arduino environment for it are at http://digistump.com/wiki/digispark/tutorials/connectingpro.

The ATTiny85 was programmed to simulate a button push on a 5 second +/- 500 ms interval for 200 ms for the RFID remote. This resulted in the waveforms shown in Figure 331. These waveforms were taken from Universal Radio Hacker https://github.com/jopohl/urh that supports the SDR from NooElec.com https://www.amazon.com/NooElec-NESDR-SMArTee-Bundle-R820T2-Based/dp/B079C4S2BT/ref=sr_1_6?keywords=nooelec&qid=1565541880&s=hi&sr=1-6-catcorr

The 200 ms provided time for seven replications of the RF code. When this was reduced to 100 ms the receiver was no longer able to lock on and decode the signal. The periods selected provide a duty cycle of 200/5000 = 4% which should provide a good balance of power vs. detection latency.

During transmission on the transmitter, and following detection the receiver, a LED blinks on each which makes the operation each to see visually. The transmitter LED could be removed if further power reduction is desired then. It is a surface mount part.

Figure 331 1517 Learning Remote RF Pulse Pattern

The ATTiny85 was also programmed to use ASK to transmit a hard-coded four character sequence followed by a space character. The RadioHead library https://github.com/adafruit/RadioHead was used to simplify the encoding implementation. To change the four characters a new source compile is required. The sketch is shown below.

```
#include <RH_ASK.h>

    // This sketch transmits both ASCII and 1517 Remote.
    // The actual RF will be determined by the hardware connected
    // P3 connects to 1517 remote channel to active the "button"
```

```
    // P4 connects to generic transmitter that will send ASK
    // It is possible to connect both types of transmit hardware,
    // but intention is that only one will be used at a time

    // P4 is used to transmit.
    // No receiver connected to ATTiny85 so P2 not used,
    // but will be initialized as input
    // ATTiny, RX on D3 (pin 2 on attiny85)
    // TX on D4 (pin 3 on attiny85), 2000 bits/sec

    RH_ASK driver(2000, 2, 4, 0);
    uint8_t id[5];

void setup()
{
    pinMode(3,OUTPUT);
    digitalWrite(3,HIGH);
    driver.init();

    // define a unique ID.
    // In this case ABC followed by A to Z
    // and terminated with space
    // receiver will look for ABC as being unique
    // and the 4th to determine if it is missing transmissions
    id[0] = 65; //ABCx space
    id[1] = 66;
    id[2] = 67;
    id[3] = 65;
    id[4] = 32;
}

void loop()
{
    // Send Encoded ASCII via ASK modulation to Pin 4
    driver.send(id,5);
    driver.waitPacketSent();
    // Send 1517 Remote Code to Pin 3 for 200 milliseconds
    digitalWrite(3,LOW);
    delay(200);
    digitalWrite(3,HIGH);
    // Ramdomize the interval to minimize collisions
    // if multiple transmitters are used
    delay(4300+random(1000));
    // Update the 4th character of the ASCII
    // so missing receptions can be detected
    id[3]++;
    if (id[3] > 90) id[3]=65;
}
```

21.18 RF Transmitter via QIACHIP

The QIACHIP was used in the Cheapest RFID evaluation project described in Section 21.17.2. It provides a very simple way to transmit a RF code that can be recognized by devices that are able to respond to keyfobs.

An ESP8266, such as with Wemos D1 Mini, can directly control any of the four QIACHIP pins to transmit four unique RF codes. Two QIACHIPs can be used with the ESP8266 for ability to have eight channels of control with the ESP8266. Of course, other processors with more GPIO could extend it even further or multiple ESP8266 used as well.

Tasmota was loaded into the ESP8266 and configured as a Sonoff 4CH Pro. This provided an easy way for a UI that controls each of the four channels of RF. The Sonoff 4CH Pro GPIO then had to be aligned with the pins on the Wemos D1 Mini and with the pins on the QIACHIP. This relationship and the color or wires used during construction as shown in Table 9.

Table 9 QIACHIP to ESP8266 Wiring

Function	GPIO	Wemos Pin	QIACHIP Pin	Wire Color
Relay 1	12	D6	1	Green
Relay 2	5	D1	2	Yellow
Relay 3	4	D2	3	Blue
Relay 4	15	D8	4	Yellow
Button 1	0	D3	N/A	Green
Button 1 return	N/A	Gnd	-	Black
Button 2	9	N/A	N/A	N/A
Button 3	10	N/A	N/A	N/A
Button 4	14	D5	N/A	N/A
Power	N/A	5V	+	Red
Gnd	N/A	Gnd	-	Black

Tasmota was configured with poweronstate of 5 for each of the four channels. This provides a negative going pulse each time the channel is commanded. The pulse duration was found to work well at 0.5 seconds. The Tasmota command for this configuration is:

```
Backlog poweronstate 5; poweronstate2 5; poweronstate3 5;
poweronstate4 5; pulsetime 5; pulsetime2 5; pulsetime3 5;
pulsetime4 5
```

A 3D printer case was made with external tabs for mounting, a peephole for reset button, an opening for the microUSB power cable and a shelf to separate the WemosD1 mini from the QIACHIP. In addition, a small pushbutton was added via hot glue to provide local contorl of the first RF channel. This part of the project is shown in Figure 333.

The MQTT Topic was selected to be QIACHIP on the Tasmota Configuration – MQTT page. Status for each of the four channels will be steady state QIACHIP/POWER1 ON. The event of transmitting RF will be indicated by QIACHIP/POWER OFF followed 0.5 seconds later by QIACHIP/POWER1 ON.

When the association is made in mcsMQTT and edited to include the publish topic and change the VSP for OFF to have a button label of Send RF. Device Management in HS was then used to make the OFF/0 control only so only the Send button remains. The ON=1 state was deleted in HS Device Management. This is shown in Figure 332.

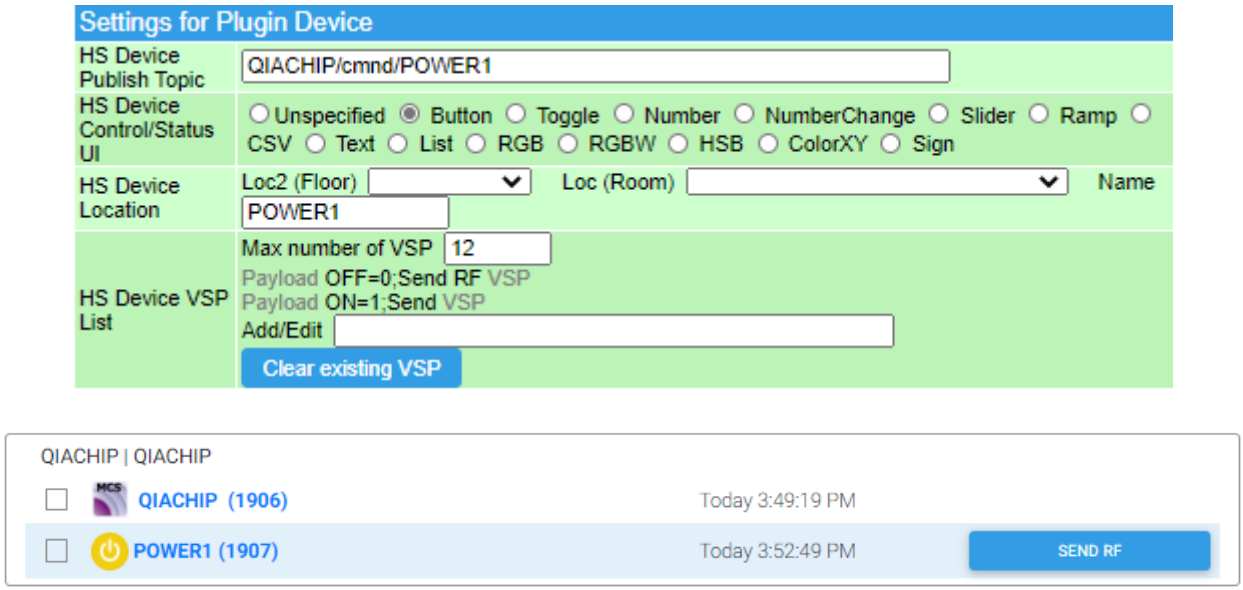


Figure 332 Setup of RF Command Channel in mcsMQTT and HS

Only one channel was setup, but a similar process is used for any of the other three if they are used.

When doing local control with the pushbutton one needs to develop the technique of doing short presses. If the button is held then two RF commands will be sent. One on the leading edge of the button press and one on the trailing edge. If the device receiving the RF, such as a Sonoff RF, does a toggle then a long button push will result in no change.

When viewing the decode QIACHIP transmission with Sonoff RF Bridge (or SDR/RTL_433) it can be seen similar to as shown below. In this case the Data 877C04 is the unique code for the one QIACHIP channel. Other channels will each have their own unique code.

17:03:14 MQT: SonoffRF/RESULT = {"Time":"2021-02-04T17:03:14","RfReceived":{"Sync":7730,"Low":250,"High":750,"Data":"**877C04**","RfKey":"None"}}

Figure 333 RF Transmitter Before and After Wiring

21.19 LED Matrix Sign

LED Matrices are commonly available in densities of 8x8 to 128x64 pixels with pitches ranging from 1 mm to 8 mm. Two technologies are available. One where the controller provides data lines to each row and each column of each of three LEDs and continuously scans to illuminate the pixel. There are commonly called RGB Matrix Panels or HUB75 Led Panels. The other is where the controller uses a single data line to address each pixel in a daisy chain manner and each pixel then maintains the color specified without further refresh. These are called Neopixels, WS2812 or similar devices.

This LED Matrix sign uses the Neopixel organized as tiles of 8x8 pixels at 8 mm pitch. They are available at https://www.aliexpress.com/item/10pcs-WS2812-LED-5050-RGB-8x8-64-LED-Matrix-for-Arduino/32658190926.html?spm=a2g0s.9042311.0.0.1d4b4c4dBPbFj0. The ten tiles were arranged to be two high and five wide thus giving a 40 pixel wide by 16 pixel high sign of dimensions 13 inches by 5.5 inches. Bigger or smaller signs can be constructed by using a different number or combination of the tiles. Each tile contains an input pin, an output pin, 5VDC pin and Ground pin. The 5VDC and Ground are bussed together and the output and input pins are daisy chained with the last output pin left unconnected.

During the construction there should be consideration for the power distribution. If 5VDC and Ground is daisy chained as was the data pins then there likely will be a perceptible voltage drop between the first and last tile and this will be reflected in a change of brightness. Bussing the power and connecting the power source in the middle of the bus will provide the best results. In my case I used Cat 5 (24 gauge) wire for all the connections.

For my initial construction I 3D printed a base with channels and an outside rim to contain the tiles. The wires were routed through the channels behind the tiles. Over the top I wrapped parchment paper to provide a diffuser and to hold the tiles from falling forward. This worked as a prototype, but suffered from the tiles not being positively attached on case thus giving a variation in the depth behind the parchment paper. It also lacked sufficient stability in keeping the tiles aligned should the sign be dropped.

For the final install double sided tape was used to attach the panels more positively. A simple piece of hardboard was used in favor of the 3D base. The tape's thickness provided sufficient clearance for the 24-gauge wire. A picture-frame style boarder was made from wood and taught tissue paper glued onto the edge to wood boarder used for the diffuser. Also attached to the edge of the frame is a light-dependent resistor to adjust the sign brightness to ambient room lighting and a switch for local on/off control.

21.19.1 LED Sign Construction

This sign was constructed from tiles containing 64 LEDs. Ten tiles were used in a two-row configuration. The ten tiles were temporarily aligned with transparent tape and then mounted on a 1/8" plywood strip using double sided tape. A wood frame was built and router used to rabbit the short sides to provide offset for mounting the t

Each tile has a digital input, digital output and two points for power and ground. The wiring strategy is to daisy-chain the digital in a consistent manner and provides a low resistance path for power with the power supply connection made in the middle of the power wiring. 24-gauge CAT5 wire was used for the

point to point solder connections. The soldering of the tiles was done after mounting on the plywood strip but before mounting the two strips in the frame.

After screwing the plywood strips onto the frame, the other electronic components were mounted. Two sensors were used. The photo-resistor was loosely placed inside a hole drilled on the side of the frame. I felt the side was a better place than the front since the front is subject to the immediate effect of the LEDs being illuminated. It was not permanently mounted to allow it to be moved in and out of the drill hole should adjustment be needed to get the correct ambient light levels.

The DH11 temperature and humidity sensor is not required, but since I had them available I included one with mounting using two-sided tape on the top inside of the frame. It is the light blue item with red, brown and black DuPont wires. I soldered the other side of these wires to the Wemos D1 mini which is visible on the plywood strip in the lower right of Figure 334. It was also attached with 2-sided tape, but screw holes are available for an alternate mount.

I used a 5V-3V voltage translator which is visible to the left of the D1 mini. It likely was not needed, but just provided a degree of robustness to match the Wemos digital output with the specs of the WS2812 LEDs. Its wiring consisted of 5V, Ground, 3.3V, D1 output of D1 mini to the 3.3V side and the 5V side went to the first LED tile in the daisy-chain. Figure 336 shows all the electrical connections.

A barrel connector was epoxied onto the frame to provide the 5VDC connection through an 18-gauge wire to the power supply.

The front of the sign is shown in Figure 335. I elected to diffuse the light from the LED to provide a muted display. This was accomplished with a polycarbonate cover over the front with orbital sander that converted the finish from clear to matte. If one is trying to achieve maximum brightness for applications in sunlight then no cover or a clear cover should be used.

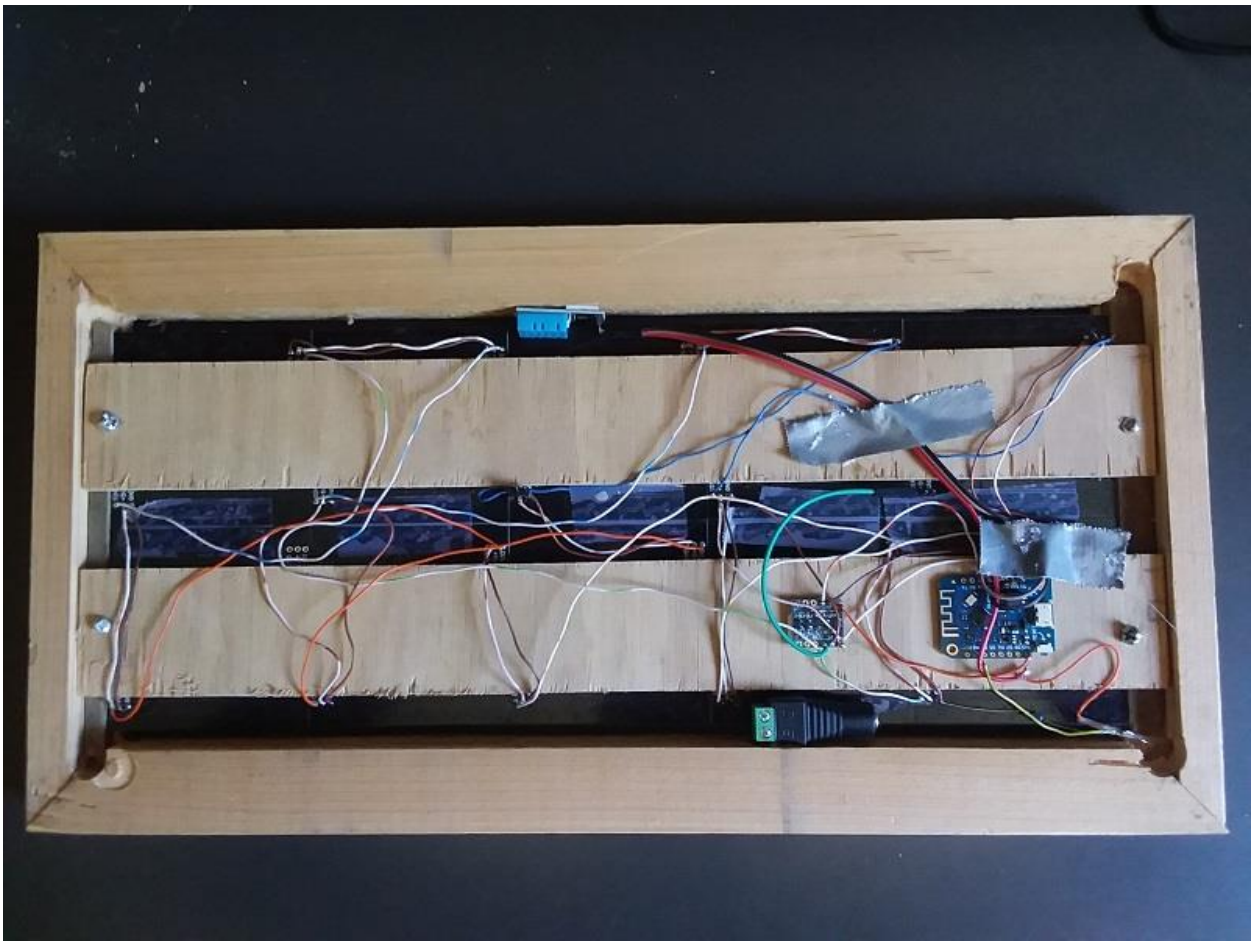


Figure 334 LED Sign Back

Figure 335 LED Sign with Diffuser Mounted

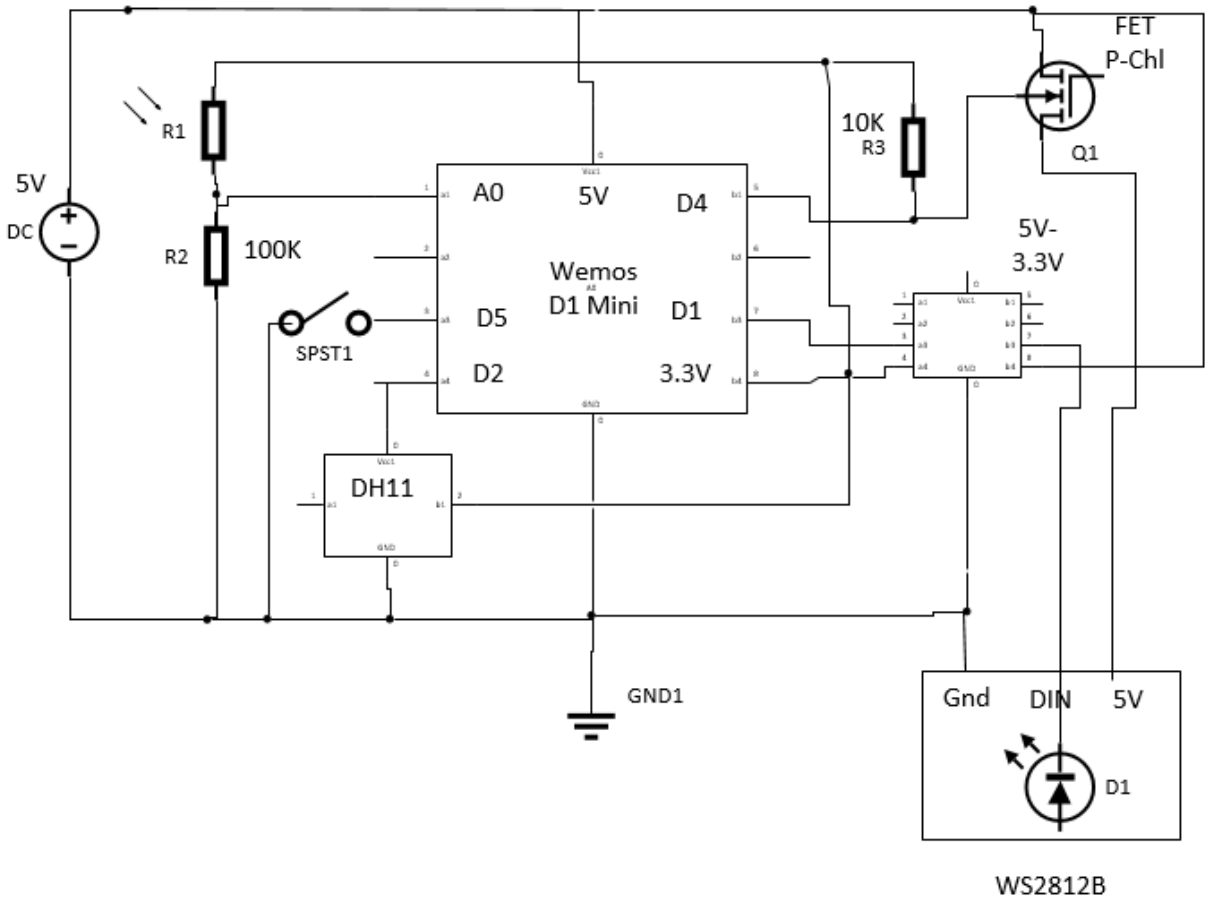


Figure 336 LED Sign Electrical Connections

21.19.2 Led Sign API Details

Table 10 shows the command topics that have been added to the standard list of Tasmota commands to support communication with the LED sign. In this example the MQTT topic of the Led sign is "LedSign" but could be changed if Tasmota Topic is configured differently.

The first twelve identify Text and the thirteenth identifies Image content to be shown. The next three affect how the text or image is displayed. The LedSign/cmnd/ClearSign topic is used to support management of the sign's retained messages. It allows removal of messages without the need to know the message id. If message id is known then a message with duration 0 will allow individual messages to be removed. The last is used to accept the API key provided by Open Weather Map that is necessary to download weather forecasts.

Table 10 LED Sign MQTT API

Topic	Payload	Description
LedSign/cmnd/PText1	JSON	Text message and properties for first non-volatile message buffer

LedSign/cmnd/PText2	JSON	Text message and properties for second non-volatile message buffer
LedSign/cmnd/PText3	JSON	Text message and properties for third non-volatile message buffer
LedSign/cmnd/PText4	JSON	Text message and properties for fourth non-volatile message buffer
LedSign/cmnd/Text1	JSON	Text message and properties for first volatile message buffer
LedSign/cmnd/Text2	JSON	Text message and properties for second volatile message buffer
LedSign/cmnd/Text3	JSON	Text message and properties for third volatile message buffer
LedSign/cmnd/Text4	JSON	Text message and properties for fourth volatile message buffer
LedSign/cmnd/Text5	JSON	Text message and properties for fifth volatile message buffer
LedSign/cmnd/Text6	JSON	Text message and properties for sixth volatile message buffer
LedSign/cmnd/Text7	JSON	Text message and properties for seventh volatile message buffer
LedSign/cmnd/Text8	JSON	Text message and properties for eighth volatile message buffer
LedSign/cmnd/Image	Binary	JPEG blocks of 120 bytes with header. See Table 12
LedSign/cmnd/Dwell	Seconds	Number of seconds a message appears until replaced by another message queued to be displayed. Default 10 seconds.
LedSign/cmnd/Scroll	Milliseconds	Number of milliseconds (with 50 ms resolution) to wait until text longer than screen size is moved left 1 pixel. Default 50 milliseconds.
LedSign/cmnd/Pan	Milliseconds	Number of milliseconds (with 50 ms resolution) to wait until image bigger than screen size is moved left/up/right/down 1 pixel. Default 150 milliseconds.
LedSign/cmnd/ClearSign	Not used	Clear all twelve of the message buffers
LedSign/cmnd/OWMKey	Ascii	Free API key obtained from Open Weather Map https://openweathermap.org/price used for forecast download when no other messages or images are shown on the last row of the display. An entry of less than 10 characters for the key will be treated as request to not download forecast.
LedSign/cmnd/TimeColor	RRGGBB	Color of HH:MM:SS displayed on first row of sign when no other message is being shown on first row

LedSign/cmnd/WeatherColor	RRGGBB	Color of weather forecast displayed on last row of sign when no other message is being shown on the last row

The LedSign/cmnd/Text and LedSign/cmnd/PText topics uses JSON payload to specify characteristics of the text that is being displayed. The JSON keys and their use are shown in Table 11.

Table 11 MQTT Text Topic JSON Payload Keys

JSON Key	Text/Number	Description
duration	Number optional	Number of seconds a message should be retained for display. After duration the message is no longer shown. 65535 is used to continue to display the message until manually removed. 0 is to no longer show the message.
row	Number optional	Text row of sign where message will be shown. Top row is 1.
color	Text optional	Six hex characters in RRGGBB format. The text will use this color unless succeeded by embedded color encoding contained in the “text” or “append” JSON keys.
text	Text optional	Text to be displayed on selected row. To change the text color on a character-by-character basis use [RRGGBB] prior to the character.
append	Text optional	Same as text key but will result in message being appended rather than replacing the text of message with the same topic. This is used to show messages greater than 120 characters. Provisions for persistent messages are 80 characters so not usually needed for these. Volatile messages can be up to 320 characters.

Table 12 MQTT Image Upload Payload Protocol

Byte	Field	Description
0	Data Type ASCII	1 = last record 2 = jpg encoding 3 = bitmap encoding (provisional)
1	Sequence Number ASCII	Index to identify which block of data is being transferred. First block is 0. Provisions exist for 10,000 bytes in RAM so maximum sequence number is (10,000/120-1 = 83)
2	Number of Bytes	Number of data bytes in payload with max of 120

	ASCII	
3	Data Binary	Up to 120 bytes of binary data comprising the image

The reporting API follows the Tasmota convention of acknowledging the commands received. It also reports a change in state of the TEXT, PTEXT and IMAGE topics. This normally when the content has been changed on a row including when the duration has expired and the text has been removed. The content of the payload is the row number that the content is shown with a 0 value for not showing. For example:

```
Topic: LedSign/TEXT1
Payload: 0
```

One special topic has been added that provides the width and height of the sign during startup. It consists of the sign topic /INFO4 with a JSON payload as shown below:

```
Topic: LedSign/INFO4
Payload: {"SignHeight":16,"SignWidth":40}
```

21.19.3 Led Sign Configuration

The hardware parameters of the LED sign are only configurable in source. User parameters are configurable via Tasmota console, serial, or MQTT messages.

Two configurations are described here. One is with an input button used to virtually toggle the screen on/off. One is with a relay output that drives a FET that physically removes/provides power to the LEDs. The FET should be used when the power supply is not sufficient to drive all the LEDs and ESP8266 during initial power up. The firmware will keep the LED power off until initialization is complete and normal message content has been setup. This assures that the startup power will not disturb the ESP8266.

Figure 337 and Figure 338 show the Module setup where GPIO5 is required to be used for the WS2812 LED matrix. I was not able to find a way to specify the WS2812 GPIO based upon RAM vs. constant memory.

Same constrains prevented for making the sign dimensions and orientation a run-time configuration. During startup an INFO4 MQTT message will be published that shows the sign height and width configured in the build.

The following is used in the source to specify the hardware parameters:

```
#define NEO_MATRIX_TOP         0x00 // Pixel 0 is at top of matrix
#define NEO_MATRIX_BOTTOM      0x01 // Pixel 0 is at bottom of matrix
#define NEO_MATRIX_LEFT        0x00 // Pixel 0 is at left of matrix
#define NEO_MATRIX_RIGHT       0x02 // Pixel 0 is at right of matrix
#define NEO_MATRIX_CORNER      0x03 // Bitmask for pixel 0 matrix corner
#define NEO_MATRIX_ROWS        0x00 // Matrix is row major (horizontal)
#define NEO_MATRIX_COLUMNS     0x04 // Matrix is column major (vertical)
#define NEO_MATRIX_AXIS        0x04 // Bitmask for row/column layout
#define NEO_MATRIX_PROGRESSIVE 0x00 // Same pixel order across each line
#define NEO_MATRIX_ZIGZAG      0x08 // Pixel order reverses between lines
#define NEO_MATRIX_SEQUENCE    0x08 // Bitmask for pixel line order
#define NEO_TILE_TOP           0x00 // First tile is at top of matrix
#define NEO_TILE_BOTTOM        0x10 // First tile is at bottom of matrix
#define NEO_TILE_LEFT          0x00 // First tile is at left of matrix
#define NEO_TILE_RIGHT         0x20 // First tile is at right of matrix
#define NEO_TILE_CORNER        0x30 // Bitmask for first tile corner
```

```
#define NEO_TILE_ROWS          0x00 // Tiles ordered in rows
#define NEO_TILE_COLUMNS       0x40 // Tiles ordered in columns
#define NEO_TILE_AXIS          0x40 // Bitmask for tile H/V orientationpaintp
#define NEO_TILE_PROGRESSIVE   0x00 // Same tile order across each line
#define NEO_TILE_ZIGZAG        0x80 // Tile order reverses between lines
#define NEO_TILE_SEQUENCE      0x80 // Bitmask for tile line order
#define NEOPIXEL_PIN 5              // GPIO 5 Wemos D1 (ESP32 pin 8) for Neopixel Sign
#define MATRIX_WIDTH 8              // 8 x 8 pixel matrix
#define MATRIX_HEIGHT 8
#define LED_SIGN_WIDTH 40           // total number of horizontal pixels on sign
#define LED_SIGN_ORIENTATION NEO_MATRIX_BOTTOM + NEO_MATRIX_RIGHT + NEO_MATRIX_ROWS +
NEO_MATRIX_PROGRESSIVE + NEO_TILE_TOP + NEO_TILE_ROWS + NEO_TILE_LEFT +
NEO_TILE_PROGRESSIVE
#define LED_SIGN_HEIGHT 16          // total number of vertical pixel on sign

FastLED_NeoMatrix *matrix = new FastLED_NeoMatrix(leds, MATRIX_WIDTH, MATRIX_HEIGHT,
LED_SIGN_WIDTH/MATRIX_WIDTH, LED_SIGN_HEIGHT/MATRIX_HEIGHT, LED_SIGN_ORIENTATION);
```

This setup corresponds to the upper left tile input going to ESP8266 GPIO 5 (D1). The output of this tile to the input of the next tile to the right. The last tile's output on this row goes to the input of the left-most tile on the next row.

If the DHT11 is used for local temperature and humidity then it can be on any available GPIO. D2 was selected for this setup in Figure 337. Any available pin can be selected for wiring the DH11T.

If the weather forecast is to be downloaded by the sign then specify location with the Tasmota commands:

Latitude xxxxx

Longitude xxxxx

Also required for forecast download will be the following new command with the API key obtained from Open Weather Map replacing the xxxxx.

OWMKey xxxxx

Another option is a button to toggle the LED Sign on and off. The on/off control can also be done from the main Tasmota browser page as shown in Figure 339, MQTT POWER command (e.g. LedSign/cmnd/Power 1) or if the Wemo emulation is setup on the Other Tasmota page then it can be done with Amazon Alexa such as "Turn Sign On" if "Sign" was setup as the friendly name on the Configure Other browser page. The Tasmota command "**poweronstate 3**" should be applied on console or other means to retain the on/off status after power cycle.

The main Tasmota browser page provides a dimmer control as well as the readouts from LDR (Analog0), and DH11T. The dimmer is set such that midpoint (50%) will provide nominal brightness if full light conditions provide an LDR reading of 1024. If the install location provides less background light then the dimmer can be increased. For example, in Figure 339 the LDR is reading 674 so it is causing the LED brightness to be reduced by 674/1024 = 65%. To increase the LED to full intensity at this background light level then the dimmer would be set to 77% (100%/65% /2). Setting it to higher values will not increase full brightness, but will increase the brightness in lower ambient light conditions.

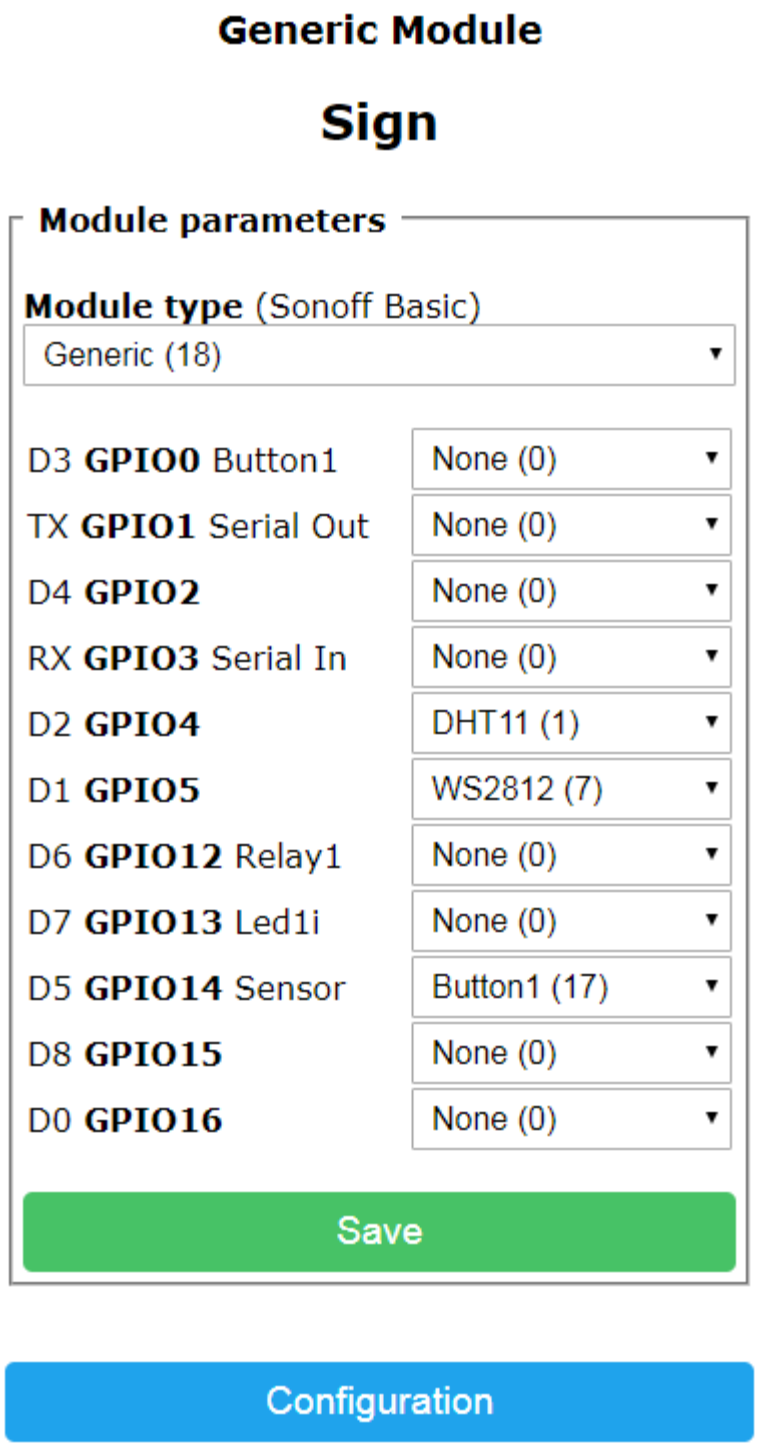


Figure 337 LED Sign Module Setup

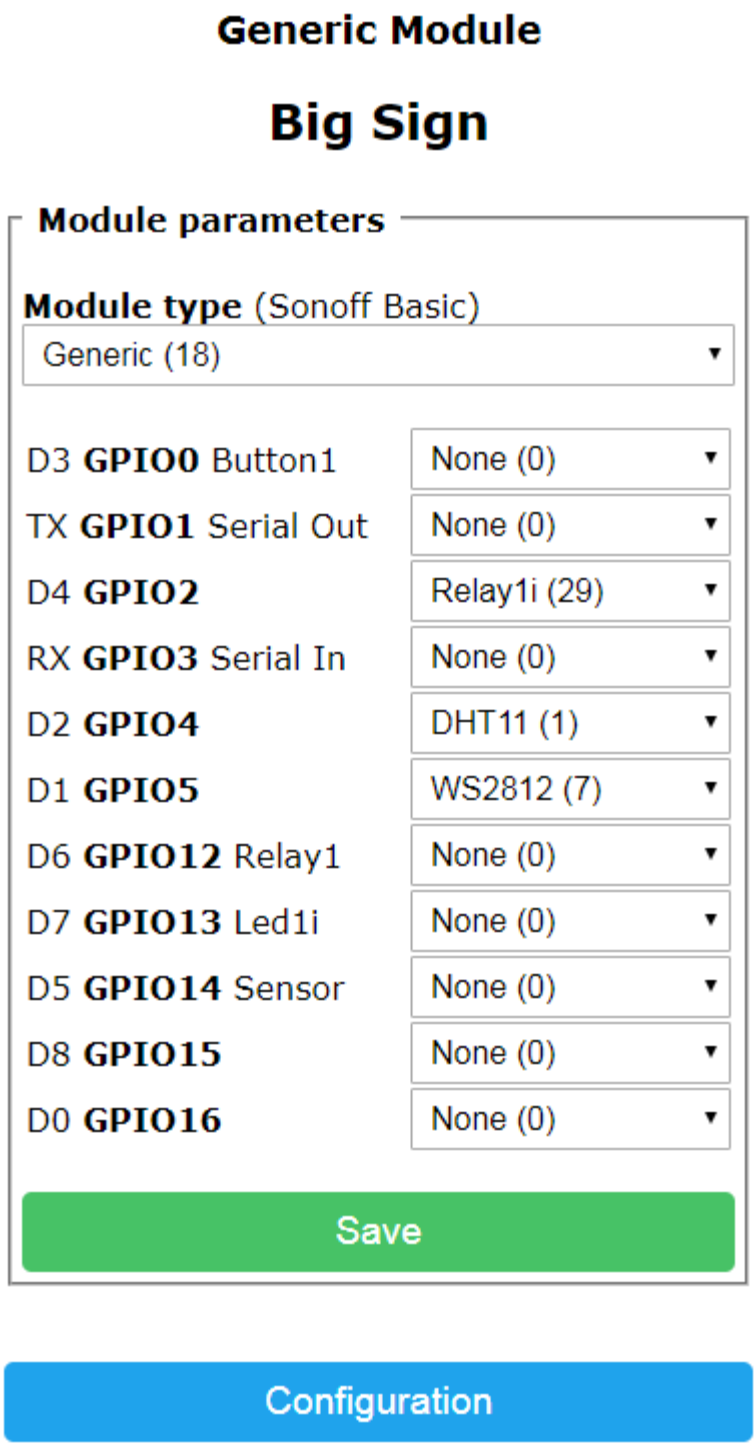


Figure 338 Sign Configuration with FET to Control LED Power

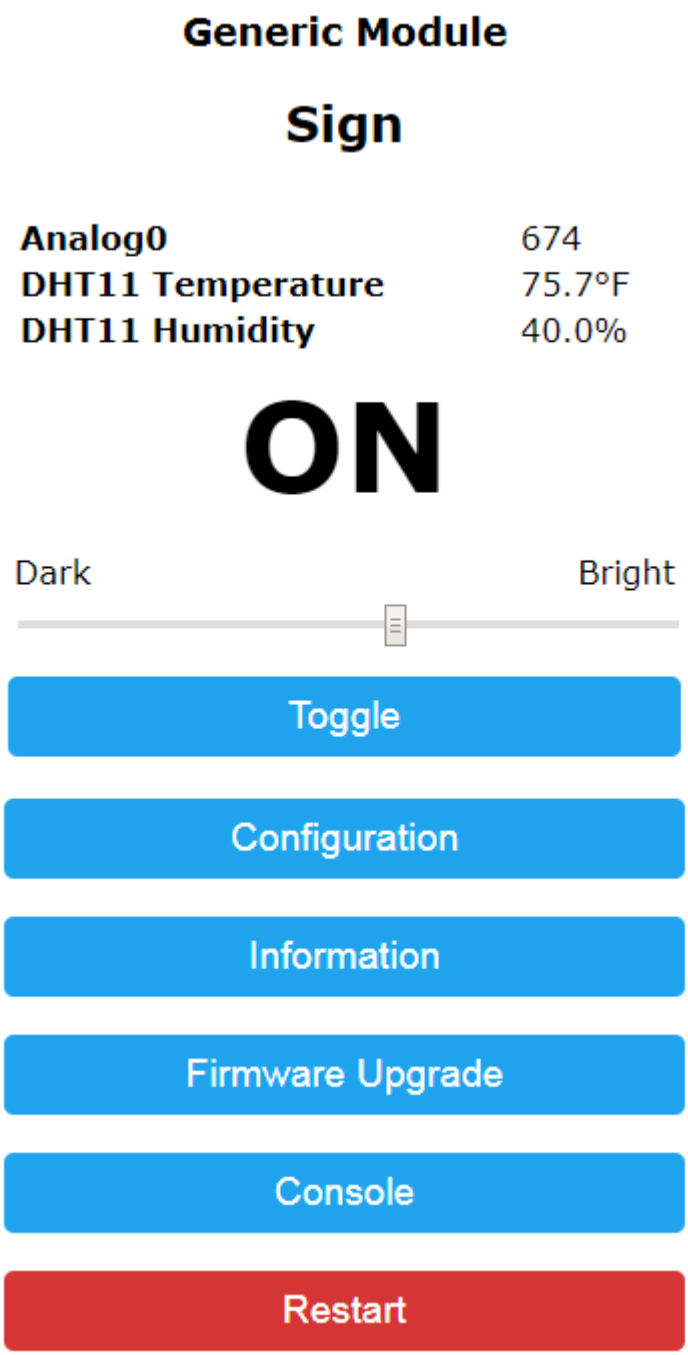


Figure 339 LED Sign Main Page

Other Tasmota pages can be configured to specify the MQTT, Logging and other parameters. There are no constraints imposed by the sign on these other parameters.

The binary image can be found at http://mcsSprinklers.com/mcsTasmota.zip with file mcsTasmota641Sign.bin. It is a 533 MB file so cannot be directly loaded after the initial flash. A process

of first loading mcsTasmotaMinimal.bin (450K) following by loading mcsTasmota641Sign.bin (533K) is needed for the OTA update.

The following parameters are given for the binary. The JPEG libraries drive the PROGRAM size beyond the ½ Meg boundary. The DATA is bumping up on the max allowed for RAM due to the 11,000 bytes reserved for the image buffer that is also used for the forecast download. If other Tasmota features are selected that make heavy use of RAM then the source would need recompilation after reducing the size of the image/forecast buffer provision.

DATA: [========] 78.4% (used 64212 bytes from 81920 bytes)

PROGRAM: [=====] 52.8% (used 540512 bytes from 1023984 bytes)

If other hardware configurations are desired then it is easy to change the source parameters and other binaries can be provided. For those who are able to compile from source the VSCode/Platform environment with source files is at http://mcsSprinklers.com/mcsTasmotaRFIDSource.zip.

21.19.4 LED Sign Usage

The usage concept of the sign is that it can show an image covering the entire sign or can show independent rows of colored text when not being used for an image. The bottom row of the sign, when no directed messages are showing, is used to display the current temperature, humidity and textual description of weather forecast over the next five days.

Upon power up the sign will show its pixel dimensions in an INFO4 message and will show the row on which each of the image and text buffers is currently being shown. Only the persistent text messages will show a non-zero row on power up. The update will be sent whenever the status of the message changes. The following is a typical power up status provided by the sign.

```
11:58:53 AM LedSign/INFO4={"SignHeight":16,"SignWidth":40}
11:58:53 AM LedSign/IMAGE=0
11:58:53 AM LedSign/PTEXT1=1
11:58:53 AM LedSign/PTEXT2=0
11:58:53 AM LedSign/PTEXT3=0
11:58:53 AM LedSign/PTEXT4=0
11:58:53 AM LedSign/TEXT1=0
11:58:53 AM LedSign/TEXT2=0
11:58:53 AM LedSign/TEXT3=0
11:58:53 AM LedSign/TEXT4=0
11:58:53 AM LedSign/TEXT5=0
11:58:53 AM LedSign/TEXT6=0
11:58:53 AM LedSign/TEXT7=0
11:58:53 AM LedSign/TEXT8=0
```

Text Display

Figure 340 shows an example of two rows of text being displayed. The first row is a short message that is statically displayed. The second row is a scrolling message the contains the five-day weather forecast that is downloaded by the sign using the latitude and longitude entered in Tasmota.

Figure 340 LED Sign Text

When the entire length of the message can be shown on its row then it will be shown statically. If it exceeds the sign width then the message will scroll left pixel-by-pixel in a marques style.

A message has properties of row, color and duration.

A message will continue to be shown for the specified duration. After the duration has expired it will be removed. A duration of 0 causes a message to be removed from flash or RAM. A duration of 65535 causes a message to never be automatically removed.

If multiple messages are being shown on a given row then the messages will be selected in round robin fashion and each displayed in sequence. The dwell time of a given message before it is swapped out for the next round-robin message is not a property of the message, but a global parameter than can be specified by MQTT command.

The scroll speed of longer messages is also not a message property, but a global parameter set by MQTT command.

Up to twelve messages can be stored in the sign. Four are stored in flash and eight in RAM. Those stored in flash will persist a power cycle. The Topic command PTEXT vs. TEXT is used to designate the location with PTEXT being stored in flash.

A message in flash can be up to 80 characters in length. Those in RAM can be 320 characters. These character counts include the characters used to override the message color.

“color” is a property in the payload of the message or color can be embedded in the text/append of the message using syntax [RRGGBB]. The override gives the ability to change the color on a character-by-

character basis. Bright colors should be selected because the sign senses ambient light and dim colors will then become black in lower lighting levels.

Kerning can be used to change the normal six-pixel per character text display with use of the non-printable bytes 01 and 02 where 01 is to move cursor left one pixel and 02 is to move cursor right one pixel. For example, a period normally has two black pixels, an illuminate pixel, and then three black pixels. To remove the left two black pixels which effectively makes the period only four pixels wide the following bytes would be used…01 01 2E which in VB is chr(1) & chr(1) & ".".

The byte 03 is a square of three pixels by three pixels. It is used as a degree symbol as the standard font library does not contain the degree symbol.

MQTT payloads have a 128-character limit. Two provisions exist to allow properties and text to span multiple MQTT messages. Each or multiple properties can be included in an JSON payload of a given topic with those not included retaining their value from prior messages for the same message id.

The second is when the text itself exceeds the payload message length. In this case the JSON key "append" rather than "text" is used to cause the text to be appended rather than replace the text of the message.

When no message is being requested in the last row then weather information is shown in that row. It consists of the local temperature and humidity provided by the DH11T sensor and the forecast obtained from Open Weather Map for the Latitude and Longitude that have been configured in Tasmota. This provided requires an API key which is entered via MQTT command.

Some examples of MQTT messages are illustrated below. In this case the based topic of the sign is "LedSign".

Use the sign for text and put message "Long Message" on row 1 with color A01020 for 3 minutes. This message will scroll because of the length of message exceeding 6 characters (40 pixels / 6 pixels/character).

```
Topic: LedSign/cmnd/Text1
Payload: {"duration":3,"color":"A01020",”row”:1,”text”:”Long Message”}
```

Use the sign for text and put message "Short" on row 0 with colors changing for the first five characters. Show it for 4 minutes. This message will be statically displayed without scrolling.

```
Topic: LedSign/cmnd/Text2
Payload:
{”duration”:4,”row”:1,”text”:”<FF0000>S<00FF00>h<0000FF>o<FFFF00>r<00FFFF>t”}
```

Extend the "Long Message" text to include additional text in red.

```
Topic: LedSign/cmnd/Text1
Payload: {”append”:”<FF0000>Appended to Message in Red”}
```

Put a message in the fourth of the sign's flash buffers that will continue to be displayed after power cycles in green on second row until manually removed.

```
Topic: LedSign/cmnd/PText4
Payload: {"duration":65535,"color":"00FF00",”row”:1,”text”:”Permanent”}
```

Remove the fourth persistent message.

```
Topic: LedSign/cmnd/PText4
```

```
Payload: {"duration":0}
```

Remove all stored messages. This will not remove the local weather message.

```
Topic: LedSign/cmnd/ClearSign
Payload: does not matter
```

```
HS Event to toggle row 2 display between inside temperature/humidity and
outside temperature1/greenhouse temperature.  Event runs every three minutes
with a 2 minute duration of the outside reading.  After two minutes the
default inside temperature and humidty is restored.

Action Send MQTT Message
LedSign/cmnd/TEXT6={"row":2,"duration":2,"color":"00FFFF","text":"<< CHAR(2)
& SROUND($$DVR:(587):,0) & CHAR(3) & CHAR(2) & CHAR(2) & CHAR(2) & CHAR(2) &
SROUND($$DVR:(588):,0) & CHAR(3) >>"}
```

Image Display

Figure 341 shows a jpeg image of a photograph that is being panned left and right to be able to see the full image. In general, images up to twice the size of the sign provide good results. Larger images tend to be difficult to comprehend because of the relatively small viewport being provided at any given instant. If the image resolution is reduced too much before downloading then it may be hard to visualize because of the low resolution provided by the smaller sign.

Figure 341 LED Sign Image

JPEG images can be downloaded to the sign. Provisions also exist for bitmap images, but this provision has not been implemented. JPEG was chosen because the critical resource on the ESP8266 is RAM and JPEG minimizes the RAM utilization. 10,000 bytes of RAM are allocated for images so no more than a 10,000-byte jpeg image can be displayed. For signs with more than two rows the RAM allocation for jpg has been reduced to 9,000 bytes to provide more RAM for the larger LED buffers.

If the image height and width can be shown in the available pixels of the sign then the image will be statically displayed. If it exceeds one or more dimensions then it will be panned up/down/left/right to show the entire image. The pan rate is a MQTT parameter.

The image will continue to be displayed until a Text topic is received.

An example of requesting an image display is accomplished by sending multiple messages of the same topic with the payload conforming to the Table 12 protocol. This protocol consists of three bytes of header and then up to 120 bytes of binary data. The last of the messages has the payload's first byte set to ASCII 1.

```
Topic: LedSign/cmnd/Image
Payload: <per Table 12 protocol>
```

21.20 Greenhouse Sensor and Control

A small greenhouse was constructed to assist with springtime seed germination and off-season vegitable growth. The structure consists of an exhaust fan with spring-loaded baffle, a salvaged space heater core, a pair of DS18B20 temperature sensor, a capacitive moisture sensor and three microcontrollers.

Figure 342 Greenhouse Structure

The primary environmental control is provided by Sonoff 4CH Pro with stock Tasmota 8.5. This provides relay control of the exhaust fan for cooling and each of two 750 watt elements of the space heater core based upon temperature measurement of the DS18B20 temperature sensor.

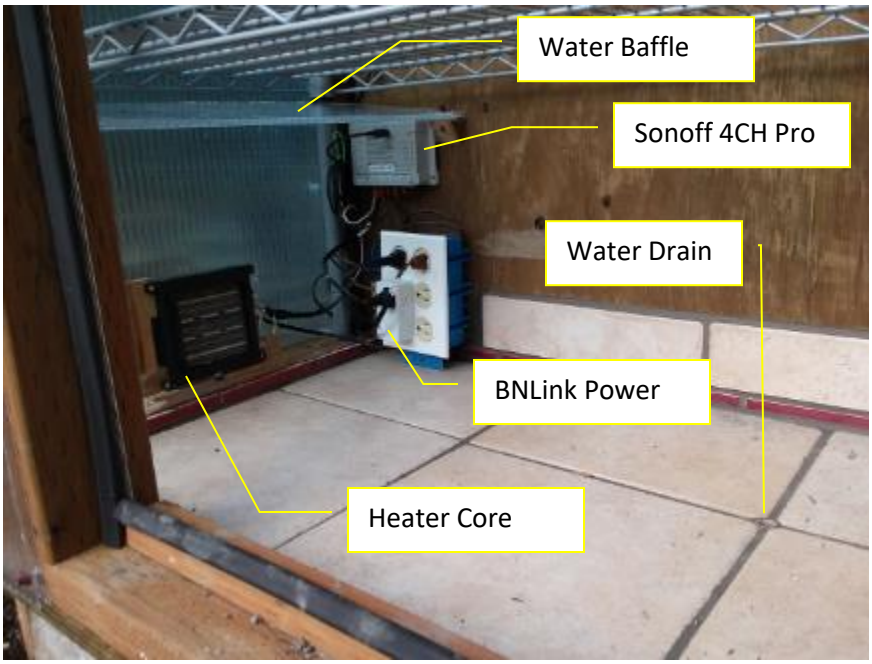


Figure 343 Greenhouse Controls Install

21.20.1 Sonoff 4CH Pro

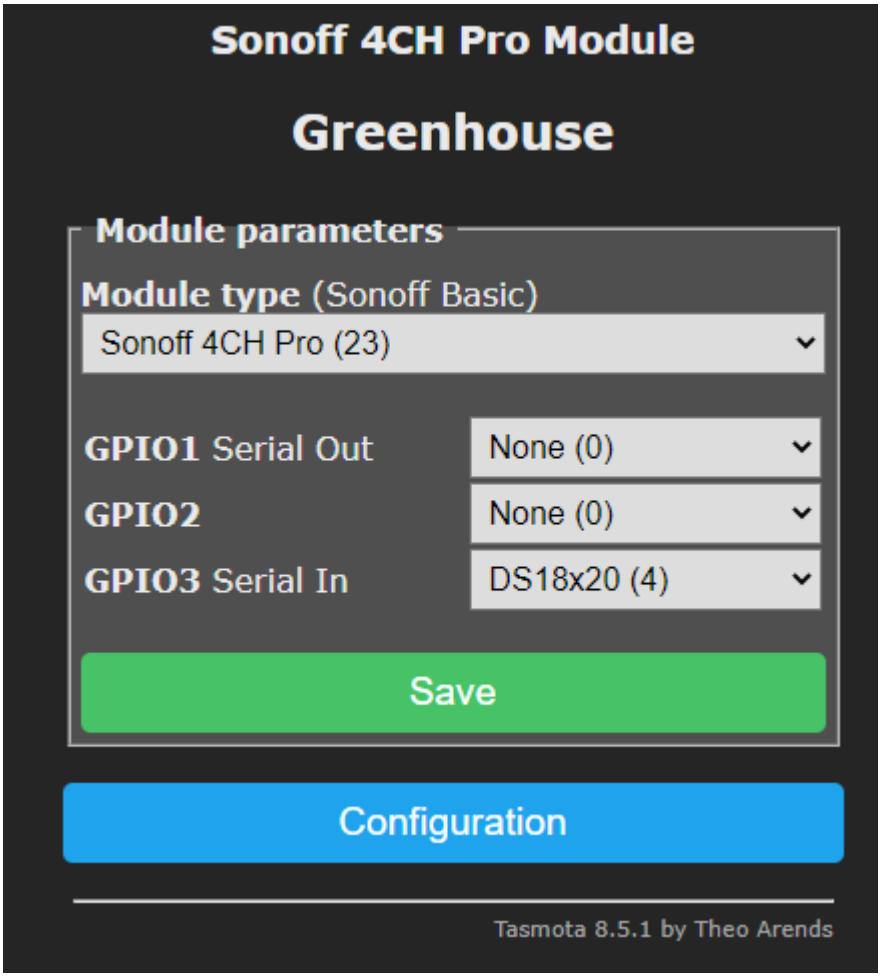


Figure 344 Greenhouse Sonoff 4CH Pro Configuration

A 4.7K pullup resistor between 3.3V and GPIO3 was used for the DS18B20. Since only a single sensor was being used it could have been done using the internal pullup resistor inside the ESP8255 with configuration through SetOption.

It would be possible to make source code changes to provide a setpoint interface via MQTT rather than the hard-coded values, but the rules were the most expedient and easy enough to update the rule via browser in the Tasmota console if desired.

<u>Pass 1 Rule that was later changed to rule during interation 2</u>

The control algorithm was implemented with the Tasmota rule shown below. The first is with control resolution of 10 seconds and the second with control every minute. The issue experienced with ruletimer is that sometimes it did not start so control period event did not trigger.

Rule1 ON system#boot DO backlog var1 0;var2 0;var3 0;ruletimer1 10 ENDON ON rules#timer=1 DO backlog ruletimer1 10;power1 %var1%;power2 %var2%;power3 %var3% ENDON ON DS18B20#Temperature<45 DO var2 1 ENDON ON DS18B20#Temperature<50 DO var1 1 ENDON ON

DS18B20#Temperature>55 DO backlog var1 0;var2 0 ENDON ON DS18B20#Temperature>95 DO var3 1 ENDON ON DS18B20#Temperature<85 DO var3 0 ENDON

Rule1 ON system#boot DO backlog var1 0;var2 0;var3 0 ENDON ON Time#Minute DO backlog ruletimer1 10;power1 %var1%;power2 %var2%;power3 %var3% ENDON ON DS18B20#Temperature<45 DO var2 1 ENDON ON DS18B20#Temperature<50 DO var1 1 ENDON ON DS18B20#Temperature>55 DO backlog var1 0;var2 0 ENDON ON DS18B20#Temperature>95 DO var3 1 ENDON ON DS18B20#Temperature<85 DO var3 0 ENDON

The same rule in a more understandable format:

> Rule1 ON system#boot DO backlog var1 0; var2 0; var3 0; ruletimer1 10 ENDON
>
> ON rules#timer=1 DO backlog ruletimer1 10; power1 %var1%; power2 %var2%; power3 %var3% ENDON
>
> ON DS18B20#Temperature<45 DO var2 1 ENDON
> ON DS18B20#Temperature<50 DO var1 1 ENDON
> ON DS18B20#Temperature>55 DO backlog var1 0; var2 0 ENDON
> ON DS18B20#Temperature>95 DO var3 1 ENDON
> ON DS18B20#Temperature<85 DO var3 0 ENDON

Var1, var2 and var 3 are the desired states of each of the relays. Var3 is the cooling. Var 2 is the second heating element. Var1 is the first heading element and the fan for the two heating elements. The desired control ranges are:

Above 95 run exhaust fan (var3) until cooling to 85 is achieved.

Below 55 run first heating element (var1) until 65 is achieved. If below 50 then also turn on the second heating element (Var2)

Control sampling is done at 10 second intervals with ruletimer1.

When the temperature change event occurs (DS18B20#Temperature) then the above control ranges are evaluated to set the three desired relay states.

<u>Iteration 2 Updated Rule</u>

It was determined that a single heater coil is sufficient for this size greenhouse. This provides for more straightforward rules. Contorl points were also tweaked. Two rules are now employed. One for Heat and one for Cool. This allows the rules to be independenly turned on and off. Mem1 is used to establish the minimum setpoint for heating with a hysteresis of 10 degrees. Mem2 is used for the cooling maximum setpoint with a 5-degree hysteresis. Control ranges are 50→55 heating, 80→75 cooling.

> (Heat mem1 setpoint 50, hysteresis mem3 5)
>
> rule1 ON DS18B20-2#temperature DO event t1=%value% ENDON ON Time#Minute DO Backlog var3 %mem1%; add3 %mem3%; var1 %var3% ENDON ON event#t1<%mem1% DO Power1 1 ENDON ON event#t1>%var1% DO Power1 0 ENDON

(Cool mem2 setpoint 80, hysteresis mem3 5)

rule2 ON DS18B20-2#temperature DO event t2=%value% ENDON ON Time#Minute DO Backlog var4 %mem2%; sub4 %mem3%; var2 %var4% ENDON ON event#t2>%mem2% DO Power3 1 ENDON ON event#t2<%var2% DO Power3 0 ENDON

21.20.2 Sonoff Basic

A Sonoff Basic was used for the moisture sensor and the external temperature. Both of these could have been done with the same Sonoff 4CH Pro, but soldering the A0 pin of the ESP8285 is difficult and I had already done it with the Sonoff Basic so just applied it to this application.

The hardware mods to the Sonoff consisted of soldering a wire to the TOUT (A0) pin of the ESP8266 and physical connection of the DS18B20 temperatue sensor to the solder point provided for GPIO14. AO is only available on the ESP8266 SMD so a very fine solder iron under microscope magnification was needed to make this available.

The Analog input used for the moisture sensor provides a range of 0 to 3 VDC. The AO input accepts voltages in the range of 0 to 1 VDC so a voltage divider was used to scale the voltage. The resistors selected are 39K on the ground side and 47K on the sensor side which provide an ambient air sensor reading of 1.0 VDC. The 1.0 VDC is reported by the Sonoff as 1024. As moisture increase the voltage decrease. The range that was experienced in the greenhouse was approximagely 600 to 1000 for the DFRobot capacitance moisture sensor.

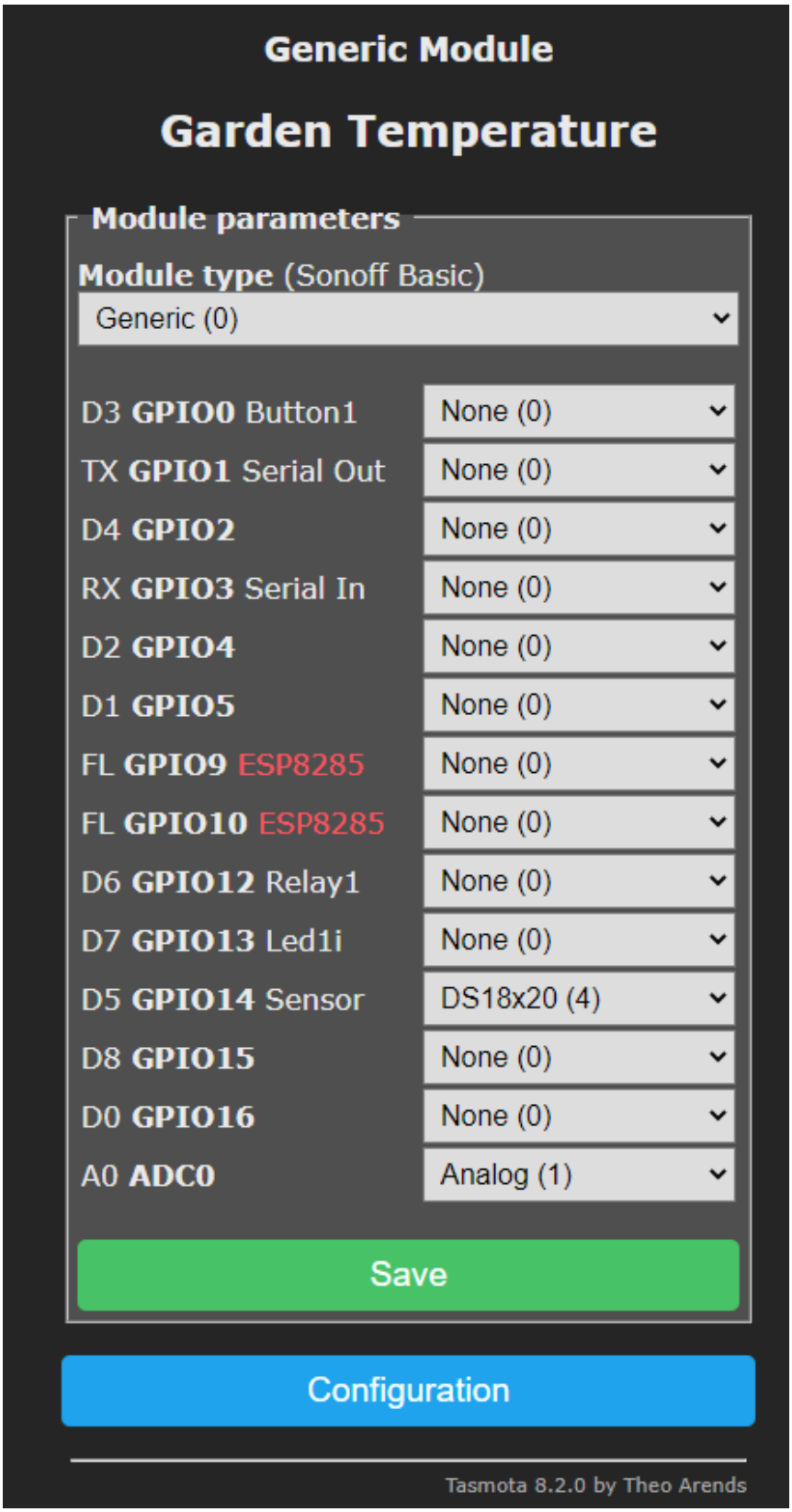


Figure 345 Greenhouse Sonoff Basic Configuration

The external temperature measurement is used for analysis of the temperature gradient inside vs. outside the greenhouse. The moisture sensor is used in a control container to gauge when watering is required.

Humidity measurement was also considered via use of DH11T rather than DS18B20, but there is no control function needed based upon humidity so it was excluded.

21.20.3 BN-Link Power Plug

The power routed thorugh the Sonoff 4CH Pro is connected through a smartplug that has energy monitoring capability. This plug is described in Section 21.13.8. It was selected based upon its 15 Amp rating for the 1500-watt heater core.

There was no additional configuration for this plug other than selecting its power on state to always be ON rather than the default of the prior state. The only parameter being monitored is the Total Energy (Wattage) being consumed, but other current or daily measures are also available for specific investigations. The Total Energy is desired to assess the cost of maintaining the environment.

21.21 Solar Ground Heater

Spring ground tempertures are low due to effects of Winter and low sun angles. To increase the growing season a water loop is run under raised bed with a coil exposed to sunlight and a recirc pump to move the water. The recirc pump (https://www.amazon.com/gp/product/B07QBMYXWY/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1) is managed by a Sonoff S31 using temperature senosrs to measure water temperature on the exposed loop and the soil temperature at six-inch soil depth.

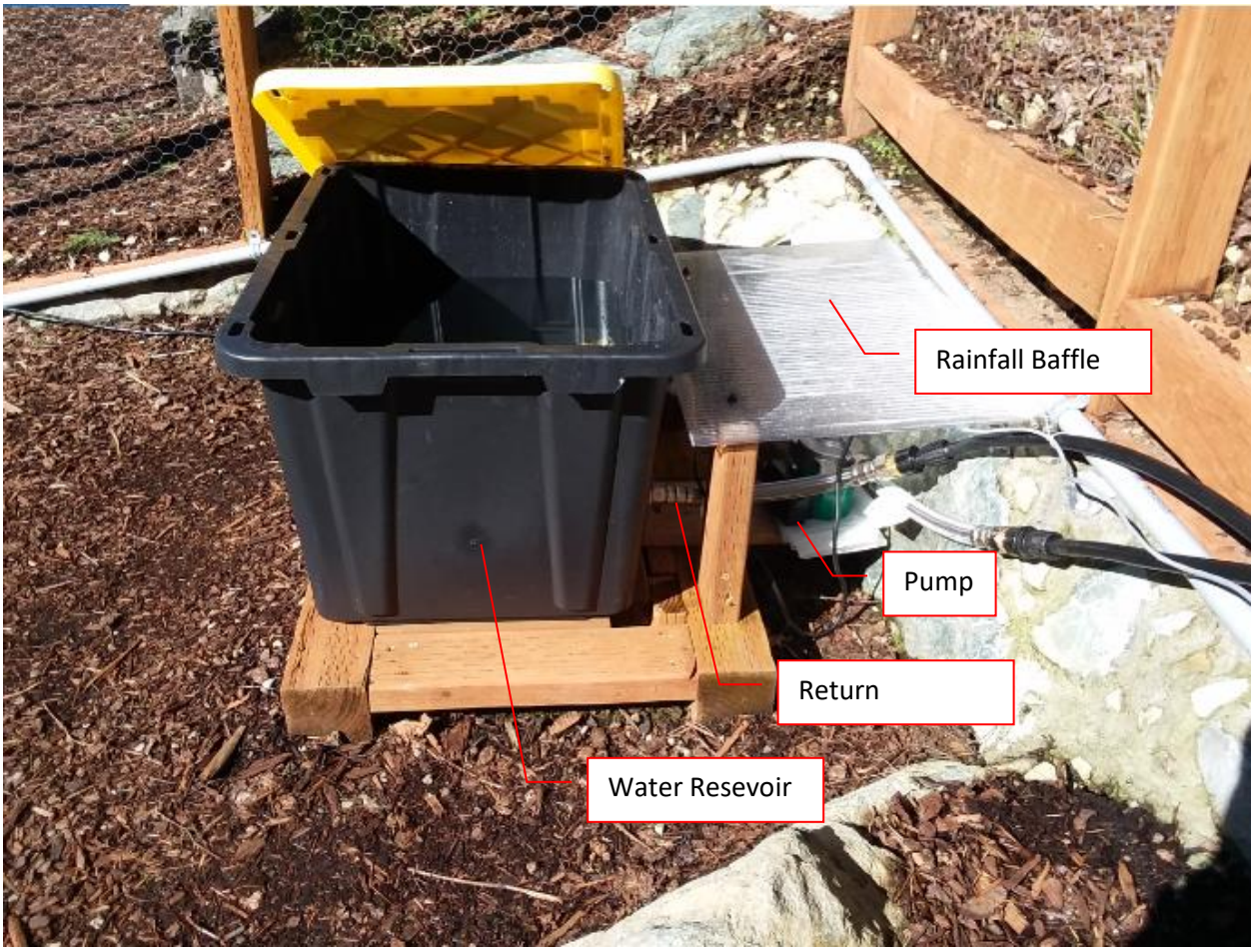


Figure 346 Solar Water Recirc Pump and Tank

Figure 347 Solar Collection Water Tubing

A rule is used to assure that water circulates in the tub to prevent freezing and also circulates when there is a positive temperature diferential between the ground temperature and the water temperature in the tube. The rule is a variant of the Tasmota Rules Cookbook solar pool heater example.

The control parameters of on/off hysteresis and the minimum air temperature for recir are shown in red in the rules below. When in a safe temperature range the recirc pump will turn on when the delta temperature is at least 2 degrees and turn off when within 1 degree.

t1: ground temp
t2: air/heated temp
var1: in valid control temp range?
var2: off threshold temp with 1 degree hysteresis
var3: on threshold temp with 2 degrree hysteresis
var5: companion to var2 so assure var2 never holds temporary value
var6: companion to var3 to assure var3 never holds temporary value
lowest valid air temp for control set to >40 for heating, >36 for exit of freeze circulation
protection from freeze temperature <34 for bypass, <35 for pump freeze circulation

<u>Rule 1 (Heat transfer control)</u>

Rule1 ON DS18B20-2#temperature DO event t2=%value% ENDON ON DS18B20-1#temperature DO event t1=%value% ENDON ON event#t2<36 DO var1 0 ENDON ON event#t2>40 DO var1 1 ENDON ON

event#t2<35 DO var1 1 ENDON ON event#t1 DO Backlog var5 %value%; add5 1; var6 %value%; add6 2; var3 %var6%; var2 %var5% ENDON ON event#t2>%var3% DO Power1 %var1% ENDON ON event#t2<%var2% DO Power1 %var1% ENDON

A second rule was added to control a bypass switch. This switch will control if the water in the tube will pass through the ground or will only return to the tank. When the recirc pump is running to protect from freezing then the water should not be routed into the ground as that will decrease the soil temperature. A Jinvoo water switch, shown in Figure 348, is used to control a two-way valve. It is commanded to position every minute based upon the the unheated water temperature used in rule 2.

<u>(Rule 2 control bypass valve)</u>

Rule2 on DS18B20-2#temperature>40 do var7 1 endon on DS18B20-2#temperature<35 do var7 0 endon on Time#Minute do publish WaterSwitch/cmnd/power %var7% endon

The plumbing for project is diagrammed below. Rule 1 controls the on/off for the pump. Rule2 controls the bypass (OFF) / normal (ON) for the 2-way valve. When the valve is in normal the water is routed though the soil. When in bypass the flow through the ground is blocked and the bypass route is the means for the water return to keep water moving in freezing tempertures. Of course, very low temperatures will still freeze the water so until Spring the system is blown out with air to protect from freezing.

```
tank  --------------> pump  ---------------------->--------- Into Unheated Tube   -----
source                                                                              | Sun-Exposed Coil
              Bypass (OFF)            ----------<------- Exit Heated Tube  <----------
               |<-------------------------| "T"
tank     <---|                          --------->--------- Ground Coil (Into Ground)  ------>|  Ground Coil
return        |<-----------------------------------<-------- Ground Coil (Exit Ground)  <-------|
              Normal (ON) – handle vertical
          2-way valve
```

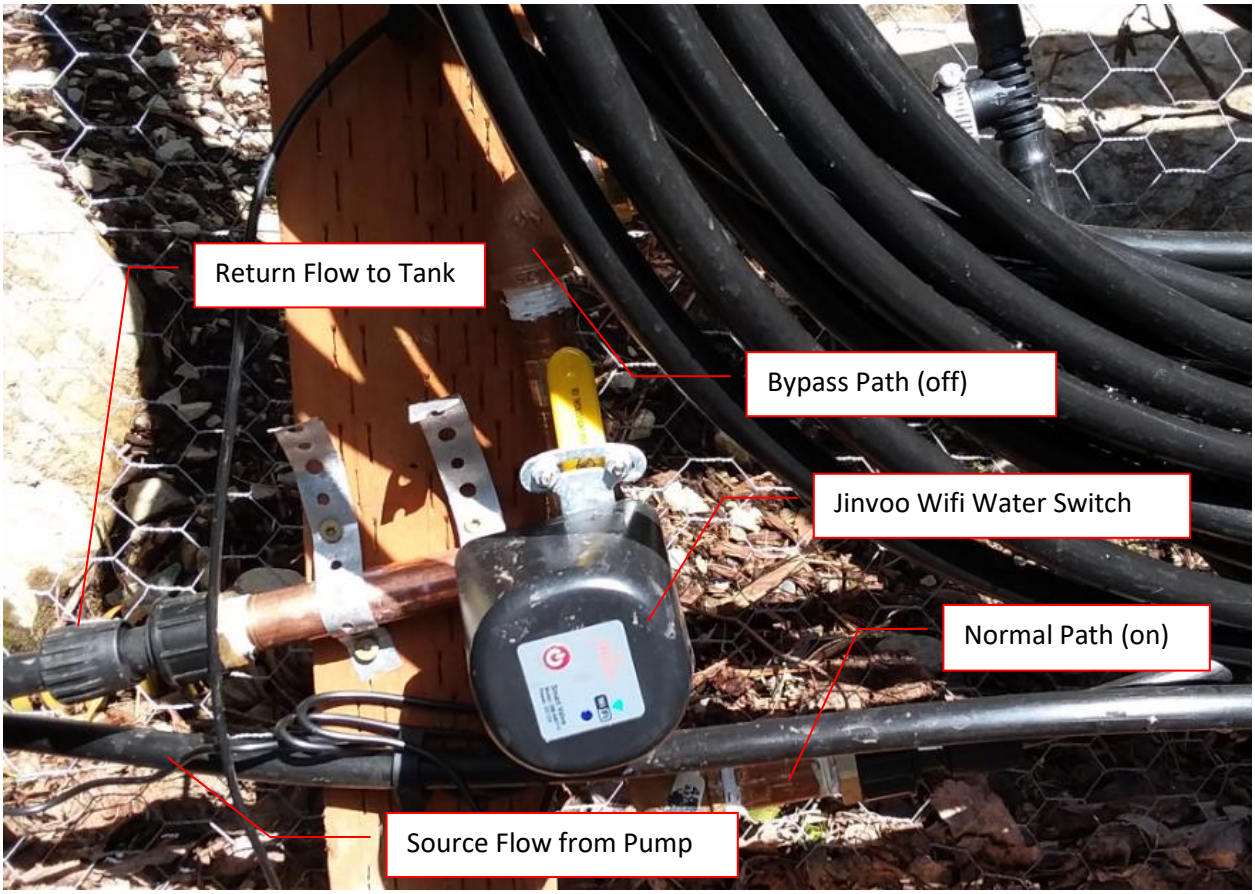


Figure 348 Bypass flow control and 2-way valve

The project was installed in early March with 1000 ft irrigation tube coiled with 200 ft in the ground under the raised bed and 800 ft coiled for sun exposure. Ground temperature six inches below the surface was measured at 42.2 degrees at time of initial install. The behavior of the heat transfer is shown in Figure 349. Even with a low sun angle the amount of heat tranfered was significant with ground temperature increasing by about 8 degrees.

The red lin in the chart is the temperature at the tube prior to entry into the ground. The white line is the ground temperature at 6 inche depth and 4 inches away from a tube. The yellow is the control for the water recirculation pump showing water was moving until 7:30 PM and then resumed around sunrise.

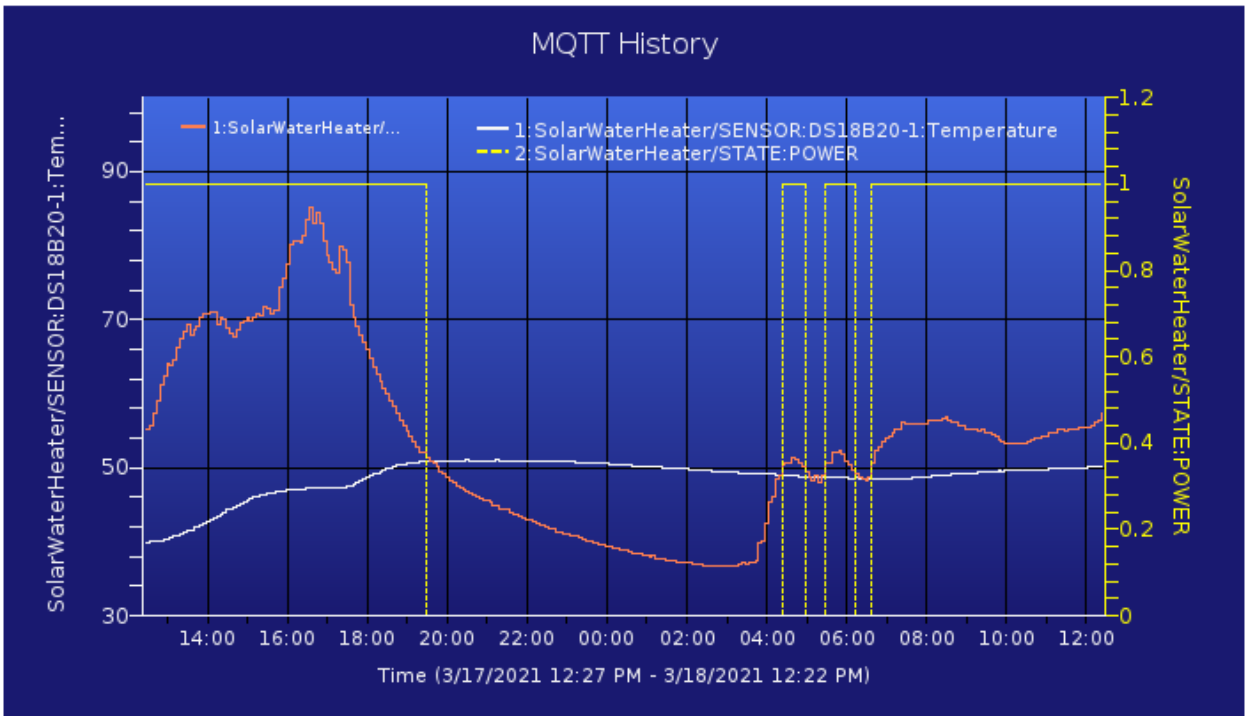
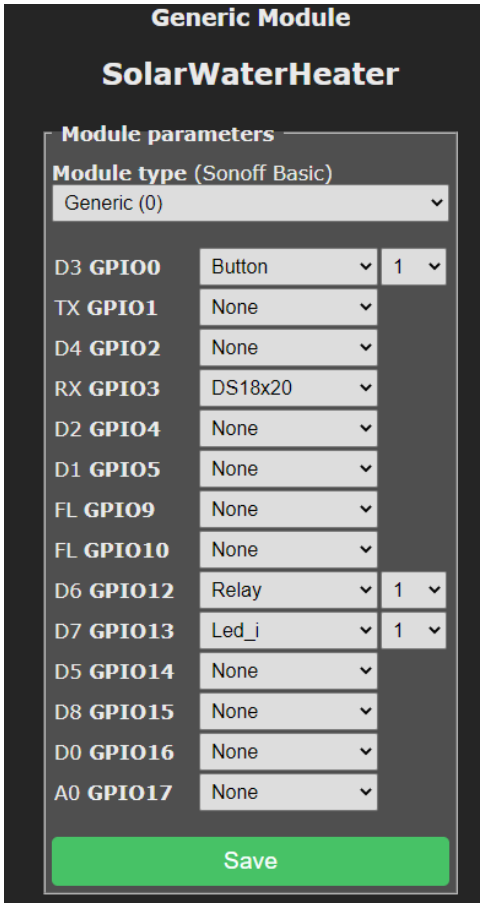


Figure 349 Solar Heat Transfer Control Performance



A 2022 revision to the solar water heater is the addition of a bucket heater for the water resevoir as an alternate means to protect the solar tubes from freezing. The 2021 approach was to turn on the recirc pump when temperatures approach freezing and divert the water so it does not pass through the ground soil. Moving water will have less chance of freezing.

The strategy for 2022 is to heat the water resevoir when the temperatures approach freezing and continue to circulate the water when it is being heated. The bucket heater is powered through a Sonoff S31 that has two temperture sensors connected. Temperature#1 is the water resevoir temperature. Temperature#2 is the air temperature. Two rules are used. Rule1 is used to determine when the the turn-on and turn-off setpoints have been traversed. Rule2 is used to control the power to the bucket heater and to monitor for the heater being on too long. The protection is setup so that withing every hours the heater cannot be on for more than 55 minutes and if it does exceed this it will be turned off for 60 minutes independent of the temperatures. The protection is to prevent a runaway of the heater. These protection thresholds still need to be tweaked based upon the observed heating rate.

<u>Rule1 Narative</u>

var1 power command
var2 on command
var4 count of on minutes
var5 too cold setpoint
var6 too hot setpoint
var7 max water temperature
DS18B20-1#temperature air temperature
DS18B20-2#temperature water temperature
DS18B20-3#temperature S31 case temperature

At startup define the variables to nominal state, setpoints, one-shot on rule1 and rule2 always on

```
Rule1 ON system#boot do backlog var1 0;var2 1;var4 0;var5 40;var6 50; var7 70;rule2 4;rule1 5;rule2 1 endon
```

Change the power command (var1) when either of the setpoint thresholds are exceeded. When Off then power is 0. Var2 is used for the On state. Normal On state is 1, but could be 0 if rule2 timer protection has been activated.

```
ON DS18B20-1#temperature<%var5% DO var1 %var2% ENDON
```

```
ON DS18B20-2#temperature>%var6% DO var1 0 ENDON
```

Twenty seconds after startup report the setpoints. Cannot do it at systemboot because everything has not yet been setup.

```
ruletimer1 20 endon
```

```
on rules#timer=1 do publish %topic%/Setpoints {"Low":%var5%,"High":%var6%,"Max":%var7%}
```

Rule2 Narative

Every 5 minutes command the relay to the rule1-determined state. If being turned on then Var4 will count up to keep track of how long it has been on during the current hour. Event c4 is set to the count so it can be used as trigger

Rule2 ON Time#Minute|5 DO backlog Power1 %var1%;add4 %var1%;event c4=%var4% ENDON

If the count exceeds hourly limit then change the On value (var2) to 0 so the relay will not be turned on based upon rule1 temperature. Init the relay command (var1) to 0 so next minute it will turn off. Publish MQTT to indicate the protection has been activated.

ON event#c4>55 do backlog var2 0;var1 0;publish %topic%/OnOvertime %var4%

Each hour reset the counter to 0. Every other hour reset the protection logic to enable the heater to be turn on again based upon temperature thresholds.

ON Time#Minute|60 do var4 0 endon

ON Time#Minute|120 do var2 1 endon

If temperature exceeds max limit, then turn everything off and report via MQTT

ON DS18B20-1#temperature>%var7% DO backlog var1 0;var2 0;power 0;publish %topic%/Overtemp %value% ENDON

If water temperature it too low then enable heater

ON DS18B20-2#temperature<%var5% DO var1 1 endon

Rules for Copy/Paste

Rule1 ON system#boot DO backlog var1 0;var2 1;var4 0;var5 40;var6 50;var7 70;rule2 4;rule1 5;rule2 1; ruletimer1 20 ENDON ON rules#timer=1 do publish %topic%/Setpoints {"Low":%var5%,"High":%var6%,"Max":%var7%} ENDON ON DS18B20-1#temperature<%var5% DO var1 %var2% ENDON ON DS18B20-2#temperature>%var6% DO var1 0 ENDON

Rule2 ON Time#Minute|5 DO backlog Power1 %var1%;add4 %var1%;event c4=%var4% ENDON ON event#c4>55 do backlog var2 0;var1 0;publish %topic%/OnOvertime %var4% ENDON ON Time#Minute|60 do var4 0 ENDON ON Time#Minute|120 do var2 1 ENDON ON DS18B20-2#temperature>%var7% DO backlog var1 0;var2 0;power 0;publish %topic%/Overtemp %value% ENDON ON DS18B20-1#temperature<%var5% DO var1 1 ENDON

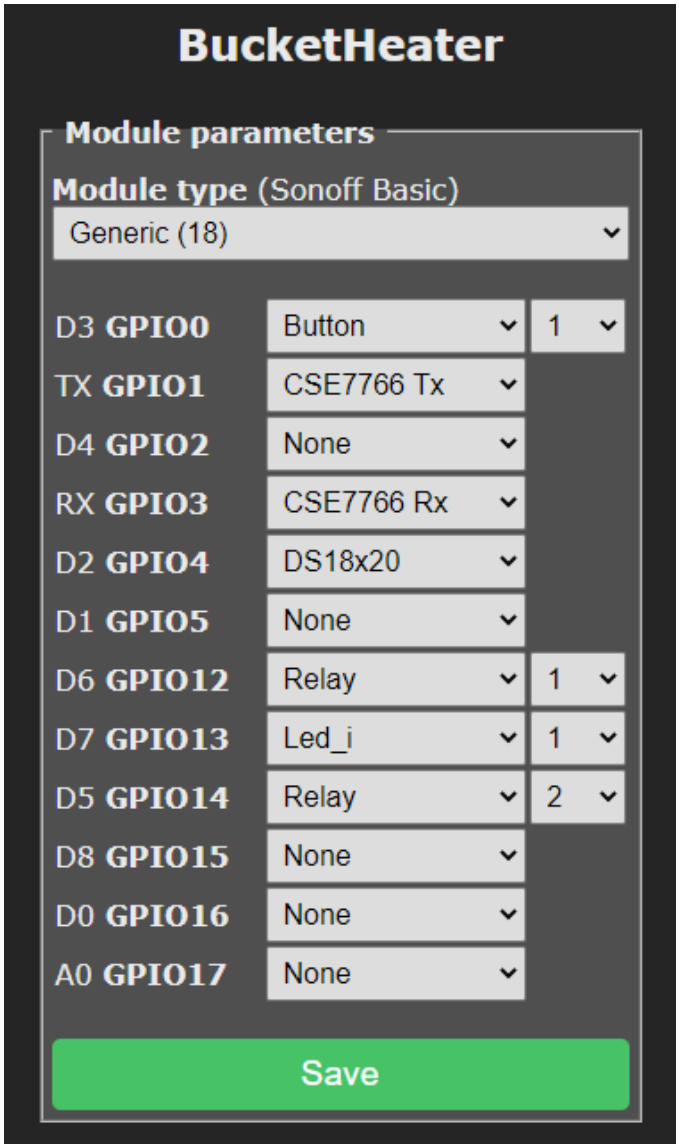


Figure 350 Bucket Heater Sonoff S31 Configuration

The Solar Water Heater Sonoff S31 Lite Rule 2 has been turned off for 2022. Rule1 remains unchanged.

<u>2022 Rules for Solar Water Heater Sonoff S31 Lite</u>

ON Rule1 ON DS18B20-2#temperature DO event t2=%value% ENDON ON DS18B20-1#temperature DO event t1=%value% ENDON ON event#t1 DO Backlog var5 %value% add5 1; var2 %var5%; var3 %value% ENDON ON event#t2<%var3% DO var1 0 ENDON ON event#t2>%var2% DO var1 1 ENDON ON event#t2<36 DO var1 1 ENDON ON Time#Minute DO Power1 %var1% ENDON

OFF Rule2 on DS18B20-2#temperature>40 do var7 1 endon on DS18B20-2#temperature<35 do var7 0 endon on Time#Minute do publish ~~WaterSwitch/cmnd/power %var7%~~ publish BucketHeater/cmnd/power %var7% endon

21.22 Mail Delivery Notification via LoRa

The problem-space being solved is awareness of US Mail delivery for a mailbox that is located 600 ft from house in a heavily wooded area. Several years ago, a mail delivery notification unit available at Radio Shack was installed with a battery-operated RF sensor on the mailbox lid and the receiver unit located about 100 ft away at the nearest power source. The audio of the receiver was interfaced to a UPB IO module so when the mail alarm sounded a UPB message was delivered to HS for notification.

While protected from the elements the "inside rated" receiver died last year and replacement no longer available. The replacement solution is a combination of Zigbee, LoRa and MQTT technologies. Zigbee was selected for the mail delivery sensor based upon its efficient battery management. LoRa was selected for RF that spans the 600 ft non-line-of-sight distance. MQTT was used as a matter of convience.

The sensor that was selected was a Xiaomi Aqara Vibration Sensor that is mounted on the door of the mailbox. The sensor is small and does not interfere with the mailbox operation.

Figure 351 Vibration (Door Position) Sensor

The sensor worked well in bench testing with Y axis motion sensed. To receive the notification a CC2531 such as shown in Figure 352 was used. It was flashed with the same firmware as used for Zigbee2MQTT.

Figure 352 CC2531 Zigbee Coodinator

In field testing it appears the metal of the mailbox interfered with the operation of the vibration sensor. The sensor would work fine in free air, but not when mounted to the metal surface.

The second choice was a Zigbee window/door sensor such as Aqara ZigBee Version Window Door Sensor shown in Figure 353. In this case the larger component was mounted at the same location as the vibration sensor on the mailbox lid. The smaller magnet was not use, but a smaller form factor maget was installed at the top of the mailbox to provide the "pull" of the reed relay in the part mounted on the door. Since the mailbox was metallic the magnet could be mounted without any adhesive.

Figure 353 Door Open Sensor

As a first choice for microcontroller to interface with Zigbee, ESP82666 running the Zigbee build of Tasmota was used. This project is early in the development cycle and out-of-the-box success was not achieved. My actual intent is to use ESP32 that has a LoRa interface integrated into its module and port the Zigbee code from ESP8266 to the ESP32 using the port developed for the BLE tracking project. Without success with ESP8266 I elected to put this approach on hold until the Tasmota-Zigbee matures.

The backup is the proven Zigbee2MQTT development that uses a processor similar to a Raspberry Pi. In this case I utilized a Hardkernel Odroid C1 with DietPi OS (Debian Lite) installed. This previous effort was described in Section 19.

Since the EPS32/Lora was on hold the LoRa interface became a pair of Ebyte EB32 LoRa transponders such as shown in Figure 354. In my case the receiver was rated at 100 mw (which does not matter) and the transmitter at 1W. Both use 433 MHz. The approved frequency for LoRa in USA is 915 MHz so the 433 MHz China-approved frequency may not be legal in US. Since it is point to point and 433 MHz was a common sensor frequency, I felt comfortable using it. 433 MHz should actually be better through trees than 915 MHz.

Figure 354 Ebyte EB32 LoRa Transponder

The EB32 uses a RS232 or RS485 to interface the controller. I elected the RS232 using a USB/Serial adapter on the Odroid C1. This was a China-supplied adapter. I normally use a FTDI-clone chip, but tried an old Prolific chipset one. It was recognized, but had issues with going offline. I added software management in the C1 software to deal with it.

On the HS side I used two different models of FTDI-based USB/Serial. Given it worked with Prolific and multiple FTDI I do not think the EB32 is very sensitive to the serial interface quality. In the first fielding I used a Lantronix EPS1 (IP/Serial) adapter on the HS side so I could relocate the receiving EB32 to a higher location that had better reception. mcsMQTT was updated to receive Serial from either IP/Serial or COM sources.

Subsequently I removed the older technology Lantronix EPS1 and replaced it with an Wemos D1 Mini to which a NS-RS232 was attached. Tasmota was installed and configured to operate on GPIO12 and GPIO14 as s software serial bridge at 9600 baud. The hardware unit is shown in Figure 355. The setup of the IP to Serial bridge is shown in Figure 356.

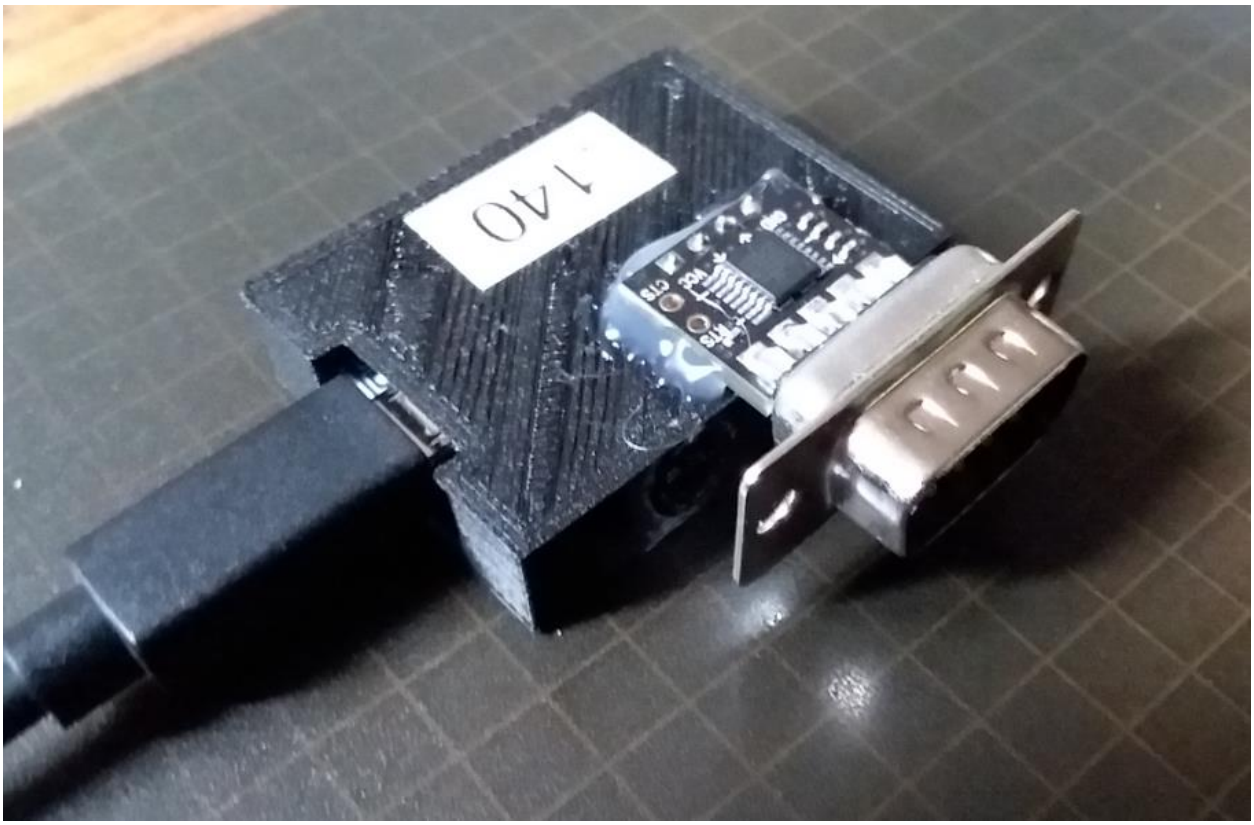


Figure 355 IP Serial Hardware

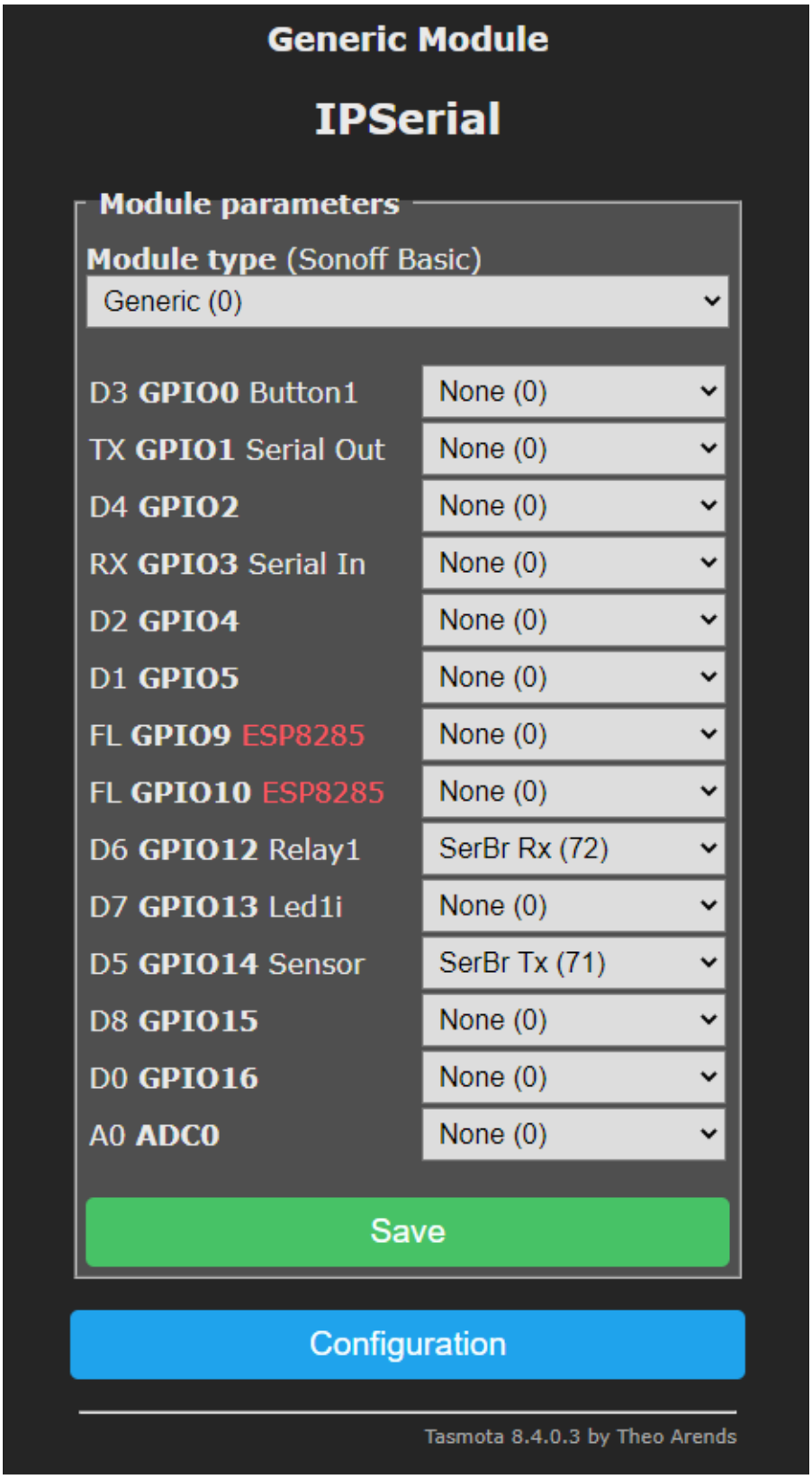


Figure 356 IP Serial Bridge Configuration

With this configuration the serial data is delivered in MQTT payloads in the RESULT topic such as:

IPSerial/RESULT = {"SSerialReceived":{"T":"Mail","P":{"hb":1,"temp":126}}

The Odroid C1 and EB32 were mounted in a repurposed Costco Chocolate Raisin container shown in Figure 357. An angle grinder was used to slice slots in the lower side for ventilation. These are not visible in figure as they are located at the bottom so that they would not be affected by rain. A drill was used to penetrate the bottom for antenna and power cable penetration. A scrap wood bock was used to provide a physical barrio between EB32 and C1. A screw was use in the center of the lid to mount to a tree near the power source. It was snug but still allowed the lid to spin to engage the remainder of the container.

Figure 357 Install of Mailbox Notification Interface

The second fielding I used an Orbit irrigation enclosure so the electronics is out of sight. This enclosue contained a power receptacle so the power strip was no longer needed and the connection has better protection from the rain.

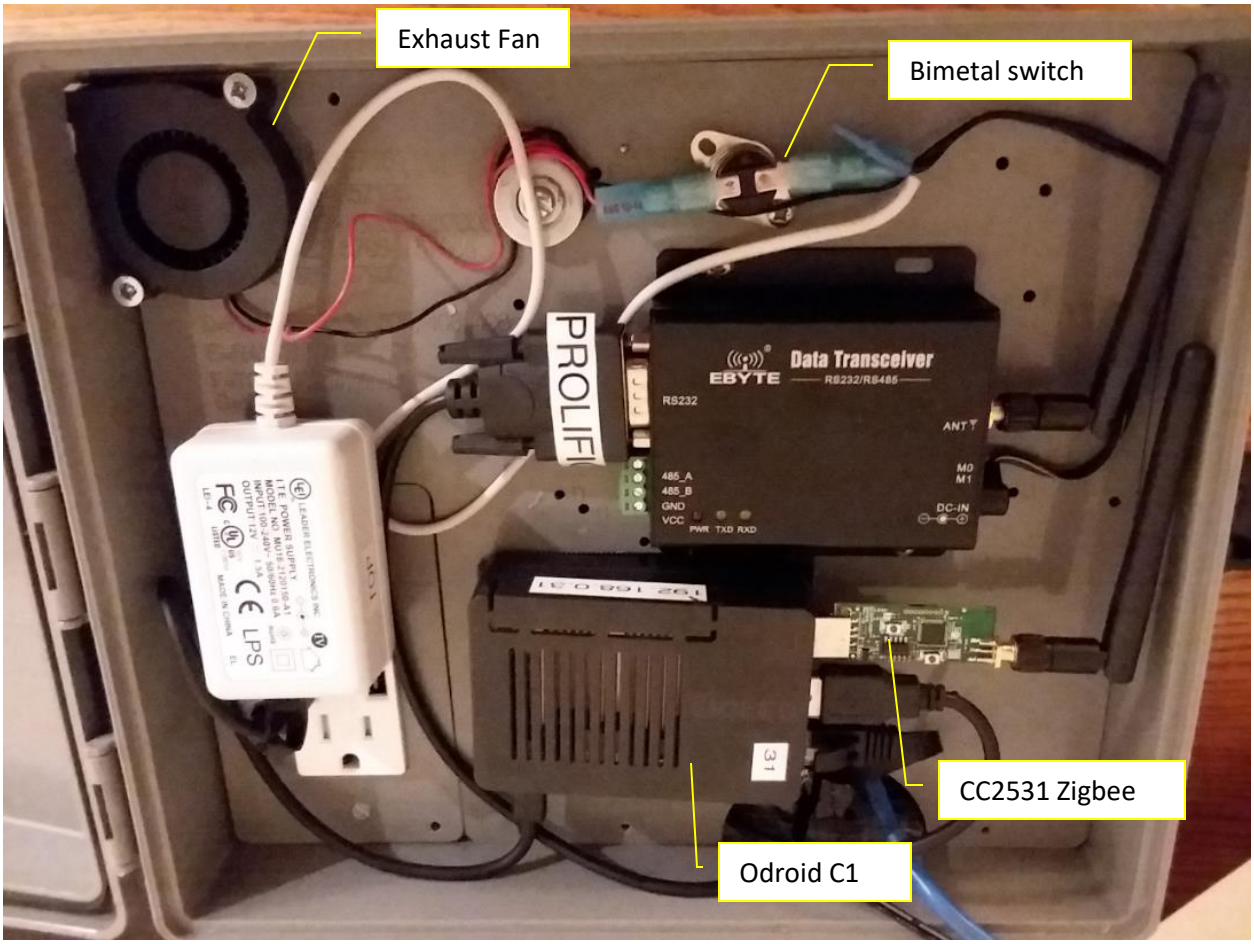


Figure 358 Orbit Irrigation Enclosure for Weatherproofing

The Orbit enclosure has provisions for wiring, but not for ventilation to dissipate heat. To deal with this I added a small fan https://www.amazon.com/gp/product/B07VYRWC3F/ref=ppx_yo_dt_b_asin_title_o09_s00?ie=UTF8&psc=1 . The power to the fan was provided by the same power source as the EByte with a bimetal switch connected in series so it ran when the temperature was about 30C. A higher temperature switch point may be more appropriate in warmer climates since the electronics should withstand warmer surrounding.

The 12VDC power for the fan and Ebyte E32 was provided by a wallwart. Power for the Odroid C1 is from the 5VDC USB plug in the power receptical. This style of receptical avoided the need for the USB-5VC plug that was used in the original mounting with a powerstrip.

Within the Odroid C1 are three packages. Zigbee2MQTT interfaces the CC2531 and delivers the changes of the mailbox sensor via MQTT. Mosquitto was installed as the MQTT broker. This was not the original choice, but became the most convenient as Mosquitto was an easy install option via DietPi. A .NET

application was developed (LORA.exe) to subscribe to MQTT topics delivered by Zigbee2MQTT and convert the message to Serial for communication via LoRa.

There were various issues on the initial installation so diagnostics were added to LORA.exe to help undertand the issues that existed. A heartbeat was added to confirm the integrity of the LoRa link. Two-way communication via LoRa was not introduced as this added another potential failure mode and I did not feel there was sufficient benefit. I also had only 100 mw at the HS side of the LoRa communication so a lack of connection could easily be attributed to a weak transmit signal.

LoRa.exe communicates using JSON payload. Figure 359 provides a snapshot that encompasses a period of time that includes multiple open/close of the mailbox lid and the heartbeat that is used to confirm the mailbox monitoring is occurring.

The serial communication is encoded with “T” to be the topic which is always “Mail”. The payload is “P” with keys of “hb” for periodic heartbeat and “door” and “link” for the open/close position of the mailbox lid and the zigbee link quality between the sensor and the CC2531. A link of over 100 is reported when the two units are next to each other. In this case the two had a separate of about 50 ft.

After this screenshot the battery status was also added. When the vibration sensor was used the key reported was “y” for the Y axis position that was typically 1 to 89 in my testing.

LoRa is a low bandwidth technology. LORA.exe assured the bandwidth was not exceeded by shorthand notation of the message payload and limiting messages to a rate of one per two seconds. A standard LoRa message can be no more than 56 characters.

Message History

/\	p/s	lastdate	topic	payload
760	S	2020-08-06 13:30:18	Serial/192.168.0.49:3001	{"T": "Mail","P":{"hb":0}}
761	S	2020-08-06 13:31:18	Serial/192.168.0.49:3001	{"T": "Mail","P":{"hb":1}}
762	S	2020-08-06 13:32:18	Serial/192.168.0.49:3001	{"T": "Mail","P":{"hb":0}}
763	S	2020-08-06 13:32:27	Serial/192.168.0.49:3001	{"T":"Mail","P":{"door":1,"link":39}}
764	S	2020-08-06 13:32:29	Serial/192.168.0.49:3001	{"T":"Mail","P":{"door":1,"link":44}}
765	S	2020-08-06 13:32:31	Serial/192.168.0.49:3001	{"T":"Mail","P":{"door":0,"link":47}}
766	S	2020-08-06 13:32:33	Serial/192.168.0.49:3001	{"T":"Mail","P":{"door":0,"link":57}}
767	S	2020-08-06 13:32:37	Serial/192.168.0.49:3001	{"T":"Mail","P":{"door":1,"link":21}}
768	S	2020-08-06 13:32:39	Serial/192.168.0.49:3001	{"T":"Mail","P":{"door":0,"link":26}}
769	S	2020-08-06 13:33:18	Serial/192.168.0.49:3001	{"T": "Mail","P":{"hb":1}}
770	S	2020-08-06 13:34:18	Serial/192.168.0.49:3001	{"T": "Mail","P":{"hb":0}}

Figure 359 Mailbox Serial Communication History

With HS an event is setup to illuminate the Mail LED on the notification sign when mcsMQTT provides the status indication that the Mail door is at 0 (open) state. This LED remains illuminated until the IR remote is used to change the LED color from red to blue on the sign. The notification sign is described in Section 21.11. The Mail indication LED was added below the Reminder LED of Figure 279.

The HS setup of the device for mail status and reporting to illuminate the Notification frame is shown below. The status values are the LED RGB hex values that control the Notification frame. Blue indicates normal. Red indicates mailbox opened. Yellow indicates that the heartbeat has been lost.

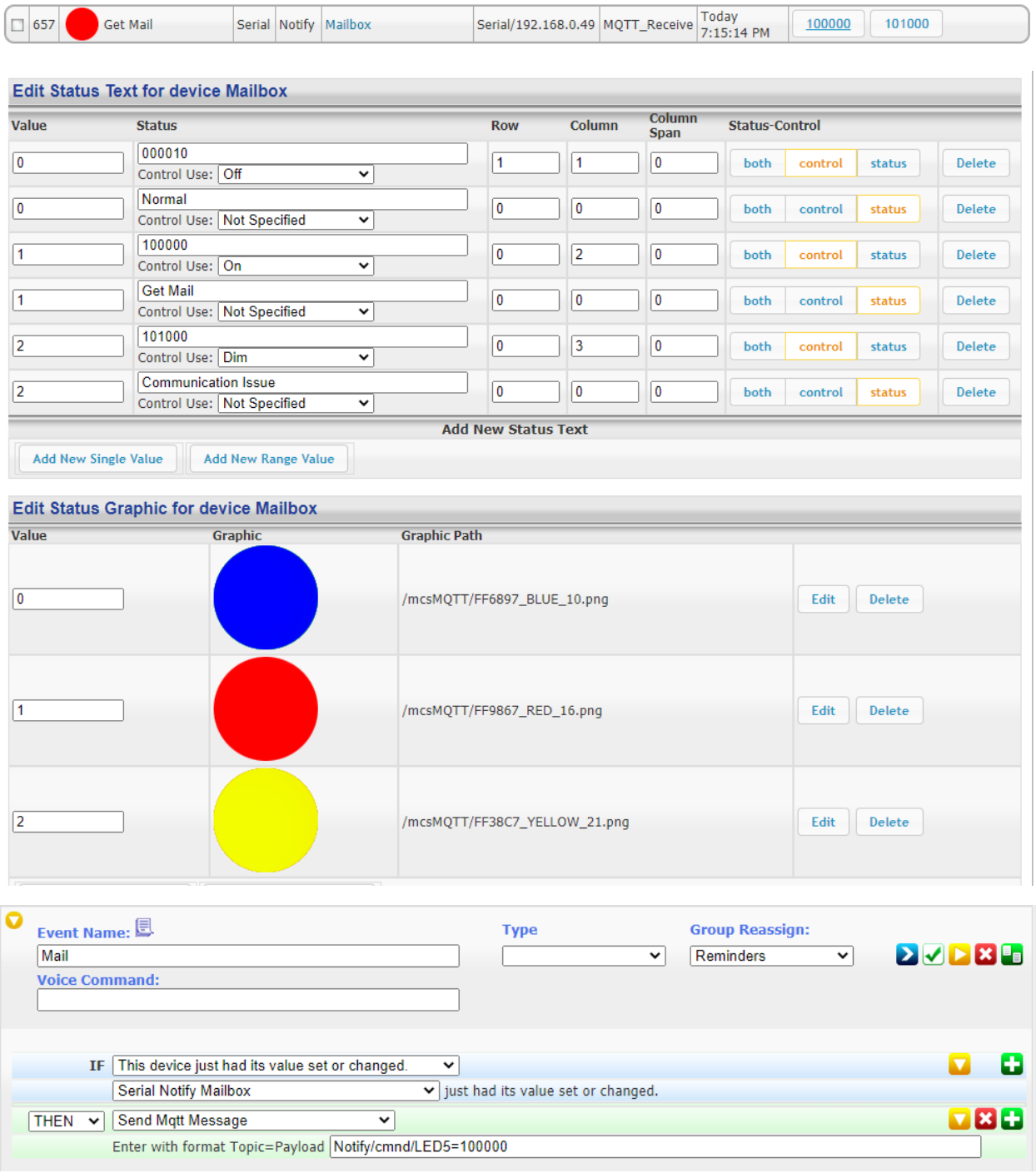


Figure 360 Homeseer Mail Setup

21.23 Alexa Controlled IR

There are many projects that use ESP or RPi for IR send and receive. Tasmota has rich IR support so this is the avenue that I took for my IR desires. The particular use case is to use Alexa to request equipment change that is performed with IR. In particular to switch the video input source on a Samsung television between HDM1 which is the normal settop box input and HDM3 which is connected to Fire TV stick. The Fire TV stick for this purpose is used to show local camera video stream.

The desired operation is to request Alexa to view a particular camera (e.g. Alexa, show road) and to switch the TV input to HDMI3. An Alexa routine would be the choice to do this, but it appears that the "show" verb for video streaming cannot be used in routines. This means two Alexa commands are needed. One to select the camera for Fire TV stick and the second to switch the input source of the TV.

The hardare configuration is based upon the schematic from https://github.com/vsimonaitis/ESP8266-MQTT-IR-Blaster and is shown in Figure 364. The device used was a Wemos D1 mini rather than NodeMCU. The resistor values were changed to match the IR LED that was used.

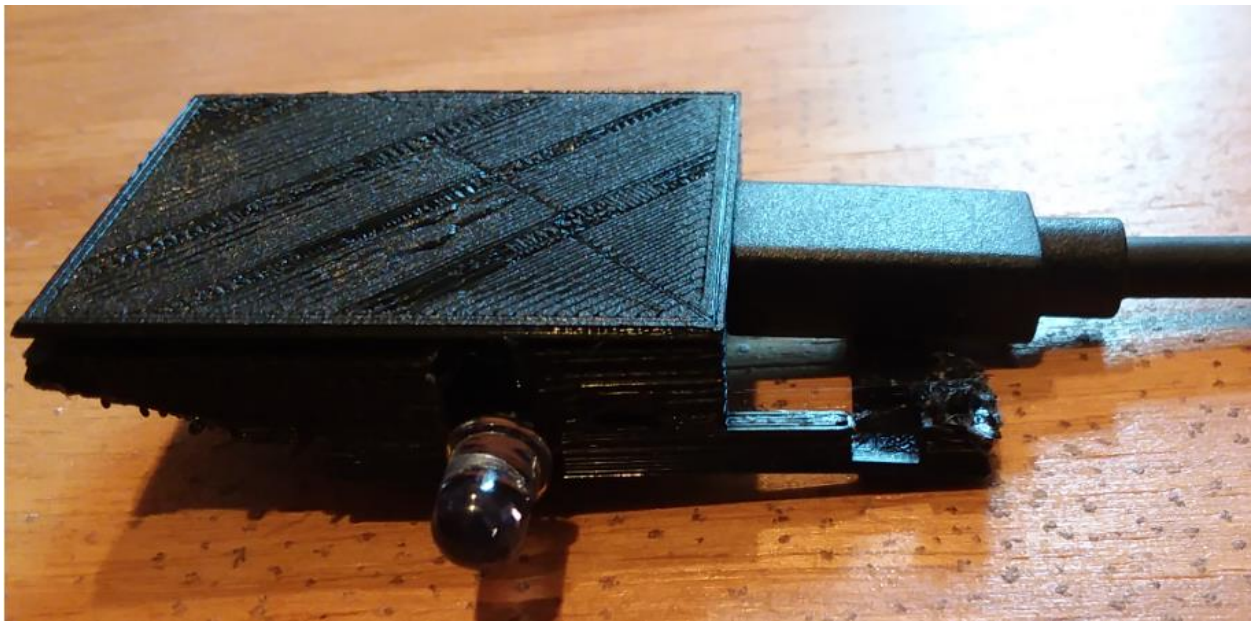


Figure 361 IR Blaster Case/Mount

Tasmota 8.4.0.3 was installed and configured initially as shown in Figure 362. The prototype development contained both IR Receive and IR Send and did not have success with reading what was being sent. A second ESP8266 was used where the second was for receive only and that was successful to collect IR codes from the remotes that were being emulated with this project. The IR receiver could be installed on the same unit as long as simultaneous transmit and receive were not done. The receiver was excluded in the production unit since the use case did not include IR translation.

TV HDMI input select and AV Receiver Volume are the two functions being performed by Alexa. The volume control is for a different use case, but uses the same hardware and firmware. Tasmota has HUE emulation so two dummy lights were setup. This is shown in Figure 363. This allows Alexa, Turn On/Off Camera that will be interpreted as HDMI3 and HDMI1 IR control via Tasmota rules. It also allows Alexa, Turn On/Off Volume which is interpreted as Volume Up and Volume Down in Tasmota rules. An Alexa

routine could also be setup to allow the friendlier Alex, Volume Up/Down. To adequately support a combination of Tasmota rules and Alexa a souce code mod was made to 8.4.0.3 so that every request looks to the rule engine as if it is a change in state of the relay so the rule is triggered and the IR command is executed.

The particular Tasmota rule used for the HDMI input select is Rule 1. The volume is Rule 2 where two IR pulse streams are sent to change the volume two notches.

```
Rule1

  ON Power1#state=1 DO IRSend
{“Protocol”:”SAMSUNG”,”Bits”:32,”Data”:0xE0E043BC,”DataLSB”:0x0707C23D
,”Repeat”:0} ENDON

  ON Power1#state=0 DO IRSend
{“Protocol”:”SAMSUNG”,”Bits”:32,”Data”:0xE0E09768,”DataLSB”:0x0707E916
,”Repeat”:0} ENDON


Rule2

  ON Event#volumeUp DO Backlog IRSend
{“Protocol”:”PIONEER”,”Bits”:64,”Data”:0xA55A50AFA55A50AF,”DataLSB”:0x
A55A0AF5A55A0AF5,”Repeat”:0}; Delay 5 ENDON

  ON Power2#state=1 DO Backlog Event volumeUp; Event volumeUp ENDON

  ON Event#volumeDn DO Backlog IRSend
{“Protocol”:”PIONEER”,”Bits”:64,”Data”:0xA55AD02FA55AD02F,”DataLSB”:0x
A55A0BF4A55A0BF4,”Repeat”:0}; Delay 5 ENDON

  ON Power2#state=0 DO Backlog Event volumeDn; Event volumeDn ENDON


Backlog Rule1 ON; Rule2 ON

In copy/paste format the rules are:
```

Rule1 ON Power1#state=1 DO IRSend {"Protocol":"SAMSUNG","Bits":32,"Data":0xE0E043BC,"DataLSB":0x0707C23D,"Repeat":0} ENDON ON Power1#state=0 DO IRSend {"Protocol":"SAMSUNG","Bits":32,"Data":0xE0E09768,"DataLSB":0x0707E916,"Repeat":0} ENDON

Rule2 ON Event#volumeUp DO Backlog IRSend {"Protocol":"PIONEER","Bits":64,"Data":0xA55A50AFA55A50AF,"DataLSB":0xA55A0AF5A55A0AF5,"Repeat":0}; Delay 5 ENDON ON Power2#state=1 DO Backlog Event volumeUp; Event volumeUp ENDON ON Event#volumeDn DO Backlog IRSend {"Protocol":"PIONEER","Bits":64,"Data":0xA55AD02FA55AD02F,"DataLSB":0xA55A0BF4A55A0BF4,"Repeat":0}; Delay 5 ENDON ON Power2#state=0 DO Backlog Event volumeDn; Event volumeDn ENDON

Backlog Rule1 ON; Rule2 ON

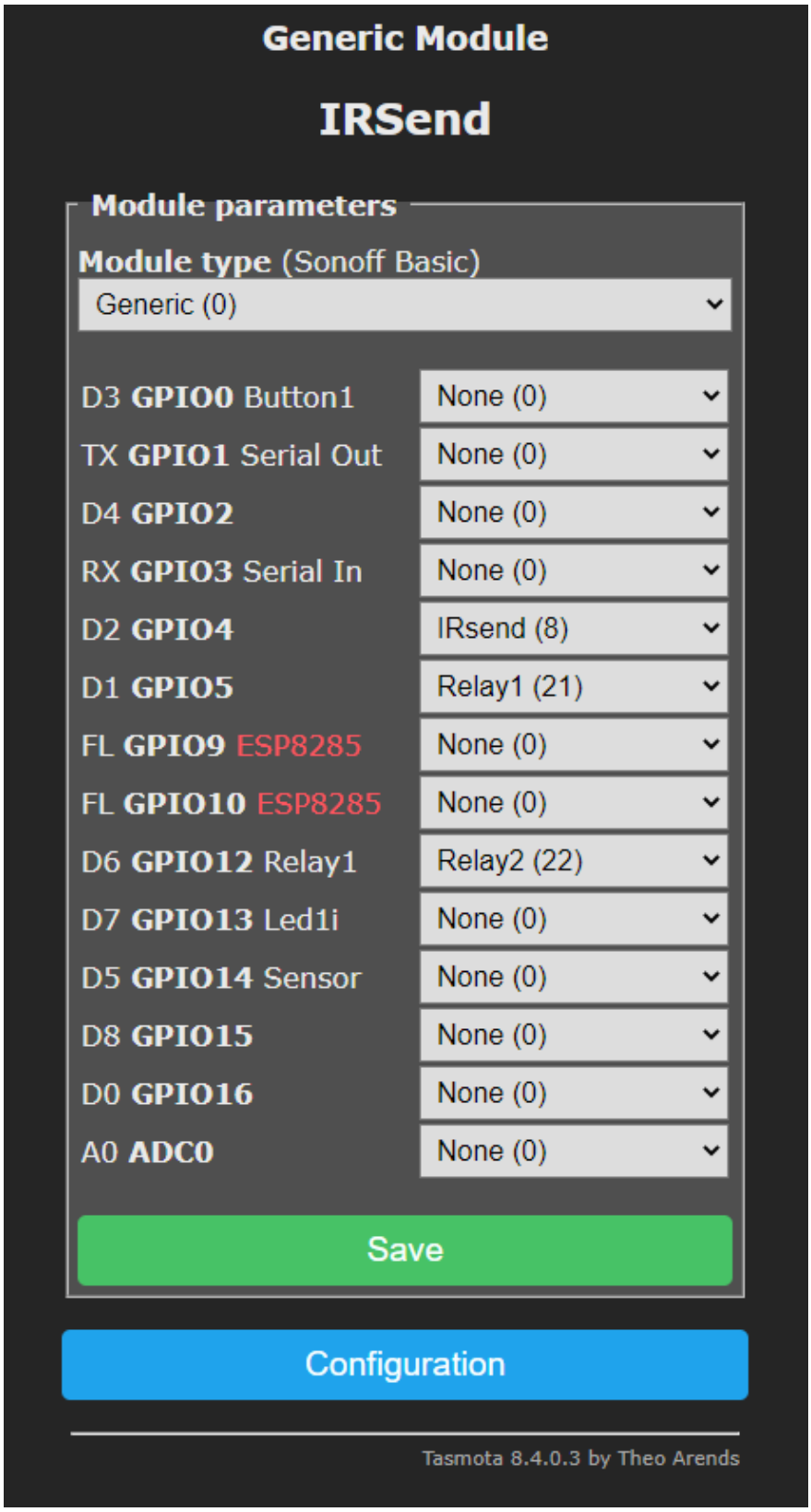


Figure 362 IR Sender Tasmota Configuration

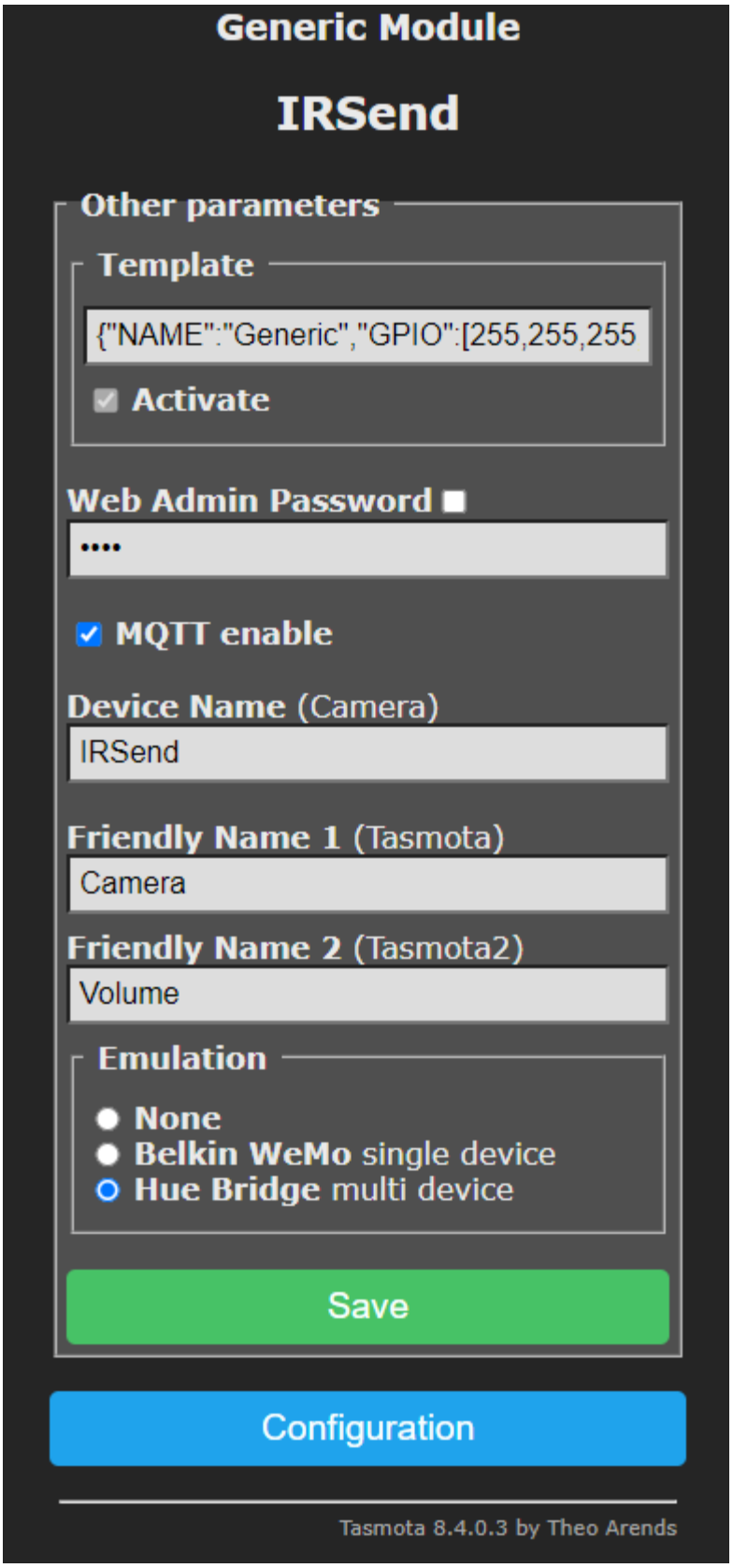


Figure 363 Alexa IR Control Setup

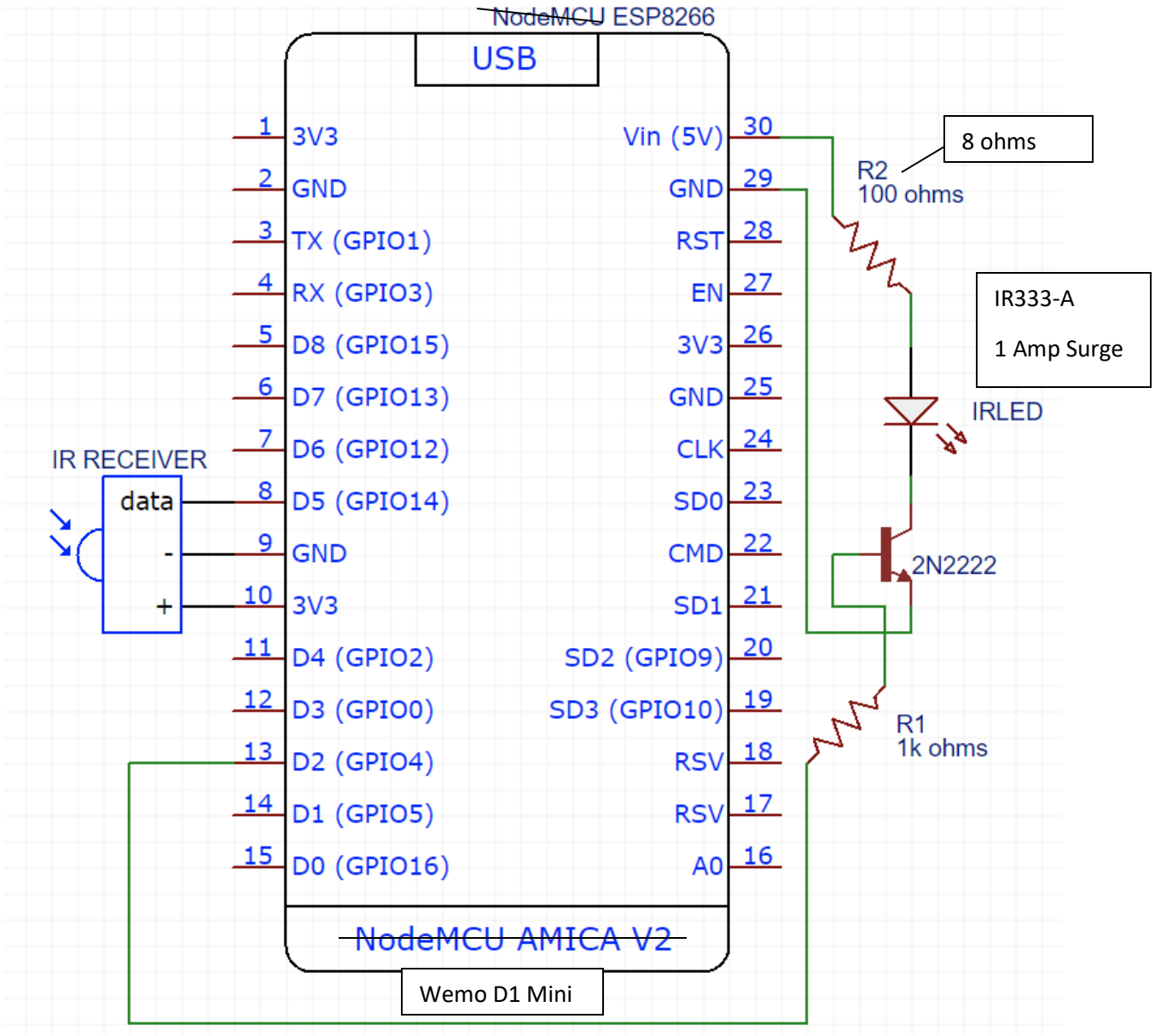


Figure 364 IR Schematic

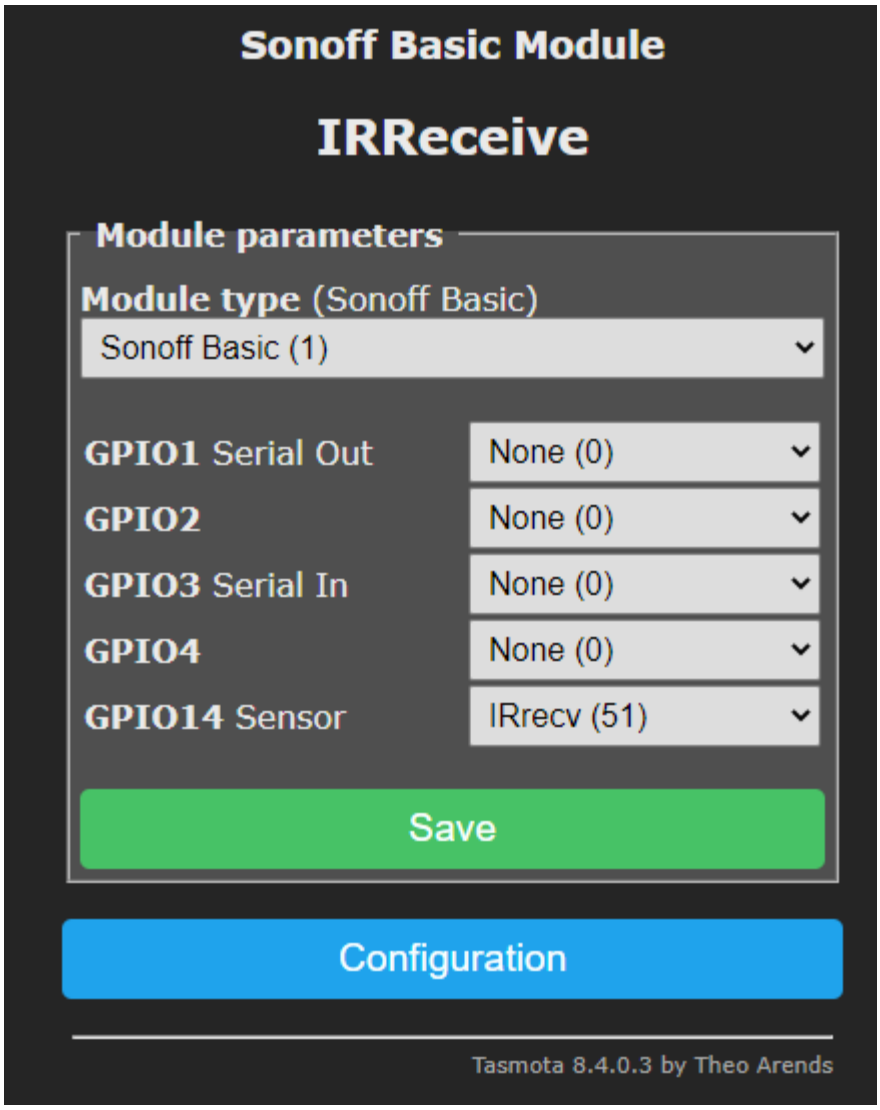


Figure 365 IR Receive Tasmota Configuration

In my case I used a IR Receiver that came with a LED strip and repurposed it for this use. If a new device is needed then it would be a TSOP 4838 or similar that has as three wires. Two for power and ground and the other is the data wire that is connected to the selected GPIO of the ESP8266. In my case if the GPIO14 such as is shown in Figure 364.Presence Detection via Ultrasonic Distance Measurement

The objective of the project is to determine when a vehicle is parked in the garage. The strategy employed is use of a sensor that measures distance to an obstruction. The device that was reused from a prior project is the MaxBiotix EZ1 shown in Figure 366. It can be interfaced via serial or analog. The microcontroller selected is the Wemos D1 Mini. Either interface method could be used, but the analog involved the fewest wires and setup so it was selected.

Figure 366 MaxBotix Ultrasonic Range Sensor

The device was placed inside a 3D-printed case and mounted on the garage wall pointing to the front of the car. Power was supplied by USB power cube plugged into a nearby power outlet.

If the distance becomes greater than 4 ft then the vehicle is no longer in the garage. This corresponds to an analog reading of 200'ish. Normally the front of the car distance is near 100. When out of the garage the distance to the closed garage door reads around 600. Open garage door provides similar larger values with more variability.

Tasmota was installed and configured with ADC for sensor and a virtual relay output to facilitate MQTT reporting of the two states of presence and non-presence as shown in Figure 367.

Stock Tasmota binary that was compiled to allow expression can be used, but a source code modification was made to reduce the amount of MQTT reporting done as rules are evaluated. With this modification the MQTT rules reporting is only done for Tasmota log levels for MQTT of INFO or higher. I set MQTT logging at NONE.

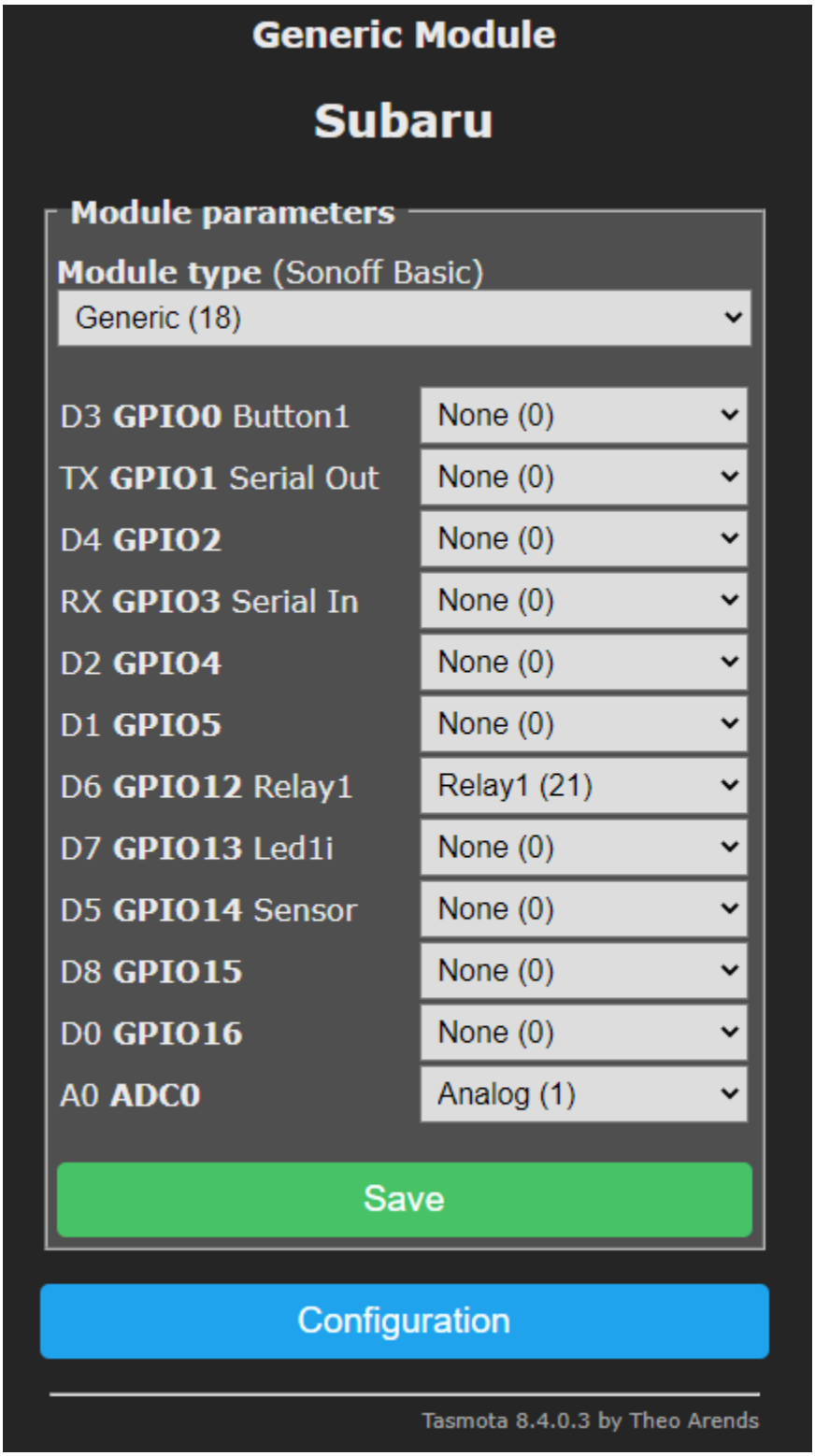


Figure 367 Tasmota Range Finder Sensor Configuration

Rules are used to convert analog distance measurement into one of two states. Two filtering algrorithms were used. One is an averaging and the other is a retriggable oneshot.

Averaging rule has the effect of throwing out spurious measurements and require the preponderance of measurements to exist for more than 60 seconds. This is done with an up/down counter that is run every two seconds. When the up count reaches 31 it changes the relay to the OFF state and limits counting to range of 30 to 0. When it reaches 0 it changes the relay to ON and limits the counting to range 1 to 31. The rule to accomplish this is shown below:

```
rule1 ON system#boot DO backlog var1 1;var2 31;var3 1;var4 0;ruletimer1 1 ENDON
ON Analog#A0div10>20 DO var4=1 ENDON
ON Analog#A0div10<=20 DO var4=-1 ENDON
ON rules#timer=1 DO backlog event CNT=%var1%; event DIST=%var4%;ruletimer1 1 ENDON
ON event#DIST DO var1=%var1%+%var4% ENDON
ON event#CNT>%var2% DO var1=%var2% ENDON
ON event#CNT<%var3% DO var1=%var3% ENDON
ON event#CNT>30 DO backlog var2=29;var1=29;var3=0;power1 0 ENDON
ON event#CNT<1 DO backlog var2=31;var1=2;var3=2;power1 1 ENDON

Rule1 1
```

Tasmota rules are organized around trigger events. The events used in this rule consist of the following

system#boot occurs at startup and in the event the variables are initialized and the periodic timer is started

ruletimer1 is used to generate the sampling interval of two seconds to be used for incrementing or decrementing the up/down count filter. Var1 is used for this counter. The timer is setup for one second, but as setup it occurs every two seconds.

Analog#A0div10 occurs when the ADC has a change of 1% in the range finder reading. Two event triggers are used based upon the reading being above or below the present/away ADC reading. When above the threshold the filter count direction (var4) is set positive. It is set negative when below the threshold.

DIST is raised by ruletimer1 every two seconds and captures the Var4 value. The DIST event is used to increment or decrement Var1 which is the current count of the up/down counter filter. Since Var4 is used directly in the event all that is important is that DIST was raised and the value of DIST is not important. %value% could have been used rather than %var4% in the DIST event expression.

CNT is also raised by ruletimer1 every two seconds. It is used to evaluate it the Var1 count has exceeded a boundry. There are four boundry cases. Two cases limit Var1 so a max (Var2) and min (Var3) are not exceeded by the counter. This assures that the counter is latched at its limits until the distance measurement changes to the other direction. The other two cases are to change the relay when the up/down counter reaches the opposite limit. When the relay position is changed the boundry limits Var2 and Var3 are adjusted so they are setup to detect when the direction of count changes.

The above rule is somewhat complex and is the one that was finally implemented. There was much learning about the event-nature of Tasmota rules and the syntax of ruletimer, variables and expressions so I wanted to capture the results of that learning.

The initial filter used was the retriggable one-shot. This filter requires that no spurious measurements are taken for a span of 30 seconds before a change the state of the relay is made. **Ruletimer1** is used to provide the 30 second time interval. Event **Analog#A0div10** occurs whenever the range sensor reading changes by more than 1%. In this event it evaluates if a false reading occurs based upon the ADC measurement being same as the state of the relay. If a false reading (i.e. reading indicates the relay is in the same position as the distance measurement indicates) then the one-shot is reset. Only after 30 seconds of ADC measurements that all indicate the relay should be in a different position will the relay be changed. Two rules are used. One rule is for the relay in the ON position. The other rule is for it being OFF. The rules toggle as the relay toggles. This could have also been done with variables rather than multiple rules to keep track of the state.

```
Rule1 ON Analog#A0div10<20 DO ruletimer1 30 ENDON
ON rules#timer=1 DO backlog  power1 off; rule2 1; rule1 0 ENDON

Rule2 ON Analog#A0div10>=20 DO ruletimer1 30 ENDON
ON rules#timer=1 DO backlog  power1 on; rule1 1; rule2 0 ENDON

Rule1 1
```

21.24 Sonoff RF and Zigbee Bridges

Itead has released two devices that bridge 433 MHz RF to Wifi and Zigbee to Wifi. Both devices are reflashable and supported by Tasmota. This means that they can be used as RF and Zigbee to MQTT bridges.

Both devices are of the same form factor of 2.5” x 2.5” x 0.75”.

Figure 368 Sonoff Zigbee Bridge

Figure 369 Sonoff RF Bridge

The flashing of both devices is a two-step process. The first is to flash the ESP8266 with Tasmota. The second is to flash the RF/Zigbee radio with compatible firmware. The Sonoff RF is a little more involved. The Sonoff Zigbee was done in about 15 minutes. See https://tasmota.github.io/docs/devices/Sonoff-RF-Bridge-433/ for the RF Bridge. See https://zigbee.blakadder.com/Sonoff_ZBBridge.html for the Zigbee Bridge.

21.24.1 Sonoff RF Bridge

Tasmota configuration of the Sonoff RF Bridge consists on only selecting “Sonoff Bridge (25)” as the device in the Module configuration page. GPIO2,4, 5,12, and 14 show as being available, but not used unless one does additional hacking. Other WiFi and MQTT setup is also necessary.

When a RF transmission data pattern is recognized a MQTT message is delivered which contains a JSON Data key where the encoded signal pattern is found. See Figure 370 for a sample of these messages. The Data field is the unique ID of the sending device.

Message History

/\	p/s	lastdate	topic	payload
0	S	2020-09-25 09:36:41	SonoffRF/RESULT	{"Time":"2020-09-25T09:36:41","RfReceived":{"Sync":14960,"Low":480,"High":1470,"Data":"105055","RfKey":"None"}}
1	S	2020-09-25 09:56:59	SonoffRF/RESULT	{"Time":"2020-09-25T09:56:59","RfReceived":{"Sync":15090,"Low":490,"High":1470,"Data":"105055","RfKey":"None"}}
2	S	2020-09-25 17:55:51	SonoffRF/RESULT	{"Time":"2020-09-25T17:55:51","RfReceived":{"Sync":7740,"Low":210,"High":680,"Data":"097BA8","RfKey":"None"}}
3	S	2020-09-25 17:58:22	SonoffRF/RESULT	{"Time":"2020-09-25T17:58:23","RfReceived":{"Sync":7710,"Low":240,"High":720,"Data":"097BA8","RfKey":"None"}}
4	S	2020-09-25 17:58:45	SonoffRF/RESULT	{"Time":"2020-09-25T17:58:45","RfReceived":{"Sync":14930,"Low":490,"High":1450,"Data":"105055","RfKey":"None"}}
5	S	2020-09-25 21:43:33	SonoffRF/RESULT	{"Time":"2020-09-25T21:43:33","RfReceived":{"Sync":7730,"Low":240,"High":710,"Data":"097BA8","RfKey":"None"}}

Figure 370 Sonoff RF Bridge MQTT Payload

There are two ways to use the received payloads. One is map the received pattern into a more meaningful name as part of the VSP. In this scenario the Control/Status UI should be set to “List”. Edits of the VSP row is done to give meaningful names to the data pattern. In Figure 371 it shows that two of the patterns were mapped into a RoadMotion notification.

The HS device status will be populated with value 2 or 4 for these two and its status will show RoadMotion when these are received. If using the RF as an event trigger then this is an attractive way to setup mcsMQTT so that the data is documented and triggering done based upon the device.

The downside of this approach is that only the last RF reception will be shown in the HS device. If the use of the RF data is to keep track of when the last reception of each code occurs then it is desirable to have one HS device for each RF code. The same triggering can be done on a device-by-device trigger basis, but in this case the trigger will not be the device with a specific value, but will be that the device has been updated. The DeviceValue of the device wil never change. Only the LastChange property will change when new messages are received.

To setup this second approach the Edit tab MQTT Subscribe Topic row, “Check to treat JSON key value as topic” checkbox is used. See Section 4.1.36 for a more complete description. This will result in every Data value being a separate row on the Association tab table. The onces of interest can then be associated with a HS device.

An application that was recently implemented is use of the QIACHIP, which is the guts of the four-button RF keyfobs, to provide notification when a dry contact has been activated from an alarm. The QIACHIP was described in Section21.17.2.

The QIACHIP provides an encoded 433 MHz message when one of its four inputs is grounded. It accepts power in the 5V to 24V range. A nice feature of the Dakota receiver to which it was connected is that it provides a 12V output that is active only after the motion alarm is triggered. I set it up for a 1 second active after alarm and used it to power the QIACHIP. One of the four Dakota relay outputs was connected to one of the four QIACHIP inputs. The result is that the message is sent at 433 MHZ when the motion notification is triggered and the Sonoff RF recognizes the transmission. The bottom line is that the tiny QIACHIP circuit card can be mounted inside the Dakota receiving station, no external power needed, and the motion notification is available on the network via MQTT.

Edit Setup or Edit of Subscription (Inbound) to a MQTT Topic

MQTT Subscribe Topic	SonoffRF/RESULT:RfReceived:Data ☐ Check to treat JSON key value as topic
Payload RegEx Match Pattern	
Payload RegEx Replace Pattern	
Payload RegEx Operation	◉ Replace Match Pattern with Replace Pattern ○ Extract Match Pattern
Low Pass Filter	Filter sensitivity of 1 (range is 0.00 to 1.00 (most sensitive))
Expression	
Add Rate Device	☐ Create a HS Rate Device with rate sensitivity of 0.75 (Range 0.00 to 1.00) ○ Per Second ○ Per Minute ◉ Per Hour
Add Accum Device	☐ Create a HS Accum Device ○ No Reset ○ Accumulation Since Midnight ◉ Delta Since Midnight
Store Payload	◉ In HS Device Value ○ In HS Device String

Settings for Plugin Device

HS Device Publish Topic	
HS Device Control/Status UI	○ Unspecified ○ Button ○ Number ○ NumberChange ○ Slider ○ Ramp ○ CSV ○ Text ◉ List ○ RGB ○ RGBW ○ HSB ○ ColorXY ○ Sign
HS Device Location	Loc2 (Floor) SonoffRF Loc (Room) RESULT Name RESULT:RfReceived:D
HS Device VSP List	Payload 18CD71=0;18CD71 VSP Payload 097BA4=1;097BA4 VSP Payload 097BA8=2;RoadMotion VSP Payload 097BA1=3;097BA1 VSP Payload 105055=4;RoadMotion VSP Payload 800722=5;800722 VSP Payload 08282A=6;08282A VSP Payload 12F750=7;12F750 VSP Payload 107055=8;107055 VSP Payload 1050D5=9;1050D5 VSP Clear existing VSP
HS Device MISC Properties	☐ NO_STATUS_DISPLAY ☐ NO_GRAPHICS_DISPLAY ☐ AUTO_VOICE_COMMAND ☐ SET_DOES_NOT_CHANGE_LAST_CHANGE ☑ SHOW_VALUES ☐ STATUS_ONLY
Grouping Parent Ref	861 Create New Parent Device
Publish Payload Template	
URI Encode Payload	☐ Encode Special Characters
Publish QOS	◉ At Most ○ At Least ○ Exactly
Publish Retain Flag	◉ Do not retain ○ Retain at broker

Figure 371 Sonoff RF VSP Capture

21.24.2 Sonoff Zigbee Bridge

The Sonoff Zigbee Bridge is a new device and the Tasmota Zigbee support is in the experimental classification. Its list of supported devices is smaller than the more mature Zigbee2MQTT (See Section 19) mechanism that uses a RPi or another general-purpose computer rather than the ESP8266 in the Tasmota implementation.

I would consider it to be a niche solution for cases where a Zigbee device is out of range of the Zigbee coordinator, yet WiFi coverage does exist. It would also suit very well for the situations where the location does not already have a Zigbee coordinator and network, but one desires to use a limited number of Zigbee devices.

Pairing is enabled from Tasmota console or MQTT message `ZbPermitJoin 1 MQTT` reporting such as shown in Figure 372 shows the success of the pairing and information about the device that was paired in the /RESULT topic.

54	S	2020-09-25 21:24:58	TasmotaZigbee/RESULT	{"ZbState":{"Status":34,"IEEEAddr":"0x00158D0002130C6F","ShortAddr":"0x89AE","ParentNetwork":"0x0000","Status":1,"Decision":0}}
55	S	2020-09-25 21:24:58	TasmotaZigbee/RESULT	{"ZbState":{"Status":30,"IEEEAddr":"0x00158D0002130C6F","ShortAddr":"0x89AE","PowerSource":false,"ReceiveWhenIdle":false,"Security":false}}
56	S	2020-09-25 21:24:58	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x89AE":{"Device":"0x89AE","ModelId":"lumi.sensor_wleak.aq1","AppVersion":4,"Endpoint":1,"LinkQuality":139}}}
57	S	2020-09-25 21:25:00	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x89AE":{"Device":"0x89AE","BatteryVoltage":3.01,"BatteryPercentage":100,"Xiaomi_64":0,"Endpoint":1,"LinkQuality":87}}}
58	S	2020-09-25 21:25:00	TasmotaZigbee/RESULT	{"ZbState":{"Status":32,"ActiveEndpoints":["0x01"]}}

Figure 372 Zigbee Tasmota Discovery Reporting

Periodic reporting occurs in the Tasmota /STATE topic about the ESP8266. Reporing intiated by the Zigbee device is in the /SENSOR topic. In this case of a Xiaomi Window/Door sensor it provides a periodic battery status as shown in row 191 of Figure 372. In rows 199 and 200 the reporting of the contact beng made and open. Note it is the Power key in the JSON message that reflects the sensor state. In essence Tasmota treats it like relay reporting.

191	S	2020-09-26 14:12:25	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x9E0A":{"Device":"0x9E0A","BatteryVoltage":3.05,"BatteryPercentage":100,"Xiaomi_64":1,"Endpoint":1,"LinkQuality":181}}}
192	S	2020-09-26 14:15:02	TasmotaZigbee/STATE	{"Time":"2020-09-26T14:15:02","Uptime":"0T16:55:09","UptimeSec":60909,"Vcc":3.480,"Heap":28,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"MqttCount":1,"Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":6,"RSSI":88,"Signal":-56,"LinkCount":1,"Downtime":"0T00:00:03"}}
193	S	2020-09-26 14:20:02	TasmotaZigbee/STATE	{"Time":"2020-09-26T14:20:02","Uptime":"0T17:00:09","UptimeSec":61209,"Vcc":3.480,"Heap":28,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"MqttCount":1,"Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":6,"RSSI":92,"Signal":-54,"LinkCount":1,"Downtime":"0T00:00:03"}}
194	S	2020-09-26 14:25:02	TasmotaZigbee/STATE	{"Time":"2020-09-26T14:25:02","Uptime":"0T17:05:09","UptimeSec":61509,"Vcc":3.480,"Heap":28,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"MqttCount":1,"Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":6,"RSSI":88,"Signal":-56,"LinkCount":1,"Downtime":"0T00:00:03"}}
195	S	2020-09-26 14:30:02	TasmotaZigbee/STATE	{"Time":"2020-09-26T14:30:02","Uptime":"0T17:10:09","UptimeSec":61809,"Vcc":3.480,"Heap":28,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"MqttCount":1,"Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":6,"RSSI":86,"Signal":-57,"LinkCount":1,"Downtime":"0T00:00:03"}}
196	S	2020-09-26 14:35:02	TasmotaZigbee/STATE	{"Time":"2020-09-26T14:35:02","Uptime":"0T17:15:09","UptimeSec":62109,"Vcc":3.480,"Heap":28,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"MqttCount":1,"Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":6,"RSSI":84,"Signal":-58,"LinkCount":1,"Downtime":"0T00:00:03"}}
197	S	2020-09-26 14:40:02	TasmotaZigbee/STATE	{"Time":"2020-09-26T14:40:02","Uptime":"0T17:20:09","UptimeSec":62409,"Vcc":3.480,"Heap":28,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"MqttCount":1,"Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":6,"RSSI":80,"Signal":-60,"LinkCount":1,"Downtime":"0T00:00:03"}}
198	S	2020-09-26 14:45:02	TasmotaZigbee/STATE	{"Time":"2020-09-26T14:45:02","Uptime":"0T17:25:09","UptimeSec":62709,"Vcc":3.480,"Heap":28,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"MqttCount":1,"Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":6,"RSSI":86,"Signal":-57,"LinkCount":1,"Downtime":"0T00:00:03"}}
199	S	2020-09-26 14:48:36	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x9E0A":{"Device":"0x9E0A","Power":0,"Endpoint":1,"LinkQuality":171}}}
200	S	2020-09-26 14:48:39	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x9E0A":{"Device":"0x9E0A","Power":1,"Endpoint":1,"LinkQuality":176}}}

Figure 373 Tasmota Zibgee Event Reporting for Window/Door Sensor

A comparision of reporting for the Xiaomi water leak sensor is shown in Figure 374. Note the battery report is similar and rather than Power, it uses Occupancy as the key to report a water leak. It obviously believes this is a motion sensor and not a water leak sensor. Being in experimental status these types of anomalies should be expected.

1	S	2020-09-25 21:25:00	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x89AE": {"Device":"0x89AE","BatteryVoltage":3.01,"BatteryPercentage":100,"Xiaomi_64":0,"Endpoint":1,"LinkQuality":87}}}
2	S	2020-09-25 21:25:03	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x89AE": {"Device":"0x89AE","Manufacturer":"LUMI","ModelId":"lumi.sensor_wleak.aq1","Endpoint":1,"LinkQuality":147}}}
3	S	2020-09-25 21:26:20	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x89AE": {"Device":"0x89AE","0500<00":"010000FF0000","ZoneStatusChange":1,"Occupancy":1,"Endpoint":1,"LinkQuality":84}}}
4	S	2020-09-25 21:26:27	TasmotaZigbee/SENSOR	{"ZbReceived":{"0x89AE": {"Device":"0x89AE","0500<00":"000000FF0000","ZoneStatusChange":0,"Occupancy":0,"Endpoint":1,"LinkQuality":121}}}

Figure 374 Tasmota Zigbee Event Reporting for Water Leak Sensor

There is no configuration needed for the ESP8266 with the binary that was provided from the Tasmota site other than customizations for WiFi SSID, MQTT broker and preferred name. The main Tasmota page reporting is shown in Figure 375.

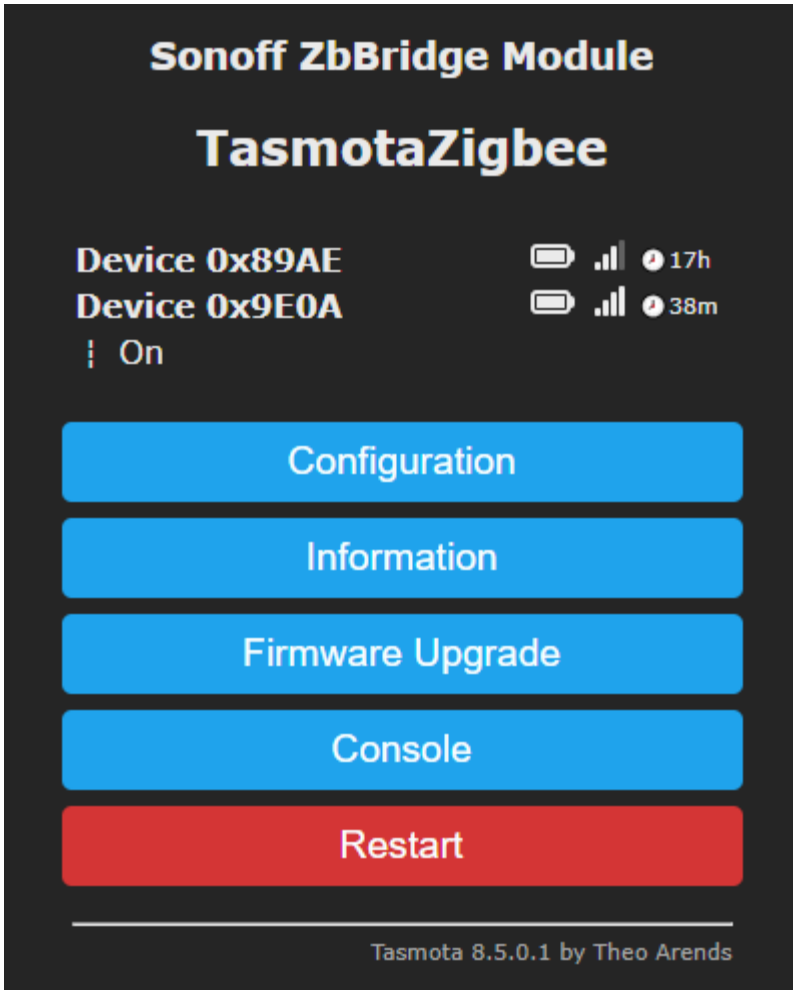


Figure 375 Tasmota Zigbee Status Page

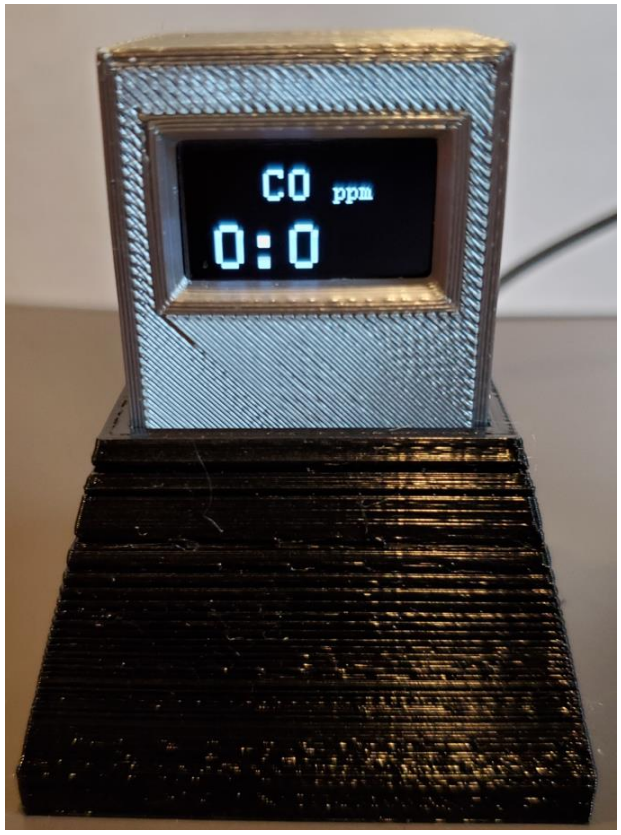
The Tasmota command ZbName can be used to change 0x123245 to a friendly name for easier recognition. (e.g. ZbName 0x89AE,WaterLeak). This is similar to changing the .yaml file for zigbee2mqtt.

It is nice to see a graphic display of status for battery, wifi and uptime. To my knowledge it is not something that exits in other Tasmota binaries. It is not clear why only one “On” status is shown as both devices have binary status.

Prior to using the Sonoff Zigbee Bridge, I used the same coordinator hardware that was used for Zigbee2MQTT. I tried both CC2530 and CC2531. Initially without reflashing the RF radio firmware from what had been successfully used with Zigbeen2MQTT and then reflashing based upon the latest information in the Tasmota Zigbee Wiki.

Since I had flashed many of these Zigbee devices in the past I felt comfortable with the flashing again, but I was never able to achieve success when integrated with a Zigbee-compiled version of Tasmota. In all cases I would get the report that Tasmota was not able to start the radio.

21.25 Carbon Monoxide Detector



Carbon Monoxide sensing is performed with ZE16B sensor that has technical description at https://www.winsen-sensor.com/d/files/ZE16B-CO.pdf. It is sold by multiple suppliers including Aliexpress https://www.aliexpress.com/item/1005002233280950.html?spm=a2g0s.9042311.0.0.52354c4dA9SWef where quantity 5 was obtained for under $20. Like many gas sensors their effectiveness fades over time so the recommendation is to replace the sensor every year or two.

The interface provided is a 5VDC UART at 9600 baud. Tasmota does support many serial-based sensors, but as of version 9.5 it does not support this one. To accommodate this sensor the code for the HXHL distance sensor was replace with code for the ZE16B. The sensor transmits continuously every second a nine-byte message of the format shown below. Bytes 4 and 5 contain the reading and Bytes 1 through 7 are used to compute a checksum against which a comparision is made for equality with Byte 8 to assure the data is valid. The source code is available at http://mcsSprinklers.com/ZE16B.zip

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
Start Byte	Gas Type	Unit	No. of decimal	Concentration (High Byte)	Concentration (Low Byte)	Full Range (High Byte)	Full Range (Low Byte)	Check sum
0xFF	CO=0x04	ppm=0x03	0 byte=0x00	0x01	0x2C	0x01	0xF4	0xD7

As part of the packaging a 0.96” LCD displayed was added to display the carbon monoxide locally. A BMP280 was also added which provides pressure measurement as well as temperature measurement. The temperature is inside-case temperature so is not reflective of ambient room temperature. https://www.amazon.com/gp/product/B08LYL7QFQ/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1 It is also provided via WiFi using MQTT. A picture of the sensor and display during bench testing is shown in Figure 376.

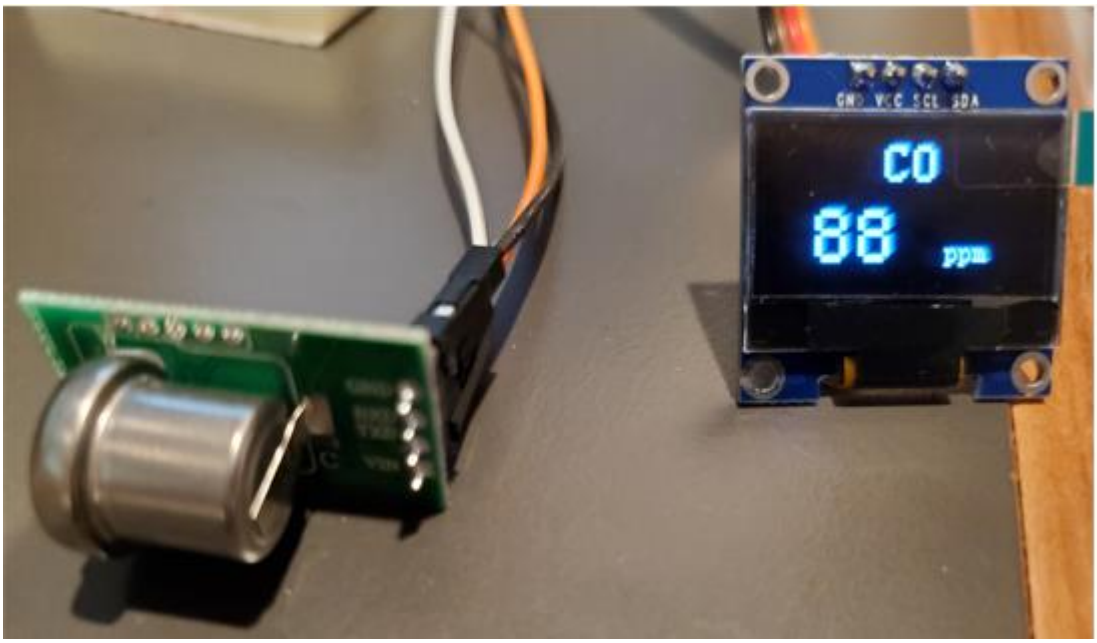


Figure 376 ZE16B CO Sensor and LCD During Bench Testing

The normal reading for the sensor is 0 ppm. To excite the sensor for testing a match was used to generate smoke that had some CO composition. It was several seconds after exposure to the stimulant before the readings moved off of zero.

The LCD has a rated life of about 1.5 years if fully illuminated. I suspect that with lower brightness levels and also the relatively sparse text being displayed this lifetime could be much longer. The approach taken with this implementation is to turn off the display unless a non-zero CO level is being reported. After power-up the LCD will be showing the CO level for 60 seconds so there is some feedback to know the display is functional (i.e., power on selftest). Rules in Tasmota were used to achieve this display management logic.

Two Rules were employed. Rule2 handles the power-up, reporting of CO level changes and formatting for the display. Rule3 handles events to detect CO level changes and when the display should be turned on and off.

var1 - format for the display with CO and ppm at the top in size 2 and 1 fonts
var2 - format for the display of the current and max CO readings in size 3 font
var3 - current CO reading
var4 - max CO reading
ruletimer1 - event to mark 2 seconds after boot. On expiration it turns on the display

ruletimer2 - event to mark 60 seconds after a CO level of 0 has been detected so display can be turned off
power – display power control

Rule 2 narrative

On boot initialize variables and timers then turn on rule3. When CO level becomes 0 then set timer to turn off display in 60 seconds as well as capture the current CO level in var3. If CO level is greater than max then update the max in var4. If the CO level changes, then update display and publish the current and max values. Assure display is on.

```
rule2 on System#Boot do backlog
        var1 [zs2x30y1]CO[s1x70y10]ppm;
        var2[s3x1y30];
        var3 0;
        var4 0;
        rule2 5;
        rule3 4;
        ruletimer1 2;
        ruletimer2 60;
        power 1;
        rule3 1 endon
    on ZE16B#CO<1 do backlog
        var3 0;
        ruletimer2 60 endon
    on ZE16B#CO>%var4% do var4 %value% endon
    on ZE16B#CO!=%var3% do backlog
        DisplayText %var1%%var2%%value%:%var4%;
        publish %topic%/SENSOR {"ZE16B":{"CO":%value%,"COMax":%var4%}};
        power 1 endon
```

Rule 3 narrative

On system boot turn off rule3. It will be turned on after rule2 completes the setup. Rule3 is setup to evaluate continuously and puts the current CO reading in var3. It also contains the logic for when the two timers expire. Timer1 to show CO level on display two seconds after startup. Timer2 to turn off power to display after one minute following power-up or when CO level returns to 0.

```
rule3 on System#Boot do rule3 0 endon
    on ZE16B#CO>0 do var3 %value% endon
    on rules#Timer=2 do power 0 endon
    on rules#Timer=1 do backlog
        DisplayText %var1%%var2%%var3%:%var4%;
        publish %topic%/SENSOR {"ZE16B":{"CO":%value%,"COMax":%var4%}};
        endon
```

Rules cut/paste

rule2 on System#Boot do backlog var1 [zs2x30y1]CO[s1x70y10]ppm;var2[s3x1y30];var3 0;var4 0;rule2 5;rule3 4;ruletimer1 2;ruletimer2 60;Power 1;rule3 1 endon on ZE16B#CO<1 do backlog var3 0;ruletimer2 60 endon on ZE16B#CO>%var4% do var4 %value% endon on ZE16B#CO!=%var3% do backlog DisplayText %var1%%var2%%value%:%var4%;publish %topic%/SENSOR {"ZE16B":{"CO":%value%,"COMax":%var4%}};power 1 endon

rule3 on System#Boot do rule3 0 endon on ZE16B#CO>0 do var3 %value% endon on rules#Timer=2 do power 0 endon on rules#Timer=1 do backlog DisplayText %var1%%var2%%var3%:%var4%; publish %topic%/SENSOR {"ZE16B":{"CO":%value%,"COMax":%var4%}} endon

Other Tasmota configuration

gpio4 640	IC SDA for LCD and pressure/temperature sensor
gpio5 608	IC SCL for LCD and pressure/temperature sensor
gpio12 4608	ZE16B sensor serial input on GPIO12
timezone -7	Pacific timezone
setoption8 1	Use F for temperature
rule2 1	Enable initial rule

backlog gpio4 640;gpio5 608;gpio12 4608;timezone -7;setoption8 1;rule2 1

The Tasmota configuration uses SDA and SCL pins for the LCD and BMP280 and GPIO12 for the Software Serial UART. The Software Serial was used to free up the Hardware Serial (GPIO3) for development testing

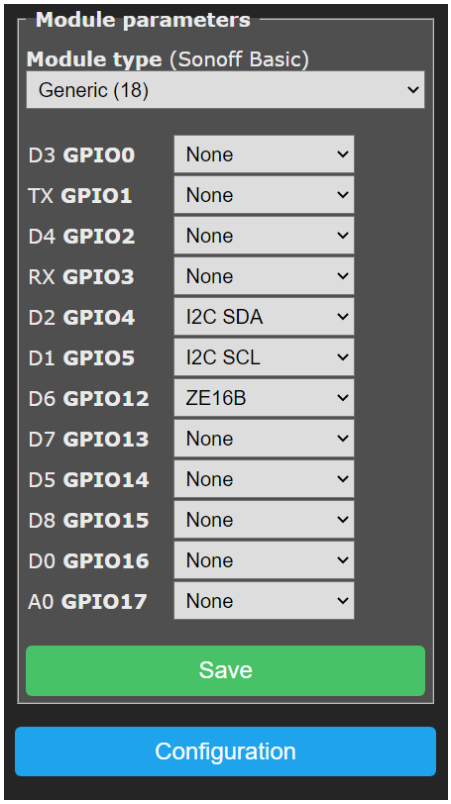
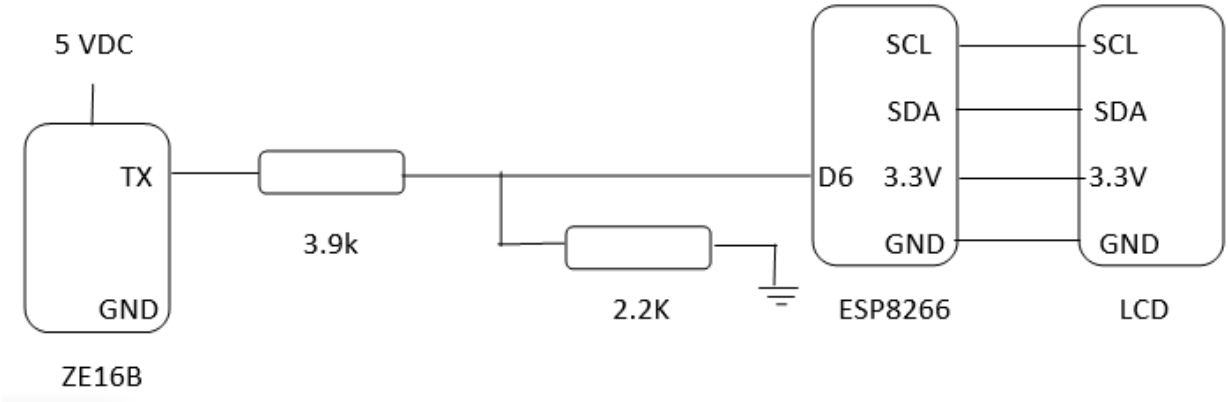


Figure 377 Tasmota Module Configuration for CO Sensing

The ZE16B sensor's specified voltage is 5VDC. It swings its UART to the rails so would produce too high a voltage for the ESP8266 GPIO12. A level shifter could be used, but a simplier solution is a voltage divider connected as shown below.



Periodic reporting is setup on the Logging page of Tasmota to be 300 seconds (5 minutes). The base Topic for the device was setup to be CO on the Tasmota MQTT page. The MQTT Broker IP address was also setup on the Tasmota MQTT page. The data reported are two JSON payloads such as:

CO/STATE = {"Time":"2021-10-15T11:35:29","Uptime":"0T00:10:09","UptimeSec":609,"Heap":28,"SleepMode":"Dynamic","Sleep":50,"LoadAvg":19,"MqttCount":1,"POWER":"OFF","Wifi":{"AP":1,"SSId":"U","BSSId":"78:8A:20:84:48:1D","Channel":11,"Mode":"11n","RSSI":64,"Signal":-68,"LinkCount":1,"Downtime":"0T00:00:03"}}

CO/SENSOR = {"Time":"2021-10-15T11:35:29","BMP280":{"Temperature":89.0,"Pressure":1024.0},"ZE16B":{"CO":0},"PressureUnit":"hPa","TempUnit":"F"}

The information of interest is the CO level, max CO level, and potentially the pressure. These can be associated to HS devices on the Association tab such as shown in Figure 378. The default reporting for pressure is hPa. The Edit tab of MQTT page in mcsMQTT can use the expression textbox to convert to other units such as $$PAYLOAD:* 0.02953 to get to inHg units.

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
0	☐	☐	☐		1197	CO/SENSOR					
1	☐	☐	☐	✓	1199	Dev: CO|SENSOR|SENSOR:\ZE16B\:\CO\ Sub: CO/SENSOR:\ZE16B\:\CO\ Pub: the following Topic on Device command	0	☐	☐	☐	2021-10-08 11:36:44
2	☐	☐	☐	✓	1198	Dev: CO|SENSOR|SENSOR:BMP280:Pressure Sub: CO/SENSOR:BMP280:Pressure Pub: the following Topic on Device command	1024.1	☐	☐	☐	2021-10-15 11:53:57
3	☐	☐	☐	☐		Sub: CO/SENSOR:BMP280:Temperature	92.1	☐			2021-10-15 11:53:57
4	☐	☐	☐	☐		Sub: CO/SENSOR:PressureUnit	hPa	☐			2021-10-15 11:53:57
5	☐	☐	☐	☐		Sub: CO/SENSOR:TempUnit	F	☐			2021-10-15 11:53:57
6	☐	☐	☐	☐		Sub: CO/SENSOR:Time	2021-10-15T11:53:57	☐			2021-10-15 11:53:57
7	☐	☐	☐	☐		Sub: CO/SENSOR:ZE16B:CO	1	☐			2021-10-15 11:53:57
8	☐	☐	☐	✓	1200	Dev: CO|SENSOR|SENSOR:ZE16B:COMax Sub: CO/SENSOR:ZE16B:COMax Pub: the following Topic on Device command	0	☐	☐	☐	2021-10-15 11:53:57

Figure 378 CO Sensor Associations

22 SDR and RTL-433

SDR is then handle for Softwrae Defined Radio which is a flexible solution of a generic RF receiver for which tuning parameters are controlled by software. Diagnostic and decoding tools have been built around the SDR including SDR Sharp for general viewing and RTL-433 for decoding digital data. In the RTL-433 case the decoded data can be put on the network using MQTT protocol.

Figure 379 Typical SDR Hardware Dongle

The software can be run on either Windows or Linux. A ground-up RPi install is described below where the MQTT component is the Mosquitto client. A Python MQTT client, and likely others, is also available

```
Build and install rtl_sdr
-------------------------
cd ~
sudo apt install pkg-config
sudo apt-get install git git-core cmake libusb-1.0-0-dev build-essential
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
sudo mkdir build
cd build
sudo cmake ../ -DINSTALL_UDEV_RULES=ON
sudo make
sudo make install
```

```
sudo ldconfig
cd ~
sudo cp ./rtl-sdr/rtl-sdr.rules /etc/udev/rules.d/
sudo reboot
# create file no-rtl.conf

sudo nano /etc/modprobe.d/no-rtl.conf
# add these three lines

blacklist dvb_usb_rtl28xxu
blacklist rtl2832
blacklist rtl2830

sudo apt-get install doxygen
sudo reboot
lsusb
rtl_test -t

Build and install rtl_433
----------------------------
sudo apt-get install libtool libusb-1.0.0-dev librtlsdr-dev
#rtl_sdr doxygen
git clone https://github.com/merbanan/rtl_433.git
cd rtl_433
sudo mkdir build
cd build
sudo cmake ../
sudo make
sudo make install

Install mosquito client
------------------------
sudo apt-get install -y mosquitto mosquitto-clients
```

As an alternate to Mosquitto client the following Python script can be installed. This is not needed if Mosquitto client is installed.

```
Install python3/pip
-------------------------
sudo apt-get update
sudo apt-get -y install python3-pip

Get rtl433-to-mqtt script
---------------------------------
https://github.com/mverleun/RTL433-to-mqtt (download zip or clone code).  I
placed mine in the same folder as rtl_433
Extract the file config.py.example to config.py and setup MQTT environment.
Extract rtl2mqtt.py
Execute in terminal window with "phython3 rtl2mqtt.py"
```

Using the Python script a MQTT set of data will look like Figure 380 for the case of a RF Remote button push.

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	d	lastdate
0	☐	☐	☐	☐		rtl_433				
1	☐	☐	☐	☐		Sub: rtl_433/Smoke-GS558/30104/code	1eb318	☐		2020-09-28 13:43:22
2	☐	☐	☐	☐		Sub: rtl_433/Smoke-GS558/30104/id	30104	☐		2020-09-28 13:43:22
3	☐	☐	☐	☐		Sub: rtl_433/Smoke-GS558/30104/learn	0	☐		2020-09-28 13:43:22
4	☐	☐	☐	☐		Sub: rtl_433/Smoke-GS558/30104/time	2020-09-28 13:43:21	☐		2020-09-28 13:43:22
5	☐	☐	☐	☐		Sub: rtl_433/Smoke-GS558/30104/unit	24	☐		2020-09-28 13:43:22
6	☐	☐	☐	☐		Sub: rtl_433:code	1eb318	☐		2020-09-28 13:43:22
7	☐	☐	☐	☐		Sub: rtl_433:id	30104	☐		2020-09-28 13:43:22
8	☐	☐	☐	☐		Sub: rtl_433:learn	0	☐		2020-09-28 13:43:22
9	☐	☐	☐	☐		Sub: rtl_433:model	Smoke-GS558	☐		2020-09-28 13:43:22
10	☐	☐	☐	☐		Sub: rtl_433:time	2020-09-28 13:43:21	☐		2020-09-28 13:43:22
11	☐	☐	☐	☐		Sub: rtl_433:unit	24	☐		2020-09-28 13:43:22

Figure 380 Python Script MQTT Message Content

When using Mosquitto client the following command is used to start RTL_433 where the blue font will be customization based upon your local environment and preference for start of the topic. The green font is the frequency being scanned. While shown as three lines below it is a single command line.

```
rtl_433 -f 310.0M -F
mqtt://192.168.0.30:1883,retain=0,devices=SDR[/type][/model][/subtyp
e][/channel:0][/id]
```

X10 RF at 310 Mhz is not normally decoded, but can be enabled in one of two ways. One is specify only specific protocols using the –R 22 key in the start command. Using this approach additional –R keys will need to be added if other protocols at 310 MHz are also going to be decoded. See list at https://github.com/merbanan/rtl_433. The other approach is to edit the source before the make/build so that X10 RF is enabled by default. In the RTL_433/srd/devices folder will be the file x10_rf.c and at the bottom will be the ".disabled" property that should be changed from 1 to 0 to enable it as a default.

```
r_device X10_RF = {
.name = "X10 RF",
.modulation = OOK_PULSE_PPM,
.short_width = 500, // Short gap 500µs
.long_width = 1680, // Long gap 1680µs
.gap_limit = 2800, // Gap after sync is 4.5ms (1125)
.reset_limit = 6000, // Gap seen between messages is ~40ms so let's get them
individually
.decode_fn = &x10_rf_callback,
.disabled = 0,
.fields = output_fields,
};
```

Multiple topics will be transmitted for each RF decode event and it is likely that the same set of messages will be repeated multiple times as digital RF transmissions are often repeated to increase reception success. An example of the X10 button and of two keychain remote buttons is shown below. Those that are likely of interest to map into HS devices are shown in red font.

`X10 Palmpad B14 ON`

Received Topic: SDR/X10-RF/B/14/time	Payload: 2020-10-02 11:30:06
Received Topic: SDR/X10-RF/B/14/id	Payload: 14
Received Topic: SDR/X10-RF/B/14/channel	Payload: B
Received Topic: SDR/X10-RF/B/14/state	Payload: ON
Received Topic: SDR/X10-RF/B/14/data	Payload: -534839041

Remote B Button

Received Topic: SDR/Generic-Remote/0/6349/time	Payload: 2020-10-02 11:58:10
Received Topic: SDR/Generic-Remote/0/6349/id	Payload: 6349
Received Topic: SDR/Generic-Remote/0/6349/cmd	Payload: 114
Received Topic: SDR/Generic-Remote/0/6349/tristate	Payload: 0ZX0101ZZ10X

Remote A Button

Received Topic: SDR/Generic-Remote/0/6349/time	Payload: 2020-10-02 11:59:42
Received Topic: SDR/Generic-Remote/0/6349/id	Payload: 6349
Received Topic: SDR/Generic-Remote/0/6349/cmd	Payload: 113
Received Topic: SDR/Generic-Remote/0/6349/tristate	Payload: 0ZX0101ZZ10Z

The X10 palmpad was recorded when tuned to 310 Mhz. The keychain remote was at 433.92 Mhz. This means either two SDR dongles are needed to do it simultaneously. An attempt was made to hop between the two frequencies at the fastest rate possible, but this resulted in missing a high percentage of the RF transmissions

```
rtl_433 –f 310.0M –f 433.92M –H 1s -F
mqtt://192.168.0.30:1883,retain=0,devices=SDR[/type][/model][/subtype]
[/channel:0][/id]
```

A more concise 433.92 MHz decode is like

```
rtl_433 -F mqtt://192.168.0.16:1883,retain=0,devices=SDR[/type][/model][/subtype]
```

If one desires to scan multiple frequencies then multiple SDR will be needed and multiple instances or rtl_433 will be started. The SDR dongle is a little larger than a typical USB dongle which means that two cannot be plugged directly into the RPi. A USB extension cable can be used for the second dongle. The "d" parameter is used when starting rtl_433 to identify which SDR dongle is being used. For example, to run both 433 and 310 then the following would be used:

```
rtl_433 -d 0 -F mqtt://192.168.0.16:1883,retain=0,devices=SDR[/type][/model][/subtype]

rtl_433 -f 310.0M -d 1 -F
mqtt://192.168.0.16:1883,retain=0,devices=SDR[/type][/model][/subtype]
```

For the Ecowitt 915 MHz on the second SDR interface it would be

```
rtl_433 -d 1 -s 250K -F
mqtt://192.168.0.16:1883,retain=0,devices=SDR[/type][/model][/subtype]
-f 915M
```

Systemd can be used to run rtl_433 at boot on Linux with something like the following:

The rtl_433.service file that is located in /etc/systemd/system contains:

```
[Unit]
Description=Decoding of SDR 433 Mhz devices
After=network.target

[Service]
ExecStart=/home/pi/rtl_433/autostart_rtl433
Restart=on-failure
TimeoutStopSec=90

[Install]
WantedBy=multi-user.target
```

autostart_rtl433 fie located at `/home/pi/rtl_433/` is made executable and contains:

```
#!/bin/sh
export LANG=en_US.UTF-8
cd /home/pi/rtl_433/build
sleep 10s
rtl_433 -F mqtt://192.168.0.30:1883,retain=0,devices=SDR[/type][/model][/subtype]
```

One time command to enable it

```
sudo systemctl enable rtl_433.service
```

If two frequencies are to be run with two dongles then a second service can be setup such as below for 910 MHz:

The second rtl_910.service file that is located in /etc/systemd/system contains:

```
[Unit]
Description=Decoding of SDR 910 Mhz devices
After=network.target

[Service]
ExecStart=/home/pi/rtl_433/autostart_rtl433_910
Restart=on-failure
TimeoutStopSec=90

[Install]
WantedBy=multi-user.target
```

autostart_rtl433_910 fie located at `/home/pi/rtl_433/` is made executable and contains:

```
#!/bin/sh
export LANG=en_US.UTF-8
cd /home/pi/rtl_433/build
sleep 10s
rtl_433 -f 910M -d 1 -F
mqtt://192.168.0.30:1883,retain=0,devices=SDR[/type][/model][/subtype]
```

One time command to enable it

```
sudo systemctl enable rtl_910.service
```

23 Pentair Pool Controller Integration

There are multiple 3rd party implementations to automate the Pentair pool controller. This section describes the one that runs on a RPi with a USB/RS-485 adapter running nodejs with extensions for MQTT. It is described at https://www.troublefreepool.com/thre...-guide.218514/. The gihub link is https://github.com/tagyoureit/nodejs-poolController and the MQTT extension is enabled per https://github.com/tagyoureit/nodejs...ns-in-2.0#mqtt.

Pool controller is an integration of nodejs-poolController with HS. Nodejs-poolController is the glue between Pentair pool equipment using Intellicenter, Intellitouch and EasyTouch or standalone equipment. It is described at tagyoureit/nodejs-poolController: An application to control pool equipment from various manufacturers. (github.com). The physical interface is RS-485 to the pool equipment that is used by nodejs-poolController. The HS interface to nodejs-poolController is ethernet.

The development thread for pool controller integration with HS is at https://forums.homeseer.com/forum/lighting-primary-technology-plug-ins/lighting-primary-technology-discussion/mcsmqtt-michael-mcsharry/1425619-poolcontroller-homeseer-interface-with-mcsmqtt/page12. Most of the development work for this integration was done by miamijerry on the HS Message Board. If one has difficulties then it may be helpful to scan this thread.

After installation the following is done to configure with MQTT operation:

Configure the following items under interfaces in config.json.
Open terminal and paste
sudo nano /home/pi/nodejs-poolController/config.json

```
"mqtt": {
    "name": "MQTT",
    "type": "mqtt",
    "enabled": true"
    "fileName": "mqtt.json",
    "globals": {},
    "options": {
     "protocol": "mqtt://",
     "host": "192.168.0.103",
     "port": 1883,
     "username": "",
     "password": "",
     "rootTopic": "pool",
     "retain": true,
     "qos": 0
```

Notes, change the following three items;
“true” enables MQTT client in poolController.
Host “IP” is the IP address of the MQTT broker ie. mcsMQTT,
RootTopic “pool” is used to sort mqtt topics.

Pool controller has the ability to send much information about the pool controller and at a rate of one or two hertz. This is in excess of the amount of information that is needed in HS and will overload mcsMQTT/HS. Two methods exist to reduce the volume of traffic. The first is with mcsMQTT that will change the subscribe list to only those topics it uses for the default set of devices it creates in HS. This is on the Client Tab as shown in Figure 381. This list is built when the topic starting with "pool/" is first observed. It will also select the third radio to listen for only this list of topics. **For those that use MQTT for more than pool controller then the radio needs to be changed to the top selection as shown in** Figure 381**.** Alternately the subscription list for Topic Discovery setting can be augmented by explicitly adding the other MQTT topis that are being used.

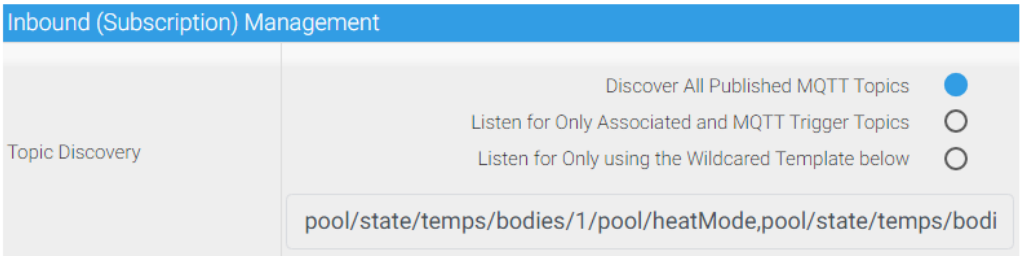


Figure 381 Pool Control MQTT Subscribe List

The second and preferred method is to limit the topics being published by the pool controller. This is with edit of the binding on nodejs to enable a specific list. This binding may change over time. The snapshot used for the initial integration in shown in Table 13.

Table 13 Pool Controller Binding Snapshot

```
{
    "context": {
        "name": "MQTT",
        "options": {
            "formatter": [
                {
                    "transform": ".toLowerCase()"
                },
                {
                    "regexkey": "\\s",
                    "replace": "",
                    "description": "Remove whitespace"
                },
                {
                    "regexkey": "\\/",
                    "replace": "",
                    "description": "Remove /"
                },
                {
                    "regexkey": "\\+",
                    "replace": "",
                    "description": "Remove +"
                },
                {
                    "regexkey": "\\$",
                    "replace": "",
                    "description": "Remove $"
                },
                {
                    "regexkey": "\\#",
                    "replace": "",
```

```
                    "description": "Remove #"
                }
            ],
            "rootTopic-DIRECTIONS": "You can override the root topic by renaming
_rootTopic to rootTopic",
            "_rootTopic": "@bind=(state.equipment.alias).replace(' ','-
').replace('/','').toLowerCase();",
            "clientId": "@bind='mqttjs_njsPC_'+Math.random().toString(16).substr(2, 8);"
        }
    },
  "events": [
    {
      "name": "config",
      "description": "Don't flood the MQTT bus.",
      "enabled": false
    },
    {
      "name": "controller",
      "description": "Emit time every minute",
      "enabled": true,
      "topics": [
        {
          "topic": "state/time",
          "message": "@bind=data.time;",
          "filter": "@bind=data.status.percent === 100;"
        }
      ]
    },
    {
      "name": "circuit",
      "description": "Populate the circuits topics",
      "topics": [
        {
          "topic": "state/circuits/@bind=data.id;/@bind=data.name;",
          "message":
"{\"id\":@bind=data.id;,\"isOn\":@bind=data.isOn?'\"on\"':'\"off\"';}",
          "description": "Bind 'on'/'off' as a message to the state topic."
        },
        {
          "topic": "state/circuits/@bind=data.id;/@bind=data.name;/isOn/string",
          "message": "@bind=data.isOn?'on':'off';",
          "description": "Bind 'on'/'off' as a message to the topic.",
          "enabled": false
        },
        {
          "topic": "state/circuits/@bind=data.id;/@bind=data.name;/isOn/boolean",
          "message": "@bind=data.isOn;",
          "description": "SAMPLE: Bind the isOn as a message to the topic.",
          "enabled": false
        },
        {
          "topic": "state/circuits/@bind=data.id;/@bind=data.name;/lightingTheme",
          "message": "{\"lightingTheme\":@bind=data.lightingTheme;}",
          "description": "SAMPLE: Bind the lighting theme to the topic.",
          "filter": "@bind=data.type.isLight === true;"
        },
        {
          "topic":
"state/circuits/@bind=data.id;/customTopicFormatter/@bind=data.name;/isOn",
          "message": "@bind=data.isOn;",
          "description": "SAMPLE: Bind the isOn as a message to the topic with a custom
replacer, qos and retain setting.",
          "formatter": [
            {
              "transform": ".toLowerCase()"
            },
```

```
            {
              "regexkey": "\\s",
              "replace": "__",
              "description": "Remove whitespace and replace with __"
            },
            {
              "regexkey": "\\/",
              "replace": "__",
              "description": "Remove / and replace with __"
            }
          ],
          "qos": 2,
          "enabled": false
        },
        {
          "topic": "state/circuits/@bind=data.id;/@bind=data.name;/object",
          "message": "@bind=data;",
          "description": "SAMPLE: Bind a JSON object as a message to the topic.",
          "enabled": false
        }
      ]
    },
    {
      "name": "virtualCircuit",
      "description": "Populate the virtual circuits topics",
      "topics": [
        {
          "topic": "state/virtualcircuits/@bind=data.id;/@bind=data.name;",
          "message":
"{\"id\":@bind=data.id;,\"isOn\":@bind=data.isOn?'\"on\"':'\"off\"';}",
          "description": "Bind 'on'/'off' as a message to the state topic."
        }
      ]
    },
    {
      "name": "valve",
      "description": "Populate the valve topics",
      "topics": [
        {
          "topic": "state/valve/@bind=data.id;/@bind=data.name;",
          "message":
"{\"id\":@bind=data.id;,\"isOn\":@bind=data.isDiverted?'\"on\"':'\"off\"';,\"isVirtual\":
@bind=data.isVirtual? true: false;,\"pinId\": @bind=data.pinId;}",
          "description": "Bind 'on'/'off' as a message to the valve state topic."
        }
      ]
    },

    {
      "name": "feature",
      "description": "Populate the features topics",
      "topics": [
        {
          "topic": "state/features/@bind=data.id;/@bind=data.name;",
          "message":
"{\"id\":@bind=data.id;,\"isOn\":@bind=data.isOn?'\"on\"':'\"off\"';}",
          "description": "Bind 'on'/'off' as a message to the state topic."
        }
      ]
    },
    {
      "name": "temps",
      "description": "Populate the temps topics",
      "topics": [
        {
          "topic": "state/temps/air",
```

```
          "message": "{\"temp\":@bind=data.air;}",
          "description": "Send air temp."
        },
        {
          "topic": "state/temps/solar",
          "message": "{\"temp\":@bind=data.solar;}",
          "description": "Send solar temp.",
          "filter": "@bind=typeof data.solar === 'undefined';"
        },
        {
          "topic": "state/temps/units",
          "message": "{\"units\":@bind=data.units;}"
        }
      ]
    },
    {
      "name": "body",
      "description": "Populate the body topic",
      "topics": [
        {
          "topic": "state/temps/bodies/@bind=data.id;/@bind=data.name;",
          "message":
"{\"id\":@bind=data.id;,\"isOn\":@bind=data.isOn?'\"on\"':'\"off\"';}",
          "description": "Bind 'on'/'off' as a message to the state topic."
        },
        {
          "topic": "state/temps/bodies/@bind=data.id;/@bind=data.name;/heatMode",
          "message": "{\"heatMode\":@bind=data.heatMode;}",
          "description": "Send heat mode."
        },
        {
          "topic": "state/temps/bodies/@bind=data.id;/@bind=data.name;/heatStatus",
          "message": "{\"heatStatus\":@bind=data.heatStatus;}",
          "description": "Send heat status."
        },
        {
          "topic": "state/temps/bodies/@bind=data.id;/@bind=data.name;/setPoint",
          "message": "{\"setPoint\":@bind=data.setPoint;}",
          "description": "Send set point."
        },
        {
          "topic": "state/temps/bodies/@bind=data.id;/@bind=data.name;/temp",
          "message": "{\"temp\":@bind=data.temp;}",
          "description": "Send temp."
        }
      ]
    },
    {
      "name": "chlorinator",
      "description": "Populate the chlorinator topic",
      "topics": [
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;",
          "message":
"{\"id\":@bind=data.id;,\"isOn\":@bind=data.isOn?'\"on\"':'\"off\"';}",
          "description": "Bind 'on'/'off' as a message to the state topic."
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/currentOutput",
          "message": "{\"currentOutput\":@bind=data.currentOutput;}",
          "description": "Send current output."
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/poolSetpoint",
          "message": "{\"poolSetpoint\":@bind=data.poolSetpoint;}",
          "description": "Send pool setpoint."
```

```
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/spaSetpoint",
          "message": "{\"spaSetpoint\":@bind=data.spaSetpoint;}",
          "description": "Send set point."
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/status",
          "message": "{\"status\":@bind=data.status;}",
          "description": "Send status."
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/superChlor",
          "message": "{\"superChlor\":@bind=data.superChlor?'\"on\"':'\"off\"';}",
          "description": "Send superChlor."
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/superChlorHours",
          "message": "{\"superChlorHours\":@bind=data.superChlorHours;}",
          "description": "Send superChlorHours."
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/saltLevel",
          "message": "{\"saltLevel\":@bind=data.saltLevel;}",
          "description": "Send salt level."
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/type",
          "message": "{\"type\":@bind=data.type;}",
          "description": "Send type."
        },
        {
          "topic": "state/chlorinators/@bind=data.id;/@bind=data.name;/targetOutput",
          "message": "{\"targetOutput\":@bind=data.targetOutput;}",
          "description": "Send targetOutput."
        },
        {
          "topic":
"state/chlorinators/@bind=data.id;/@bind=data.name;/virtualControllerStatus",
          "message": "{\"virtualControllerStatus\":@bind=data.virtualControllerStatus;}",
          "description": "Send virtualControllerStatus."
        }
      ]
    },
    {
      "name": "lightGroup",
      "description": "Populate the lightGroup topic",
      "topics": [
        {
          "topic": "state/lightgroups/@bind=data.id;/@bind=data.name;",
          "message":
"{\"id\":@bind=data.id;,\"isOn\":@bind=data.isOn?'\"on\"':'\"off\"';}",
          "description": "Bind 'on'/'off' as a message to the state topic."
        },
        {
          "topic": "state/lightgroups/@bind=data.id;/@bind=data.name;/action",
          "message": "{\"action\":@bind=data.action;}"
        },
        {
          "topic": "state/lightgroups/@bind=data.id;/@bind=data.name;/lightingTheme",
          "message": "{\"lightingTheme\":@bind=data.lightingTheme;}"
        },
        {
          "topic": "state/lightgroups/@bind=data.id;/@bind=data.name;/type",
          "message": "{\"type\":@bind=data.type;}"
        }
```

```
      ]
    },
    {
      "name": "pump",
      "description": "Populate the pumps topic",
      "topics": [
        {
          "topic": "state/pumps/@bind=data.id;/@bind=data.name;",
          "message":
"{\"id\":@bind=data.id;,\"isOn\":@bind=data.isOn?'\"on\"':'\"off\"';}",
          "description": "Bind 'on'/'off' as a message to the state topic."
        },
        {
          "topic": "state/pumps/@bind=data.id;/@bind=data.name;/rpm",
          "message": "{\"rpm\":@bind=data.rpm;}"
        },
        {
          "topic": "state/pumps/@bind=data.id;/@bind=data.name;/flow",
          "message": "{\"flow\":@bind=data.flow;}"
        },
        {
          "topic": "state/pumps/@bind=data.id;/@bind=data.name;/watts",
          "message": "{\"watts\":@bind=data.watts;}"
        },
        {
          "topic": "state/pumps/@bind=data.id;/@bind=data.name;/status",
          "message": "{\"status\":@bind=data.status;}"
        }
      ]
    },
    {
      "name": "chemController",
      "description": "Populate the chemControllers topic",
      "topics": [
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/acidTankLevel",
          "message": "{\"acidTankLevel\":@bind=data.acidTankLevel;}",
                "enabled": false
        },
        {
          "topic": "config/chemControllers/@bind=data.id;/@bind=data.name;/alkalinity",
          "message": "{\"alkalinity\":@bind=data.alkalinity;}"
        },
        {
          "topic":
"config/chemControllers/@bind=data.id;/@bind=data.name;/calciumHardness",
          "message": "{\"calciumHardness\":@bind=data.calciumHardness;}"
        },
        {
          "topic": "config/chemControllers/@bind=data.id;/@bind=data.name;/cyanuricAcid",
          "message": "{\"cyanuricAcid\":@bind=data.cyanuricAcid;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/orpDosingTime",
          "message": "{\"orpDosingTime\":@bind=data.orpDosingTime;}",
                "enabled": false
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/orpLevel",
          "message": "{\"orpLevel\":@bind=data.orpLevel;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/orpSetpoint",
          "message": "{\"orpSetpoint\":@bind=data.orpSetpoint;}"
        },
        {
```

```
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/orpTankLevel",
          "message": "{\"orpTankLevel\":@bind=data.orpTankLevel;}",
                "enabled": false
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/pHDosingTime",
          "message": "{\"pHDosingTime\":@bind=data.pHDosingTime;}",
                "enabled": false
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/pHLevel",
          "message": "{\"pHLevel\":@bind=data.pHLevel;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/pHSetpoint",
          "message": "{\"pHSetpoint\":@bind=data.pHSetpoint;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/saltLevel",
          "message": "{\"saltLevel\":@bind=data.saltLevel;}"
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/saturationIndex",
          "message": "{\"saturationIndex\":@bind=data.saturationIndex;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/status",
          "message": "{\"status\":@bind=data.status;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/type",
          "message": "{\"type\":@bind=data.type;}"
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/virtualControllerStatus",
          "message": "{\"virtualControllerStatus\":@bind=data.virtualControllerStatus;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/alarms/flow",
          "message": "{\"flow\":@bind=data.flow;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/alarms/ph",
          "message": "{\"ph\":@bind=data.ph;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/alarms/orp",
          "message": "{\"orp\":@bind=data.orp;}"
        },
        {
          "topic": "state/chemControllers/@bind=data.id;/@bind=data.name;/alarms/phTank",
          "message": "{\"phTank\":@bind=data.phTank;}",
                "enabled": false
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/alarms/orpTank",
          "message": "{\"orpTank\":@bind=data.orpTank;}",
                "enabled": false
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/alarms/probeFault",
          "message": "{\"probeFault\":@bind=data.probeFault;}",
```

```
                  "enabled": false
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/warnings/waterChemistry",
          "message": "{\"waterChemistry\":@bind=data.waterChemistry;}"
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/warnings/phLockout",
          "message": "{\"phLockout\":@bind=data.phLockout;}",
                  "enabled": false
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/warnings/phDailLimitReached",
          "message": "{\"phDailLimitReached\":@bind=data.phDailLimitReached;}",
                  "enabled": false
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/warnings/orpDailyLimitReached",
          "message": "{\"orpDailyLimitReached\":@bind=data.orpDailyLimitReached;}",
                  "enabled": false
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/warnings/invalidSetup",
          "message": "{\"invalidSetup\":@bind=data.invalidSetup;}"
        },
        {
          "topic":
"state/chemControllers/@bind=data.id;/@bind=data.name;/warnings/chlorinatorCommError",
          "message": "{\"chlorinatorCommError\":@bind=data.chlorinatorCommError;}",
                  "enabled": false
        }
      ]
    },
    {
      "name": "*",
      "description": "DEFAULT: Sends the entire emitted response.",
      "body": "@bind=data;",
      "enabled": false
    }
  ]
}
```


When the first MQTT topic starting with “pool/” is observed, mcsMQTT will create a set of default devices that represent the dashpanel for the pool controller. These as well as others will appear in the Association table of mcsMQTT so the additional can also be associated with HS devices if desired. The Device and Features is shown in Figure 382. If issues exist with the initial creation then remove the pool topics from mcsMQTT using “pool/#” in the text box on Client tab, Inbound Management section and let them be created again on the first observation of the pool/ topic.

pool | pool				
pool (1612)		Today 7:55:38 PM		
pool | state				
Temps-bodies_1_pool_heatMode (1613)	Off	Today 7:55:38 PM	OFF	ENABLE
pool | state				
Temps-bodies_2_spa_heatMode (1614)	Off	Today 7:55:38 PM	OFF	ENABLE
pool | state				
Temps-air (1615)	0	Today 7:55:38 PM		
pool | state				
Temps-bodies_1_pool_temp (1616)	0	Today 7:55:38 PM		
pool | state				
Temps-bodies_2_spa_temp (1617)	0	Today 7:55:38 PM		
pool | state				
Temps-bodies_1_pool_setPoint (1618)	0	Today 7:55:39 PM	Value 0	SUBMIT
pool | state				
Temps-bodies_2_spa_setPoint (1619)	0	Today 7:55:39 PM	Value 0	SUBMIT
pool | state				
Temps-bodies_1_pool_heatStatus (1620)	Off	Today 7:55:39 PM		
pool | state				
Temps-bodies_2_spa_heatStatus (1621)	Off	Today 7:55:39 PM		
pool | state				
Circuits-1-spa (1622)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Circuits-2-aux1 (1623)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Circuits-3-aux2 (1624)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Circuits-4-poollight (1625)	on	Today 7:55:41 PM	OFF	ON
pool | state				
Circuits-5-aux4 (1626)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Circuits-6-pool (1627)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Circuits-7-aux5 (1628)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Circuits-8-aux6 (1629)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Circuits-9-aux7 (1630)	off	Today 7:55:39 PM	OFF	ON

pool | state				
Features-11-feature1 (1631)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Features-12-feature2 (1632)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Features-13-feature3 (1633)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Features-14-feature4 (1634)	off	Today 7:55:39 PM	OFF	ON
pool | state				
Features-15-feature5 (1635)	off	Today 7:55:40 PM	OFF	ON
pool | state				
Features-16-feature6 (1636)	off	Today 7:55:40 PM	OFF	ON
pool | state				
Features-17-feature7 (1637)	off	Today 7:55:40 PM	OFF	ON
pool | state				
Features-18-feature8 (1638)	off	Today 7:55:40 PM	OFF	ON
pool | state				
Features-auxextra (1639)	off	Today 7:55:40 PM	OFF	ON
pool | state				
ChemController-pool_state_chlorinators_1_intellichlor--40_saltLevel (1640)	0	Today 7:55:40 PM		
pool | state				
ChemController-pool_state_chemControllers_1_chemcontroller1_alarms_ph (1641)		Today 7:55:40 PM		
pool | state				
ChemController-pool_state_chemControllers_1_chemcontroller1_alarms_orp (1642)		Today 7:55:40 PM		
pool | state				
ChemController-pool_state_chemControllers_1_chemcontroller1_alarms_ph (1643)		Today 7:55:40 PM	Value 0	SUBMIT
pool | state				
ChemController-pool_state_chemControllers_1_chemcontroller1_alarms_orp (1644)		Today 7:55:40 PM	Value 0	SUBMIT

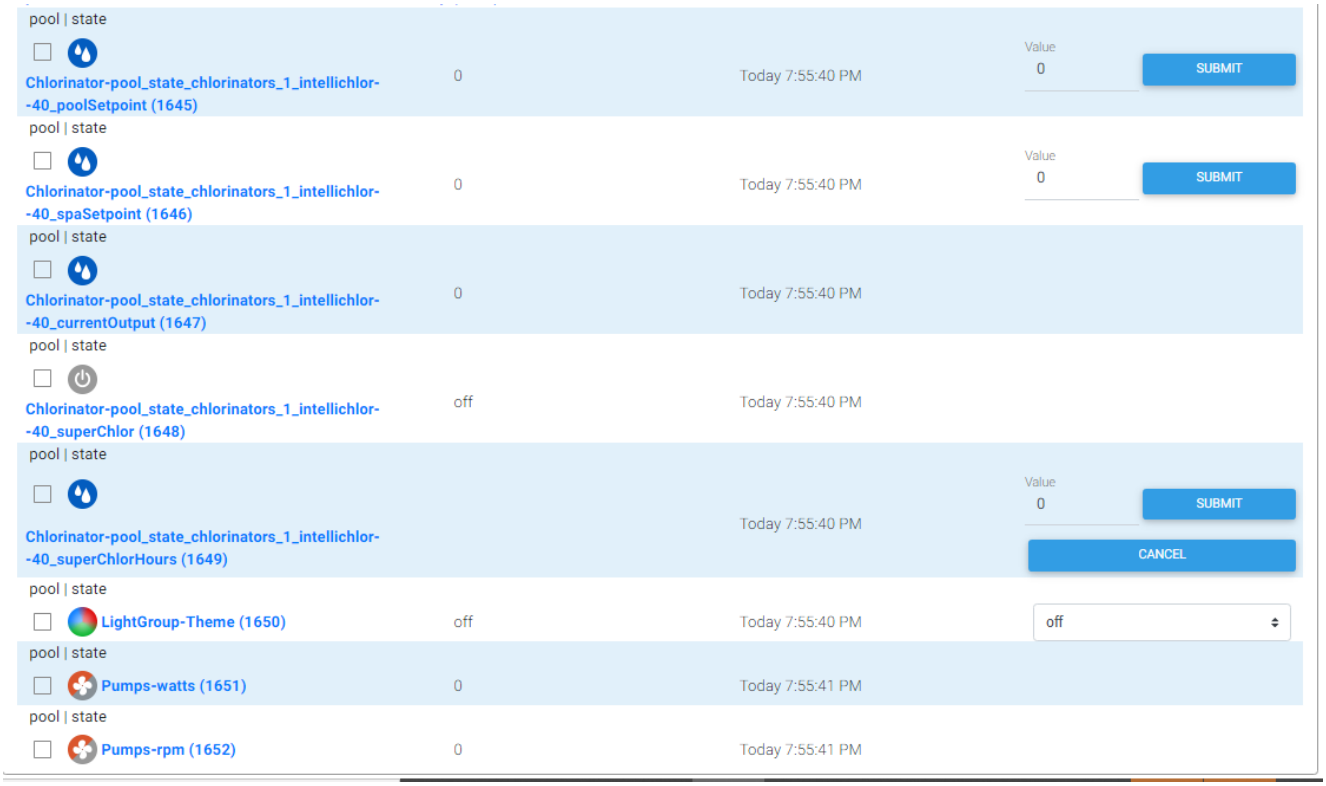


Figure 382 Pool Controller Default Device and Features

Table 14 MQTT Message Predefined Setup

HS Device	MQTT Subscribed Topic	HS Device Published Topic	Published Payload Template	PUT
Pool 1	pool/state/circuits/6/pool:isOn Payload off=0;off VSP Payload on=1;on VSP	pool/state/circuits/setState	{“id”:6,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:6,”state”:true}
Pool Heat Mode 2	pool/state/temps/bodies/1/pool/heatMode:heatMode:desc Gas Heater VSP values Off=1;Off Heater=2;Enable Payload Off=1;Off VSP Payload Heater=2;Enable VSP Solar VSP values Off=1;Off Heater=2;Enable	pool/state/body/heatMode	{“id”:1, “heatMode”:$$VALUE:}	PUT /state/body/heatMode {“id”:1,”mode”:1}
Spa Heat Mode 3	pool/state/temps/bodies/2/spa/heatMode:heatMode:desc Gas Heater VSP values Off=1;Off Heater=2;Enable Payload Off=1;Off VSP Payload Heater=2;Enable VSP Solar VSP values Off=1;Off Heater=2;Enable	pool/state/body/heatMode	{“id”:2,”heatMode”:$$VALUE:}	PUT /state/body/heatMode {“id”:2,”mode”:1}
Air Temp 4	pool/state/temps/air:temp	Display Only		

Pool Temp 5	pool/state/temps/bodies/1/pool/temp:temp	Display only		
Spa Temp 6	pool/state/temps/bodies/2/spa/temp:temp	Display only		
Pool Setpoint 7	pool/state/temps/bodies/1/pool/setPoint:setPoint	pool/state/body/setPoint	{“id”:1,”setPoint”:$$VALUE:}	PUT /state/body/setPoint {“id”:1,”setPoint”:76}
Spa Setpoint 8	pool/state/temps/bodies/2/spa/setPoint:setPoint	pool/state/body/setPoint	{“id”:2,”setPoint”:$$VALUE:}	PUT /state/body/setPoint {“id”:2,”setPoint”:71}
Heat Status Pool 9	pool/state/temps/bodies/1/pool/heatStatus:heatStatus:desc	Display only		
Heat Status Spa 9.1	pool/state/temps/bodies/2/spa/heatStatus:heatStatus:desc	Display only		
Spa 10	pool/state/circuits/1/spa:isOn	pool/state/circuits/setState	{“id”:1,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:1,”state”:true}
Aux 1 11	pool/state/circuits/2/aux1:isOn	pool/state/circuits/setState	{“id”:2,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:2,”state”:true}
Aux 2 12	pool/state/circuits/3/aux2:isOn	pool/state/circuits/setState	{“id”:3,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:3,”state”:true}
Aux 3 13	pool/state/circuits/4/aux3:isOn	pool/state/circuits/setState	{“id”:4,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:4,”state”:true}
Aux 4 14	pool/state/circuits/5/aux4:isOn	pool/state/circuits/setState	{“id”:5,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:5,”state”:true}
Aux 5	pool/state/circuits/7/aux5:isOn	pool/state/circuits/setState	{“id”:7, “isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:7,”state”:true}

15				
Aux6 16	pool/state/circuits/8/aux6:isOn	pool/state/circuits/setState	{“id”:8,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:8,”state”:true}
Aux7 17	pool/state/circuits/9/aux7:isOn	pool/state/circuits/setState	{“id”:9,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:9,”state”:true}
	New Equipment - IntelliCenter			
Aux 8 Intellicenter	pool/state/circuits/10/aux8:isOn	pool/state/circuits/setState	{"id":10,"isOn":"$$LABEL:"}	
Aux 9 Intellicenter	pool/state/circuits/11/aux9:isOn	pool/state/circuits/setState	{“id”:11,”isOn”:”$$LABEL:”}	
	New Equipment - IntelliCenter			
Feature 1 Intellicenter	pool/state/features/129/feature1:isOn	pool/state/features/setState	{"id":129,"isOn":"$$LABEL:"}	
Feature 2 Intellicenter	pool/state/features/130/feature2:isOn	pool/state/features/setState	{"id":130,"isOn":"$$LABEL:"}	
Feature 3 Intellicenter	pool/state/features/131/feature3:isOn	pool/state/features/setState	{"id":131,"isOn":"$$LABEL:"}	
Feature 4 Intellicenter	pool/state/features/132/feature4:isOn	pool/state/features/setState	{"id":132,"isOn":"$$LABEL:"}	
Feature 5 Intellicenter	pool/state/features/133/feature5:isOn	pool/state/features/setState	{"id":133,"isOn":"$$LABEL:"}	
Feature 6 Intellicenter	pool/state/features/134/feature6:isOn	pool/state/features/setState	{"id":134,"isOn":"$$LABEL:"}	
Feature 7	pool/state/features/135/feature7:isOn	pool/state/features/setState	{"id":135,"isOn":"$$LABEL:"}	

Intellicenter				
Feature 8 Intellicenter	pool/state/features/136/feature8:isOn	pool/state/features/setState	{"id":136,"isOn":"$$LABEL:"}	
Feature 9 Intellicenter	pool/state/features/137/feature9:isOn	pool/state/features/setState	{"id":137,"isOn":"$$LABEL:"}	
	Original Equipment - EasyTouch			
Feature 1 EasyTouch 18	pool/state/features/11/feature1:isOn	pool/state/features/setState	{“id”:11,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:11,”state”:true}
Feature 2 19	pool/state/features/12/feature2:isOn	pool/state/features/setState	{“id”:12,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:12,”state”:true}
Feature 3 20	pool/state/features/13/feature3:isOn	pool/state/features/setState	{“id”:13,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:13,”state”:true}
Feature 4 21	pool/state/features/14/feature4:isOn	pool/state/features/setState	{“id”:14,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:14,”state”:true}
Feature 5 22	pool/state/features/15/feature5:isOn	pool/state/features/setState	{“id”:15, “isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:15,”state”:true}
Feature 6 23	pool/state/features/16/feature6:isOn	pool/state/features/setState	{“id”:16,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:16,”state”:true}
Feature 7 24	pool/state/features/17/feature7:isOn	pool/state/features/setState	{“id”:17,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:17,”state”:true}
Feature 8 25	pool/state/features/18/feature8:isOn	pool/state/features/setState	{“id”:18,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:18,”state”:true}
Aux Extra 26	pool/state/features/20/auxextra:isOn	pool/state/features/setState	{“id”:20,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:20,”state”:true}

Salt Level 27	pool/state/chlorinators/1/intellichlor--40/saltLevel:saltLevel	Display only		
PH Level 28	EasyTouch pool/state/chemControllers/1/chemcontroller1/alarms/ph:level IntelliCenter pool/state/chemControllers/1/chemcontroller1/pHLevel:pHLevel poolController v6.5.2 next pool/state/chemControllers/1/chemcontroller1/ph/level	Display only		
ORP Level 29	EasyTouch pool/state/chemControllers/1/chemcontroller1/alarms/orp:level IntelliCenter pool/state/chemControllers/1/chemcontroller1/orpLevel:orpLevel poolController v6.5.2 next pool/state/chemControllers/1/chemcontroller1/orp/level	Display only		
PH Setpoint EasyTouch 30	pool/state/chemControllers/1/chemcontroller1/alarms/ph:setpoint	pool/state/chemController	{"id":1,"ph":{"setpoint":$$VALUE:}	PUT /state/chemController {“id”:1,”ph”:{“setpoint”:7.5},”orp”:{“setpoint”:720},”alkalinity”:25,”calciumHardness”:25,”cyanuricAcid”:0,”saturationIndex”:”-1.0”}
PH setpoint IntelliCenter	pool/state/chemControllers/1/chemcontroller1/ph/setpoint or pool/state/chemControllers/1/chemcontroller1/pHSetpoint:pHSetpoint	pool/state/chemController	{"id":1,"ph":{"setpoint":$$VALUE:}	
Orp setpoint 31	EasyTouch pool/state/chemControllers/1/chemcontroller	pool/state/chemController	{"id":1,"orp":{"setpoint":$$VALUE:}	PUT /state/chemController {“id”:1,”ph”:{“setpoint”:7.4},”orp”:{“setpoint”:730},”alkalinity”:25,”calciumHardness”:25,”cyanuricAcid”:0,

	r1/alarms/orp:setpoint IntelliCenter pool/state/chemControllers/1/chemcontroller1/orp/setpoint or pool/state/chemControllers/1/chemcontroller1/orpSetpoint:orpSetpoint poolController v6.5.2 next pool/state/chemControllers/1/chemcontroller1/orp/setpoint			”saturationIndex”:”-1.0”}
Chlorinator Pool Setpoint 32	pool/state/chlorinators/1/intellichlor--40/poolSetpoint:poolSetpoint	pool/state/chlorinator	{"id":1,"poolSetpoint":$$VALUE:}	PUT /state/chlorinator/poolSetpoint {“id”:1,”setPoint”:95}
Chlorinator Spa Setpoint 33	pool/state/chlorinators/1/intellichlor--40/spaSetpoint:spaSetpoint	pool/state/chlorinator	{"id":1,"spaSetpoint":$$VALUE:}	PUT /state/chlorinator/spaSetpoint {“id”:1,”setPoint”:7}
Chlorinator Current Output 34	pool/state/chlorinators/1/intellichlor--40/currentOutput:currentOutput	Display only		
Chlorinator SuperChlor 35	pool/state/chlorinators/1/intellichlor--40/superChlor:superChlor	pool/state/chlorinator	{"id":1,"superChlorinate":$$VALUE:}	PUT /state/chlorinator/superChlorinate {“id”:1,”superChlorinate”:true}
Chlorinator Super ChlorHours 36	pool/state/chlorinators/1/intellichlor--40/superChlorHours:superChlorHours	pool/state/chlorinator or pool/config/chlorinator	{"id":1,"superChlorHours":$$VALUE:}	PUT /state/chlorinator/superChlorHours {“id”:1,”hours”:2}
Intellibrite Themes 37	EasyTouch pool/state/lightgroups/192/intellibrite/lightingTheme:lightingTheme:name Payload off=0;off VSP Payload on=1;on VSP Payload thumper=208;thumper VSP	pool/state/circuit/setTheme	EasyTouch {“id”:192,”theme”:$$VALUE:} IntelliCenter	EasyTouch PUT /state/circuit/setTheme {“id”:192,”theme”:128} IntelliCenter

	Payload hold=209;hold VSP Payload reset=210;reset VSP Payload mode=211;mode VSP Payload colorsync=128;colorsync VSP Payload colorswim=144;colorswim VSP Payload unknown=254;unknown VSP Payload colorset=160;colorset VSP Payload party=177;party VSP Payload romance=178;romance VSP Payload carribean=179;carribean VSP Payload american=180;american VSP Payload sunset=181;sunset VSP Payload royal=182;royal VSP Payload save=190;save VSP Payload recall=191;recall VSP Payload blue=193;blue VSP Payload green=194;green VSP Payload red=195;red VSP Payload white=196;white VSP Payload magenta=197;magenta VSP Payload none=198;none VSP IntelliCenter pool/state/circuits/5/poollight/lightingTheme :lightingTheme:desc White = 0 Green = 1 Blue = 2 Magenta = 3 Red = 4 Sam mode = 5 Party = 6 Romance = 7 Carribean = 8 American = 9 Sunset = 10 Royal = 11		{“id”:5,”theme”:$$VALUE:}	PUT /state/circuit/setTheme {"id":5,"theme":2}
Pump Watts 38	EasyTouch pool/state/pumps/1/intelliflovs/watts:watts IntelliCenter	Display only		

	pool/state/pumps/1/vs/watts:watts			
Pump RPM 39	EasyTouch pool/state/pumps/1/intelliflovs/rpm:rpm IntelliCenter pool/state/pumps/1/vs/rpm:rpm	Display only		

HS Device	MQTT Subscribed Topic	HS Device Published Topic	Published Payload Template	PUT
Pool 1	pool/state/circuits/6/pool:isOn Payload off=0;off VSP Payload on=1;on VSP	pool/state/circuits/setState	{“id”:6,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:6,”state”:true}
Pool Heat Mode 2	pool/state/temps/bodies/1/pool/heatMode:heatMode:desc Gas Heater VSP values Off=1;Off Heater=2;Enable Solar VSP values Off=1;Off Heater=3;Enable	pool/state/body/heatMode	{“id”:1, “heatMode”:$$VALUE:}	PUT /state/body/heatMode {“id”:1,”mode”:1}
Spa Heat Mode 3	pool/state/temps/bodies/2/spa/heatMode:heatMode:desc Gas Heater VSP values Off=1;Off Heater=2;Enable	pool/state/body/heatMode	{“id”:2,”heatMode”:$$VALUE:}	PUT /state/body/heatMode {“id”:2,”mode”:1}

	Solar VSP values Off=1;Off Heater=3;Enable			
Air Temp 4	pool/state/temps/air:temp	Display Only		
Pool Temp 5	pool/state/temps/bodies/1/pool/temp:temp	Display only		
Spa Temp 6	pool/state/temps/bodies/2/spa/temp:temp	Display only		
Pool Setpoint 7	pool/state/temps/bodies/1/pool/setPoint:setPoint	pool/state/body/setPoint	{“id”:1,”setPoint”:$$VALUE:}	PUT /state/body/setPoint {“id”:1,”setPoint”:76}
Spa Setpoint 8	pool/state/temps/bodies/2/spa/setPoint:setPoint	pool/state/body/setPoint	{“id”:2,”setPoint”:$$VALUE:}	PUT /state/body/setPoint {“id”:2,”setPoint”:71}
Heat Status Pool 9	pool/state/temps/bodies/1/pool/heatStatus:heatStatus:desc	Display only		
Heat Status Spa 9.1	pool/state/temps/bodies/2/spa/heatStatus:heatStatus:desc	Display only		
Spa 10	pool/state/circuits/1/spa:isOn	pool/state/circuits/setState	{“id”:1,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:1,”state”:true}
Aux 1 11	pool/state/circuits/2/aux1:isOn	pool/state/circuits/setState	{“id”:2,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:2,”state”:true}
Aux 2 12	pool/state/circuits/3/aux2:isOn	pool/state/circuits/setState	{“id”:3,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:3,”state”:true}
Aux 3	pool/state/circuits/4/aux3:isOn	pool/state/circuits/setState	{“id”:4,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:4,”state”:true}

13				
Aux 4 14	pool/state/circuits/5/aux4:isOn	pool/state/circuits/setState	{“id”:5,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:5,”state”:true}
Aux 5 15	pool/state/circuits/7/aux5:isOn	pool/state/circuits/setState	{“id”:7, “isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:7,”state”:true}
Aux6 16	pool/state/circuits/8/aux6:isOn	pool/state/circuits/setState	{“id”:8,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:8,”state”:true}
Aux7 17	pool/state/circuits/9/aux7:isOn	pool/state/circuits/setState	{“id”:9,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:9,”state”:true}
Aux 8 Intellicenter	pool/state/circuits/10/aux8:isOn	pool/state/circuits/setState	{"id":10,"isOn":"$$LABEL:"}	
Aux 9 Intellicenter	pool/state/circuits/11/aux9:isOn	pool/state/circuits/setState	{“id”:11,”isOn”:”$$LABEL:”}	
Feature 1 Intellicenter	pool/state/features/129/feature1:isOn	pool/state/features/setState	{"id":129,"isOn":"$$LABEL:"}	
Feature 2 Intellicenter	pool/state/features/130/feature2:isOn	pool/state/features/setState	{"id":130,"isOn":"$$LABEL:"}	
Feature 3 Intellicenter	pool/state/features/131/feature3:isOn	pool/state/features/setState	{"id":131,"isOn":"$$LABEL:"}	
Feature 4 Intellicenter	pool/state/features/132/feature4:isOn	pool/state/features/setState	{"id":132,"isOn":"$$LABEL:"}	
Feature 5 Intellicenter	pool/state/features/133/feature5:isOn	pool/state/features/setState	{"id":133,"isOn":"$$LABEL:"}	
Feature 6 Intellicenter	pool/state/features/134/feature6:isOn	pool/state/features/setState	{"id":134,"isOn":"$$LABEL:"}	

Feature 7 Intellicenter	pool/state/features/135/feature7:isOn	pool/state/features/setState	{"id":135,"isOn":"$$LABEL:"}	
Feature 8 Intellicenter	pool/state/features/136/feature8:isOn	pool/state/features/setState	{"id":136,"isOn":"$$LABEL:"}	
Feature 9 Intellicenter	pool/state/features/137/feature9:isOn	pool/state/features/setState	{"id":137,"isOn":"$$LABEL:"}	
Feature 1 EasyTouch 18	pool/state/features/11/feature1:isOn	pool/state/features/setState	{“id”:11,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:11,”state”:true}
Feature 2 19	pool/state/features/12/feature2:isOn	pool/state/features/setState	{“id”:12,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:12,”state”:true}
Feature 3 20	pool/state/features/13/feature3:isOn	pool/state/features/setState	{“id”:13,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:13,”state”:true}
Feature 4 21	pool/state/features/14/feature4:isOn	pool/state/features/setState	{“id”:14,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:14,”state”:true}
Feature 5 22	pool/state/features/15/feature5:isOn	pool/state/features/setState	{“id”:15, “isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:15,”state”:true}
Feature 6 23	pool/state/features/16/feature6:isOn	pool/state/features/setState	{“id”:16,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:16,”state”:true}
Feature 7 24	pool/state/features/17/feature7:isOn	pool/state/features/setState	{“id”:17,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:17,”state”:true}
Feature 8 25	pool/state/features/18/feature8:isOn	pool/state/features/setState	{“id”:18,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:18,”state”:true}
Aux Extra 26	pool/state/features/20/auxextra:isOn	pool/state/features/setState	{“id”:20,”isOn”:”$$LABEL:”}	PUT /state/circuit/setState {“id”:20,”state”:true}

Salt Level 27	pool/state/chlorinators/1/intellichlor--40/saltLevel:saltLevel	Display only		
PH Level 28	EasyTouch pool/state/chemControllers/1/chemcontroller1/alarms/ph:level IntelliCenter pool/state/chemControllers/1/intellichem1/pHLevel:pHLevel	Display only		
ORP Level 29	EasyTouch pool/state/chemControllers/1/chemcontroller1/alarms/orp:level IntelliCenter pool/state/chemControllers/1/intellichem1/orpLevel:orpLevel	Display only		
PH Setpoint EasyTouch 30	pool/state/chemControllers/1/chemcontroller1/alarms/ph:setpoint	pool/state/chemController	{"id":1,"ph":{"setpoint":$$VALUE:} **2	PUT /state/chemController {“id”:1,”ph”:{“setpoint”:7.5},”orp”:{“setpoint”:720},”alkalinity”:25,”calciumHardness”:25,”cyanuricAcid”:0,”saturationIndex”:”-1.0”}
PH setpoint IntelliCenter	pool/state/chemControllers/1/intellichem1/pHSetpoint:pHSetpoint	pool/state/chemController	{"id":1,"ph":{"setpoint":$$VALUE:} **2	
Orp setpoint 31	EasyTouch pool/state/chemControllers/1/chemcontroller1/alarms/orp:setpoint IntelliCenter pool/state/chemControllers/1/intellichem1/orpSetpoint:orpSetpoint	pool/state/chemController	{"id":1,"orp":{"setpoint":$$VALUE:} **2	PUT /state/chemController {“id”:1,”ph”:{“setpoint”:7.4},”orp”:{“setpoint”:730},”alkalinity”:25,”calciumHardness”:25,”cyanuricAcid”:0,”saturationIndex”:”-1.0”} PM] info: [12:49:35 PM] 192.168.0.128 PUT /state/chemController {"id":1,"ph":{"setpoint":7.3,"dailyVolumeDosed":null},"orp":{"setpoint":730,"dailyVolumeDosed":null},"alkalinity":25,"calciumHardness":25,"cyanuricAcid":0,"saturationIndex":"-1.0"}
Chlorinator Pool	pool/state/chlorinators/1/intellichlor--	pool/state/chlorinator	{"id":1,"poolSetpoint":$$VALUE:}	PUT /state/chlorinator/poolSetpoint

Setpoint 32	40/poolSetpoint:poolSetpoint			{“id”:1,”setPoint”:95}
Chlorinator Spa Setpoint 33	pool/state/chlorinators/1/intellichlor--40/spaSetpoint:spaSetpoint	pool/state/chlorinator	{"id":1,"spaSetpoint":$$VALUE:}	PUT /state/chlorinator/spaSetpoint {“id”:1,”setPoint”:7}
Chlorinator Current Output 34	pool/state/chlorinators/1/intellichlor--40/currentOutput:currentOutput	Display only		
Chlorinator SuperChlor 35	pool/state/chlorinators/1/intellichlor--40/superChlor:superChlor	pool/state/chlorinator	{"id":1,"superChlorinate":$$VALUE:} **1	PUT /state/chlorinator/superChlorinate {“id”:1,”superChlorinate”:true}
Chlorinator Super ChlorHours 36	pool/state/chlorinators/1/intellichlor--40/superChlorHours:superChlorHours	pool/state/chlorinator	{"id":1,"superChlorHours":$$VALUE:}	PUT /state/chlorinator/superChlorHours {“id”:1,”hours”:2}
Intellibrite Themes 37	EasyTouch pool/state/lightgroups/192/intellibrite/lightingTheme:lightingTheme:name Payload off=0;off VSP Payload on=1;on VSP Payload thumper=208;thumper VSP Payload hold=209;hold VSP Payload reset=210;reset VSP Payload mode=211;mode VSP Payload colorsync=128;colorsync VSP Payload colorswim=144;colorswim VSP Payload unknown=254;unknown VSP Payload colorset=160;colorset VSP Payload party=177;party VSP Payload romance=178;romance VSP Payload carribean=179;carribean VSP Payload american=180;american VSP Payload sunset=181;sunset VSP Payload royal=182;royal VSP Payload save=190;save VSP Payload recall=191;recall VSP Payload blue=193;blue VSP Payload green=194;green VSP Payload red=195;red VSP Payload white=196;white VSP	pool/state/circuit/setTheme	EasyTouch {“id”:192,”theme”:$$VALUE:} IntelliCenter {“id”:5,”theme”:$$VALUE:}	EasyTouch PUT /state/circuit/setTheme {“id”:192,”theme”:128} IntelliCenter PUT /state/circuit/setTheme {"id":5,"theme":2}

	Payload magenta=197;magenta VSP Payload none=198;none VSP IntelliCenter pool/state/circuits/5/poollight/lightingTheme :lightingTheme:desc White = 0 Green = 1 Blue = 2 Magenta = 3 Red = 4 Sam mode = 5 Party = 6 Romance = 7 Carribean = 8 American = 9 Sunset = 10 Royal = 11			
Pump Watts 38	EasyTouch pool/state/pumps/1/intelliflovs/watts:watts IntelliCenter pool/state/pumps/1/vs/watts:watts	Display only		
Pump RPM 39	EasyTouch pool/state/pumps/1/intelliflovs/rpm:rpm IntelliCenter pool/state/pumps/1/vs/rpm:rpm	Display only		

**1 Chlorinator Commands

```
Friend Function ChlorinatorCommand(ByVal oMQTT As MqttReport, ByVal nValue As Double) As Boolean
  'return true if command was already sent
  With oMQTT
    If InStrRev(.Source, "superChlorHours") > 0 Then
      Dim sPayload As String = "{""id"":1,""superChlorHours"":" & nValue.ToString & "}"
      If oMQTTClient(.Broker) Is Nothing Then
```

```
            Return True
        End If
        Dim sTopic As String = .Topic
        Dim oMQTT2 As New MqttReport
        Dim sTopic2 As String
        Dim sPayloadOff As String
        Dim sPayloadOn As String
        If MQTTReceiveDictionary.TryGetValue("pool/state/chlorinators/1/intellichlor--40/superChlor:superChlor", oMQTT2) Then
            sTopic2 = oMQTT2.Topic
            sPayloadOff = ExpandedPayload(ZERO, oMQTT2, "")
            sPayloadOff = ExpandedPayload(oMQTT2.Template, oMQTT2, "", False, ZERO)
            sPayloadOn = ExpandedPayload(oMQTT2.Template, oMQTT2, "", False, ONE)
        Else
            sTopic2 = "pool/state/chlorinator"
            sPayloadOn = "{""id"":1,""superChlorinate"":""on""}"
            sPayloadOff = "{""id"":1,""superChlorinate"":""off""}"
        End If
        If nValue > 0.0 Then
            StatPublish(sTopic, sPayload, .History)
            oMQTTClient(.Broker).Publish(sTopic, Encoding.UTF8.GetBytes(sPayload), .QOS, .Retain)
            StatPublish(sTopic2, sPayloadOn, .History)
            oMQTTClient(.Broker).Publish(sTopic2, Encoding.UTF8.GetBytes(sPayloadOn), .QOS, .Retain)
        Else
            sPayload = ExpandedPayload(nValue.ToString, oMQTT, "")
            StatPublish(sTopic, sPayload, .History)
            oMQTTClient(.Broker).Publish(sTopic, Encoding.UTF8.GetBytes(sPayload), .QOS, .Retain)
            StatPublish(sTopic2, sPayloadOff, .History)
            oMQTTClient(.Broker).Publish(sTopic2, Encoding.UTF8.GetBytes(sPayloadOff), .QOS, .Retain)
        End If
        Return True
    End If
    Return False
  End With
End Function
```

```
**2 Chemcontroller Payllod Template

  Private Const EASYTOUCH_PH_SETPOINT_TOPIC As String = "pool/state/chemControllers/1/chemcontroller1/alarms/ph:setpoint"
  Private Const EASYTOUCH_ORP_SETPOINT_TOPIC As String = "pool/state/chemControllers/1/chemcontroller1/alarms/orp:setpoint"
  Private Const INTELLLICENTER_PH_SETPOINT_TOPIC As String = "pool/state/chemControllers/1/chemcontroller1/pHSetpoint:pHSetpoint"
  Private Const INTELLLICENTER_ORP_SETPOINT_TOPIC As String = "pool/state/chemControllers/1/chemcontroller1/orpSetpoint:orpSetpoint"

Friend Sub ChemControllerPayloadTemplate(oMQTT As MqttReport)
      'populate the MQTT template for ph and orp commands

      If InStr(oMQTT.Source, "chemController") > 0 Then
          Dim arrPayload() As String = {"7.1", "720", "25", "25", "0", "-1.0"}
         Dim arrTopic() As String = {"", ""}

          If gPoolEquipment = PoolEquipment.EasyTouch Then
              arrTopic(0) = EASYTOUCH_PH_SETPOINT_TOPIC
              arrTopic(1) = EASYTOUCH_ORP_SETPOINT_TOPIC
          Else
              arrTopic(0) = INTELLLICENTER_PH_SETPOINT_TOPIC
              arrTopic(1) = INTELLLICENTER_ORP_SETPOINT_TOPIC
          End If

          For i As Integer = 0 To arrTopic.Length - 1
              Dim oMQTT2 As New MqttReport
              Dim sTopic2 As String = arrTopic(i)
              If MQTTReceiveDictionary.TryGetValue(sTopic2, oMQTT2) Then
                  If IsNumeric(oMQTT2.Payload) Then
                      arrPayload(i) = oMQTT2.Payload
                  End If
              End If
          Next

          If InStr(oMQTT.Source, "ph", vbTextCompare) > 0 Then
              oMQTT.Template = "{""id"":1,""ph"":{""setpoint"":" & "$$VALUE:" & "},""orp"":{""setpoint"":" & arrPayload(1) &
"},""alkalinity"":" & arrPayload(2) & ",""calciumHardness"":" & arrPayload(3) & ",""cyanuricAcid"":" & arrPayload(4) &
",""saturationIndex"":""" & arrPayload(5) & """}"
          End If
          If InStr(oMQTT.Source, "orp") > 0 Then
              oMQTT.Template = "{""id"":1,""ph"":{""setpoint"":" & arrPayload(0) & "},""orp"":{""setpoint"":" & "$$VALUE:" &
"},""alkalinity"":" & arrPayload(2) & ",""calciumHardness"":" & arrPayload(3) & ",""cyanuricAcid"":" & arrPayload(4) &
",""saturationIndex"":""" & arrPayload(5) & """}"
          End If
      End If
  End Sub
```

24 IP Relay – Ethernet, WiFi, RS-485, CAN (Dingtian)

There is a very attractive product from China that provides relay control and digital input reporting via wired ethernet or WiFi as well as product variants of RS-485 and CAN when WiFi is not needed. The pricing is interesting with the 8-channel being the lowest cost at $14.25 and 4 and 8 channel at slightly higher price when a smaller form factor is desired. I obtained the 2-channel version for evaluation. The case is also nice with strong plastic and DIN rail capability.
https://www.aliexpress.com/item/4000999069820.html?spm=a2g0o.detail.1000060.1.49bb7aafhc4Kem&gps-id=pcDetailBottomMoreThisSeller&scm=1007.13339.169870.0&scm_id=1007.13339.169870.0&scm-url=1007.13339.169870.0&pvid=ebd8ac2d-a04f-4b4d-abd2-97afe5b420e0&_t=gps-id:pcDetailBottomMoreThisSeller,scm-url:1007.13339.169870.0,pvid:ebd8ac2d-a04f-4b4d-abd2-97afe5b420e0,tpp_buckets:668%232846%238116%232002&&pdp_ext_f=%7B%22sceneId%22:%223339%22,%22sku_id%22:%2210000013555389040%22%7D

The firmware is very capable and comes with much documentation at http://www.dingtian-tech.com/sdk/relay_sdk.zip Sample programs are given, interface with HA platform Domoticz, and multiple interface protocols supported including MQTT.

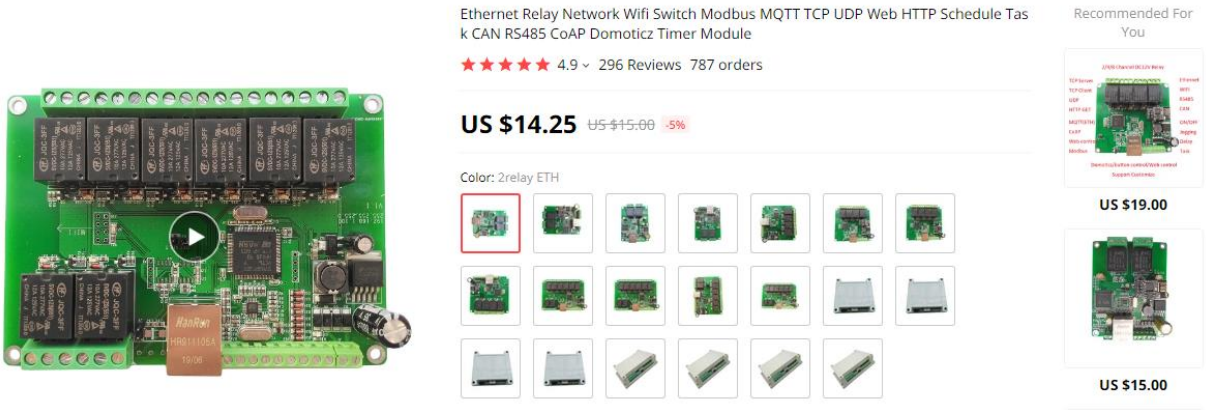


Figure 383 Dingtian IP Relay/Input/WiFi/RS-485/CAN Product Listing

I was not able to get a response from my device so I contacted supplier via email. Response came that evening showing how to do factory reset and offer for WhatApp or TeamViewer as more immediate ways that they could provide support.

The setup is via browser at default IP 192.168.1.1 admin/admin login credentials. The IP can then be changed to match your network. The MQTT setup is done on the Relay Connect tab of the browser menu where th MQTT Broker Address is entered and login credential to Broker if needed. See Figure 384. There is also a Keep Alive setting on the page that comes default at 30 seconds. The Keep Alive results in period status reporting at the specified interval. In my case I desired only event reporting so set it to 0.

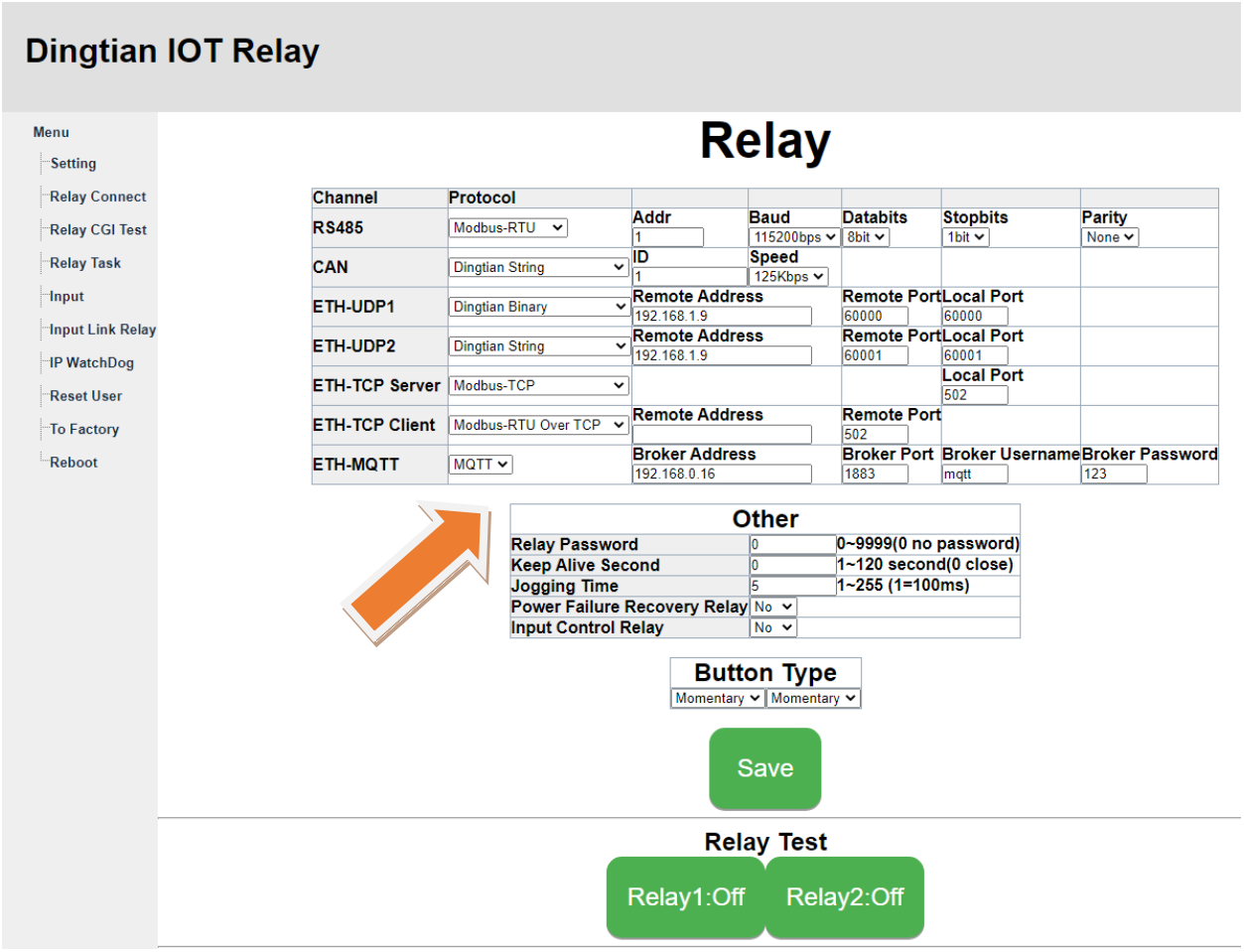


Figure 384 Dingtian IOT Relay MQTT Configuration

Clicking on the two Relay Test buttons will result in MQTT messages that will be visible on the mcsMQTT Association tab. Using a jumper wire between Gnd and each of the two digital inputs also results in MQTT messages. "A"ssociating the appropriate topics produces HS devices. Note the syntax of the Pub Topic to be able to control the relay in Figure 386. The base Topic is suffixed with /in/r# where # is the relay number.

Figure 385 Dingtian IOT Relay / Inputs as HS Devices

Association Table for Auto Association of MQTT Topic and HS Device

Λ	o	r	e	a	ref	TOPIC	payload	h	d	i	lastdate
0	☐	☐	☐	☐		Sub: /dingtian/relay4334/in/r1	OFF	☐			2021-09-23 12:10:53
1	☐	☐	☐		632	/dingtian/relay4334/out					
2	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/i1	ON	☐			2021-09-23 11:58:17
3	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/i2	ON	☐			2021-09-23 11:58:24
4	☐	☐	☐			/dingtian/relay4334/out/input1					
5	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/input1:idx	1	☐			2021-09-23 11:58:17
6	☐	☐	☐	☑	635	Dev: dingtian|dingtian|input1:status Sub: /dingtian/relay4334/out/input1:status Pub: the following Topic on Device command	LOW	☐	☐	☐	2021-09-23 11:58:17
7	☐	☐	☐			/dingtian/relay4334/out/input2					
8	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/input2:idx	2	☐			2021-09-23 11:58:25
9	☐	☐	☐	☑	636	Dev: dingtian|dingtian|input2:status Sub: /dingtian/relay4334/out/input2:status Pub: the following Topic on Device command	LOW	☐	☐	☐	2021-09-23 11:58:25
10	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/lwt_availability	online	☐			2021-09-23 12:06:14
11	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/r1	OFF	☐			2021-09-23 12:10:53
12	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/r2	ON	☐			2021-09-23 12:13:06
13	☐	☐	☐			/dingtian/relay4334/out/relay1					
14	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/relay1:idx	1	☐			2021-09-23 12:10:53
15	☐	☐	☐	☑	638	Dev: dingtian|dingtian|relay1:status Sub: /dingtian/relay4334/out/relay1:status Pub: the following Topic on Device command /dingtian/relay4334/in/r1	OFF	☐	☐	☐	2021-09-23 12:10:53
16	☐	☐	☐			/dingtian/relay4334/out/relay2					
17	☐	☐	☐	☐		Sub: /dingtian/relay4334/out/relay2:idx	2	☐			2021-09-23 12:13:06
18	☐	☐	☐	☑	639	Dev: dingtian|dingtian|relay2:status Sub: /dingtian/relay4334/out/relay2:status Pub: the following Topic on Device command /dingtian/relay4334/in/r2	ON	☐	☐	☐	2021-09-23 12:13:06

Figure 386 Dingtian IOT Relay MQTT Topics

25 WLED Support

WLED is an application targeted for ESP8266 and ESP32 that performs very flexible control of LED light strips such as WS2811 and WS2812B. The main support page for this firmware is https://github.com/Aircoookie/WLED/wiki. Supportin pages of interest are

https://github.com/Aircoookie/WLED/wiki/MQTT

https://github.com/Aircoookie/WLED/wiki/HTTP-request-API

https://github.com/Aircoookie/WLED/wiki/JSON-API

There are multiple source for install of the firmware that is available at https://github.com/Aircoookie/WLED/releases or source at https://github.com/Aircoookie/WLED with some suggestions at the link to the source.

WLED MQTT default setup contains communications on topic starting with “wled/”. mcsMQTT will recognize this topic and customize a HS set of devices for control and status of WLED. The default setup for HS4 /deviceutility is shown in Figure 387. Note that the /c topic is represented in two color spaces. One is RGB as a color picker and the other as three sliders for Hue, Saturation and Value. Either can be used and the plugin will keep synchronization with WLED.

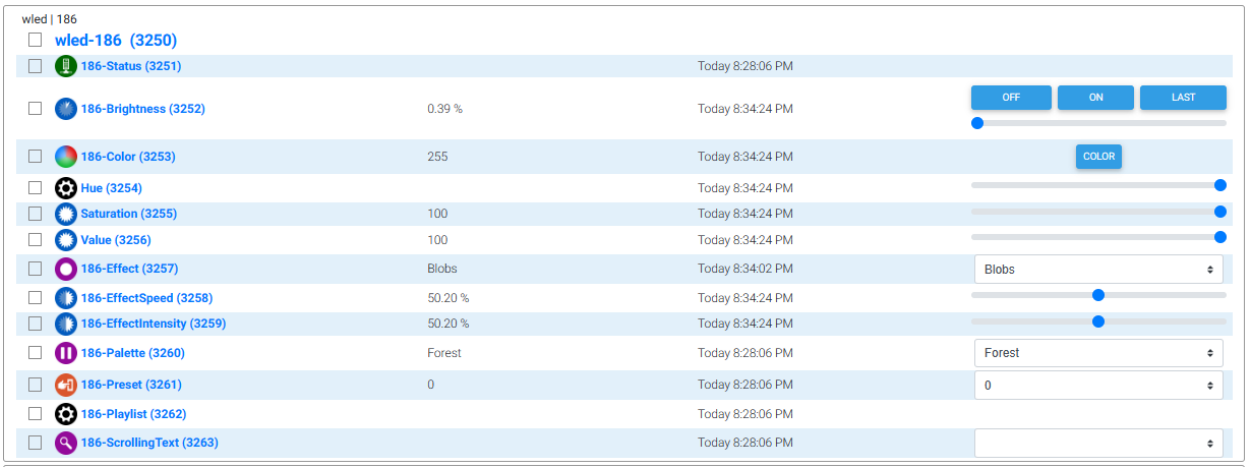


Figure 387 Default WLED HS Devices

Staged and ready for HS device creation, if desired, are a number of other end points reported on the /api topic. The full set is shown in Figure 388.

It is also possible to create other end points based upon the API defined in the links above. In this case new mcsMQTT device would be created manually from the Edit tab. If any staged device on the Associatiion tab is to be included as a HS device there may be some edits needed on the Edit tab after clicking the “a” checkbox. These edits will depend upon the contents of the API schema vs. the initial guess provided by mcsMQTT.

7	☐	☐	☐	☐		wled/55b1f4/v				
8	☐	☐	☐	☐		Sub: wled/55b1f4/v:ac	255	☐		2020-01-30 19:22:30
9	☐	☐	☐	☐		Sub: wled/55b1f4/v:cl	4	☐		2020-01-30 19:22:30
10	☐	☐	☐	☐		Sub: wled/55b1f4/v:cs	0	☐		2020-01-30 19:22:30
11	☐	☐	☐	☐		Sub: wled/55b1f4/v:ds	WLED	☐		2020-01-30 19:22:30
12	☐	☐	☐	☐		Sub: wled/55b1f4/v:fp	0	☐		2020-01-30 19:22:30
13	☐	☐	☐	☑	1852	Dev: |wled|55b1f4-Effect Sub: wled/55b1f4/v:fx Pub: the following Topic on Device command wled/55b1f4/api	0	☐	☐	2020-01-30 18:43:59
14	☐	☐	☐	☑	1854	Dev: |wled|55b1f4-EffectIntensity Sub: wled/55b1f4/v:ix Pub: the following Topic on Device command wled/55b1f4/api	76	☐	☐	2020-01-30 18:46:15
15	☐	☐	☐	☐		Sub: wled/55b1f4/v:md		☐		2020-01-30 19:22:30
16	☐	☐	☐	☐		Sub: wled/55b1f4/v:nd	60	☐		2020-01-30 19:22:30
17	☐	☐	☐	☐		Sub: wled/55b1f4/v:nf	1	☐		2020-01-30 19:22:30
18	☐	☐	☐	☐		Sub: wled/55b1f4/v:nl	0	☐		2020-01-30 19:22:30
19	☐	☐	☐	☐		Sub: wled/55b1f4/v:nr	1	☐		2020-01-30 19:22:30
20	☐	☐	☐	☐		Sub: wled/55b1f4/v:ns	0	☐		2020-01-30 19:22:30
21	☐	☐	☐	☐		Sub: wled/55b1f4/v:nt	0	☐		2020-01-30 19:22:30
22	☐	☐	☐	☑	1853	Dev: |wled|55b1f4-EffectSpeed Sub: wled/55b1f4/v:sx Pub: the following Topic on Device command wled/55b1f4/api	178	☐	☐	2020-01-30 18:44:03
23	☐	☐	☐	☐		Sub: wled/55b1f4/v:th		☐		2020-01-30 19:22:30
24	☐	☐	☐	☐		Sub: wled/55b1f4/v:ws	0	☐		2020-01-30 19:22:30
25	☐	☐	☐	☐		Sub: wled/55b1f4/v:wv	-1	☐		2020-01-30 19:22:30

Figure 388 WLED /api topic end points

Support for WLED Segments was added based upon WLED 0.10.0. A segment is a division of a LED strip into smaller strips that operate relatively independently. This feature of WLED is still somewhat immature and has some limitations, but basic functionality does exist. I have observed that segment setup did not persist of power cycle of the WLED controller. A WLED segment setup of two segments is shown in Figure 389. This is done with browser URL set to the IP of the ESP8266/ESP32 containing the WLED firmware.

Figure 389 WLED Segment Definition

Segments are not exposed via MQTT or the XML API interface. They are only visible using the JSON API. To support segments an additional tab was added to the Local page. For each WLED controller a row will be present to enter the IP of the controller and to identify how many segments the controller supports.

Segment 0 is the main segment that contains all LEDs in the strip so the first logical subdivision is 1. A max of 10 is supported by WLED. Figure 390 shows this Local tab setup.

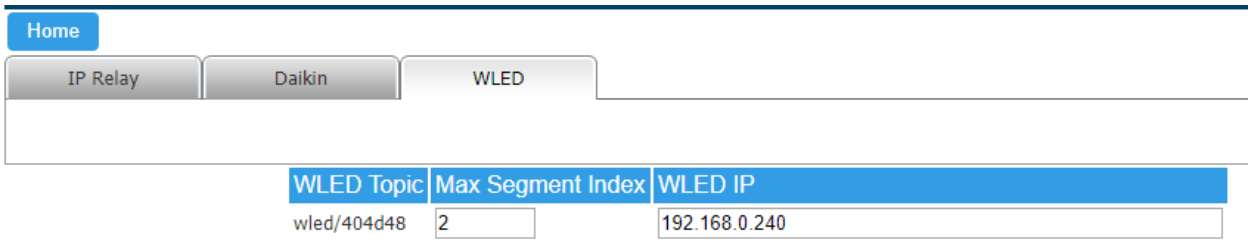


Figure 390 WLED Setup on Local Page

mcsMQTT will evaluate to align the MQTT-equivalent devices with the segments specified Any time either of the two Figure 390 fields are edited. The segment index will be added to the topic as shown in Figure 391. In this example segment 1 and the start of segment 2 are shown.

T1	T2	T3	T4	T5	T6
	404d48				
J1	J2	J3	J4	J5	J6

Show Selected Associations

Prev 0 of 36 Next

Association Table for Auto Association of MQTT Topic and HS Device

/\	o	r	e	a	ref	TOPIC	payload	h	d	lastdate
0	☐	☐	☐	☑	2743	wled/404d48				
1	☐	☐	☐	☑	2756	Dev: wled|404d48|1-Color Sub: wled/404d48/1/c:RRGGBB Pub: the following Topic on Device command wled/404d48/1/col	#220000	☐	☐	2020-05-25 14:00:53
2	☐	☐	☐	☑	2752	Dev: wled|404d48|1-Brightness Sub: wled/404d48/1/g:Brightness Pub: the following Topic on Device command wled/404d48/1	110	☐	☐	2020-05-25 14:00:53
3	☐	☐	☐	☑	2753	Dev: wled|404d48|1-OffOn Sub: wled/404d48/1/g:OffOn Pub: the following Topic on Device command wled/404d48/1	ON	☐	☐	2020-05-25 14:00:53
4	☐	☐	☐	☑	2760	Dev: wled|404d48|1-Effect Sub: wled/404d48/1/v:fx Pub: the following Topic on Device command wled/404d48/1/api	0	☐	☐	2020-05-25 14:00:53
5	☐	☐	☐	☑	2764	Dev: wled|404d48|1-EffectIntensity Sub: wled/404d48/1/v:ix Pub: the following Topic on Device command wled/404d48/1/api	0	☐	☐	2020-05-25 14:00:53
6	☐	☐	☐	☑	2762	Dev: wled|404d48|1-EffectSpeed Sub: wled/404d48/1/v:sx Pub: the following Topic on Device command wled/404d48/1/api	0	☐	☐	2020-05-25 14:00:53
7	☐	☐	☐	☑	2758	Dev: wled|404d48|2-Color Sub: wled/404d48/2/c:RRGGBB Pub: the following Topic on Device command wled/404d48/2/col	#CC3300	☐	☐	2020-05-25 14:00:53

Figure 391 Virtual MQTT Topics for WLED Segments

HS devices are created for each segment. This is shown in Figure 392. The devices created include On/Off, brightness, color, white channel if appropriate, and effects parameters.

Since segments are not visible via MQTT there is no update of the HS devices except when a segment parameter is commanded, segment setup is changed or at plugin startup. To force an update then command a segment of the controller. This command can be no change such as the On/Off device commanded into the same state as it currently exists.

Display Filters: Floor | Room | Device Type | Show All

Ref	Status	Category	Floor	Room	Name	Last Change	Control
2743			wled	404d48	wled-404d48	Today 1:58:07 PM	
2752	98%		wled	404d48	1-Brightness	Today 2:33:27 PM	
2756	4325384		wled	404d48	1-Color	Today 2:33:27 PM	420008
2760			wled	404d48	1-Effect	Today 2:33:27 PM	Solid
2764	0%		wled	404d48	1-EffectIntensity	Today 2:33:27 PM	
2762	0%		wled	404d48	1-EffectSpeed	Today 2:33:27 PM	
2753	ON		wled	404d48	1-OffOn	Today 2:33:27 PM	OFF ON
2754	69%		wled	404d48	2-Brightness	Today 2:33:27 PM	
2758	12436480		wled	404d48	2-Color	Today 2:33:27 PM	BDC400
2761			wled	404d48	2-Effect	Today 2:33:27 PM	Solid
2765	0%		wled	404d48	2-EffectIntensity	Today 2:33:27 PM	
2763	0%		wled	404d48	2-EffectSpeed	Today 2:33:27 PM	
2755	ON		wled	404d48	2-OffOn	Today 2:33:27 PM	OFF ON
2745	46%		wled	404d48	404d48-Brightness	Today 2:31:28 PM	
2747	10026831		wled	404d48	404d48-Color	Today 2:31:28 PM	98FF4F
2748			wled	404d48	404d48-Effect	Today 2:31:29 PM	Solid
2750	50%		wled	404d48	404d48-EffectIntensity	Today 2:31:29 PM	
2749	50%		wled	404d48	404d48-EffectSpeed	Today 2:31:29 PM	
2746	ON		wled	404d48	404d48-OffOn	Today 2:31:28 PM	OFF ON
2744			wled	404d48	404d48-Status	Today 2:31:29 PM	
2766	50%		wled	404d48	404d48-White	Today 2:31:28 PM	

Figure 392 WLED Segments as HS Devices

The “Scrolling Text” special effect has been added to WLED and applies to a WLED 2D configuration in the LED setup. If this effect is available and IP of the WLED controller is setup on the Local Page, WLED Tab, then mcsMQTT will create a Scrolling Text HS Feature for each WLED segment.

The setup includes some date/time-oriented selections on the control selector. These can be changed and augmented from the MQTT Page, Edit Tab of the Feature. The initial VSP are shown below where the payload field (first item) is an inline expression for a date-related function and the next is the text shown in the selector of the HS Feature. When selected, the function will be evaluated and the text delivered to the associated WLED segment as a scrolling text effect. mcsMQTT will automatically change the effect to scrolling text when this HS Feature is used.

```
(" ";                0; " ")
("<<Now()>>";        1; "Date-Time")
("<<Today()>>";      2; "Date")
("<<Time()>>";       3; "Time")
("<<NameOfDay()>>";4; "WeekDay")
```

The VSP will provide a “preset” capability that can be used from the HS Devices page or can be used as an Event Action to control a HS Device Feature.

It is also possible to send any text from a HS Event Action using a one-line script where the Feature Ref, a 0, and the desired text passed to the SendControlForFeatureByString function. The third (text) parameter can contain replacement variables and inline functions what will be evaluated at time of the event action.

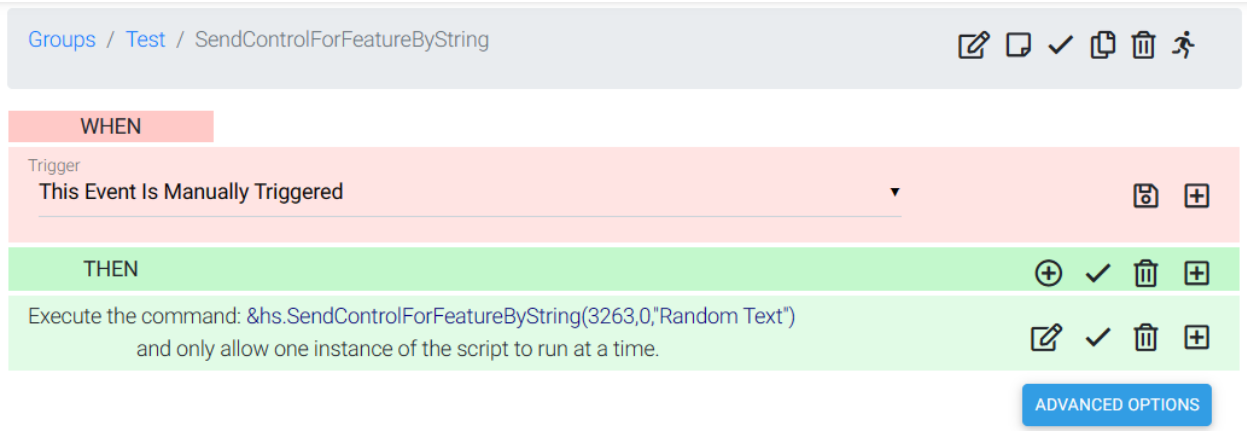


Figure 393 Event Action to send scrolling text to WLED display

For my evaluation I used a $20 8 x 32 LED panel Amazon.com: BTF-LIGHTING WS2812B ECO RGB Alloy Wires 5050SMD Individual Addressable 8X32 256 Pixels LED Matrix Flexible FPCB Full Color Works with K-1000C,SP107E,etc Controllers Image Video Text Display DC5V : Tools & Home Improvement which is similar to the other messaging signs described in this document. They can be cascaded and tiled for long or large matrix displays. This means the display can be partitioned into segments with one or more of the segments doing scrolling text while others are used for other purposes.

26 Plex Integration

Tautulli GitHub - Tautulli/Tautulli: A Python based monitoring and tracking tool for Plex Media Server. provides a MQTT conduit with Plex. The Plex status is reported in a JSON payload containing subject, body and topic keys. Unfortunately, the body key uses escape-encoding rather than JSON-encoding so standard means to decode the body cannot be used.

To deal with this mcsMQTT looks for topics starting with “Plex” and a JSON key of “body”. In these cases it will discard the subject and topic parts of the payload the reformat the body part to be standard JSON. When Tautuili is configured, assure that the topic starts with “Plex” to enable this capability.