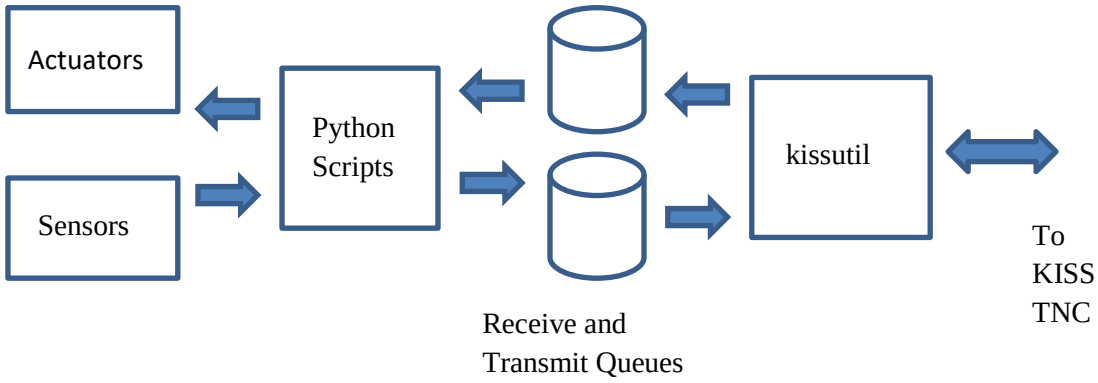
Ham Radio of Things (HRoT)

Part 2 – Build a "Thing"

In Part 1 of this series, we discussed the Publish / Subscribe model often used for the **Internet of Things** (IoT). We also installed a "Broker" and added a gateway so it can be used over Ham Radio with specially formatted APRS "messages."

Now we will build a multipurpose "thing" with different sensors and actuators. This also uses some of the components we previously covered. We already know how to set up a radio and a KISS TNC on a Raspberry Pi. (See xxx for more deails). We also need **kissutil** as used in Part 1. Don't start up **kissutil** until instructed.



This time we will use Python scripts to:

- Subscribe to topics for our actuators.
- Monitor the incoming Receive Queue and command the actuators.
- Watch the sensors and send messages periodically and/or when values change.

Build a "Thing"

I used an old Raspberry Pi model 1 because I have an extra one sitting around but there are certainly other possibilities.

Install Circuit Python Libaries:

```
sudo pip3 install --upgrade setuptools

pip3 install RPI.GPIO

pip3 install adafruit-blinka

pip3 install adafruit-circuitpython-dht
```

The devices used here don't use SPI or I^2C so it shouldn't matter if those are enabled.

Grab a breadboard and wire up some peripherals. A kit like this would be a great way to get started.

Temperature Humidity Sensor

Wire up a DTH22 Temperature-Humidity sensor as explained here.

Download HRoT application:

```
git clone https://github.com/wb2osz/hrot

cd hrot/things
```

Edit dht22.py to have your own callsign for mycall. Then run it.

```
./dht22.py
```

Don't be upset by messages like:

```
A full buffer was not returned.  Try again.
Checksum did not validate.  Try again.
```

The Adafruit documentation warned us that the DHT22 driver was in bad shape and not reliable. If all goes well, you should see the temperature and humidity printed periodically.

Look in /dev/shm/TQ and you should find files containing something like this.

```
WB2OSZ-1>APMQ01,WIDE1-1::WB2OSZ-15:pub:temperature=64.8

WB2OSZ-1>APMQ01,WIDE1-1::WB2OSZ-15:pub:humidity=26
```

RGB LED

Sorry, I don't have a part number on this. It has 4 wires. The common anode connects to 3.3 volts. Each of the others has its own 560 ohm resistor which connects to an RPi pin. For each one we need the pin number for wiring and the GPIO number for the software

	RPi pin	RPi GPIO number
Red (flat edge)	11	17
Common Anode	3.3 volts	
Green	15	22
Blue	21	9

We are going to drive these pins with pulse width modulation (PWM) to vary the brightness of each color individually. There is only a single hardware PWM output available (GPIO 18, pin 12) that we want to keep for the servo. These will be driven with software PWM. Don't worry someone else did all the hard work and it all happens magically.

Let's test it.

If you haven't done so yet, download HRoT application:

git clone https://github.com/wb2osz/hrot

cd hrot/things

Edit actuators.py to have your own callsign for mycall. Then run it.

./actuators.py

We are going to manually stick packets into the receive queue. Be sure to change the addressee to the station you used for "mycall" above. Remember to add spaces if the addressee is less than 9 characters.

```
echo 'X>Y::WB2OSZ-1 :top:rgbled=(255,0,0)' > /dev/shm/RQ/1

echo 'X>Y::WB2OSZ-1 :top:rgbled=(0,255,0)' > /dev/shm/RQ/1
```

```
echo 'X>Y::WB2OSZ-1 :top:rgbled=(0,0,255)' > /dev/shm/RQ/1

echo 'X>Y::WB2OSZ-1 :topx:rgbled=(100,100,100)' > /dev/shm/RQ/1
```

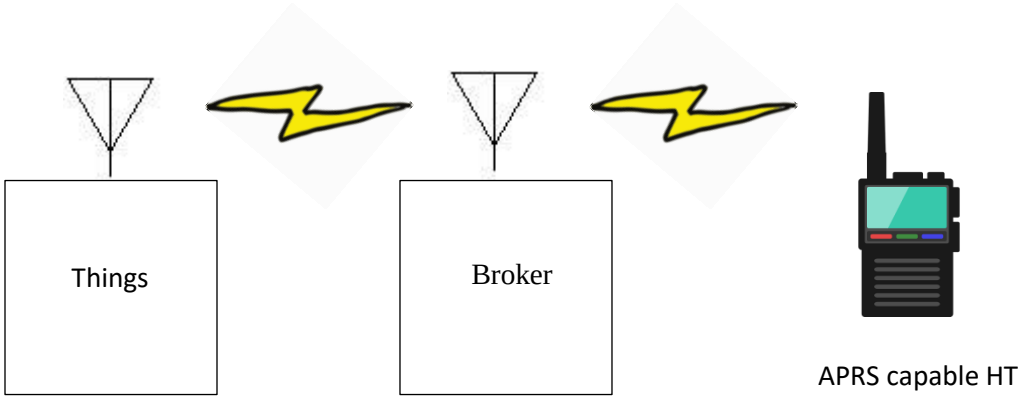
The LED should change to red, then green, then blue, then a dimmer white. This also demonstrates that the values are not limited to just a number.

Servo, Motion Sensor, etc.

Get back to this later. I want to get on with the final end to end test.

Let's Put it All Together

This is what we are going to attempt to do. Be sure to use appropriate callsigns (not mine) if you do this over the radio. We have three different radios and associated equipment.



Preparation:

Make sure that

(1) Radios are all operating correctly. Associated TNCs are OK.
(2) **kissutil** and **hrotgw** are running on the broker computer.
(3) **kissutil**, **actuators.py,** and **dht22.py** are running on the "thing" computer.

This is what should happen:

The "thing" computer:

- Subscribes to the "rgbled" topic.
- Publishes "temperature" and "humidity" values periodically.

The HT (D72 or similar):

- User manually keys in a publish message, to the broker station, with rgbled topic and value.

The broker:

- Sends a “top” message to the thing station because it subscribed to “rgbled.”

The “thing” computer:

- The LED color should change in response to the message sent from the HT.

The HT:

- User manually keys in a subscribe message for the temperature topic.
- Should start receiving temperature messages.

Be Creative

In the example here, I used bits and pieces just sitting around. There are countless other possibilities. Let your imagination go wild.

Radios:

- Rather than an HT, how about a little transceiver module such as the SA818? Has anyone built a Raspberry Pi HAT with this or something similar?
- ULARI Transceiver

Audio interfaces:

- USB audio adapter.
- Use GPIO pin from USB audio adapter for PTT.
- DRAWS.
- Some single board computers, such as the Nano Pi NEO have built-in audio input so a separate "soundcard" would not be needed.

Sensors / Actuators:

- Look around. What would you like to monitor or control remotely?
- How about a motion sensor to detect misbehaving cats that go on the kitchen table? Respond in a way to break them of that habit.

Computing:

- Control devices based on data from sensors.
- Make status information available on a web page.
- Log information to a database.
- Send e-mail or text messages when interesting events occur.

Future Subjects

Future installments might contain more advanced subjects such as:

- Add a computation part to process sensor information and decide how to command actuators.
- Authentication to prevent others from controlling your devices. (Is someone impersonating the sender?)
- Quality of Service (QoS) – Improving the chances of messages getting there. At a price, of course.
- Having the HRoT gateway communicate directly through an IGate besides going over a radio channel.
- More ready to go examples of peripherals.
- More interesting use cases.
- … and anything else you suggest.

Conclusion

If we want to communicate, we need a common language. This has been an introduction into how the Internet of Things can be extended over Ham Radio. The information here should be complete enough for you to replicate these results. By using industry standards at the core, we can take advantage of the MQTT ecosystem rather than having to reinvent everything and ending up with incompatible ad hoc approaches that don't talk to each other.