Work Experience - RF Guide

Patrick Mintram

April 29, 2019

Contents

List of Figures

1 Introduction

This guide has been produced to help you work through the Radio Frequencies (RF) workshop as part of your work experience. In this workshop you will learn

1. What things use RF.
2. How we can make something that uses RF.
3. How using RF can expose your projects to vulnerabilities.
4. What tools we can use to help when using RF.

1.1 What you will be doing

In following this workshop you will be using tools available at home, to look at some of the information being sent through the air as RF. You will be able to see the different frequencies used by different kinds of devices, such as doorbells, WiFi, remote control cars and bluetooth connected items. You will then send some secret messages between some microcontrollers and use these tools to spy on the message as part of a man in the middle (MITM) attack.

1.2 Using this guide

There may part of this guide which aren't explained very in depth, that is because the subject of RF and signals is really complicated, so the detail has been left out. If you want to find out more there are some good overviews available online[1]. This guide is meant at more of a practical workshop than an academic exercise, so if something is glossed over a useful link will be provided in the footnotes, as you have already seen. It isn't expected that you full understand the subjects covered, but it is expected that you'll take some time in the future to have a play with the tools and techniques and learn a bit more about the things you've touched on.

1.3 Feedback

The author of this guide is keen to know what you think; the good, the bad and the ugly. Please feel free to send any comments their way, or if you're that way inclined use the github system and raising an issue or creating a pull request.

[1] http://www.ti.com/lit/ml/slap127/slap127.pdf, for example

2 Equipment

In order to complete this guide you will need the following equipment.

1. Laptop with the following:
   (a) The Software Defined Radio (SDR) Drivers. These are usually available from the manufacturers website.
   (b) The Arduino Integrated Development Environment (IDE)[2].
   (c) The RadioHead-Extras library should be installed and made available to the Arduino IDE. The library is in the `src` folder of this repo.
   (d) gnuradio[3].
   (e) A clone of this repo and performed recursively[4].
2. An SDR with an appropriate antenna for looking at the 430-440MHz frequency range. Its up to you which you use, there are loads available for a reasonable price[5].
3. Two Adafruit Feathers with an RFM69 packet radio module attached[6] as shown in fig. 1. These should ideally have antennas attached as described in the Adafruit documentation[7].

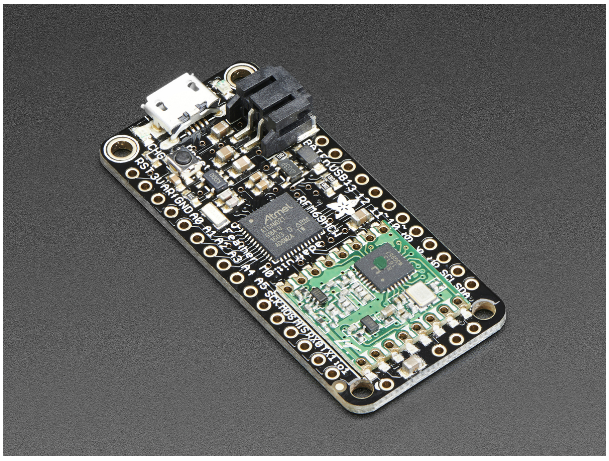


Figure 1: An Adafruit Feather M0 with RFM69 Packet Radio

[2] `https://www.arduino.cc/en/Main/Software`

[3] `https://www.gnuradio.org`

[4] git clone –recursive https://github.com/geekskick/wex-guide

[5] `https://www.rtl-sdr.com`

[6] `https://learn.adafruit.com/adafruit-feather-m0-radio-with-rfm69-packet-radio/overview`

[7] `https://learn.adafruit.com/adafruit-feather-m0-radio-with-rfm69-packet-radio/antenna-options`

3 Looking at the Spectrum

The first thing we need to understand is what the RF spectrum looks like. There are plenty of electromagnetic waves around, which you may or may not be aware of. Let's quickly revise what a wave looks like, by looking at fig. 2[8].

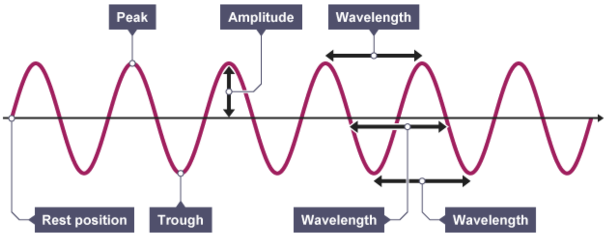
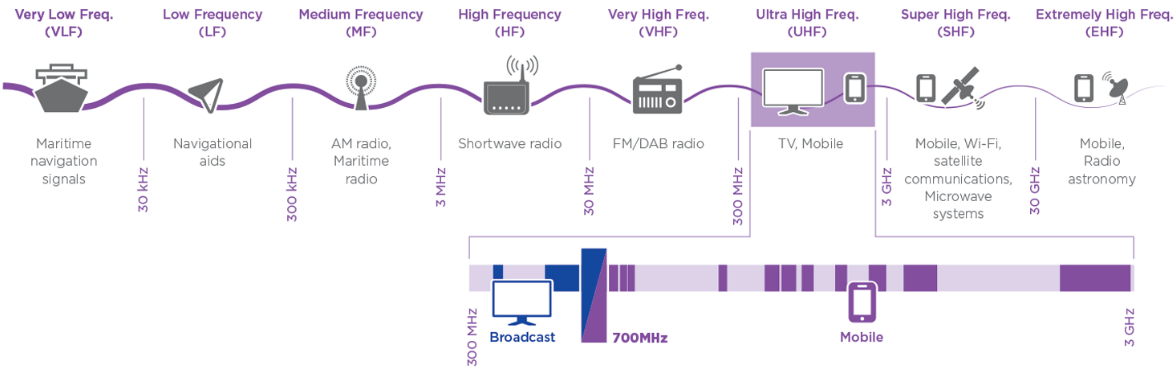


Figure 2: The RF spectrum

The key thing we care about from this diagram is the wavelength because that determines how long it takes for the wave to happen; it's period. This can be used to calculate how many times it' repeats in a second, this is measured in *Hertz* and is a result of the equation shown in eq. (1).

$$\text{Frequency (Hz)} = \frac{1}{\text{Time for one cycle of the wave (s)}} \tag{1}$$

For example a signal that repeats every 2.309469 nanoseconds has a frequency of 434MHz - it repeats 434 million times a second. There are loads of different frequencies in the RF spectrum and 434MHz fits in the ultra high frequency (UHF) part of this, as shown in fig. 3[9].



The UHF band (470 to 862 MHz) is critical to the delivery of broadcast services, and is the only frequency range that can be used for TV broadcasting. Parts of the UHF band have already been freed up for mobile use, and there is increasing pressure from mobile operators to open the 700MHz band.

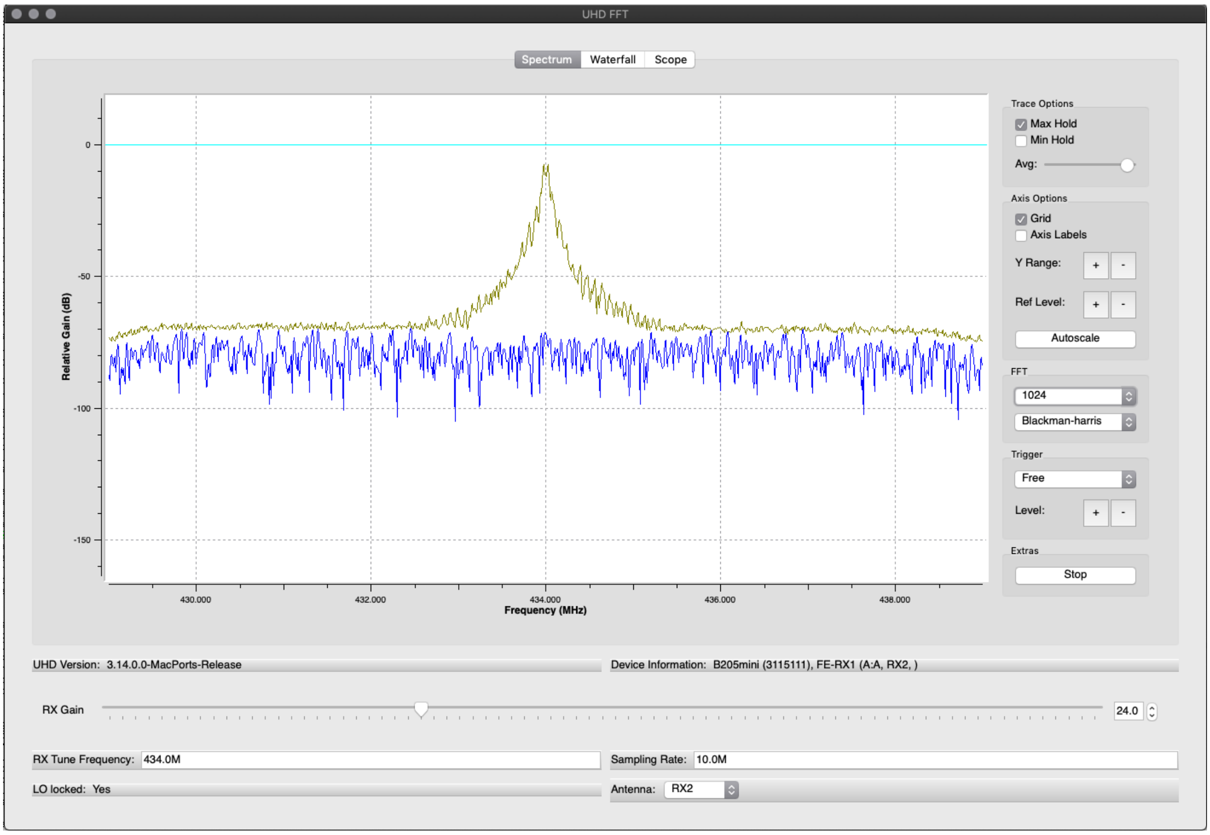
Figure 3: The RF spectrum

We can easily see the effect of these signals by using equipment which uses them; if our radio doesn't work then we know that the signals aren't present at 90MHz. What about if we want to see a signal at

[8] https://www.bbc.com/bitesize/guides/zgf97p3/revision/1

[9] https://www.ecnmag.com/blog/2017/06/understanding-rf-spectrum

434MHz though? The radios in our car only tune into parts of the very high frequency (VHF) frequencies so we can't use those. Here is where our SDR comes in useful because we can tell it a frequency to tune into (centre frequency) and we can tell it how quickly to process data (sample rate). We can then plug this into the gnuradio software to see on a graph which frequencies are most powerful, as seen in fig. 4.



☑ Max Hold
□ Min Hold
☑ Grid
□ Axis Labels
FFT: 1024
Blackman-harris
Trigger: Free

UHD Version: 3.14.0.0-MacPorts-Release
Device Information: B205mini (3115111), FE-RX1 (A:A, RX2,)
RX Gain 24.0
RX Tune Frequency: 434.0M
Sampling Rate: 10.0M
LO locked: Yes
Antenna: RX2

Figure 4: The output of the uhd_fft command with a 434MHz signal present.

The spike on the yellow line in fig. 4 shows that there is some signal present at the 434MHz frequency. We can change the settings on this window to see other signals which might be present, but first you need to run the following command from the command line to open the window.

```
uhd_fft −f 434000000 −s 10000000
```

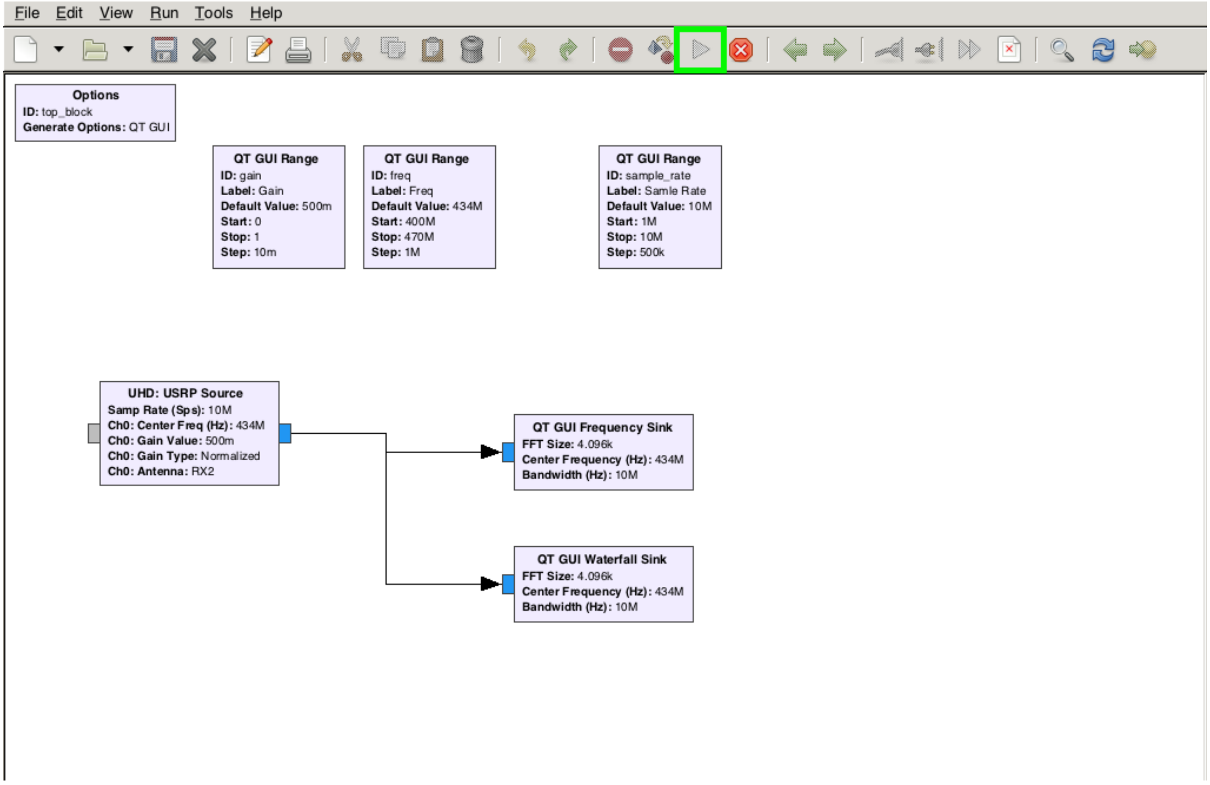
Try looking somewhere around 2.4GHz and seeing how busy it is. What do you think these signal might be?

3.1 gnuradio flows

Rather than using a prepackaged command like uht_fft we can make our own graphical user interface (GUI)s using gnuradio companion. This can be a bit weird, so to start you off one has been provided. From the command line enter:

```
gnuradio−companion
```

From here open up the provided file FFT.grc and you can click the play button highlighted in fig. 5 and see the spectrum as we did before. At this point it's worth taking some time to familiarise yourself with gnuradio using the tutorial available here: `https://wiki.gnuradio.org/index.php/Guided_Tutorial_GRC`, that way it wont be a shock if the instruction is to 'add in the Throttle block', for example.



Options
ID: top_block
Generate Options: QT GUI

QT GUI Range
ID: gain
Label: Gain
Default Value: 500m
Start: 0
Stop: 1
Step: 10m

QT GUI Range
ID: freq
Label: Freq
Default Value: 434M
Start: 400M
Stop: 470M
Step: 1M

QT GUI Range
ID: sample_rate
Label: Samle Rate
Default Value: 10M
Start: 1M
Stop: 10M
Step: 500k

UHD: USRP Source
Samp Rate (Sps): 10M
Ch0: Center Freq (Hz): 434M
Ch0: Gain Value: 500m
Ch0: Gain Type: Normalized
Ch0: Antenna: RX2

QT GUI Frequency Sink
FFT Size: 4.096k
Center Frequency (Hz): 434M
Bandwidth (Hz): 10M

QT GUI Waterfall Sink
FFT Size: 4.096k
Center Frequency (Hz): 434M
Bandwidth (Hz): 10M

Figure 5: gnuradio flow for viewing the spectrum

4 Sending a secret message

In this section we will use our feathers to send some messages to each other. Fortunately this is pretty simple to get started with thanks to the code provided by the RadioHead-Extras library provided.

Open up the *SimpleFSKSend.ino* sketch and the *SimpleFSKRx.ino* files in the arduino editor and load them on to different feathers. You should see that one is sending a message and the other is receiving it, and printing it to the serial monitor. This is our secret message!

Verify that it is working, and then try changing the code to send another message. For now, keep the settings of the RF chip the same, we will come back and change them later.

5 Spying on a secret message

In this section we will use our gnuradio flow to perform a MITM attack on our two feathers, as shown in fig. 6.

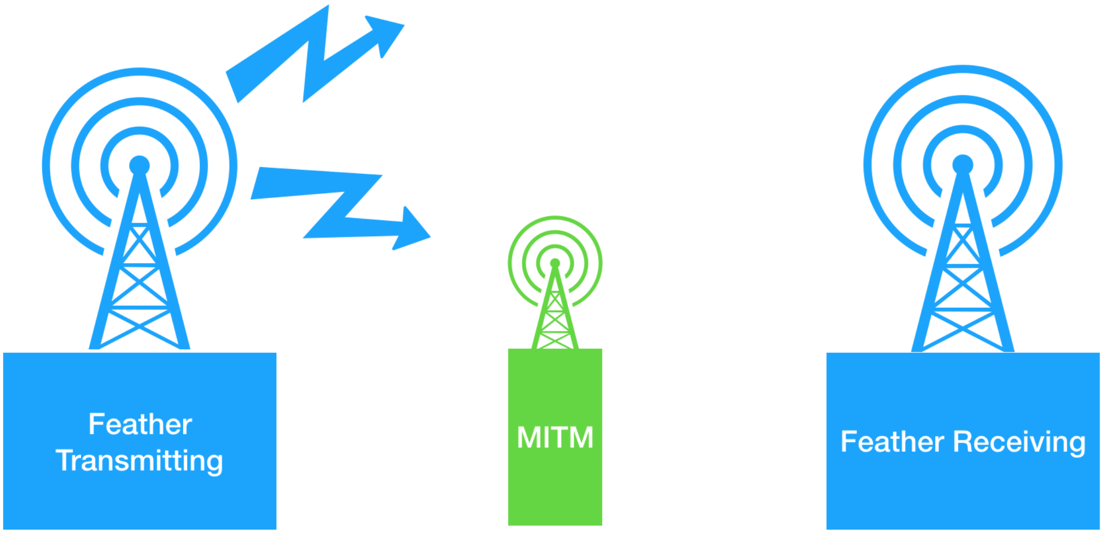


Figure 6: MITM attack on our two feathers

Using the gnuradio flow from before, find the frequency which the feather is transmitting on. It's probably going to be the loudest signal you can see, since the transmitter is close to the antenna.

5.1 Filtering

The first step is to use an low pass filter (LPF) to make sure we receive only the transmission we are interested in. Drag an LPF into the gnuradio flow, and add another QT Frequency Sink to see the output. In addition, it's a good idea to add some QT Range Widgets so you can change the parametres of the filter as it's running. Our aim for the LPF is to have an output that looks abit like barad-dur.

Figure 7: Sauron's home.

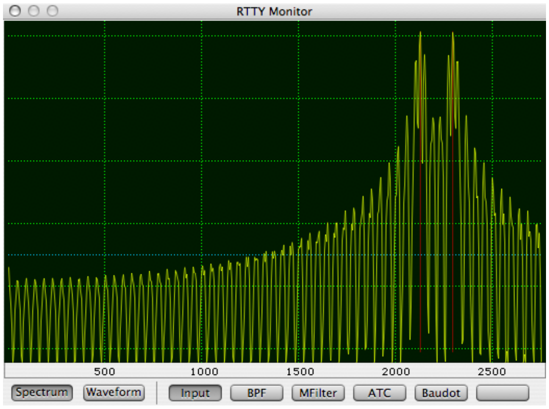


Figure 8: FSK Signal as seen on a frequency plot.

6 Real World Examples and Further Projects

6.1 Tyre Pressure Monitoring System

This technique can be used in the 'real world' as demonstrated here: `https://www.sharebrained.com/downloads/toorcon/dude_wheres_my_car_toorcon_sd_2013.pdf` where someone has used the same technology to find out the tyre pressure of different cars near them. Work through the slides and the code provided and see what you can see.

6.2 FM Reciever

You can listen to the radio, and watch the spectrum and signals change as you do it by following the guide here: `https://www.instructables.com/id/RTL-SDR-FM-radio-receiver-with-GNU-Radio-Companion/`

6.3 Satellite Images

There is a rough guide for receiving images from weather satellites available which may require some special configuration of antenna, if you're up for it!

`http://oz9aec.net/radios/gnu-radio/noaa-weather-satellite-reception-with-gnu-radio-and-usrp/`