FAMU-FSU College of Engineering

Spring Final Report

ECE Team #1
IEEE SoutheastCon Autonomous Robot

Members:
Michael Pelletier (map13m)
Colin Fortner (crf13)
Hunter Fitch (hmf14)
Nicole Perry (np13b)

Faculty Advisor
Dr. Bruce Harvey

Faculty Reviewers
Dr. Rodney Roberts
Dr. Ming Yu

Instructor
Dr. Jerris Hooker

TABLE OF CONTENTS

I. Introduction

A. Executive Summary

The F2-P2 robot is designed to run on a 45” by 93” arena with four Star Wars themed stages and is based on the 2017 IEEE SoutheastCon Hardware Challenge rules [1]. After starting the robot, it can autonomously navigate to each stage and complete each of the stages’ unique tasks described below.

Stage One will have the robot “uncovering the unknown” by testing five various circuits in unknown configuration to determine what the hidden component is. This is accomplished using a six pronged plate arranged like the stage design. Each prong is attached to a signal with the middle prong connected to ground. Each component, a resistor, a wire, a diode, a capacitor, and an inductor, will be assigned a numeric value between and one and five. The robot will then store this information into memory for later use, as well as display the gathered code on an LCD screen.

Stage Two will require the robot to compete in a “lightsaber duel”. After detecting the presence of a time variant electromagnetic field, the robot will initiate repeated contact with the arena’s “Lightsaber,” only when the field is being momentarily produced. The field is detected using a Hall Effect sensor mounted at one side of the robot. The robot’s “lightsaber” is 3D printed and controlled with a servo.

Stage Three will call upon the values gathered in stage one to “bring down the shields”. The robot will turn a rotary encoder a full rotation for each value collected from the circuit in stage one. To change to the next value, the encoder must be turned in the opposite direction. There is an allowable ±15° between each turn. To rotate the encoder, the robot uses two servo motors connected to a pulley. The pulley rests under the encoder. As the stepper motors turn, the encoder spins as well.

Stage Four tasks the robot to “fire the proton torpedo”. The robot will be loaded with 3 Nerf N-Strike© Darts and has to launch the darts through a square target. The target will be positioned 7.5” from the base of the arena and will consist of a 6”x6” square. An automated dart gun with a relay is mounted on the robot. The gun has the range to shoot from the farthest distance the arena will allow.

The full robot measures 12” by 11” by 10”. The chassis has two levels of an eighth inch aluminum with sixteenth inch vex robotics sides for the bottom level. The robot has plastic tank treads and are controlled with two motor controllers and two DC motors. For navigation, short and long range IR sensors are incorporated into the design. There are one short and one long range sensor on each of the front and back plates. On the left plate, there are two long range sensors spaced on the plate. All components are controlled using an Arduino Mega. The robot is powered using a 7.2 V, NiMH, 3000 mAh Vex Robotics battery and two 1.5 V batteries.

The robot was tested on an arena designed for each stage. The success of the robot is based on its ability to autonomously navigate this arena and complete the stages. While the robot was able to complete the designated stages individually, the robot could not accurately approach each stage

B. Problem Statement

The overall objective of this project was to develop an autonomous robot that can complete the tasks described in the 2017 SoutheastCon Robotics competition [1]. These task are described further in the report.

C. Operating Environment

The robot is designed to run on a specific arena outlined in the challenge rules [1]. This arena needs to be fully built in order to use the robot. The robot will be tested in a 45" wide by 93" long playing field. There are four levels. Each stage is located on one of the walls. Stages 1, 2, and 3 will on the lowest and largest level. On this level is the 15" by 15" starting square. See the rules for complete details on how to build the arena and each stage [1]. Figure 1.1 shows a fully built arena.

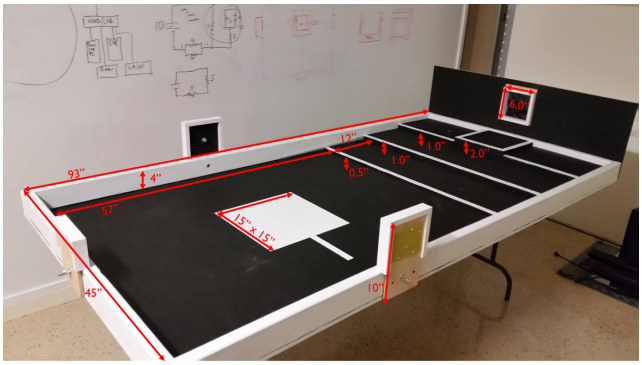


Figure 1.1: Arena

D. Intended Users and Users

The robot can be used by anyone. However, it is intended to be used at the SoutheastCon competition or in the arena described in the operating environment. The robot can be programmed for different uses, but it's components are designed to complete specific tasks.

E. Assumptions and Limitations

It was assumed that arena at the competition and the one build for testing were of equal size and quality. Further, the testing arena did not have full scoring capabilities. Thus, the sensitivity of the vibration sensor for stage 2 and the degree of error allowed for stage 3 were estimated. During the competition, the teams had limited time in the competition arenas for testing and error correcting to the arena. Outside of the competition, the robot has limited function without the stages.

F. Needs Statement

The needs of this project are to design and construct a robot capable of the following. The robot must be small and low cost, capable autonomous operation to navigate and move around the arena, as well as test a five-point circuit, detect using sensors for an electromagnetic field, rotate a combinational dial, and fire projectiles.

G. Constraints

The constraints on the robot include its cost and size. The estimated budget for this project is $750. Thus, the complete construction of the robot must be at or under this budge. The robot must fit entirely in a 12" by 12" by 12" cube at the start of the match. However, after the run has started, the robot can expand in size or break into multiple independent robots if the robot does not exceed the playing surface [1].

H. Acknowledgements

The Southeast Con group would like to extend a special thank you to Dr. J. Hooker, Dr. B. Harvey, Dr. R Roberts, Dr. M. Yu, the FAMU-FSU COE Machine Shop, and the Region 3 Institute of Electrical and Electronic Engineers for their roles in the designing of this robot.

II. System Design

A. Stage 1

Stage 1 requires the robot to detect multiple different circuit components. The components being a wire, a 10K Ohm resistor, a .1uF capacitor, a 100mH inductor, and a diode in either forward or reverse bias. The stage consists of six holes where there are five holes on the outside and the sixth in the center. Each of the outside holes has a random component and they all connect to the center ground. The circuit was detected by sending a voltage to each of the outer holes and measured the voltage coming out of the center pad to the common ground (Figure 2.1).

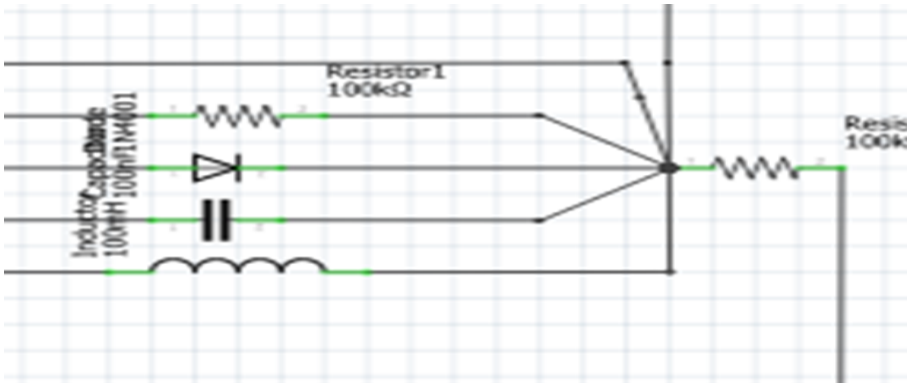


Figure 2.1: Circuit layout for Stage 1

The design choice for the attachment was whether to use a single prong to connect the robot to the stage component and rotate through the other terminals, or to have all five prongs attach to each of the terminals and read them individually. The idea of using one prong was not used due to it adding unnecessary difficulty with rotating to find the next prong. Because the approach on stage one is largely dependent on decoding the capacitor and inductor, these components were the focus for designing the stage. Initially, the plan was to use a pulse width modulation pin to simulate a sine wave. From there, we would be able to use an oscilloscope reading to determine if the output voltage was leading or lagging. This could determine if it was a capacitor or inductor respectively. This approach was scrapped when it was determined that using sensors determining the power factor would be difficult to do autonomously. The next approach that was ultimately chosen as the solution our team went with was a simple circuit that took voltage readings at certain time intervals to determine the component (Figure 2.2).

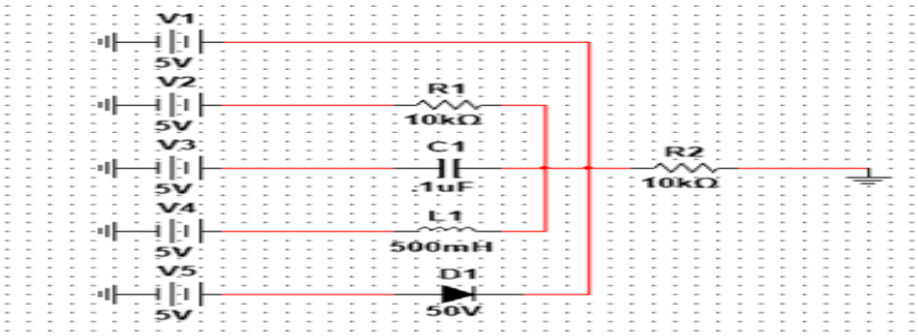


Figure 2.2: Stage 1 layout

Each of the five components in parallel are the ones hidden behind the stage. The voltage reading is taken over the highlighted resistor initially when the pin was turned on, and then again one second later. For the wire, resistor, and diode, voltage will remain constant. For the capacitor, as the capacitor is charging, the voltage across the resistor will drop, and vice versa for the inductor. These differences are recorded and can be used to determine the components.

Voltage calculations are taken across each path seen in *Figure 11* and are as followed:

Wire: Total voltage from pin to ground = 5V

Resistor: $V_R = V_{in} * \frac{R_2}{R_1+R_2} = 5 * \frac{10K}{10K+10K}$ = 2.5V

Diode (forward bias): $V = V_{in} - V_F = 5 - 0.6$ = 4.6V

Diode (reverse bias): Will act as an open circuit = 0V

Capacitor: $V_R = V_{in} - V_{in}(1 - e^{\frac{-t}{RC}})$

When t=0: $V_R = V_{in}$ = 5V

When t= 1: $5 - 5(1 - e^{\frac{-1}{10K*.1\mu}}) \approx 0$V

Inductor: $V_R = V_{in}(1 - e^{\frac{-Rt}{L}})$

When t=0: $V_R = 0V$

When t= 1: $5\left(1 - e^{\frac{-1*10K}{-500m}}\right) \approx 5$V

The values for the resistor voltage are going to be the basis for determining the hidden component. The approach for coding this section would be to first determine if the voltage in the beginning matches the voltage after one second. If the initial voltage is significantly higher than the voltage at one second, the component must be the capacitor. If the voltage after one second is significantly greater than the voltage at the beginning, the component must be an inductor. If the two voltages are the same, then the voltage will be checked to determine the component. If the voltage reading is 5 volts, the component must be the wire. If the voltage is 2.5 volts, the component must be the resistor. Finally, if none of those are true, then the component must be the diode. Checking the diode at the end allows us not to worry if it is in forward or reverse bias.

B. Stage 2

Stage 2 requires the robot to detect an electromagnetic field produced by a coil constructed from 40 turns of #20 stranded copper wire wound around a 0.5" diameter bobbin. A current of 1 A will be fed through the magnet to generate a positive field (as measured from the front side of the wall on the robot arena side) (IEEE Southeast Division, 2016). The lightsaber will be constructed to contain 8 LED indication lights and a vibration sensor to accurately detent contact, made from the robot to the arena mounted components. These components are as listed in the Table 2.1.

Table 2.1: Lightsaber Technical Components

Vibration Sensor	Tech Specs ● Maximum Operating Temperature: 260°C ± 10°C ● Contact Time: 2 - 2.5ms Dimensions: ● Diameter: 5mm / 0.2" ● Height (w/ pins): 23mm / 0.9" ● Height (w/o pins): 11mm / 0.4" ● Weight: 0.2g
LED Indication Lights	Dimensions: 51.10mm/2"x10.22mm/0.4"xC.19mm/0.12" Weight: 2.57g

This stage has been approached from several different angles until ultimately deciding on the use of a Hall Effect sensor. The original design used a Reed Switch. A reed switch is a mechanical switch typically comprised of two metal contacts inside a glass or plastic housing, in the presence of an electromagnetic field, the contacts touch thus closing the switch. The reed switch will be implemented into a parallel circuit with the microcontroller. The reed switch was not sensitive enough to determine a change in the field.

To meet the needs of the electromagnetic fields, the robot was equipped with an electromagnetic field sensor. The Hall Effect Sensor used in combination with the Arduino microcontroller to determine if the magnetic field is active. Once the field has been determined to be active, the Arduino microcontroller sends the signal to activate the Servo used to make contact between the robot and arena lightsaber. The block diagram below shows the process flow utilized for Stage 2.

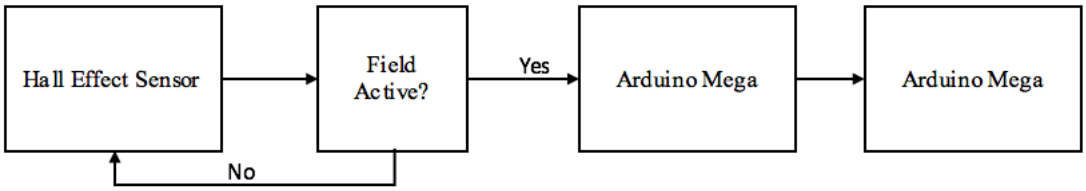


Figure 2.3: Block Diagram of Stage 2

C. Stage 3

Stage 3 implements the code found in stage 1 on a quadrature encoder. The encoder is able to rotate freely and records the direction and number of turns. The robot needs to turn this knob a full 360 degrees to represent a value of one. The knob is continually turned in the same direction until the turns equal the value of the digit. To change digits, the knob is turned in the opposite direction. There is +/- 15 degrees for each turn. A total of five figures needs to be implemented. The approach is to use a motor that has some type of gripper attached to it. The motor needs to rotate 360 degrees and should be fast and precise.

The initial design utilized a continuous rotation servo. Servos are fast and can rotate smoothly. Without the position sensor of a regular servo, the continuous rotation servo was controlled through time delays. However, this resulted in too much precision lost and greater then +/- 15 degrees for each turn.

To get more precision, a stepper motor was used. To control the stepper motor, an h-bridge was added as well. The stepper motor was more difficult to implement then the servo due to the use of the h-bride. It also weighed significantly more than the servo. However, the stepper motor's precision was the deciding factor in choosing it. Two stepper motors were used to turn a conveyor. The conveyor was made from a rubber band that rested underneath the encoder when approached. To get the band underneath the encoder, a servo held the band down until stage 3 was approached. Figure 2.4, shows a system design of stage 3.

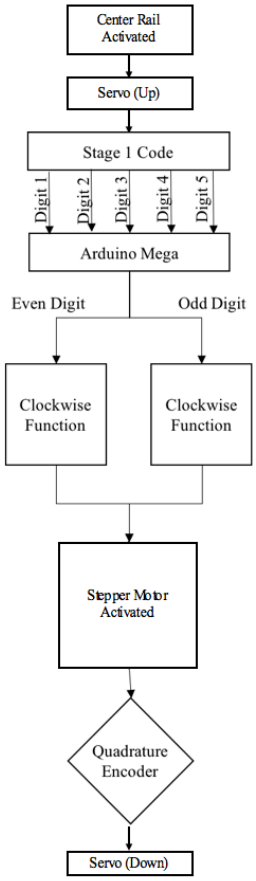


Figure 2.4: Stage 3 System Design

D. Stage 4

The solution for stage four has may different possibilities, these possibilities include a purpose built spring powered nerf gun, an air powered nerf gun, or repurposing an actual nerf gun. The first solution explored was building an air powered nerf gun that would mount to the top of the robot and be fired when the round was over. After having looked at the air powered nerf gun a little more, other options were explored more thoroughly, mainly the repurposing of an actual nerf gun. With the idea of repurposing a nerf gun while looking at old robots in the lab, an old battery powered nerf gun was found. After some careful measurements it was determined that after the nerf gun was deconstructed it would fit in the place that the air powered gun would go. This decision to go with the deconstructed nerf gun over the hand built solutions was made not only for reliability, but for the ease of implementation. The deconstructed nerf gun was battery powered which made the only thing needed to fire a simple relay to allow the current to flow or not, thereby turning the nerf gun on or off. Figure 2.5 shows the system design for stage 4.

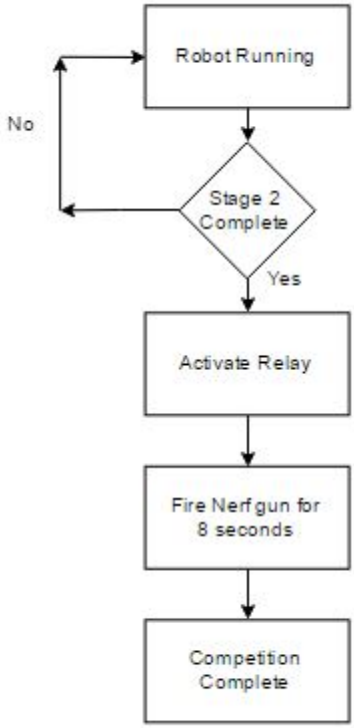


Figure 2.5: Stage 4 System Design

E. Navigation

The sensors to be used in the design have been narrowed down to two cost-effective options. The first option is an ultrasonic sensor, these sensors are extremely cheap, and has accuracy to match the cost. The second option is to use infrared sensors that are about twice as much as ultrasonic sensors, and have more accuracy as to distance and location. The ultrasonic sensors would be a good option, but with the competition environment, it is unknown how much outside noise from the crowd will be there, and if the short wall of the arena will be a large enough reflector for the sound pulses. Infrared is a proven cost-effective sensor that has been used in many applications effectively, mainly with previous competition senior design groups. That led to the selection of infrared sensors for use on the robot because of the track record previous groups had with them, and the ease of installation on the robot, and their implementation on the microcontroller.

F. Total Integration

The design for our project is set to have the front and back of the robot to have two extensions each. Each extension will correspond to an individual stage of the competition. For stage one and three, the heights are at a fixed position that cannot be altered, due to this fact, it makes sense that these sensors be on opposite sides of each other. The light saber for stage two and the projectile launcher in stage four do not have designated locations so they can be mounted above the extensions for the previous two stages. The sides of the robot would be difficult to use with extensions due to the fact that we are using tracks and it would be unnecessary to approach an obstacle sideways as opposed to head on. For that reason, the sides of the robot will be used to monitor the display screens, distance sensors as well as the drive train of the robot.

Other designs that were considered for the build of the robot include a rotating platform that could change 90 degrees in order to have one extension on each side. However due to the fact that we have plenty of room vertically, this added complication was not necessary. Wheels were another option for our motion. It would have been just as easy to implement and design, with faster movement. However, the decision to use tracks was made for the purpose of stage four. The initial design was for the robot to climb the stairs to make firing into stage 4 easier. However, after learning the range of the Nerf gun, the stairs were not climbed.

The system is shown in Figure 2.6. Subsystems are described below in *Solutions and Designs*. The input will just be an "On and Off" switch only due to the autonomous nature of the robot. The output will be mechanical movement.

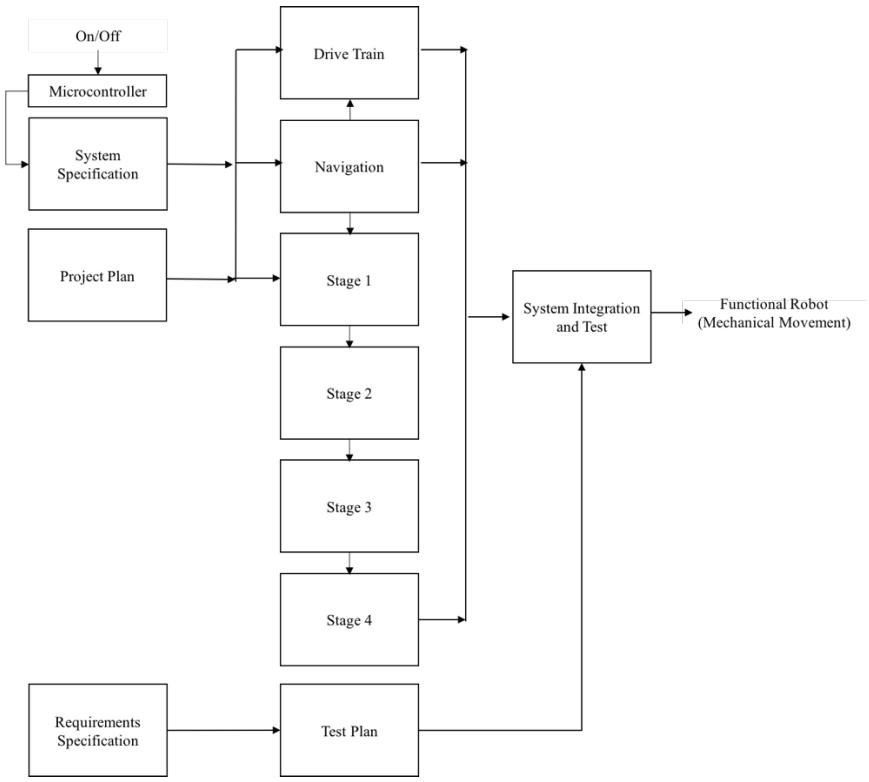


Figure 2.6: System Design

III. Components

A. Chassis

The robot chassis consists of many components. The base of the chassis is a 7 ¾" x 10" plate made of 1/8" aluminum with 1" fold on all four sides to create a bottomless box. Attached to this base is another aluminum plate, again 7 ¾" x 10", but with no sides. Between the base and top plate are four 2" standoffs, each held to the base plate with a ½" 6-32 bolt, and a ¾" 6/32 bolt to attach the top plate to the standoffs. Between the standoff and the top plate there is a 6/32 nut to allow the top plate to be raised or lowered ¼" from the top of the standoff.

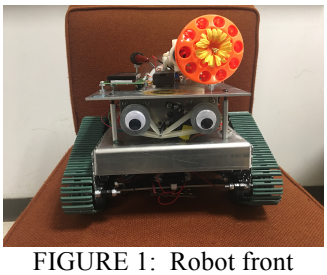


FIGURE 1: Robot front

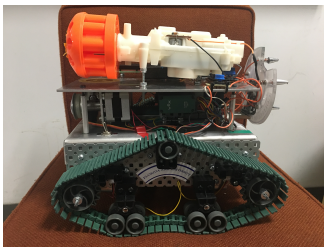


FIGURE 2: Robot right

Attached to the 1" x 10" long side of the base plate is the drive train assembly. To stabilize the drive train assembly, two 1" x 7 ¾" long supports are fastened on the inside of the drive axles. Each support is constructed of 1/8" aluminum with a ½" x 1" fold on both ends to allow them to mount to the drive assembly plate. Mounted along the bottom of the robot base, are the Arduino microcontroller, and battery with Velcro for easy removal. Between the two supports is a 2" x 5" VEX plate. Held in place with rubber bands and Velcro, this plate creates a door to the bottom of the robot to secure wire from dragging, as well as holds the power distribution board. Additional attachments to the base include sensors for navigation and magnetic sensing.

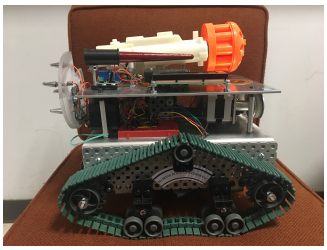


FIGURE 3: Robot left

Attached to the top of the base plate is the components of stage 2 and stage 4. Between the Top and Base plates is the linear rail system for stages one and three. Parallel to the long side of the base is three plastic VEX Rail mounts equidistant from both sides and from each other. Resting on the rail mounts are two pieces of VEX linear railing. The railing for stage one is 3" long and the railing for stage three is 7" long. The longer rail has 3 pieces of VEX toothed track attached. Equidistant from all four edges is the linear rail motor with square axle and VEX Medium Gear mounted inside a VEX C-Bracket. The Linear rail is moved by a VEX 2 Wire Motor with the VEX 2-3 wire motor controller.

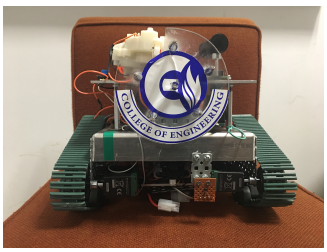


FIGURE 4: Robot back

B. Microcontroller – Ardunio Mega & Pushbutton

The microcontroller chosen is an Arduino Mega. The Arduino is powered from the battery using the VIN Pin. One of the ground pins goes directly to the power distribution line. The code is started using a generic blue pushbutton mounted on top of the robot. The digital pins used are 22, 31, 45, 47, 49, and 51-53. The analog pins used are A0, A4-A9, and A14. The PWM pins used are 3 and 5-11.

See Appendix I for pin placement for each stage.

The COMM pins used are SCL and SDA.

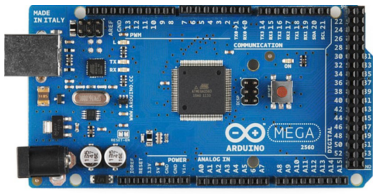


FIGURE 5: Arduino Mega

FIGURE 6: Pushbutton

C. Battery – Vex Robotics 7.2V Robot Battery NiMH 3000mAh

This battery can supply 7.2 volts and lasts for 3000 milliamp hours. It is rechargeable and includes connectors for charging. For optimal performance of the robot, the battery needs to be changed or recharged after two hours of use. A toggle switch is attached to the battery's ground to turn on and off the power. The switch is located on the left side of the robot next to stage 3. The red wire of the battery connects to the 7.2 power distribution plate underneath the robot. The black wire with the switch attached is connected to the ground line on the power distribution plate.

FIGURE 7: Battery

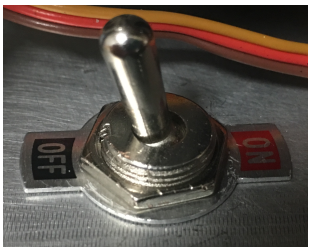


FIGURE 8: Toggle Switch

D. Drivetrain — Vex Robotics Tank Treads Kit

1) Tracks

A Vex Robotics tank tread kit is chosen. The full track measures 11.25 inches. In Figure 5, (a) is the plastic tread links, (b) is the tensioner, (c) is the gear, and (d) is wheels. On the other side of the robot is another set of tank treads with the same set up shown in Figure 5.

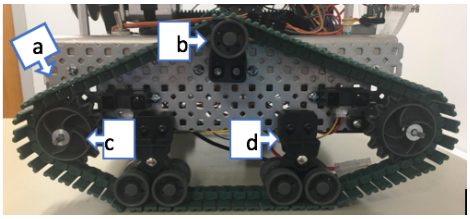


FIGURE 9: Track setup

2) Motors – Vex Robotics DC Motor 393 & Motor Controller 29

Each track is controlled with a motor and motor control. They are located at the back and on the inside of the robot. The motor is connected directly to the motor controller that attaches to the arduino. The motors used are Vex Robotics DC Motor 393, and the controller is a Vex Robotic Motor Controller 29. The red wire of the motor connects to the 7.2 power line, while the black connects to the ground line. The white signal connects to the Arduino's PWM pin.

FIGURE 10: Motor (Left) & Controller (Right)

E. Navigation

1) Short Range – Sharp GP2Y0A41SK0F

This sensor ranges from 4-30 centimeters. There are two mounted on the robot: one on the front and rear. The red wire is connected to 5 volts Vcc, the black wire is connected to ground, and the yellow wire will be connected to an analog pin on the microcontroller to collect the data being output by the IR sensor.

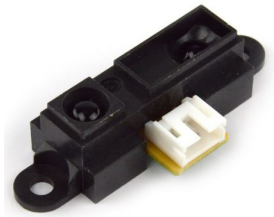


FIGURE 11: Sharp GP2Y0A41SK0F

2) Long Range – Sharp GP2Y0A02YK0F

This sensor ranges from 20-150 centimeters. There are four total in use: one on the front and rear and two on the left face. The long range sensors are connected the same as the short range sensors.

FIGURE 12: Sharp GP2Y0A02YK0F

F. Stage 1

1) Plastic Plate

The design for stage one consisted of a plastic plate with a radius of 2.5". This plate consists of six holes; each hole is a 0.25" diameter hole. The first hole is directly in the center of the plastic plate, and the holes on the perimeter are arranged in a circle with a 1.5" radius around the center (the 1.5" radius circle passes through the center of each of the 0.25" diameter holes). The holes begin at the top 0° position, with the next 4 counting off sequentially in a clockwise manner at 72°, 144°, 216°, and 288°.

2) Springs & Resistor

Attached in each hole is a .25" spring that extended .5" from the base. Each of the outer springs had a wire connected to the tip of the spring that ran through the middle of the spring and attached into the Arduino microcontroller. The spring in the middle of the plate is recognized as the ground pin, and this one connected to a 10K resistor before connecting to ground. The entire plastic plate is connected to the rail system in the middle of the robot that is later.

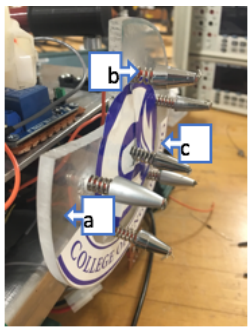


FIGURE 13: Stage 1 Outline
(a) Plastic Plate
(b) Springs
(c) Ground Pin

3) LCD Screen –IC/I2C/TWI Serial 2004/20x4 LCD Module

The LCD screen displays the code found in Stage 1. The red wire connects to the 5V line, the brown wire represents ground, and the orange and yellow wire connect to the COMM pins on the Arduino.

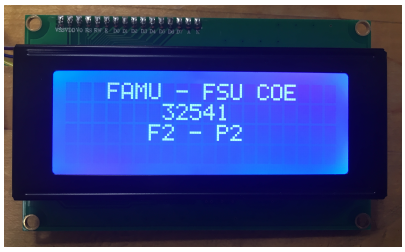


FIGURE 14: LCD Screen

G. Stage 2

1) Allegro A1301KUA-T ratio-metric Hall Effect sensor

Stage 2 consists of 2 major components connected to the main robot Controller Arduino. The first component is the electromagnetic sensor. Using an Allegro A1301KUA-T ratio-metric Hall Effect sensor mounted on 1" x ¾" printed circuit board with 3 connector wires for attachment to the power distribution and microcontroller. Each prong of the chip corresponding a separate wire. This build is attached to the front of the chassis via a 1 ½" x 1" mounting bracket and a ¾" 6-32 bolt and 3 6-32 nuts per bolt. As shown in Figure 14.

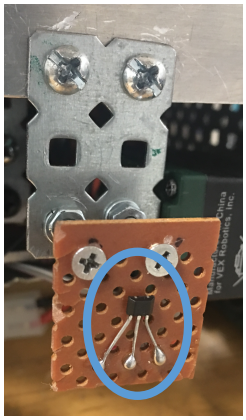


FIGURE 15: Figure 1: Magnetic Field Sensor Build with PCB and spacing components

2) VEX Brand 3-Wire Servo

The second component is a VEX Brand 3-Wire Servo. This device is mounted directly to the top plate of the chassis using the supplied hardware. After the plate is a Vex Brand bearing to allow the axle to freely spin. A 3" square axle is inserted through the bearing into the servo clutch, with an axle collar about 1" from the top. Resting on the axle is the 6" 3D printed "Lightsaber". The setup can be seen in Figure 15. The orange wire on the servo connects to the 5V line, the black wire connects to the ground line, and the white wire goes to the a PWM pin.

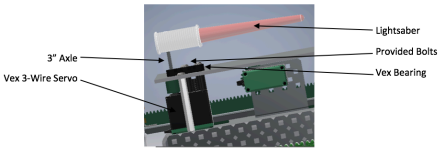


FIGURE 16: Stage 2 Servo Setup

H. Stage 3

1) Stepper Motor—ROB-09238

Two stepper motors are used behind each gear as shown in Figure 19. These stepper motors are bipolar and turn 360° through 2000 steeps. The holding torque is 2.3 kg*cm. The motors should never be powered above 12 V. The two motors are connected via a metal plate. The plate is mounted to the center track.

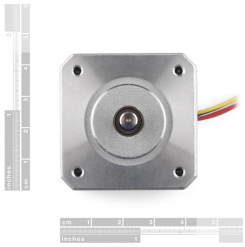


FIGURE 17: Stepper Motor

2) H-Bridge—TEXAS INSTRUMENTS SN754410NE IC, PERIPHERAL DRIVER, HALF-H, 1A, DIP16

Two of these H-bridges are used. Each one connects to the servo. The signal pins of the two bridges are connected together. The H-bridges are placed on a breadboard with two voltage lines: 5 V and 7.2 V. See Appendix II for the stepper motor and h-bridge set-up.

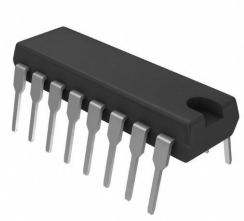


FIGURE 18: H-Bridge

3) Servo Motor—ROB-10333

The servo holds down the band as shown in Figure 19. This model is a micro servo with metal gears. It can be powered between 4.8 and 6 volts. It rotates 180°. The servo is powered with 5 V using the breadboard with the H-Bridges on them.

FIGURE 19: Servo

4) Gears & Band

In Figure 19, (a) shows the rubber band, (b) points to the gear, and (c) displays the servo.

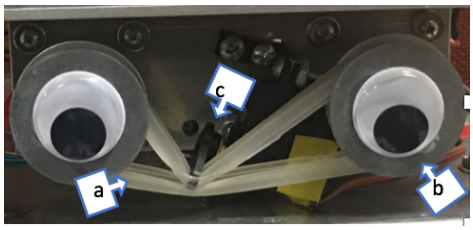


FIGURE 20: Stage 3 Setup

The rubber band is a third of an inch thick. The gears are made from three 1.25” fender washers. On each side of three washers is two 1” fender washers. Enclosing the other washers is a 1.5” fender washer on each side. The gears are connected to the servo through a coupler.

I. Stage 4

1) Battery Powered Nerf Gun

The nerf gun was removed from the housing to get the actual firing component down to the smallest size possible to ensure that it fit on the robot. Mounting the nerf gun on the robot required drilling out the holes where the gun is mounted to its exterior housing to 3/16”. This allowed the nerf gun to be mounted to the robot with 3” machine screws, as shown in Figure 20, The longer screws allow the nerf gun to have the elevation adjusted to make the nerf darts fired out of the gun more accurate.

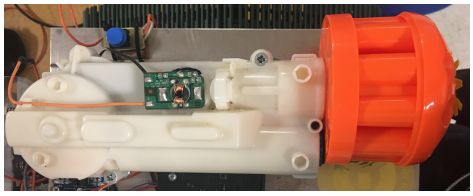


FIGURE 21: Automated Nerf Gun

2) Sunfounder 2 Channel DC 5V Relay Module

This relay allows the nerf gun to be fired when it is needed since the trigger mechanism had been removed. The relay is mounted to the robot with ½” machine screws. The relay is two channel, but for this application only one is needed. To connect the relay, 5 volts from the microcontroller goes to Vcc, ground goes to ground, and In1 goes to a digital pin.

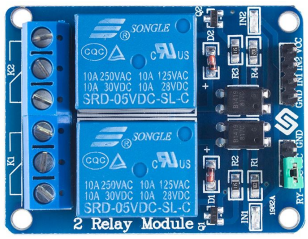


FIGURE 22: 5V Relay

3) 3 Volt Battery Pack

The battery pack is used to power the nerf gun, the black wire on the nerf gun is ground, and the orange wire is the voltage. To make the nerf gun fire, connect the black wire of the battery pack to the black wire of the nerf gun. The red wire from the battery pack connects to the most inward screw of K1, and the orange wire is placed on the screw next to that in K1.

FIGURE 23: Battery Pack

IV. TEST PLAN

Table 4.1: Robot Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.		
2	All wires connected.		
3	The robot is in the white square with front facing stage 3.		
4	Robot is turned ON. LCD is on and not flashing.		
5	Push blue button. Robot is moving toward stage 1.		
6	Center rail pushed out toward stage 1.		
7	Stage 1 makes contact with the stage 1 arena.		
8	LCD displays code.		
9	Robot reverses toward stage 3.		
10	Center rail pushed out toward stage 3.		
11	Servo arm is lifted.		
12	Band makes contact with stage 3 encoder.		
13	Stepper motor turns.		
14	Stepper motor inputs code on LCD.		
15	Servo arm resets.		
16	Robot moves toward stage 2.		
17	Robot strikes arena lightsaber.		
18	Robot strikes when field is active.		
19	Robot strikes at final 28 seconds.		
20	Robot launches 3 nerf darts toward stage 4 into the hole.		
21	Robot is turned OFF.		

Table 4.2: Stage 1 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.		
2	All wires connected.		
3	Upload Stage 1 Test code.		
4	Voltage is coming through pins.		
5	LCD is displaying code.		

Table 4.3: Stage 2 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.		
2	All wires connected.		
3	Upload Stage 2 Test code.		
4	A difference is observed when the magnetic field is ON verses OFF.		
5	The lightsaber hits when the field is ON.		

Table 4.4: Stage 3 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.		
2	All wires connected.		
3	Upload Stage 3 Test code.		
4	Servo arm comes up.		
5	Servo turns counterclockwise.		
6	Servo turns clockwise.		
7	Servo arm resets.		

Table 4.5: Stage 4 Test Plan

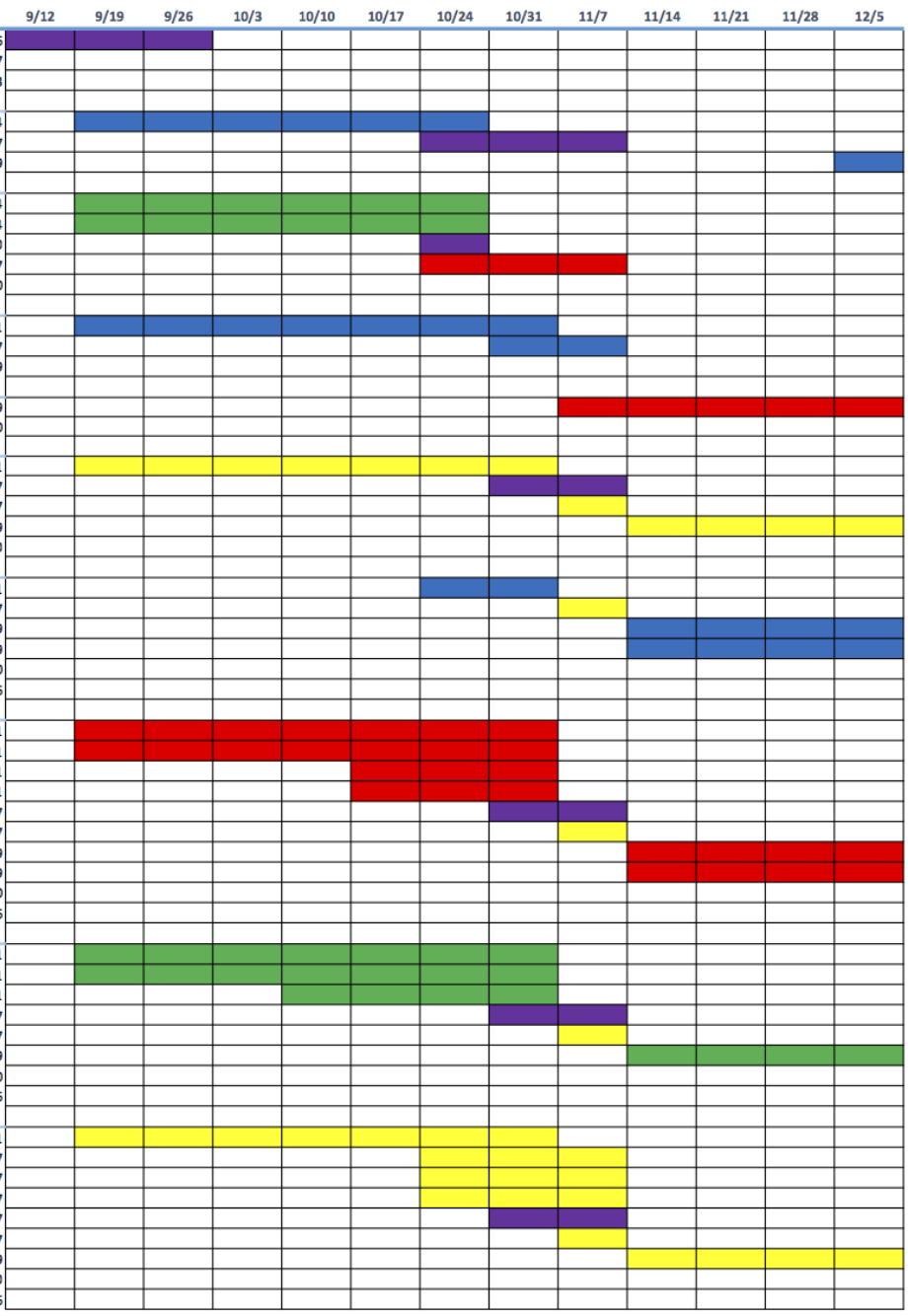
Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.		
2	All wires connected.		
3	Battery pack is ON.		
4	3V is going to the gun.		
5	Shots are delayed.		
6	Gun is positioned correctly at target.		
7	3 darts fire through the hole at stage 4.		

V. SCHEDULE

A. Fall 2016 Schedule

For the Fall 2016 semester, the robot was built on schedule.

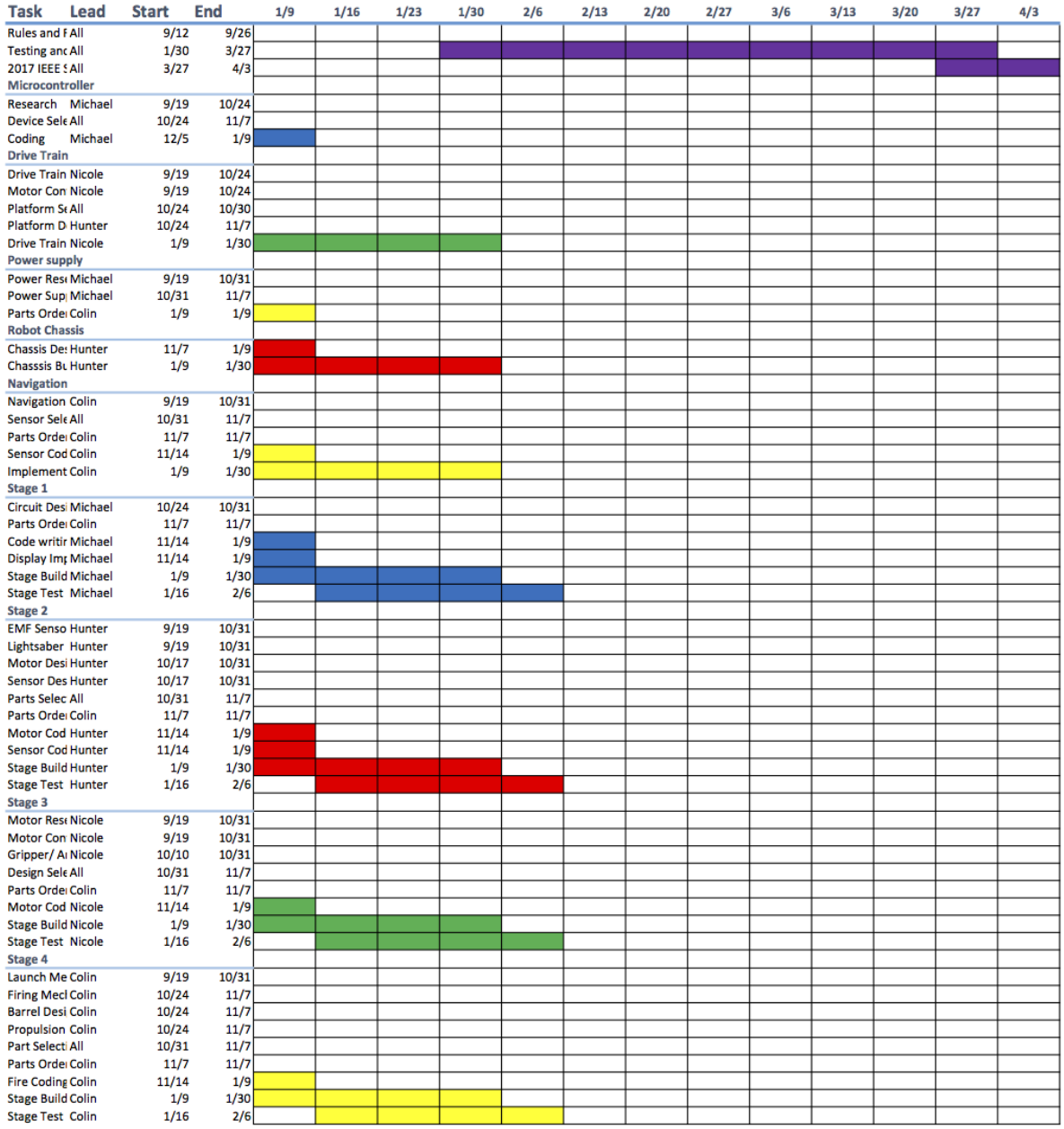
Task	Lead	Start	End
Rules and F	All	9/12	9/26
Testing and	All	1/30	3/27
2017 IEEE S	All	3/27	4/3
Microcontroller			
Research	Michael	9/19	10/24
Device Sele	All	10/24	11/7
Coding	Michael	12/5	1/9
Drive Train			
Drive Train	Nicole	9/19	10/24
Motor Con	Nicole	9/19	10/24
Platform Se	All	10/24	10/30
Platform D	Hunter	10/24	11/7
Drive Train	Nicole	1/9	1/30
Power supply			
Power Resi	Michael	9/19	10/31
Power Sup	Michael	10/31	11/7
Parts Orde	Colin	1/9	1/9
Robot Chassis			
Chassis De	Hunter	11/7	1/9
Chasssis Bu	Hunter	1/9	1/30
Navigation			
Navigation	Colin	9/19	10/31
Sensor Sele	All	10/31	11/7
Parts Orde	Colin	11/7	11/7
Sensor Cod	Colin	11/14	1/9
Implement	Colin	1/9	1/30
Stage 1			
Circuit Des	Michael	10/24	10/31
Parts Orde	Colin	11/7	11/7
Code writir	Michael	11/14	1/9
Display Im	Michael	11/14	1/9
Stage Build	Michael	1/9	1/30
Stage Test	Michael	1/16	2/6
Stage 2			
EMF Senso	Hunter	9/19	10/31
Lightsaber	Hunter	9/19	10/31
Motor Desi	Hunter	10/17	10/31
Sensor Des	Hunter	10/17	10/31
Parts Selec	All	10/31	11/7
Parts Orde	Colin	11/7	11/7
Motor Cod	Hunter	11/14	1/9
Sensor Cod	Hunter	11/14	1/9
Stage Build	Hunter	1/9	1/30
Stage Test	Hunter	1/16	2/6
Stage 3			
Motor Resi	Nicole	9/19	10/31
Motor Con	Nicole	9/19	10/31
Gripper/ A	Nicole	10/10	10/31
Design Sele	All	10/31	11/7
Parts Orde	Colin	11/7	11/7
Motor Cod	Nicole	11/14	1/9
Stage Build	Nicole	1/9	1/30
Stage Test	Nicole	1/16	2/6
Stage 4			
Launch Me	Colin	9/19	10/31
Firing Mecl	Colin	10/24	11/7
Barrel Desi	Colin	10/24	11/7
Propulsion	Colin	10/24	11/7
Part Selecti	All	10/31	11/7
Parts Orde	Colin	11/7	11/7
Fire Coding	Colin	11/14	1/9
Stage Build	Colin	1/9	1/30
Stage Test	Colin	1/16	2/6



B. Spring 2017 Schedule

In the spring time, it took longer than expected to get the part together and get the robot built. We expected to have three additional weeks at the end of the semester to test our full design before the competition. Due to needing excess parts and waiting for the machine shop to finish milling out our stage, our deadline of finishing the robot in its fullest was delayed two weeks. Once everything was ready, it took an extra couple day to fully wire the robot and get it ready to test. Due to this we were left to only a couple days to test our navigation coding for the entire robot.

Initial Gantt Chart:



Adjusted Gantt Chart:

Task	1/9	1/16	1/23	1/30	2/6	2/13	2/20	2/27	3/6	3/13	3/20	3/27	4/3	4/10	4/17
Rules and Research															
Testing and Design Implementation															
2017 IEEE Southeast Con Hardware Chall.															
Microcontroller															
Research															
Device Selection															
Coding															
Drive Train															
Drive Train Research															
Motor Controller Selection															
Platform Selection															
Platform Design															
Drive Train Build															
Power supply															
Power Research															
Power Supply Selection															
Parts Ordered															
Robot Chassis															
Chassis Design															
Chasssis Build															
Wiring/Electrical															
Navigation															
Navigation Research															
Sensor Selection															
Parts Ordered															
Sensor Coding															
Implementation															
Stage 1															
Circuit Design															
Parts Ordered															
Code writing															
Display Implementation															
Stage Build															
Stage Test															
Stage 2															
EMF Sensor Research															
Lightsaber Motor Research															
Motor Design															
Sensor Design															
Parts Selection															
Parts Ordered															
Motor Coding															
Sensor Coding															
Stage Build															
Stage Test															
Stage 3															
Motor Research															
Motor Controller Research															
Gripper/ Arm Selection and Design															
Design Selection															
Parts Ordered															
Motor Coding															
Stage Build															
Stage Test															
Stage 4															
Launch Mechanism Research															
Firing Mechanism Design															
Barrel Design															
Propulsion Design															
Part Selection															
Parts Ordered															
Fire Coding															
Stage Build															
Stage Test															

*The red bars represent tasks that were behind schedule. Green was completed items and yellow were in progress items.

VI. Budget

The initial budget for the project was $750. Table 6.1 shows the purchases that were made for the project. After the final purchase of all the needed components, the remaining budget after the third presentation was $190. That budget was left without any contingency parts. Some money was saved in the development of stage one and stage four, allowing the budget to grow a little more, allowing $190 of contingency parts to be bought and taken to competition, all while leaving $20 left in the total $750 budget. The budgeting in the project went very well and the ability to adhere to the budget throughout the project has been great. There were no real budget complications, stage two and three needed some of the money saved from stage one and stage four. However, the greatest accomplishment was that there were no budget overruns and were also able to keep from scraping other old robots. This proved effective because the robot got what it needed, not what could be reused because of budget concerns.

Table 6.1: Budget Spend

Product	*Quantity*	*Total*
Rotary Encoder	1	$3.95
Clear Plastic Knob	1	$0.95
Continuous Servo	1	$11.95
Drivetrain	1	$78.93
Arduino Mega	2	$7C.98
Magnetic Sensor	1	$2.00
IR Sensors	8	$108.20
Hardware	1	$34.19
Chassis	1	$32.25
Hall Effect Sensors	4	$10.27
MisC. Vex	1	$98.99
MisC. RobotShop	1	$79.92
MisC. Hardware	1	$191.90
Shipping	1	$21.15
Total	-	$730

VII. Conclusion

The overall objective of this project was to develop an autonomous robot that could compete in the 2017 SoutheastCon Robotics competition. The robot was somewhat successful. There were various tasks the robot had to complete. The robot was successfully designed to address each stage. The main problems with were a short circuit caused by a loose power wire, navigation, and the drivetrain. Future improvements could be made to the wiring. All wires could have caps placed on them to prevent any metal from touching. For navigation, it was difficult to get stable readings from the sensors. When readings were acquired, the tank treads made navigating difficult. Initially, tank treads were chosen so the stage 4 target could be approached closer. However, the Nerf gun proved enough range to shoot from stage 2. Omni-directional wheels would have been more beneficial for this project due to their ability to move in the x and y planes. Finally, navigation should have been made a larger priority. Due to the number of arenas at the completion, there was limited time to perfect navigation. If navigation was tested more at the testing arena, then stage testing could have been the focus instead. This would have been easier since the stages were portable. To conclude, the robot designed, F2-P2, can function correctly if navigation is adjusted. The robot has the capability to complete all stages well if the robot can accurately approach the stages.

VIII. REFERENCES

[1] IEEE Southeast Division. "Southeast Con 2017 Hardware Competition Rules."Episode MMXVII: The Engineering Force Awakens (20117): 1-18.Southeast Con 2017 Hardware Competition Rules. IEEE, 3 Apr. 2017. Web. 30 Sept. 2016. <http://sites.ieee.org/southeastcon2017/files/2016/04/MMXVII.pdf>.

APPENDIX A – COMPONENTS & DATA SHEETS

A. Extra Component Information

Table A.1: Arduino Mega Pin Connections

Component	PIN
Stage 1	Digital – 45, 47, 49, 51, 53
	Analog – A14
Stage 2	Analog – A0
	PWM – 3
Stage 3	Digital – 22
	PWM – 8-10
Stage 4	Digital – 31
Sensors	Analog – A4-A9
LCD	COMM – SCL, SDA
Rail System	PWM – 6
Motors	PWM 5, 7

APPENDIX II

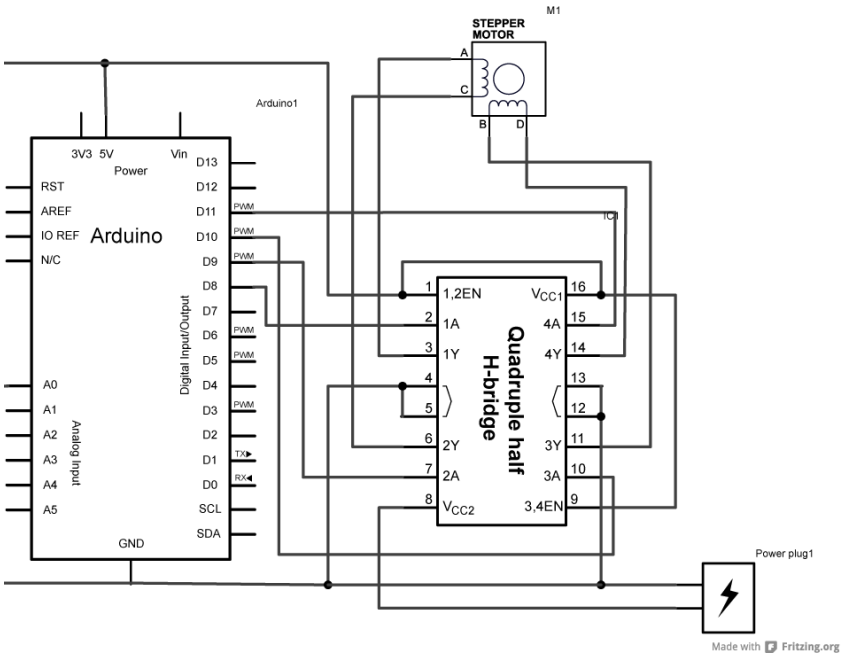


Figure A.1: Stage 3 Connection

B. Data Sheet Links

- *Stage 2 Servo – https://content.vexrobotics.com/docs/inventors-guide/servo-motor-1-06-05.pdf*
- *Stage 2 Hall Effect Sensor – www.allegromicro.com/~/media/Files/Datasheets/A1301-2-Datasheet.ashx*
- Stage 3 Stepper Motor – https://www.sparkfun.com/datasheets/Robotics/SM-42BYG011-25.pdf
- Stage 3 H-Bridge – https://www.ti.com/lit/ds/symlink/sn754410.pdf
- Stage 3 Servo – http://cdn.sparkfun.com/datasheets/Robotics/S05NF%20STD.pdf
- Drivetrain Motors – https://content.vexrobotics.com/docs/instructions/276-2177-instr-0414_v2.pdf
- Navigation Long Range – http://www.robotshop.com/media/files/pdf/datasheet-gp2y0a02yk0f.pdf
- Navigation Short Range – http://www.robotshop.com/media/files/pdf/datasheet-gp2y0a41sk0f.pdf

C. Code

Final Code for Full Design

```
#include <LiquidCrystal_I2C.h>// include the library code
#include <Servo.h>
#include <Stepper.h>

const int stepsPerRevolution = 113;  //146 change to fit
encoder rotation
const int startButton = 52;

//motor pin assignment
const int motorLeft = 7;
const int motorRight = 5; //backwards
const int motorRail = 6;

//speed variables set
int fullFront = 230;
int fullBack = 150;
int halfFront = 210;
int halfBack = 170;
int off = 190;

//sensor pin assignment
const int frontShort = A6;
const int frontLong = A7;
const int backShort = A4;
const int backLong = A5;
const int sideFront = A8;
const int sideBack = A9;

//stage 1
const int s11 = 45;
const int s12 = 47;
const int s13 = 49;
const int s14 = 51;
const int s15 = 53;
const int aRead = A14;

//stage 2 pins
const int gaussPin = A12;

//stage 3 pins

//stage 4 pins
const int nerfPin = 31;



// initialize the stepper library on digital pin 22, and PWM
9-11
Stepper myStepper(stepsPerRevolution, 22, 9, 10, 11);
Servo myservoS2, myservoS3;
LiquidCrystal_I2C lcd(0x27,20,4);


void setup() {
  Serial.begin(9600);                    // setting for screen
output

  pinMode(startButton, INPUT);


  pinMode(gaussPin, OUTPUT);                //stage 2 pin
  myservoS2.attach(3);

  myservoS3.attach(2);                    //stage 3 pin
  myStepper.setSpeed(60);                   // set the speed at
60 rpm:

  pinMode(nerfPin, OUTPUT);                 //stage 4 pin
}

void loop()
{
  digitalWrite(nerfPin, HIGH);
  lcd.clear();

  int code1, code2, code3, code4, code5, val, i;

//PushButton code
  do{
    val = digitalRead(startButton);
```

```
  Serial.println(val);
 }while (val == LOW );

 delay(1000);

 moveToStage1();

 railSystem1();

 code1 = stage1(45);
 code2 = stage1(47);
 code3 = stage1(49);
 code4 = stage1(51);
 code5 = stage1(53);

 screen();

 moveToStage3();

 railSystem3();

 stage3(code1);
 stage3(-code2);
 stage3(code3);
 stage3(-code4);
 stage3(code5);

 moveToStage2();

 stage2();

while(1);
}
```

Stage 1 Code

```
int stage1(int pin)
{

/************************Declarations*************
*************/

 pinMode(pin, OUTPUT);                    //set load pin to
an output

 int sensorRead0;
 int sensorRead1;
 float voltage0;
 float voltage1;
 float grace = 0.2;                       //additional range for
leniency in calculations
 int component;                           // Output
/************************Implementation**********
***************/

 digitalWrite(pin, HIGH);                 // turn on pullup
resistors
 sensorRead0 = analogRead(A14);           // Retrieve
value from anaolg A0
 delay(1000);                             // wait 1 second
 sensorRead1 = analogRead(A14);           // Retrieve
value from anaolg A0
 digitalWrite(pin, LOW);                  // turn off pullup
resistors

 voltage0 = sensorRead0 * (5.0/ 1023.0);  //Scale value
to the  actual voltage reading
 voltage1 = sensorRead1 * (5.0/ 1023.0);  //Scale value
to the  actual voltage reading

/************************Output
Logic************************/

 if( voltage0 > (voltage1 + grace) )      //If voltage drops
over time, capacitor is chosen
 {
  Serial.println("Capacitor");
  component = 3;
 }
 else if( voltage0 < (voltage1 - grace) )  //If voltage rises
over time, inductor is chosen
 {
  Serial.println("Inductor");
  component = 4;
 }
 else                                     //Voltage doesn't change
 {
  if( voltage0 < 5.2 && voltage0 > 4.8 )   //If around 5V,
wire is chosen
  {
   Serial.println("Wire");
   component = 1;
  }
  else if( voltage0 < 2.7 && voltage0 > 2.3 ) //If around
2.5V, resistor is chosen
  {
   Serial.println("Resistor");
   component = 2;
  }
  else if( voltage0 < 4.6 && voltage0 > 4.2 ) //If around
4.4, diode is chosen
  {
   Serial.println("Diode");
   component = 5;
  }
  else
  {
   Serial.println("other");
   component = 0;
  }
 }

 pinMode(pin, INPUT);                     //Pinmode needs to
be reet before next pin is set
```

```
/************************Serial Print for
testting************************/

   Serial.print("Voltage 0 = ");
   Serial.println(voltage0);
   Serial.print("Voltage 1 = ");
   Serial.println(voltage1);
   Serial.println(pin);

   return component;
}
```

Stage 2/4 Code

```
void hit();

void stage2() {
/************************Declarations****
*********************/

                    //Pin assignment for gauss
reading
myservoS2.attach(4);                  //servo object
assigned to pin 4
  int gss;                            //measures gauss
reading
  int gaussPin = A12;                  //Pin
assignment for gauss reading
  int basefeild;
  int verify = 0;                     //check value for
false hits
  int time = 0;

  myservoS2.write(90);                  //servo gets
into position
  delay(800);

  gss = analogRead(gaussPin);   //reads sensor
value
  basefeild = gss;  //initalize gauss reading of feild
  Serial.print abs(gss);
  Serial.println(" Gauss \tBase Gauss \t");

  hit();
  Serial.println("HIT \t");

  //this should wait for 28.5 seconds + 3 seconds
from hit
  //25.5 from while loop and 3 seconds for hit()
function
  while( time <= 255 )
  {
    gss =analogRead(gaussPin);
    Serial.print (gss);
    Serial.print(" Gauss \t");
    if( (gss > (basefeild +1)) || (gss < (basefeild -1))
)               //electromagnetic field is produced
      verify++;
    else
      verify = 0;
    if( verify == 3 )                  //feild stays on
for .3 second
    {                                 //(check for false
positives)
      hit();
       Serial.print("HIT \t");
      verify = 0;
    }
    delay(100);                       //wait for .1
seconds
    time++;
    Serial.print(time/10);
    Serial.print(" sec \t");
    Serial.print(verify);
    Serial.println(" verified");
  }

  hit();
  Serial.println("HIT \t");
  myservoS2.write(180);
  delay(500);

  digitalWrite(nerfPin, LOW);
  delay(10000);
  digitalWrite(nerfPin, HIGH);
}


/************************Output
Logic************************/

void hit()
{
    myservoS2.write(45);
    delay(750);
    myservoS2.write(90);
```

```
   delay(750);
}
```

Stage 3 Code

```
void stage3(int digit)
{
/************************Declarations**********************/

  int turn = 1;
  int steps;

/************************Output Logic************************/

  if ( digit > 0 )
    steps = -stepsPerRevolution;
  else
    steps = stepsPerRevolution;

  digit = abs(digit);

  while (turn <= digit)
  {
    myStepper.step(steps);
    delay(500);
    turn++;
  }
}


void servoSet(int value)
{
  if ( value == 0 )
  {
    myservoS3.write(180);
    delay(2000);
  }
  else
  {
    myservoS3.write(90);
    delay(1000);
  }
}
```

Rail System Code

```
void railSystem1()
{
  analogWrite(motorRail, 160);
  delay(1000);
  analogWrite(motorRail, off);
}

/************************************
*********************************/

void railSystem3()
{
  analogWrite(motorRail, halfFront);
  delay(1900);
  analogWrite(motorRail, off);
}
```

Navigation Code

```
int frontShortLength;
  int frontLongLength;
  int backLongLength;
  int backShortLength;
  int sideFrontLength;
  int sideBackLength;
  int baseRead;

  float frontShortCm;
  float frontLongCm;
  float backLongCm;
  float backShortCm;
  float sideFrontCm;
  float sideBackCm;
  float baseReadCm;

void moveToStage1()
{
    int avg=0;
    int sum=0;

    for(int i=0; i<100; i++)
    {
    delay(15);
    baseRead = analogRead(sideFront);
    baseReadCm = 9462 / (baseRead-16.92);
    sum+= baseReadCm;
    }
    avg = sum/100.0;

    frontLongLength = analogRead(frontLong);
    sideFrontLength = analogRead(sideFront);
```

```
   sideBackLength = analogRead(sideBack);

   frontLongCm = 9462 / (frontLongLength-
16.92);
   sideFrontCm = 9462 / (sideFrontLength-16.92);
   sideBackCm = 9462 / (sideBackLength-16.92);



 analogWrite(motorLeft, halfFront);
 analogWrite(motorRight, halfFront);
 delay(3000);

 do{
    sideFrontLength = analogRead(sideFront);
    sideFrontCm = 9462 / (sideFrontLength-
16.92);

    if(sideFrontCm > avg +2){
      analogWrite(motorLeft, halfFront);
      analogWrite(motorRight, off);
    }
    else if(sideFrontCm < avg -2){
      analogWrite(motorLeft, halfFront);
      analogWrite(motorRight, off);
    }
    else{
      analogWrite(motorLeft, halfFront);
      analogWrite(motorRight, halfFront);
    }
      Serial.print(avg);
 }while(frontShortCm > 6);

 analogWrite(motorLeft, off);
 analogWrite(motorRight, off);


}
void moveToStage3()
{
   int avg=0;
   int sum=0;

   for(int i=0; i<100; i++)
   {
   delay(15);
   baseRead = analogRead(sideBack);
   baseReadCm = 9462 / (baseRead-16.92);
   sum+= baseReadCm;
   }
   avg = sum/100.0;

   int avgF=0;
   int sumF=0;

   for(int i=0; i<100; i++)
   {
   delay(15);
   baseRead = analogRead(sideFront);
   baseReadCm = 9462 / (baseRead-16.92);
   sumF+= baseReadCm;
   }
   avgF = sumF/100.0;
   do{

   backLongLength = analogRead(backLong);
   sideFrontLength = analogRead(sideFront);
   sideBackLength = analogRead(sideBack);

   backLongCm = 9462 / (backLongLength-
16.92);
   sideFrontCm = 9462 / (sideFrontLength-16.92);
   sideBackCm = 9462 / (sideBackLength-16.92);


      if( sideBackCm > (avg + 2) )
//veering left correction
      {
        analogWrite(motorLeft, halfBack);
        analogWrite(motorRight, off);
      }
      else if( sideBackCm < (avg - 2) )
//veering right correction
      {
        analogWrite(motorLeft, off);
        analogWrite(motorRight, halfBack);
      }
      else if( sideFrontCm < (avgF - 2) )
//veering left correction
      {
        analogWrite(motorLeft, halfBack);
        analogWrite(motorRight, off);
      }
```

```
      else if( sideFrontCm > (avgF + 2) )
//veering right correction
      {
       analogWrite(motorLeft, off);
       analogWrite(motorRight, halfBack);
      }
      else
      {
       analogWrite(motorLeft, halfBack);
       analogWrite(motorRight, halfBack);
      }

  Serial.print(avg);
  Serial.print('\t');
  Serial.print(avgF);
  Serial.print('\t');
  Serial.print(sideFrontCm);
  Serial.print('\t');
  Serial.println(sideBackCm);

  }while(backLongCm > 25);
   do{

   backShortLength = analogRead(backShort);
   sideFrontLength = analogRead(sideFront);
   sideBackLength = analogRead(sideBack);

   backShortCm = 9462 / (backShortLength-
16.92);
   sideFrontCm = 9462 / (sideFrontLength-16.92);
   sideBackCm = 9462 / (sideBackLength-16.92);

  if( sideBackCm > (avg + 2) )            //veering
left correction
      {
       analogWrite(motorLeft, halfBack);
       analogWrite(motorRight, off);

      }

      else if( sideBackCm < (avg - 2) )
//veering right correction
      {
       analogWrite(motorLeft, off);
       analogWrite(motorRight, halfBack);
      }
      else
      {
       analogWrite(motorLeft, halfBack);
       analogWrite(motorRight, halfBack);
      }
  Serial.print(avg);
  Serial.print('\t');
  Serial.print(avgF);
  Serial.print('\t');
  Serial.print(sideFrontCm);
  Serial.print('\t');
  Serial.println(sideBackCm);

  }while(backShortCm > 6);

}

void moveToStage2()
{
   analogWrite(motorLeft, fullBack);
   analogWrite(motorRight, fullFront);
   delay(950);
   analogWrite(motorLeft, off);
   analogWrite(motorRight, off);
}


void moveToStage22()
{
     digitalWrite(nerfPin, HIGH);
  lcd.clear();

  int code1, code2, code3, code4, code5, val, i;

//PushButton code
  do{
   val = digitalRead(startButton);
   Serial.println(val);
  }while (val == LOW );

  delay(7000);

   digitalWrite(nerfPin, LOW);
   delay(7000);
   digitalWrite(nerfPin, HIGH);

   analogWrite(motorLeft, fullFront);
   analogWrite(motorRight, fullFront);
```

```
delay(3000);
analogWrite(motorLeft, off);
analogWrite(motorRight, off);
delay(27000);
analogWrite(motorLeft, halfBack);
analogWrite(motorRight, halfBack);
delay(1000);
analogWrite(motorLeft, fullFront);
analogWrite(motorRight, fullFront);
delay(1000);
analogWrite(motorLeft, off);
analogWrite(motorRight, off);

digitalWrite(nerfPin, LOW);
delay(20000);
digitalWrite(nerfPin, HIGH);

}
```

APPENDIX B – TEST PLAN DATA

A. March 22, 2017

Table B.1: Robot Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.		X
3	The robot is in the white square with front facing stage 3.		X
4	Robot is turned ON. LCD is on and not flashing.		X
5	Push blue button. Robot is moving toward stage 1.		X
6	Center rail pushed out toward stage 1.		X
7	Stage 1 makes contact with the stage 1 arena.		X
8	LCD displays code.		X
9	Robot reverses toward stage 3.		X
10	Center rail pushed out toward stage 3.		X
11	Servo arm is lifted.		X
12	Band makes contact with stage 3 encoder.		X
13	Stepper motor turns.		X
14	Stepper motor inputs code on LCD.		X
15	Servo arm resets.		X
16	Robot moves toward stage 2.		X
17	Robot strikes arena lightsaber.		X
18	Robot strikes when field is active.		X
19	Robot strikes at final 28 seconds.		X
20	Robot launches 3 nerf darts toward stage 4 into the hole.		X
21	Robot is turned OFF.		X

Table B.2: Stage 1 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 1 Test code.	X	
4	Voltage is coming through pins.	X	
5	LCD is displaying code.	X	

Table B.3: Stage 2 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 2 Test code.	X	
4	A difference is observed when the magnetic field is ON verses OFF.	X	
5	The lightsaber hits when the field is ON.	X	

Table B.4: Stage 3 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 3 Test code.	X	
4	Servo arm comes up.	X	
5	Servo turns counterclockwise.	X	
6	Servo turns clockwise.	X	
7	Servo arm resets.	X	

Table B.5: Stage 4 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Battery pack is ON.	X	
4	3V is going to the gun.	X	
5	Shots are delayed.	X	
6	Gun is positioned correctly at target.	X	
7	3 darts fire through the hole at stage 4.	X	

Stage 1:

After completion of the code test, a number sequence was written onto the LCD. The LCD would read “FAMU - FSU COE” followed by the sequence for the components, the name of the robot “F2 - P2”.

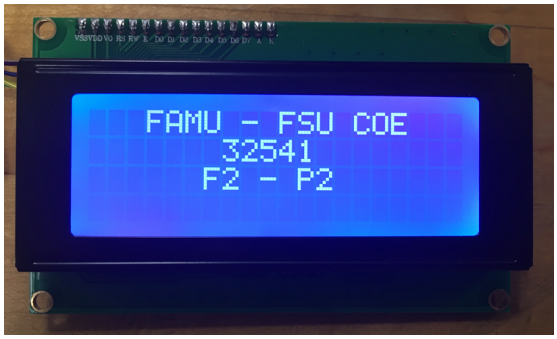


Figure B.1: LCD Display

Stage 2:
The following results were obtained for the Allegro Hall Effect sensor.

```
|----------------------0----------------------|-5Gauss
|----------------------0----------------------|-7Gauss
|----------------------0----------------------|-5Gauss
|----------------------0----------------------|-7Gauss
|----------------------0----------------------|-5Gauss
|----------------------0----------------------|-7Gauss
```

Figure B.2: Data with no field

```
|----------------------0----------------------|-7Gauss
|----------------------0----------------------|-7Gauss
|----------------------0----------------------|-7Gauss
|----------------------0----------------------|-8Gauss
|----------------------0----------------------|-8Gauss
|----------------------0----------------------|-8Gauss
```

Figure B.3: Data with field

B. March 29, 2017

Table B.6: Robot Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	The robot is in the white square with front facing stage 3.	X	
4	Robot is turned ON. LCD is on and not flashing.	X	
5	Push blue button. Robot is moving toward stage 1.	X	
6	Center rail pushed out toward stage 1.	X	
7	Stage 1 makes contact with the stage 1 arena.	X	
8	LCD displays code.	X	
9	Robot reverses toward stage 3.	X	
10	Center rail pushed out toward stage 3.	X	
11	Servo arm is lifted.	X	
12	Band makes contact with stage 3 encoder.	X	
13	Stepper motor turns.	X	
14	Stepper motor inputs code on LCD.	X	
15	Servo arm resets.	X	
16	Robot moves toward stage 2.		X
17	Robot strikes arena lightsaber.		X
18	Robot strikes when field is active.	X	
19	Robot strikes at final 28 seconds.		X
20	Robot launches 3 nerf darts toward stage 4 into the hole.	X	
21	Robot is turned OFF.	X	

Table B.7: Stage 1 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 1 Test code.	X	
4	Voltage is coming through pins.	X	
5	LCD is displaying code.	X	

Table B.8: Stage 2 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 2 Test code.	X	
4	A difference is observed when the magnetic field is ON verses OFF.	X	
5	The lightsaber hits when the field is ON.	X	

Table B.9: Stage 3 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 3 Test code.	X	
4	Servo arm comes up.	X	
5	Servo turns counterclockwise.	X	
6	Servo turns clockwise.	X	
7	Servo arm resets.	X	

Table B.10: Stage 4 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Battery pack is ON.	X	
4	3V is going to the gun.	X	
5	Shots are delayed.	X	
6	Gun is positioned correctly at target.	X	
7	3 darts fire through the hole at stage 4.	X	

C. April 12, 2017

Table B.11: Robot Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	The robot is in the white square with front facing stage 3.	X	
4	Robot is turned ON. LCD is on and not flashing.	X	
5	Push blue button. Robot is moving toward stage 1.	X	
6	Center rail pushed out toward stage 1.	X	
7	Stage 1 makes contact with the stage 1 arena.		X
8	LCD displays code.	X	
9	Robot reverses toward stage 3.		X
10	Center rail pushed out toward stage 3.		X
11	Servo arm is lifted.	X	
12	Band makes contact with stage 3 encoder.	X	
13	Stepper motor turns.	X	
14	Stepper motor inputs code on LCD.	X	
15	Servo arm resets.	X	
16	Robot moves toward stage 2.		X
17	Robot strikes arena lightsaber.	X	
18	Robot strikes when field is active.	X	
19	Robot strikes at final 28 seconds.		X
20	Robot launches 3 nerf darts toward stage 4 into the hole.	X	
21	Robot is turned OFF.	X	

Table B.7: Stage 1 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 1 Test code.	X	
4	Voltage is coming through pins.	X	
5	LCD is displaying code.	X	

Table B.8: Stage 2 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 2 Test code.	X	
4	A difference is observed when the magnetic field is ON verses OFF.	X	
5	The lightsaber hits when the field is ON.	X	

Table B.9: Stage 3 Test Plan

Step	*Instructions*	*Pass*	*Fail*

1	Battery is charged.	X	
2	All wires connected.	X	
3	Upload Stage 3 Test code.	X	
4	Servo arm comes up.	X	
5	Servo turns counterclockwise.	X	
6	Servo turns clockwise.	X	
7	Servo arm resets.	X	

Table B.10: Stage 4 Test Plan

Step	*Instructions*	*Pass*	*Fail*
1	Battery is charged.	X	
2	All wires connected.	X	
3	Battery pack is ON.	X	
4	3V is going to the gun.	X	
5	Shots are delayed.	X	
6	Gun is positioned correctly at target.	X	
7	3 darts fire through the hole at stage 4.	X	

APPENDIX C – USER GUIDE

I. SETUP

A. Correct Setup

The underside of the robot should be checked for any loose wires on the Arduino and power distribution shield. If any wires are unattached, reconnect them and refer to the Troubleshooting section if issues persist. The gate holding the power distribution shield needs to be closed. The battery is unplugged for safety and needs to be reattached to the connectors before use. The robot can be placed anywhere within the staring 15 x 15 white square. The front of the robot should be facing Stage 3, with the rear facing Stage 1. The sides of the robot should be parallel with the short walls of the arena. The front is designated with the plastic eyes. If the robot is not lined up correctly, all stages will not be completed. Stage 1 needs to be reset by pushing it toward the robot until it hits the plate. Stage 3's rubber band should be pulled past the servo arm to hold the band down. Nerf darts should be loaded into Stage 4's barrel, and the Nerf gun's battery pack turned on. The toggle switch can now be flipped to the ON position.

B. Saftey Precautions

The underside of the robot has many wires. The robot should only be handled by grabbing the top plate of the robot or the tracks. The LCD acts as an indicator for any problems with the robot. If the LCD is not lit up or is flickering, immediately turn of the robot using the toggle switch and refer to the Troubleshooting section. Once the blue button is pressed, the robot will begin moving. Avoid the blue button until the user is ready for use.

II. Operation

A. Proper Operation

Since the robot is fully autonomous, the robot will move and complete all the stages after pushing the blue button. To terminate or reset the robot, the toggle switch should be moved to the OFF position. After completely turning off the robot, follow the setup instructions again.

B. Operational hazards

The lightsaber for Stage 2 swings outside the robot. Avoid going near the lightsaber when in use. The nerf guns' fires projectiles and should never be pointed at a person's face while in use. Further, the arena does not have a catching net for the darts. The projectiles are only fast enough to hurt an eye if shot into one. No other damage should occur.

III. Troubleshooting

Before troubleshooting, ensure the toggle switch is in the OFF position.

A. Chassis

1) Why are components or metal plates moving easily?

Due to the stationary manner of the chassis, troubleshooting consists purely of verifying that all bolts and nuts have been tightened and are securely holding each component to the chassis. Replace bolts and nuts as needed.

2) Why is Stage 1 and Stage 3 not moving out?

The toothed rail section has a tendency to over extend from the motor and bracket and may need to be reset by slowly pushing the rail back into the robot will the robot is off to avoid the motors from grinding. The non-toothed section of the rail can also slip too far and will also need to be pushed back into the robot. To assist in this, a rubber band can also be attached between the standoffs and wrapped around the mounting bolt of stage one components to the linear rail. The elasticity of the rubber band will allow the rail to return inside the robot when the motor extends the other portion of the rail outside the robot. If the motor has no function, replace the motor with the motor listed in the Components section.

B. Microcontroller

1) Why is the Arduino light off?

First, make sure the battery is operational by going to section C. If the battery is functioning correctly, the Arduino light being off can represent two problems. First, check to make sure there is a wire from the VIN pin on the Arduino to the 7.2 V line on the power distribution shield. It this wire is missing, connect a wire at these two pins. If the Arduino still will not power on, turn off the battery and check for any loose wires. No light on the Arduino may represent a short circuit. If the Arduino still does not function after checking the wiring, replace the Arduino with another Arduino Mega. The Arduino is connected via a fastener.

2) Where does this wire that came out of the Arduino go?

Check Appendix I for pin placement of all components. Replace the wire in the correct pin for proper function.

C. Battery

1) I think the battery died.

First, check to make sure the battery is connected to the battery wires, and the toggle switch is on the ON postition. The battery can be checked using a multimeter. Place one prong of the multimeter on the 7.2 V line and one on the ground. If the reading is above 7.2 V, then the battery is functioning correctly.

However, all components run best when the battery is not used for more than two hours. If the battery needs to be replaced, remove the battery and replace it with the spare battery. Charge the old battery. Recheck the function of the battery.

D. Drivetrain

1) Why is the robot not moving?

First, make sure the battery is operational by going to section C. Check the motor wiring. All motors should be connected to the 7.2 V and ground lines. If the motors are still not operational, replace the motors. If there is still not function, the motor controllers may need to be replaced.

2) Why is the robot moving in the wrong direction?

If the robot is moving in the wrong direction, check the motor controller and motor connectors. Flip the orientation of both connectors labeled "R".

3) Why are each track moving in different direction?

If the motor is moving in different direction, one of the motor controller and motor connector is in the wrong direction. Flip one of the connectors until both tracks are moving in the same direction.

4) Why are the tracks sliding?

Sliding tracks mean the tank treads are too loose. Take out a tread or more until the tracks are tight.

5) Why am I misalighned with the stage?

Misalignment represents a problem with the sensors. Check wiring of the sensors to ensure everything is still connected. Troubleshooting the IR sensors requires opening the serial port monitor for the microcontroller and looking at the values that are being given for a certain distance. One thing to be wary of is to be sure that when troubleshooting that the IR sensor is being tested within its effective range.

E. Stage 1

1) Why am I reading the code "12345" on the LCD?

This code is the default code if there are no connections made. Ensure that the wiring is correct for this stage. If the wire is in order, push out the wires on the ends of the springs to make better contact with Stage 1.

2) Why is the LCD off or flickering?

First, make sure the battery is operational by going to section C. Check the LCD wiring. If both are correct, a short may be present. Go to secotion B of Troubleshooting if this is the case.

F. Stage 2

1) Why is the lightsaber not swinging or swinging too often?

Start by verifying that all wiring is fully connected and is hooked to the correct locations. Power to the power distribution, ground to ground, and signal to the correct I/O pin of the Arduino. Once all wiring has been confirmed, try double checking the electromagnetic sensor. The following test code can be uploaded to the Arduino to verify that the sensor is working and reading values that will allow for the stage to operate.

```
int gss;
void setup(){
  Serial.begin(9600);
}
void loop(){
  while(1)
  {
    int aValue =analogRead(0); // take note of the pin assignment here
    gss = map(aValue, 102, 922, -640, 640);
    gss = abs(gss);

    Serial.print(gss);
    Serial.println("Gauss");
    delay(100);
  }
  while(1);
}
```

Be sure to update the pin assignment of the test code before uploading. Once run, open the serial monitor and you will be able to see the Gauss readings from the sensor, there should be a noticeable change between no active field and an active field. If none is present, the sensor may need to be replaced.

If there is a noticeable field difference, and the stage still does not work, the issue may be the servo. Again, verify the wiring for the components. If the servo does not work, replace the servo with the same model and test again.

G. Stage 3

1) Why is the servo making a "clicking" noise?

This noise is normal and does not mean the servo is not functioning.

2) Why Is the belt not turning?

If the belt is not turning, there is a problem with the stepper motor or h-bridges. Ensure that the circuit is correct for Stage 3 by referring to Appendix II. If the wiring is correct, check the power lines on the breadboard using a multimeter to see if the correct voltage is coming through. If the wrong voltage is read, go to section B and C of Troubleshooting. Replace parts if needed.

3) Why is the servo arm not down?

The arm should reset after each run. If the arm does not reset, check wiring of the servo. The brown wire connects to ground, the red wire connects to the 5V line on the breadboard, and the signal should be in PWM 8 on the Arduino. If this is correct, the servo may need to be replaced.

H. Stage 4

1) Why are the darts firing at the correct time?

Be sure to make sure that the relay is not sitting on the bare aluminum top plate and has an insulator between it. Check that the Nerf gun battery is turned on. Replace batteries if the batteries have died. Check the wiring of the relay and nerf gun. If this is correct, ensure the battery and microcontroller are functioning correctly.