University of Southern Indiana
Pott College of Science, Engineering, and Education
Engineering Department

8600 University Boulevard
Evansville, Indiana 47712

IEEE SoutheastCon 2023

The Claw

Grant Davis and Dylan Fox
ENGR 491 – Senior Design (or TECH 471 - Senior Project)
Fall 2022

Approved by: ____
Faculty Advisor: Daniel Elg, Ph.D. Date

Approved by: ____
Department Chair: Paul Kuban, Ph.D. Date

Acknowledgements

Throughout the semester, several professors assisted the group in the completion of this project. Professor Kicklighter highlighted the basics of Arduino code and was always willing to meet to discuss hardware specifications. Dr. Elg worked on developing the report and presentation as well as being the project's advisor. Dr. Chlebowski worked with the development of the robot's communication system. Our thanks go to each of them. A special thanks goes out to Dr. Kuban and the USI engineering department for cultivating learning and supporting students to the best of their abilities.

Abstract:

The purpose of this project is to design and build an autonomous robot that can navigate the IEEE course. This design will be built in compliance with the rules specified by the IEEE SoutheastCon 2023 Hardware Competition. The group will use Arduino micro controllers and DC motors to move around the course. The robot will be designed to be able to complete all tasks in less than two minutes so that time is not an issue. The robot will also be required to not collide with the other autonomous robot or course walls. The design will keep this requirement in mind to ensure no collisions will occur.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The task of this project was to create an autonomous robot with the ability to navigate a track, sort items by color, and stack those items. The robot will be built for the IEEE SoutheastCon 2023 Hardware Competition and in compliance with the competition rules. While there are numerous ways in which this robot could be designed, the primary goal is to design an efficient system that can meet the competition objectives and score the maximum possible points given the time limit.

2 Objectives and Requirements

2.1 Objectives

Every year a hardware competition takes place with different IEEE teams from across the country. Typically, the hardware competition has teams build a robot to complete some list of tasks. This year a hurricane has wrecked across the IEEE SoutheastCon. The hurricane forced ducks (rubber ducks) out of the duck pond and knocked over three duck statues (3-D printed marshmallows). The ducks and statues are scattered throughout the park. The park also includes caged manatees and alligators that are very hungry. The group must design some type of hardware that will clean up the ducks and statues (putting them into a white circle) while also feeding the animals. The marshmallows are different colors and must be stacked in the center of a white circle or moved in a rectangular zone. To finish the clean up a switch must be pulled to start the firework celebration. Teams are given a maximum of three minutes to complete all tasks. This group will focus on the ducks and marshmallows.

2.2 Requirements

The objective for the IEEE SoutheastCon is to create an autonomous robot that can navigate a park and organize rubber ducks into a white circle while also stacking colored marshmallows in a specific order. The alternative object for the competition is moving the objects into a rectangular zone. The design will include Arduino controllers and DC motors for navigation. The robot design will be able to complete all tasks in under two minutes so that time is not an issue. The robot will be designed to not collide with other objects such as other robots or course walls.

3 Competition Background

The 2023 IEEE Hardware Competition addresses the autonomous robot field. The following information was gathered from the revised rules offered on the competition's official website [1].

3.1 Course Description

The main course is constructed of a 4-foot x 8-foot sheet of plywood with a boundary of 1-inch by 4-inch wooden beams on three sides (Actual: 0.75 in. x 3.5 in.) and 2 – 2-inch x 4-inch (Actual 1.5 in x 3.5 in) by 3 foot and 10.5 inches wooden beams on one end.

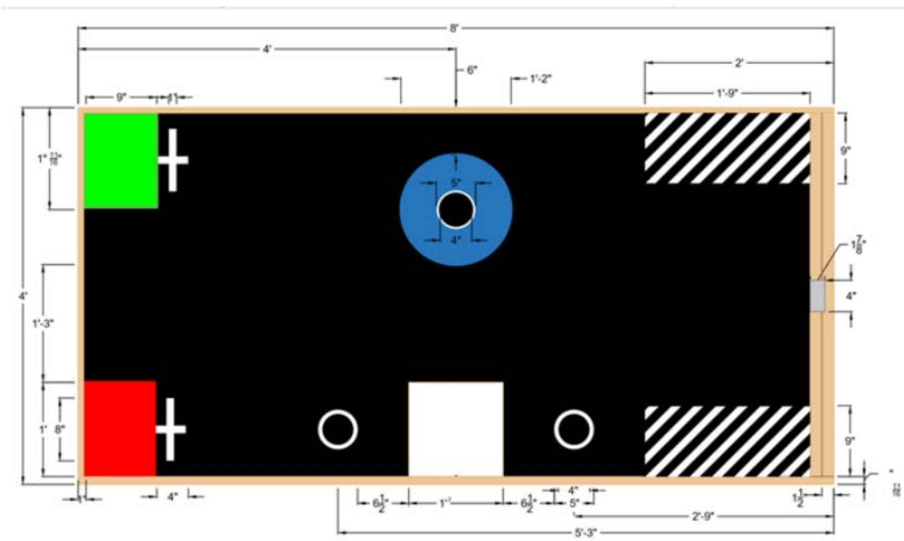

Figure 1: Main Course Design (AUTOCAD) [1]

The main course design is shown above in Figure 1. The inside dimensions are 3 feet by 10.5 inches wide and 7 feet by 8.25 inches in length with a .75-inch outline on 3 sides and a 3-inch outline on one end. The diagonal lines in the two recycle areas are at a 45-degree angle to the side of the board. A 12" x 12" starting zone is located in the bottom center portion of the track designated by the white square. Chip drop off positions are located in the top left and bottom left corners designated by the green and red rectangles. The green and red rectangles' positions can be swapped during the competition; however, they will always be located in the top left and bottom left corners. The blue circle that is five inches in diameter represents the location where the rubber ducks will be stacked.

3.2 Competition Rules

3.2.1 Robot Rules

The robot must be autonomous and must fit completely within the 12 inches by 12 inches by 12 inches starting area at the start of each match. The robots must always be wholly contained within the playing surface but may reach a maximum of 1 inch beyond the outside edge of any arena walls, and a maximum of 0.5 inch down the backside of the arena walls. If multiple independent robots are used, they must all start within the same starting space and split after the competition begins. The robots must have an easily reachable emergency cutoff switch to allow the robot team to disable the robot if necessary. The start switch for the robots must be a physical switch located on the robot either toggle or pushbutton. The robot shall not present any danger to the judges, spectators, playing arena, or area surrounding the arena [1].

3.2.2 Procedure Rules

1. Once teams have been called from the pre-match staging area to the match, they will have an additional two minutes to get their robots and their arenas ready for the match.
2. A single team member must place the robot inside the starting area. No portion of the robot may extend past the bounds of the starting area before the time starts.
3. The start switch will be actuated by the robot team once the judge indicates the start of the match.
4. Each run will last a total of three minutes.
5. The robot may complete any of the challenges in any order and may skip any of the challenges.
6. Once the robot activates the fireworks switch, the round ends.
7. The robot team can stop their robot at any time before the three-minute period and signal to the judge that they are finished with their match.
8. Each team will be allowed three separate runs to achieve the most points possible.
9. After the preliminary rounds, the top eight teams will advance to a single-elimination bracket. The teams will be seeded based on their placing in the preliminary rounds. The

top 2 teams will advance to the final round. The final round will be held at the awards banquet.

3.2.3 Scoring System

Points will be awarded for performing certain tasks throughout the competition. The points system is shown below in Table 1.

Points	Task
36	3 pedestals stacked on the duck pond statue location inside the inner circle and in the correct order (base level – white, second level – green, third level – red) with a pink duck on top. Without touching anything.
33	3 pedestals stacked on the duck pond statue location inside the inner circle and in the correct order (base level – white, second level – green, third level – red) with a yellow duck on top. Without touching anything.
30	3 pedestals stacked on the duck pond statue location inside the inner circle and in any order with a pink duck on top. Without touching anything.
27	3 pedestals stacked on the duck pond statue location inside the inner circle and in any order with a yellow duck on top. Without touching anything.
27	3 pedestals stacked on the duck pond statue location inside the inner circle and in any order with a yellow duck on top. Without touching anything.
24	3 pedestals stacked on the duck pond statue location inside the inner circle and in any order . Without touching anything.
24	3 pedestals stacked on the duck pond statue location inside the outer circle and in the correct order (base level – white, second level – green, third level – red) with a pink duck on top. Without touching anything
21	3 pedestals stacked on the duck pond statue location inside the outer circle and in the correct order (base level – white, second level – green, third level – red) with a yellow duck on top. Without touching anything.
18	3 pedestals stacked on the duck pond statue location inside the outer circle and in any order. Without touching anything.
18	3 pedestals stacked on the duck pond statue location anywhere in the park and in the correct order (base level – white, second level – green, third level – red) with a pink duck on top. Without touching anything
15	3 pedestals stacked on the duck pond statue location anywhere in the park and in the correct order (base level – white, second level – green, third level – red) with a yellow duck on top. With or without touching anything.
12	pedestals stacked on the duck pond statue location anywhere in the park and in any order. Without touching anything.
33	2 pedestals stacked on a non-pond statue location inside the inner circle and in the correct order (base level – white, second level – green) with a pink duck on top. Without touching anything.
30	2 pedestals stacked on a non-pond statue location inside the inner circle and in the correct order (base level – white, second level – green) with a yellow duck on top. Without touching anything
27	2 pedestals stacked on a non-pond statue location inside the inner circle and in any order with a pink duck on top. Without touching anything.

24	2 pedestals stacked on a non-pond statue location inside the inner circle and in any order with a yellow duck on top. Without touching anything.
21	2 pedestals stacked on a non-pond statue location inside the inner circle and in the correct order (base level – white, second level – green). The one with the duck above is 21. Without touching anything.
18	2 pedestals stacked on a non-pond statue location inside the inner circle and in any order . Without touching anything.
18	2 pedestals stacked on a non-pond statue location inside the outer circle and in the correct order (base level – white, second level – green) with a pink duck on top. Without touching anything.
15	2 pedestals stacked on a non-pond statue location inside the outer circle and in the correct order (base level – white, second level – green,) with a yellow duck on top. Without touching anything
12	2 pedestals stacked on a non-pond statue location inside the outer circle and in any order. Without touching anything.
9	2 pedestals stacked on a non-pond statue location anywhere in the park and in the correct order (base level – white, second level – green) with a pink duck on top. With or without touching anything.
6	2 pedestals stacked on a non-pond statue location anywhere in the park and in the correct order (base level – white, second level – green) with a yellow duck on top. With or without touching anything.
3	2 pedestals stacked on a non-pond statue location anywhere in the park and in any order . With or without touching anything.
2	Placing a duck, food chip, or pedestal in the recycling attraction
7	Each correct food chip fed to the manatees
7	Each correct food chip fed to the alligators
3	Each incorrect food chip fed to the manatees
3	Each incorrect food chip fed to the alligators
5	Each duck in the pond
10	Start the fireworks
10	Auto-start feature used and successful
5	Submitting an entry into a SoutheastCon 2023 Student Branch Promotion Competition. Limited to one entry

Table 1: Points System

3.2.4 Format

1. Each team will have three minutes per round.
2. The highest score of a single run will advance.
3. In the event of a tie, the team with the lowest total time will advance.
4. After the preliminary rounds, the top 8 teams will advance to a single elimination bracket. The teams will be seeded based on their placing in the preliminary rounds.

3.2.5 Strategy

For the IEEE Southeast con event there are two different ways of attacking the 2023 hardware competition. Time, efficiency, and overall points accumulated were some factors in creating the attacks for the robot.

The first attack was to have the robot focus on the stacking portion of the competition. In this, the robot would identify the objects based on color and shape to determine the stacking order. When looking at the distribution of points the IEEE club noticed that stacking the marshmallows in any order (i.e. ignoring color) with a pink duck on top would gain the club 30 points just 6 less than the max awarded points. The club believed this would be an achievable goal that was somewhat difficult. Within this strategy the question was asked “Should the robot carry more than one pedestal (marshmallow)?”. This question brought up the time constraint aspect of the competition. The club decided that the robot should in fact carry all three pedestals before stacking. The logic behind this decision is based on the fact the robot would be less time efficient if the robot stacked one object at a time rather than multiple at once.

The second attack was to approach the competition from a more simplistic view. In the bottom and top right portion of the track there are two spaces known as trash bins. For each object that is put into a trash bin 2 points are awarded. With 10 ducks and 7 marshmallows there are a total of 17 objects in play. This would allow for 34 points to be gained if all objects were placed in the trash bins. For this design the robot would need some type of dump truck feature to help carry several objects at one time. Although this design is simpler than the first it would require more time than the first design.

In the end, the IEEE club decided to use the second attack for the robot. This design would be simplistic and allow for more testing.

4 Robot Design

The robot’s design has been altered numerous times throughout the process of this project. However, as stated previously, there are many ways in which this design could be approached.

4.1 Design Requirements

1. The robot should be capable of stacking objects by color.
2. The robot must finish the tasks in three minutes or less.
3. The robot must be able to fit in a 1’ by 1’ by 1’ by 1’ cube (maximum size).
4. The robot must be able to run autonomously after start.
5. The robot cannot contain any explosive, pyrotechnic, toxic, or corrosive materials. Flammable liquids or gases are also prohibited.
6. The robot shall not present any danger to the judges, spectators, playing arena, or area surrounding the arena.
7. Robots must not collide into any other robots when completing tasks.
8. The start switch must be a physical switch located on the robot.
9. The Robot must operate within the duck garden (the built track shown in Figure 1).
10. The robot must have an emergency cut-off switch.

4.1.2 Performance

Design Requirements	**Engineering Requirements**	**Justification**
1	1.1 The Robot should be able to sort several objects based on different colors.	Rubber Ducks and Marshmallows of different color will be placed randomly throughout the track.
1	1.2 The Robot should then be able to stack these different color objects in a specific order.	Stacking the objects in the correct order allows for maximum points to be gained within the competition.
2	1.3 The Robot must finish its competition in three minutes.	IEEE allots three minutes of time for each run and allows no more than three minutes.

4.1.3 Functionality

Design Requirements	**Engineering Requirements**	**Justification**

4	2.1 The robot must run without outside intervention during the competition.	IEEE states that any robot within the Hardware competition must be autonomous.
8	2.2 The robot must be started with a mechanical switch.	IEEE competition rules state that a switch must be used to start the robot not a Bluetooth signal.
10	2.3 The robot must have a built-in emergency cut-off switch.	Competition rules state that in case of emergency a built in switch must be used to turn the robot off.

4.1.4 Economic

The robot is only being built for the IEEE Hardware Competition and will not be put into production, thus there are no economic requirements.

4.1.5 Energy

Design Requirements	**Engineering Requirements**	**Justification**
2	4.1 The robot should have battery life cable of running for 20 minutes.	Robots are allowed to run the track for three minutes with each test. The 20 minutes will allow the robot to run for several tests.

4.1.6 Environmental

Design Requirements	**Engineering Requirements**	**Justification**
7	5.1 Multiple robots will be running at the same time and should be designed so that they will not run into each other.	Robot collisions will cause interruption in their task completions. Ultimately increasing the overall time of the run.
9	5.2 The robot must stay within the track to complete the run.	Failure to stay within the required track will result in disqualification and an overall loss for the competition.

4.1.7 Public Health, Safety, and Welfare

Design Requirements	**Engineering Requirements**	**Justification**
6	6.1 The robot must not be harmful to any people spectating the run.	Harming a judge or spectator will result in disqualification.
5	6.2 A robot design must not be designed with harmful material.	Other than this being an IEEE rule, it goes without saying

		the robot cannot harm anyone.
10	6.3 The robot must be able to be shut down quickly in case of an emergency.	If something goes wrong the robot should be shut down in order to not get penalty points.

4.1.8 Manufacturability

Design Requirements	Engineering Requirements	Justification
3	7.1 The robots must be able to fit inside a 1' cube before turning the robot on.	This is with the rules published by the IEEE SoutheastCon 2023 rules.

4.1.9 Operational

Design Requirements	Engineering Requirements	Justification
9	8.1 The robot must stay within the duck garden while it completes its tasks.	If the robot leaves its zone than it could mess with the tasks of the other robot or it could run off the track.

4.1.10 Political

Since this robot will only be used in the IEEE Hardware Competition, compliance with government regulations is unnecessary. However, the group must follow the IEEE guidelines and rules.

4.1.11 Global

Since this robot will only be used in one country, the United States, there are no global requirements.

4.1.12 Social and Cultural

Design Requirements	Engineering Requirements	Justification
-	11.1 Any code or drawings must use the English language when needed.	This robot will only be used in the United States.

4.1.13 Usability

The competition clearly states that the robot must be autonomous meaning that there will be no user interface needed within the robot. Thus, there will be no usability requirements.

4.2 Division of Labor

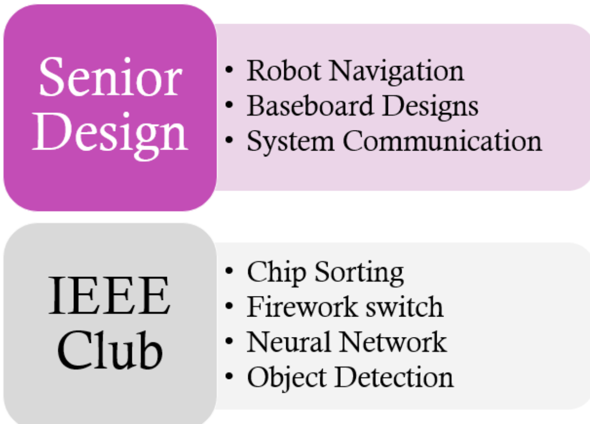

Figure 2: Workload Division

Within the IEEE hardware competition, there were two groups working together to build a robot to represent USI in the competition. There must be an important distinction made between the senior design team's work from the IEEE club's work. In Figure 2 above, there is a graphic showing the division of labor between the two groups. Senior design solely focused on robot navigation, the communication system within subsystems, and baseboard design and construction. The IEEE club focused on flipping the firework switch, sorting chips, and detecting objects with a neural network robotic claw. Although these ideas are not from senior design, they are worth mentioning because robot navigation must communicate with the IEEE club systems.

4.2 Scooping Idea/Design

The first idea for the robot's design would have been shaped in the frustum of a square pyramid, with one side having a scoop device capable of opening in order to pick up objects. The remaining three sides would be hollow cylinders, used to hold the sorted objects. The top of the robot would hold the sorting system, which will detect objects' color and shape and sort them accordingly. The robot will work to scoop rubber ducks and marshmallows. After successfully identifying and scooping an object, the scoop device would tilt backward, allowing the object to fall on top of the robot. Here, objects will be sorted based on color and shape, successfully completing the robot's task.

4.3 Robotic Arm Idea/Design

Another robot design idea included a claw that would pick up the objects and subsequently dump them into a bin or cylinder receptacle. In order to pick up and identify objects, a robotic arm and its camera would be used. This idea was selected to use for the robot's design.

4.4 Baseboard Construction

Throughout the project's lifetime, several baseboard designs were made. All baseboard designs were constructed using 3/8" aluminum pieces, which were cut using the waterjet. Three smaller baseboards and one larger baseboard were made. These designs are shown below.

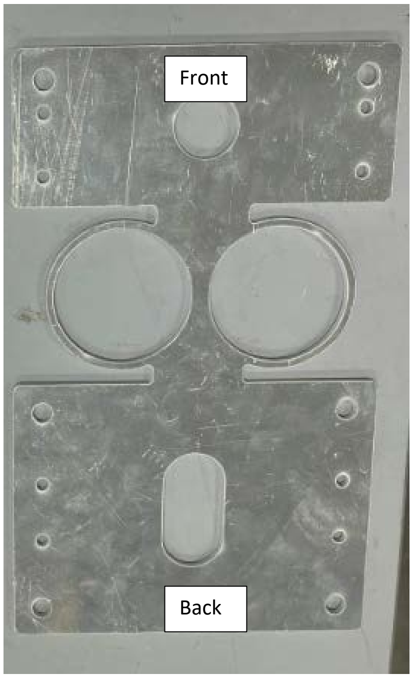

Figure 3: Baseboard Design One

Initially, the IEEE club focused on constructing two robots for the hardware competition. As a result, the first three baseboards were designed based around this idea. The first three baseboards were made with a smaller width to allow a second robot to fit within the course's starting area. The first baseboard, in Figure 3 above, was designed to be six inches in length by ten inches. The two center rings would be used as silos to hold the marshmallows for stacking. On the front side of the baseboard, a robotic arm would be placed to pick up the marshmallows. Holes were made on the front and back side of the board to allow wires to easily go from the bottom to the top of the robot.

Figure 4: Baseboard Design Two

Baseboard Two, in Figure 4 above, was designed to have a width of six inches and a length of eleven inches. The two center rings would be used as silos to hold the marshmallows for stacking. In this design the silos were faced on the same side, which is altered from Baseboard 1 above. On the front side of the baseboard a robotic arm would be placed to pick up the marshmallows, as in the previous design. Holes were then put on the front and back side of the board to allow wires to easily go from the bottom to the top of the robot. The main difference between Baseboard One and Two is the length, with Baseboard 1 being slightly smaller. This small difference would have helped to maximize space if two robots were used.

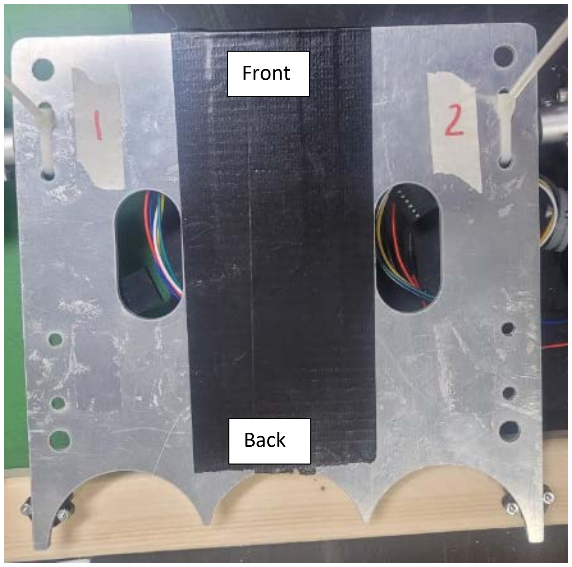

Figure 5: Baseboard Design Three

The third baseboard (Baseboard Design Three in Figure 5) was designed to have a length and width of eight inches. The back side of the board has three half rings cut in, to allow three silos to be attached. The three silos would have been used for holding three columns of marshmallows. A robotic arm would have then been mounted to front of the baseboard to pick up the marshmallows. The marshmallows would have then been placed into three neat stacks within the silos. This square baseboard was much smaller than the first two designs, allowing for more space for a second robot.

Commented [DF1]: These pics are awesome!

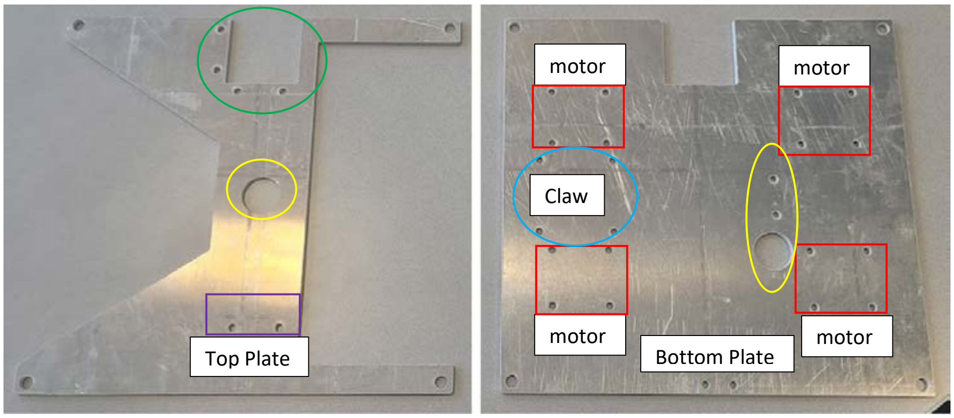

Figure 6: Baseboard Design Four

Midway through the project, the IEEE club began to focus on singular robot instead of the original two robots. This allowed for the full space of the starting zone to be allocated to a single robot. Baseboard Four was designed to be eleven inches square. The bottom and top plate were both designed in AutoCAD with specific locations for each component that would be attached to the robot. This CAD drawing can be found in Appendix C Figure 29 and Figure 30. In Figure 6, the specific locations at which attachments will be put on the bottom baseboard can be seen. The four motors would be attached to the undercarriage of the bottom baseplate with the use of four screw holes. These screw holes are shown within a red rectangle. A yellow oval is shown on both the top and bottom base plates, these ovals show holes which were made for wire management. The blue oval on the bottom base plate shows where the robotic arm would be mounted using four screw holes. The purple rectangle shows where the system power switch would be mounted. On the top base plate, several holes and cuts were made to account for some of the IEEE club's designs. The green oval shows where two servo motors and a chute would be placed. Since a singular robot was chosen to be used, Baseboard 4 was chosen for the final design.

5 Hardware

5.1 Motor Considerations/Selection

Figure 7: 12V DC High Speed 300RPM Gear Motor with Encoder for Arduino [2]

Encoded metal gear motors would be used to control the wheels of the robot. Two different motors were considered when designing the robot's navigation. Further information about these parts will be in the Navigation Considerations section.

The first set of motors are pictured above in Figure 7 with their approximate dimensions. These motors were purchased through Amazon but were manufactured by Bemonoc. When looking at the specifications of the motors, one can see that it runs on 12 V DC with a high speed of 300 RPM. The motors were considered because of their size, speed, and encoder capabilities. Previously, the IEEE club used larger motors that took up more space than needed. The Bemonoc motors are much smaller and help keep the robot below the height requirement of one foot. Their speed seems to be moderate and will allow the robot to travel around seven inches every wheel rotation. The encoders for the motors use channels for A and B, meaning that the motors can be controlled in a clockwise and counterclockwise movement.

Figure 8: FIT0186 Gearmotor 251 RPM 12 V with Encoder [3]

The second set of motors that were considered are pictured above in Figure 8. They were purchased from Digikey but were designed by DFROBOT. These units have a 0.61" long, 6 mm-diameter D- shaped output shaft. This motor is intended for use at 12V, though the motor can begin rotating at voltages as low as 1V. The face plate has six mounting holes evenly spaced around the outer edge threaded for M3 screws. These mounting holes form a regular hexagon with the centers of neighboring holes are 15.5 mm apart [3]. These motors are much larger than the those previously mentioned, causing the robot to have a much greater height. An additional difference between the motors is their speeds. These motors are slower when compared to the 300 RPM motors. However, the DFROBOT motors can handle much heavier robots without stalling.

When looking at how much weight would be attached to the robot, a decision was made to use the DFROBOT motors. Although these motors are slower the Bemonoc motors, they can handle the excess weight created by other subsystems.

5.2 Time of Flight Distance Sensor Considerations/Selection

A common theme with previous autonomous robots from other IEEE teams was difficulties in colliding with other robots and track walls. This was taken into consideration by the senior design team, in efforts to avoid this issue. To prevent “the claw” from colliding with different objects several Time-of-Flight Distance Sensors were implemented on its frame. A Time-of-Flight distance sensor uses the basic principle of measuring the distance between a sensor and a target. The distance is measured based on the time difference between the emission of a signal and its return to the sensor after being reflected by an object (target) [4].

Figure 9: Adafruit VL53L4CX Time of Flight Distance Sensor [5]

The claw will use four Adafruit VL53L4CX Time of Flight (Figure 9) distance. Each sensor will be located on a different side of the robot representing a different cardinal direction for each side. The VL534CX has a minimum sensing range of 1 mm and a maximum sensing range of 6000 mm. This particular sensor is great for large distances and easy to use with microcontrollers due to its 3-5V power logic. These Time of Flight sensors are great for determining the distance of the surface directly in front of it. This works perfectly when trying to determine the distance between the robot and the track walls.

Although the Adafruit VL53L4CX Time of Flight distance sensor had a lot of upsides, one major downside was that this sensor required a significant amount of flash memory that was not available on the current microcontroller. For this reason, it was decided to switch to the Pololu VL53L0X Time-of-Flight Distance Sensor Carrier with Voltage Regulator, 200cm Max and this can be seen in Figure 10, below.

Figure 10: VL53L0X Time-of-Flight Distance Sensor Carrier [6]

The VL53L0X (Figure 10) Time-of-Flight Sensor is very similar to the VL53L4CX Time-of-Flight sensor when it comes to libraries and the data sheet. This sensor has a maximum sensing range of 200cm, and it requires very little flash memory, which works with all the selected Arduino Micro-Controllers.

The original plan for the time robot was to install one Time-of-Flight sensor on each side of the robot to determine the direction of its movement, with one sensor for each cardinal direction. Later, it was decided to add two Time-of-Flight sensors to the left side of the robot and two to the front. By installing multiple sensors on the same side, the robot's alignment could be ensured, and it could travel in a straight line instead of diagonally. Having two sensors on the front and two on the side would provide two different ways to ensure the robot's alignment in case one set of sensors was obstructed. The subsequent plan was to mount one sensor on the front and one on the side of the robot. The front sensor would detect any impending collisions with objects or walls, while the side sensor would square the robot by taking a measurement, having the robot move forward a short distance, and then taking another measurement. The robot would compare these distances and adjust its direction accordingly.

These sensors have the same digital address, and they need to be connected to the SDA and SCL pins of the Arduino board that are being using. Since that are using the same pins and have the same address, an additional microcontroller is needed for each additional sensor. This means that having 4 sensors would require 4 microcontrollers. This would take up a lot or space and the more board that are used the more of an issue timing would be, so if would be better to use less boards in a more effective manner rather than more in a similar way.

5.3 Battery Considerations/Selection

Voltage and current were a big concern when choosing a battery. The motors as well as the robotic arm ran off a max of 12 volts while the rest of the systems like the Arduino uno and the time-of-flight sensors. The plan originally consisted of getting a 12-volt battery to power half of the robot and a voltage regulator to down step the voltage to 5 volts to power the rest of the robot. Two considerations changed this. The motors are technically 12-volt motors, but they can be run on a lesser voltage with no drawbacks. The other consideration is that the battery can supply the Arduino uno or mega with 12 volts to power itself, and it can automatically provide a 5-volt pin that can be connected to a rail on a breadboard to power the rest of the robot.

The battery that was selected in the end was the Turnigy 2200mAh 3S LiPo Battery Pack 11.1V (Figure 11). This battery was also used in previous years robots with success, so it is familiar to the club members.

Figure 11: Turnigy 2200mAH Battery Pack [7]

This battery has 2200 mAh of life, so running out of charge during a test run or the middle of the competition should be of no concern.

Using two of these batteries was also briefly considered since this robot has to many subsystems that require so much current. In the end it was believed that one battery could be used to power the entire robot with the robot arm being powered by a separate battery with its own fuse. The team was never able to power the entire robot at once so this was never fully implemented.

5.4 Wheel Considerations/Selection

One of the most important parts for any robot that deals with navigation is wheels. Regular wheels could have been selected, but omni wheels (Figure 12) were selected. The benefit of these omni wheels is increased maneuverability, and more options when it comes to movement. The omni wheels have the option to drive horizontally in addition to forward and backwards. Omni wheels can also have a zero turning radius which allows for tighter turns with less of a chance of slipping what can happen with regular wheels. When parts for this robot were being researched, these wheels were selected because they provided the most options when it came to navigation. This was also at the time when navigation was not fully designed, and omni wheels didn't limit navigation.

Figure 12: Mecanum Omni Wheels [8]

5.5 Motor Driver

Since motors with encoders are being used, a motor driver is needed to control the signal of the motor and the encoder count. The TB6612FNG Dual Motor Driver Carrier was selected as motor driver. This can be seen below in Figure 13.

Figure 13: TB6612FNG Dual Motor Driver Carrier [9]

The one of the benefits of this motor driver is that it can be used to control 2 motors instead of one. This simplifies the circuit slightly because only 2 dual motor drivers have to be used instead of 4 individual motor drivers. The recommended motor voltage for this h-bridge motor driver is from 4.5 volts to 13.5 volts which is compatible with the selected motors. The logic voltage for this motor driver is from 2.7 volts to 5.5 volts which is compatible with the 5 volt output of the Arduino uno's 5 volt output pin. This motor drive has also been used by previous USI teams for the IEEE competition.

5.6 Micro Controllers

The robot's system required several micro controllers. The most user-friendly to the group is the Arduino Integrated Development Environment (IDE) because of the wide range of example codes.

5.6.1 Arduino Uno

Figure 14: Arduino Uno Rev 3 [10]

Above in Figure 14, is the Arduino Uno Rev 3. Two of these controllers were used in tandem with two motor drivers to control four 12 V DC motors. The Arduino Uno Rev 3 has an operating voltage of 5 V, 6 PWM pins, 2 hardware interrupt pins, and an operating current of 0.5 A. Of the 6 PWM pins, two are required for the motor control being used. The two hardware interrupt pins would prove helpful if encoder counts were to be used for navigation.

5.6.2 Arduino Mega

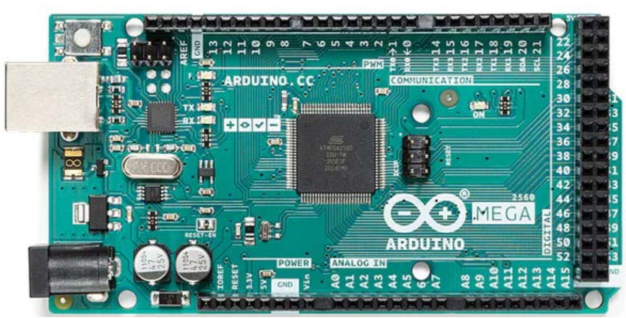

Figure 15: Arduino Mega 2560 Rev3 [11]

Above in Figure 15 is the Arduino Mega 2560 Rev3. This larger board is acting as the brain of the robot. One reason for this is that is has an excess of pins that can be used for communication between the other micro controller. The Arduino Mega communicates with each

Arduino UNO using 2-bit communication. This larger board needs 4 pins to communicate with the 2 Arduino Uno's. The Time-of-flight sensors also need a board with more flash memory and the Arduino Mega is compatible with the Time-of-Flight sensors. The Arduino Mega has an input voltage of between 7 and 12 volts with is compatible with the battery that was selected.

5.6.3 Jetson Nano

Figure 16: Jetson Nano Developer Kit [12]

Above in Figure 16 is the Jetson Nano micro controller. The reason that this board was selected was because it was compatible with DOFBOT robotic claw. This micro controller also has multiple USB ports that can be used for Serial Communication with the Arduino Mega. The Jetson Nano is also a powerful computer that can be used to run multiple neural networks in parallel. The robotic claw is using a neural network for its vision system and this microcomputer is very capable of handling that.

5.8 Voltage Regulator

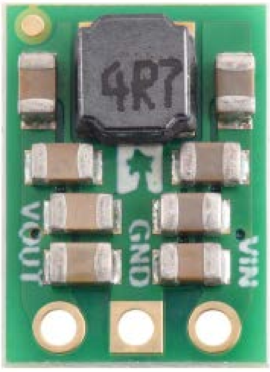

Figure 17: 5V, 1.5A Step-Up/Step-Down Voltage Regulator S13V15F5 [13]

As mentioned before, an 11.1 V rechargeable LiPo battery would be used to power the robot's electrical system. Several pieces of hardware required 5 V as their operating voltages. To combat this problem, research was done into several voltage regulators that would step down voltage. When looking at the Pololu website, several alternatives were found to step down voltage. The main concern was to find a regulator that meant the navigation's current and voltage needs. Within the S13VxF5 family there are four regulators that all step-down voltage to 5 V, however they all have different output currents. For the navigation's electrical system, a total of three amps would be required for operation: 0.5 A for each Arduino Uno and 1 A for each motor driver. The S13V15F5 voltage regulator (Figure 17) can step up or step-down voltage for inputs between 2.8 V and 22 V. The regulator also can have an output current of 1.5 A. Although, this is not 3 A, two of these would be used to be added up to 3A. One S13V15F5 voltage regulator would be paired with an Arduino Uno and motor driver, meaning that two regulators would be required for the navigation system.

5.9 Robotic Claw Consideration

Since grabbing and storing objects is such a large part of this competition the DOFBOT Robotic Arm was selected for this system, and this can be seen below in Figure 18.

Figure 18:DOFBOT Robotic Arm [14]

The DOFBOT Robotic Arm is highly customizable with 6 different axes of movement. This allows for very precise movement and allows the arm to grab, move, and sort objects with a high level of precision and accuracy. The robotic arm or claw has several libraries and examples that are helpful for learning about the claw and all the different options that are available to us. Different claws can also be 3D printed and attached to the robot depending on the application. For this robot, a bigger claw was printed and mounted.

This DOFBOT Robotic Arm needs a microcomputer to run. One great example of a microcomputer that is compatible is the Jetson Nano microcomputer. This Jetson Nano microcomputer is an advanced microcomputer that can process AI algorithms. One of the IEEE student club members made a neural network for artificial intelligence. Several hundred photos are taken and uploaded to the Jetson Nano for the AI to recognize the images and sort them accordingly. This claw is not a part of the senior design, but it is a part of the robot, and it is necessary to mention it in this report.

6 Robot Navigation

Mapping and navigation are crucial parts of this design. A perfectly working robot with no navigation will receive zero points at the competition.

6.1 Path of Navigation

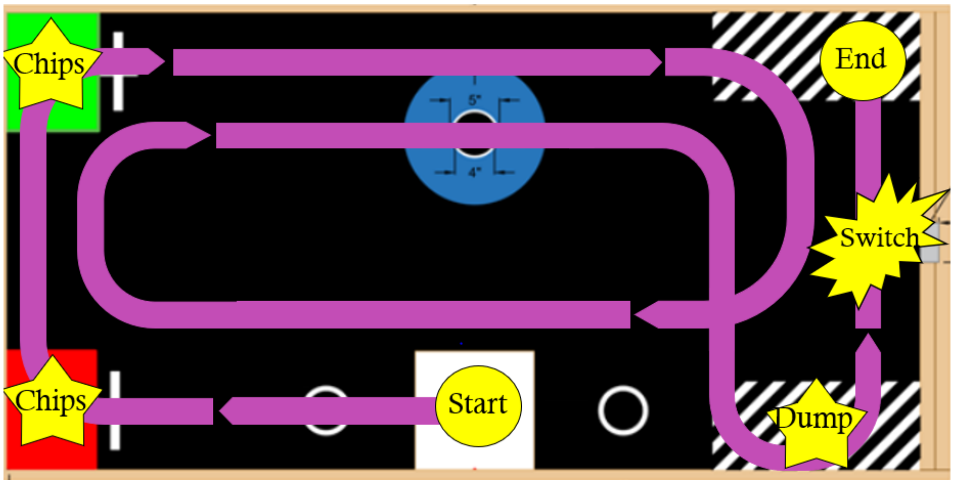

Figure 19: Path of Navigation

Before deciding on how to ensure the robot would move around the course, an idea was conceived on how the robot should navigate through the course. This idea is shown in Figure 19 above. The robot would start moving toward the manatee and alligator attractions shown as the red and green rectangles. The attractions are where chips could be dumped which increases the overall run score by seven points per chip of the correct color. After dumping the chips, the robot would start sweeping across the arena, moving marshmallows and rubber ducks in the process. After finishing several sweeps, the robot would dump the objects into the bottom right hazard zone and run into the firework switch, ending the run.

6.2 Encoder Based Movement

The first approach to navigation was an encoder-based movement system. Encoders inherently monitor the displacement or position but can be used to measure linear distance by calculating the number of pulses compared to the known number of pulses per arc length and designed into a system that returns linear feedback [15]. In short, the pulses shown from an encoder can be used to calculate linear distances. The below equations can be used to calculate distance from encoder pulses.

$$\textbf{Eq 1: } PPI = \frac{PPR}{2\pi r}$$

$$\textbf{Eq 2: } L = \frac{pulses}{PPI}$$

Equation 1 shows the formula for solving the pulse per inch (PPI), where PPR represents the pulses per one revolution and r being the radius of the wheel attached to the encoder shaft. Generally, the PPR is found in motor data sheets provided by the seller of the motor. Equation 2 shows the formula for length (L), where the pulses are the appropriate counts seen by the encoder. In order to select the distance at which a robot will move, Equations 1 and 2 can be used to solve for the pulses with the length and PPI. This pulse count can act as a conditional: a cut-off for when vehicles should brake. The main problem with encoders is trying to maintain their accuracy. The most important aspect of measuring length with a measuring wheel encoder is avoiding slippage. This is a problem when using omni wheels and can make things very difficult when trying to sink four wheels to stop at the same time. As mentioned previously, omni wheels were selected for the robot. This selection led to later difficulties (encoder inaccuracy) ruling out encoder counts for robot navigation.

6.3 Time Based Movement

After ruling out encoder movement, a more simplistic approach was taken to solve robot movement. A timer and time-of-flight based system were developed to control robot movement. This system is shown below in Figure 20.

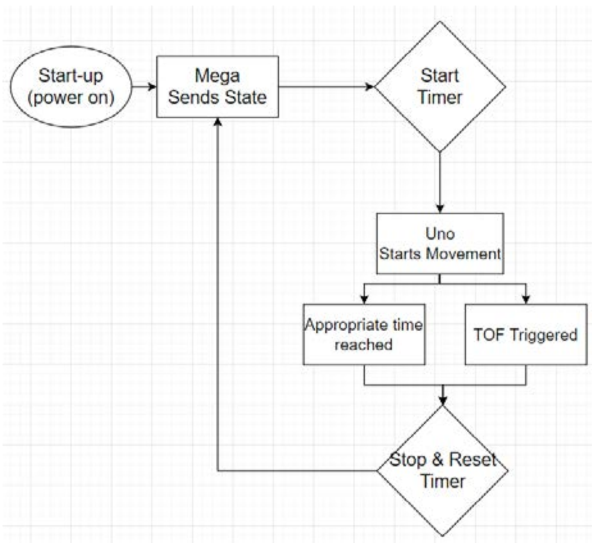

Figure 20: Time-Based Navigation Flow Chart

In theory, the system would be powered up and an Arduino Mega would send a signal to the two Arduino Uno's to start movement. Simultaneously, the Mega would start a timer with a cutoff for when the robot needs to brake. If the appropriate time is reached, then the timer stops and resets with the Mega sending another signal for the next movement. However, if the time-of-flight sensor saw that the robot is at its appropriate distance then the timer would also be stopped and reset. This would then initiate the next movement within the Mega.

6.4 Navigation Electrical System

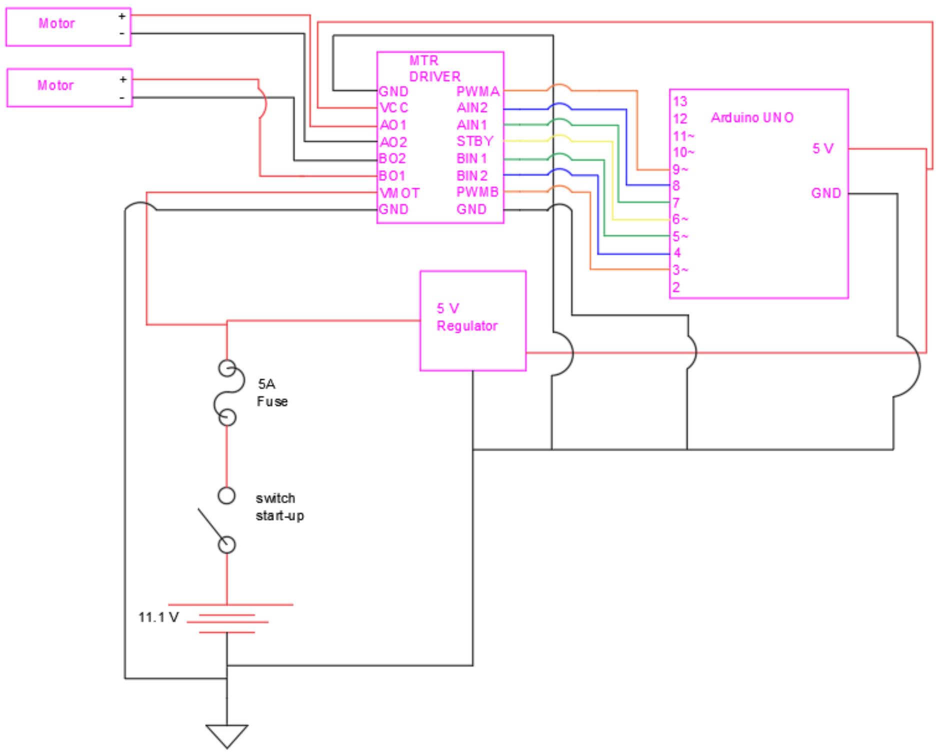

Figure 21: Two Motor Electrical Schematic

Above in Figure 21 is schematic for a two-motor electrical system. This system is used twice within the robot because it is a four-motor system. In the schematic, all red wires are positive voltages, and all black wires are considered ground. The system starts off with its 11.1V LiPo battery and a power switch. A 5 A fuse was then put at the beginning of system to protect the hardware being used. A 5 A fuse was selected due to the system needing a total of 3 A to operate. The IEEE team would also be using the same battery for their design, so a 5 A fuse was put in instead of 3A to allow current to reach the club's devices. This 3 A system is divided amongst the two Arduino controllers and the two motor drivers. After, the fuse the 11.1 V goes into two different connections, the motor driver's V_{mot} pin and the voltage regulator's V_{in} pin. The regulator then steps down the 11.1 V to 5V. These 5 V connections are then run into the Arduino Uno 5 V pin and the motor driver's V_{cc} pin. These connections establish power to the motors, motor driver, and Arduino Uno. When looking at the Arduino Uno's connections, the most important is the connections made to pins 9 and 3. These pins are pulse width modulation (PWM) signified by the ~ symbol. The motor drivers PWMA and PWMB pins must be attached to PWM pins of the Arduino in order for the motors to be controlled. The 12 V motors in the top right are attached to the motor driver.

6.5 Scooping Considerations

For the scooping component of the robot, the DOFBOT robotic arm was selected. This arm has various upsides like it has 6 different axes of rotation for very precise grabbing angles, it comes with lots of examples and libraries that can be used for troubleshooting problems, and it is combatable with the Jetson Nano, a very advance microcomputer that can be used for advanced algorithms.

Before the claw was decided on there were a few previous ideas for scooping. One previous idea that was considered was a servo scoop. This idea would consist of two half bucket shaped pieces that would find and close around the object. A servo motor would then scoop the object to the top of the robot in a dump truck motion. The object would then be sorted based on color.

Another idea that was thought of was a claw on a track similar to the claw in an arcade machine. This idea was abandoned due to difficulty implementing a track that was over a foot long as well as difficulty in installing a claw that can be raised and lowered onto the track.

The final idea that was considered was an origami grabber. During research into scoops and grabbers, an origami claw was discovered that seemed possible. This claw would be made out of a flexible material that could be 3-D printed and reinforced with a more rigid material. An appointment with Dr. Nelson was made to discuss this idea and the professor even showed examples of it working. The idea of using a servo motor to open and close the claw seemed possible but this idea was also abandoned.

6.6 Sorting and Stacking Considerations

Every baseboard design that was fabricated and designed had the idea of on-board storage in mind. The initial designs involved a silo like device with a funnel on top to stack the marshmallow. One idea consisted of collecting three marshmallows and afterward, opening the silo and driving away to leave a stack of stacked marshmallows. Most of the baseboard designs include this or a similar idea.

Later in designing the robot, the strategy changed to involve dumping any objects that were collected rather than stacking them. This changed the design by having a large cubby with a door that is controlled by a servo motor that would dump out these objects into the hazard zone. This would be the final design that was selected due to its simplicity. This idea would also allow the robot to only focus on storing the objects rather than sorting the objects.

7 Robot Construction

Construction of the robot is an important part of the overall robot. Poor construction will not allow the robot to stay together in one piece or perform in its intended way. A reliable construction includes secure connections of screws and bolts, pieces mounted in the correct orientation, and materials placed to maximize their efficiency.

7.1 Motor Placement and Attachment

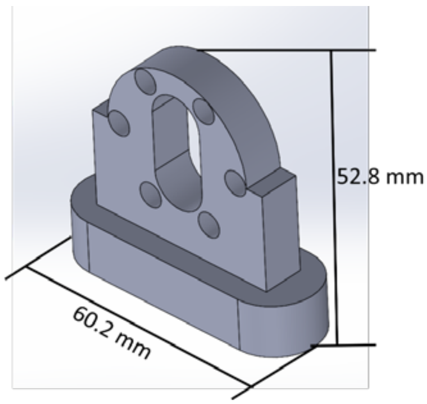

Figure 22: Motor Mount

Due to the large size and importance of the motors, secure motor mounts are needed. Each motor was secured by a motor mount that was 3D printed. The design of these motor mounts was based off the motor mount from a previous year's robot. Some sagging was noticed in the motor, meaning that the motors were not perfectly level. Out of fear that this would affect the motors and the robot's performance, a zip tie was added to the opposite end of the motor for additional support. The motor mount (Figure 22) is secured to the motor by six half inch bolts that were provided with the motors when the were purchased. Four sets of four holes were drilled into the bottom of baseplate 4 [6]. Each set of 4 holes is in a square pattern with 1.5 inches between the center of the holes. Each set of 4 holes is also designed to secure one motor. The two holes closest to the side of the robot are for one-inch-long screws to secure the motor mount to the baseboard. The two other holes closer to the center of the robot were originally made for a second motor mount but now these two holes are being used by the zip tie to add additional stability to the motor.

7.2 Baseboard Attachment

Baseboard design 4 shown above in Figure 6, has 2 levels that need to be securely attached to each other. In order to secure these baseplates to each other, the first 4 holes with a diameter of 0.3125 inches were machines into each corner of each baseplate. Next, a 4-foot thread all with a diameter of ¼ inch was purchased and cut into 4 1-foot sections. A thread-all was put through each corner and secured with a lock nut on top and on the bottom of the baseplate. The high of this could be adjusted later depending on the needs. A nut was then screwed onto the thread-all approximately 6 inches above the bottom baseplate. The top level could then be placed on top of these nut and secured as needed with another nut on top of the second level baseplate.

7.3 Claw Attachment

For the claw to be attached to the baseboard 4 holes were drilled into the baseboard in a square pattern with the holes being 2 inches apart from each other. These holes with dimensions can be seen above in Figure 6. The robotic class is then removed from its own baseboard by removing half inch bolts from the bottom of the claw. The robotic arm is then placed above the 4 drilled holes with 2 inch spacing and the half inch screws are reinstalled on the claw, this time with the robot's baseboard instead of the claws included baseboard. A washer is added to each screw above the baseplate for added security and to help with spacing.

7.4 Overall Placement

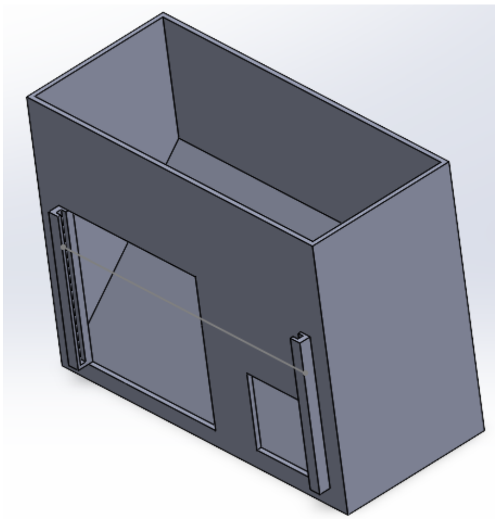

Figure 23: Storage Bin

There are other subsystems such as the color sensor that are also attached to this robot. Shown below in Figure 31, there is a 2-inch by 2-inch square that was cut out of both the top and bottom layer of baseplate 4. This slot is designed for a chip chute. Due to limited space on the bottom layer the servo motors designed to dump the chips had to be mounted to the second layer. This was done by 3D printing a mount for the servo motor and attaching this mount to the baseplate using a half inch bolt and nut. Whenever the chip servo was mounted to the second later and the chips were dumped, they would bounce and react unpredictably. This is where the chip chute comes in. The chip chute serves the purpose of slowing down the chips and directing the chips in a more reliable way. A color sensor is also needed for this subsystem and this color sensor was mounted to the bottom of the chip chute. Thanks to the customizability of 3D printing as well as experienced IEEE student members this was designed and printed with no difficulty.

Seen above in Figure 6, there is a 9-inch by 4-inch slot cut out of the top layer. The purpose of this slot is to hold out storage device (Figure 23) for the marshmallows as well as rubber ducks. A hollow rectangular prism measuring 4 inches deep, 9 inches wide and 6 inches tall was 3D printed to serve as the storage device. This rectangular prism has a hole 8inch wide and a 4-inch-tall rectangular hole in the side to act as an exit for the objects to fall out of. The bottom of this shape has a 45-degree slope towards the exit to allow objects to fall out easier. This shape also has no top to allow for the robotic claw to place objects in the storage device. The storage device fits very snuggly in the 9-inch by 4-inch slot for the robot. This secure fit would then be reinforced with superglue so that this connection is permanent.

Microcomputers such as the Arduino Mega, and Jetson Nano will be mounted in the middle of the robot between both layers, and behind the robotic arm. Each microcomputer has a 3D printed mount for it that is that is secured to the baseboard using Velcro straps. The Arduino Uno's, along with a breadboard that is being used for motor drivers are placed in a hollowed out rectangular prism that is mounted to the bottom of the robot. The reason for this bottom placement is due to limited space in the middle of the robot, and extra space being available underneath the robot. This spacing can be seen below in Figure 29.

8 Testing/ Integration

8.1 Characterization of Motors

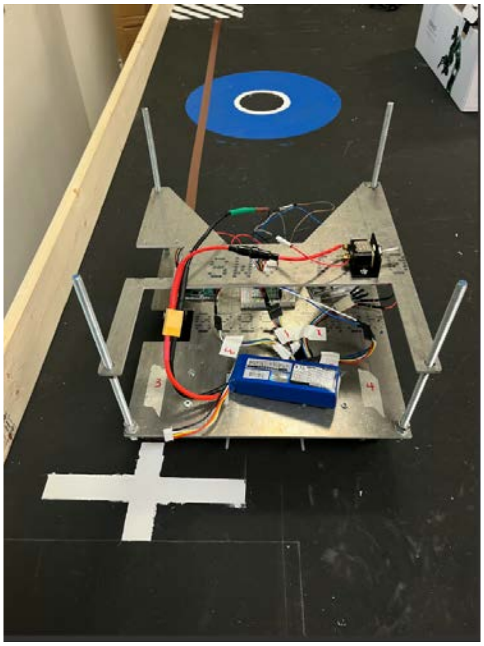

Figure 2424: Motor Strength Testing

Before implementing the time-based movement, motor strength testing was conducted. Although all electrical motors used for the robot had the same specifications, some tended to be stronger than others leading to drifting. This testing was done to make sure the robot could move in a straight line and is shown above in Figure 24. To do this testing, a long piece of electrical tape was applied to the course in a straight line. For each test run, the time of when the robot strayed from the straight line was recorded to see if a positive progression was made. The strength each of motor was recorded for each run. This data is shown below in Table 2.

Test	Motor 1 Strength	Motor 2 Strength	Motor 3 Strength	Motor 4 Strength	Time of Displacement (seconds)
1	**90**	**90**	**90**	**90**	**2.4**
2	**95**	**90**	**95**	**90**	**3.6**
3	**97**	**90**	**97**	**90**	**4.4**
4	**99**	**90**	**99**	**90**	**6.8**
5	**101**	**90**	**101**	**90**	**7.3**
6	**102**	**90**	**101**	**90**	**7.8**
7	**103**	**90**	**101**	**90**	**9.2**
8	**104**	**90**	**101**	**90**	**12.2**
9	**105**	**90**	**101**	**90**	**14.1**

Table 2: Motor Strength Testing Data

For the first test, a baseline of 90 was established for all four motors and it was seen that motors two and four were much stronger than motors one and three. To compensate for this, the strengths of motors one and three were increased slowly until no more positive progression was seen. Halfway through the test runs it was observed that motor three was slightly stronger than motor one. To compensate for this motor, motor one was increased while keeping the other motors at their respective strengths. On the final test run, it was determined that the motor strengths were as follows: Motor 1 Strength is 105, Motor 2 Strength is 90, Motor 3 Strength is 101, and Motor 4 Strength is 90. These values would allow the robot to go in a straight for a little over fourteen seconds, which is more than enough time for the robot to go from end to end of the competition arena.

8.2 Time of Flight Sensors

The first stage of testing the time-of-flight sensors involved opening an example code provided in the VL53L0X library in Arduino. The basis of how the sensors worked and their capabilities were learned through this code. The code had a serial print statement that provided the distance read by the sensor every half second. The next step was to test the accuracy of the sensors. Several tests were conducted, which involved setting up a ruler at the sensor's base and using a foot-long 2 by 4 painted black. The piece of wood was moved back and forth, and readings were taken multiple times to collect as much data as possible. It was found that the sensor would give an incorrect reading if an object was closer than approximately 5 cm to the sensor. This distance was taken as the minimum distance. According to the sensor, the maximum distance reading is 200 cm, but it was discovered that the sensor would act unpredictably after 100 cm, which was acceptable for the intended purpose.

8.3 Navigation States

The robot's navigation can be broken down into four different states. These states are strafe, forward, turn right and brake. Strafe is the command that is being used initially for the robot to move away from the boarder wall, and since omni wheels are being used strafing is a valid movement pattern. This strafe command is being used to move the robot to the right. The forward command drives all 4 motors forward at a constant speed. This robot is only designed to turn right, and the right command turns the robot slowly to the right by having the right 2 wheels drive backwards at the same speed that the left two wheels are driving forward. The brake command stops all movement of the motors. Using this command also minimizes wear and tear on the motors especially after rigorous testing. Using these four commands in the correct pattern will allow this robot to complete its navigation.

8.4 Communication Control

The way that navigation was planned was that the Arduino Mega would send a 2-bit word to each Arduino Uno and these 2-bit words related to a specific command for the Arduino Uno. 00 correlates to a strafe. 01 correlates to driving forward. 10 correlates to turning right. Lastly 11 correlates to braking. The Arduino Mega has 2 pins to send each Arduino Uno to communicate these 2-bit word. The Arduino Uno receives this 2-bit word and does the relevant navigation based on the word. The Arduino Mega also has a timer function that can be paused, resumed, and reset. Whenever the Mega reaches its timer limit, it will pause the timer, reset the timer, send the next navigation command, and then resume the timer. This process would repeat itself as many times as it needs to until navigation is completed.

The Jetson Nano is also communicating with the Arduino Mega, this time through a serial port using a 1-bit word. If the Jetson Nano's camera sees an object that it wants to grab, it will send a 1 to the Arduino Mega. This 1 that was sent to the Arduino Mega acts as an interrupt to the Navigation. When the Arduino Mega receives this 1, it will pause the timer, and brake the motors until the Jetson Nano sends a 0 to the Arduino Mega. Once the Arduino Mega receives a 0, the timer will resume, and the navigation will continue. The Jetson Nano sending a 0, acts as an all clear to the Arduino Mega, and the Mega will continue its navigation as normal. The Jetson Nano will continuously send a 1 or a 0 to the Arduino Mega if it has power.

9 Results

9.1 Competition Placing

The claw robot placed 14th place out of 29 competing teams. With a total of three runs the robot accumulated an 42 points overall. Points were awarded for dumping the green chips, turning on, and moving off the starting square. For each run, 10 points were awarded if the robot moved off the starting square within 3 seconds. The claw moved out of the starting square within three seconds in two of the three runs. Seven points were awarded for each chip dumped in its corresponding color. On the robot's second run, it was able to reach the green square and dump two chips into the zone. Four points were awarded for the robot turning on. The robot turned on two out of three times.

9.2 What Went Right

For the competition, three main designs worked well including overall communication, chip sorting, and the time-of-flight sensors. Communication was established between an Arduino Mega and two Arduino Uno's. Communication was also established between the robot arm and the Arduino Mega. These two aspects were crucial for system integration because the arm tells the mega when an item is seen and the Mega tells the Uno's when to stop navigation. A time-of-flight sensor was hooked up to the Arduino Mega and was able to read the distances between the robot and the course walls. With this being successful, the robot had more information on how to move around the course. The last aspect that performed well was the IEEE club's idea on how to dump the feeding chips. Their system used to servo motors and a singular color sensor. One servo motor was for the green chips and the other was for the red chips. The color sensor would tell the servo motors which color they were above and dump the corresponding chips to their respective zone.

9.3 What Went Wrong / How to Improve

The four main issues with the robot were the electrical system, limited physical space, a non-modular design, and system integration. The main problem with the electrical system was that voltage connections were not labeled in the best way. This resulted in wrong connections being made. During the competition, an 11.1 V connection was put into a 5 V rail which resulted in an Arduino Uno being overloaded. During this process, a motor was also fried which caused several problems for the competition. As mentioned previously, motor strength testing was done for four motors but not a backup motor. When attaching a backup motor in place of the fried motor, the robot was no longer able to stay on a straight path in the competition. To stop this from happening with future competitions, electrical systems should be fully colored coded and labeled with their appropriate voltages. The next two problems go hand in hand with each other. Not making a modular design meant that things were crammed together resulting in limited space. To fix this, baseboards should be made to have several small screw holes throughout, allowing component pieces to be placed in any location. This would allow for components to be swapped in and out to meet the task at hand. The final problem occurred due to poor scheduling amongst the group. System integration proved to be time-consuming and is a difficult task that takes a great deal of testing. The group did not allow enough time for testing to take place to integrate the jetson nano, the Arduino Mega, and the Arduino Uno’s. At the competition, the jetson nano was able to communicate with the Arduino Mega and the Mega was able to communicate with the Uno’s. However, it was never proven to work with all three controllers. The group should have scheduled a large amount of time for system integration to occur.

10 Conclusion

The robot for the IEEE Southeast Con 2023 student hardware competition was challenging. During the competition, the team faced multiple challenges leading up to/during the competition. This is due to the frequent designing, tweaking, and removal of various ideas and developments in order to create the ideal and most efficient robot. This robot used serial communication, an AI neural network, color sensors, a timer, time-of-flight distance sensor, and a giant robotic arm. These components individually are complex, making it difficult to integrate and implement them all into one cohesive unit that required a multitude of testing. Some of these components worked well with others such as the communication, the sensors, and the chip system. Some things did not go well with the robot such as the electrical system, and the very limited design. That is how these competitions go. One day everything is working, and the next half of your files get corrupted. The navigation from this robot will also provide a good starting point for future years robots.

10.1 Disposal Program

The disposal program for this robot is to be handed down to next year's IEEE student club to be used in any capacity that it can serve. This robot could be used as a base for next year's IEEE robot, it could be used for spare parts, or it could be placed in a display case. This robot is not planned to be thrown away, but it can be recycled or incorporated into another robot.

10.2 Future Works

As mentioned in the Disposal Program section, all senior design work will be given to the IEEE club. In addition to the physical robot and its components, the senior design group will be passing a comprehensive folder to the IEEE club, which will include all navigation code, parts ordering list, and design schematics. These materials should prove to be helpful resources for future IEEE club teams at USI. This senior design group hopes that the IEEE club will build upon the work left behind. More specifically, the senior design group hopes that materials passed down will help to further develop the time-based navigation system. Future groups should be able to develop better systems with encoder counts for competition. This would help future groups to solve slippage problems mentioned in the Encoder Movement section. Finally, the navigation system should give future teams a good starting point, preventing them from having to build one from the ground up.

References

[1] IEEE, "IEEE SoutheastCon 2023 Student Competitions," IEEE, 1 November 2023. [Online]. Available: https://ieeesoutheastcon.org/student-competitions/. [Accessed 5 December 2023].

[2] Amazon, "Industrial & Scientific-Industrial Electrical-Passive Components-Switch Encoders," Bemonoc, 2023. [Online]. Available: https://www.amazon.com/Encoder-Gearmotor-300RPM-Arduino-Printers/dp/B07GNGCNK1/ref=asc_df_B07GNGCNK1/?tag=hyprod-20&linkCode=df0&hvadid=309803885710&hvpos=&hvnetw=g&hvrand=10726019617567647468&hvpone=&hvptwo=&hvqmt=&hvdev=m&hvdvcmdl=&hvlocint=&hvlocphy=90. [Accessed 8 February 2023].

[3] DFROBOT, "12V DC Motor 251rpm w/Encoder (SKU: FIT0186)," Digikey , 2017. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/FIT0186_Web.pdf. [Accessed 19 March 2023].

[4] Terabee, "Time-of-Flight principle," Terabee, 2023. [Online]. Available: https://www.terabee.com/time-of-flight-principle/#:~:text=Time%2Dof%2Dflight%20(ToF)%20sensors%20are%20used%20for,the%20distance%20between%20the%20points.. [Accessed 10 March 2023].

[5] Adafruit, "Sensors / Proximity / Light & LIDAR / Adafruit VL53L4CX Time of Flight Distance Sensor - ~1 to 6000mm," Adafruit, 2023. [Online]. Available: https://www.adafruit.com/product/5425. [Accessed 16 February 2023].

[6] Pololu Corporation, "Pololu," Pololu Corporation, 2023. [Online]. Available: https://www.pololu.com/product/2490. [Accessed 22 March 2023].

[7] "HobbyKing.com," 2023 HOBBYKING, [Online]. Available: https://hobbyking.com/en_us/turnigy-2200mah-3s-25c-lipo-pack.html. [Accessed 25 April 2023].

[8] Youyeetoo, "Youyeetoo," Youyeetoo Geek Shop, 2023. [Online]. Available: https://youyeetoo.com/products/omni-wheel-4ps-60mm-a-set-of-4-mecanum-wheels-60mm?VariantsId=10399. [Accessed 15 3 2023].

[9] Pololu, "Pololu Robotics & Electronics," Pololu Corporation, 2023. [Online]. Available: https://www.pololu.com/product/713. [Accessed 15 March 2023].

[10] Arduino, "Arduino Uno Rev3," Arduino, 24 March 2023. [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3. [Accessed 10 January 2023].

[11] Arduino, "arduino-mega-2560-rev3," Arduino, 2021. [Online]. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3. [Accessed 25 March 2023].

[12] nvidia, "Jetson Nano Developer Kit," nvidia, 2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Accessed 25 February 2023].

[13] Pololu Corporation, "5V, 1.5A Step-Up/Step-Down Voltage Regulator S13V15F5," Pololu Corporation, 2023. [Online]. Available: https://www.pololu.com/product/4084. [Accessed 15 March 2023].

[14] Yahboom, "Yahboom," 2023 Yahboom, 2023. [Online]. Available: https://category.yahboom.net/products/dofbot-jetson_nano. [Accessed 13 March 2023].

[15] Dynapar, "How to Measure Linear Distance with Rotary Encoders," Dynapar, 2023. [Online]. Available: https://www.dynapar.com/knowledge/applications/encoder-linear-measurement/. [Accessed 15 February 2023].

[16] DFROBOT, "Fermion: TCS3200 RGB Color Sensor," DFROBOT, [Online]. Available: https://www.dfrobot.com/product-540.html. [Accessed 22 October 2022].

[17] DFROBOT, "DFRobot SEN0101 TCS3200 Color Sensor/Color Detector," AMAZON, 2022. [Online]. Available: https://www.amazon.com/DFRobot-SEN0101-TCS3200-Sensor-Detector/dp/B009MK24JE/ref=sr_1_1?crid=R8NZDVF7Z04A&keywords=SEN0101&qid=1669074779&sprefix=sen0101%2Caps%2C104&sr=8-1. [Accessed 14 December 2022].

Appendix A

Bill of Materials

Bill of Materials		
Product	**Price**	**Quantity**
Bemonoc 12 V Encoded Motors	$ 14.88	4
VL53L0X Time of Flight Distance Sensor	$ 14.95	1
Turnigy 2200mAh 3S 40C Lipo Pack	$ 28.00	2
DOFBOT AI Vision Robotic Arm	$ 389.00	1
Mecanum wheels 60mm	$ 59.00	1
Arduino Uno	$ 27.60	2
Arduino Mega	$ 48.40	1
Dual Motor Driver Carrier	$ 9.95	2
Jetson Nano	$ 149.00	1
5V, 1.5A Step-Up/Step-Down Voltage Regulator	$ 8.95	2
Total Price		$ 868.87

Figure 2525: Bill of Materials

Appendix B

Shown below in Figure 26 is the system architecture. This Figure shows how all the subsystems are connected.

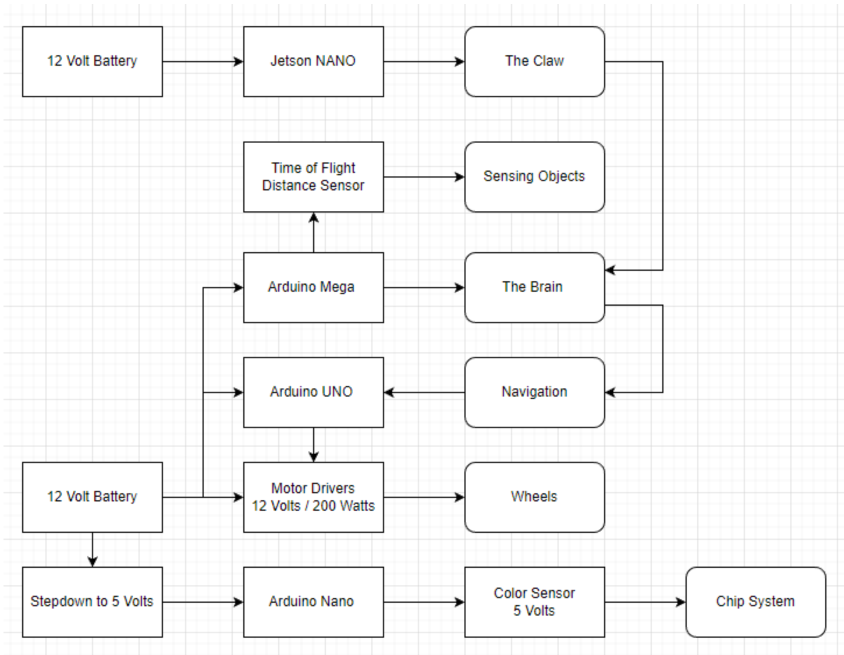

Figure 2626: System Architecture

Appendix C

Below, one can find the AutoCAD drawings for each baseboard and the physical design for the chip and chute.

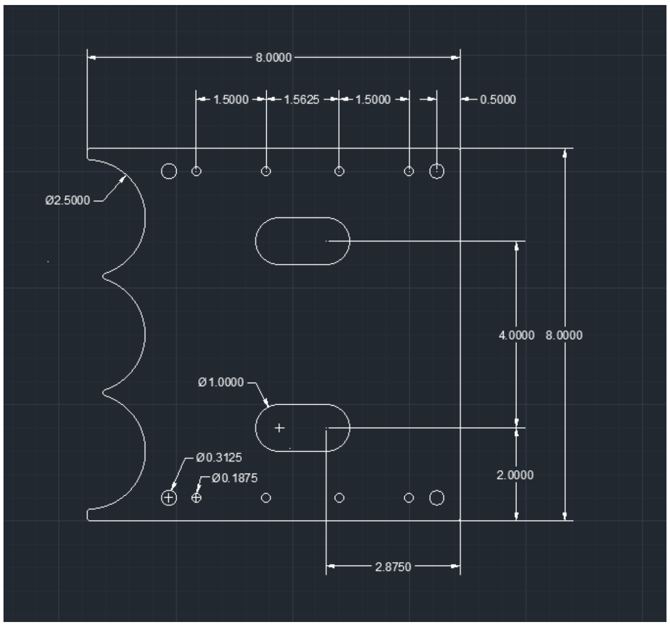

Figure 2727: Baseboard 3 Design with Dimensions

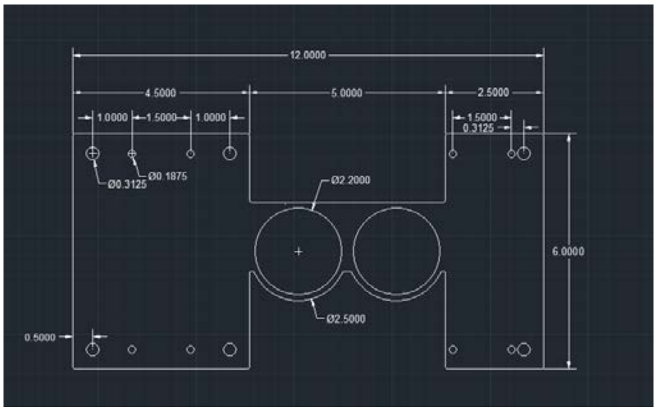

Figure 2828:Baseboard 2 Design with Dimensions

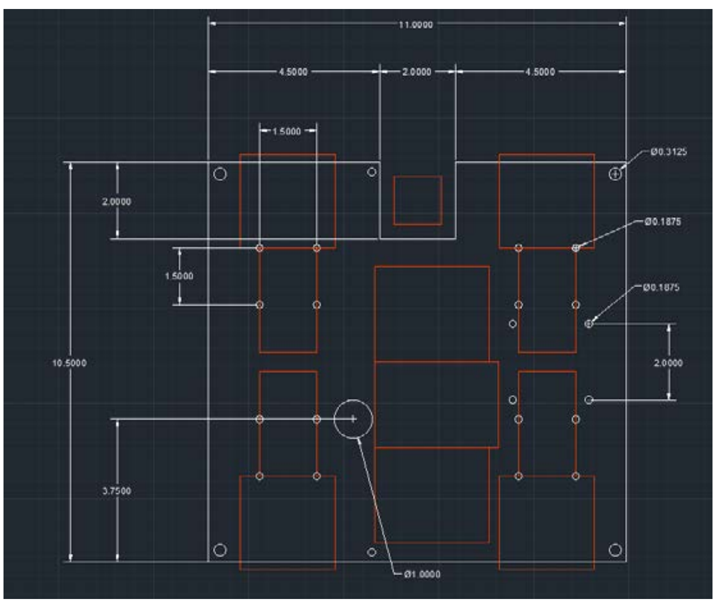

Figure 2929: Baseboard 4 Bottom Layer Design with Dimensions

Shown above in Figure 29 is the bottom layer of baseboard 4. This is the baseboard that was selected in the final run of the robot. Shown in red outline is the approximate dimensions of the components such as the omni wheels, the motors, the Arduino Uno's, a small bread board, and etcetera.

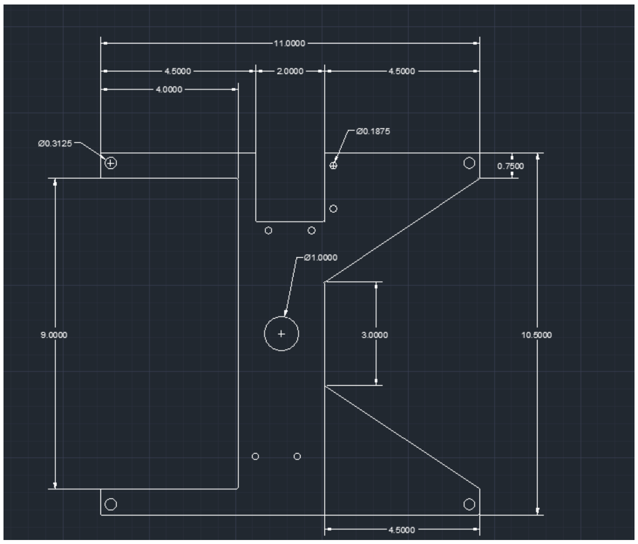

Figure 3030: Baseboard 4 Top Layer Design with Dimensions

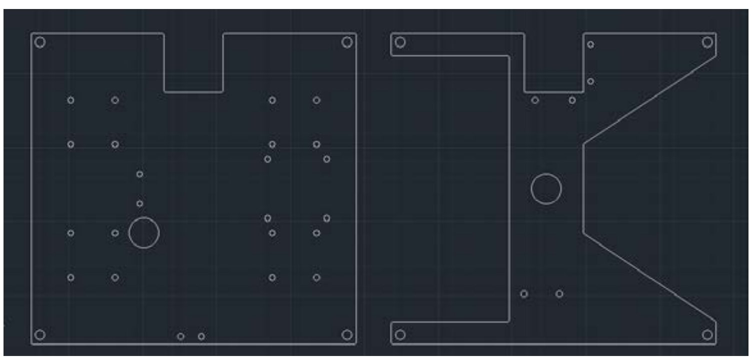

Figure 3131: Baseboard 4 Top and Bottom Layer Design

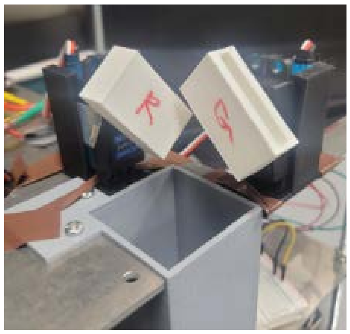

Figure 3232: Chip Servos and Mount

Figure 3333: Chip Chute

Appendix D

ABET Outcome 2, Design Factor Considerations

Design Factor	Page number, or reason not applicable
Public health safety, and welfare	Page 8-9
Global	Page 9
Cultural	Page 9
Social	Page 9
Environmental	Page 8
Economic	Page 8
Ethical & Professional	Not within a specific page but all IEEE rules were followed
Reference for Standards	Page 7

Table 3: Design Factors Considered

Appendix E

Schedule Spring 2023 Semester	
Date	Task
1/2/2023	Begin Navigation Research
1/7/2023	Workday with IEEE
1/21/2023	Workday with IEEE
1/23/2023	Finish Navigation Research
1/30/2023	Begin Implementing Navigation
2/2/2023	Revise Pre-senior Report
2/4/2023	Workday with IEEE
2/15/2023	Start Navigation Portion of Report
2/28/2023	Workday with IEEE
2/28/2023	Have all parts completely ordered
3/1/2023	Start Hardware Section of Report
3/4/2023	Workday with IEEE
3/15/2023	Finish Implementing Navigation
3/18/2023	Workday with IEEE
3/31/2023	Design Presentation Review (85%)
4/6/2023	Senior Design Report Draft Due
4/15/2023	IEE Hardware Competition
4/21/2023	Design Presentation Day
4/27/2023	Senior Design Final Report Due

Table 4: Semester Schedule

Appendix F

The below code is the first iteration of time based PWM movement before adding in the Arduino mega control. This code is just for Arduino Uno’s.

```
/*
Name: Grant Davis and Dylan Fox
After seeing encoder counts would not be reliable the team decided to focus
on a time-based Navigation system with different states.
The delays are used for time and have been mapped out for the IEEE arena. For
future teams they will need to use delays according to their
constraints.
Last Modified: 04/14/2023
/2023
*/
#include <SparkFun_TB6612.h>


// these constants are used to allow you to make your motor configuration
// line up with function names like forward.  Value can be 1 or -1
const int offsetA = 1;
const int offsetB = 1; // was negative

// these are the constants for our motor speeds for going forward
const int FWD_MTR1=105;
const int FWD_MTR2=90;
const int FWD_MTR3=101;
const int FWD_MTR4=90;
// these are the constants for our motor speeds for going right
const int Right_MTR1=52.5;
const int Right_MTR2=-45;
const int Right_MTR3=51.5;
const int Right_MTR4=-45;
int Count_pulses = 0; // intiate pulses for encoders
#define AIN1 9
#define BIN1 7
#define AIN2 10
#define BIN2 6
#define PWMA 11
#define PWMB 5
#define STBY 8
#define Encoder_output_A 2
#define Encoder_output_B 3

// Initializing motors.  The library will allow you to initialize as many
// motors as you have memory for.  If you are using functions like forward
// that take 2 motors as arguements you can either write new functions or
// call the function more than once.
Motor motor1 = Motor(AIN1, AIN2, PWMA, offsetA, STBY);
Motor motor2 = Motor(BIN1, BIN2, PWMB, offsetB, STBY);
int state=0;
void setup() {
}

void loop() {
  // put your main code here, to run repeatedly:
```

```
delay(500);
if(state==0)
{
delay(100);
motor2 = Motor(BIN1, BIN2, PWMB, 1, STBY);
motor1.drive(105); // FWD_MTR1=105; FWD_MTR3=101;
motor2.drive(90);
delay(6900);
brake(motor1,motor2);
delay(200);
state++;
}
if(state==1){
    // first forward motion to move toward corner
    motor2 = Motor(BIN1, BIN2, PWMB, -1, STBY);
    delay(1000);
    brake(motor1,motor2);
    delay(100);
    motor1.drive(105); // FWD_MTR1=105; FWD_MTR3=101;
    motor2.drive(90);  // FWD_MTR2=90; FWD_MTR4=90;
    delay(6900); // either 6450 or 6500
    brake(motor1,motor2);
    delay(200);
    state=99;
  }// end if (end state==0)
 if(state==2)
 {
  // this will be the state for turning corners
    motor1.drive(52.5); // Right_MTR1=52.5; Right_MTR3=51.5;
    motor2.drive(-45);  // Right_MTR2=-45;  Right_MTR4=-45;
    delay(4800);  //tried 4760, 4775
    brake(motor1,motor2);
    state++;
 }//end if (end state==1)
 if(state==3)
 {
  // side piece straight away
  delay(100);
  motor1.drive(105);  // FWD_MTR1=105; FWD_MTR3=101;
  motor2.drive(90);   // FWD_MTR2=90; FWD_MTR4=90;
  delay(4850); // tried 4700
  brake(motor1,motor2);
  state++;



                                          ;
 }// end if (end state==2)
 if(state==4)
 {
    // second right turn in the first corner
    delay(100);
    motor1.drive(52.5); // Right_MTR1=52.5; Right_MTR3=51.5;
    motor2.drive(-45);  // Right_MTR2=-45;  Right_MTR4=-45;
    delay(4825); // was 4775
    brake(motor1,motor2);
    state++;
```

```
 }//end if (end state==3)
 if(state==5)
 {
  // first long straight away
  delay(100);
  motor1.drive(105);  // FWD_MTR1=105; FWD_MTR3=101;
  motor2.drive(90);   // FWD_MTR2=90; FWD_MTR4=90;
  delay(12800);
  brake(motor1,motor2);
  state++;
 }// end if (end state==4)
 if(state==6)
 {
 // third right turn in the opposite corner with an upward movement of a foot
    delay(100);
    motor1.drive(52.5); // Right_MTR1=52.5; Right_MTR3=51.5;
    motor2.drive(-45);  // Right_MTR2=-45;  Right_MTR4=-45;
    delay(4800);
    brake(motor1,motor2);
    state++;
 }//end if (end state==5)
 if(state==7)
 {
 // moving up a foot
  delay(100);
  motor1.drive(105);  // FWD_MTR1=105; FWD_MTR3=101;
  motor2.drive(90);   // FWD_MTR2=90; FWD_MTR4=90;
  delay(1500);
  brake(motor1,motor2);
  state++;
 }
 if(state==8)
 {
 // RIGHT TURN
 delay(100);
 motor1.drive(52.5); // Right_MTR1=52.5; Right_MTR3=51.5;
 motor2.drive(-45);  // Right_MTR2=-45;  Right_MTR4=-45;
 delay(4800);
 brake(motor1,motor2);
 state=99;
 }
}
```

The next code is the combined states control when the Arduino Mega was added. This below code is for the Arduino Uno.

```
#include <SparkFun_TB6612.h>


// these constants are used to allow you to make your motor configuration
// line up with function names like forward.  Value can be 1 or -1
const int offsetA = 1;
const int offsetB = -1;

// these are the constants for our motor speeds for going forward
const int FWD_MTR1=105;
```

```
const int FWD_MTR2=90;
const int FWD_MTR3=101;
const int FWD_MTR4=90;
// these are the constants for our motor speeds for going right
const int Right_MTR1=52.5;
const int Right_MTR2=-45;
const int Right_MTR3=51.5;
const int Right_MTR4=-45;
int Count_pulses = 0; // intiate pulses for encoders
#define AIN1 9
#define BIN1 7
#define AIN2 10
#define BIN2 6
#define PWMA 11
#define PWMB 5
#define STBY 8
#define Encoder_output_A 2
#define Encoder_output_B 3
// Initializing motors.  The library will allow you to initialize as many
// motors as you have memory for.  If you are using functions like forward
// that take 2 motors as arguements you can either write new functions or
// call the function more than once.
Motor motor1 = Motor(AIN1, AIN2, PWMA, offsetA, STBY);
Motor motor2 = Motor(BIN1, BIN2, PWMB, offsetB, STBY);
int state=0;
void setup() {
  pinMode(13,INPUT);
  pinMode(12,INPUT);
  //Serial.begin(9600);
}

void loop() {
// put your main code here, to run repeatedly:
//delay(500);
//Serial.print("start");

if(digitalRead(13) == 0 && digitalRead(12)==0)
{
// strafe into and out of the wall to energize the motors
 //delay(2000);
 //Serial.print("strafe");
 motor1.drive(FWD_MTR1);  // FWD_MTR1=105; FWD_MTR3=101;
 motor2.drive(-FWD_MTR2);  // FWD_MTR2=90; FWD_MTR4=90;
} // end if (end of strafe state)


if(digitalRead(13) == 0 && digitalRead(12)==1){
    // This will be the second state of the program as the forward state
   // delay(100);
     //Serial.print("Forward");
    motor1.drive(FWD_MTR1); // FWD_MTR1=105; FWD_MTR3=101;
    motor2.drive(FWD_MTR2);  // FWD_MTR2=90; FWD_MTR4=90;
  }// end if (end of forward state)


 if(digitalRead(13) == 1 && digitalRead(12)==0)
 {
```

```
    // this will the be the third state of the program as the right turn
state
    //delay(100);
     //Serial.print("right");
    motor1.drive(Right_MTR1); // Right_MTR1=52.5; Right_MTR3=51.5;
    motor2.drive(Right_MTR2);  // Right_MTR2=-45;  Right_MTR4=-45;
 }//end if (end of right turn)


 if(digitalRead(13) == 1 && digitalRead(12)==1)
 {
    //Serial.print("stop");
    brake(motor1,motor2);
 }//end if (end of brake state)
} // end of main loop
```

The next code is from the Arduino Mega using a timer and time of flight sensor to control the Arduino Uno.

```
#include <Wire.h>
#include <VL53L0X.h>
#define JetsonPin 11
int Jetson=0;
#define SensorFPin 10
int SensorF=0;
#define SensorSPin 9
int SensorS=0;
#define NanoIn 12 //Input
#include <StopWatch.h>
uint32_t ti = 0;
#include <Servo.h>

Servo Door;
int pos = 0;

int state = 0;
StopWatch myStopwatch;
VL53L0X sensor1;// define object for sensor 1

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  sensor1.init();// initialize sensor 1 //look at this
  sensor1.setTimeout(500);// set time out for sensor 1
  sensor1.startContinuous();// measure continuously for sensor 1  //look at
this
  pinMode(SensorS, OUTPUT);
  pinMode(SensorF, OUTPUT);
  pinMode(Jetson, OUTPUT);
  pinMode(NanoIn, INPUT);
  pinMode(7,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(4,OUTPUT);
  Door.attach(3);
}
```

```
void loop()
{
  int distance1 =sensor1.readRangeContinuousMillimeters();// get distance for
sensor 2

 distance1 = distance1 -10;// -55 is to compensate for error. Change or set
it to zero to make it work for your sensor

  Serial.print("Distance 1: ");
  Serial.print(distance1);// print distance from sensor 1
  Serial.print("mm");
  delay(100);
  if (sensor1.timeoutOccurred())
  {
    Serial.print(" TIMEOUT 1");
  }
  Serial.println();
  if (distance1 < 40) //about to hit something
  {
    digitalWrite(SensorFPin, HIGH);
    SensorF = HIGH;
  }
  else
  {
    digitalWrite(SensorFPin, LOW);
    SensorF = LOW;
  }
Serial.print(SensorF);

    Serial.print(digitalRead(NanoIn)); //pin 10 for the nano
    if (digitalRead(NanoIn)) //nano is below 40mm
    {//if 1
      digitalWrite(SensorSPin, HIGH);
      SensorS = HIGH;
    }
    else
    {//if zero
      digitalWrite(SensorSPin, LOW);
      SensorS = LOW;
    }

    int JetsonIn = Serial.read();
    Serial.print(JetsonIn);
  if (JetsonIn == -1)
  { //we have no data yet
  }
  else
  { //we have data from the jetson nano
    if (JetsonIn ==49)
    {//stop moving //turn on led
       digitalWrite(JetsonPin, HIGH);
       Jetson = HIGH;
       delay(500); //delete this when we hook it up to the nano
    }
    else
    {//keep moving //led should be off
```

```
      digitalWrite(JetsonPin, LOW);
      Jetson = LOW;
    }
  }




  //start of the main code!!!!
if (Jetson ==0)
{
if (state == 0) //brake
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 0   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);

  if (ti >= 100) //brake
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 1) //strafe
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 1   ");
  digitalWrite(7, LOW);
  digitalWrite(6, LOW);
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);

  if (ti >= 1000) //strafe for 1 seconds
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 2) //brake
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 2   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);

  if (ti >= 100) //brake
```

```
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state ==3) //go straight
{

  Serial.print("State 3");
    myStopwatch.start(); //start the timer
   ti = myStopwatch.elapsed();
   Wire.write("state1");
    digitalWrite(7, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(4, HIGH);

  if ((ti >= 5000)||(SensorF == 1)) //drive forward for 5 seconds
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 4) //brake
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 4   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);

  if (ti >= 100) //brake
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 5) //turn
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 5   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);

  if (ti >= 4800) //turn
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
```

```
  }
}
else if (state == 6) //brake
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 6   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);

  if (ti >= 100) //brake
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state ==7) //go straight
{

  Serial.print("State 7");
    myStopwatch.start(); //start the timer
   ti = myStopwatch.elapsed();
   Wire.write("state1");
    digitalWrite(7, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(4, HIGH);

  if ((ti >= 4850)||(SensorF == 1)) //drive forward for 4.8 seconds
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 8) //brake
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 8   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);

  if (ti >= 100) //brake
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 9) //turn
{
```

```
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 9   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);

  if (ti >= 4800) //turn
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 10) //brake
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 10   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);

  if (ti >= 100) //brake
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state ==11) //go straight
{

  Serial.print("State 11");
    myStopwatch.start(); //start the timer
   ti = myStopwatch.elapsed();
   Wire.write("state1");
    digitalWrite(7, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(5, LOW);
    digitalWrite(4, HIGH);

  if ((ti >= 12800)||(SensorF == 1)) //drive forward for 12.8 seconds
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 12) //brake
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 12   ");
  digitalWrite(7, HIGH);
```

```
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);

  if (ti >= 100) //brake
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 13) //turn
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 13   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, LOW);
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);

  if (ti >= 4800) //turn
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else if (state == 14) //brake
{
  myStopwatch.start(); //start the timer
  ti = myStopwatch.elapsed();
  Serial.print("   State 14   ");
  digitalWrite(7, HIGH);
  digitalWrite(6, HIGH);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);

  if (ti >= 100) //brake
  {
    state++;
    myStopwatch.stop();
    myStopwatch.reset();
  }
}
else
{

Serial.print("0 degrees \n");
  Door.write(270);
  delay(1000);
  Door.write(0);
  Serial.print("State Done"); //turn right
}
}//Jetson =0
else //Jetson =1
{
```

```
  myStopwatch.stop();
  delay(500);
  //don't reset
}

} //Void(loop)
```