Robot Final Report

Engineering 1282.01H

Spring 2020

Thomas Krisak

Keith Kriston

Alekzander Srode

Kevin Wang

[redacted] MWF 10:20

Date of Experiment: 02/04/2020

Date of Submission: 04/20/2020

Team H1 - Executive Summary

Carmen’s Diner faces increased demand from implementation of an online ordering app. To meet this demand increase, the diner looks to utilize automated vehicles to solve simpler tasks, allowing staff to focus on more complicated work. Multiple teams were tasked with designing such vehicles, each creating a robot which could quickly, precisely, and consistently perform tasks about a premade diner course [2]. These tasks included pressing a jukebox button based on light color, dropping off trays, flipping a certain lever up and down on the ice cream machine, flipping a burger patty, sliding a ticket on its rack, and pressing the final red button [1].

Team H1 initially intended to use a triangular chassis with a three-wheel omnidirectional drivetrain. For task completion, the team would employ a forklift-shaped arm which could rotate and move up and down. After some development, however, the team ended up changing the design to a laser-cut wooden rectangular chassis driven by two powered wheels and two sliders. The task completion mechanisms developed along with the performance tests of the robot, finalized as a front and back arm. Additionally, both a light sensing mechanism and a line following set up supported the arms and drivetrain in completing tasks and movement.

The wheels, motors, and motor mounts were already available, but the chassis and sliders were specially designed by and manufactured for the team. The drivetrain and chassis were mated using screws and 3D printed parts. For navigation, the robot used its drivetrain to follow a pre-coded path determined and heavily tweaked by the team in testing. Early on, the wheels often detached from the motors, an issue fixed by securing the wheels to the motor axle with glue. While this prevented them from falling off, one wheel was glued at an angle that hindered proper movement. Additionally, the chassis’ holes were slightly too small for screw heads, a problem fixed by sanding.

Three optosensors were also attached to the chassis, allowing the robot to determine if it was over any lines about the diner course. An algorithm was developed which utilized these sensors to keep the robot moving along the line. Despite brief testing, neither the optosensors nor RPS was not utilized. Given more time, the team would have included these systems.

Furthermore, a CDS cell was attached to the bottom of the robot, enabling it to detect the brightness of any light below it. A piece of cardboard surrounded the cell, minimalizing the amount of ambient light that reached it. Meanwhile, a red filter covered the bottom of this cylinder to increase the contrast between red and blue light. This cover allowed the robot to more accurately detect the color of light beneath it. The robot was able to sense red light but struggled to accurately detect blue. Thus, the robot defaulted to blue unless red light was detected.

The first arm saw a cardboard tray attached to an axle, in turn connected to a servo motor. Additionally, an erector set piece was attached to the same axle, acting as a longer front arm. It had enough clearance to turn roughly 90 degrees, started resting at an angle level to the floor. Tasked with depositing the tray, flipping the grill, and pulling the ice cream lever, initially this arm reliably met its requirements. Despite early success, however, the axle which powered this arm broke off the servo motor which drove it. The arm failed to complete tasks after this despite attempts to repair it. A better joint between the axle and motor would be needed for future tests.

Another servo, modified to act as a DC motor, was paired with a gear and toothed erector set piece, enabling the piece to extend and retract. This arm was tasked with moving the ticket but proved problematic during testing. Both movement and arm extension required precise code tweaking only accomplished through testing, and the arm also had to be repaired when it became misaligned during said testing. Eventually, it was able to accomplish its task.

Table of Contents

List of Figures

List of Tables

1. Introduction

Due to an increase in business from the addition of the FEHpingo online ordering app, Carmen's Diner was struggling to keep up with the number of orders. In order to keep up with orders, the owners of Carmen's Diner are investing in launching a strategy that utilizes automated vehicles to perform the simpler tasks within the diner. This way, the staff working can focus on the complicated tasks, such as human interaction. In order to achieve this strategy, management contracted the Ohio State Research and Development (OSURED) team to select a prototype to complete these tasks. The design chosen would be used in a restaurant relaunch.

In order to choose the best robot, a scale model of the diner (referred to as "the course") was constructed by a research team at The Ohio University. Robot prototypes are required to complete the tasks setup on the course in a timely fashion and with precision to be eligible for final selection [1].

Multiple teams of four were created in order to have different possible robot concepts and designs. The team Hash Browns 1 was made up of the people Kevin Wang, Keith Kriston, Thomas Krisak, and Alekzander Srode. Over the course of the design process, performance tests for each team acted as checkpoints, one on each day as follows: February 21st, February 28th, March 6th, and March 23rd. Each of these tests would assess different robot capabilities [3]. The four performance tests were planned to lead up to a final individual test on March 27th, with a final group competition on April 4th, 2020.

The second section, Preliminary Concepts, covers the initial ideas and designs the team had in mind for the robot, as well as how the team brainstormed and how the robot was initially going to complete the tasks. The third section, Analysis Testing and Refinement, covers what problems were found in the robot during testing and how these problems were fixed. The fourth section, Individual Competition, covers the plan and process to finish the individual competition. The fifth section, Final Design, covers the details of the finished robot, including drivetrain, chassis, task completion mechanisms, and how these components factored into the budget. The sixth section, Final Competition, covers how the robot performed in the final competition. The seventh section, Summary and Conclusion, goes over the most important aspects of the project and discusses what would have been done had the team been given more time or money. The eighth section, Reference, lists all the outside resources the team used when creating this document.

2. Preliminary Concepts

This section covers the brainstorming and design decisions of the team's robot, along with the rules and regulations regarding the robot course runs.

2.1 Brainstorming Process

Each member individually brainstormed task completion methods and mechanisms prior to Team H1's first meeting. These ideas were then reconciled following discussion at the first team meeting. The brainstorming criteria included order of task completion, chassis design, drivetrain design, and mechanism design to complete each task. Each idea was accompanied by a drawing. After discussing their ideas, the team worked together to create a decision matrix as shown in Table B1 in Appendix B. The decision matrix involved rating criteria weighted by importance. Each idea presented by a group member was rated using the criteria listed within the table. The total grade of the idea was listed at the bottom. The decision matrix helped the team make unbiased decisions based on the most important aspects for each component. Because the decision matrix was a dynamic table, if the team encountered problems after settling on a decision using the matrix, the team would be able to make changes pertinent to their discoveries. The team made three initial designs, as shown in Appendix C in Figure C1, which utilized the highest rated components in the decision matrices. The first design involved a splitting forklift, a triangular chassis, and three omni wheels. The second design was different from the first in that it used a square chassis rather than a triangular one, and rather than three omni wheels, it used two omni wheels and one slider. And finally, the third design used a complex lift, a triangular chassis, and three omni wheels.

Following the creation of the decision matrix, the team decided to combine the concepts of the first, second, and third designs. Instead of omni wheels, the team selected normal wheels. This was mainly because the team was unsure of how the omni wheels worked. The combined design involved a triangular chassis, with two wheels on the back side, a slider in the front, and a single complex arm on the front. The team then proceeded to create a mockup model of the design out of cardboard, foam, tape, and spare pencils. Figure 1 below depicts the mockup model. Figure C2 in Appendix C depicts a different view of the mockup.

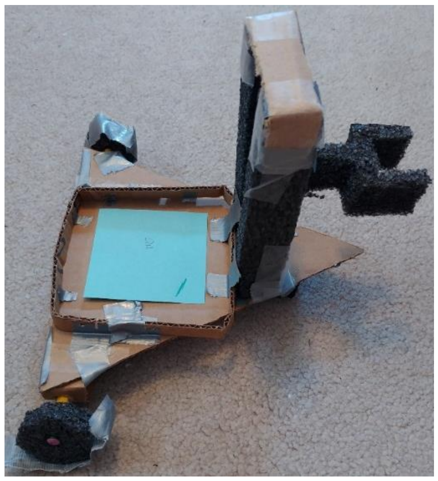


Figure 1: Picture of robot mockup model from the side.

After the mockup model was created, the team noted a problem with the design. The problem was that a triangular chassis was not a good decision as it did not allow for proper room on the robot to place the proteus, or any complex mechanisms. And so, instead of the triangular chassis, the team decided to go with a square chassis. A second problem was later discovered, which was that the intricacy of the complex arm, an arm that could move up and down and turn, was determined to be too difficult and would not be precise enough. The idea was scrapped, and two new concepts were created in its place. Rather than one single arm, the robot would include two arms. An arm on the back side that could extend in and out, and an arm on the front that would tilt up and down.

Once the team had decided on the robot's chassis, drivetrain, and task completion mechanisms, the team needed to decide what would be the best sensors for the robot to use. The team knew that there was a dead zone on the course, where RPS would not be available to the robot for use. But there were lines on the course available to the robot for following. Because of this, the team chose to include a line following kit on the bottom of the robot using three sensors and a circuit board.

Besides the navigation, the robot needed to detect light colors and differentiate between blue and red light. For the robot to accomplish this, the team chose to include a CDS cell on the bottom of the robot, with a red plastic filter placed over top of the cell. The team chose to include a red filter because it was found that the cell could better differentiate between the colors, blue and red, when a red filter was placed over top of the cell.

Along with the hardware decisions made, the team also needed to make initial software decisions as well. For the course, a Robot Positioning System (RPS) was included for easier navigation and position checks of the team’s robot. But because the team had little to no knowledge, at the time of brainstorming, when it came to the RPS, the team decided shaft encoding would be the best route for programming at the start of the project.

Another sensor discussed by the team was the bump sensors. At the time of the initial design process, the team was unsure whether the robot would need bump sensors or not. Based on the team’s initial understanding of the course and how the software would work, it was suggested that bump sensors would not be included. But because the team wanted to be prepared for possible need of bump sensors, the sensors were also involved in the layout of the robot’s chassis and drivetrain.

2.2 Robot Tasks, Conditions, and Rules

Teams’ robots had to quickly, precisely, and consistently perform several different tasks under multiple conditions [2], outperforming other teams’ robots in order to progress in the competition. These tasks included pressing a jukebox button based on color, dropping off used trays, flipping a lever up and down on the ice cream machine, flipping a burger patty, sliding a ticket on the ticket rack, and finishing by pressing the final red button [1]. On the next page, Figure 2 depicts the robot course with the location of each task, with Table 1 that lists each task the robot had to perform, along with the points assigned to each task.

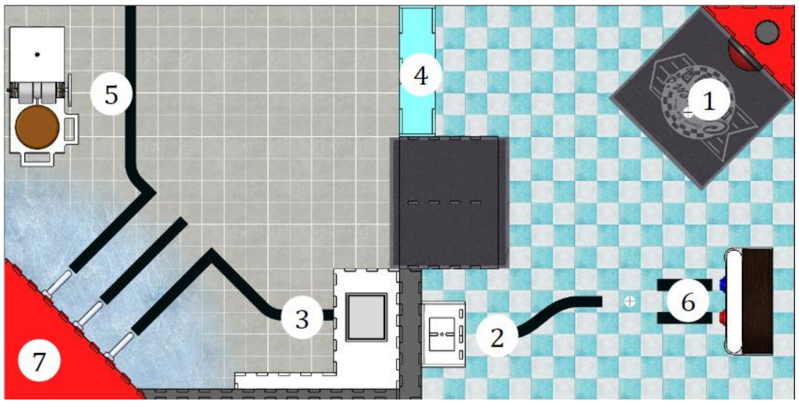


Figure 2: Picture of the robot course (Diner Scale Model). The numbers on the figure depict the locations of tasks. One (1) is the final button and start light. Two (2) is the trash bin. Three (3) is the sink. Four (4) is the ticket rack. Five (5) is the burger. Six (6) is the jukebox. And Seven (7) is the ice cream machine [4].

Table 1: Tasks for robot to complete on the course, along with points allotted to each task [4].

Primary Tasks	**Points**
Initiate on start light	8
Press the final charging button	8
Return tray to either trash bin or sink	7
Move ticket from starting position	8
Any ice cream lever is flipped down	8
Any ice cream lever is flipped back up	7
Press any juke box button	8
Press the correct juke box button	7
Burger flip is initiated	7
Burger flip is completed	7
Possible Primary Task Points	**75**
Secondary Tasks	
Move ticket to final position	6
Ticket remains in final position	4
Only correct ice cream lever is flipped down	7
Ice cream lever is left in down position for at least 7 seconds	5
Hot plate is returned to initial position	3
Possible Secondary Task Points	**25**
Total Possible Task Point	**100**
Penalties	
Interfering with a competitor's robot	DQ

The conditions that the robot had to meet were varied. The first condition was that the robot had to be within the size constraints of 9 inches long, 9 inches wide, and 12 inches tall. The second condition that had to be met was the robot was required to be fully automated. Using a provided proteus, the team needed to make sure the robot worked only though automated code. The robot could not transmit or receive any wireless signals, except from RPS and the robot was not allowed to be touched or controlled by anyone during its tests. Constraints also pertaining to the proteus included no adhesives permitted on the proteus, the proteus could not be used as a sensor, accept the when using the accelerometer, and the proteus was not allowed to be used in the structure or mechanisms of the robot.

The third condition for the robot was that the cost of all parts that went into making the robot, whether used on the robot or not, had to be within a total of $160. Teams that exceeded their $160 budget would lose one point for every $0.50 over the budget. Budgets that were under the $160 limit received an additional score multiplier of 0.005 for every dollar saved, up to $40. For example, if the team scored the maximum score possible for the final competition, which was 100 points, and the team saved the maximum amount of money allowed for extra points, $40, then the total extra points the team would have received would have been 0.005 times 40 times 100, or 20 extra points.

The fourth condition was that all robots were required to have a QR code placed 9 inches above the course ground during all performance tests, practice runs, and competitions. The QR code also was required to be fully visible to RPS. This was required so that all robots could be kept track of while on the course. This was a requirement following Performance Test 2, as QR codes had not been distributed to teams up until that point.

The fifth condition the robot had to meet was that no adhesive material could contact the course surface at any point during a robot's run. And the sixth condition was that all parts on the robot, including fasteners and adhesives, had to come from the FEH Robot Store. Any parts or adhesives that a team wanted to use on their robot that did not come from the FEH Robot Store had to be reviewed and approved by the team's GTA. The cost of the item used also had to fit within the robot budget of the team using that item.

Along with the conditions, the robot also had to follow a set of rules for the competitions. A list of the rules is given as follows:

1. The team will have one minute to setup their robot. Once the run has started, the robot must complete all tasks within two minutes.

2. The robot must be connected to the RPS system, and the QR code should be detectable.

3. The robot cannot interact with another robot on another course. Any object placed in or tossed onto another course is not allowed. The robot must also not leave behind any object on the course that could cause interfere with another robot.

4. A part that detaches or is lost from a robot, intentionally or unintentionally, is defined as a loose or disposable part. Course officials may choose to confiscate any loose or disposable parts at the end of a run.

2.3 Team's Task Completion Decisions

To start the task completion decisions, the team needed to figure how the robot would complete each task. Through long discussions, the team eventually came to an initial layout for completing the course. The robot would begin at the starting light. From there, the robot would go up the ramp and place the tray into the sink. Then the robot would move to the receipt and push it to the opposite side. Next, the robot would move to the burger and flip it. Then the robot would move to the ice cream machine and flip the lever down, wait seven seconds, and flip it back up. Next, the robot would head back down the ramp to the jukebox, press the correct button the jukebox, and finally head back to the starting position to press the final button. Figure 3 below depicts the path the team chose for the robot to traverse the course.

Next, the team determined how the robot would complete each of these tasks. From some more discussion, it was decided that the front arm would be used to complete the tray, ice cream lever, and burger tasks. Because the front arm could move up and down, it could hold the tray, and then slide it into the sink when needed. It could move down to flip the lever down, and then move back up to flip the lever back up. And it could move up, along with the turning of the robot, to flip the burger and put the grill back down.

To complete the receipt task, the robot would use the back arm. Because the back arm could move in and out, when the robot was in the right position, the arm could extend out between the edge of the receipt rack and the receipt itself. From there, the robot could move

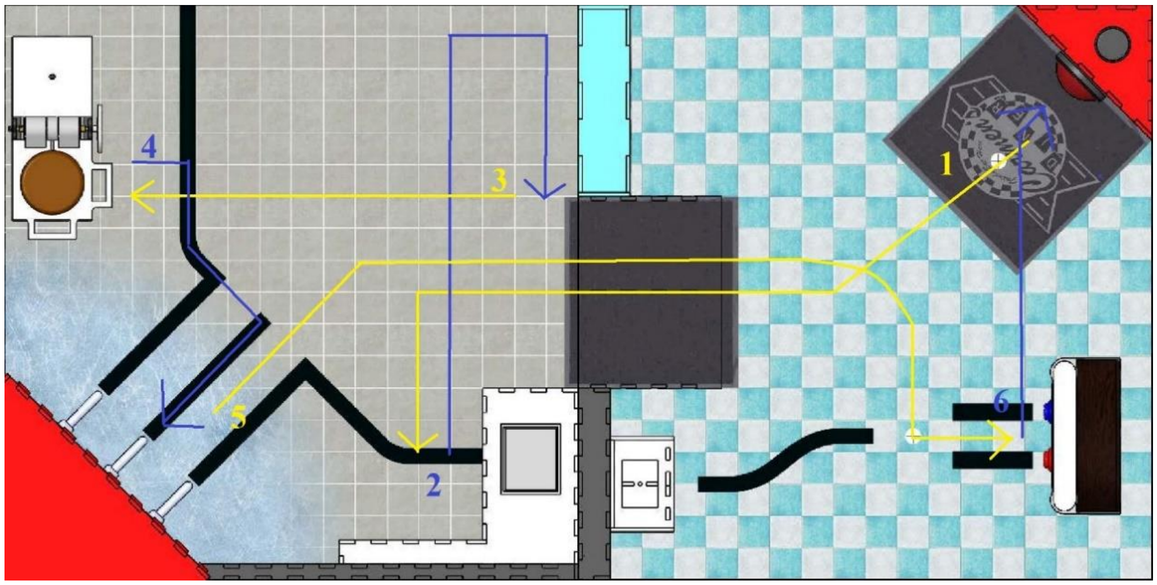


Figure 3: Picture of planned full path taken by robot in competitions.

forward, pulling the receipt along with it. And once the robot reached the end of the receipt rack, the arm could be pulled back in, allowing the robot to move to the next task.

To complete the jukebox task and the final button pressing task, it was determined that the robot could run into the buttons. No arm was necessary for pressing the buttons.

3. Analysis, Testing, and Refinements

This section covers the analysis and refinements of the robot’s drivetrain and chassis and mechanisms. It also covers an explanation and analysis of testing done on the robot over time.

3.1 Analysis of the Drivetrain

The drivetrain included anything that helped the robot move. In the team’s case, the drivetrain included sliders, motors, and wheels. At first, the team wanted to use omni wheels to allow the robot to move more freely. But because the team members did not understand how the omni wheels worked at the time of analyzing, the team began considering dubro rubber wheels more. Upon further inspection of the dubro wheels, the team decided that the robot would work just as well with the dubro wheels as it might with the omni wheels. The team decided this because when considering how the code would work, the team knew how to turn the robot without omni wheels. And since no team member at the time understood the how the omni wheels functioned, the team wanted to play it safe. So dubro wheels were selected. But the dubro wheels came in three different diameters, 2.0 inches, 2.5 inches, and 3.0 inches.

Table 2: Distances between task locations

Locations	**Distances (in.)**
Start to Sink	46
Sink to Receipt Start	30
Receipt End to Burger	25
Burger to Ice Cream	25
Ice Cream to Jukebox	48
Jukebox to Finish	30
Total Distance	**204**

To determine the best diameter of wheel for the robot, the team analyzed the motors that were available to teams. To start the analysis, the team measured distances between tasks on the course. The distances between different tasks are given in Table 2 above.

The total distance between the tasks the robot needed to complete, in the order the team decided upon, was 204 inches. Next, the team needed to estimate how long it would take the robot to complete the tasks. For the tray, the estimated time to complete was 5 seconds, as the robot only had to place the tray into the sink. For the receipt, the estimated time was 10 seconds, as the arm had to extend outward, allow the robot to move from start to finish, and then allow the arm to retract back in. For the burger, the estimated time was 5 seconds as the robot could move the front arm up and turn at the same time, therefore not requiring long to complete the task. For the Ice Cream lever, the estimated time for completion was 10 seconds. 3 seconds were dedicated to moving the arm of the robot up and down, while 7 seconds were dedicated to waiting and repositioning the robot, as the requirement of the secondary task indicates. For the Jukebox, the estimated time of completion was 10 seconds, as the robot needed to read the light to get the correct button, and then move forward and press the correct button. Finally, the estimated time for the last button was less than a second, as pressing a button is almost instant.

From the estimated times for task completion, the total time required for task completion was 40 seconds. As a recommendation from the staff in charge of the project, the team also added a buffer time of 10 seconds to our total so that the team could make sure the motor and wheels we chose worked well and had some leniency. That brought the total time for tasks to 50 seconds. Because for competitions the robot had only 2 minutes, or 120 seconds, to complete the course, subtracting the task time from the allotted time gave the team an estimate of 70 seconds that the robot would have to move between tasks.

Using the time of 70 seconds between tasks and the total distance the robot would have to travel, the team determined the minimum linear speed required for the robot to traverse the whole course in time. This was done using Equation 1 below. Upon calculating, the found minimum was 2.9 inches/second. Next, the team selected a wheel size believed to work well with the robot. Because the team was already thinking about using Igwan motors, and since there were adapters between Igwans and dubro wheels for 2.0-inch and 2.5-inch diameters, the team went with the 2.5-inch diameter wheels.

$$Minimum\ Linear\ Speed = \frac{Total\ Distance\ to\ Travel}{Total\ Time\ for\ Travel} \tag{1}$$

Using the dimensions of the wheel, the rotational speed of the motors required were calculated using Equation 2 on the next page. Since the wheels selected had a diameter of 2.5 inches, to find the circumference of the wheels, 2.5 inches was multiplied by pi (π), which gave a circumference of about 7.854 inches. Then the rotational speed was calculated, dividing the minimum linear speed, 2.9 inches/second, by the circumference of the wheels, 7.854 inches. This gave a rotational speed of 0.369 revolutions/second. But because the rotational speeds of the motors were given in revs/min (RPM), the value was multiplied by 60 sec/min to convert revs/sec into RPM. This gave a rotational speed of 22.14 RPM.

$$Rotational\ Speed = \frac{Minimum\ Linear\ Speed}{Wheel\ Circumference} \tag{2}$$

Now the team needed to determine the minimum torque required by the motors to move the robot. This minimum would be the torque required to move the robot up the ramp on the course. To find this torque, the team started by estimating the combined weight of all the parts hat were expected to go on the bot at the time. This total added up to an estimate of ~1556 grams, or ~54.89 ounces. Then the team had to find the angle of the ramp. To find this, the team measured the height, length, and hypotenuse of the ramp, which were measured to be 3 inches, 11 inches, and 11.4 inches, respectively. Then using Equation 3 below, the angle of the ramp was calculated. The calculated angle came out to be ~15.3 degrees.

$$Angle\ of\ Ramp = \arctan\left(\frac{Height}{Length}\right) \tag{3}$$

Now that the team had an estimated weight, and the angle of the ramp, the required torque to get up the ramp could be calculated. First, using the relationship given in a PowerPoint by the faculty in charge of the project [5], the team could calculate the required minimum force to move the robot. This force turned out to be 22.484 ounces of force. With the minimum force needed, the torque to move the robot up the ramp could be found. To do this, Equation 4 below was used, and the minimum torque required to move the robot up the ramp was calculated to be 28.105 oz-inch. And since the robot was using two motors, the required minimum torque of one motor turned out to be half that, which was 14.0525 oz-inch.

$$Minimum\ Torque\ Required = (Minimum\ Required\ Force) * (Wheel\ Radius) \tag{4}$$

With the required minimum torque, 14.0525 oz-inch, and the minimum rotational speed, 22.14 RPM, the team could select the best motor for the robot. Using a provided graph of the torque versus rotational speed of all motors available [6], the team plotted a point on the graph representing the minimum rotational speed and torques required for the robot. This graph with the added point is shown in Figure 4 below.

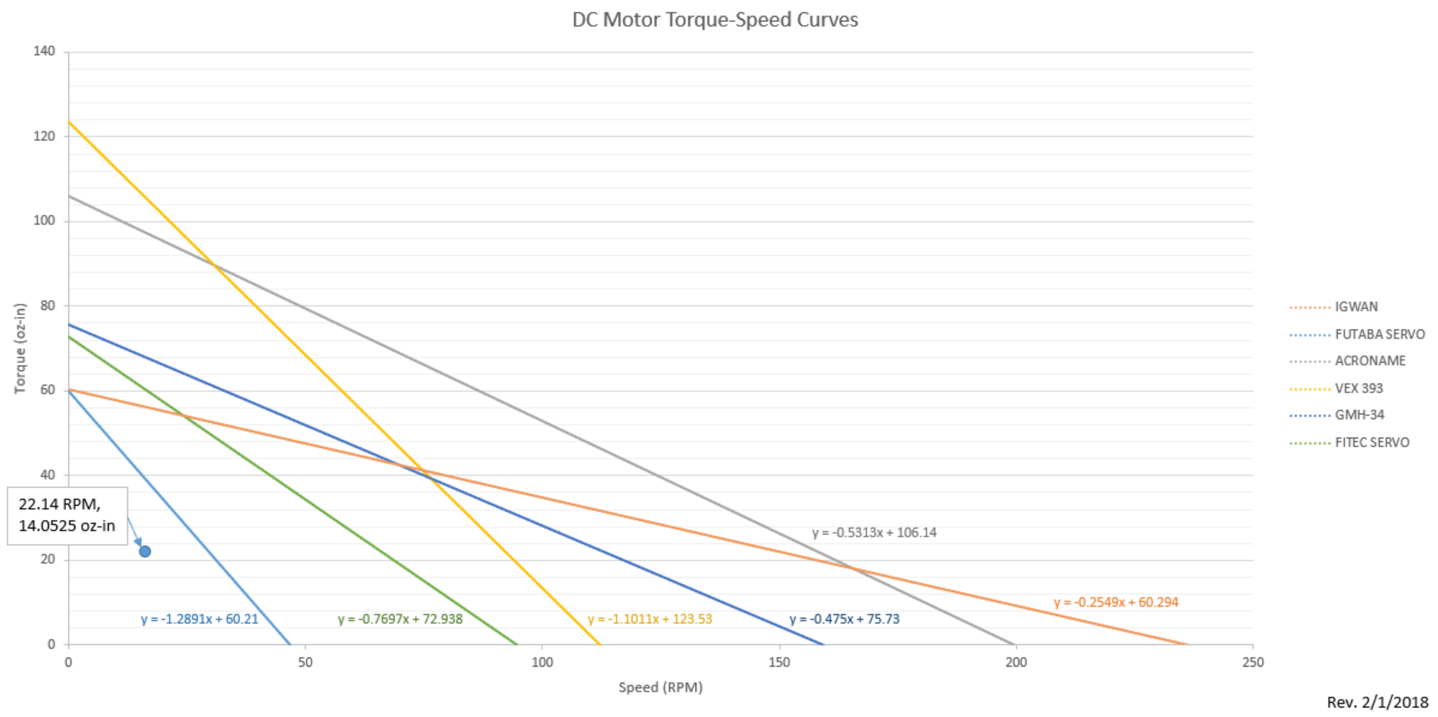


Figure 4: Graph of motor torques (oz-in) and rotational speeds (RPM) [6]. Blue point indicates team's minimum required torque and rotational speed.

Using the graph and the added point, the team made note of all the motors that met or exceeded the requirements for the robot. Turned out that with the team's calculations and estimates, all motor options passed the requirements. Because of this, the team decided to stick with the idea of using Igwan motors for the robot as the Igwans did pass the team's requirements. The Igwans also included shaft encoders, reducing the work the team would have had to put into making shaft encoders.

3.2 Analysis of the Chassis and Mechanisms

Closer analysis of the preliminary designs and mockup model revealed multiple flaws in the initial idea. One observation made was that using a triangular chassis to fit the 9 inches long by 9 inches wide condition made it difficult to fit both the complex arm and proteus onto the robot. This was because accommodating for the width of the wheels rendered the chassis too small, not leaving enough room for the proteus or mechanisms on top.

Another observation made from the mockup model was that the preliminary arm design may have been too complex and heavy to be easily constructed and fixed. Because the complex arm would have been able to move up, down, left, and right, the arm would have required at least two motors. And since fitting one arm onto two motors, or vis versa, was deemed too difficult and may have caused the arm to end up being inaccurate, the idea was replaced.

As a result of these observations and the drivetrain analysis, the team decided to alter the robot design, and replaced the triangular chassis with a square chassis with two sliders towards the front, and two dubro wheels powered by igwan motors in the back. Then the complex arm was replaced with a single arm on the front that tilted up and down, and a single arm on the back that extended in and out.

3.3 Testing and Refinement

Testing logs were made to record the changes and refinements that were made for each performance test (2/21, 2/28, 3/06). These testing logs made note of the date, the location, the reason for test, the members present, the changes made as a result, number of runs, observations of the test, and notable results. Videos were taken of random runs during each test.

The refinements made during testing mainly involved tweaks in the programming of the robot. Throughout all three of the performance tests, movement was a constant issue, which needed big fixes in the beginning and small changes toward the end. Big fixes included repairing the movement function to allow for accurate forward travel in instances greater than four inches, as well as accurate turning functions for frequently used directions (90 degrees either way). Small changes involved altering the starting and ending position of the arms, as well as small movement tweaks for better execution of the performance test. Further small changes included sleep functions and motor power modifications.

For Performance Test 1 (pressing the jukebox button), movement was a big concern, constant tweaks had to be made and many of the early test runs were ruined by crooked forward driving and inconsistent turns. After the team made the driving and turning relatively consistent on flat ground, the robot managed to make it to the jukebox light. It managed to detect red light just fine but could not properly detect blue light. In order to fix this problem, the team changed the code so that if the robot detected any light, it would default to blue unless it detects red light. Afterward, the robot was having trouble fully pressing the jukebox buttons (chassis was too short). This was fixed by adding a tall barrier to the front of the robot in order to better reach the button. It was also noted that the robot was going crooked left up the ramp, this problem was fixed by increasing the motor power (speed) going up the ramp. The robot strayed less from its intended path.

Inconsistent forward travel distance was another bane while testing for Performance Test 2, this test involved dropping the tray in the proper area and sliding the receipt. Constant small tweaks were made for the robot's movement and turning because the route chosen required accurate placement prior to sliding the receipt. The tray deposit task was successfully completed after the fifth time testing for it. The receipt sliding was where the team spent most of the testing. After depositing the tray in the sink. The robot kept bumping into the receipt box, positioning crooked, or positioning too far from the receipt. These problems kept the team from being able to complete the receipt slide sooner. A small tweak in movement to solve one problem caused another. Eventually, the robot arm tasked with sliding the receipt was extended to make it easier to position the robot. This refinement made the difference in being able to slide the receipt or not.

Performance Test 3 involved flipping the patty. The team initially found great success with flipping the patty, able to move the patty and return the hot plate to its original position in one smooth motion, but constant hardware failures caused the team to backtrack, fix the robot, and retest. The front arm was tasked with flipping the patty. The connection between the motor and the axle of the front arm was the main source of problems. Due to stress from testing and an inherently weak connection, the axle got disconnected from the motor multiple times. Each time, the team tried a different method of fixing the two pieces back together. First epoxy, then gorilla glue, then super glue, then hot glue. The team eventually got the two pieces relatively stable by attaching a rubber tubing to the metal of the axle in order to give the glue a better grip.

Following Performance Test 3, the epoxy, gorilla glue, super glue, and hot glue all came loose from around the motor and axel for the front arm. Because of this, the rubber tubing was removed from the axel, as it seemed to add no extra help, and the gluing was redone.

4. Prototype Status

This section covers the status of the robot and the time and budget that went into the creation of the prototype.

4.1 Budget and Time Spent

The total budget allotted to the team at the start of the project was $160. Over the course of the project, the team made 17 different orders. The orders totaled a cost of $144.11, with a remaining $15.89. Of the spent budget, there were five categories that the money fell into. These categories included the robot's chassis, drivetrain, front arm, back arm, and sensor electronics. Depicted in Figure 5 below is each of these five categories and the remaining budget, with the percent of the total budget that went into each section. A sixth section was included for cardboard alone to help show how cheap the material was for the robot's construction.

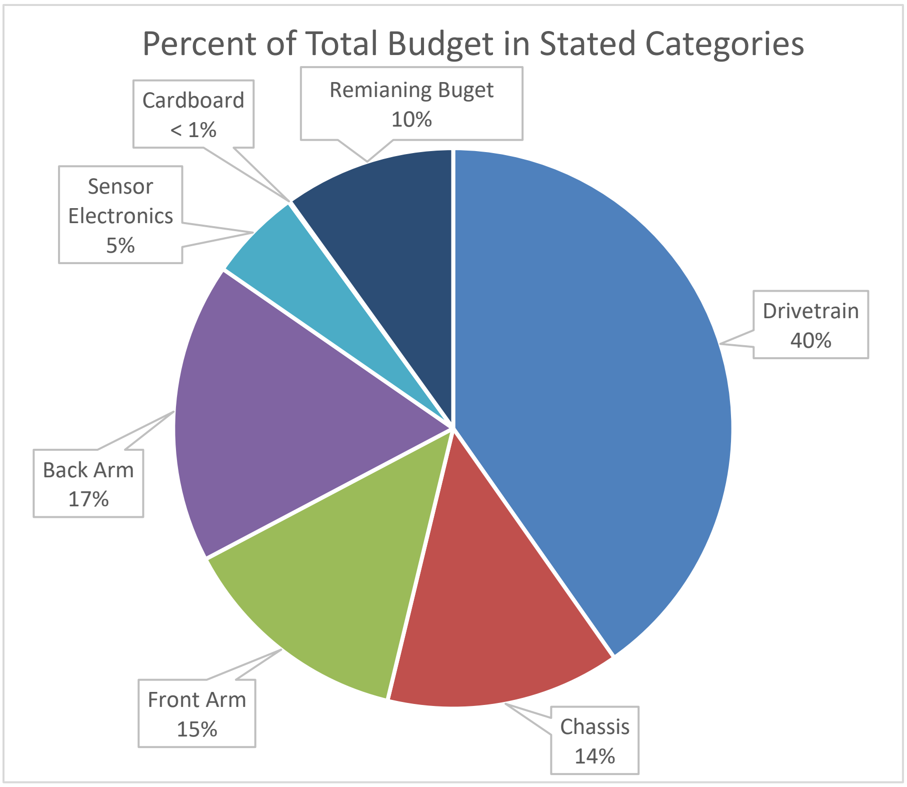


Figure 5: Pie chart depicting the rounded percentages of the total budget that went into each category.

Tables D1 and D2 in Appendix D give a closer breakdown of parts bought, in what quantity, and their total cost, as well as the cost from each order that went into what category of the robot, and the progression of the decreasing budget after each order. Figure D3 also located in Appendix D gives a visual representation of the budget progression in Table D2.

Along with budget, each team member had to keep track of their time spent on different parts of the robot. These areas of work included Documentation, Project Management, CAD, Coding, Testing, Building and Construction, and any Other smaller categories, or when time was spent equally on multiple sections at once. Table 3 below is the team's representation of how much time was spent individually and together on each category of the project, along with the total time spent, by each person, working on the project.

Table 3: A table depicting individual times for each category, group effort times for each category, and total time spent by each person on the project overall. Times are given in minutes. (NOT YET COMPLETE)

	Documentation	Project Management	CAD	Coding	Testing	Building and Construction	Other		Total
Thomas Krisak	920 min	340 min	390 min	0 min	925 min	465 min	410 min		3450 min
Keith Kriston	1020 min	735 min	345 min	480 min	525 min	1335 min	150 min		4590 min
Alek Srode	2355 min	660 min	320 min	840 min	955 min	700 min	865 min		6695 min
Kevin Wang	1795 min	1175 min	440 min	500 min	1345 min	785 min	405 min		6445 min
Total	6090 min	2910 min	1495 min	1820 min	3750 min	3285 min	1830 min		

Tables D3, D4, D5, and D6 in Appendix D provide a time sheet breakdown of each group member's time spent over the course of the project.

4.2 Chassis and Drivetrain

As outlined in preliminary concepts and brainstorming, the team initially intended to use a triangular chassis with a three-wheel omnidirectional drivetrain. After a brief mockup and some further brainstorming and discussion, the team changed the design. The final robot instead utilized a rectangular chassis with a powered two-wheel and two-slider drivetrain, a design which resulted in a cheaper yet more reliable and workable robot.

4.2.1 Chassis

The chassis was designed as three layers, each laser-cut into MDF per design of the team. A three-layer design was chosen to enable different holes to be cut for different purposes in each layer. With the separate pieces held together using machine screws and nuts, it provided the platform for all the mechanisms of the robot and the Proteus controller.

The bottom layer of the chassis had multiple holes laser-cut into it for multiple purposes. At the top and bottom, two large rectangular holes were used for wire management, helping organize wires both from the drivetrain and arm mechanisms on the robot. The smallest holes in the corners of the piece were designed to mount bump sensors, although these sensors never came into use during testing. Additional hole were cut into the chassis piece for mounting the drivetrain and the line following kit respectively. The holes were shaped hexagonally to accommodate the heads of the screws used to secure these mechanisms and additionally secure the chassis together. Initially too small for their purpose, these holes required the only post-design modification to the chassis: sanding them to the proper size.

The middle layer of the chassis also had multiple cuts, many of which aligned with the bottom layer’s holes for the same purposes. Firstly, the wire management holes were placed in the same position with the same size and shape as those on the bottom layer. Furthermore, the holes for mounting the line following kit and drivetrain were aligned with those on the layer below. These holes were circular rather than hexagonal, as they did not need to accommodate the heads of the screws.

Like the other two layers, the top layer featured certain cuts for specific purposes. Once again, two large holes were placed at the top and bottom for wire management. In between these holes, one large cut was made to hold the Proteus controller in place on the top of the chassis. Holes to secure the chassis together, aligned with those used to secure the drivetrain and line following kit, were placed in places appropriate to the holes on the other two chassis pieces.

4.2.2 Drivetrain

The drivetrain used two wheels paired with two sliders in a rear-wheel-drive configuration. As outlined in Analysis of the Drivetrain, the team chose to use two-and-a-half-inch diameter wheels. These wheels were powered by Igwan motors, mounted to the chassis via prior-designed 3D-printed motor mounts. This configuration, including wheels and motors, was placed on the rear of the robot while the two sliders were mounted on the front. Unlike the motor mounts, the sliders were designed solely by the team to their desired specifications.

The first problem with the drivetrain arose during testing, where the wheels would become detached from the rest of the drivetrain. This problem occurred frequently enough to cause the team to permanently attach the wheels to the motors using super glue and hot glue. This fix was the only change to the drivetrain design that the team made. However, due to a hastily done glue job, one of the wheels was permanently angled inward, a second drivetrain problem that likely resulted in inaccurate movement.

4.3 Task Completion Mechanisms

The first Performance Test required the robot to press one of two colored buttons on the jukebox corresponding to color of the light in front of the jukebox. In order to accomplish this, a CDS cell was attached to the bottom of the robot, as seen in Figure 6 below. The CDS cell allowed the robot to determine the brightness of a light below it. A piece of cardboard was crafted into a cylinder that surrounded the cell, minimalizing the amount of ambient light that reached it. A red filter was attached at the bottom of the cylinder, which increased the contrast beneath it.

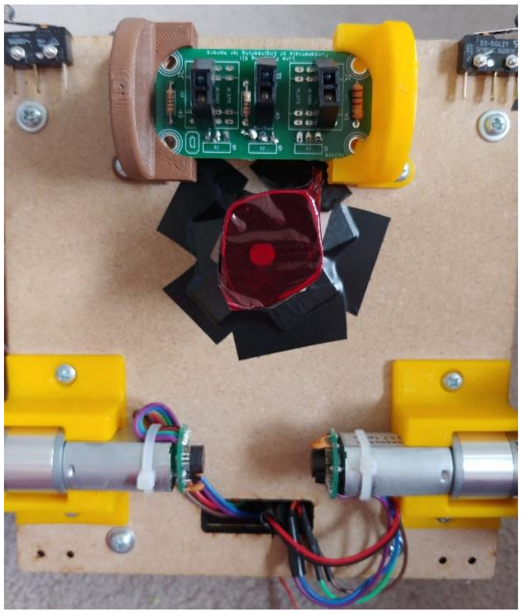


Figure 6: Picture of robot's underside showing the CDS cell, motors, and line following kit.

The CDS cell was able to detect the difference between the two colors of the light, however it had some inconsistencies. The range of values for the colors were fairly large, as the cell may see the light as a different brightness under different circumstances, such as difference in ambient light, a different positioning of the robot over the light, or variance between courses.

The red filter helped to differentiate the colors, however the ranges needed to be tweaked to guarantee that the colors were detected properly.

The chassis was not tall enough to allow the robot to push the button. A piece of cardboard was attached to the front of the robot and reinforced with tape, as shown in Figure 6 on the previous page. The cardboard added the necessary height in order to reach the button and was sturdy enough to press the button without collapsing or falling over.

The second Performance Test required the robot to deposit a tray into either the trash can or sink and slide the ticket across its track. In order to deposit the tray, an arm was fashioned out of cardboard to hold the tray, as seen in Figure 7 below. The arm was then fastened to an axle connected to a servo motor. The arm had enough clearance to turn roughly 45 degrees up or down. When the robot was powered on, the arm would angle itself upward to allow the tray to be placed on the arm, then when the robot reached the sink, it would angle the arm downward, causing the tray to fall into the sink.

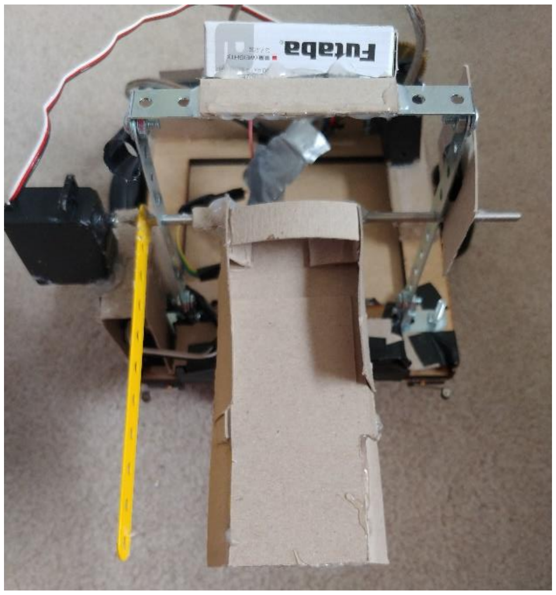


Figure 7: Picture of robot’s front arm.

Initially, the arm had issues holding the tray; the weight of the tray would push the arm down. At one point, the hot glue used to hold the arm onto its axle broke from the weight. The axle also had issues staying connected to the servo motor. Various methods were used to strengthen the axle's connection to both the arm and the motor. To hold the arm in place, superglue was applied to the axle, then hot glue was added to strengthen the connection. To connect the axle to the motor, superglue was applied to the axle and then shrink tubing was fitted over the connection and shrunk to hold the axle in place.

In order to slide the ticket, another arm was attached to the back of the robot, as seen in Figure 8 below. Another servo motor was modified to act as a DC motor, which lacked the precision of a servo motor, but allowed it to turn multiple rotations. A gear was attached to the motor, which allowed the arm to extend or retract. When the robot was properly aligned with the receipt, the arm would extend and the robot would drive forward until the receipt reached the end of the track, then finally the arm would retract allowing the robot to move independent from the receipt.

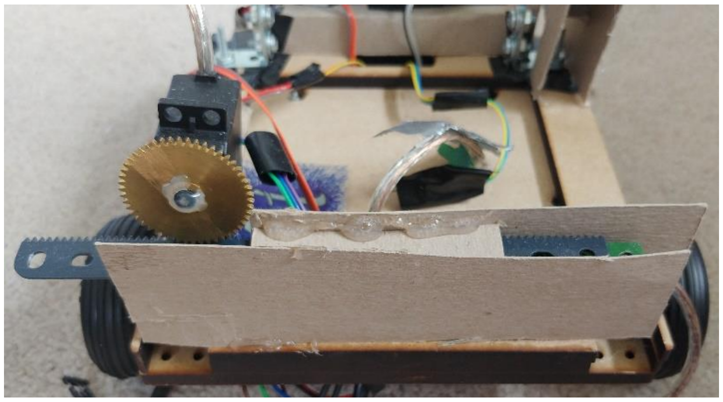


Figure 8: Picture of robot's back arm.

The back arm had inconsistencies when pushing the receipt. One such inconsistency was that the arm would not always extend or retract. The gear on the motor and the teeth on the arm were not the same shape, causing the arm to sometimes fall out of alignment with the gear and stop moving. This was solved by creating a cardboard track for the arm to slide down that would keep it aligned with the gear. The arm would also break the track if caught on an object. The track was reinforced with more cardboard; however, the problem was never solved. Whenever the track would break, it was hastily repaired.

For the third performance test, the robot needed to flip the burger patty to the other side of the stove. An erector set piece was attached to the same axle as the front arm, as seen in Figure 9 on the next page. The erector set piece acted as a longer front arm, which was able to reach the hot plate and flip the burger patty. To more easily move the hot plate from its resting position, the robot would move the arm upwards while turning to the right. After flipping it, the robot would move the arm a little further upwards, as not to catch the hot plate, and move a little further to the right. Then the arm would move back down, and touch the handle of the hot plate, flipping the hot plate back down if it got stuck in the up position.

Had the fourth Performance Test been held, it would require the robot to navigate to the correct ice cream lever, flip it down, then flip it up after seven seconds had passed. In order to navigate to the correct ice cream lever, three optosensors were attached to the robot, allowing it to determine if it was on a line. An algorithm was being developed to keep the robot moving across the line using the optosensors. Once the robot had reached the correct lever, it was planned to use the same arm that held the tray in order to flip the lever.

The erector set piece often had issues staying connected to its axle, as it was longer and moved a heavier object. Both attributed to the increased torque the arm experienced, causing it to sometimes break off from the axle. The arm was reinforced with superglue and hot glue to keep it connected to the axle. Figure 9 on the next page depicts the robot following the changes made from the third performance test. For a side view of the robot, see Figure C3 in Appendix C.

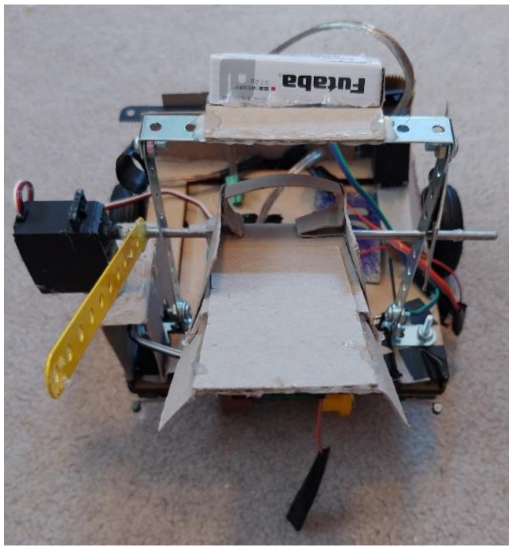


Figure 9: View of robot from front side.

4.4 Code

The first performance test required the robot to be able to navigate on its own, detect the color of a light on the course, and press the similarly colored button on the jukebox. In order to navigate on its own, the robot used the shaft encoding built into the IGWAN motors. Functions such as turnFor, driveForwardFor, and driveBackwardFor took an input in inches, then converted it into the counts of the motor, which could be used to make the motors run until it reached the specified distance.

In order to detect the color of the light on the course, the robot used a CDS cell, which would send a voltage between 0 and 3.3 Volts, with a higher voltage representing a lower brightness. The function determineLightValue would then compare the voltage to two ranges that contained one of the colors, then change the color of the touch screen to the color of the light. The robot used the color to determine its path toward the jukebox, pressing the button matching the light.

The second Performance Test required the robot to deposit the tray into either the sink or trash can and slide the ticket across its track. The arm holding the tray would turn with a servo motor. The motor could be adjusted with the member function SetDegree, which allowed the arm to deposit the tray into the sink. The back arm was extended and retracted by a motor, allowing it to reach the ticket. The functions extend and retract ran the back motor until the arm extended to its maximum length and retracted the arm to its minimum length, respectively.

The third Performance Test required the robot to flip the burger patty. The same motor that controlled the arm holding the tray also controlled a longer front arm that could reach the stove and flip the burger. The same member function setDegree was called to adjust the arm and flip the patty.

5. Future Development

This section covers the planned future developments to be made to the robot and code.

5.1 Code Development

The fourth Performance Test, had it been held, would have required the robot to navigate to the correct ice cream lever and flip it down, then flip it back up after seven seconds had passed. RPS was unavailable near the ice cream lever, requiring the robot to navigate using another method. The robot was equipped with three optosensors, which were planned to be used to follow lines and ensure the robot had navigated toward the correct lever. An algorithm was being developed, utilizing the optosensors to keep the robot on a line marked on the course, but was not finished. To finish the development, the time required would be a small amount as most of the code was already written. The code just needs to be tweaked to the course and robot's standards. The planned algorithm would check which optosensors could see the line and determine the robot's position relative to the line. The robot would then use that information to adjust its angle to stay on the line.

The same arm that had held the tray was planned to be used to flip the lever. Once again, setDegree was planned to lift the arm up, push the arm down, maneuver the arm under the lever, and finally push the lever back up after seven seconds had passed. In order to finish the code for the fourth Performance Test, approximately one hour would be necessary to transition the line following algorithm, and another would be necessary to test and debug the code.

The functions debugAllSensors and displayCoordinates were designed to aid in testing and debugging the robot. debugAllSensors displayed the information each sensor was receiving. This was called at the beginning of each run to ensure that all sensors were connected properly and gave appropriate values. displayCoordinates displayed the information RPS sent to the robot about its current position. The method would display the x and y positions as well as the direction the robot faced as an angle in degrees counterclockwise of the x-axis.

The functions turnTo and driveTo were being developed in order to utilize RPS to make navigation easier. turnTo was designed to accept an RPS coordinate, then calculate the angle it needed to turn in order to face it, then turn until it reached that angle. driveTo was designed to call turnTo in order to face the desired coordinate then calculate the distance between its current position and the destination. The robot would then drive until it reached the desired coordinate. These functions never reached implementation as bugs in the code prevented the team from doing so. Table A1 lists the user defined functions that were implemented or were meant to be implemented and describes what each function was meant to do. Figure A2 provides a flowchart that depicts how the team planned on coding the robot, using the user defined functions from Table A1, to complete the course. These functions still needed to be debugged, roughly one hour for each of them would be necessary to finish debugging.

5.2 Hardware Development

After assembling the chassis of the robot, a problem noticed by the team, but of not too much concern at the time, was that the adapters that connected the wheels to the motor shafts did not have a strong connection neither to the wheel nor the shaft. Because of this, the wheels kept falling off the adapters, or the adapters off the shafts. The solution at the time was using hot glue to connect the wheels to the adapters. That had fixed the problem between the wheels and the adapters. But later, the hot glue did not maintain its hold between the two parts and so the wheel started coming off again. So, the team used super glue. When this worked well, the team also decided to go ahead and do the same between the shaft and the adapter. But due to poor gluing quality to the shaft, one of the wheels ended up at an angle with the ground that was not 90^{o}. Therefore, one future development that should be made is the correction of the wheel's slant. This way, the robot will not move unpredictably and will instead be more accurate and repeatable. This development would take a short amount of time as the adapter would only have to be removed from the shaft, and then re-glued back onto the shaft in a more careful fashion.

The next area that would require further development is the back arm. One problem with the back arm is that the connection between the gear and the toothed rack strip is too wide. Because of this, the rack strip would not extend out or in properly. Therefore, the fix that would be made to the back arm is to tighten the gap between the rack strip and the gear. To do so, the stand of cardboard on which the motor for the gear stood would need to be moved closer to the rack strip. But not so close that it would get in the path of the strip and cause unneeded friction. So, if the gear needed to be moved closer to the strip still, then the motor itself would be moved closer towards the strip. This would not take long at all as the stand can easily be removed, along with the motor, because they were glued in place using hot glue. The stand may have to end up being remade though, depending on how much damage it receives from being removed.

Another problem with the back arm was that the carboard walls on either side of the stand that the rack strip sat on were too close to the strip, and the hot glue used had managed to get on the stand. Because of this, the rack strip was experiencing too much friction, further limiting its extending and retracting capabilities. To fix this, the cardboard walls would have to be removed and the excess hot glue removed. Then the walls would have to be re-glued, carefully, onto the stand. And if the walls were damaged in the removal, then the walls would also need to be remade. This development's time requirement, like the last, depends on how badly damaged the walls are. If they are badly damaged, they would have to be remade. Other than that, because the walls only require re-gluing carefully, the task would not long.

Lastly, the front arm was too fragile to withstand testing and perform reliably. This was due to the connection between the axel, the part that the arm feature attached to, and the gear of the motor was fragile. The front arm's axel was attached to the gear of the motor by first filing down one end of the axel, and then inserting it into the hole of the gear. After that, the connection was cemented using hot glue. But from testing the arm, the connection between the gear and the axel was too slipper. Because the axel was made from metal, the hot glue did not want to stick to it. To remedy this, the team attempted to use epoxy instead, as epoxy would have been more solid and held tighter. But because the epoxy never dried properly between the plastic and the metal, the epoxy would not hold either. As a last attempt, the team used super glue as well, but to no avail.

Therefore, based on the previously failed attempts to properly connect the front arm axel to the motor’s gear, the team had decided the best method would be to improve the connection using the key method, where the gear and axel are formed, molded, and filed down in such a way that the two parts fit together perfectly. This would improve the fit and connection between the two parts, and hopefully fixing the strength of the front arm. This development would take the most time out of all the Hardware Developments, taking upwards of an hour or two, because the front arm would have to be taken completely apart. From there, the axel and gear would have to be shaped in just the right fashion that they match each other’s shape and fit together snuggly. Then the arm would have to be reassembled, making sure any glue used in the reassembly of the arm does not get stuck to the stand that holds the arm’s axel up.

If these developments are incorporated into the robot, the robot would meet a complete prototype state. With the front arm and back arm fixed and developed, the mechanisms for completing each task will be ready to take on the full course. And with the improvement of the wheel’s positioning, the robot will be able to move in a more predictable, repeatable, and reliable fashion. Therefore, making the hardware aspect of the robot ready for the course.

6. Summary and Conclusions

The team was assigned the job of creating a robot that would complete kitchen tasks for the company Carmen’s Diner. Before creating the full robot, the team created a fully functioning robot prototype of the size 9” x 9” x 12”. A model course was built by a research team at The Ohio State University for testing on.

The required tasks for the robot to complete included pushing a button on a jukebox depending on the color of a light nearby. The robot also had to place a tray in a sink, or on the top of a trash bin. Other tasks included pulling an ice cream lever and waiting seven seconds before pushing it back up, flipping a burger over from a hot plate, and pushing a receipt down a ticket rack. And finally, the robot was to return to its starting position and press a button. The robot would have two minutes in total to complete these tasks.

To start the project process, each team member brainstormed multiple different robot designs and ideas. These designs involved separate models and designs of drivetrains, chassis, and task completion mechanisms. The selected design was to have a triangular chassis, two motors for the drivetrain using omni wheels, and a single complex arm on the front. But after careful reconsideration due to problems with the design and team preferences, the design was changed. The final selected design was a robot that would have a rectangular chassis, two normal wheels on two motors for the drivetrain, and two separate arms. One front arm and one back arm. The team also decided to include a three optosensor line following circuit with a CDS cell on the bottom of the robot.

The path the team had planned on running the robot was decided to be in an almost figure eight fashion. To start, the robot would head up the ramp and place the tray in the sink. From there, the robot would head to the ticket and push it all the way over. Then the robot would move to the burger and flip it. Following that, the robot would head to the ice cream machine and flip the lever. Lastly, the robot would head back down the ramp to the jukebox and press either the red or the blue button, before heading back to the start to press the final button for course completion.

For prototype construction, the team was given a starting budget of $160 for buying parts to build. To buy needed parts, the team used the FEH Robot Store, as all parts, unless checked with the team's GTA, were to come from the Robot Store [7]. Over the course of the project, the team spent a total of $144.11 and had $15.89 remaining for any other parts should they be needed.

The plan for the code was to develop multiple user defined functions that would allow the construction of code to flow smoother. These such functions included one that would have the robot turn a requested angle amount either left or right. One that would have the robot move forward a requested distance and one that would move backward a request distance. And functions that would connect with the RPS to help check the robot's position and angle to make sure it was where it was supposed to be. But due to unforeseen events, the project was halted and the RPS functions could not be finished. The robot was to also use a function that helped it follow the lines on the course in conjunction with the other functions but ended up incomplete due to the halted project.

The robot ultimately had its strengths and its weaknesses. The first of its strengths, the code was workable in that it was easily tweaked between tests, allowing changes to be made wherever and whenever they were needed. Adding to this list of strengths, the chassis proved a solid base for the robot, facing no problems during testing. Finally, the drivetrain's repair was successful in that the team did not see the wheels detach after they were secured with glue. The glue, however, led to the first weakness of the robot: its poor navigation. Due to the aforementioned subpar glue job, the robot's movement was difficult to predict, requiring extensive testing and tweaking to achieve the route that the team wanted. The arm mechanisms were also delicate, not only requiring extensive tweaking but also constant repairs and adjustments to function. Due to these constant changes, the routes the robot took were often unrepeatable, requiring even more tweaking even in the wake of a successful test.

Since the project was halted, the team was unable to finish the wanted work on the robot. This work included both software and hardware aspects. On the software side, multiple functions were left unfinished and others with bugs. On the hardware side, multiple aspects of the robot needed fixes or developments. This included the front arm needing repairs and reworking, the back arm needed adjustments made to it, and one of the robot's wheels required repositioning. If the team were given more time, these adjustments and developments may have been made and the robot would have been a finished prototype.

7. References

[1] Robot Scenario. 2020, February 24. www.carmen.osu.edu

[2] Robot Course CAD. 2020, February 24. www.carmen.osu.edu

[3] Robot Performance Tests. 2020, February 24. www.carmen.osu.edu

[4] Robot Scenario. 2020, April 6. www.carmen.osu.edu

[5] Pre_R03 – Drivetrain Calculations. 2020, April 6. www.carmen.osu.edu

[6] FEH Motors Graph. 2020, April 6. www.carmen.osu.edu

[7] FEH Robot Store. 2020, April 8. www.feh.osu.edu/store

APPENDIX A
Code

This appendix provides the final code.

Performance Test 3 Code

```
#include <FEHLCD.h>

#include <FEHUtility.h>

#include <FEHIO.h>

#include <FEHServo.h>

#include <FEHMotor.h>

#include <LCDColors.h>

#include <FEHRPS.h>

#include <math.h>

/*

* Left and Right global variables (For Readability)

*/

#define LEFT false

#define RIGHT true

#define INCH 1


/*

* Drivetrain motors

*/

FEHMotor rightMotor(FEHMotor::Motor1, 9.0);

FEHMotor leftMotor (FEHMotor::Motor0, 9.0);


/*

* Motor digital encoders

*/

DigitalEncoder right_encoder(FEHIO::P0_0);

DigitalEncoder left_encoder(FEHIO::P2_7);


/*

* Line follower sensors

*/

AnalogInputPin rightLine(FEHIO::P1_5);
AnalogInputPin middleLine(FEHIO::P1_6);

AnalogInputPin leftLine(FEHIO::P1_7);


/*

* CDS Cell sensor

*/

AnalogInputPin cdsCell(FEHIO::P1_4);


/*

* Front bump sensors

*/

DigitalInputPin frontRightBump(FEHIO::P0_3);

DigitalInputPin frontLeftBump(FEHIO::P0_6);


/*

* Back bump sensors

*/

DigitalInputPin backRightBump(FEHIO::P0_4);

DigitalInputPin backLeftBump(FEHIO::P0_5);


/*

* Servo motor

*/

FEHServo frontArm(FEHServo::Servo0);


/*

* Hacked servo motor

*/

FEHMotor backArm(FEHMotor::Motor2, 5.0);


/*

* Scales to adjust the motor speed in order to get the robot to drive straight

*/
```

```
float leftMotorScale = 1.003;

float rightMotorScale = 1.011;

float leftTurnAdjust = 5;

float rightTurnAdjust = 5;


/*

* The average speed we want our robot to go at all times

*/

int normalSpeed = 50;

int turningSpeed = 25;


/*Multiplier for retracting backArm

* (retracts a little less than it extends)

*/

float backArmMotorRetractMultiplier = 1.047;


/*Constant for backArm motor power*/

int backArmMotorPower = 50;


/*

* Code to make sure each sensor is working properly

*/

void debugAllSensors() {

float x, y;

while (!LCD.Touch(&x, &y)) {

  LCD.Clear(BLACK);

        LCD.Write("Right Line Sensor: ");    LCD.WriteLine(rightLine.Value());

        LCD.Write("Middle Line Sensor: ");   LCD.WriteLine(middleLine.Value());

        LCD.Write("Left Line Sensor: ");     LCD.WriteLine(leftLine.Value());


        LCD.Write("CDS Cell Value: ");       LCD.WriteLine(cdsCell.Value());


        LCD.Write("Front Bumper Right: ");   LCD.WriteLine(frontRightBump.Value());
        LCD.Write("Front Bumper Left: ");    LCD.WriteLine(frontLeftBump.Value());


        LCD.Write("Back Bumper Right: ");    LCD.WriteLine(backRightBump.Value());

        LCD.Write("Back Bumper Left: ");     LCD.WriteLine(backLeftBump.Value());

}

}


/*

* Code to move the robot (hopefully) straight forward

*/

void driveForward(int speed = normalSpeed) {

leftMotor.SetPercent(speed * leftMotorScale);

rightMotor.SetPercent(speed * rightMotorScale);

}


/*

* Code to move the robot (hopefully) straight backwards

*/

void driveBackwards(int speed = normalSpeed) {

leftMotor.SetPercent(speed * leftMotorScale * -1);

rightMotor.SetPercent(speed * rightMotorScale * -1);

}


/*

* Code to make both motors stop

*/

void stop() {

leftMotor.Stop();

rightMotor.Stop();

}


/*

* Get and store the value of light from the CDS Cell
```

```
*/

float getLightValue() {

 float cdsCellVal = cdsCell.Value();

 return cdsCellVal;

}


/*

* Determine what the color of the light is and display on the screen

*/

int determineLightValue(float val) {

 if (val < 0.40) {

     LCD.Clear(RED);

     return 333;

 } else if (1.25 > val && val > 0.75) {

     LCD.Clear(BLUE);

     return 999;

 } else {

     LCD.Clear(BLUE);

     LCD.WriteLine("BLUE");

     LCD.WriteLine(cdsCell.Value());

 }

 return 0;

}


/*

* Turn in some direction (LEFT or RIGHT) for some amount of degrees

*/

void turnFor(float degrees, bool direction, int speed = turningSpeed)

{

if(direction)

{

   degrees += rightTurnAdjust;

}
```

```
else

{

    degrees += leftTurnAdjust;

}

float inches = (degrees / (360.0)) * 25.13;

float counts = inches * 41.67;


left_encoder.ResetCounts();

right_encoder.ResetCounts();


if (!direction) {

     rightMotor.SetPercent(speed * rightMotorScale);

     leftMotor.SetPercent(speed * leftMotorScale * -1);

} else if (direction) {

     rightMotor.SetPercent(speed * rightMotorScale * -1);

     leftMotor.SetPercent(speed * leftMotorScale);

} else {

     LCD.WriteLine("I refuse to turn!");

}


while((left_encoder.Counts() + right_encoder.Counts()) / 2 < counts);

stop();

}


/*

* Code to drive forward a specific distance

*/

void driveForwardFor(float inches, int speed = normalSpeed)

{

LCD.Clear();

float counts = inches * 41.67;

left_encoder.ResetCounts();

right_encoder.ResetCounts();
```

```
driveForward(speed);


while((left_encoder.Counts() + right_encoder.Counts()) / 2.0 < counts);

stop();

LCD.Write(counts / 41.67);   LCD.WriteLine(" inches");

}


/*

* Code to drive backwards a specific distance

*/

void driveBackwardsFor(float inches, int speed = normalSpeed)

{

float counts = inches * 41.67;

left_encoder.ResetCounts();

right_encoder.ResetCounts();


driveBackwards(speed);


while((left_encoder.Counts() + right_encoder.Counts()) / 2.0 < counts);

stop();

}


/*

* Code to extend backArm

*/

void extend(){

 backArm.SetPercent(backArmMotorPower);

 Sleep(2.25);

 backArm.Stop();

}


/*
```

```
* Code to retract backArm

*/

void retract(){

 backArm.SetPercent(-backArmMotorPower * backArmMotorRetractMultiplier);

 Sleep(2.25);

 backArm.Stop();

}


/*

* Code to make the robot turn towards a specific point

*/


void turnTo(float x, float y, int speed = turningSpeed)

{

   double angle;

   if(x - RPS.X() == 0)

   {

       angle = y > RPS.Y() ? 90 : 270;

   }

   else if(y - RPS.Y() == 0)

   {

       angle = x > RPS.X() ? 0 : 180;

   }

   else

   {

       angle = tan((y - RPS.Y()) / (x - RPS.X()));

   }


   double turnAngle = RPS.Heading() - angle;


   turnFor(turnAngle, abs(turnAngle < 180), speed);

}
```

```
/*

* Code to make the robot drive to a specific point

*/

void driveTo(float x, float y, int speed = normalSpeed, int turnSpeed = turningSpeed)

{

   double xDistance = x - RPS.X();

   double yDistance = y - RPS.Y();

   double distance = pow(xDistance, 2) + pow(yDistance, 2);


   turnTo(x, y, turnSpeed);


   driveForwardFor(distance, speed);

}


bool lineFollow(){

    {

        float x,y;


        bool lineDetected = true;

        bool leftDetected = false;

        bool centerDetected = false;

        bool rightDetected = false;


        LCD.Clear(FEHLCD::Black);

        LCD.SetFontColor(FEHLCD::White);


        /*Sensor test code

         * .Write("Left: "); LCD.WriteLine(leftLine.Value());

            LCD.Write("Center: "); LCD.WriteLine(centerLine.Value());

            LCD.Write("Right: "); LCD.WriteLine(rightLine.Value());


            LCD.Clear(FEHLCD::Black);

        */
        //int onOFF = 0;
    //while(!LCD.Touch(&x, &y)){}

        while(lineDetected)
        {
            /*reinitialize detector booleans*/
            leftDetected = false;
            centerDetected = false;
            rightDetected = false;

            /*if left optosensor detects line, turn true*/
            if(leftLine.Value() > 1.5){
                leftDetected = true;
            }

            /*if center optosensor detects line, turn true*/
            if(centerLine.Value() > 0.9){
                centerDetected = true;
            }

            /*if right optosensor detects line, turn true*/
            if(rightLine.Value() > 1.2){
                rightDetected = true;
            }

            if (leftDetected && centerDetected){
                rightMotor.SetPercent(20);
                leftMotor.SetPercent(30);
            }
            else if(rightDetected && centerDetected){
                rightMotor.SetPercent(30);
                leftMotor.SetPercent(20);
            }
            else if(centerDetected){
                rightMotor.SetPercent(25);
                leftMotor.SetPercent(25);
            }
            else if(leftDetected){
                rightMotor.SetPercent(15);
                leftMotor.SetPercent(50);
            }
            else if(rightDetected){
                rightMotor.SetPercent(50);
                leftMotor.SetPercent(15);
            }
            else {
                lineDetected = false;
            }
    //------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
    ---------------
    int main(void)
    {
      float x,y;
      LCD.Clear();

      /*
       * Calibrate the servo motor
       * Max: 2410
       * Min: 500
       */
      frontArm.SetMin(500);
      frontArm.SetMax(2410);

      /*
       * Debug mode
       * 0 is straight up
```

```
 * 90 is parallel to the ground

 * 180 is straight down

 */


frontArm.SetDegree(30.);

debugAllSensors();


/*

 * Wait for light to start

 */

while(cdsCell.Value() > 1.5);


*

 * Drive forward a bit, turn to the left 90 degrees

 */

driveForwardFor(4 * INCH);

turnFor(90, LEFT);


/*

 * Drive forward to center of ramp, turn right 90 degrees

 */

driveForwardFor(7.25 * INCH);

turnFor(90, RIGHT);


/*

 * Drive up the ramp

 */

driveForwardFor(12 * INCH);

driveForwardFor(12 * INCH, 90);

driveForwardFor(10 * INCH);

/*

 * Drive to the stove

 */
```

```
  /*
   * Flip the burger patty
   */
  /*
   * Drive to the beginning of the ice cream line
   */
  /*
   * Follow the line to the lever
   */
  /*
   * Flip the lever
   */
  return 0;
}
```

Table A1: This is where the table mentioning all the code functions with their descriptions, returns, parameters, and names will go.

Function Name	Parameters	Returns	Description	Function Creation was Complete or Started
turnFor	float degrees bool direction int speed	void	Have the robot turn in the requested direction for the specified amount	Started
driveBackwards	int speed	void	Have the robot drive forward	Complete
driveForward	int speed	void	Have the robot drive backwards	Complete
stop		void	Stop the robot's wheels	Complete
driveForwardFor	float inches int speed	void	Have the robot drive forward for the specified amount	Started
driveBackwardsFor	float inches int speed	void	Have the robot drive backwards for the specified amount	Started
getLightValue		float	Get a value from the CDS cell	Complete
determineLightValue	float val	int	Take the CDS cell's value and determine of the color is red or blue and display the color on the proteus screen	Complete
extend		void	Move the back arm out	Complete
retract		void	Move the back arm back in	Complete
debugAllSensors		void	Displayed the current output from each of the sensors onto the proteus screen to aid the team in determining if all sensors were working properly	Complete
displayCoordiantes		void	Displayed the coordinates on the proteus screen that the RPS sent to the robot	Complete
turnTo	float x float y int speed	void	Would allow the robot to read RPS and turn the front of the robot in a certain direction based on the RPS coordinates	Started
driveTo	float x float y int speed int tunrSpeed	void	Would allow the robot to read RPS and turn in the needed direction, followed by driving forward or backward based on the RPS coordinates	Started
followLine	int speed	void	Allows the robot to follow a line in the direction the robot is currently facing	Started
getIceCreamLever		int	Returns the value the robot requires to determine which ice cream lever the robot should flip	Not Started

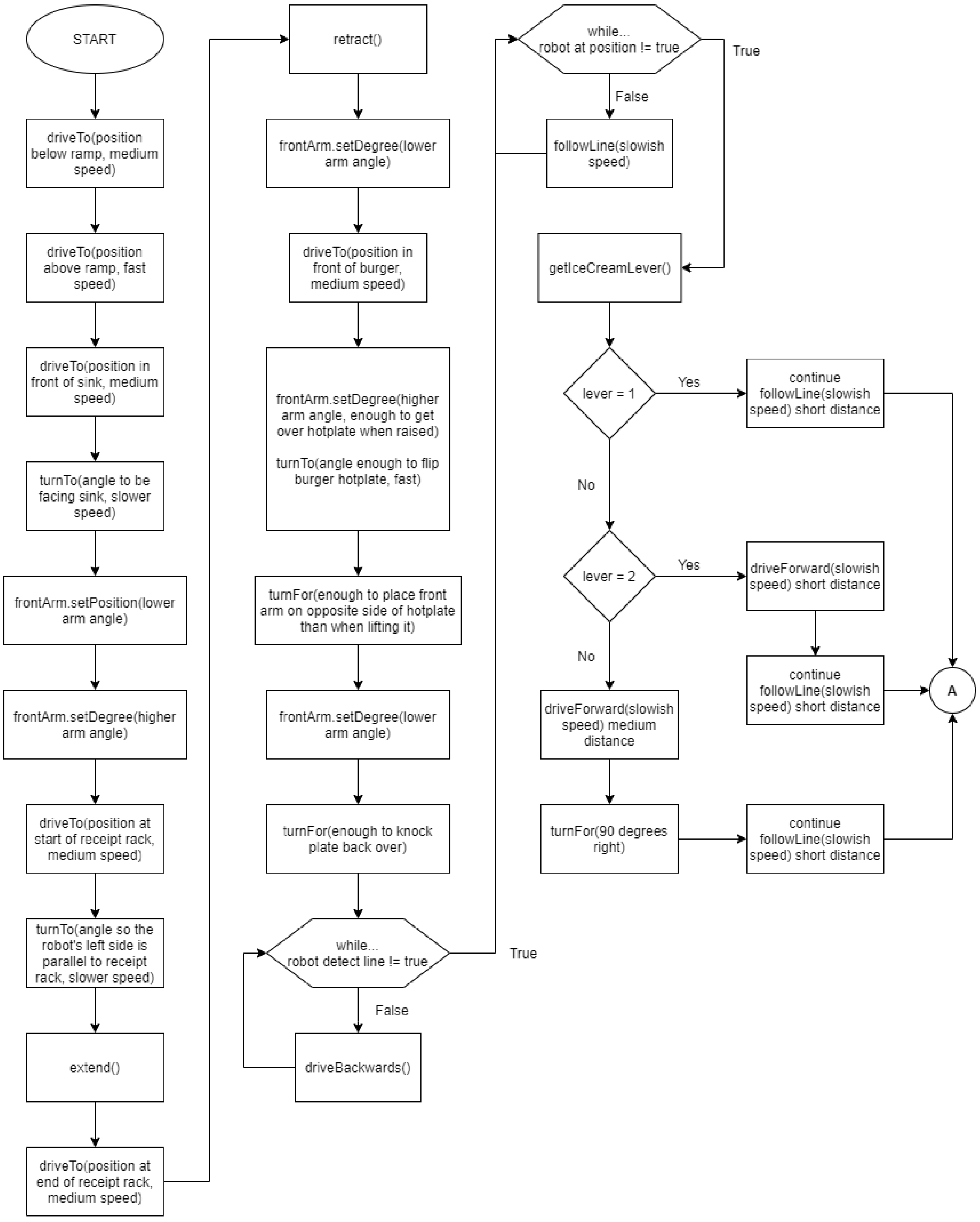

START
driveTo(position below ramp, medium speed)
driveTo(position above ramp, fast speed)
driveTo(position in front of sink, medium speed)
turnTo(angle to be facing sink, slower speed)
frontArm.setPosition(lower arm angle)
frontArm.setDegree(higher arm angle)
driveTo(position at start of receipt rack, medium speed)
turnTo(angle so the robot's left side is parallel to receipt rack, slower speed)
extend()
driveTo(position at end of receipt rack, medium speed)
retract()
frontArm.setDegree(lower arm angle)
driveTo(position in front of burger, medium speed)
frontArm.setDegree(higher arm angle, enough to get over hotplate when raised)
turnTo(angle enough to flip burger hotplate, fast)
turnFor(enough to place front arm on opposite side of hotplate than when lifting it)
frontArm.setDegree(lower arm angle)
turnFor(enough to knock plate back over)
while... robot detect line != true
True
False
driveBackwards()
while... robot at position != true
True
False
followLine(slowish speed)
getIceCreamLever()
lever = 1
Yes
continue followLine(slowish speed) short distance
No
lever = 2
Yes
driveForward(slowish speed) short distance
continue followLine(slowish speed) short distance
No
driveForward(slowish speed) medium distance
turnFor(90 degrees right)
continue followLine(slowish speed) short distance
A

Figure A1: Flowchart depicting the choices and code overview the robot will do for the decided run path.

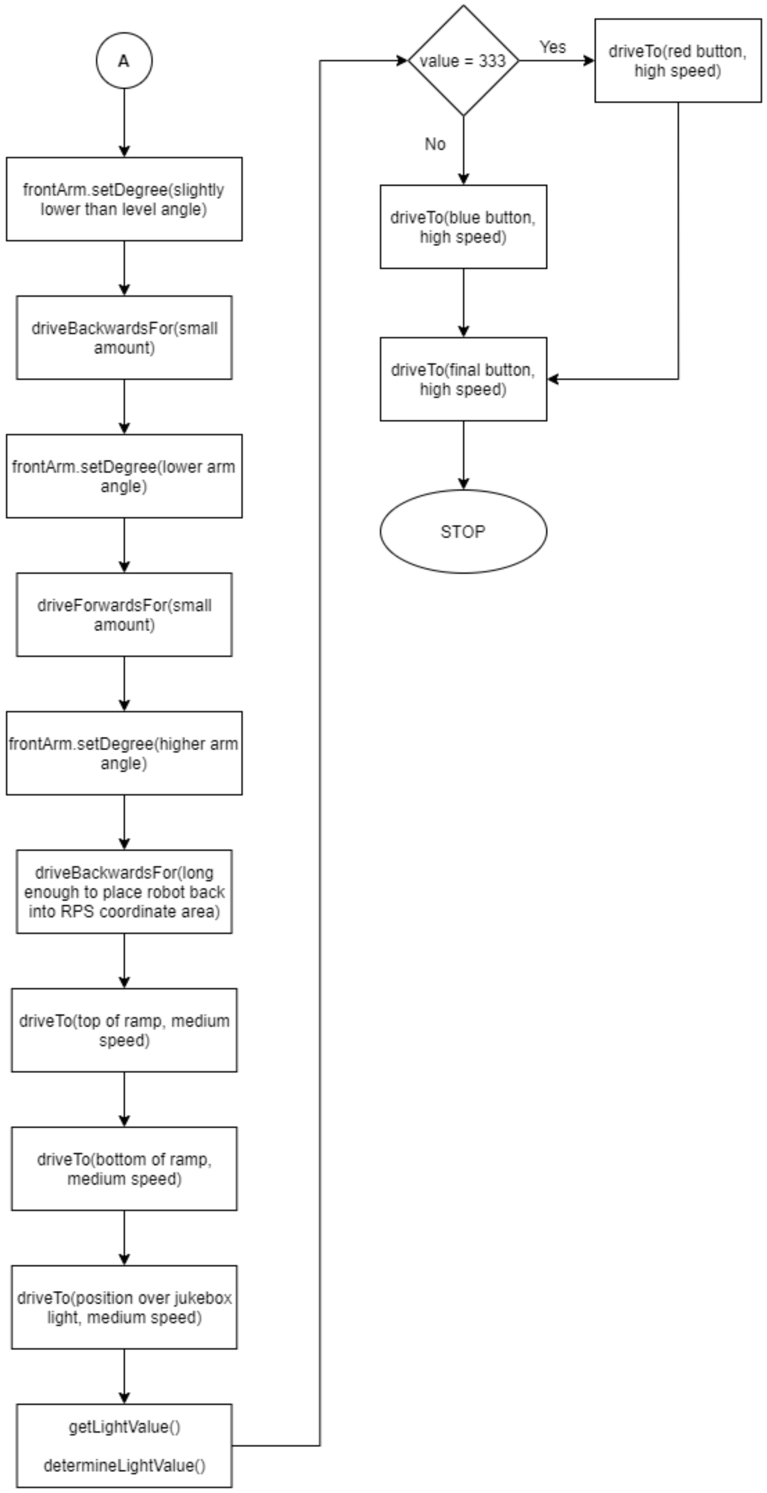
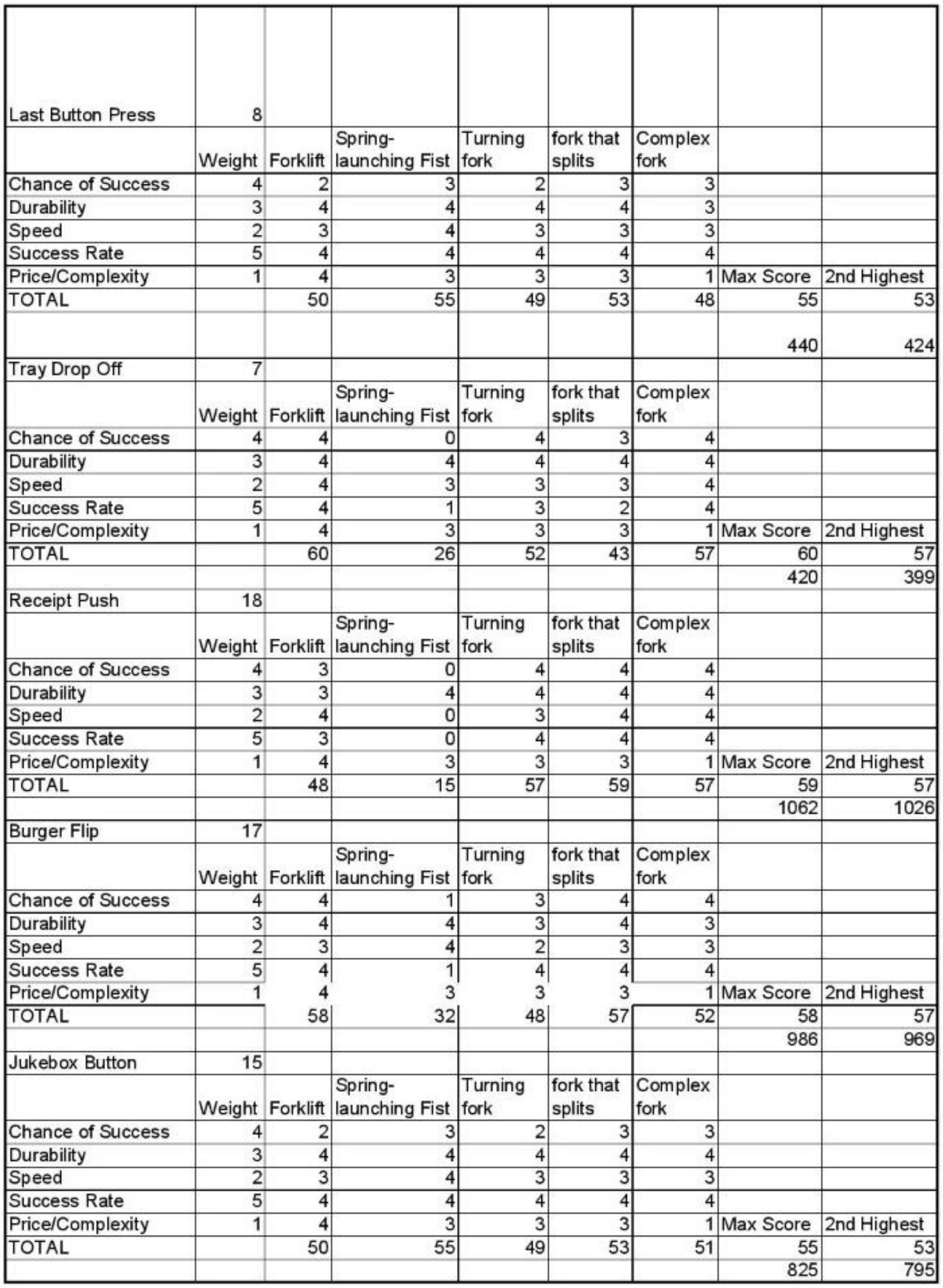

A
frontArm.setDegree(slightly lower than level angle)
driveBackwardsFor(small amount)
frontArm.setDegree(lower arm angle)
driveForwardsFor(small amount)
frontArm.setDegree(higher arm angle)
driveBackwardsFor(long enough to place robot back into RPS coordinate area)
driveTo(top of ramp, medium speed)
driveTo(bottom of ramp, medium speed)
driveTo(position over jukebox light, medium speed)
getLightValue()
determineLightValue()
value = 333
Yes
driveTo(red button, high speed)
No
driveTo(blue button, high speed)
driveTo(final button, high speed)
STOP

Figure A2: Continuation of flowchart from Figure A1 on the last page.

APPENDIX B
Decision Matrices

This appendix provides the decision matrices.

Table B1: Decision Matrix



Last Button Press	8							
	Weight	Forklift	Spring-launching Fist	Turning fork	fork that splits	Complex fork		
Chance of Success	4	2	3	2	3	3		
Durability	3	4	4	4	4	3		
Speed	2	3	4	3	3	3		
Success Rate	5	4	4	4	4	4		
Price/Complexity	1	4	3	3	3	1	Max Score	2nd Highest
TOTAL		50	55	49	53	48	55	53
							440	424
Tray Drop Off	7							
	Weight	Forklift	Spring-launching Fist	Turning fork	fork that splits	Complex fork		
Chance of Success	4	4	0	4	3	4		
Durability	3	4	4	4	4	4		
Speed	2	4	3	3	3	4		
Success Rate	5	4	1	3	2	4		
Price/Complexity	1	4	3	3	3	1	Max Score	2nd Highest
TOTAL		60	26	52	43	57	60	57
							420	399
Receipt Push	18							
	Weight	Forklift	Spring-launching Fist	Turning fork	fork that splits	Complex fork		
Chance of Success	4	3	0	4	4	4		
Durability	3	3	4	4	4	4		
Speed	2	4	0	3	4	4		
Success Rate	5	3	0	4	4	4		
Price/Complexity	1	4	3	3	3	1	Max Score	2nd Highest
TOTAL		48	15	57	59	57	59	57
							1062	1026
Burger Flip	17							
	Weight	Forklift	Spring-launching Fist	Turning fork	fork that splits	Complex fork		
Chance of Success	4	4	1	3	4	4		
Durability	3	4	4	3	4	3		
Speed	2	3	4	2	3	3		
Success Rate	5	4	1	4	4	4		
Price/Complexity	1	4	3	3	3	1	Max Score	2nd Highest
TOTAL		58	32	48	57	52	58	57
							986	969
Jukebox Button	15							
	Weight	Forklift	Spring-launching Fist	Turning fork	fork that splits	Complex fork		
Chance of Success	4	2	3	2	3	3		
Durability	3	4	4	4	4	4		
Speed	2	3	4	3	3	3		
Success Rate	5	4	4	4	4	4		
Price/Complexity	1	4	3	3	3	1	Max Score	2nd Highest
TOTAL		50	55	49	53	51	55	53
							825	795

Table B1 (continued): Decision Matrix

Softserve Machine	27							
	Weight	Forklift	Spring-launching Fist	Turning fork	fork that splits	Complex fork		
Chance of Success	4	4	0	3	4	4		
Durability	3	4	1	3	3	3		
Speed	2	4	4	4	4	4		
Success Rate	5	3	0	3	3	3		
Price/Complexity	1	4	3	3	3	1	Max Score	2nd Highest
TOTAL		55	14	47	51	49	55	51
							1485	1377
WEIGHTED TOTALS		688	502	654	738	705		
Drivetrain / Chassis	84							
	Weight	Omni / Squar	Omni / Triangle	Omni / Slider /	Omni / Slider /	4 Two-Direction		
Control	5	2	5	4	4	3		
Durability	4	2	4	4	4	4		
Speed	2	3	4	3	3	2		
Price/Complexity	1	3	3	3	3	4	Max Score	2nd Highest
TOTAL		27	52	45	45	39	52	45
							4368	3780

	Overall Design Decision Matrix		Splitting Forklift + Triangular Chassis + 3 Omniwheels	Splitting Forklift + Square Chassis + Omniwheels and slider	Complex Lift + Triangular Chassis + Omniwheel
		Weight			
	Last button	8	49	49	48
	Tray Drop Off	7	52	52	57
	Receipt Push	18	57	57	57
	Burger Flip	17	48	48	52
	Jukebox Push	15	49	49	51
	Softserve	27	47	47	49
	DriveTrain/Chassis	84	52	45	52
	TOTAL		8970	8382	9149

APPENDIX C
Robot Pictures and Diagrams

This appendix provides the pictures of the physical mock-up.

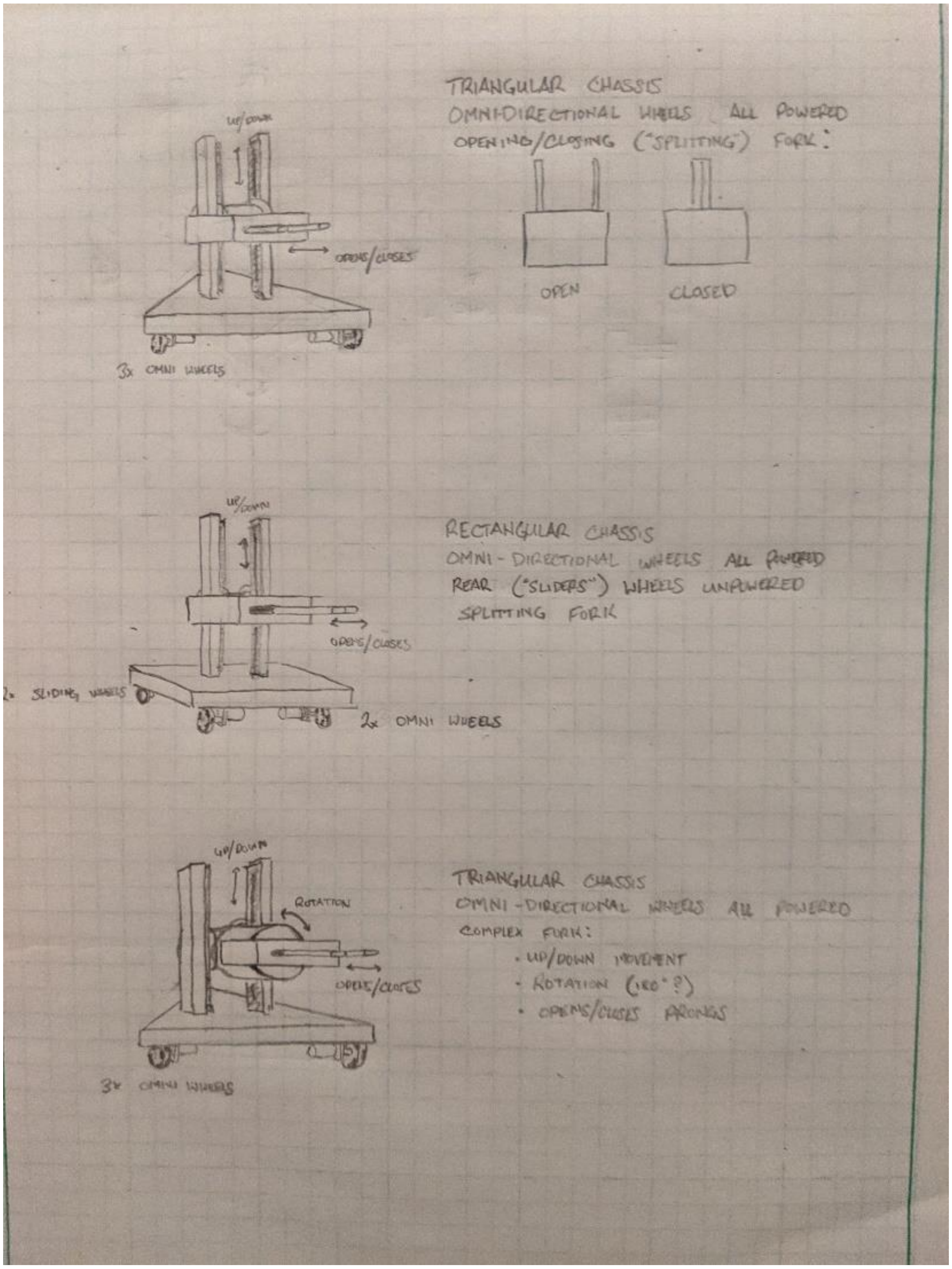


Figure C1: Three Preliminary Robot Designs.

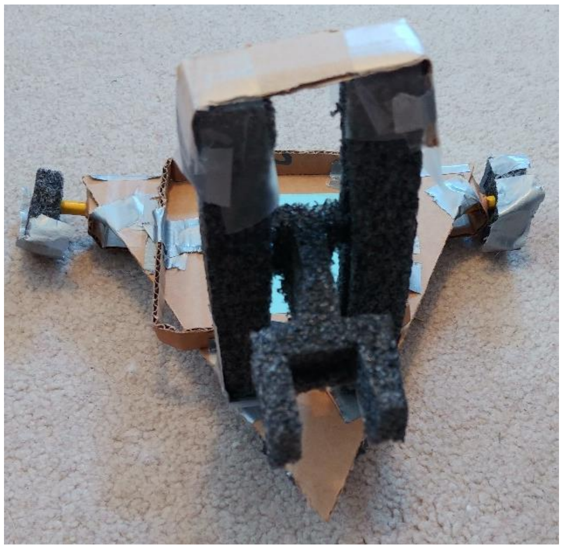


Figure C2: Robot Mockup Front View.

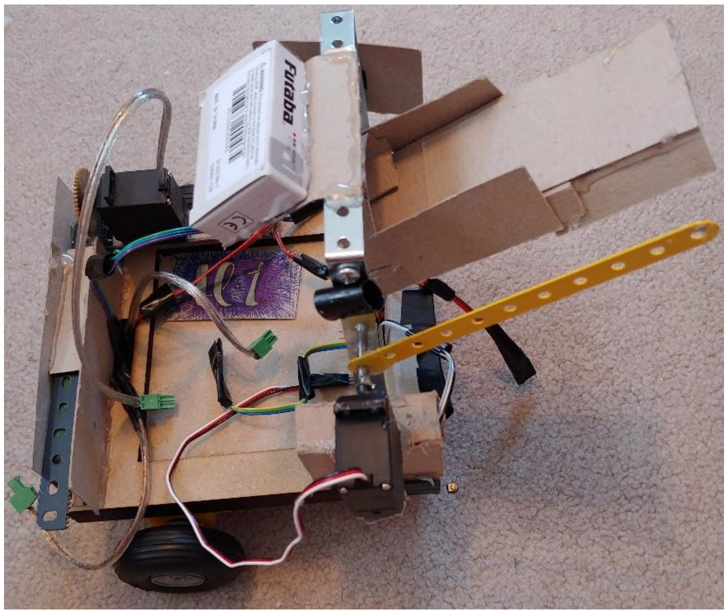


Figure C3: Side View of Final Robot Design.

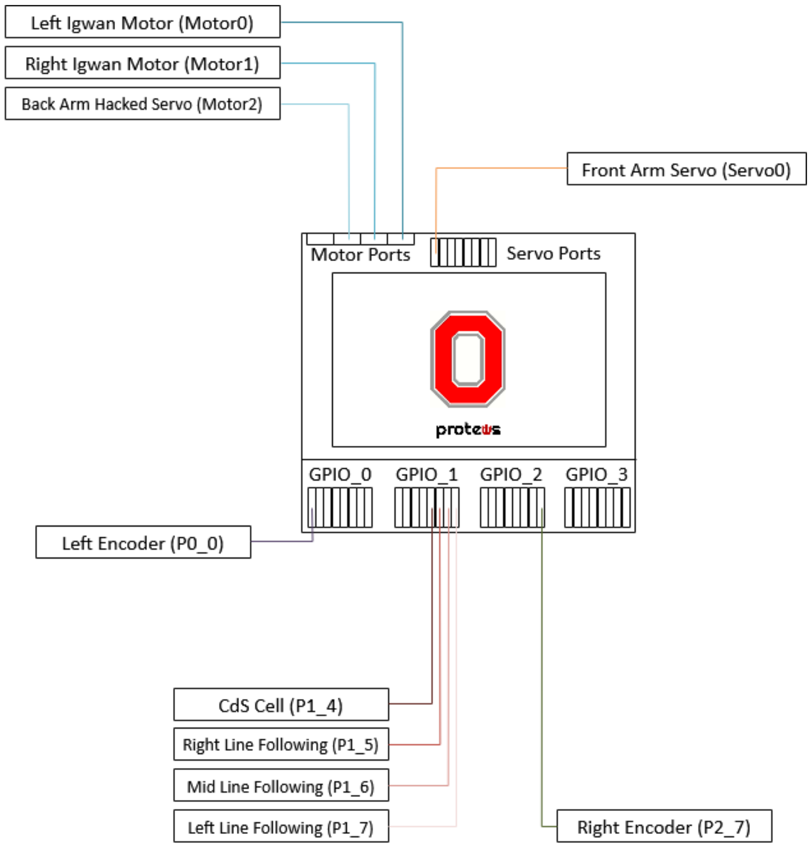

Left Igwan Motor (Motor0)
Right Igwan Motor (Motor1)
Back Arm Hacked Servo (Motor2)
Front Arm Servo (Servo0)
Motor Ports
Servo Ports
proteus
GPIO_0
GPIO_1
GPIO_2
GPIO_3
Left Encoder (P0_0)
CdS Cell (P1_4)
Right Line Following (P1_5)
Mid Line Following (P1_6)
Left Line Following (P1_7)
Right Encoder (P2_7)

Figure C4: Electrical diagram of proteus ports and what sensors/motors were plugged into which ports.

APPENDIX D
Budget and Time Sheets

This appendix provides tables showing the time spent on various project areas and the budget breakdown.

Table D1: A list of items used or bought for the robot, the quantity bought, and the total cost of all parts bought.

Item	Quantity	Cost
2 ft Length of Solder (Lead Based)	1	1.00
Red Filter	1	0.10
Motor Wire 2-Conductor 18AWG (1 ft)	2	0.40
Terminal Block for DC Motors	3	3.00
10 Pack #8 Screws (.38"), Nuts, Washers	2	0.40
Heat Shrink Tubing for Motor Wire (1 in)	14	0.28
Male Header Strip (36 Pin)	1	0.60
Ribbon Cable (1 ft)	1	0.50
3D Printed Chassis Mount for IGWAN Motor	2	5.00
DuBro 250T 2.5" Wheel	2	5.20
DuBro Wheel Adapter for IGWAN Shaft- For 2.5" Wheel	2	2.00
IG-22 IGWAN Motor	2	50.00
3D Printed Slider	2	4.60
4 Pack, 16mm Screws w/ Washers, M2x16	2	0.10
6 Pack of #6 Screws (1.5"), Nuts, Washers	2	0.40
Optosensor (100 Ohm and 10 kOhm Resistors)	2	1.54
Line Following Circuit Board (Optosensors Not Included)	1	1.00
Microswitch, Roller Blade (Medium)	3	3.60
Chassis (3 * 1/4 Inch Layers of MDF)	1	16.54
Epoxy Packet w/ Stick and Cup	3	2.64
8.5" x 11" Cardboard Sheet	3	0.12
1x1" Angle Bracket w/ 2x2 Hole	4	4.00
6.5" Axle Rod	1	1.24
5.5 x 0.5" Double Angle Strip w/ 1 x 11 x 1 Hole	1	1.92
2.5" Strip w/ 5 Hole	2	1.20
5.5" Strip w/ 11 Hole	3	2.67
Futaba Servo Motor w/ Hardware	2	20.00
50 Teeth Gear (Includes One 069S)	1	9.01
6.5" Rack Strip	1	3.44
Double Angle Strip 4.5" x 0.5" 1x9x1 Hole	1	1.41
3/32" dia. Shrink Tubing	1	0.20
	Total Cost	**144.11**

Table D2: Table that tracks the cost of each order, how much went into each category, and the effect each order had on the budget over time.

Robot Section	Total Cost	Order 1	Order 2	Order 3	Order 4	Order 5	Order 6	Order 7	Order 8	Order 9	Order 10	Order 11	Order 12	Order 13	Order 14	Order 15	Order 16	Order 17
Drivetrain	64.40	62.20	2.20															
Chassis	21.64	4.60	17.04															
Front Arm	21.60						10.00	6.22	0.20			0.88	0.88	1.41	0.89	0.04	0.20	0.88
Back Arm	27.77						22.45	3.92	0.20	0.20	1.00							
Sensor Electronics	8.58		6.14	1.00	0.10				1.34									
Cardboard (Used Across Sections)	0.12					0.04		0.08										
OVERALL TOTAL SPENT	**144.11**	66.80	25.38	1.00	0.10	0.04	32.45	10.22	1.74	0.20	1.00	0.88	0.88	1.41	0.89	0.04	0.20	0.88
	Start	After Order 1	After Order 2	After Order 3	After Order 4	After Order 5	After Order 6	After Order 7	After Order 8	After Order 9	After Order 10	After Order 11	After Order 12	After Order 13	After Order 14	After Order 15	After Order 16	After Order 17
Budget Remaining	160.00	93.20	67.82	66.82	66.72	66.68	34.23	24.01	22.27	22.07	21.07	20.19	19.31	17.90	17.01	16.97	16.77	15.89

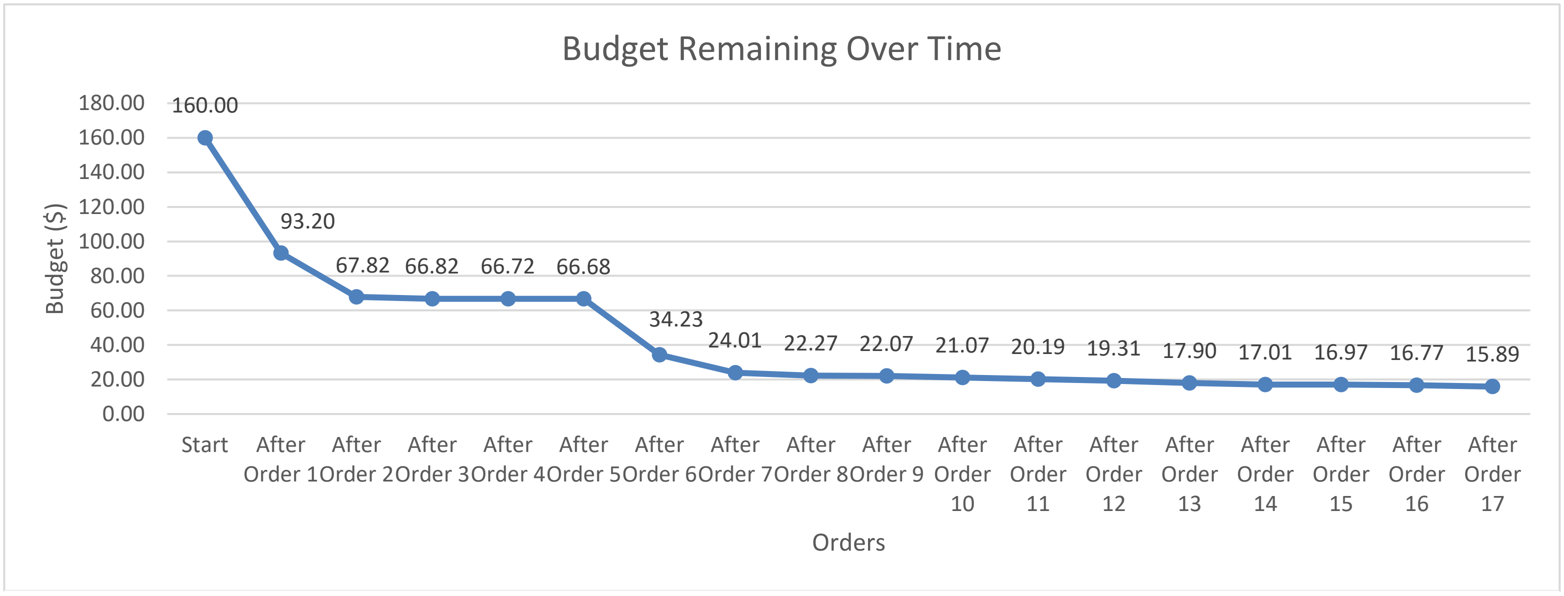


Figure D1: Graph depicting the trend in budget over time. The order 1, 2, 6, and 7 were the teams major orders, consisting of most of the parts needed for the robot. All other orders were made in order to get extra parts or a part that was discovered to have been needed that was not ordered earlier.

Table D3: The time sheet tracking time spent by Alek Srode. Lists date, item (Work Category), start time, end time, and time spent in minutes.

Date	Item	Start Time	End Time	Time Spent (min)
2020-02-05	Documentation	11:35 AM	12:25 PM	50
2020-02-06	Project Management	7:20 PM	9:20 PM	120
2020-02-07	CAD	9:00 AM	9:30 AM	30
2020-02-07	CAD	9:30 AM	10:05 AM	35
2020-02-07	Coding	10:45 AM	12:25 PM	100
2020-02-08	CAD	2:10 PM	4:20 PM	130
2020-02-08	Building/Construction	5:05 PM	6:15 PM	70
2020-02-10	Project Management	9:35 AM	10:05 AM	30
2020-02-10	Project Management	10:15 AM	12:20 AM	125
2020-02-11	Project Management	8:30 PM	9:30 PM	60
2020-02-12	Documentation	9:00 AM	10:15 AM	135
2020-02-13	Project Management	5:30 PM	5:55 PM	25
2020-02-14	CAD	9:30 AM	10:05 AM	35
2020-02-14	Coding	10:15 AM	12:20 PM	125
2020-02-14	CAD	3:00 PM	3:45 PM	45
2020-02-17	Project Management	9:30 AM	10:15 AM	45
2020-02-17	Project Management	10:20 AM	12:20 PM	120
2020-02-17	Building/Construction	1:05 PM	1:45 PM	45
2020-02-18	Coding	4:45 PM	5:30 PM	45
2020-02-18	Other	5:30 PM	8:10 PM	160
2020-02-19	Documentation	9:00 AM	10:05 AM	65
2020-02-19	Building/Construction	10:15 AM	12:25 PM	130
2020-02-19	Building/Construction	8:30 PM	9:00 PM	30
2020-02-20	Other	2:10 PM	4:50 PM	160
2020-02-20	Other	6:00 PM	9:00 PM	180
2020-02-21	Testing	10:20 AM	12:20 PM	120
2020-02-21	Building/Construction	3:30 PM	4:00 PM	30
2020-02-22	Documentation	4:45 PM	5:20 PM	35

2020-02-24	Documentation	9:00 AM	10:20 AM	80
2020-02-24	Building/Construction	10:20 AM	12:20 PM	120
2020-02-24	Building/Construction	2:50 PM	5:20 PM	150
2020-02-24	Coding	6:00 PM	8:50 PM	170
2020-02-25	Testing	4:15 PM	6:40 PM	145
2020-02-26	Documentation	9:05 AM	10:20 AM	75
2020-02-26	Building/Construction	10:20 AM	12:25 PM	125
2020-02-27	Testing	5:30 PM	9:00 PM	210
2020-02-28	Documentation	9:30 AM	10:10 AM	40
2020-02-28	Testing	10:20 AM	12:20 AM	120
2020-03-02	Project Management	9:00 AM	10:15 AM	75
2020-03-02	Other	10:20 AM	12:25 PM	145
2020-03-02	Testing	7:00 PM	9:00 PM	120
2020-03-03	Coding	4:20 PM	9:00 PM	280
2020-03-04	Coding	10:20 AM	12:20 AM	120
2020-03-04	Other	5:20 PM	9:00 PM	220
2020-03-05	Testing	2:10 PM	6:10 PM	240
2020-03-25	Documentation	1:00 PM	5:30 PM	270
2020-03-27	Documentation	1:15 PM	3:15 PM	120
2020-04-06	Documentation	2:30 PM	8:00 PM	330
2020-04-06	Documentation	8:45 PM	10:30 PM	105
2020-04-08	Documentation	8:15 AM	1:00 PM	285
2020-04-08	Documentation	3:45 PM	7:00 PM	195
2020-04-08	Documentation	10:30 PM	11:30 PM	60
2020-04-10	CAD	11:15 PM	12:00 AM	45
2020-04-12	Project Management	7:50 PM	8:20 PM	30
2020-04-12	Documentation	8:30 PM	8:45 PM	15
2020-04-13	Project Management	5:30 PM	6:00 PM	30
2020-04-14	Documentation	4:00 PM	7:00 PM	180
2020-04-15	Documentation	5:15 PM	6:30 PM	75
2020-04-16	Documentation	6:00 PM	7:30 PM	90
2020-04-17	Documentation	1:00 PM	2:45 PM	105
2020-04-17	Documentation	3:15 PM	4:00 PM	45

Table D4: The time sheet tracking time spent by Keith Kriston. Lists date, item (Work Category), start time, end time, and time spent in minutes.

Date	Item	Start Time	End Time	Time Spent (min)
2/5/2020	Documentation	11:30	12:30	60
2/6/2020	Project Management	7:30	9:30	120
2/7/2020	CAD	9:30	10:00	30
2/8/2020	Building/Construction	5:00	6:15	75
2/10/2020	Project Management	9:30	12:30	180
2/11/2020	Project Management	8:30	9:30	60
2/12/2020	Documentation	9:00	10:15	75
2/13/2020	Project Management	5:30	6:00	30
2/14/2020	Project Management	9:30	10:00	30
2/14/2020	Coding	10:00	12:30	150
2/17/2020	Project Management	9:30	12:30	120
2/18/2020	Coding	4:00	5:30	90
2/18/2020	Testing	5:30	8:00	150
2/19/2020	Documentation	9:00	10:00	60
2/19/2020	Building/Construction	10:00	12:30	150
2/20/2020	Testing	6:00	9:00	180
2/21/2020	Testing	10:00	12:30	150
2/22/2020	Documentation	5:00	6:00	60
2/24/2020	Building/Construction	10:15	12:00	105
2/24/2020	Testing	7:00	9:00	120
2/25/2020	Testing	5:30	7:30	120
2/26/2020	Building/Construction	9:45	12:25	150
2/27/2020	Testing	5:45	9:00	255
2/28/2020	Testing	9:45	12:30	135
3/2/2020	Project Management	9:45	12:30	135
3/2/2020	Coding	7:00	8:30	150
3/4/2020	Coding	10:15	12:30	135
3/4/2020	Testing	5:15	6:00	45
3/5/2020	Testing	5:45	8:45	180
3/25/2020	Documentation	1:00	3:00	120

3/27/2020	Documentation	1:15	3:15	120
4/8/2020	Documentation	11:30	12:30	60
4/8/2020	Documentation	1:00	2:15	75
4/8/2020	Documentation	5:30	6:00	30
4/8/2020	Documentation	6:15	7:15	60
4/10/2020	CAD	11:45	12:30	45
4/10/2020	CAD	12:45	2:45	120
4/10/2020	CAD	5:15	5:45	30
4/10/2020	CAD	8:00	12:00	120
4/13/2020	Documentation	5:30	6:00	30
4/15/2020	Documentation	12:45	1:15	30
4/16/2020	Documentation	5:00	5:30	30
4/16/2020	Documentation	5:45	6:30	45
4/16/2020	Documentation	7:00	9:00	120
4/17/2020	Other	1:00	3:30	150
4/20/2020	Documentation	6:45	7:30	45
4/20/2020	Project Management	8:15	9:15	60

Table D5: The time sheet tracking time spent by Thomas Krisak. Lists date, item (Work Category), start time, end time, and time spent in hours and minutes.

Date	Item	Start Time	End Time	Time Spent (hh:mm)
2020-02-05	Documentation	11:35 AM	12:25 PM	00:50
2020-02-06	Project Management	7:20 PM	9:20 PM	02:00
2020-02-06	Building/Construction	9:20 PM	10:20 PM	01:00
2020-02-07	CAD	9:30 AM	10:05 AM	00:35
2020-02-08	Building/Construction	5:05 PM	6:15 PM	01:10
2020-02-10	Project Management	9:35 AM	10:05 AM	00:30
2020-02-10	Project Management	10:15 AM	12:20 PM	02:05
2020-02-11	Project Management	8:30 PM	9:30 PM	01:00
2020-02-12	Documentation	9:00 AM	10:15 AM	01:15
2020-02-12	Project Management	8:30 PM	9:30 PM	01:00
2020-02-14	Project Management	10:15 AM	12:20 PM	02:05
2020-02-14	CAD	3:00 PM	3:45 PM	00:45
2020-02-17	Project Management	9:45 AM	10:15 AM	00:30
2020-02-17	Project Management	10:20 AM	12:20 PM	02:00
2020-02-18	Other	5:30 PM	8:10 PM	02:40
2020-02-18	Building/Construction	7:00 PM	8:10 PM	01:10
2020-02-19	Documentation	9:35 AM	10:05 AM	00:30
2020-02-19	Building/Construction	10:15 AM	12:25 PM	02:10
2020-02-19	Building/Construction	8:30 PM	9:00 PM	00:30
2020-02-20	Project Management	4:00 PM	5:00 PM	01:00
2020-02-20	Testing	7:00 PM	9:00 PM	02:00
2020-02-21	Testing	10:20 AM	12:25 PM	02:05
2020-02-24	Documentation	9:00 AM	10:20 AM	01:20
2020-02-24	Testing	7:00 PM	9:00 PM	02:00
2020-02-25	Building/Construction	6:00 PM	6:40 PM	00:40
2020-02-26	Documentation	9:05 AM	10:20 AM	01:15
2020-02-26	Building/Construction	10:20 AM	12:25 PM	02:05
2020-02-27	Testing	5:30 PM	9:00 PM	03:30
2020-02-28	Documentation	9:30 AM	10:10 AM	00:40
2020-02-28	Testing	10:20 AM	12:20 PM	02:00

2020-03-02	Testing	9:00 AM	10:20 AM	01:20
2020-03-02	Other	10:20 AM	12:25 PM	02:05
2020-03-02	Testing	7:00 PM	9:00 PM	02:00
2020-03-04	Other	10:20 AM	12:25 PM	02:05
2020-03-05	Testing	6:00 PM	8:30 PM	02:30
2020-03-25	Documentation	1:00 PM	4:00 PM	03:00
2020-03-25	Project Management	1:00 PM	9:20 PM	08:20
2020-03-28	Project Management	2:00 PM	6:10 PM	04:10
2020-04-06	Documentation	3:00 PM	8:00 PM	05:00
2020-04-08	Project Management	3:00 PM	5:00 PM	02:00
2020-04-09	CAD	7:00 PM	9:40 PM	02:40
2020-04-10	CAD	9:00 PM	11:30 PM	02:30
2020-04-16	Project Management	4:00 PM	5:00 PM	01:00
2020-04-20	Project Management	6:00 PM	7:00 PM	01:00
2020-04-20	Documentation	8:30 PM	10:00 PM	01:30

Table D6: The time sheet tracking time spent by Kevin Wang. Lists date, item (Work Category), start time, end time, and time spent in minutes.

Date	Item	Start Time	End Time	Time Spent (min)
2020-02-05	Documentation	11:35 AM	12:25 PM	50 min
2020-02-06	Project Management	7:20 PM	9:20 PM	120 min
2020-02-07	CAD	9:30 AM	10:05 AM	35 min
2020-02-07	Coding	10:45 AM	12:25 PM	100 min
2020-02-08	Building/Construction	5:05 PM	6:15 PM	70 min
2020-02-10	Project Management	9:35 AM	10:05 AM	30 min
2020-02-10	Project Management	10:15 AM	12:20 PM	125 min
2020-02-11	Project Management	8:30 PM	9:30 PM	60 min
2020-02-12	Documentation	9:00 AM	10:15 AM	75 min
2020-02-12	CAD	4:30 PM	5:30 PM	60 min
2020-02-13	Project Management	5:30 PM	5:55 PM	25 min
2020-02-14	Project Management	9:30 AM	10:05 AM	35 min
2020-02-14	Coding	10:15 AM	12:20 PM	125 min
2020-02-17	Project Management	9:30 AM	10:15 AM	45 min
2020-02-17	Project Management	10:20 AM	12:20 PM	120 min
2020-02-18	Documentation	4:00 PM	5:30 PM	90 min
2020-02-18	Building/Construction	5:30 PM	8:10 PM	160 min
2020-02-19	Documentation	9:00 AM	10:05 AM	65 min
2020-02-19	Building/Construction	10:15 AM	12:25 PM	130 min
2020-02-19	Building/Construction	8:30 PM	9:00 PM	30 min
2020-02-20	Testing	4:00 PM	5:00 PM	60 min
2020-02-20	Testing	6:00 PM	9:00 PM	180 min
2020-02-21	Testing	10:20 AM	12:25 PM	125 min
2020-02-22	Documentation	5:00 PM	6:00 PM	60 min
2020-02-24	Testing	7:00 PM	9:00 PM	120 min
2020-02-25	Coding	3:45 PM	5:00 PM	75 min
2020-02-25	Documentation	5:00 PM	6:00 PM	60 min
2020-02-25	Coding	6:00 PM	7:15 PM	75 min
2020-02-26	Building/Construction	10:20 AM	11:20 AM	60 min
2020-02-26	Other	11:20 AM	12:25 PM	65 min

2020-02-28	Testing	10:20 AM	12:25 PM	125 min
2020-03-02	Coding	10:20 AM	12:25 PM	125 min
2020-03-02	Testing	7:20 PM	9:00 PM	100 min
2020-03-02	Documentation	9:10 PM	9:30 PM	20 min
2020-03-02	Documentation	11:00 PM	12:00 AM	60 min
2020-03-03	Testing	4:00 PM	5:00 PM	60 min
2020-03-03	Testing	5:00 PM	9:00 PM	240 min
2020-03-04	Testing	10:20 AM	12:25 PM	125 min
2020-03-22	Documentation	9:00 PM	10:00 PM	60 min
2020-03-25	Documentation	10:00 AM	11:45 AM	105 min
2020-04-01	Other	4:00 PM	6:00 PM	120 min
2020-04-02	Other	12:45 PM	1:10 PM	25 min
2020-04-03	Other	3:00 PM	5:00 PM	120 min
2020-04-08	Documentation	3:00 PM	5:30 PM	150 min
2020-04-09	CAD	3:30 PM	4:45 PM	75 min
2020-04-09	Documentation	4:45 PM	6:30 PM	105 min
2020-04-10	CAD	11:00 PM	12:00 AM	60 min
2020-04-14	Documentation	4:00 PM	5:00 PM	60 min
2020-04-15	Other	10:30 AM	11:45 AM	75 min
2020-04-15	Documentation	4:00 PM	4:30 PM	30 min
2020-04-17	Documentation	1:00 PM	2:30 PM	90 min
2020-04-17	Documentation	3:15 PM	4:00 PM	45 min
2020-04-19	Documentation	4:00 PM	6:30 PM	150 min

APPENDIX E
Testing Logs

This appendix provides the testing logs

Table E1: Complete Testing Log (Pages E2 – E10)

When: 2/20	Where: Hitchcock	Why: PT1					
Test No.	No. of Runs	Reason for test	Action / changes made prior	Observations	Changes to be made	Good Results?	Members Present
1	1	PT1 get to the jukebox light	made robot and made code for PT1	Pointed at jukebox light. Right motor malfunctioned due to exposed unconnected wire	fix that		Kevin, Alek, Keith, Thomas
2	1	-	checked connections	Left wheel drove a little faster, did not wait for light to start, wheels were slightly wobbly, slider got stuck on light for jukebox.	improve driving code		Kevin, Alek, Keith, Thomas
3	2	-	debugged movement	POINTED AT RECEIPT BOX Run 1: stuck on boot screen. Run 2: successfully waited for light, drove relatively straight, did not stop due to unplugged encoder, successfully went up ramp.	plug stuff back in		Kevin, Alek, Keith, Thomas
4	1	-	check wire connections	Right wheel fell off. turned full 180 instead of 90 when backwards instead of forwards when going to receipt	fix code		Kevin, Alek, Keith, Thomas
5	1	-	halved the turning code, changed movement code from driveForward() to driveForwardFor().	Correctly turned 90 degrees, overshot the light	work on light sensing and movement		Kevin, Alek, Keith, Thomas
6	1	-	Decreased initial driveForwardFor() distance	Gets very close to light, but does not detect the jukebox light. Slightly crooked to the left.	Improve movement code some more		Kevin, Alek, Keith, Thomas
7	1	-	Decreased 14 inches to 11.5 initial, decreased right motor power	Overshot slightly	more code improvement for movement		Kevin, Alek, Keith, Thomas
8	1	-	Small movement tweak	Crooked left when moving forward, after hitting wall (missed light) backed up successfully instead of continuously running into the wall.	Work on movement		Kevin, Alek, Keith, Thomas
9	1	-	Distance changed from 10 to 7, right motor scale 1, left motor scale 1.01	Turn undershot due to weaker right motor, overshot the light.	work on turning and movement		Kevin, Alek, Keith, Thomas
10	1	-	tweaking	Got close to the light, but did not reach it	more tweaking		Kevin, Alek, Keith, Thomas
11	1	-	more tweaking	Almost made it to the ramp	tweak		Kevin, Alek, Keith,

							Thomas
12	1	-	tweaking	Did not reach jukebox light, after turning toward it, did not make it. wheel fell off going up ramp.	tweak		Kevin, Alek, Keith, Thomas
13	1	-	tweaking	Short of light, backed up a bit too far when making way to ramp	movement tweak		Kevin, Alek, Keith, Thomas
14	1	-	tweak	Detected A light, but hit the wrong button. (hit blue when light was red) Made it to the ramp, but did not make it all the way up. While going up the ramp, it was a little crooked to the left.	changed to get closer to the jukebox light, (originally just barely detects light then stops)	Detected a light. Hit a button	Kevin, Alek, Keith, Thomas
15	1	PT1 Improve consistency	increased movement toward jukebox light before turning toward jukebox.	Went without detecting the light at the start. Got stuck hitting the jukebox button. Going up ramp, went crooked to the left. ground up against the left wall while going up the ramp.	movement tweaks		Kevin, Alek, Keith, Thomas
16	1	-	tweaks	Still gets tuck on jukebox button (did not reach limit count to move back, but stops after hitting the button. It essentially froze). Still crooked going up ramp, did not make it back down (crooked to the left the entire time).	straighten movement up ramp		Kevin, Alek, Keith, Thomas
17	1	-	Increased left motor power in general	Does not properly detect the blue jukebox light, but properly detects the red light. Accidentally hit both buttons but did not completely press either. Crooked up ramp still.	Improve ramp movement and jukebox light detection		Kevin, Alek, Keith, Thomas
18	1	-	Increased motor power of both motors when going up the ramp. Changed light detection code so that the robot defaults to hitting the blue button if the red light is not detected (will hit blue button unless red is detected). Added cow catcher for improved button actuation.	Red button detected, hit red button, went up ramp fast (and way too far), went back slow, undershot ramp, only slightly crooked, bumped into the right wall.	decrease distance going up ramp, increase distance going down ramp		Kevin, Alek, Keith, Thomas
19	2	-	decreased distance up ramp, increased distance down ramp	Run 1: red light registered blue, got stuck hitting button, got too close to right wall going up ramp after hitting the blue button. Run 2: red light, properly hit the red button, went up the ramp, and back down successfully	Nothing	Run 2 got a perfect	Kevin, Alek, Keith, Thomas
20	1	-	nothing	OFFICIAL RUN 1: Red light, tru positive, proper button hit, but got stuck on the button. (15 points)	Adjust movement and get robot unstuck from jukebox button		Kevin, Alek, Keith, Thomas

21	2	-	Adjusted movement to ramp after blue button press, implemented button press kill timer for all three button guess conditions, red, blue, default	Run 1:Blue, true positive (default), wheel fell off backing up toward ramp. Run 2: Red, true positive, hit button, not a true press, continued on. Up ramp, down ramp, slightly crooked right, wheel fell off on way down.	Fix rubber wheel to hubcap		Kevin, Alek, Keith, Thomas
22	2	-	Hot glued rubber part of right wheel to hubcap	Run 1: Everything Smooth. Run 2: OFFICIAL RUN 2: Everything smooth. (23 points out of 20).			Kevin, Alek, Keith, Thomas
When: 2/24 Where: Hitchcock Why: PT2							
Test No.	No. of Runs	Reason for test	Action / changes made prior	Observations	Changes to be made	Good Results?	Members Present
1	2	See if we can shorten travel time	Test PT1 Code at double speed (all) (25% to 50%)	turns over shot, left wheel fell off, straight driving good.	Return turning speed to normal (50% back to 25%), hot glued rubber part of left wheel to hub cap	Straight drive @ 50% seems to drive an accurate distance	Kevin, Keith
2	4	Improve turning accuracy	Return turning speed to 25%	Turning is accurate, wheels stayed on, drove straight, jukebox light detected, proper button was pressed, went up ramp successfully, went too far and hit patty flip, drove down ramp too fast, flipped over	None	Accurate turns, accurate and faster driving.	Kevin, Keith
3	1	Get to bottom of ramp	Using provisional PT2 code	Did not actually use PT2 code	Use PT2 code		Kevin, Keith
4	2	-	Actually use PT2 code	Drove too far, apparently turns slightly more than 90 degrees.	Print out some lines to determine why the robot overshoots driving	none	Kevin, Keith
When: 2/25 Where: Hitchcock Why: PT2							
Test No.	No. of Runs	Reason for test	Action / changes made prior	Observations	Changes to be made	Good Results?	Members Present
1	1	Left Encoder has not been working	Test code created that prints counts recorded by left encoder	Does not print count value, Realized Left Encoder was plugged into a faulty bank (P3_7)	Moved left motor encoder to bank P2_7		Kevin, Keith

2	1	-	Moved Left encoder to P2_7	Worked fine	None	Driving function work now!	Kevin, Keith
3	1	Testing extension and retraction of back arm	Created void methods extend() and retract() for the back arm.	back arm slides into place and is able to interact with ticket slider. Placement is a little tight	Now test robot capability to move the ticket by itself		Kevin, Keith
4	1	-	Added to code, included basic drive function	Chassis held up when pushing ticket. Did not test reaction when pushed all the way through	Put back arm methods into PT2 Code, add precises movement for ticket push in PT2 Code	Chassis holds up when pushing the ticket a little	Kevin, Keith
When: 2/26 Where: Hitchcock Why: PT2							
Test No.	No. of Runs	Reason for test	Action / changes made prior	Observations	Changes to be made	Good Results?	Members Present
1	2	Test front arm, test all code	constructed front arm, included front arm code, secured arm	Arms seem stable and in working condition. Turns were half, undershot	Debug turning	Arms work	All
2	1	Test all code PT2	Doubled turning values (	Turning works but did not make it far enough up ramp	Debug driving straight		All
3	1	-	Increased forward speed going up ramp				All
4	1	-	Increased distance run before dropping off tray	Tray almost went into the sink, fell off on the side closer to ramp	Small adjustments		All
5	1	-		Tray went into the sink, the robot missed the receipt	Improve driving	Tray went into sink	All
When: 2/27 Where: Hitchcock Why: PT2							
Test No.	No. of Runs	Reason for test	Action / changes made prior	Observations	Changes to be made	Good Results?	Members Present
1	2	sink & receipt line up		tray in sink, back left wheel caught on ticket holder, back arm malfunction	increase distance up ramp	tray is in	All
2	1	-	9 in to 10 in past ramp	No longer caught on receipt holder, Undershot to receipt though. Too far away, and not far enough right.	Tweak Values		All

3	2	-	back up to receipt 11 in to 13 in; Drive from wall (from being flush) 1 in so that the back arm aligns with receipt	hit wall next to sink. closer to receipt now. Still ~2 inches in front of receipt	Further tweak		All
4	1	-	Back up to wall further (to flush) 13in to 15in. Drive forward from wall, changed from 1in to 1.5in	backing up to wall, did not get flush. back arm did not extend fully because of colliding with receipt. thus retracted too far.			All
5	2	-	Back up to wall changed 15in to 17in; distance pulling away from wall after flush changed 1.5in to 1.125in;	Backed up flush to wall. Drove forward from wall far enough. Back Arm did not extend far enough.	Adjust Back arm, Adjust distance from wall next to receipt.	Backed up flush to wall.	All
6	1	-	distance pulling away from wall after flush changed 1.125in to 1.1in	did not go flush with wall. Went into receipt holder (overshot turn)	change turning and dstiance to receipt		All
7	2	-	Charged Proteus	motors were stronger, did not get close enough to receipt.	tweak distances		All
8	3	-	Distance backing up to ramp (on way to receipt) from 1in to 2in	run1:backed up too far to receipt. ran into receipt holder; Run2: backed up flush to receipt wall. back arm did not extend far enough; Run3: went into receipt holder	Go a little further after going up ramp		All
9	2	-	Going a little further after going up ramp changing from 10in to 10.5in	not close enough to receipt. Back arm did not reach receipt.	back up closer to the receipt slider		All
10	2	-	1.5 in to 3.5in backing up to receipt holder from tray drop	broke back arm holding contraption due to outside interference	fix back arm holding contraption		All
11	2	-	fixed back arm holding contraption	not close enough to receipt slider. Back Arm was not	increase back up to receipt		All
12	1	-	drive back to receipt change from 3.5 to 4in	backed up too much, ran into receipt holder	tweak		All
13	1	-	Drive back to receipt change from 4in to 3.75in	Did not back up enough	tweak		All
When: 2/28 Where: Hitchcock Why: PT2							
Test No.	No. of Runs	Reason for test	Action / changes made prior	Observations	Changes to be made	Good Results?	Members Present
1	2	Sink and receipt line up (PT2)	fully charged proteus	Left motor too strong, going up ramp, goes slightly to the right. Backing up to the receipt holder. Runs into the receipt holder.	Adjust turning		Kevin, Keith
2		-		going up ramp, ran into receipt holder. messed everything else up	decrease turning		Kevin, Keith

3	2	-		tray did not make it into sink, going crooked to the right up the ramp.			Kevin, Keith
4	1	-	hot glued back arm gear to motor. changed going up ramp 75% speed to 90% speed.	Accurate tray deposit. Still too far from receipt. back arm worked properly but arm fell out of socket.	Increase backing up distance to receipt.		Kevin, Keith
5	1	-	Backing up to receipt, changed from 3.75in to 3.875in. Back arm construction overhaul (reglued parts) more sturdy, can extend farther now)	One encoder was plugged in incorrectly. We changed it accordingly. Second run, the back arm hit the front of the receipt. did not position into the gap of receipt and holder. closer now.	properly plug stuff in	getting closer to sliding receipt.	Kevin, Keith
6	1	-	Plugged encoders in correctly	Arm fell out due to extending too far. Was not close enough to receipt to push it. Seemed that the motors undershot many of the distances. Did not bump into wall of sink like usual. Did not get close to the receipt like that last test			
7	1	-	extend and retract for 2.25sec instead of 2.0 sec	ran too far now?? hit sink wall. But did not get close enough to receipt slider			
8	2	-	Improved back arm	seemingly inconsistent			
9	1	-	Backing up 4in instead of 3.875in to the receipt. Sturdified the back arm	Official: Seemingly .5in away from receipt	Increase arm length or something		All
10	1	-	added stylus onto arm to lengthen	Official: Arm fit into gap, moved receipt. not all the way. Almost, very very close.	Increase receipt slide distance jussst a little		All
11	1	-	Lengthened stylus a little more	Official 3: arm got wobbly, did not finish sliding receipt	sturdify arm		All
12	1	-	Sturdified stylus	Official 4: fully slid receipt but got stuck turning toward hot plate	shorten stylus a litttttle bit	Receipt fully slid	All
13	1	-	retracted stylus juuust a little bit	Official 5: Full 23 points		Full Points, Made it to the hot plate after full slide and tray drop.	All
When: 3/3 Where: Hitchcock Why: PT3							
Test No.	No. of Runs	Reason for test	Action / changes made prior	Observations	Changes to be made	Good Results?	Members Present
1	1	Test Patty flip for PT3	Coded program to run PT3	Made it up ramp perfectly, Turned right but did not move forward, just turned straight again and then "flipped" the patty.	Actually drive up to the patty flip	Good going up ramp (code	Kevin, Alek

						taken from PT2)	
2	1	-	Found error in code: driveForward(***); Solution: changed to: driveForwardFor(***)).	Made it to the patty flipper, arm bumped into the plate, arm slipped from under the plate when trying to flip. Robot slid (rotated left)	Lower arm slightly, add hook to arm so robot does not slide	Made it to patty flip	Kevin, Alek, Thomas
3	1	-	Added small cardboard hook to the end of the robot, lowered front arm starting angle by 5 degree (to 25 degrees)	arm started high, made a mistake changing arm angle	correct arm angle		Kevin, Alek, Thomas
4	1	-	Changed starting angle to 125 degrees (from 5 degrees)	Right wheel got loose. Driving went haywire as a result	Secure wheels		Kevin, Alek, Thomas
5	3	-	Superglued DuBro wheels to the IGWAN/Wheel Adapters	Run1: unplugged wire to servo; Run2: Misses Hot plate; Run3: Basically run 2, but further info, turns left too much when turning to face patty flip. Front arm barely misses to the left of the plate. Also, Going up ramp, robot rotates slightly to the left	Decrease left turn when facing patty flip		Kevin, Alek, Thomas
6	4	-	Changed left turn from 90 degrees to 85 degrees	Run 1: Patty successfully flipped, plate was not returned to original position, very smooth performance. Run 2: Patty successfully flipped and plate returned to original position. Run 3: same result as run 2. Run 4: moved the plate but did not flip the patty. Seems that the closer the arm is to the rotating point of the hot plate, the higher the chance that the robot manages to get full points.	Slightly increase the distance driven to the right wall before going to the patty flip, which may make the flip more consistent and efficient.	Successfully flipped patty and returned hot plate	Kevin, Alek, Thomas
7	2	Test Patty Flip and ice cream lever and PT3	Increased distance driven right before going up to patty from 9in to 9.2in. Added new code to flip ice cream lever	Run1: front arm missed hot plate from opposite side, slides out from under on the right side now, instead of the left. Run2: same thing	Slightly decrease right movement, slightly decrease turn after patty flip		Kevin, Alek, Thomas
8	1	-	Changed distance moving right before patty flip from 9.2in to 9.1in. Decreased left turn (after patty flip) from 87 degree to 70 degrees	Front arm for patty got loose. Did not flip patty	Secure front patty arm, decrease left turn after patty going to the ice ream levers		Kevin, Alek, Thomas
9	2	-	Front patty arm hot glued to front axle, old glue taken off.	Run 1: Patty was not flipped, But ice cream lever succesfully flipped. Run 2: Neither action successful, arm slides in to wheel for patty, arm missed left lever, ended up right of it.	Change turn degrees, tweak	Ice cream lever flipped.	Kevin, Alek, Thomas

10	2	-	end angle for robot front patty arm changed from 55 degrees to 38 degrees (to bottom). Added a 3 degree turn left once robot reaches patty flip. Changed starting angle of front arm from 135 degrees to 133 degrees (was hitting cow catcher).	Run 1: Patty was not flipped. Ice cream not flipped. Arm missed to left of patty flipper, did not go all the way up. Ended up in between left and center ice cream levers. Whenever patty not flipped correctly, arm orientation is changed (forced downward a little). Run 2: Same thing happened.	Tweak		Kevin, Alek
11	2	-	Remove the left turn 3 degrees at patty flip. Change distance moving right before patty flip from 9.1in to 9.0in. Incease distance moving to icecream from 15in to 17in.	Run 1: ended up outside the patty flip. Ran into middle ice cream lever. aka went too far. Run 2: ended up inside patty flip. Starting arm position got changed.	decrease distance traveled to ice cream lever. tweak patty flip.		Kevin, Alek
12	2	-	Drive power moving right changed from 50 to 25. Decreased distance traveled to ice cream lever from 17in to 16in.	Run 1: Arm ended way outside of patty flip. Ran into ice cream lever, was left of left lever but close. Run 2: same and consistent	Increase right turn after going up ramp		Kevin, Alek
13	2	-	Increase right turn after going up ramp from 90 degrees to 100 degrees	Run 1: Front arm makes contact with the patty flip, servo is now weaker?? Cannot even lift plate anymore. Misses ice cream lever. Run 2: same thing	Figure out problem with front arm servo.		Kevin, Alek
When: 3/4 Where: Hitchcock Why: PT3							
Test No.	No. of Runs	Reason for test	Action / changes made prior	Observations	Changes to be made	Good Results?	Members Present
1	1	Code for PT3	Resecured front axle to servo motor. Rescrewed motor block (it had gotten loose).	Robot went wonky. IGWAN Motor block screwed on incorrectly.	Rescrewed motor block (of IGWAN motor)		Kevin, Alek, Thomas, Keith
2	2	-	Correctly rescrewed motor block. Removed stylus from back arm due to complaints	Run 1: too far in on the patty flip, flipped left ice cream lever. Run 2: same, consistent.	change distance	Flipped left Icecream lever	Kevin, Alek, Thomas, Keith
3	1	-	Distance moving right before patty flip changed from 9.0in to 8.5in	Hot plate was starting to flip. Patty was not fully flipped. ice cream lever kinda flipped? in between left and center levers	Increase distance moved right		Kevin, Alek, Thomas, Keith
4	3	-	Distance moving right before patty flip changed from 8.5in to 8.75in	Run 1: Too far outside patty flip. Center ice cream lever was flipped, kinda sorta ran into it. Run 2: same thing. Run 3: Too close inside patty flip.	Add some sleeps to check problems		Kevin, Alek, Thomas, Keith
5	3	-	Added 1.5s sleep after getting to hot plate before turning arm. Added 1.0 s sleep after lifting arm and turning for patty flip.	Run 1: Ended up too far inside patty flip, flipped middle ice cream lever. Run 2: Ended up too far OUTSIDE patty flip, flipped left lever.	Figure out why we are so inconsistent		Kevin, Alek, Thomas, Keith
6	2	-	Slowed down movement speed to increase consistency.	Run 1: Ended up inside patty flip. Flipped left ice cream lever. Run 2: same but got near hot plate and got stuck???	-		Kevin, Alek, Thomas,

							Keith
7	3	-	Tweaks (turning and moving)	Run 1: Ended up too far outside patty flip. Ran into ice cream levers, in between left and center. Run 2: Close to outside patty flip, successfully flipped patty and returned hot plate to original position. Run 3: Closer to center, patty flipped successfully, hot plate stayed up. Ran into ice cream levers again.	Slightly decrease left turn before reaching patty flip.		Kevin, Alek, Thomas, Keith
8	2	-	Changed left turn before hot plate from 93 degrees to 91 degrees.	Run 1: Too close to patty flip, did not flip. Ran into ice cream levers. Run 2: same.	Charge proteus to decrease variations in runs.		Kevin, Alek, Keith
9	4	-	Created a template	Run 1: encoder unplugged. Run 2: Ran too close to left wall when going up ramp. was too far outside hot plate. Ran into center lever. Run 3: close outside hotplate. But performed without flaw, ran into middle and right ice cream levers. OFFICIAL: Close, flipped patty but hot plate stayed up. ran into levers but did not pull them down.	Charge proteus, tweak numbers and add code to flip patty down.		Kevin, Alek, Keith, Thomas
10	2	-	Distance run to ice cream lever changed from 16in to 15in and front arm for ice cream going down farther (85 degrees to 80 degrees). Added code to lower hot plate manually just in case.	Run 1: bumped into right wall going up ramp, killed run. Run 2: perfect patty flip. ran into machine on way to ice cream flip.	Charge proteus. Tweak code going to ice cream lever.		Kevin, Alek, Keith, Thomas
11	3	-	Backing up 1.5in (instead of 1in) after patty flip. Increasing turn after patty flip by 10 degrees.	Run 1: Going up ramp, ended up turning slightly left, ruined run (too far outside patty flip). Run 2: Going up ramp, ended up turning slightly right, ruined run (too far inside patty flip). Run 3: Going up ramp, good. patty flipped, (front arm caught on hot plate coming down). Ruined front arm axle to motor connection.	Fix front arm connection again.		Kevin, Alek, Keith, Thomas