Project Delivery Report

Team Lead: Christian Millar
Ho Jae Kim
Andrew Rojas
Anthony Baselious
Abdulla Al Kinani

Sponsors:

Mechanical & Aerospace Engineering Department
Henry Samueli School of Engineering & Applied Science
University of California, Los Angeles (UCLA)
Los Angeles, CA. 90095-1597

Figure 1: From left to right: Ho Jae Kim, Andrew Rojas, Abdulla Al Kinani, Christian Millar, Anthony Baselious

Abstract

This project intends to create a completely autonomous robot capable of performing the duties of a fast food worker or waiter with high efficiency and timeliness. The robot is designed to travel to a given location, following a predetermined path to complete activities such as assembling food orders from random picks and delivering them to the relevant table. It incorporates advanced capabilities such as line tracking, obstacle avoidance, and precise item handling, all while adhering to severe design restrictions such as size, autonomy, budget, and power source limitations. Unlike existing systems that necessitate multiple machines and human supervision, this innovative single-device solution provides a low-cost, self-sufficient alternative to streamline food service operations, increasing productivity while reducing the need for extensive human labor and complex machinery configurations.

Table of Contents

List of Figures

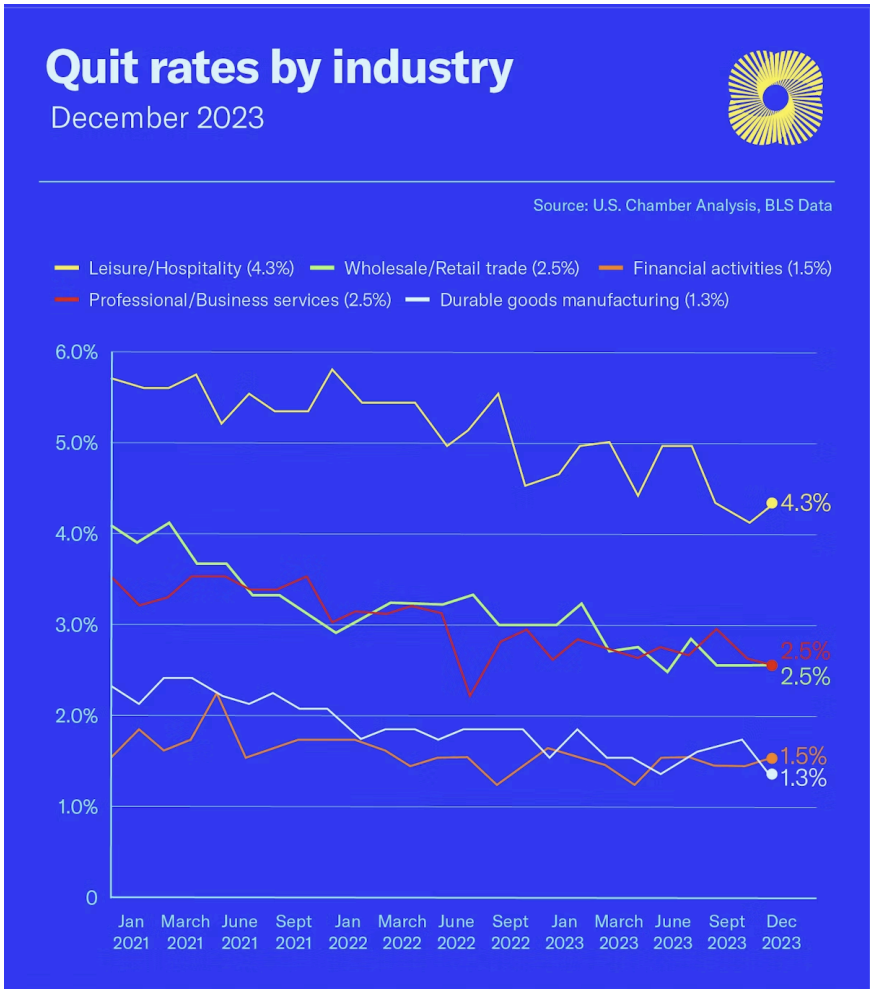
List of Tables

List of Symbols

Symbol	Unit
F_g	N
m_{disk}	kg
g	m/s^2
F_{Max}	N
W	N
$F_{friction}$	N
τ	Nm
r	m
μ	N/A
F_N	N
$F_{required}$	N
F_{servo}	N

1. Introduction

Most people are aware of the growing issues in the world of food service, such as rising prices due to inflation and labor shortages. However, there are also issues that come into play such as lack of quickness in preparing meals, lack of hygiene, and lack of safety in the workplace. The US Chamber of Commerce writes “the leisure and hospitality industry has experienced the highest quit rates of all industries, with the accommodation and food services subsector of this industry experiencing a quit rate consistently above 4.5 percent since July 2021.” Additionally, the CDC recognizes that there are increasing challenges in food safety, due to “Changes in our food production and supply, including more imported foods, Changes in the environment leading to food contamination, and New and emerging bacteria, toxins, and antimicrobial resistance.” Finally, a NPR article from recent weeks was published with the headline “Food prices are going up — and at levels Americans haven't seen in decades”. It is clear from this alone that steps need to be taken to help calm these glaring problems.



- Beef and veal: +16.2%
- Pork: +14%
- Poultry: +12.5%
- Fish and seafood: +10.4%
- Eggs: +11.4%
- Dairy: +5.2%
- Fats and oils: +11.7%
- Fresh fruits: +10.6%
- Fresh vegetables: +4.3%
- Processed fruits and vegetables: +7.6%
- Sugars and sweets: +7%
- Cereals and bakery products: +7.8%

A solution to this problem is obvious with the increasing industry of robotics and autonomous systems. In essence, a food robot. A food robot has the goal of this product is to perform the tasks of an average fast food worker/waiter in an efficient and timely manner. Although no potential customers were interviewed directly, there are a plethora of articles on the internet that claim AI is the future of restaurant operations. For example, Forbes states that "In the kitchen, AI is optimizing operations and food preparation. [For example] Zume Pizza, for instance, uses AI-driven robots to assist in pizza preparation, improving efficiency and

consistency. These robots can spread sauce perfectly and assess cooking times, enhancing the quality of the pizzas." Reducing the cost of hiring a human worker would have the effect of reducing prices for the consumer as well. Additionally, it is more hygienic than a real human, and can perform its duties quite accurately. For this reason, this project will create a fully autonomous robot that must be able to navigate a venue.

For this robot, it is highly encouraged to follow the tape via line tracking. It will start by following a curved path before making the first 90° turn to enter the kitchen. Once there, it must decide whether to assemble a burger or a chicken sandwich (based on the roll of a die). A second die roll will determine which table the order must be dropped off at in the dining room. The product must be able to grab and stack a minimum of two wooden disks from an elevated surface. Upon exiting the kitchen, the product must lower the disks through the doorway. The device will turn again to navigate the second turn, which is angled, and avoid a dynamic obstacle. The vehicle must stop and allow the customer to pass in front of it before continuing. Next, it will go up and down a ramp with a 15° incline and no gradient. This leads into the venue's second 90° turn, and that is followed by the area with the various tables. The product must drop off the disks on an elevated surface at the correct location that was determined by the second die roll. The device must drop off the disks in the correct, upright orientation before going to the end and stopping. The end use of the product is to perform the tasks of a fast food worker more efficiently.

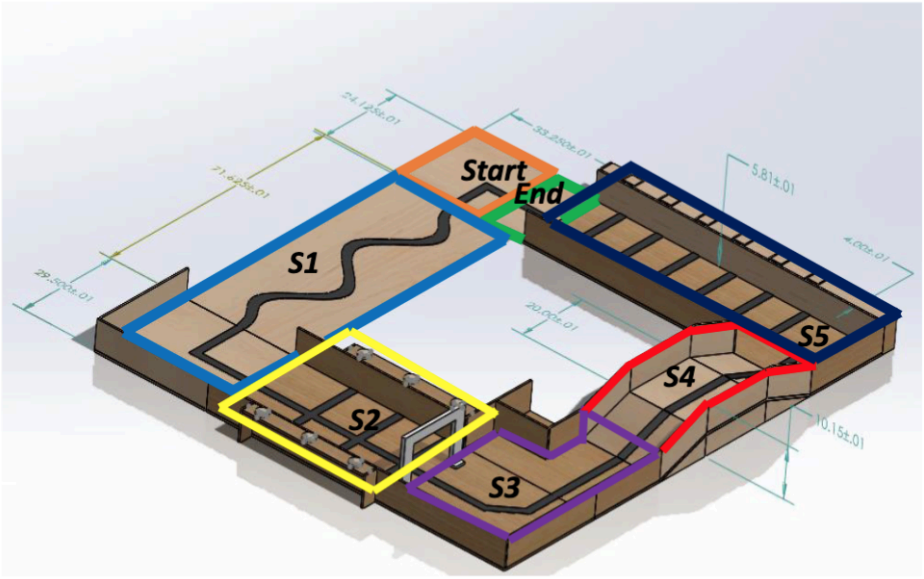


Figure 2: Venue

The client has issued a few design constraints that the product must adhere to in order to qualify for consideration. The device must fit in the venue (i.e. not be too large to fit within the walls). As mentioned, the device must operate autonomously, but remote operation may be considered as a backup. Pre-designed arm kits are not allowed to be purchased. Electric batteries are the only source allowed to power the device. All of the sensors must be mounted on the vehicle. The product cannot leave any parts or components behind. The task set must be completed within 5 minutes. Besides motors, any actuators must be designed and fabricated by the team. Certain off-the-shelf components for the hardware, power transmission, and electronics

are available for purchase. This includes, but is not limited to rods, fasteners, screws, channels, rails, springs, hinges, wheels, motors, gears, gear boxes, rack and pinions, pulleys, belts, strings, flex shafts, steel cables, bearings, switches, wires, cables, bearings, connectors, fuses, relays, motor controls, breadboards, LEDs speakers, etc. The purchasing budget is limited to $500. Microcontrollers, routers, cables, etc. will be provided by the client and do not count against this limit. The client's electric power requires a decision between disposable and rechargeable batteries; a mix between the two is not allowed. If disposable batteries are chosen, then the robot is allowed to have ten 1.5V (AAA, AA, C, or D), and two 9V batteries (12 total). If rechargeable batteries are chosen, then the robot can have three 3.7V, three 9V, and six 1.2V batteries (12 total). The details of the design requirements are subject to change throughout the design process.

There are currently very few products that perform similar functions, however, they are not able to perform every task that this product can. There are multiple robots and conveyor belts working in conjunction with each other to deliver the food. The robots involved each only perform one task. For example, one robot cooks the patties, while the other places the buns, and they work together to make a burger. Afterwards, they place it in a box on a conveyor belt that takes it to another robot that brings the food out on trays for people to grab. These robots also require at least one human operator each, to ensure that they are working as programmed. Since this robot is fully autonomous, it does not require a full-time human operator to monitor the device. This singular product does all of this, thus reducing the cost of having to buy multiple robots, install conveyor belts, and pay for workers to monitor the several robots' operations. In some cases, the robots are unable to cook, which means that human staff is still required to perform these tasks. The robots that can perform multiple tasks such as the Flippy ROAR are expensive, but more importantly, are not mobile like this product.

Figure 3: Similar Products on the Market

It is estimated that the device will stack three disks, go through the door, and avoid the

dynamic obstacle. It is approximated that the product will complete the track within 4 minutes by using line following to navigate the course quicker. The plan is to design the product from scratch, and some parts will be fabricated with a 3D printer.

High-Level and Low-Level Design Requirments

1.1.1. High-Level Design Requirments

Table 1: High-Level Design Requirements*HLDR: High-level Design Requirement

HLDR *	Description
1	Navigate the course from start to finish autonomously
2	Deliver a stacked combination of at least two disks from point A to point B
3	Vehicle begins at Start
4	Send command to rover
5	Press “start”
6	Vehicle passes through S1 (90° turn)
7	Vehicle acquires item (2 disk combo) S2
8	Vehicle must go to corresponding shelf locations and remove disks from shelf
9	Assemble disks into stack
10	At least 2 of the following: Dynamic obstacle, external die roll & coin toss CMD, Navigate over hill, Navigate through gate
11	Vehicle drops off item at pre-selected drop off (table) location S5
12	Vehicle drives to End and stops End
13	Cannot purchase pre-designed arm kits
14	Vehicle must be powered by electric batteries: ten 1.5V (AAA, AA, C, or D), and two 9V batteries or three 3.7V, three 9V, and six 1.2V batteries
15	All sensors must be mounted on the vehicle
16	Vehicle may not leave behind any parts or components
17	Device must complete the task within 5 minutes

18	Allotted $500 budget

1.1.2. Low-Level Design Requirments

Table 2: Low Level Design Requirements

LLDR*	Description
1	Line following navigation
2	Vehicle avoids obstacle S3
3	Stack 3 disks
4	Complete run in 4 minutes
5	Average speed is
6	Shape of device is 2 level box
7	Size of device is 11.5 x 9.2 x 3.6 in
8	Steering type is AWD
9	Delivery mechanism is 4-bar mechanism with claw
10	Rechargeable batteries
11	7 motors
12	Distance between wheels should be 7.5 in

The low-level design requirements were chosen as a team. It was felt that these would make the project manageable while earning the most points possible. The line following navigation is easier to code with Arduino than wall following or other navigation systems. The product is designed to fit through the kitchen door, so this will not be an issue. The code for detecting an obstacle is relatively simple, as the device can back up to give the obstacle more clearance. The claw is designed to stack 3 disks. AWD was chosen because the motors were very cheap, and it provides excellent traction and torque. As a result, turning will be easier and the device will have enough power to go up the ramp. It will also allow more space for sensors and motors as there will be no need for a steering motor or a rack and pinion. The delivery mechanism will feature a four-bar mechanism that has a claw on the end to grab the disks. The idea is that the four-bar mechanism will go to the exact height of the disks to pick them up and drop them off and stack them in the payload bay. This design will allow the product to fit through the kitchen door exit. Rechargeable batteries were chosen because they offer more voltage to power the motors, sensors, and arduino. There are seven motors: 4 DC motors (one for each wheel), 2 servo motors (one for each side of the four-bar mechanism), and 1 servo motor for the claw. The most important low level design requirements are line following, size of device,

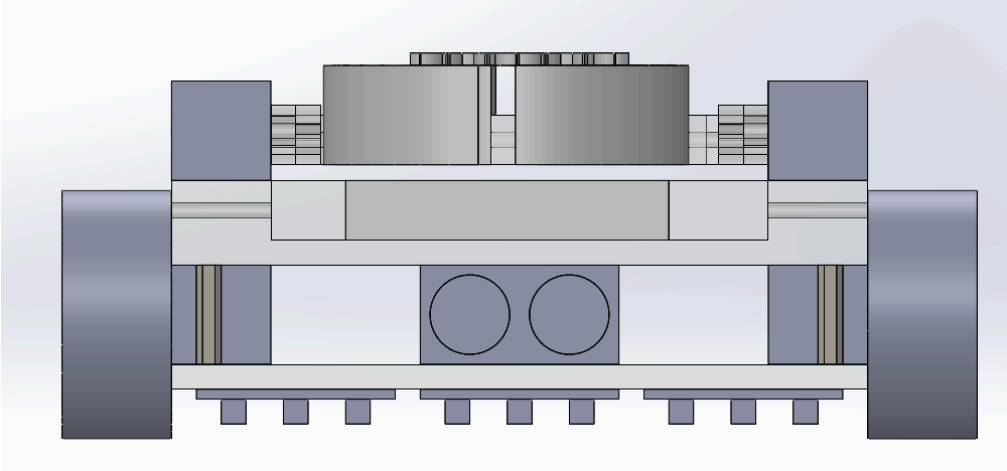
steering type, delivery mechanism, battery type, and number of motors used.

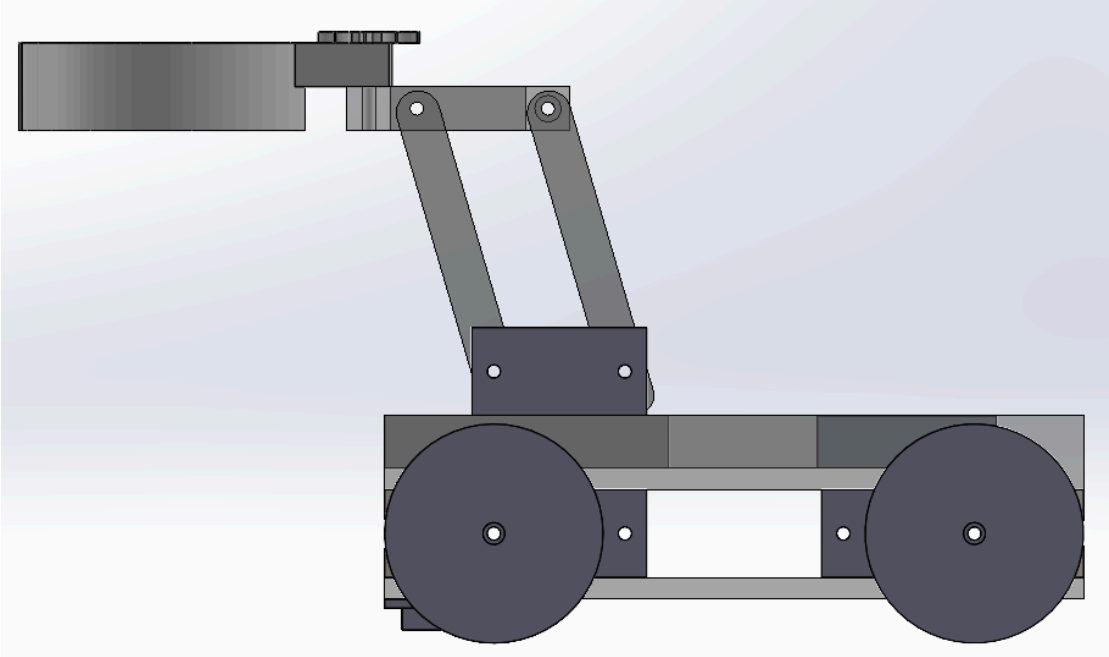
2. Design Description

2.1. Design Concept Development

2.1.1. Design Concept 1

This concept uses a four-bar mechanism. The four-bar linkage will be designed so that when fully extended it is at the right height and depth of the disc. The four-bar mechanism will pick up the disk using a claw which takes the disk into a platform in the chassis. The chassis of the robot consists of two metal plates that are separated vertically by supports. This will be good for weight reduction as well as having space for the electronic components. Having the sides open will allow for better placement of the motors that will hold the wheels, as well as the sensors that will allow the robot to navigate the maze. The benefit of this design is that there are no components outside of the main chassis. This allows the robot to be stable as there are no overhanging parts. This in turn gives the root more stability and maneuverability compared to other designs. Another benefit of the four-bar is that it can use the same mechanism to pick up the disks at the station, stack them within it, and place them at the end goal, making it efficient.





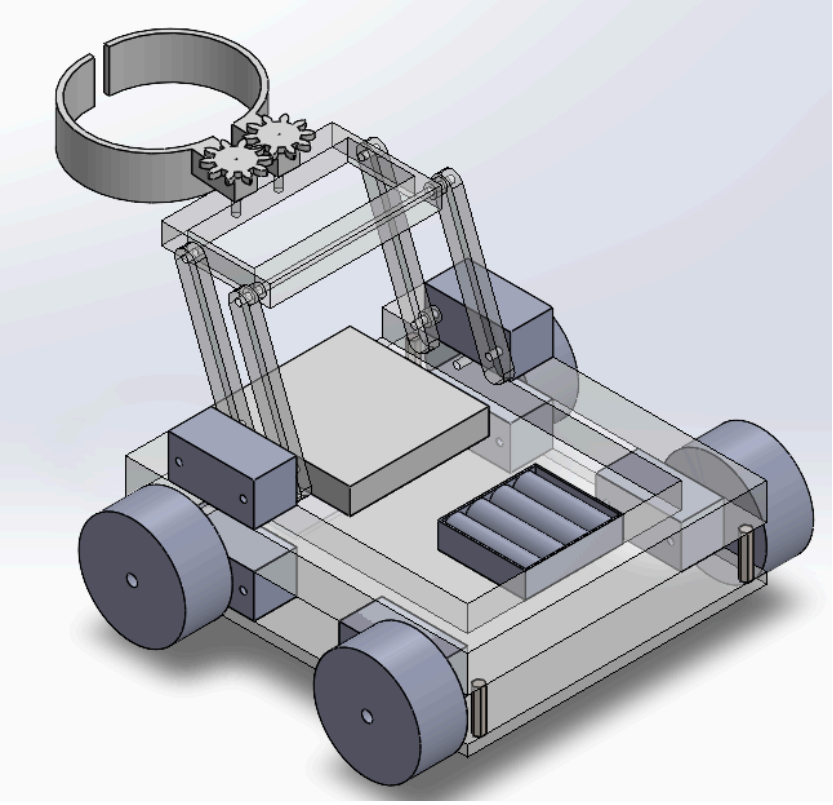


Figure 4 : Different Views of Four-Bar Mechanism Concept Design

2.1.2. Design Concept 2

Concept 2 utilizes an arm that will pick up and drop the disks.The drive system of the robot is also made up of tank treads, making the overall design resemble an excavator. The benefit of this design is that the arm will have multiple joints, allowing it to have a greater degree of freedom and being more maneuverable. This allows it to grab and place the disks with greater flexibility. The base would also have multiple wheels, as well as tank treads. This will give the robot a lot of stability as well as more power as more motors would have to be used.

However, this design also has some flaws, like being highly unstable and slow. The arm would require a lot of motors to be able to move at different degrees. This woul

d make it heavy and imbalanced. The base while being stable would also be slow and maneuverable as it uses a tank driving system.

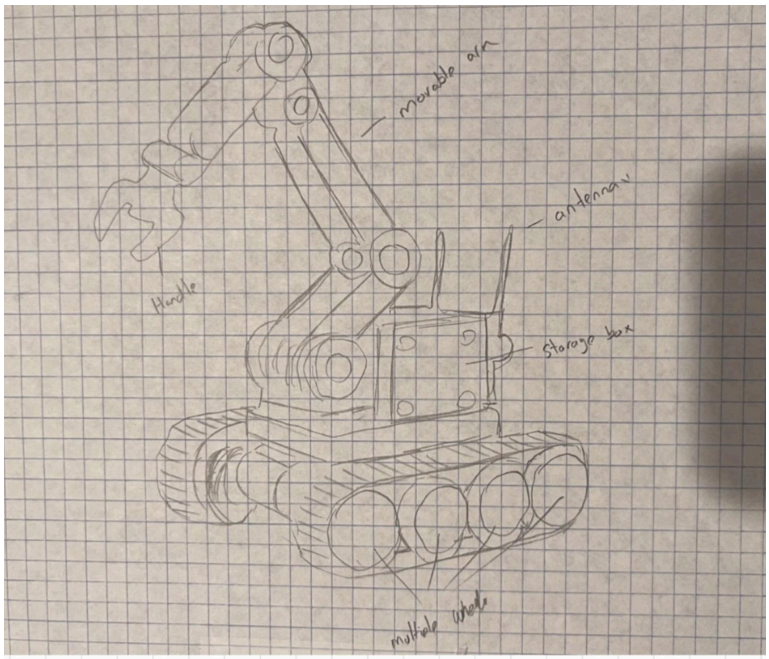


Figure 5 : Tank with Arm Concept [3].

2.1.3. Design Concept 3

Concept 3 features a box chassis with a lift and grabber in the front. The lift is designed to go up and down, so that the device can grab the disks from the elevated surface and lower them to fit through the door. The grabber is meant to hold the disks in place, in order to prevent the stacked assembly from coming undone at the dropoff location. The design also features larger and thicker rear wheels to increase the overall traction of the vehicle. It will be able to handle the 90° turns and ramp better.

There will be one sensor for the line tracking on the front face of the vehicle. There will be four motors total: one for the lift, one each for the rear wheels, and a steering motor. The device is RWD, so it will have more power to go up the ramp and have more maneuverability.

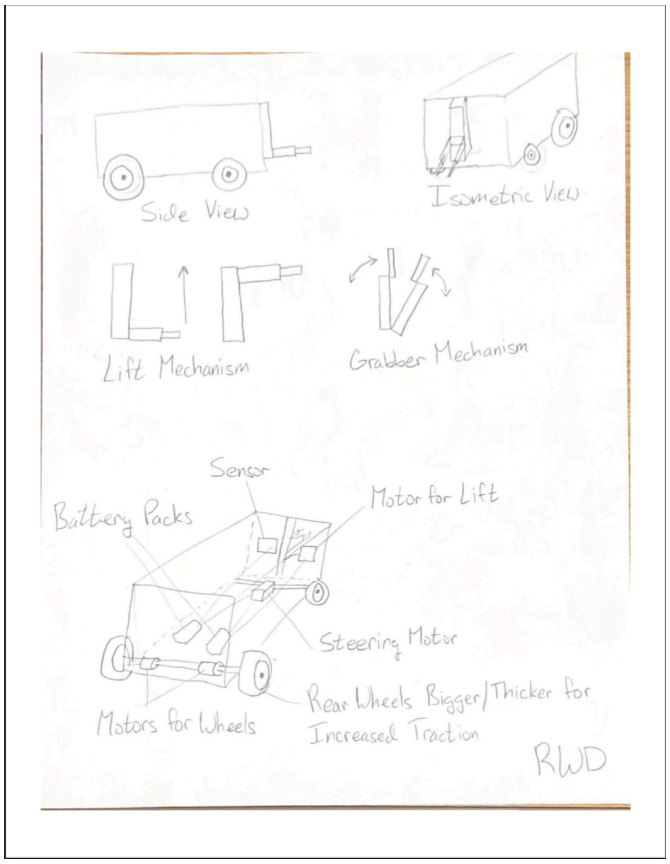


Figure 6: Forklift Concept [3]

2.1.4. Objectives Tree and Pairwise Comparison Chart

Table 1: PCC and Objective Tree

OBJECTIVES TREE							
		Design Scores			Weighted Design Scores		
Design Objective Factors	PCC Weight	Design 1	Design 2	Design 3	Design 1	Design 2	Design 3
Cost	0.1	6	2	7	0.6	0.2	0.7
Weight	0.15	6	1	7	0.9	0.15	1.05
Safety	0.1	8	8	5	0.8	0.8	0.5
Durability	0.2	8	8	8	1.6	1.6	1.6
Aesthetics	0	0	0	0	0	0	0
Complexity	0.05	6	2	6	0.3	0.1	0.3

Power	0.2	7	7	8	1.4	1.4	1.6
Maneuverability	0.2	8	5	7	1.6	1	1.4
	1			Total Score:	7.2	5.25	7.15

1. Design Concept Selection

The design that was chosen was the design that uses the four-bar mechanism. While the score was extremely close to the fork lift method design, the four bar concept was better in safety and maneuverability, giving it the edge. The benefit of this design is that there are no components outside of the main chassis body. It is a boxy shape, and it allows the robot to be stable as there are no overhanging parts. This is because the center of mass is closer to the middle of the chassis. This in turn gives the root more stability and maneuverability as compared to other designs. Another benefit of the four-bar mechanism is that it can use the same mechanism to pick up the disks at the station, stack them within it, and place them at the end goal. This is because it uses the same action, but in forward and in reverse, making it highly efficient and easier to assemble, code, and ultimately complete the course. The fasteners are not included in the CAD model. In the Bill of Materials section, many of the components come with their own set of fasteners. For this reason, we have included the holes where we will fasten our components, but did not deem it necessary to show all the individual fasteners.

2.2. Design Overview

This system consists of many subsystems. Starting at the bottom we have an acrylic plate. This acrylic plate hold the electrical components such as the breadboard and batteries. These batteries power many of the subsystems of the vehicle. The 4 motors are then also attached to the side of the acrylic plate. These motors are attached to their corresponding wheels. These motors power the system and allow the vehicle to move.

Also attached to the plate are the IR sensors. These sensors are then attached with using 3D printed spacers to the bottom of the plate. This is meticulously designed so it can properly detect if the vehicle is following the path of the black tape or not. This bottom acrylic plate is then attached to the top plat using long spacers. Attached on this top plate is the 4 bar mechanism. This mechanism is fixed to the plate using 3D print elements. This allows the 4-bar mechanism to rotate while the bottom of the bar is fixed to the plate. Each side of the bar is powered by a servo. At the end of the 4 bar mechanism the 3D print claw is attached. This is what will pick up and grab the disks. This claw is powered by a servo.

2.3. System Specifications

2.4. Mechanical Systems

The four-bar linkage consists of four rigid bars connected by pivots on the servos to form a closed chain, creating a system that can transfer motion and force in a controlled manner. The primary components include a fixed frame, a crank (input link), a coupler (connecting link), and

a follower (output link) which is attached to the claw. As the crank rotates, it imparts a corresponding motion to the coupler and follower, enabling the claw to open and close. This configuration allows for precise and repeatable movements, making it ideal for tasks requiring dexterity and accuracy, such as picking up and manipulating disks. The efficiency of the four-bar linkage in converting rotational motion into linear or oscillatory motion is leveraged to ensure the claw's movements are smooth and synchronized, providing reliable performance in various industrial and robotic applications.

2.5. Control Systems

The control systems of this robot, powered by an Arduino Mega, are crucial for its precise and autonomous operation. The robot utilizes three infrared (IR) sensors for line tracking, enabling it to follow predefined paths with accuracy. These sensors detect the contrast between the line and the surface, sending signals to the Arduino Mega to adjust the robot's trajectory accordingly. Additionally, another IR sensor is employed for wall detection, allowing the robot to navigate around obstacles and prevent collisions by signaling the Arduino to alter its course when a wall is detected. The system is further enhanced by two input buttons, which provide manual control options, allowing users to initiate specific pick up and dropoff locations. The Arduino Mega, acting as the central processing unit, integrates inputs from the IR sensors and buttons, processes the data, and controls the actuators driving the robot's movement and the claw mechanism, ensuring seamless and responsive operation.

3. Subsystem Design Description

3.1. Structural

3.1.1. Subsystem Description

For this system the structural components play a crucial role in ensuring task completion. As seen in the CAD model the system is designed with an acrylic plate on the bottom to hold the electronic components on it and to attach to the motor system. The plate is then attached using spacers to a second acrylic plate on top of it which hosts the retrieval mechanism. The bottom of the retrieval mechanism, in this case, a 4-bar mechanism, is then fixed to this plate to allow rotation of the 4-bar mechanism along this point.

3.1.2. Design requirements

Many factors must be considered when designing the structural subsystem which in our case is the chassis of the vehicle. The placement of the motors, ground clearance for the ramp, and sensor placement must be taken into account when design the bottom plate. There also must be consideration to make space for the wiring and breadboard to ensure proper integration. For the second plate, the distance between plates needs to be calculated to ensure that the retrieval mechanism will be placed at a proper height. There also should be enough room for the 4-bar mechanism and the servos along with the claw and disks.

3.1.3. Subsystem CAD Models and Engineering Drawings

3.2. Drive System

3.2.1. Subsystem Description

For this system, it was decided that the drive system would be powered using an all-wheel drive system. All-wheel drive was chosen to help the vehicle go up the hill on the course. The motors will control the movement of the vehicle and allow it to run the course, go to the disk location, go up a hill, and drop off location.

3.2.2. Design requirements

The drive system must be able to power the vehicle to run the course, go to the disk location, go up a hill, and drop off location. The power of motors and speed must be taken into account. All wheel drive was chosen to help the vehicle get up the hill and complete its objective.

3.2.3. Subsystem CAD Models and Engineering Drawings

3.3. Retrieval/ Delivery Mechanism

3.3.1. Subsystem Description

The goal of the retrieval system is to retrieve one circular disk, stack it on top of another disk, carry it to a delivery location and successfully deliver the stack at the delivery location. To achieve this purpose this vehicle will use a 4-bar mechanism connected to a claw. The four-bar mechanism will rotate up with the claw attached to the front to allow the claw to be around the disk. The claw will then grab the disk and the 4-bar mechanism will then bring the claw and the disk back down to the vehicle. It will then repeat this process with the second disk stacking the first disk on the second disk when retrieving it. Then the mechanism will deliver both disks stacked at the delivery station.

3.3.2. Design requirements

To successfully complete this task, there are several factors that must be considered. First is how the circular disk will be grabbed. Next, there needs to be consideration of how the first disk will be stacked on top of the second disk. Then the mechanism must be able to deliver both disks stacked on top of each other at the delivery station. The four-bar mechanism will allow the claw to get from the vehicle to the pickup station for the disks and the claw will pick up the disks. Then the four-bar mechanism will bring the disk back to the vehicle. Afterward, the four-bar mechanism will then put the claw at the delivery station where the claw will release the stacked disks.

3.3.3. Subsystem CAD Models and Engineering Drawings

3.4. Sensors

3.4.1. Subsystem Description

Sensors are used to feed information into the system to allow the system to decide how to operate and make the necessary changes to make the vehicle run the course. There are IR sensors that allow the vehicle to detect if it detects white or black under it. This allows the vehicle to properly line follow along the path. It also allows it to detect intersections which allows the vehicle to stop at the right drop off and pick up locations.

3.4.2. Design requirements

The sensors must be properly chosen so they provide useful and accurate

information regarding the current location of the vehicle on the track. It needs to be able to sense whether the vehicle is currently on the black line and be able to provide accurate information so they vehicle can adjust and properly follow the black tape. There is also a different sensor that must provide information on how close an obstacle is in front of the vehicle. This can help it to detect how close it is from walls and to not crash into obstacles in its path.

3.4.3. Subsystem CAD Models and Engineering Drawings

3.5. Electronics

3.5.1. Subsystem Description

Power System

- **Batteries:**
    - We used an Elegoo lithium-ion battery to power the Arduino and the four motors.
    - Additionally, we used four 1.5V AA batteries (providing a total of 6V) to power the three servos.

Sensor System

- Infrared (IR) Sensors:
    - We had three standard IR sensors for detecting objects or obstacles.
    - We also had one RML IR sensor, which consists of three modules for enhanced detection capabilities.

Control System

- Arduino:
    - Our Arduino was connected to the lithium-ion battery, which powered the four motors.
    - The Arduino provided a 5V power supply for the four IR sensors and other low-power components.

Actuation System

- Servos:
    - We used three servos connected to the 6V battery supply (four 1.5V AA batteries).
- Motors:
    - We had four motors powered by the lithium-ion battery connected to the Arduino.

Additional Components

- Pull-up Resistor and Button:

- We connected a pull-up resistor and a button to the 5V supply from the Arduino for input control and to ensure stable readings.

3.5.2. Design requirements

Power System

- We required the Elegoo lithium-ion battery to have enough capacity to power the Arduino and four motors for our operation.
- We needed the 4 x 1.5V AA batteries to provide a stable 6V supply to power the three servos reliably.

Sensor System

- We strategically placed the three standard IR sensors and the RML IR sensor to ensure comprehensive coverage and accurate obstacle detection.
- We calibrated the sensors to avoid false positives and ensure reliable detection.

Control System

- We programmed the Arduino to handle inputs from the IR sensors and provide outputs to the servos and motors.
- We managed power efficiently to ensure the stable operation of the entire system.

Actuation System

- We ensured the servos operated within their specified voltage range (6V) and handled the required mechanical loads smoothly and precisely.
- We used motor driver circuits to control the motors efficiently, ensuring they could handle the power supplied by the lithium-ion battery.

Additional Components

- We correctly rated the pull-up resistor to ensure the button operated reliably.
- We used a responsive and durable button for repeated use.

Safety and Reliability

- We securely connected all components to prevent short circuits or loose connections.
- We provided adequate insulation and protection for all wiring to avoid electrical hazards.
- We tested the system to ensure consistent performance under different conditions and included redundant systems or fail-safes where necessary to handle potential component failures.

Subsystem Models and Engineering Drawings

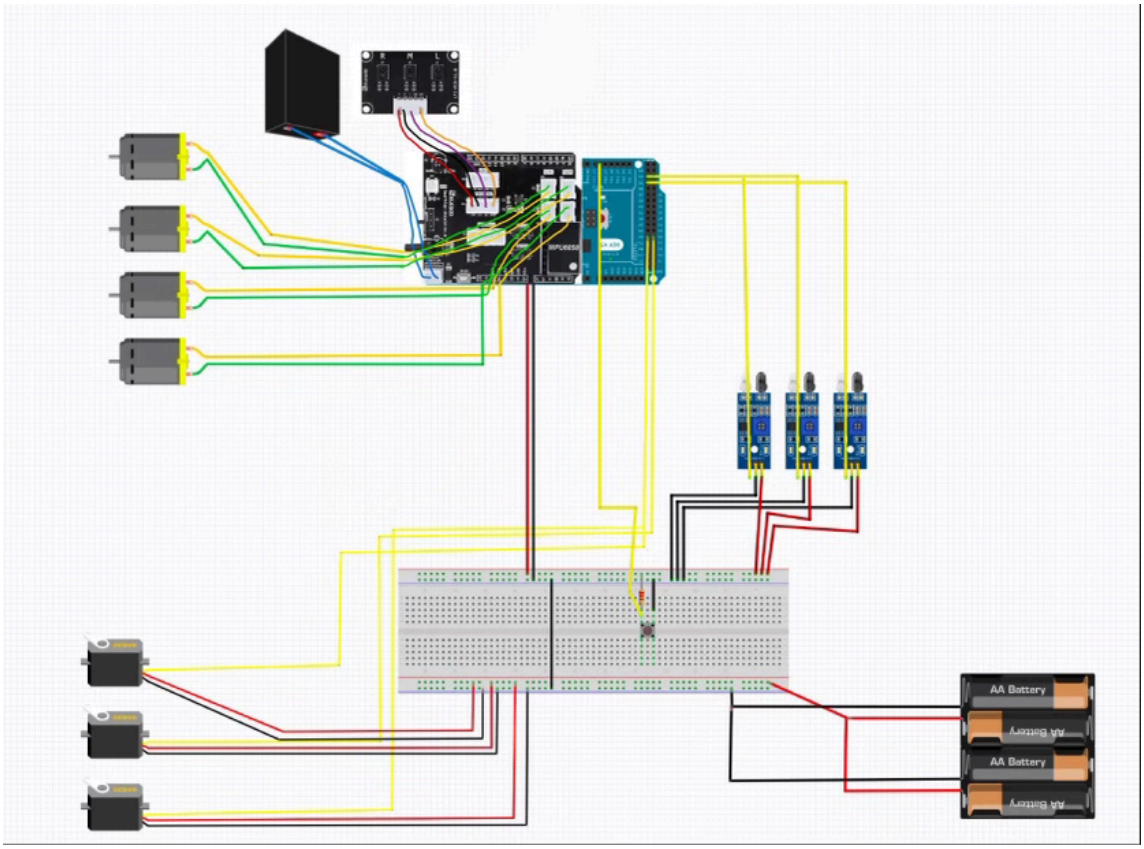


Figure 7: Wiring diagram

To wire everything, we first connected the Elegoo lithium-ion battery to the Arduino to power it and the four motors. We then used four 1.5V AA batteries in series to create a 6V power supply for the three servos. The three standard IR sensors and the RML IR sensor were connected to the Arduino's 5V power supply, ensuring they received adequate power for operation. Each sensor's signal pin was connected to the appropriate digital input pins on the Arduino, allowing for precise detection and control. The three servos were wired to the 6V battery pack, with their signal wires connected to the designated PWM output pins on the Arduino. The four motors were also powered by the lithium-ion battery, with control wires running to motor driver circuits connected to the Arduino for efficient operation. Additionally, we included a pull-up resistor and a button, which were connected to the 5V supply from the Arduino, ensuring stable input readings. All connections were securely soldered or attached using breadboards and jumper wires, with careful attention to insulation and proper routing to prevent short circuits and ensure reliable performance.

4. Design Analysis

4.1. Analysis and Calculations

4.1.1. Drive System Power Requirements

If an **arbitrary location** is selected for the sake of this report (say kitchen location #1, and Table #1), the path segment diagram is as follows. The purpose of Segment 4 and Segment 8 being the kitchen and table activities is to allow the distance traveled to be the same regardless of location. For example, if location #2 or #3 were to be visited instead of location #1, where the robot turns and performs the action would be different, but the total distance traveled from the start of the segment to the end of the segment would **be the same.** This is similar to the segmented path image shown above, but with more details required for calculation.

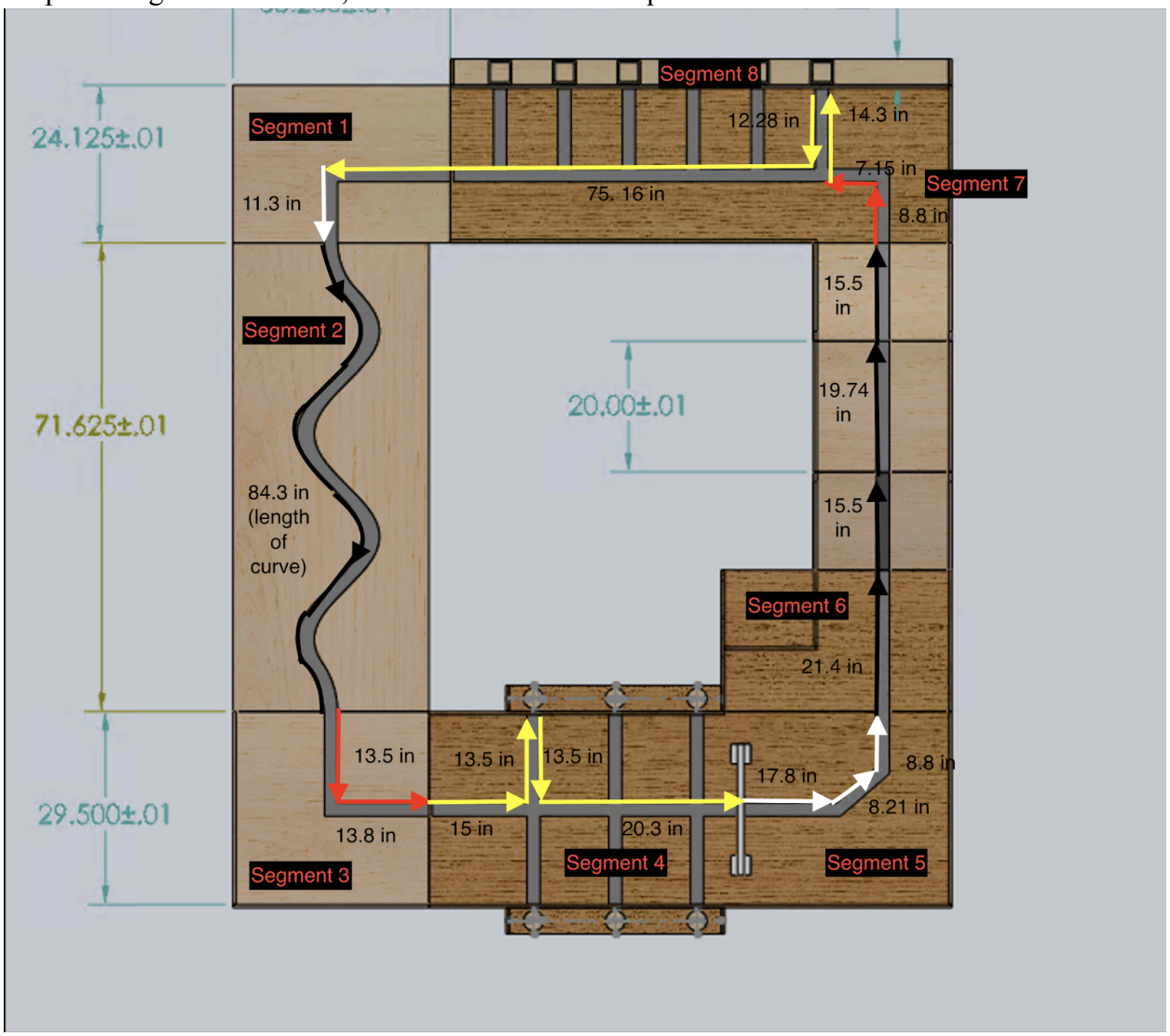


Figure 8 : Venue Divided Into Segments

Table 3: Estimated length and total time of travel for each path segment

Table 1: Estimated length and total time of travel for each path segment		
Path Segment	Segment Length (in)	Time Along Segment (s)
1	11.3	4.52
2	84.3	33.72

3	27.3	10.92
4	62.3	24.92
5	34.81	13.924
6	72.14	28.856
7	15.95	6.38
8	101.74	40.696
TOTAL TIME (seconds)		163.936

Table 4 : Estimated time-line for each path segment

Table 2: Estimated Timeline for each path segment					
Path Segment	Move Profile Type	Accel Time (s)	Const Vel Time (s)	Decel Time (s)	Total Time (s)
1	Trapezoidal	1.506666667	1.506666667	1.506666667	4.52
2	Trapezoidal	11.24	11.24	11.24	33.72
3	Trapezoidal	3.64	3.64	3.64	10.92
4	Triangular	12.46	0	12.46	24.92
5	Trapezoidal	4.641333333	4.641333333	4.641333333	13.924
6	Triangular	14.428	0	14.428	28.856
7	Trapezoidal	2.126666667	2.126666667	2.126666667	6.38
8	Triangular	20.348	0	20.348	40.696

Table 5 : Estimated velocity and acceleration for each path segment

Tawble 3: Estimated Velocity and Acceleration for each path segment				
Path Segment	dtot(m)	ttot(s)	Vmax(m/s)	amax(m/s^2)
1	0.28702	4.52	0.09525	0.06321902655
2	2.14122	33.72	0.09525	0.008474199288
3	0.69342	10.92	0.09525	0.02616758242

4	1.58242	24.92	0.127	0.007644462279
5	0.884174	13.924	0.09525	0.02052212008
6	1.832356	28.856	0.09525	0.006601746604
7	0.40513	6.38	0.09525	0.04478840125
8	2.584196	40.696	0.127	0.004681049735

Table 6 : Estimated propulsion force for each path segment

Table 4: Estimated Propulsion Force for each path segment					
Path Segment	Finertia(N)	Fweight(N)	Ffriction(N)	Frol(N)	Fprop(N)
1	0.1264380531	19.62	4.905	1.962	26.61343805
2	0.01694839858	19.62	4.905	1.962	26.5039484
3	0.05233516484	19.62	4.905	1.962	26.53933516
4	0.01528892456	19.62	4.905	1.962	26.50228892
5	0.04104424016	19.62	4.905	1.962	26.52804424
6	0.01320349321	19.62	4.830935259	1.932374103	26.39651286
7	0.08957680251	19.62	4.905	1.962	26.5765768
8	0.009362099469	19.62	4.905	1.962	26.4963621

Table 7: Estimated required propulsion power

Table 5: Estimated required propulsion power	
Path Segment	Pprop(W)
1	2.534929975
2	2.524501085
3	2.527871674
4	3.365790693
5	2.526796214
6	3.352357133
7	2.53141894
8	3.365037987

4.1.2. Move Profile

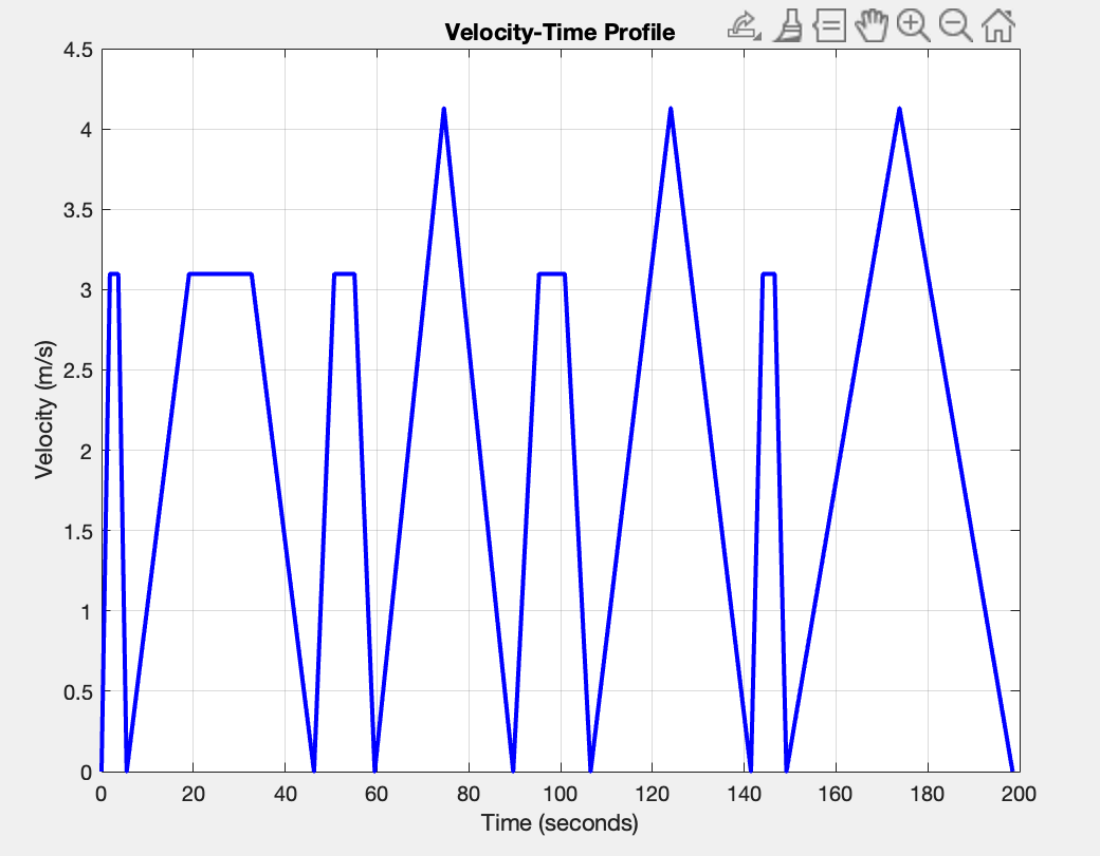


Figure 9 : Move Profile of Different Segments

Table 8 : Frictional Coefficients For Drive Type

Minimum Friction Coefficient for FWD	μf	mu_f	0.3247549876	-
Minimum Friction Coefficient for RWD	μr	mu_r	1.531847875	-
Minimum Friction Coefficient for AWD	μawd	mu_awd	0.2679491924	-

Table 9: Wheel Torques

Front Wheel Torque	Tfw	T_Fw	0.5245687178	N-m
Rear Wheel Torque	**Trw**	T_Rw	0.5245687178	N-m
All Wheel Torque	Taw	T_Aw	1.049137436	N-m

Table 10 : Normal Force on Wheels

Normal Force on both Front Wheels	Nf	Nf	15.63649477	N
Normal Force on both Rear Wheels	Nr	Nr	3.314969944	N

It should be noted that the force of gravity acting along the slope is 26 Newtons.

Table 11 : Beta

Fraction of weight on front wheel	Bf	betaf	0.8250810692	-
Fraction of weight on rear wheel	Br	betar	0.1749189308	-
FWD propulsion Power Based On Weight Distribution	PB_FWD	P_b-FWD	33.28245203	W
RWD propulsion Power Based On Weight Distribution	PB_RWD	P_b-RWD	7.055950184	W
AWD propulsion Power Based On Weight Distribution	PB_AWD	P_b-AWD	40.33840221	W

Based on this information, although RWD requires the least power, it requires a very high frictional coefficient. However, FWD is more feasible given that its frictional coefficient is actually attainable (prevents slippage). Overall, AWD is the best option because it provides the best traction, and for that reason it has been selected.

4.1.3. Motor Torque Requirements

Table 12: Estimated maximum required propulsion power and motor torque along each path segment

Table 6: Estimated maximum required propulsion power and motor torque along each path segment		
Path Segment	Pprop(W)	TProp(N-m)
1	2.534929975	1.176694154
2	2.524501085	1.171853147
3	2.527871674	1.173417748
4	3.365790693	1.171779775
5	2.526796214	1.172918527
6	3.352357133	1.167102961
7	2.53141894	1.17506436
8	3.365037987	1.171517724

4.1.4. Disk Retrieval Calculations

5. Picking up the sandwich disks:

$$F_g = (2 * m_{disk}) * g = 2 * 0.0195\,kg * 9.81\,m/s^2 = 0.382\,N$$

For lifting: $F_{Max} > W$

For sliding: $F_{Max} > F_{friction}$

Torque: 1.7 kg-cm at 4.8V

1.9 kg-cm at 6.0V = 1.9 * 9.81 N * 0.01 m = 0.18639 Nm

Radius: 0.01m

F_{Max} = □/r = 0.18639 Nm/0.02 m = 9.3195 N

This means the motor can exert a force of 0.95 kg at a radius of 1 cm which is sufficient to lift or slide the disks as long as they are lighter than this force.

6. Control System Design

6.1. Drive Motor Selection and Motor Specifications

6.1.1. Motor Selection

We used the motors from the ELEGOO smart car because it saved costs, allowed us to use AWD and we already knew how the motors functioned.

6.1.2. Motor Description

The motors are DC3-6V TT Motor and have a gear ratio of 1:48 at a max speed of 200 RPM which meets our needs.

6.1.3. Circuit Wiring Diagram

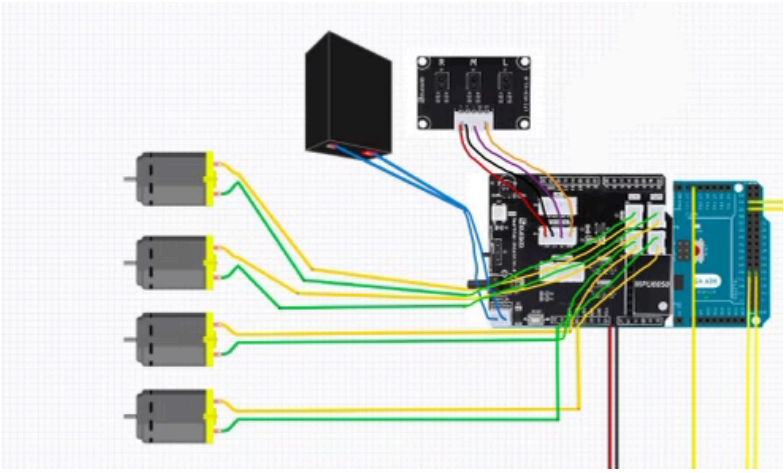


Figure 10: Circuit Wiring Diagram for the Wheel Motors

6.2. Disk Retrieval / Delivery Mechanism

The goal of the Retrieval/Delivery Mechanism is to autonomously fetch a circular disk, stack it on top of another disk, transport the stack to a delivery location, and successfully deliver it. This system can be utilized in various environments such as warehouses, manufacturing facilities, or automated storage systems. To achieve this purpose, the vehicle uses a 4-bar mechanism connected to a claw. The 4-bar mechanism rotates upward with the claw attached to the front, allowing the claw to surround the disk. The claw then grabs the disk, and the 4-bar mechanism brings the claw and the disk back down to the vehicle. This process is repeated to stack a second disk on top of the first one. Finally, the vehicle transports the stacked disks to the delivery station, where the claw releases them.

6.2.1. Motor Selection

Selecting the appropriate motors is crucial for the efficient operation of the mechanism. The criteria for motor selection include torque requirements to handle the weight of the disks and any resistance during movement, speed requirements that balance efficiency and safety, power consumption for longer battery life, and the size and weight of the motor to fit within the design constraints. Additionally, motors with precise control are preferred for smooth and accurate operations. Various types of motors were considered, including DC motors, servo motors, and stepper motors, each offering different advantages based on the specific needs of the project.

6.2.2. Motor Description

Three servo motors are employed in this project for their precise control capabilities. These motors typically offer torque ranging from 2 kg-cm to 20 kg-cm, operate at speeds within 0.1 to 0.2 seconds per 60 degrees, and are controlled through PWM (Pulse Width Modulation) signals. Servo motors are advantageous due to their high precision, ease of control, and relatively compact size. In this project, two servo motors control the opening and closing of the claw, while

one servo motor manages the vertical movement of the 4-bar mechanism to lift and place the disks.

6.2.3. Circuit Diagram

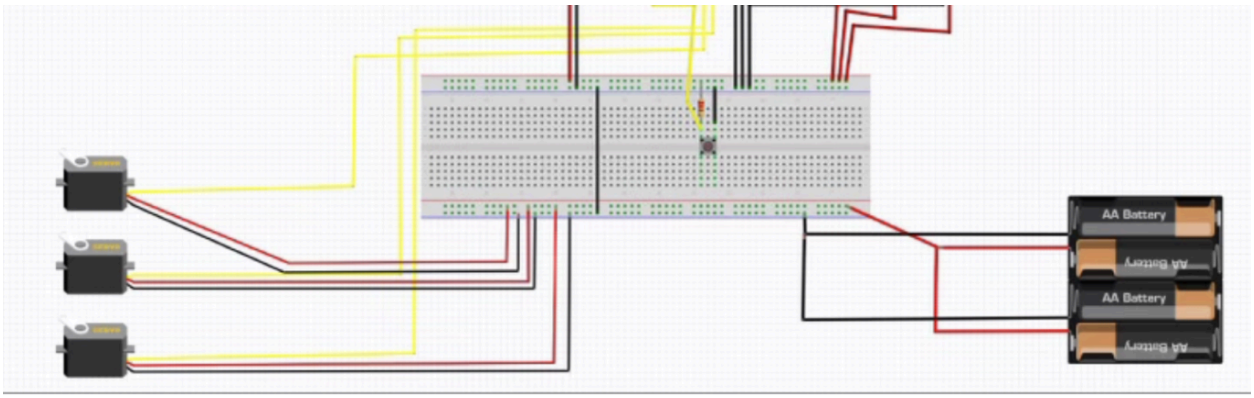


Figure 11: Motor Circuit Diagram

6.3. Sensors and Theory of Operation

Sensors are crucial for feeding information into the system, enabling it to make decisions and adjustments necessary for navigating the course. This vehicle utilizes a combination of sensors to ensure proper operation and successful completion of tasks. IR sensors detect the presence of white or black surfaces beneath the vehicle, allowing it to follow a designated path and recognize intersections for accurate stops at pickup and drop-off locations. Additional sensors, such as ultrasonic sensors and limit switches, provide critical data to avoid obstacles and ensure precise movements.

The sensors must be carefully selected to provide accurate and useful information regarding the vehicle's current location and surroundings. IR sensors need to effectively detect whether the vehicle is on the black line and provide precise feedback for correct line following. Additionally, the system requires sensors to detect obstacles in front of the vehicle to prevent collisions. Accurate sensing ensures the vehicle can navigate the course smoothly, stop at the correct locations, and avoid obstacles effectively.

6.3.1. Ultrasonic Sensors

Ultrasonic sensors measure the distance to objects using sound waves. These sensors emit a sound wave that bounces off objects and returns to the sensor, which calculates the distance based on the time it takes for the echo to return.

Ultrasonic sensors operate by emitting a burst of ultrasonic sound waves (typically at a frequency of 40 kHz) and measuring the time it takes for the echo to return after bouncing off an object. The sensor calculates the distance to the object based on the speed of sound in air and the time delay. This information allows the vehicle to detect obstacles and measure the distance to walls or other objects, ensuring it can navigate without collisions.

6.3.2. Limit Switches

Limit switches are mechanical devices that detect the presence or absence of an object by making or breaking an electrical connection. They are typically used to determine the position of moving parts within a system.

Limit switches operate through a physical actuator that is triggered by contact with an object. When the object pushes the actuator, the switch changes state (open or closed), sending a signal to the control system. This signal informs the system that a specific position has been reached or that an object is present. In the vehicle, limit switches can be used to detect the position of the claw or the disks, ensuring accurate operations during pickup, stacking, and delivery.

6.3.3. Circuit Diagram of all Sensors

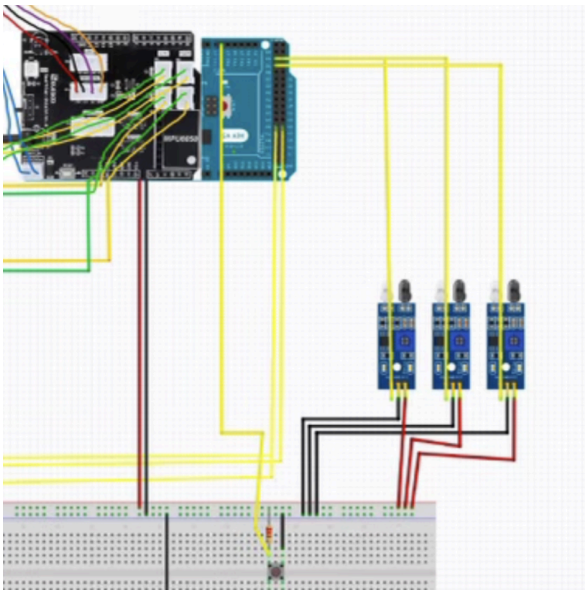


Figure 12: Sensor circuit diagram

6.4. State Diagram

6.4.1. Stateflow Charts

Simulink was utilized specifically due to its ease of use, programmer familiarity, readability, and limited need for handwritten C Code. It is primarily used to minimize the use of C Code needed in the IDE itself. This is preferable because State Flow logic is significantly easier to visualize than written C Code in the IDE. The Stateflow begins with inputs and outputs that wrap around the State Machine. These inputs are the 6 IR Sensors, as well as two input buttons for determination of the pickup location, and the drop off location. The output side features PWM for steering control, as well as motor power, servo output values, and finally a debug number for error resolution.

To begin, the robot's autonomous movement is initialized using two buttons. One is the KeyIn attached to an Arduino Shield. This has a built in pull-up resistor that increments values. The button pressed once indicates that the robot delivers on the left side, and the button pressed twice indicates that the robot delivers on the left. Additionally, a self-built pull up resistor was added, and increments in the same way with values of 1-6 to indicate which location the payload is delivered too. This was done in StateFlow by entering a state, and everytime the button is pressed, the robot increments another value that is stored in the local data, and passed to the Arduino when necessary.

The Stateflow itself is designed around a central “Drive Forward” state. The reason the Drive Forward was selected as the central state is because everything branches off of the line following. The line following was tested first, and the rest of the code is built around it. Another reason why it was selected as the central state is because line following is objectively the hardest part of robot design, at least in this case. The design of the logic is as follows. For driving forward, if the middle and left IR sensor read black, but the right is reading white, the object turns left until all black. The same is done in the inverse if the left IR sensor reads white. The values for the IR sensors are analog based, and read values 1-1000. 1 indicates whitemost conditions, whereas 1000 reads blackmost conditions. Careful measurement taking was done over countless hours to determine a threshold that allowed the robot to have very tight turning and line following.

As stated previously, counters and timers play a large role in the control of this system. As the robot begins driving forward, and enters the kitchen, the IR Sensors read all black approaching the first junction. At this point, the robot increments +1, and depending on the KeyIn, decides whether to turn left or right. When this occurs, the robot then makes its determination, and resumes driving in the drive forward state. As the robot approaches the wall, a front IR sensor is used for object detection.

The object detection IR Sensor has two purposes. One purpose is for determining when the robot should stop for payload pickup and delivery. The other is for sensing an obstacle in the venue. As the robot reads the wall the first time, a digital input is sent to the stateflow in binary. 1 indicates that the wall is reached, while 0 indicates that no object is detected. The first time the IR sensor detects an object, the servos are engaged.

The servos are primarily based on time. Using the after() function, and an arm superstate, the servos are given values in each state, and then executed in the IDE. Essentially, this means that the servos have a function call in the Arduino IDE that is initiated when entering the arm state. Using a series of five movements, the servos first move forward, the claw opens, the claw drops down, the claw closes on the disk, and the servos move back. Attached to this state is the logic for reversing, and then driving forward again. This seems simple, but actually includes a series of about four steps. The first is reversing until reading all black on the main line following strip (this does not trigger an intersection increment, as this state is not connected directly to the drive forward), idling for .1 seconds, turning right sightly until the the two rightmost IR sensors read black, and then finally driving forward until it catches the main black line, resuming the drive forward state.

At this point, the robot approaches the second junction. At the second junction, the robot once again reads all black, and because it's connected to the drive forward, it increments that an intersection is present. It turns left or right once again, and then initiates the same steps as the first pickup. This was especially useful, because the servo movement was designed so that when the claw moves forward and then opens, it drops the first disk on the second disk, drops down, and then pinches and picks up the object.

Here, the difficult part of the logic is complete. About half of The State Machine was composed of states just for the kitchen area. Now, the robot leaves the kitchen and approaches a dynamic object. Because the IR sensor for a dynamic object is already triggered twice, the third trigger indicates that an obstacle instead of a wall is present. The robot idles until the object clears, at which point it is sent back to the drive forward.

The next part of the logic is the high climb. At first, it was theorized when testing that the robot could climb the hill. With the added weight of the claw, prior to testing servos, the robot was unable to do so. A speed boost was attempted using a timer, but this proved futile due to battery issues, where the timer could not be engaged at the right time due to depleting battery life. After finding a way to increase the

speed, it was found that the robot could still not navigate the hill, likely due to a lack of torque. The solution was a weight distribution change, where the four bar mechanism was kept forward instead of all the way up when picking up the last object. With this alternative weight distribution, the robot successfully navigated the hill.

For the last part of the course, the robot navigates down the hill in the drive forward state, and makes another 90 degree left turn using the same drive forward and turning logic. The robot then counts all black on all sensors except the leftmost sensor. The counter for the table intersection increases by 1, and because the drop off button is pressed once, The robot initiates the same type of turn as in the kitchen, turning right into the dropoff. The IR sensor detects a wall, increments +1, and understands that a wall is present. It initiates the arm, and using the inverse of the servo pickup logic, the servo drops off the stacked disks in the same way. It drives reverse, once again using the same reverse logic as the kitchen, and initiates a left turn instead of a right turn, exiting the course.

6.4.2. Simulink Code/C-code

The Simulink Code was generated into a folder of .c, .h, and .mk files. These help initialize the inputs and outputs, as well as provide the logic for the entire system in the same language, C, as the IDE. These files are then added to the IDE. In the IDE specifically, the script was designed to be the interface between the robot's movements and the written logic. Essentially, this means that libraries were imported, inputs and outputs were given variables, the correct pins were initialized for each sensor, and the values were properly passed from the Simulink to the circuit board over and over using a step module. As the IDE runs the script, it consistently steps through the Simulink and passes the logic to the robots outputs so that it performs the proper autonomous movements. Three additional purposes of the IDE were the use of the servoWrite function to allow the values written in the Simulink to be converted from arbitrary values into degree values that the servos could understand. The button for the dropoff was created so that the Arduino could understand when the button was pushed, and a Debug Number, which was added to each state as a separate numeric value, was printed so that it was possible to determine which state the logic was getting stuck in during testing. Additionally, all the input values, timers, and counters from the Simulink were printed. This served as an extra way of debugging should the error state be identified, but still with an unidentifiable issue. Overall, it was the printing out of the values that made testing much simpler.

7. Product Fabrication

7.1. Chassis

For the Chassis the main method of product fabrication was laser cutting. The chassis involved a bottom plate and a top plate. Holes were then drilled into both plates to fit various parts such as the motors, electrical components, and sensors, and to be able to attach both plates together. This is seen in the plate drawings in the Appendix.

7.2. Drive system

The drive system consists of 4 motors used from the ELEGOO device. To ensure proper integration between the motor and the wheels, the ELEGOO wheels will also be used. This allows us to save costs down the line as the parts are already given.

7.3. Steering System

For the steering system, the IR sensor will be used to provide information regarding the

position of the vehicle with respect to black tape on the floor. This information is then fed into the Simulink code which directs the vehicle on what to do with this information. This sensor is bolted to the bottom of the chassis. This is designed so that the sensors have a good angle of the floor to be able to read the path the vehicle is on. The spacers used to lower the IR sensor from the bottom of the chassis were fabricated using 3D printing.

7.4. Juice box Item retrieval/Delivery Mechanism

For the retrieval and delivery, the 4-bar mechanism was used. The bars for the mechanism were fabricated using laser cutting from an acrylic material. The supports which fixed the bars to the acrylic plate of the chassis were 3D printed. The claw mechanism that picks up the disks was also fabricated using 3D printing. Since the 3D printers have some issues with tolerancing when it comes to holes, it was sometimes necessary to drill holes to make them a bit bigger or more circular. This is also seen in the engineering drawing in the appendix

7.5. Final Product Pictures

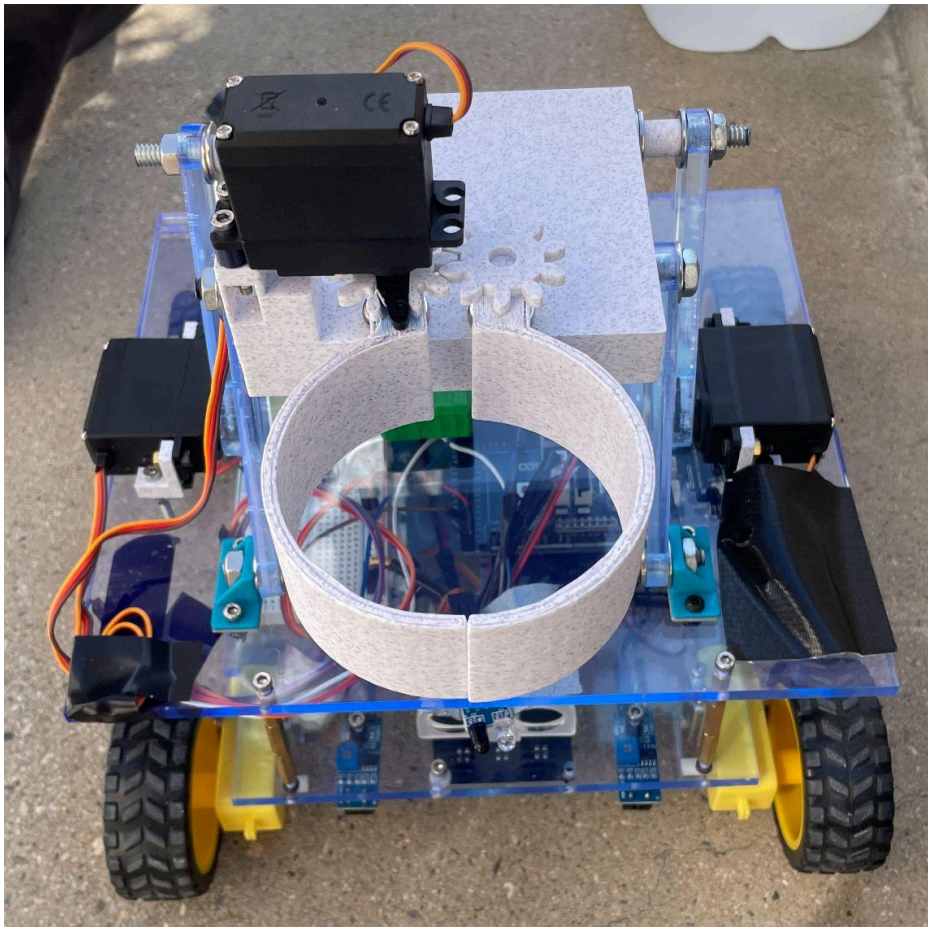


Figure 13: Final Product

8. Product Performance Testing and Evaluation

8.1. Run times

8.1.1. Patty Stack Retrieval

The total time for stacking is 36 seconds, from the second it approaches the wall until both patties are both stocked and leaving the second pickup location. An initial problem was the approach to the wall because the bot did not have enough time to correct its line following before the IR sensor detected the wall. Because of this, changes had to be made numerous times to ensure that it approached the disk properly. These include forcing the robot to turn for an extra .2-.4 seconds at the T-Junction before going back to its drive forward state, and moving the IR sensor from the side of the bot to the middle. This was an absolutely painstaking process that took a period of two weeks, including three all-nighters, to ensure that it could navigate in and out of the kitchen area. Another massive issue that contributed to the time sink was problems with battery discharge. Often, the code would work, but after successive tries it would fail. This led to constant adjustments, not realizing until we had discussed with Omar that a low battery does indeed affect the logic. Lastly, reversing out of the first pickup location and getting the robot to drive forward once again was difficult because the IR sensors for line following were placed so close together. For example, line following was not figured out when reversing, so when approaching the T a IR sensor would often catch black slightly, and the robot would enter a spin. This required reversing backwards past the drive forward line, turning right slightly, and then driving forward until the bot arrived on the black line with its three middle sensors, all before hitting the turn for the second pickup locations.

8.1.2. Starting Area

The total time for the starting area is 35 seconds, from the time that it leaves the starting box until it approaches the wall of the first pick up. The starting area was relatively easy to test, and the line following was completed in less than an hour. The only adjustments needed were steering and increasing the sensitivity of the sensor transitions to ensure that the adjustments were tighter.

8.1.3. Obstacle Area

The robot detects any dynamic object. It halts and stays idle until the object clears. During testing, an object was placed in front of the bot for a total of 1-1.5 seconds before clearing, and initiating its drive forward state once again. The obstacle area was quite easy to configure. The robot approached the dynamic object and would idle until the object cleared. Because of this all that had to be done logic-wise was driving forward, idling in a detection state, and then back to driving forward. An unexpected win was the hill climb, which the bot was unable to do with the claw placed fully back down to its starting “flat” position. With the claw being placed at about 45 degrees from zero due to issues with the four barm, the weight distribution on the bot changed, and it generated more torque on the front wheels, ensuring that they would not slip when performing the 15 degree climb.

8.1.4. Wilson restaurant

When coming down the hill into its first turn into the dropoff location, the robot takes a total of 11 seconds from the turn into its transition to driving forward out of the kitchen area. This was particularly difficult to code because the robot approached the turn into the restaurant with a lot of downhill speed. It would often miss the turn entirely. Additionally, the robot failed to increment properly with the IR sensors (when all sensors read black and the leftmost sensor reads white). This was due to the reversing action in the kitchen. When driving to the second dropoff location, the robot would read the same incrementation criteria, and think that it was now in the restaurant area. It would often turn right in the kitchen thinking it was in the dropoff location, and then the counter would be too high, missing the dropoff turn entirely. A workaround for this was to use a timer present in the stateflow, making sure the bot could not increment unless it had been running the course for a set period of time, and thus only incrementing in the restaurant. Once this was perfected, the robot was able to successfully make the turn to the dropoff with a consistency of about 20%.

8.1.5. Burger/Sandwich Delivery

If the robot successfully turned into the delivery location, the IR sensor always read the wall and would engage the servos. Because the servo action for the restaurant is the inverse of the pickup, logic design was relatively easy. The only issue here was the back wall when reversing out of the dropoff location. The bot was only to turn when all the IR sensors read black, and then it would turn left. Unfortunately, this meant the back of the bot would nearly run into the back wall, and when turning, would scrape its back left well on the wall. This nearly caused the robot to enter a tail spin. After adjusting the amount of time of the idle before turning, the robot was able to engage the turn faster, and thus exit the dropoff location and initiate driving forward with few issues.

8.2. Overall Performance

Overall, the bots performance is decently consistent. After running the bot during testing nearly 50 times, adjustments were made to the object detection IR sensor moving it closer to the middle, and the chair holding down a lip on the venue near the 90 degree turn into the restaurant was removed. After this change, the bot ran the venue on the first try, and successfully delivered the payload with no issues.

9. Work Breakdown

9.1. Work Breakdown Schedule Diagram

Week 1	**Part Ordering**
Week 2	**Design Review**
Week 3	**Electronics and Wiring Ideation**
Week 4	**Initial Printing and Laser Cutting**
Week 5	**Control Strategy Review**
Week 6	**Bottom Plate Assembly and Sensor Testing, Initial Top Plate Assembly**
Week 7	**Further Printing and Laser Cutting /Wiring Implementation**
Week 8	**Arduino and Simulink Integration + Line Following Testing**
Week 9	**Testing: Line Following, Kitchen Turns, Hill Climb, Restaurant Approaching**
Week 10	**Completed Claw and Top Plate Assembly. Testing: Line Following, Pickup and Dropoff with Servos, Object Detection, Hill Climb. Full demo and delivery Wednesday Morning at 1:45 AM**

10. BOM and Cost Analysis

10.1. Assembly Drawings

10.2. BOM

Electronics				Batteries				Body			
Price	QTY	Details	Link	Price	QTY	Details	Link	Price	QTY	Details	Link
		Servo	https://www.ama	$6.99	1	AA holder	https://www.ama	$66.54	1	24x36" Acryllic	https://www.mcmaste
$15.99	1	Servo	https://www.ama	$4.99	1	9V holder	https://www.ama	$0.00	1	Screws	-
$9.99	1	L298N	https://www.ama	$26.99	1	3.7V Charger/ba	https://www.ama	$0.00	1	Bolts	-
$9.99	1	Ultrasonic	https://www.ama	$25.99	1	9V Charger/batt	https://www.ama	$0.00	1	Washers	-
$7.79	1	IR Sensor	https://www.ama	$24.99	1	1.2V Charger/ba	https://www.ama	$0.00	1	Nuts	-
$0.00	1	Wheels		$9.99	1	3.7 V Holder	https://www.ama	$0.00	1	Caliper	
$0.00	4	Wheel Motors	-					$9.99		Hexagonal Spac	https://www.amazon.
Total Costs:	$220.23										

Figure 14: Bill of materials

The total cost of all the materials is $220.23 before tax. This is well under the given budget.

10.3. Final Cost Analysis

10.3.1. Material Costs

Component	Vendor	Cost Per Unit	# of Units to Purchase	# of Components	ASIN Number
Wheel Motors	Taking from ELEGOO	$0	0	4	N/A
Wheels	Taking from ELEGOO	$0	0	4	N/A
Infrared Sensor	Amazon	$7.79	1	5	B08215B7TF
L298N	Amazon	$9.99	1	4	B0C5JCF5RS
Servos	Amazon	$15.99	1	4	B09V5BR7J5
Ultrasonic Sensor	Amazon	$9.99	1	5	B01JG09DCK
1.2V Charger/batteries	Amazon	$24.99	1	8	B083ZMYF55
1.2 V Holder	Amazon	$6.99	1	8	B07BNMKNQX

3.7V Charger/batteries	Amazon	$26.99	1	6	B0BCW9Q5QQ
3.7 V Holder	Amazon	$9.99	1	8	B0BCW9Q5QQ
9V Charger/batteries	Amazon	$25.99	1	4	B08SL9X2YC
9 V Holder	Amazon	$4.99	1	10	B08SL9X2YC
Hexagonal Spacer	Amazon	$9.99	1	120	B0B2WDRK7W
24” x 36” Acrylic	McMaster	$66.54	1	1	N/A
MG966R Servos	Amazon	$19.99	1	4	B07MFK266B
Arduino Mega 2560	Amazon	$48.90	2	2	B0046AMGW0
TCRT500 Infrared Sensors	Amazon	$8.79	1	10	B00LZV1V10
Digital-Only Infrared Sensor	Amazon	$9.63	1	10	B07W97H2WS
Total		$307.54			

10.3.2. Labor Costs

There are 5 engineers working on this project. Using UCLA’s guidelines of 3 hours of work per week per unit in class this means that for a 4 unit class there are 12 hours of work per week. Using an hourly rate of $40 dollars per hour, this leads to a cost of $4,800 per engineer for the entire quarter. During the final phase of testing, over a period of three weeks, at any given point, a duo or trio of engineers were working an average of 10-12 hours per day for a three week period. This increased the labor cost massively. This means that the labor costs for this quarter were $24,000, and the additional cost from the final phase of testing was $26,400. This leads to a generous team cost of $50,400.

11. Testing

During the development and testing phases, several significant control issues were identified and addressed. One major problem was battery discharge, where minor changes in battery life caused significant disruptions in the robot's line-following ability. This issue was

discovered after two exhaustive nights of modifying code. If the battery was fully charged, the robot would drive too fast and miss the intersections. If the battery was not charged enough, the robot could not make the turns sharp enough. Additionally, the added weight from the arm affected the robot's performance. Keeping the arm down hindered its ability to climb hills, while keeping it up prevented it from fitting through gates. In the end, we decided to leave the arm up to allow the robot to go over the hill.

Although line following and hill navigation were largely resolved, the inclusion of a top plate and claw mechanism substantially impaired the robot's turning capability in confined spaces like the kitchen and table areas. The Ultrasonic Sensor initially used caused "jerky" movements, leading to its replacement with a TCRT5000 module. Moreover, the servo motors required extensive external testing due to an internal "zero degree" position that could not be easily calibrated, necessitating their removal and manual adjustment of the servo horns. Lastly, discrepancies between the CAD designs and real-life components led to tolerancing and manufacturing issues, further complicating the assembly and functionality of the robot.

Results

The robot successfully navigated the course using line tracking, efficiently picked up two disks, and correctly stopped in response to a dynamic object. It managed to climb the hill without any issues, successfully dropped off the disks at the designated location, and returned to the starting point. The entire run was completed in an impressive 2 minutes and 5 seconds.

Conclusion

Enhancements to the robot's design include better attachment methods for the servo parts to the claw or links, ensuring more secure and reliable connections. Adjusting the length of the four-bar links could optimize the claw's range of motion and improve overall efficiency. Additionally, creating a casing for the chassis would protect the electronics from damage and environmental factors, enhancing the robot's durability and longevity.

The robot exhibits several weaknesses that impact its performance. It sometimes loses its way while navigating the path, particularly entering the kitchen at a bad angle and backing up too far in the kitchen and drop-off table area. Additionally, it occasionally misreads a wall as a dynamic obstacle and gets stuck. The need for constant battery charging hampers its usability, and the guiding servo frequently slips out of its fitting on the four-bar linkage, affecting the robot's precision and stability.

Future improvements should focus on making the robot more compact, which would enhance maneuverability and efficiency. Adding additional electronics, such as motor controllers, could increase the accuracy of the robot's runs, while more efficient batteries would reduce the frequency of recharging and extend operation time. These steps will contribute to a more robust and reliable robotic system, capable of performing tasks with greater precision and consistency.

To conclude, this project took on the challenging task of designing and building an

autonomous robot that would help perform the normally performed by fast food workers or waiters. The unique design allows the robot to move through a specific location, following an intended route and performing operations such as putting together food orders using randomly generated combinations. Including modern technologies, such as line tracking and obstacle avoidance along with precise item handling ensures that the robot operates within the size autonomy budget power source constraints.

During the design process, several concepts and designs were looked into and it was identified that the most efficient and viable option through a comprehensive analysis using a Pairwise Comparison Chart with an Objectives Tree. The solution satisfies high-level design requirements set by the client, such as autonomy in navigation, completion of work within 5 minutes, and strict budget constraints with no compromise on performance and efficiency.

This autonomous robot is unique in comparison to existing solutions, as it combines several tasks into one device and does not require human supervision, thereby reducing the cost of deployment of multiple robots and conveyor systems. This project highlights the ability of robotics to improve operational efficiency in fast-food restaurants and provides a base for future developments concerning autonomous service robots.

Summing up, the completion of this project can be considered a great step forward in applying robotics to the service industry and provides an idea about how automated food services will look like with robots' involvement that would replace humans thus optimizing workflow and improving performance within fast-paced environments.

12. References

1] CNBC. "How fast food robots are helping address the labor shortage." [Online]. Available: www.cnbc.com/2023/01/20/how-fast-food-robots-are-helping-address-the-labor-shortage.html. [Accessed: 20-Jan-2023].

[2] The Economic Times. "McDonald's Opens First Automated Store in Texas. Check Facilities Here." [Online]. Available: economictimes.indiatimes.com/news/international/us/mcdonalds-opens-first-automated-store-in-texas-check-facilities-here/articleshow/96480220.cms?from=mdr. [Accessed: 7-Feb-2024].

[3] V. B. Ramirez, "A New and Improved Burger Robot's on the Market-and Everyone Wants One," Singularity Hub, Nov. 2020. [Online]. Available: singularityhub.com/2020/11/06/flippy-the-fast-food-robot-just-went-commercial-and-its-selling-like-crazy/.

[4] NPR, "Grocery store food prices increase, 2022 USDA report," Mar. 2022. [Online]. Available: www.npr.org/2022/03/31/1090086246/grocery-store-food-prices-increase-2022-usda-report.

[5] Centers for Disease Control and Prevention, "Food Safety Challenges." [Online]. Available: www.cdc.gov/foodsafety/challenges/index.html.

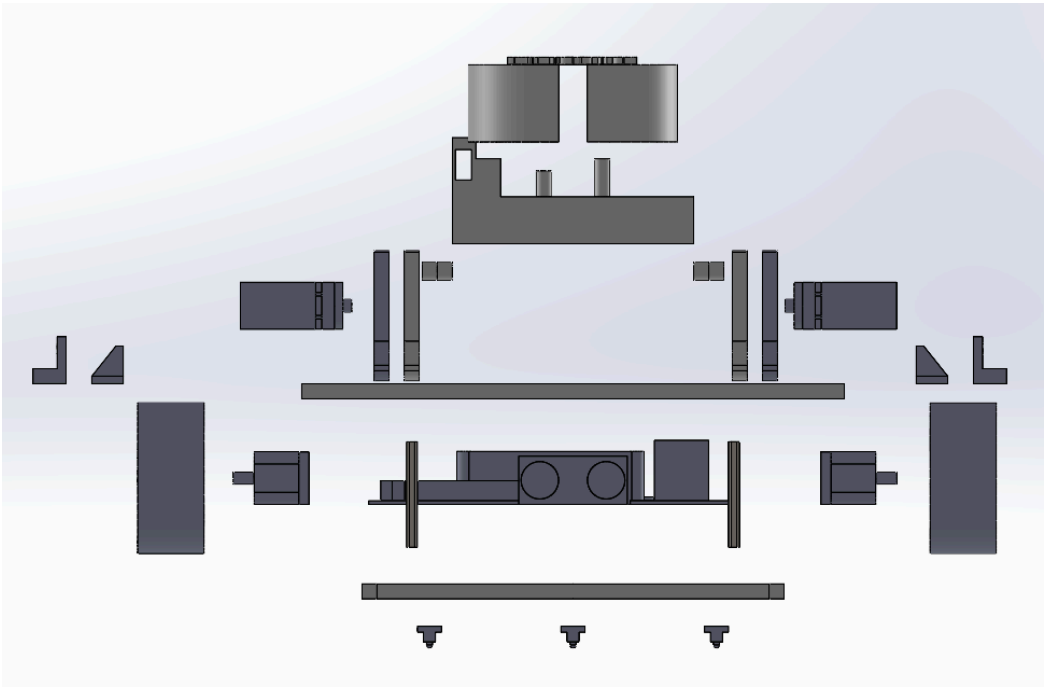
[6] U.S. Chamber of Commerce, "Understanding America's Labor Shortage: The Most Impacted Industries." [Online]. Available: www.uschamber.com/workforce/understanding-americas-labor-shortage-the-most-impacted-industries.

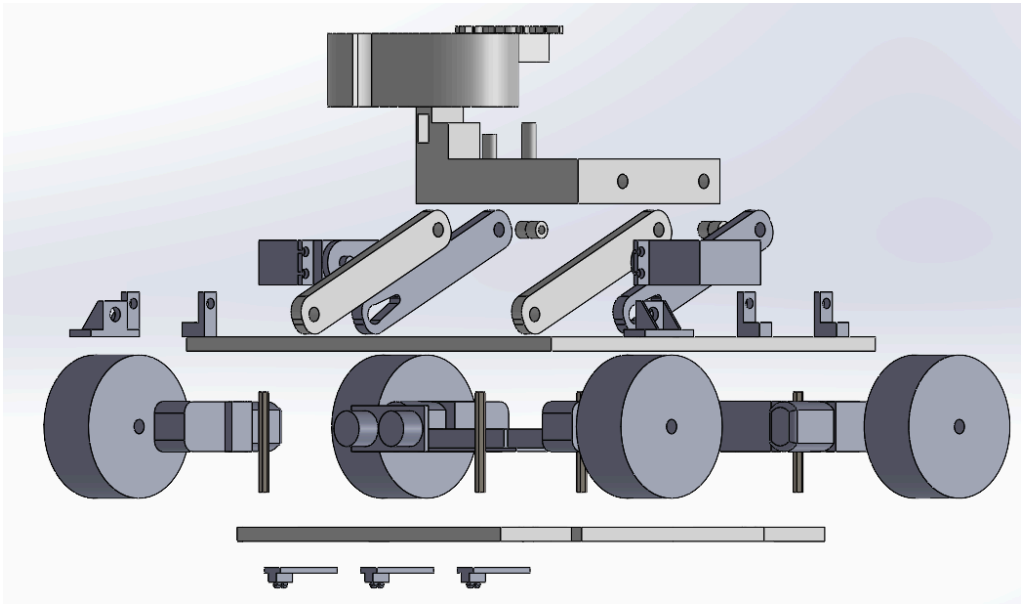
[7] N. Sahota, "AI in the Culinary World: Revolutionizing Restaurant Ops & Customer Experience," Forbes, Mar. 2024. [Online]. Available: www.forbes.com/sites/neilsahota/2024/03/13/ai-in-the-culinary-world-revolutionizing-restaurant-ops--customer-experience/?sh=5e85112f77da.

13. Appendix

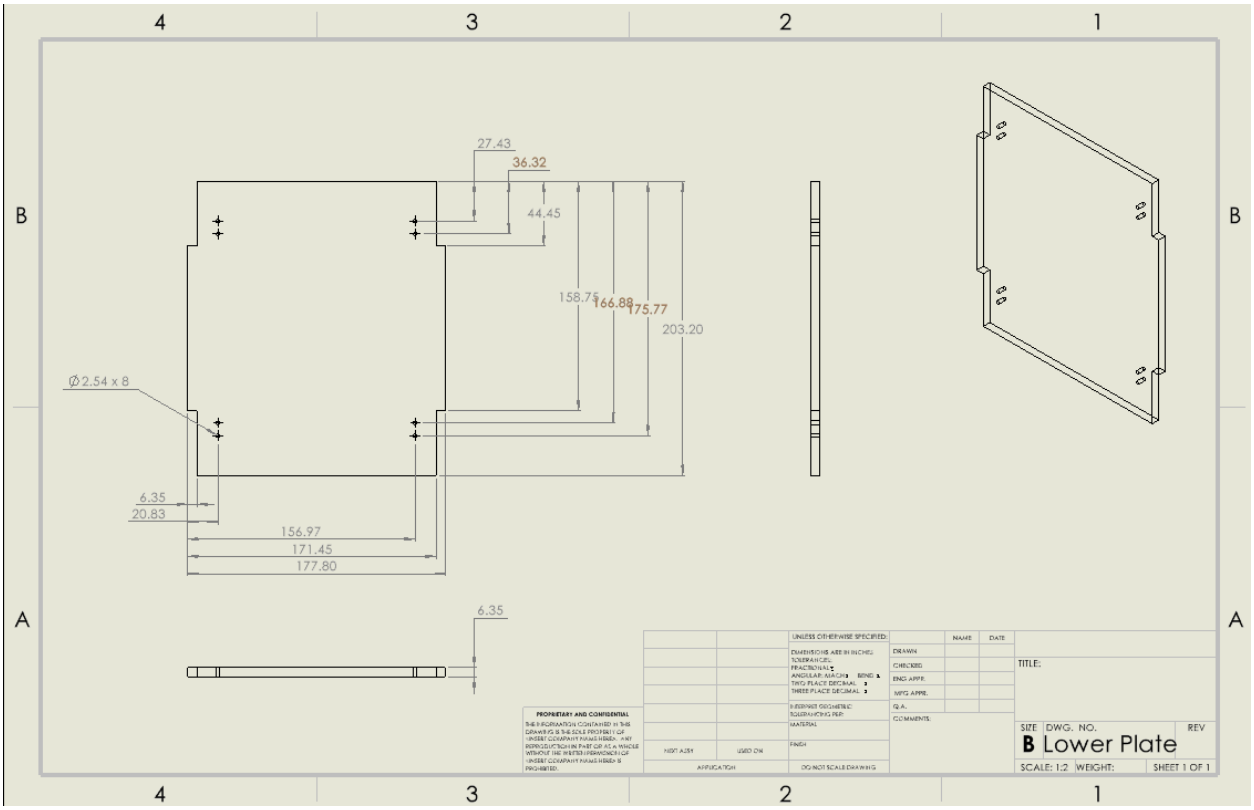
13.1. Engineering Drawings

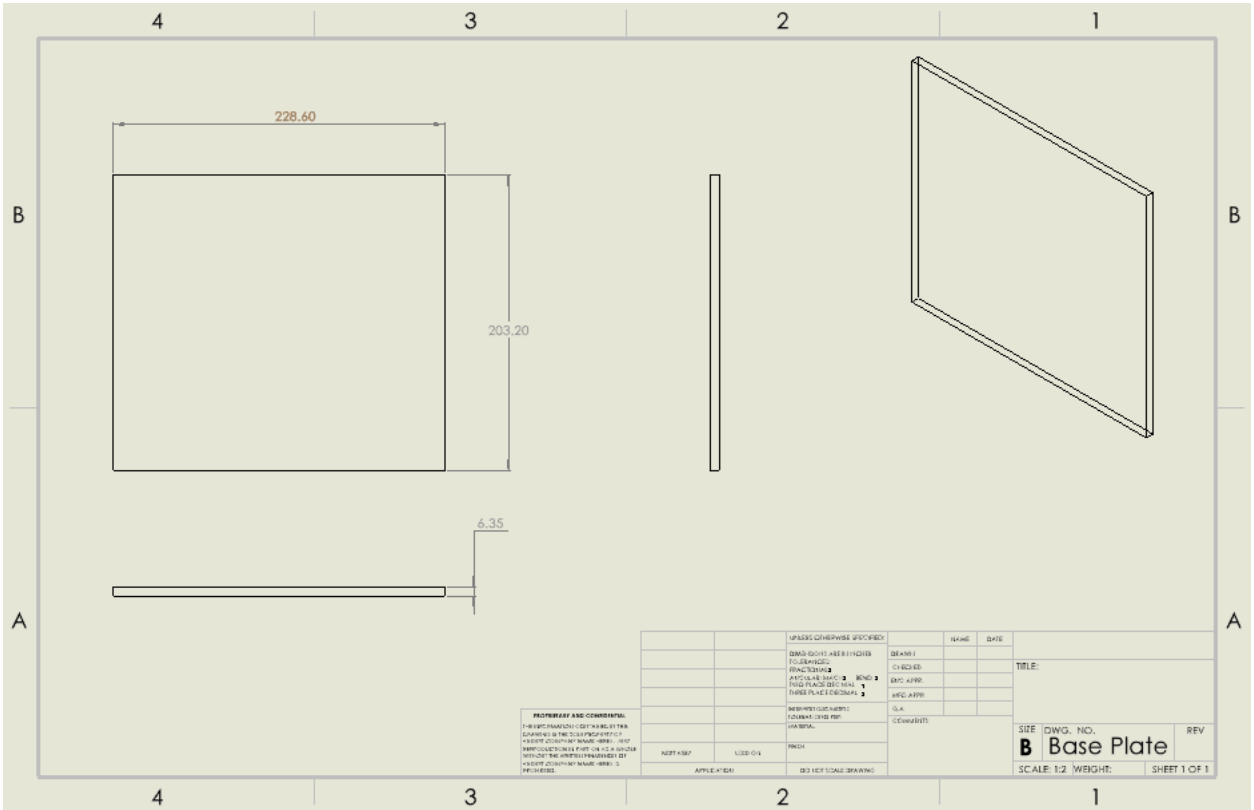
13.1.1. Exploded Views

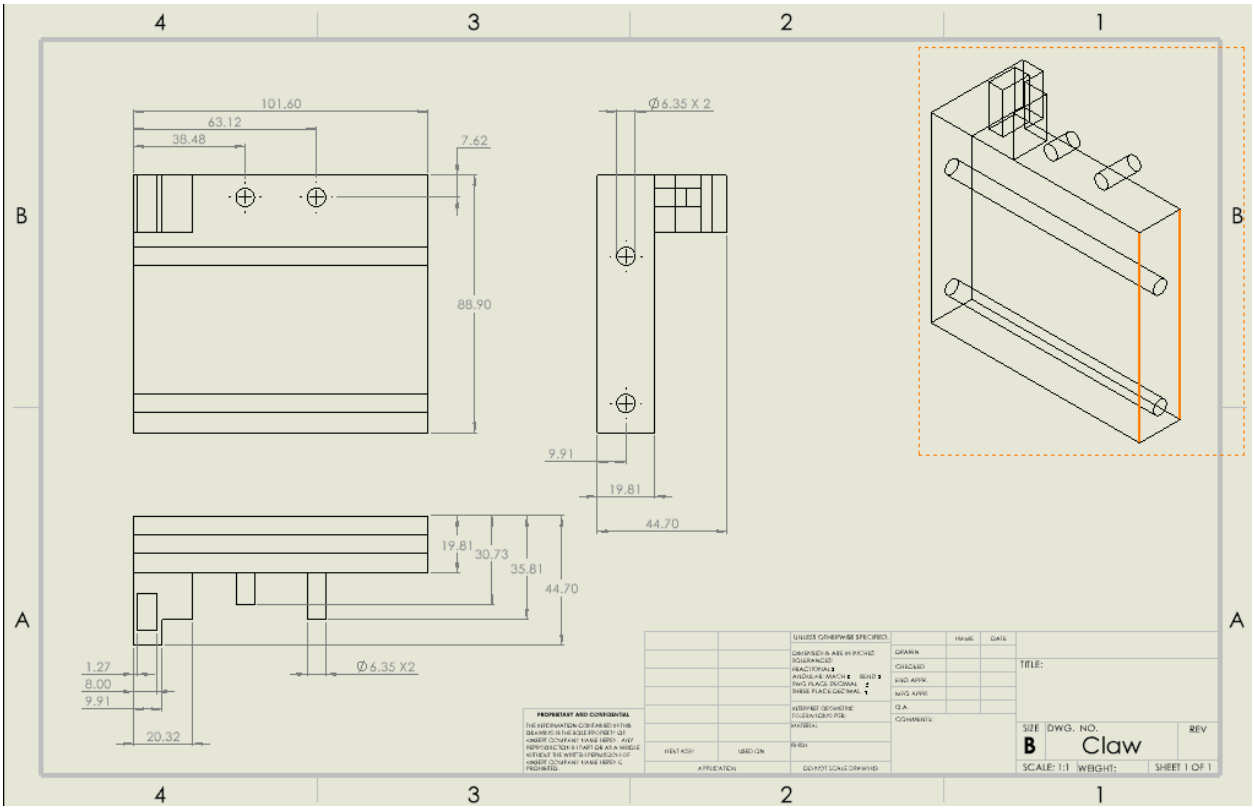


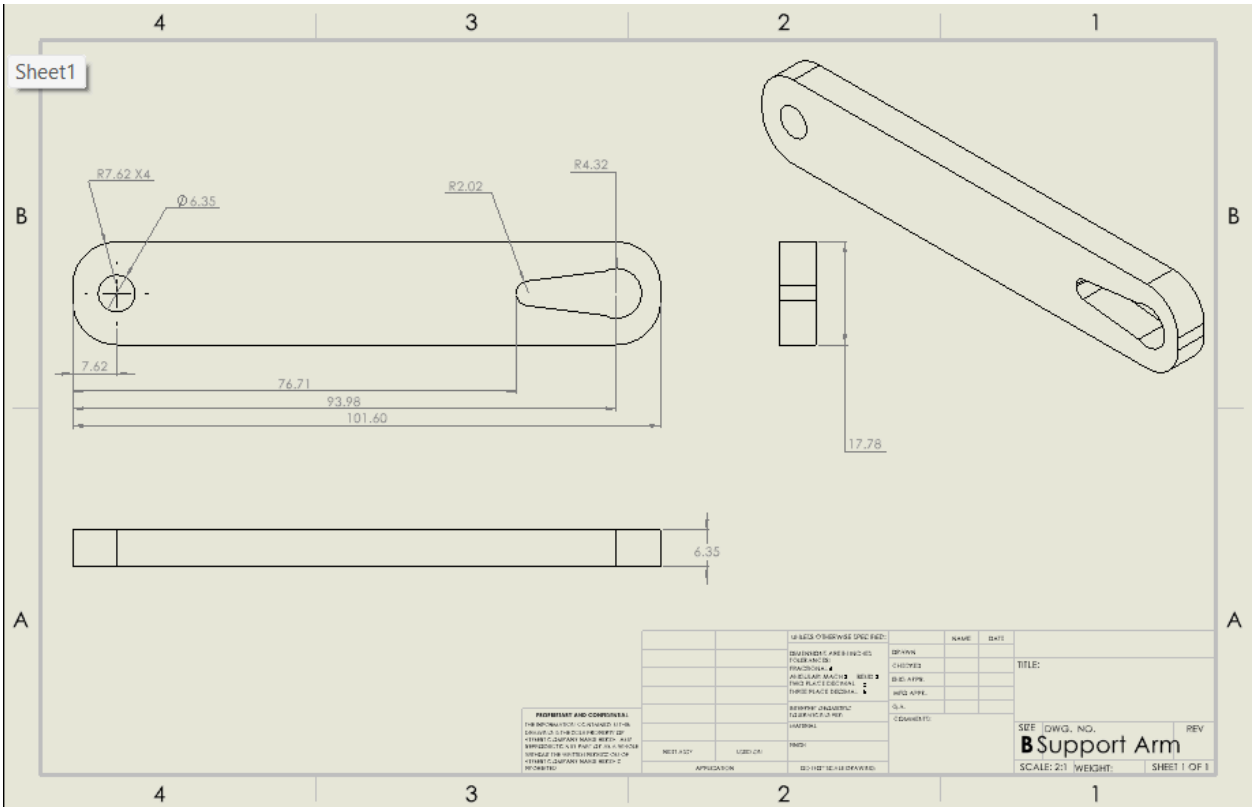


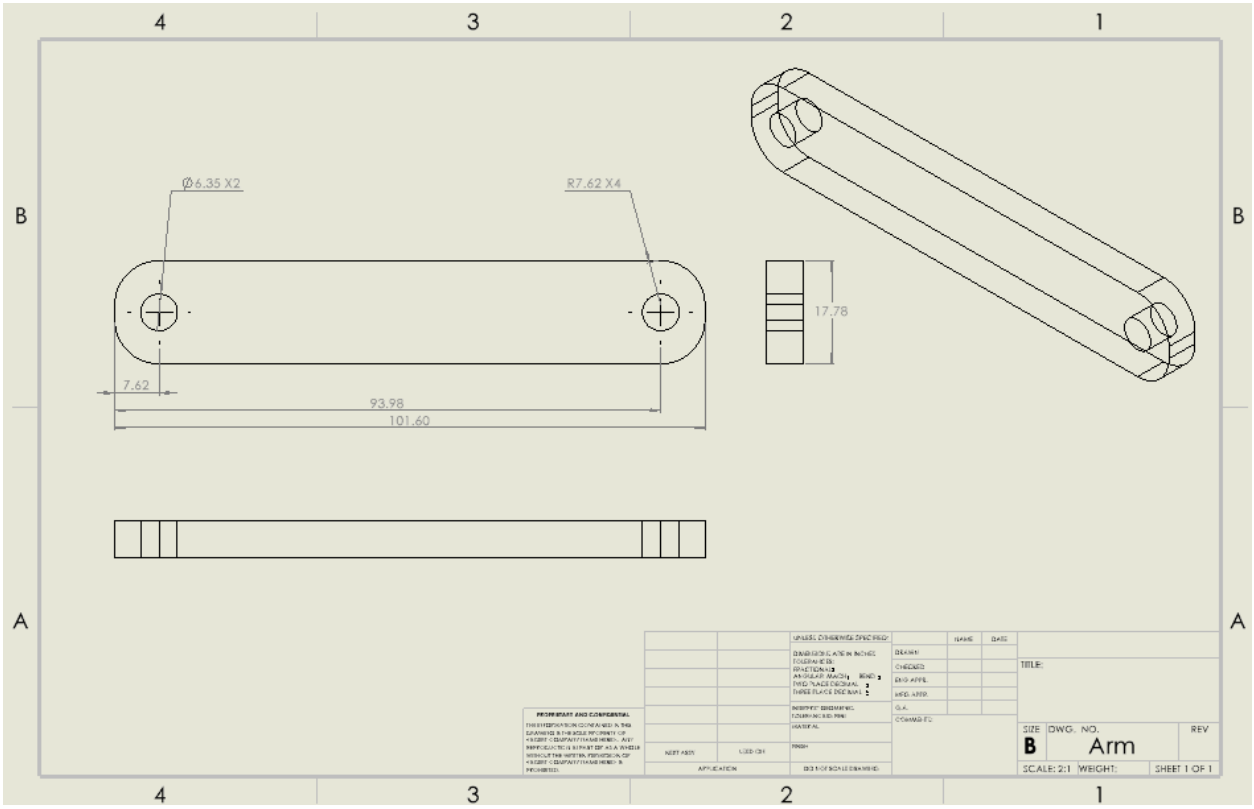
13.1.2. Engineering Drawings of Parts

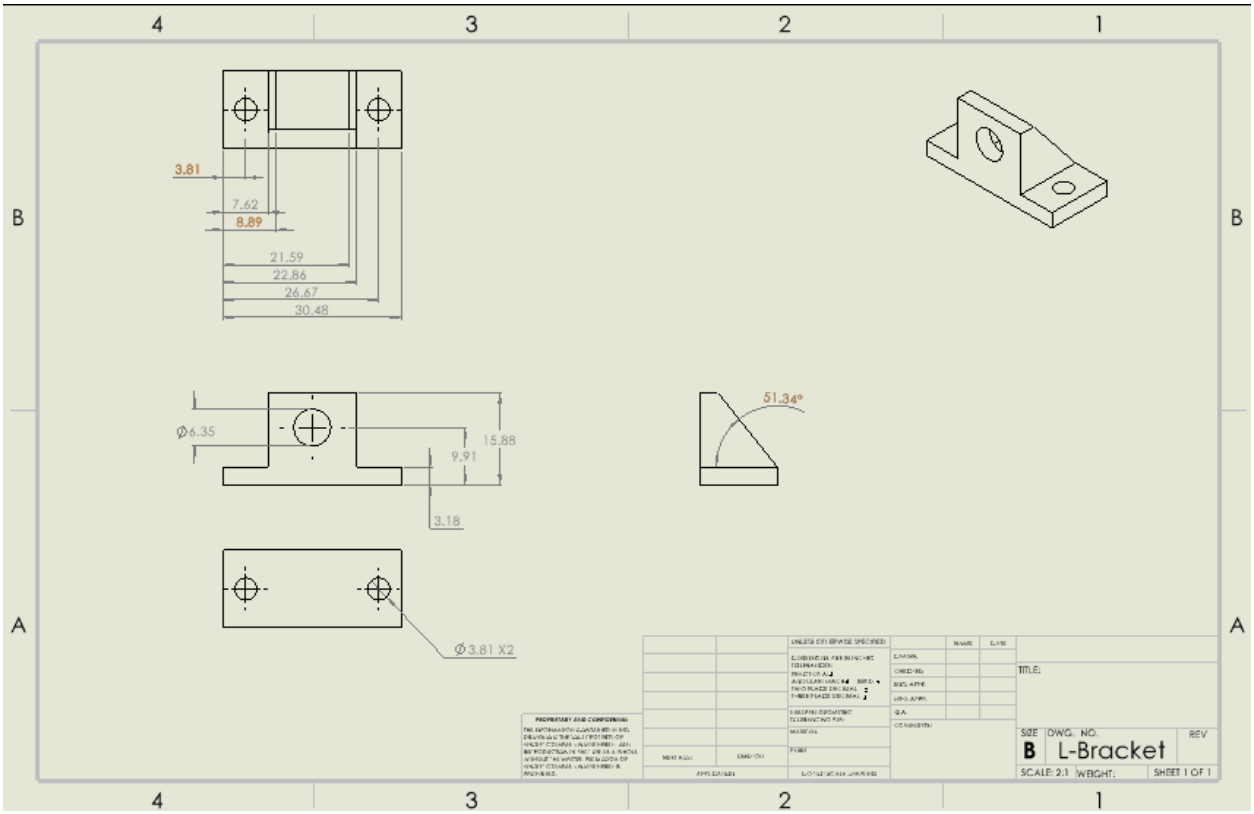

27.43
36.32
44.45
158.75
166.88
175.77
203.20
Ø2.54 x 8
6.35
20.83
156.97
171.45
177.80
6.35
B Lower Plate
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1

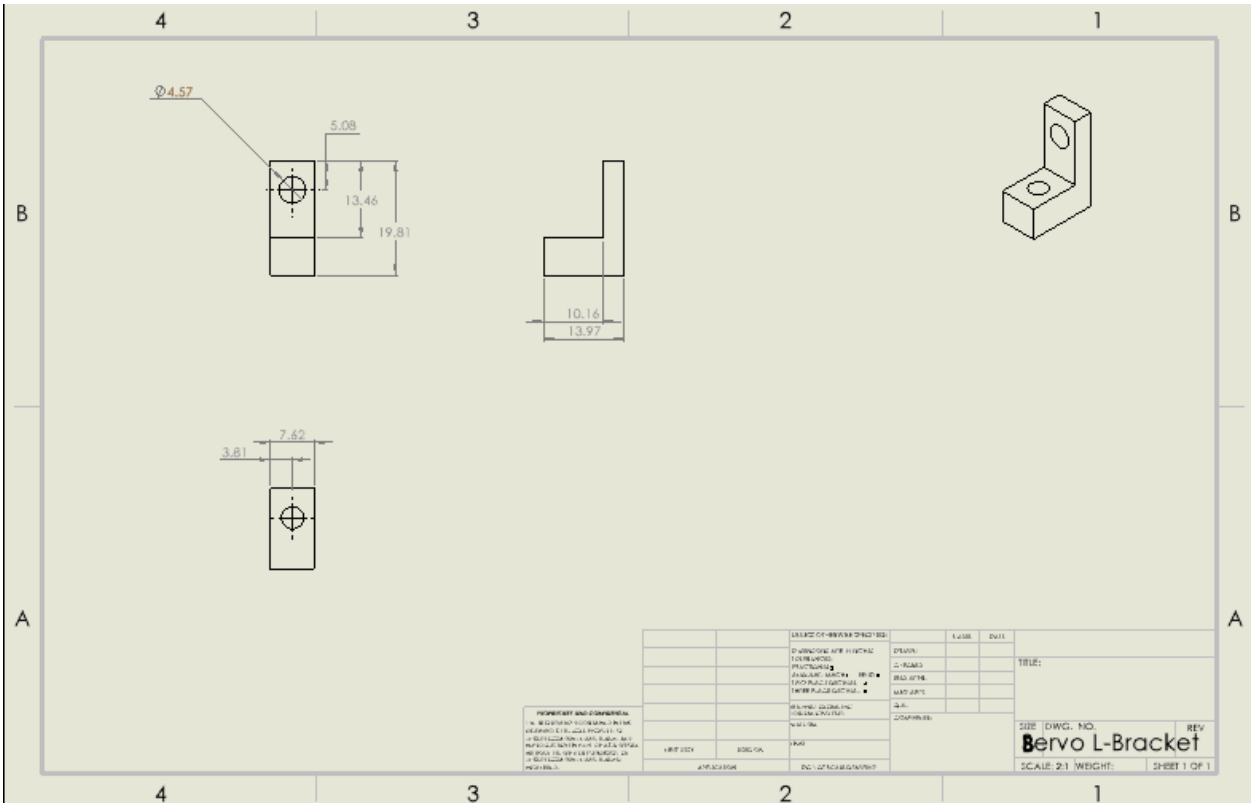

228.60
203.20
6.35
B Base Plate
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1


101.60
63.12
38.48
7.62
88.90
Ø6.35 X 2
9.91
19.81
44.70
19.81
30.73
35.81
44.70
1.27
8.00
9.91
20.32
Ø6.35 X2
Claw
SCALE: 1:1
SHEET 1 OF 1

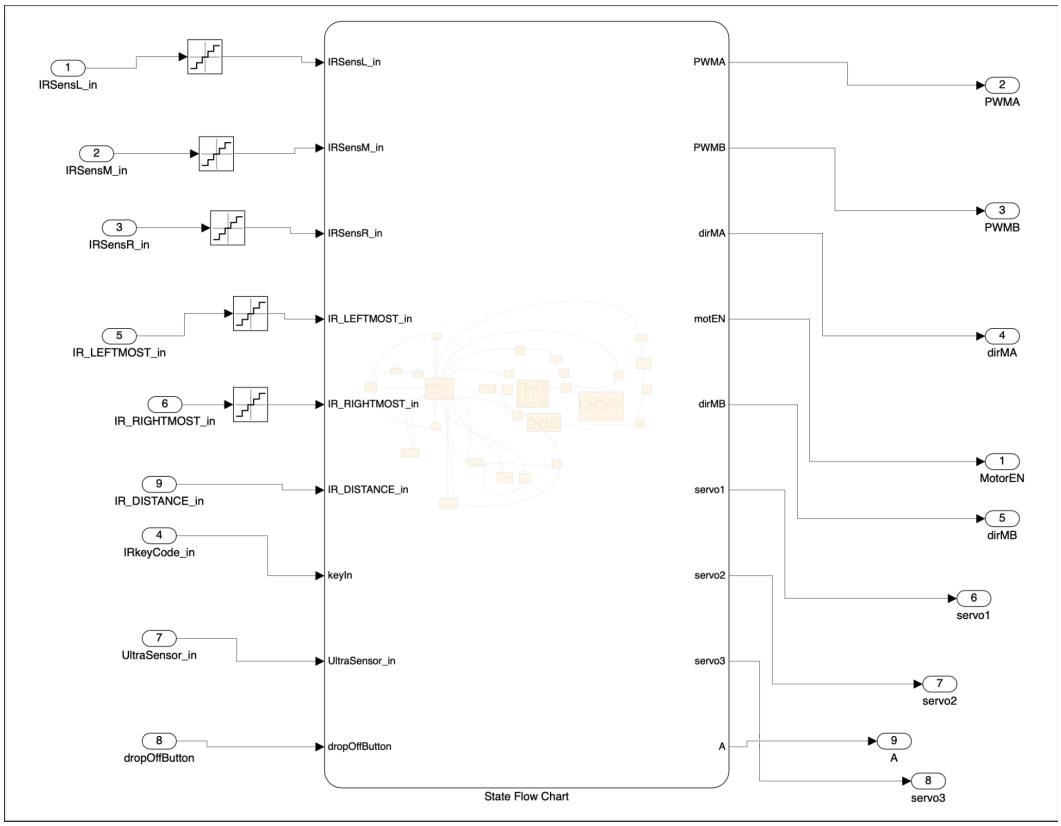

Sheet1
R7.62 X4
Ø6.35
R2.02
R4.32
7.62
76.71
93.98
101.60
17.78
6.35
Support Arm
SCALE: 2:1
SHEET 1 OF 1

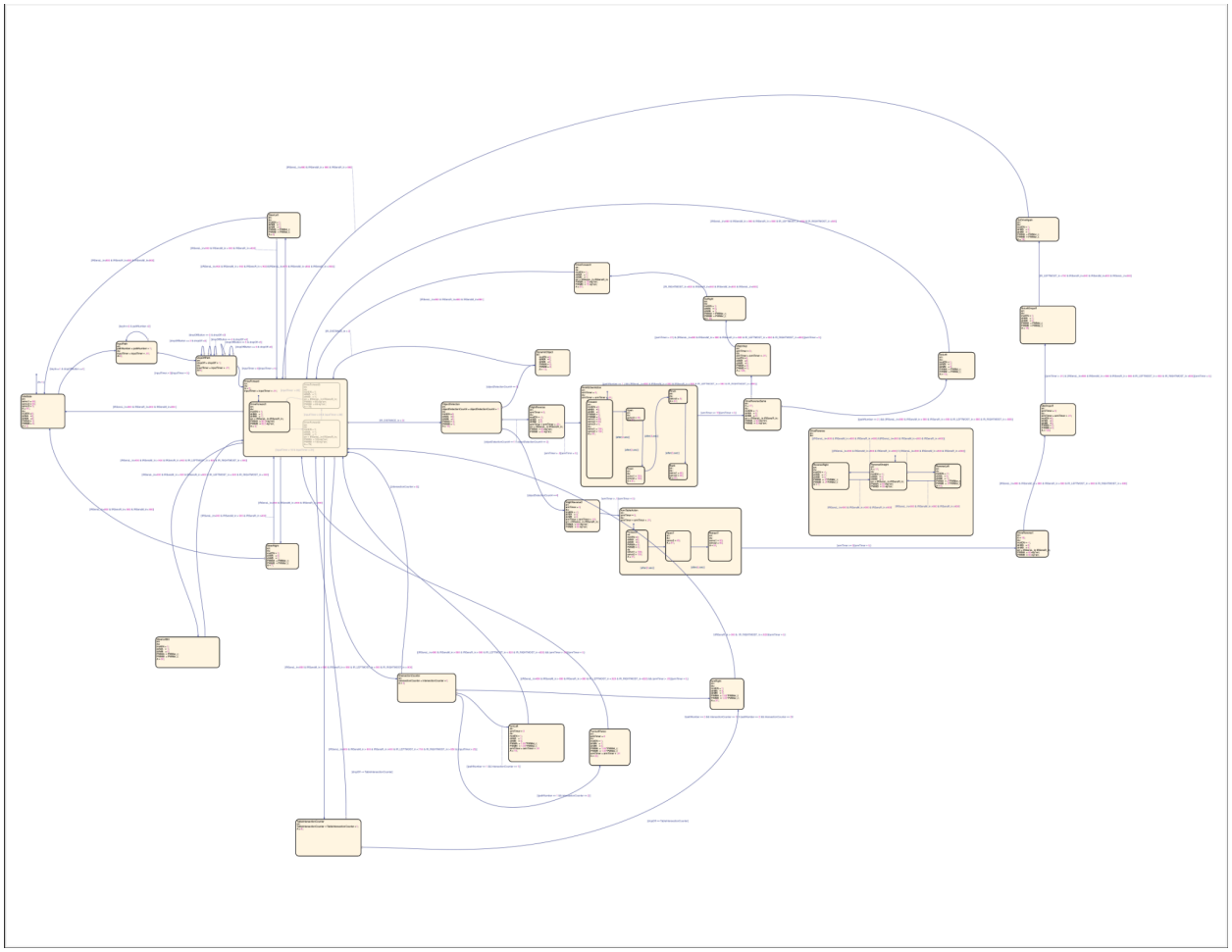

Ø6.35 X2
R7.62 X4
17.78
7.62
93.98
101.60
Arm
B
SCALE: 2:1
WEIGHT:
SHEET 1 OF 1


3.81
7.62
8.89
21.59
22.86
26.67
30.48
Ø6.35
15.88
9.91
3.18
51.34°
Ø3.81 X2
L-Bracket
B
SCALE: 2:1
WEIGHT:
SHEET 1 OF 1



13.2. Stateflow Charts





13.3. Simulink Code/ C-Code

CODE APPENDIX: /*

@Author: Christian Millar
@Date: 2024-03-24 11:30:00
@LastEditTime: 2020-06-06 01:45:00
@LastEditors: Christian Millar
@Description: OMAR 1 Robot
@FilePath:

***/**

```
#include <avr/wdt.h>
//#include <hardwareSerial.h>
#include <stdio.h>
#include <string.h>
#include "DeviceDriverSet_xxx0.h"
#include <Arduino.h>
#include "ArduinoJson-v6.11.1.h" //ArduinoJson
#include "MPU6050_getdata.h"
#include "UltrasoundByWill.h"*/
//#include <NewPing.h>
extern "C" {
#include "FinalTest.h"
#include "FinalTest_private.h"
```

```
#include "FinalTest_types.h"
}
/*Hardware device object list*/
MPU6050_getdata AppMPU6050getdata;
DeviceDriverSet_RBGLED AppRBG_LED;
DeviceDriverSet_Key AppKey;
DeviceDriverSet_ITR20001 AppITR20001;
DeviceDriverSet_Voltage AppVoltage;
DeviceDriverSet_Motor AppMotor;
DeviceDriverSet_ULTRASONIC AppULTRASONIC;
DeviceDriverSet_Servo AppServo;
//DeviceDriverSet_IRrecv AppIRrecv;
/*f(x) int */
// static boolean
// function_xxx(long x, long s, long e) //f(x)
// {
//   if (s <= x && x <= e)
//     return true;
//   else
//     return false;
// }
// static void
// delay_xxx(uint16_t _ms)
// {
//   wdt_reset();
//   for (unsigned long i = 0; i < _ms; i++)
//   {
//     delay(1);
//   }
// }
void ApplicationFunctions_Init(void)
{
  bool res_error = true;
  AppVoltage.DeviceDriverSet_Voltage_Init();
  AppMotor.DeviceDriverSet_Motor_Init();
  AppServo.DeviceDriverSet_Servo_Init(90);
  AppKey.DeviceDriverSet_Key_Init();
  AppRBG_LED.DeviceDriverSet_RBGLED_Init(20);
  //AppIRrecv.DeviceDriverSet_IRrecv_Init();
  AppULTRASONIC.DeviceDriverSet_ULTRASONIC_Init();
  AppITR20001.DeviceDriverSet_ITR20001_Init();
  //res_error = AppMPU6050getdata.MPU6050_dveInit();
  //AppMPU6050getdata.MPU6050_calibration();
  // Intialize DemoWeek 5 Parameters
  //FinalTest_P.controlEN = true;
  //FinalTest_P.dir_MA    = true;
  //FinalTest_P.dir_MB    = true;
  //FinalTest_P.speed_MA  = 128;
  //FinalTest_P.speed_MB  = 64;
```

```
}
// Initialize some variables
float Yaw; // yaw angle from the IMU
int IRSensL; // Left IR sensor
int IRSensM; // Middle IR sensor
int IRSensR; // Right IR sensor
uint8_t keyValue; // key value
float device_voltage; // pin voltage
uint16_t ultrasonic_fb; // ultrasonic reading
bool IRerror; // IR receive error
uint8_t IRrecv_code; // IR receive code
unsigned long previous_time = millis();
//for distance IR Sensor
const int pinIRd2 = 25;
const int pinIRa2 = A0;
const int pinLED2 = 9;
int IRvalueA2 = 0;
int IRvalueD2 = 0;
/* Motor Inputs */
bool dirMA;
bool dirMB;
bool motEN;
uint8_t PWMA;
uint8_t PWMB;
int Servo1_Output, Servo2_Output, Servo3_Output;
#define Servo1Pin 44
#define Servo2Pin 45
#define Servo3Pin 46
#include <Servo.h>
Servo servo1;
Servo servo2;
Servo servo3;
#define IR_LEFTMOST_PIN A8
#define IR_RIGHTMOST_PIN A9
#define Button_PIN 18
#define MAX_DISTANCE 200
// Button-related variables
int dropOffButton = 0;
static int lastButtonState = HIGH;
static unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50;
//NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DISTANCE);
void setup() {
  Serial.begin(9600);
  ApplicationFunctions_Init();
  FinalTest_initialize();
  /*UltrasoundInit();*/
  //pinMode(Servo1Pin,OUTPUT);
  //pinMode(Servo2Pin,OUTPUT);
```

```
  //pinMode(Servo3Pin,OUTPUT);
  servo1.attach(Servo1Pin);
  servo2.attach(Servo2Pin);
  servo3.attach(Servo3Pin);
  servo3.write(5);
  pinMode(IR_LEFTMOST_PIN, INPUT);
  pinMode(IR_RIGHTMOST_PIN, INPUT);
  pinMode(Button_PIN, INPUT);
  FinalTest_P.PWMsl_l = 210; // range = 0-> 255 (uint8) baseline 200,50,50,200 2baseline 200,150,150,200
3baseline 180,150,150,180 3baseline 195,185,185,195
  FinalTest_P.PWMsl_r = 205;
  FinalTest_P.PWMsr_l = 205;
  FinalTest_P.PWMsr_r = 210;
  pinMode(pinIRd2,INPUT);
  pinMode(pinIRa2,INPUT);
  pinMode(pinLED2,OUTPUT);
  //attachInterrupt(digitalPinToInterrupt(Button_PIN),ButtonStuff,RISING);
/*
  Interrupt Initialization
  TCCR1A = 0;
  TCCR1B = B00010010;  //CNCx ICESx – WGMx3 WGMx2 CSx2 CSx1 CSx0
  ICR1 = 20000;  // Set Timer Interrupt 100 Hz. If you want 100*(10)= 1 kHz, just put ICR1=10000/(10)=1000,
Similarly, using 2000 will produce 500 Hz
  //Note: Be sure the timer interrupt frequency matches with the simulink block diagram. Otherwise, the
"after" function in simulink or any other functions related to time won't be accurate.
  TIMSK1 = B00000001; // Enable Timer Interrupt
  */
}
unsigned long PreT = 0;
/*
unsigned long UltraSoundTime = 0;
bool UltraSoundWaiting = false;
int UltrasoundDis = 0;
int UltraSoundChecking()
{
  if (!UltraSoundWaiting)
  {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    //while(!digitalRead(ECHO_PIN)){}
    delay(1);
    UltraSoundTime = millis();
    UltraSoundWaiting = 1;
  }
  else
  {
```

```
   if (!digitalRead(ECHO_PIN))
   {
     UltrasoundDis = (millis() - UltraSoundTime) / 58;
     UltraSoundWaiting = 0;
   }
 }
  return UltrasoundDis;
}
*/
int DebugNum;
// ISR(TIMER1_OVF_vect){

// }
void loop() {
  //Serial.print("T:");
  if (millis() - PreT >= 10) // Runs at 100 Hz
  {
    //Serial.println(millis() - PreT);
    PreT = millis();
     //delay(50);                  // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay
between pings.
     //Serial.print("Ping: ");
     //Serial.print(sonar.ping_cm()); // Send ping, get distance in cm and print result (0 = outside set distance
range)
     //Serial.println("cm");
    // put your main code here, to run repeatedly:
    //AppMPU6050getdata.MPU6050_dveGetEulerAngles(&Yaw); // Get vehicle orientation
    IRSensL = AppITR20001.DeviceDriverSet_ITR20001_getAnalogueexxx_L();
    IRSensM = AppITR20001.DeviceDriverSet_ITR20001_getAnalogueexxx_M();
    IRSensR = AppITR20001.DeviceDriverSet_ITR20001_getAnalogueexxx_R();
    AppKey.DeviceDriverSet_key_Get(&keyValue);
    //device_voltage = AppVoltage.DeviceDriverSet_Voltage_getAnalogue();
    //AppULTRASONIC.DeviceDriverSet_ULTRASONIC_Get(&ultrasonic_fb);
    //AppIRrecv.DeviceDriverSet_IRrecv_Get(&IRrecv_code);
    /* Send fb data to Simulink Module */
    FinalTest_U.IRSensL_in     = IRSensL;
    FinalTest_U.IRSensM_in     = IRSensM;
    FinalTest_U.IRSensR_in     = IRSensR;
    //DemoWeek5_U.VoltageDetect_in = device_voltage;
    //FinalTest_U.UltraSensor_in   = ultrasonic_fb;
    FinalTest_U.UltraSensor_in  = UltrasoundChecking();
    FinalTest_U.IRkeyCode_in = keyValue;
    //DemoWeek5_U.MPU6050IMU_yaw_in= Yaw;
    //DemoWeek5_U.IRSensorCode_in  = IRrecv_code;
    FinalTest_U.IR_LEFTMOST_in = analogRead(IR_LEFTMOST_PIN);
    FinalTest_U.IR_RIGHTMOST_in = analogRead(IR_RIGHTMOST_PIN);
    FinalTest_U.dropOffButton = dropOffButton/2;
    //IR Distance
    IRvalueA2 = analogRead(pinIRa2);
```

```
  FinalTest_U.IR_DISTANCE_in = digitalRead(pinIRd2)+2;


  // Read and debounce the button
  int buttonState = digitalRead(Button_PIN);
  if (buttonState != lastButtonState) {
    lastDebounceTime = millis();
    lastButtonState = buttonState;
    dropOffButton++;
  }
  // if ((millis() - lastDebounceTime) > debounceDelay) {
  //   if (buttonState == LOW && lastButtonState == HIGH) {
  //     dropOffButton++;
  //     Serial.print("Button Pressed! Count: ");
  //     Serial.println(dropOffButton);
  //   }
  // }
  //lastButtonState = buttonState;
  /* Step Simulink Module*/
  FinalTest_step();
  /* Extract outputs from Simulink Module */
  PWMA  = FinalTest_Y.PWMA;
  PWMB  = FinalTest_Y.PWMB;
  motEN = FinalTest_Y.MotorEN;
  dirMA = FinalTest_Y.dirMA;
  dirMB = FinalTest_Y.dirMB;
  Servo1_Output = FinalTest_Y.servo1;
  Servo2_Output = FinalTest_Y.servo2;
  Servo3_Output = FinalTest_Y.servo3;
  DebugNum = FinalTest_Y.A;
  /* Send commands to actuators */
  AppMotor.DeviceDriverSet_Motor_control(dirMA, PWMA, dirMB, PWMB , motEN);
  //AppMotor.DeviceDriverSet_Motor_control(1, 0, 1, 0 , 1);
 if (DebugNum == 91) {
  servo1.write(Servo1_Output);
  servo2.write(Servo2_Output);
  servo3.write(Servo3_Output);
}
 }
 /* Verify remaining outputs */
 if (millis() - previous_time >= 1000) { // Print things here
  Serial.print(IRSensL);
  Serial.print("\t");
  Serial.print(IRSensR);
  Serial.print("\t");
  Serial.print(IRSensM);
  Serial.print("\t");
  Serial.print(FinalTest_U.IR_LEFTMOST_in);
  Serial.print("\t");
```

```
    Serial.print(FinalTest_U.IR_RIGHTMOST_in);
    Serial.print("\t");
    Serial.print(FinalTest_U.UltraSensor_in);
    Serial.print("\t");
    Serial.print(FinalTest_U.IR_DISTANCE_in);
    Serial.print("\t");
    Serial.print(keyValue);
    Serial.print("\t");
    Serial.print(DebugNum);
    Serial.print("\t");
    Serial.print(dropOffButton/2);
    Serial.println();
    Serial.print(FinalTest_DW.intersectionCounter);
    Serial.print("\t");
    Serial.print(FinalTest_DW.TableIntersectionCounter);
    Serial.print("\t");
    Serial.print(FinalTest_DW.inputTimer);
    Serial.print("\t");
    Serial.println();
    previous_time = millis();
  }
}
//void ButtonStuff(){
  //dropOffButton+=1;
  //delay(100);
//}
```

13.4. Work Breakdown Schedule

13.5. Packing Slips and Receipts