

Carmen's Diner Robot Project

Engineering 1282.01H

Spring 2020

Team: PI3 Fighters (PIE3)

Liam Evans

Harrison Hecker

Zach Jones


Cam Powers

Instructor: Bart Krieger

Section: 3:00pm-5:05pm

GTA: Tony Heglas

Executive Summary

Carmen's Diner has tasked the team with designing and constructing a robot to complete several different tasks on a course: pressing a jukebox button, flipping a burger, sliding a ticket, depositing a tray, flipping an ice cream lever up and down, and pressing a large button. Carmen's Diner needs this robot to deal with the influx of business, maintain the qualities that made it successful, and continue to grow and thrive. 

The team designed an arm attached to a servo motor that could rotate to and press the correct jukebox button and slide into and rotate the burger crank. The team also designed an arm that slid in between the ticket and its holder and moved it as the robot moved forward. Additionally, the team designed tray holder that utilized a motor to keep the tray from sliding out until the robot was at the correct location. The team designed wedges that would force the ice cream lever up or down when the robot drove forward. Finally, the team decided that any of the previous mechanisms, particularly the tray deposit and jukebox button/burger flip mechanisms, would easily be able to press the final button. The team built and attached the mechanisms for the jukebox button, burger crank, ticket slider, tray deposit, and final button tasks. The team has not yet built the ice cream lever mechanism. However, development is on schedule, as the team put in a total of 231.5 hours up to this point, and the build is underbudget by \$41.82. Additionally, the robot meets all the different specifications, requirements, and constraints laid out by Carmen's Diner prior to the start of the build.

The current robot has several strengths. The tray deposit and jukebox mechanisms are quick, consistent, and efficient. Several parts, such as the jukebox button/burger crank arm, are easily replaceable, and many other parts, such as the tray deposit mechanism, are versatile and allow for more than one task to be completed with one mechanism structure. However, the

positioning code and code accounting for a weak left motor are inconsistent and unpredictable. Additionally, the ticket mechanism requires great precision and a small amount of luck to successfully complete its task. Also, the arm that rotates the burger crank is consistently torn off the motor due to the stress of the crank on it.

To improve the robot, the team needs to be reassembled and given time and materials. The team will take time to analyze, refine, and fix the positioning code and the code accounting for a weak left motor. Additionally, the team will take time to refine the code for the ticket mechanism and, in the event that refining the code does not solve the problem, redesign and rebuild the ticket arm itself to allow for less precision and luck. The team will use the materials to create duplicate jukebox button/burger crank arms in the event that one breaks. Additionally, the team will use screws or epoxy instead of glue to secure the arm to the motor. Finally, the team will use the rest of the materials to construct the ice cream lever wedges and attach them to the tray deposit mechanism structure.

Table of Contents

List of Figures	vii
List of Tables	viii
Section 1: Introduction	1
Section 1.1: Project Problem Statement.....	1
Section 1.2: Team Members and Dates.....	1
Section 1.3: Report Organization	2
Section 2: Preliminary Concepts	4
Section 2.1: Requirements and Constraints.....	4
Section 2.1.1: Specifications.....	4
Section 2.1.2: Construction.....	5
Section 2.1.3: Course Description	5
Section 2.2: Brainstorming Process	7
Section 2.2.1: Individual Brainstorming.....	7
Section 2.2.2: Group Brainstorming	8
Section 2.2.3: Creation and Use of Mechanism Design Matrices	8
Section 2.2.4: Creation and Use of an Overall Design Matrix	9
Section 2.3: Descriptions of Preliminary Concepts for Mechanism Designs	10
Section 2.3.1: Drivetrain/Chassis.....	10
Section 2.3.2: Jukebox	12
Section 2.3.3: Ticket	12
Section 2.3.4: Tray Deposit	14
Section 2.3.5: Ice Cream Lever.....	15
Section 2.3.6: Burger Crank.....	16
Section 2.3.7: Final Button	17
Section 2.3.8: Strategy	17
Section 2.3.9: Final Design.....	18
Section 2.4: Preliminary Code	19
Section 3: Analysis, Testing, and Refinement	20
Section 3.1: Preliminary Concepts to Final Design	20

Section 3.1.1: Tray Deposit Mechanism.....	20
Section 3.1.2: Ticket Sliding Arm	21
Section 3.1.3: Jukebox Button/Burger Crank Arm.....	22
Section 3.1.4: Chassis	23
Section 3.2: Analysis and Calculations	23
Section 3.2.1: Microswitches and CdS Cells	23
Section 3.2.2: Drivetrain Data	24
Section 3.2.3: Shaft Encoding and Line Following	25
Section 3.2.4: RPS and Data Logging	25
Section 3.3: Tests	26
Section 3.3.1: Performance Test #1	27
Section 3.3.2: Performance Test #2	28
Section 3.3.3: Performance Test #3	28
Section 4: Current Status of the Prototype.....	29
Section 4.1: Specifications	29
Section 4.2: The Robot Itself.....	30
Section 4.2.1: Physical Robot	30
Section 4.2.1.1: Jukebox Button/Burger Crank Mechanism.....	30
Section 4.2.1.2: Tray Deposit Mechanism.....	31
Section 4.2.1.3: Ticket Slider Mechanism	31
Section 4.2.1.4: Final Button Mechanism.....	31
Section 4.2.1.5: QR Holder.....	32
Section 4.2.2: Code	32
Section 4.2.2.1: Driving Functions	32
Section 4.2.2.2: Turning Functions.....	33
Section 4.2.2.3: Checking Functions	33
Section 4.2.2.4: Jukebox Button Function	33
Section 4.2.2.5: Performance Test Main Codes.....	34
Section 4.3: Time, Money, and Development.....	34

Section 4.3.1: Time	34
Section 4.3.2: Money	35
Section 4.3.3: Development.....	36
Section 4.4: Strengths and Weaknesses	37
Section 5: Future Development.....	39
Section 5.1: Projected Necessary Resources	39
Section 5.1.1: Personnel.....	39
Section 5.1.2: Materials	40
Section 5.1.3: Time	41
Section 5.2: Anticipated Challenges	41
Section 5.3: Projected Strengths and Weaknesses	42
Section 6: Summary and Conclusions.....	44
Section 6.1: The Past.....	44
Section 6.2: The Present.....	45
Section 6.3: The Future	45
Section 7: References	47
Section 7.1: Online Resources	47
Section 7.2: FEH Resources	47
Appendix A: Preliminary Concepts	A1
Appendix B: Analysis, Testing, and Refinement.....	B1
Appendix C: Testing	C1
Appendix D: Physical Robot Mechanisms.....	D1
Appendix E: Code.....	E1

List of Figures

Figure No.	Figure Description	Page
1	Picture of course with location numbering.	6
2	Example of a mechanism design matrix	5
3	Second drivetrain/chassis concept	7
4	First and second jukebox button mechanism design concept	8
5	First ticket slider mechanism design concept	8
6	Second ticket slider mechanism design concept	9
7	First tray deposit mechanism design concept	9
8	Second tray deposit mechanism design concept	10
9	First ice cream lever design concept	10
10	Second ice cream lever design concept	11
11	Second burger crank design concept	11
12	Both possible robot routes	13
13	Initial final design as a result of brainstorming	13
14	Final tray deposit mechanism design	19
15	Final ticket sliding arm design	20
16	Final jukebox button/burger crank arm design	21
17	Example of a testing log and the testing format the team followed	23
18	Summary of total time the team spent in different areas of the project.	31
19	Summary of team's spending up until development and testing ceased	41
1A	Second initial final robot design after brainstorming	A2
2A	Third initial final robot design after brainstorming	A2
3A	SolidWorks mockup of team's final choice for an initial overall design.	A3
4A	Cardboard mockup of team's final choice for an initial overall design.	A3
5A	The team's initial idea for the complete modular course code	A4
1B	The torque-speed graph used to pick the motors for the robot [4].	B2
1D	SolidWorks exploded and condensed view of the current robot.	D2
2D	SolidWorks drawing of the jukebox button/burger crank mechanism.	D3
3D	SolidWorks drawing of the jukebox button/burger crank arm.	D4
4D	SolidWorks drawing of the jukebox button/burger crank motor holder.	D5
5D	SolidWorks drawing of the chassis.	D6
6D	SolidWorks drawing of the tray deposit mechanism.	D7
7D	SolidWorks drawing of the walls for the tray holder.	D8
8D	SolidWorks drawing of the tray holder ramp.	D9
9D	SolidWorks drawing of the ticket slider mechanism	D10
10D	SolidWorks drawing of the ticket arm sheath.	D11
11D	SolidWorks drawing of the QR code holder.	D12
1E	Driving for a certain distance method code.	E2
2E	Driving for a certain time method code.	E3
3E	Driving up the ramp method code.	E4
4E	Right turn method code.	E5
5E	Left turn method code.	E6

6E	Right turn with one wheel method code.	E6
7E	Left turn with one wheel method code.	E7
8E	Code checking “x” coordinate if robot is moving in positive “x” direction.	E7
9E	Code checking “y” coordinate if robot is moving in positive “y” direction.	E8
10E	Code checking “x” coordinate if robot is moving in negative “x” direction.	E8
11E	Code checking “y” coordinate if robot is moving in negative “y” direction.	E9
12E	Check heading method code.	E9
13E	Code for pressing correct jukebox button.	E10
14E	Performance Test #1 main code	E11
15E	Performance Test #2 main code	E12
16E	Performance Test #3 main code	E13

List of Tables

Table No.	Table Description	Page
1	Important Project Due Dates before COVID-19	2
2	Important Project Due Dates after COVID-19	2
3	Description of numbered locations on course.	6
4	Example of a mechanism design matrix.	9
5	Overall robot design matrix	10
6	Parts for building ice cream lever mechanism and their prices.	35
1C	Summary of Purpose, Outcome, and Changes from all tests	C2

1. Introduction

The following subsections explain the problems facing the team, explain why solving them is important, present team members and dates, and outline the organization of the report itself.

1.1 Project Problem Statement

The American diner used to be the primary source of food and socialization for the American people, but after the 1960s, the fast food business pushed it to the side until there were only about 5000 diners left operating [1]. Despite this, some diners have survived and even grown stronger by incorporating technology, such as online ordering, into their business models. One such diner, Carmen's Diner, did so, but was struggling to cope with increased demand and failing to complete simple tasks: changing music, placing trays on the trash or in the sink, sliding ticket orders, flipping burgers, and switching ice cream levers. If this problem was not solved, Carmen's Diner would be unable to cope with the influx of business. Therefore, it decided to incorporate a new form of technology to free the workers to interact with customers: robots. By utilizing an effective robot, the restaurant could retain its "Friends Eat Here" atmosphere, the qualities that made it successful, and continue to grow and thrive. The purpose of the Diner's project was to have a research team design and build a robot to complete several mundane restaurant tasks on a scale model of the diner to help improve business [2].

1.2 Team Members and Dates

The team consisted of four people: Zach Jones, Liam Evans, Cam Powers, and Harrison Hecker. Each of the team members worked diligently to complete certain parts of the project by their due dates, found below in Tables 1 and 2 on the next page.

Table 1: Important Project Due Dates before COVID-19.

Task	Project Due Date
Individual Brainstorming	29-Jan
Team Working Agreement	31-Jan
Sketches, Decisions and Strategy	5-Feb
Design Schedule	7-Feb
Prototype/Mockup	10-Feb
Exploration 1 Progress Report	14-Feb
Agendas and Notes	17-Feb
Final Report Outline	19-Feb
Exploration 2 Progress Report	21-Feb
Code Representation	24-Feb
Budget and Testing Log	26-Feb

Unfortunately, a global pandemic surrounding the COVID-19 virus prevented the team from meeting in person from March 15rd, 2020 onward. Table 2 below lists the due dates for assignments after the team was forced to only work together online.

Table 2: Important Project Due Dates after COVID-19.

Task	Project Due Date
Exploration 3 Progress Report	25-Mar
Final Report First Draft	25-Mar
Electrical Systems	27-Mar
Final Report Second Draft	8-Apr
New - 1	27-Mar
Working Drawing Set	10-Apr
Oral Report	17-Apr
Final Written Report	20-Apr
Project Portfolio	22-Apr

1.3 Report Organization

Following the Introduction is Preliminary Concepts, which details the requirements and constraints of the project, the team's brainstorming process, and descriptions of preliminary concepts for the mechanism designs and code. Then, Analysis, Testing, and Refinement shows

how the preliminary concepts transitioned to the final design and what analyses, calculations, and tests the team performed. Current Status of the Prototype then details what state the current prototype is in, including areas such as specifications already met, time and money spent, and the robot's strengths and weaknesses. Next, Future Development describes what is needed to completely develop the robot, such as what resources would be needed and how much, any anticipated challenges, and the projected strengths and weaknesses of the final design. Following that, Summary, Conclusions, and Recommendations summarizes and concludes the project and report in addition to suggesting any future work and/or improvements. Next, References provides any resources the group used in the process of building or documenting the robot and writing the report. Finally, Appendices contains different categories of supplemental information referenced in the report, such as coding, course testing logs, and SolidWorks drawings.

2. Preliminary Concepts

The following subsections describe the requirements and constraints the team worked under, the brainstorming process the team opted to use, and the preliminary concepts, both in written and visual format, for the different mechanisms to complete the objectives of the course.

2.1 Requirements and Constraints

The team operated under a myriad of different requirements and constraints throughout the build. They can be grouped into two specific areas, Specifications and Construction, which are detailed below in the first subsections. Additionally, because the entire build was based around a scale model of the restaurant, the third subsection provides a detailed description of the course in both written and visual format.

2.1.1 Specifications

There were several specifications the team abided by throughout the process. First, the robot, *in its starting position*, had to fit within and 9" x 9" box and could be no taller than 12". Additionally, the team could only use a programmable Proteus microcontroller to control all powered systems located on the robot. There were a series of rules concerning the Proteus the team was required to follow as well. First, the team could not attempt any repair work on their controller. Second, no adhesives could be applied to the Proteus except Velcro, which could not be applied anywhere on the face containing the screen. Third, the Proteus could not be used as a touch sensor or any other type of sensor. Finally, the Proteus could not be used as part of the structure of the robot and could never be used as a mechanical element of the robot.

The team was subjected to numerous other constraints as well. The team had a budget of no more than \$160. Additionally, for every \$.50 increment over the budget, *the team was reprimanded by the owner's of Carmen's Diner*. The robot was required to be autonomous. The

robot could not perform any action that could potentially harm another robot or the viewing audience. It could not receive or transmit any wireless signals, with the exception of communication with RPS. To communicate with RPS, a QR code was required to be at least 9” above the course surface and present at all practice runs, performance tests, and competitions after Performance Test 2. Additionally, because adhesives and paint were not intended as primary structural materials, if an adhesive material was deemed a structural element, the team would be charged accordingly. Also, adhesive materials could not intentionally come into contact with the course at any moment. Finally, the team was allowed a maximum of three omnidirectional wheels [2].

2.1.2 Construction

The team was subjected to a number of constraints regarding the construction of the robot. First and foremost, only team members were allowed to perform robot construction. All resources or parts used for construction, off-the-shelf or custom-made, had to be purchased through the company store, unless the team had the project manager’s approval to acquire them from an additional supplier. If the project manager granted approval, the price of the resources or parts had to be factored into the team’s budget. Finally, Carmen’s Diner placed great emphasis on reusability, so the team could return any unused parts for a credit of 75% of the purchase price. However, this did not apply to any parts left over from any kit, there was no buyback credit for special-order parts, no electrical parts, sensors, or motors could be returned, and all returns had to be approved by the Company Store before credit was issued [2].

2.1.3 Course Description

The course was comprised of two different levels connected by a ramp. It contained seven main objectives, all of which are named and described in Figure 1 and Table 3, found on the next page. Additionally, the course was digitally overlaid with a coordinate system provided by RPS,

or Robot Positioning System. The origin is located at the red square shown in Figure 1. The coordinate system progressed as if the viewer was standing on the right side of the course looking forward, so the positive “y” direction moves leftward and the positive “x” direction moves upward. There is also an RPS “dead zone” within the blue shape, found in the bottom left corner of Figure 1. In this “dead zone,” RPS did not function so the team relied on other methods of navigation.

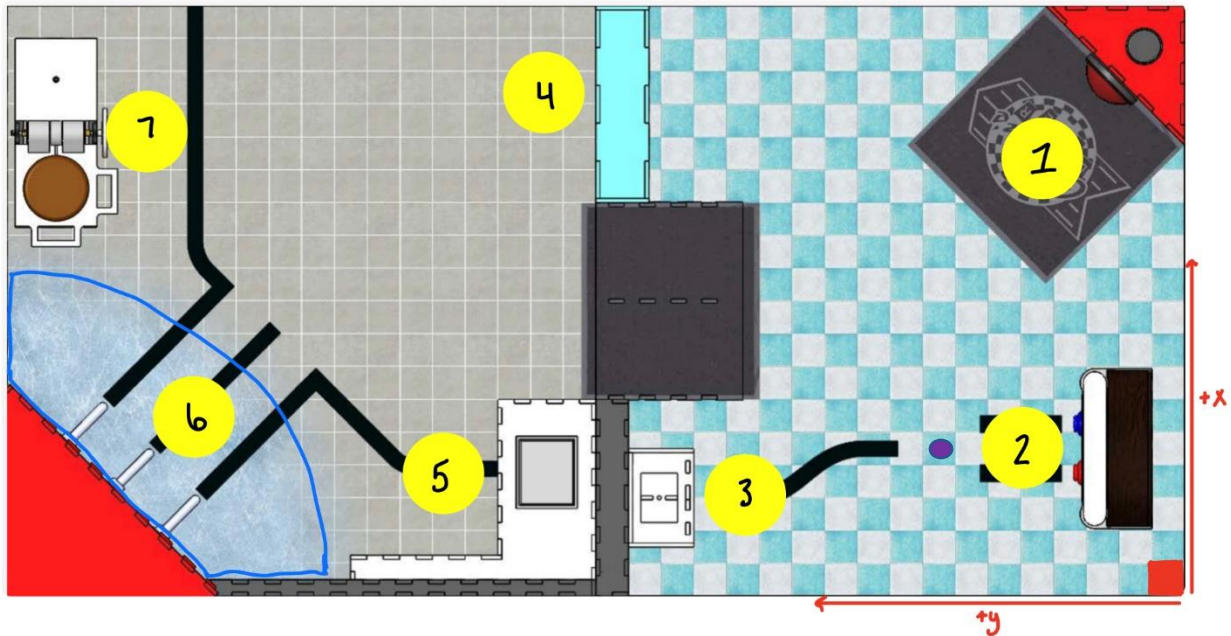



Figure 1: Picture of course with location numbering.

At each location, described in Table 3 on the next page, the robot completed a task. At Location 1, the robot was not required to complete a task at the *start* of the run but had to press the red button at the *end* of the run to stop the timer. At Location 2, the robot had to detect what light was being displayed from the small purple circle in front of the jukebox and then proceed forward and press the button corresponding to the light it detected. The robot also had to deposit a tray in either Location 3 or 5. At Location 3, the robot had to drop the tray on top of the trashcan. At Location 5, the robot had to drop the tray into the sink. At Location 4, the robot was required to slide a ticket from its starting position near the wall of the course to its final position near the ramp. At Location 6, the robot was required to use RPS in order to figure out what ice cream lever

to flip. Then, it had to proceed forward, flip the correct lever down, wait 7 seconds, and then flip the lever back up. Finally, at Location 7, the robot was required to raise the gray platform with the burger on it high enough to flip the burger to the other side, and then lower the platform back down.

Table 3: Description of numbered locations on course. 

Location	Description
1	Starting position and start button (red circle)
2	Jukebox
3	Trash cans and possible tray deposit location
4	Ticket slider
5	Sink and possible tray deposit location
6	Three ice cream levers
7	Burger crank

2.2 Brainstorming Process

The team utilized an extensive but effective brainstorming process for the creation of the robot design. It consisted of four main steps: individual brainstorming, group brainstorming, creation and use of mechanism design matrices, and creation and use of an overall robot design matrix.

2.2.1 Individual Brainstorming

During individual brainstorming, each of the team members separated and synthesized designs for the drivetrain/chassis and mechanisms of the robot. To maximize the effectiveness of this step, the team implemented some rules. To ensure that there was a substantial amount of designs for the synthesis of the overall robot, each team member created a minimum of two different ideas for the strategy/route, chassis, drivetrain, and different mechanisms for completing the objectives of the course. Additionally, to facilitate effective communication of these ideas, the team members drew rough sketches, either by hand or in SolidWorks, of each idea that he or she

had. In keeping with effective communication, and to ensure that proper documentation was maintained, each team member typed his or her ideas, sketches, any notes about the robot course itself, or any other ideas she or he had in a neat and easy to follow document. Finally, the group gave itself a maximum of five days for this step to increase the number and quality of ideas before moving on to the next step.


2.2.2 Group Brainstorming

The group brainstorming began with a simple structure: one group member presented his or her ideas for each of the elements of the robot, often using a sketch to supplement the explanation while the others listened intently, asked clarifying questions, and wrote down any notes. After this, each member discussed his or her notes with the others, sharing ideas he or she liked, ideas he or she thought worked well together, and ideas he or she thought could prove useful if improved or altered in some way. Next, the team combined any ideas that were similar and discarded others until it had four designs for the chassis, drivetrain, and course objective mechanisms and three possible routes for the robot to navigate the course.

2.2.3 Creation and Use of Mechanism Design Matrices

This step provided the team with some method of quantifying the effectiveness of each of the designs with respect to different criteria, such as cost, simplicity, or versatility. In order to accomplish this, the team went through each element of the robot and created a list of criteria it considered to be important that the designs satisfy. While the team used the same criteria in many matrices, many parts had unique criteria, such as surface area in the chassis design matrix. Then, the team created a weighted point-based system. The team gave a weight of one through five to each criterion, with five being the heaviest weight. Then, each team member went through each design for all the elements of the robot and assigned it a value one through five, with five meaning

that the design met the criterion strongly and one meaning that the design met the criterion weakly. Before performing any calculations, the team decided that in the event of a tie, it would brainstorm more criteria and pit the two concepts against each other. Next, the group averaged the scores for each design and multiplied them by the weight of the criterion. These values were then added together to give the designs a total score. The team selected the two designs with the highest score to move on to the next step. The jukebox mechanism matrix, which is similar to the other matrices, can be seen below in Table 4.

Table 4: Example of a mechanism design matrix. 

Criteria	Weight	arm on wheel that can rotate or press button	arm on front at same height as button robot slightly rotates based on color	line following	corner of robot aligned to correct button
Cost	3	3.5	4.8	5	5
Speed	3	4.3	5	3	2.5
Repeatability	5	4.8	4	4	4
Simplicity	4	3.8	4.8	4.8	4.8
Multi-Use	3	4	3.3	2	1
Total		74.6	78.5	74.2	64.7

2.2.4 Creation and Use of an Overall Design Matrix

With the top two designs for each robot element in hand, the team then synthesized three different overall designs of a robot using variations of the first and second picks for each of the robot elements. After that, the team created one more design matrix, similar to the others, for the overall design of the robot. As with the individual mechanism design concepts, in the event of a tie, the team would pit the two sketches against each other with additional criteria. However, it differed slightly in that, as shown in Table 5 below on the next page, the team wanted the design to satisfy more criteria and selected only one design from the matrix, rather than two. The team then used this design moving forward to build the robot. Sketch 2 and 3 can be found in Appendix A in Figures 1A and 2A, respectively.

Table 5: Overall robot design matrix.

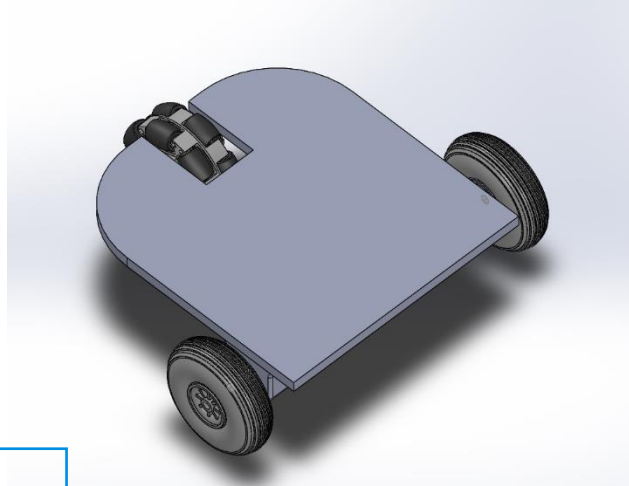
Criteria	Weight	Sketch 1	Sketch 2	Sketch 3
Cost	4	3	2	2
Simplicity (building)	3	4	4	3
Simplicity (coding)	3	4	3	4
Precision (mechanisms)	2	3	2	2
Adaptability	5	4	4	4
Efficiency (moving from task to task)	4	4	2	4
Total		78	61	69


2.3 Descriptions of Preliminary Concepts for Mechanism Designs

This section describes the top two preliminary concepts for each of the robot's main components in words, pictures, sketches, and mockups.

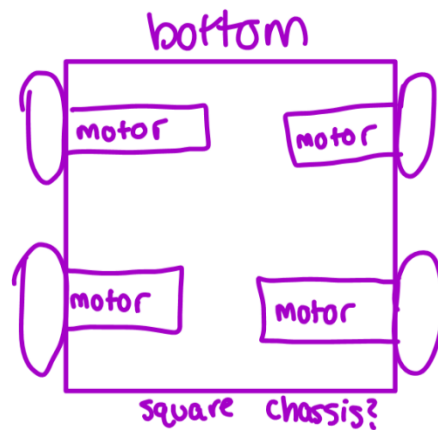
2.3.1 Drive Train / Chassis


The drivetrain and chassis concept that the team decided to implement was a rectangular chassis with two drive wheels in front, one on each side of the robot, and one omnidirectional wheel, located at the center of the back of the robot. Its rough sketch is shown on the next page in Figure 4. The team judged this design to have the best balance of maneuverability and power, with very little friction in turns and powerful front-wheel IGWAN motors. However, that meant the turning axis would be in between the front two wheels, rather than in the center of the chassis.




Figure 4: First drivetrain/chassis concept: a curved chassis with three wheels.

The team's second choice for chassis featured a rectangular body with four wheels, arranged like that of a car. Each set of two wheels on the same side would be mechanically linked through gears or chains, so the robot would drive and control like a tank. While turning would be less sharp and exact, this design scored highly because of its simplicity of design and is shown in Figure 3 below.




Figure 3: Second drivetrain/chassis concept: a square chassis with four wheels.

2.3.2 Jukebox Button

The team's top choices for pressing the jukebox button were very similar. The first, shown in Figure 4 below, utilized a wheel that had an extended peg that would press the appropriate button depending on its rotation. Similar to the first, the team's second choice contained an arm that would rotate depending on what light was shown so the robot would not have to turn to the correct button and would just drive forward. The only difference would be that the arm would be directly connected to a motor, rather than to an additional piece that was connected to a motor. The team chose the arm over the wheel because it cost less to produce just the arm and the arm's possible multi-functionality.

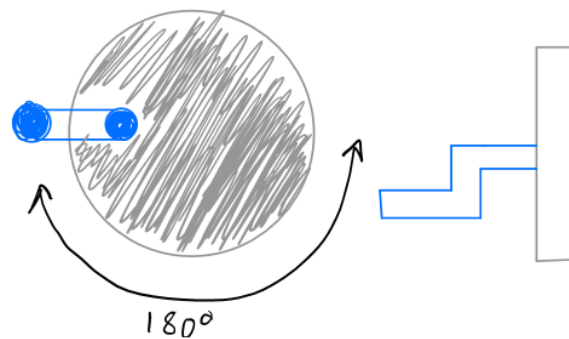


Figure 4: The first jukebox mechanism design concept.

2.3.3 Ticket Slider

For sliding the ticket, the team decided that a long, stiff arm would be the simplest and most efficient design. It would leverage the turning of the robot to move the ticket, being mounted in the center back side of the robot. The team judged this design to be simple, cost-effective, and repeatable. It is depicted in Figure 5 on the next page.

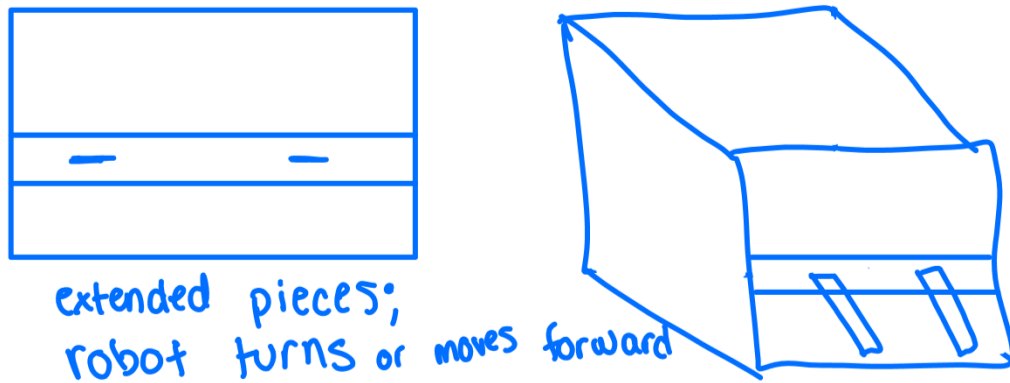


Figure 5: First ticket mechanism design concept: static arms that push the ticket as the robot turns.

The team chose a hook mounted on the front side of the robot to be the second-best option. With this design, the robot would pull up alongside the ticket slider, turning so that the hook would be behind the ticket and drive forward to slide the ticket. Its design is shown in Figure 6 below.

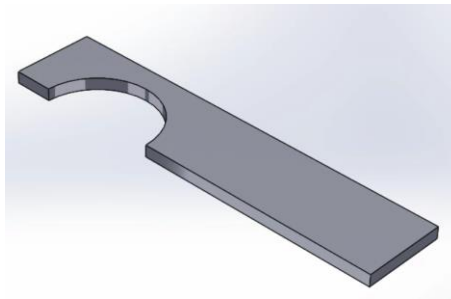


Figure 6: Second ticket mechanism design concept: a hooked arm that slides the ticket.

2.3.4 Tray Deposit

The team decided that the design with the most potential was one that held the tray in spokes, rotating to drop it onto the trash cans or into the sink. This design showed potential to be combined with other ideas that involved the wheel and peg, such as the design for a button-pressing wheel discussed earlier. The sketch of this design can be seen in Figure 7 below.

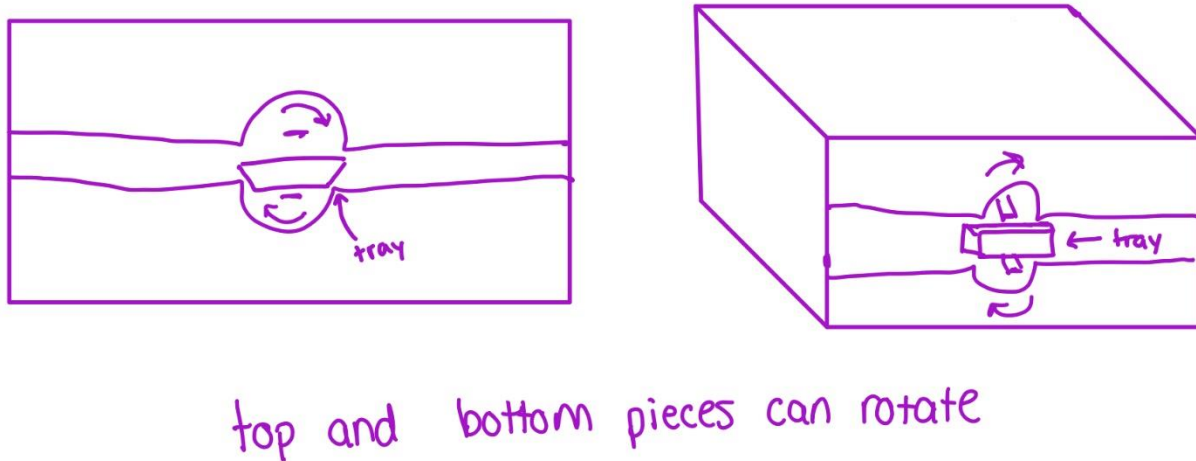


Figure 7: First tray mechanism design concept: spinning static spokes that hold and then drop the tray.

The second idea utilized a platform that would have a support be pressed inward as the robot backed into the trash can. Once the support was pressed in, it would allow the platform to hinge forward and cause the tray to slide forward onto the trash can. This idea was simple and did not require a motor to work. Its design is in Figure 8, found below on the next page.

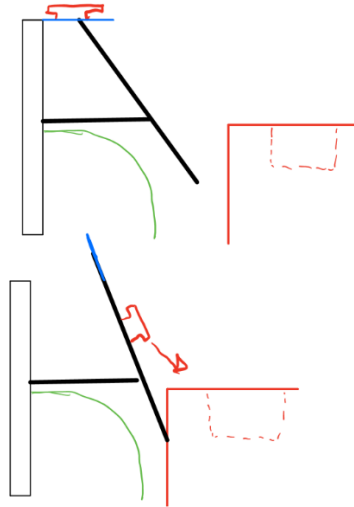


Figure 8: Second tray deposit design concept: a rotating platform that holds and then drops the tray

2.3.5 Ice Cream Lever

The team's top design choice for flipping the ice cream levers used a combination of wedges that would lower a raised lever or vice versa if driven into the lever. This, though a complex design due to the calculations needed to design it and precision to create it, would be very cost effective. It is depicted in Figure 9 below.

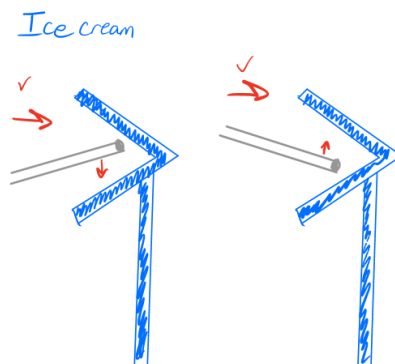


Figure 9: First ice cream lever design concept: angled wedges that force the lever up or down.

The team chose a motorized bar between two raised arms as its second choice. These motorized arms would raise or lower the arm to the appropriate height to raise or lower the lever as required. It is depicted in Figure 10, found below.

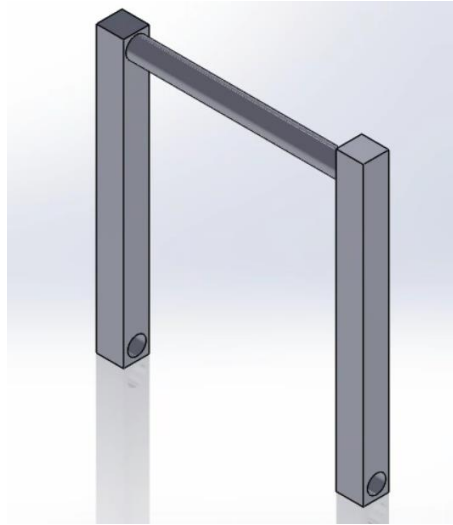


Figure 10: Second ice cream lever design concept: a bar that can be raised or lowered to push the lever up or down.

2.3.6 Burger Crank

The top design for flipping the burger featured a wheel with a peg design as described in Figure 5. This wheel would put its extended peg into one of the holes of the crank of the burger mechanism, turn 90° to flip the patty, and then flip 90° back to put the hot plate back in the starting position. Though the size would be slightly different, it would look exactly like the design in Figure 5.

The second-best design had an arm that would grab the hot plate's handles on its side. The arm would then rotate with an axis concentric to that of the hot plate's hinge, flipping the burger. It would then be used to push the plate back down when the robot drives around it to release its hold. This design is shown in Figure 11, found below.

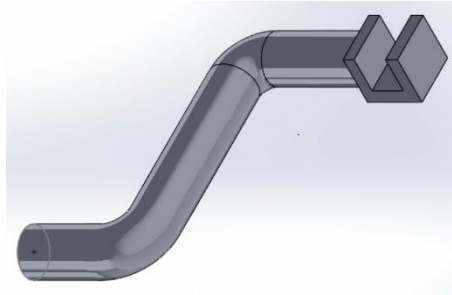


Figure 11: Second burger crank design concept: a hooked arm that grab the plate handles and rotate the plate.

2.3.7 Final Button

The size of the button and its low resistance to being pressed made the designs for this task simple. The team's top choice was the lack of any special mechanism to press the final button, allowing for resources to be allocated elsewhere.

The second choice was to have a simple arm or peg that stuck out of the front of the robot that would be dedicated to pressing the final button. Its design would be similar to that of the ticket arm in Figure 5.

2.3.8 Strategy

The strategies for navigating through the course can be seen in Figure 12 on the following page. The team chose them because they required the least amount of turning. Additionally, they

allowed for the greatest flexibility as the appropriate mechanisms could be placed on opposite sides of the robot to save time.

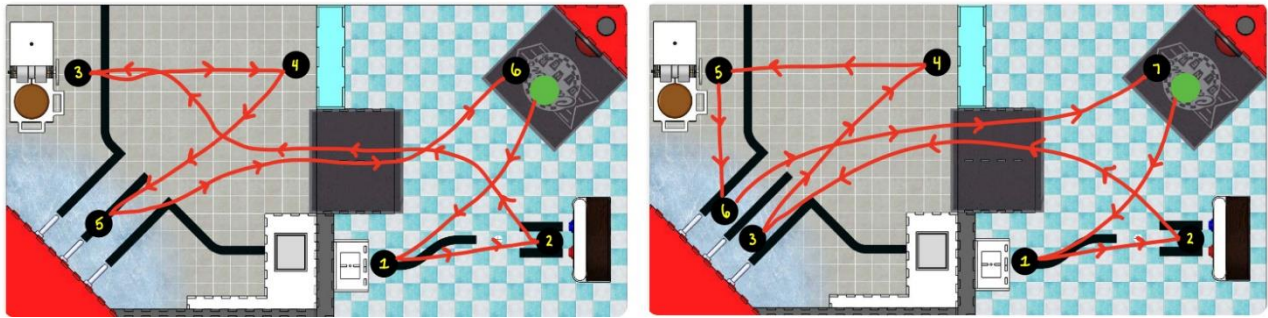


Figure 12: First (left) and second (right) choices for the route the robot would take.

2.3.9 Final Design

The final design the team chose utilized a variety of first and second picks from the drivetrain/chassis and different objective mechanisms. A comprehensive sketch of the final design can be found below in Figure 13. Additionally, the team used this final design to create a SolidWorks mockup and a cardboard mockup of the robot to better understand the size of the mechanisms and how much room it would have to work with. Both mockups can be found below in Figures 3A and 4A, respectively, in Appendix A.

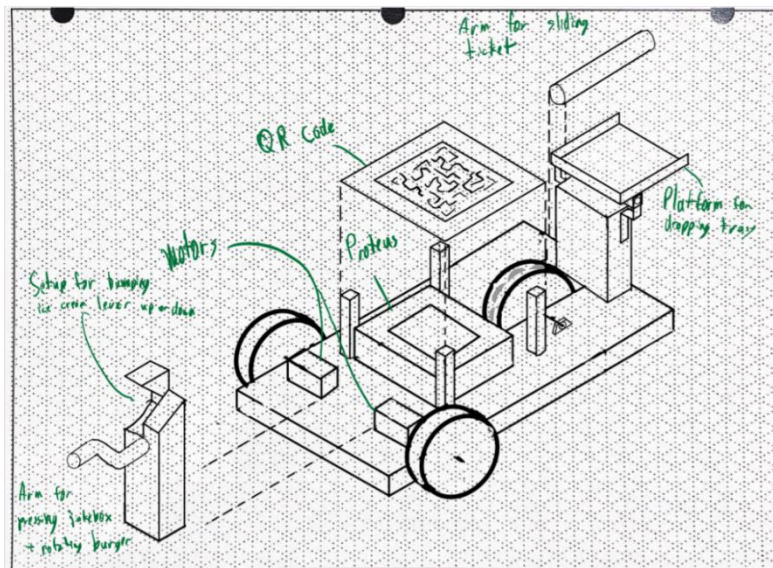


Figure 13: Initial final design as a result of brainstorming.

2.4 Preliminary Code

The team did not synthesize as detailed primary code designs as mechanism designs. However, the team did come up with a few guiding principles to follow as the code became more and more developed. In an effort to maintain as many singular points of control for the program and increase modularity, the team decided to write as many functions as it could for different areas of the robot. For example, the team knew immediately that a drive function, left turn function, and right turn function would be incredibly helpful in writing and understanding different programs. Additionally, the team also knew that functions for the different mechanisms would be extremely useful in writing the program for completing the entire course. Unfortunately, the team did not yet have enough knowledge of the syntax for the code or a specific idea of how the Proteus communicated with the motors. This prevented the team from creating any in-depth pseudocodes or flowcharts for the different methods. However, the team created an overall flowchart consistent with the strategy the team decided the robot would employ for completing the entire course. This flowchart can be found in Figure 5A in Appendix A.

3. Analysis, Testing, and Refinement



This section details the process of going from the preliminary concepts to the final design and any analysis, calculations, or tests performed. Additionally, it explains what was learned from the tests and how products were refined after tests.

3.1 Preliminary Concepts to Final Design

In the process of building the robot, the team made a few design changes to account for certain constraints or circumstances that popped up. The team altered four aspects of the robot: the tray deposit mechanism, the ticket sliding arm, the jukebox button/burger crank arm, and the chassis.

3.1.1 Tray Deposit Mechanism

As seen above in Figure 13, the original design for the tray deposit mechanism placed the tray on a platform that was controlled by a servo motor. When the time came, the motor would rotate and drop the tray onto the trash cans or into the sink. However, the team realized that there was a simpler option that was not only cheaper, due to less materials and a cheaper motor, but also easier to construct. The team designed a new mechanism where the tray sat on a slanted platform and was held back by the arm of a servo motor. When it came time, the motor rotated the arm away and the tray would slide down onto the trashcan or into the sink. The new mechanism can be seen in Figure 14 on the following page.

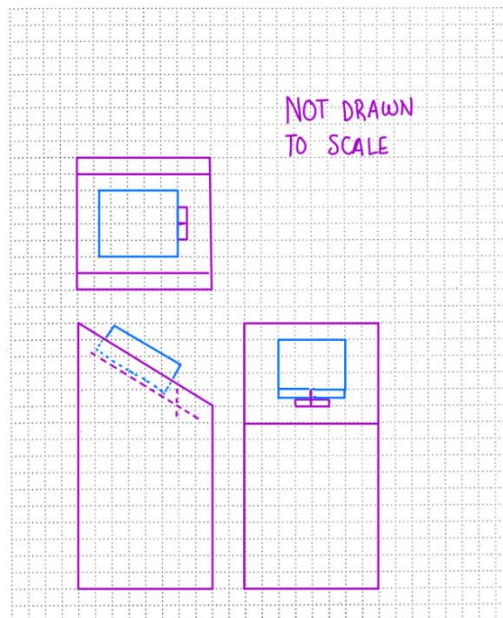


Figure 14: A sketch of the final tray mechanism used on the robot.

3.1.2 Ticket Sliding Arm

As demonstrated in Figure 13, the team planned for the ticket arm to be a rod that extended out of the back of the robot. To slide the ticket, the robot would line the arm up correctly, back up until the arm was between the ticket and the wall, and then turn to the left and slide the ticket. However, the team discovered through building the cardboard mockup that in placing the arm in that location and orientation, the robot would violate the 9" x 9" size constraint. To solve this problem, the team designed a new ticket sliding mechanism shown in Figure 15 on the next page. It would be standing up before the start of the run so as to not violate the size constraint and would then drop once the run began. To slide the ticket, the robot would turn to the right, slide the arm in between the ticket and the wall, and drive forward.

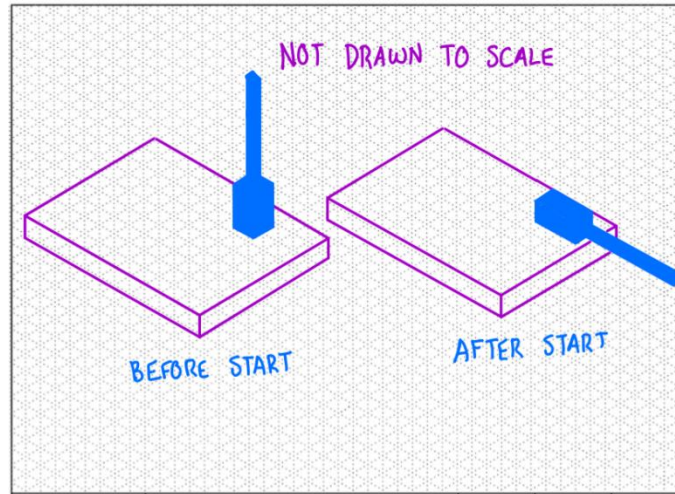


Figure 15: A sketch of the final ticket arm design implemented on the robot.

3.1.3 Jukebox Button/Burger Crank Arm

Figure 13 shows the initial concept for the arm that was to press the correct jukebox button and turn the burger crank. Originally, the team planned to design it in SolidWorks and 3D print it as a custom part. However, the team noticed that it had a surplus of wood and would cost less money to take the correct measurements, build the arm out of wood, and attach it to the servo motor. The new arm, shown in Figure 16 below, had two prongs cut into it. The outermost prong pressed the jukebox button. The innermost prong allowed the arm to fit into the burger crank and provided more surface area to relieve the force on the outer prong when turning the crank.

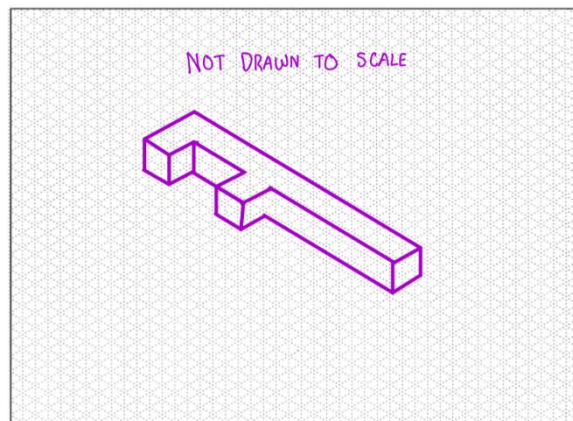


Figure 16: A sketch of the final jukebox/burger crank arm design implemented on the robot.

3.1.4 Chassis

Finally, throughout the course of the build, the team made a few minor changes to the chassis design itself. First, the team raised the main platform two inches to allow the team to place a CdS cell underneath it to read different colors of lights. Additionally, to provide more room for the mechanisms, the team placed the motors underneath the main platform and moved the omnidirectional wheel closer to the back. The team also rounded off edges of the main platform so as to not damage the course in any way. Finally, the team built the QR code holder using two supports rather than four because it cost less and performed just as well as four supports.

3.2 Analysis and Calculations

The first calculations the team made dealt with the course itself. Specifically, the team measured the height and circumference of the burger crank, the height of the ice cream levers at the up and down position, the height of the jukebox buttons, the height of the trash can and sink, and the distance between the ticket and the side of the wall. However, as time progressed, the analyses and calculations increased in complexity. The team performed multiple experiments, analyses, and calculations to improve the robot's ability to move around the course, respond to certain color lights, navigate to each mechanism, and complete each course objective. The team worked with microswitches and CdS cells, different drivetrain data, shaft encoding, line following, RPS, and data logging.

3.2.1 Microswitches and CdS Cells

The team worked with microswitches and CdS cells to analyze a way to move about the course and read different color lights, respectively. During this analysis, the team used a provided sample robot and wrote a program to move the robot through a small maze using the microswitches. The team learned that the microswitches could be used for squaring off against a

wall or lining a robot up with an objective on the course. Additionally, the team worked with a provided servo motor and CdS cell and wrote a program that rotated the servo based on how much light the CdS cell detected. The team then placed colored filters over the CdS cell and recorded the voltages the cell output. This helped the team discover which filter provided the greatest differences in voltage between the three types of light used on the course: no light, red light, and blue light. The green filter provided the greatest differences and the team placed it over the CdS cell for the remainder of the build.

3.2.2 Drivetrain Data

The team worked with the drivetrain data to perform calculations to determine what motor would best suit the team's robot. To accomplish this, the team measured out an estimate of the distance the robot would travel to complete the entire run. Then, the team estimated how much time it would take to complete each task, added the times together, and then subtracted that from two minutes. Using those values, the team calculated the average linear speed the robot needed to have to complete the run in two minutes using Equation 1.

$$\text{linear speed} = \frac{\text{distance}}{\text{time}} \quad (1)[3]$$

Using the diameter of the wheels, the team converted the linear speed to rotational speed in rotations per minute using Equation 2.

$$\text{rotational speed} = \frac{\text{linear speed}}{\text{wheel circumference}} * \frac{2\pi \text{ radians}}{1 \text{ revolution}} * 60 \text{ seconds} \quad (2)[3]$$

Next, the team estimated the total mass of the completed robot. Then, the team measured the angle of the ramp on the course. Using the angle, the mass of the robot, and Equation 3, the team determined the force needed from the motors to propel the robot up the ramp.

$$F_{motors} = W_{robot} * \sin(\theta) + F_{internal\ friction} \quad (3)[3]$$

The team then used Equation 4 to find the total torque needed and then divided the torque by the number of motors to find out how much torque per motor the robot required.

$$\tau_{motors} = F_{motors} * r_{wheel} \quad (4)[3]$$

Using this value and the speed-torque graph found in Appendix B, the team determined that the IGWAN motors best fit the robot's needs.

3.2.3 Shaft Encoding and Line Following

The shaft encoding and line following analysis provided the team with another alternative to course navigation more exact than microswitches. For shaft encoding, the team calculated the number of counts needed to travel one inch. Using that value and a provided robot with shaft encoders attached, the team wrote a program that drove the robot short distances and completed several 90° turns. This could be used for when the team needed the robot to travel a very specific distance or make more exact turns. The line following portion provided the team with a more precise method of navigating to specific course objectives. To accomplish this, the team performed two tests. The first utilized only one optosensor and a small program that had the robot drive in a zig-zag motion based on if the cell could detect the black line or not. The second test utilized three optosensors and a program that had the robot follow a smooth path. The robot would self-correct depending on which two cells could not see the black line.

3.2.4 RPS and Data Logging

The final analysis the team performed surrounded RPS, or Robot Positioning System, and data logging. The purpose of testing RPS was to provide a supplemental tool for navigation. By implementing provided code, the team learned how the RPS system used a coordinate system to track the robot's location. Additionally, the team learned how to use this feature to its advantage

to write code that would help the robot self-correct during the run. For example, if the robot turned and was not at the correct degree, it would keep slowly turning until it reached it. The team's work with the data logging system provided an additional tool for error analysis. The team learned that the data logging system stored data from the entire run and that after the run, this data could be analyzed to provide a quantitative reason why the robot performed the way it did.

3.3 Tests

Even though the COVID-19 pandemic cut the building process short, the team performed several tests throughout the robot build process. Before building, the team created a testing log format that was to be followed every time any team member(s) performed a test on any aspect of the robot. This structure included several key items that needed to be recorded: who did the testing, what was tested, the date, the location, the number of tests, the time needed, the purpose of the testing, the outcome, and any changes made to the robot as a result of the testing. The team deemed these items as necessary for many reasons, with the influential ones being repeatability, validity, clarity, and documentation. An example of a testing log can be found below in Figure 17.

Who Did the Testing	Harrison, Zach, Cam
What was Tested	Driving portion of performance test #1 code
Date of Testing	2/18/2020
Location of Testing	Hitchcock 214
Number of Tests	6
Time	3:10pm - 3:30pm
Purpose for Testing	Determine if the wheels operated as they were supposed to
Outcome of Testing	They did not. The code was skipping the turning functions
Resulting Changes	A mathematical statement calculating the number of counts for the IGWAN motors was resulting to 0 because of integer arithmetic. This was causing the code to skip over a while loop allowing the robot to turn. Eliminating this problem allowed the robot to turn.

Figure 17: An example of a testing log and the format the team followed.

The team made a wide array changes as a result of testing, from robot mechanisms to software to the starting position of the robot. Table 1C, found in Appendix C, summarizes the purpose, outcome, and resulting changes from all the tests the team performed. In addition to completing several smaller tests throughout the build, the team performed three special tests, called Performance Tests, that analyzed key components of the robot.

3.3.1 Performance Test #1

The first performance test dealt with the robot's ability to detect what color of the light in front of the jukebox, press the correct jukebox button, and navigate to and up the ramp. The team ran multiple smaller tests before completing the performance test in its entirety. First, the team performed a test on the robot's ability to drive the correct route for the performance test. During this test, the robot drove and turned twice as far as it needed to, and the team changed the code calculating the shaft encoder counts as a result. The team then tested the driving route of the performance test again and received unreliable results. Sometimes the robot traveled the route perfectly, and other times it did not. After much analysis, the team discovered that the type of wires used to connect one of the IGWAN motors was not motor wire and inhibiting the full current from the Proteus from reaching the motor. After replacing the wiring, the robot drove the route correctly. Finally, the team performed the performance test, and received unreliable results once again. Proteus experts made the team aware that the Proteus was low on power and likely not sending the specified amount of current to the motors. After the team charged the Proteus, the robot completed the performance test perfectly.

3.3.2 Performance Test #2

The second performance test concerned the robot's ability to deposit the tray into the sink or onto the trashcan and slide the ticket to its final position. As a result of the first performance test, the team did not perform any preliminary smaller tests to analyze smaller aspects of the robot. Instead, the team wrote and implemented a code designed to complete the performance test. Unfortunately, it took multiple runs for the robot to complete the test. The robot consistently dropped the tray on top of the trashcans as the team intended, but it didn't slide the ticket arm into place. It took multiple refinements of the turning code for that portion of the test before the robot slid the arm into place and moved the ticket to its final position.

3.3.3 Performance Test #3

The third and final performance test dealt with the robot's ability to rotate the burger crank and drive to an ice cream lever and flip it down. The team worked diligently to prepare for this performance test, as it concerned a difficult aspect of the robot: RPS. In this performance test, the robot needed to travel to the ice cream lever using only RPS as a guide. As with the first performance test, the team performed preliminary testing on the robot's RPS capabilities. The team tested the robot's ability to read in and display RPS values and then used that ability to map out the route the robot needed to take. The team then used those values to test the robot's ability to drive the correct route, and it did. The team then moved on to completing the performance test. Much to the team's surprise, the robot completed the test within the first couple of attempts. However, in attempting to achieve a more consistent performance, the team refined the RPS code. This caused the robot to not complete the performance test at all and the team had to revert to the original code for it to complete it again.

4. Current Status of the Prototype

The following sections describe several areas pertaining to the build at the end of development and testing. The first is how well the robot met the specifications laid out at the start of the build. The second describes the physical robot itself and the code for it. The third is how much time and money went into the build and whether development was on schedule and above or below budget. Finally, the last subsection details the robot's strengths and weaknesses.

4.1 Specifications

As of now, the robot meets many of the specifications laid out before the build began. First and foremost, it satisfies the 9" x 9" x 12" size constraint. Additionally, the robot satisfies all the specifications surrounding the Proteus, such as only using Velcro to secure it to the chassis and not using it as a sensor, structural support, or mechanical element. The robot also meets the myriad of other structural specifications, such as being completely autonomous, only receiving wireless signals from RPS, not using adhesive materials as structural support, never performing any action that would harm the course or another robot, and utilizing no more than three omnidirectional wheels. In addition to structural specifications, the robot meets several mechanism specifications as well. The jukebox button, tray deposit, and final button mechanisms complete their objectives consistently, quickly, and efficiently. However, there are some mechanism specifications that the robot does not meet.

The robot has no mechanism for flipping the ice cream lever. There is an edge on the tray deposit mechanism that inconsistently flips the lever down, but there is no aspect of the robot that can flip it back up. Therefore, the robot cannot complete the ice cream lever objective. Additionally, the robot cannot consistently or efficiently turn the burger crank or slide the ticket. The rotating arm will rotate the burger crank, but the weight of the burger pulls the arm off the

servo motor incrementally until the arm can no longer rotate the crank and eventually rips the arm off the motor entirely. The robot is also inconsistent in sliding the ticket. It has only slid it successfully a few times and requires great precision and a small amount of luck to even slide the arm into place. Finally, while the robot can successfully communicate with RPS, it does not report consistent values. Therefore, any code utilizing RPS, such as positioning and turning checks is flawed and performs unpredictably.

4.2 The Robot Itself

There are two major components to the robot: the physical robot itself and the code it uses. The physical robot has different course objective mechanisms and several custom parts, including the drivetrain/chassis. The code the robot uses includes a myriad of different methods and three different main codes for the performance test. There is no code for any competitions, as they did not occur. A drawing of the robot in exploded and condensed view can be found in Figure 1D in Appendix D, as well as all other figures mentioned in this section.

4.2.1 Physical Robot

The robot contains mechanisms for 5 out of the 6 different course objectives: the jukebox button, tray deposit, ticket slider, burger crank, and final button. Several of them utilize custom parts to attach to the robot. Because of when testing and development ended, the robot does not have a mechanism for flipping the ice cream lever.

4.2.1.1 Jukebox Button/Burger Crank Mechanism

The jukebox button and burger crank mechanism, found in Figure 2D, is comprised of the arm pictured in Figure 3D glued to a servo motor, which is attached to the chassis via a wooden custom holder and placed at the front of the robot. It operates by rotating to the correct jukebox

button and pressing it or by sliding into a slot in the burger crank and rotating it. Drawings for the custom holder and the chassis can be found below in Figures 4D and 5D.

4.2.1.2 Tray Deposit Mechanism

The tray deposit mechanism, seen below in Figure 6D, is comprised of a micro servo motor and a custom wooden tray holder, which is placed near the rear of the robot. The pieces that make up the tray holder can be found in Figures 7D and 8D, respectively. The tray is placed in the holder and the axle of the motor stops it from sliding out. To drop the tray, the motor simply rotates the axle and the tray then drops out of the holder.

4.2.1.3 Ticket Slider Mechanism

The ticket slider mechanism, located in Figure 9D, consists of a wheel axle inserted into a custom wooden sheath, pictured in Figure 10D, that is screwed onto the chassis near the rear of the robot. To slide the ticket, the robot approaches it at a 90° angle and then turns right, sliding the arm in between the ticket and the wall. The robot then drives forward, sliding the ticket as it does so, and then backs up and turns left once the ticket is as far as it will go.

4.2.1.4 Final Button Mechanism

Unlike the others, there is no defined final button mechanism, as several different parts of the robot can press the button. The robot could back into the button and the tray deposit structure would press the button or the robot could drive forward and the jukebox button/burger crank arm would press the button. As none of the performance tests required the robot to press the button, the team planned to decide how it would press the button as the only when the competitions drew nearer.

4.2.1.5 QR Holder

In order to communicate with RPS, the robot had to have a QR code mounted at least 9” above the course surface. The mechanism for holding the QR code, found below in Figure 11D, consists of several screws and erector set pieces as well as a 3” by 3” piece of wood. The team glued the QR code onto the piece of wood. The mechanism is placed on the structure such that the QR code sits directly above the Proteus.

4.2.2 Code

In an effort to create a modular code, the team wrote a number of different methods to complete different robot functions. The team wrote a method for driving a specific distance, driving for a specific time, driving up the ramp, turning right, turning left, turning right with one wheel, turning left with one wheel, checking the heading, checking “x” and “y” coordinates, and pressing the right jukebox button. All figures mentioned in this section can be found in Appendix E.

4.2.2.1 Driving Functions

The driving a specific distance function takes a distance in inches and a speed as inputs and its code can be found in Figure 1E. It sets the motors to drive at a specific speed until the encoders determined the robot drove the correct distance. The driving for a certain time method takes a time in seconds and a speed as inputs and its code can be found in Figure 2E. It starts a timer and set the motors to a certain speed until the timer reached the time passed into the function. The driving up the ramp function takes a speed as an input, sets the motors to that speed, and increments the speed as the robot drives up the ramp. The team wrote this function because robot experts advised the team that it would place less stress on the motors if utilized. Its code is located in Figure 3E.

4.2.2.2 Turning Functions

The left and right turn functions operate in a similar manner to each other. Both take an angle in degrees and a speed as inputs and turn the robot the specified amount of degrees. The right wheel moves forward and the left wheel moves backwards for the right turn function and vice versa for the left turn function. Their codes can be found in Figures 4E and 5E, respectively. The turning right and left with one wheel methods take the same inputs as well, but only set the right or left motor, respectively, to a certain speed. Their codes can be found in Figures 6E and 7E, respectively.

4.2.2.3 Checking Functions

There are four functions that checked “x” and “y” coordinates. They all take a desired coordinate as an input and move the robot forward or backward depending on if the robot was behind or in front of the coordinate. There is a function for both the “x” and “y” coordinates in the case that the robot is moving in the positive direction and in the case that the robot is moving in the negative direction. Their codes can be found in Figures 8E, 9E, 10E, and 11E, respectively. The check heading function takes a heading in degrees as its input and turns the robot to the correct degrees if it is not already there. Its code is located in Figure 12E.

4.2.2.4 Jukebox Button Function

The method that presses the right jukebox button takes no inputs. It is called when the robot is positioned over the light in front of the jukebox. It uses the CdS cell attached to the bottom of the robot to determine what color the light is, rotates the arm to the correct position, drives forward to press the button, and then backs up to exactly where it was. The code can be found below in Figure 13E.

4.2.2.5 Performance Test Main Codes

The main codes for the three performance tests utilized many of the different functions outlined above. They all implemented at least a drive function and one turning function. Someone of them also used more specific functions as well. The first performance test used the jukebox button function and the third performance test used all four of the coordinate checking functions as well as the check heading functions. The codes for all three performance tests can be found in Figures 14E, 15E, and 16E, respectively.

4.3 Time, Money, and Development

There are three specific areas of the robot that were and are integral to construction: time, money, and development. Time is the most important, because it has to spread between all areas of the robot, such as documentation, coding, building, modeling, and testing. Money is also critical, as it is the first hurdle to jump when building the robot. Finally, the current development of the robot is also important, as what is underdeveloped, finished, and on schedule greatly affects what the team will work on.

4.3.1 Time

The team members split their time between five major areas: documentation, project management, coding, testing, CAD/modeling, and building/construction. Each member split his or her time according to what area he or she was responsible for. For example, those in charge of building or construction invested more time into building/construction and CAD/modeling. Additionally, as the build progressed, different areas took precedence over the others. For example, at the start of the build, the team spent much time brainstorming and building the robot. As such, the team spent a lot of time in CAD/modeling and building/construction. Found below,

Figure 18 provides a summary of the time the team spent in different areas of the build outside of scheduled project meetings.

Out of Class									
Team Member	Total Hours		Documentation	Project Management	Coding	Testing	CAD	Building/Construction	Other
Zach	58		12	3.75	3.5	9.25	16	13.5	0
Liam	33		7	3	8	9	2	4	0
Cam	39.5		11	4	2	10	3	9.5	0
Harrison	67		38.25	9.75	3.5	3.5	4	8	0
Team Total	197.5		68.25	20.5	17	31.75	25	35	0
In Class						Total = Out of Class + In Class = 231.5			
Number of Robot Classes		Class Time (hours)		Total					
17		2		34					

Figure 18: Summary of total time the team spent in different areas of the project.

4.3.2 Money

The team had a budget of \$160 when the build began and planned to have \$40 remaining at the end of the build to compensate for any unseen costs or challenges that arose. To achieve this goal, team scrutinized every aspect of the robot and strove to minimize costs wherever possible. Figure 19 below shows the team's spending habits as the build progressed.

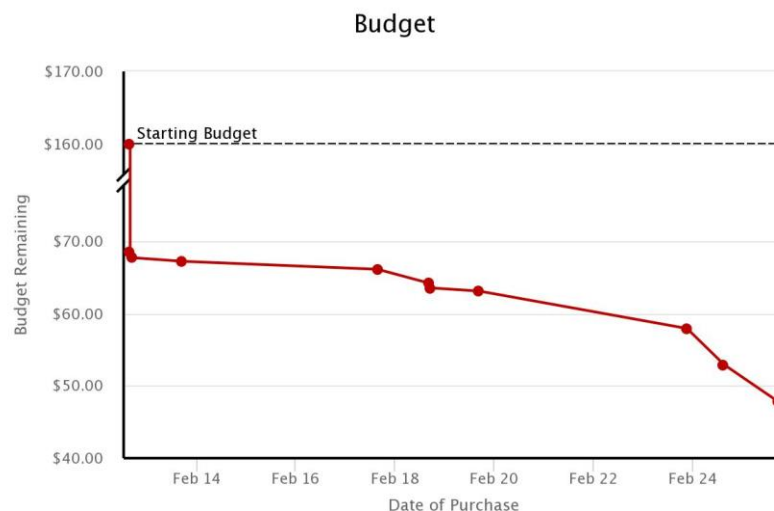


Figure 19: Summary of team's spending up until development and testing ceased [5].

At the time development and testing ended, the team had \$47.82 left to spend. The last aspect of the robot left to build is the ice cream lever mechanism. Table 6 below provides all the pieces the team planned to buy to build the mechanism and their prices.

Table 6: Parts for building ice cream lever mechanism and their prices.

Part	Number	Price	Total
1x0.5" Angle Bracket	2	\$0.50	\$1.00
Micro Servo Motor	1	\$5.00	\$5.00
Total			\$6.00

As Table 6 shows, the total cost would not bring the team under the \$40 goal set at the beginning of the build and thus the robot is underbudget, but only by \$1.82. However, setting the team's goal aside, the robot is well underbudget by \$41.82.

4.3.3 Development

It was critical that the team stay on schedule for all aspects of the build, as falling behind in one would've caused the team to fall behind in the others. However, there were certain areas that allowed for more leeway than others. For example, the schedule for creating the website was more flexible and allowed the team more freedom to choose when to start certain items. The schedule for building the robot, however, was more rigid and structured and had concrete starts and deadlines for different items. Fortunately, up until development and testing ended, the team stayed on schedule. The team completed all objectives in the more flexible items, such as the website, CAD, and coding, on time. Additionally, the team finished all objectives in the more rigid aspects, such as documentation and building/construction, on schedule. For example, the team finished the first two performance tests on time. Furthermore, the team completed the third performance test a day early, providing a full week to build the ice cream mechanism and refine the RPS code and putting the robot's development ahead of schedule.

4.4 Strengths and Weaknesses

Even though the COVID-19 pandemic prevented the team from building the robot, the robot still has many strengths that it utilized before development and testing ended. First and foremost, it is versatile. Many of the parts the team designed can or have the potential to complete more than one course objective. Also, several of the course objective mechanisms work reliably and efficiently, such as the tray deposit, jukebox button, and final button mechanisms. Additionally, many of the parts are replaceable and attachable. For example, the burger crank/jukebox arm is not only easily replaceable but can be reattached quickly and effectively. Finally, the strategic placement of the axis of rotation in front of the center of the robot allows it to navigate the course and complete the objectives in a unique way. For example, when sliding the ticket, the robot can simply drive forward and then turn 90° to slide the ticket arm into place, rather than approaching at an angle, which would be required if the axis of rotation was in the center of the robot. However, there are aspects of the robot's navigation, as well as certain mechanisms, that are a detriment to the robot rather than a benefit.

Because the pandemic stopped development, the robot also contains weaknesses that the team would have eradicated. The first two lie within the code and the last two lie within the robot mechanisms. Concerning the first coding issue, the positioning code is inconsistent. The team utilizes RPS in self-check algorithms for positioning and turning, and they do not perform accurately or reliably. Sometimes they overcorrect and other times they do not correct enough. Additionally, the team wrote special code to compensate for a weak left motor. Despite this, the robot drifts to the left when driving forward or backward and turns inconsistently when relying on the left motor, indicating that the compensation code is not working correctly. Concerning the robot's mechanisms, the ticket slider and burger arm are unreliable. The robot has only fully slid

the ticket a few times, and it takes great precision to even place the ticket arm in the correct location. The burger arm can successfully rotate the burger crank. However, the weight of the burger will slowly weaken the glue holding the arm in place until the arm can no longer turn the crank and eventually falls off.



5. Future Development

The next three subsections concern any future development of the robot. The first subsection details any projected necessary resources for the team to complete the robot and how those resources would be allocated. The second subsection explains any challenges the team anticipates facing if development were to continue. The third and final subsection outlines the projected strengths the team believes the robot would have if completed.

5.1 Projected Necessary Resources

In the event that development and testing were to continue, the team would need a few resources. The first is personnel. The team needs to be re-assembled exactly the way it was before testing and development ended. The second is materials. There are a number of aspects of the robot that need additional work and a few that haven't been built yet, and the team needs the materials to do so. Finally, the third and most important resource the team would need is time. It needs time to build, code, refine, and test.

5.1.1 Personnel

One resource the team would need if the development and testing were to continue is personnel. More specifically, the team would need all four of its members to continue construction. Each member plays an integral role in different aspects of the project, and the loss of any one of them would severely damage the team's ability to stay on schedule. Additionally, it is important that no one team member is replaced. Each member brought his or her own strengths to the table and as the build progressed, the team learned how to effectively utilize those strengths. Also, the team learned how to work efficiently and communicate effectively in the shortest amount of time possible. If a member is replaced by another, the team will have to start that process again and development will be slowed.

If the team is reassembled the way it was before development ended, it will lead to a completed prototype. The team members will continue to play their roles in the project and improve in those roles as the build progresses. This will allow for all the different areas of the project to continue on schedule and not delay the others. Additionally, the team will continue to make use of and improve its ability to operate as a well-oiled machine. This will allow the team to work quickly and efficiently, utilizing each member's strengths and maintaining effective communication.

5.1.2 Materials

At the time development stopped, there were number of elements of the robot that needed additional construction or didn't exist at all. The most glaring of these is the ice cream lever mechanism. Currently, the robot has no mechanism to flip the ice cream lever up and down. As flipping the levers is an objective on the course, it is critical that the team builds this mechanism. The materials needed for this mechanism can be found above in Table 6. Additionally, there are some aspects of the robot that need to be refined or possibly redesigned. The arm that turns the burger crank and presses the jukebox button needs to be duplicated and a stronger adhesive material, such as epoxy, must be used to secure the arm to the motor. Furthermore, the ticket slider arm might have to be redesigned, and the team would need to have materials at its disposal if that was the case.

Provided that the team has the necessary materials, it could build a complete robot. The team would first build the ice cream lever using the Erector Set pieces and the additional motor. Next, the team would take wood and construct multiple duplicates of the burger crank and jukebox arm in the event that a replacement is needed. The team would then take the stronger adhesive material and carefully secure the arm in the correct place so that the burger crank would

no longer rip it off. Finally, if the team determined that the ticket arm needed to be redesigned, it would brainstorm and order the necessary parts and build the new arm.

5.1.3 Time

The most important resource the team needs is time. All of the obstacles facing the team when development ended will take time to overcome. As the build was nearing the close when development ended, most of the obstacles are construction and coding based. The team needs time to build the ice cream lever, write code for the ice cream lever, and test the mechanism and the code. Additionally, the team needs time to finish the code for the robot to run the whole course and refine all of the objective mechanisms. Once that occurs, the team can move onto the final stage of the build: repeatedly testing that the robot successfully completes the entire course. To accomplish all this, the team needs three weeks.

In the first week, the team will simultaneously build the ice cream lever mechanism, write the code for it, and, once those have been finished, test them to make sure they work correctly. In the second week, the team will finish the code for the overall course run, write self-checking algorithms for different areas of the course, and test and refine the individual mechanisms to ensure they all operate efficiently and quickly. Finally, in the third week, the team will run repeated tests of the robot's ability to complete the entire course. The team will try and iron out any small issues that occur and, if necessary, change the order in which the robot completes the objectives to decrease the overall time of the run.

5.2 Anticipated Challenges

The team is bound to run into unforeseen challenges if development were to continue. However, there are a few the team knows will occur. First, the design for the ice cream lever requires precise measurements for when the lever is up and when the lever is down, as well as

the length and largest diameter of the lever. Obtaining these measurements, and writing code that effectively utilizes them, will be a challenge. Another potential problem surrounding the ice cream mechanism is the Erector Set pieces. The team noticed that the metal Erector Set pieces, especially the longer ones, tend to flex and bend. This could present a challenge as the pieces may bend instead of forcing the lever up or down. An additional challenge in the area of mechanisms is the burger and jukebox arm being ripped off. If epoxy is so strong that the arm cannot be taken off if another portion of the arm breaks, then it cannot be used and the team must come up with another method of securing the arm to the motor while still allowing it to be detached. Finally, the team will face an obstacle dealing with RPS. The current RPS code is unpredictable and unreliable. It reports inconsistent values wherever it is used. This is a problem because the team has turning and positioning checks that utilize RPS. Furthermore, the self-checking algorithms the team plans to write will also rely heavily on RPS. Thus, the team must analyze the code, find out what's causing the issue, and fix it for the self-checking algorithms to work.

5.3 Projected Strengths and Weaknesses

In the event that the robot is completed, there will be multiple aspects of it that are unique and effective. First and foremost, all the objective mechanisms will be fully operational. Furthermore, many of them will be efficient, consistent, and quick. Secondly, many parts will be replaceable and versatile. For example, the robot will have an arm that can press the jukebox button, rotate the burger crank, and press the final button in a pinch. Additionally, the arm is also detachable for easy repairs and/or replacement. Another example is the tray deposit mechanism. Attached to it is the ice cream lever mechanism, allowing the one structure to deposit the tray and flip the ice cream lever. Finally, the robot will have self-checking algorithms for various

aspects of the course. For example, there will be an algorithm that checks if the robot has inserted the arm into the burger crank correctly. If the robot moves forward and isn't at the correct position, it will continually back up, adjust itself, and move forward again until it is. These types of algorithms will play an extensive role in some of the aspects of the robot that aren't as beneficial.

One of the mechanisms that is not as efficient as the others is the ticket slider mechanism. Unless the team elects to redesign it completely, the mechanism will require precise driving and turning and a robust correction algorithm to successfully complete its objective. Additionally, the robot's navigation and turning abilities are not exact, and turning and positioning checks must be utilized for them to be effective. Finally, unless they are replaced during additional development, the servo motor controlling the jukebox and burger crank arm and the left motor are at risk for breaking during the run. The jukebox mechanism relies on the arm pressing the button, which puts lateral stress on the gears of the servo motor, something which the motor was not built to handle. After repeated presses, the motor could break. Additionally, the left motor has consistently output weaker power than a fully functioning motor. While the team has written code to compensate for this, it is still at risk to break or start outputting even less power.

6. Summary and Conclusions

In an attempt to help Carmen's Diner deal with its influx of customers, maintain the qualities that make it great, and improve business, the team designed and is constructing a robot to help with menial restaurant tasks. The following sections summarize what the team completed up to the point that development and testing was forced to end, the current state of the robot, and any plans or needed resources for future development of the robot.

6.1 The Past

To begin, the team members individually brainstormed different ideas for the objective mechanisms, the drivetrain/chassis, and what route the robot would take to complete the course. The team then created design matrices for the mechanisms, drivetrain/chassis, and strategy, and used them to evaluate the different ideas and choose the top two designs. The team then utilized an additional design matrix to create three different overall concepts of the robot and select the top choice out of the three. Afterwards, the team began constructing the robot. The team built the chassis/drivetrain and jukebox button mechanism and its corresponding code first and completed the first performance test. Next, the team built the ticket slider and tray deposit mechanisms and their corresponding codes and completed the second performance test. The team's final act of construction before development ended dealt with building the burger crank mechanism and its corresponding code and completing the third performance test. Smattered between the performance tests were several exploration days, where the team exposed itself to different potential aspects of the robot, such as motor encoders, microswitches, line following, and RPS. The team also published a number of documents throughout the building process, including a summary of the brainstorming process, several exploration progress reports, drivetrain analysis,

code representation, and electrical systems. Finally, the team began constructing a comprehensive website for the build.

6.2 The Present

At the current moment, the robot is not complete. The ice cream lever mechanism is not built, some of the mechanisms need refined or possibly redesigned, and the code utilizing RPS needs to be analyzed and fixed. Also, the ticket arm mechanism is inconsistent and needs to be refined or redesigned. Finally, some of the parts on the robot are at risk for breakage, such as the left drivetrain motor and the servo motor controlling the burger crank/jukebox button arm. However, many aspects are finished, and development is on schedule. First and foremost, the robot can move about the course and satisfies all construction constraints specified at the start of the build. Second, the tray deposit, jukebox button, and final button mechanisms are built and can be relied upon to complete their respective objectives consistently and efficiently. Third, many of the parts are replaceable and versatile, such as the burger crank/jukebox arm, just as the team planned. Finally, the team is underbudget by \$47.82 and the remaining potential costs of continued development will not push the team overbudget.

6.3 The Future

If development and testing were to continue in the future, the team will complete the robot. To do so, the team needs three primary resources: personnel, materials, and time. The team needs to be reassembled the way it was before development ended. This will ensure that the team will operate at the same speed and efficiency as it did before development was stopped. The team needs materials to build the ice cream lever, duplicate the burger crank/jukebox button arm, secure the arm to the motor, and possibly redesign or refine the ticket slider mechanism, such as creating a motorized arm that extends out from the robot to behind the ticket, an idea that

would require much less complex driving. The team needs time to complete all of these objectives. Additionally, there are some challenges the team anticipates facing. First, getting the precise measurements and code to flip the ice cream levers will be difficult. Second, Erector Set pieces tend to bend, and this presents a problem if they are being used for the ice cream lever and bend instead of pushing the lever. Third, the RPS code is inconsistent and will require intense analysis and work to fix. Finally, the team must solve the dilemma of securing the burger crank/jukebox button arm in a way that allows it to be detachable but also will not succumb to the stress of lifting the burger. If the team is given the necessary resources, then it will complete the robot. The final button, tray deposit, jukebox button, burger crank, and ice cream lever mechanisms will be consistent and efficient. While it will require the precise turning and a robust correction algorithm, the ticket slider mechanism will be consistent as well. Many of the parts will be replaceable and versatile. Finally, there will be self-correction algorithms utilizing RPS to compensate for inconsistencies and variability in navigation and turning.

7. References and Resources

Below are groupings of different resources and references the team utilized in the design and building of the robot and synthesis of this report.

7.1 Online Resources

- [1] Nemick, Cassidy. “Decline of the American Diner.” *Scalar.com*, Scalar, scalar.usc.edu/works/the-evolution-of-the-american-diner/decline-of-the-american-diner.

7.2 FEH Resources

- [2] “ENGR 1282.01H Design Problem Statement & Specifications, Spring 2020.” *Carmen Canvas*, Ohio State, Feb. 2020, osu.app.box.com/s/mb14f4tncgf72me94jxp44taqj9hsvl
- [3] “Drivetrain Calculations.” *Carmen Canvas*, Ohio State, Feb. 2020, <https://osu.app.box.com/s/ud3umti4wubvuykmagxpeb009itqc2nf>
- [4] “FEH Motors Graph.” *Carmen Canvas*, Ohio State, Feb. 2020, <https://osu.app.box.com/s/73nq79bfperbgcmqwel30zd7cqgtmv0t>
- [5] “FEH Robot Store” *FEH Robot Store*, Ohio State, Apr. 2020 <https://feh.osu.edu/store/team>

Appendix A

Preliminary Concepts

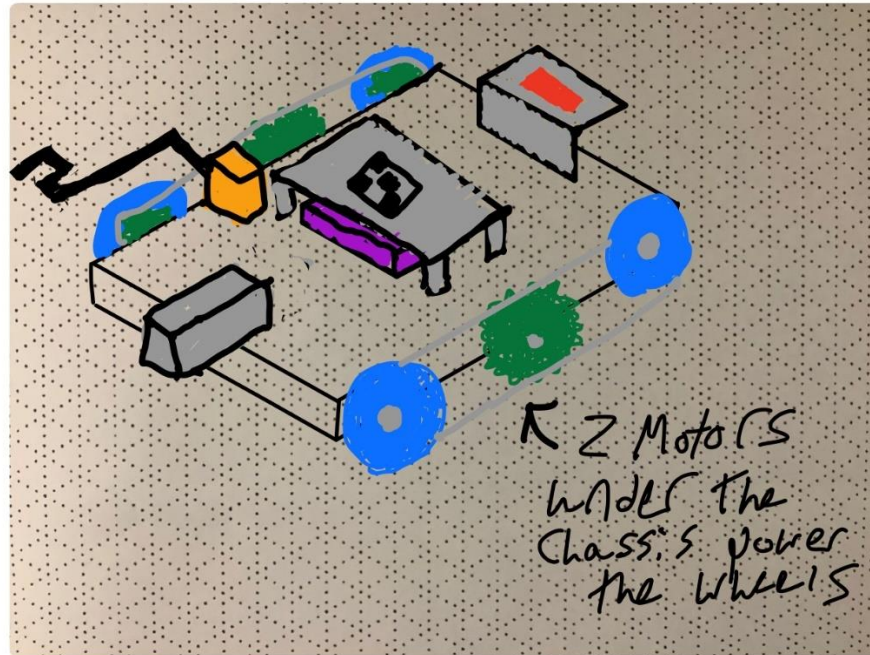


Figure 1A: Second initial final design for the robot after brainstorming.

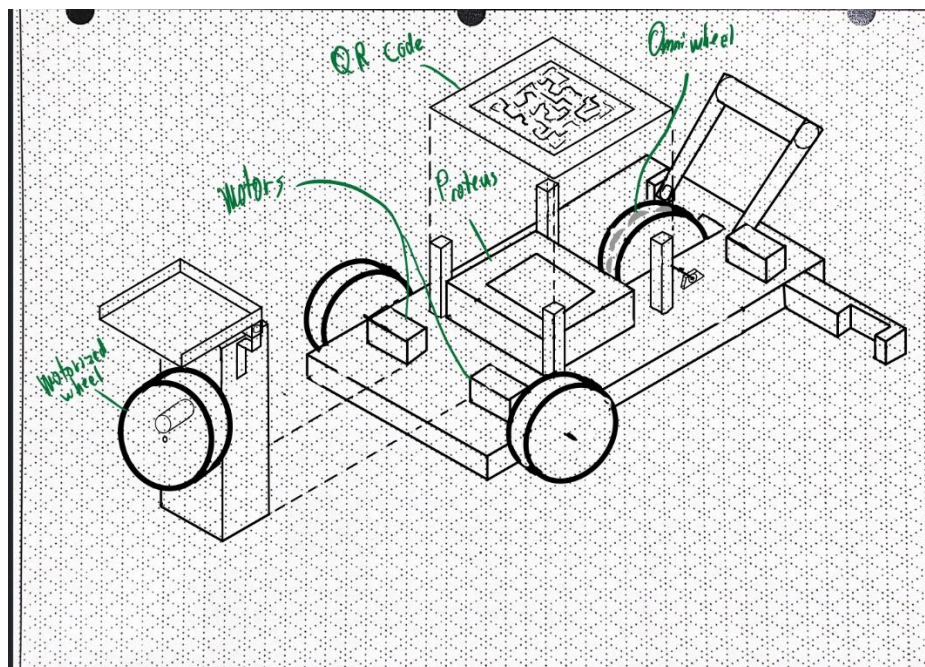


Figure 2A: Third initial final design for the robot after brainstorming.

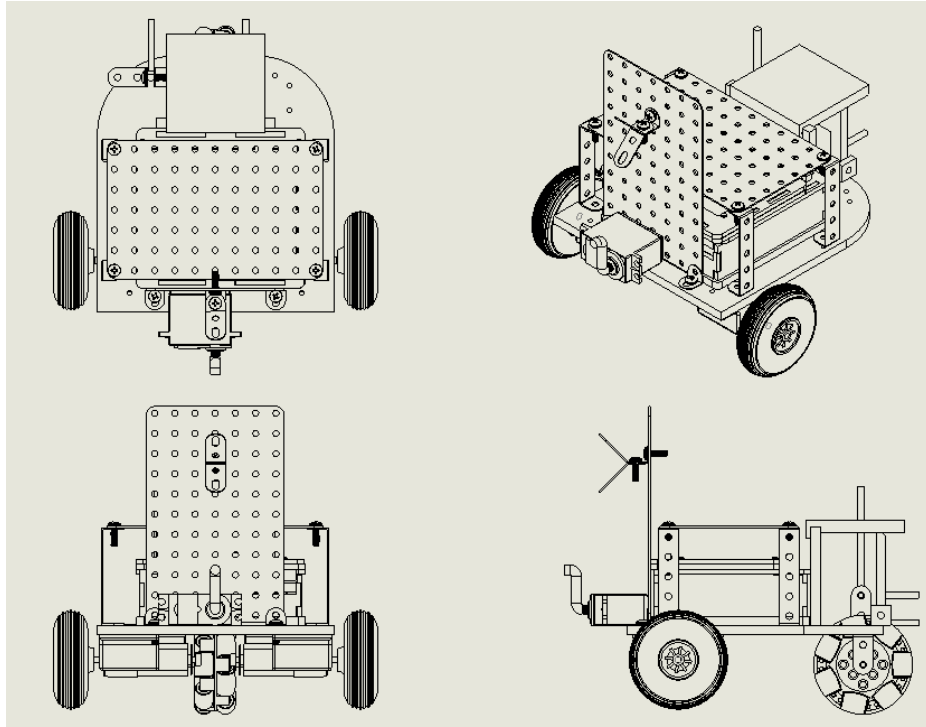


Figure 3A: SolidWorks mockup of team's final choice for an initial overall design.



Figure 4A: Cardboard mockup of team's final choice for an initial overall design.

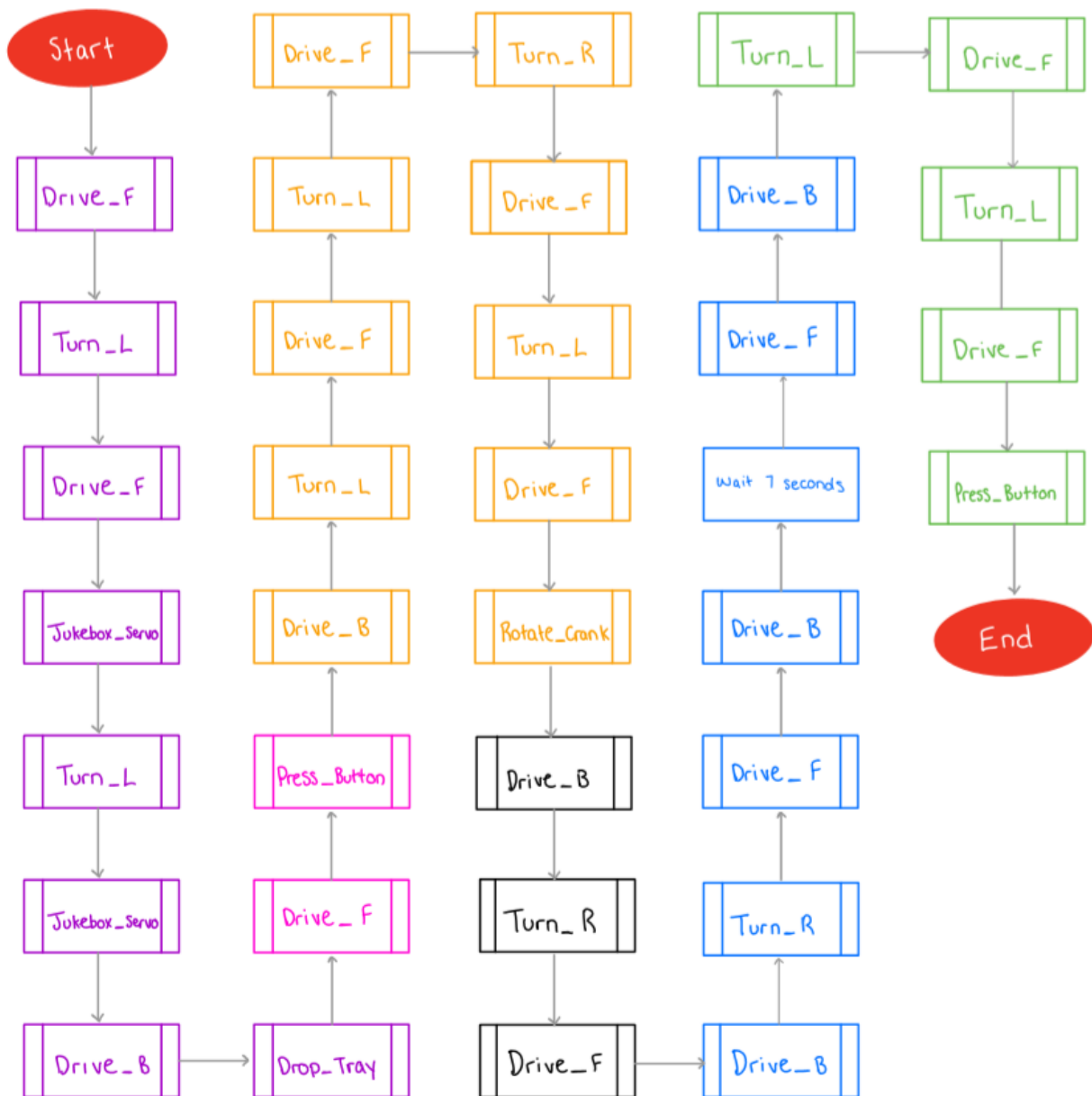
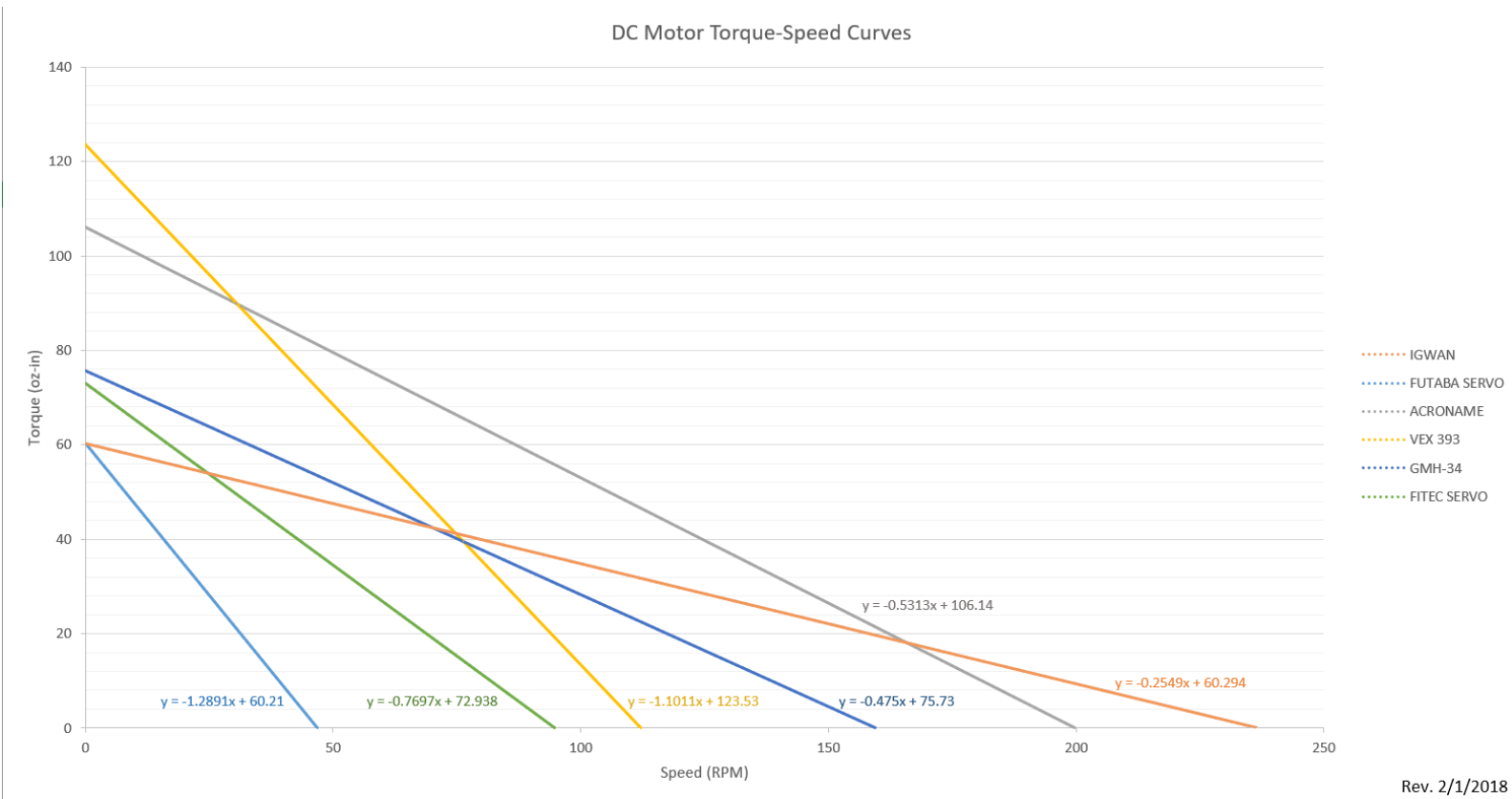


Figure 5A: The team's initial idea for the complete modular course code.

Appendix B

Analysis, Testing, and Refinement



Rev. 2/1/2018

Figure 1B: The torque-speed graph the team used to choose the motors for the robot [4].

Sample Calculation for Equation 1

$$\text{linear speed} = \frac{\text{distance}}{\text{time}}$$

$$\text{linear speed} = \frac{227 \text{ inches}}{84 \text{ seconds}}$$

$$\text{linear speed} = 2.70 \text{ in/s}$$

Sample Calculation for Equation 2

$$\text{rotational speed} = \frac{\text{linear speed}}{\text{wheel circumference}} * \frac{2\pi \text{ radians}}{1 \text{ revolution}} * 60 \text{ seconds}$$

$$\text{rotational speed} = \frac{2.70 \text{ in/s}}{5\pi} * \frac{2\pi \text{ radians}}{1 \text{ revolution}} * 60 \text{ seconds}$$

$$\text{rotational speed} = 64.8 \text{ rpm}$$

Sample Calculation for Equation 3

$$F_{\text{motors}} = W_{\text{robot}} * \sin(\theta) + F_{\text{internal friction}}$$

$$F_{\text{motors}} = 44.5 \text{ oz} * \sin(18.43) + 8$$

$$F_{\text{motors}} = 22.07 \text{ oz}$$

Sample Calculation for Equation 4

$$\tau_{\text{motors}} = F_{\text{motors}} * r_{\text{wheel}}$$

$$\tau_{\text{motors}} = 22.07 \text{ oz} * 2.5 \text{ in}$$

$$\tau_{\text{motors}} = 55.18 \text{ oz-in}$$

Appendix C

Tests

Table 1C: Summary of Purpose, Outcome, and Changes for all tests.

Test	Purpose	Outcome	Resulting Changes
1	Determine if wheels function properly	They did not	Code for calculating counts altered
2	Determine if robot drove correct route for 1 st performance test	It did not	Code for calculating distance altered
3	Determine if robot lined up at correct spots for 1 st performance test	It did not	Wiring for motors fixed
4	Determine if robot lined up at correct spots for first performance test after fixing motor wire	It did	None
5	Determine if the robot read the correct color lights and moved the servo motor accordingly	It Did	Proteus charged after many inconsistencies
6	Determine if robot traveled correct distance utilizing both encoders	It did	None
7	Determine if robot dropped the tray and made it up the ramp	It did not	Turn counts recalculated and motors recalibrated
8	Determine if robot completed 2 nd performance test	It did not	Turning code for ticket portion refined
9	Determine if robot completed 3 rd performance test	It did	Code for turning burger crank refined

Appendix D

Physical Robot Mechanisms

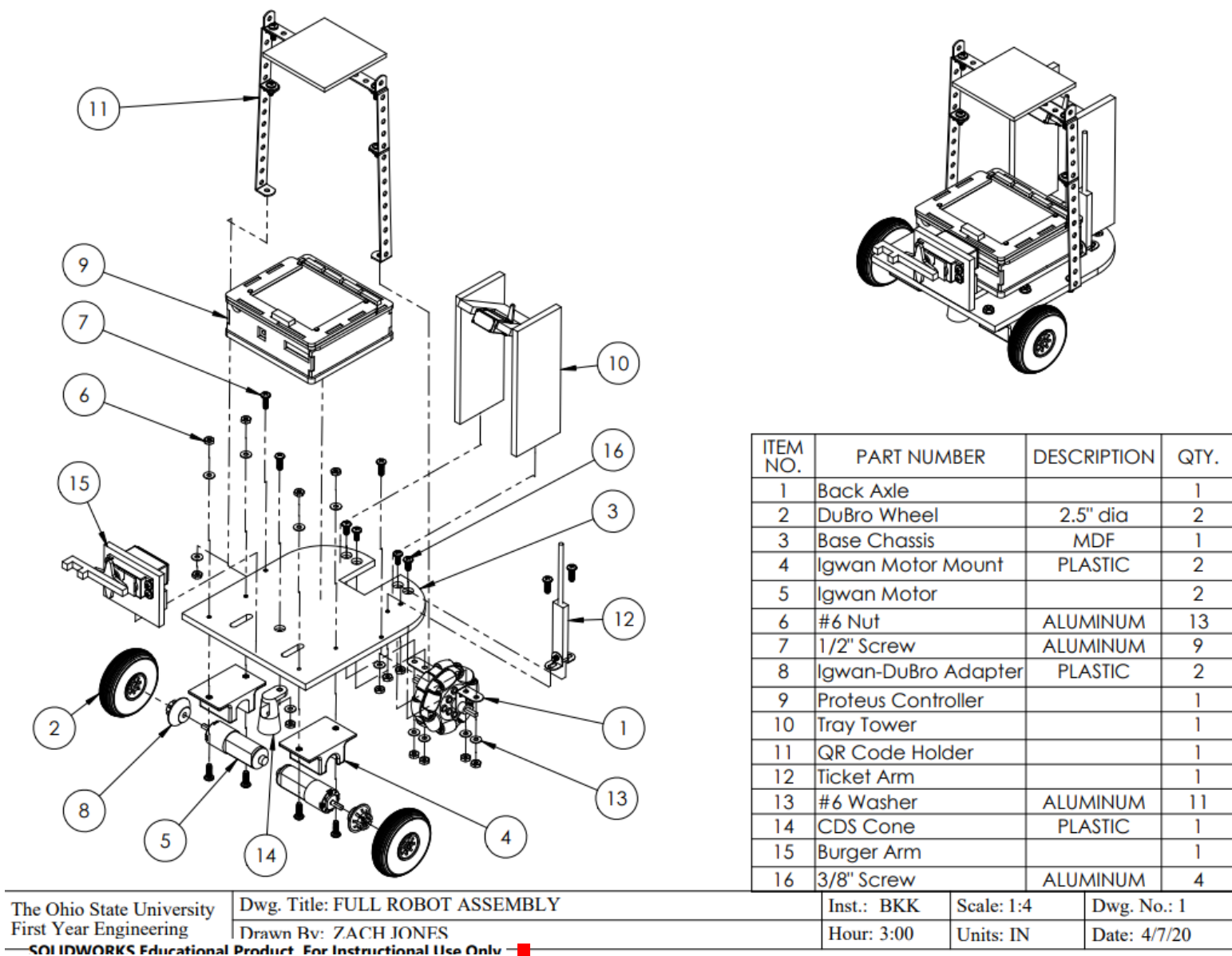
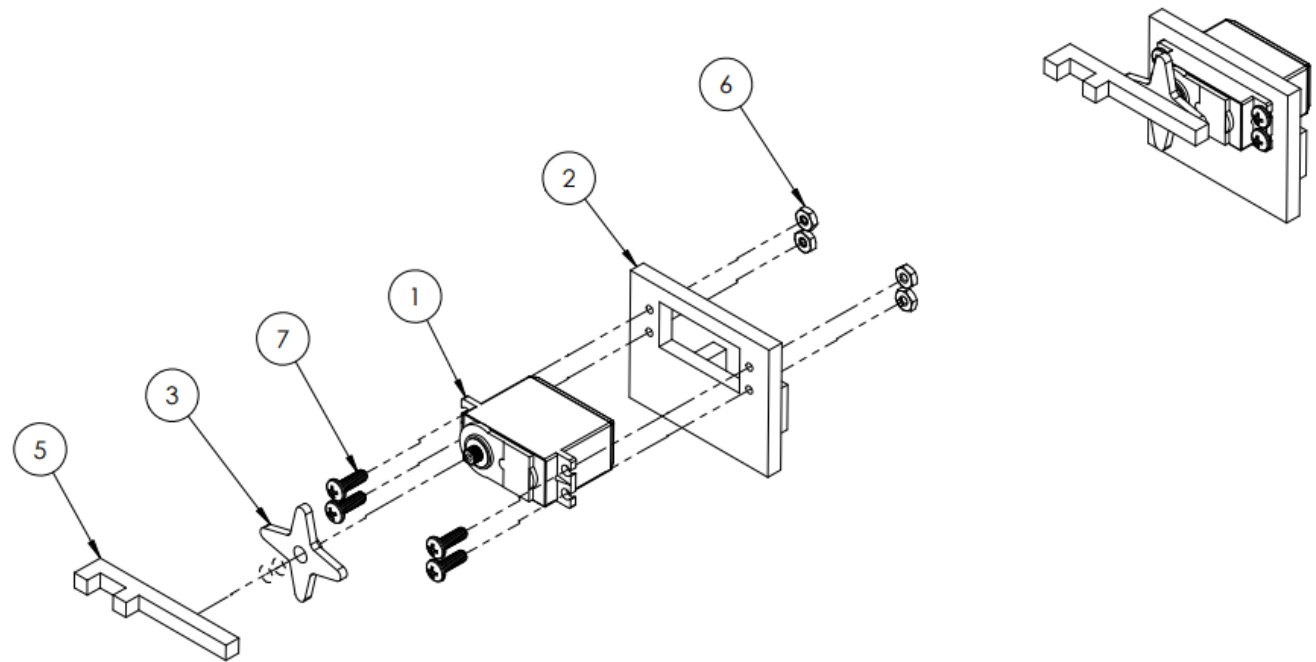


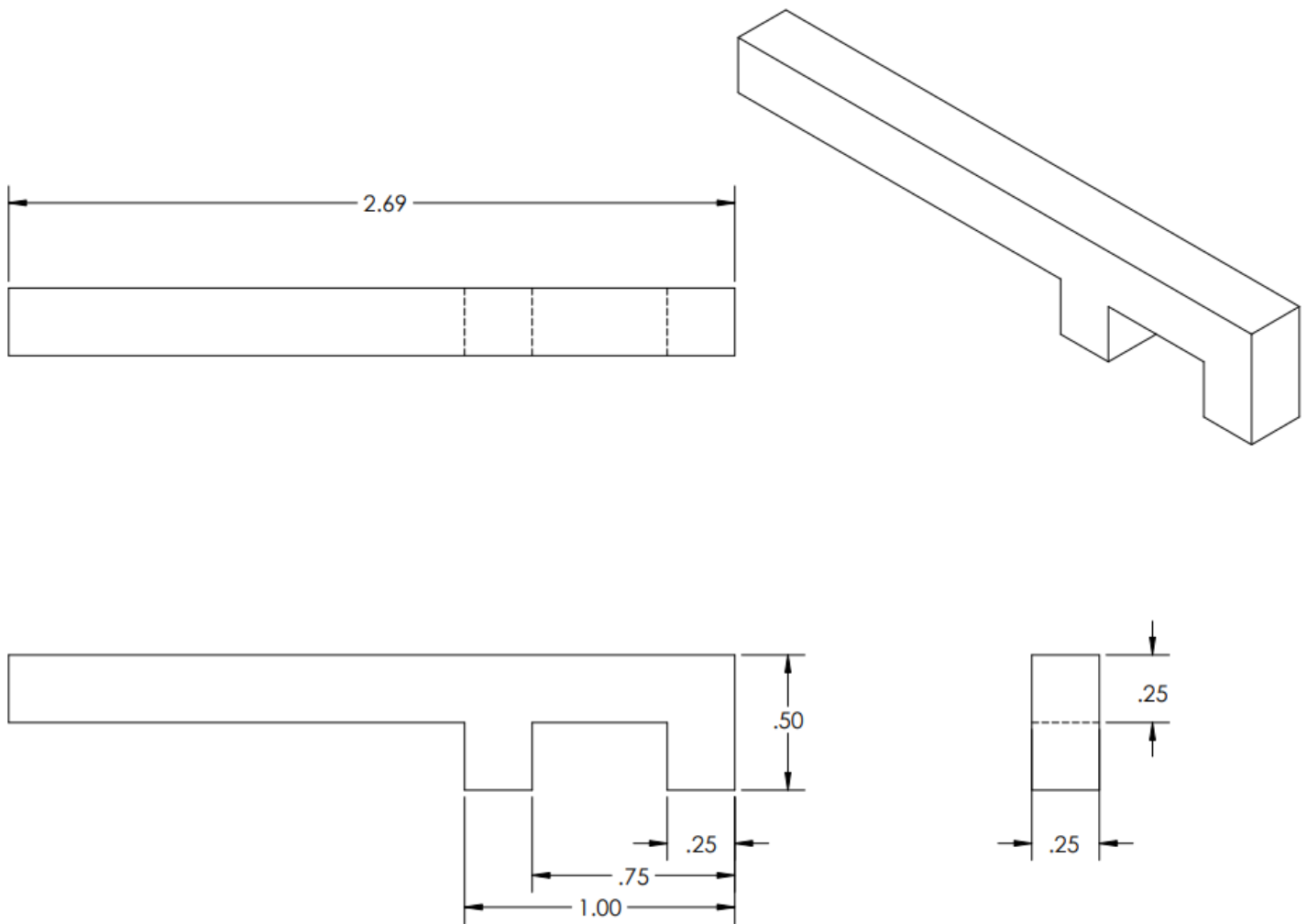
Figure 1D: SolidWorks exploded and condensed view of the current robot.



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Fitec Motor		1
2	Motor Mount	MDF	1
3	Servo Wheel	PLASTIC	1
4	1/2" Screw	ALUMINUM	4
5	Burger Arm	MDF	1
6	#6 Nut	ALUMINUM	4

The Ohio State University First Year Engineering	Dwg. Title: POWERED BURGER ARM	Inst.: BKK	Scale: 1:2	Dwg. No.: 2
	Drawn By: ZACH JONES	Hour: 3:00	Units: IN	Date: 4/7/20

Figure 2D: SolidWorks drawing of the jukebox button/burger crank mechanism.



The Ohio State University
First Year Engineering

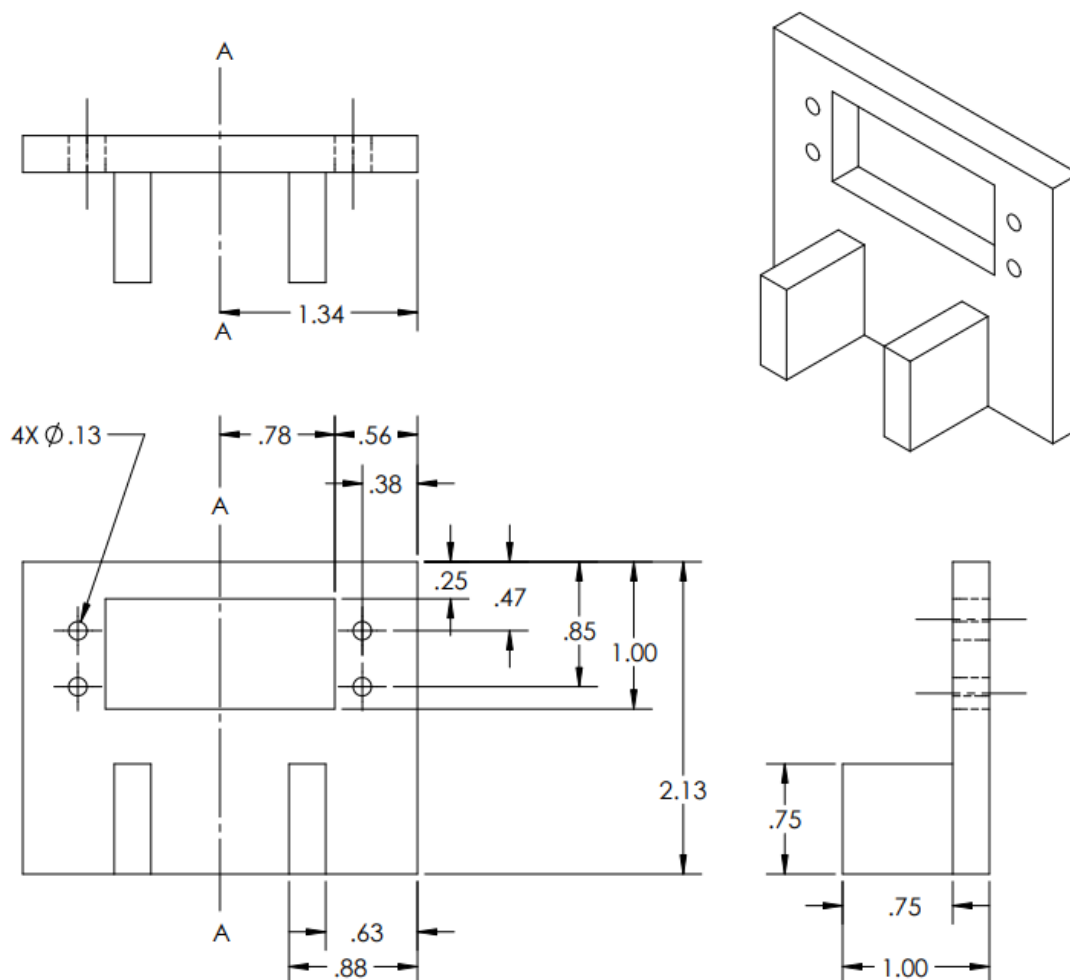
Dwg. Title: BURGER ARM/JUKEBOX ARM
Drawn By: HARRISON HECKER

Inst.: BKK
Hour: 3:00

Scale: 1:0.5
Units: IN

Dwg. No.: 7
Date: 4/5/20

Figure 3D: SolidWorks drawing of the jukebox button/burger crank arm.



Part is symmetric about Plane A-A.

The Ohio State University
First Year Engineering

Dwg. Title: MOTOR MOUNT

Drawn By: HARRISON HECKER

Inst.: BKK

Hour: 3:00

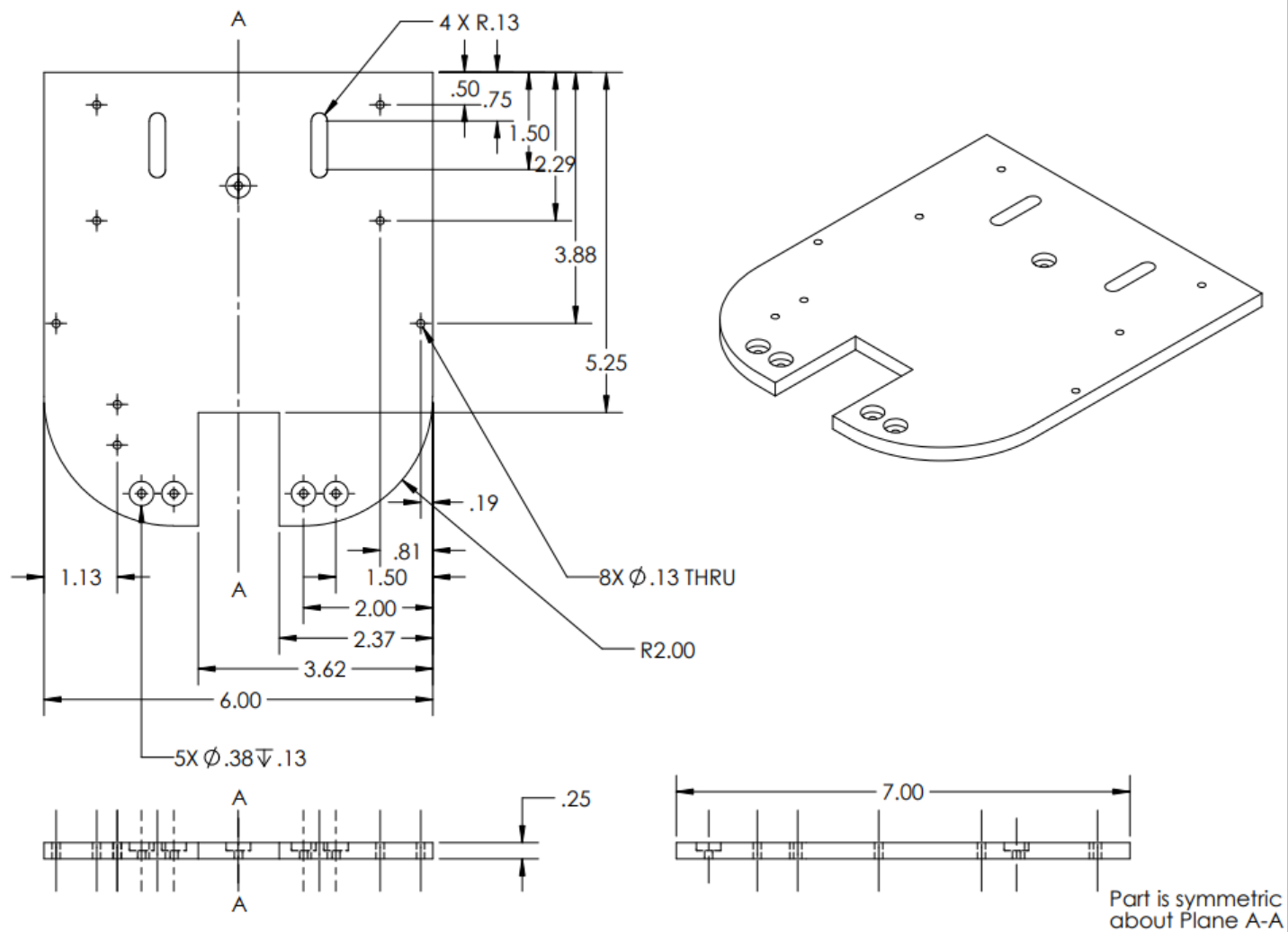
Scale: 1:1

Units: IN

Dwg. No.: 8

Date: 4/5/20

Figure 4D: SolidWorks drawing of the jukebox button/burger crank servo motor holder.



The Ohio State University
First Year Engineering

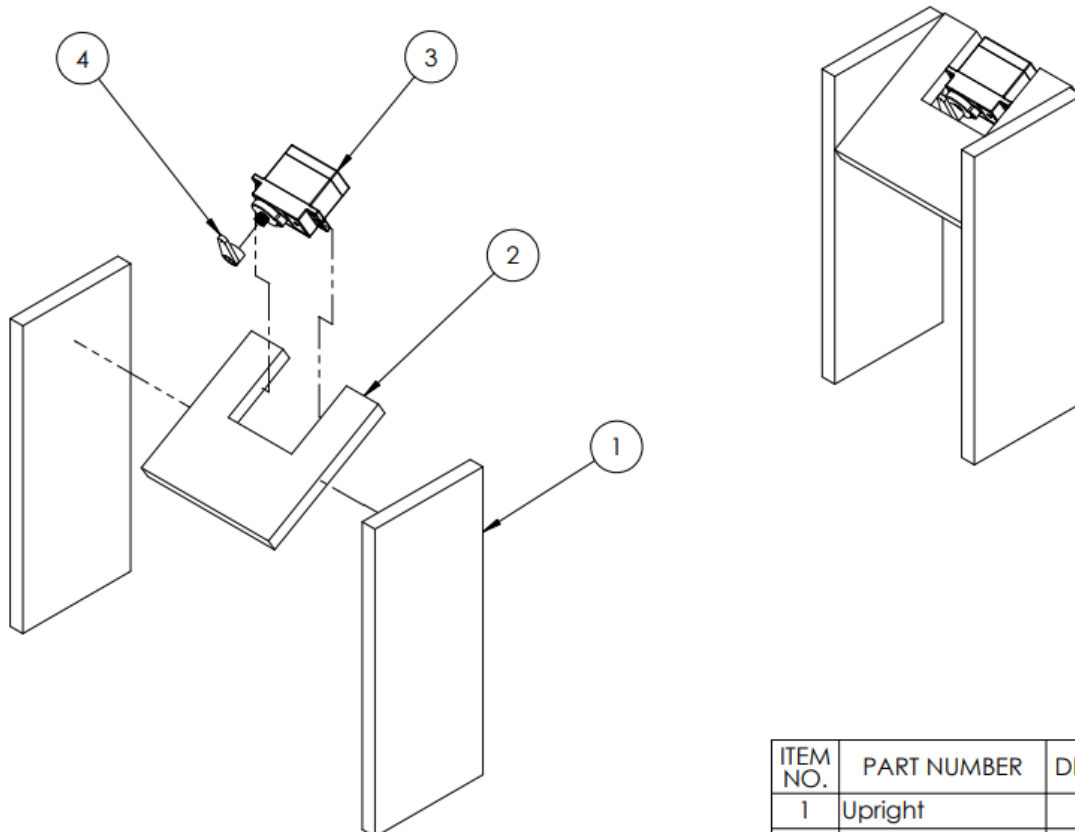
Dwg. Title: CHASSIS
Drawn By: CAM POWERS

Inst.: BKK
Hour: 3:00

Scale: 1:2
Units: IN

Dwg. No.: 12
Date: 4/7/20

Figure 5D: SolidWorks drawing of the chassis.



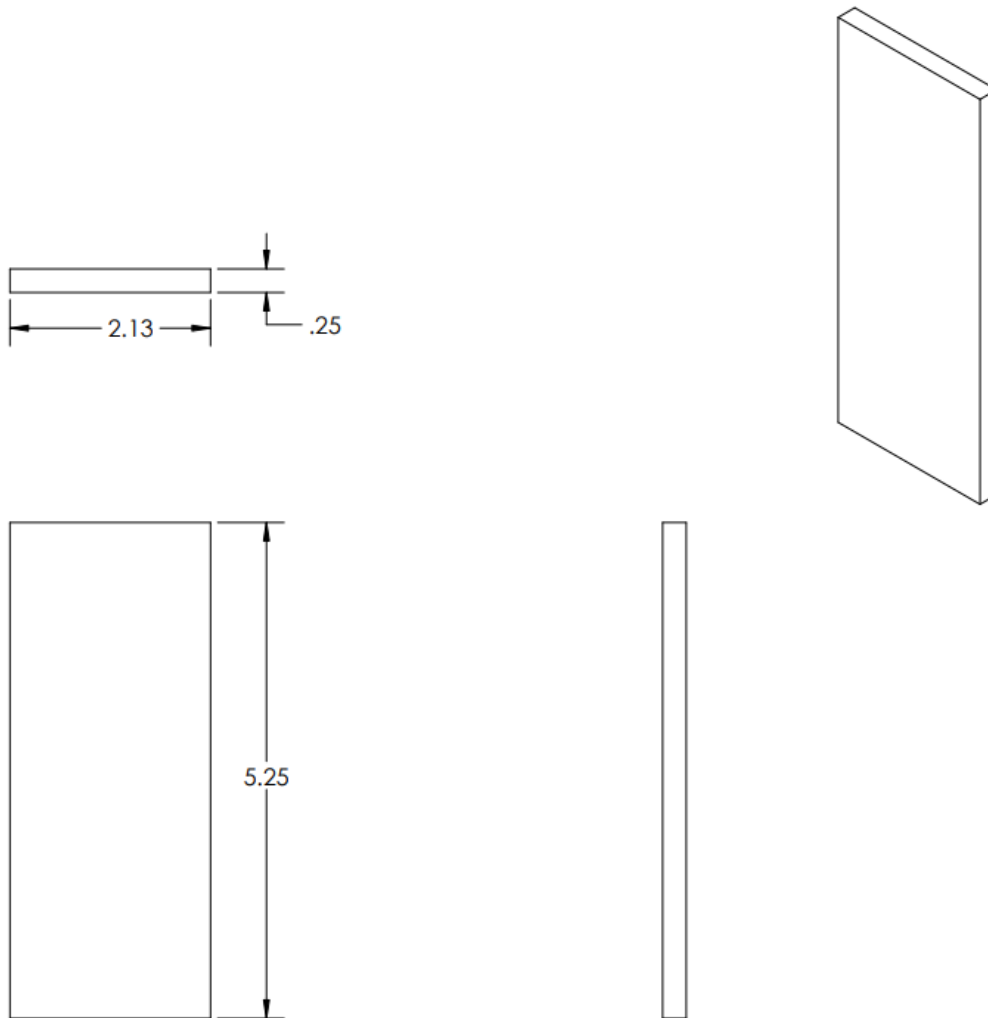
ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Upright	MDF	2
2	Ramp	MDF	1
3	Micro Servo		1
4	Mini Servo Arm	PLASTIC	1

Inst.: BKK	Scale: 1:2	Dwg. No.: 3
Hour: 3:00	Units: IN	Date: 4/8/20

The Ohio State University
First Year Engineering

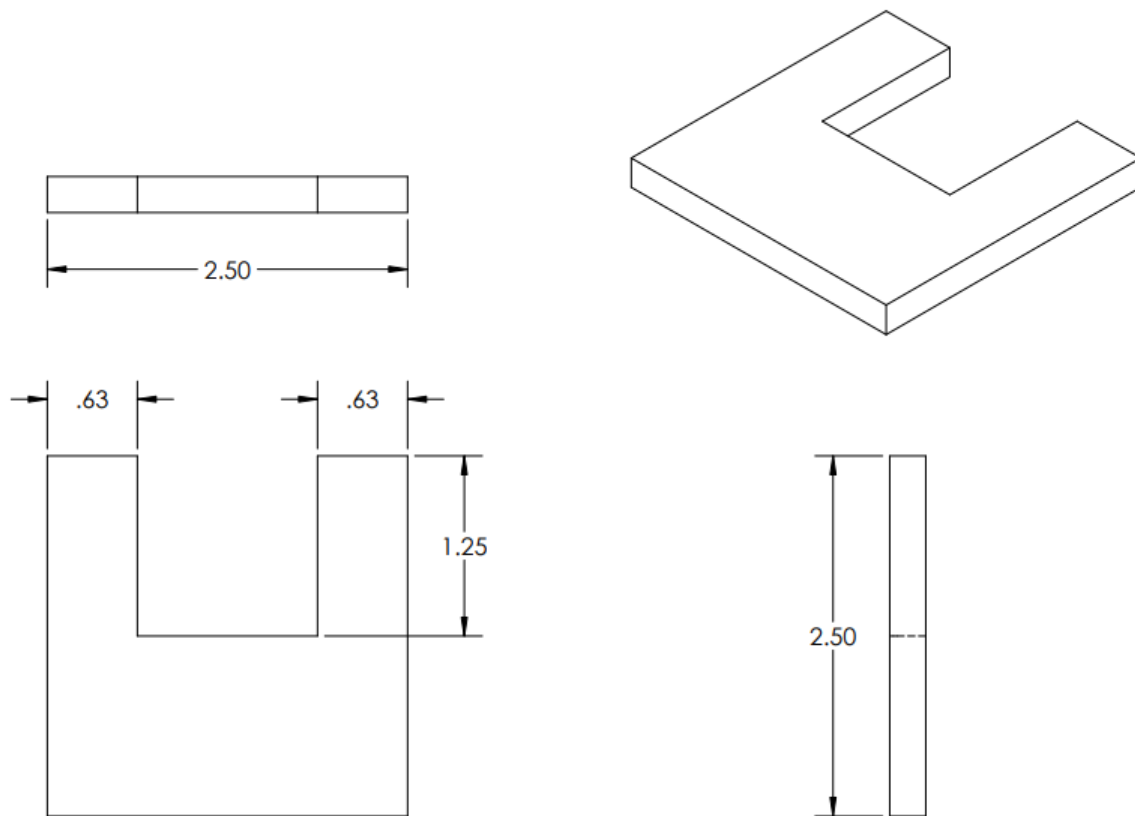
Dwg. Title: TRAY TOWER
Drawn By: ZACH JONES

Figure 6D: SolidWorks drawing of the tray deposit mechanism.



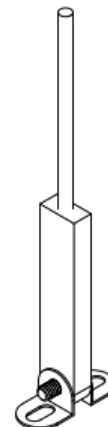
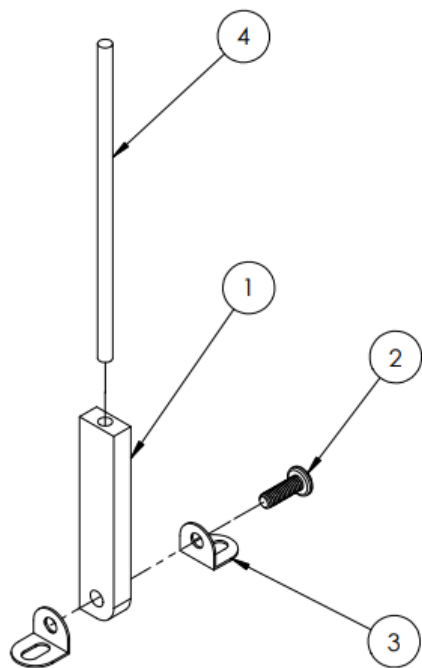
The Ohio State University First Year Engineering	Dwg. Title: UPRIGHT	Inst.: BKK	Scale: 2:3	Dwg. No.: 9
	Drawn By: LIAM EVANS			
		Hour: 3:00	Units: IN	Date: 4/9/2020

Figure 7D: SolidWorks drawing of the walls for the tray holder.



The Ohio State University First Year Engineering	Dwg. Title: RAMP	Inst.: BKK	Scale: 1:1	Dwg. No.: 10
	Drawn By: LIAM EVANS	Hour: 3:00	Units: IN	Date: 4/9/2020

Figure 8D: SolidWorks drawing of the tray holder ramp.



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Ticket Arm Sheath	MDF	1
2	1/2" Screw	ALUMINUM	1
3	Angle Bracket 90deg 1x1 Hole	ALUMINUM	2
4	3.5IN Round Axle	ALUMINUM	1

The Ohio State University
First Year Engineering

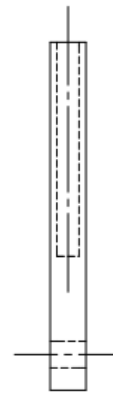
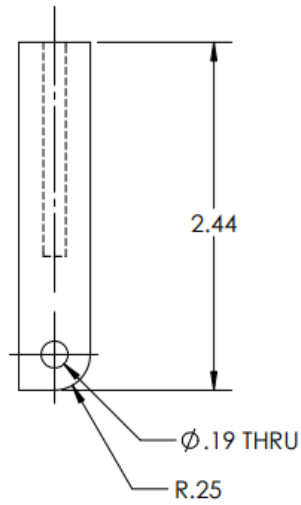
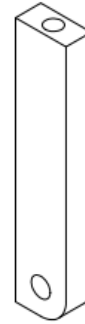
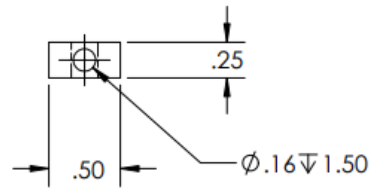
Dwg. Title: TICKET ARM
Drawn By: ZACH JONES

Inst.: BKK
Hour: 3:00

Scale: 2:3
Units: IN

Dwg. No.: 6
Date: 4/8/20

Figure 9D: SolidWorks drawing of the ticket slider mechanism.



The Ohio State University
First Year Engineering

Dwg. Title: TICKET ARM SHEATH
Drawn By: CAM POWERS

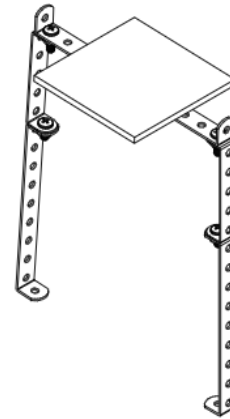
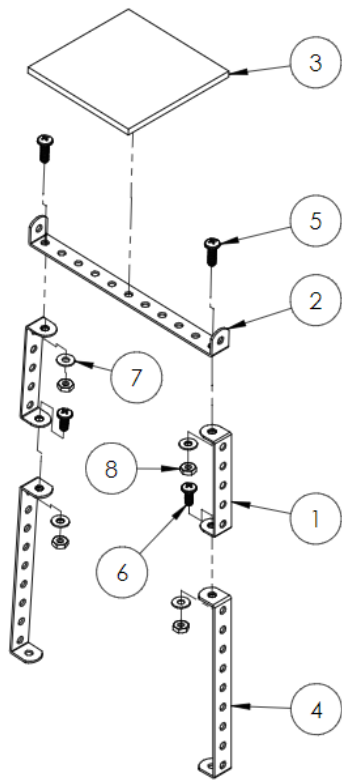
Inst.: BKK
Hour: 3:00

Scale: 1:1
Units: IN

Dwg. No.: 11
Date: 4/8/20

SOLIDWORKS Educational Product For Instructional Use Only

Figure 10D: SolidWorks drawing of the ticket arm sheath.



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Double Angle Strip - 5 Holes	ALUMINUM	2
2	Double Angle Strip - 11 Holes	ALUMINUM	1
3	QR Code Mount	ACRYLLIC	1
4	Double Angle Strip - 9 Holes	ALUMINUM	2
5	1/2" Screw	ALUMINUM	2
6	3/8" Screw	ALUMINUM	2
7	#6 Washer	ALUMINUM	4
8	#6 Nut	ALUMINUM	4

The Ohio State University First Year Engineering SOLIDWORKS Educational Product - For Instructional Use Only	Dwg. Title: QR CODE HOLDER	Inst.: BKK	Scale: 1:3	Dwg. No.: 5
	Drawn By: ZACH JONES	Hour: 3:00	Units: IN	Date: 4/8/20

Figure 11D: SolidWorks drawing of the QR holder.

Appendix E

Code

```

void Drive(int speed, double inches) //using encoders
{
    int counts = inches*Counts_Per_Inch;
    //Reset encoder counts
    right_encoder.ResetCounts();
    left_encoder.ResetCounts();

    if(speed > 0){
        int i = 0;
        while(i < speed){
            right_motor.SetPercent(i);
            left_motor.SetPercent(i);
            Sleep(10);
            i++;
        }
        //While the average of the left and right encoder is less than counts,
        //keep running motors
        while((right_encoder.Counts() + left_encoder.Counts())/2<counts){
            if (right_encoder.Counts()>left_encoder.Counts()+2){
                left_motor.SetPercent(speed+Correction);
                right_motor.SetPercent(speed);
            }
            if (left_encoder.Counts()>right_encoder.Counts()+2){
                right_motor.SetPercent(speed+Correction);
                left_motor.SetPercent(speed);
            }
        }
    }
    //If reverse
    if(speed < 0){
        int i = 0;
        while(i > speed){
            right_motor.SetPercent(i);
            left_motor.SetPercent(i);
            Sleep(10);
            i--;
        }
        //While the average of the left and right encoder is less than counts,
        //keep running motors
        while((right_encoder.Counts() + left_encoder.Counts())/2<counts){
            if (right_encoder.Counts()>left_encoder.Counts()+2){
                left_motor.SetPercent(speed-Correction);
                right_motor.SetPercent(speed);
            }
            if (left_encoder.Counts()>right_encoder.Counts()+2){
                right_motor.SetPercent(speed-Correction);
                left_motor.SetPercent(speed);
            }
        }
    }

    //Turn off motors
    right_motor.Stop();
    left_motor.Stop();
}

```

Figure 1E: Driving for a certain distance method code.

```

void Drive_Time(int speed, double seconds) //using encoders
{
    int start_time = TimeNow();
    right_encoder.ResetCounts();
    left_encoder.ResetCounts();

    if(speed > 0){
        int i = 0;
        while(i < speed){
            right_motor.SetPercent(i);
            left_motor.SetPercent(i);
            Sleep(10);
            i++;
        }
        while(TimeNow()-start_time < seconds){
            if (right_encoder.Counts()>left_encoder.Counts()+2){
                left_motor.SetPercent(speed+Correction);
                right_motor.SetPercent(speed);
            }
            if (left_encoder.Counts()>right_encoder.Counts()+2){
                right_motor.SetPercent(speed+Correction);
                left_motor.SetPercent(speed);
            }
        }
    }

    //If reverse
    if(speed < 0){
        int i = 0;
        while(i > speed){
            right_motor.SetPercent(i);
            left_motor.SetPercent(i);
            Sleep(10);
            i--;
        }
        while(TimeNow()-start_time < seconds){
            if (right_encoder.Counts()>left_encoder.Counts()+2){
                left_motor.SetPercent(speed-Correction);
                right_motor.SetPercent(speed);
            }
            if (left_encoder.Counts()>right_encoder.Counts()+2){
                right_motor.SetPercent(speed-Correction);
                left_motor.SetPercent(speed);
            }
        }
    }

    //Turn off motors
    right_motor.Stop();
    left_motor.Stop();
}

```

Figure 2E: Driving for a certain time method code.


```

void Ramp(int speed){

    //Start Motors
    right_motor.SetPercent(speed);
    left_motor.SetPercent(speed);

    right_encoder.ResetCounts();
    left_encoder.ResetCounts();

    //While the encoder reads less than 12 inches
    while((right_encoder.Counts() + left_encoder.Counts())/2 < 19*Counts_Per_Inch){
        if (right_encoder.Counts()>left_encoder.Counts()+2){
            left_motor.SetPercent(speed+Correction);
            right_motor.SetPercent(speed);
        }
        if (left_encoder.Counts()>right_encoder.Counts()+2){
            right_motor.SetPercent(speed+Correction);
            left_motor.SetPercent(speed);
        }
    }

    //Stop motors
    right_motor.Stop();
    left_motor.Stop();
}

```

Figure 3E: Driving up the ramp method code.

```

void Turn_R(int speed,double angle)//using encoders
{
    //Reset encoder counts
    right_encoder.ResetCounts();
    left_encoder.ResetCounts();

    //Calculate counts for turn
    int counts = ((Dis_Btwn_Wheels*PI*(angle+2)/360.))/(PI*Wheel_Diameter)*Counts_Per_Rotation;
    LCD.WriteLine(counts);

    //Set both motors to desired percent
    right_motor.SetPercent(-speed);
    left_motor.SetPercent(speed*1.06);

    //While the average of the left and right encoder is less than counts,
    //keep running motors

    while((right_encoder.Counts() < counts) || (left_encoder.Counts() < counts));

    //Turn off motors
    right_motor.Stop();
    left_motor.Stop();
}

```

Figure 4E: Right turn method code.

```

void Turn_L(int speed,double angle)//using encoders
{
    //Reset encoder counts
    right_encoder.ResetCounts();
    left_encoder.ResetCounts();

    //Calculate the counts for turn
    int counts = ((Dis_Btwn_Wheels*PI*(angle)/360.))/(PI*Wheel_Diameter)*Counts_Per_Rotation;
    LCD.WriteLine(counts);

    //Set both motors to desired percent
    right_motor.SetPercent(speed);
    left_motor.SetPercent(-speed*1.06);

    //While the average of the left and right encoder is less than counts,
    //keep running motors

    while((right_encoder.Counts() < counts) || (left_encoder.Counts() < counts));

    //Turn off motors
    right_motor.Stop();
    left_motor.Stop();
}

```

Figure 5E: Left turn method code.

```

void One_Wheel_R(int speed, double angle){
    right_encoder.ResetCounts();
    int counts = ((2*Dis_Btwn_Wheels*PI*(angle)/360.))/(PI*Wheel_Diameter)*Counts_Per_Rotation;

    right_motor.SetPercent(speed);

    while(right_encoder.Counts()<counts);

    right_motor.Stop();
}

```

Figure 6E: Right turn with one wheel method code.

```

void One_Wheel_L(int speed, double angle){
    left_encoder.ResetCounts();
    int counts = ((2*Dis_Btwn_Wheels*PI*(angle)/360.))/(PI*Wheel_Diameter)*Counts_Per_Rotation;

    left_motor.SetPercent(speed);

    while(left_encoder.Counts()<counts);

    left_motor.Stop();
}

```

Figure 7E: Left turn with one wheel method code.

```

void check_x_plus(float x_coordinate) //using RPS while robot is in the +y direction
{
    //check if receiving proper RPS coordinates and whether the robot is within an acceptable range
    while(RPS.X() > 0 && (RPS.X() < x_coordinate - 1 || RPS.X() > x_coordinate + 1))
    {
        if(RPS.X() > x_coordinate)
        {
            //pulse the motors for a short duration in the correct direction

            Drive(-power,.2);
        }
        else if(RPS.X() < x_coordinate)
        {
            //pulse the motors for a short duration in the correct direction

            Drive(power,.2);
        }
    }
}

```

Figure 8E: Code for checking “x” coordinate if robot is moving in positive “x” direction.

```

void check_y_plus(float y_coordinate) //using RPS while robot is in the +y direction
{
    //check if receiving proper RPS coordinates and whether the robot is within an acceptable range
    while(RPS.Y() > 0 && (RPS.Y() < y_coordinate - 1 || RPS.Y() > y_coordinate + 1))
    {
        if(RPS.Y() > y_coordinate)
        {
            //pulse the motors for a short duration in the correct direction

            Drive(-power,.2);
        }
        else if(RPS.Y() < y_coordinate)
        {
            //pulse the motors for a short duration in the correct direction

            Drive(power,.2);
        }
    }
}

```

Figure 9E: Code for checking “y” coordinate if robot is moving in positive “y” direction.

```

void check_x_minus(float x_coordinate) //using RPS while robot is in the -x direction
{
    //check if receiving proper RPS coordinates and whether the robot is within an acceptable range
    while(RPS.X() > 0 && (RPS.X() < x_coordinate - 1 || RPS.X() > x_coordinate + 1))
    {
        if(RPS.X() > x_coordinate)
        {
            //pulse the motors for a short duration in the correct direction

            Drive(power,.2);
        }
        else if(RPS.X() < x_coordinate)
        {
            //pulse the motors for a short duration in the correct direction

            Drive(-power,.2);
        }
    }
}

```

Figure 10E: Code for checking “x” coordinate if robot is moving in negative “x” direction.

```

void check_y_minus(float y_coordinate) //using RPS while robot is in the -y direction
{
    //check if receiving proper RPS coordinates and whether the robot is within an acceptable range
    while(RPS.Y() > 0 && (RPS.Y() < y_coordinate - 1 || RPS.Y() > y_coordinate + 1))
    {
        if(RPS.Y() > y_coordinate)
        {
            //pulse the motors for a short duration in the correct direction

            Drive(power,.2);
        }
        else if(RPS.Y() < y_coordinate)
        {
            //pulse the motors for a short duration in the correct direction

            Drive(-power,.2);
        }
    }
}

```

Figure 11E: Code for checking “y” coordinate if robot is moving in negative “y” direction.

```

void check_heading(float heading) //using RPS
{
    //you will need to fill out this one yourself and take into account
    //checking for proper RPS data and the edge conditions
    //(when you want the robot to go to 0 degrees or close to 0 degrees)
    Sleep(.4);
    while (RPS.Heading() > heading + .6 || RPS.Heading() < heading - .6){
        if (RPS.Heading() > heading - .6){
            Turn_R(10,.35);
            //Sleep(.3);
        } else if (RPS.Heading() < heading + .6){
            Turn_L(10,.35);
            //Sleep(.3);
        }
    }

    //Turn off motors
    right_motor.Stop();
    left_motor.Stop();
}

```

Figure 12E: Check heading method code.

```

void Jukebox_Servo()
{
    int color;
    //Red Light
    if(Cds_Cell.Value() > RED_LIGHT_LOWER_BOUND && Cds_Cell.Value() < RED_LIGHT_UPPER_BOUND){
        arm_servo.SetDegree(180);
        LCD.WriteLine("Red");
        color = 0;
    }
    //Blue Light
    if(Cds_Cell.Value() > BLUE_LIGHT_LOWER_BOUND && Cds_Cell.Value() < BLUE_LIGHT_UPPER_BOUND){
        arm_servo.SetDegree(0);
        LCD.WriteLine("Blue");
        color = 1;
    }
    Drive(30,7.4);//Drive 7 3/8 inches
    Sleep(1.0);
    if(color == 0){
        Turn_L(15,6);
    }
    else{
        Turn_R(15,9);
    }
    Drive(-20,7.5);//reverse to where it wa
}

```

Figure 13E: Code for pressing the correct jukebox button.

```

int main(void)
{
    //Initialize the screen
    LCD.Clear(BLACK);
    LCD.SetFontColor(WHITE);

    //Set servo angles
    arm_servo.SetMax(2391);
    arm_servo.SetMin(726);
    servo_arm.setDegree(180);

    LCD.WriteLine("Waiting to Start");
    while(Cds_Cell.Value() > BLUE_LIGHT_UPPER_BOUND); //Wait for the start to light up

    //Drive(50,7.5); //Drive 7.5 inches
    //Turn_L(25,46);
    //Drive(50,10); //Drive 10.5 inches
    //Jukebox_Servo();
    //Turn_L(25,94);
    Jukebox_Servo(); //second call to make sure that it hits the light
    Drive(20,7); //Drive 7 3/8 inches
    Sleep(1.0);
    Drive(-20,7); //reverse to where it was
    //Turn_L(25,94);
    //Drive(50,9); //Drive 10 inches
    //Turn_L(25,94);
    //Drive(50,12); //Drive 12 inches to the ramp
    //Ramp(80);
    //Sleep (1.0);
    //Drive(-50,20); //Reverse to where it was

    return 0;
}

```

Figure 14E: Performance Test #1 main code.


```

int main(void)
{
    //Initialize the screen
    LCD.Clear(BLACK);
    LCD.SetFontColor(WHITE);

    //Set servo angles
    arm_servo.SetMax(2391);
    arm_servo.SetMin(726);

    tray_servo.SetMin(512);
    tray_servo.SetMax(2488);
    tray_servo.SetDegree(120);

    LCD.WriteLine("Waiting to Start");
    while(Cds_Cell.Value() > BLUE_LIGHT_UPPER_BOUND); //Wait for the start to light up

    Drive(20,10.5);
    Turn_L(15,42);
    Drive(20,13);
    //Sleep(.2); // line up for trash can
    Turn_L(15,90.);
    Drive_Time(-20,2.5);
    tray_servo.SetDegree(20);
    Sleep(1.0);
    tray_servo.Off();
    Drive(20,8);
    Turn_L(15,90.);
    Drive(20.5,10.5);
    //Sleep(.2); //line up for ramp
    Turn_R(15,90.);
    Drive(-20,10);
    Ramp(-75);

    /* Ticket Slide */
    Sleep(.5); //pause after the fast ramp portion
    Turn_L(15,90.);
    Drive(20,9.7);
    Turn_R(15,90.);
    Drive(20,5.8);
    Turn_R(15,40);
    One_Wheel_R(-15,55);
    Drive(20,4.5);

    /* To the Burger flip */
    Sleep(.2);
    One_Wheel_L(-15,90);
    Drive(-20,20);

    return 0;
}

```

Figure 15E: Performance Test #2 main code.

```

int main(void)
{
    //Set servo angles
    arm_servo.SetMax(2391);
    arm_servo.SetMin(726);

    RPS.InitializeTouchMenu();
    LCD.Clear(BLACK);
    LCD.WriteLine("Waiting to Start");
    //Write initial screen info
    LCD.WriteRC("X Position:",2,0);
    LCD.WriteRC("Y Position:",3,0);
    LCD.WriteRC("  Heading:",4,0);
    LCD.WriteRC("  Battery:",5,0);

    while(Cds_Cell.Value() > BLUE_LIGHT_UPPER_BOUND){
        LCD.WriteRC(RPS.X(),2,12); //update the x coordinate
        LCD.WriteRC(RPS.Y(),3,12); //update the y coordinate
        LCD.WriteRC(RPS.Heading(),4,12); //update the heading
        LCD.WriteRC(Battery.Voltage(),5,12);
    }//Wait for the start to light up
    Drive(20,11);
    Turn_L(15,137.);
    LCD.WriteLine(RPS.Heading());
    //check_heading(270);
    LCD.WriteLine(RPS.Heading());
    Drive(-20,8);
    Ramp(-75);
    Drive(-20,14);
    Turn_L(15,90.);
    Drive(20,9.5);
    //check_heading(0);
    check_x_plus(26.5);
    Turn_L(15,90);
    //check_heading(90);
    arm_servo.SetDegree(0);
    Drive_Time(15,3.2);
    //check_y_plus(63);
    Sleep(.5);
    arm_servo.SetDegree(120);
    Sleep(1.0);
    arm_servo.SetDegree(0);

    return 0;
}

```

Figure 16E: Performance Test #3 main code.