Senior 2 Project: Group 2B

Speed Radar Gun

Spring 2017

Antonio Rodriguez, Project Leader
College of Electrical and Computer Engineering
Oklahoma State University
Stillwater, Oklahoma
antor@okstate.edu

James Touthang
College of Electrical and Computer Engineering
Oklahoma State University
Stillwater, Oklahoma
james.touthang@okstate.edu

Jingkun Liang
College of Electrical and Computer Engineering
Oklahoma State University
Stillwater, Oklahoma
jingkun.liang@okstate.edu

Rachel Campbell
College of Electrical and Computer Engineering
Oklahoma State University
Stillwater, Oklahoma
rachel.leigh.campbell@okstate.edu

Abstract*—Our purpose was to create a Doppler radar gun within one semester using a budget of $250. Our radar gun can be broken down into five subsystems, specifically the antenna and transceiver, amplifier and filter circuit, microcontroller, power, and outer casing subsystems. We chose the MACOM transceiver joined to a coated horn antenna to send and receive signals and produce our intermediate or Doppler frequency. Our amplifier and filter circuit used the MAX414CPD operational amplifier for its high performance and low noise. Our bandpass filter has ideal cut off frequencies at 700 Hz and 7300 Hz, which allows us to accurately measure speeds between 10 and 100 mph. The actual cut off frequencies of out filter circuit were 884 Hz and 10.6 kHz, to compensate for a more gradual cutoff slope. The total gain of our amplifier and filter circuit is approximately 7700. We chose the Arduino as the controller for our radar gun and used the FHT library to determine the true Doppler frequency from a given intermediate frequency. The Arduino also computes the speed from the Doppler frequency and displays the speed in mph and Km/hr on the LCD. The power for the radar gun is supplied by two 9V batteries which allow for over 2 hours of continuous runtime. Finally, the outer casing was created using Solidworks modeling and 3d printing. Our testing of our completed Doppler radar gun showed that the gun can measure ball speeds from 10 - 46 mph (this is not the limit of the radar gun's capabilities, only the limit of our throwing speed) at a range of 0.5 - 50 feet with an accuracy of approximately plus or minus 1 mph. Our radar gun meets all required specifications and performs admirably in real world conditions.

***Keywords*—Doppler; radar gun; Capstone; Group 2B; Oklahoma State University; ECEN Department;**

I. Introduction

Our goal was to create a Doppler radar gun with a professional, handheld appearance within the timeframe of one semester and the budget of $250. A Doppler radar gun works under the premise of transmitting a signal at a moving object, receiving a signal back from that moving object and mixing those two signals together to get an intermediate or Doppler frequency. This Doppler frequency can then be used to calculate the speed of the moving object. Doppler radar guns are frequently used by the police force to measure the speed of vehicles and by sports enthusiasts to measure the speed of balls. Our radar gun will be used to measure the speed of tennis balls and will most likely by used in sporting settings.

II. Design Problems and Objectives

A. *Theoretical Foundation*

The Doppler effect is the increase in frequency of waves as the observer and source move towards each other. The Doppler effect can be observed in the changing of pitch in police sirens as a police car approaches and drives away from a stationary observer. Using a transceiver and microcontroller, the Doppler effect can be used to determine the speed of passing objects. To do so the transceiver sends out a radio frequency of fixed bandwidth in a directed fashion, often using an antenna of some kind. This frequency will hit all objects in its path. These objects which are struck by the transmitted frequency will produce a return frequency which is picked up by the transceiver. If the object is stationary the difference between the transmitted and returned frequencies will be zero. However, if the object is in motion the difference between the transmitted and returned frequencies will be proportional to the speed at which the object is traveling and be termed the intermediate or Doppler frequency. The Doppler frequency relates to the speed of the object in the following manner:

$$Fd \;=\; 2V\left(\frac{Ft}{c}\right)cos\theta$$
$$Formula\ 1.$$

Where Fd is the Doppler or intermediate frequency, Ft is the transmitted frequency, V is the velocity of the object, and $\theta$is the angle between the object's path of motion and the transceiver's position. Using the Doppler effect to measure speed is most effective when the measurement device and object's motion occur parallel to each other, with the object moving either directly toward or directly away from the transceiver.

B. Design Problem

We were tasked with creating a Doppler radar gun with the ability to measure the speed of tennis or baseballs. This radar gun needed to be able to accurately measure a broad range of speeds from relatively large distances away. This gun also needed to be handheld and have a professional appearance.

C. Specifications

The Doppler radar gun we created needed to measure speeds from a minimum of 10 mph to a maximum of 100 mph. It needed to have an operational range from 0.5 feet to 50 feet with an accuracy of plus or minus 1 mph at or below 25 feet and an accuracy of plus or minus 3 mph at 50 feet. The power supply of this radar gun needed to be sufficient for at least 2 hours of continuous runtime. Finally, in order for the radar gun to be considered handheld it needed to be less than 12 inches long, 9 inches high and 6 inches wide and have a weight less than 3 pounds.

D. Subsystems and Team Responsibilities

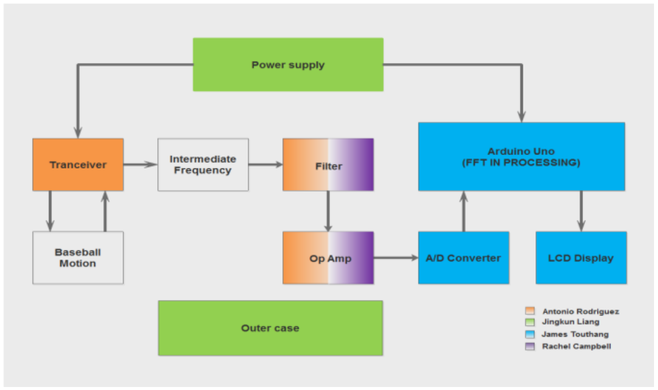


Figure 1. The block diagram

The figure above shows the layout of the radar gun and the responsibilities of each team member. For a larger version of the block diagram please refer to Appendix E. James worked on the microcontroller, ADC, and LCD display. In other words, he was responsible for converting the signal processed by the filter and amplifier circuit to a digital signal, getting the Doppler frequency the transceiver received through Fourier transform, storing it to the microcontroller, calculating the speed, and display the speed on the LCD display.

Antonio was responsible for the hardware including the transceiver, antenna, and filter/amplification circuit. Rachel helped Antonio with the filter design and the amplification circuit. Their main objective was to design the filter and OP amp circuit which connected to the output of the transceiver. As soon as the transceiver sends the signal and gets the return or intermediate frequency, the filter circuit filters out most of the noise to improve the accuracy of the measurement, and the OP amp amplifies the signal in order to be readable by the microcontroller.

Jingkun Liang worked on the power subsystem and the case design. He designed the case using Solidworks and printed it with a 3D printer. After the printing, he need to modified the detail over and over again to fit the size for every parts. For the power subsystem, Jingkun designed a circuit which can produce a stable voltage that can satisfy power needs.

III. Detailed Design Documentation

A. Transceiver

The transceiver is the heart of the radar gun. The MACOM MA-7801-17 was implemented into the design. The transceiver operates at 24.1 GHz, which fall into the K-Band radio spectrum. It accepts a 5-Volt input and outputs the intermediate or Doppler frequency in millivolts.

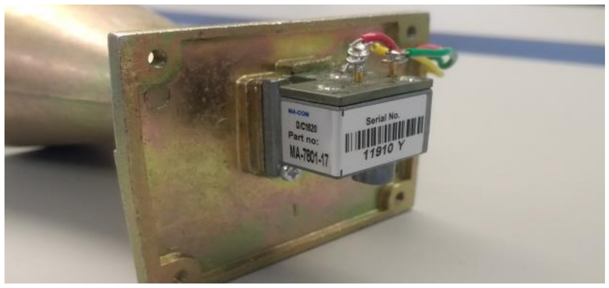


Figure 2. Transceiver module

It operates using a gunn diode and a schottky diode in the form of a mixer circuit.

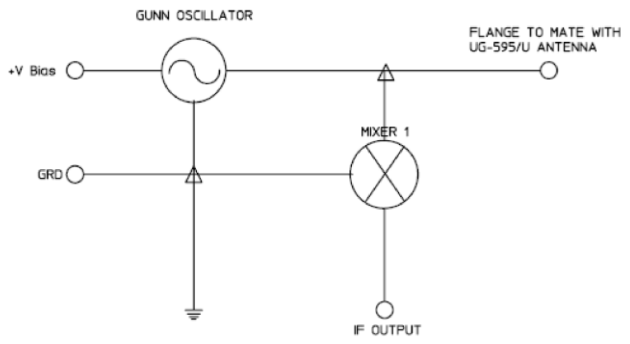


Figure 3. Block Diagram of the Transceiver

Since the transceiver we used was cannibalized from a Bushnell radar gun, it came with a horn antenna that was designed to work with the transceiver.

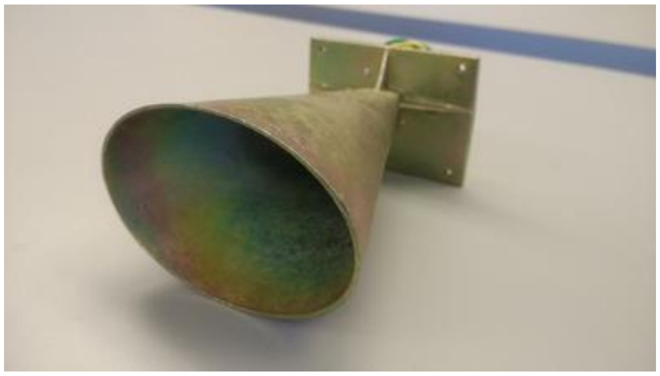


Figure 4. Horn Antenna

B. Filter/Amplifier Circuit

An active band pass filter was designed to filter out undesired frequencies and noise. A practical gain was chosen for the filter stage to see the intermediate frequency on the oscilloscope. Please refer to Appendix A. for the Amplifier/Filter circuit.

The component values were calculated using the frequent cutoff equation:

$$f_{3dB} = \frac{1}{2 * \pi * R * C}$$

Formula 2.

R denotes the resistor component and R denotes the capacitor component. π is a mathematical constant of 3.14159.

For the high pass filter, a cutoff of 884 Hz has chosen.

$$\text{High Pass Filter cutoff: 884 Hz} = \frac{1}{2*\pi*0.1uF*1800\Omega}$$

Formula 3.

For the low pass filter, a cut of 10.6 kHz has chosen.

$$\text{Low Pass Filter cut off: 10.6 kHz} = \frac{1}{2*\pi*100pF*150K\Omega}$$

Formula 4.

The theoretical frequency response of the final design can be found in Appendix B.

The gain on the Filter stage is:

$$\text{Gain: } \frac{Rf}{Ri} = \frac{150K}{1.8K} = 83.3\,\frac{V}{V}$$

Formula 5.

Rf denotes the feedback resistance of the filter and Ri stands for the input resistance of the filter.

Because of the limited possible combinations of capacitors and resistors, these cutoffs were chosen in order to set a gain high enough to amplify the signal without saturating on the oscilloscope. For the amplifier stage, an inverting amplifier was configured with a gain of 83V/V. The feedback resistance was set with a potentiometer for quick adjustments.

A 2.5 volt DC offset was incorporated into both stages via the positive inputs of the amplifiers. This was done in order to avoid amplifying only the positive half of the signal since the transceivers output was centered at 0 Volts.

C. Spice Software

OrCAD Capture was used to layout the circuit in Spice. This program allowed quick prototype testing and to observe the frequencies that our filter circuit could hypothetically operate.

D. Circuit Hardware

Maxim MAX414 amplifiers were chosen because of their low noise characteristics and high gain bandwidth product.

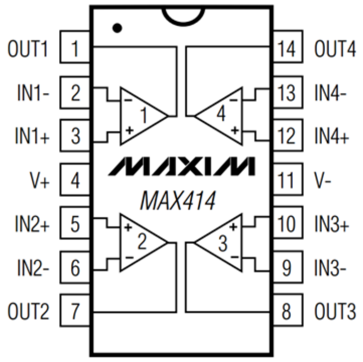


Figure 5. Layout of the MAX414 28 MHz Precision Amplifier [2]

Passive resistor and capacitor components from OSU issued kits were used for this project to reduce the overall costs.

E. Arduino C Programming

Processing the amplified signal was done through C language and used the FHT library from Open Music Labs[5].The library uses the Fast Hartley Transform that converts the time domain signals to the frequency domain. This is exactly what the Fast Fourier Transform (FFT) does, except the FHT is designed for real data and does not account for complex data. This way, the FHT uses half as much computation and half the memory compared to the FFT Library.

In the Arduino, pin A0 is used to input the signal coming from the amplified transceiver signal. The signals goes through the built in ADC and turns the voltage reading from a voltage reading to a 10-bit signed number. The 10 bit signed number is then turned into a 16 byte signed number where it is put into an allocated array called fht_input[].

The array is then computed with the fht_window() function that increases the frequency resolution. After, the fht_reorder() function is called which reorders the input data. The program then finally calls fht_run() which process the real samples and put it it into the same array. Lastly, fht_mag_log() is called to take the magnitude of the fht_input and stores it into an array called fht_log_out. This function squares the real and imaginary values, sums them, takes square root, and finally the log base 2 so that it is compressed in a logarithmic function using a lookup table. Below is the final calculation:

$$\text{fht\_log\_out[i]} = 16 * log_{\ 2}(\sqrt{real_i^2 + imaginary_i^2})$$

Formula 6.

The output is an 8 byte signed number. That is stored in memory that will be used to find the frequency with the highest magnitude. This is done by iterating through each array in the fht_log_out and stores the number that has a set threshold value. With the radar gun. It is set so that if it stored above the threshold value, then the index of the array would be stored so that the program can compute the frequency based on the bin.

The FHT is set to compute 256 points so the total bin size of the output is 128 (256/2). The frequency is calculated by taking the index of the array, i and multiplying it by the sample rate divided by the 256 samples points. Due to the variation and unexpected computation time of the arduino, the program has a variable called bin_multiplier which is set to 146.05, which means the index array at fht_log_out of 1 would have a frequency of 146.05 Hz, and index of 3 would be 438.15 Hz and so on.

That frequency would then be stored to a global variable called highest_freq which will be used to calculate the speed using the toMilesPerHour() function.

The toMilesPerHour() function passes in the doppler frequency, or the frequency calculated from the fht_log_out iteration. The overall equation can be seen below:

$$\text{Speed in MPH} = \frac{dopplerFrequency * c * 3600}{2 * transmitting\ Frequency * 1609.34}$$

Formula 7.

Transmitting frequency is given by the bushell's transceiver where it operates at 24.125 Ghz multiplied by 2 to because of the sending wave and returning. Lastly *c* is the speed of light in meters per second and the 3600/ 1609.34 is the number used to output the result in miles per hour, where there is 3600 seconds in 1 hour and 1609.34 meters in 1 mile.[1] The equation is provided by the data sheet of the HB100 transceiver. [10].

The program utilizes a button that is connected to pin 9 in the arduino. If the digital read on pin 9 is high, then the button has been released and displays the current highest speed computed by passing in the highest_freq global variable. When the trigger has been pulled, the digital read is low, and the highest_freq is set to 0 so that the program can run the FHT and determine the highest_freq.

Speed is then displayed on a basic LCD 16x2 screen connected to digital 6, 2, and 4 pins. This was done using a backpack that essentially converts the 16 pins of the LCD into 5 pins, where 2 pins are left for VCC and GND. The LCD is attached to the backpack pins where data pin is connected to pin 6, clk pin is connected to pin 2, and latch pin is connected to pin 4.

In summary, the code is constantly running the FHT on the analog signal regardless if the transceiver is on. Once the trigger is pushed, the highest_freq determined by the code is set to 0 so that once the trigger is released, it will display the appropriate speed in both miles per hour and kilometers per second.

F. Casing

For the primary objective, the outer case should accommodate all of the parts in a satisfying volume. But most importantly, it needs a professional appearance to make the user measure the speed more conveniently.

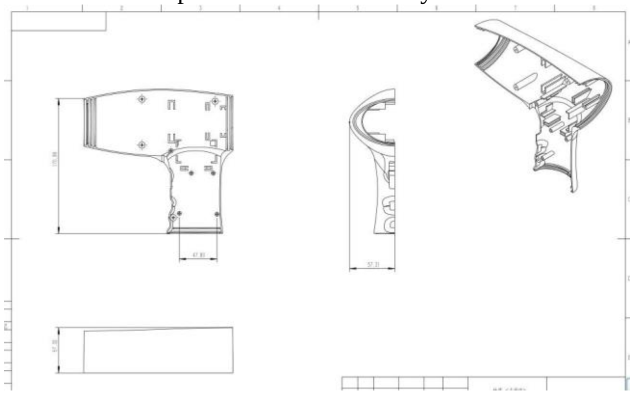


Figure 6. Three-view drawing of the radar gun

The figure above shows all the professional features of the gun. Firstly, at the body of the radar gun, we increase the radius of the case according to the part we need to place so that we can get a 30% volume increase while the outer appearance can be more stylish compared with the boxy look.

Secondly, we designed different kinds of grooves to fit all the parts. Like the figure shown below, Arduino Uno is fixed by screws. At the same time, there are 4 holders to hold the batteries and we remain a space to let the electrodes out of the box so that it will be convenient to link to the circuit. Last, we add a trigger in the speed gun in order to let users use the gun easily and our speed gun can be more like the real gun and have the same usage that users can know how to use it quickly.

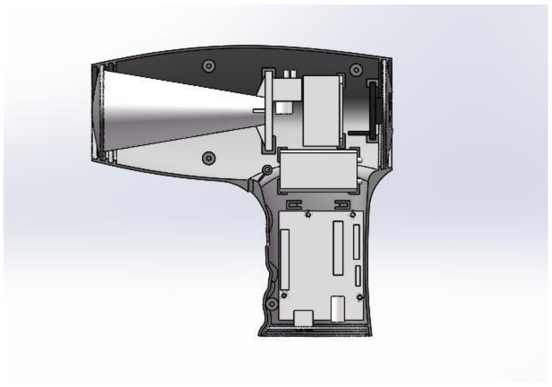


Figure 7. Inner structure

Figure 8. The overall appearance

E. Trigger

The trigger from the cannibalized Bushnell gun was modified to work in our radar. The momentary switch was connected the 5 Volt rail of our circuit and the gates of two 2N2222 NPN transistors. One transistor signaled the Arduino that the trigger was closed. The other transistor powered the transceiver. Configuring the transceiver this way allowed us to conserve battery power and meet our battery life specification. Figure 9. shows how the trigger was implemented.

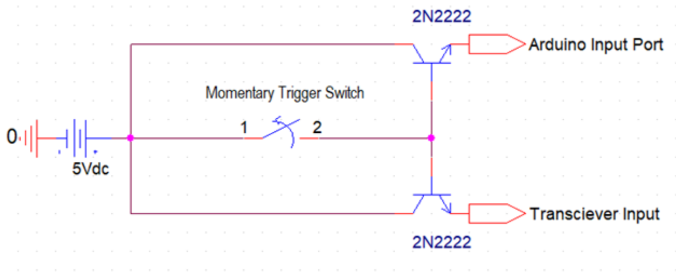


Figure 9. Trigger Layout

F. Power subsystem (Jingkun)

The figure below shows the circuit of the 7805-power model. The advantage of this circuit is the output power of the circuit is 5W and it can provide a stable 5V voltage and 1A current. So we do not worry about the varying of power. In these days, we have done an experiment to measure the efficiency of the 7805 circuit.

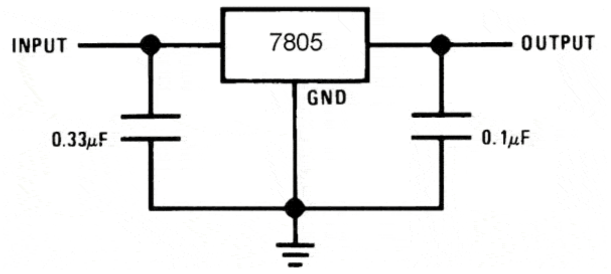


Figure 10. 7805 circuit

By referencing the datasheet, the input voltage range of the 7805 is 7 - 24V, we used different values of input voltage and recorded their output voltages.
The result is in below:

Input voltage / V	**Output voltage / V**	**Efficiency**
7.00	**4.99**	**71.29%**
11.00	**5.03**	**45.73%**
15.00	**5.00**	**33.33%**
19.00	**5.03**	**26.47%**
23.00	**5.01**	**21.78%**

Table 1. Table of the efficiency of the 7805 circuit

Although the efficiency of this circuit is much lower than the power bank, it can provide a stable voltage with a low current while the output current of the power bank is usually between 2.5 - 4A that may damage the components.

Noise is produced by 7805 circuit. To help with noise, a capacitor was added in parallel to the output of the circuit.

IV. Laboratory and Test Plans and Results

A. Transceiver

The HB100 transceiver was initially chosen. After researching the transceiver more intensely, the team realized that the transceiver has designed for short range motion sensing. The sensitivity of the transceiver was uncertain, so the MACOM transceiver was chosen instead. This transceiver was chosen because it was already used to detect baseball-sized objects from a distance of 90 feet in the Bushnell radar gun. This option eliminated the transceiver of being a limiting factor in the design.

The team experienced issues reading a frequency from the transceiver. It was discovered that the transceivers were being damaged as they were being removed from the commercial guns. This was acknowledged after another radar gun team loaned us their transceiver to test. The team was able to move ahead with a better filter and amplifier design after this issue was corrected.

B. Filter/Amplifier Circuit

This first filter design encompassed a second order high pass filter cascaded with a second order low pass filter and amplifier. This design was not able to function because the output of the bandpass filter was always at the maximum voltage. This issue was witnessed using other amplifiers. Using a single stage band pass filter solved our problem and allowed us to move forward.

During the initial tests of the project, it could only detect baseballs and aluminum foil coated tennis balls. This was fixed when noise was removed from the circuit which allowed us to increase the gain high enough to detect uncoated tennis balls

C. LCD/Arduino

The initial steps to determine how the frequency will respond was using the FreqCount library. This library works by setting a time frame when the digital read is high and low. This would result in calculating the time it takes for the pin to be on and and would determine the sample of points per that time, or the frequency. From the figure below shows a graphical representation of the process.

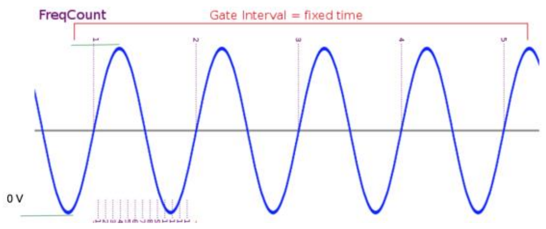


Figure 11.: FreqCount Process

The problem with this implementation is that the rising time must read in a voltage that has to be greater than 1.5V to be considered a high. This means that the amplified circuit must have an offset above that value. The reading must also be periodic or else it will output a frequency that will be distorted and incorrect since the signal will be consistently varied in terms of frequency. Since the intermediate frequency varies in amplitude and in frequency, using the FreqCount Library is not suitable for this project.

Using the trigger to collect the data and output the maximum frequency as it is pressed means that the button will occur a bouncing effect. This can be seen by the figure 2 below [3]:

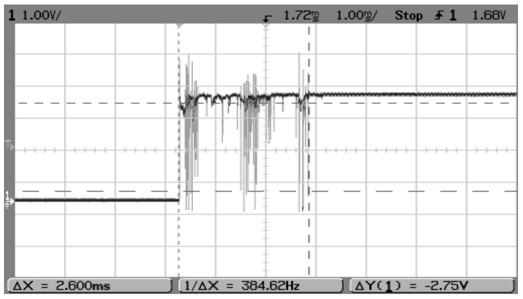


Figure 12. Oscope showing Bouncing Effect

To account for the bouncing effect, the button needs to incur a delay as it is about to be pushed. This means, that once a change in state has occurred once the button is pressed, there will be a 10ms delay and the a final correct digital read would take place.

The last thing to account for the displaying of the speed is to output white space characters. This is known as white space tracing and was used to overwrite previous characters that had been shown on the LCD. This will allow the arduino to print numbers that vary in length as well as print the units without having to overwrite previous results. This also avoids using the clear() function since the clear() function takes a huge amount of delay.

C. Doppler Radar Gun

Our primary testing apparatus was implemented using the following tools: our Doppler radar gun, a commercial Doppler radar gun, measuring tape, tennis balls, and a sheet netting system for safety. One team member would throw a tennis ball straight ahead into the sheet netting system placed a variable distance from the start line. A second team member holds both the commercial radar gun and our radar gun at the start line and triggers them both simultaneously. The speeds displayed by the two radar guns were recorded and the difference between the two values calculated. This test was repeated at a distance of 0.5, 1, 5, 10, 15, 20, 30, 40, and 50 feet. The same team member used the same technique on every throw in order to maintain a semi-constant speed at each distance. At no point in our testing did the difference recorded exceed 3 mph.

The maximum range of our gun was found using the primary testing apparatus and a variation of the above testing procedure. Rather than simply recording the difference between measured speeds, the testing distance was extended until our measurement accuracy fell below our specification threshold. We accurately measured a speed of 14.2 mph from 50 feet away using our Doppler radar gun. It took several tries to achieve this result, likely due to the difficulty of matching the radar gun's area of operation to the tennis ball's path at such a distance.

The maximum speed of a tennis ball measured by our Doppler radar gun was 46.3 mph. This was the fastest any of our team could throw a tennis ball while keeping safety in mind. We measured car speeds at an intersection at upwards of 49 mph. Theoretically our radar gun is capable of measuring speeds up to 143 mph with the limiting factor being the designed filter limitations.

The final dimensions of our radar gun were 7.5 inches long, 2.94 inches wide and 6.69 inches tall, well within specification limits. The final weight of the radar gun was 1.5 pounds, well below the 3 pound limit. Our radar gun is both handheld and portable.

The total current draw of all systems in radar gun was 160 mA. The two 9V batteries used by the power subsystem each supplied 175 mAh for a total of 350 mAh. Thus the total continuous runtime of our radar gun was 2.1875 hours.

V. Bill of Materials

The project was to be completed with a budget of $250 total. The table belows shows the bill of materials that were used in the final product.

Quantity	**Materials**		
	Description	***Unit Price***	***Total Price***
1	Arduino Uno	22.69	22.69
2	Batteries	18.4	36.8
1	LCD 16x2	15.57	15.57
1	LCD Backpack	10.00	10.00
1	Quad 28MHz Low-V OpAmp	0.6	0.6
1	Bushnell (Transceiver)	89.79	89.79
1	Battery Clip Pack	3.73	3.73
		Total	**179.18**
		Left	***70.82***

Table 2. Bill of Material

VI. Gantt Chart

The general form of our groups Gantt chart can be seen in the figure below.

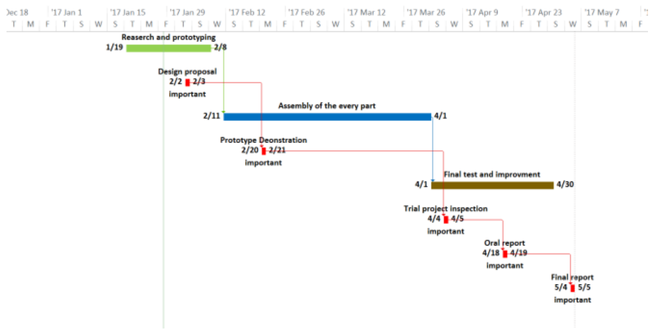


Figure 13. The Gantt chart

A more detailed Gantt chart can be found in Appendix D.

VII. Future improvements

Our Doppler radar gun could be improved through further refinement of the amplifier/filter circuit in order to minimize noise and maximize bandwidth and cutoff slope to form a more ideal response. Our radar gun could also be improved through the consolidation of our microcontroller on to a more compact and personalized printed circuit board. This would minimize wasted space within the casing and remove extraneous components from the assembly. The outer casing of our radar gun could also be improved upon, with minor changes made to eliminate gapping and strengthen the areas used for joining of the two halves.

In the future, a Bluetooth module could be added to our existing radar gun to allow for the transmission of speed data to some kind of external storage device. This would allow for monitoring and aggregation of data to improve sporting practices. This Bluetooth module could for instance transmit speeds to a mobile device that allows users to set goals and view their progress as part of a tennis or baseball training application.

To improve the accuracy and range of the radar gun a different transceiver could be implemented. The MACOM transceiver cannot detect speeds under 10mph and has some limitations in both the accuracy of its speed measurements and its operational range. A different transceiver could eliminate these problems and allow for enhanced functionality of the radar gun.

VIII. Results and Conclusions

Throughout the semester our team has utilized concepts from electromagnetics, circuits theory, embedded systems, signal analysis, physics and computer science to create a functional Doppler radar gun. We have gained competence not only in the application of various technical concepts but also in areas such as project management, public speaking, technical writing and teamwork. Our completed radar gun meets all the specifications required of it. It can measure speeds between 10 - 143 mph at a range of 0.5 - 50 feet with an accuracy of plus or minus 2 mph. Our radar gun is powered by two 9 V batteries and has a continuous runtime exceeding 2 hours. It was completed in one semester and cost $180 to produce, $70 under the maximum budget.

Acknowledgment

This project acknowledges Dr. Rama Ramakumar for his guidance and expertise in the preparation and execution of our project. We also would like to thank, Dr. Krasinski, Dr. Hutchens for their help in the design of the filter.

References

[1] "Miles to Meters (m) - How many meters in a mile ?", Asknumbers.com,2017.[Online].Available:http://www.asknumbers.com/MilesToMeters.aspx. [Accessed: 29- Apr- 2017].

[2] MACS-007801-0M1RM0, 1st ed. MACOM, 2017, p. 1.

[3] "Crypto Shield Hookup Guide - learn.sparkfun.com", Learn.sparkfun.com, 2017. [Online]. Available: https://learn.sparkfun.com/tutorials/crypto-shield-hookup-guide/maxim-integrated-ds3231m-real-time-clock. [Accessed: 29- Apr- 2017].

[4] *Single/Dual/Quad, 28Mhz, Low-Noise, Low-Voltage, Precision Op Amps*. 1st ed. 2009. Web. 4 May 2017.

[5] ARTICLES, TECHNICAL et al. "Teardown Tuesday: Radar Gun". *Allaboutcircuits.com*. N.p., 2017. Web. 4 May 2017.

[6] "ArduinoFFT - Open Music Labs Wiki", *Wiki.openmusiclabs.com*, 2017. [Online]. Available: http://wiki.openmusiclabs.com/wiki/ArduinoFFT. [Accessed: 04- May- 2017].

[7] Blog.arduino.cc. (2017). *Arduino Blog » Nice drawings of the Arduino UNO and Mega 2560*. [online] Available at: https://blog.arduino.cc/2011/01/05/nice-drawings-of-the-arduino-uno-and-mega-2560/.

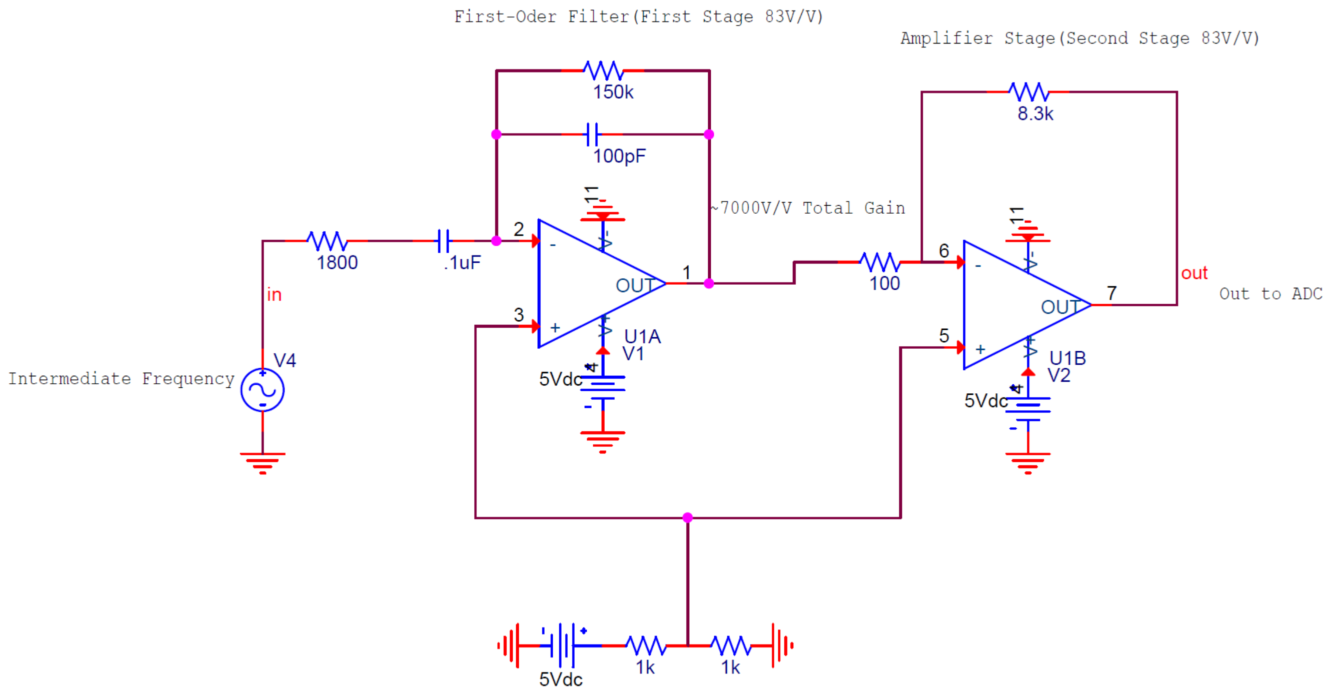
[8] 7805 Datasheet. (2004). 1st ed. [ebook] Texas Instruments Incorporated. Available at: https://www.sparkfun.com/datasheets/Components/LM7805.pdf.

[9] Arduino Uno Drawing. (2013). 1st ed. [ebook] Available at: http://www.krekr.nl/wp-content/uploads/2013/08/Arduino-uno.pdf.

[10] 2017. [Online]. Available: https://www.limpkin.fr/public/HB100/HB100_Microwave_Sensor_Application_Note.pdf. [Accessed: 04- May-2017].

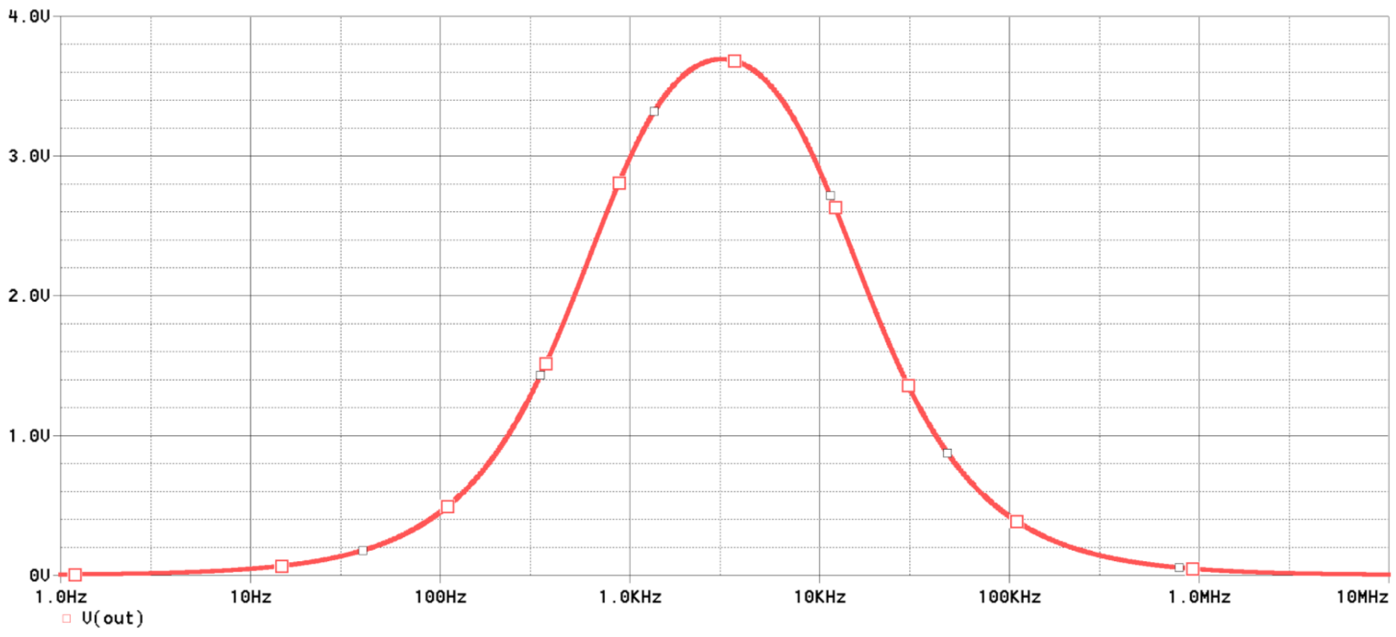
Appendix

Appendix A.



Filter/Amplifier Schematic

Appendix B.



Theoretical Frequency Response of the Filter/Amplifier Circuit

APPENDIX C..

Code_With_Button.ino

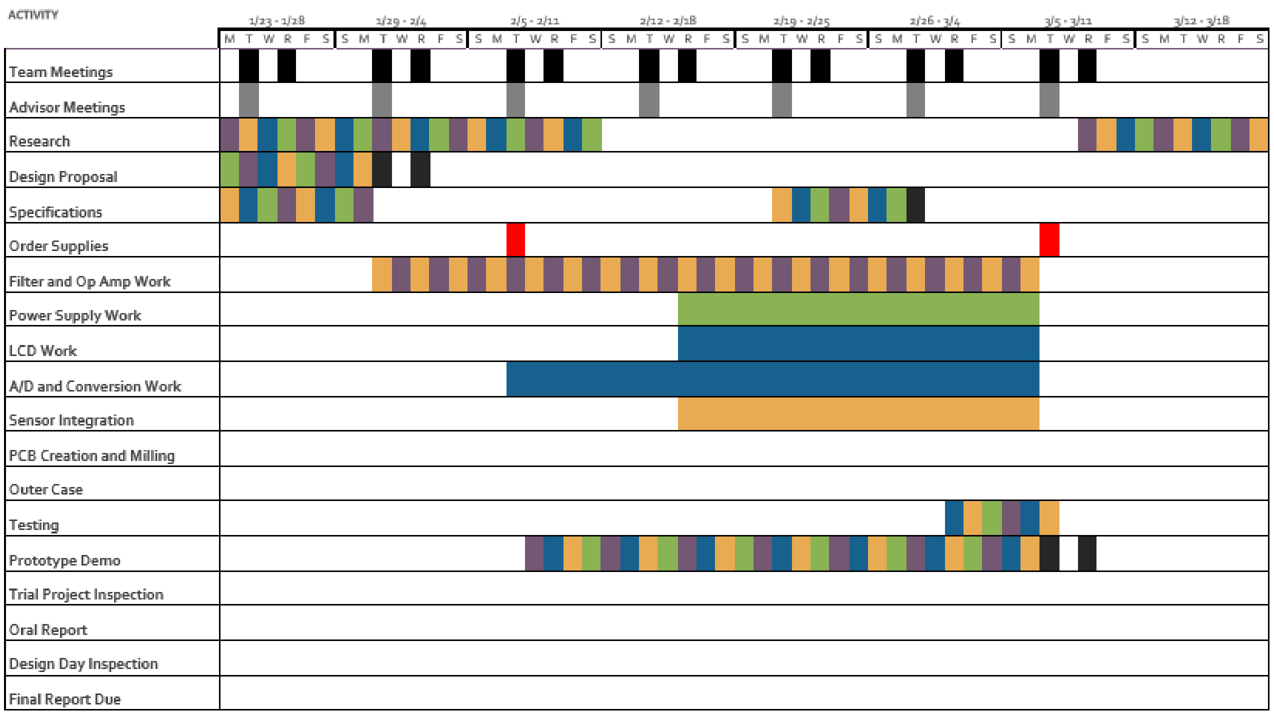
```
/*
 * Runs FHT with real data. When button pressed, Records max freq bin.
 * Outputs the freq and speed when button is released
 * Author: James Touthang
*/

// Definitions
#define LOG_OUT 1         // use the log output function
#define FHT_N 256         // set to 256 point fht

// Libraries
#include <FHT.h>          // include the library
#include "Wire.h"
#include "Adafruit_LiquidCrystal.h"

Adafruit_LiquidCrystal lcd(6, 2, 4);

// Optimizing Variables for better reading
int max_mag = 160;                    // magnitude of bin threshold, anything above gets stored as max
int bin_multiplier = 146.05;            // samplerate/fht. bin multiplier
//----------------------------------------------------------------------------------------------------

double highest_freq = 0;                // Storing the highest Frequency
int buttonState = HIGH;                 // button state is initialized as "pressed"

// custom Character
byte arrow1[8] = { 0B10000,0B11000,0B11100,0B11110,
                   0B11100,0B11000,0B10000,0B00000 };
byte theo[8] = { 0b00000,0b01110,0b10001,0b10001,
                 0b01110,0b00000,0b11111,0b00000};
byte theS[8] = { 0B01110,0B11011,0B10000,0B11100,
                 0B00111,0B10001,0B11011,0B01110};
byte theu[8] = { 0B00000,0B11111,0B00000,0B10001,
                 0B10001,0B10001,0B01110,0B00000};

double toMilesPerHour(double dopplerFreq){
  /*Doppler Formula
  // Transmitting Frequency = 24.125 GHz
  // Speed of light = 299,792,458 m /s
  // 3600 sec in 1 hour
  // 1609.34 meters in 1 mile
  */
  return (dopplerFreq * 299792458.0 * 3600.0 ) / (2.0 * 24125000000.0 * 1609.34);
}

double toKilometersPerHour(double dopplerFreq){
  return (dopplerFreq * 299792458.0 * 3600 ) / (2.0 * 24125000000.0 * 1000);
}

// method to get max of two values.
int maxValue(int x, int y){
  if(x >= y) return x;
  return y;
}

// method to account for the bouncing during the button press.
boolean debounceButton(boolean state){
  boolean stateNow = digitalRead(9);
  if(state!=stateNow){                // if buttonState and digitalRead is diff, it delays for 10ms
    delay(10);                        // waits 10 ms to to account for the bouncing
    stateNow = digitalRead(9);
  }
  return stateNow;
}

void setup() {
  Serial.begin(115200);               // use the serial port
  pinMode(9,INPUT);                   // Button
  lcd.begin(16, 2);                   // LCD Initialization
  lcd.setBacklight(HIGH);             // LCD Backlight
  lcd.createChar(1, arrow1);          // saves custom char to address 1
  lcd.createChar(2, theo);            // saves custom char to address 2
  lcd.createChar(3, theS);            // saves custom char to address 3
  lcd.createChar(4, theu);            // saves custom char to address 4
  TIMSK0 = 0;                         // turn off timer0 for lower jitter
  ADCSRA = 0xe5;                      // set the adc to free running mode
  ADMUX = 0x40;                        // use adc0
  DIDR0 = 0x01;                       // turn off the digital input for adc0
}

void loop() {
 while(1) {                           // reduces jitter
    cli();                            // UDRE interrupt slows this way down on arduino1.0
    for (int i = 0 ; i < FHT_N ; i++) { // save 256 samples
      while(!(ADCSRA & 0x10) );       // wait for adc to be ready
      ADCSRA = 0xf5;                  // restart adc
      byte m = ADCL;                  // fetch adc data
      byte j = ADCH;
      int k = (j << 8) | m;           // form into an int
      k -= 0x0200;                     // form into a signed int
      k <<= 6;                        // form into a 16b signed int
      fht_input[i] = k;                // put real data into bins
    }

    fht_window();                     // window the data for better frequency response
    fht_reorder();                    // reorder the data before doing the fht
    fht_run();                        // process the data in the fht
    fht_mag_log();                    // take the output of the fht
    sei();                            // sets back interrupt

    // Writes to Serial to show graphic spectrum analyzer
    Serial.write(255);              // send a start byte
    Serial.write(fht_log_out, FHT_N/2);     // send out the data

    /*
     * fht_log_out[i],
     *    i is the frequency bin
     *    fht_log_out[i] is the magnitied
     * frequency: f(i) = i * sample_rate / FHT_N
     * sample_rate = 16 Mhz / some prescaler
     */
    int largest_index = 0;
    int last = 0;
    for (byte i = 6; i < FHT_N/2; i++) {    // iterates through each bin and if is the largest it stores it.
      if( fht_log_out[i] >= max_mag) {
        largest_index = i;
      }
    }
    last = largest_index;             // sets the last largest_index

    // takes the two values and gets max and multiplies by a multiplier to get freq
    double cur_freq = maxValue(last,largest_index) * bin_multiplier;

    // sets the highest_freq during reading to get the maximum value
    if(cur_freq > highest_freq ) {
      highest_freq = cur_freq;
    }

    // Code for button pressed. Sets the highest_freq to 0 and display words
    if(debounceButton(buttonState)==HIGH && buttonState == LOW){
      highest_freq = 0;
      lcd.setCursor(0,0); lcd.print("Collecting..       ");
      lcd.setCursor(0,1); lcd.write(1);lcd.write(1);lcd.write(1);lcd.write(1);lcd.print("  ");
                      lcd.write(2);lcd.write(3);lcd.write(4);lcd.print("  ");
                      lcd.write(1);lcd.write(1);lcd.write(1);lcd.write(1);

      buttonState = HIGH;           // after exetuing all LoW states, set ButtonState to HGIH
    }

    // Code for when button released. prints the data to lcd
    else if ( debounceButton(buttonState)==LOW && buttonState == HIGH){
      //lcd.setCursor(0,0); lcd.print(highest_freq); lcd.print(" Hz        ");
      lcd.setCursor(0,0); lcd.print(toMilesPerHour(highest_freq)); lcd.print(" MPH           ");
```

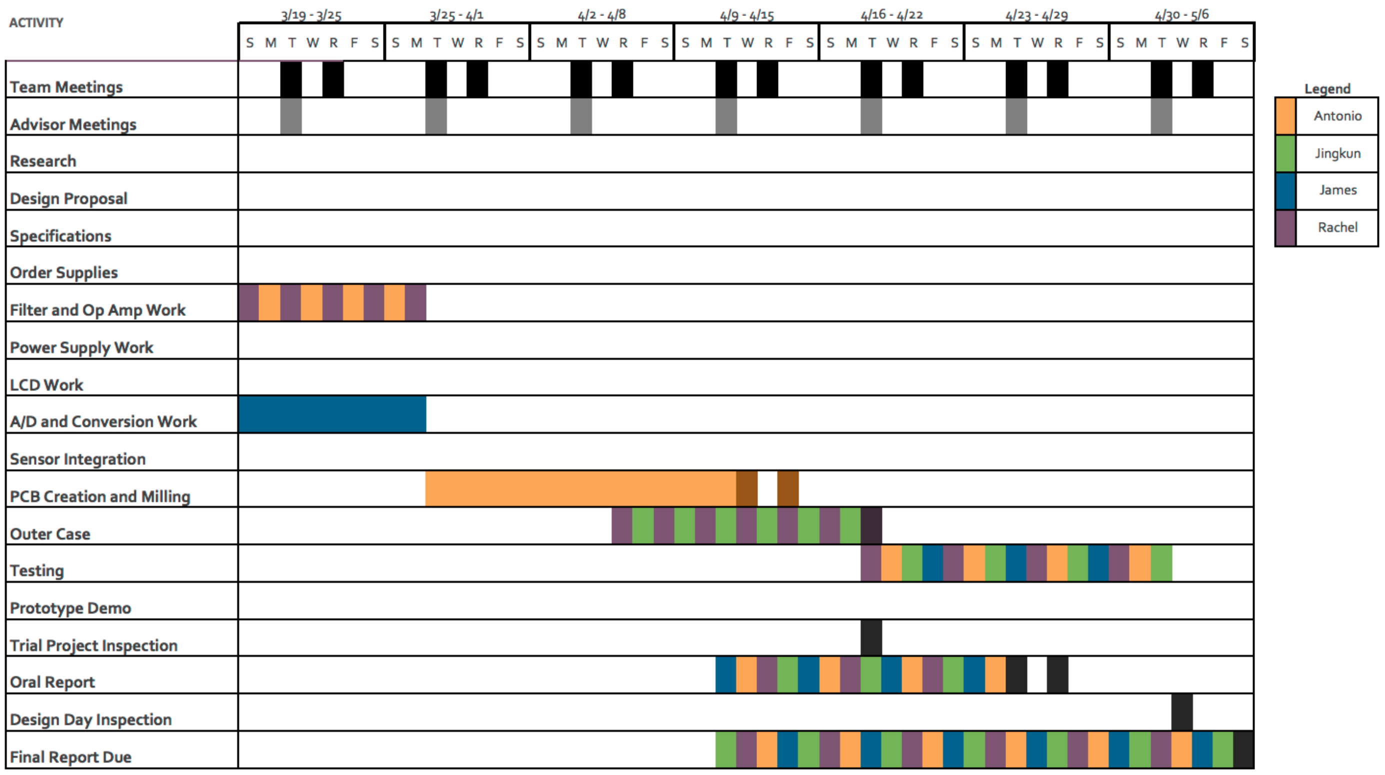
```
        lcd.setCursor(0,1); lcd.print(toMilesPerHour(highest_freq)*1.61); lcd.print(" km/hr        ");
        buttonState = LOW;            // after executing all HGIH states, set ButtonState to LOW
      }
  }// end of while
}
```

Source Code for Arduino Uno

APPENDIX D.

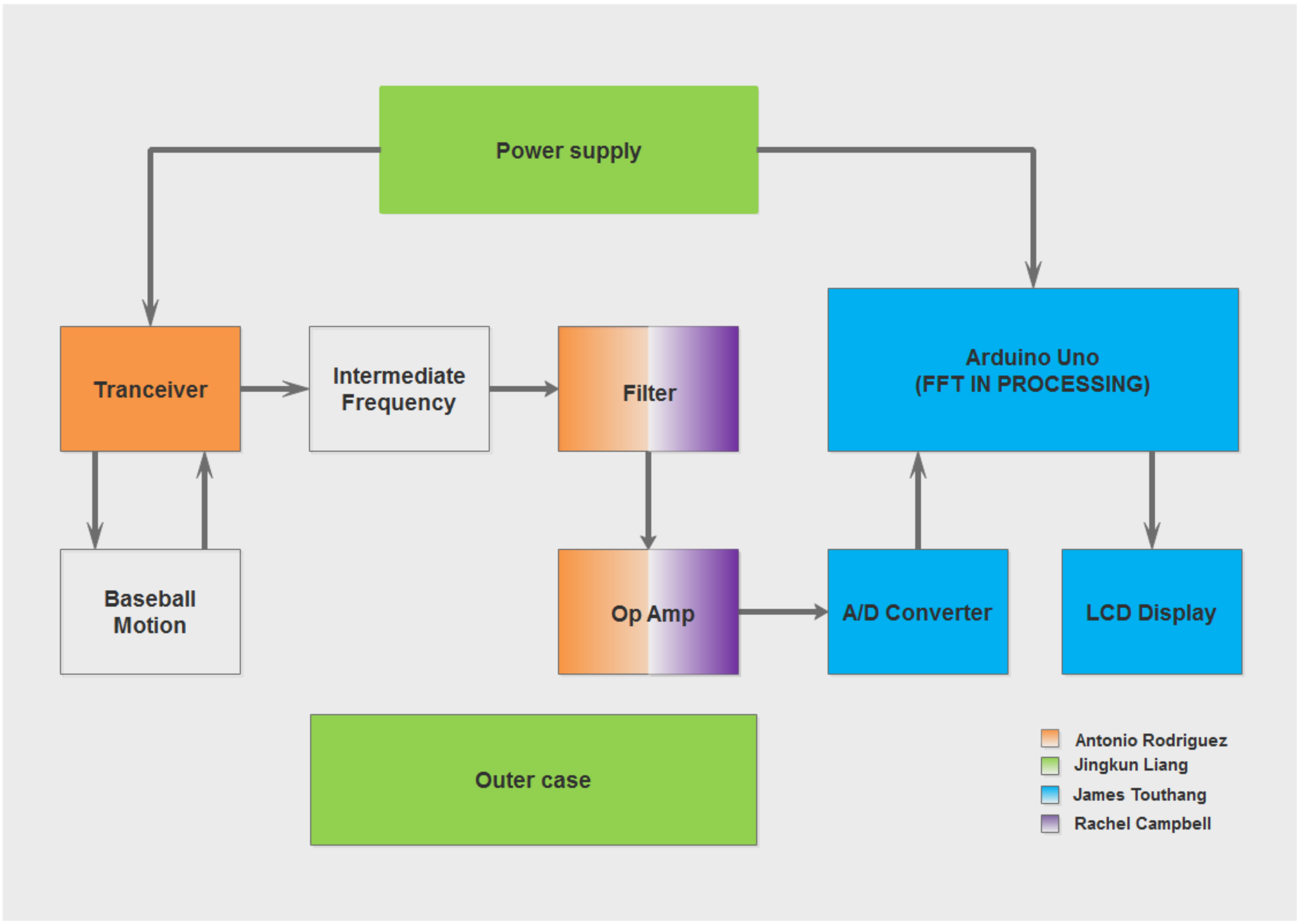
Gantt Chart





Gantt Chart

APPENDIX E.



Block Diagram