Lab 3: Software Defined Radio
7/24/2021
Alex Young

Write-Up

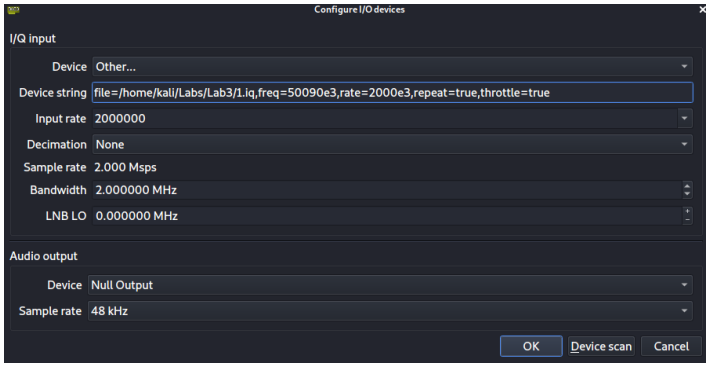
Introduction

In this lab I was given 3 .iq files and the instructions to determine the contents of the files using the tools in the kali vm previously installed. As of the time I am writing this, I am not entirely sure if I truly found all the data that was included in these files because I never seemed to find any complete message. Given the vagueness of the lab description and having already spent 8+ hours working on the lab coming to multiple dead ends, I decided to compile my efforts and conclusions rather than continuing to search.

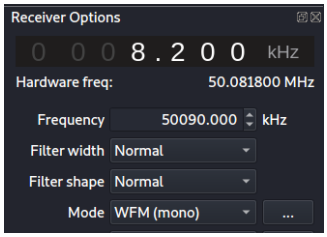
File 1

This file was tricky to get started with and decipher because I had to first understand how gqrx worked and get audio setup on the kali vm (and also unzip all the .iq files in the laggy vm). I started out by installing gqrx (using either sudo apt-get install gqrx-sdr or sudo apt install kali-tools-sdr). Gqrx is a sdr receiver that can read the .iq files and display the sounds generated by them at different frequencies and bandwidth. I went on to configure the audio to play using the volume control application.

As file 1 has the filename → 1_2Msps_50.090MHz_2MHz.iq, the string parameters for setting up gqrx are set. The screenshot below shows how each value in the filename corresponds to a parameter in the “configure I/O devices”.



Device string:
file=/home/kali/Labs/Lab3/1.iq,freq=50090e3,rate=2000e3,repeat=true,throttle=true
In addition to setting these parameters, the “Receiver Options → Frequency” needs to match the 50.090 MHz frequency in the device string.



For this file it didn't seem that the mode particularly mattered, and from here I ran the file. The first thing I noticed was that the file sounded like a quick series of beeps. These sorts of beeps led me to think that this was morse code. However, the beeps were happening so fast that I knew that I could not decipher them manually. A quick google search found this helpful website: https://morsecode.world/international/decoder/audio-decoder-adaptive.html
I ran the audio with this website running in the background repeatedly and got results that looked like this:

EAPTURED IETE46520KHZ SAMPRATESOMMTETTTETTTEEEIWT WAS FAPTURED ETT 146520KHZ SAMPRATEE2M EEETTTANLE J WAS FAPTURED UT 146520KHZ EAMPRATEIJM EEEATTETTEETEEESTTTEMASEFAPTURED UT E46520KHZIEAMPRATEIOM EETMPLE WAS CAPTURED ITT 146520KHZ SAMPRATEEEEOM ETTETTTETTEEDE 2 WAS RAPTURED UT 146520KHZ SAMPRATESOM EEETMPLE EJ WAS+APTURED VT146520KHZ IAMPRATEEJM IETTTTETTEETEEEEETMWAS CAPTURED ET 146520KHZ EEAMPRATE 2M EEETMPLE J WAS FAPTURED ETT 46520KHZ EAMPRATE EJM E

While there appears to be some gibberish, there are also quite a few apparent patterns that stand out. Below I show what I deciphered from this morse code decryption:

- IETTTTETTEETEEEEET WAS CAPTURED ET 146520KHZ SAMPRATE 2M
- **___ was captured at 146520 KHz, sample rate 2 M**

At this point I assumed that ___ could mean either files 1, 2, or 3, so I went to this file location to try out the audio for all three files.

File 2
File 2 was the most straightforward for me out of any of the files.
Previously (when I was originally testing out the .iq files) I had been using mode → RAW I/Q. While using this mode and testing in all 3 files I didn't find anything at all in the file location I found from the morse code. While I was not completely sure that the previous message was exclusive to file 2, it seemed quite likely, so I focused my attention on file 2.

I found that from googling 146520 KHz that it is the national calling frequency for 2m, which matched with the morse code. From here I found that this frequency uses FM and decided to change up the mode I was listening to the audio in. Using the device string below and mode → wfm (mono), I was able to hear a clear audio output.

file=/home/kali/Labs/Lab3/2.iq,freq=146520e3,rate=2000e3,repeat=true,throttle=true

The audio that played was - **"The third file was captured at 7.171 MegaHertz (MHz) at 2.5 Megasamples per second (Msps)"**. Before looking at file 3 I made sure that files 1 and 2 didn't have any hidden pieces at this frequency - which they did not.

File 3

This file was by far the most difficult for me to decipher, and I am still not completely sure that I found what I was supposed to find for the lab. To follow the steps of what I ended up doing, first I set up the frequency, sample rate and device string as described by the audio from file 2.

- file=/home/kali/Labs/Lab3/3.iq,freq=7171e3,rate=2500e3,repeat=true,throttle=true

At this point I ran audio for the file while trying all sorts of different modes, and found not much of anything. There was no morse code or message. Here I was quite stuck, yet I felt like I was at the right spot because there were clearly some abnormal sounds, so I googled for different ways that radio could decipher messages.

At first I thought that the message might be hidden in the waterfall, but a zoom-in found this to be false. Eventually I began thinking that this might be a SSTV (slow scanning television) that created an image based off of the audio. I thought this could be the case because the sounds that file 3 were making sounded similar to SSTV audio files. Later on, I googled 7.171 MHz and found that this band is actually used for SSTV.

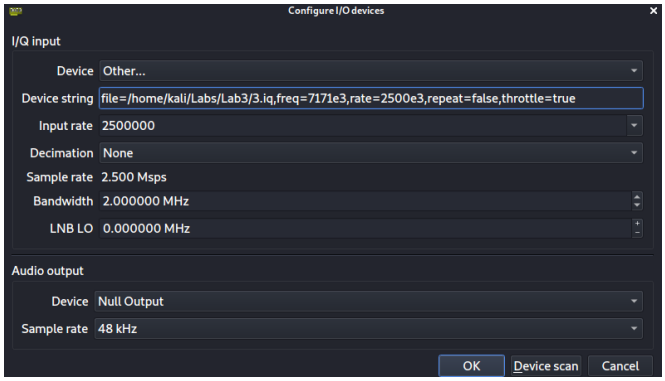
Frequencies [edit]

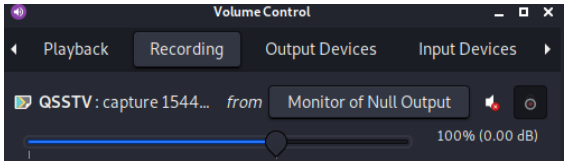
Using a receiver capable of demodulating single-sideband modulation, SSTV transmissions can be heard on the following frequencies:

Band	Frequency	Sideband
80 meters	3.845 MHz (3.73 in Europe)	LSB
43 meters	6.925 MHz (Pirate Radio)	USB
40 meters	7.171 MHz (7.165 in Europe)	LSB
40 meters	7.180 MHz (New Suggested Frequency to include General Classes)	LSB
20 meters	14.230MHz Frequency 1 Analog .	USB
20 meters	14.233MHz Frequency 2 Analog to alleviate crowding on 14.230.	USB
15 meters	21.340 MHz	USB
10 meters	28.680 MHz	USB
11 meters	27.700 MHz (Pirate Radio)	USB

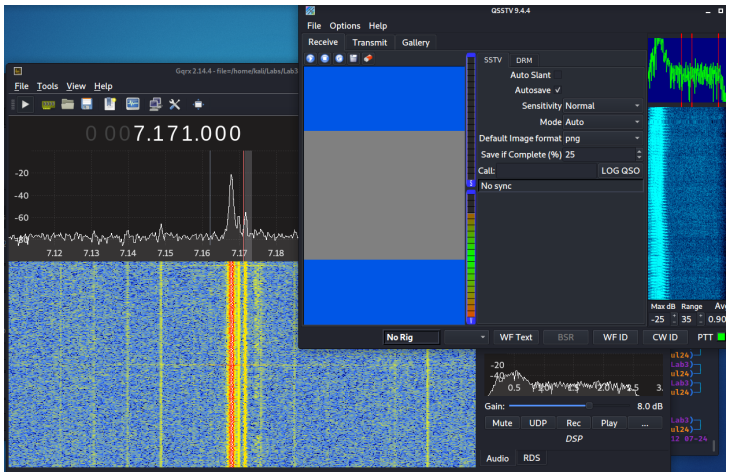
https://en.wikipedia.org/wiki/Slow-scan_television

With this in mind, the next step was to figure out how to generate the image from the audio file. After quite a bit of research I followed the instructions here: https://charlesreid1.com/wiki/Qsstv to install qsstv. Using the volume control system, I connected the output of gqrx to the input recording of qsstv and started receiving from qsstv (using the monitor of null output).

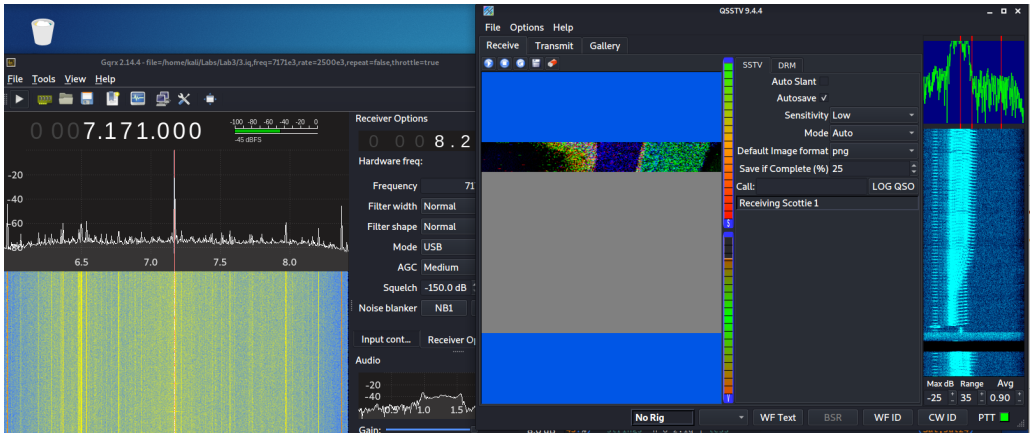


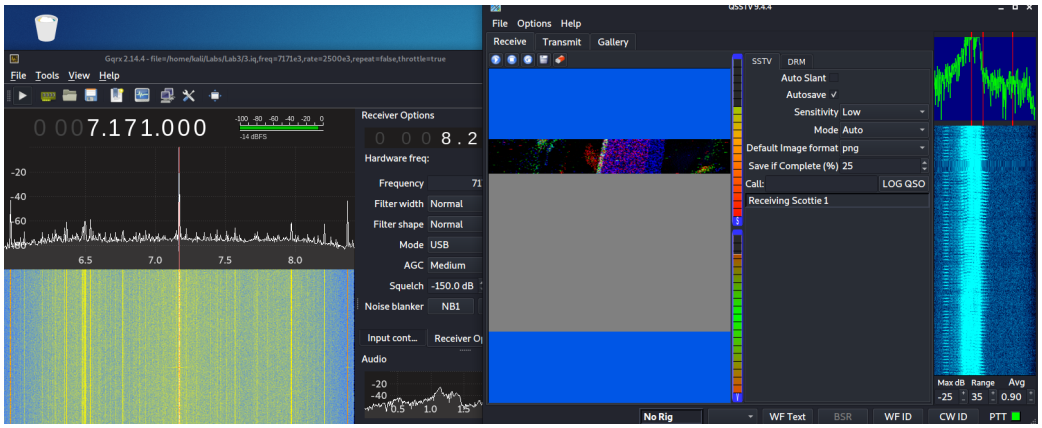


Despite this seeming like it would work, I was stuck receiving no image at all for quite a long time, with the results looking like the screenshot below.



When I found the wikipedia article that I screenshotted above, I tried further experimentation with the mode → lsb, because that is what the table shows. This led me to be further stuck until eventually I found that running the file using mode → usb in fact starts generating an image in qsstv. **Here I was able to confirm that file 3 generates a sstv image, as shown below.**



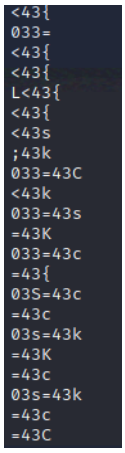


However, as shown, the image is never able to fully complete itself. No matter what bandwidth is set, the receiver would always break some point before 20% of the image was completed. At this point I was not sure if this break was man made or the fault of the extremely laggy vm. In either case I felt like there was not any path forward for me to complete the image (considering that file 3 is 2.4 GiB large I was not sure how to “fix” it on the vm).

Conclusion

From here I stopped with the conclusion that the contents of file 3 is an sstv image. I am not sure what this image shows and if it leads to more steps to find hidden information or not.

The thing that makes me uncertain about this is that the assignment says that the optional software installation on kali is unnecessary for finishing the labs (and these optional installations include gqrx). What this means is that using tools like gdb-gef and strings we were supposed to have enough information to reach where I had gotten using gqrx. To test this out, I tried using strings on each .iq file, but nothing was decipherable at all. An example is shown below:



Overall, I used gqrx to find messages in files 1 and 2 that pointed me towards frequencies to use to find information in files 2 and 3 respectively. In file 3 I found a hidden sstv image and arrived at a dead end. I hope I was on the right path here and I also hope that the complete solutions can be shared for this lab.