**Testing Continued**

Since the attenuator is completely working, no further tests were needed for that board. Returning back to the testing of the SDR Receiver, I picked off of where I was last week. Last week, I still needed to rewire the bandpass filters (as this was thought to be the cause of frying the arduino nano). I was able to successfully cut the traces and find some wire I could use to re-solder the pins to the right places. After I finished doing this, I used a multimeter to test the connections to make sure that they were solid. Once I had done everything I could to make sure the bandpass filters were wired correctly, and I had explored any other possibility of the Arduino frying, I mustered up some courage and tried plugging in my Arduino nano once again. This time, it did not burn anything! So, the incorrect wiring in the bandpass filters was, in fact, the problem.

However, an issue remained. I had programmed the Arduino to select the 2nd bandpass filter, but it was still selecting the first filter. After checking the datasheet for the multiplexer, checking my code, and checking the board with no luck – I decided that I could afford to tackle this problem at another time. I figured this was something I could work on after I knew I had the rest of the board working since I knew I had at least the first bandpass filter working.

Proceeding to test the rest of the board, I put in a 760kHz as the input signal, and set the si5351 chip output to 3MHz, so that the local oscillator is at 750KHz. To my disappointment, the tayloe mixer did not output the four signals as I expected it to. Instead of having signals that were at 0 degrees, 90 degrees, 180 degrees, and 270 degrees, the signals were somewhat arbitrary. This was somewhat expected because I saw some of this result in my LTspice simulation. I determined that this was due the “loading” of the circuit – not seeing exactly a 50 ohm impedance at the output. I figured that this was partially the same reason for what I was seeing out of the Tayloe mixer. But, looking at the final output I and Q signals, I discovered that they were exactly 180 degrees out of phase instead of 90 degrees out of phase. After talking to Dr. Frohne and Josh, I found that we had mislabeled the 180 and 90 signals on the inputs to the summing and low pass filter op-amps. Having the signals switched would explain the different loading effects. It also would explain why the final signals were 180 instead of 90 degrees out of phase. I cut the traces on our circuit board and attached new wires to the outputs of the tayloe mixer. Before attaching the wires to the correct op-amps, I decided to check the signals coming out of the mixer without any loading from the op-amps. What I saw on the oscilloscope was an absolute blessing – I saw sine waves that were 0, 90, 180, and 270 degrees out of phase — perfection. Soldering on the wires to the correct op-amp inputs, I again looked at the output from the tayloe mixer and saw the same loading effects that I noticed in the LTspice simulation. Finally, looking at the final I and Q output signals, I saw sine waves that were exactly 90 degrees out of phase and with low noise! This was a cause for celebration!

The next step in testing the radio was getting quisk to work to control the frequency of the local oscillator and selecting the bandpass filter on the mux instead of the Arduino test program. I downloaded the Arduino files from Dr. Frohne’s github, flashed the Arduino with the openradio\_quisk.ino file and related .h and .cpp files, changed the file in quisk to the quisk\_conf.py file and made edits within the .py files so that the USB port name on my mac matched those in the code. I also needed to change the sound input on the quisk program to the USB :C Media Pulse\_Left channel. At this point, I was able to see the signal at two peaks at 760MHz and 740MHz – both of which were 10MHz away from the 750 MHz local oscillator signal set by quisk. Unfortunately, it appeared that my mac was only using 1 channel, instead of 2. As a result, the 769MHz and the 740MHz signal were equal in strength. I essentially had no image rejection. But, this simply a software issue, not an issue with the radio hardware and circuit design. Dr. Frohne helped me contact a forum that is continuing to look into the issue. Hopefully by next week sometime Mr. Eric Thornton will be able to tell me what the best way to manually set the number of channels being used are. Once I am able to use both channels on the sound card, I should be able to test for image rejection.

Some things that still need to be done are

1. Get my sound card to use both channels so that I can test for image rejection
2. Verify / or alter and update the code controlling the use of bandpass filters in the Arduino and python code to work with my multiplexer functions
3. Perform tests for signal discernibility. (this one will definitely be done by the time of the presentation next week and should be included in the documentation)

For more information on testing procedures, I have started adding them to my github repository: <https://github.com/greenjacketgirl/SDR_Receiver>

Images corresponding to this weeks’ tests:









