# Lab 6: A Few Different Digital Signals

## Overview

For this lab, we will look at a few different digital signals that you can capture with your SDR. Digital signals are interesting because there are so many different ways that information is encoded, and it is fun to try to sort out what a particular signal does. However, getting beyond decoding the digital data is hard, because then we need to know what the communication protocol is. In this lab we'll just decode the digital data, and leave it at that.

We'll look at three different signals. The first is the signal your key fob emits to communicate with your car. The second is a look at some of the packet data that is sent in the 915 MHz ISM band. Finally we'll look at a digital radio signal.

## Aims of the Lab

The goal of the lab is to find, identify, and decode different digital signals.

## Your Car's Remote

The first signal we'll look at is that generated by your wireless car key fob. This is call a Remote Keyless System (RKS). In the U.S. these operate at 315 MHz, +/- 2.5 MHz. My Prius key turned out to be at 312.590 MHz.

To capture signals from your key fob, first you need to figure out what frequency it transmits on. Connect your SDR and use GQRX or SDR# to monitor the spectrum. When you push a button on the fob, you should see a brief jump in the spectrum. You may need to shift the frequency band up or down by a couple of MHz to find the signal, mine was almost 2.5 MHz low.

One word of caution. Don't get too carried away pushing the button! The RKS system uses a rolling pseudo-randomly generated code. Both the key fob and the car keep in synch, so that the car recognizes the next code. However, if the key fob gets too far ahead in the sequence (100s of button pushes) the car won't recognize it. That makes the key (and the car) considerably less useful!

Here is the command to capture data for my car:

```
> ./rtl_sdr -f 312590000 -s 2048000 -n 20480000 -g 29.7 rks312590.dat
```

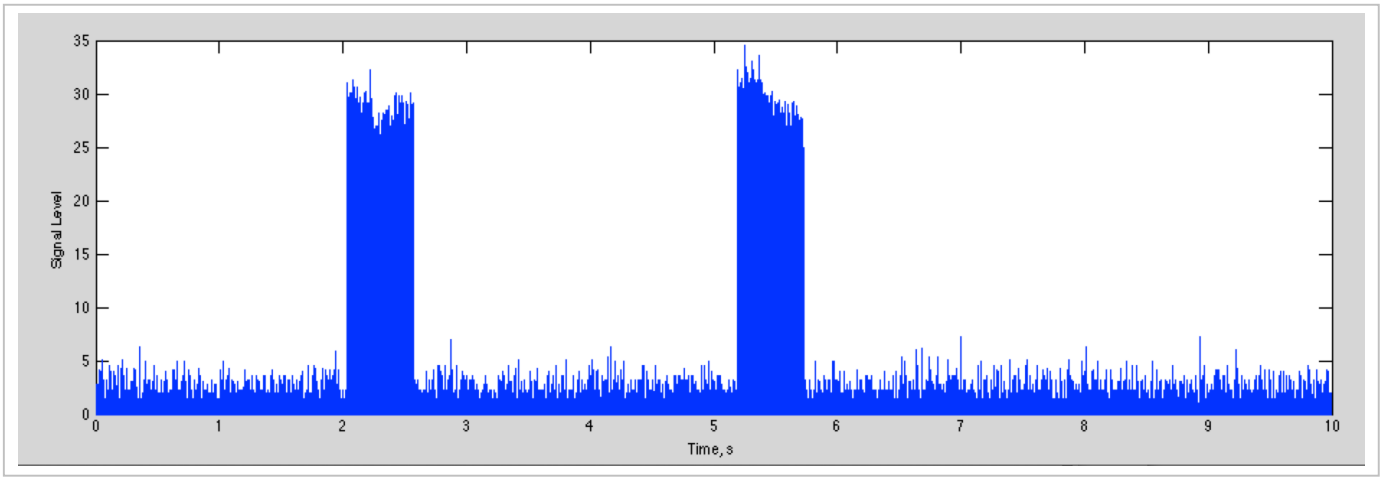
I specified the gain to be 29.7 dB (one of the values you see with you run rtl_test), because if it auto ranged (the default) it sets the gain for the background noise, and then completely saturates with the key fob signal appears. A sample data set for my car is here:

rke312590.dat

First we need to find where the signals are. We can find them by plotting the magnitude of the signal

```
> dk = loadFile('rks312590.dat');
> fs = 2048000;                          # Sampling Frequency
> t = [1:length(dk])/2048000;            # Time
> plot(t(1:1000:end),abs(dk(1:1000:end)));
```

where we skipping by 1000 samples to keep the plot manageable. The result is shown below

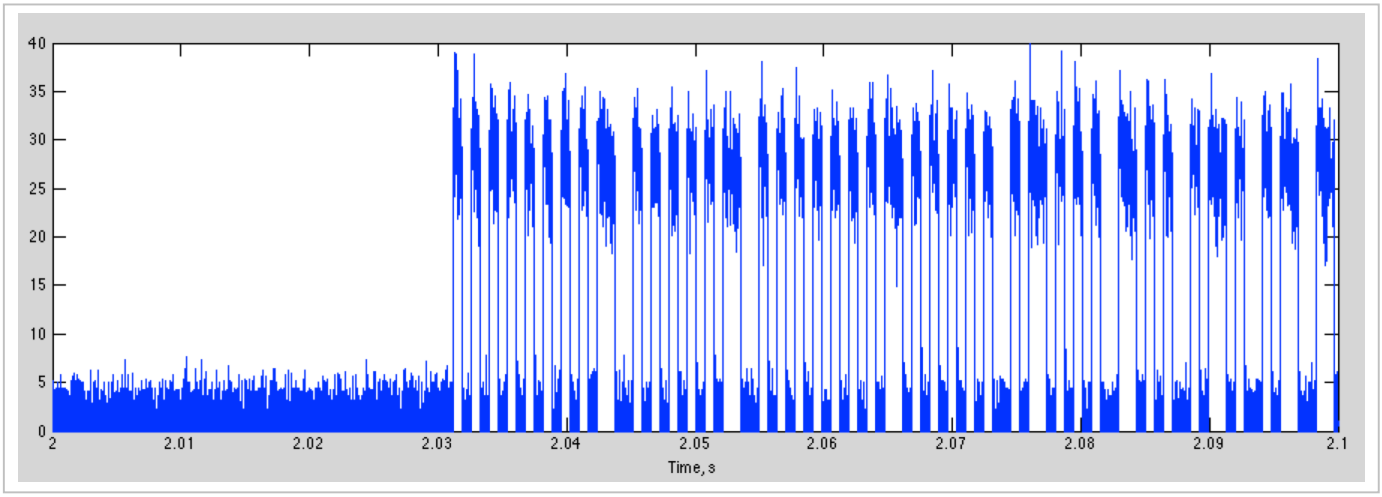



The total width of the plot is 10 seconds, so you can see there is one key press shortly after 2 seconds, and another shortly after 5 seconds.

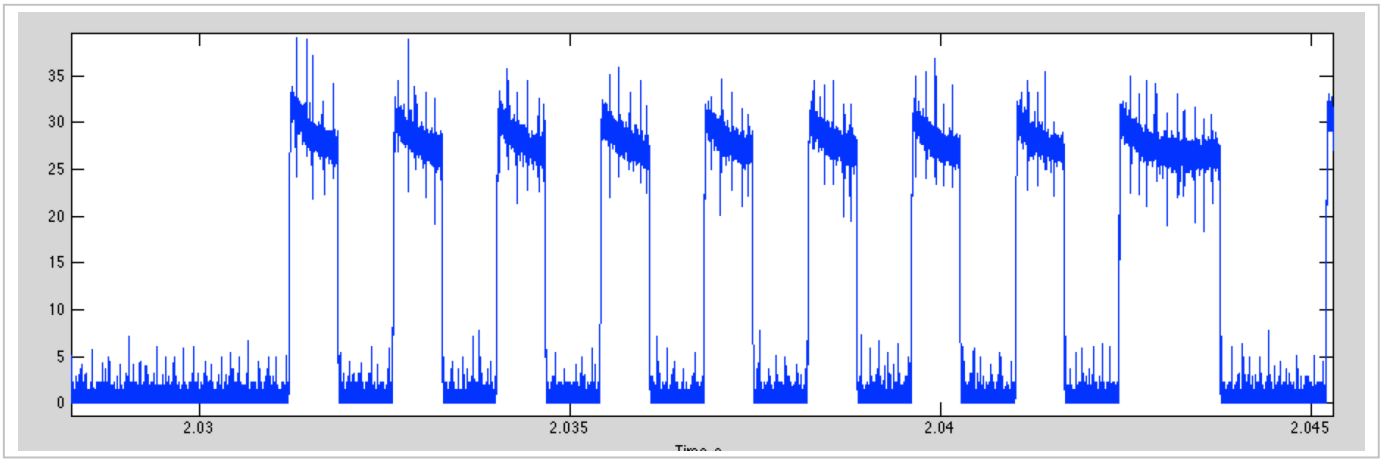
If we plot 100 ms starting at 2 seconds, we can see the digital signal we are looking for:

```
> plot(t(2*fs:2.1*fs),abs(dk(2*fs:2.1*fs)));
```

which looks like this



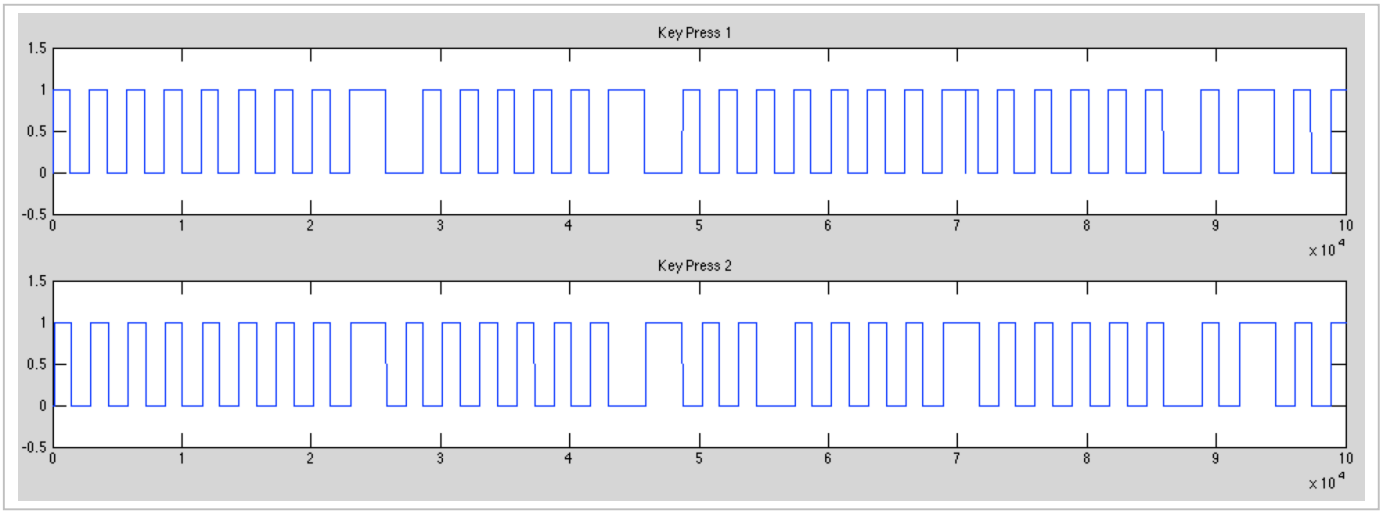

Zooming in to the first couple of bits, we get



The bits are easy to identify. A decision threshold of 15 will give almost perfect detection (your data will have a different level)

```
> dkd = abs(dk) > 15;
```

If we do this, and then plot first part of the digital data for the two key presses, we get this




Although the two start the same, they rapidly diverge. This is fortunate, because if the signal was the same every time, you'd have enough information to steal my car now!

Convert each key press logical waveforms to sequence of bits. You will need one value (1 or 0) for each bit. Determine the width of each bit, and the number of total bits. Include the bit pattern for this data, or your data, in your report.

This is a simple on-off-keying signal (OOK).

## Digital Data Packets

The next signals we will look at are digital packets that are transmitted in the 915 MHz ISM band. This goes from 902 MHz to 928 MHz. ISM stands for “Industrial, Scientific, and Medical”, and is largely unregulated. Almost all of the radio wave commutation you care about is in an ISM band! This includes WiFi, bluetooth, cordless phones and speakers, and microwave ovens. This is also the 33 cm an amateur band. There is lots of different traffic from the usual FM repeaters, to amateur TV, and digital radio.

If we use GQRX or SDR# to monitor the spectrum we see the spectrum jump around every once in a while, with different frequencies and spectra. We won't see a continuous communication at a single frequency, as we did with the previous analog signals.

Capture 10 seconds of data to take a look at, centered at 910 MHz. Here is a file I captured at home:

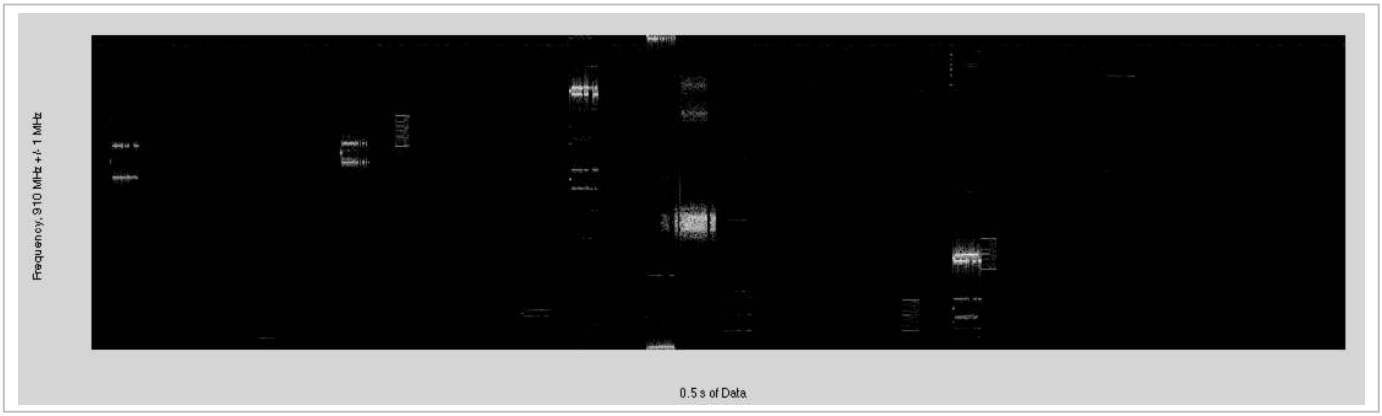
ism910.dat

Your environment probably has more traffic, and will be more interesting than this!

If we compute a spectrogram with

```
> di = loadFile('ism910.dat');
> msg(di,1,512,2048);
```

we will get something like this



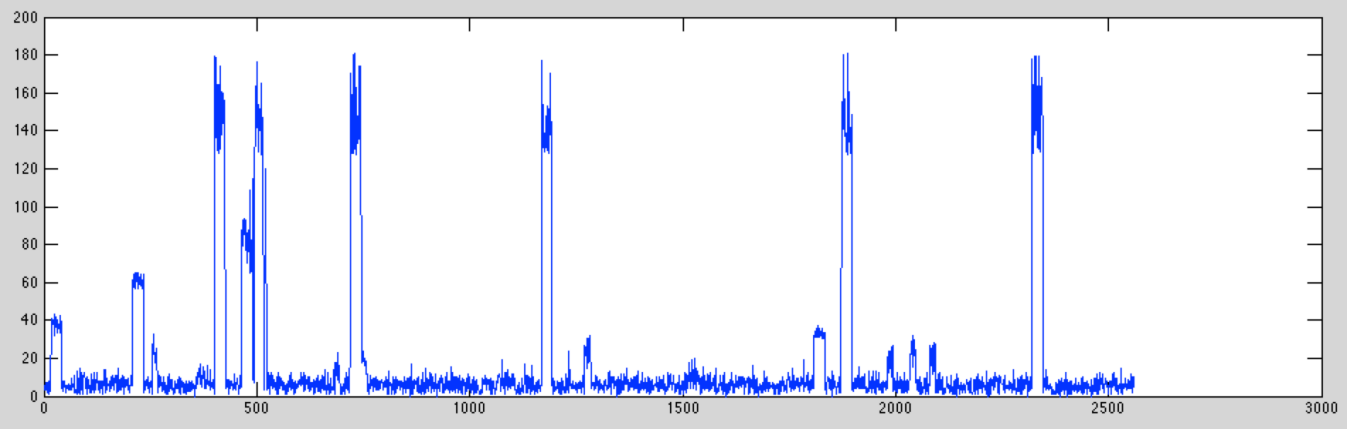

This is about a half second of data. What we see are lots of different digital packets, each with different frequencies and encoding patterns. Every once in a while they interfere. We'll just look at a couple of packets to see how they are

encoded. At this level it is hard to tell what is happening.

We can get a better idea of the timing of the packets by plotting the magnitude of the signal. If we plot every 1000th sample of the first 1/8th of the data

```
> plot(abs(di(1:1000:end/8)));
```

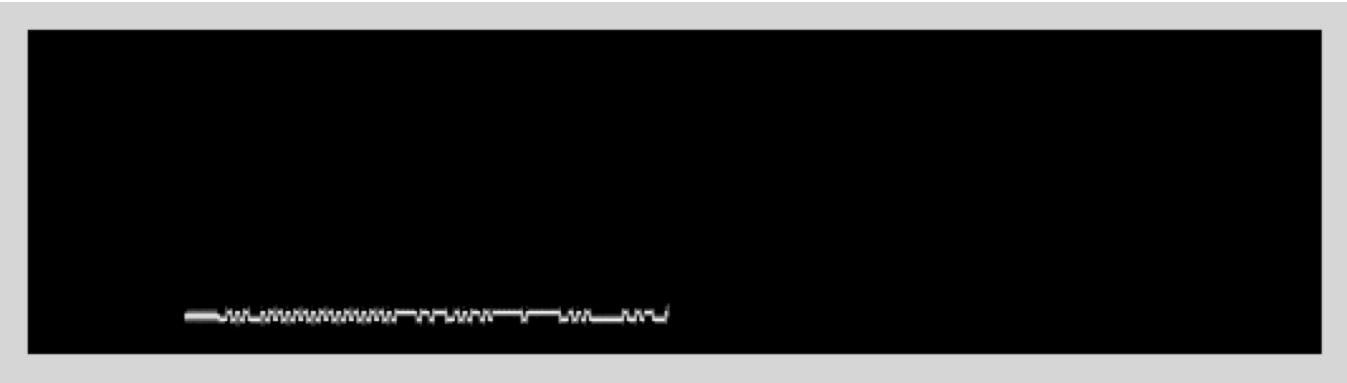
we get something like this




There are lots of different packets to look at. If we zoom in, we can see that there is an isolated packet starting at 1160. Since we skipped 1000 samples per plot point, this corresponds to sample 1160000 in the raw data. In the spectrogram above we can't really sort out what the signal is doing because we used a frame size of 512 samples to compute the frequency spectrum for each line, and it is changing more quickly than that. If we decrease the frame size to 128 samples, and limit the dynamic range of the spectrogram to 20 dB, we get a very nice depiction of the digital data for this packet,

```
> msg(d,1160000,128,512,20)
```

which looks like this:



The digital data is easy to identify here. This uses frequency shift keying (FSK), with each bit being either +f0 or -f0, after an initial calibration interval at a frequency of 0.

Choose another packet in the data, and see if you can figure out how to decode it's contents. Some of the packets are much easier than others to decode. You need to set the frame size small enough to resolve individual bits, but large enough to spectrally resolve the frequencies used to encode the bits. Include a plot of the spectrogram for the packet of interest in your report.

## Digital Radio

When you were exploring the amateur and police radio bands, you undoubtedly encountered channels that made an oscillating beeping sound. Let take a look at how these channels are encoded, and why they sound this way.

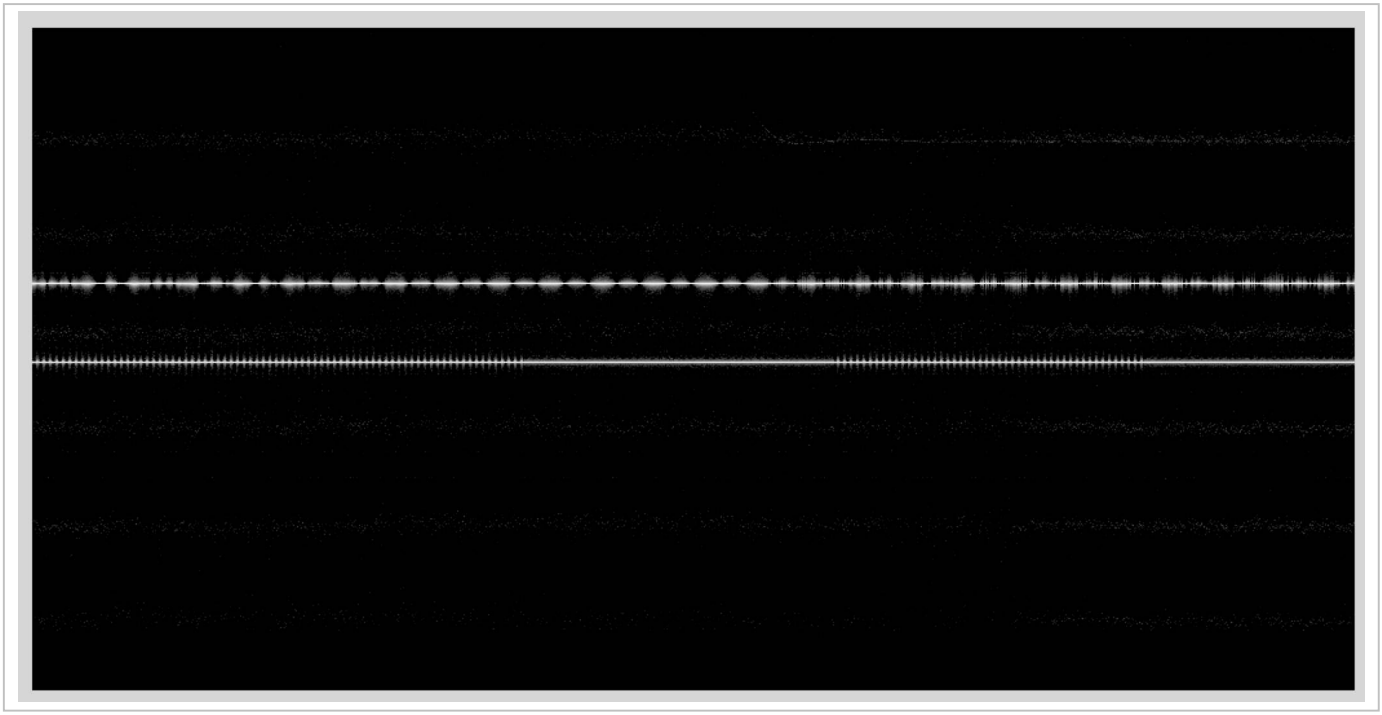
There are lots of channels to choose from. The one I picked is in the VHF amateur band at about 144.985 MHz. Many of the digital police and fire bands around 460 MHz are encoded in the same way, and would also work.

The only trick is that there isn't always a signal at 144.985. Use GQRX to monitor the frequency, and then when you have a strong signal, kill GQRX to free up the SDR, and then acquire 10 s of data using rtl_sdr as usual. Data I acquired at 145 MHz is available here:

vhf145.dat

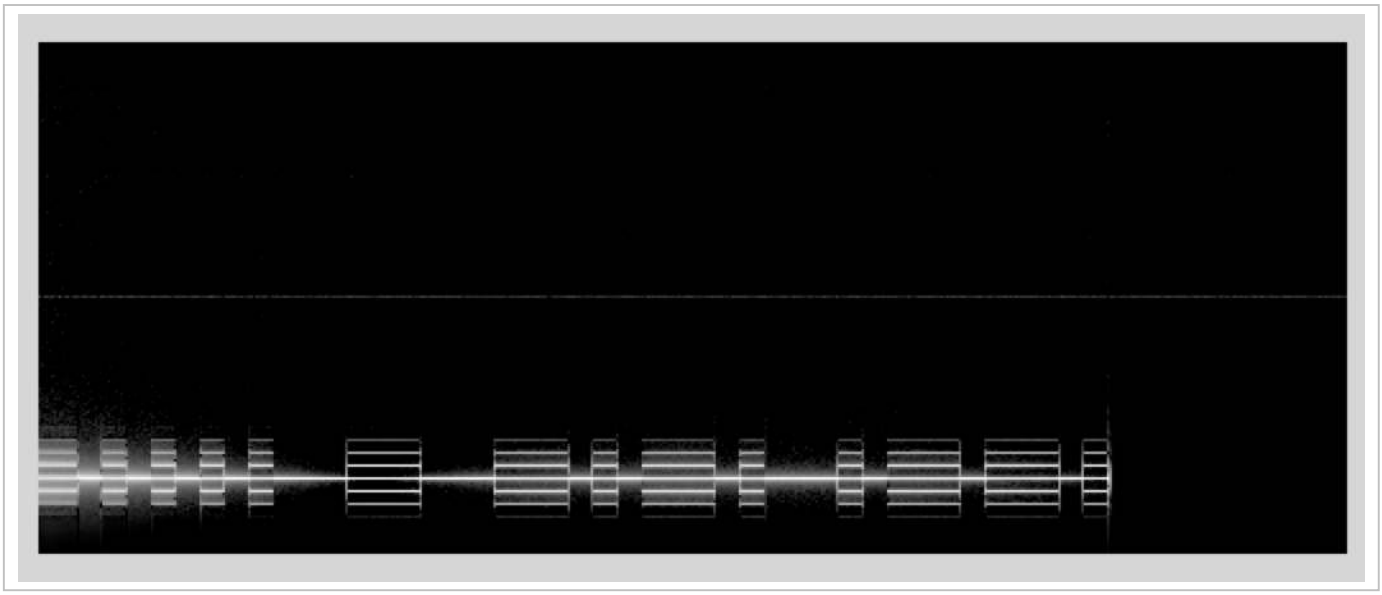
This is the usual sampling rate of 2.048 MHz, centered at a frequency of 145 MHz.

If we display the spectrogram with a block size of 512, and a dynamic range of 30 dB, we get this,

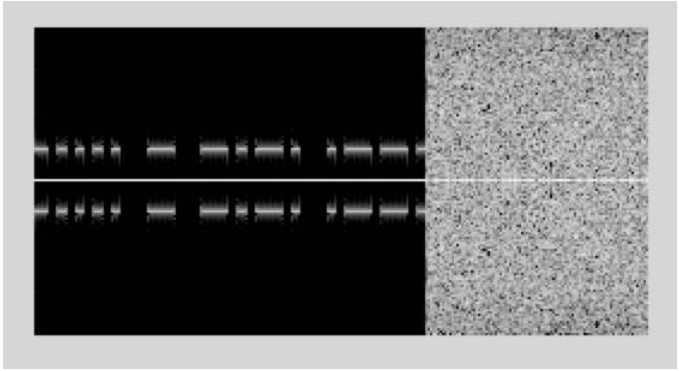


Checking the frequencies with ffreq we find the signal we want at about -10 kHz. The other signal we see is N6FNI identifying itself with Morse code. We'll ignore that.

If we decimate the signal down by a factor of 256 (two decimations of 8 each), and compute the spectrogram with a smaller 200 sample window, we get this intriguing plot,



This shows a signal is being keyed on and off, but it has an odd structure. This is what you hear. To get a better idea of what going on, recall that most of the signals in this band are FM, so maybe we should FM decode first, and then see what we get. Using your narrowband FM receiver code, and a frequency shift by -10 kHz, we get this spectrogram, using a block size of 128,



We get a nice clean frequency shift digital waveform (at least unit the transmission ends). This is digital FM encoded in an analog FM signal! You can plot the waveform itself, and what you will see is a constant plus a cosine superimposed every once in a while.

Determine the length of the bits, and decode the waveform that was being transmitted into a sequence of 1's and 0's.

## Lab Report

There are a couple of things to include in your lab report

- The bit pattern for two key presses.
- A spectrogram (or waveform plots) that show how another packet was encoded in the 915 MHz band.
- The decoded bit pattern for the FM digital radio signal.

  You can also look at another digital signal if you like from the 460 MHz band. There are many variations. This totally optional, though.

Page generated 2017-05-18 11:27:08 PDT, by jemdoc.