Adding/Modifying MQTT Devices

This document deals with adding MQTT enabled devices to the Oasis Spa controller. These devices can be any (IoT) device that is enabled with MQTT. MQTT is a communications protocol used by many small networked devices such as Relays, Light controllers, lights, and switches to just name a few. The whole process has a few steps and several files that need to be modified. There is nothing too complicated to do, just make sure you get the formatting correct.

You can add as many MQTT devices as you'd like (until you run out of unique GPIO pins on your Raspberry, but multiple MQTT devices can listen to a single pin). For example, in my Oasis Spa setup I have a shower which is connected to the same outdoor tankless heater as the circulation pump. The shower is activated by a Sonoff Touch device. This triggers a cascade of actions:

1. Shuts off the main spa circulation pump (spa pump is 110v connected to a Sonoff TH)

2. Closes the water flow from the heater to the spa (12v valve connected with a Sonoff SV)

3. Opens a valve to let fresh water flow through the heater and to the shower. (12v valve connected with a Sonoff SV)

Also, traditional GPIO pin (WiringPi) controlled relays, switches or devices can be added. For example, if your spa heater takes power beyond spec of a 16amp networked IoT switch, then you can directly hook a high-powered relay up to your Raspberry for normal GPIO control.

As mentioned above, I use Sonoff IoT relays flashed with ESPurna firmware. ESPurna was chosen because it is simple, and has the features needed. Sonoff makes a full line of 'hackable' devices that can accept most any firmware written for the esp8266 chip. I use a mix of low voltage (Sonoff SV) and normal 110v (SonoffTH, Sonoff basic, Sonoff Dual etc..) relays in my setup. The Tub itself is a Coleman/Bestway inflatable tub. The motivation for this project is this inexpensive spa-in-a-box has a very weak heater and frustratingly unconfigurable controls in this all-in-one device from Amazon. The tub itself is great! Portable, and very well made thick plastic. Unfortunately, the heater can only raise the temperature less than 3° an hour. (in cool weather, even worse) With a tankless propane water heater, we can now heat the tub at a rate of over 15°/hr. With the added benefit of being able to control it from anywhere in the world. With the addition of MQTT communication, I can make the spa experience seamless, configurable, and easy to use.

My setup looks like this:

PUMP: 110v controlled with Sonoff TH10 sensor/relay module. 16gpm flow rate, but with added hoses and heater, and elevation rise, it is more like 7-10gpm. Perfectly acceptable for this tub which has only one hot water input and no jets.

Tankless propane water heater: 3v controlled with a Sonoff SV and a small buck converter to get to the correct voltage. It was designed to be run from 2x alkaline batteries, but I've directly hooked the Sonoff relay output to the battery terminals, so it can be switched ON/OFF. One 20lb tank of gas lasts about a month if I use the built-in heater to maintain heat above 90°f/32°c during the day. If I use the propane heater exclusively, I get about 2 weeks from each tank (refills are $11us, a month of electricity is around $40 if I maintain ready-to-tub heating with the built-in heater) The propane and tub insulation is efficient enough that I don't heat the spa at night aside from frost protection. If I run the pump at night, the spa cools and looses heat quicker (about 1.5° from tub to return) therefore, I only run the pump during the day. In my current setup, there is about 20ft of hose and the heater which when no flame is applied acts as a heatsink, removing heat from water flow.

Lighting: 12v LED strips controlled with a Magic Home LED device. This also accepts the ESPurna firmware and supports full RGBW MQTT control

Temp Sensors: Several Sonoff TH10 sensors. Currently only one of the relays are being used, for the pump. The rest are simply reading temperature and their relay/switches can be used later for device expansion. The Sonoff th10 and TH16 sensors are normal A/C relays with a ¼" plug for a DS18B20 (OneWire) waterproof sensor wire. The sensor and relay are independent from each other, and it is not necessary to have the relay output connected. Temp readings are taken by ESPurna's web API, which outputs via the website into a text file. I'm utilizing 3 sensors:

A. In the filter basket for live tub temp readings

B. Inline just after the heater to measure incoming water temp.

C. An ambient air temp for good measure

Functionality

The Oasis Spa Controller was built on Rick Feenstra's excellent project. He built it for his own purposes, controlling a traditional Hot Tub with traditional pumps and heater. He built the website which controls the Raspberry Pi's GPIO pins directly. The functionality is perfect, but I wanted to be able to use wireless IoT switches. Since my tub is an inexpensive inflatable model, and I'm using propane heat, my power requirements don't get into high-voltage territory. Instead of re-writing the website's functions to adapt it to MQTT communication we are running a program which listens to the state of the Raspberry's GPIO pins. If this program detects that a change has been made from ON/OFF it will fire off the appropriate MQTT message to the device associated with that Pin. In this way the website will still function as Rick set it up to do, as well as adding new IoT functionality.

Currently, the process for assigning, and configuring MQTT devices is a manual one. You'll have to edit the config file yourself for every device you want to add. It isn't ideal, but I don't have the scripting chops to make it happen via the website directly, maybe you do? This functionality is accomplished with the GitHub project: Hallonlarm. The program comes installed with the Oasis Spa Controller Raspberry Pi image. Your job will be to supply the MQTT Broker address and configure each MQTT device so the program knows how to communicate to/from and which pin it should be assigned to. Additionally, there is handy functionality that can invert ON/OFF (1/0) state. Let's get started!

Be sure to backup a copy of the file you're modifying before you save it just in case you need to roll back.

Setup

This document assumes you have got your Pi's SD card set up with Oasis Spa's image and everything is setup and running. You can access the http://oasis-spa website (or whatever name you've assigned to it). We'll move into setting up MQTT relay switches now. They will control the pump, heater and whatever lighting or other devices you'd like to have Oasis control.

Hallonlarm's files have been symlink'd to a samba share named 'Config'. If you have name resolution on your network up and running correctly browsing to \\oasis-spa\config will get you there (if that doesn't work for your network, \\RASPBERRY_IP_ADDRESS\config will work. The default username/password is pi/raspberry . If you're on a domain network oasis-spa\pi and raspberry. Please change the password. Hallonlarm is the "middle man" listener program that detects if one of the Pi's pin states have changed. The config files below are how we set it up for your network.

There are two Hallonlarm files of interest

1. Hallonlarm.service is where you specify your MQTT broker. One needs to be running on your network. A MQTT broker is the "server" that all the devices connect to and publish/subscribe to. All your switches/relays/temp sensors should use the same broker. The Oasis Spa's website is both a Publisher and a Subscriber. If it detects that a MQTT message has been published for one of its configured devices, Hallonlarm will change the state of the Raspberry pin. If you turn on a device from the web interface, Hallonlarm will detect a pin state change and publish a message to the broker to change the the associated device. I use an external MQTT broker for this on

my network since I use other MQTT devices in my home, but you can use the Raspberry’s internal broker, which is the default. (to do!) You don’t have to change anything in this file to make Oasis Spa work, but if you already have a MQTT broker this is where you place It’s IP address/port.

2. Hallonlarm.config is where all the MQTT IoT devices are defined and mapped. Here’s an example with items in bold annotated notes for illustrative purposes.

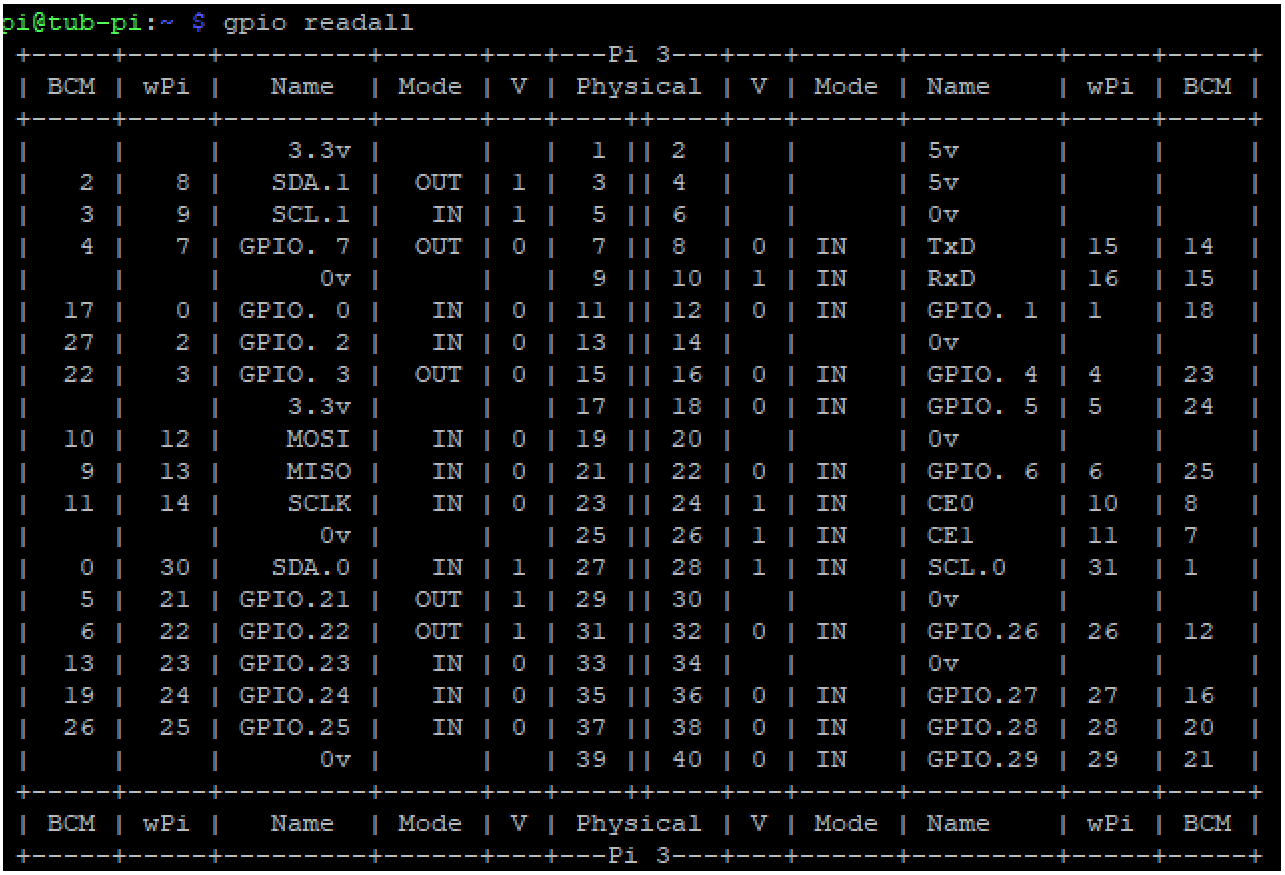
```
/*
PUMP RELAY
Friendly name for finding it in this config file. Below is where the device is defined by it’s topic. All messages
from this device would append to the tub/pump/relay/0 topic.
This is how Hallonlarm listens to each device.
 */
device "tub/pump/relay/0" {
  /* Published name */
  name = "Spa Pump"
  /* Device type */
  device = "pump"

  /* Exported features */
  feature = {
    contactSensorState = {
      info = {
        min = 0
        max = 1
        step = 1
        /*
         Default is <base topic>/<feature>/get and set. With espurna the Get/Set topic is the same.
         */
        getTopic = "tub/pump/relay/0/set",
        setTopic = "tub/pump/relay/0/set",
      }
      gpioIn = {
        /* Hallonlarm uses BCM2835 pin numbers . This is the pin that will be monitored and mapped to
MQTT states. Check your Raspberry’s model for mapping from PIN to BCM pin map.
        Remember BCM2835 pin !=Raspberry/WiringPi pin #. The tub controller uses WiringPi pin layout. So
in this case BCM pin 5 = WiringPi Pin 21 for my Raspberry Pi 3b 1.2 */
        pin = 5
        /* Set to true to invert the state before reporting to MQTT */
        invert = false

        /*
         Interval between reading GPIO state (in milliseconds). Lower value gives better precision but uses
more CPU.
         There is no reason to make this any faster. In fact, it could probably be set much lower, such as 1
second
        */
        readInterval = 200
```

```
            /*
              Don't consider state changed to until 5/10 reads of the same value.
              Useful for devices that might be flapping between states. Error checking, but also adds a delay to
state change.  10 reads at 200ms apart means 2 seconds delay total.
            */
            minReadOpened = 5
            minReadClosed = 10
         }
      }
   }
}
```

It is all straight forward as you can see. Your Hallonlarm.config file comes with several devices already defined, simply modify them to your needs, or to make it easier just setup your ESPurna firmware to match! If you don't use them all, either delete or comment out the unused sections/devices.

Relay Setup in Controller Web config

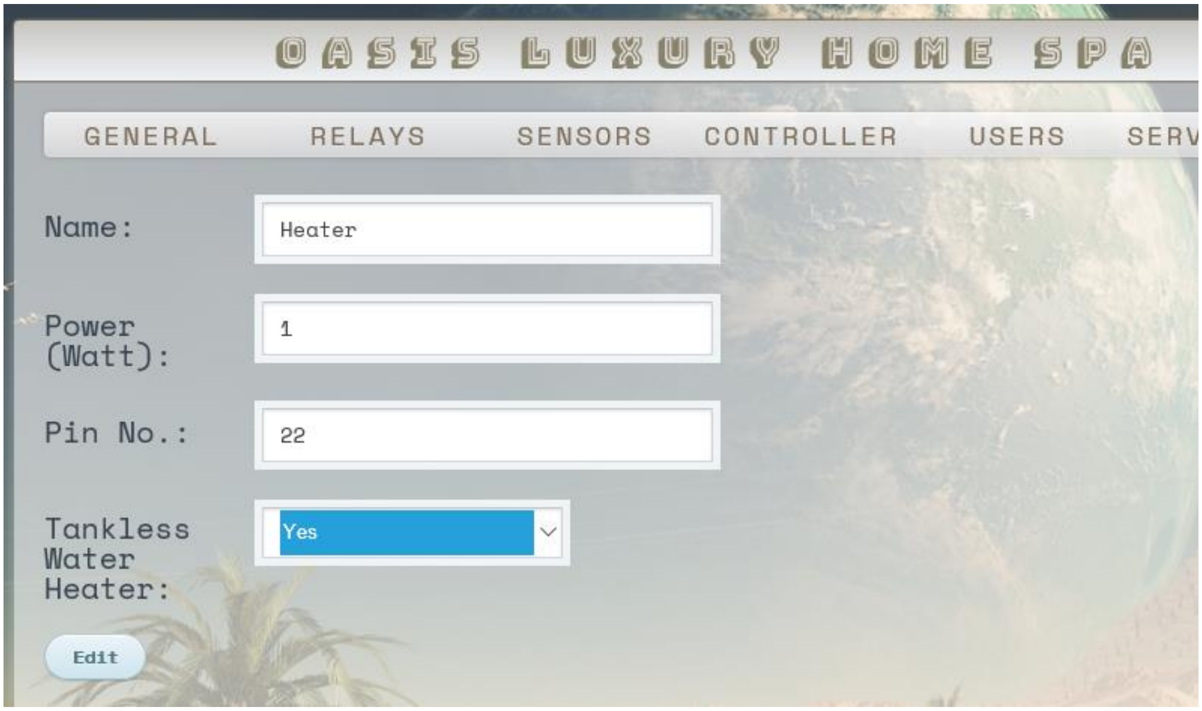
Browse to the Relay configuration section of the controller (http://oasis-spa/index.php?p=CONF.relays). There is already some relays setup there for you. This is where you can edit/rename/remap/delete if you wish. When editing or adding new relays, remember the Pin you will be using is the WiringPi pin numbering. So…in the Hallonlarm config above I used BCM pin 5, which maps to Pin 21 on my Raspberry model. Models differ, so you may need to change things here. WiringPi's website has published pin maps for all the models. To get your specific layout, use SSH to connect and type `gpio readall` this will show a map of your PI like so. Your model may vary



```
pi@tub-pi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |  OUT | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |  OUT | 0 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |  OUT | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |  OUT | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |  OUT | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

Tankless Heaters

Tankless water heaters by law these must cut off gas/electricity after 20 minutes of operation. To reset them, you must power cycle the heater. There is an option when creating/editing a relay for this. When that is selected, your heater will cut off operations for 1 minute after 18 minutes of continuous operation. It will then reignite to resume heating after one minute. If you are using a traditional electric spa heater, or any other type of device, be sure to select NO to the tankless option.



Name: Heater

Power (Watt): 1

Pin No.: 22

Tankless Water Heater: Yes