Spring 2020

Digital, Automated Reactive Target System

Nicholas Haas
njh63@zips.uakron.edu

SaiPranay Vellala
sv92@zips.uakron.edu

Trandon Ware
ttw13@zips.uakron.edu

Thomas Martin
tam150@zips.uakron.edu

Digital Automated Reactive Target

(D.A.R.T) System

Project Design Report

DT14

Nicholas Haas

SaiPranay Vellala

Thomas Martin

Trandon Ware

Dr. Adams

April 24, 2020

Table of Contents

List of Figures

List of Tables

Abstract

In this era, technology is woven into almost every facet of our leisure activities. Although technology has innovated hobbies ranging from chess to soccer, the art of shooting has been neglected. Unnecessary insufficiencies such as bullet ricochets off of mechanical steel targets, ineffective progress tracking, and general inaccessibility to outdoor training facilities are all improvable areas of this sport. The Dynamic Automated Reactive Target (D.A.R.T) System aims to fill some of these gaps and help modernize recreational marksmanship. Modeling the system after a dueling tree will optimize the use of the system and allow for different training models to challenge the marksman. The system will utilize accelerometers to collect accurate data regarding target tracking from the user during training, and custom motor controllers allow for optimized movement response times of each target. To ensure safety and increase efficiency, Bluetooth communication will grant the marksman control of the system from anywhere within the range with the use of any portable device. With the algorithms developed, multiple different training simulations can be practiced. In all, the D.A.R.T. System will revolutionize the safety and convenience of shooting sports.

1. The D.A.R.T system circuitry should be protected from environment to provide durable structure.
2. The D.A.R.T system needs to be safe for indoor use.
3. The D.A.R.T app should be easy to control and display information.
4. The D.A.R.T system should give real time target feedback.
5. The D.A.R.T system needs to be able to perform multiple training simulations.
6. The D.A.R.T system needs to be mobile and easy to set up or tear down

Problem Statement

Need

In order to make the sport of shooting more accessible to the public, safe alternatives to outdoor reactive target training need to be created to enable practice on indoor firearm ranges. Although an estimated 51 million people participated in shooting sports in 2014, only about 14 million people did so at an indoor range, even though indoor ranges tend to be more accessible to the public [11]. The current reactive target systems are limited to outdoor use since the majority of the targets are made of steel plates, which are hazardous to use indoors. This in turn limits the amount of places one can practice on reactive targets, as well as the amount of days that he/she can utilize this type of shooting due to the season and weather. [NH, TM, SV, TW]

Objective

The objective of this project will be to create a digital, automated reactive target (D.A.R.T) system. The design and implement an indoor reactive target system that is safe, interactive, affordable, and can be an alternative to outdoor steel targets. The system will have sensors that can detect a hit on target and respond accordingly. Data from the use of the target system should be displayed in an easy to understand and accessible format for the user. [NH, TM, SV, TW]

Background

In the professional marksman community, there is a lack of reactive target systems that are ineffective due to safety, practicality and inadaptability. Training outdoors is not a consistent option and using the same equipment as outdoor target systems poses a threat to human safety due to flying shrapnel. To construct challenging and unique scenarios for a marksman, the system must be electronically operated and

must be programmed to create dynamic situations. In order to provide an effective reactive training simulation, the DART System was designed to provide a safe environment where the system dynamically responds to a marksman and provides real-time feedback. [SV]

The primary form of reactive target training is done using steel targets that react to hits on target with sound or movement, based on the mechanical design of the target. This form of training is limited to outdoor use only due to the fragmentation of bullets on impact with the steel, which if done indoors could present a hazard to participants. A study done by Bui shows the scale and impact that ricochet fragments can have on their surroundings [10]. Training outdoors also limits the versatility since it is difficult to track each bullet's path point of impact on the target; outdoor targets are static and not electronically operated, making the reactive aspect of an outdoor target fairly basic. Outdoor ranges have limited availability based on weather and season. Alternatively, indoor firearm ranges are not equipped to protect either the marksmen or the building structure from ricochet of shooting steel targets, making them unsafe for general use indoors. Some solutions to bring this form of reactive shooting indoors include both laser and altered reality training simulations. The laser training systems allow for dynamic reactions of steel targets to be more accessible to marksmen. By using a system that is not dependent on a projectile, there is more opportunity to safely train both indoors and outdoors. The United States Army Research Institute for the Behavioral and Social Sciences did a study where they found that there was a strong correlation to training scores on a laser system compared to live fire drill [2]. Similarly a study at Duke University showed that using a virtual reality simulation can accurately represent a

marksman ability [3]. The main limitation with these types of training simulations are that even though a dynamic target response can be obtained there is a loss of response from the firearm in the form of recoil. As suggested by Morelli [1], recoil adds a measurable impact on the performance of the shooter. These systems may be able to predict a marksman's ability, but it does not fully simulate a live fire situation. There exist live fire electronic targets that can be used on indoor firearm ranges, but they tend to be used for precision shooting verses reactive target training. This means that most electronic targets lack the ideal dynamic response and versatility that obtained on steel targets. [NH, TM]

The design for the DART System would combine the safety of the outdoor ranges with the quality and convenience of the indoor ranges while including additional features that would enhance the experience for the person(s) using it. A virtual targeting system called The Live Fire Screen has some similarities to the design of the DART System. Both of these systems allow for safe indoor usage and can determine the position of a hit remotely. They also both give the user real-time feedback on whether or not they hit a shot, and which shots they hit. In addition, both systems are very portable and can be easily set up and torn down. The Live Fire system supports specific weapons such as revolvers and submachine guns. Its qualified ammunition includes: NATO 9mm, NATO 5.56mm, and 7.62mm rounds. The screen itself is designed to be able to take up to 50,000 rounds before a replacement screen is needed. [TW]

Another system that is of a similar design to the DART System is the XWT GEN4 Wireless Target Carrier. This system includes programmable scenarios and trials to improve a user’s skills, which is one of the marketing goals for the design of the

DART System. They are both easy to control and display the information in an easy-to-understand manner. The XWT GEN4 system also features a quiet internal system, without bulky and noisy motors, and it also does not collect debris or lead casings from gunfire. In addition, it can move along a rail system and provide “advance” or “retreat” training exercises. [TW]

Both the Live Fire Screen and the XWT GEN4 Wireless Target Carrier lack some aspects that make the design more convenient, and they also have some features that would make the DART System better. Looking at what else is out there could provide some insight in making the DART System more efficient and more convenient for all who use it. This in turn, will make it more desirable to consumers. The suggested design for the system will allow for the marksmen to safely enter into a dynamic training simulation in a life fire setting. [TW]

Ever since the inception of guns, marksmen, in tandem with inventors, have been aiming to improve their firearm control and reactiveness through gun training simulations. As a result, there are many existing designs for target systems dating back to the early 1900s. The early days of target system engineering showcased the rotary target [4], well known for the bullseye in the center. Even though the patent has expired, the design paved the way for the improvement of training simulations; the DART system intends to use a rotary target as its fundamental target design and will build upon this landmark patent. As time progressed, the designs for the target systems became increasingly complex, allowing marksman to train their reactiveness in addition to their aim: the concept of a reactive training simulation [5] was born. Because the static behavior of the rotary target does not lend itself to effective training when it comes to

quick accuracy and awareness, creating a scenario where there is target movement and the marksman must track down the target allows for training in a more diverse environment. However, in recent years, the preexisting target training simulation market has been weakly disrupted, causing little to no innovation to occur. Around 2008, lasers started to make an entrance in the method of training: instead of an actual gun being fired and a bullet penetrating a surface, a laser, sensing devices and a processor were used to track an imaginary bullet [6]. Although this patent cleverly detects when a surface is 'hit', the marksman is not firing any weapon and is not impacted by any natural gun responses, such as recoil and acoustic feedback. Because reactive training simulations intend to prepare marksmen for real life situations, having no feedback from a firearm is ineffective and dissolves the challenge of a reactive target training simulation. The DART system intends to use electronic targets to create dynamic responses and will house a plethora of sensors to detect real bullets being fired and hitting the target. The sensors will also collect and transmit data back to the marksman to give real-time feedback of his or her performance. In 1980s, real-time feedback equated to a computer-controlled target system that automatically scored the marksman and provided simple visual feedback to indicate when a target has been hit [7]; the user was not provided with any statistics or data trend. Providing real-time feedback is crucial to the improvement of a marksman, and the DART system intends to improve upon the 'Automated Target System' and its antiquated design in order to allow the marksman to actively monitor performance during the simulation and analyze the data trends from the collected data. The DART system intends to take existing patents, and redesign them in order to create an effective reactive targeting system that has real-time feedback and dynamic responses

to marksman interaction. Although firearm training simulations have been around for ages, no reactive training system is as safe, effective and versatile as the DART system. [SV]

Marketing Requirements

7. The D.A.R.T system circuitry should be protected from environment to provide durable structure.
8. The D.A.R.T system needs to be safe for indoor use.
9. The D.A.R.T app should be easy to control and display information.
10. The D.A.R.T system should give real time target feedback.
11. The D.A.R.T system needs to be able to perform multiple training simulations.
12. The D.A.R.T system needs to be mobile and easy to set up/tear down.

[NH, TM, SV, TW]

Engineering Analysis

Circuit

Actuating Circuit [NH, TM]

A DC motor will be implemented to actuate the rotation of the target from the resting to active position. To prevent the momentum of the bullet hitting the target, pushing the arm of the target or motor away from its active position, a worm gear and drive will be utilized between the motor and the target arm to prevent any undesired movement. This unwanted movement could otherwise interfere with the accelerometer calculations or position of the target when the motor is not active. Additionally with a worm gear drive, the motor is not required to continuously expend energy to hold its position.

First in doing this analysis, it is important to know the moment of inertia for calculating the work done. Before calculating the moment of inertia (I), the mass must be determined with the following equation, where m_c is the mass of the target in kg:

$$m_c = Density * Volume \quad \textit{Equation 1}$$

Multiplying density of cardboard ($68.9\frac{g}{cm^3}$) by the volume of the cardboard present in the target will give the mass of the target, m_c [12]. The volume is assumed by treating the target as a thin cylinder by taking the target face area and multiplying by the depth of the cardboard, which is assumed to be 1 cm. The weight in grams can then be calculated by dividing mass in grams by one thousand, resulting in m_c= 0.487 [kg]. The diameter *d* of the target face [cm] is assumed to be 30cm.

$$I = m_c * d^2 \quad \textit{Equation 2}$$

Because this calculation is based off of a moment of a point particle, I, the work required to move the target from one position to another will be a viable approximation to use for further calculations. The following equation can be used to find the work done by the motor(W_d), where ω^2 is the angular velocity measured in $\frac{rad}{sec}$:

$$W_d = I * \omega^2 \quad \textit{Equation 3}$$

Angular velocity is calculated as π^2, where the motor moves $\frac{\pi}{2}$ radians, and the desired time of actuation is set to 0.5 seconds.

From this equation, the power used by the motor can be discovered, where power

(*P*) is defined as the differential of work with respect to time. From this concept, the power exerted by the motor for one movement can be calculated by the following equation, where *t* is the interval of time over which the work was done:

$$P = \frac{W_d}{t} \qquad \textit{Equation 4}$$

Assuming a constant 12V DC source is available, the current needed to drive this motor can be determined by the following equation:

$$I = \frac{P}{V} \qquad \textit{Equation 5}$$

Finally, when using a diameter of 30cm, assuming a weight of 0.487 kg, and a 12V source, the project requires a DC motor that can handle around a 600 mA current.

The following code was created using Matlab and was used to calculate motor torque required to maintain position of a target that was hit with a bullet.

```
%_________________variables_________________
d = 30; %target Diameter (cm)
de= 0.689;  %density of cardboard (g/cm^3)
deg = 90; %degrees traveled
t = 0.5;%time to actuate (s)
c = 1; %thickness of target (cm)
V = 12;
Cpm = 75;%cycle per mode
D_motor = .5
hr = 3;

%____________Conversions____________________
rad = (deg/360)*2*pi; %convert to radians
w = rad/t; %angular velocity of the target(rad/s)
r = d/2; %radius of target (cm)
d_m = d/100; %convert to meter
```

```
a= pi*r^2; %target face area (cm^2)
mc_g = de*a*c; %mass of target (g)
mc_kg = mc_g/1000; % convert to kg
hr_motors = hr * D_motor;
%_____Calculations Power_____
I = mc_kg*d_m^2;
W_tot =I*w^2
P = W_tot/t

%_____Motor Current Draw_____
A=2*P/V; %for one cycle
I_draw = A*Cpm; %current drawn in 1 mode
Ahr=I_draw*hr_motors %amp hours of the Motors
```

Actuating Circuit Uncompensated Transfer Function Derivation [TM]

Based on the motor calculation a PD controller was chosen to be used for the system. The proposed controller will used a potentiometer attached to the driving shaft of the motor to act as feedback to the closed loop system. The proposed system will look as follows:

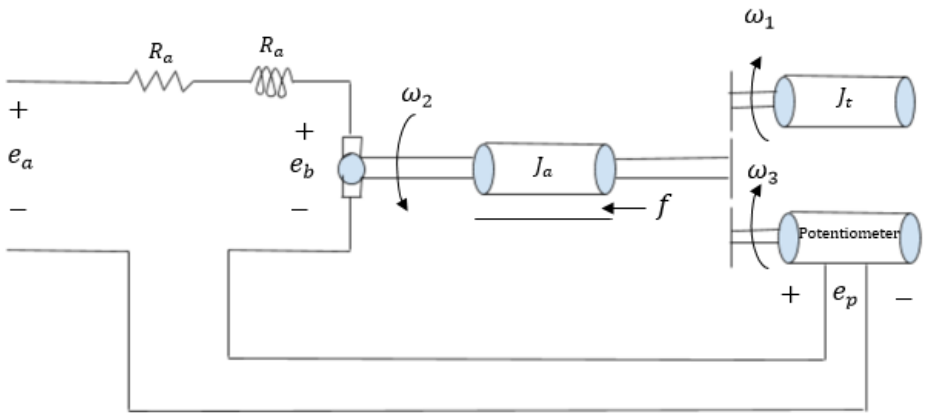


Figure 1: Electromechanical representation of the actuationg system.

From here the system can be broken down into a block diagram that represents the control loop for the derived equations.

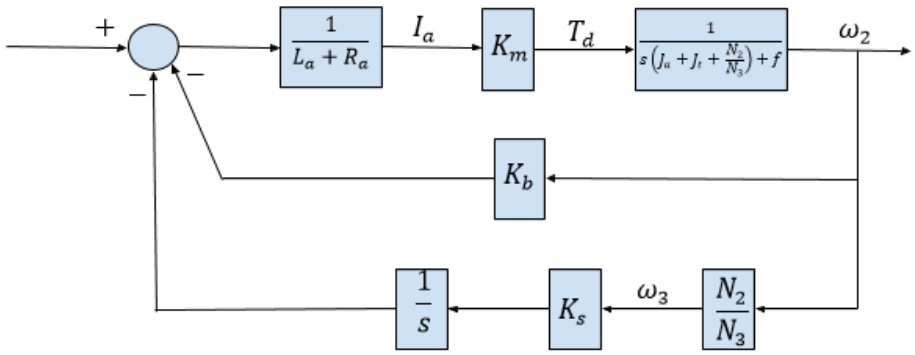


Figure 2: Control Loop Block Diagram of the actuating system.

From this the transfer function of the system can be derived. The first equation needed to find the uncompensated transfer function of the system is to isolate an equation for the armature voltage, this is found by using Kirkoff's current law to solve for armature emf in terms of the resistance R, current through the armature i(t), potentiometer emf, and motor emf

$$e_a = R_a i_a(t) + L_a i_a(t) + e_b(t) + e_p(t) \qquad \textit{Equation 1}$$

Next we must find an equation for the back emf of the motor, the motor constant multiplied by the rotational velocity of the motor shaft as shown below

$$e_b(t) = K_b\, \omega_1(t) \qquad \textit{Equation 2}$$

Now we can relate the Torque developed in the motor shaft with the current in the armature with a motor constant k

$$T_d(t) = k_m\, i_a(t) \qquad \textit{Equation 3}$$

Knowing that the torque developed by the motor is the same as the load torque, we can write an equation relating the moments of the target arm and motor back to the torque and friction using reflecting coefficients where necessary, as shown.

$$T_L(t) = J_a\,\theta''_2(t) + f\theta''_2(t) + J_t\theta''_2(t)\frac{N_2}{N_3} \qquad \textit{Equation 4}$$

The aforementioned reflecting coefficients based off the rotational velocity is shown below

$$\omega_2 = \omega_3\frac{N_3}{N_2} \qquad \textit{Equation 5}$$

Since the transfer function is eventually going to be in terms of $T(S) = \frac{\theta_2(S)}{E_a(S)}$, it is important to relate the position of the second arm to the to the emf seen by the potentiometer, as show here

$$e_p(t) = K_s\,\theta_2(t)\frac{N_2}{N_3} \qquad \textit{Equation 6}$$

The Las equation represents how to relate the number of turns N in a potentiometer, the emf E, and the potentiometer constant k

$$K_s = \frac{E}{2\pi N} \qquad \textit{Equation 7}$$

Now that all the equations have been written in time domain, it is important to turn these transform into the frequency domain to create the uncompensated transform function. Using the Mason's gain formula on the block diagram as shown below, we are able to get a uncompensated transform function of:

$$T(S) = \frac{\theta_2(S)}{E_a(S)} = \frac{1}{(L_a+R_a)[s^2\left(J_a+J_t+\frac{N_2}{N_3}\right)+s(fk_mk_b)+k_mk_s]} \qquad \textit{Equation 8}$$

Analog Circuit PD Controller Derivation [TM]

The first thing needed for the design of the controller is the uncompensated transfer function. This was found earlier in the electromechanical section, and is represented by,

$$T(S) = \frac{\theta_2(S)}{E_a(S)} = \frac{1}{(L_a+R_a)[s^2\left(J_a+J_t+\frac{N_2}{N_3}\right)+s(fk_mk_b)+k_mk_s]} \qquad \textit{Equation 9}$$

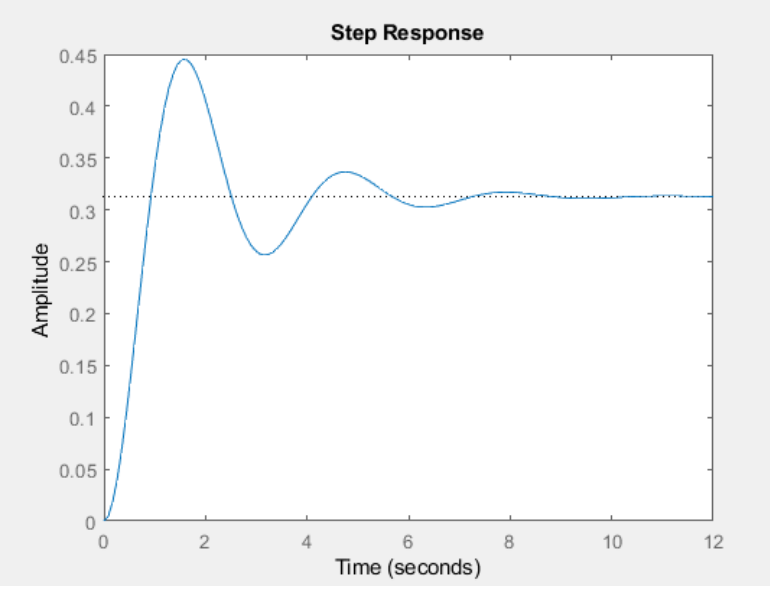


Figure 3: This represents the response of the uncompensated system in the time domain. As this figure illustrates a Proportional Derivative controller is needed to lower the oscillations of the response, as well as lessen the rise time to within the predetermine

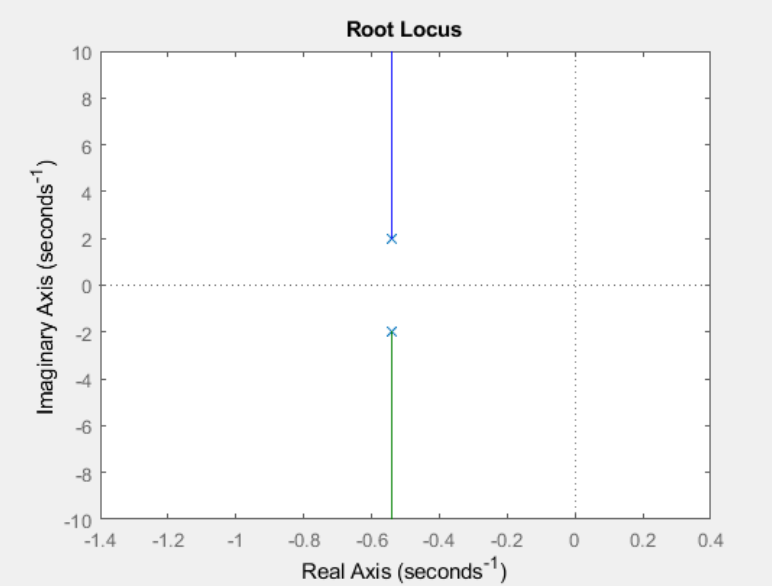


Figure 4: This shows the root locus of the uncompensated system, which we will try to manipulate with our controller to drive a better response for the system.

To implement the first part of the controller, the proportional controller. This sees the uncompensated system multiplied by a constant k to try and reduce the overshoot of the system, and is implemented as shown below,

$$T_c(S) = \frac{\frac{k_n k_m}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}}{[s^2+s\frac{(fk_m k_b)}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}+\frac{k_m k_s}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}]} \qquad \textit{Equation 10}$$

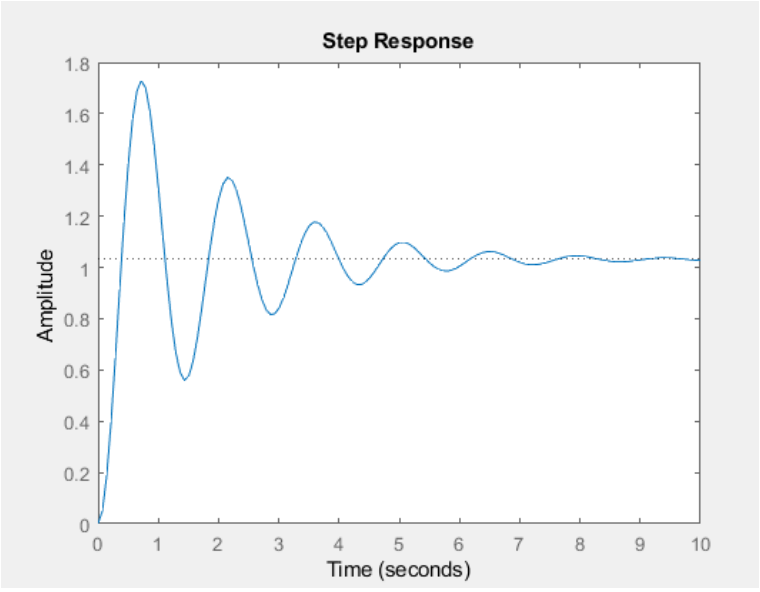


Figure 5: This shows that the proportional controller, while marginally improving the rise time of the system, has added noticeable levels of destabilization to the system in the forms of new oscillations.

With a less stable system a Derivative controller is also needed to further decrease the rise time and add a bit more stability back in the system. To do this the uncompensated transfer function will need to be multiplied by $sk_d + k_n$, where k_d is the gain used for the derivative controller and k_p is the constant found through the proportional controller. Multiplying this together, the new compensated transfer function will look like this,

$$T_c(S) = \frac{\frac{sk_dk_m+k_nk_m}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}}{[s^2+s\frac{(fk_mk_b)}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}+\frac{k_mk_s}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}]} \qquad \textit{Equation 11}$$

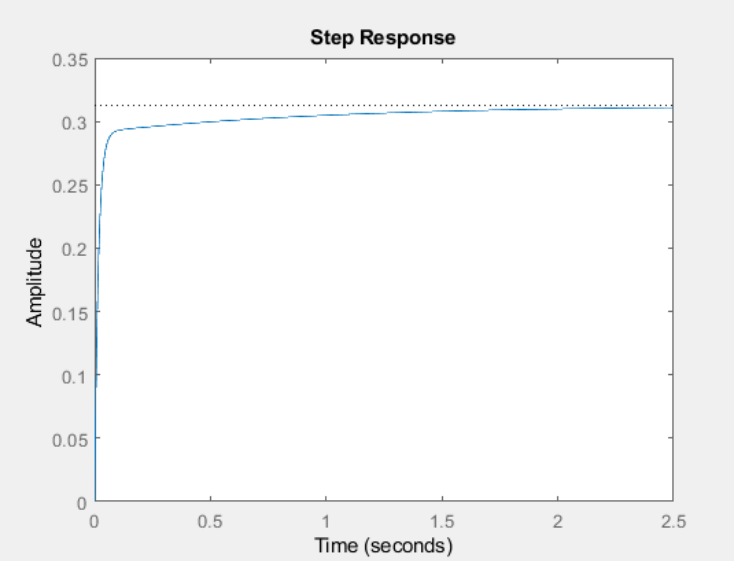


Figure 6: This shows the response of the PD controlled compensated transfer function. This is now within the desired rise time of 0.5 seconds and shows a much more stable system, that won't overshoot the end position.

Somewhat ironically this response, while idealistically would work great, in practice a rise time that more closely hugs the 0.5 second limit is more desirable. This is because the project in real time would be more likely to experience complications due too quick of a movement, which could damage equipment effecting later performance of the actuating system. It is for this reason

a PID controller will likely be necessary, the likes of which are still being worked out, but the equation for it can be found below.

$$T_c(S) = \frac{\frac{sk_d k_m + k_n k_{m+\frac{1}{s}k_p k_m}}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}}{[s^2+s\frac{(fk_m k_b)}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}+\frac{k_m k_s}{(L_a+R_a)\left(J_a+J_t+\frac{N_2}{N_3}\right)}]} \qquad \textit{Equation 12}$$

MATLAB Code for PD Controller:

```
clc
clear
km=2.5;      %(kg*cm)/A
kb=0.81851113590118;  %(v*s)/rad
ks=3.197186342055   ;
%za=50;  magnitude of armature impedance (I think)
Jt=0.27520351645447;
Ja=1.8849555921539;
%f=.30;
N1=44;
N3=1;
N2=1;

excess = 3;

num0 = [km/(Ja+Jt*(N2/N1))];
den0 = [1,kb*km/(Ja+Jt*(N2/N1)),km*ks/(Ja+Jt*(N2/N1))];
r0 = roots(den0)
bp = .07589;

num = [km/(Ja+Jt*(N2/N1))];
den = [1,kb*km/(Ja+Jt*(N2/N1)),km*ks/(Ja+Jt*(N2/N1))];

TF = tf(num,den);
figure(1)
rlocus(TF)
figure(2)
step(TF)
S1 = stepinfo(TF)
r1 = roots(den)

% kp = 1;
% for n = 1:3
%     p = abs(abs(pole(n)) - bp);
%     kp = kp * p;
% end


%Porpotional Controller only

kn = 15;
num2 = [kn*km/(Ja+Jt*(N2/N1))];
den2 = [1,kb*km/(Ja+Jt*(N2/N1)),km*ks/(Ja+Jt*(N2/N1))+kn];
TF2 = tf(num2, den2);
```

```
r2 = roots(den2)
figure(3)
step(TF2)
S1 = stepinfo(TF2)


%PD controller
kd=14;
num3 = [kd*km/(Ja+Jt*(N2/N1)),kn*km/(Ja+Jt*(N2/N1))];
den3 =
[1,(kb*km/(Ja+Jt*(N2/N1)))+kn*(km*ks/(Ja+Jt*(N2/N1))),km*ks/(Ja+J
t*(N2/N1))*kn];
TF3 = tf(num3, den3);
r3 = roots(den3)
figure(4)
step(TF3)
S1 = stepinfo(TF3)


%PID Controller
%ki=14;
%num4 = [kd*km/(Ja+Jt*(N2/N1)),kn*km/(Ja+Jt*(N2/N1))];
%den4 =
[1,(kb*km/(Ja+Jt*(N2/N1)))+kn*(km*ks/(Ja+Jt*(N2/N1))),km*ks/(Ja+J
t*(N2/N1))*kn];
%TF4 = tf(num4, den4);
%r4 = roots(den4)
%figure(5)
%step(TF4)
%S1 = stepinfo(TF4)
```

Sensing Circuit Calculations [NH, TM]

For this project, two options for a sensing circuit were considered: the first option utilizes an inductive loop to track whenever a conductor (bullet) passed through the loop changing the magnetic field and creating a measurable current in the inductive loop, and the second option utilizes an accelerometer to monitor the vibrations due to g-forces created by a projectile passing through the cardboard.

The inductive sensor acts as a metal detector using the principle of Lentz's Law to detect a change in a magnetic field due to a conductor moving through. For these calculations, the case of a .22 caliber bullet is used to calculate the worst

possible scenario: the bullet is simulated as a conductor 15mm long with a diameter of 5.5mm and the conductivity of lead 4.55 x10^6 S [13].

To determine the baseline magnetic field for the sensing circuit, the flux in the inductor B_0 can be calculated by the following equation, where n is the number of turns in the inductor, $\mu_0$is the permeability of free space, I is the initial current in the inductor prior to the projectile passing through in amps and r is the radius of the target in meters:

$$B_0 = \frac{n\mu_0 I}{2\pi r} \qquad \textit{Equation 6}$$

Using this baseline, we can apply Lentz's law; the change in power in an inductor due to a conductor moving through can be calculated by the following equation, where c is the length of the bullet in meters, R is the radius of the projectile in meters, w is the angular frequency in $\frac{rad}{sec}$ and σ is the conductivity of lead in Siemens [14]:

$$dP(t) = \frac{w^2\pi\sigma c B_0 (cos(wt))^2 R^3 dr}{2} \qquad \textit{Equation 7}$$

Integrating this equation from 0 to the radius of the projectile outputs the power change in watts of the system as shown below:

$$P(t) = (\tfrac{1}{8})w^2\pi\sigma c B_0 (cos(wt))^2 R^4 \qquad \textit{Equation 8}$$

Given that only the magnitude of power is important to the sensing circuit, the power in the system is demonstrated by:

$$|P| = (\tfrac{1}{8})) = (\tfrac{1}{8})w^2\pi\sigma c B_0 R^4 \qquad \textit{Equation 9}$$

Since there will be a constant voltage on the ring, the change in power will

appear as a change in current, and can be calculated as follows:

$$\Delta I = \frac{P}{V} \qquad \textit{Equation 10}$$

The following code was created using Matlab and was used to determine if an inductor could be used to act as a detector for a bullet moving through a target.

```
%_____Constants_____________________________
u0 = (4*pi)*10^-7;
sigma = 4.55e6; %conductivity of Lead
f = 60; %frequency (Hz)
w = 2*pi*f; %angular frequency (rad/sec)

%_____Variables_____________________________
r = .07;%radius of the target (m)
I = 1;%initial current in the inductor (A)
V = 12; %initial voltage in the inductor (VAC)
n = 300; %number of turns in inductor
L = .0057; %diameter of bullet (m)
dr = L/2; % radius of Bullet (m)
c = .015; % length of Bullet (m)

%_____CALCULATIONS_____________________________

B0 = (n*u0*I)/(2*pi*r); %calculate the flux in the ring
dP = (1/8)*(w^2)*pi*(dr^4)*(B0^2)*sigma*c
% should be a *cos(wt)^2, but we are focused on the abs mag
dI = dP/V
```

For proof of concept, the values for the variables were chosen as follows and plugged into the Matlab code above:

$$Radius\ of\ Target = 7cm$$
$$w = 2\pi(60Hz)$$
$$I = 1A$$
$$V = 12V$$
$$n = 300\ loops$$

Upon running the Matlab code, the program outputs an expected current change value of $13na$, which is deemed too small to be measured in any reasonable fashion. The variables in the Matlab code were altered to see what changes could be made to produce a reasonable change in current. Altering the frequency or number of turns produced the biggest effect in the system; however, the number of turns would have to be increased to more than one million or the frequency of the AC power would have to increase to 30 kHz. After some investigation into the plausibility of both of these options, both were deemed either too expensive or too complicated to be kept within the scope of the project.

The next possibility for sensing was to use an accelerometer to measure the acceleration and vibration in the target as a bullet passes through. The Work-Energy Principle can be used to calculate the local acceleration of the impact of the bullet.

First the work done by the bullet by the target must be calculated as follows, where m is mass in kilograms, V_{enter} is the velocity of the projectile in meters per second as it enters the target and V_{exit} is the exit velocity of the projectile in meters per second as it leaves the target:

$$W_{projectile} = \frac{1}{2} m({V_{enter}}^2 - {V_{exit}}^2) \qquad \textit{Equation 11}$$

In this equation, work done to the bullet is equal to the work done by the target. Given that the target has a measurable thickness, the work can be defined as the force applied over the thickness of the target as seen below, where m is the mass of the target in kilograms, d is the thickness of the target in meters and a is

the acceleration of the target in meters per second squared.

$$W = mad \qquad \textit{Equation 12}$$

The equation above can be manipulated to solve for the local acceleration and can be divided by the gravitational constant (9.81 meters per second^2) to find the G-forces seen upon impact.

The following code was created using Matlab and was used to determine if an accelerometer could be used to act as a detector for a bullet moving through a target.

```
m =0.0019; %mass of bullet (kg)
vi= 700;    %Entering velocity of bullet(m
vf= 699.5;  %exit velocity of bullet (m/s)
c= 1;    %thickness of target (cm)
de= 0.689;  %density of cardboard (g/cm^3)
r= 8;       %radius of the target (cm)

%_____________________Calculations_____________________________
a= pi*r^2; %target face area (cm^2)
mc_g = de*a*c; %mass of target (g)
mc_kg = mc_g/1000; % convert to kg

% Used the principle of Work and energy. by knowing the change in velocity
% through the target we can calculate the work done to the target by the
% bullet (W=.5m(vi^2-vf^2))

% work is also force over distance (W=fd=mad) so knowing how thick the
% target we can calculate the force applied to the target. and by knowing the
% mass of the target we can determine the acceleration acting on the target.

% dividing acceleration by 9.81 will give the G force. Below is the
% derived formula
t=(vi^2)-(vf^2);
W = (m*((vi^2)-(vf^2)))/2;
F =100*W/(c);
```

```
a=F/mc_kg;
g= a/9.81

%calculate torque on target pivot
Tq=F*2*r/100
```

For proof of concept, a .22 caliber bullet weighing 0.0019 kilograms striking a 1 centimeter thick cardboard target were assumed as the constants for this simulation. The average velocity of a .22 caliber bullet is 700 meters per second. It is assumed that the enter and exit velocity of a .22 caliber bullet differs by 0.5 meters per second, and the mass of the cardboard is estimated to be 0.1385 kilograms given the density of cardboard is 0.683 kilogram per centimeter^3 and the radius of the target is 8 centimeters[12]. From these estimations, the estimated local G-forces seen on the target max out around 50G, which is a measurable value. This G-force can be reduced further by increasing the size of the target.

There are pros and cons to each of the methods; one of the major pros to the inductive method was that it acted independently of the cardboard target, which meant more sessions could be run without having to change the cardboard target for fear the bullet passing through an existing hole. In addition, the inductive circuit would be highly accurate in correctly determining which target was hit, and would not falsely trigger the target hit command if the structure of the targeting system was hit. That said, its major drawback was that it generated extremely low current differentials, which works fine theoretically, but in practice could easily be disguised or hidden by the presence of electronic noise in the system. The accelerometer approach however, was much easier to fine tune to any

target design used, and methods were brought up to mitigate the effects of vibration interference when other targets were struck by projectiles.

Accelerometer Test [NH, TM, SV, TW]

An experiment was conducted to understand the realistic use and effectiveness of an accelerometer based sensor to detect hits on a cardboard target.

For the experiment three different calibers of bullets were used: a .22 cal CCF Standard Velocity 40 Grain fired from a Ruger Mark IV Competition firearm, a 9mm Blazer 115 Grain fired from a Sig P320 firearm and .40 cal Blazer 180 Grain fired from a H&K USP firearm.

An Analog Devices ADXL326 accelerometer was used as the hit detection Sensor. Based on the calculations above this sensor was chosen because it can detect up to 16g with about 57mV/g sensitivity. The sensor was attached to the target shaft using a binder clip. The sensor was fed 3V from an Agilent E3631A Power Supply and the signal was measured using an Agilent MSO6012A Mixed Signal Oscilloscope.

The distance for the target was arbitrarily chosen to be 20 feet from the marksman. Three different target setups were used. For the first setup a metal coat hanger was bent to act as the target shaft as seen bellow and the cardboard face was taped to the shaft.

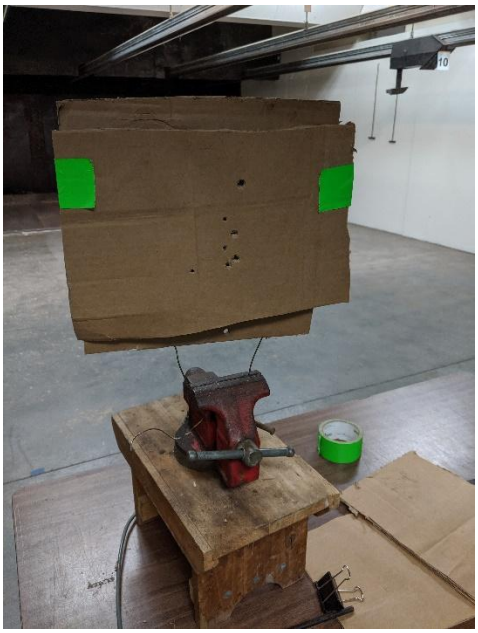


Figure 7: Test Fixture 1

The sensor was attached to the shaft near the clamp. Three data points were collected, one from each caliber bullet. And the magnitude of the voltage was collected and logged. The next test was conducted by moving the sensor so it was attached to the shaft at the base of the target. Again the voltage magnitude of three data points were collected, one from each caliber bullet.

The next test a rigid target holder was used and the target faced was taped to the target shaft as seen in the Test Fixture 2 figure below. The sensor was attached midway on the target shaft. Based on the results of the first few tests a worst case scenario of the .22 caliber round was tested first. Two data points were collected with this round, and then in the interest of the limited time on the firing range we moved on to the next test fixture.

Figure 8: Test Fixture 2

For the third test fixture the coat hanger was cut into 2 shafts which were placed through the cardboard target face in order to try and increase the vibration transfer between the target and target shaft as seen in the Test Fixture 3 figure below. For this test 2 data points were collected using the 9mm round before the allotted range time expired.

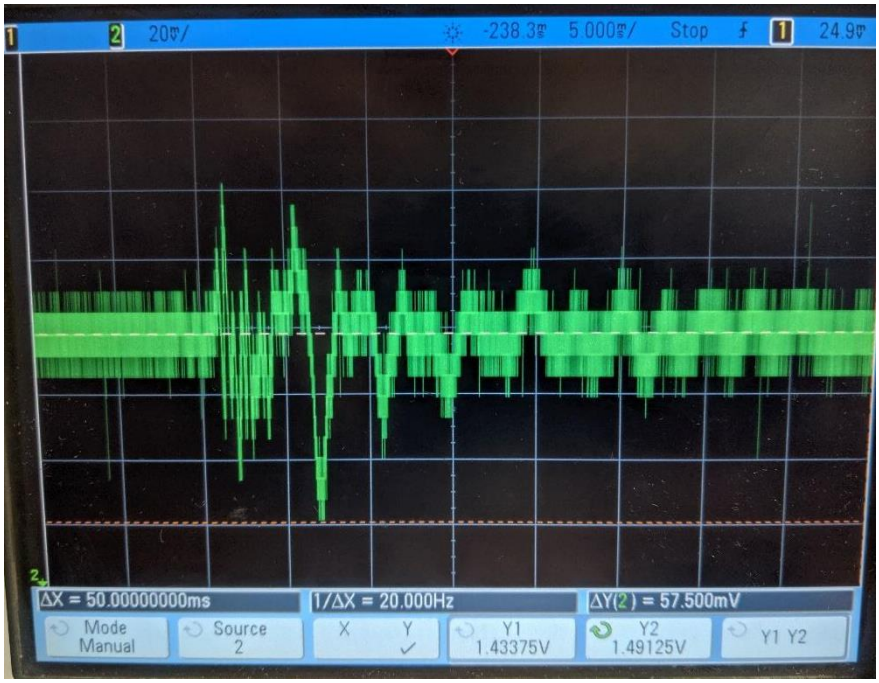


Figure 9: Test Fixture 1, Test 1, .22 cal round

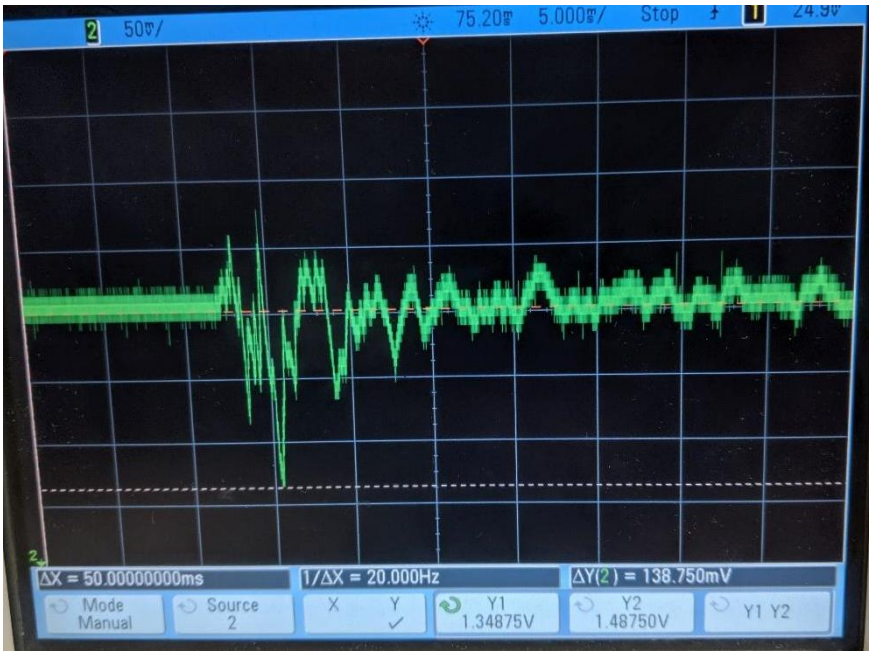


Figure 10: Test Fixture 1, Test 2, 9mm round

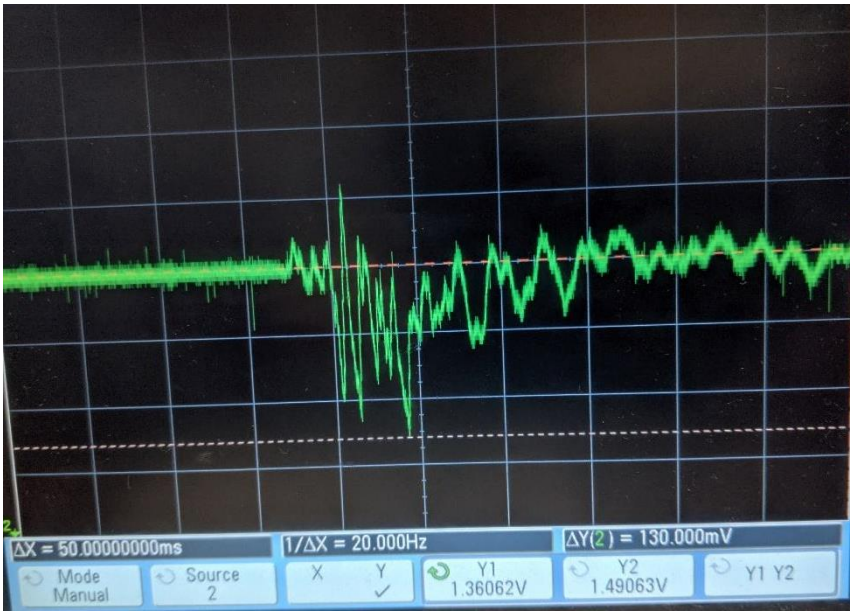


Figure 11: Test Fixture 1, Test 3, .40 cal round

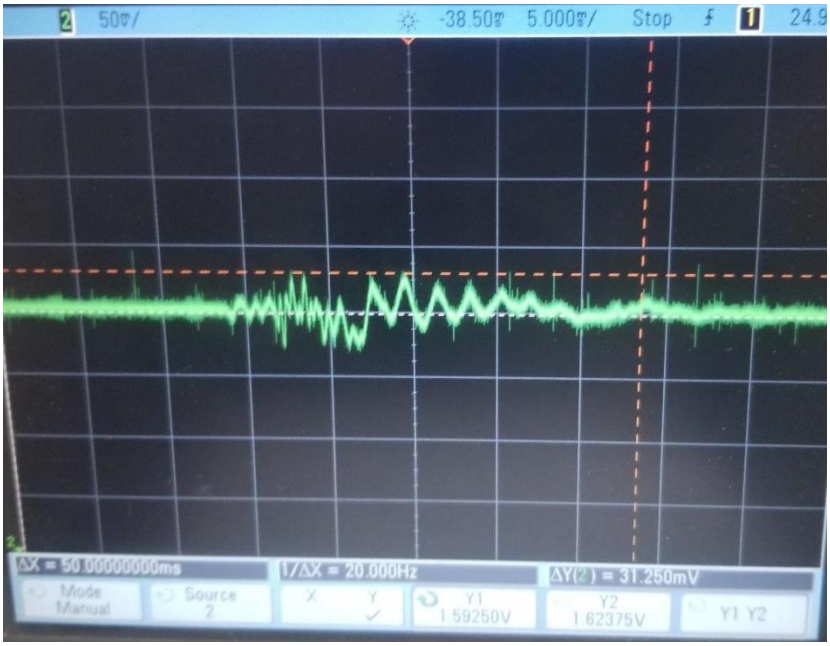


Figure 12: Test Fixture 1, Test 4, .22 cal round

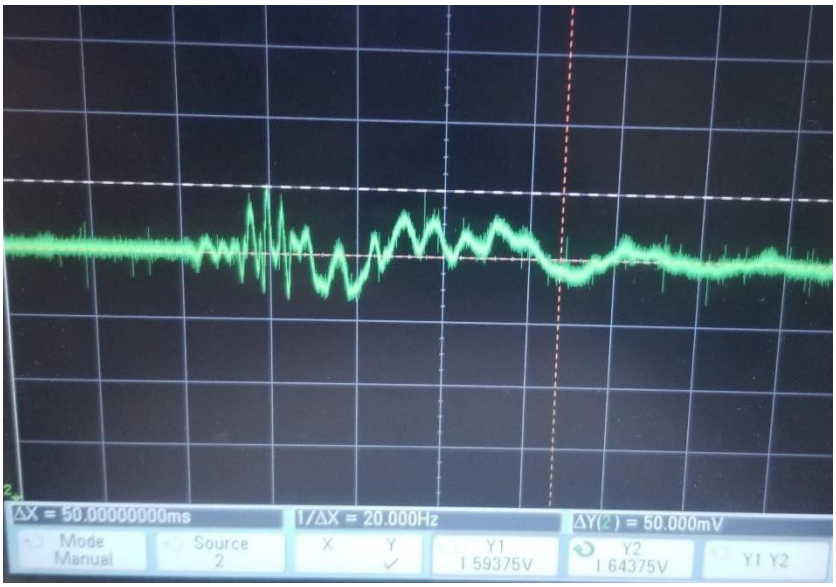


Figure 13: Test Fixture 1, Test 5, .22 cal round

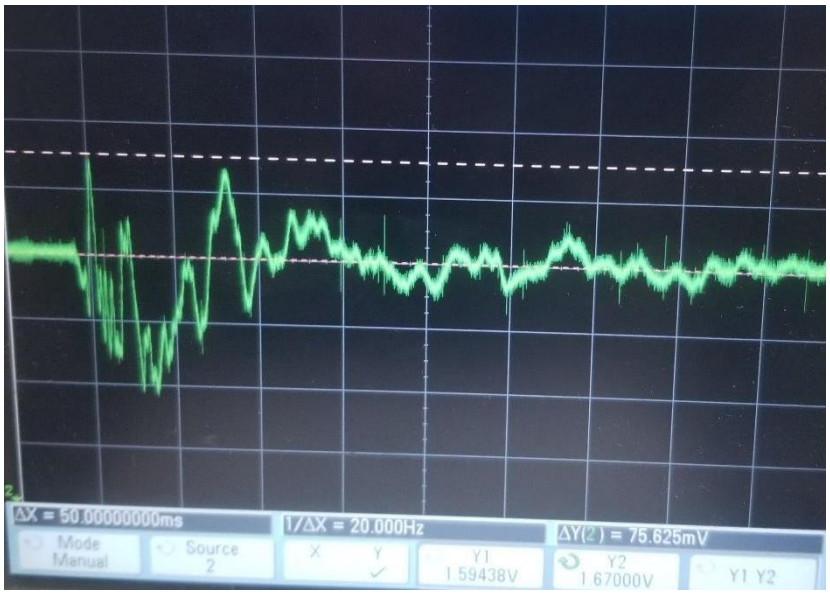


Figure 14: Test Fixture 1, Test 6, 9mm round

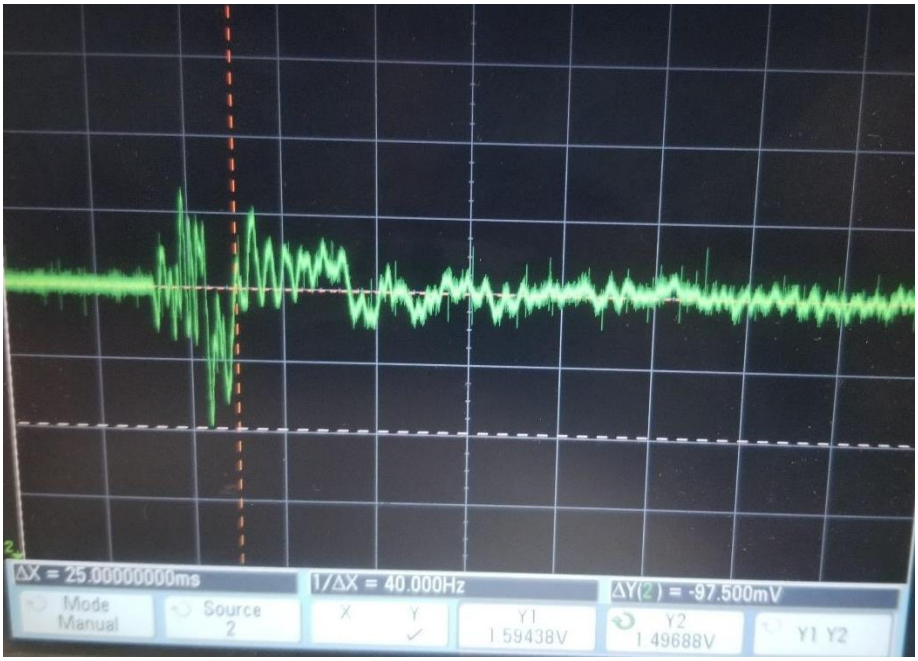


Figure 15: Test Fixture 1, Test 7, .40 cal round

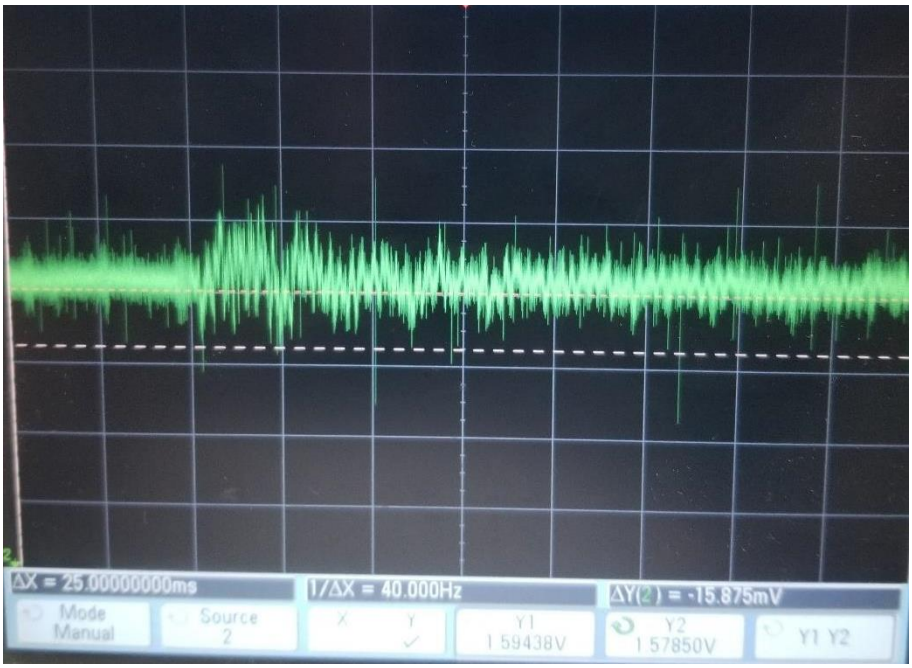


Figure 16: Test Fixture 2, Test 8, .22 cal round

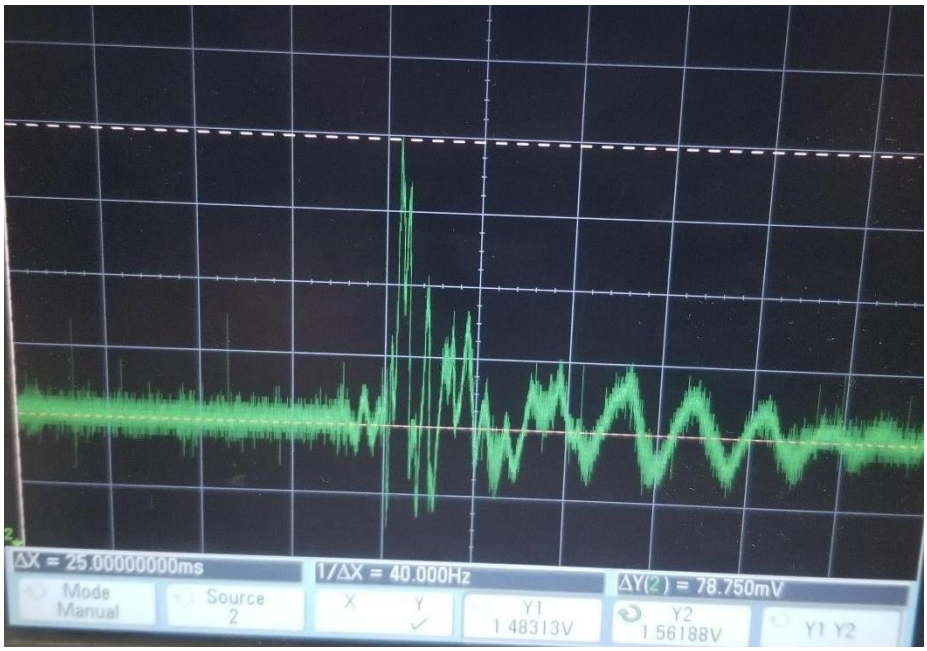


Figure 17: Test Fixture 2, Test 9, .22 cal round

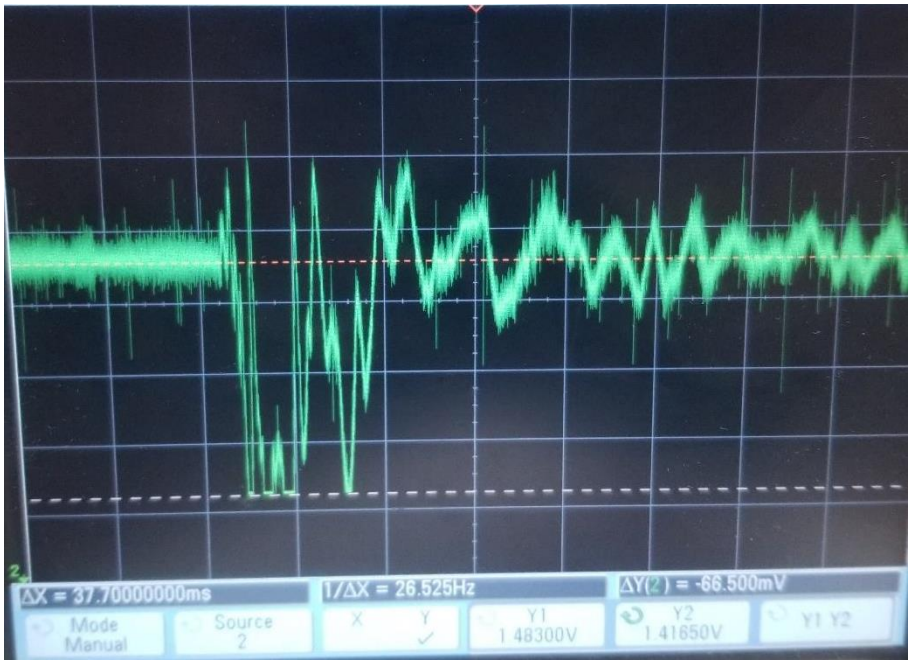


Figure 18: Test Fixture 3, Test 10, 9mm round.

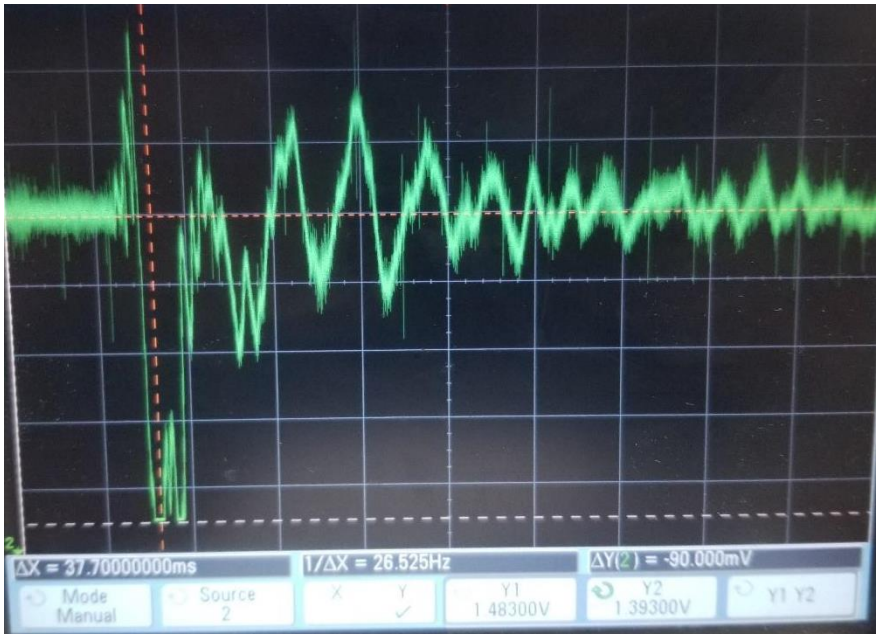


Figure 13: Test Fixture 3, Test 11, 9mm round

Test	Voltage (V)	g Forces (g)	Note
1	0.0574	1.01	Test Fixture 1, .22 cal round
2	0.1388	2.44	Test Fixture 1, 9mm round
3	0.1300	2.28	Test Fixture 1, .40 cal round
4	0.0500	0.88	Test Fixture 1, .22 cal round
5	0.0700	1.22	Test Fixture 1, .22 cal round
6	0.0756	1.33	Test Fixture 1, 9mm round
7	0.1147	2.01	Test Fixture 1, .40 cal round
8	0.0159	0.28	Test Fixture 2, .22 cal round
9	0.0787	1.38	Test Fixture 2, .22 cal round
10	0.1180	2.07	Test Fixture 3, 9mm round
11	0.090	1.58	Test Fixture 3, 9mm round

Table 1: g Forces Results

From this data it can be seen that the purchased sensor was too large. The expected g forces were approximately $1/10^{th}$ of the calculated values. It is estimated that the error in the calculations are unexpected losses in the elasticity of the materials used. From this data it can also be seen that the vibration

frequency caused by the impact is approximately 20 Hz. With this the rest of the sensing circuit can be designed.

Electronics

Power Consumption [NH, TM]

The power consumption of the system will be determined by two main features: the motors and the microcontroller. The power consumption of the Explorer 16/32 board that is being used for development is 1.6Ah. To find the power required to run the motors, a few assumptions need to be made. The calculations for the motors' power consumption are made on assumptions of use of the system. First the system needs to be run for three hours, where half of the time is spent in a mode where the motors are operational in 2 minute durations and the other half is spent idle. Based on the motor power calculations done in section 2.1.1 and assuming 12 volt motors will be used, the current is calculated as follows:

$$I = \frac{P}{V} = \frac{0.8652W}{12V} = 0.07235A\ per\ 90^{o}\ Motor\ Movement \qquad \textit{Equation 13}$$

In each two minute period, an estimated 150 motor movements occur. Multiplying the 150 motor movements by the current required to move a target results in the total current drawn during each two minute session. Then dividing this number by 30 to get the amp hours required per two minute session is shown below:

$$Ah_{per\ session} = 0.07235A * 150\ moves * \frac{1}{30} hr = 0.0368Ah \qquad \textit{Equation 14}$$

In three hours, there are 45 possible two minute sessions of system operation; by

multiplying the amp hour per session value by 45, the total amp hours needed to run the motors over the battery life can be calculated as follows:

$$Ah_{total} = Ah_{per\ session} * 45 = 16.28\ Ah \quad \text{Equation 15}$$

Since this number was estimated using values that were assumed to be extreme cases, a built in buffer exists. Between the two components of the system, a minimum of 17.88 Ah rechargeable energy supply is required to operate this system for three hours.

Mechanical Structure

Vibration Attenuation [NH]

The impact of a bullet on the cardboard will generate a vibration that will propagate through the system. For the sensing circuit to work properly the attenuation of vibration seen throughout the system needs to be negligible compared to the initial amplitude of the vibration. The calculation for vibration noise was done under the assumption that two sensors would be placed on a piece of metal and spaced 10 cm apart. This represents a worst case scenario. With a damping coefficient of metal assumed to be 5.51 dB/cm [15]. We can then assume that the relationship of the magnitude of the signals between the two sensors is as follows.

$$\frac{S_2}{S_1} = e^{-5.51\frac{dB}{cm}(10cm)} = 1.3x10^{-24} \quad \text{Equation 16}$$

This gain is on the order of 10^{-24} and thus it is safe to assume that the noise seen on other sensors in the system is negligible and can be filtered out by a adding a trigger level to not account for values under a desired amplitude.

Signal Processing

Analog Input Signals [TW, SV]

Signal Processing is necessary to convert the data collected in the actuating and sensing circuits in order to accurately determine the status of the circuit and respond accordingly. The actuating circuit, which uses a DC motor, gives feedback data to locate the current position of the target, and the sensing circuit, which uses accelerometers, gives feedback data in terms of G-force to help indicate when a target has been struck. In the case the accelerometer is used without an external analog-to-digital converter, the analog input signal going into the microcontroller must be converted from the accelerometer's native analog output to a digital variable that registers and stores a hit on the sensor. Supposing the accelerometer uses a 3.3V source, each input signal must be converted to either a 0 or a 1, depending on the threshold set for each analog input signal.

Analog Output Signals [SV, TW]

To dictate target position, an analog signal must be processed and outputted to the actuating circuit, in the case that a hardware controller does not convert the signal midway through. There will be two motor locations: 0 degrees and 90 degrees. The necessary voltage required to force the DC motor to change position will be outputted from the embedded controller depending on conditions met by the motor position and the target hit variables.

Digital Input Signals [SV, TW]

Digital Input signals can be received from both the actuating and sensing circuits, depending on the specific hardware used. In the case that digital inputs are available for the microcontroller, motor location, motor location feedback, and hit on target will be inputted to the microcontroller and integrated in the software as boolean variables; this allows all major components of the subsystems to be controlled and monitored with digital variables exclusively.

Digital Output Signals [SV, TW]

Depending on the specific hardware and analog controller(s) used, a Digital Output signal can be outputted by the microcontroller to dictate the desired motor location; this allows the microcontroller to output either a 0 or 1, and the controller that is being used in between the microcontroller and the actuating/sensing circuits can convert the signal from the digital signal that the microcontroller is outputting to a format or protocol that the specific hardware can understand.

Communications

Bluetooth [SV,TW]

In order to allow communication between the microcontroller and the user interface, a plug in Bluetooth module using the UART protocol will be used. Bluetooth allows for the user to be physically away from the device while still being able to communicate seamlessly. A single user will be able to connect directly to the embedded controller without any intermediary devices, and will receive the control and feedback data that is defined in the controller to be outputted to the user.

Embedded Systems

High level Embedded System Logic [SV, TW]

The logic of the embedded system depends on three arrays that hold the real-time locations of the motors, the instantaneous target hit detection, and hit detection over a session.

// startup, commence UART communication to user,

start timer for 1 minute

rand var to distribute targets - create two possible options and randChoose options

motorLocationArray_everything = above pseudocode

latchReset(everything) to clear caps

// latchReset for accel for muxControl

if (motorLocation is active), send a HIGH signal to accel to unlatch the sensor readings FOR .1?? second.

// hit Detect for accel

if (motorLocation is active && hitDetect from sensingController is HIGH)

change TargetHit

trigger latchResetForAccel()

// sensitivity Pot for sensingController

if (depend on user){

send DO to triggerLevel to sensingController

}

// motorLocationfeedback from actuatingController

this is a get function, HIGH means active target

store in array, this array will be referenced in loop

// motorMove for actuatingController

if(motorLocation = HIGH, && hitDetect = HIGH), send to actuatingController an idle(LOW)

else if (motorLocation = LOW && hitDetect == HIGH), latchReset();

else if (motorLocation = HIGH, && hitDetect == LOW), keep polling hitDetect and motorLocation.

else if (motorLocation = LOW && hitDetect == LOW && targetHit_parallel = HIGH), send actuatingController a HIGH.

// loop

if(motorLocationArray_11 is HIGH) keep polling hitDetect_11

else if(motorLocationArray_11 is LOW) keep polling hitDetect_12

if(motorLocationArray_21 is HIGH) keep polling hitDetect_21

else if (motorLocationArray_21 is LOW) keep polling hitDetect_22

if(motorLocationArray_31 is HIGH) keep polling hitDetect_31

else if (motorLocationArray_31 is LOW) keep polling hitDetect_32

// duelingTreeEvent END

after 1 minute, copy motorLocationArray_everything copy to final outputArray and then motorLocationArray_everything send LOW

check real time clock calendar, trigger ISR

// user defined ISR

user is shooting target -- normal operation

user hits target --- triggers this ISR

ISR ops:

the target, which was hit, is retracted --- increment hitStore[] and latchReset()

the parallel target, which will be in use of the opponent, is displayed

the ES sends UART data to the interface

update data structures

end ISR;

user shoots targets...

Universal Asynchronous Receiver/Transmitter (UART) [SV, TW]

The DART system requires the user interaction to ensure system mode and sensitivity are set according to the user's desired preferences, and also displays feedback from the DART system regarding target hits. In order to accomplish interaction from a safe distance reliably, the Universal Asynchronous Receiver/Transmitter protocol was chosen to establish communication between the standalone DART system and the user through Bluetooth Low Energy. The Explorer 16/32 board, which houses the main embedded components, has an expansion board, namely the RN4871 MikroBus extension board. The Explorer 16/32 and the RN4871 act as one unit and allow the system developer to assign pin outputs from the main Explorer board to the RN4871.

There lies one main caveat and limitation with the RN4871; its communication settings must align perfectly with the outputted signal from the Explorer board. Because of minute differences caused by propagation delays, the baud rate generation must match within a few percent error margins. Since the default clock speed of the Explorer board did not allow for pristine baud rate generation, a PLL of 4x was chosen and implemented, which is configurable through the Oscillator Source Selection config **FNOSC** through #pragma. This sets the oscillator to the PLL instead of the Oscillator Divider. Then, the PLL mode config must be set to PLL4x in order to generate a clock speed where the desired baud rate can be achieved. According to the RN4871 configuration settings, the default baud rate is 115200; to output a signal from the Explorer

board with the default baud rate requires the use of BRGH. The BRG (baud rate generator) value is determined by

$$Baud\ Rate = \frac{Fcy}{4 * (UBRGx + 1)} \qquad U2BRG = \frac{16M}{4 * 115200} - 1 \qquad \textit{Equation 18}$$

This calculation results in a value of 34, and requires the mode of the BRG to be set to BRGH=1. After accurately setting the baud rate settings, the data transmission settings need to match the RN4871's default settings of 8 data bits, 1 stop bit, no flow control, and no parity. These are set by the U2MODE register to a value of 0x8008.

To transmit a character, the UART communication must be initiated using the settings configured above, and each cycle of transmission requires a handshake to determine which party is transmitting/receiving. When a handshake signal allows a Clear-To-Send, the CTS signal is asserted LOW, and characters are sent individually into the U2TXREG, which the RN4871 sends to the user over Bluetooth low energy. To end transmission, the CTS signal is asserted HIGH.

To receive a character, the same UART configuration settings must be utilized, but the RTS signal is used in place of CTS, and the U2RXREG buffer register is filled with characters sent from the user. When the buffer register is full, the RTS signal reverts to its idle HIGH state.

Engineering Requirement Specification

Marketing Requirement	Engineering Requirement	Justification
2, 6	The D.A.R.T system needs to have a rechargeable energy storage device that can operate the system for a minimum of 3 hours.	The system needs to be able to last through a standard practice time
1, 5, 6	The D.A.R.T system's mechanical structure needs to include 2 columns with 3 rows, totaling 6 targets.	The system will be modeled off of a dueling tree design which will allow for the most versatile use of the system.
4	One motor movement of the target subsystem needs to be completed within 0.5 seconds.	The reaction of the system should be done quicker than the reaction of the marksman.
4	The D.A.R.T system needs to be able to sense an impact on the target within 0.5 seconds.	The reaction of the system should be done quicker than the reaction of the marksman.
2, 3	The D.A.R.T system needs to be able to communicate wirelessly with the user within a standard 50 foot firing range	The system needs to be able to communicate to the user within a standard indoor firing range.
3, 4	The user of the D.A.R.T system needs to be able to see a hit on target on the interface within 1 second.	The system needs to display data about its usage to the user and bystanders in a reasonable amount of time.
1, 2	The D.A.R.T system needs to be structurally able to withstand an impact of 50 G-forces.	The structure needs to be stable and able to withstand stray shots without toppling over.
2, 5	The D.A.R.T System sensing circuit needs to be able to be tuned to detect impacts from at least 4 different projectiles.	The system needs to be firearm agnostic in order to be the most effective.
5	The D.A.R.T system's sensing circuit needs to be able to sense a hit on target from 8cm to 20cm diameter.	The sensing system needs to be able to detect on a multitude of standard target sizes.
3, 4	The D.A.R.T system's user interface needs to be display the statistics for each mode, such as length of time per run and total target hits per session.	The app needs to be able to display real-time control and feedback data to the user. It should also provide some type of standardized statistics to easily compare sessions.

Table 2: Breakdown the Engineering Requirements of the D.A.R.T System.

[NH, TM, SV, TW]

Engineering Standards Specifications

	Standard	Description
Safety	DOE: Firearm Range Design	The standards set by the Department of Energy for firearm range design will be used to safely design the target system to minimize the chance of ricochet. [17]
Communication	UART BLE	Bluetooth allows for local communication within 50 feet, which is the expected distance of an indoor firing range.
Design Methods	IEEE std. 506-2007	The method for routing cable throughout the system will follow the standards set forth in IEEE std 506-2007. [16]
Programming Languages	C, Kotlin	The C Language allows for programming of the Explorer 16/32 board used for development. Kotlin is the programming language used for the Android app development

Table 3: Design Standards Table

[NH,TM,SV,TW]

Accepted Technical Design

System Block Diagram

System Level Zero Block Diagram [NH]

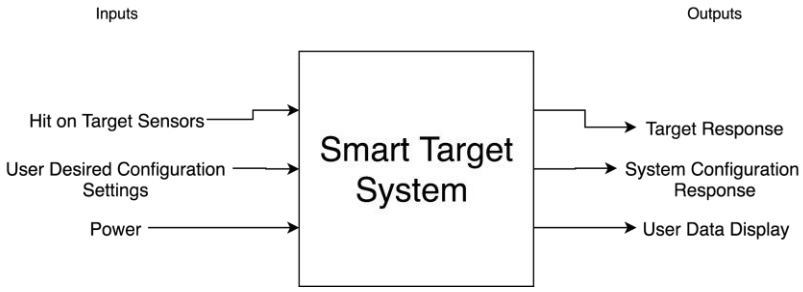


Figure 19: Zero Level Block Diagram showing the fundamental inputs and outputs of the system.

Module	D.A.R.T System
Designers	Nicholas Haas, Trandon Ware, Sai Vellala, Thomas Martin
Inputs	Power, Hit on Target sensor- a sensing system to tell when a target has been struck, User Desired Configuration Settings- To control certain aspects of the system like sensitivity, lighting, speed etc.
Outputs	Target Response- the target system should react to certain stimulus (i.e. the motor should go up or down), Configuration Response- the settings of the system should chance when inputs are changed User Data Display- The data from the user session should be displayed for user viewing
Description	The system will act as a target system that will allow the user to enter different training modes. When the target is hit there will be a response based on the training mode. This could be lowering one target while raising up another.

Table 4: System Level One Block Diagram Breakdown Table.

System Level One Block Diagram [NH]

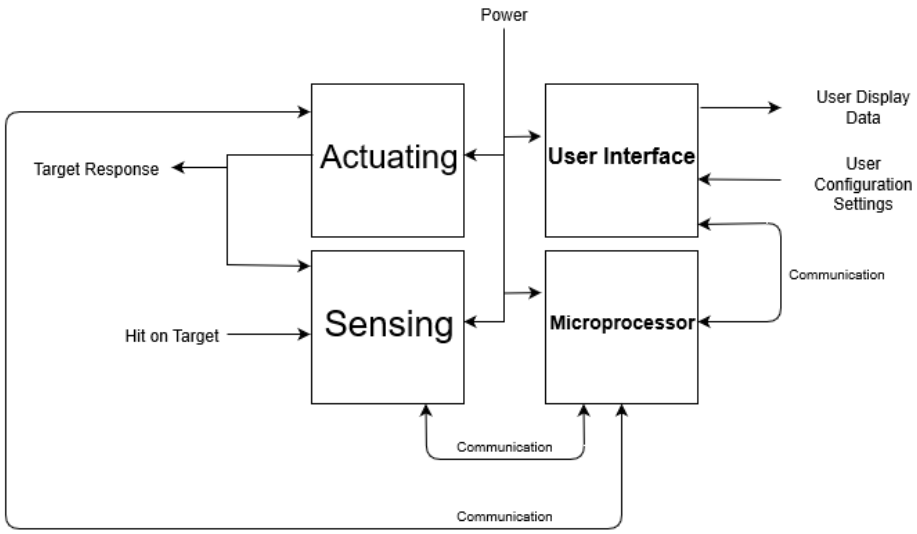


Figure 20: System Level Two Block Diagram for the sub-category breakdown of the system.

Module	D.A.R.T System
Designers	Nicholas Haas, Trandon Ware, Sai Vellala, Thomas Martin
Inputs	Power, Hit on Target sensor- a sensing system to tell when a target has been struck, User Desired Configuration Settings- To control certain aspects of the system like sensitivity, lighting, speed etc.
Outputs	Target Response- the target system should react to certain stimulus (i.e. the motor should go up or down), Configuration Response- the settings of the system should change when inputs are changed User Data Display- The data from the user session should be displayed for user viewing
Description	The system will act as a target system that will allow the user to enter different training modes. When the target is hit there will be a response based on the training mode. This could be lowering one target while raising up another.

Table 5: System Level Two Block Diagram Breakdown Table.

Each category was broken down into their own block diagrams and displaced in the following sections.

Hardware Block Diagrams

Actuating Level One Block Diagram [TM]

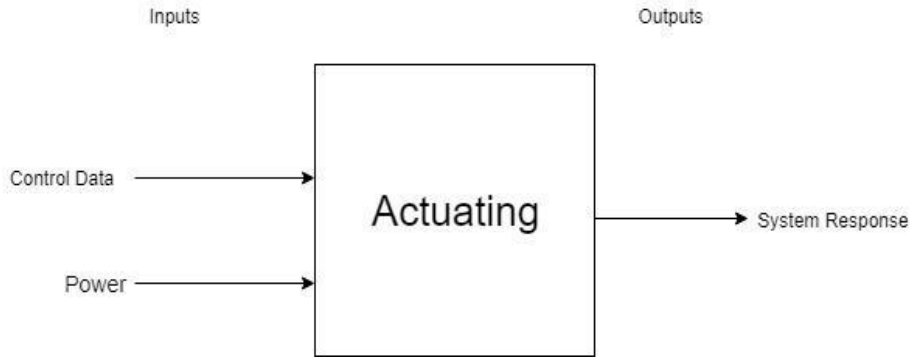


Figure 21: Actuating System Level One Block Diagram

Module	Actuating
Designers	Thomas Martin
Inputs	Power, Control data
Outputs	System Response
Description	The block will read in the control data from the microcontroller and then will use an actuating circuit to make the targets respond in the desired way

Table 6: Actuating Level One Block Diagram Breakdown Table.

Actuating Level Two Block Diagram [TM]

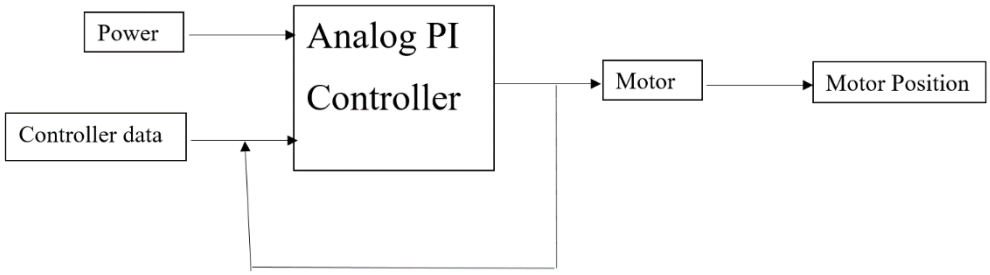


Figure 22: Actuating System Level Two Block Diagram

Module	Actuating
Designer	Thomas Martin
Inputs	Power, Controller Data, Error
Outputs	Motor Position
Description	This analog PI controller will be used to take in an analog output from the controller, and compare it with the voltage of the motor. This controller must be able to compare the position of the motor to the desired position as stated by the controller.

Table 7: Actuating Level Two Block Diagram Breakdown Table

Actuating Level Three Diagrams [TM]

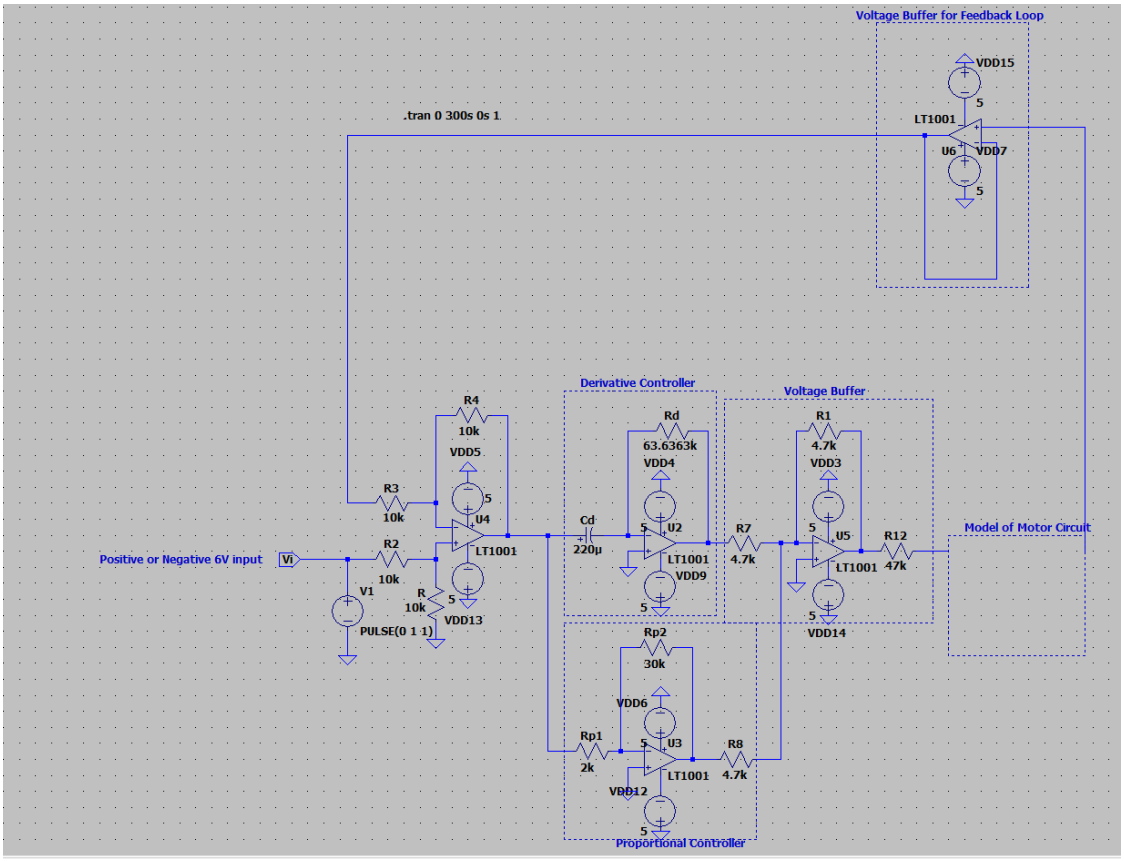


Figure 23: Schematic of PD controller in LTSpice

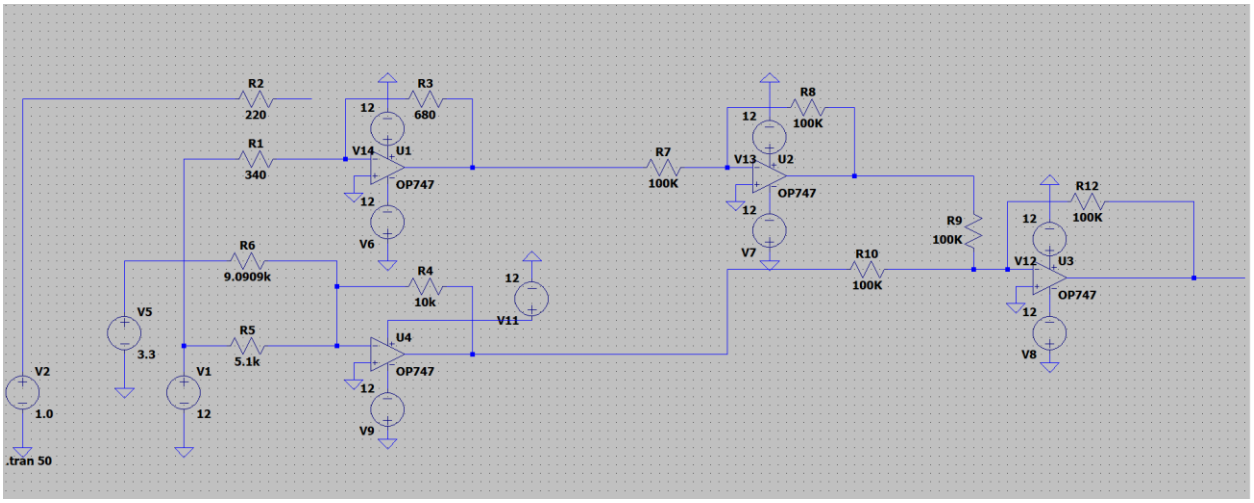


Figure 24: Schematic of the Input Voltage Level Converter circuit in LTSpice

Sensing Level One Block Diagram [NH]

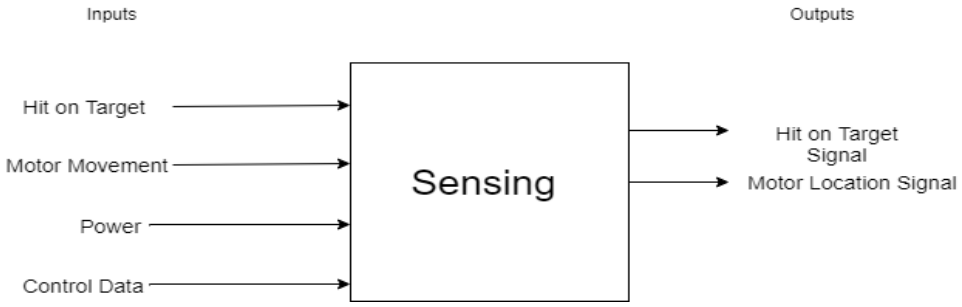


Figure 25: Level One Sensing Block Diagram

Module	Sensing
Designers	Nicholas Haas
Inputs	Power, Physical Hit on Target, Motor Movement, Control data
Outputs	Hit Signal, Motor Location Signal
Description	There sensing circuit will need detect a hit on the target and the location of the motors at a given point in time. The data collected will need to be communicated to the microprocessor and sensor control data will need to be read from the microprocessor.

Table 8: Sensing Level One Block Diagram Overview

Sensing Level Two Block Diagram [NH]

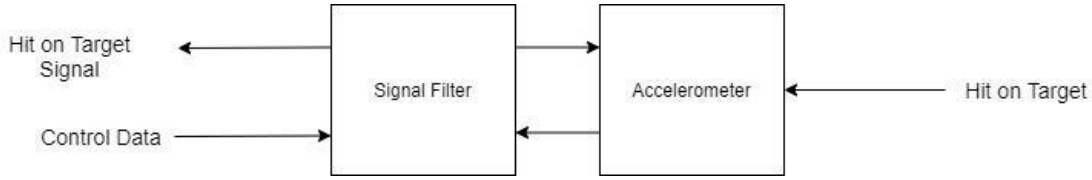


Figure 26: Sensing Level Two Block Diagram

Module	Sensing
Designers	Nicholas Haas
Inputs	Power, Physical Hit on Target, Control data
Outputs	Hit Signal, Motor Location Signal
Description	There sensing circuits will need detect a hit on the target using an accelerometer at a given point in time. The data collected will need to be filtered such that when a hit occurs a positive voltage is sent back to the controller, otherwise there is zero volts going to the processor. Sensor control data will need to be read from the microprocessor and be used to set the trigger level of the circuit and reset the circuit.

Table 9: Sensing Level Two Block Diagram Breakdown Table.

Sensing Level Three Block Diagram [NH]

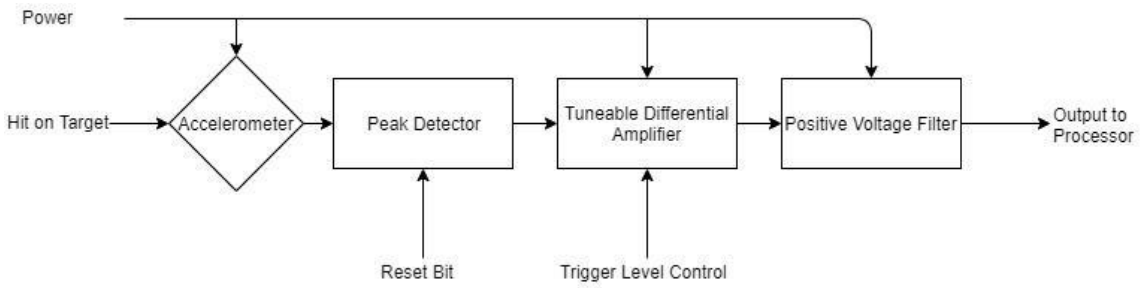


Figure 27: Sensing Level Three Block Diagram

Module	Accelerometer Sensing
Designers	Nicholas Haas
Inputs	Power, Physical Hit on Target, Reset, Trigger Level Control
Outputs	Output to Processor
Description	There sensing circuits will need detect a hit on the target using an accelerometer and be able to process the signal from the sensor by running the data through a series of electrical circuits that will output a high voltage when the target is hit until the reset signal is sent. The Trigger Level Control will be used to set the sensitivity of the sensing circuit to trigger when a vibration above a desired amount is detected.

Table 10: Sensing Level Three Block Diagram Breakdown Table.

Sensing Level Four Block Diagram [NH]

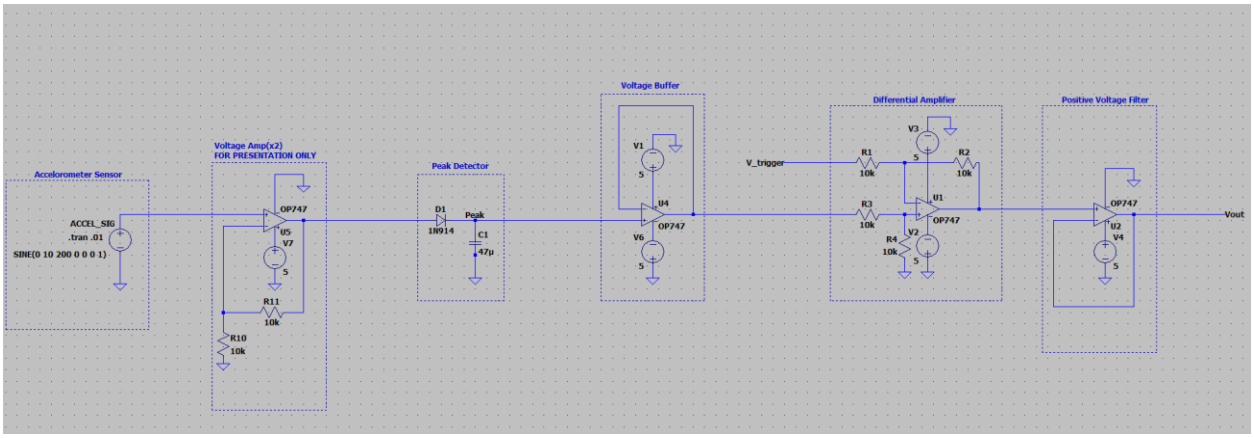


Figure 28: Sensing Level Four Block Diagram, Sensing Filter Schematic

Qty.	Refdes	Part Num.	Description
1	UC1	EVAL-ADXL326Z	Small, Low power, 3 axis, +/- 16 g Accelerometer
1	U1	UA747CN	LPD OP AMP
6	R1,R2,R3,R4,R5,R6	691104	10K 0.25W Resistor
1	C1	1946261	47uf Capacitor
1	D1, D2	1N4004	Diode
2	U2,U3,U4	LM358	LPD OP AMP

Table 11: Parts List for Sensing Filter Circuit

Module	Accelerometer Sensing
Designers	Nicholas Haas
Inputs	+5V/-5V power Rails, 3.3V sensor Power, Physical Hit on Target, Trigger Level Control
Outputs	Hit Signal
Description	There sensing circuits will need vibration using an accelerometer and be able to process the signal from the sensor by running the data through a series of electrical circuits that will output a positive voltage when the amplitude of vibration goes above the trigger level. The Trigger Level Control will be used to set the sensitivity of the sensing circuit to trigger when a vibration above a desired amount is detected.

Table 12: Sensing Level Four Block Diagram Breakdown Table.

Sensing Level Five Block Diagram [NH]

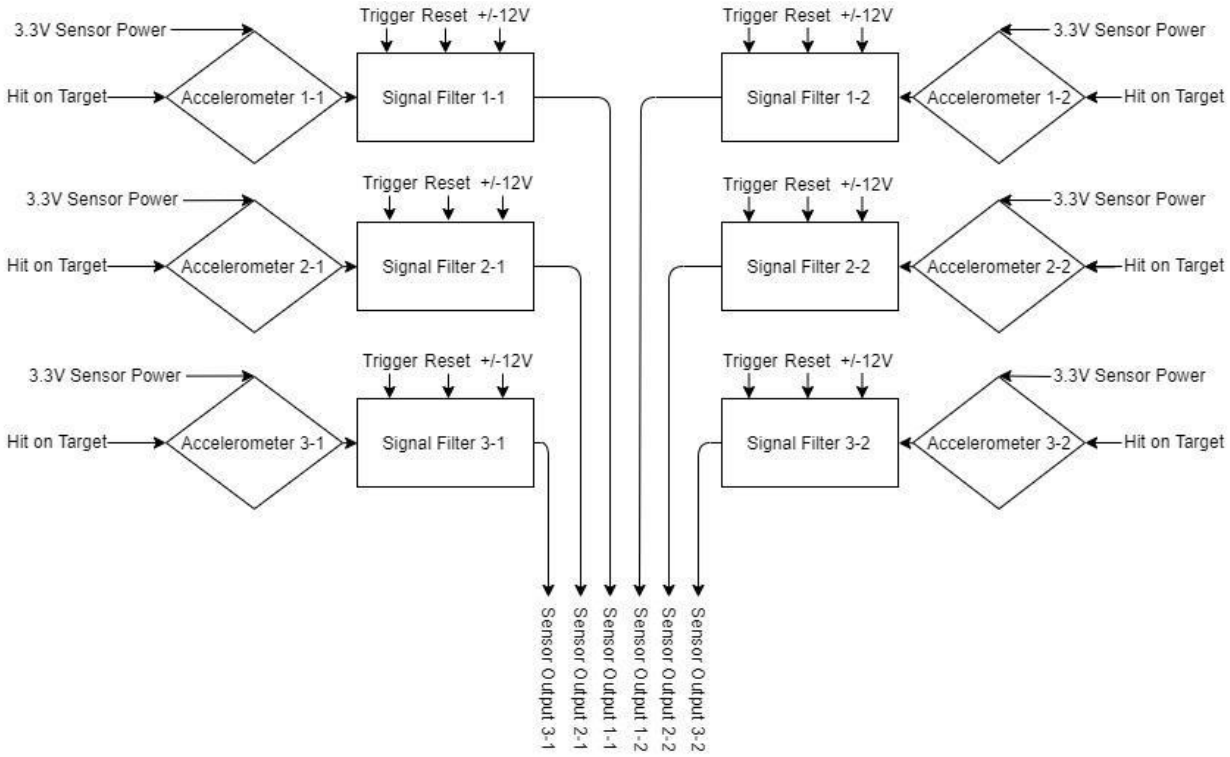


Figure 29: Sensing Level Five Block Diagram for the overview of the system wide the sensing circuit.

Module	System Hit on Target Sensing
Designers	Nicholas Haas
Inputs	3.3V, +/- 12V , Physical Hit on Target (x6), Reset (x6), Trigger Level Control (x6)
Outputs	Sensor Output (x6)
Description	The sensing system will need to be able to filter 6 separate target sensors and properly output correct values to the processor given the proper input signals of reset and trigger control to each sub filter.

Table 13: Sensing Level Five Block Diagram Breakdown

Sensing Level Six Block Diagrams [NH]

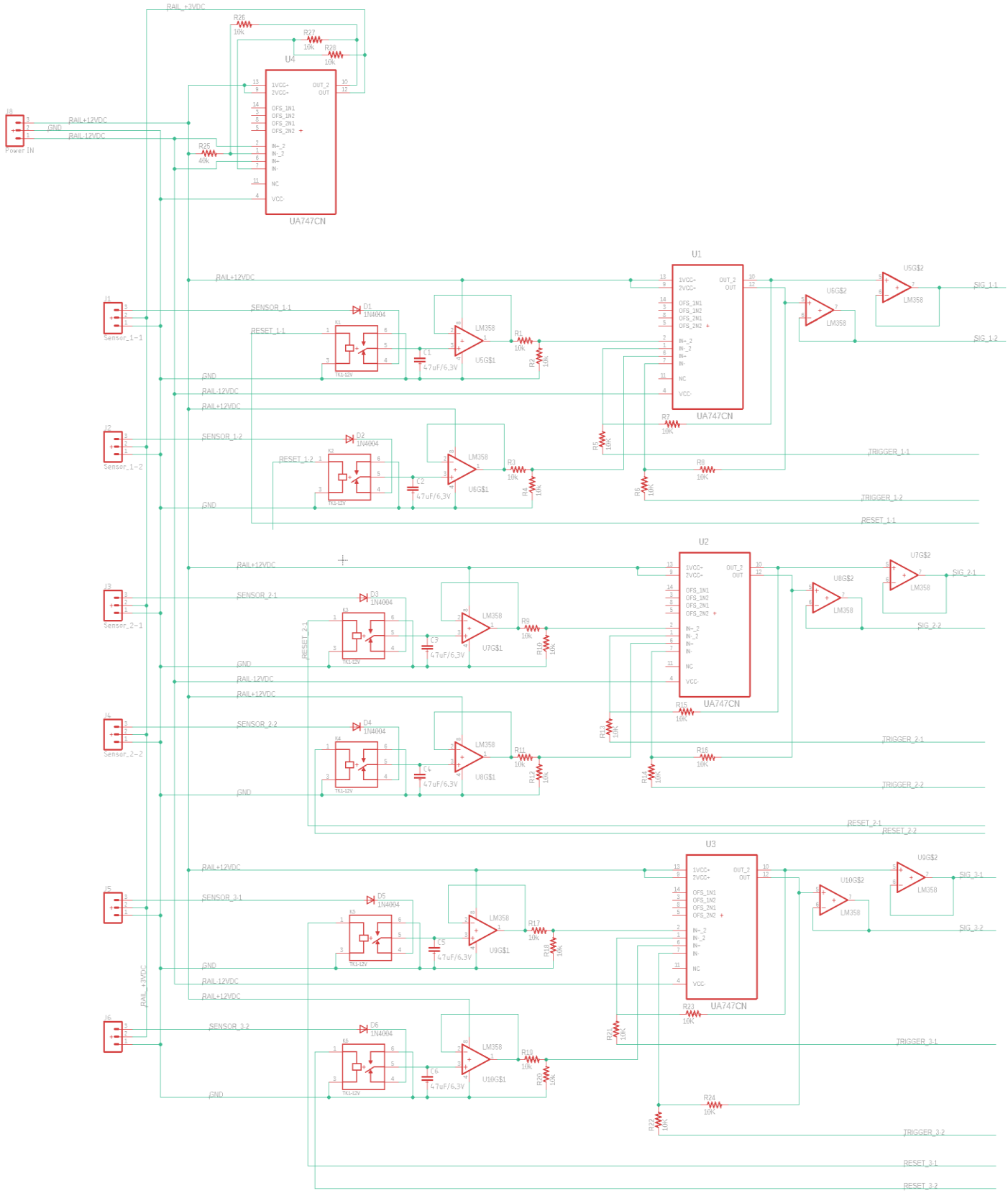


Figure 30: Sensing Level Six Block Diagram, Whole System Sensing Circuit

Qty.	Refdes	Part Num.	Description
4	U1,U2,U3,U4	UA747CN	LPD OP AMP
27	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28	691104	10K 0.25W Resistor
6	K1, K2, K3, K4, K5, K6	TK1-3V	RELAY GEN PURPOSE SPDT 2A 3VDC
6	C1, C2, C3, C4, C5, C6	1946261	47uf Capacitor
6	D1,D2, D3, D4, D5, D6	1N4004	Diode
6	U5, U6, U7, U8, U9, U10	LM358	LPD OP AMP
6	J1, J2, J3, J4, J5, J6	ADXL325BCPZ-RL8	ACCELEROMETER 5G ANALOG 16LFCSP

Table 14: Parts List for Whole System Sensing Circuit

IO Connector Pinout	Description
1	Sensor 1-1 Signal out
2	Sensor 1-2 Signal out
3	Sensor 2-1 Signal out
4	Sensor 2-2 Signal out
5	Sensor 3-1 Signal out
6	Sensor 3-2 Signal out
7	Trigger Voltage 1-1 Signal in
8	Trigger Voltage 1-2 Signal in
9	Trigger Voltage 2-1 Signal in
10	Trigger Voltage 2-2 Signal in
11	Trigger Voltage 3-1 Signal in
12	Trigger Voltage 3-2 Signal in
13	Reset 1-1 Signal in
14	Reset 1-2 Signal in
15	Reset 2-1 Signal in
16	Reset 2-2 Signal in
17	Reset 3-1 Signal in
18	Reset 3-2 Signal in

Table 15: Pinout for the IO coming to and from the controller

Module	Accelerometer Sensing
Designers	Nicholas Haas
Inputs	+5V/-5V power Rails, 3.3V sensor Power, Physical Hit on Target, Trigger Level Control
Outputs	Hit Signal
Description	There sensing circuits will need vibration using an accelerometer and be able to process the signal from the sensor by running the data through a series of electrical circuits that will output a positive voltage when the amplitude of vibration goes above the trigger level. The Trigger Level Control will be used to set the sensitivity of the sensing circuit to trigger when a vibration above a desired amount is detected.

Table 16: Sensing Level Four Block Diagram Breakdown Table.

The following schematics will be a zoomed on the different elements in the Level Six Block Diagram. Starting with the Power Converter followed by the sensing circuits for the three individual rows of targets.

Software Block Diagrams

Microprocessor Level One Block Diagram [TW]

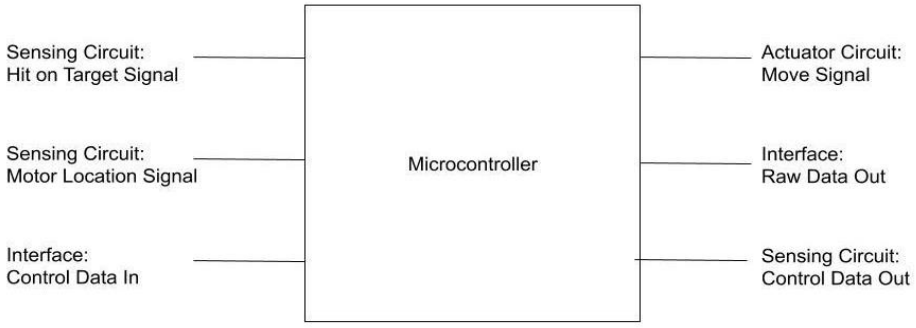


Figure 31: Microprocessor Level Two Block Diagram

Module	Microcontroller
Designers	Sai Vellala, Trandon Ware
Inputs	Power, Target Hit Sensor - the sensing circuit outputs a signal when a target has been struck Motor Location Sensor - the sensing circuit outputs a signal detailing where each motor in the system is located
Outputs	Actuator Move Signal - the microcontroller calculates when the motor should move a target and will send a signal to move the target Raw Data Out - the microcontroller sends all the raw data collected from the system to the interface
Description	The microcontroller is the component that controls every other subsystem; it receives real-time data defining target hits and locations, but returns data to the actuator regarding which location each target should be located at. It also sends all data received from the circuits to the interface. The microcontroller receives control data from the interface which is processed and sent to the sensing circuit.

Table 17: Microcontroller Level One Block Diagram Breakdown.

Microprocessor Level Two Block Diagram [TW]

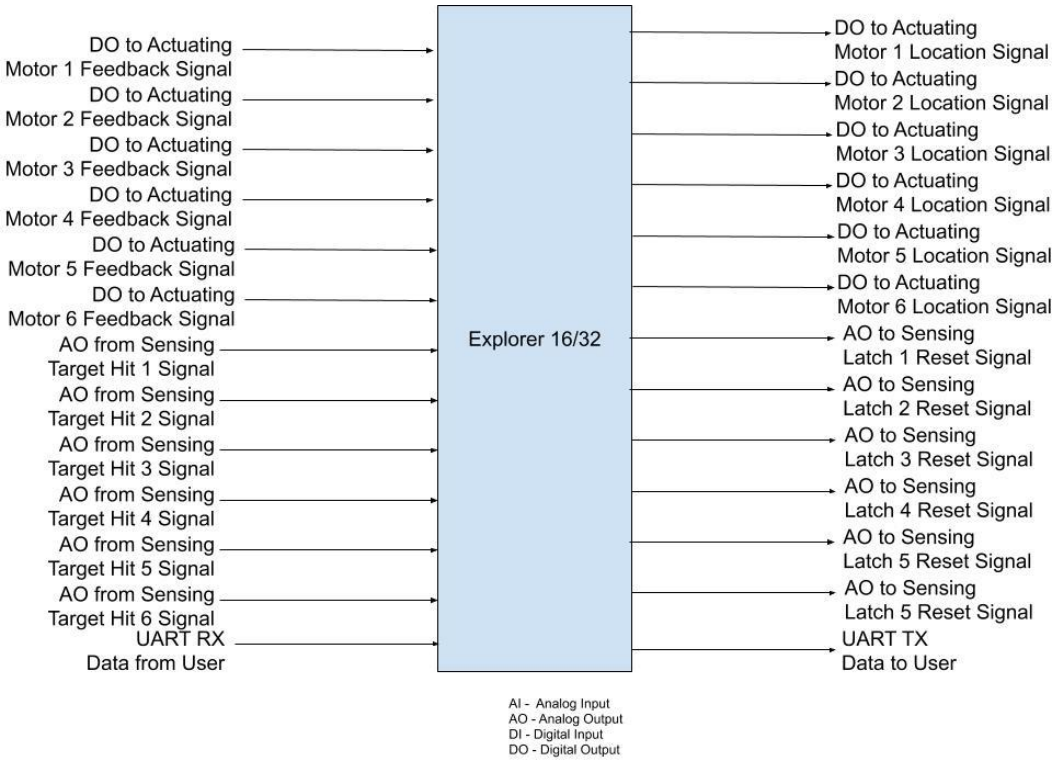


Figure 32: Microprocessor Level Two Block Diagram

Module	Microcontroller
Designers	Sai Vellala, Trandon Ware
Inputs	Power, DO to Actuating Motor 1 Signal, DO to Actuating Motor 2 signal, DO to Actuating Motor 3 Signal, DO to Actuating Motor 4 Signal, DO to Actuating Motor 5 Signal, DO to Actuating Motor 6 Signal, AO From Sensing Target Hit 1 Signal, AO From Sensing Target Hit 2 Signal, AO From Sensing Target Hit 3 Signal, AO From Sensing Target Hit 4 Signal, AO From Sensing Target Hit 5 Signal, AO From Sensing Target Hit 6 Signal
Outputs	AO to Actuating Motor 1 Location, AO to Actuating Motor 2 Location, AO to Actuating Motor 3 Location, AO to Actuating Motor 4 Location, AO to Actuating Motor 5 Location, AO to Actuating Motor 6 Location, AO to Sensing Latch 1 Reset Signal, AO to Sensing Latch 2 Reset Signal, AO to Sensing Latch 3 Reset Signal, AO to Sensing Latch 4 Reset Signal, AO to Sensing Latch 5 Reset Signal, AO to Sensing Latch 6 Reset Signal, UART TX Data to User
Description	The microcontroller is the component that controls every other subsystem; it receives real-time data defining target hits and locations, but returns data to the actuator regarding which location each target should be located at. It also sends all data received from the circuits to the interface. The microcontroller receives control data from the interface which is processed and sent to the sensing circuit.

Table 18: Microprocessor Level 2 Block Diagram Overview

Microprocessor Level Three Flowchart for I/O Simulation [TW]

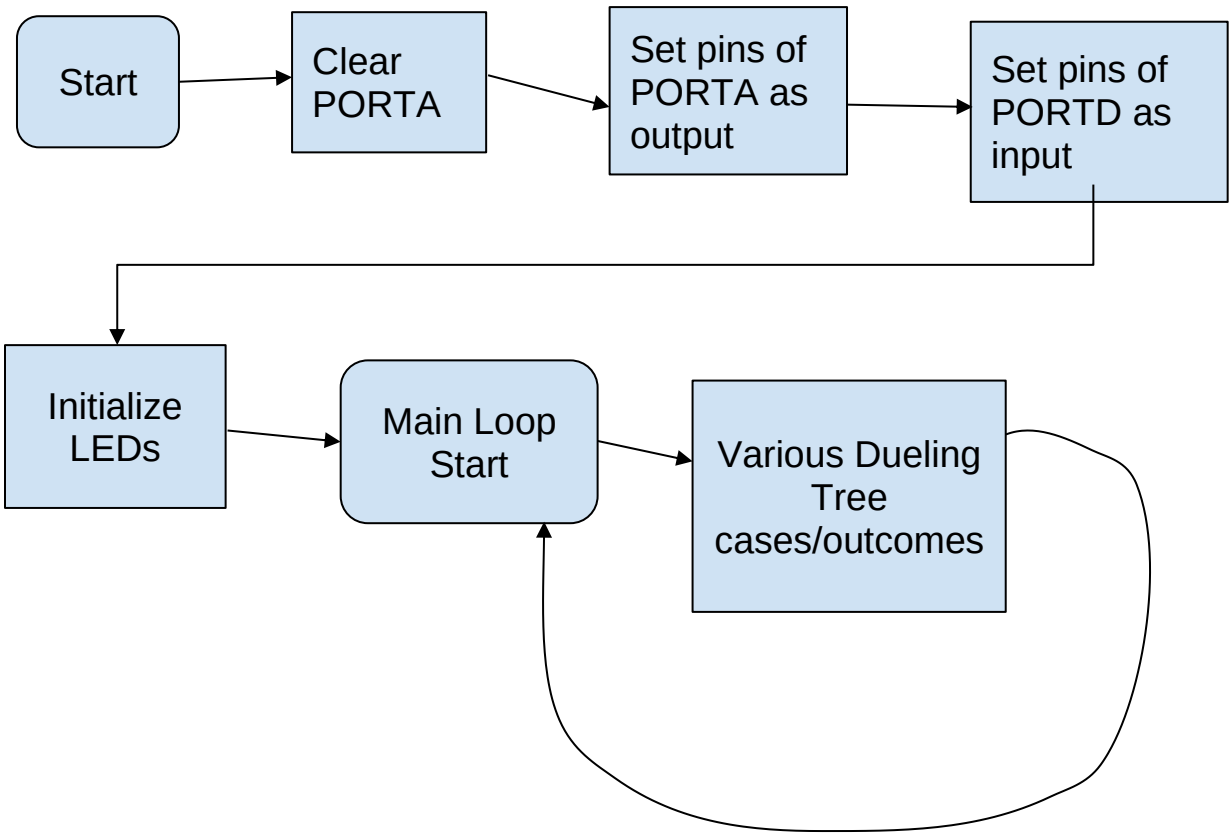


Figure 33: Flow Chart Representation of the microprocessor code.

The flowchart above is a rough design of what the final program will look like. The block named “Various Dueling Tree cases/outcomes” will be a series of if-statements that represent specific targets being moved in response to a target being hit. The final code is subject to change.

The pins of PORTA need to be set as outputs (TRISA = 0x00) so the LEDs on the board can be used as the outputs in the simulation. PORTA then needs to be cleared so no leftover data interferes with programming the board. The pins of PORTD must be set as inputs (TRISD = 0xffff) so the pushbuttons can be used as inputs in the simulation.

Microprocessor Level Four Design Code: [TW]

```
1
2
3   #include <p24FJ1024GB610.h>
4   #include <xc.h>             // Generic header for XC16 Compiler
5
6
7   /*pin #'s for LEDs
8    D3 = p17 = RA0
9    D4 = p38 = RA1
10   D5 = p58 = RA2
11   D6 = p59 = RA3
12   */
13
14  void ms_delay(int ms)
15  {
16      T2CON = 0x8030;         // Timer 2 on, TCKPS<1,0> = 11 this 1:256
17      TMR2 = 0;
18      while(TMR2 < ms*63)     // 1/(16MHz/(256*63)) = 0.001008 close to 1ms
19      {
20      }
21  } //ms_delay
22
23  int main(void)
24  {
25
26      ANSA = 0x0000; //set up portA i/o as digital so button S5 will work
27      ANSD = 0x0000;
28      PORTA = 0x00; //clear PORTA
29      TRISA = 0x0000; // set pins of PORTA as output
30      TRISAbits.TRISA7 = 1;
31      TRISD = 0xFFFF; // set pushbuttons of PORTD as input
32      LATAbits.LATA0 = 1; // set "targets" '1' -> target raised
33      LATAbits.LATA1 = 0; // '0' -> target lowered
34      LATAbits.LATA2 = 1;
35      LATAbits.LATA3 = 0;
36
```

Figure 34: Microprocessor Code: Initialization and Setup

```
37
38      while(1) //main loop start
39      {
40          if(PORTDbits.RD13 == 0) // if S4 pressed (Right target, row 1 hit)
41          {
42              LATAbits.LATA0 = 0; //lower right target, row 1
43              LATAbits.LATA1 = 1; //raise left target
44          }
45
46          if(PORTAbits.RA7 == 0) // if S5 pressed (left target, row 1 hit)
47          {
48              LATAbits.LATA1 = 0; //lower left target, row 1
49              LATAbits.LATA0 = 1; //raise right target
50          }
51
52          if(PORTDbits.RD7 == 0) // if S6 pressed (right target, row 2 hit)
53          {
54              LATAbits.LATA2 = 0; //lower right target, row 2
55              LATAbits.LATA3 = 1; //raise left target
56          }
57
58          if(PORTDbits.RD6 == 0) // if S3 pressed (left target, row 2 hit)
59          {
60              LATAbits.LATA3 = 0; //lower left target, row 2
61              LATAbits.LATA2 = 1; //raise right target
62
63          }
64      }
65  }
```

Figure 35: Microprocessor Code: Main Loop

The code above programs the Explorer 16/32 Development board such that it simulates the dueling tree session using the buttons as the method for hit detection, and using the LEDs as a method to display which targets are raised and which are lowered. A delay function was included for safety, but it was ultimately not needed for the demonstration. The code was written in the C programming language as it was the most convenient and effective language to use with the Explorer 16/32 Development board.

One problem that was encountered during the process of programming the board was the configuration of button S5. This particular pushbutton is connected to pin 92 on the board, which is also connected to LED D10. This caused issues since it became impossible to properly use the button as an input since the LED was always off, which caused the button to always be read as high (active low).

To resolve this, PORTA and PORTD had to be configured as digital input/output. The issues were resolved after this configuration.

For the Dueling Tree session on our final design, the case that two targets are hit at the same time will not matter (two targets on the same row will never be up at the same time, and it is irrelevant if two targets on different rows are hit at the same time), so this code was also written with that in mind. The following figures show how each LED signal changes depending on which button is pressed.

Figure 36: Microprocessor Simulation: Right Target, Row 2 (LED D5, Channel 3) is hit and lowered; Left Target, Row 2 (LED D6, Channel 2) is raised in response

This diagram shows the case that a target on the right is hit and is lowered, then the target on the left is raised in response so the other player can hit said target.

Figure 37: Microprocessor Simulation: Left Target, Row 2 (LED D6, Channel 2) is hit and lowered; Right Target, Row 2 (LED D5, Channel 3) is raised in response.

This diagram shows the case that a target on the left is hit and is lowered, then the target on the right is raised in response so the other player can hit said target.

Figure 38: Microprocessor Simulation: Right Target hit, then Left Target hit

This diagram shows the case that the right target is hit by the first player, then the left target is hit by the second player.

Communications Level One Block Diagram [SV]

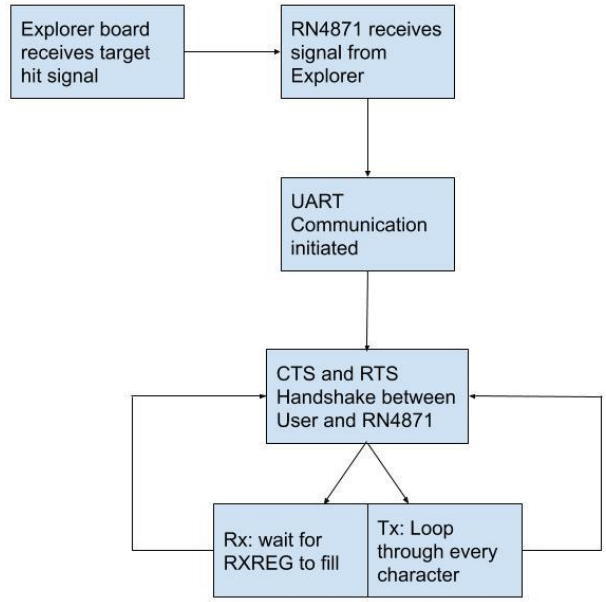


Figure 39: Communication Level One Block Diagram

Module	UART Communications
Designers	Sai Vellala
Inputs	Power, Signal Received
Outputs	Signal Transmitted
Description	The System will utilize UART communication for control of the system at any given point within a firing range (assumed to be 50ft)

Table 19: Communications Level One Block Diagram Overview

Communication Level Two Flow Chart [SV]

Functions that will be used:

void initU2(void): This function will configure the UART module

char putU2(char c): This function will allow the UART module to transmit data

char getU2(void): This function will allow the UART module to receive data

void ms_delay(int)/ void us_delay(int): this function will be used for timing between certain commands

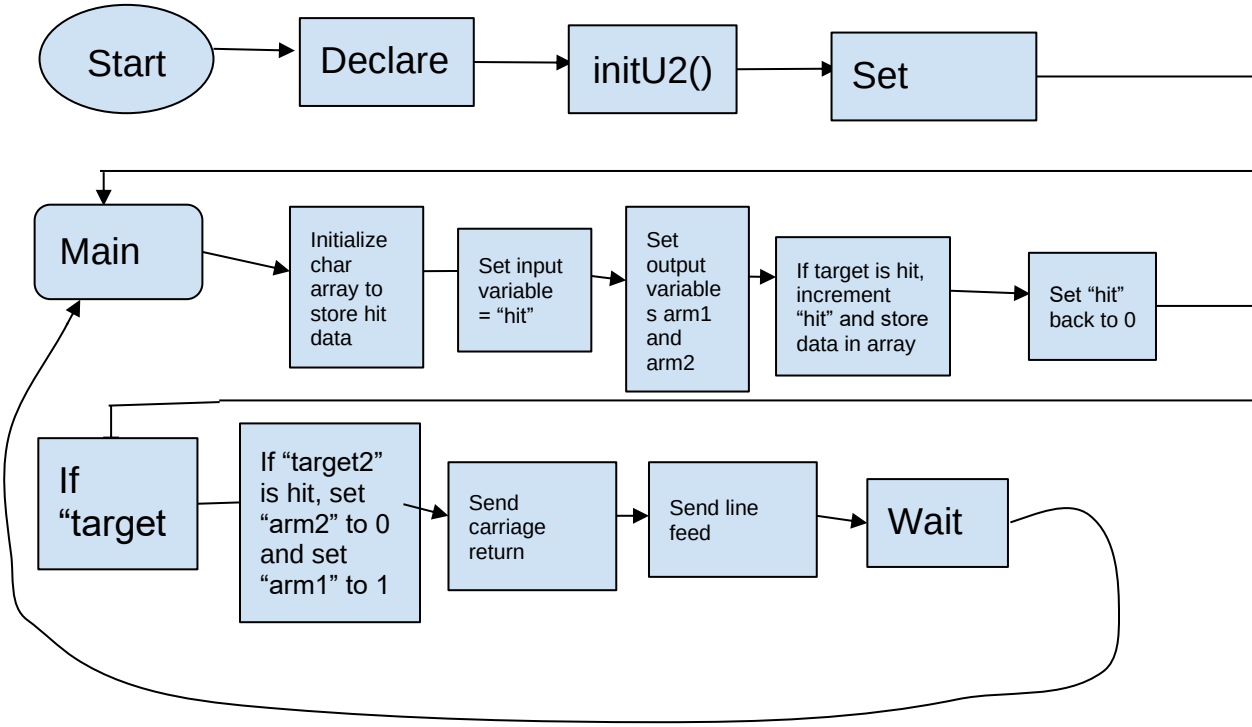


Figure 40: Flow Chart for the UART communication code.

Communications Level Three Designed Code [SV]

```
// FOSCSEL
#pragma config FNOSC = PRIPLL           // Oscillator Source Selection (Primary Oscillator with PLL module (XT + PLL, HS + PLL, EC
+ PLL))
#pragma config PLLMODE = PLL4X          // PLL Mode Selection (4x PLL selected)
#pragma config IESO = ON                // Two-speed Oscillator Start-up Enable bit (Start up device with FRC, then switch to
user-selected oscillator source)

// FOSC
#pragma config POSCMD = XT              // Primary Oscillator Mode Select bits (XT Crystal Oscillator Mode)
#pragma config OSCIOFCN = OFF           // OSC2 Pin Function bit (OSC2 is clock output)
#pragma config SOSCSEL = ON             // SOSC Power Selection Configuration bits (SOSC is used in crystal (SOSCI/SOSCO) mode)
#pragma config PLLSS = PLL_PRI          // PLL Secondary Selection Configuration bit (PLL is fed by the Primary oscillator)
#pragma config IOL1WAY = ON             // Peripheral pin select configuration bit (Allow only one reconfiguration)
#pragma config FCKSM = CSDCMD           // Clock Switching Mode bits (Both Clock switching and Fail-safe Clock Monitor are
disabled)
```

Figure 41: Communication Code: Configuration File

```
void InitU2(void){
    U2BRG = 34; // Fcy = 16MHz, desired BR = 115200, BRG=34
    U2MODE =0x8008; //BRGH = 1, 8N1N
    U2STA = 0x0400;
    TRISFbits.TRISF13 = 1; // enable RTS
    RTS = 1; // not RTS
}

char putU2(char c)
{
    //while (CTS); //wait for !CTS (active low)
    while (U2STAbits.UTXBF ); // Wait if transmit buffer full.
    U2TXREG = c; // Write value to transmit FIFO
    return c;
}


char getU2 ( void )
{
    RTS = 0; // telling the other side !RTS
    while (! U2STAbits . URXDA ); // wait
    RTS =1; // telling the other side RTS
    return U2RXREG ; // from receiving buffer
} //getU2
```

Figure 42: Communication Code: Initialize, Get and Put

```
int main(void) {
    int i, j, count = 0;
    char demo[25];
    char transmission[25];
    RPINR19bits.U2RXR = 10;
    RPOR8bits.RP17R = 5;
    ms_delay(100);
    InitU2();
    ms_delay(100);
    sprintf(demo, "Welcome to the demo!");
    size_t len_demo = strlen(demo);
    while(1){
        sprintf(transmission, "This is transmission %d", count);
        size_t len_trans = strlen(transmission);
            for(j = 0; j < len_demo; j++)
            {
                putU2(demo[j]);
            }
            putU2(0x0A);
            ms_delay(100);
        for(i = 0; i < len_trans; i++)
        {
            putU2(transmission[i]);
        }
        putU2(0x0A);
        ms_delay(1000);
        count = count + 1;

    }
    return 0;
}
```

Figure 43: Communication Code: Main Loop

Accepted Final Design

Sensing Circuits Design [NH]

There will be six total Breakouts for the Sensor circuits, one for each arm of the D.A.R.T system. This is a simple circuit that consists of only the ADXL325BCPZ-RL8 accelerometer. This component was decided to have a separate board for two reasons. First this would allow for the sensor to be placed as close to the impact as possible to get the maximum transfer of signal to the sensor. The low profile and weight of having a single component would allow for the circuit to be mounted on the physical target arm without affecting the dynamics or putting unnecessary load on the motors. The second reason is that this would allow for only one component to be at risk of being damaged by wayward projectiles or fragmentation. The breakout board would be outfit with a plug in case of any damage the part could be easily replaced. The ADXL325BCPZ-RL8 Accelerometer is a surface mount component, so for this reason it was chosen to have a custom breakout board designed instead of a solder board. The sensor requires 3.3VDC And Ground as an input and will output three analog signals that are proportional to the acceleration in the X, Y and Z axis relative to the sensor. For this reason a six pin plug was chosen to connect this circuit to the rest of the system.

The original design of this breakout included an additional circuit that would add the three axis analog signals together because the rest of the system was designed to be direction agnostic, the Sensing Filter circuit only required the detection of a change in acceleration. However after the Circuit board was ordered it was discovered that a voltage buffer was required between the sensor and the sensing filter. Unbeknownst to us this buffer existed on the test board that was used for the preliminary design, which is why it was not present in the board we designed. The design of the breakout board was still able to be repurposed as without using the addition circuit and treating it as a true breakout board. This was done to help preserve the

budget and time of the project. The original and final circuit that was designed using the Autodesk Eagle software can be seen below.

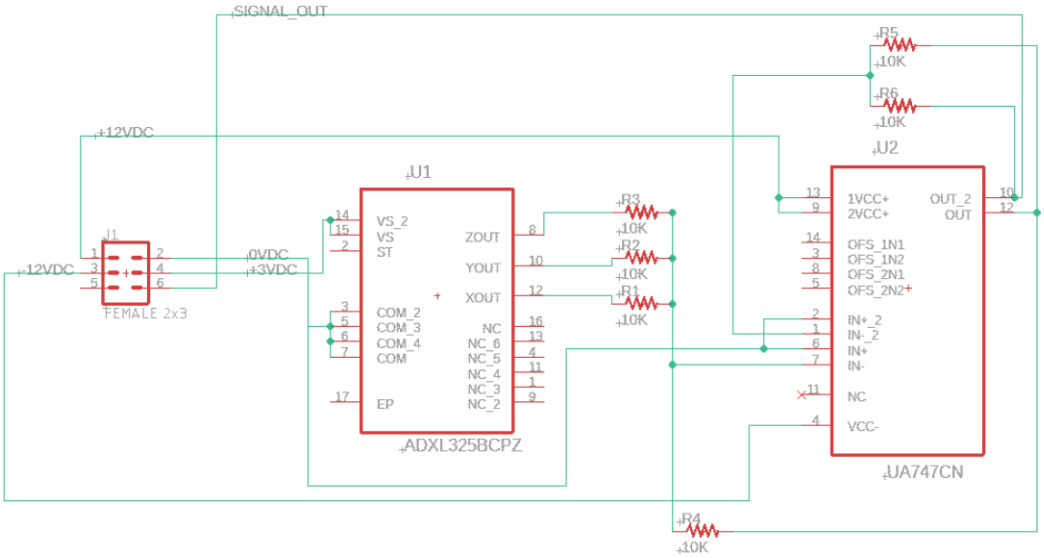


Figure 44: Breakout Sensor Design Prior to Redesign

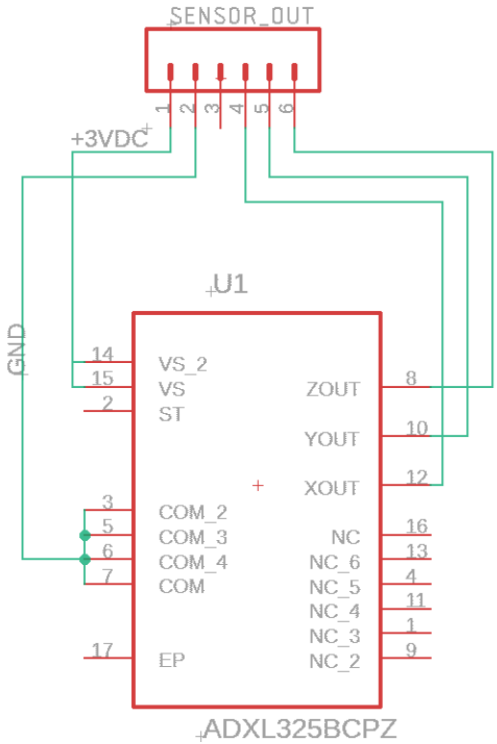


Figure 45: Eagle Schematic of Final Accepted Design of the Breakout Sensor

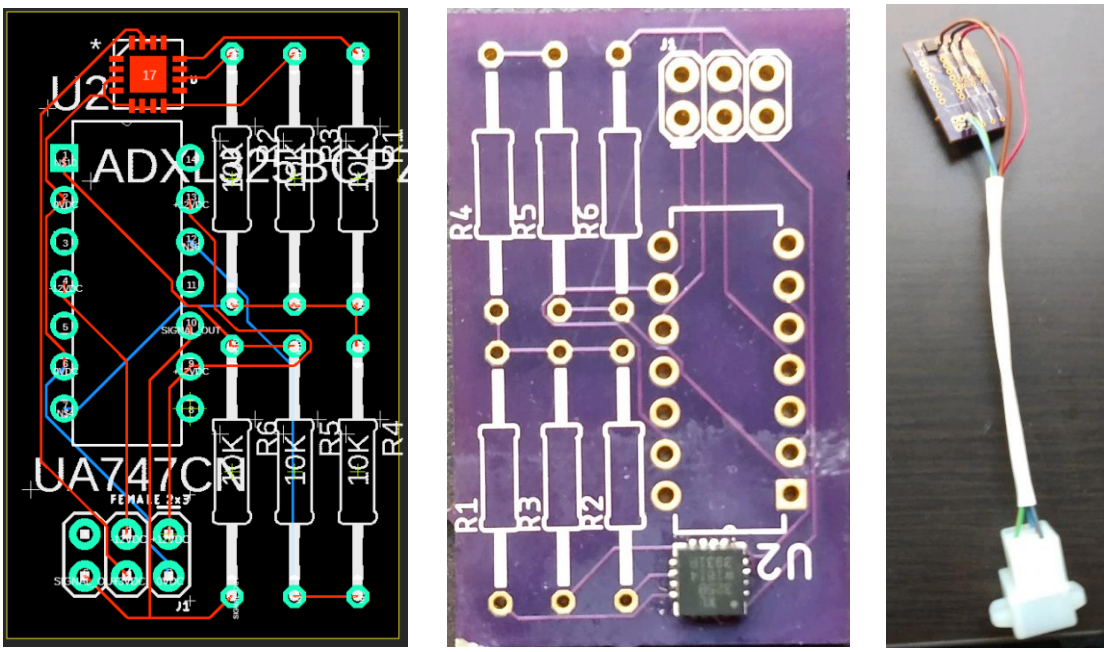


Figure 46: Progression of Breakout Sensor form Eagle PCB Design (Left), Delivered PCB from OSHPARK (middle) and then Final wired board (Right)

The purpose of the Sensing Filter is to read the raw analog signal coming in from the sensor. The output of this circuit will be one signal for each sensor. This signal for each sensor will be 0VDC until an acceleration higher than that set by the trigger is detected by the filter, in which the signal would become nonzero. Unlike the diagram in Figure 23, it was chosen to separate the sensing filters based on rows. This was done to keep the size of the circuit down to allow for easier debugging when constructing the final solder boards. This also was the ideal amount to support the resetting circuitry that was added and changed. The original TK1-3V relay that was going to be used to reset the capacitors were unable to be ordered so the EC2-3TNU was ordered which is a double pull double throw and could support resetting two sensors. This design strategy of using solder boards proved to be useful when issues like the missing voltage

buffer arose. The designed circuit schematic can be seen below with a picture of the physical circuit.

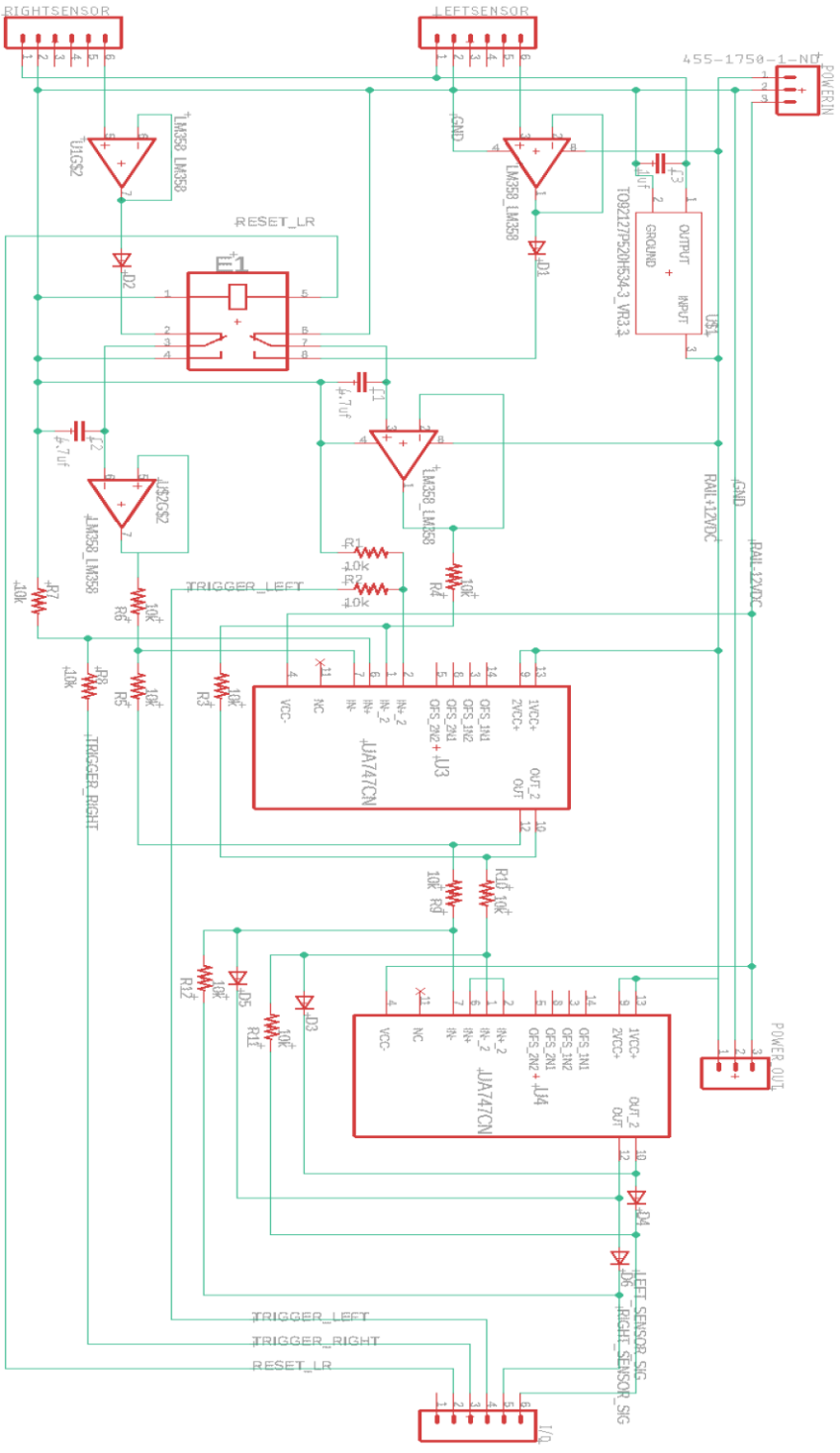


Figure 47: Eagle Schematic of the final Sensing Filter

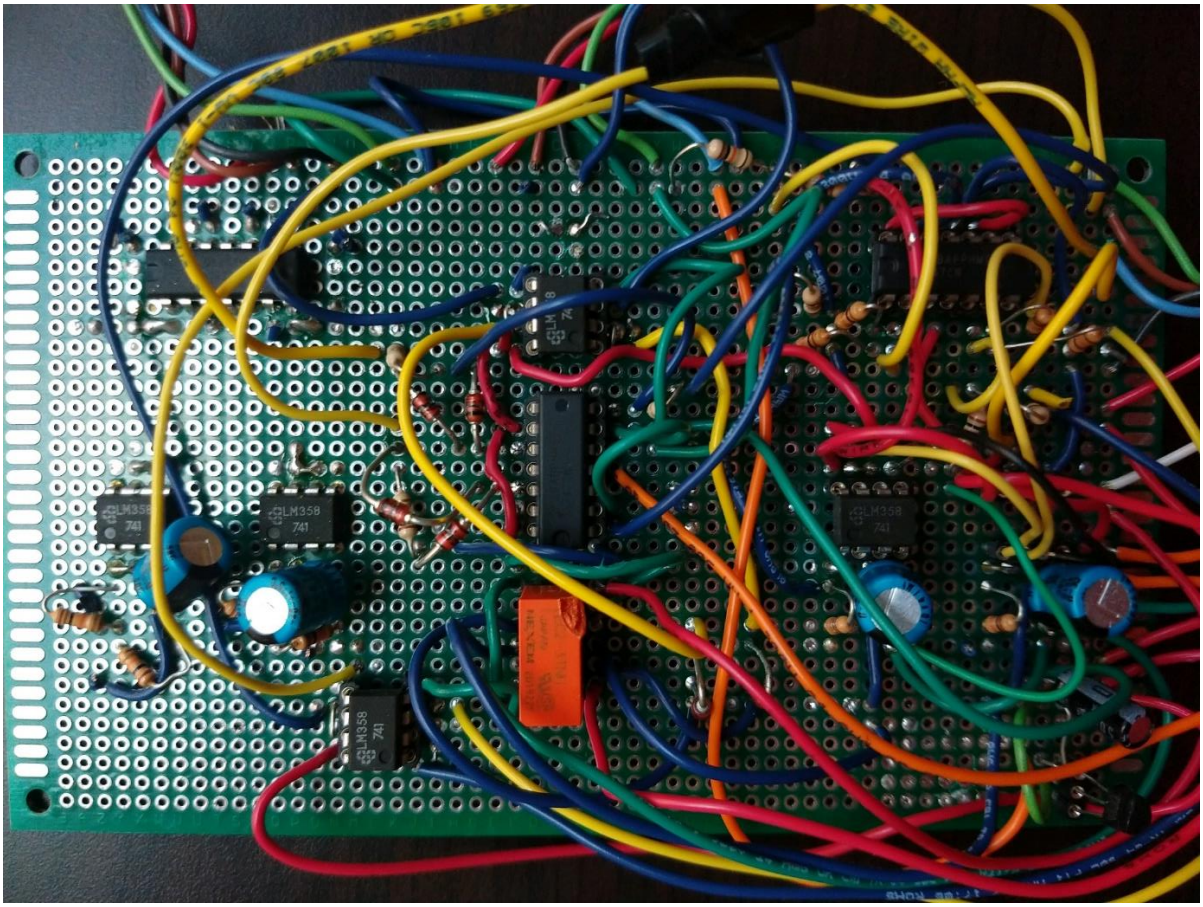


Figure 48: Photo of solder board of first designed Sensing Filter

Qty.	Refdes	Part Num.	Description
2	U3, U4	UA747CN	LPD OP AMP
12	R1,R2,R3,R4,R5,R6,R7, R8,R9,R10,R11,R12	691104	10K 0.25W Resistor
1	E1	EC2-3TNU	RELAY GEN PURPOSE SPDT 2A 3VDC
3	C1, C2, C3	1946261	47uf Capacitor
6	D1, D2, D3, D4, D5, D6	1N4004	Diode
2	U1,U1	LM358	LPD OP AMP
1	U5	LP2950ACZ-3.0G	IC REG LINEAR 3V 100MA TO92-3
1	POWERIN	A1401-ND	3 Pin Male Plug
1	POWEROUT	A1400-ND	3 Pin Female Plug
2	LEFTSENSOR, RIGHTSENSOR	A1405-ND	6 Pin Male Plug
1	I/O	A1404-ND	6 Pin Female Plug

Table 20: Parts List for Final Sensing Filter

Prior to the campus shutdown due to COVID-19 all six Breakouts for Sensor boards were completed and verified as working properly and one of the sensing filter boards was completed. The last two Sensing Filter boards were in the process of being completed and debugged. A power supply was used to simulate +12VDC, -12VDC and ground that would come from the battery circuit on the Sending Filter. From this we were able to properly power both the Sensing Filter and the attached Breakout for the Sensor. Another power supply was used to simulate the trigger voltage and reset signals that would be coming from the embedded system. With all the inputs properly simulated we were able to see the proper response on the output when each sensor was hit with a thrown object. The waveforms seen on the oscilloscope appeared to match the waveforms on the test circuit that was used for proof of concept with live ammunition.

Given more time I would like to do more wire management of the circuits to help the debugging process moving forward. Other than this the next steps to take with this system will start with testing the accuracy with live ammunition; however, this will not be possible in a timely manner due to the COVID-19 Pandemic. The waveforms seen on the oscilloscope during debugging of the Sensing Filter appeared to match the waveforms on the test circuit that was used for proof of concept with live ammunition. This seemed very promising that this system would have worked properly if used with live ammunition. Once the live ammunition test was completed integration with the battery circuit and the embedded system could begin.

Battery Circuit Design [NH, TM]

Power for the system would be generated from two 12VDC, 230CCA batteries to create the positive and negative rails that will be used throughput the system (see below). Some of the components that are used in the system were very sensitive to having exactly 12VDC applied so a voltage regulator was fit for the top rail to smooth out the voltage and handle any changes in

the charging and unchanging of the battery over time. The circuit was going to have a 3 pin connector output so the power could be unplugged from the rest of the system and easily recharged when needed.

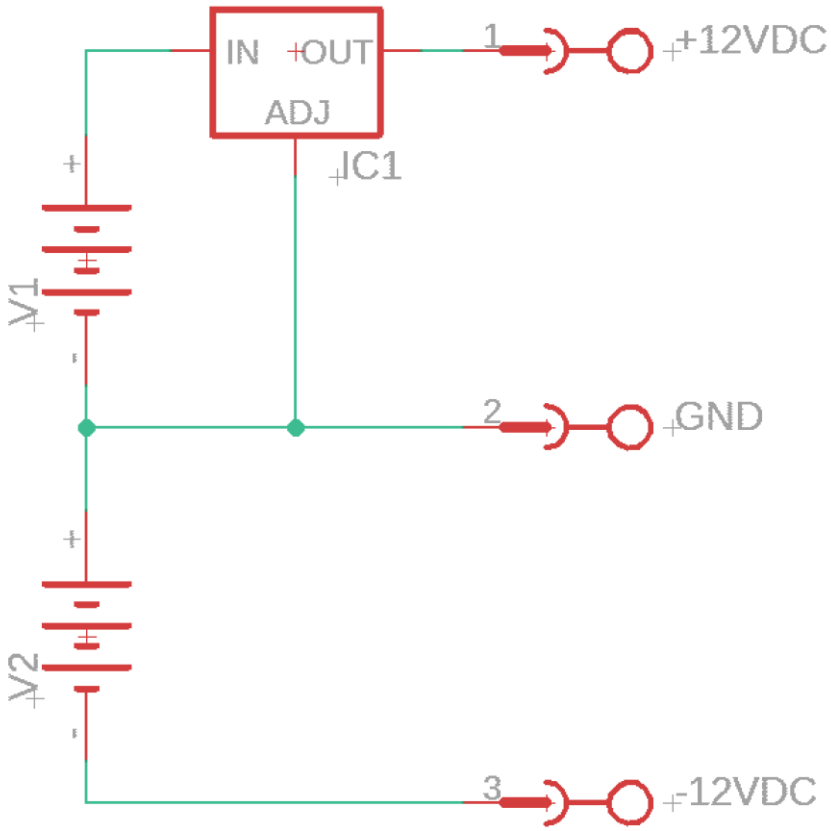


Figure 49: Eagle Schamatic of the Battery Circuit

Qty.	Refdes	Part Num.	Description
2	V1,V2	725-1750A	12VDC, 230CCA Battery
1	IC1	LT1084CT-12#PBF	12VDC Regulator

Table 21: Parts List for the Battery circuit

Actuating Circuits Design [TM]

Last semester the design to regulate the input voltage to the motor did not work successfully as planned, where there was an issue regulating the voltage to -6V. Due to this two solutions were created to attempt to address this problem, and also make a more compact and cost efficient circuit.

Design One:

This design tried to condense the existing design from 6 operational amplifier down to just two. The way this was accomplished was to make one of the operational amplifier a NIC (Negative Impedance Converter) to simulate a negative resistance, so that if V2 is set to 0V DC, the output to the PID controller would correctly show as -6V DC. In the case where the voltage V2 was set to 3V DC, then the output would read as 6V DC, where in both scenarios the voltage from the battery (12VDC) was applied to V1. This allowed for the circuit to be considerably more compact. The issue with this design however, was that the VI curve for the operational amplifier affected the effectiveness of the NIC, and due to this it was judged to be too unaccountable for a final design so a newer design was implemented. This design was successfully tested in the lab however, and was working for the time of the midterm presentations.

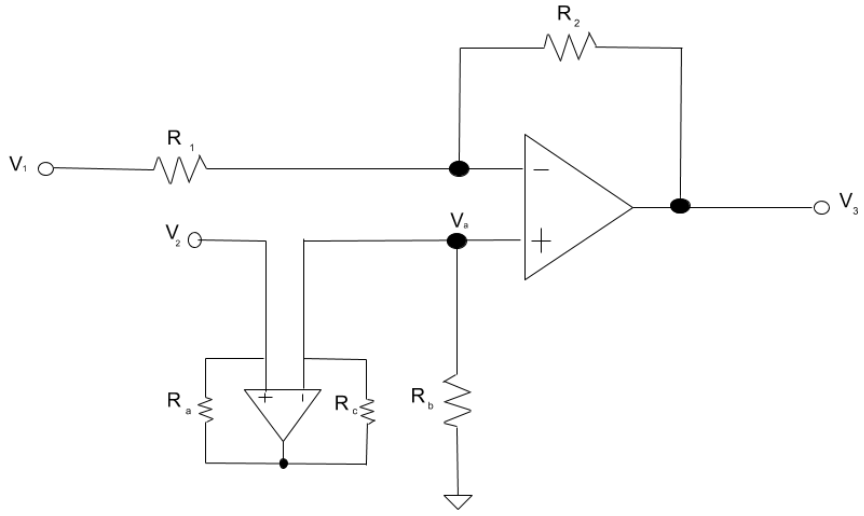


Figure 50: Shows the design of the implemented input voltage device with an NIC being used.

Voltage	Case 1	Case 2
V1	12VDC	12VDC
V2	0	3VDC
V3	-6VDC	6VDC

Table 22: Shows the Expected Voltage levels in both stages for the circuit in figure 49

R1	10kOhm
R2	5KOhm
Rb	4.7KOhm
Ra	1KOhm
Rc	1.2KOhm

Table 23: Shows the component values for both stages for the circuit in figure 49

Design 2:

The second design created was one that used an operational amplifier and a switching voltage regulator. This regulator would only start after the voltage reached a threshold of 3VDC from the embedded system. Essentially this design saw the NIC replaced with the voltage regulator circuit. Unfortunately this design was only being implemented just before the break from school and all information on it is in the school computers.

The final design, shown below, is the finalized design used for the motor actuating circuit. This design was fully functional and worked mostly as predicted. Small issues arose due to using an operational amplifier whose slew rates surpassed the needed rate for the transient response to be perfect, however the design of the circuit worked as expected. The solution to this problem was going to be implementing OP27 operational amplifiers instead of the OP747 that were currently being used, because the OP27 had a slew rate that could handle the quick rise in voltage level the transient produced in 0.5 seconds. The PID Design from last semester was unchanged and worked as anticipated.

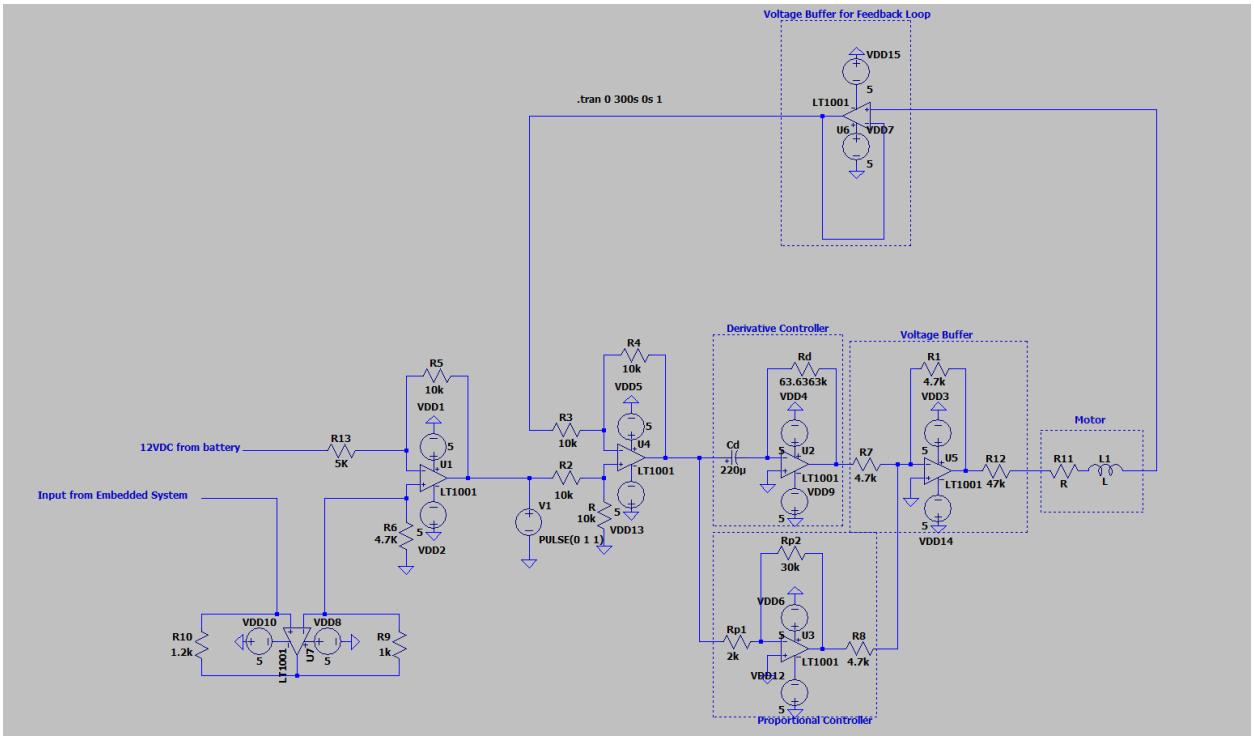


Figure 51: Shows the final accepted design for the Actuating Circuit

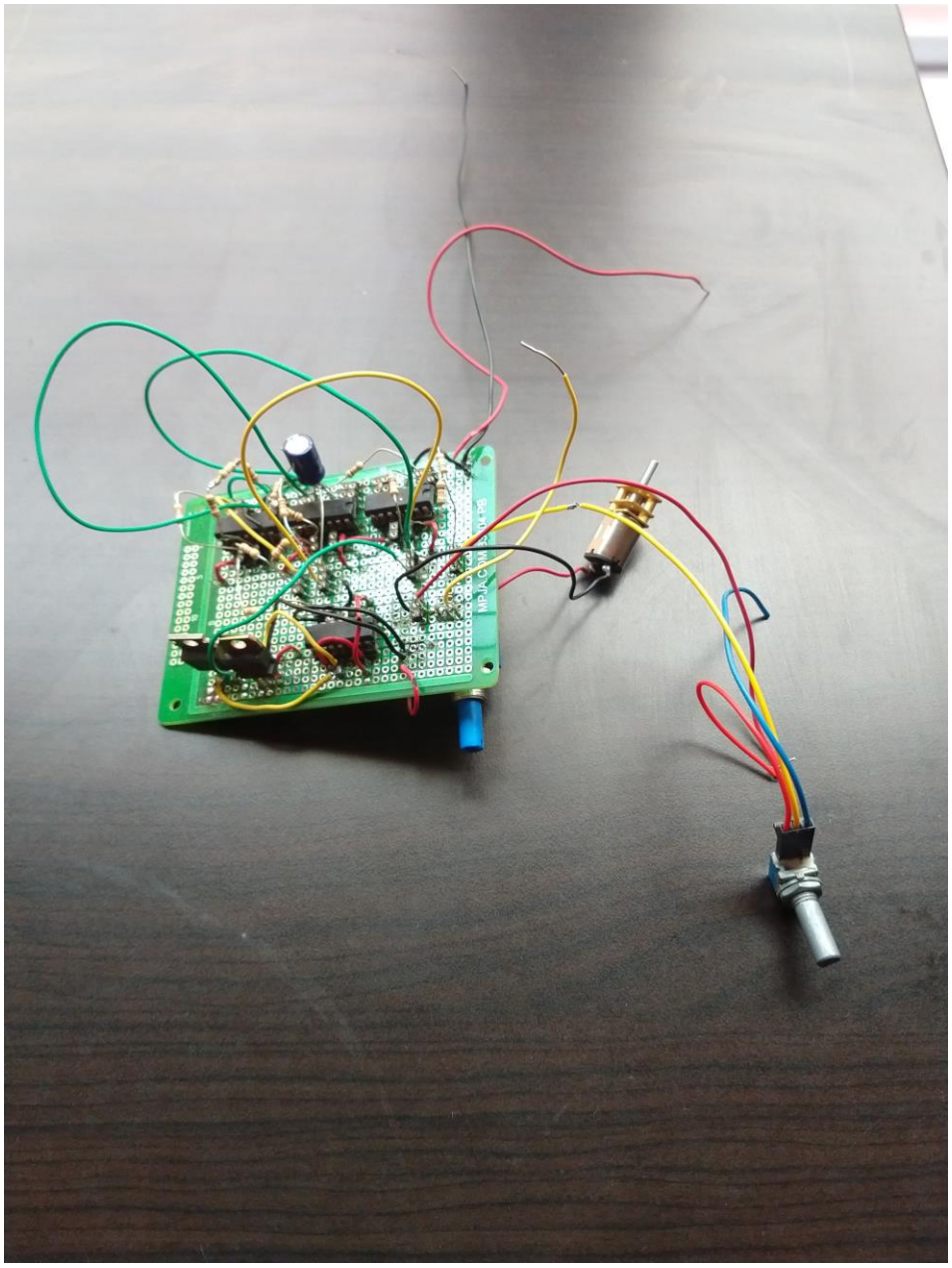


Figure 52: Shows the implemented working Motor Circuit

Next Steps Taken:

If the project was allowed to continue, the first step would be to print out the full actuating structure and fasten the circuit to it as anticipated. With this in order, the final step

needed would be to integrate it to the battery and embedded system, which would not have taken long, since it would only take an analog pulse from the embedded controller. If all went well PCB parts would be ordered for the other identical actuating circuits and after assembly this part of the project would be completed.

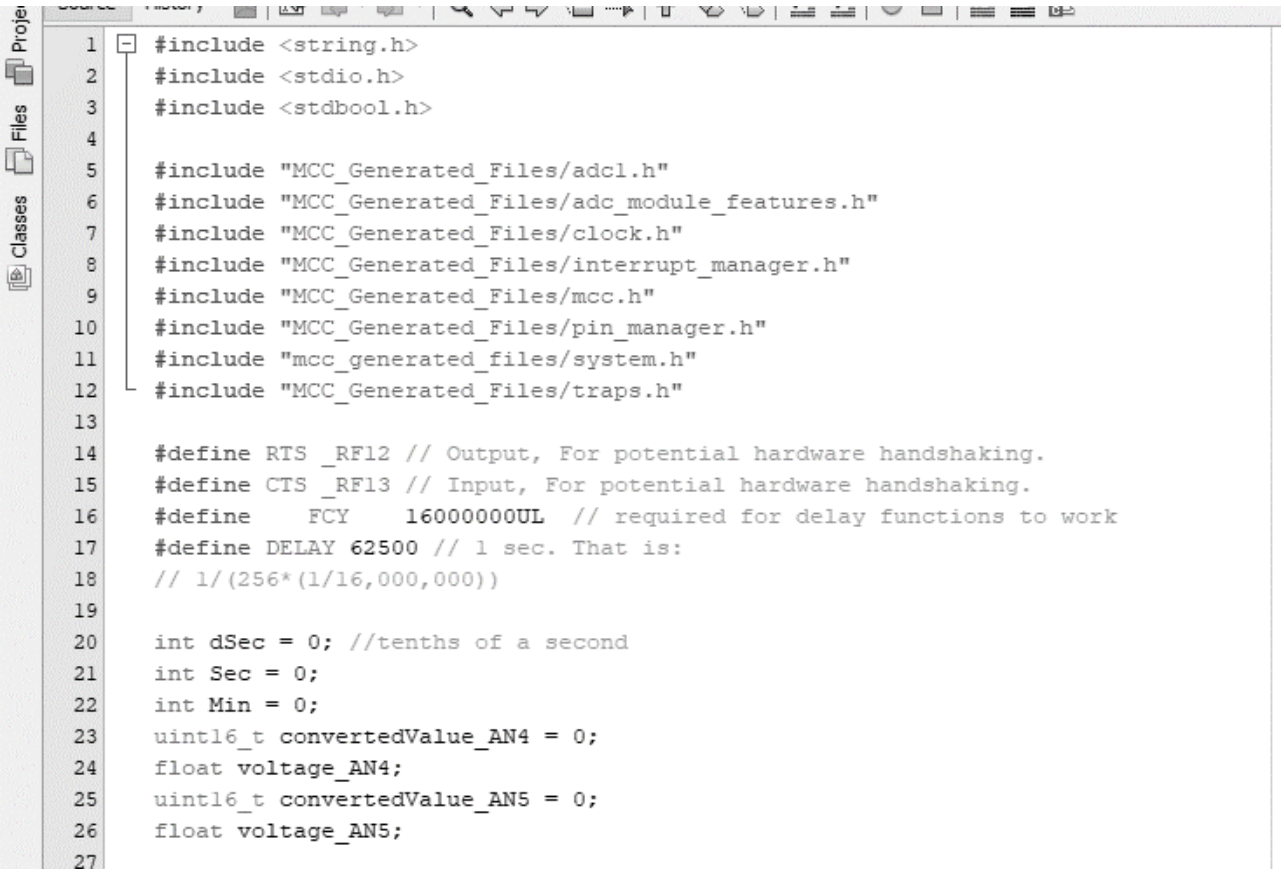
Microprocessor Design [TW]

During this semester, we began to work on integration of the input/output control with the sensing circuit, the wireless communication, and the web interface. The first task we focused on was displaying the data of which target was hit and the data of where each arm was currently located. This data would be put into an array, which would then be printed as an 8-bit string and displayed on the screen because of the Bluetooth module. This process was done with the 'sprintf' function and multiple arrays. We ran into some difficulties in displaying the correct data as we would either get garbage data or incorrect numbers. The only numbers displayed are a combination of 0's and 1's. The first 4 characters in the string show which target was hit, and the last 4 characters show where the active targets are currently located.

The next task we worked on was putting the entire loop on a 60 second timer to simulate a single Dueling Tree session. To do this, we had to create an interrupt service routine that would allow us to count up to 60 seconds using Timer 1 on the microcontroller. It would also allow us to count in milliseconds, but this was not needed since we had a dedicated millisecond delay function.

The next task to be completed was integration with the sensing circuit. We were able to connect the circuit to the Explorer 16/32 board and use it as a simulation for a single row of 2 targets (one on each side) and switch them between the states of being raised and lowered. However, we were not able to configure the code to manually set the voltage of specific pins. We were in the

process of figuring this out, and we did manage to set pins to either 0V or 3.3V, but could not set any values in between such as 1.5V, 2V, etc.

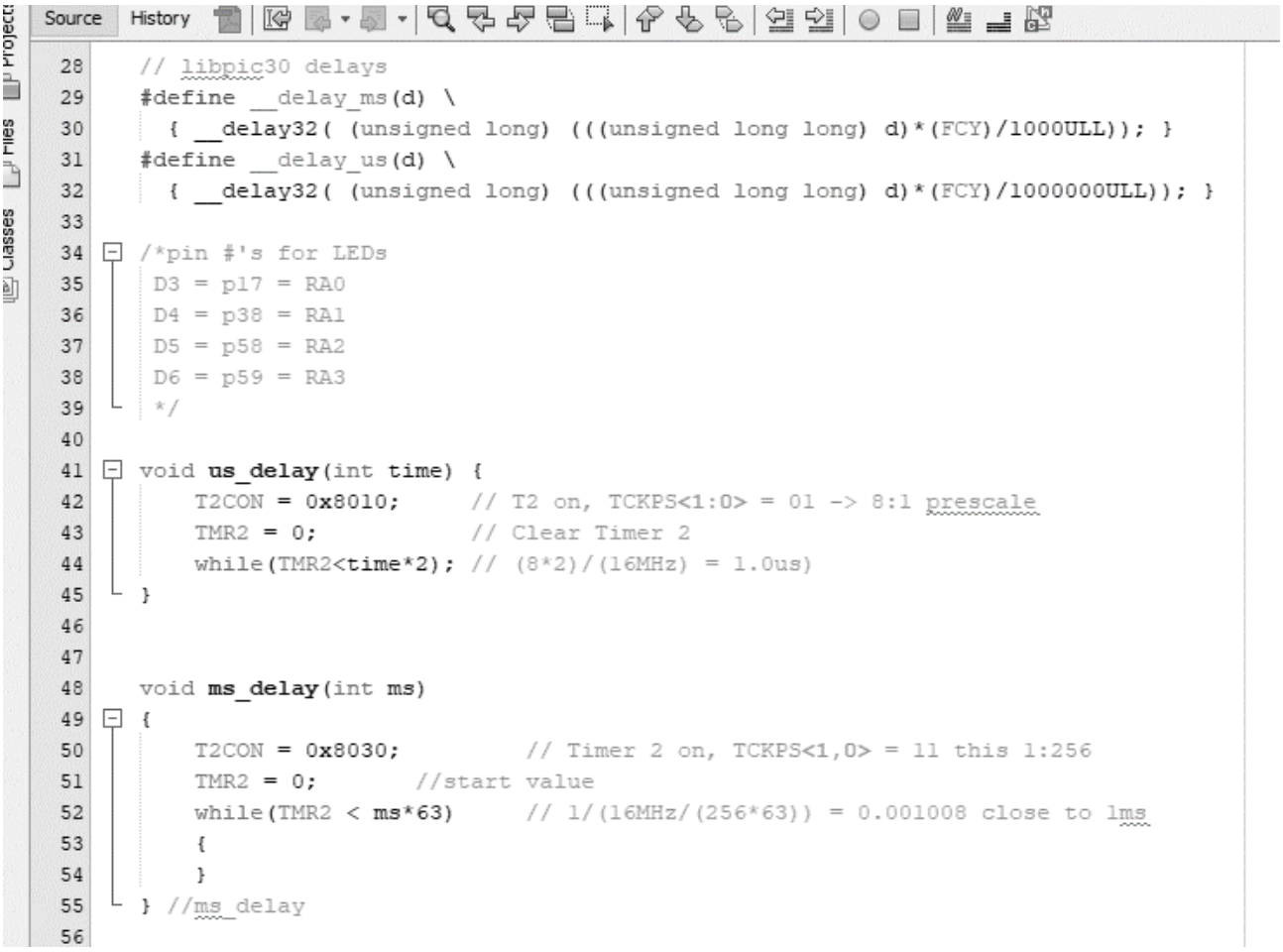


```
#include <string.h>
#include <stdio.h>
#include <stdbool.h>

#include "MCC_Generated_Files/adcl.h"
#include "MCC_Generated_Files/adc_module_features.h"
#include "MCC_Generated_Files/clock.h"
#include "MCC_Generated_Files/interrupt_manager.h"
#include "MCC_Generated_Files/mcc.h"
#include "MCC_Generated_Files/pin_manager.h"
#include "mcc_generated_files/system.h"
#include "MCC_Generated_Files/traps.h"

#define RTS _RF12 // Output, For potential hardware handshaking.
#define CTS _RF13 // Input, For potential hardware handshaking.
#define    FCY    16000000UL  // required for delay functions to work
#define DELAY 62500 // 1 sec. That is:
// 1/(256*(1/16,000,000))

int dSec = 0; //tenths of a second
int Sec = 0;
int Min = 0;
uint16_t convertedValue_AN4 = 0;
float voltage_AN4;
uint16_t convertedValue_AN5 = 0;
float voltage_AN5;
```



```
28  // libpic30 delays
29  #define __delay_ms(d) \
30    { __delay32( (unsigned long) (((unsigned long long) d)*(FCY)/1000ULL)); }
31  #define __delay_us(d) \
32    { __delay32( (unsigned long) (((unsigned long long) d)*(FCY)/1000000ULL)); }
33
34  /*pin #'s for LEDs
35   D3 = p17 = RA0
36   D4 = p38 = RA1
37   D5 = p58 = RA2
38   D6 = p59 = RA3
39   */
40
41  void us_delay(int time) {
42      T2CON = 0x8010;      // T2 on, TCKPS<1:0> = 01 -> 8:1 prescale
43      TMR2 = 0;            // Clear Timer 2
44      while(TMR2<time*2); // (8*2)/(16MHz) = 1.0us)
45  }
46
47
48  void ms_delay(int ms)
49  {
50      T2CON = 0x8030;          // Timer 2 on, TCKPS<1,0> = 11 this 1:256
51      TMR2 = 0;        //start value
52      while(TMR2 < ms*63)      // 1/(16MHz/(256*63)) = 0.001008 close to 1ms
53      {
54      }
55  } //ms_delay
56
```

```
57  void InitU2(void){
58      U2BRG = 34; // Fcy = 16MHz, desired BR = 115200, BRG=34
59      U2MODE =0x8008; //BRGH = 1, 8N1N
60      U2STA = 0x0400;
61      TRISFbits.TRISF13 = 1; // enable RTS
62      RTS = 1; // not RTS
63  }
64
65  char putU2(char c)
66  {
67      //while (CTS); //wait for !CTS (active low)
68      while (U2STAbits.UTXBF ); // Wait if transmit buffer full.
69      U2TXREG = c; // Write value to transmit FIFO
70      return c;
71  }
72
73
74  char getU2 ( void )
75  {
76      RTS = 0; // telling the other side !RTS
77      while (! U2STAbits . URXDA ); // wait
78      RTS =1; // telling the other side RTS
79      return U2RXREG ; // from receiving buffer
80  } //getU2
```

```
82  void _ISR _T1Interrupt(void)
83  {
84      dSec++;
85      if(dSec > 9)
86      {
87          dSec = 0;
88          Sec++;
89          if(Sec > 59)
90          {
91              Sec = 0;
92              Min++;
93              if(Min > 59)
94              {
95                  Min = 0;
96              }
97          }
98      }
99      _T1IF = 0;
100 }
101
102 void InitPMP( void)
103 {
104     // PMP initialization.
105     PMCON1 = 0x8303;          // Following Fig. 13-34. Text says 0x83BF (it works
106     //PMMODE1 = 0x03FF;         // Master Mode 1.  8-bit data, long waits.
107     PMCON4 = 0x0001;          // PMA0 enabled
108 }
```

```
109
110  void initLCD (void)
111  {
112      // PMP is in Master Mode 1, simply by writing to PMDIN1 the PMP takes care
113      // of the 3 control signals so as to write to the LCD.
114      PMCON4 = 0;                 // PMA0 physically connected to RS, 0 select Control register
115
116      PMDIN1 = 0b00111000;      // 8-  bit, 2 lines, 5X7. See Table 9.1 Function set
117      ms_delay(1);                // 1ms > 40us
118
119      PMDIN1 = 0b00001100;        // ON, cursor off, blink off
120      ms_delay(1);                // 1ms > 40us
121
122      PMDIN1 = 0b00000001;        // clear display
123      ms_delay(2);                // 2ms > 1.64ms
124
125      PMDIN1 = 0b00000110;        // increment cursor, no shift
126      ms_delay(2);                // 2ms > 1.64m
127  }
128
129  char ReadLCD (int addr)
130  {
131      // the first read has previous value in PMDIN1
132      int dummy;
133      while( PMCON2 = 0x8000);    // wait for PMP to be available
134      PMCON4 = addr;              // select the command address
135      dummy = PMDIN1;             // initiate a read cycle, dummy
136      while( PMCON2 = 0x8000);    // wait for PMP to be available
137      return( PMDIN1);            // read the status register
138  } // ReadLCD
139
```

```
139
140  // In the following, addr= 0 -> access Control, addr= 1 -> access Data
141  #define BusyLCD()    ReadLCD( 0) & 0x80 // D<7> = Busy Flag
142  #define AddrLCD()    ReadLCD( 0) & 0x7F     // Not actually used here
143  #define getLCD()     ReadLCD( 1)           // Not actually used here
144
145  void WriteLCD( int addr, char c)
146  {
147      while (BusyLCD());
148      while( PMCON2 = 0x8000);        // wait for PMP to be available
149      PMCON4 = addr;
150      PMDIN1 = c;
151  }
152
153  #define putLCD( d)  WriteLCD( 1, (d))
154  #define CmdLCD( c)  WriteLCD( 0, (c))
155  #define HomeLCD()   WriteLCD( 0, 2)
156  #define ClrLCD()    WriteLCD( 0, 1)
157
158  void putsLCD( char *s)
159  {
160      while( *s) putLCD( *s++);
161
162  } //putsLCD
163
```

```
165  void SetCursorAtLine(int j)
166  {
167
168      if(j == 1)
169      {
170          CmdLCD(0x80); // sets cursor to top left
171      }
172      else if(j == 2)
173      {
174          CmdLCD(0xC0); // sets cursor to bottom left
175      }
176  }
177
178  /*...63 lines */
241
242
243
244
245
246  //...70 lines
316
317  int main(void)
318  {
319      T1CON = 0x8030; // Timer 1 On, prescale 1:256, Tcy as clock
320      _T1IP = 4; //default value
321      TMR1 = 0; // clear timer
322      PR1 = 6250-1;  // Set period register for T1 for 0.1 seconds
323      _T1IF = 0; // Make sure T1 interrupt flag is cleared
324      _T1IE = 1; // Enable the T1 interrupt emitter
325
```

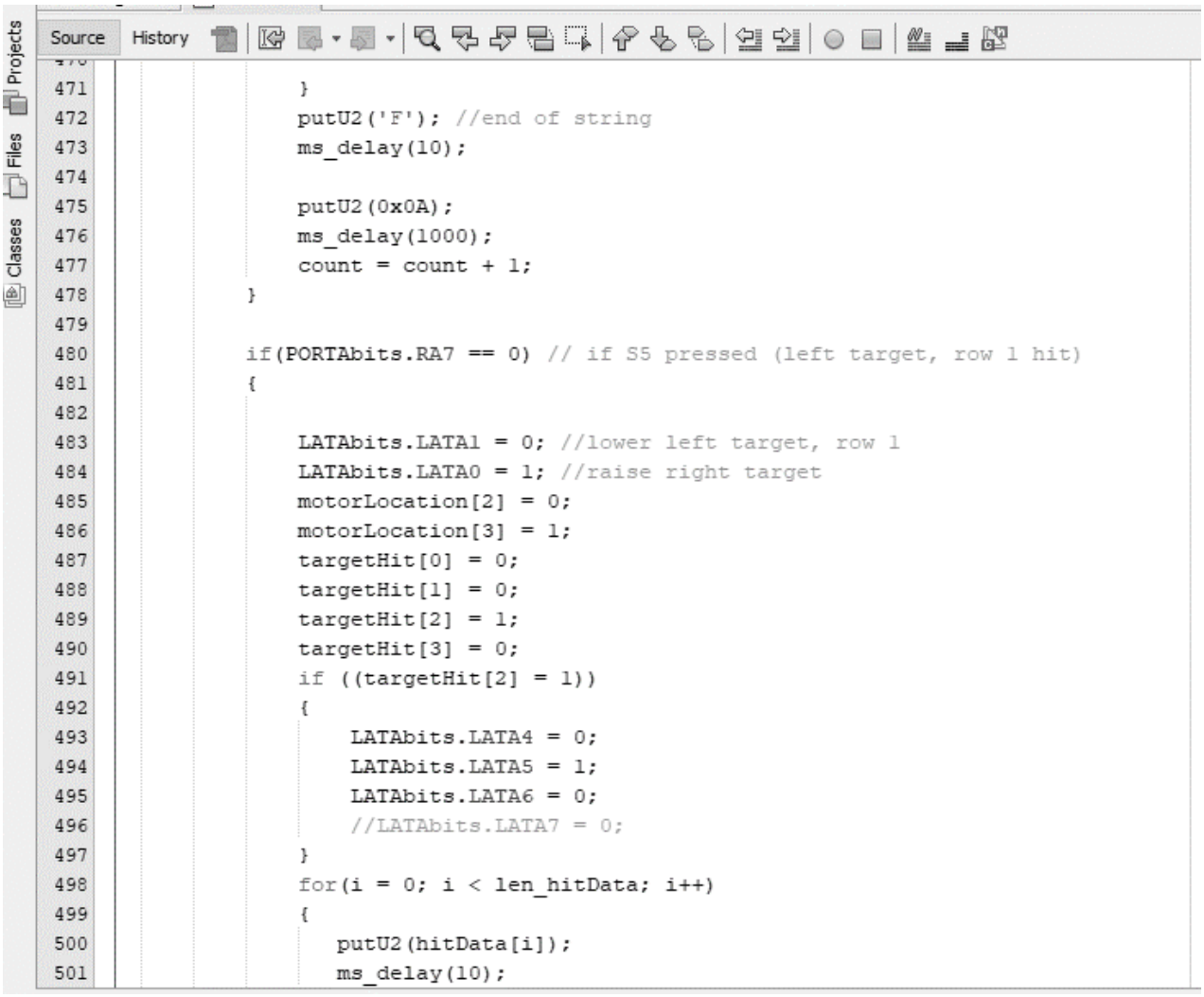
```
326     SYSTEM_Initialize();
327     ADC1_Initialize();
328     ADC1_ChannelSelect(channel_AN4);
329     while(ADC1_IsConversionComplete(channel_AN4));
330     convertedValue_AN4 = ADC1_ConversionResultGet(channel_AN4);
331     voltage_AN4 = (convertedValue_AN4 * 3.3)/1023;
332
333
334     ANSA = 0x0000; //set up portA i/o as digital so button S5 will work
335     ANSD = 0x0000;
336     PORTA = 0x00; //clear PORTA
337     TRISA = 0x0000; // set pins of PORTA as output
338     TRISE = 0x0000; //set pins of PORTE as output
339     TRISG = 0xFFFF; //set pins of PORTG as input
340
341
342     //initialize trigger voltages
343     // need a way to manually set voltages
344     //initialize resets
345     //pins are active high
346
347     LATEbits.LATE2 = 0; //Reset 1 low
348     LATEbits.LATE3 = 0; //Reset 2 low
349     //ms_delay(1000);
350     LATEbits.LATE2 = 1; //Reset 1 hi
351     LATEbits.LATE3 = 0; //Reset 2 low
352     ms_delay(1000);
353     LATEbits.LATE2 = 0; //Reset 1 low
354     LATEbits.LATE3 = 0; //Reset 2 low
355     //ms_delay(1000);
```

```
356     LATEbits.LATE2 = 0; //Reset 1 low
357     LATEbits.LATE3 = 1; //Reset 2 hi
358     ms_delay(1000);
359     LATEbits.LATE2 = 0; //Reset 1 low
360     LATEbits.LATE3 = 0; //Reset 2 low
361     //ms_delay(1000);
362     LATEbits.LATE2 = 1; //Reset 1 hi
363     LATEbits.LATE3 = 0; //Reset 2 low
364     ms_delay(1000);
365
366
367
368     TRISAbits.TRISA7 = 1;
369     TRISD = 0xFFFF; // set pushbuttons of PORTD as input
370     LATAbits.LATA0 = 1; // set "targets" '1' -> target raised
371     LATAbits.LATA1 = 0; // '0' -> target lowered
372     LATAbits.LATA2 = 0;
373     LATAbits.LATA3 = 1;
374     LATAbits.LATA7 = 1;
375     //LATAbits.LATA4 = 0;
376     //LATAbits.LATA5 = 0;
377     //LATAbits.LATA6 = 0;
378     //LATAbits.LATA7 = 0; //this command currently does not work
379     bool motorLocation[4] = {true,false,false,true};
380     bool targetHit[4] = {false, false, false, false};
381
```

```
383     //Wireless comms
384     int i, j, count = 1;
385     int k = 5;
386     char demo[25];
387     char session[50];
388     char hitData[25];
389     RPINR19bits.U2RXR = 10;
390     RPOR8bits.RP17R = 5;
391     ms_delay(100);
392     InitU2();
393     ms_delay(100);
394     //sprintf(demo, "Welcome to the demo!");
395     //sprintf(session, "This session will have a duration of %d seconds", k);
396     size_t len_demo = strlen(demo);
397     size_t len_session = strlen(session);
398     for(j = 0; j < len_demo; j++)
399         {
400             putU2(demo[j]);
401         }
402         putU2(0x0A);
403         //ms_delay(10);
```

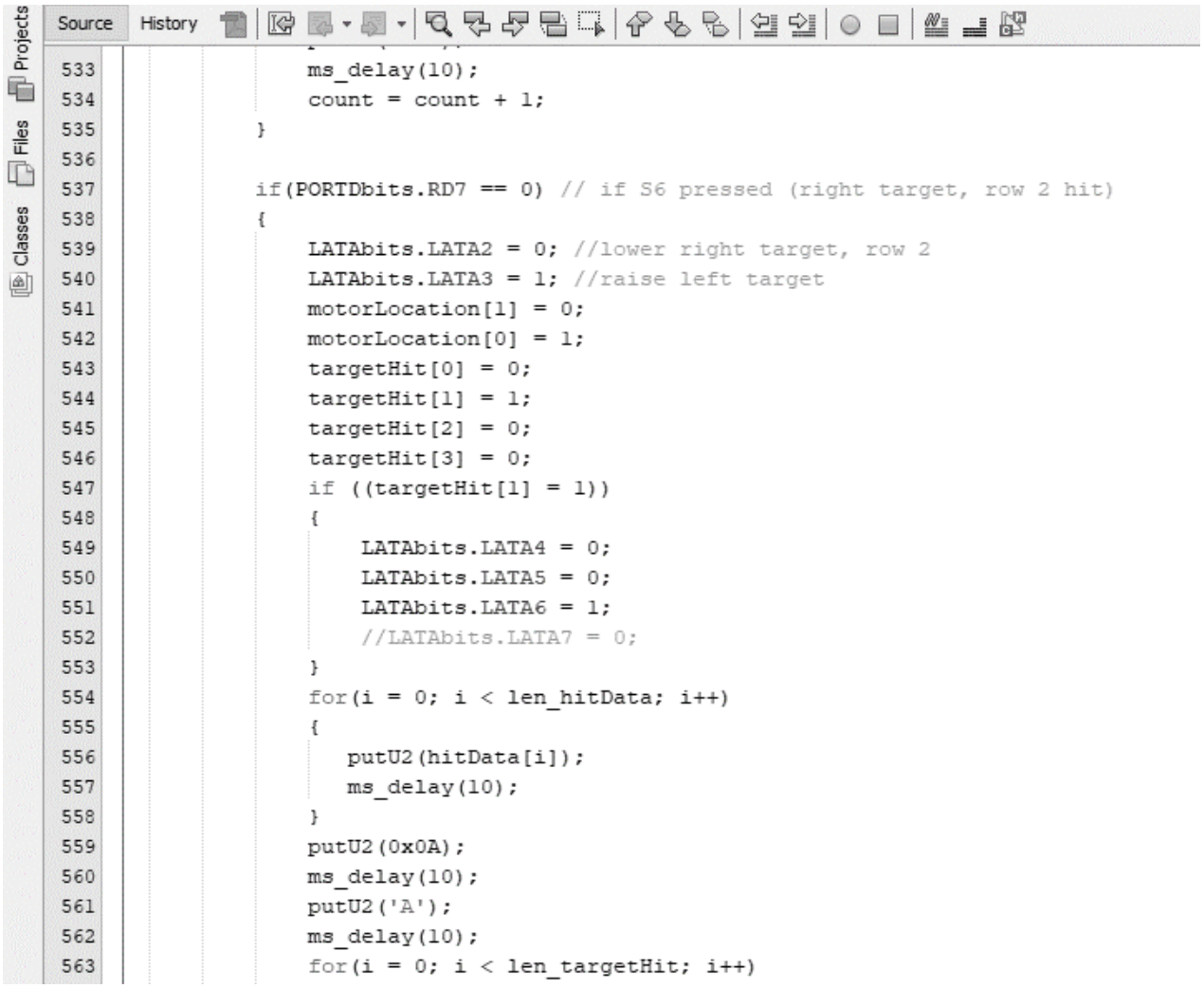
```
406
407     while(1) //main loop start
408     {
409         size_t len_hitData = strlen(hitData);
410         int len_targetHit = sizeof(targetHit)/sizeof(targetHit[0]);
411         int len_motorLocation = sizeof(motorLocation)/sizeof(motorLocation[0]);
412
413         // Initialized target setup:
414         //1001 -> 10
415         //        01
416         if(PORTDbits.RD13 == 0) // if S4 pressed (Right target, row 1 hit)
417         {
418             LATAbits.LATA0 = 0; //lower right target, row 1
419             LATAbits.LATA1 = 1; //raise left target
420             motorLocation[3] = 0;
421             motorLocation[2] = 1;
422             targetHit[0] = 0;
423             targetHit[1] = 0;
424             targetHit[2] = 0;
425             targetHit[3] = 1;
426             if ((targetHit[3] = 1))
427             {
428                 LATAbits.LATA4 = 1;
429                 LATAbits.LATA5 = 0;
430                 LATAbits.LATA6 = 0;
431                 //LATAbits.LATA7 = 0;
432             }
433             for(i = 0; i < len_hitData; i++)
434             {
435                putU2(hitData[i]);
436                ms_delay(10);
437             }
```

```
438             putU2(0x0A);
439             ms_delay(10);
440             putU2('A'); //beginning of string
441             ms_delay(10);
442             for(i = 0; i < len_targetHit; i++)
443             {
444                 if(targetHit[i] == true)
445                 {
446                     putU2('1');
447                 }
448                 else
449                 {
450                     putU2('0');
451                 }
452                ms_delay(10);
453             }
454
455
456
457             ms_delay(10);
458
459             for(i = 0; i < len_motorLocation; i++)
460             {
461                 if(motorLocation[i] == true)
462                 {
463                     putU2('1');
464                 }
465                 else
466                 {
467                     putU2('0');
468                 }
```



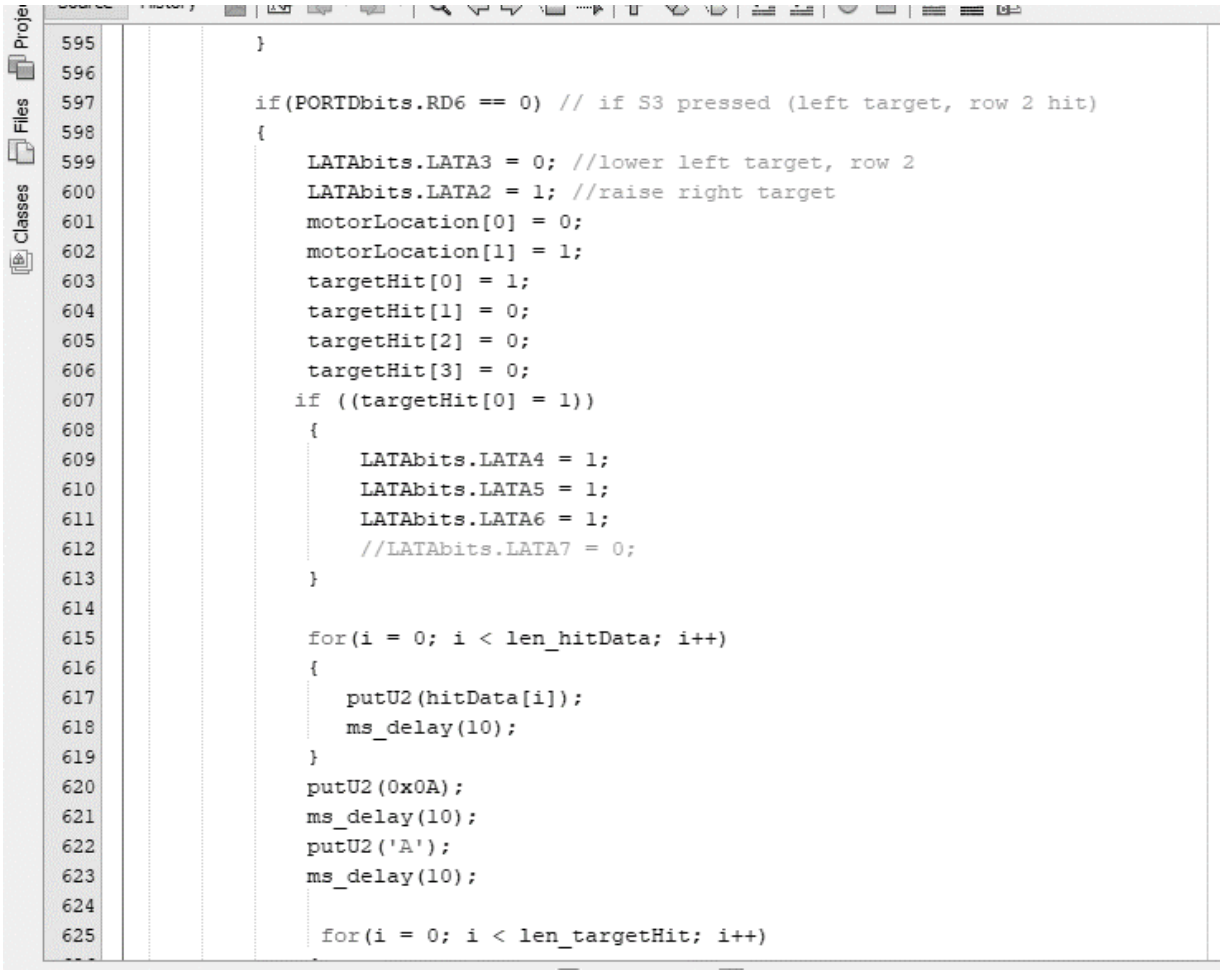

```
471             }
472             putU2('F'); //end of string
473             ms_delay(10);
474
475             putU2(0x0A);
476             ms_delay(1000);
477             count = count + 1;
478         }
479
480         if(PORTAbits.RA7 == 0) // if S5 pressed (left target, row 1 hit)
481         {
482
483             LATAbits.LATA1 = 0; //lower left target, row 1
484             LATAbits.LATA0 = 1; //raise right target
485             motorLocation[2] = 0;
486             motorLocation[3] = 1;
487             targetHit[0] = 0;
488             targetHit[1] = 0;
489             targetHit[2] = 1;
490             targetHit[3] = 0;
491             if ((targetHit[2] = 1))
492             {
493                 LATAbits.LATA4 = 0;
494                 LATAbits.LATA5 = 1;
495                 LATAbits.LATA6 = 0;
496                 //LATAbits.LATA7 = 0;
497             }
498             for(i = 0; i < len_hitData; i++)
499             {
500                 putU2(hitData[i]);
501                 ms_delay(10);
```

```
502             }
503             putU2(0x0A);
504             ms_delay(10);
505             for(i = 0; i < len_targetHit; i++)
506             {
507                 if(targetHit[i] == true)
508                 {
509                     putU2('1');
510                 }
511                 else
512                 {
513                     putU2('0');
514                 }
515                 ms_delay(10);
516             }
517
518             ms_delay(10);
519             for(i = 0; i < len_motorLocation; i++)
520             {
521                 if(motorLocation[i] == true)
522                 {
523                     putU2('1');
524                 }
525                 else
526                 {
527                     putU2('0');
528                 }
529                 ms_delay(10);
530             }
531
532             putU2(0x0A);
```




```
533             ms_delay(10);
534             count = count + 1;
535         }
536
537         if(PORTDbits.RD7 == 0) // if S6 pressed (right target, row 2 hit)
538         {
539             LATAbits.LATA2 = 0; //lower right target, row 2
540             LATAbits.LATA3 = 1; //raise left target
541             motorLocation[1] = 0;
542             motorLocation[0] = 1;
543             targetHit[0] = 0;
544             targetHit[1] = 1;
545             targetHit[2] = 0;
546             targetHit[3] = 0;
547             if ((targetHit[1] = 1))
548             {
549                 LATAbits.LATA4 = 0;
550                 LATAbits.LATA5 = 0;
551                 LATAbits.LATA6 = 1;
552                 //LATAbits.LATA7 = 0;
553             }
554             for(i = 0; i < len_hitData; i++)
555             {
556                putU2(hitData[i]);
557                ms_delay(10);
558             }
559             putU2(0x0A);
560             ms_delay(10);
561             putU2('A');
562             ms_delay(10);
563             for(i = 0; i < len_targetHit; i++)
```

```
564             {
565                 if(targetHit[i] == true)
566                 {
567                     putU2('1');
568                 }
569                 else
570                 {
571                     putU2('0');
572                 }
573                 ms_delay(10);
574             }
575
576             ms_delay(10);
577             for(i = 0; i < len_motorLocation; i++)
578             {
579                 if(motorLocation[i] == true)
580                 {
581                     putU2('1');
582                 }
583                 else
584                 {
585                     putU2('0');
586                 }
587                 ms_delay(10);
588             }
589             putU2('F');
590             ms_delay(10);
591
592             putU2(0x0A);
593             ms_delay(1000);
594             count = count + 1;
```



```
595             }
596
597             if(PORTDbits.RD6 == 0) // if S3 pressed (left target, row 2 hit)
598             {
599                 LATAbits.LATA3 = 0; //lower left target, row 2
600                 LATAbits.LATA2 = 1; //raise right target
601                 motorLocation[0] = 0;
602                 motorLocation[1] = 1;
603                 targetHit[0] = 1;
604                 targetHit[1] = 0;
605                 targetHit[2] = 0;
606                 targetHit[3] = 0;
607                if (targetHit[0] = 1))
608                 {
609                     LATAbits.LATA4 = 1;
610                     LATAbits.LATA5 = 1;
611                     LATAbits.LATA6 = 1;
612                     //LATAbits.LATA7 = 0;
613                 }
614
615                 for(i = 0; i < len_hitData; i++)
616                 {
617                    putU2(hitData[i]);
618                    ms_delay(10);
619                 }
620                 putU2(0x0A);
621                 ms_delay(10);
622                 putU2('A');
623                 ms_delay(10);
624
625                  for(i = 0; i < len_targetHit; i++)
```

```
626             {
627                 if(targetHit[i] == true)
628                 {
629                     putU2('1');
630                 }
631                 else
632                 {
633                     putU2('0');
634                 }
635 
636                 ms_delay(100);
637             }
638 
639             ms_delay(10);
640               for(i = 0; i < len_motorLocation; i++)
641             {
642                 if(motorLocation[i] == true)
643                 {
644                     putU2('1');
645                 }
646                 else
647                 {
648                     putU2('0');
649                 }
650                 ms_delay(10);
651             }
652             putU2('F'); //termination char
653             ms_delay(10);
654 
655             putU2(0x0A);
656             ms_delay(1000);
```

```
            count = count + 1;
        }

        if(PORTGbits.RG0 == 1) // if Left Sensor signal high
        {
            // Right Sensor signal low
            LATEbits.LATE2 = 0; //Reset 1 low
            LATEbits.LATE3 = 0; //Reset 2 low
            ms_delay(1000);
            LATEbits.LATE2 = 0; //Reset 1 low
            LATEbits.LATE3 = 1; //Reset 2 hi
            LATAbits.LATA0 = 0;
            LATAbits.LATA1 = 1;
            ms_delay(1000);
        }

        if(PORTGbits.RG14 == 1) // if Right Sensor signal high
        {
            // Left Sensor signal low
            LATEbits.LATE2 = 0; //Reset 1 low
            LATEbits.LATE3 = 0; //Reset 2 low
            ms_delay(1000);
            LATEbits.LATE2 = 1; //Reset 1 hi
            LATEbits.LATE3 = 0; //Reset 2 low
            LATAbits.LATA0 = 1;
            LATAbits.LATA1 = 0;
            ms_delay(1000);
        }

    }

    return 0;
}
//RPINR19 = 0x210A;
//RPINR19bits.U2CTSR = 33;
```

Figure 53: Screenshots of the embedded systems Code

Communications Design [SV]

Transparent UART and Bluetooth Low Energy [SV, TW]

Because the user relies on instantaneous feedback presented on an interface a safe distance away from a hot range, the DART system hinges on the successful and efficient transmission of data. To bridge the physical gap,

Bluetooth Low Energy utilizing transparent UART has been chosen. Bluetooth allows the user to receive pertinent information while being physically away from the device, and transparent UART allows for a free-flowing conversation to occur between the microcontroller and the interface. Whenever the RN4870 module receives data from the Explorer, the module simply transmits the data with no further processing or analyzing, acting as a transparent device that just acts as a gateway to transmit or receive data; a similar process is done when the module receives data from an external source, such as user interface.

JavaScript Implementation of Google's Web Bluetooth API [SV, TW]

After thorough research on the most efficient communication between the embedded system and the back-end user interface which utilized Bluetooth Low Energy, it has been determined that utilizing Google's Web Bluetooth API will satisfy all pertinent needs of the user and the developer. Although the Web Bluetooth API is only available on Google Chrome browser, every single device that can run Google Chrome will be able to utilize the interface. Because this greatly increases the portability and eases the stringent requirements some applications place, the user interface was ultimately designed exclusively for Google Chrome.

The API was built to connect and communicate with Bluetooth Low Energy devices with the Generic Attribute Profile (GATT). To fully automate the interface and minimize and user input, the user interface is primarily written with JavaScript asynchronous functions, allowing the program to "await" responses from its intended recipient.

When a user first attempts a connection to the embedded system, an async function is called that initializes the connection. Using the Service and Characteristic UUIDs determined when configuring the Transparent UART as inputs for the navigator.bluetooth.requestDevice function, a device object is returned which is stored as the primary device. Using the device object as an parameter for GATT.connect and using that output as an parameter to connect specifically to transmit (Tx) and receive (Rx) characteristic, receiving GATT notifications that are pertinent to the user can be initialized. By using an event listener that listens to when notifications are received, the raw data sent by the embedded system can be deciphered.

The event listener results in a function call to another function that uses a Text Decoder to translate the raw data into an intelligible format. Because UART works by sending one character at a time, the decoded data is “pushed” into an array to preserve the order of characters. Because the embedded system design compressed all important data into one array that was a fixed length of 11 bits, the array is shifted once the maximum length of 11 was detected; this was created to ensure no data was mismanaged and every transmission was being decoded and parsed accurately.

Once the entire transmission of 11 bits was contained in an array and the array was pushed correctly, a synthetic event is dispatched which separates and sorts the bits into the respective data categories (which targets have been hit and which targets are currently active) which are stored as separate arrays. The completion of sorting the data into the two arrays calls a function that graphically

notifies the change of location for each target and notifies which targets have been hit.

User Interface [TW, SV]

The end product of the user interface for the DART system will be implemented on a custom domain that can be accessible from anywhere; for a proof of design and concept, the user interface is currently hosted locally.

Upon entrance into the interface, the user will see this screen:

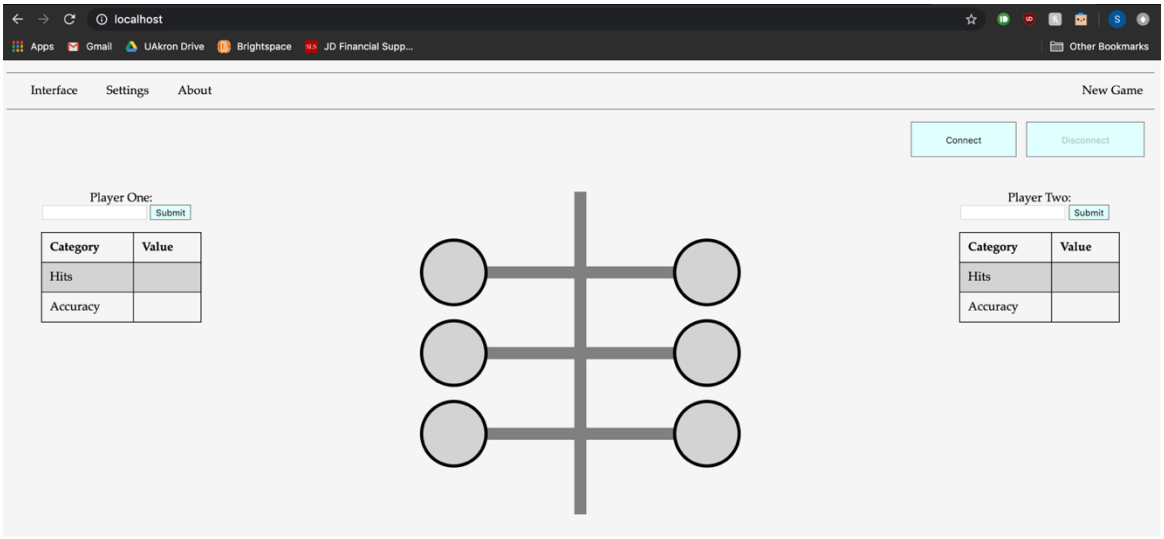


Figure 54: Default User Interface

The interface will require a connection to the embedded system; by clicking the light blue connect button, the following prompt (from Google Bluetooth API) will appear:

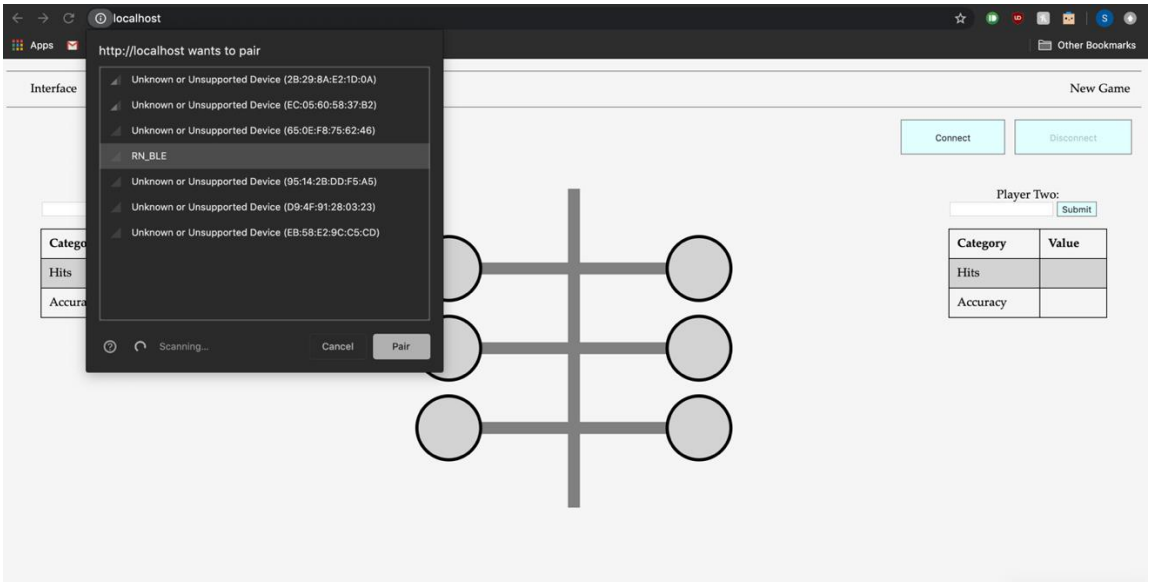


Figure 55: Pairing Window

Upon selecting the appropriate Bluetooth device, Google Chrome will pair with the device, which can be confirmed by noticing a green text (on the left side) indicating what device the interface has connected to.

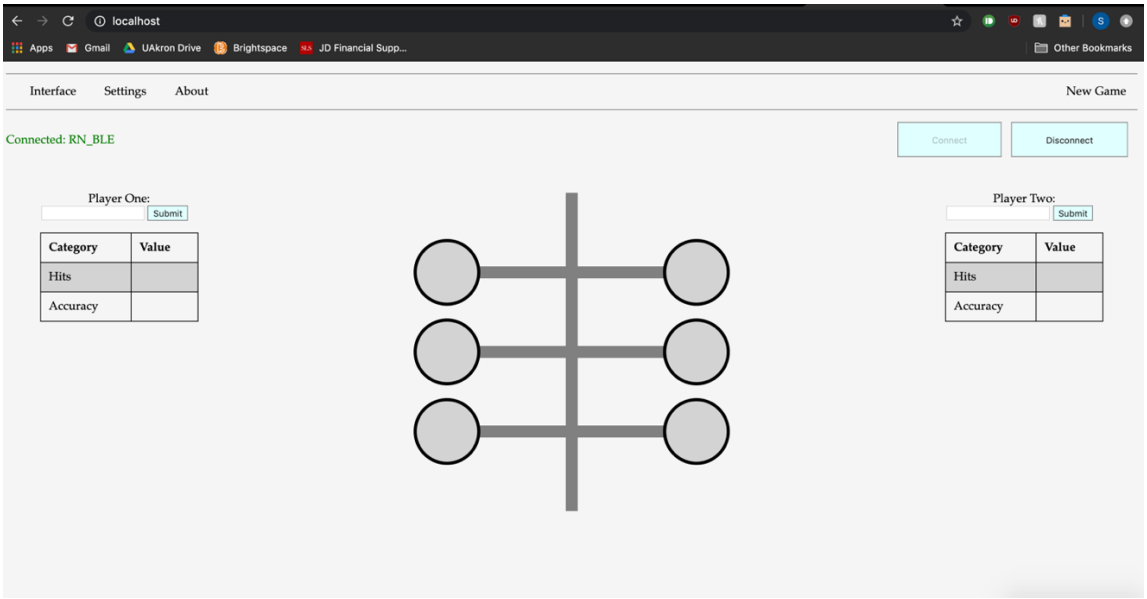


Figure 56: Connected Text

After typing the names of both players (and clicking submit), the user interface will show which targets are active by coloring its corresponding target symbol on the interface.

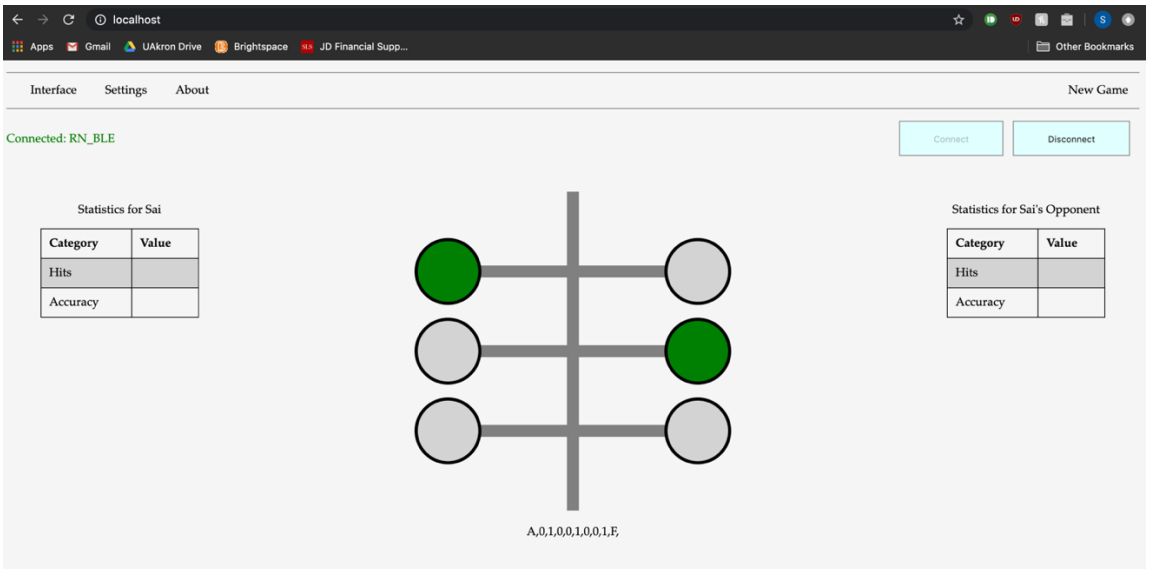


Figure 57: Example of Active Targets

Because integration between the software and the physical target was untestable, the hit and accuracy statistics along with the simulated hit color (red on the circular target symbol) were not showable; however, the active target moving locations can still be simulated with software.

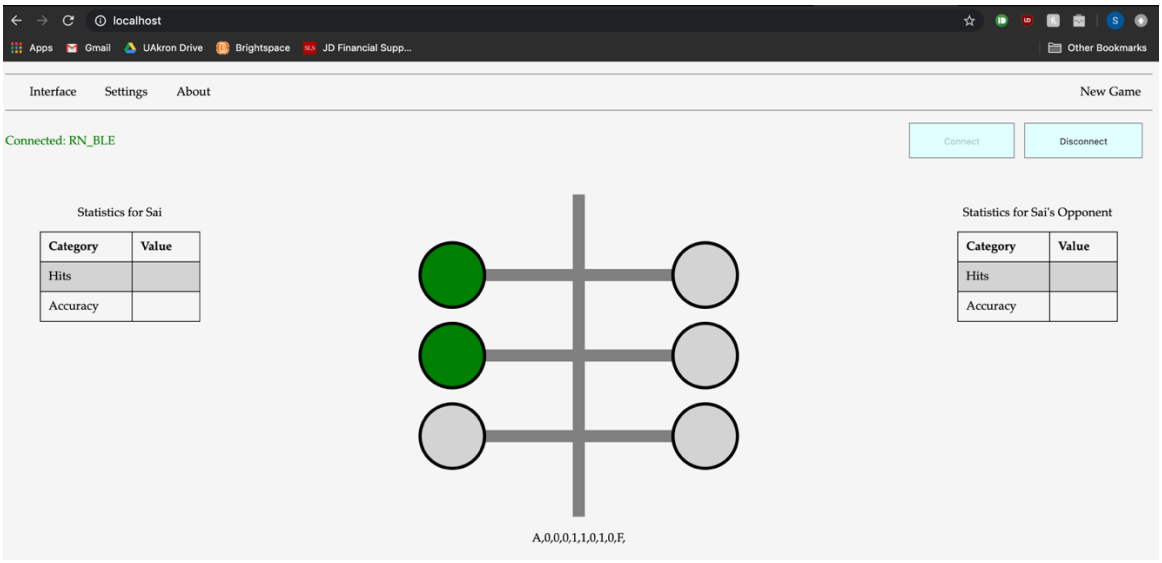


Figure 58: Active Target Changed

After the user is finished with their current session, they can terminate their Bluetooth connection by clicking the light blue Disconnect button and the status of their connection will be updated in the left corner with red text.

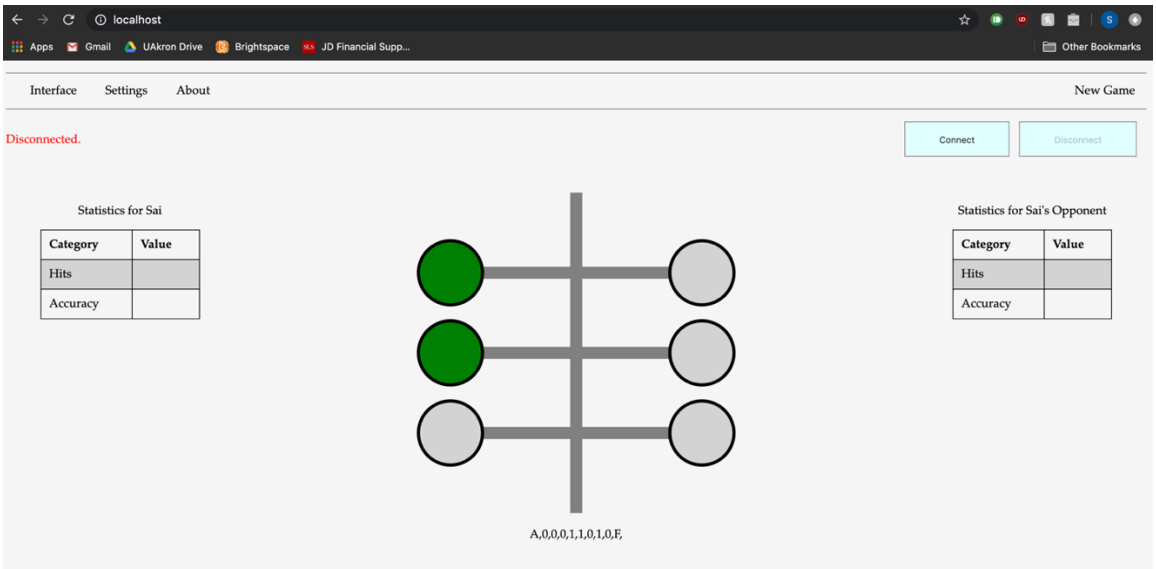


Figure 59: Disconnected Text

Team Information

Nicholas Haas, Project Leader, Electrical Engineer, ESI: No

Thomas Martin, Hardware Design Manager, Electrical Engineer, ESI: No

SaiPranay Vellala, Archivist, Computer Engineer, ESI: Yes

Trandon Ware, Software Lead Manager, Computer Engineer, ESI: Yes

Mechanical Sketches [TM]

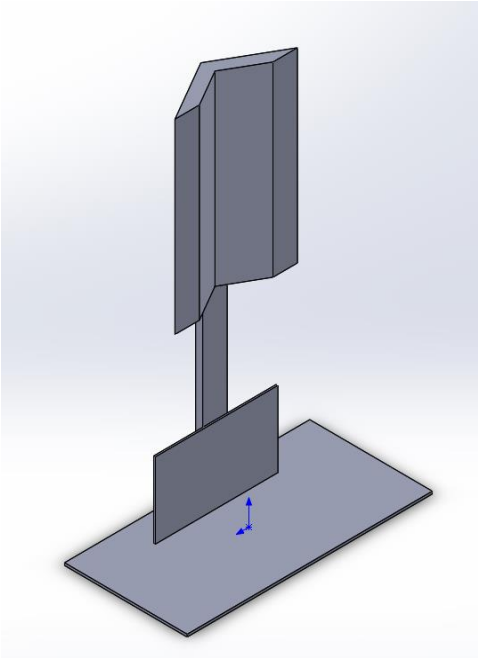


Figure 60: Mechanical Drawing of Target Stand

A custom bracket will be needed to complete the motor controller. The motor will spin a worm drive gear system. A spindle will be attached to the other end of the worm drive. This spindle will both turn the target and a potentiometer. The potentiometer will act as the feedback to the motor controller.

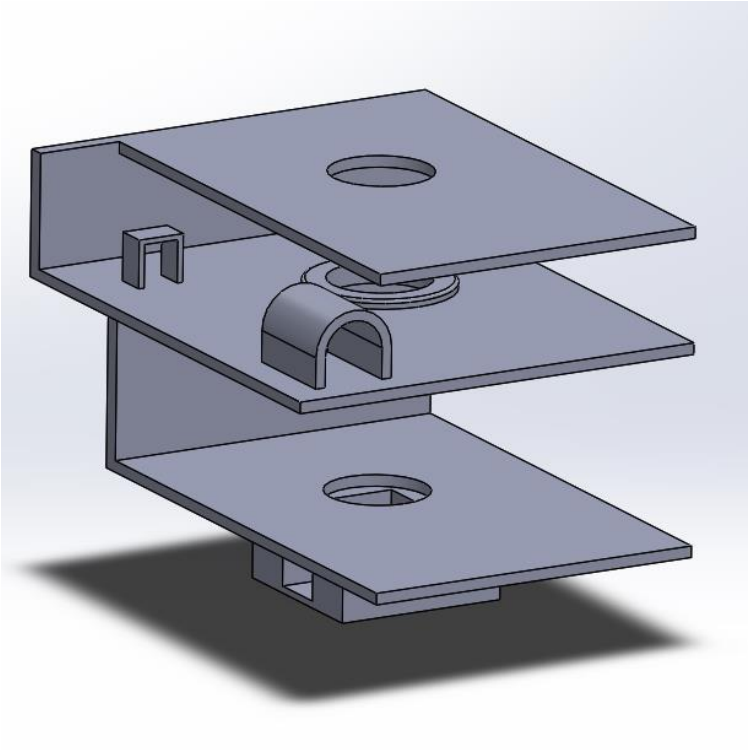


Figure 61: Drawing of the right side motor and worm gear bracket.

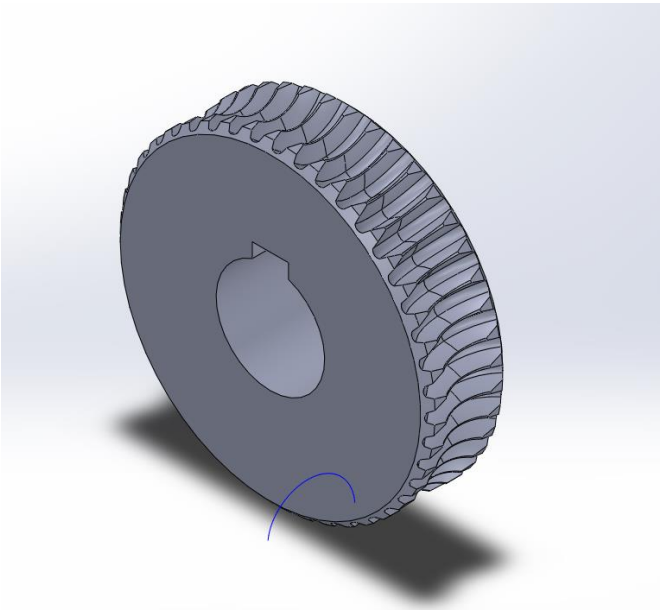


Figure 62: Drawing of the drive gear that will operate the spindle

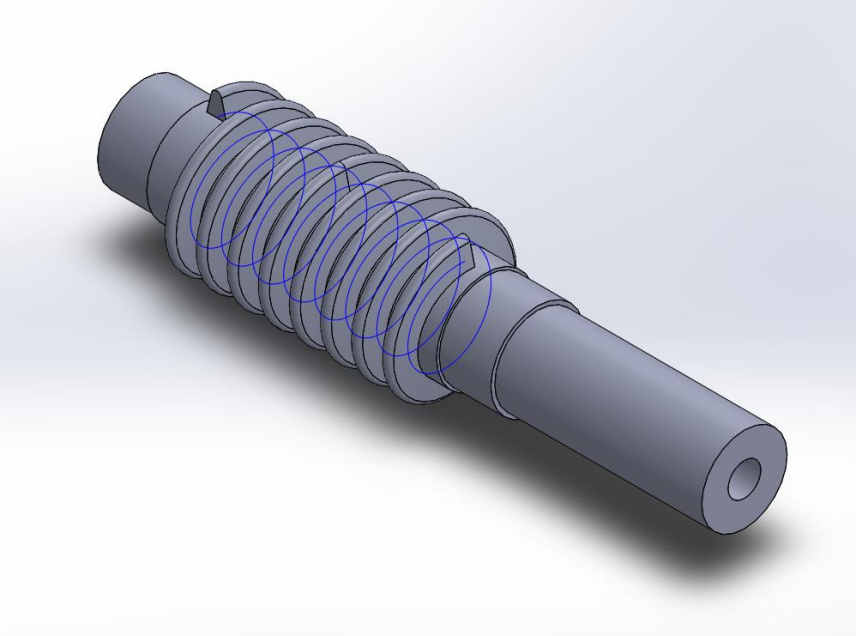


Figure 63: Drawing of the worm gear that will be driven by the motor.

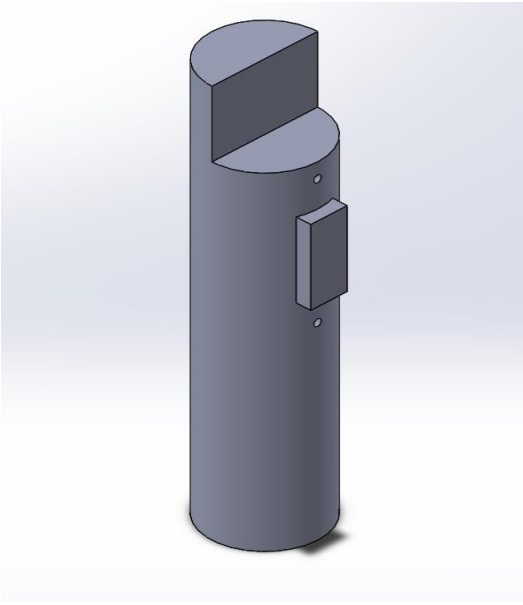


Figure 64: Drawing of the spindle that will move the target and potentiometer.

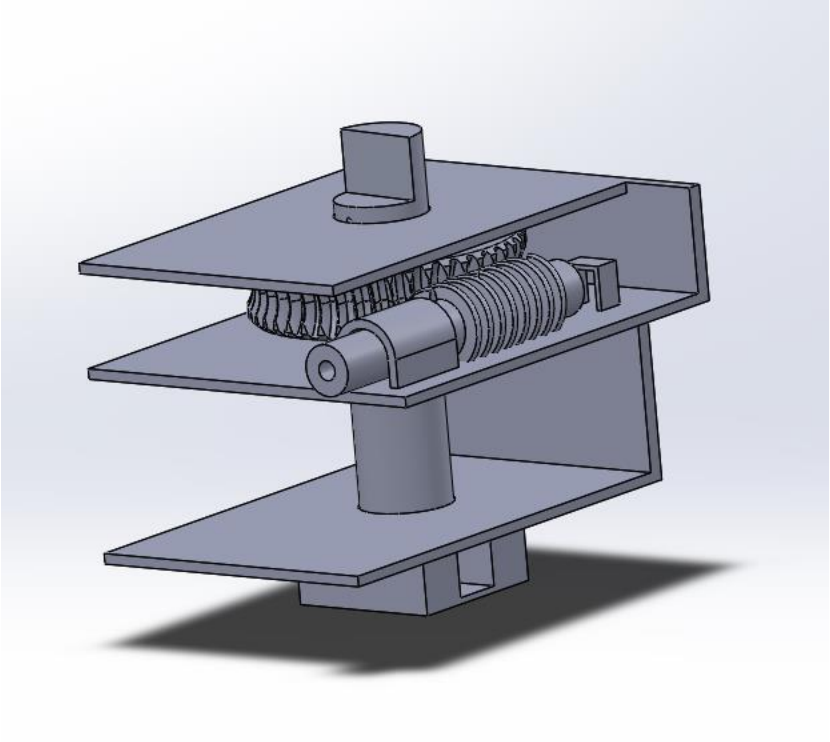


Figure 65: Drawing assembly of the motor bracket

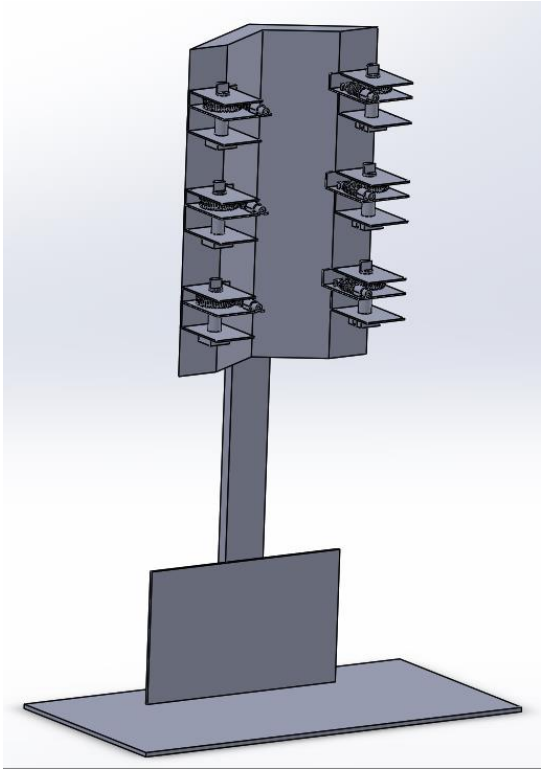


Figure 66: Drawing of how the motor brackets will sit on the target frame.

Structural Changes to the Design

This semester saw a few changes to the structural design of the targeting system. Most significantly was an enlargement of the actuator structure. This allowed for the inclusion of three ball bearings to act between the structure and the shaft. This concept came about after realizing that the friction resulting from a 3D printed shaft and structure could cause issues. These issues include adding extra stress to the motor, to altering the position of the target when the motor finishes being actuated by taking energy out of the system through heat generation. The ball bearings therefore should help to medicate this effect, and enable the system to run with far less required power.

The second change to the structural design was in the joints. Instead of attempting to 3D print most of the actuating structure as one rigid structure, it was decided to utilize bolts, piecing the shaft and target arm at 45 degrees and putting a bolt with a nut on the end in-between them. This was done to make printing and assembly of the structure easier. In addition if the target arm was ever shot by mistake, it means you would only have to print a new target arm, not the whole assembly again, streamlining replacement of parts. Unfortunately the Solidwork part files were left on the lab computer in ASEC and unable to be retrieved after the COVID-19 shutdown, so replication to show the improved design was not possible.

The next step for this part of the project was to 3D print the actuating structure and implement it with the motor circuit. After that was accomplished work on the large steel tree could begin, were some welding and assembly would be required.

Project Budget and Parts

Parts List Table

Qty.	Refdes	Part Num.	Description
1	UC1	EVAL-ADXL326Z	Small, Low power, 3 axis, +/- 16 g Accelerometer
11	U1,U2,U3,U4	UA747CN	LPD OP AMP
66	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28	691104	10K 0.25W Resistor
13	C1, C2, C3, C4, C5, C6	1946261	47uf Capacitor
20	D1,D2, D3, D4, D5, D6	1N4004	Diode
19	U2,U3,U4,U5, U6, U7, U8, U9, U10	LM358	LPD OP AMP
6	J1, J2, J3, J4, J5, J6	ADXL325BCPZ-RL8	ACCELEROMETER 5G ANALOG 16LFCSP
3	NH	EC2-3TNU	RELAY GEN PURPOSE DPDT 2A 3VDC
6	NH	LP2950ACZ-3.0G	IC REG LINEAR 3V 100MA TO92-3
5	TM	91A1A-B28-A18L	50kohm 2W potentiometer
6	TM	TIP32A	Tip 32A PNP Transistor
6	TM	TIP31C	TIP 31C NPN Transistor
6	NH	A1401-ND	3 pin Plug
6	NH	A1400-ND	3 pin Cap
6	NH	A1405-ND	6 pin Plug
6	NH	A1404-ND	6 pin Cap
12	NH	A1420-ND/A1421-ND	Socket/ Pin
1	NH	LT1084CT-12#PBF	12VDC Regulator
2	NH	8KPEPJF4	Breakout for Sensor PCB
1	NH	8KPEPJF5	Sensing Ffilter PCB
6	TM	EMC12565	12VDC Motors

Table 24: Final Parts List

Material Budget List

In the end we were able to use less than half of the allowed $400 budget. There was still more that would have been required to be ordered, including the mechanical frame. However this would not have taken the rest of the budget. For this project we were able to successfully adhere to the allotted budget.

Qty.	Part Num.	Description	Cost	Cost
1	EVAL-ADXL326Z	Small, Low power, 3 axis, +/- 16 g Accelerometer	$31.50	$31.50
11	UA747CN	LPD OP AMP		
66	691104	10K 0.25W Resistor		
13	1946261	47uf Capacitor		
20	1N4004	Diode		
19	LM358	LPD OP AMP		
3	CD4051BCN	8 Channel Analog Mux		
3	TIP32A	PNP Transistor		
6	ADXL325BCPZ-RL8	ACCELEROMETER 5G ANALOG 16LFCSP	3.54	21.24
3	EC2-3TNU	RELAY GEN PURPOSE DPDT 2A 3VDC	2.69	8.07
6	LP2950ACZ-3.0G	IC REG LINEAR 3V 100MA TO92-3	0.51	3.06
5	91A1A-B28-A18L	50kohm 2W potentiometer	5.91	29.55
6	TIP32A	Tip 32A PNP Transistor		
6	TIP31C	TIP 31C NPN Transistor		
6	A1401-ND	3 pin Plug		
6	A1400-ND	3 pin Cap		
6	A1405-ND	6 pin Plug		
6	A1404-ND	6 pin Cap		
12	A1420-ND/A1421-ND	Socket/ Pin		
1	LT1084CT-12#PBF	12VDC Regulator	10.60	10.60
2	8KPEPJF4	Breakout for Sensor PCB	7.00	14.00
1	8KPEPJF5	Sensing Ffilter PCB	13.80	13.80
6	EMC12565	12VDC Motors	10.26	61.56
			Total	$193.38

Table 25: Final Material Budget List

Project Schedule

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Add New Column
1		**Project Design**	**173 days**	**Fri 8/30/19**	**Tue 4/28/20**		**Nick,Sai,Tom,Trandon**	
2		Midterm Report	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon	
3		Cover page	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon	
4		T of C, L of T, L of F	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon	
5		**Problem Statement**	**28 days**	**Fri 8/30/19**	**Tue 10/8/19**		**Nick,Sai,Tom,Trandon**	
6		Need	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon	
7		Objective	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon	
8		Background	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon	
9		Marketing Requirements	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon	
10		Engineering Requirements Specification	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon	
11		**Engineering Analysis**	**28 days**	**Fri 8/30/19**	**Tue 10/8/19**		**Nick,Sai,Tom,Trandon**	
12		**Circuits (DC, AC, Power, ...)**	**28 days**	**Fri 8/30/19**	**Wed 10/9/19**			
13		**Sensing**	**10 days**	**Thu 9/26/19**	**Wed 10/9/19**		**Nick,Tom**	
14		**Hit Sensing**	**45 days**	**Thu 9/26/19**	**Wed 11/27/19**		**Nick**	
15		Research Sensors/Analysis	45 days	Thu 9/26/19	Wed 11/27/19		Nick	
16		Spec out Sensors	45 days	Thu 9/26/19	Wed 11/27/19		Nick	
17		Sensor Design	45 days	Thu 9/26/19	Wed 11/27/19		Nick	
18		**Motor Location Sensing**	**45 days**	**Thu 9/26/19**	**Wed 11/27/19**		**Tom**	
19		research sensors	45 days	Thu 9/26/19	Wed 11/27/19		Tom	
20		Spec out Sensors	45 days	Thu 9/26/19	Wed 11/27/19		Tom	
21		Sensor design	45 days	Thu 9/26/19	Wed 11/27/19		Tom	
22		Design Sening Circ	45 days	Thu 9/26/19	Wed 11/27/19		Nick,Tom	
23		**Actuating**	**10 days**	**Thu 9/26/19**	**Wed 10/9/19**		**Tom**	
24		Do Torque Calulations	10 days	Thu 9/26/19	Wed 10/9/19		Tom	
25		Spec out motors	10 days	Thu 9/26/19	Wed 10/9/19		Tom	
26		Investigate mounting options	10 days	Thu 9/26/19	Wed 10/9/19		Tom	
27		**Power**	**1 day**	**Fri 8/30/19**	**Fri 8/30/19**		**Tom,Nick**	
28		Look up power requiremets for each part	1 day	Fri 8/30/19	Fri 8/30/19		Tom,Nick	
29		power consumption calculations	1 day	Fri 8/30/19	Fri 8/30/19		Tom,Nick	
30		Design Voltage Regulators	1 day	Fri 8/30/19	Fri 8/30/19		Tom,Nick	
31		**Aux Circuits**	**1 day**	**Fri 8/30/19**	**Fri 8/30/19**		**Nick**	
32		Lighting	88 days	Fri 8/30/19	Tue 12/31/19		Nick,Tom	
33		Time Counter	88 days	Fri 8/30/19	Tue 12/31/19		Nick,Tom	
34		**Communications (analog and digital)**	**120 days**	**Fri 8/30/19**	**Thu 2/13/20**		**Sai,Trandon**	
35		**Bluetooth Module**	**1 day?**	**Fri 8/30/19**	**Fri 8/30/19**		**Sai**	
36		Research/Review how to com	1 day?	Fri 8/30/19	Fri 8/30/19		Sai	
37		**WiFi**	**41 days?**	**Fri 8/30/19**	**Fri 10/25/19**		**Sai,Trandon**	
38		Research/Review how to com	1 day?	Fri 8/30/19	Fri 8/30/19		Sai	
39		Electromechanics	41 days	Fri 8/30/19	Fri 10/25/19		Sai	
40		Computer Networks	41 days	Fri 8/30/19	Fri 10/25/19		Sai	
41		Embedded Systems	41 days	Fri 8/30/19	Fri 10/25/19		Trandon	
42		**UART Communication**	**1 day?**	**Fri 8/30/19**	**Fri 8/30/19**		**Sai,Trandon**	
43		review UART code/functions a	1 day?	Fri 8/30/19	Fri 8/30/19		Sai,Trandon	
44		**Embedded Systems**	**120 days**	**Fri 8/30/19**	**Thu 2/13/20**		**Sai,Trandon**	
45		**Explorer 16/32**	**1 day?**	**Fri 8/30/19**	**Fri 8/30/19**		**Sai,Trandon**	
46		review datasheet for the expl	1 day?	Fri 8/30/19	Fri 8/30/19		Sai,Trandon	
47		**Code/Programming**	**1 day?**	**Fri 8/30/19**	**Fri 8/30/19**		**Sai,Trandon**	
48		Review Embedded Systems n	1 day?	Fri 8/30/19	Fri 8/30/19		Sai,Trandon	
49		**Accepted Technical Design**	**28 days**	**Fri 8/30/19**	**Tue 10/8/19**		**Nick,Sai,Tom,Trandon**	
50		**Hardware Design: Phase 1**	**28 days**	**Fri 8/30/19**	**Tue 10/8/19**		**Nick,Tom**	
51		**Hardware Block Diagrams Levels 0 thru N (w/ FR tables)**	**120 days**	**Fri 8/30/19**	**Thu 2/13/20**		**Nick,Tom**	

52			Sensing Block Diagrams	120 days	Fri 8/30/19	Thu 2/13/20		Nick
53			Actuating Block Diagrams	120 days	Fri 8/30/19	Thu 2/13/20		Tom
54			**Software Design: Phase 1**	**28 days**	**Fri 8/30/19**	**Tue 10/8/19**		**Sai,Trandon**
55			Software Behavior Models Levels 0 thru N (w/FR tables)	120 days	Fri 8/30/19	Thu 2/13/20		Sai,Trandon
56			**Mechanical Sketch**	**120 days**	**Fri 8/30/19**	**Thu 2/13/20**		**Nick,Tom**
57			Update Mechanical Sketch	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Tom
58			**Project Schedules**	**120 days**	**Fri 8/30/19**	**Thu 2/13/20**		**Nick,Sai,Tom,Trandon**
59			**Midterm Design Gantt Chart**	**120 days**	**Fri 8/30/19**	**Thu 2/13/20**		**Nick,Sai,Tom,Trandon**
60			Update and fix midterm gantt chart					Nick,Sai,Tom,Trandon
61			References	120 days	Fri 8/30/19	Thu 2/13/20		Nick,Sai,Tom,Trandon
62			**Midterm Parts Request Form**	**28 days**	**Fri 8/30/19**	**Wed 10/9/19**		
63			**Sensing Circuit Materials**	**21 days**	**Thu 9/26/19**	**Thu 10/24/19**		**Nick**
64			Order Sensor	1 day	Thu 9/26/19	Thu 9/26/19		Nick
65			Order Circuit Components	21 days	Thu 9/26/19	Thu 10/24/19		Nick
66			**Actuating Circuits Materials**	**21 days**	**Thu 9/26/19**	**Thu 10/24/19**		**Tom**
67			Order Motor and Componets	21 days	Thu 9/26/19	Thu 10/24/19		Tom
68			Preliminary Design Presentations	0 days	Thu 9/19/19	Thu 9/19/19		Nick,Sai,Tom,Trandon
69			Project Poster	39 days	Thu 10/10/19	Tue 12/3/19		Nick,Sai,Tom,Trandon
70			Final Design Report	144 days	Thu 10/10/19	Tue 4/28/20		Nick,Sai,Tom,Trandon
71			Abstract	144 days	Thu 10/10/19	Tue 4/28/20		Nick,Sai,Tom,Trandon
72			**Hardware Design: Phase 2**	**34 days**	**Thu 10/10/19**	**Wed 11/27/19**		
73			**Modules 1...n**	**34 days**	**Thu 10/10/19**	**Wed 11/27/19**		
74			**Simulations**	**34 days**	**Thu 10/10/19**	**Wed 11/27/19**		
75			**Motor Controller**	**12 days**	**Wed 10/30/19**	**Thu 11/14/19**		
76			PSPICE Schamatics for controller	1 day	Wed 10/30/19	Wed 10/30/19		Tom
77			Design Motor Circuitry	12 days	Wed 10/30/19	Thu 11/14/19		Tom
78			**Accelerometer**	**16 days**	**Wed 10/23/19**	**Thu 11/14/19**		**Nick**
79			Proof of Concept (Sensing Projectiles)	0 days	Wed 10/23/19	Wed 10/23/19		Nick
80			Saimulate PSPICE Sensing Circuit	1 day	Wed 10/30/19	Wed 10/30/19		Nick
81			Design Sensing Circuitry	12 days	Wed 10/30/19	Thu 11/14/19		Nick
82			**Schematics**	**34 days**	**Thu 10/10/19**	**Wed 11/27/19**		
83			Eagle Scamatics for Sensing Circuits	42 days	Thu 10/10/19	Fri 12/6/19		Nick
84			Eagle Scamatics for Motor Controller	42 days	Thu 10/10/19	Fri 12/6/19		Tom
85			**Software Design: Phase 2**	**35 days**	**Thu 10/10/19**	**Wed 11/27/19**		**Sai,Trandon**
86			**Modules 1...n**	**35 days**	**Thu 10/10/19**	**Wed 11/27/19**		**Sai,Trandon**
87			**Simulations**	**35 days**	**Thu 10/10/19**	**Wed 11/27/19**		
88			**Code (working subsystems)**	**35 days**	**Thu 10/10/19**	**Wed 11/27/19**		**Sai,Trandon**
89			**UART Demonstration**	**35 days**	**Thu 10/10/19**	**Wed 11/27/19**		**Sai**
90	✓		Baud Rate Generator	35 days	Thu 10/10/19	Wed 11/27/19		Sai
91	✓		Signal Comparison with Saleae	35 days	Thu 10/10/19	Wed 11/27/19		Sai
92	✓		Rx setup and testing	35 days	Thu 10/10/19	Wed 11/27/19		Sai
93			Tx setup and testing	81 days	Thu 10/10/19	Thu 1/30/20		Sai
94			CTS and RTS Handshaking	81 days	Thu 10/10/19	Thu 1/30/20		Sai
95			**I/O Simulation Demonstration**	**35 days**	**Thu 10/10/19**	**Wed 11/27/19**		
96			Write Code for I/O simulati	35 days	Thu 10/10/19	Wed 11/27/19		Trandon
97			Perform testing with Explorer 16/32	35 days	Thu 10/10/19	Wed 11/27/19		Trandon
98			Create diagrams to analyze LED signals	35 days	Thu 10/10/19	Wed 11/27/19		Trandon
99			**User Interface: Android Application**	**117 days**	**Thu 10/10/19**	**Fri 3/20/20**		**Sai**
100			Bluetooth Terminal	117 days	Thu 10/10/19	Fri 3/20/20		Sai
101			GUI	117 days	Thu 10/10/19	Fri 3/20/20		Sai

102			Control Data Functionality	117 days	Thu 10/10/19	Fri 3/20/20		Sai	
103			Instantaneous Feedback Data Functionality	117 days	Thu 10/10/19	Fri 3/20/20		Sai	
104			System integration Behavior Model	144 days	Thu 10/10/19	Tue 4/28/20			
105			**Parts Lists**	**34 days**	**Thu 10/10/19**	**Wed 11/27/19**			
106			**Parts list(s) for Schematics**	**34 days**	**Thu 10/10/19**	**Wed 11/27/19**			
107			Sensing Circuit	144 days	Thu 10/10/19	Tue 4/28/20		Nick	
108			Actuating Circuit	144 days	Thu 10/10/19	Tue 4/28/20		Tom	
109			Materials Budget list	144 days	Thu 10/10/19	Tue 4/28/20		Nick,Tom	
110			**Final Parts Request Form**	**9 days**	**Tue 10/15/19**	**Fri 10/25/19**		**Nick,Tom**	
111			Order All Actuating Componets	1 day	Tue 10/15/19	Tue 10/15/19		Tom	
112			Order All Sensing Circuit Items	1 day	Tue 10/15/19	Tue 10/15/19		Nick	
113			Order Power Components	1 day	Tue 10/15/19	Tue 10/15/19		Nick,Tom	
114			Final Design Presentations	0 days	Thu 11/14/19	Thu 11/14/19		Nick,Sai,Tom,Trandon	
115			Parts Request Form for Spring Semester	27 days	Wed 11/27/19	Thu 1/2/20		Nick,Sai,Tom,Trandon	

Figure 67: Gantt chart layout of the D.A.R.T System project schedule for Fall 2019.

Conclusions

In all, the work done in the fall semester has acted as a proof of concept for the development of the D.A.RT System. Each of the four sections of the project, Sensing Circuits, Actuating Circuits, Microprocessor Coding and Communications, has been shown to be a reliable avenue for completion of the system through the theoretical simulations, testing and development done over the course of the fall 2019 semester. This team was able to successfully expand on this knowledge and produce working subsystems over the course of the first half of the spring 2020 semester using the knowledge gained to debug and improve upon the theories established in the fall. As it stands, the sensing circuit, Bluetooth transmission and embedded system design simulation of the system behave as expected, and the actuating circuit is undergoing adjustment. These system were prepared for integration that would have led to the completion of the system as a whole. In conclusion, the proof of concept established in this course shows that a complete system would have been created given the allotted amount of time that was lost due to the COVID-19 campus shutdown.

References

[1] F. Morelli, J. Neugebauer, M. LaFiandra, P. Burcham, and C. Gordon, "Recoil Measurement, Mitigation Techniques, and Effects on Small Arms Weapon Design and Marksmanship Performance,"*IEEE Transactions on Human-Machine Systems,* vol. 44, no. 3, June 2014

[2] M.Smith and J. Hagman, "Predicting Rifle and Pistol Marksmanship Performance With the Laser Marksmanship Training System," United States Army Research Institute for the Behavioral and Social Sciences, Alexandria, VA, Tech, Report, Oct 2000

[3] J. Clements, "Predicting Performance during a Dynamic Target Acquisition Task in Immersive Virtual Reality," Department of Electrical and Computer Engineering, Duke University, USA, Mar, 2018

[4] S. Daniels, 'Rotary Target', US687873A, Dec. 3, 1901

[5] E. Steil, 'Reactive Target System', US9470482B2, Oct. 18, 2016

[6] T. Kandir, M. Shechter, and J. Clark, 'Firearm laser training system and method facilitating firearm training for extended range targets with feedback of firearm control', US7329127B2, February 12, 2008

[7] D. Allen, R. Kilmer, T. Guy, and W. Fosnow, 'Automated scoring target system', US4222564A, Sept. 16, 1980

[8] S. Jihong, Z. Feimeng, L. Yi, and W.Shengsheg, "Research on Key Technologies in Automated Target-Reporting," Hefei New Star Applied Technology, Hefei, Anhui, China

[9] A. Brown, and C. Coelho, "Modeling Goal Directed Movements in Modern Pistol Competition," Banff Center, Banff, Canada, 2017

[10] X. Bui, J. Komenda and R. Vitek, "Frangibility of Fragile Bullet upon Impact on a Hard Target," *2017 International Conference on Military Technologies,* Brno, Czech Republic, 2017

[11] National Shooting Sport Foundation. (2015). *Sport Shooting Participation in the United States in 2014.* Harrisonburg, VA:

[12] “Density of Cardboard in 285 Units and Reference Information.” *Density of Cardboard in 285 Units and Reference Information*, www.aqua-calc.com/page/density-table/substance/cardboard.

[13] Helmenstine, Anne Marie. “A Table of Electrical Conductivity and Resistivity of Common Materials.” *ThoughtCo*, ThoughtCo, 27 June 2019, www.thoughtco.com/table-of-electrical-resistivity-conductivity-608499

[14] IDA, NATHAN. *ENGINEERING ELECTROMAGNETICS*. SPRINGER NATURE, 2019.

[15] Wu, Shih-Jeh, et al. “Measurement of Elastic Properties of Brittle Materials by Ultrasonic and Indentation Methods.” *Applied Sciences*, vol. 9, no. 10, 2019, p. 2067., doi:10.3390/app9102067.

[16] *IEEE Guide for the Design and Installation of Cable Systems in Substations* , IEEE Standard 525, 2007

[17] United States, Congress, Office of Health, Safety and Security, and Larry D Wilcher. “Range Design Criteria.” *Range Design Criteria*, 2012, pp. 1–53.

Appendix I: D.A.R.T System Terminology

Mode - The type of training simulation you wish to conduct.

Session - The completion of any single training mode.

Time Trial - A training mode defined by: Given X minutes, every target starts in the upright position, lowering one at a time as the sensor detects hits. Once every target has been lowered they will return to the upright position.

Whack-A-Mole - A training mode defined by: Targets present themselves in a randomized order one at a time, seeding a new random target number each time the user hits the previous target.

Dueling Tree - A training mode defined by: Only one target per row will show at one time. At the beginning of the event there will be an equal number of targets in each of the two columns (note the targeting system will have 4 rows with 2 columns). Every time a target registers a hit it will disappear and the other target in the row will appear switching back and forth until time runs out.

Hits per Unit Time - This is the primary measurement for accuracy of the user while operating they system. It will be calculated by taking the number of hits in a given session and dividing by the duration of the session. It will be displayed as hits/min.

Parallel Target - The parallel target refers to the target that shares a row with the specified target, but is in the opposite position.