Notes on SDR

J.D. McCormack

2015-07-01

1 Introduction

The following document will contain information related to my efforts in software defined radio. From time to time, the document may include information on other topics as well. It will initially serve as a day to day journal of my activites. After a certain discovery phase, this document will be updated to create a manual or tutorial for future students to learn from. It is not recommended that the document be used for self study until a newer version is created. Many of the steps that are outlined will be dead ends or poor practices while the disocvery process takes place.

2 7-1-2015

Currently working on using the RTL SDR.

What I have learned so far:

- This SDR is different than a FUNCUBE dongle
- It will work from around 500 Hz to 1700 Hz
- It can only RECIEVE

The fact that this can only recieve is by far the largest setback. This will ultimately not be what we can use for cognitive radio.

However, the tutorials still seem more widely available than those found for the BladeRF which is bi-directional. Therefore I will continue for a bit. It may serve a purpose as part of the larger sensor network at some point. We could still use it to relay information over a large distance.

There are "Cubelites" being sent out that are basically funcube dongle based satellites. These sound pretty cool.

I want to start taking notes in Latex so they will be more readable than a plain .txt file and still useable with version control.

I need to install texlive-full and texmaker.

Currently I am set back by needing to update ubuntu. I can't install anything else while the updates are installing.

Success So far I have made decent progress. The RTL is working with GNU SDR. I had trouble at first when the kernel was loading a separate driver that bogs it down.

I used this command to fix that problem:

```
# sudo rmmod dvb_usb_rtl28xxu rtl2823
```

There are more permanent ways of fixing the problem but I wanted to start with this as it is non-permanent and will default back to normal on a restart.

The new file in the RTL-SDR will allow you to hear FM stations. But they are extremely faint. I'm going to continue to play with the file settings to see if I can get a clearer transmission.

3 7-2-2015

I began the day with a quick interlude into learning LaTex. After about an hour and a half of studying I was able to produce the document you are currently reading. I'm not making use of many libraries but I believe it already seems more organized and official than my traditional notes. Also, it is much more portable than a normal word document and less prone to formating failures seen in google docs.

Yesterday I got the SDR to recieve FM stations. However, they were staticy and sounded slowed down. I'm not sure if this is an error in my demodulation values or just that the computer that I'm using is too slow. I tried using a different one but that ended up being slower than I remembered. I will try to get linux installed on my laptop later tonight when I'm done working, but I'm not confident it will work as I had trouble in the past.

Today I will try to replicate the results I had yesterday, but through the use of the python libraries instead of relying on the GNURadio GUI interface. A small note about the gui interface. I did learn that its possible to make variables, attach them to GUI widgets, and then alter then live. It appears that having more than 3 of these can severely impact performace. To use the GUI widgets, simple drag one onto your workspace (I used the Wx widgets) and then give it a unique name, and appropriate value range (if its a slider). Then in the blocks for different components, you can use these variable names instead of hardcoded values. This is useful for gains and frequencies.

/paragraphGNURadio in Python

I will be following the tutorial found here. I will begin with the dial tone generator.

Listing 1: dial_tone.py

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

# Gr is the basic gnu radio module
# gr is always required
from gnuradio import gr
# audio and analog are libraries of blocks
from gnuradio import audio, analog

# Here we define a top block that inherits from
# the gr class top block.
class my_top_block(gr.top_block):
  def __init__(self):
    gr.top_block.__init__(self)

    # standard quality sample rate and amplitude value
    sample_rate = 32000
    ampl = 0.1

    # We create two sine waves with the same amplitude and sameple rate
    # from above.

    src0 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 350, ampl)
    src1 = analog.sig_source_f(sample_rate, analog.GR_SIN_WAVE, 440, ampl)

    # dst is the "destination" or audio sink block
    dst = audio.sink(sample_rate, "")

    # We then connect the two sine waves to the destination block
```

```
        self.connect(src0, (dst, 0))
        self.connect(src1, (dst, 1))

if __name__ == '__main__':
    try:
        my_top_block().run()
    except [[KeyboardInterrupt]]:
        pass
```