California Polytechnic State University, SLO
College of Engineering
Computer Engineering Program

Roborodentia Final Report

Submitted by:
Zeph Nord, Mitchell Myjak, Trevor Gesell
June 2018

Faculty Advisor:
Dr. John Seng

Table of Contents

Introduction

The Senior Project consisted of competing in Roboredentia, a competition in which groups build robots to complete a particular task. This event took place at the Cal Poly Open House on Saturday, April 12th, 2018. For the competition, the robot was to collect Nerf balls from supply tubes raised approximately 7” from the board and shoot them into nets placed along the opposite side of the course. The design, manufacture, and testing of the robot began the first week of Cal Poly winter quarter and lasted until the day of the competition. The process of creating a robot to compete in the competition will be described in subsequent sections.

Problem Statement

Design a fully autonomous and self-contained robot capable of loading Nerf Rival-like balls and firing them into targets on the opposite side of the course. In order to accomplish this task, a custom chassis, ball collector, and ball launcher must be designed and fabricated, as well as programming a microcontroller to make use of sensors and a motor controller for autonomy.

The autonomous robot must navigate the entire course, loading the Nerf balls from supply drops at each end, afterwards launching them into various nets. Landing the Nerf balls into multiple nets results in higher bonus scoring combos, but may be less efficient for time.

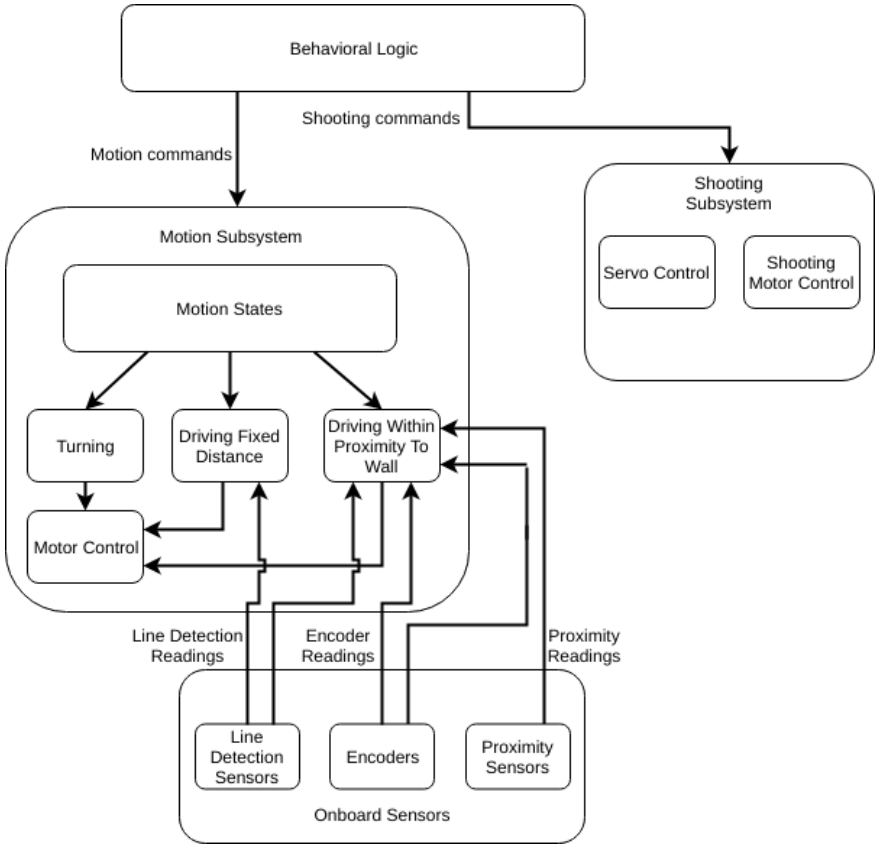
The robot must have a footprint within a 12" x 14" at the start of the match, and at no point be larger than 14" x 17". It must start the match with a height less than 14" but there is no height limit after the match begins. There cannot be interference with other robots, and the robot may not leave the course (become airborne).

There are no constraints on budget on materials used.

Software

Software Architecture Block Diagram

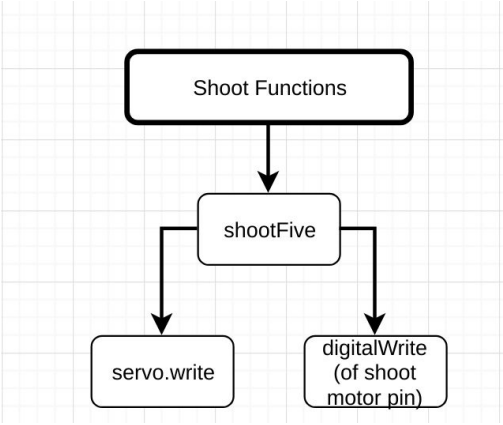
The logic for the system is as follows. The top level logic consisted of either motion commands or shooting commands. The motion subsystem consisted of three different motions states needed to traverse the field. In order to ensure correct traversal, onboard sensors were used with their respective values being sent to the necessary functions. The shooting subsystem maintained control of the servo as well as the shooting mechanism motor.

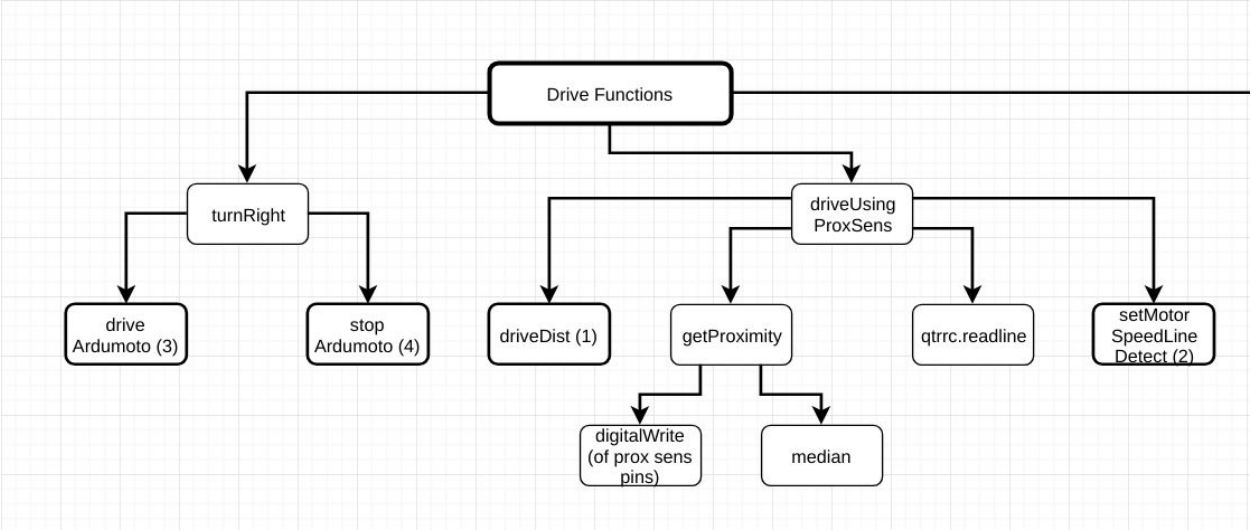


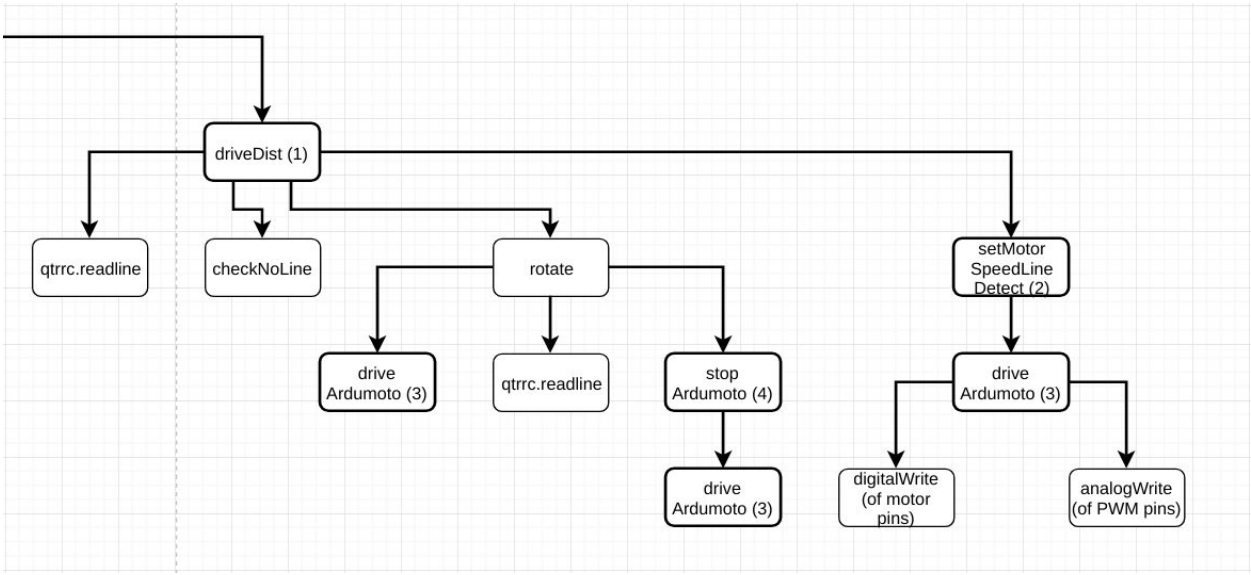
Function Flow Overview

The function flow consisted of two main components, shoot functions and drive functions. In the diagrams below, functions that are used multiple places and contain function calls nested inside of them have bolded boxes and contain a number so other calls to the function can be found.

However, only one instance of each bolded function contains the nested function calls. This is done in order to make the diagram more visually understandable.







Code Functionality

shootFive

- shootFive shoots five balls into the targeted net

servo.write

- servo.write changes the angle of the servo horn to allow the balls to fall through the funnel, thereby being shot by the shooting mechanism

turnRight

- turnRight turns the robot to the right to a specified degree until the encoders read that the motors reached said degree of rotation

driveArdumoto

- driveArdumoto sends a signal of specified magnitude and direction to a motor

stopArdumoto

- stopArdumoto stops a motor by sending a magnitude of zero to driveArdumoto

driveUsingProxSens

- driveUsingProxSens first drives the bot a minimum distance, calling driveDist. It then gets the proximity of the bot to the wall in question by calling getProximity. If the distance to the wall is within a certain margin, then the function ends, ceasing movement of the bot. If the distance to the wall is not within the margin, then the line detector is used by calling qtrrc.readline, and the motors are driven forward using setMotorSpeedLineDetect with the line detector readings passed as a parameter.

driveDist

- driveDist drives the bot a predetermined distance using encoders. It first gets line detector readings, then uses those readings to check if the bot is off the line by calling checkNoLine. If the bot is off the line, rotate is called to rotate the bot back onto the line until the line detector reads that the bot is back on the line. Once it is clear the bot is on the line, the motor speeds are set using setMotorSpeedLineDetect. This continues until one of the encoders reads it has reached the specified destination.

checkNoLine

- checkNoLine determines if the bot is off the line by checking the values of the line detector. If all the values of the six reflectance sensors in the array are less than a minimal value, then a boolean is returned which indicates the bot is off the line to a certain degree of accuracy

getProximity

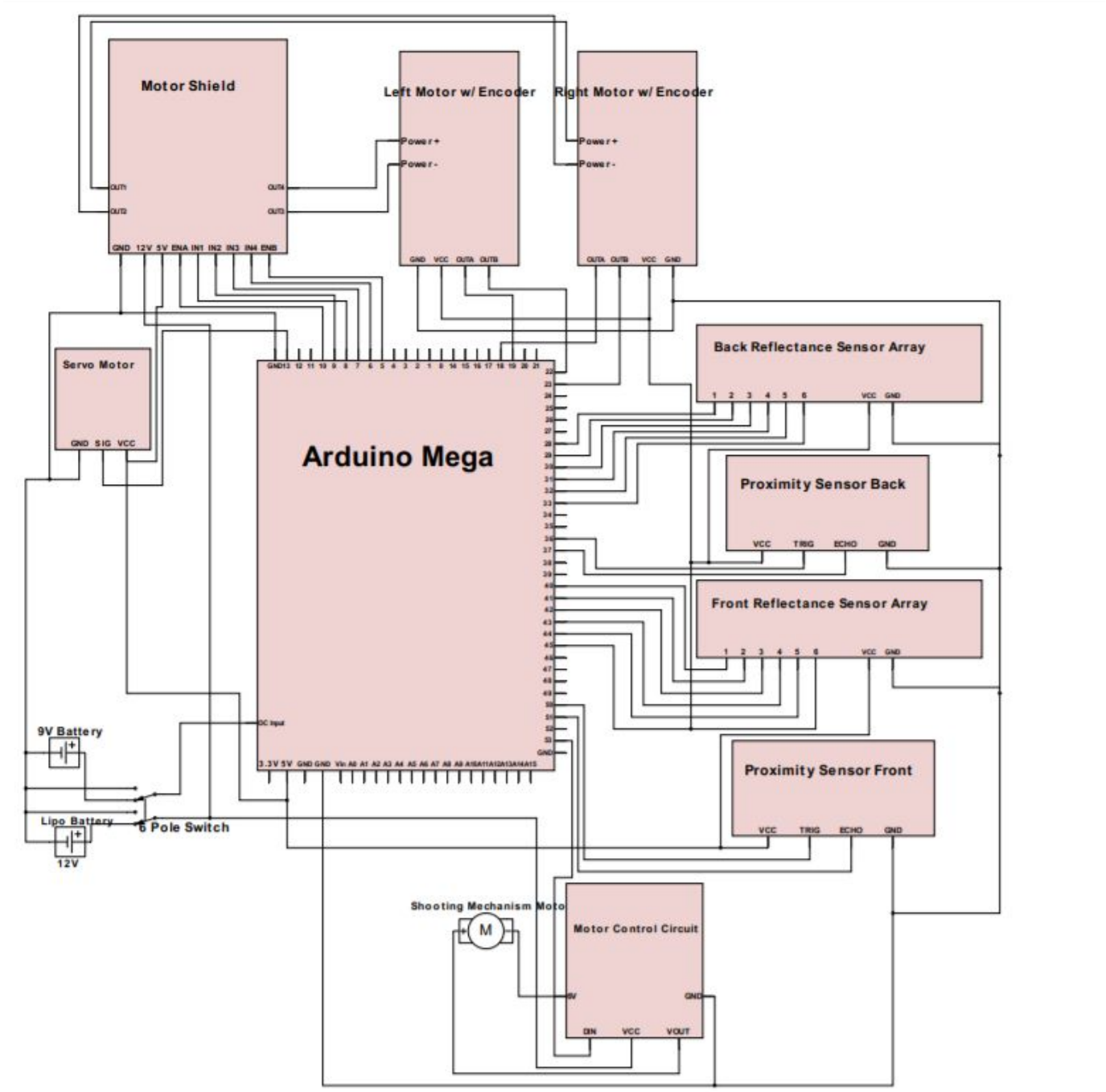
- getProximity gets the distance between the proximity sensor and the wall by sending out an ultrasonic signal, waiting for it to bounce off the wall and return, and divides by a constant to determine the distance. One median reading is found among ten sequential readings in order to reduce noise, and this reading is returned.

setMotorSpeedLineDetect

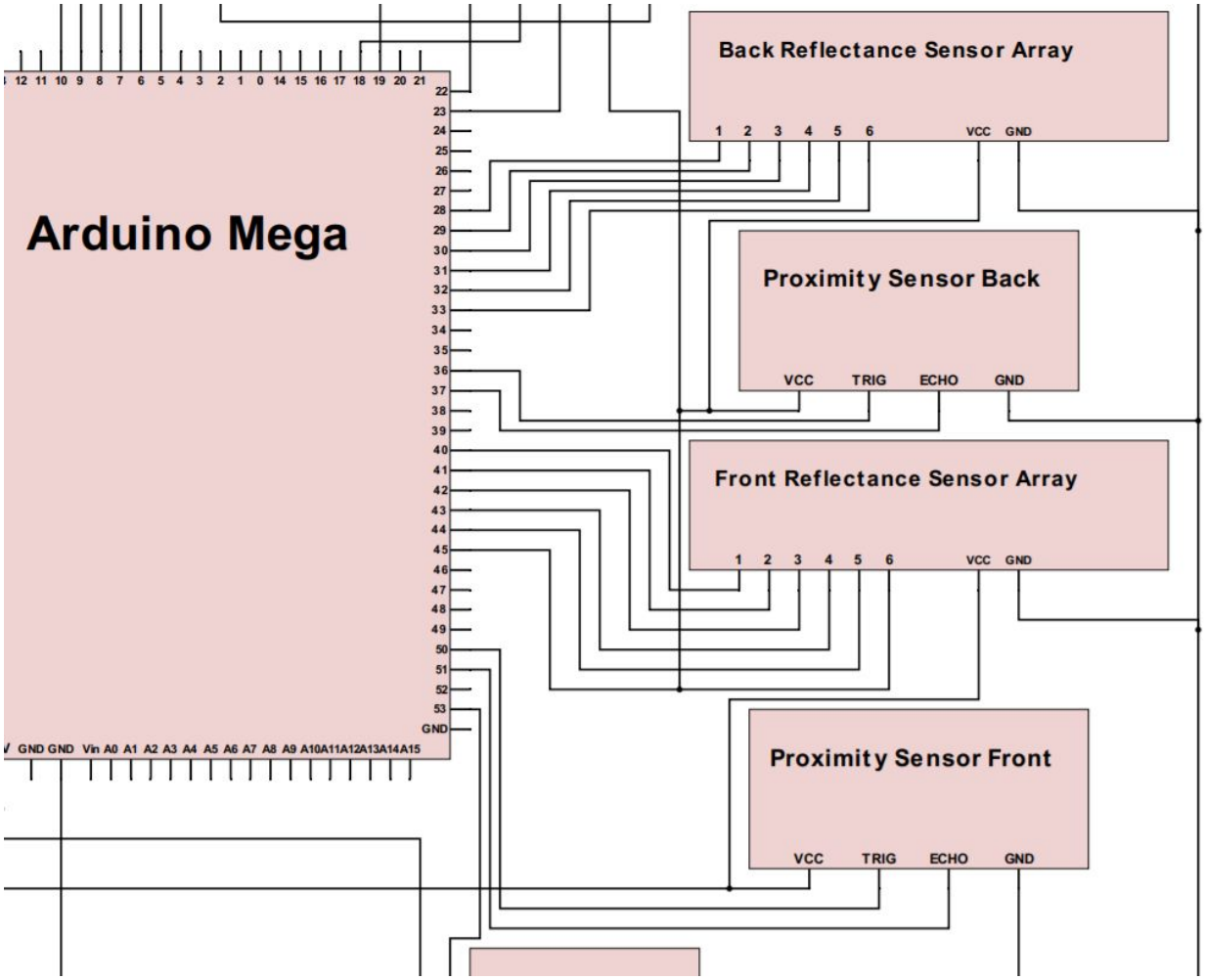
- setMotorSpeedLineDetect moves the left and right motors according to the readings gathered from the line detector. PID control is used in order to set the speeds of the motors accordingly. Once the appropriate values for the motor speeds have been found, driveArdumoto is called to drive the motors.

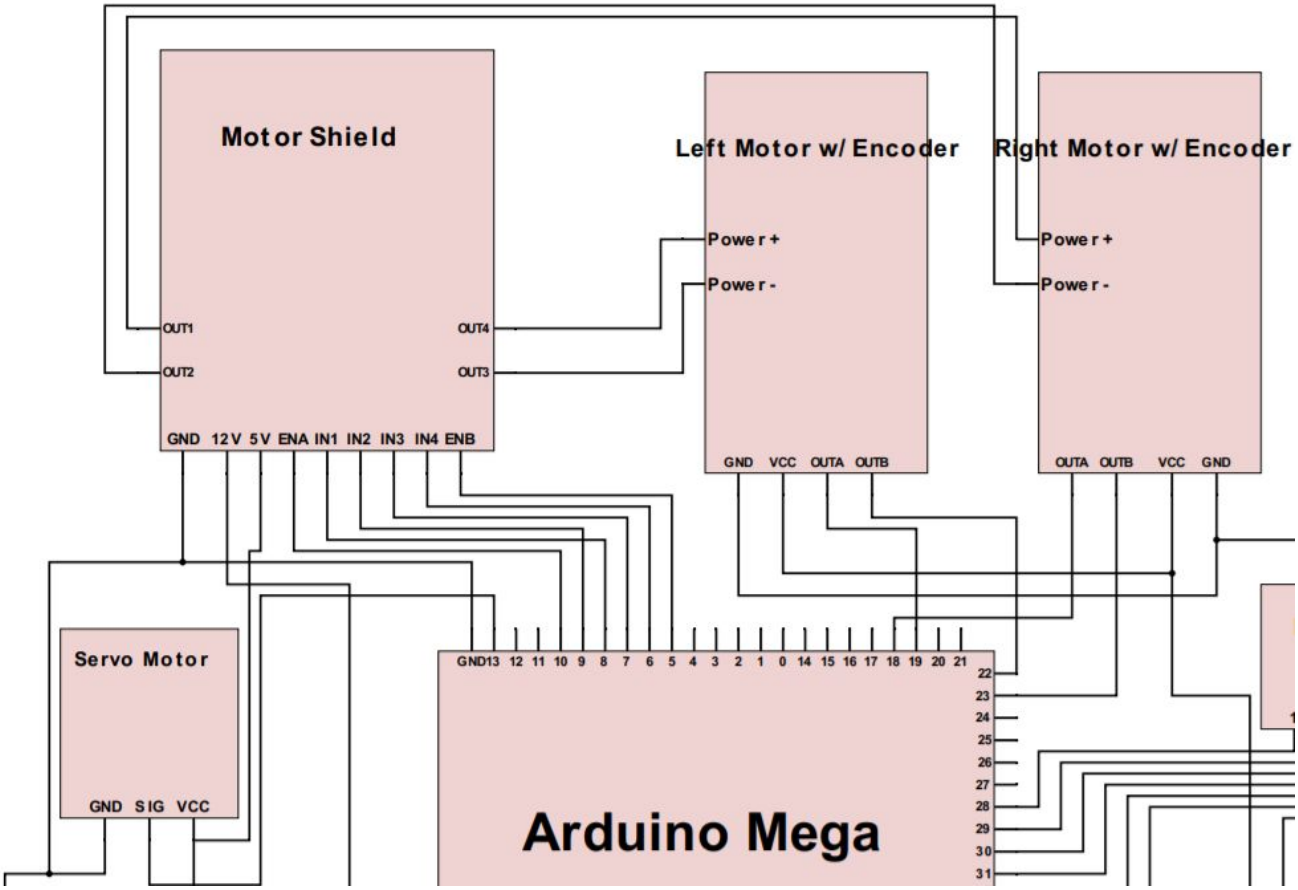
Hardware

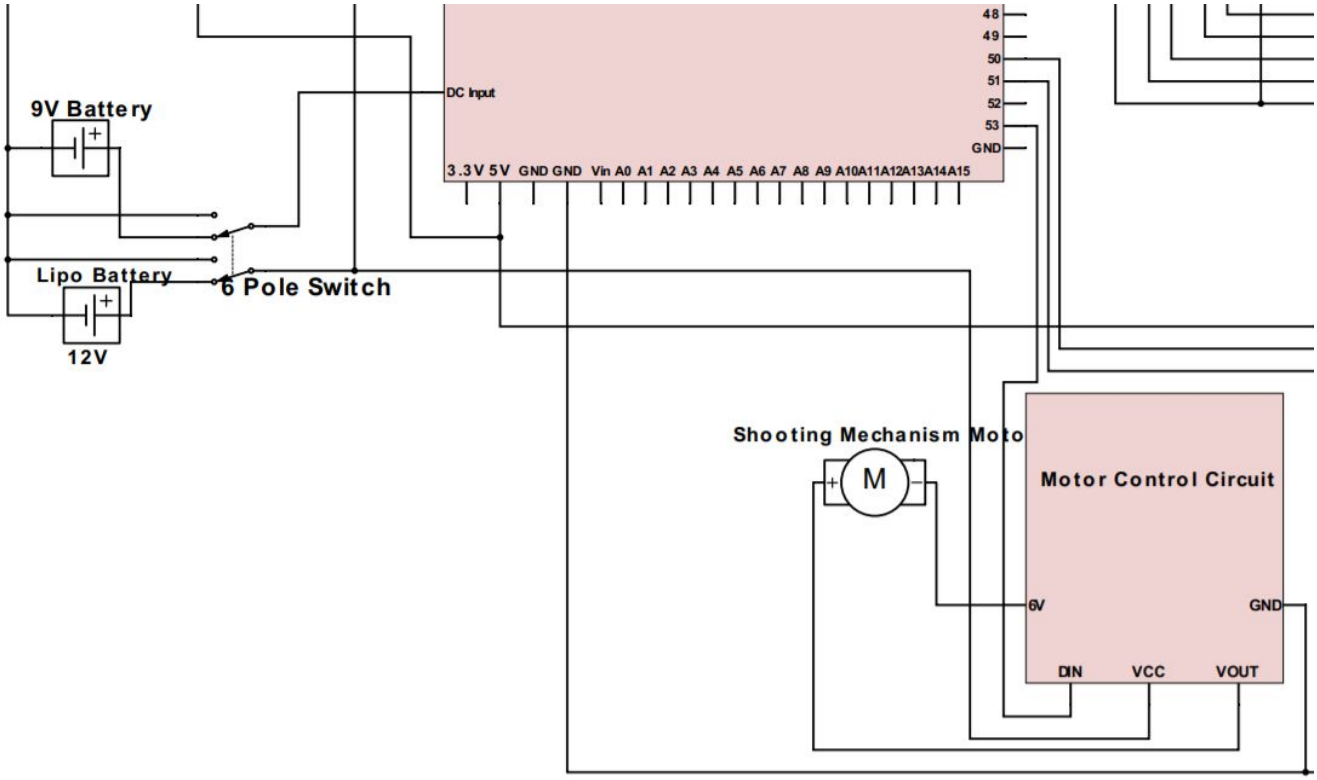
Hardware Architecture Block Diagram



Closer Views of Hardware Architecture Block Diagram Sections




Motor Shield
OUT1
OUT2
OUT4
OUT3
GND 12V 5V ENA IN1 IN2 IN3 IN4 ENB
Left Motor w/ Encoder
Power +
Power -
GND VCC OUTA OUTB
Right Motor w/ Encoder
Power +
Power -
OUTA OUTB VCC GND
Servo Motor
GND SIG VCC
GND13 12 11 10 9 8 7 6 5 4 3 2 1 0 14 15 16 17 18 19 20 21
22
23
24
25
26
27
28
29
30
31
Arduino Mega



Interactions between hardware devices

All of the components directly interface to the Arduino besides the geared DC motors, which are interfaced with the motor shield. How the systems interact with each other is detailed below:

Ultrasonic Module Distance Sensor

The ultrasonic module distance sensor is used to determine the distance from the front or the back of the robot to a wall. The Arduino sets the trig pin of the distance sensor low, waits a couple microseconds, then sets it high, wait a couple more microseconds, and then sets it back low. When this happens, the ultrasonic distance sensor sends out a series of ultrasonic pulses. It waits for an echo from a nearby object and pluses the echo pin high when it is received. The arduino recognizes this pulse and is able to calculate the distance based on how long it took for the ultrasonic distance sensor to pulse the echo pin.

Reflectance Sensors

The reflectance sensors are used to determine where on the black line the robot is. If the robot is off the line, the reflectance sensors can determine this. Also, the reflectance sensors are crucial in

determining the speed of each motor in a way the the center of the line sensors is over the center of the line. In the array of six reflectance sensors, when the Arduino wants to check the positioning of the robot, it sets the pins connected to the reflectance sensors to outputs and sets them high. Then, after waiting a small amount of time, the Arduino sets the I/O pins to inputs and measures the time it takes for the voltage to decay by checking the I/O pins. The shorter the decay time, the greater the reflection is. When the sensors are over the black tape, the reflectance will be low. The Arduino software is able to interpret this and adjust the position of the robot accordingly.

Encoders

The encoders, which are connected to the motors of the robot, tell the Arduino how many times the motor completed a revolution. This allows the Arduino to determine the distance the robot moved by multiplying the revolutions by the circumference of the wheel. The encoder utilizes a hall effect sensor to determine the number of revolutions the motor has completed. By counting the number of rising edges of one channel, one revolution is completed for every sixteen rising edges. Essentially, the frequency of the square wave outputted from the encoder is sixteen times the frequency of the motor rotation frequency. In addition, the phase shift between the A and B inputs can determine if the wheel is moving forwards or backwards.

DC motor

The DC motor receives its input from the motor controller circuit. This input is a voltage drop across the motor which is always 6V when the motor is to run and 0V when the motor is to not run. The motor controller circuit essentially pulls up the output from the Arduino so that it can run the motor at a speed faster than that if it could only be provided 5V.

Geared DC motor

Like the DC motor, the geared DC motors receive their input in the form of a voltage drop across its two terminals. This voltage drop dictates the speed at which the motor is to rotate. The geared DC motors receive their input from the motor shield.

Servo Motor

The servo motor is used to control the balls dropping through the funnel. When it is time to shoot the balls, the servo motor will turn, causing balls to fall through the hole in the funnel. The servo motor receives its input in the form of a PWM signal. The angle in which to turn the shaft is determined by the duty cycle of the PWM signal. The higher the duty cycle, the more the shaft

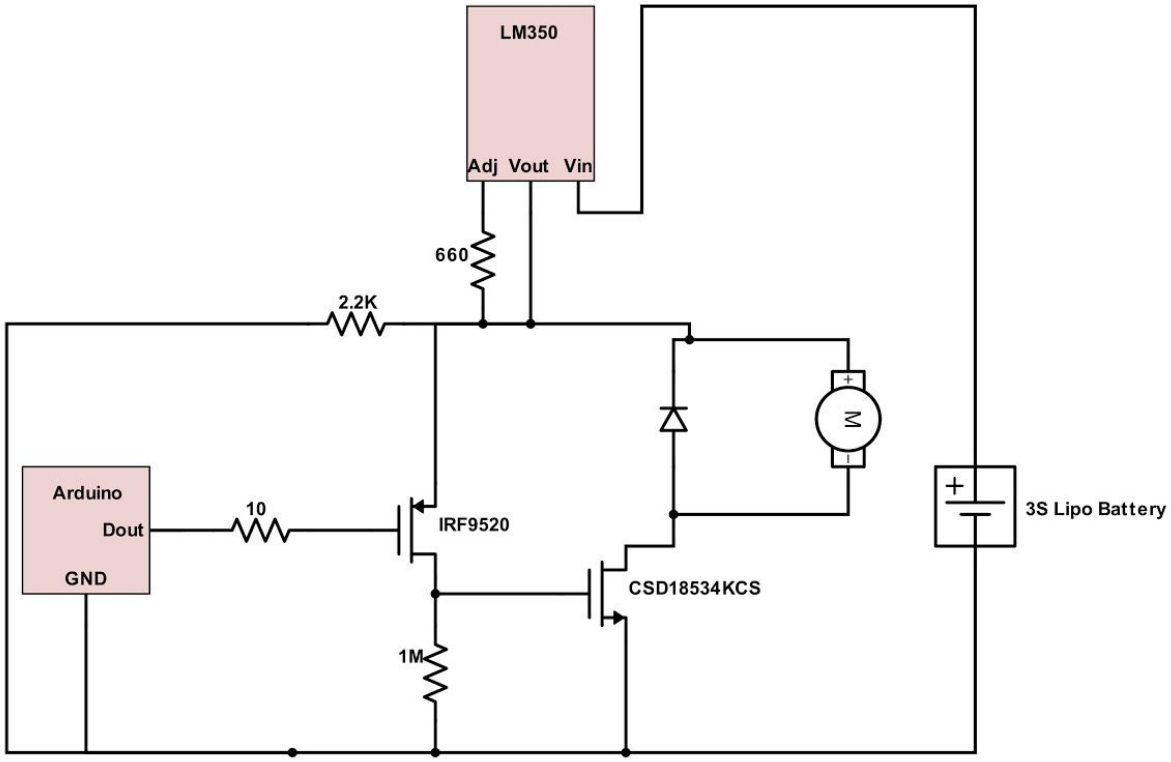
will turn. The servo motor receives this PWM input from the Arduino. The Arduino will set the angle for the servo motor to turn to based on a PWM signal it sends out an one of its output pins.

Motor Shield

The motor shield is used to power the motors, allowing them to rotate. It takes its inputs from the Arduino. For each attached motor there are three inputs, two which decide direction of rotation and one which decides speed of rotation. The input that decides speed is a PWM while the the other two are DC voltages. The motor shield receives these inputs and produces a particular voltage drop at its two output terminals in which the motor is connected to. The voltage difference across the two terminals causes the motor to rotate at a particular speed and direction. In addition to the inputs received by the Arduino, the motor shield is also connected to the Lipo battery which allows the motors to rotate at a higher voltage than that of the Arduino (12V compared to 5V), allowing rotations to occur much faster.

Circuit Schematics

There was only one circuit that needed to be created for the project. This circuit set a particular voltage drop across the DC motor for the shooting mechanism. This voltage drop depended on whether or not the DC motor was to run or sit idle. Essentially, this is an active low circuit, meaning when the motor is to run, the Arduino outputs a low signal from the I/O pin connected to the circuit. The purpose of the circuit is to be able to use the Lipo battery's additional voltage to run the motor at a faster pace. For instance, if the motor was not connected to the Lipo battery, it could only run at a maximum speed set by a 5V voltage drop, which was too slow. The use of the Lipo battery allowed for a maximum voltage drop of about 11.1V.

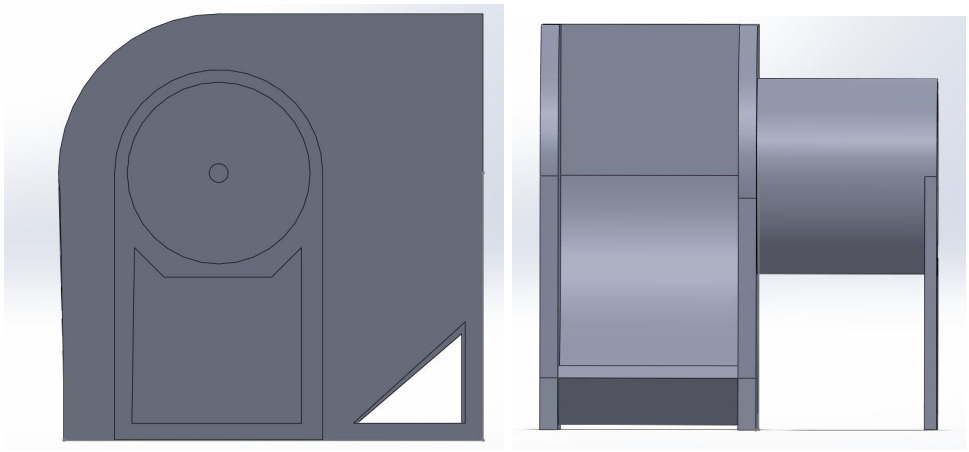


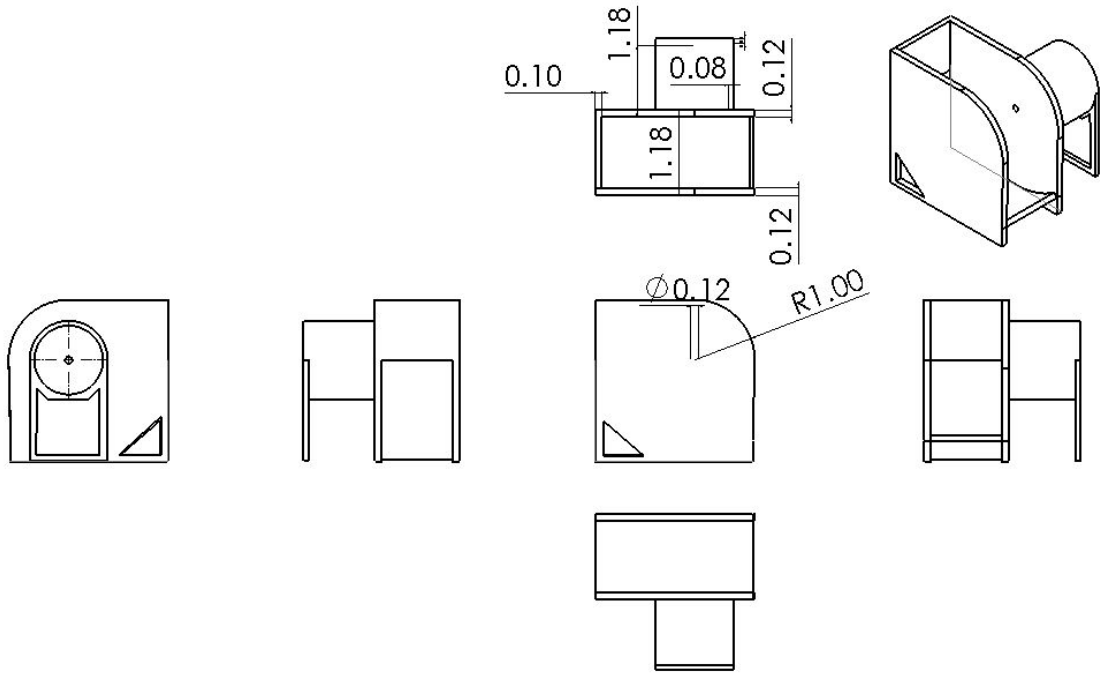
Mechanical

*****All measurements are in inches.

The Shooting Mechanism

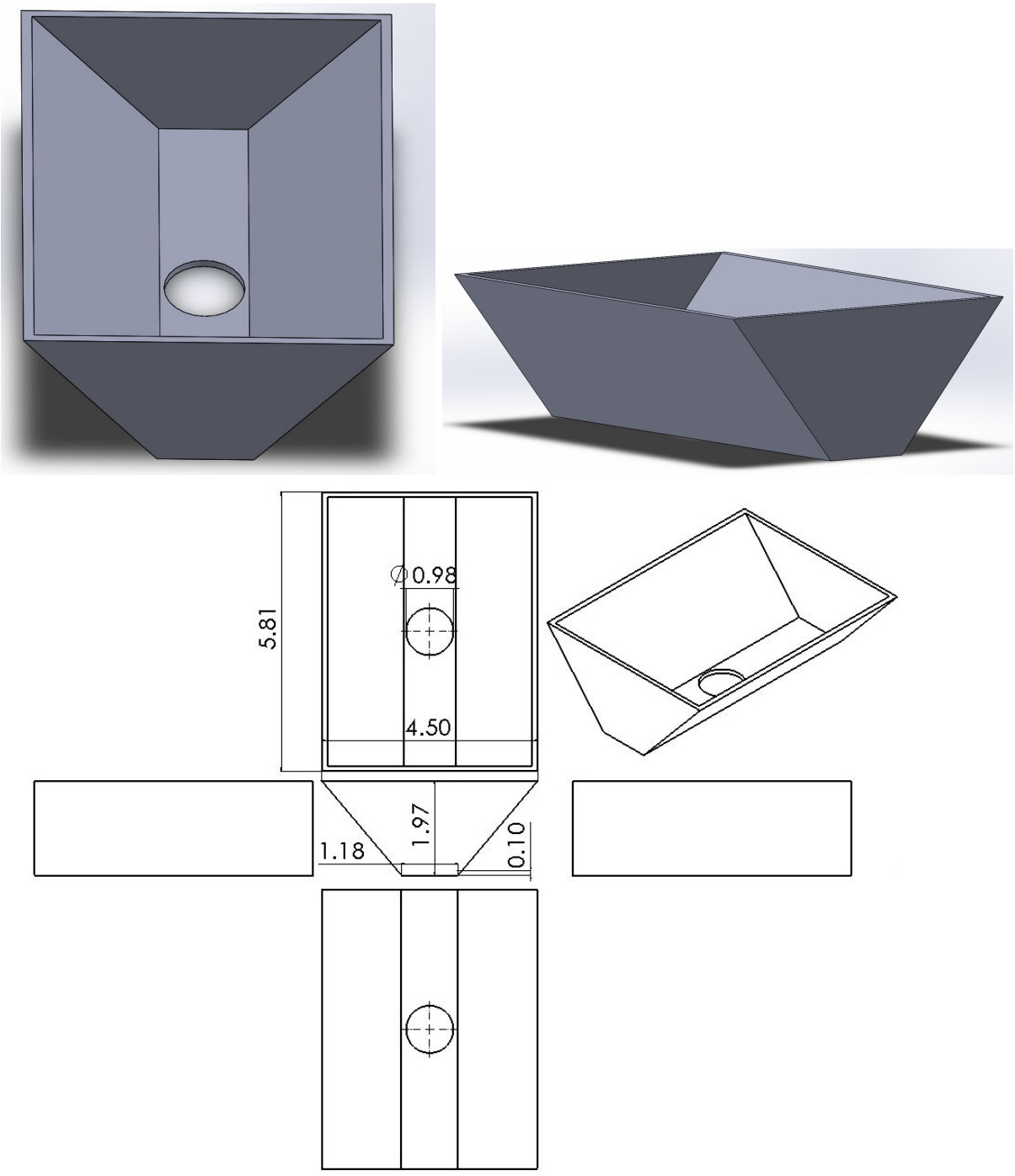
Step one of the project was designing and prototyping something can launch the Nerf balls. After a few iterations we designed a sloped wheelhouse driven by a small dc motor that would spin the wheel inside, propelling the ball as it fell into the well.





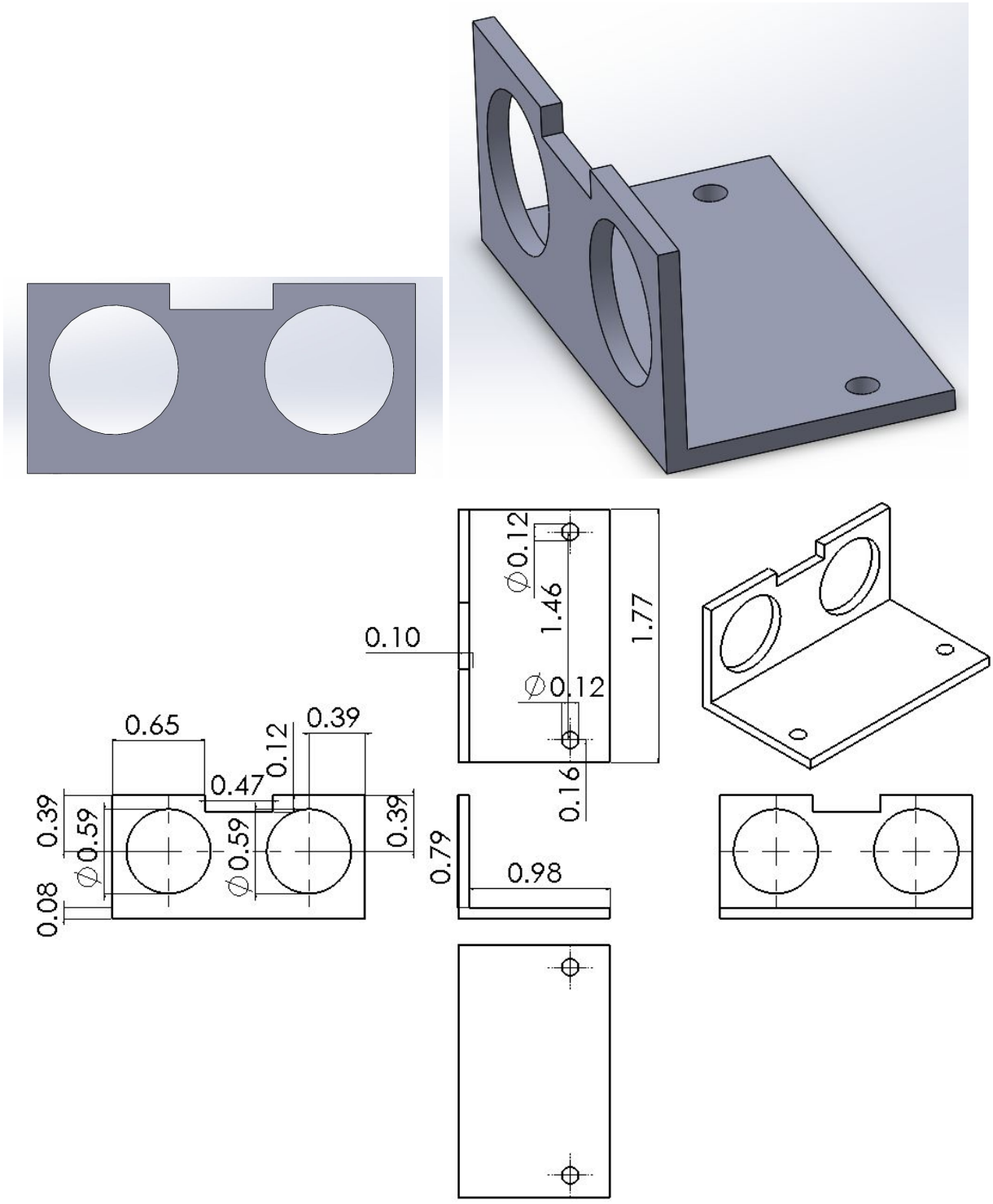
The Funnel

The funnel was designed to be the highest point on the robot, requiring clearance of the underside of the supply drops. Later on large screws were added to each side of the funnel in order to "catch" the release mechanism as the robot drove under the supply drop, as the funnel was directly underneath it.



Ultrasonic Sensor Housing

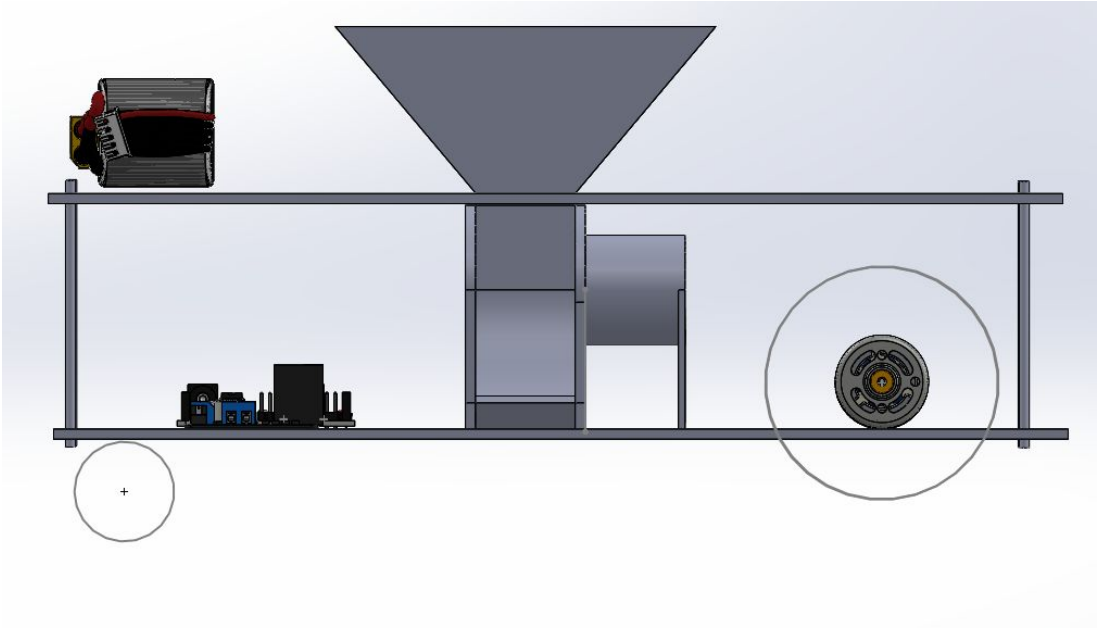
The final 3D-printed part was housing for the two ultrasonic distance sensors on each side of the bot. This was to ensure accuracy and consistency between trials.

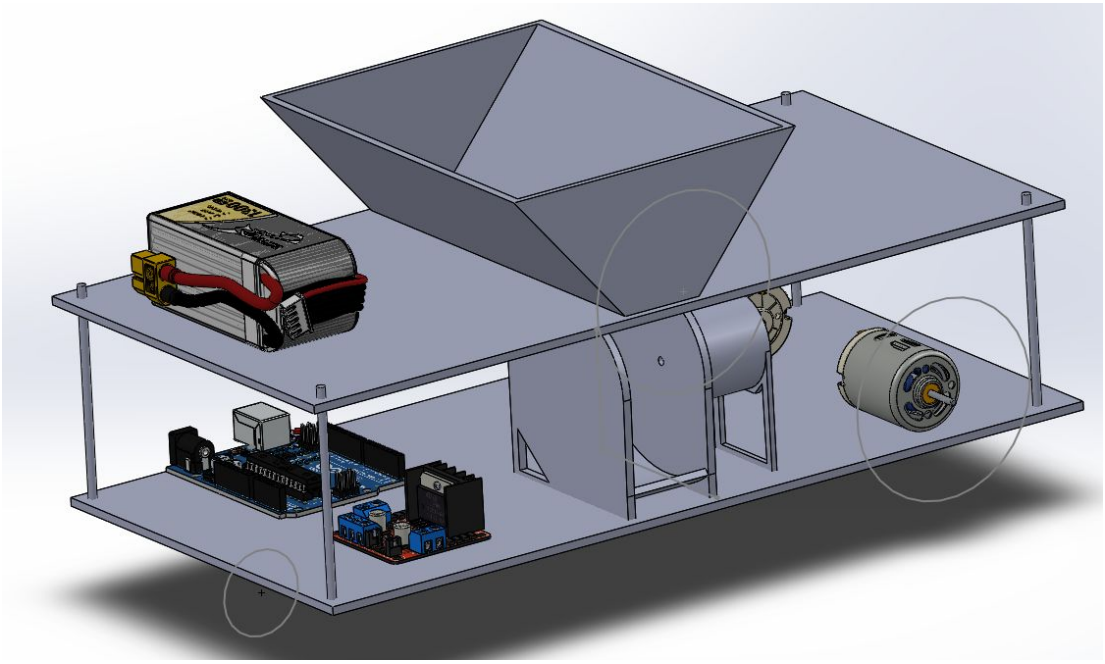


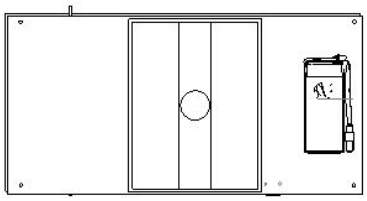
The Final Design

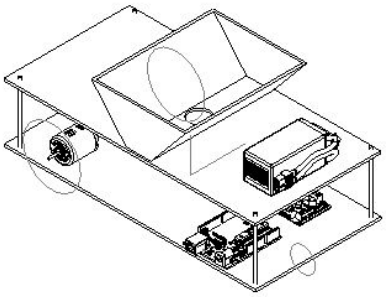
All parts were ensured to fit together by creating an assembly in SolidWorks. The final robot used in the competition had small alterations from the final design, such as screws added on each side of the funnel with tape over the grooves, and a different LiPo battery than originally

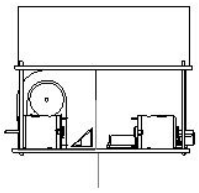
planned. However, the core design stayed the same and different modifications if this project is to be redone might produce better results.

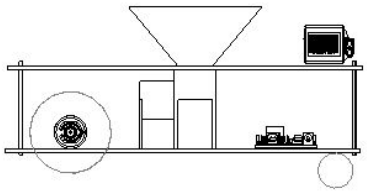


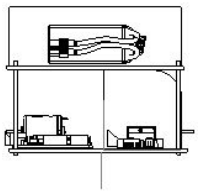


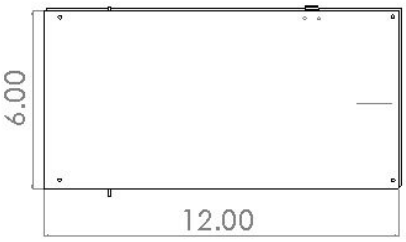











6.00
12.00

Budget and Bill of Materials

Budget

There was no strict budget for this project, though keeping the spending below 200 dollars was the goal.

Bill of Materials

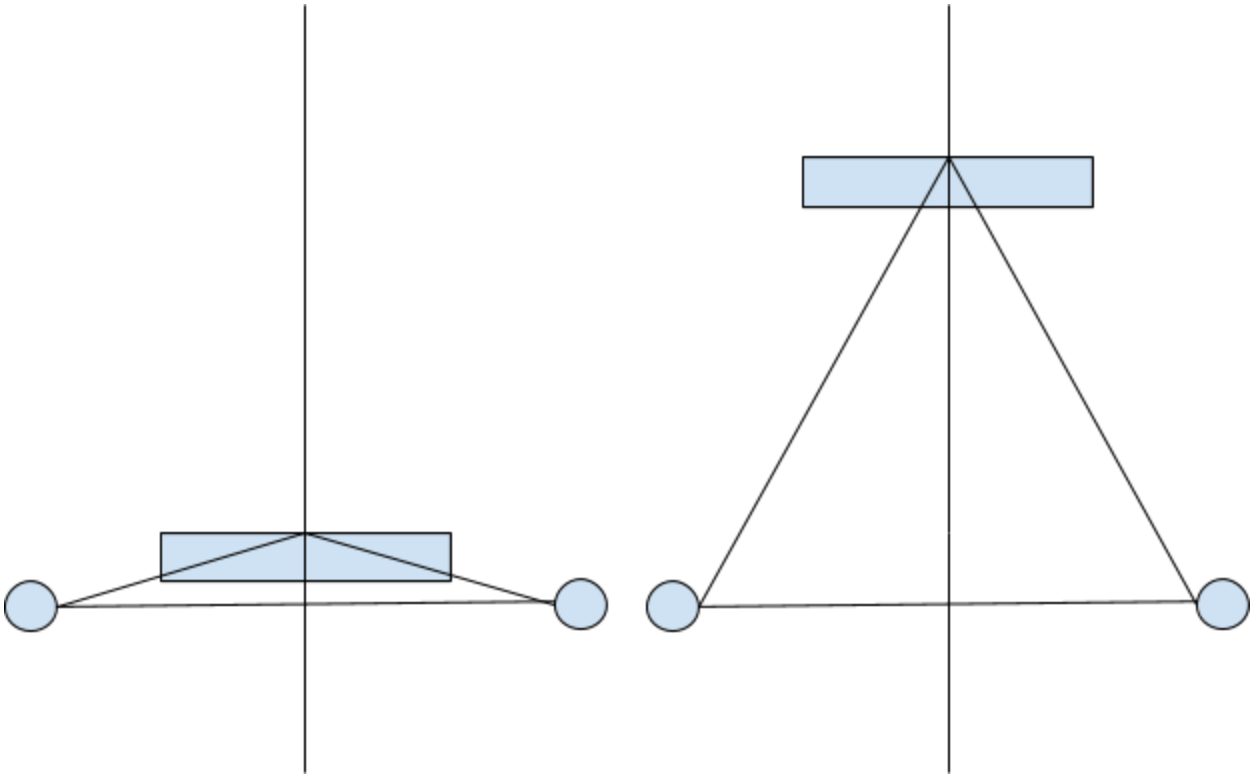
Item	Quantity	Cost per unit	Total cost
Shooting mechanism motor	1	$7.22	$7.22
Elegoo Mega	1	$15.00	$13.99
Expanded PVC	1	$12.70	$12.70
4-½” screw set	1	$9.83	$9.83
3S lipo battery	1	$26.99	$26.99
Solder wire roll	1	$7.59	$7.59
Silicone wire 10’	1	$5.48	$5.48
Sparkfun Ardumoto Shield Kit	1	$24.95	$24.95
Ultrasonic sensor	1	$3.95	$3.95
Miscellaneous nuts, bolts, and washers	-	-	$22.16
50:1 motors	2	$12.00	$24.00
QTR-RA reflectance sensor	1	$9.95	$9.95
Power HD Micro Servo	1	$6.95	$6.95
Solarbotics wheel	2	$3.50	$7.00
Nylon spacer pack 25 mm	2	$1.29	$2.58

Nylon spacer pack 10 mm	1	$1.49	$1.49
Nylon spacer pack 8 mm	1	$1.79	$1.79
Ball caster	1	$1.99	$1.99
Pololu wheel pair	1	$9.95	$9.95
L bracket motor mount pair	1	$7.95	$7.95
Wheel mounting hub for 6 mm shaft 2 pack	1	$7.95	$7.95
QTR-RC reflectance sensor	2	$9.95	$19.90
Rocker switch	2	$0.75	$1.50
Sales tax (all orders)	-	-	$5.54
Shipping and handling (all orders)	-	-	$23.51
3D printed launcher	1	$15.00	$15.00
Motor shaft adapter	1	$8.00	$8.00
3D printed funnel	1	$25.00	$25.00
-	-	-	-
TOTAL			$314.91

Lessons Learned

Concepts Learned

- The reflectance sensors for the robot needs to be placed as far away from the robot's wheel axle as possible. Placing the sensors close to the axle of the wheels will cause the robot to not follow the line well. The reason for this is due to the robot overcompensating its turns when the reflectance sensors are too close to the axel. Looking at the diagram below, in the figure on the left, a small right turn will move the line to the far left side of the line sensor. The robot will adjust to get back into the center and make a larger right turn, displacing the line sensor even more. In the figure on the right, a small right turn will still keep the line towards the middle of the line sensor, thus it does not need to make a hard turn to keep the line centered.



- While the design of the robot should be careful and patient, adequate time must be allocated for prototyping and building the robot. While many ideas looked good in simulation or sounded good in concept, during construction of the robot, many different things can go wrong, such as damaging a part or making a slight error in measurements. In the building of the project, the ramp in the center of the course could have been detrimental. It had not been planned for , which resulting in having to raise the line sensors higher than they should have been (about 0.5" off the ground, where specs said the maximum height should be 0.375"). Had building occurred earlier, it would have

been clear the design would not make it up the ramp and additional parts could be ordered to assist with this. Unfortunately, this was not possible as this realization occurred with just a week until the competition.

- Duplicate parts should be purchased with every online order assuming the cost of the part is under that of the shipping cost. Many times in this project, a part would get blown out and it was a scramble to order a new one and hope in came in before the competition. Additionally, the cost of shipping multiple orders added up quickly and was a significant cost to the project. Having duplicate parts ensures there are extra parts waiting in case on stops working. This saves significant time and can potentially save money.
- Alternative prototyping methods should be explored, such as creating parts from cardboard or other cheap and available material. Much of the prototyping in this project was done using 3D printing, which can be expensive and time consuming for both the design and printing stages. We were fortunate enough to be able to reuse a modified version of some of our 3D printed parts for something other than the original intended use, which saved money and time in the end.

Future Improvements

- If the Arduino is to be used solely for the project and no additional hardware is to be added, the leads of the wires should be soldered to all hardware components. There were many times a component in the robot would stop working and after careful troubleshooting, it turned out to be a loose wire. Soldering the wires to the leads would ensure the connections stay intact over the course of testing and the competition.
- The voltage from the adjustable voltage regulator should be changed from 6V to 6.1V. With the angle the shooting mechanism was set to, the Nerf balls were shot slightly underneath the collection nets. This meant many had to bounce into the nets if points were to be scored and a lot simply just never made it into the net. Increasing the voltage at the adjustable voltage regulator would increase the power going into the DC motor which would result in the balls getting shot further and higher. The shooting mechanism was tested at 6.2V but this was too much as the balls were shot above the nets. Before the competition, the output voltage at the regulator was set to 6.1V but the voltage regulator could not handle it because too much current was needed to set the output at that particular voltage. Another method of getting the voltage down to 6.1V should have been used such as a voltage divider at the output of the voltage regulator.
- In addition, if the ramp underneath the shooting mechanism was lowered, the motor could be given a high voltage and not have the balls shoot above the collection nets. This would have been a better alternative to adjusting the voltage at the regulator since the motor could be set at a high voltage, such as 7V, ensuring balls had enough speed to get

into the collection nets. Unfortunately, the ramp was secured to the chassis of the robot in a way that would have damaged the entire structure of the robot if it were to be moved.

- A redesign of the funnel, as many times multiple Nerf balls would attempt to enter the drop through-hole at once, blocking the passage. This issue was semi-deterred by placing a micro servo with an arm that swept across the top of the hole, but a better solution would be a more cylindrical shaped funnel, with an attached motor that would vibrate and shake the entire funnel to ensure the Nerf balls did not become static.
- Since the shooting mechanism was placed only on one side of the robot, it was required to turn the robot completely around depending on which side of the course it was loading from, in order to face the nets properly. This could be remedied by placing a second shooting mechanism with a second hole in the funnel on the other side of the robot.
- Another likely improvement would be to use omni-directional wheels. This would allow for ideal placement of the shooting mechanism, and drop the requirement of very precise turns in order for the funnel to line up properly to the supply drops.

Conclusion

The project proved to be successful as the robot was able to consistently collect Nerf balls from the supply tubes and shoot them into the scoring nets. The robot competed well against other groups, resulting in a 5th place finish. However, just a few minor modification to the robot, as described above, could have easily resulted in a 3rd place finish. In other words, the robot came close to winning the $400 given to the third place team. Overall, the robot stood out from the competition by its ability to navigate the course without any errors. In fact, the robot only had to be reset on the course once after it got caught underneath a supply tube. In the end, the shooting mechanism and the speed of the robot prevented the robot from competing with the top-tier ones. The knowledge gained from building this robot would surely help with creating a more successful robot in the future.