# Team RadioNauts:

Eric Andler, Josue Herrera, Ben Sawtelle, Vi Thai

# Final Report
# EEC 134 - RF Systems

## Design

The main idea behind our design was to make the parts as modular as possible. By keeping the different systems separated we thought it would make debugging and problem solving easier to accomplish. We separated our radar into the following subsystems: function generator, radio frequency circuit (RF), baseband, and digital signal processing (DSP). Each of these subsystems was self contained, either on a PCB or breadboard.

Below is the overall block diagram for the entire radar system. Included in this diagram are the calculated power ratings for each component, along with relevant linearity ratings. This should show that all components are operating well within their optimal linear regions.

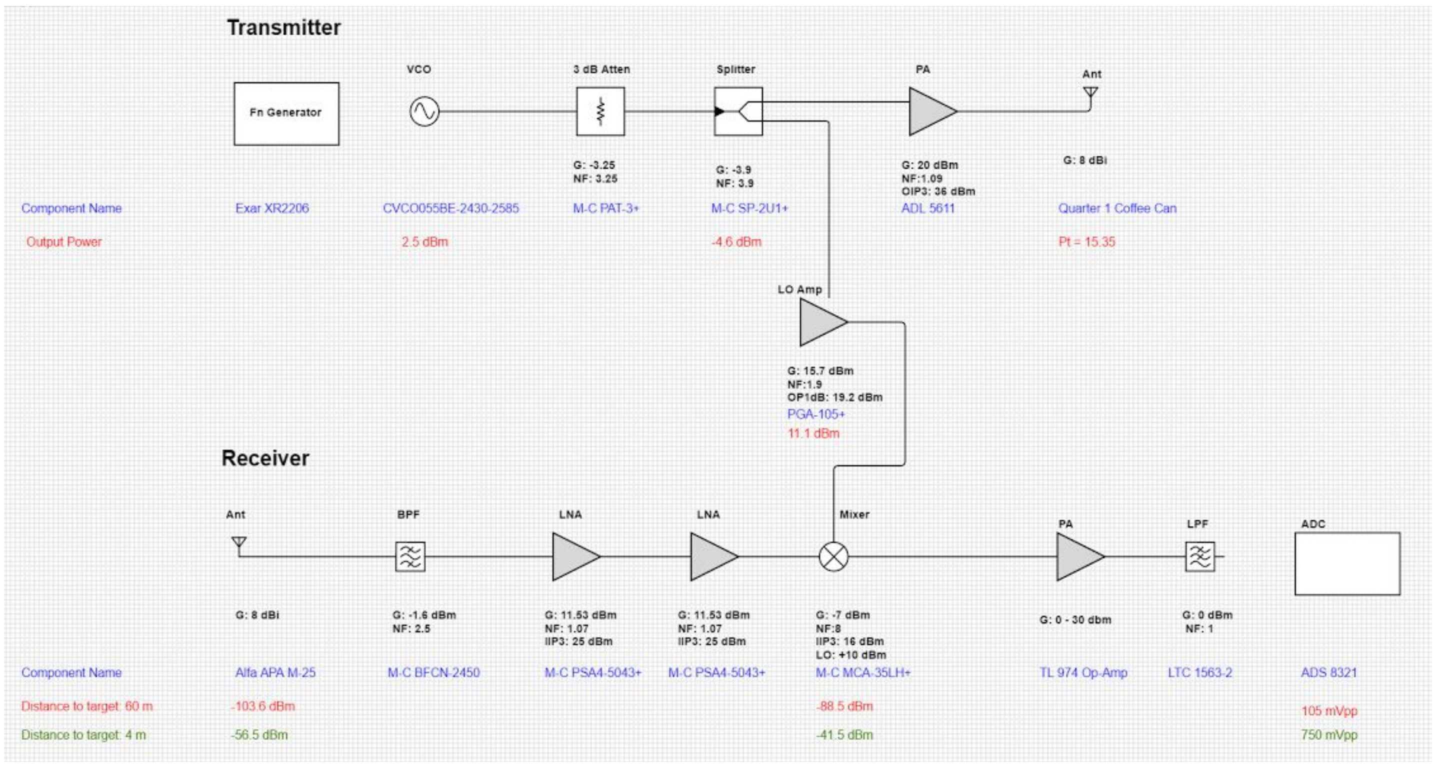

Figure 1. Block diagram for entire radar system

### *RF system*

The main objective when designing the RF PCB is was to keep the RF signal traces as straight as possible and as isolated as possible.  Because signal leakage was expected to be a problem, the signal generated by the VCO was designed to be as far away from the input signal as possible, which can be observed on the PCB when looking at the footprints for the SMA connectors.

Because the traces are made to resemble a microstrip transmission line for frequencies between between 2.4 and 2.5 GHz, but are not ideal, one must keep the traces short in order to avoid losses, but they were also made long enough to allow for easy soldering and desoldering of the components.

When designing the RF PCB, one must not cross the waveguide traces for this would add noise to the signal. This restriction became an issue when providing the 5V Vcc bias to the LO Amp that runs from the splitter of the VCO signal to the mixer, for there was no possible way of providing the needed bias without crossing the waveguide traces. This issue was resolved by using a header pin that would provide needed bias through a wire over the waveguide trace and the PCB itself.

One can also observe from the design of the PCB that in order to keep the number of loose wires to a minimum and to easily transfer the IF signal from the mixer to the baseband, a 3.5mm female audio jack was added as the output of the RF Band.

A final note on the PCB is that it follows the design shown on the block diagram for the radar system and using the tool available we were able to verify the power on before and after each component; however one must note that due to the inaccuracy of the crude probe made to measure the power at each point, the power was not the same as that indicated on the block diagram, since the probe introduced large losses. Each component was verified to be working by measuring the power before and after each component, which allowed one to identify problems such as bad soldered joints. An example of this behavior was when debugging the power out of the transmitter. Using the probe to measure the power after the power amplifier, one was able to read a power of approximately 1dBm. This was about 15dBm lower than expected, however, measuring the power before the amplifier one read a power of about -14dBm, which meant that the amplifier was amplifying about 15 dBm, which was very close to the 16dBm that the amplifier was intended to amplify.

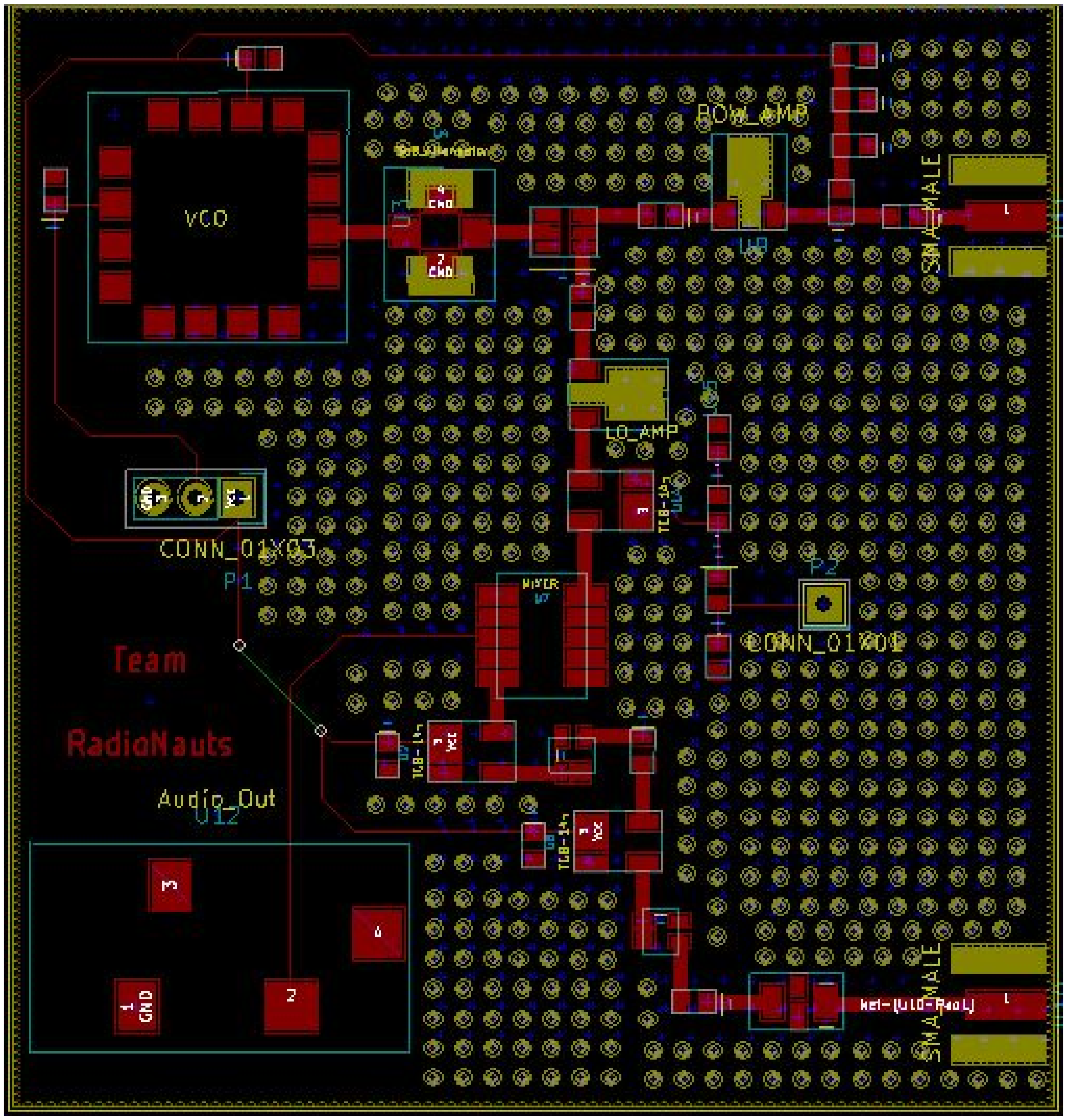

Figure 2. RF pcb

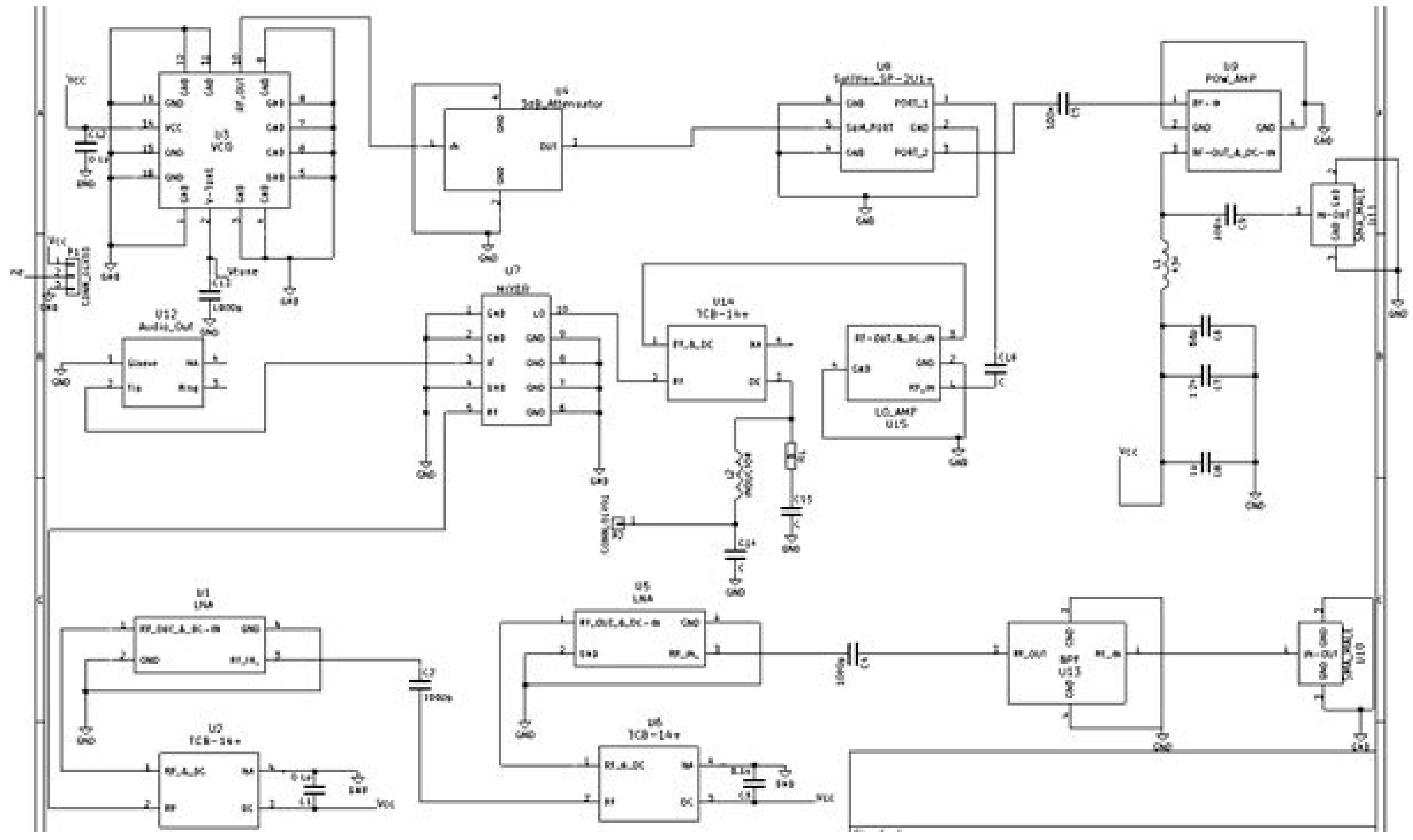
Figure 2a. Schematic for RF pcb

***Function Generator/Power Junction***

In an attempt to save power and weight on our system, we decided to use an IC as a function generator. The chip we chose was the Exar XR 2206 (datasheet). This IC is capable of outputting triangle or sine wave as well as a square wave of varying amplitudes and frequencies by simply adjusting the biasing points. Our design required the Vtune going into the VCO to be a triangle wave running between 0 and 2.5 volts at a frequency of 25 Hz. In order to achieve this waveform, we had to use negative voltages. The schematic below is from the datasheet for this IC. We followed this very closely, with the only exception being that all of the ground symbols were replaced with a -2.5 V bias.

Throughout the entire radar system there was a need for four different voltages (+/- 5 V and +/- 2.5 V). There were four different systems that were separated out into different boards, all with their individual power needs. In order to make the wiring easier we decided to include all of the power implementation into the pcb that also housed the function generator. The +5 V was needed for all four systems. Initially we chose a low pass filter for the baseband system that needed a -5V bias, however that component was eventually replaced with a filter that only needed +5V. The power pcb had already been sent to manufacture by that point, so it was too late to erase that part of the design. The +2.5 V reference point was needed for both the baseband circuit and the DSP. In addition to the voltage biasing outputs, the function generator outputs also needed to be sent out of the board. The triangle wave output needed to go to the RF board, and the square wave to the DSP. Each of the voltages needed would be regulated using a linear voltage regulator (-5 V, -2.5 V, +5V), with the exception of the +2.5 voltage, which utilized a reference voltage IC. The idea is that all the power needed for the radar system would come into the board, and then using locking test points as outputs should be easy to run the different bias points needed for other boards. Included below are the schematic and pcb design (copper fills are not pictured for clarity sake, but ground planes were added on top and bottom).

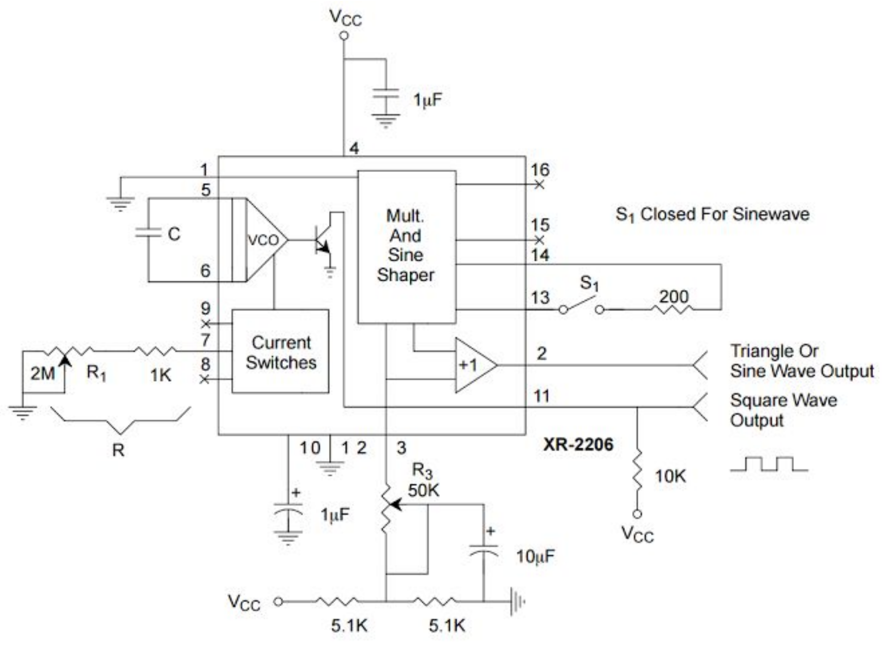

Figure 3. Schematic for function generator using XR 2206

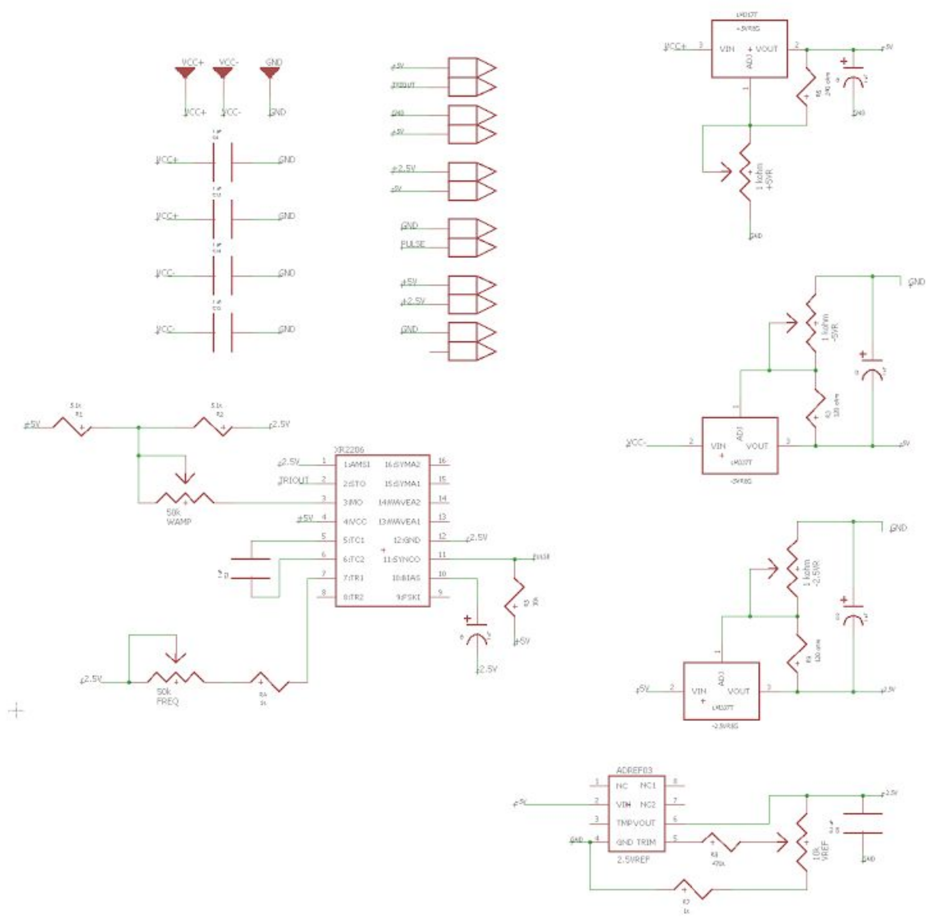

Figure 4. Schematic for function generator/power junction pcb

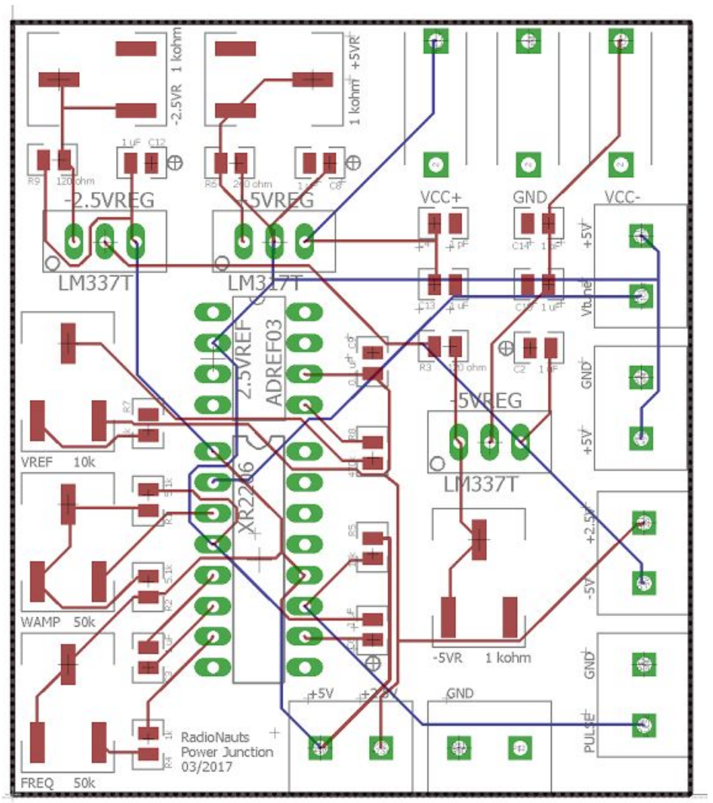

Figure 5. Pcb design for function generator/power junction

***Baseband***

For amplification, you may choose whichever topology to achieve the necessary amount of gain. We will examine the common non-inverting amplifier topology for simplicity sake.

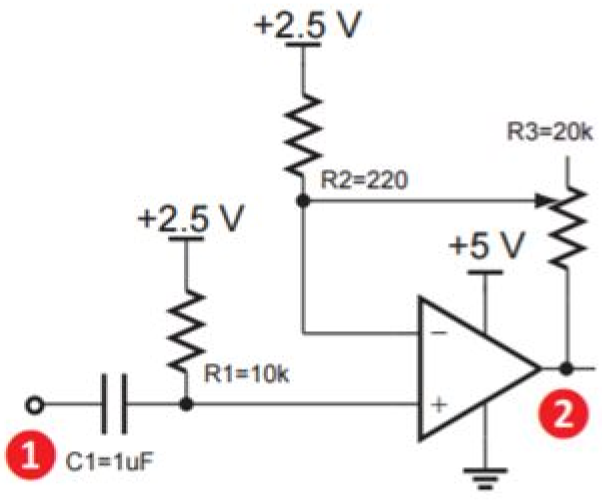

Figure 6. Non-inverting Amplifier Schematic

From figure 6, the gain of the amplifier is determined by R2 and R3. It would be a good idea that R3 be switched with a potentiometer to have an easily adjustable gain. Examining the schematic, the role of C1 is to DC couple the input signal so as to eliminate any DC components from the signal. The +2.5 Volt reference in conjunction with the +5 Volt supply rail merely sets the maximum output swing if only a +5 Volt source is used to power the amplifier. This can be modified depending on the user's intended application. From our experience, it may be a good idea to cascade another non-inverting amplifier to add versatility to the gain stage. You may need to increase the gain substantially than what was calculated from the link budget analysis in order to obtain a desirable signal for DSP. However, since amplifying the signal also amplifies the noise, it is important to find a good balance between the two, which is easily achieved with potentiometers as the gain controller.

Like the baseband amplifiers, there are numerous ways to implement a low pass filter. One of the easiest method is to use an IC from companies such as Analog Devices or MAXIM Integrated. We used MAX7408 (datasheet) since the typical operating circuit was easy to build. The MAX7408 provides a stopband rejection of approximately 53 dB with cutoff frequencies between 1 Hz to 15 KHz.

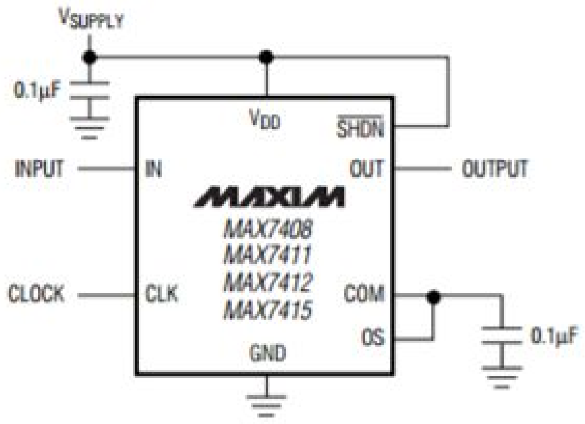

Figure 7. Typical Operating Circuit for MAX7408

Since an external clock cannot be utilized, the MAX7408 can be internally clocked by shorting the CLK pin to GND with a capacitor value to yield the desired cutoff frequency.  The relationship between the cutoff frequency and oscillation frequency of the internal clock can be expressed as:

$$f_C = \frac{f_{CLK}}{100}$$

The necessary clock frequency can directly be calculated by the cutoff frequency. With these parameters, the expression to determine the capacitor value is given as

$$f_{CLK}(KHz) = \frac{k}{C_{OSC}(pF)} \qquad where\ k = 27 * 10^3$$

In principle, the cutoff frequency of the low-pass filter should be as low as possible so that unwanted components from the amplified signal and background noise will be filtered out. The cutoff should be high enough to just cover the bandwidth of the signal.

***Digital Signal Processing***

The signal processing design was fairly simplistic and consisted of an ADC, a microcontroller, a small display, and a few tactile switches that acted as an input to switch through menus and move a cursor. This design ended using a low amount of power, only drawing about 50-60mA at 5V. Unfortunately there wasn't enough time to create a fully functioning system that could output the results in realtime, but the display output a frequency spectrum of the system every second (or half second). This data needed further processing to create a meaningful output and the design plan was to create a similar display to the Python/Matlab code that was used in its stead, but in one second intervals or quicker. While time was the main obstacle, there was also some issues arising from a lack of RAM. Memory issues, however, could have been eliminated with some proper code optimization.

This design used the ADS8321 16-bit ADC, which had an SPI interface with the timing diagram shown below. Due to the strange timing, bit-banging was used rather than a dedicated hardware SPI channel. This meant that the timing was created by manually toggling the pins in the code instead of sending the data to a queue register on the microcontroller. The results are often slower, but more control over the data flow is offered.

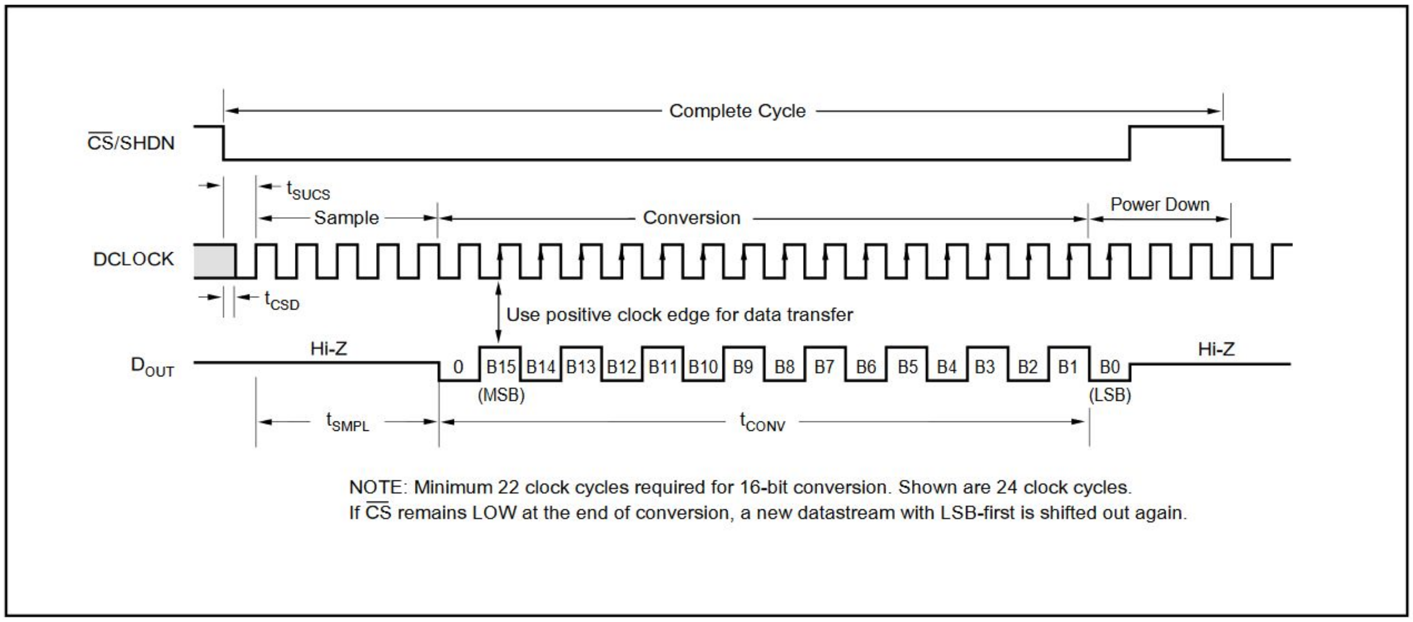
Complete Cycle
CS/SHDN
tSUCS
Sample
Conversion
Power Down
DCLOCK
tCSD
Use positive clock edge for data transfer
DOUT
Hi-Z
0
B15
B14
B13
B12
B11
B10
B9
B8
B7
B6
B5
B4
B3
B2
B1
B0
(MSB)
(LSB)
Hi-Z
tSMPL
tCONV
NOTE: Minimum 22 clock cycles required for 16-bit conversion. Shown are 24 clock cycles.
If CS remains LOW at the end of conversion, a new datastream with LSB-first is shifted out again.

Figure 8. ADS8321 Basic Timing Diagram

The ADS8321 was implemented using the suggested schematic within the datasheet (Figure 9), with the reference of the ADC, which sets the max readable voltage level, set to around 1.25V rather than 2.5V.

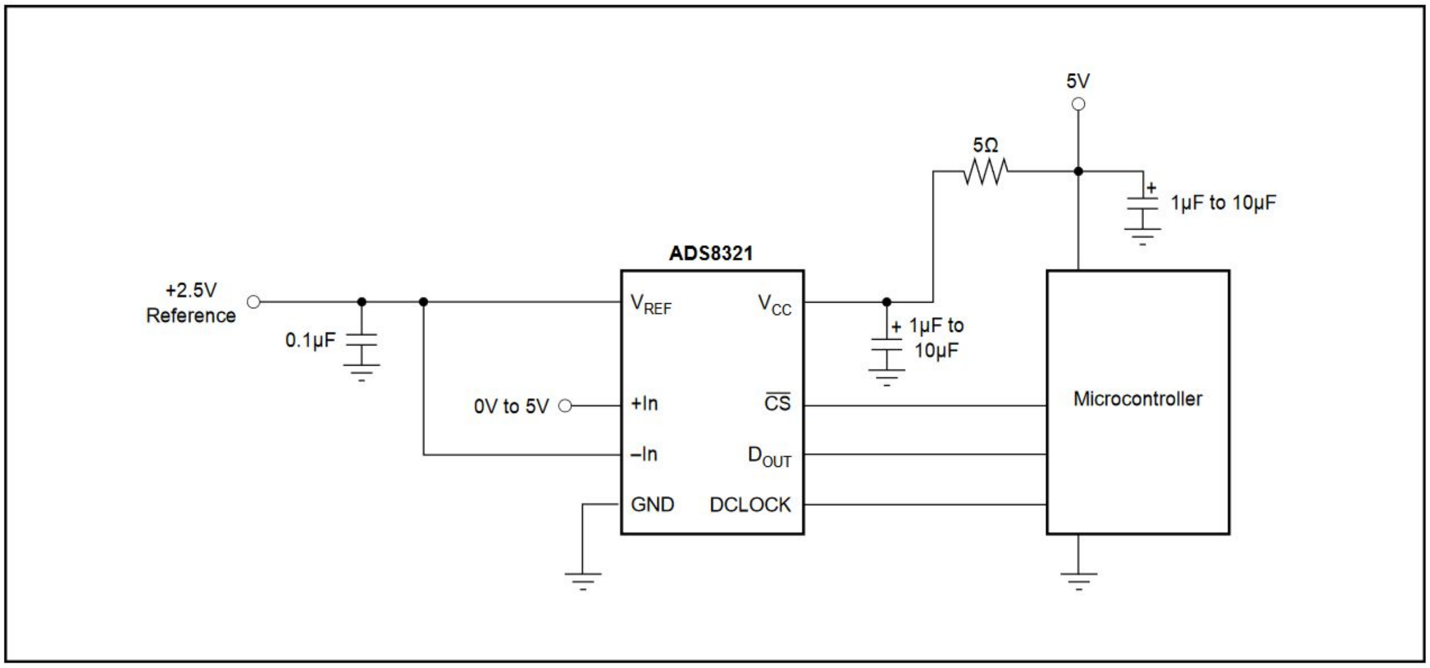

Figure 9: ADS8321 Example Schematic

The original microcontroller used was the Raspberry Pi Zero, which runs a Linux operating system by default. The Raspberry Pi has an FFT library that uses the onboard GPU to quickly process large amounts of data within a few hundred microseconds. This coupled with the high speed of the Raspberry Pi made it a very tempting choice for this application, but it was eventually eliminated due to the unnecessary difficulty that the Linux operating system imposed on the data acquisition process. Because Linux is not a “real-time” operating system, several interrupts that keep the OS flowing smoothly disrupt the read timing of the ADC. This means that a consistent sampling rate can be difficult to achieve. Perhaps the more important reason that the Raspberry Pi Zero was abandoned was the difficulty of easily coding and running the program. There is only one available micro-USB port, which means a powered USB hub is required to connect the Pi to the network and use a keyboard. Although there are other methods for logging into the Pi such as serial connections and network SSH terminals, they all created some difficulties including issues with shutting down the microcontroller correctly to prevent damage to the SD card. The Teensy 3.1 was eventually used, as it had plenty of power to process the data, but it’s memory capacity was a bit lacking. This wouldn’t have been an issue for the Teensy 3.6, but there wasn’t time to order a new microcontroller. The Teensy is fairly simplistic to program, since it uses the same IDE as an arduino and holds all of the same functionality with a few added features such as a faster clock speed.

The display used was a 16-bit 128x128 OLED from https://www.adafruit.com/. A completely functioning library had been written for the Arduino that included printing text and a list of simple primitives, but the commands for interfacing with the display driver circuit directly can be found in the datasheet.

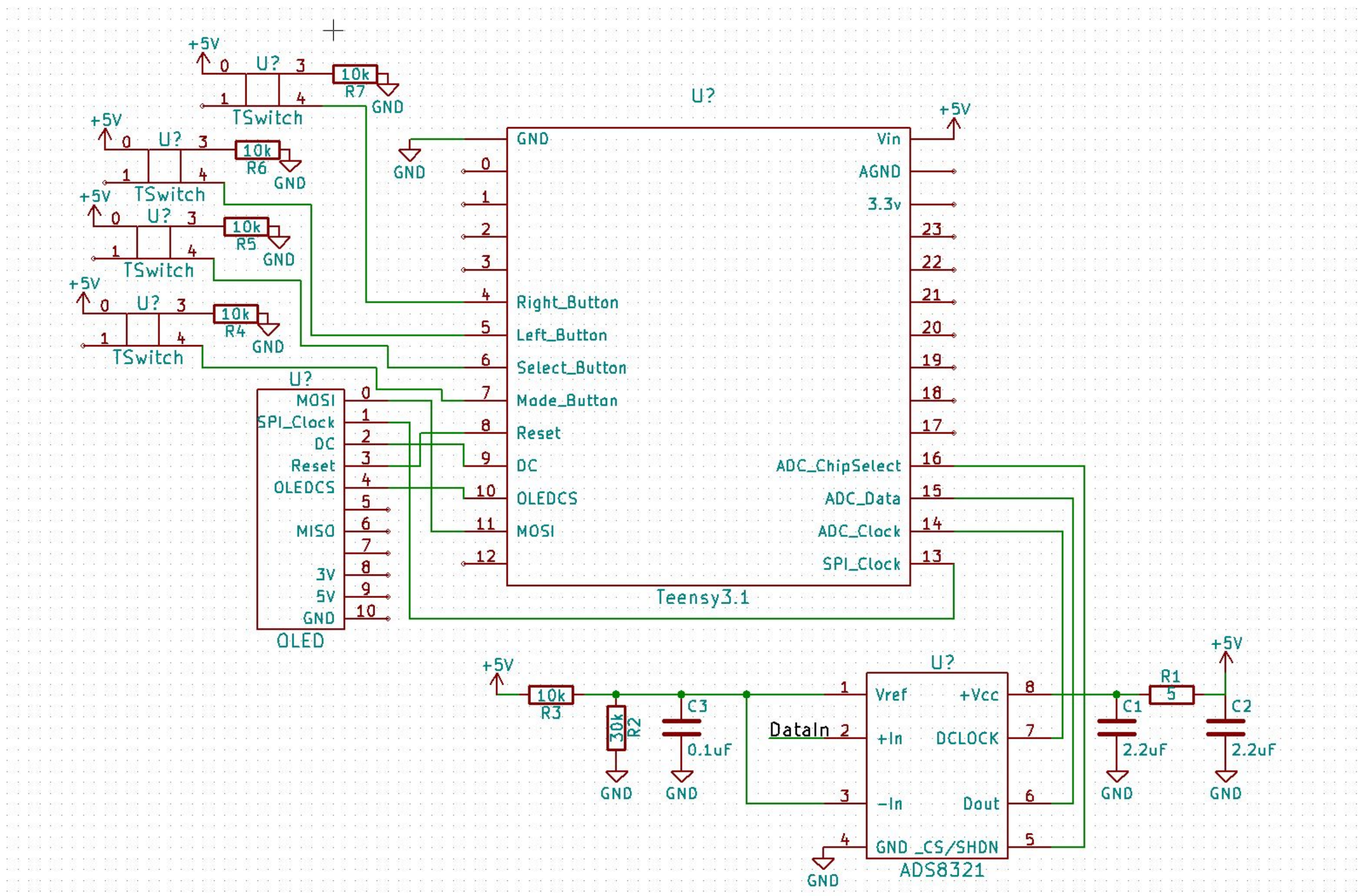

Figure 10. DSP Schematic. Tactile switches with pull-down resistors were used as input devices.

The code written for the teensy can be found here. It's functional enough to get produce a fourier spectrum on the screen, but the data types can be optimized a little more intelligently and the code can be organized to be a little more clean.

Unfortunately the matlab code used within the labs for the class was used to process the data in the end. The DSP was only a few steps away from working correctly though.

## Testing

Since we took the modular approach for the system, it was simpler to test each board for functionality.

### *Baseband*

The baseband was easily tested due to being built on a breadboard. A test signal was used as an input to the amplifier stage. We decided that in order to add versatility and improve gain, two non-inverting amplifiers were cascaded together. The needed gain to amplify a wave traveling a forward 60 meters was approximately 30. By adding a potentiometer as one of the gain control resistors to the non-inverting amplifier, we were able to see on the oscilloscope that the required gain was met. With the cascaded amplifiers, the maximum gain was approximately 900 or 60 dB. Next the low pass filter IC was tested using another external test signal. We decided that the cutoff frequency of the filter should be less than 10 KHz. This was easily achieved by shorting a desired capacitor value from CLK to GND. The amplifier stage and filtering stage was cascaded together and the observed signal on the oscilloscope showed that the test signal was amplified and filtered as desired.

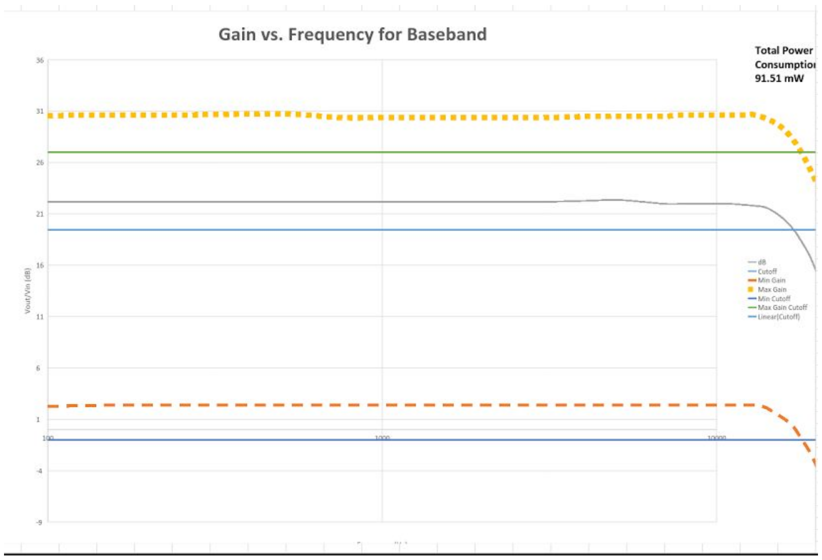

Figure 11. Gain characteristics for baseband system

### *RF Board*

There was difficulty in testing the RF board since there was not a precise instrument to measure the RF signal due to impedance matching. One of the methods to testing the gain and frequency of each component was attaching the probe to a spectrum analyzer and probing the input and output pins to see whether the gain or frequency was consistent with the datasheet. A constant voltage was used for the tune voltage of the VCO. It was measured that at 0V and 2.5V, the frequency span of the VCO ranged from 2.4 GHz to 2.5 GHz, respectively. Next the power and low noise amplifiers were tested on and their gain was found to be close with the datasheet. To test whether the mixer was mixing properly, a test RF signal from the synthesizer was used to determine whether the signal, with a constant tune voltage, from the VCO was downconverted by the synthesizer signal; this method, showed that the mixer was working as intended.

### *Entire System*

For a preliminary test of the entire radar system, the output signal going into the DSP was viewed with an oscilloscope. We wanted to see whether the signal was changing as a reflective surface was placed in front of the antennas. By adjusting the gain from the baseband, the final output signal could be modified for better DSP. From early field tests, we were not able to get satisfactory data with last quarter's DSP. With the help of team Radio Freqs, we realized that signal processing using the code for MATLAB provided far better results than with the code for Python. In addition, at much farther distances from the antenna, the signal could be seen better as the gain from the baseband was increased substantially higher than what was calculated. For distances less than 20 meters, the signal could be seen better for lower gain. Several configurations for the antennas were tested. The two ALFA antennas as transmitter and receiver seemed to include undesirable crosstalk that interfered with the signal reflected by the metal surface. We found that one coffee can waveguide for the transmitter and one ALFA antenna as the receiver provided the best results. With this configuration and the baseband gain set to approximately 50 dB, the maximum range detected was close to 135 meters.

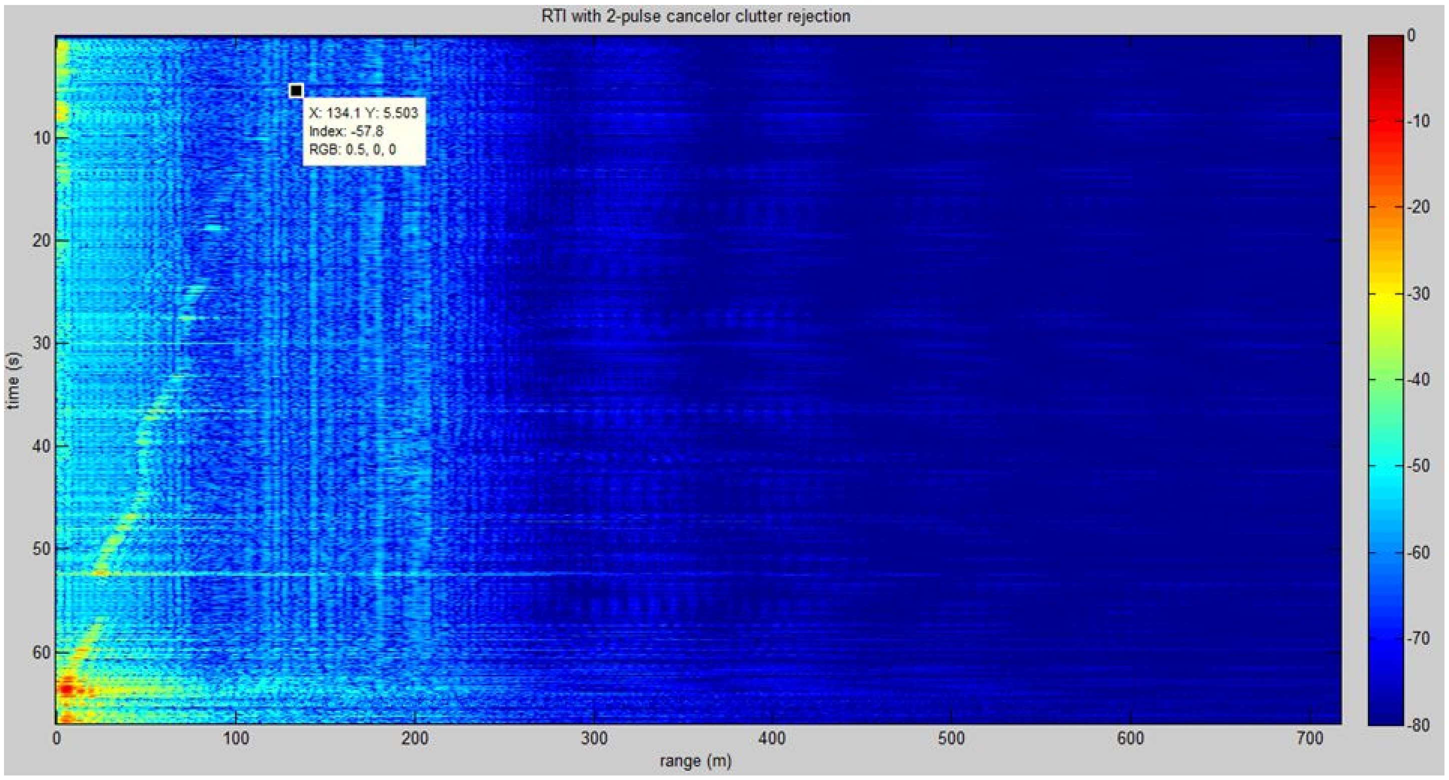

Figure 12. Final results from our best test run

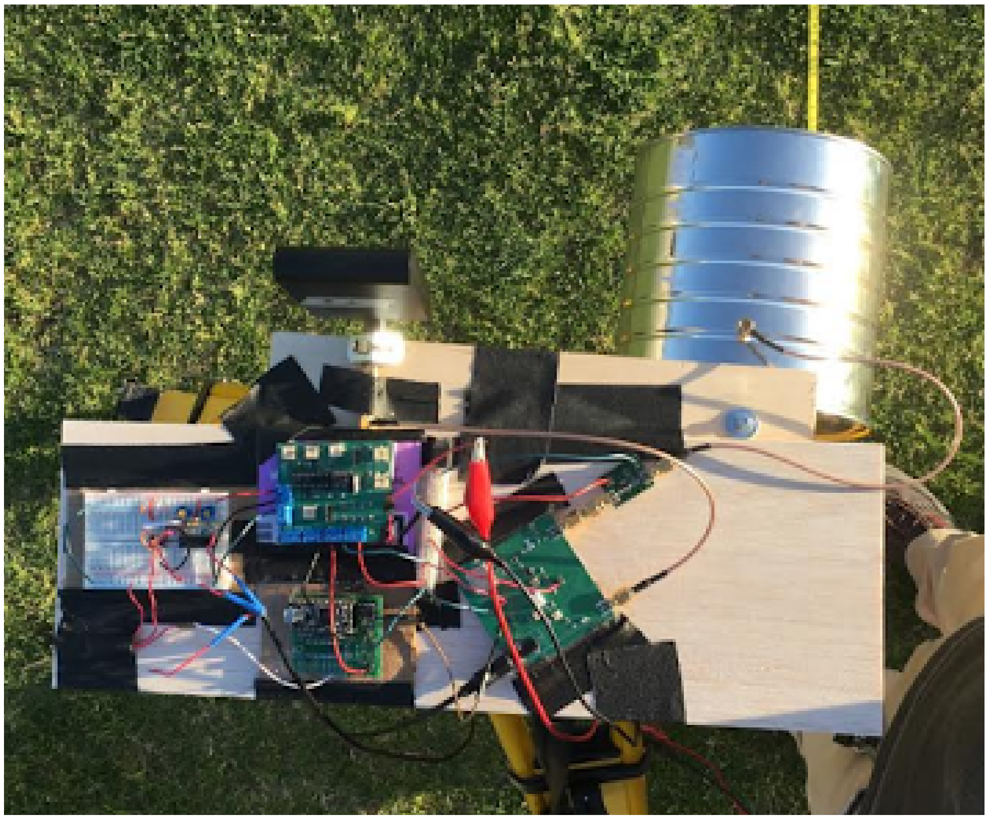

Figure 13. Top view of final implementation

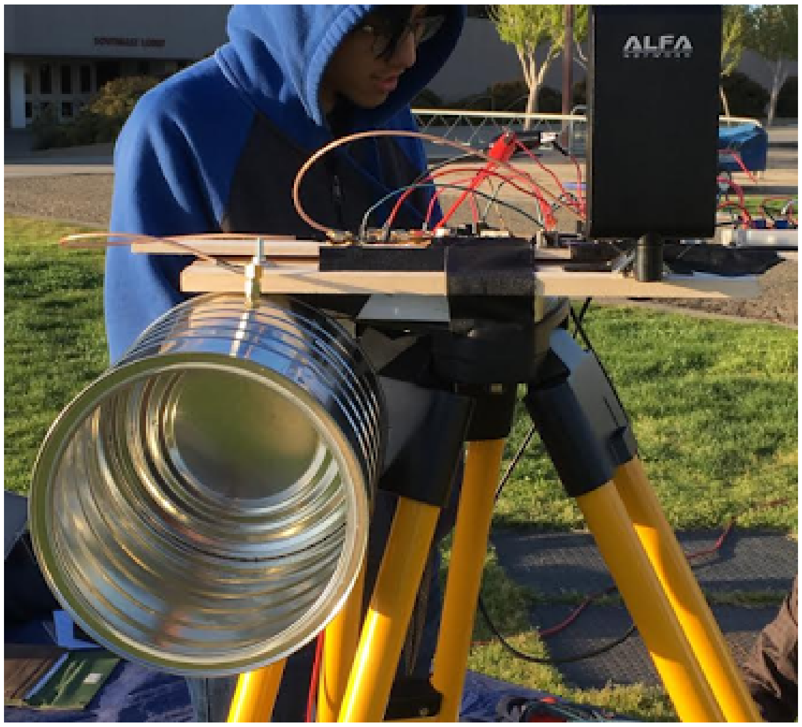

Figure 14. Side view of final implementation with antenna mounts

## **Discussion**

Eventually we ended up replacing the function generator with the teensy and DAC system used from quarter one of this class. We were having problems with getting unreliable data from the DSP portion, and we thought that perhaps there may have been too much jitter in the triangle wave output from the function generator IC. Eventually we found that there was too little gain coming from our baseband amplifier, and that was the major problem we were having. At this point, though, we had run out of time to sufficiently test the difference between the teensy and the IC to see if there was a significant difference. The teensy had already been put in place so we decided to leave it there. Our suspicion is that the triangle wave would still have worked, and with a little more time that would be a place where we could save some significant power consumption. The teensy takes approximately 30 mA to run, with the XR 2206 only a few uA.

There ended up being a lot of empty space in the circuitry portion of our system. This was an unfortunate by product of the modular approach that we took. While this approach was very useful for debugging, we unfortunately ran out of time to really cut down on the excess board space taken up. An example is with the baseband breadboard and the teensy pcb. We ended up modifying the baseband pcb from quarter one in order to make the new teensy function generator work. However there was still a lot of empty space left on the baseband circuit which was on a breadboard. To save space we could have combined these two.

When it comes the PCB, we decided to maintain our first PCB, for it had initially worked without needing much work, but when one of our amplifiers burnt due to incorrect biasing, discovered that the PCB lacked an important feature; test points. As was discussed earlier, the probe used to measure power along the RF traces was unreliable to the point where it was impossible to determine what component or number of components were the ones that did not work. If a revision of the PCB had been made, we would have included test points in the RF lines that went after each component to the side of the board and would be terminated on footprints for

SMA connectors. By doing this, one could simply solder a temporary SMA connector what could then be connected directly to the spectrum analyzer in order to obtain more precise readings of the power.

The signal processing was nearly complete by the end of the project, but too much time was wasted using the Raspberry Pi. The code just needed to display a plot similar to the MATLAB code in Figure 12 in real time with a few techniques to reduce the background noise. A few simple clutter rejection techniques would be simple to add and the real time plot would not require much to code with a spectrum already being generated every second. There are also a few easy ways that the image could have been compressed to accommodate the lack of RAM on the Teensy 3.1.