YardStick One Replay Attack Lab

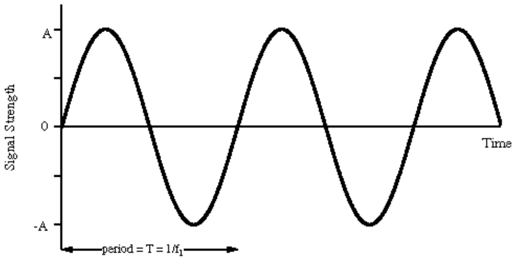
Objectives:

- Introduction to SDR recording
- Basic Signal Characteristics
- Waveform Analysis
- Reproduction of a signal vulnerable to replay attacks
- Signal Replay using the YardStick One

Background

Wireless communication is a very complex and deep topic that can encompass many volumes. As a result, this background is going to provide information relevant to the lab so that participants understand what they are investigating and reproducing at a basic level.

With regard to wireless communication, three types of modulation are typically discussed in a basic communications course. In this context, modulation is the process of altering one signal with another to convey information between communicating partners using the air as a transmission medium. The two signals involved are the carrier signal and the information signal. The carrier signal is usually a sinusoidal waveform that operates at a frequency that the sender and receiver must be tuned to in order to exchange information.

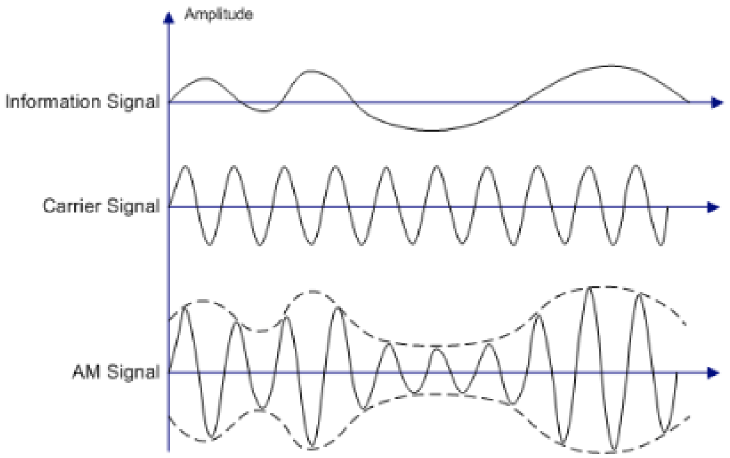


(a) Sine Wave

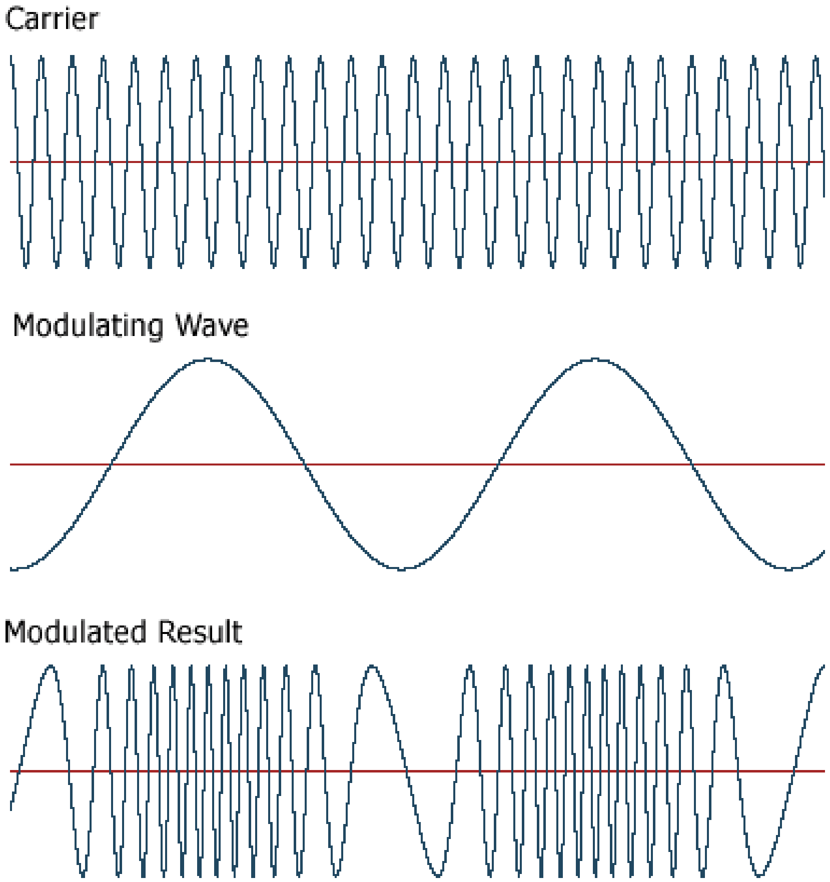
The information signal is used, along with the rules of the modulation technique, to modify the carrier in such a way that the original information signal can be recovered by the receiver of the modulated waveform by demodulating (reversing the modulation process) the received stream. An example information signal could be human speech.

The three basic transmission modulation techniques that are relevant to this lab are amplitude modulation, frequency modulation, and phase modulation. Amplitude and phase modulation should be familiar from the broadcast radio system. AM and FM radio are one application of these modulation techniques.

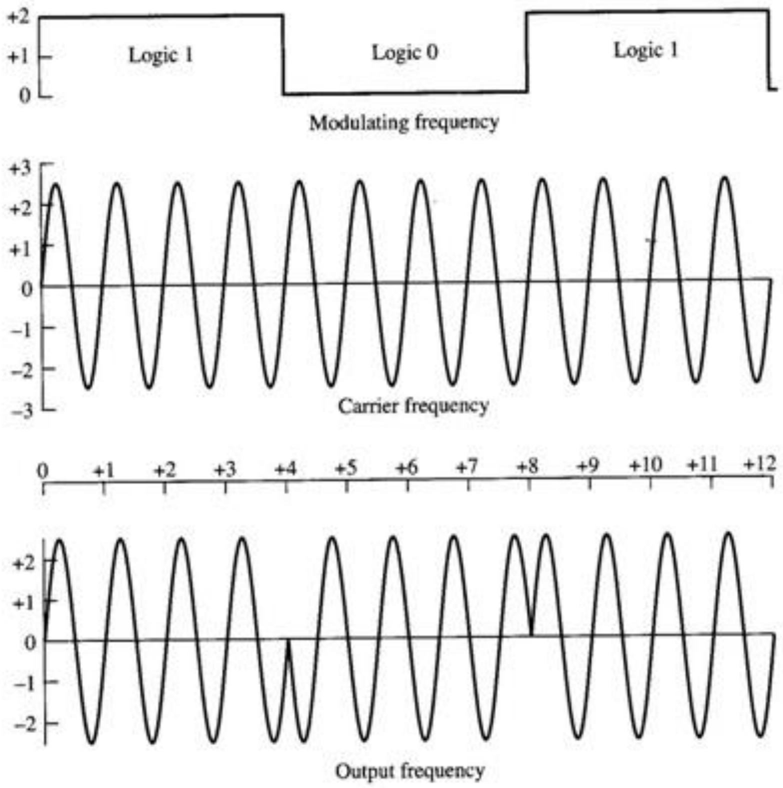
Amplitude modulation uses the information signal to modify the amplitude of the carrier waveform. Graphing the resulting waveform, the height of the peaks mimic the behavior of the input information waveform. A depiction of this behavior can be seen below.



In contrast, frequency modulation uses the information signal to modify the frequency of the carrier waveform. As a result, the modulated waveform appears to compress and decompress based on the input information waveform. This behavior can be seen in the graph below.

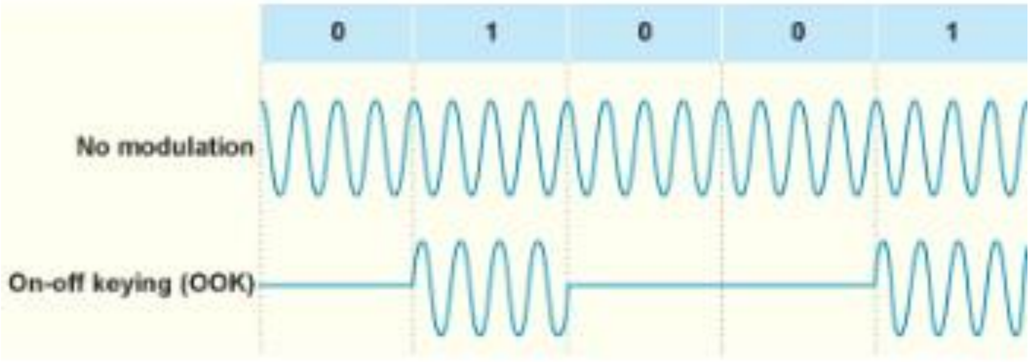


The final transmission modulation technique is phase modulation. Many variants of phase modulation exist. However, they all perform the same operation to transmit digital information. The phase of the carrier wave is modified by the input information signal in order to form symbols that represent the digital input stream. Phase modulation systems with more symbols typically lead to higher throughput and better compression. An example phase modulation technique, Binary Phase Shift Keying (BPSK) can be seen below.

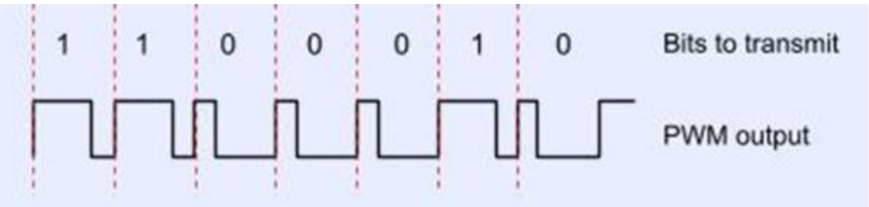


In the diagram above, the phase of the carrier wave is shifted by 180 degrees when a transition between a logic one and a logic zero occur. In BPSK, the carrier wave is inverted at each transition.

Communication systems that have to transmit a small amount of data, like the one we explore in this lab, use On-Off Keying. On-Off keying can be considered an extension of amplitude modulation where the data waveform is digital and the carrier waveform is essentially turned on and off where transitions occur.



The target device in this lab employs AM OOK with Pulse Width Modulation (PWM). Pulse width modulation is a technique to encode the On-Off Keyed signal to form symbols. The width of the pulses and gaps are interpreted by the receiver to recover the transmitted data.



Overview

Many simple communication systems that we use for various purposes (doorbells, garage door openers, key fobs, etc) employ Amplitude Modulated On-Off Keying with Pulse Width Modulation for communication. In some cases, the communication systems send information without protecting the transmitted data. This vulnerability can be present in the form of lack of encryption and/or replay protection.

In this lab, we will explore one such system (a wireless doorbell) and demonstrate replay of the target signal to generate a desired effect (ring the doorbell without pressing the button). To accomplish this task, we will be using the following list of hardware and software:

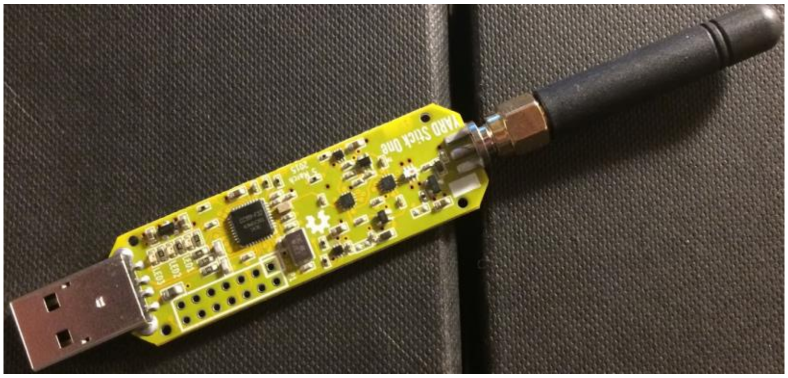
- Rtl-sdr dongle - An inexpensive receive-only software defined radio.
- Yardstick One - A sub-1GHz digital wireless transmitter device.
- GQRX - An open source SDR receiver software.
- Rfcat - An open source SDR transmitter software capable of generating an ASK OOK PWM signal.
- Audacity - An open source audio editing program that can be used to inspect the target waveform.

Wireless Doorbell Replay Attack Lab

The target device for this attack is a wireless doorbell like the one seen below. The transmitter for this device produces a signal that is vulnerable to replay attacks. We will illustrate this vulnerability by replaying the signal to cause the doorbell to ring without pressing the transmitter button.

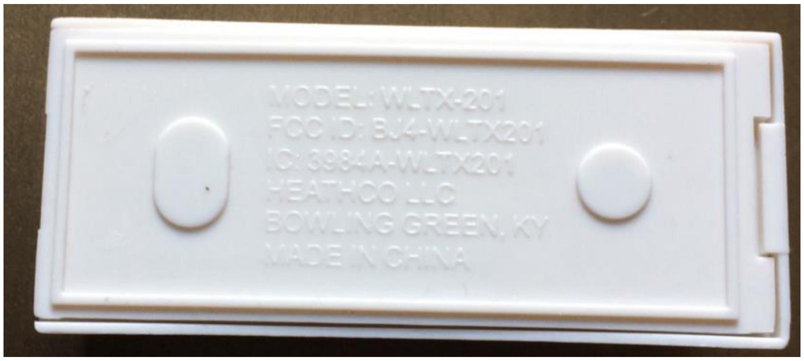
After logging onto your computer, open two terminal windows.

The RTL-SDR should be plugged into one of the computer USB ports while the YardStick One should be plugged into another. These devices can be seen below for reference.

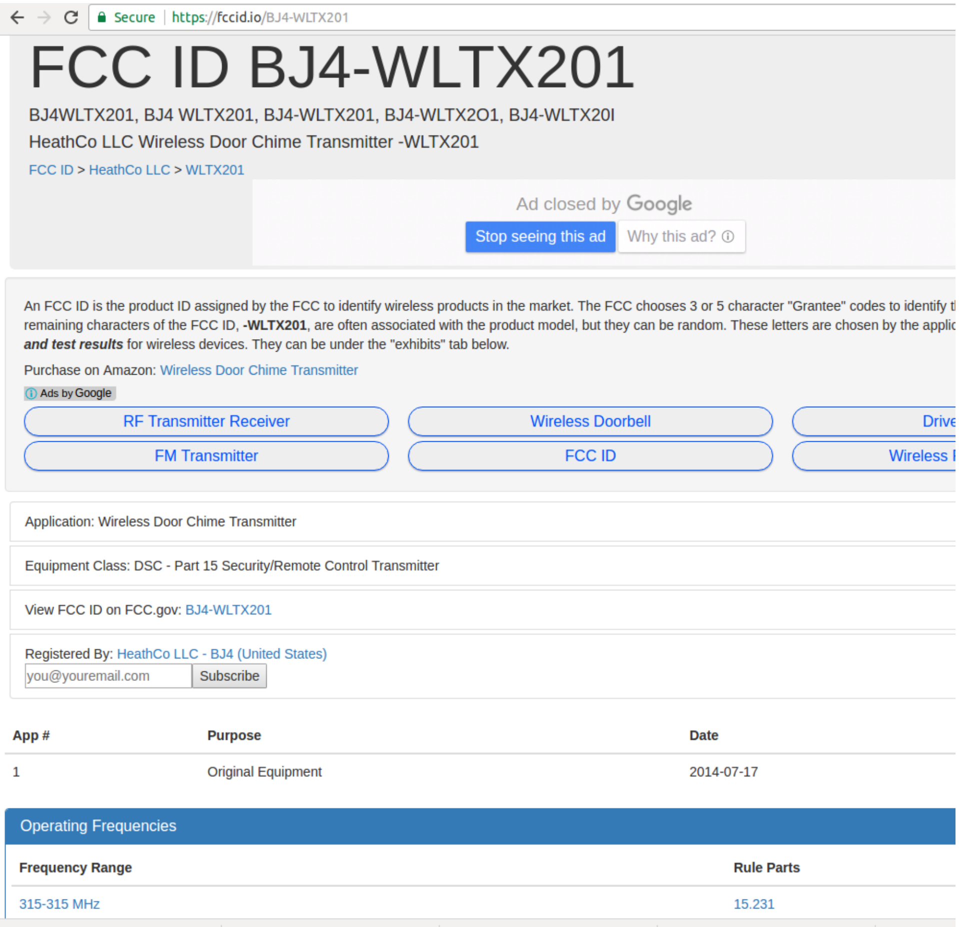


The RTL-SDR will be used to receive our transmitted signal and the YardStick One will be used to replay the signal to cause the doorbell to ring.

Before attempting to attack the target device, we need to know what frequency to monitor for the vulnerable signal. On the back of the transmitter, the FCC ID is molded into the plastic body as seen below.



The FCC ID in the graphic above is BJ4-WLTX201. This piece of information is valuable because companies must submit disclosures to the FCC for any wireless device that operates in the United States. These disclosures are public record and may include the operating frequency of the device, pictures of the internals of the device, and circuit schematics that may be useful in staging an attack. Submitting the FCC ID above to the web application at http://fccid.io results in the following result.



FCC ID BJ4-WLTX201

BJ4WLTX201, BJ4 WLTX201, BJ4-WLTX201, BJ4-WLTX2O1, BJ4-WLTX20I

HeathCo LLC Wireless Door Chime Transmitter -WLTX201

FCC ID > HeathCo LLC > WLTX201

An FCC ID is the product ID assigned by the FCC to identify wireless products in the market. The FCC chooses 3 or 5 character "Grantee" codes to identify t
remaining characters of the FCC ID, **-WLTX201**, are often associated with the product model, but they can be random. These letters are chosen by the applic
and test results for wireless devices. They can be under the "exhibits" tab below.

Purchase on Amazon: Wireless Door Chime Transmitter

Application: Wireless Door Chime Transmitter

Equipment Class: DSC - Part 15 Security/Remote Control Transmitter

View FCC ID on FCC.gov: BJ4-WLTX201

Registered By: HeathCo LLC - BJ4 (United States)

App #	Purpose	Date
1	Original Equipment	2014-07-17

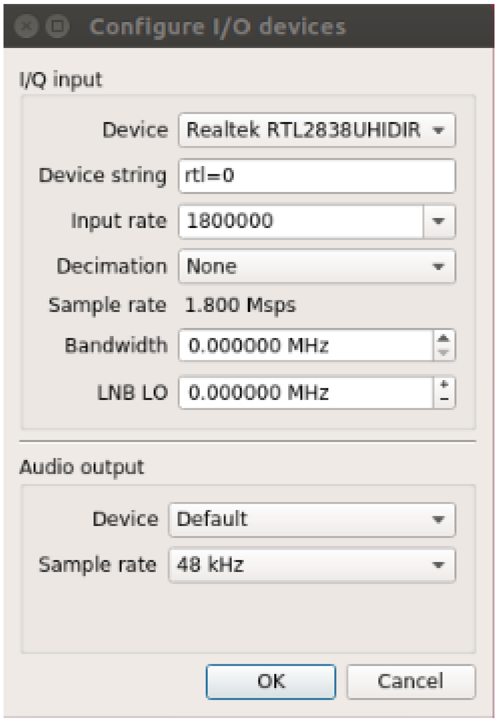
Operating Frequencies

Frequency Range	Rule Parts
315-315 MHz	15.231

The operating frequency is identified as 315 MHz and several documents describing the device are included with the record.

In the first terminal window on the lab computer, type “gqrx -r” to launch the GQRX receiver application. Initially, the application will prompt for configuration of the I/O device. This is a consequence of using the “-r” switch and allows the operator to configure the target device before initializing the radio to avoid errors.

Ensure that the “Realtek RTL2838UHIDIR” device is selected and click OK.



Configure I/O devices

I/Q input

Device: Realtek RTL2838UHIDIR

Device string: rtl=0

Input rate: 1800000

Decimation: None

Sample rate: 1.800 Msps

Bandwidth: 0.000000 MHz

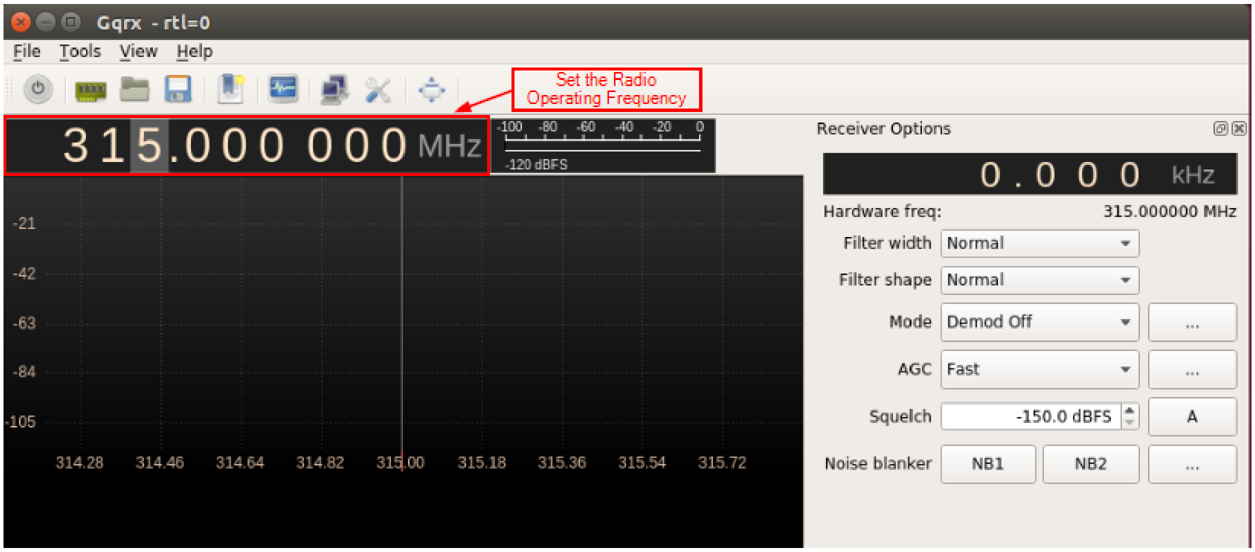
LNB LO: 0.000000 MHz

Audio output

Device: Default

Sample rate: 48 kHz

After clicking OK, the GQRX GUI application appears. Click on the numbers in the large frequency display and change the reading to be 315.000 000 MHz as seen below. This is easiest to accomplish by clicking the most significant digit and then using the arrow keys to modify the frequency.



Set the Radio Operating Frequency

Gqrx - rtl=0

315.000 000 MHz

Receiver Options

0.000 kHz

Hardware freq: 315.000000 MHz

Filter width: Normal

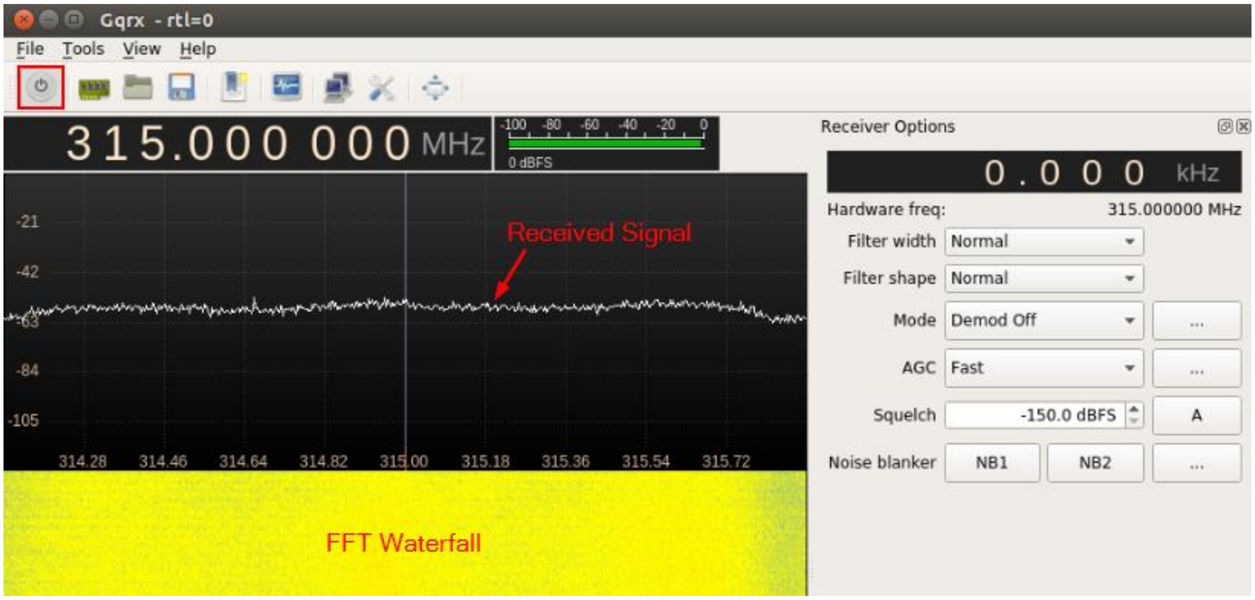
Filter shape: Normal

Mode: Demod Off

AGC: Fast

Squelch: -150.0 dBFS

Next, click the power button on the far left side of the GUI toolbar to enable the radio. You should see the signal appear and the FFT waterfall should turn yellow as seen below.



Gqrx - rtl=0

315.000 000 MHz

Hardware freq: 315.000000 MHz

Filter width: Normal

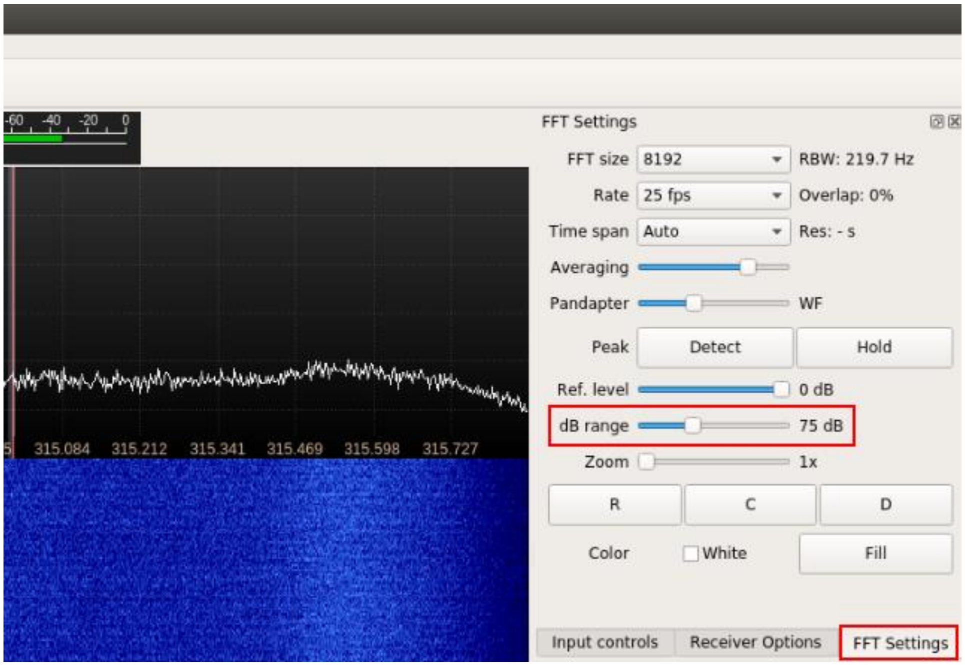
Filter shape: Normal

Mode: Demod Off

AGC: Fast

Squelch: -150.0 dBFS

Next, select the FFT settings tab on the right hand side of the display and adjust the dB range parameter to be about 75 dB. The display should turn blue and this will make it easier to distinguish the transmitted signal from noise.



FFT Settings

FFT size: 8192 RBW: 219.7 Hz

Rate: 25 fps Overlap: 0%

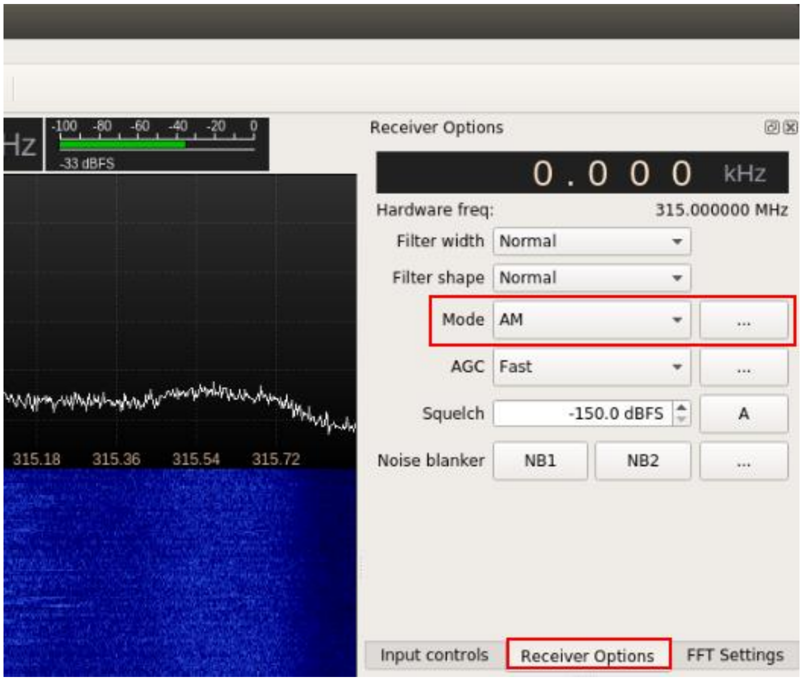
Time span: Auto Res: - s

Ref. level: 0 dB

dB range: 75 dB

Zoom: 1x

Finally, select the Receiver Options tab and set the mode to AM. This will demodulate the AM transmission so we can observe the OOK PWM signal. As a side effect, we will also hear the pulses generated by the transmitter as an audible signal.



Receiver Options

0.000 kHz

Hardware freq: 315.000000 MHz

Filter width: Normal

Filter shape: Normal

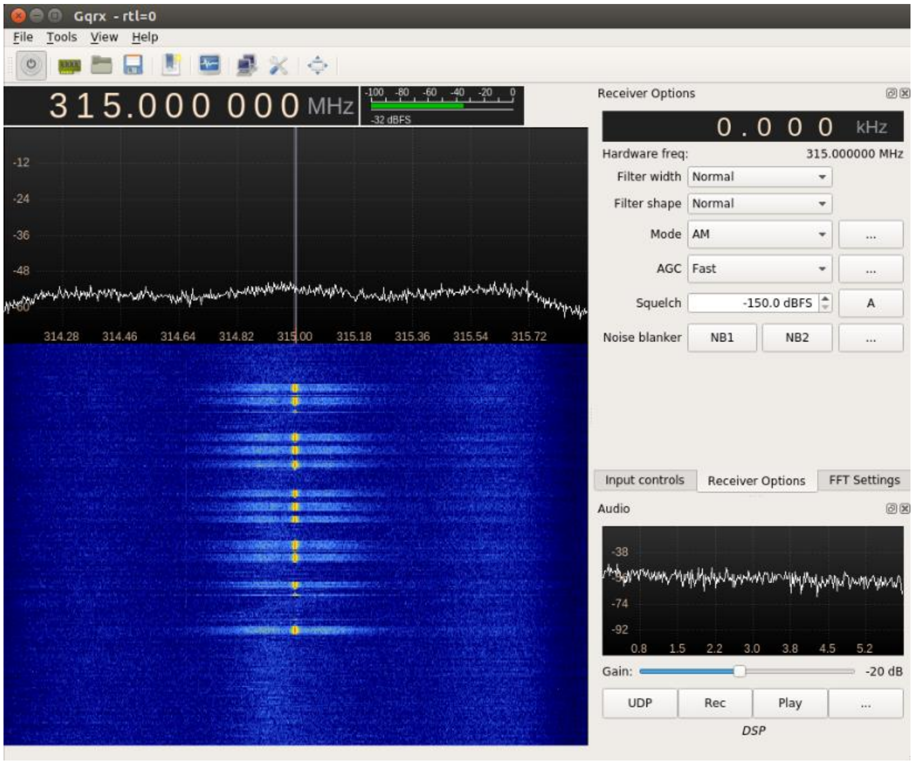
Mode: AM

AGC: Fast

Squelch: -150.0 dBFS

Noise blanker: NB1 NB2

With the receiver configured, press the button on the transmitter for the wireless doorbell. You should hear an audible signal generated from the laptop and see the signal visually on the FFT waterfall display. The signal will look like a red line surrounded by a yellow cloud as seen below. The output below is the result of several successive button pushes so the output can be seen clearly.



315.000 000 MHz

Receiver Options

0.000 kHz

Hardware freq: 315.000000 MHz

Filter width: Normal

Filter shape: Normal

Mode: AM

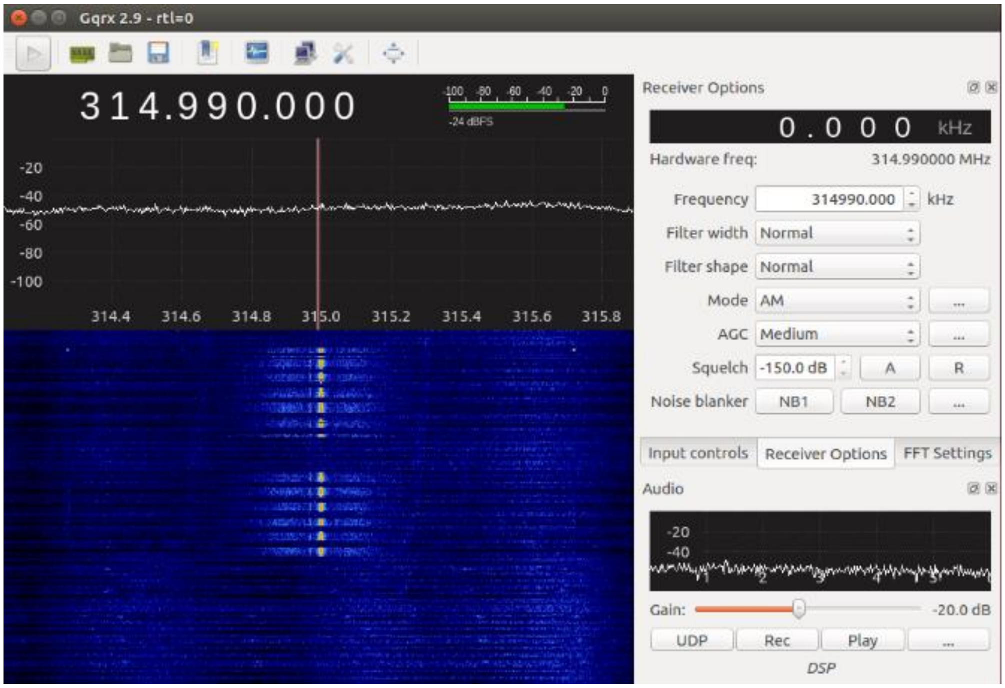
AGC: Fast

Squelch: -150.0 dBFS

Noise blanker: NB1 NB2

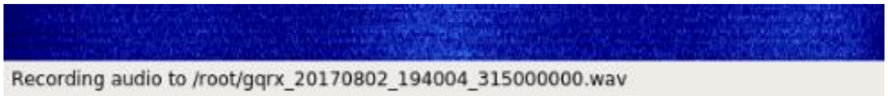
Gain: -20 dB

Before recording, make sure that the radio is properly tuned. It is likely that your transmitter (the button) is not transmitting at exactly 315 MHz. As a result, the radio must be fine-tuned to receive the transmitted signal without distortion. The objective in tuning is to line up the recorded pulses in the FFT waterfall to the tuned center frequency (center vertical line). While tuning, you also want to make sure that the value below "Receiver Options" reads 0.000 kHz. With the Audio gain up, you will hear a distinct change in the received output. An improperly tuned GQRX instance can be seen below. Although the difference between the center frequency and the transmitted frequency is minor, this will prevent proper reception of the transmitted code.



Hardware freq: 314.990000 MHz
Frequency 314990.000 kHz
Filter width Normal
Filter shape Normal
Mode AM
AGC Medium
Squelch -150.0 dB
Noise blanker
Gain: -20.0 dB

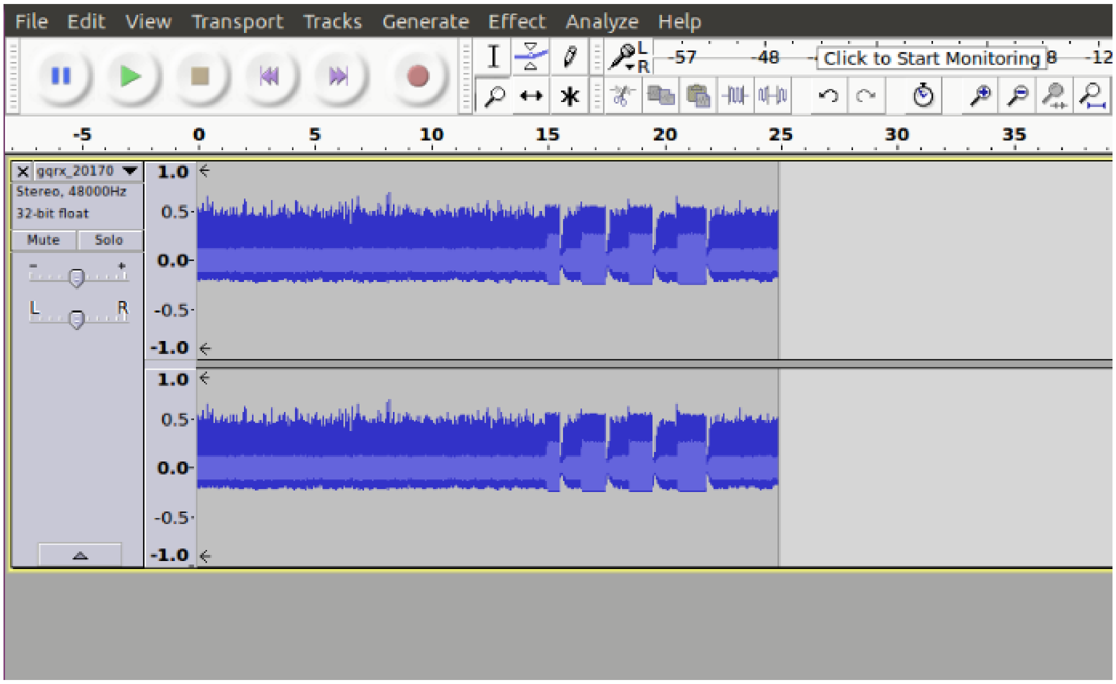
After confirming reception of the transmitter signal, repeat the process. However, this time, click the "Rec" button to record the demodulated waveform to a "WAV" file. The filename and path of the recording are displayed in the status bar of the GUI and in the terminal window you launched GQRX from for reference.



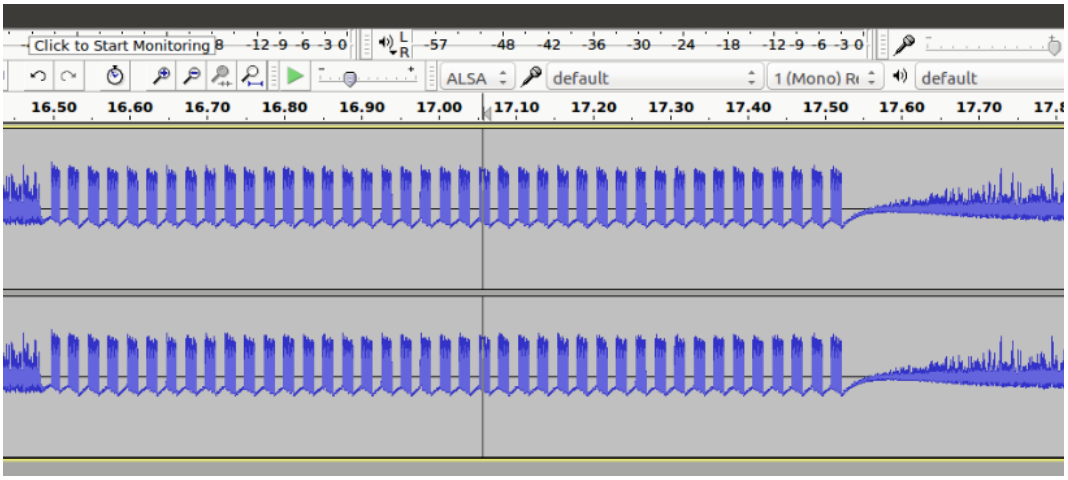
Recording audio to /root/gqrx_20170802_194004_315000000.wav

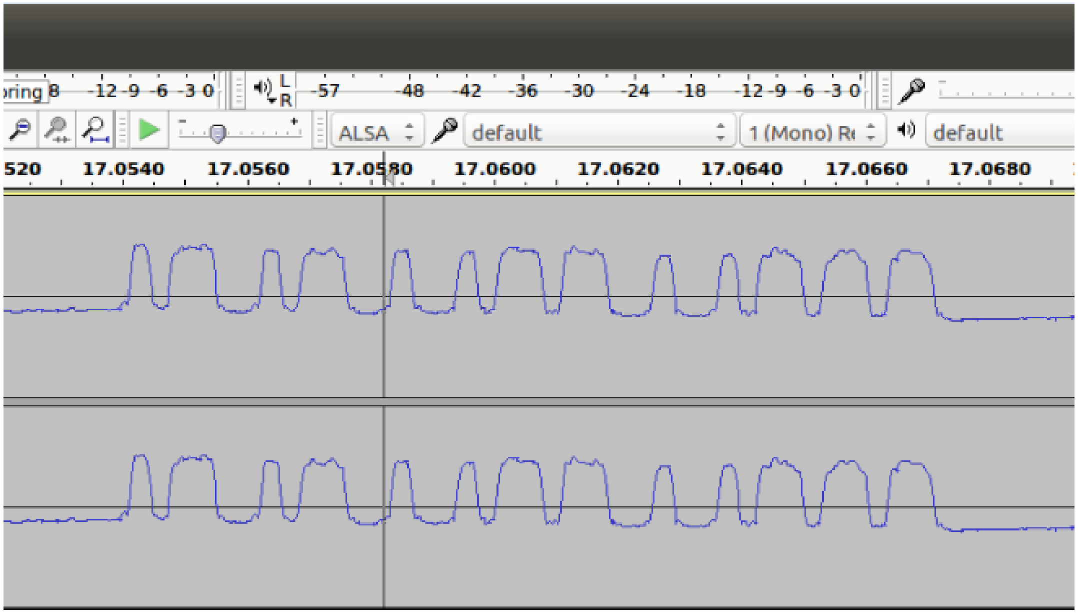
After pushing the button on the transmitter several times, click the "Rec" button to stop the recording. GQRX can be closed at this point. It will not be used through the rest of the lab. If you wish, you can leave GQRX running so you can observe transmissions from the YardStick One along with those of the original transmitter.

Next, use the launcher to open Audacity on the laptop and open the resulting "WAV" file using the File > Open menu option. Choose the default import method when prompted and click OK. The waveform will open looking similar to the display seen below.

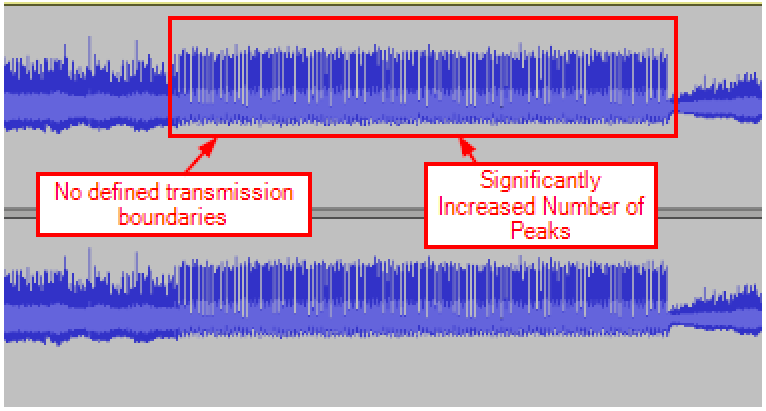


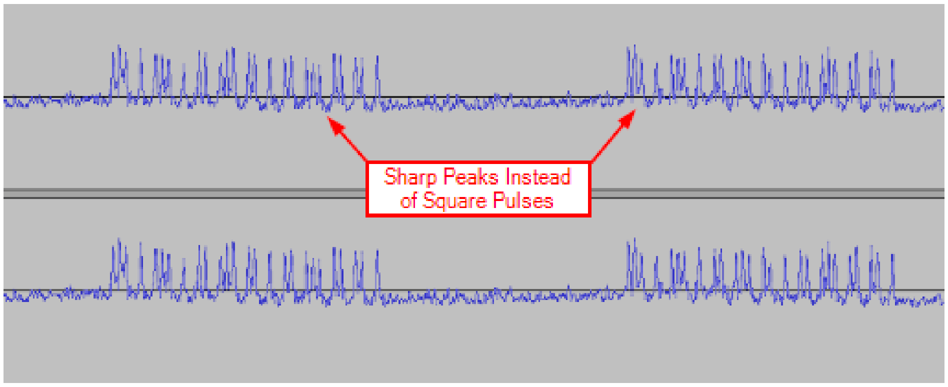
Click in the signal editor area inside one of the pulse regions and press Ctrl + 1 until the signal looks like the one seen below. The first image shows the pulse train zoomed in and the second shows a single iteration of the key being transmitted to the receiver.





If GQRX needs to be fine-tuned, the output in Audacity will resemble the following graphics.



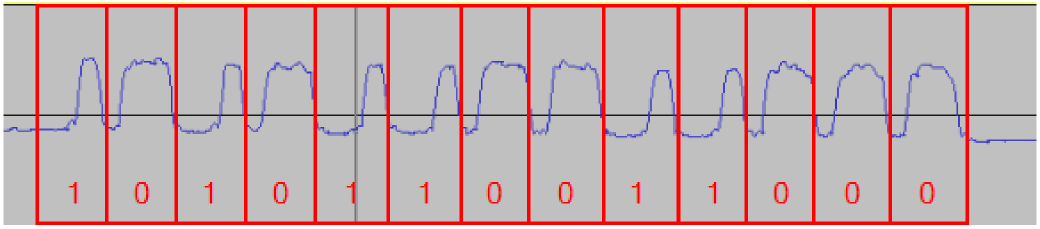


Should your recorded waveform resemble these, or lack defined pulses in the Audacity output, re-tune your GQRX to more closely match the transmitted frequency.

The single iteration screen capture shows the Pulse Width Modulated (PWM) digital signal. This signal must be decoded into its binary equivalent so that we can recreate the transmitted signal with the

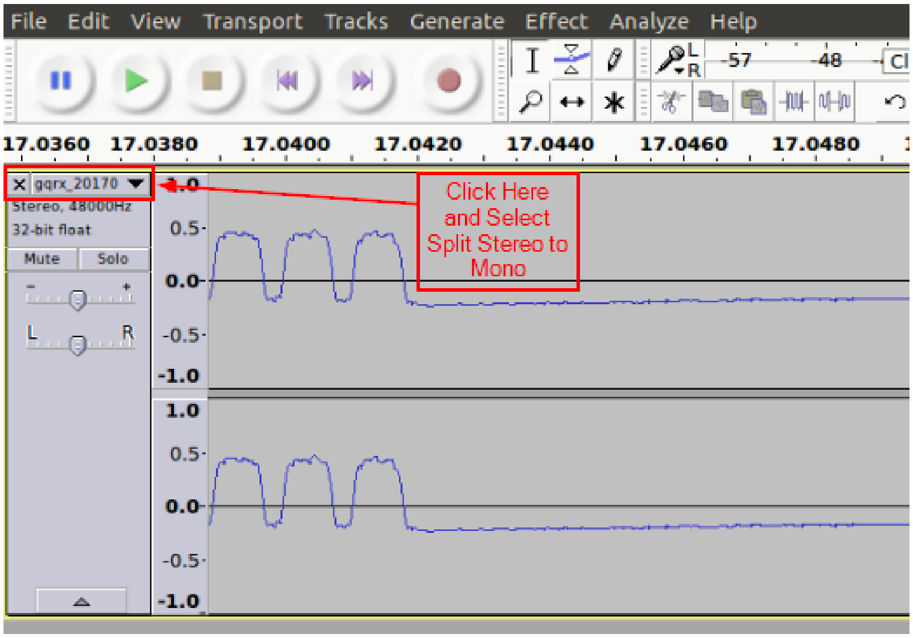
YardStick One. This is most easily accomplished by trying to identify patterns both forward and backward within the signal.

Reading the signal backward you should be able to see that there are two symbols being transmitted using the varying pulse widths. One signal consists of a short gap and long pulse while the other consists of a long gap and a short pulse. This is depicted with the binary equivalent annotated below.



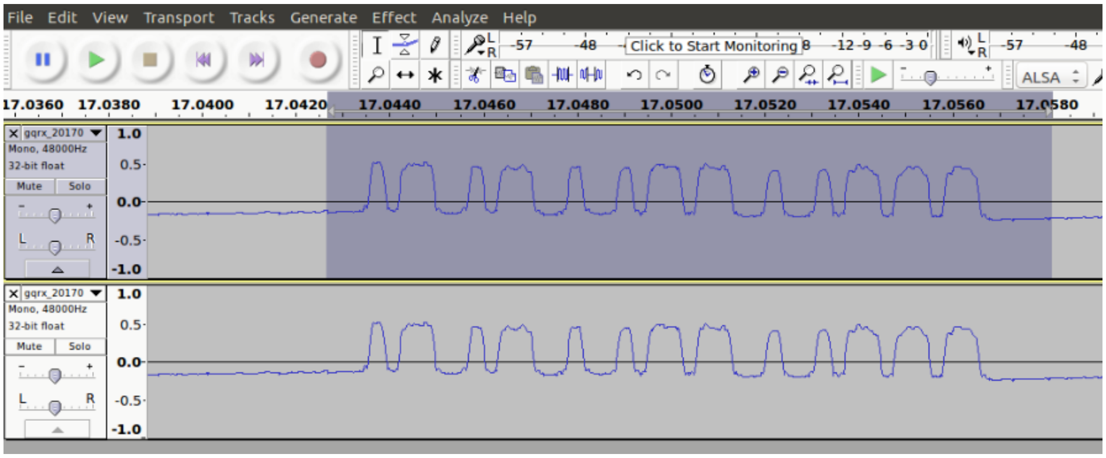
The pulses in this signal appear to be on the order of 25-33% duty cycle for a binary one and 66-75% duty cycle for a binary zero. This means that the signal is being held high for that portion of the repeating period of the waveform. This will become important when we attempt to recreate the waveform.

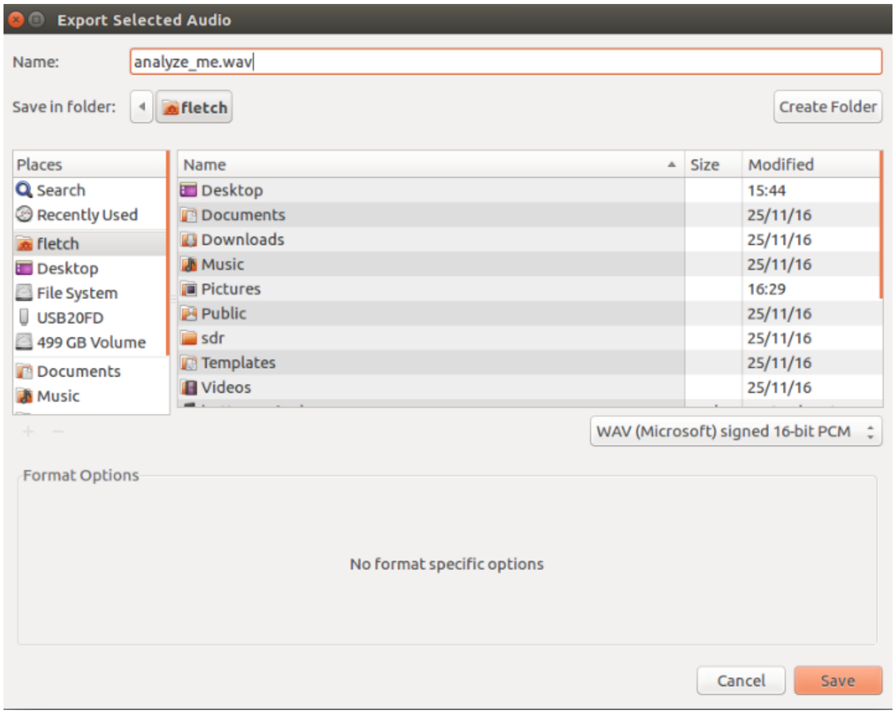
This manual analysis can be confirmed using ooktools, which is also installed on the laptop. To do so, select the track name at the top of the window to the left of the signal display and select the “Split Stereo to Mono” option.



Click Here and Select Split Stereo to Mono

After splitting the track into mono, highlight a single run of the PWM code as seen below and select File > Export Selected Audio and save the file as “analyze_me.wav”. Click OK when prompted for metadata to describe the file.





Export Selected Audio

Name: analyze_me.wav

Save in folder: fletch

Create Folder

Places
Search
Recently Used
fletch
Desktop
File System
USB20FD
499 GB Volume
Documents
Music

Name	Size	Modified
Desktop		15:44
Documents		25/11/16
Downloads		25/11/16
Music		25/11/16
Pictures		16:29
Public		25/11/16
sdr		25/11/16
Templates		25/11/16
Videos		25/11/16

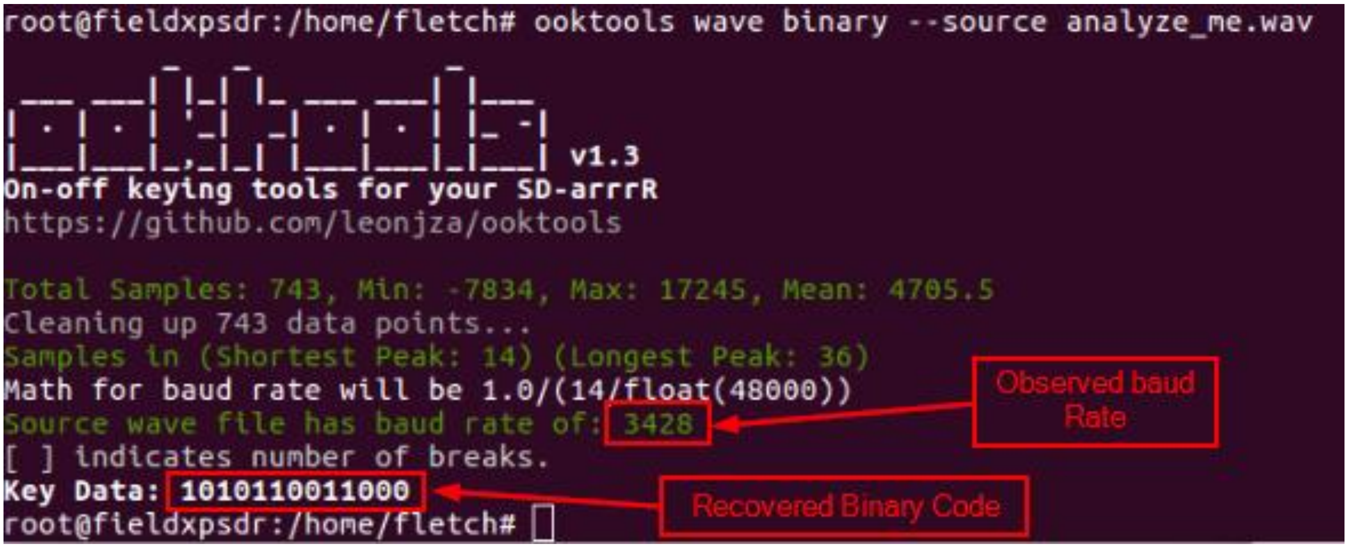
WAV (Microsoft) signed 16-bit PCM

Format Options

No format specific options

Cancel Save

In one of your terminal windows, run the command “ooktools wave binary --source analyze_me.wav”. The proper path to the WAV file must be provided in order for this to work. Output from this tool should resemble the graphic below and confirm the manually derived binary code.



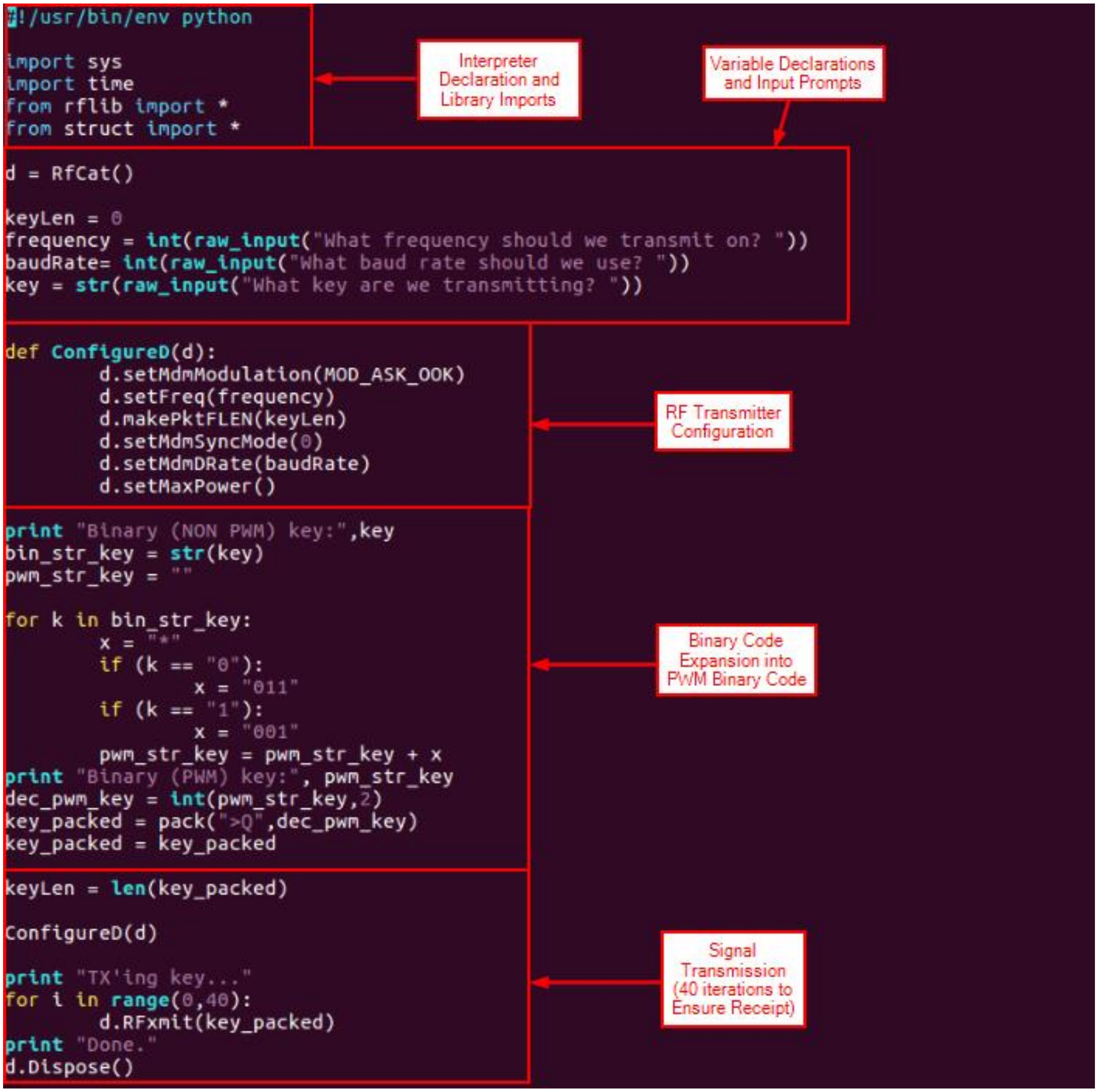
```
root@fieldxpsdr:/home/fletch# ooktools wave binary --source analyze_me.wav
          _   _           _
 ___ ___| |_| |_ ___ ___| |___
| . | . | '_|  _| . | . | |_ -|
|___|___|_,_|_| |___|___|_|___| v1.3
On-off keying tools for your SD-arrrR
https://github.com/leonjza/ooktools

Total Samples: 743, Min: -7834, Max: 17245, Mean: 4705.5
Cleaning up 743 data points...
Samples in (Shortest Peak: 14) (Longest Peak: 36)
Math for baud rate will be 1.0/(14/float(48000))
Source wave file has baud rate of: 3428
[ ] indicates number of breaks.
Key Data: 1010110011000
root@fieldxpsdr:/home/fletch#
```

With the code confirmed, we can now attempt to replay the captured signal. The rfcat Python library will be used to perform this operation in a script. Without any additional encoding, we could use rfcat to set the modulation, frequency, baud rate, and length then transmit interactively. Because the binary data is Pulse Width Modulated, we have to expand the binary code that we extracted from the signal to recreate the PWM signal.

Returning to the duty cycle mentioned above, we stated that a binary one can be represented by a 25-33% duty cycle pulse while a binary zero can be represented by a 66-75% duty cycle pulse. This means that we can expand the ones and zeroes into three or four digit numbers, concatenate those, and transmit. As a result:

- Binary 1 = 001 or 0001
- Binary 0 = 011 or 0111

Fortunately, the python script that we use in the lab already does this and can be found at http://andrewmohawk.com/2012/09/06/hacking-fixed-key-remotes/. The only difference in our script is the interpretation of the PWM signal. The source code can be seen below.


Interpreter Declaration and Library Imports
Variable Declarations and Input Prompts
RF Transmitter Configuration
Binary Code Expansion into PWM Binary Code
Signal Transmission (40 iterations to Ensure Receipt)

```
#!/usr/bin/env python

import sys
import time
from rflib import *
from struct import *

d = RfCat()

keyLen = 0
frequency = int(raw_input("What frequency should we transmit on? "))
baudRate= int(raw_input("What baud rate should we use? "))
key = str(raw_input("What key are we transmitting? "))

def ConfigureD(d):
        d.setMdmModulation(MOD_ASK_OOK)
        d.setFreq(frequency)
        d.makePktFLEN(keyLen)
        d.setMdmSyncMode(0)
        d.setMdmDRate(baudRate)
        d.setMaxPower()

print "Binary (NON PWM) key:",key
bin_str_key = str(key)
pwm_str_key = ""

for k in bin_str_key:
        x = "*"
        if (k == "0"):
                x = "011"
        if (k == "1"):
                x = "001"
        pwm_str_key = pwm_str_key + x
print "Binary (PWM) key:", pwm_str_key
dec_pwm_key = int(pwm_str_key,2)
key_packed = pack(">Q",dec_pwm_key)
key_packed = key_packed

keyLen = len(key_packed)

ConfigureD(d)

print "TX'ing key..."
for i in range(0,40):
        d.RFxmit(key_packed)
print "Done."
d.Dispose()
```

Execute this script by executing “python ringmemaybe.py”. The script will prompt for the transmission frequency, baud rate, and key discovered while executing the previous portions of the lab.

During the first transmission, you should have GQRX running to ensure that the YardStick One transmits at the appropriate frequency. If not, adjust the transmit frequency according to the observation in GQRX. The transmitted signal should appear at the center frequency where the doorbell signal was recorded.

Once the appropriate characteristics have been achieved, the doorbell should activate without pressing the transmitter button.

If you desire, make a copy of the transmission script and see if you can make the doorbell ring with a 25/75% duty cycle PWM signal and a 10/80% duty cycle PWM signal. You will have to modify the values assigned to the variable x and the may have to adjust the baud rate.

Cleanup

Please delete your saved WAV file and close all open application and terminal windows.

Thank you!!