EEC-134AB: RF System Design Senior Project 2015-16

Team J (AKA J-Crew)
Members:
Joe Cooney
Jimmy Hua
Jesus Beltran

PURPOSE

The purpose of this RF System Design Senior Project was to design an radar system that could detect objects at a maximum distance of 50 meters. In addition, the radar system would be judged by three criteria: weight, power consumption, and accuracy. At the very end of the term, groups were able to compete against each other to see who best met the radar criteria. The criteria would be judged by measuring the weight, and power consumption, as well as by having the team determine the location of an object in a field, based solely off of their radar results.

DESIGN

We decided to design of the radar system was based on Lab 6's block level diagram. Since none of the members of our group had extensive RF background we determined it was best to stick to a circuit topography we already had worked with and understood. What was unique in our design was our use of two amplifiers in the receiver end of the radar system to better boost the signal while at high frequency, in order to suppress noise, and to achieve a larger signal out of the mixer.

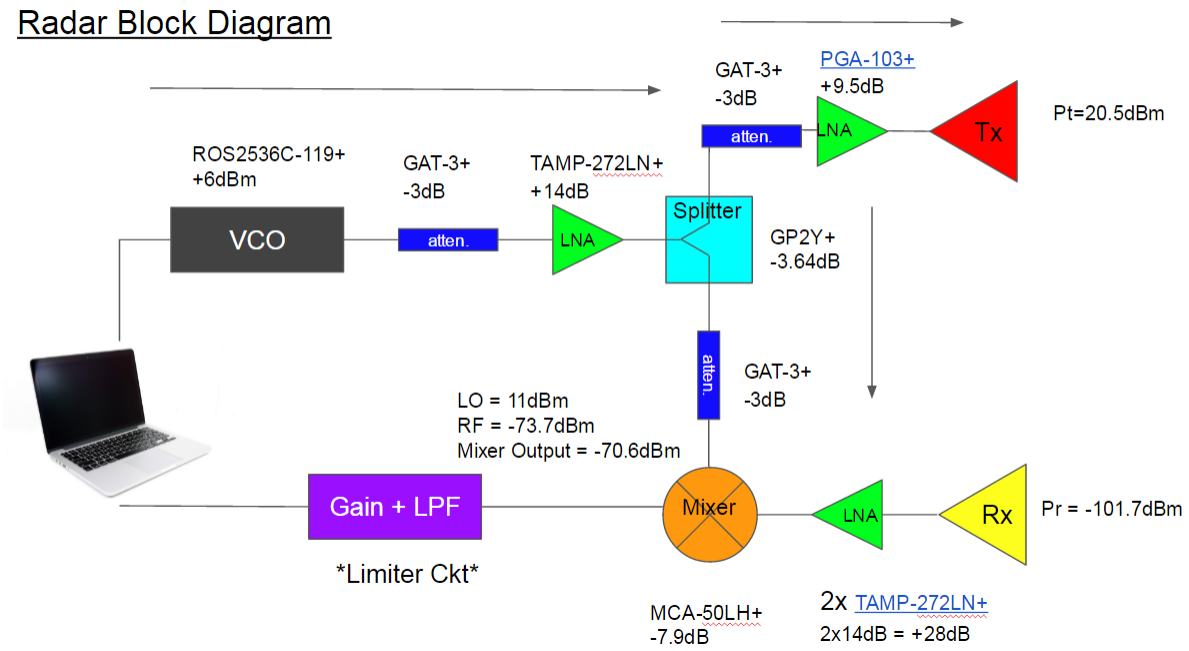

Figure #1: Radar System Block Level Diagram

The design was split into two separate printed circuit boards (PCBs). One of the PCBs was an RF PCB, containing all of the high frequency components (2.4 GHz), while the other PCB was used for baseband filtering and amplification (approx. 35 Hz). We did this because troubleshooting the boards would have been difficult if both RF and baseband PCBs were on one board. By grouping together everything with similar frequencies, it is easier to understand where some of the errors and issues are during testing. This also was a benefit as our boards

were essentially backwards compatible, allowing us to test new boards with our setups from lab 6 which we had already been able to get working.

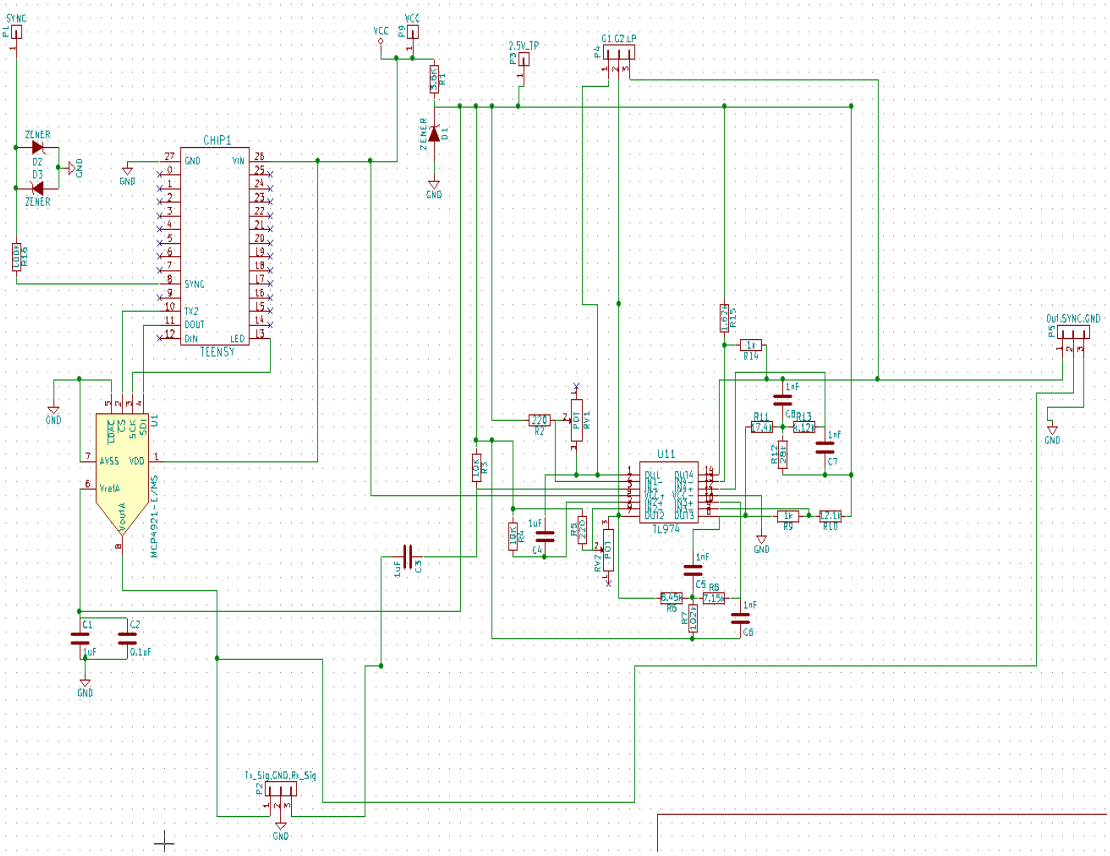

Figure #2: Baseband Schematic

The design for the baseband circuit very closely followed the design of the circuit used in Lab 6 with the addition of a clipper circuits for the SYNC, and final filtered signal.

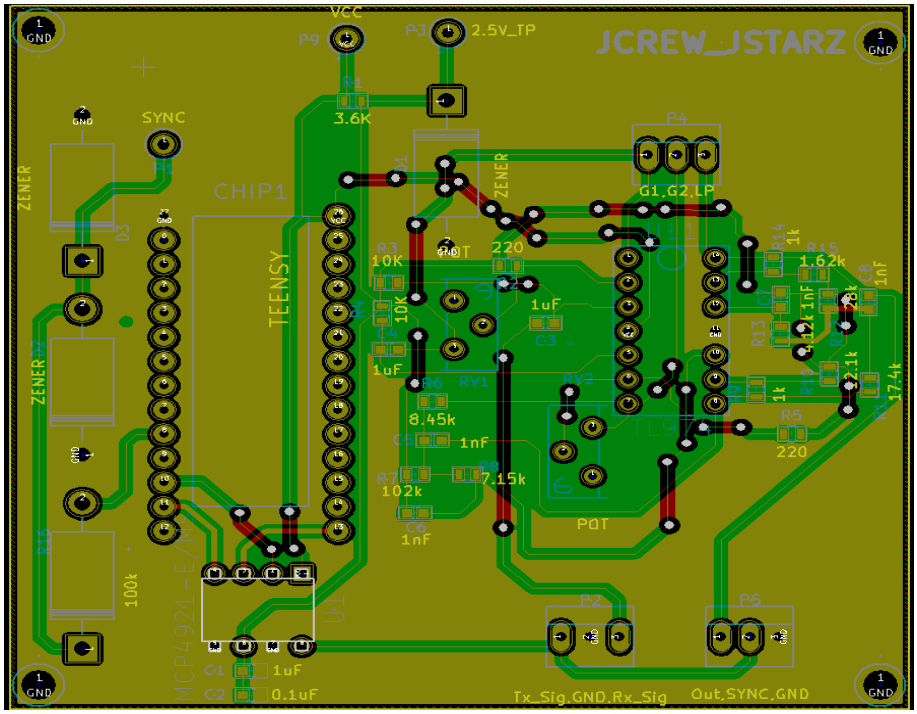

Figure #3: Baseband PCB

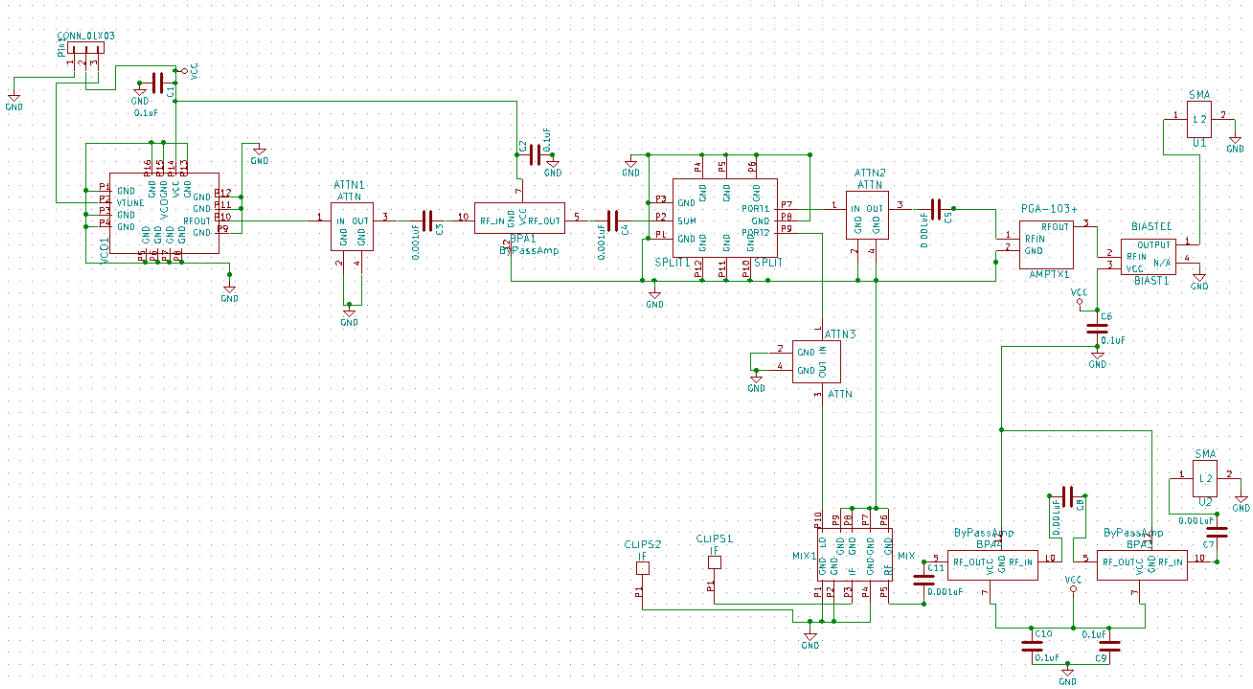

Figure #4: RF Schematic

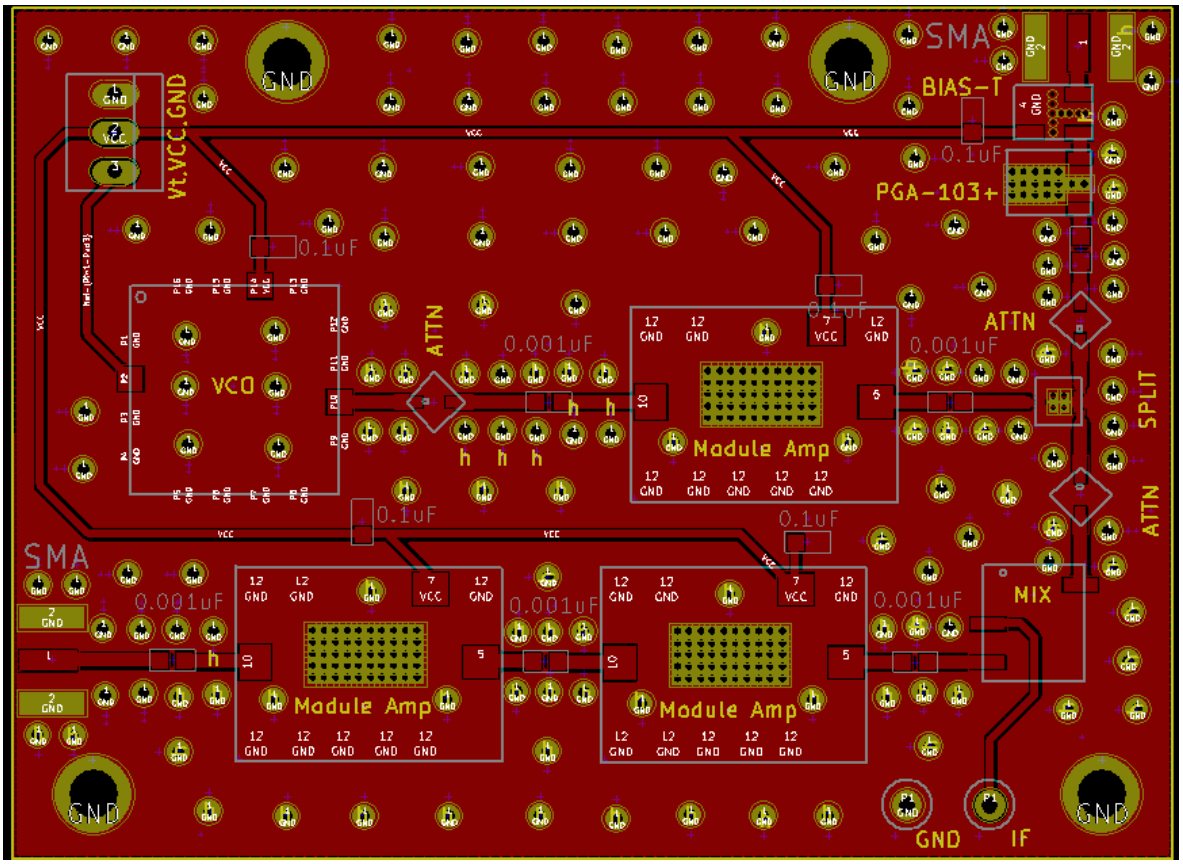

Figure #5: RF PCB

When choosing a transmitted power we were advised to shoot for the range of about 20-30 dBm. With that in mind, we designed our RF PCB with a transmitted power of approximately 20.5 dBm. By our power calculations, the received power was expected to be relatively low. This turned out to be around -100 dBm at the Rx antenna (for a distance of 50 meters). This was then set to be our input power for the receiver portion of the RF PCB in our power simulation (ADISimRF). The reason why we wanted to have a larger signal after the power amplifiers was so that the noise would be reduced according to Friis' Formula for noise figures. Also if the received signal going into the mixer is too small, the output of the mixer does not contain viable information. The purpose of the mixer is to create two sideband signals which is the sum and difference of the received signal compared to the output signal. Therefore, if no signal is detected at the Rx, then there is nothing that can be mixed, and the result is just noise at the output of the mixer.

Tx Power Calculations

Table | Chart | Device Selection

	Stage 1	Stage 2	Stage 3	Stage 4
Transmit	Atten	LNA	Device	LNA
Toggle Tx/Rx	Temp Part	Temp Part	Temp Part	Temp Part
Output Freq (MHz)	2400	2400	2400	2400
Zin (Ohms)	50	50	50	50
Zout (Ohms)	50	50	50	50
Power Gain (dB)	-3	14	-6	9.5
Voltage Gain (dB)	-3	14	-6	9.5
OIP3 (dBm)	100	30	100	30
OP1dB (dBm)	90	19.5	90	22.7
Pout (dBm)	3	17	11	20.5
Pout Backoff	87	2.5	79	2.2
Peak Backoff (dB)	87	2.5	79	2.2
Noise Figure (dB)	0	0	0.8	0
Voltage (V)	0	5	0	5
Current (mA)	0	55	0	60

Pt = 20.5 dBm
Pwr Cosumption = 0.58 W

Input

Number of Stages	4	
Input Power	6	dBm
Analysis Bandwidth	1	MHz
PEP-to-RMS Ratio	0	dB
P1dB Backoff Warning	2	dB
Peak Backoff Warning	1	dB

Analysis

Output Power (rms)	20.5	dBm	Noise Figure	0.07	dB	OIP3 (Po/2 per tone)	28.4	dBm
Output Voltage (rms)	2.37	Vrms	Output NSD	-159.2	dBm/Hz	IIP3 (Pin/2 per tone)	13.9	dBm
Output Voltage (pp)	6.69	Vpp	Output NSD	2.4	nV/rtHz	IMD (Po/2 per tone)	-21.8	dB
OP1dB	19.84	dBm	Output Noise Floor	-99.2	dBm	SFDR	85.1	dB
IP1dB	6.3	dBm	SNR	119.7	dB	ACLR (est.)	-37	dB
Power Gain	14.5	dB				Pwr Consumption	0.58	W
Voltage Gain	14.5	dB						

Figure #6: TX ADISim Power Calculation

Above can be seen the power consumption of the Tx and how we came up with our value of 20.5 dBm output power. Note that the VCO is not included in this (not available in ADISimRF), and that some of the attenuating components have been combined (splitter -3dB combined with atten. -3 dB).

Rx Power Calculations

Table | Chart | Device Selection

	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Receive	LNA	LNA	LNA	LNA	Mixer (Rx)
Toggle Tx/Rx	Temp Part	Temp Part	Temp Part	Temp Part	Temp Part
Input Freq (MHz)	2400	2400	2400	2400	2400
Zin (Ohms)	50	50	50	50	50
Zout (Ohms)	50	50	50	50	50
Power Gain (dB)	14	14	14	14	-7.9
Voltage Gain (dB)	14	14	14	14	-7.9
IIP3 (dBm)	30	30	30	30	100
IP1dB (dBm)	19.5	19.5	19.5	19.5	91
Pin (dBm)	-101.7	-87.69	-73.69	-59.69	-45.69
Pin Backoff (dB)	121.2	107.2	93.2	79.2	136.7
Peak Backoff (dB)	121.2	107.2	93.2	79.2	136.7
Noise Figure (dB)	0	0	0	0	0
Voltage (V)	5	5	5	5	0
Current (mA)	55	55	55	55	0

Output Power is actually higher (-42.6 dBm)since LO power not taken into account for mixer

Input

Number of Stages	5	
Input Power	-101.69	dBm
Analysis Bandwidth	400	MHz
PEP-to-RMS Ratio	0	dB
P1dB Backoff Warning	5	dB
Peak Backoff Warning	1	dB

Analysis

Output Power (rms)	-53.59	dBm	Noise Figure	0	dB	OIP3 (Po/2 per tone)	35.92	dBm
Output Voltage (rms)	0.47	mVrms	Output NSD	-125.7	dBm/Hz	IIP3 (Pin/2 per tone)	-12.2	dBm
Output Voltage (pp)	1.32	mVpp	Output NSD	116	nV/rtHz	IMD (Pin/2 per tone)	-185	dB
OP1dB	24.42	dBm	Output Noise Floor	-39.7	dBm	SFDR	50.4	dB
IP1dB	-22.7	dBm	SNR	-13.9	dB	ACLR (est.)	14	dB
Power Gain	48.1	dB	Input Rx Sensitivity	-77.8	dBm	Pwr Consumption	1.1	W

Figure #7: RX ADISim Power Calculation

In **Figure #7** the Rx power calculations can also be seen. However, this is not the final design we used. Since it is more efficient to boost signals at baseband using Op-Amp amplifiers we opted to reduce our LNA's from four to two. Remarkably the power consumption did end up to be about the same in total (1.5 Watts). But that is coincidental since this does not take into account the filtering stages, teensy microcontroller, and other components that draw power throughout the system.

Another design assumption that we kept similar to lab 6 was the signal's frequency that was set at 2.4 Ghz. Since we successfully tested the can-tennas in Lab 5 and did not have any background in radar design, we decided to stick with the cantennas and design a lightweight system to offset the bulkiness of the cans. This also allowed us to avoid the issue of the spectrum analyzer's limitations because the analyzer cannot work above 3 Ghz. Shown below is the gain of our cantennas that were used in Lab 5.

$$G = G_1 = G_2 = \frac{4\pi r}{\lambda}\sqrt{\frac{P_r}{P_t}}.$$

With r = 1 meter, and wavelength 12.5 cm, and experimentally determined Pr, and Pt. The Gain was found to be 3.87 dBi

Figure #8: Can-tenna Calculated Gain

Something that we came across during our redesigning stage was the use of a clipper circuit for both the signal after the low pass filter and the SYNC signal. The clipper circuit was connected in a 2 diode configuration to cap both maximum and minimum voltage at a preferred output level. Since the signal that is going into the laptop for signal processing can be relatively high voltage if the object is close to the antenna/receiver, the laptop could experience hardware failures because of the laptop's ADC maximum voltage tolerance. Therefore, it is advised to send a signal into the laptop's microphone port, and observe at what point does increasing the voltage of the input signal result in a maximum use of the ADC. At this point, any input signal with significantly higher voltage levels does not contain any additional information and may destroy the hardware in the laptop.

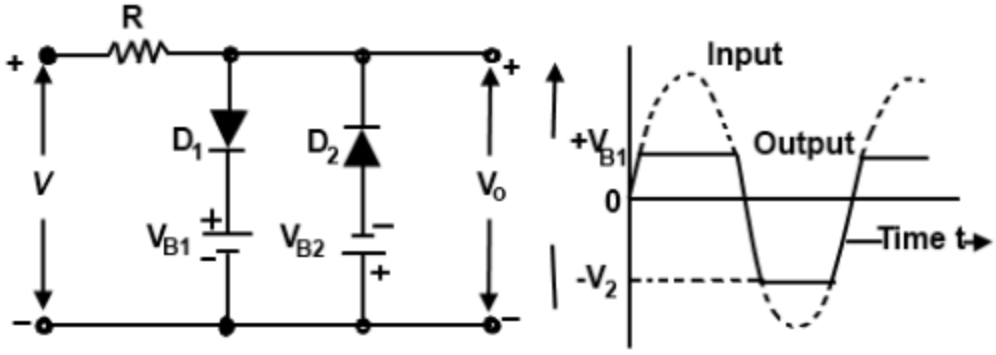

Figure #9: Dual Diode Clipper Circuit

We encourage the PCB designers to implement test clipper points to easily analyze and troubleshoot both RF and baseband PCB prototypes. The following are test point recommendations to help identify the signal integrity: GND, VCC, SYNC, RX(received signal after the mixer), TX(transmitted signal after the power amplifier), OPAMP1(after first operational amplifier), LPF(after second stage low pass filter).

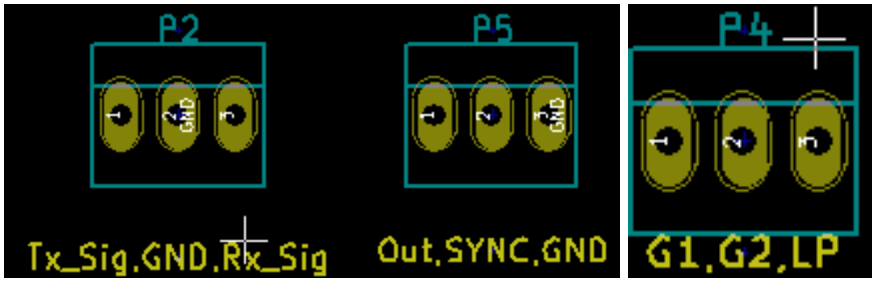

Figure #10: Test Clipper Points

IMPLEMENTATION

All the RF components were obtained through Mini-Circuits. Most of the RF components that we planned on using we received through Mini-Circuits' as free samples. Through free sampling, we were able to obtain 4 of each component. This allowed us to have enough samples to make two boards, it is preferable to not have so few components as mistakes happen often, and reordering more components can set you back weeks. The components that we were not able to obtain through free sampling, getting in touch with a Mini-Circuits representative made it possible to receive a small sample size of the needed RF components. **Table #1** contains the list of the RF components used. All other components used were standard resistors, capacitors, and diodes available to us in the lab.

Component	Mini-Circuits Part Number
Voltage Control Oscillator	ROS-2536C-119+
Attenuator	GAT-3+
Mixer	MCA-50LH+
Splitter	GP2Y+
Power Amplifier	TAMP-272LN+
Power Amplifier for Tx	PGA-103+
Bias T for Power Amplifier for Tx	TCBT-14+

Table 1: Mini-Circuits RF Component List

The baseband PCB and the RF PCB designs were fabricated through Bay Area Circuits. Each design we received multiple copies of each PCB. Having more than one copy made it possible to make a spare PCB, provided we had spare components. Provided in **Figure #11** and **Figure #12** are the soldered baseband PCB and RF PCB.

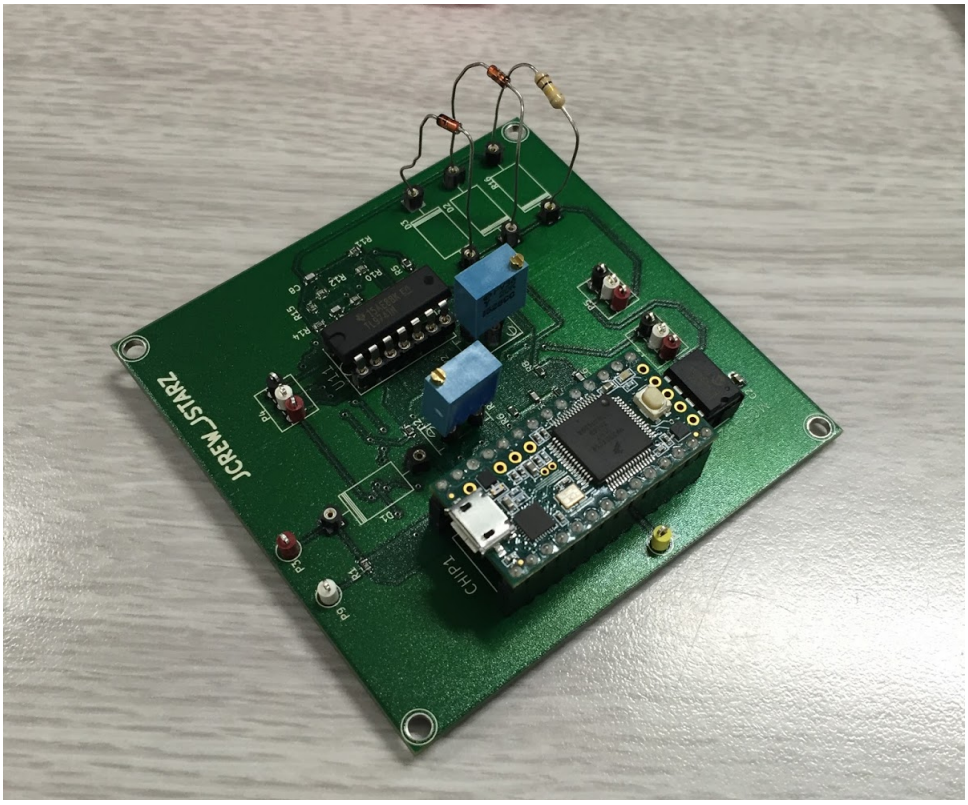

Figure #11: Baseband PCB Soldered on Components

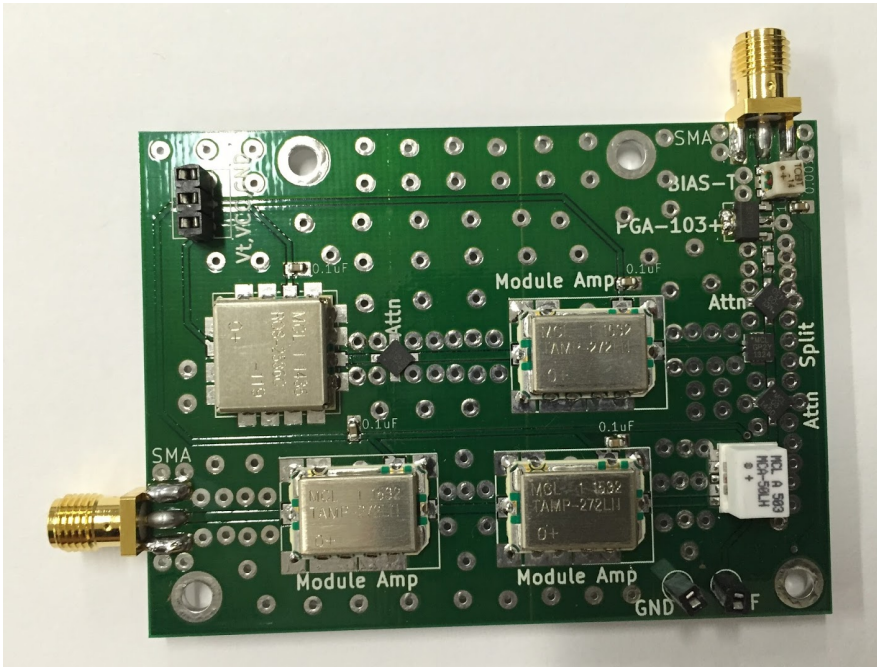

Figure #12: RF PCB Soldered on Components

In order to make the RF system lightweight, we chose to use a plastic black box that would contain the PCBs. In order to secure the PCBs in place, through holes for standoffs were incorporated in the PCB design as seen in **Figure #12**.

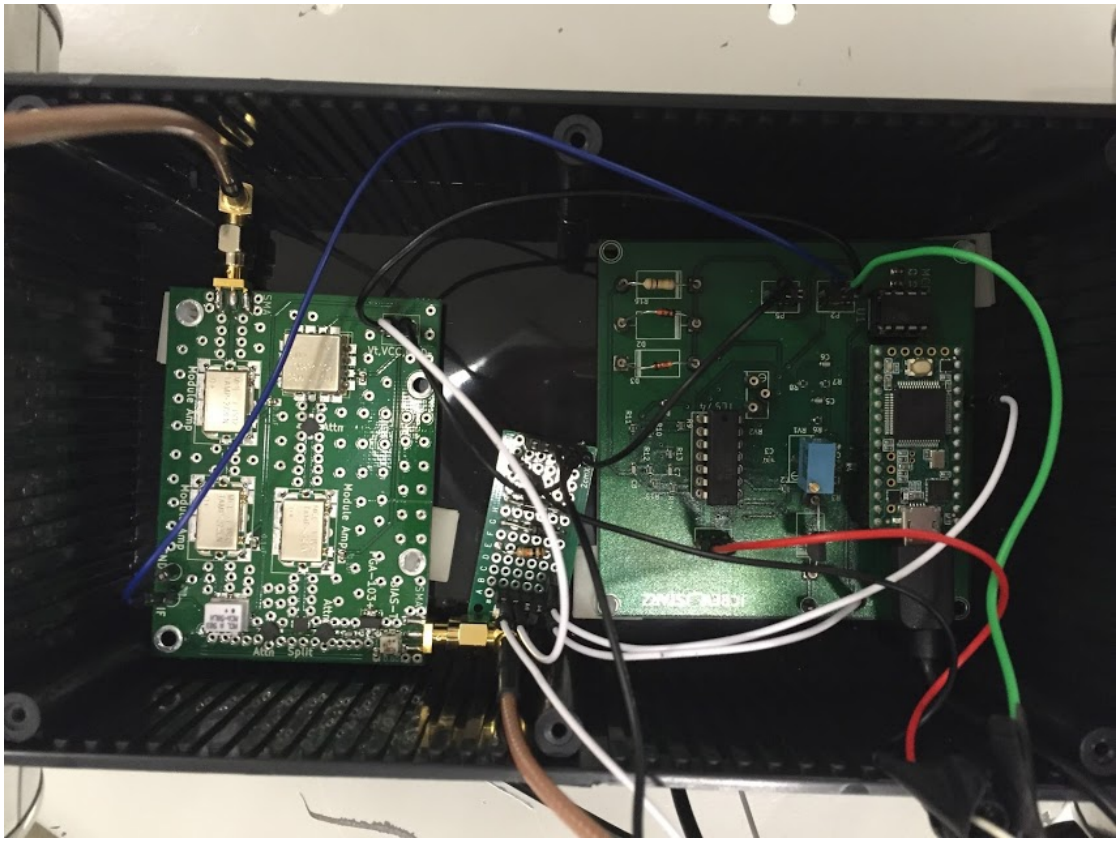

Figure #13: PCBs Mounted Inside Black Box

Since none of us had any experience designing RF antennas, we chose to use the tin can antennas for our RF system design. To mount the antennas on the black box, at first it was suggested to screw the antennas to the front of the black box, but upon receiving the black box, the dimensions were too small to accommodate the antennas such that the SMA cables would be able to reach the RF SMA connectors. Instead, we chose to use velcro to mount the antennas on the side of the black box. The final product of the RF system design is seen in **Figure #14**.

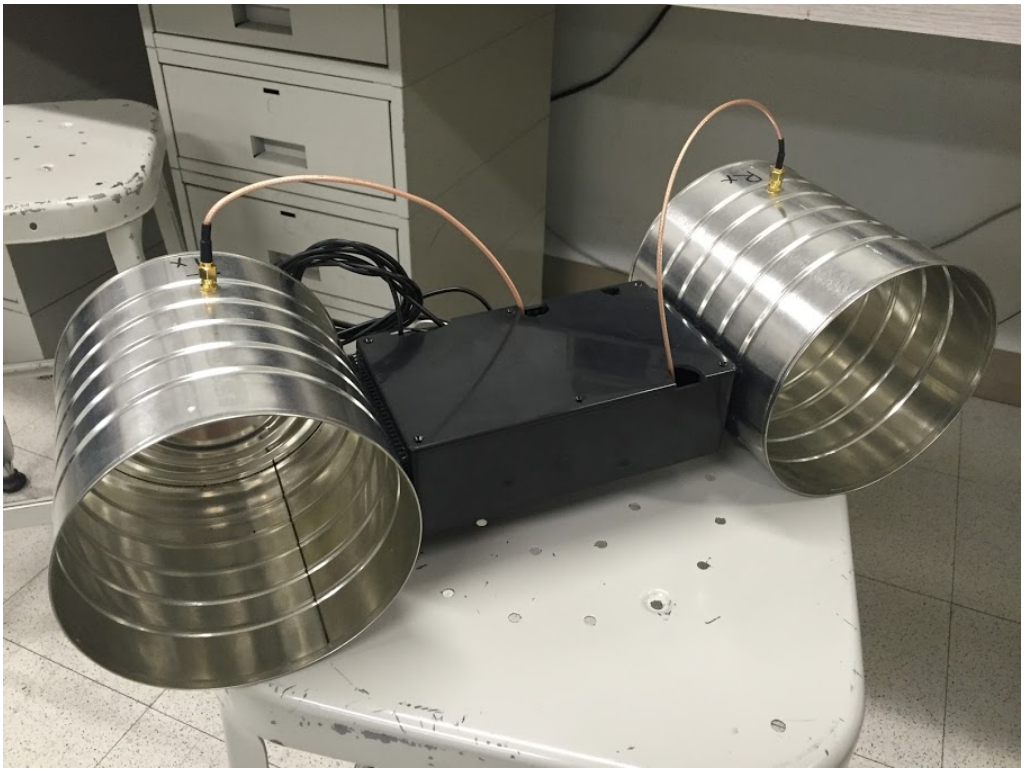

Figure #14: RF System Final Design

During testing, it was observed that the baseband PCB only required one gain stage instead of two from the original PCB design. We used a sinusoidal input signal as the received signal with a peak-to-peak amplitude of 100 mV. The first observation was the clipping of the sinusoidal signal for the low peak when there was about a 30 to 1 amplification. In addition, when actually testing the baseband PCB with the RF components from lab 6, we were not getting the results as intended, rather we observed too much background noise and/or didn't detect an object. Therefore we chose to use one gain stage instead. To get the received signal to go through one gain stage instead of two on the PCB, we removed a couple of components and shorted two test points together. From this, we tested the one gain stage using a sinusoidal signal as the received signal and still observed clipping of the low peak. However, using the RF components from lab 6 we were actually able to observe less background noise and detect stationary objects. Therefore, we decided to keep the current baseband configuration with one-gain stage.

There were two implementations of the RF PCB; we had enough components to make a spare. The first implementation of the RF PCB had rework done on several occasions, for example the VCO had been soldered on incorrectly. This required removing pins to create a flat bottom surface in order to reheat the PCB to remove the VCO and set it back on PCB correctly. During testing, in conjunction with the baseband PCB, the first RF PCB worked fairly well. We were able to detect objects to about 30 meters. We were able to do field tests on several occasions with the first RF PCB, however, at some point and it's not entirely clear what happened, but we were no longer able to detect objects any further than 20 meters. We came to the conclusion

that the rework done on the first RF PCB may have damaged the other components from repetitive heating. Therefore, we decided to make another RF PCB since we had enough spare RF components and another RF PCB. Fortunately, soldering on all the components took one single effort on the second RF PCB. After the second RF PCB was made, we conducted indoor tests and observed much better results than the first RF PCB. Upon conducting field tests with the second RF PCB, we observed that we could detect objects much further than 50 meters, which was the goal of the upcoming competition.

Code Implementation:

When we went to alter the code to better work with stationary objects we opted to use existing MATLAB code from the MIT Open Course "Introduction to Radar Systems". Our team is more familiar with MATLAB than with the provided Python code.

Our code went through many different renditions over time, and we ended up creating 5 different versions of the code. The main goal of our code was to be able to find the location of a stationary object without relying in anyway on vision or knowing where the object was located. We assumed that we could easily be asked to find a target without being able to look at the field. This is a very important aspect of any good radar system, after all there isn't much benefit in a radar if you can already tell where an object is located by vision alone.

In order to find the object we relied used the data that had already been scaled and displayed in the "Without Clutter Rejection" figure of the existing code. This simplified the problem for us and allowed us to look at the data as just a large matrix with different intensity levels for each location of the matrix. We recognized that for a still image we are concerned with the vertical lines that run from top to bottom (indicating a non-moving object). Using this knowledge and the assumption that there is only a single object in the field we need to look for a peak in intensity along the x axis. It took time to settle on our final design, since it isn't necessarily the most intense vertical line that indicates the object, but rather how the intensity varies about the object. This issue is very apparent when trying to find an object far away from the radar, because there is a distance at which the intensity close to the radar exceeds the intensity away from it, or the reflections from the ground close to the radar become the strongest returned signal. For sometime we used code that looped through the matrix starting at the right of the image and once it found a much higher signal than the average intensity it would isolate that area and then find the maximum intensity, and choose that as the object location. However, we found that there was much variation using this code and that although it could work successfully in one location, hard coded values had to be changed depending on the spot, or time of day. So while it would have been nice to have a fully automatic system, we opted to use more human involvement in our final design to insure maximum accuracy. A fully automated system however seems very achievable although it would likely require a much more sophisticated algorithm than we currently have.

Our final code design first presents the user with a plot showing both the scaled time vs. distance plot and directly below it an intensity vs. distance 2d plot. This way the user can look at two different representations of the same data and determine roughly where they think the object is located. By having the 2d plot of intensity one can look at where the intensity has local maximum to determine a guess at the distance of the object. They can then enter this distance into code and it will isolate the area about the guess. It can then determine the maximum in this isolated area and that is returned to the user as the distance. In addition to this distance a rounded and linearly scaled version distance is returned, since it is assumed that the object was placed at a whole number (either in ft or meters). We noticed that our system went nonlinear as distance increased and this linear factor attempted to mitigate some of the non-linearity. Finally a filtered plot is returned that shows graphically the same result as the returned printed distance.

RESULTS & DISCUSSION

The signals that were observed under an oscilloscope was what we had expected. The further the object was from the radar system, the smaller the characteristic frequency of the signal. The closer the metallic object is to the radar system, the larger the frequency is. The results also shows a signal that was static rather than varying when constantly fixed.

The result of our radar system went according to our expectations. During the Hutchinson field testing, we were able to achieve a minimum range of 50m and a maximum range of approximately 90m. The test was conducted by having a person hold the metallic plate and walked back and forth from the radar system to a designated distance.

Code Results:

Below can be seen a typical .wav file that would be sent into the code. It can be seen that the channel on top is a sync square wave while the one below is the received signal that has been filtered and amplified. The code then performs an FFT on the signal and is able to then transform the frequency information into distance and intensity information since the difference in returned frequency from output frequency is directly proportional to distance, and the intensity can be determined by the power in frequency domain. We would often check how our signal looked in audacity to assure ourselves that we were picking up a periodic signal that somewhat followed the frequency of our sync signal.

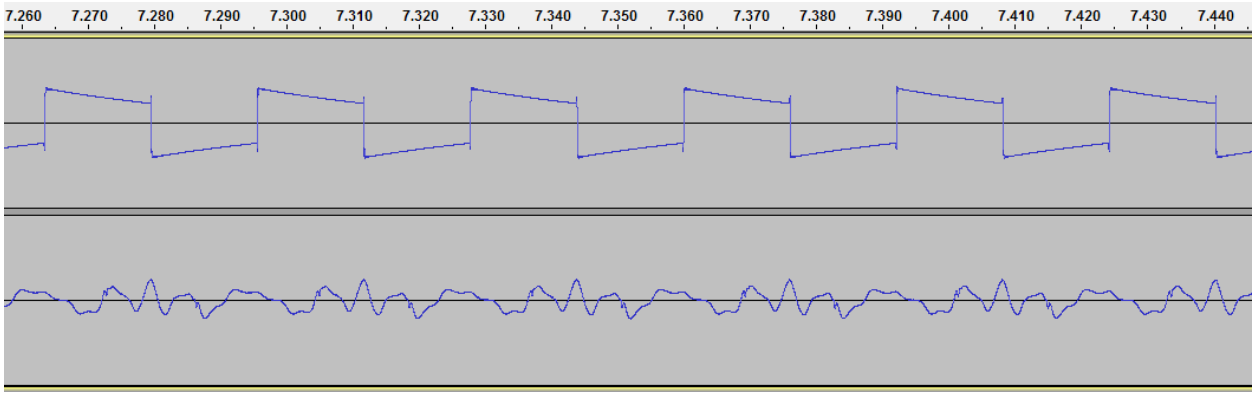

Figure #15: Waveforms in Audacity

Below is some of the preliminary testing we did prior to the final competition testing.

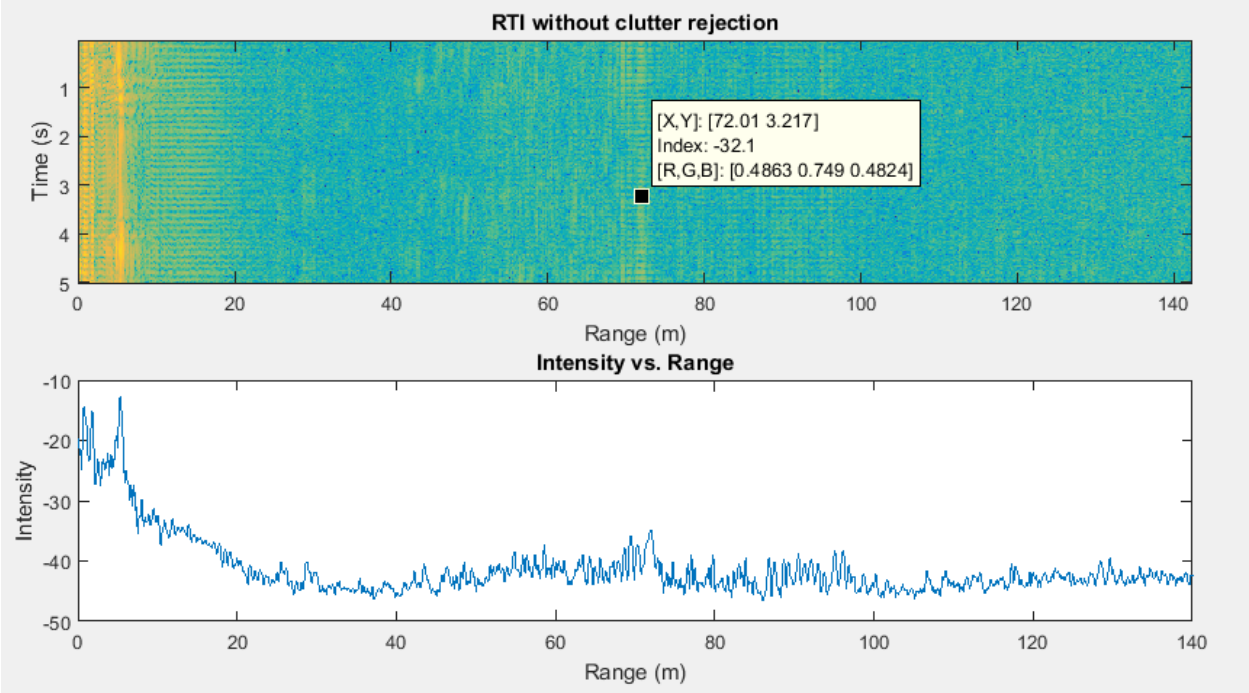

Figure #16: Still Object Testing

We can see in the above plot that there is a vertical line at approx. 72 meters, but we can also see that there is a peak in the bottom graph (local maximum) also at 72 meters. The user can also click on the 2d plot and determine their “guess location” by selecting the peak seen there. The guess value is then typed into matlab as the estimated position in meters.

```
Estimated Position in Meters?: 72

 Guess is at 72.00 meters, 236.22 feet


The object seems to be located at 72.049 meters, or 236.381 feet


Most Likely Value: 240 Feet (approx. 73 Meters)
```

Figure #17: MATLAB Printed Input/Output

The guess at 72 meters, allows the program to then isolate the area around 72 meters and search for the maximum. In this case it found the object to be located at 72.049 meters. But then a linear scaling, and rounding was applied to this result and was able to place the object at 240 feet as the “most likely” location. We were very happy with this result since for this measurement we had used a tape measure to place the object at exactly 240 ft. This told us that even without any extra scaling we are accurate at 240 ft to within roughly 2% or about a meter. With the added scaling and rounding we were able to directly pinpoint the location.

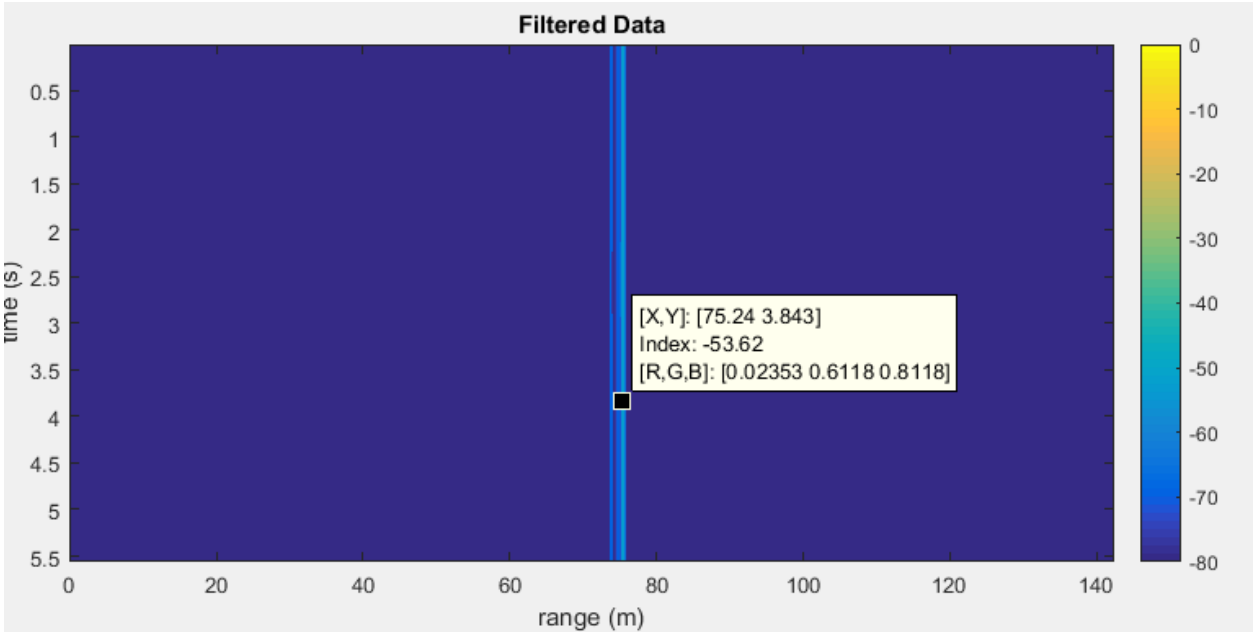

Figure #18: Competition Result (Clutter Rejection Only)

The above shows a graphical representation of what the code uses to determine where the local maximum is, and thereby the location of the object.

During the testing phase we often used a slightly modified version of the original “walking” code to determine how the radar was functioning and what our maximum distance might be. We were able to see that the radar was functioning well at 76 meters which was 26 meters more than we had set out to achieve. We felt very confident at this point since the radar was far exceeding our expectations.

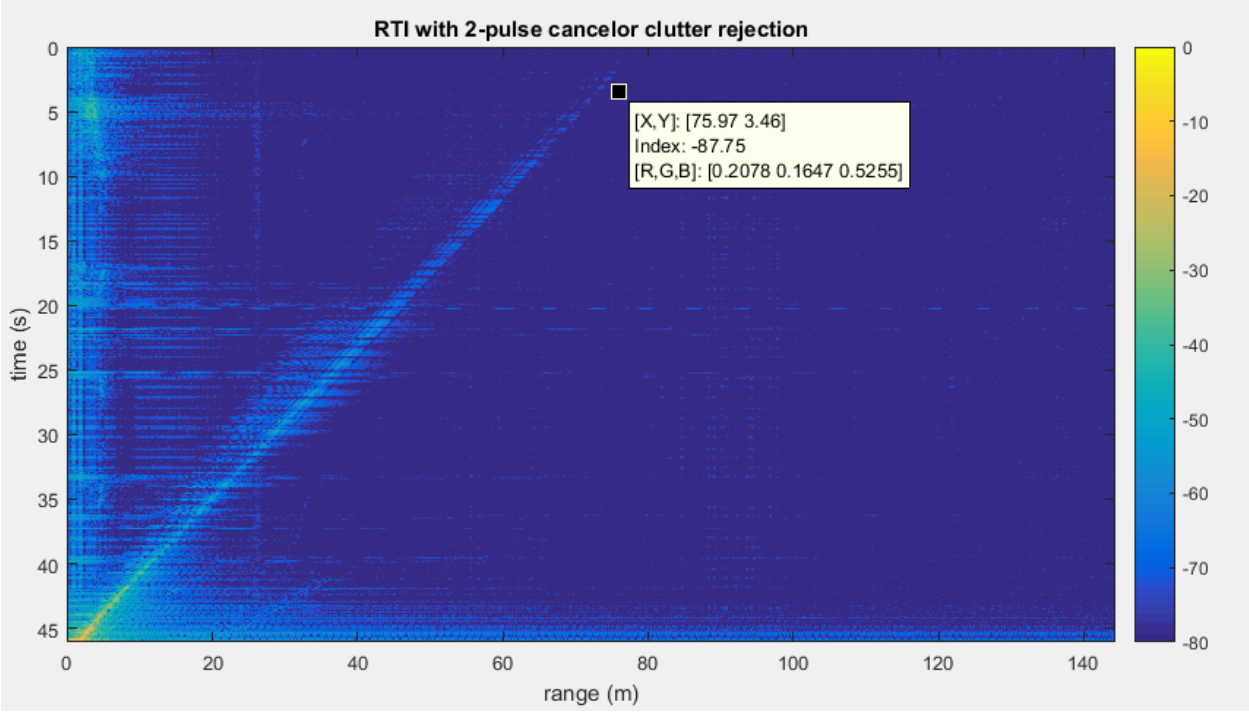

Figure #19: Preliminary Testing (Clutter Rejection Only)

The figures below show our final results taken from the day of testing. The testing process did not occur quite as we thought it would. There was much more of a focus on distance, power consumption, and weight than on the accuracy of the system for still objects. We were a little surprised that the final measurement that we were judged on was based more off the fact that we could detect something at long distance rather than we could detect the position of an unknown object. We ended up using original code to determine the final max distance. We had someone stand at 300 meters for a few seconds and then walk back towards the radar. It can be seen in the results that there is a straight line for a few seconds and then a sharp bend as the object begins to walk in towards the radar.

In **Figure #20** one can be seen the raw data, and then a highly contrasted image of the same data in **Figure #21** helps accentuate the path of the object.

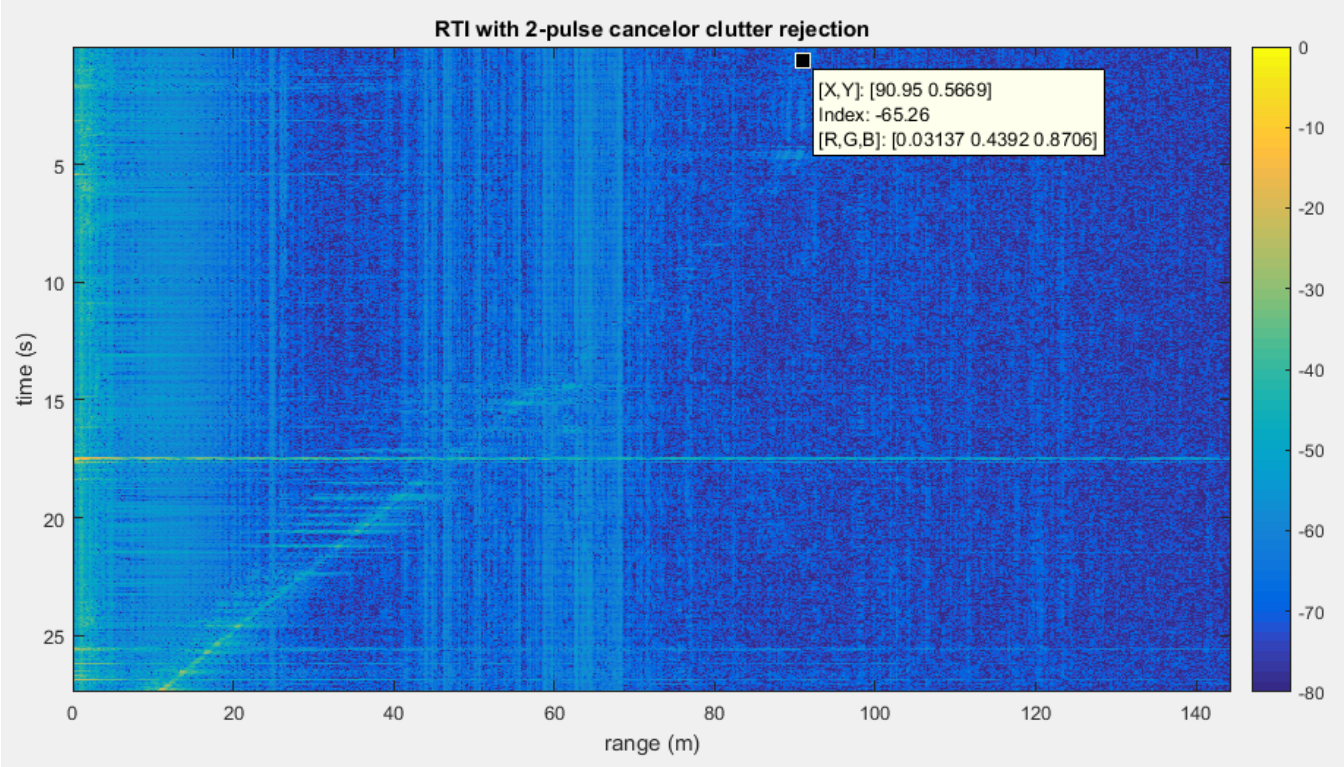

Figure #20: Competition Result (Clutter Rejection Only)

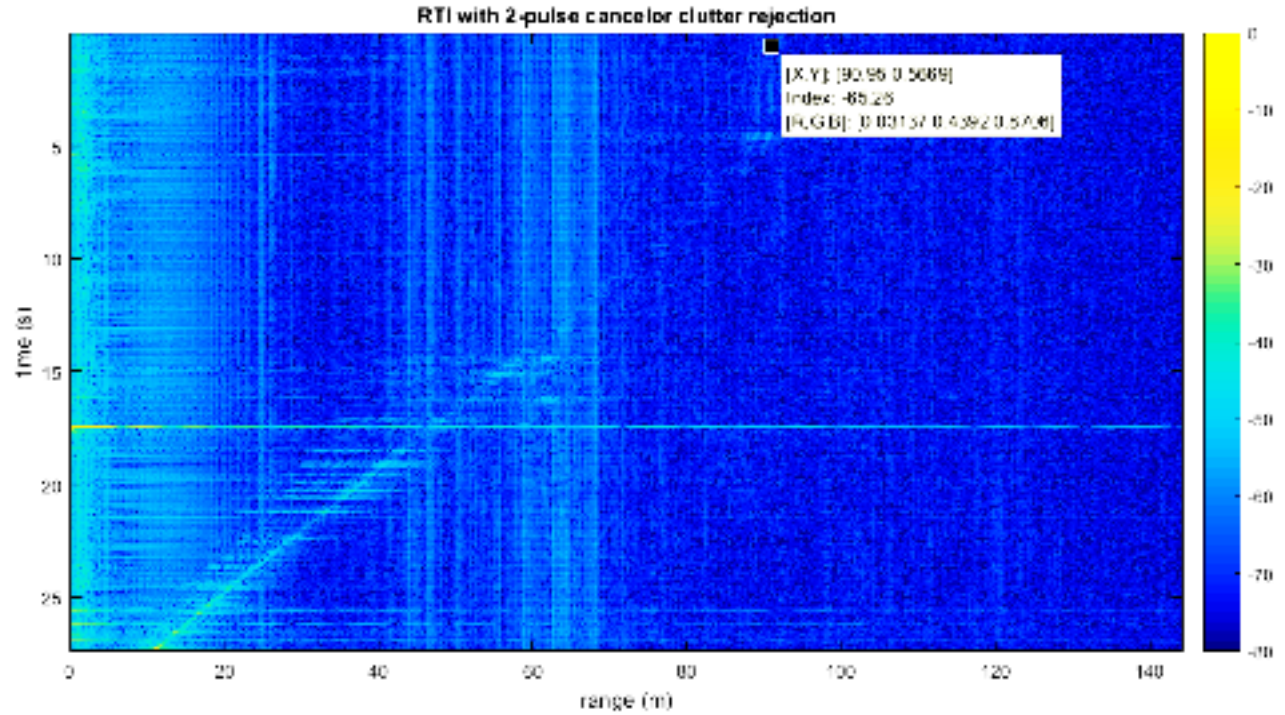

Figure #21: Competition Result (Clutter Rejection and High Contrast)

At the bottom right of the information box in **Figure #21** there can be seen a “corner”, this is what we based our final measurement off of, which was 91 meters. Since the objects max distance was at 300 ft or 91.4 meters we were able to have less than 0.5% error at 91 meters. We were quite pleased with this result, since we only set out to make a radar capable of observing objects at around 50 meters.

CONCLUSION

On the day of competition we were able to achieve first place in the competition with a radar that locate a target at 91 meters with 0.43% accuracy. The radar weighed 400 grams, and consumed 1.5 Watts. This project really helped develop our knowledge on RF and embedded systems. None of the members of our group had any RF experience beyond basic electromagnetics, and this design project greatly broadened our understanding of RF systems. Our goals all along was to keep things as simple as possible, and even though initially it took a long time to get a working system; through hard work and perseverance we were able to design a system that exceeded our expectations, and that we are immensely proud of.

Figure #22: Day of Competition (4/15/16), (from left to right) Jesus Beltran, Jimmy Hua, & Joe Cooney

APPENDIX

References:

KiCAD PCB Tutorial:
https://github.com/ucdart/UCD-EEC134/blob/master/labs/appendices/kicad/pcb-tutorial.pdf
Lab 6 Manual:
https://github.com/ucdart/UCD-EEC134/blob/master/labs/lab6/EEC134_Lab_Manual_6.pdf

MATLAB Code:

```
% Process Range vs. Time Intensity (RTI) plot
% Original Credit to Gregory L. Charvat
% Edited and added to by Joe D. Cooney (Winter/Spring 2016) v5 (4/15/2016)

clc;
clear all;
close all;


[Y,FS] = audioread('4_12_2016/240ft.wav');% Main Signal

%constants
c = 3E8; %(m/s) speed of light

%radar parameters
Tp = 20E-3; %(s) pulse time
N = Tp*FS; %# of samples per pulse
fstart = 2260E6; %(Hz) LFM start frequency
fstop = 2590E6; %(Hz) LFM stop frequency
BW = fstop-fstart; %(Hz) transmit bandwidth
f = linspace(fstart, fstop, N/2); %instantaneous transmit frequency

%range resolution
rr = c/(2*BW);
max_range = rr*N/2;

%the input appears to be inverted
trig = -1*Y(:,1);
s = -1*Y(:,2);

clear Y;

count = 0;
thresh = 0;
start = (trig > 3*thresh);
```

```
for ii = 100:(size(start,1)-N)
      if start(ii) == 1 & mean(start(ii-11:ii-1)) == 0
         count = count + 1;
      sif(count,:) = s(ii:ii+N-1);
      time(count) = ii*1/FS;
      end
end

%subtract the average
ave = mean(sif,1);
for ii = 1:size(sif,1);
      sif(ii,:) = sif(ii,:) - ave;
end

zpad = 8*N/2;
R = linspace(0,max_range,zpad)*0.71; % scaling factor of 0.71
%RTI plot
figure(10);
subplot(2,1,1);
v = dbv(ifft(sif,zpad,2));
S = v(:,1:size(v,2)/2);
m = max(max(v));

K1_x = imagesc(R,time,S-m,[-80, 0]);
%colorbar;
ylabel('Time (s)');
xlabel('Range (m)');
title('RTI without clutter rejection');
datacursormode on; hold off;
K1 = copy(K1_x);

scaling = R(end)/size(K1.CData,2);  % scaling factor

subplot(2,1,2);
plot(scaling*(0:length(max(K1_x.CData))-1),(mean(K1_x.CData)))
xlabel('Range (m)');
ylabel('Intensity');
title('Intensity vs. Range');
datacursormode on;
axis([0,140,-50,-10]);  % This can be changed depending on your
                        % data... will be specific to radar

distance = input('Estimated Position in Meters?: ');
fprintf('\n\n Guess is at %0.2f meters, %0.2f feet\n\n', ...
            distance,distance*3.28084);

ii = round(distance/scaling);
```

```
      % Creates a buffer zone around first object occurence (50 is arbitrary)
H1 = ii - 50;
H2 = ii + 50;
      % Finds minimum of data
K1Min = min(min(K1.CData));

      % Sets all data not within the buffer zone to the minimum value
K1.CData(:,1:H1) = K1Min;
K1.CData(:,H2:end) = K1Min;

      % Finds maximum of new matrix with limited columns containing valued
      % information
[K1Max idx] = max(max(K1.CData));
[xMax yMax] = ind2sub(size(K1.CData),idx);

d = 2;
% Threshold Check
[a,b] = size(K1.CData);
L = a*b;

      % Again cuts out values lower than the maximum
loc = K1.CData < K1Max - d*5;
      K1.CData(loc) = K1Min;

for ii = 1:size(K1.CData,2)-1
      K1.CData(:,ii) = mean(K1.CData(:,ii));
end

[K1Max idx] = max(max(K1.CData));

K1.CData(:,idx) = -20;

Location = idx*scaling;
Feet = 3.28084*Location;


fprintf('\n\nThe object seems to be located at %0.3f meters', ...
      ' or %0.3f feet \n\n',Location,Feet);

Guez = 1/0.9756*(Feet - 1.8529);    % linear adjustment (tries to
GuezM = round(Guez/3.28084);        % account for nonlinearity)
Guez = round(Guez);                 % experimentally determined

fprintf('\n\nMost Likely Value: %d Feet (approx. %d Meters)\n',Guez,GuezM);

      % Plots final result
figure(30);
imagesc(K1.XData,K1.YData,K1.CData,[-80,0]); colorbar;
```

```
ylabel('time (s)');
xlabel('range (m)');
title('Filtered Data');
datacursormode on;

figure(40);
subplot(2,1,1);
imagesc(R,time,S-m,[-80, 0]);
colorbar;
ylabel('time (s)');
xlabel('range (m)');
title('RTI without clutter rejection');
datacursormode on; hold off;

subplot(2,1,2);
imagesc(K1.XData,K1.YData,K1.CData,[-80,0]); colorbar;
ylabel('time (s)');
xlabel('range (m)');
title('Filtered Data');
datacursormode on;
```