Proximity Detector via Frequency Modulated Continuous Wave (FMCW) Radar

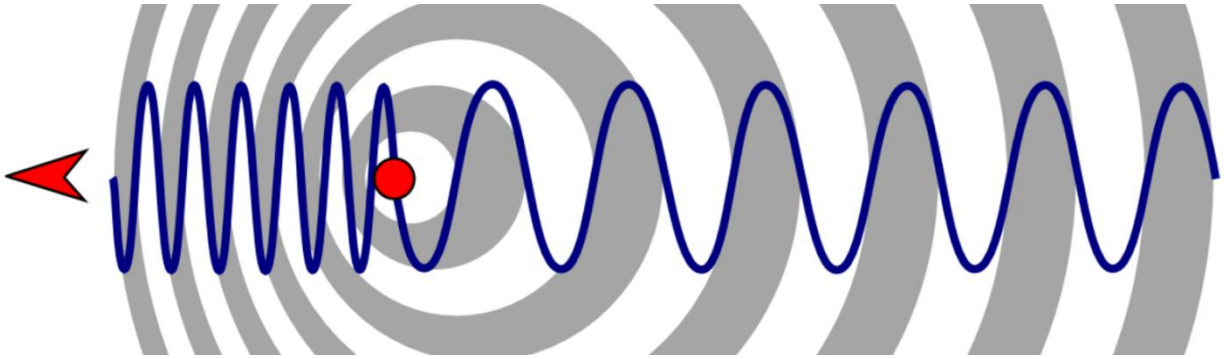
Group: Mansoor Wahab, David Rangel, Daniel Kuzmenko, Michael Moon
danielkuzmenko@gmail.com, drangelalarcon@gmail.com
mansoorw1992@hotmail.com, mfmoon1@gmail.com

Abstract

In this project a frequency modulated continuous wave (FMCW) radar was designed and built to be used with Arduino microcontrollers. FMCW is a radar system where a frequency modulated signal is mixed with an echo from a target to produce a beat signal. The operating frequency is chosen to be at 2.5GHz with the modulation bandwidth of 700 MHz starting at 2.1-2.8GHz. The design shield gets sandwiched between an Arduino Uno on the top and the Arduino Due on the bottom. The Arduino Uno is used to control an oscillator which modulates the 2.5GHz tone and the Due is used to sample the mixer output signal (after LPF), which has a frequency dependent of the distance of the object, and then send the data to the computer via Bluetooth. This radar was designed to be used as a proximity detector but because of the frequency modulation nature it can be used also to measure speed.

Introduction:

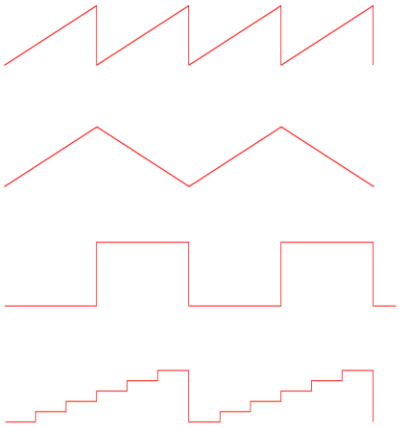
Frequency modulated continuous wave (FMCW) radar is variant of a continuous-wave radar. Continuous-wave (CW) radar is a type of radar system where a known stable frequency continuous-wave energy is transmitted and then received from any reflecting objects. CW radar uses Doppler Effect to measure speed. Return frequencies are shifted away from the transmitted frequency based on the Doppler Effect when objects are moving. There is no way to evaluate distance so frequency modulation is added to the outgoing wave.



(Change of frequency caused by the motion of the target)

In the FMCW system, the transmitted signal of a known stable frequency continuous wave which varies up and down in frequency over a fixed period of time by a modulating signal. Frequency difference between the received signal and transmitted signal increases with delay,

and hence with distance. There are many different modulation patterns which can be used in FMCW radar.



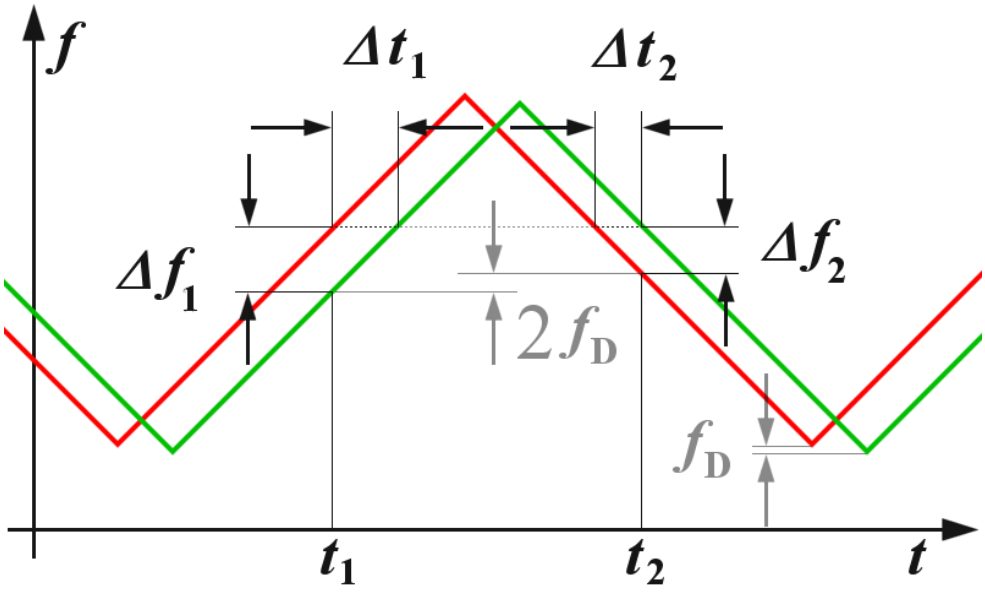
Sawtooth

Triangle

Square-wave

Stepped modulation

For this design the modulation pattern that was chosen was a combination of the triangle and the stepped modulation. This modulation pattern allows for easy separation of the delta frequencies on the rising and the falling edge so distance and speed could be extracted at the same time.



In the figure above an echo signal is shifted due to the running time compared to the transmission signal to the right (green is the echo and red is the transmission signal). Without a Doppler frequency, the amount of the frequency difference during the rising edge is equal to the measurement during the falling edge. A Doppler frequency shifts the echo signal in height (green graph in the figure). It appears the sum of the frequency difference Δf and the Doppler frequency f_D at the rising edge, and the difference between these two frequencies at the falling edge. This opens up the possibility of making an accurate distance determination, despite the frequency shift caused by the Doppler frequency, which then consists of the arithmetic average

of the two parts of measurements at different edges of the triangular pattern. Knowing the Δf and the BW of the system, one can simply calculate the distance using a formula shown below.

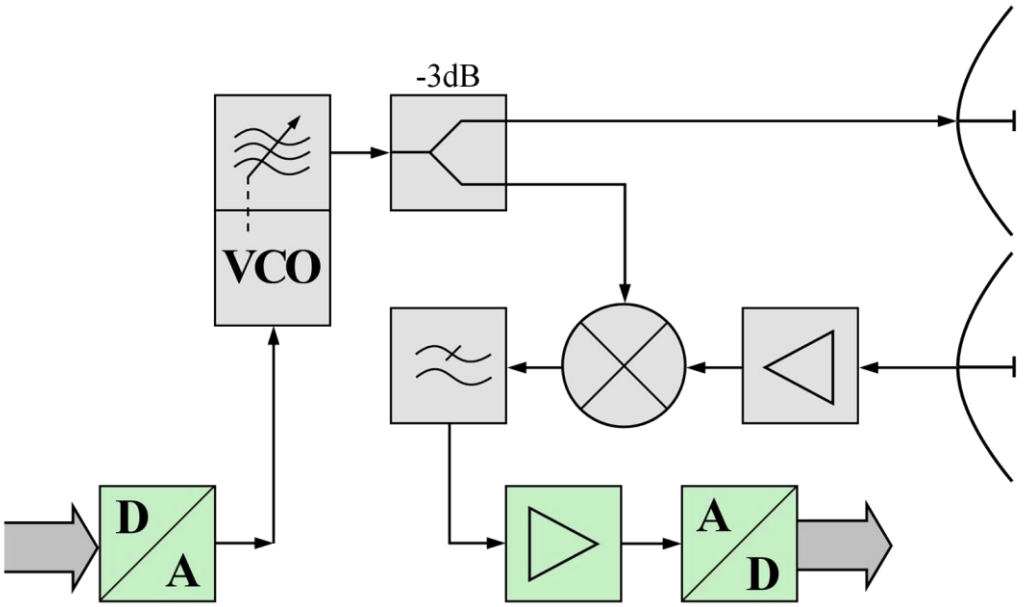
$$Distance = \frac{c \times T_{modulation} \times \Delta f}{BW \times 2}$$

Where T-modulation is the half period of the modulation triangle, Δf is as described above, and the BW (bandwidth) is how much the signal is modulated around 2.5GHz.

At the same time the accurate Doppler frequency can be determined from two measurements. The difference between the two difference frequencies is twice the Doppler frequency. This leads to the calculation of speed as follows

$$Speed = \frac{f_{D*}c}{2 * f_o}$$

Where f_D is the Doppler shift and fo is the starting frequency. A block diagram of FMCW radar is shown below.



A voltage controlled oscillator (VCO) is used to modulate the RF signal around 2.5GHz. This signal is then amplified and sent to a power splitter where the signal is split into two. The first is used as the transmission signal and the other as a local oscillator for a mixer. Once the echo signal returns to the receiver, it passes through a low noise amplifier and then gets mixed down to an intermediate frequency. This IF is then passed through a low pass filter to get rid of the entire high order spurious signal. At this point the signal is so small that it needs to be amplified so it passes through a baseband amplifier. Next, the signal gets digitized and is processed to extract the Δf and the f_D. The DSP can be performed on the computer since sound cards have

high-sampling rate ADCs. However since the computer is not portable, it is practical to use microcontrollers and FPGAs.

Description of the Project:

Our project is a FMCW radar shield for the Arduino Due able to precisely detect an object's distance up to 15 meters, updating the distance measurement twice a second. The shield is a compact PCB with the analog system on it, using surface mounted and a few through-hole components. The system is fairly self contained and can send data to a laptop or cell phone using a Bluetooth connection.

Design details:

The RF/Analog portion is based from a typical FMCW radar schematic which we prototyped in the first ten weeks. The prototype operates at a frequency band of 2.3 to 2.7 GHz, uses the Arduino Uno to control the A/D and create the triangle wave, uses the computer sound card to sample the IF signal, and finally processes the information in Matlab.

The new system operates at 2.1 to 2.8 GHz, uses the Arduino Uno still to control the A/D, and utilizes the Arduino Due and a D/A for sampling the IF signal. The A/D has a 16 bit resolution to ensure that even the smallest signal gets digitized. Sampled data is stored in the Due and is later transmitted over the Bluetooth chip to a laptop to finish the signal processing. The FFT can also be done on the Arduino Due, though for presentable results the laptop processes the data and displays the information. The Due and Computer must work in tandem to sample, transmit, take the fft and then display the data. This is done by code in Matlab that first queries the Due to send the data over. The Due then transmit previously sampled data for Matlab to take the fft. Once the fft is calculated, Matlab takes the magnitude of the data and plots a range-time intensity plot. This plot also has data from the previously transmitted samples so that one could see how things have changed. While Matlab is busy calculating the fft and plotting data, the Due code is sampling new data so that when Matlab queries for data the Due will already have the sampled data ready to transmit with no delay.

To create this schematic on an Arduino Due shield, we found surface mounted component versions of these blocks from Minicircuits. In order for the components to work together, we needed to choose the components to match our desired frequency band of 2.1 to 2.8 GHz, operate on 5V DC power supply, match to 50 Ohm impedance, and deliver the right amount of power to one another.

The IC components used are as follows:

- **VCO: JTOS-3000** - This voltage controlled oscillator is rated for 2.3 to 3 GHz controlled with a voltage between .5 and 12 Volts. The outputted signal power is about

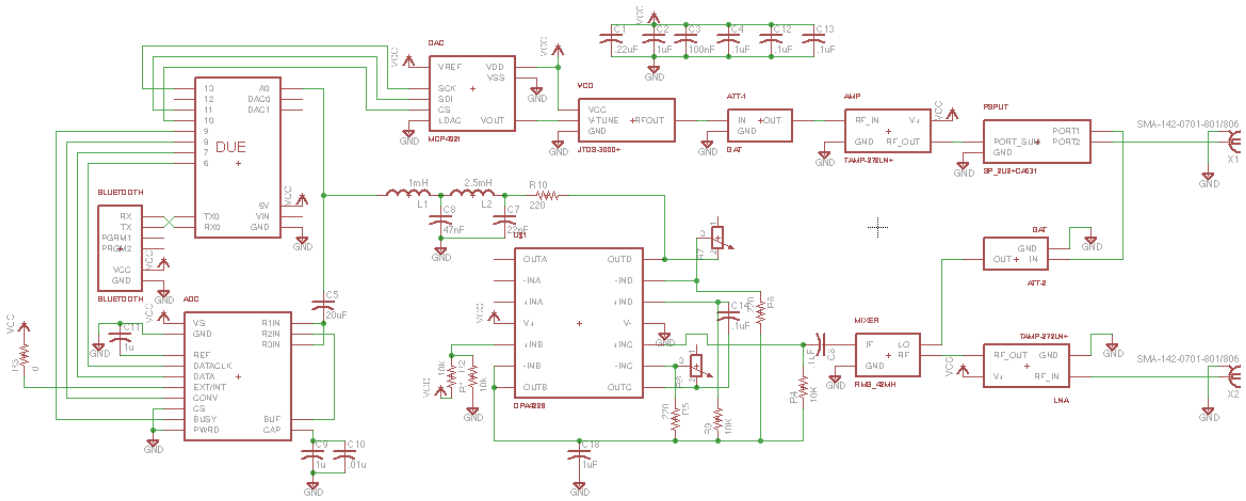
10dBm. For our project we are controlling it from .69 to 5 Volts and getting a range of roughly 2.1 to 2.8 GHz.

- **Power AMP/Low Noise AMP: TAMP-272LN+** - Our low noise amplifier, which we are also using as a power amplifier, gives around 14dB of gain within the 2.3 to 2.7 GHz band. It has a small noise figure of .85dB, a one dB compression point of 19.5dBm, and an output IP3 of 30 dBm.
- **Mixer: RMS-42MH+** - This is a double balanced mixer able to take a wide band of .8 to 4.2 GHz. It is passive, and requires a LO signal of 13dBm in order to create the mixing products, which usually has about 5.3 dB conversion loss.
- **Power Splitter: SP-2U2+** - This power splitter operates at 1720 to 2850 MHz, creates an even power split and has a 50 ohm match at each port. It has an insertion loss of about .7 dB from each port.
- **Attenuator: GAT-5+** - These are small 50 ohm matched attenuators which add 5 dB of attenuation between the power amp and the mixer to avoid overpowering the mixer, and between the VCO and the power amp to avoid pushing the power amp to the 1dB compression point.
- **Op Amps: OPA4228** - This op amp IC has 4 different op amps in it that have low noise of 3nV/Hz^.5, and a high open loop gain of 160dB. These op amps are used on the board to create a two stage IF amplifier, and another is used to create a virtual ground for that amplifier at (VCC+Ground)/2
- **ADC: ADS7813** - This is a 16-bit Analog to Digital converter that can sample at 40kHz. For this project we were able to implement it with the Arduino Due to sample at 25kHz, which is more than fast enough to sample our IF signal which is at 5kHz.
- **DAC: MCP4921** - This is a 12-bit Digital to Analog converter with an SPI of 20 MHz. This chip is still controlled by the Arduino Uno in our project, and thus the SPI connection is limited by the Uno's 16MHz clock.

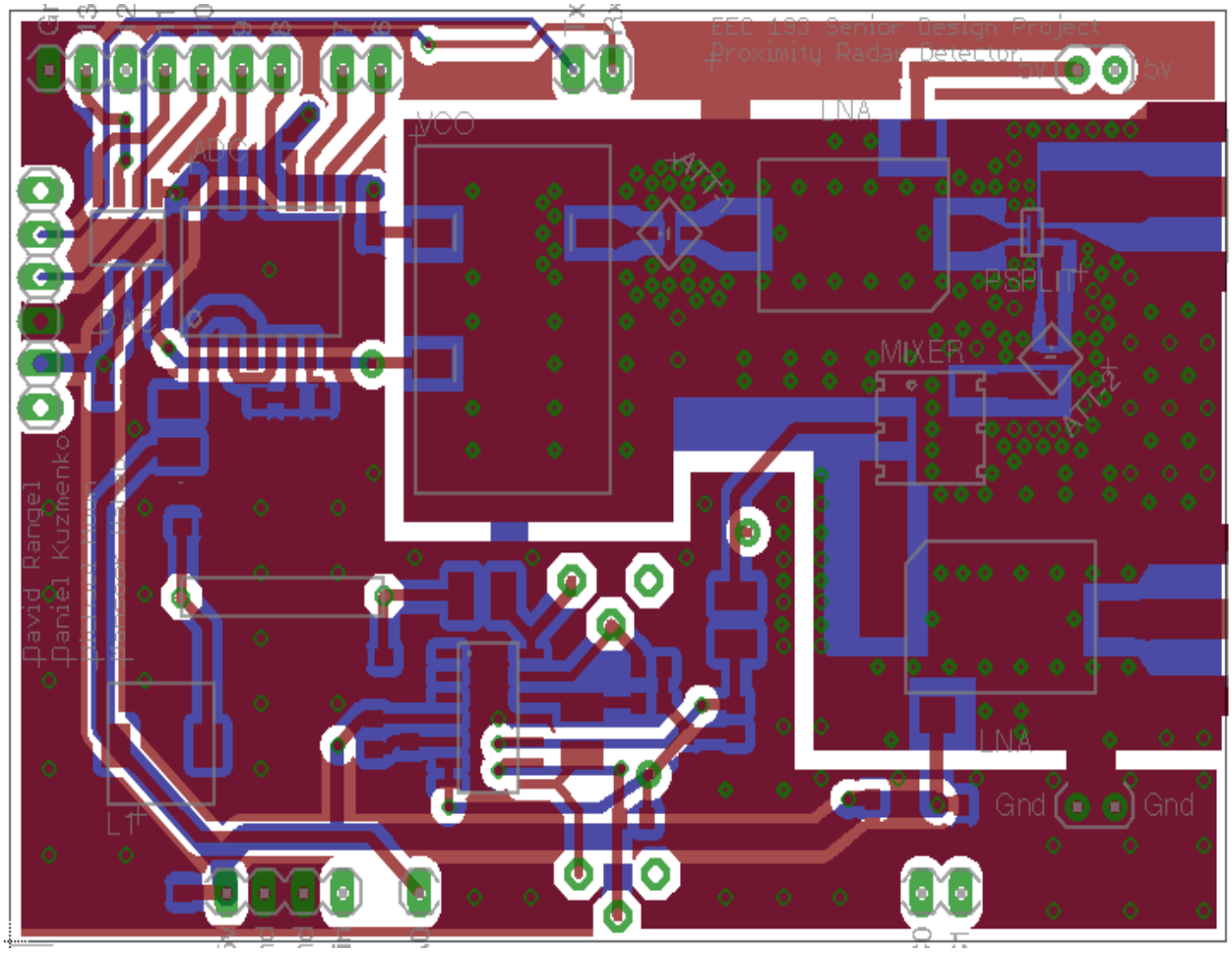
The low pass filter used after the amplification of the IF signal is a passive lumped-component low pass filter. It is designed as a fourth order maximally flat filter, with a 3dB cutoff around 15kHz, with the IF signal coming out below 5kHz.

There are also DC blocks, 22uF capacitors in series with the system to block the DC component coming out of the mixer. This also helps attenuate the 60Hz noise signal coming from the triangle wave.

The following picture is the EAGLE schematic for the entire system. The analog portion has the same signal flow as the block diagram given below. The DAC is being fed from the Arduino Uno (which is the same block as the DUE in the schematic below) and the ADC is feeding to both the Arduino Due and the ADC, which is controlled by the Arduino Due.

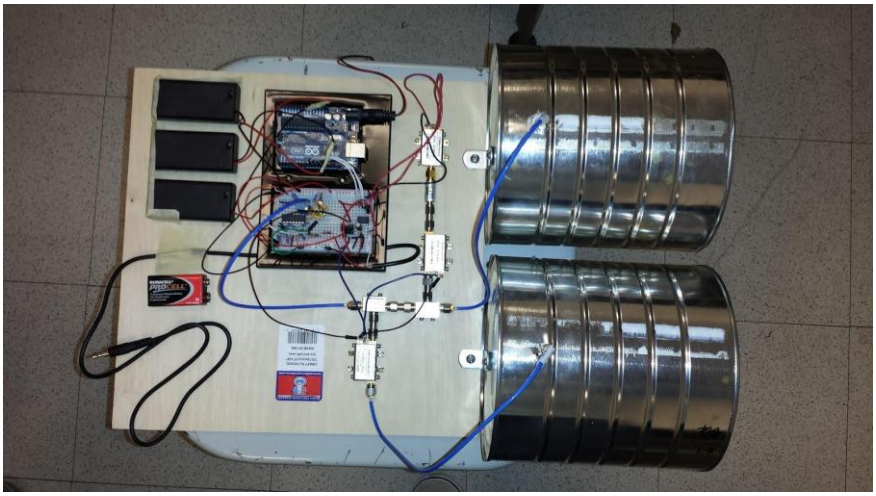


The EAGLE PCB layout is given below. The RF portion is isolated from the low frequency/digital portion by means of separate grounds and multiple vias. The holes on the top and bottom are all there for the Arduino connections, and the left side holes are there for the Bluetooth module connection.

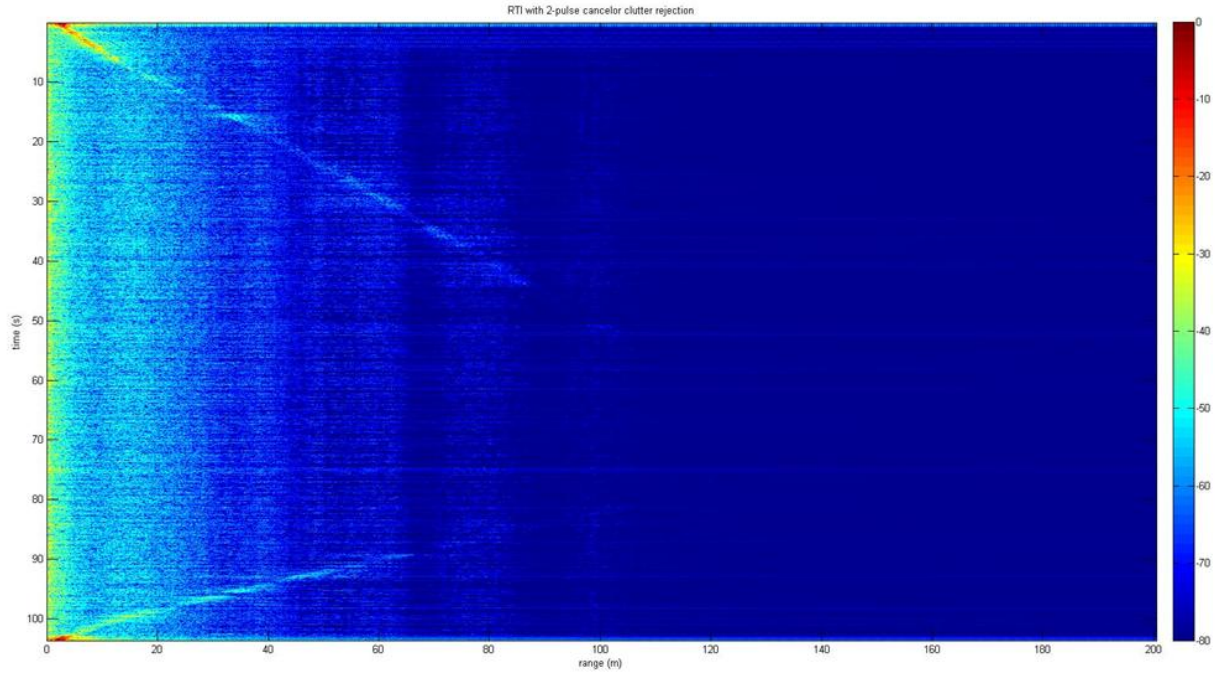


Test/Measurement Results:

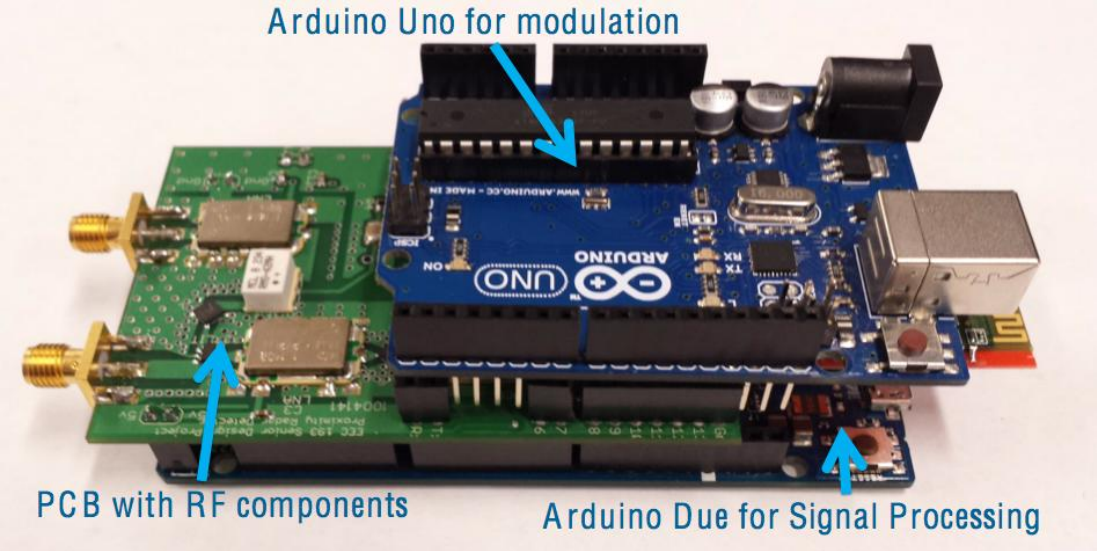
The very first iteration of this radar project was the MIT connectorized prototype which was assembled from scratch. The RF front end was all connectorized Mini-circuits components with the baseband amplifier and the low pass filter built on the breadboard.



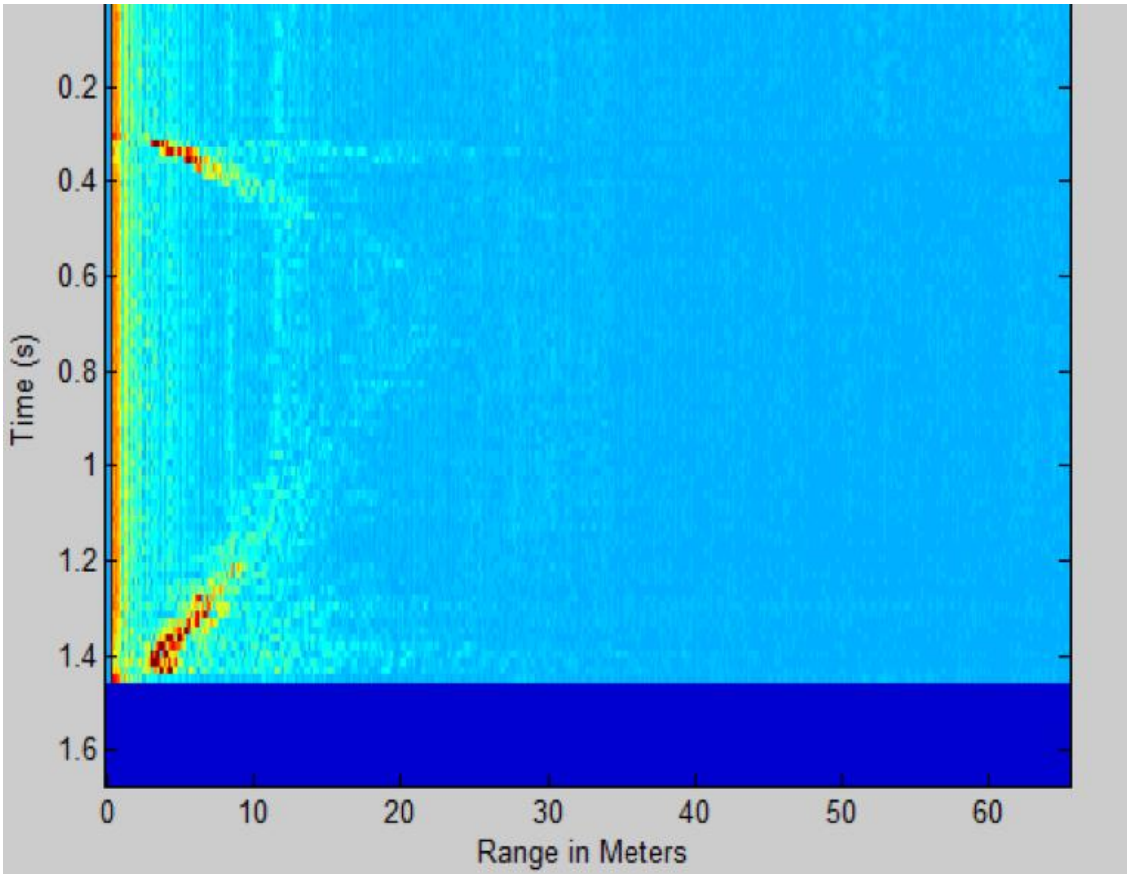
This system was designed for post-processing so the signal needed to be sampled by a computer via an audio card and then ran in Matlab to get a range-time intensity plot. With this system, we were able to get a reading distance out to 90 meters, which can be seen in this range vs. time graph of a moving object:



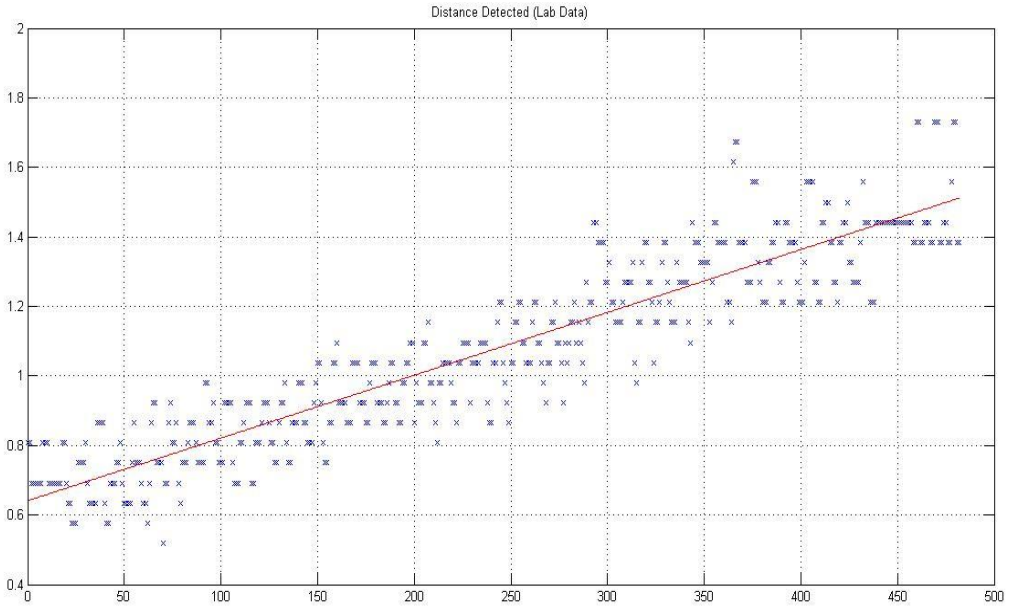
Our project is a much more compact PCB that fits right in between an Arduino Uno and an Arduino Due:



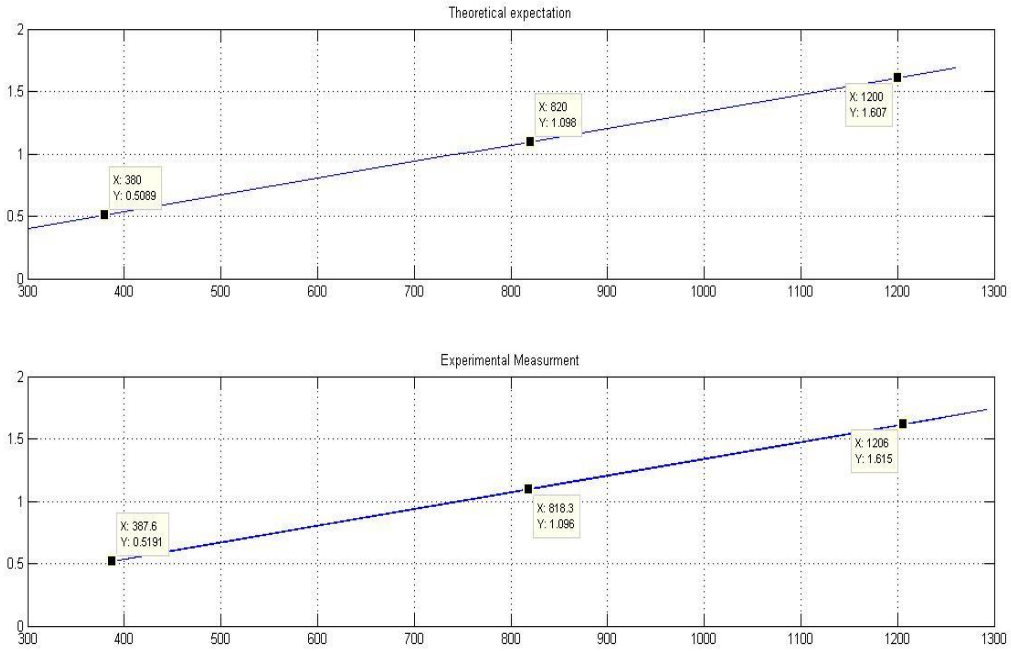
With this system, we can constantly update and provide a stream of data, and provide a distance measurement up to 20 meters, tested on a balcony. We can judge the distance of an object within plus or minus half a meter. The results of the system are given below, showing range versus time of a moving object:



The following plot shows the experimental data (distance) from a 2m walk. We ran a Least Squares regression to fit a curve to the data. This is a vital step since realistic results have some degree of error between successive samples.



The following shows the theoretical and experimental relationships between distance and frequency. We see they are very close to each other.



For estimation of the tone frequency, we relied primarily on two algorithms: Frequency estimator proposed by Rife and Vincent and simple High-Peak Detection. We found High-Peak Detection to give more accurate results. The following table proves this.

Theoretical Frequency	High-Peak Approximation	Frequency Estimator	Theoretical Distance	High-Peak Approximation	Distance Estimator
60	43.066	105.67	0.080357	0.057678	0.14152
120	129.2	181.56	0.16071	0.17303	0.24316
180	172.27	230.64	0.24107	0.23071	0.3089
240	258.4	310.04	0.32143	0.34607	0.41523
300	301.46	357.43	0.40179	0.40375	0.47871
360	344.53	405.89	0.48214	0.46143	0.5436
420	430.66	482.67	0.5625	0.57678	0.64643
480	473.73	532.51	0.64286	0.63446	0.71318
540	559.86	581.14	0.72321	0.74982	0.77832
600	602.93	656.78	0.80357	0.8075	0.87962
660	646	707.8	0.88393	0.86517	0.94795
720	732.13	782.73	0.96429	0.98053	1.0483
780	775.2	832.2	1.0446	1.0382	1.1146
840	818.26	911.73	1.125	1.0959	1.2211
900	904.39	956.7	1.2054	1.2112	1.2813
960	947.46	1007.2	1.2857	1.2689	1.349
1020	1033.6	1084.8	1.3661	1.3843	1.4528
1080	1076.7	1133.6	1.4464	1.442	1.5182
1140	1119.7	1182.7	1.5268	1.4996	1.5839
1200	1205.9	1259.9	1.6071	1.615	1.6873
1260	1248.9	1309.8	1.6875	1.6727	1.7542
1320	1335.1	1385.1	1.7679	1.788	1.855
1380	1378.1	1434.9	1.8482	1.8457	1.9218
1440	1421.2	1485.1	1.9286	1.9034	1.989
1500	1507.3	1561.4	2.0089	2.0187	2.0911
1560	1550.4	1610.6	2.0893	2.0764	2.157
1620	1636.5	1687.2	2.1696	2.1918	2.2597
1680	1679.6	1734.9	2.25	2.2495	2.3235
1740	1722.7	1785.3	2.3304	2.3071	2.3911
1800	1808.8	1863.1	2.4107	2.4225	2.4952
1860	1851.9	1910.9	2.4911	2.4802	2.5593
1920	1938	1987.4	2.5714	2.5955	2.6617
1980	1981.1	2036.8	2.6518	2.6532	2.7279
2040	2024.1	2086.6	2.7321	2.7109	2.7946
2100	2110.3	2163	2.8125	2.8262	2.8968
2160	2153.3	2211.2	2.8929	2.8839	2.9615

We can see that for high frequencies, the frequency estimator algorithms improves, but still does not deliver results any close to high-peak estimation. For low frequency, the frequency estimator is egregious.

Discussion:

Our results show some decent precision in the 0 - 20 meter range, meeting our set goals from the beginning. In this range it is easy to follow the path someone takes away and towards the antennas. The system also updates twice per second, meeting our goal of frequent distance updates.

The biggest limiting factor of the system is currently noise. As the signal goes further and further out, the IF signal quickly diminishes into the noise floor, limiting how far out we can see. The signals being transmitted are too high frequency and too high power for the FR-4 material, and are leaking out into the rest of the board. Another source of noise is our use of attenuators after generating the FM signal.

Another source of noise is the 60 Hz triangle wave somehow appearing on the IF signal. This can actually be seen on the measured range vs. time plot in the results as a line going from the top to the bottom of the graph. The solid red line going from the top left to the bottom left side of the graph seems to be a near DC constant noise similar to the 60 Hz noise. Fortunately, these are constant and easily ignorable when looking at a moving target.

Possibility for future extension:

To improve the RF system, the noise should be diminished. In order to do so, following the discussion of the noise source, a 4 layer board with a better dielectric could be used, to better contain the RF signals. This would bring the ground plane closer to the RF traces, increase the capacitance, decrease the size of the RF traces, and increase the isolation between the traces.

Changing the schematic to remove the attenuators would also help eliminate some noise. If a different mixer were used with a smaller LO power requirement, possibly an active mixer, then the power amplification could occur after the splitter, and thus all the amplified power would go to transmitting the signal. This would also help in reducing the amount of high power RF noise on the board.

Even though the refresh rate is twice a second, this is still not a real time system. After some timing analysis it was found that it takes the system 30 ms to acquire 700 samples and 400 ms to transfer the data to a computer. Once on the computer, it takes .3 ms to take an fft of the data and then 5ms to create the plot and all the other necessary calculations. From this timing analysis it can be seen that the bottleneck in this system is the data transfer rate. A possible way to improve timing is to use a different communication method. A serial link sends one bit at a time. What if there was a parallel link? This way multiple bits can be sent over in parallel which will speed up the process significantly. Another possible solution is compression of the transmit data. If data can be compressed to say 50% of its original size that would almost cut the data rate in half. If more compression can be achieved, then the data transfer rate will decrease.

Although this system has the fft taken on the computer, it would be desirable to have the system itself do the signal processing. The Arduino Due is somewhat limited on this factor because of its speed and the fact that it does not have a floating point unit. It has been shown that an fft can be done on the Due, but at a rate of 150ms per 1024 points. An alternative would be to redevelop the system around a much more powerful signal processing device. FPGA would be a wonderful choice because of how fast it can calculate an fft. There are also some new systems coming to the market that have GPUs in microcontrollers that can also calculate ffts with amazing speeds (nvidia jetson tk1).

Suggestions for this class:

- More Pizza.
- There should be a prize for the team that wins the competition.
- The quarter one labs can be done in a shorter time span, to start designing and building a system earlier. This would especially be helpful if the class needs to be fit within 2 quarters.
- Encourage ECE students specializing in Digital Electronics to join the project/class. It is one thing to have a working signal processing algorithm in Matlab and another to actually implement it on a processor efficiently.

Conclusion:

The radar works! We must say it meets the expectation given we were on a student budget. As with any engineering system, ours has areas that need to be improved and further developed. Every member of our team has learned valuable lessons.

Acknowledgements:

MIT radar project
TI Website
Minicircuits Sales Department