Ham Radio of Things (HRoT)

Part 1 – Getting Started

A while back, K7UDR gave a talk about the "***APRS of Things***." In his real life example, there was cabin on an island with no electricity. A generator would automatically start up periodically and run for a while to charge batteries. It would be bad if the generator did not start and worse if it kept going and depleted the fuel. How could he know if there was a problem? Without Internet access on the island, Ham Radio was chosen to convey this important information.

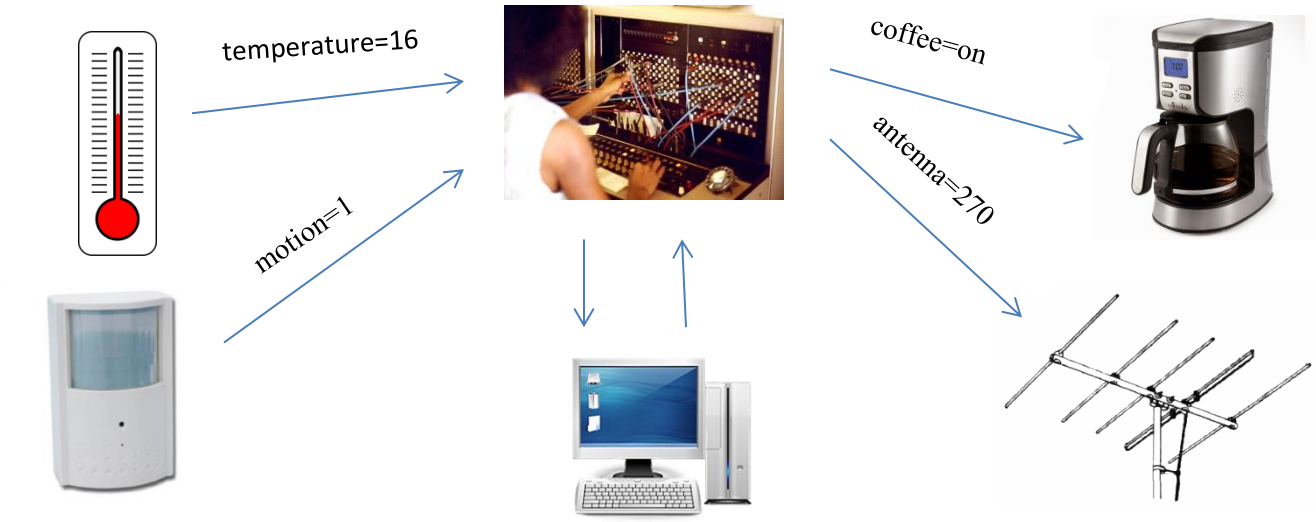
There have been other mentions of merging Ham Radio with the Internet of Things but only ad hoc incompatible narrow applications. Here is a proposal for a more general standardized method so different systems can communicate with each other.

Internet of Things

With billions of computers and mobile phones (which are handheld computers) all connected by the Internet, the large growth is now expected from the "Internet of Things." What is a "thing?" It could be a temperature sensor, garage door opener, motion detector, flood water level, smoke alarm, antenna rotator, coffee maker, lights, a home thermostat, …, just about anything you might want to monitor or control.

For example, a vending machine could report its inventory levels so the delivery schedule could be optimized. Farms can monitor soil moisture levels in different locations to optimize irrigation. The possibilities are endless.

How do these "things" all talk to each other? A popular protocol for connecting this together is called MQTT. Rather than everyone talking directly to everyone else, all communication goes through a "broker."



All the sensors “publish” information to the broker. This information is in the form of a name, called a “topic,” and a value. Those who are interested in obtaining information “subscribe” to the desired topics and the broker passes along values provided by the sensors. Computing elements can monitor appropriate sensors and send commands to the actuators, commanding them to do something. It’s a simple and very flexible approach.

In the first part of this series we will set up a broker with a gateway to ham radio. In the second part, we will a multi-function “thing.” You are encouraged to play along at home and develop your own innovative uses.

Install the Broker

Our first step is to install “Mosquitto” (note the double T) for the broker.

In this tutorial, the Raspberry Pi is being used. A few minor adjustments would need to be made to the procedure for a different type of Linux system.

Adapted from: Installing MQTT Broker Mosquitto on Raspberry Pi.

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key

sudo apt-key add mosquitto-repo.gpg.key

cd /etc/apt/sources.list.d/

sudo wget http://repo.mosquitto.org/debian/mosquitto-buster.list

sudo apt-get update

sudo apt-get install libmosquitto-dev libmosquitto1 libmosquittopp-dev
libmosquittopp1 mosquitto mosquitto-clients
```

That was easy.

Fun with Topics

"Topic" names are composed of letters, numbers, and slashes. The slashes allow you to organize related information into a hierarchical structure, much like directories (or folders) in a computer file system.

building1/floor1/room101/temperature
building1/floor1/room101/humidity
building1/floor1/room101/lights

building2/floor3/room307/temperature
building2/floor3/room307/humidity
building2/floor3/room307/lights

This is useful for keeping things organized and allows use of pattern matching (wildcard characters) to select related information. For example, you could subscribe to the temperature for all rooms, on all floors, of building1 by specifying "building1/+/+/temperature".

Let's see it in action.

(1) In one command window, type:

```
mosquitto_sub -v  -t coffee
```

(2) In another command window type:

```
mosquitto_sub -v  -t  antenna
```

(3) In yet another command window type:

```
mosquitto_pub -t coffee -m 1

mosquitto_pub -t coffee -m 0

mosquitto_pub -t antenna -m 270

mosquitto_pub -t antenna -m 45

mosquitto_pub -t lights -m red
```

Things to notice:

- The message values were delivered to the subscribers of those messages.
- No one subscribed to "lights" so it was not delivered anywhere.
- Message values can be text strings, not just numbers.
- By default, it looks for the MQTT broker on the same host. You can specify a remote host on the command line.

How to Send over Ham Radio

MQTT normally uses TCP/IP. Traditional AX.25 packet radio with "connected" mode would be one way to replace the transport layer. It is similar to TCP/IP in several ways:

- One-to-one communication between stations.
- Acknowledgment that data was received.
- Automatic retry if a transmission doesn't get through.
- Notification if the other station disappears.

However, the infrastructure is not in place. Range would be limited without having existing digital repeaters (digipeaters) available in many places. There are countless APRS digipeaters everywhere but the traditional connected mode requires a different type of digipeater and they are about as rare as hen's teeth nowadays.

Another alternative would be APRS "messages." This is a very attractive approach because there are APRS digipeaters everywhere and they are joined into a global network through Internet Gateway (iGate) stations. You can send a "message" to someone across town or on the other side of the world. Simply construct a packet like this:

```
WB2OSZ>APRS::N2GH     :Hello, Dave!
```

- Source – your station name.
- Destination doesn't matter in this case. It is usually the system type.
- ":" data type indicator for "message."
- Addressee, padded out with spaces to make 9 characters.
- Another ":" character.
- The message text.

There is also an option to put a message identifier at the end and receive acknowledgment that the message was received but we will get back to that later.

The global network of APRS-IS servers and gateway (IGate) stations connect the disjoint radio networks together.

You can find more details in Successful APRS IGate Operation.

Add a Gateway between Broker and Ham Radio

Download the source code and install the gateway.

```
git clone https://github.com/wb2osz/hrot
cd hrot
cd hrotgw
make
sudo make install
```

Just for demonstration purposes, we will give it the station name BROKER-15.

```
hrotgw -m BROKER-15
```

One side of this connects to the MQTT Broker. The other side uses a file-based queueing system. This allows you to write additional applications in any language, without the complexity of dealing with network protocols. By using a small building block approach, it is easier to mix and match different components for your particular needs.

This is how it works.

Incoming APRS packets, from the radio will be placed in the receive queue directory, normally /dev/shm/RQ. We don't have the radio part connected yet so we will test it manually.

Suppose N2GH had a remote control antenna rotator that he wants to control remotely. He could sent a message like this:

```
N2GH>X::BROKER-15:sub:antenna
```

We can simulate that by typing:

```
echo "N2GH>X::BROKER-15:sub:antenna" > /dev/shm/RQ/1
```

The **HRoT** gateway creates a subscription to the "antenna" topic for station N2GH. Open another command window, and do what we learned earlier: Manually publish something to the antenna topic.

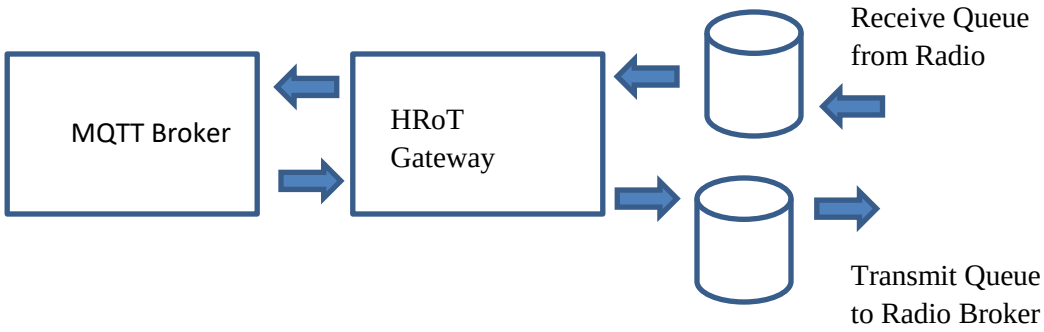
```
mosquitto_pub -t antenna -m 123
```

Now look in the transmit queue, /dev/shm/TQ. You should find a file containing:

```
[0] BROKER-15>APMQ01,WIDE1-1::N2GH     :top:antenna=123
```

This is an APRS "message" containing the topic name and value.

This is what we have so far:



In the next section, we will set up another application to:

- Put received packets in the receive queue.
- Transmit packets found in the transmit queue.

Talk to a KISS TNC

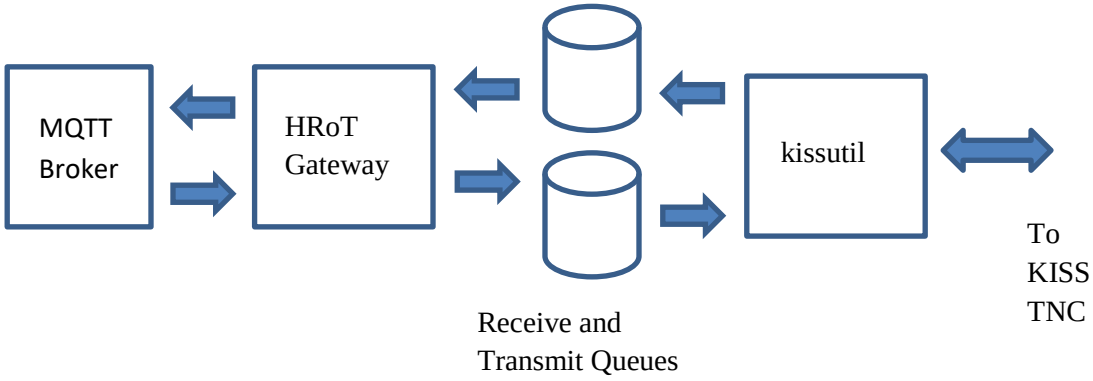
The Dire Wolf software TNC comes with a handy application called "**kissutil**." It has several functions for talking to a KISS TNC. One of them is sending to and from a file based queue just like we've been talking about. **Kissutil** can connect with a TNC over a serial port or TCP/IP so it should be usable with just about any KISS TNC.

In my case I already have Digipeater/IGate which can be used as a general purpose KISS TNC at the same time so I simply supply the network address.

```
kissutil -f /dev/shm/TQ -o /dev/shm/RQ -h 192.168.1.35
```

Received packets go into the receive queue (RQ) directory where the HRoT gateway processes and deletes them.

Kissutil takes any files from the transmit queue (TQ) directory and transmits them over the radio.



Future Subjects

That's enough for Part 1.

In Part 2 we will build a "thing" with sensors and actuators and communicate with it over the radio.

Future installments might contain more advanced subjects such as:

- Add a compute node 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 types 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.