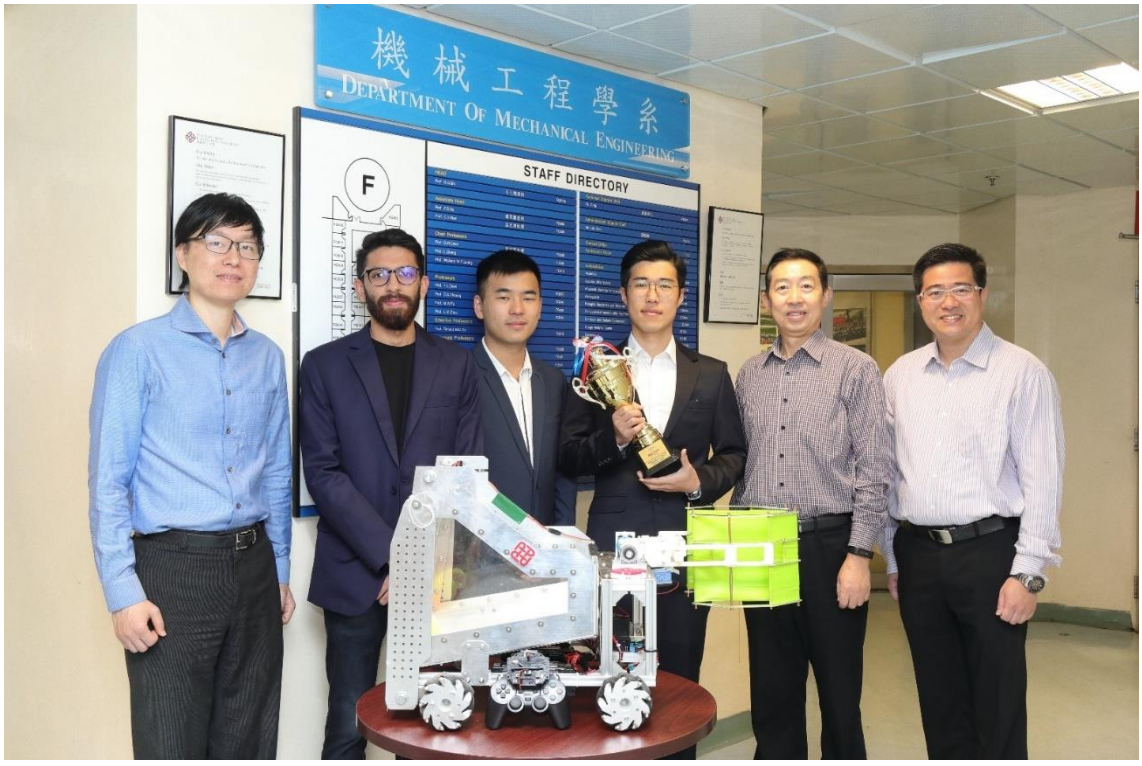

機械工程學系
DEPARTMENT OF MECHANICAL ENGINEERING
STAFF DIRECTORY
F

Acknowledgements

We would like to express our gratitude to our supervisor Dr. Henry Chu for providing constant support and feedback through the duration of this project and for actively taking part during the competition preparation phase. He was always ready to meet us and it has been a pleasure to work with him.

We would like to extend our thanks to Dr. Curtis Ng for giving us multiple opportunities to showcase our talents and for supporting us through 2 major competitions. He was always by our side when we required technical support or motivation.

The Industrial Centre at the Hong Kong Polytechnic University as well as the Project Laboratory for the Mechanical Engineering Department provided valuable support during the manufacturing stage of this project. The technical staff at the lab were very kind and extremely helpful at all times. Mr. Jack Wong and Mr. Dawson Chan travelled with us for our competitions and assisted and influenced our mechanical design greatly. We dedicate our victories to them.

We are very thankful for the assistance in procurement provided by the Mechanical Engineering Department, the University Financial Office, and the International Affairs Office (IAO).

All benefactors have played a significant role in helping us achieve our victories and in pushing us towards the completion of this project and we are truly grateful.

Abstract

Every year, ASME (American Society of Mechanical Engineers) organizes a Stu dent Design Competition (SDC) that enables engineering students from all around the w orld to participate and challenge themselves with innovative ideas and designs. The 201 9 SDC challenges team's creativity in robotics and technical design skills to brainstorm, design, develop, and manufacture a fully functional prototype that can quickly also care fully secure a variety of different balls which will be balancing on tube stands in the mid dle of a flat playing surface. After background research, the team adopted iterative desig n methodology to develop the ideas from sketch to an Arduino controlled robot, and app ropriate analysis, such as mechanism and material property, as well as the modification are conducted for performance optimization in each evaluation iteration.

Further development by augmentation of dynamic object detection and image processing abilities was carried out to demonstrate the potential of transforming the device into an AGV for garbage collection system. The team researched several different techniques and algorithms used for mainstream image processing applications and developed a program which could detect the target object, the garbage bin, and send instructions to travel from the current position of the robot to the bin via HSV filtering and other functions to complement and optimize it.

The team not only successfully secured the 2nd runner up award in ASME 2019 E-Fest Asia Pacific Student Design Competition, but also integrated additional features to meet the declared objectives with outstanding performance.

Table of Content

Table of Figures

Chapter 1. Introduction

The background of the ASME Competition and AGV application is mentioned in this chapter followed by establishing the scope for this project, problem identification, objectives, and information about project management.

1.1 Background

For this capstone project, the students have designed and created a Robot to compete in the 2019 ASME Student Design Competition (Asia-Pacific region) organized by the American Society of Mechanical Engineers, and held Vellore, India. For this competition, teams from numerous institutes around the Asia-Pacific region are required to build a device for the pick-and-place race that can quickly, but carefully secure a variety of different balls from the play area and place them in the collection area within limited time using a wireless remote controller. It is aimed at testing the competitors' technical design skills and imagination and is a perfect opportunity for the team to display their talents.

After the competition is over, the team plans to optimize and improve the robot which is why the team explored several areas of application for the robot in the real world to choose a path for development of the robot. One of the major trends catching up around the world is "going green". The Hong Kong Government, as well as The Hong Kong Polytechnic University (PolyU) are making various policies and implementing solutions around Hong Kong, and on campus to collect several kinds of waste items like recyclable waste, and food waste which accounts for 37.2% of the total solid waste in Hong Kong according to the Environmental Protection Department under the Government of HKSAR [1]. The different kinds of waste or garbage are collected separately to give it a second

life and reduce the overall waste generated. Thus, in this waste management system, collection is an important and rather resource-consuming activity, which the team thinks is achievable with our robot design. Automated Guided Vehicles (AGV) can work 24/7 with some amount of down-time for maintenance purposes, can work in the absence of light, do not need any air-conditioning, and also provide more efficiency and less errors as compared to humans, which makes them a cost-effective solution which can be employed by organizations to cut down their costs. Thus, the path forward for our machine after the ASME competition is to develop and enhance it to be used as an Automated Guided Vehicle (AGV) equipped with vision sensors to be used for the purpose of garbage and waste collection.

1.2 Scope of the Project

Since both time, and resources are limited for the team to develop a full-scale working model of a garbage collection AGV, a scope is set for this capstone project. The team will design, manufacture, and assemble a smaller, fully functional prototype for the project to generate a proof-of-concept for the design and feasibility of the idea. This prototype will also be used to compete in the ASME competition. Furthermore, the prototype will be equipped with a vision sensor to test and demonstrate its ability to identify and locate a garbage bin using python programming, traveling to it, and aligning itself with the bin, ready to then collect the garbage from it. Thus, the ultimate goal of the project is to develop a working prototype of a semi-automated vehicle which serves as a garbage/ waste collecting machine.

1.3 Identification of Problems

Upon analysing the ASME competition rules and requirements (Appendix I), the engineering problem can be broken down into several tasks and an efficient and precise robot must be designed and built using several systems to perform the different tasks. Firstly, the robot must fit the dimension requirements (50 x 50 x 50cm) and start from rest. It must then manoeuvre across the play-area with limited space available for turning, such that it does not knock over any other balls. Next, the gripper must pick spherical objects of sizes ranging from a table-tennis ball to a basketball. The robot must also be able to store a certain amount of balls for scoring maximum points; and lastly, it must have an appropriate mechanism for dropping/placing the collected objects within the starting zone. Based on these tasks for the competition, a set of clear objectives can be structured for the capstone project.

As for the machine being able to align itself with the garbage bin for collection, it can also be split into several tasks. Firstly, the object must be capable to identify the garbage bin within its field of view. Next, it must be able to compute the location of this garbage bin with respect to the robot and travel to it. Lastly, to compensate for any error, it must be able to know the orientation of the garbage bin, to check if it is misaligned, or tilted and compensate for the error.

1.4 Objectives of the Project

1.4.1 Design a Mobile Platform with High Maneuverability

An appropriate steering or manoeuvrability mechanism must be selected that allows the machine to move freely in any direction within the plane of the floor (X-Y

Plane). The selected mechanism should allow the machine to turn with a minimum radius of curvature and preferably allow it to rotate about its central axis (Z-axis). It must also allow the robot to perform the mentioned movements quickly as time is a critical constraint of the engineering problem.

1.4.2 Suitable Pick and Place Mechanism

This objective can further be broken down into three parts for a clearer structure:

(i) Universal Gripper (for spherical rigid objects)

A universal gripper must be designed such that it is able to pick up objects of various sizes. It must be strong, and sturdy enough for picking large objects such as a basketball, and precise enough to pick up small objects such as a table-tennis ball.

(ii) Storage Compartment

The storage compartment must accommodate as many spherical objects (balls) as possible and be designed in a way that prevents the balls from bouncing out / escaping it while being dropped by the gripper.

(iii) Unloading Mechanism

The machine must have a mechanism that can unload the collected balls within the play area. The balls cannot leave the unloading zone by bouncing or rolling out, so, it is important that the balls are stationary on the ground after being unloaded from the machine.

1.4.3 Efficient Microcontroller-Based System for Robot Control

For the machine to perform all the tasks mentioned above, it should implement a smart control system that allows a user to easily command it wirelessly. Its response time must be minimal with a maximum range. The user-friendliness of the controls is vital in order to reduce human error.

1.4.4 Vision Based Approach for Semi-Automation

For the machine to perform the task of achieving the correct position and alignment between the robot itself and the target object, which is a garbage bin, an appropriate and optimized software solution must be proposed, researched, implemented, and demonstrated, so that the concept of developing an AGV for the garbage collection can be proven.

A diagram to visualize and summarize the aforementioned problems can be found in Appendix II. All the objectives mentioned must be implemented on the robot and function satisfactorily, and corrective measures must be taken to solve any observed problems.

1.5 Project Management

This project having multiple objectives phases of development was expected to be difficult to manage. Thus, a Gantt Chart was created to aid the team in this aspect and can be found in Appendix III.

In addition, students have good team chemistry. Each of them has their strong points and work was allocated by keeping this in mind. All team members contributed and participated actively in all parts of this project. To smoothen the execution of different tasks, they were assigned one ‘Person-in-Charge’ (PIC). The major roles of each team member are mentioned below:

1.5.1 Shin Jiho

Jiho is the team leader. He is the PIC for the mechanical design of the robot and also for project management.

1.5.2 Chow Hung Ming Roy

Roy is the PIC for material selection and manufacturing processes. He also utilized his SolidWorks skills for parts analysis.

1.5.3 Someshwar Rudra Ajay

Rudra is the PIC for software development including micro-control system designing and programming.

Chapter 2. Literature Review

Among the market, there are many resources and products that we could refer to and learn from. In this section, a few items would be discussed and evaluated in terms of items that we can learn from.

2.1 Forklift

Forklifts and pallets are widely used in the logistics industry, especially for places where moving of heavy loads are very frequent. For example, storage warehouses, cargo handling at airports or even logistics. A forklift has a simple lifting mechanism by applying the concepts of hydraulics. In hydraulics, by using a fluid that is having high resistance to pressure, the force from one end could be effectively delivered to the other ends due to the pressure and area difference [2]. One application of hydraulic technology is hydraulic cylinders, which provides a great end linear force. This application allows the forklift to overcome heavy weight items, and to lift the heavy loads vertically.

When the forklift approaches its target, it inserts the fork extension to the pallets and lifts them up. Since heavy loads are always applied to the forklift in the front, a counterweight balance is installed at the back of the forklift for canceling out the moment generated. This application brings up design concerns of a forklift. Safety and stability are emphasized for the usage and design of a forklift [3]. In the industry, overturning of a forklift that caused injuries or fatalities accompany for 24% of forklift accidents [4]. As a result, measurements of preventing overturning incidents are focusing on the analysis of overturning. This implies that the analysis of the location of the center of gravity, moment generated by the load and counterweights and dynamics of an operating forklift should be thoroughly considered during the design of a forklift. For simplicity of stability

analysis, the stability triangle system is introduced and utilized to model the center of gravity location limits as shown in Figure 1 [5]. The stability triangle is simply having the front axle as the base of the triangle, and the tip at the middle point of the rear axle.

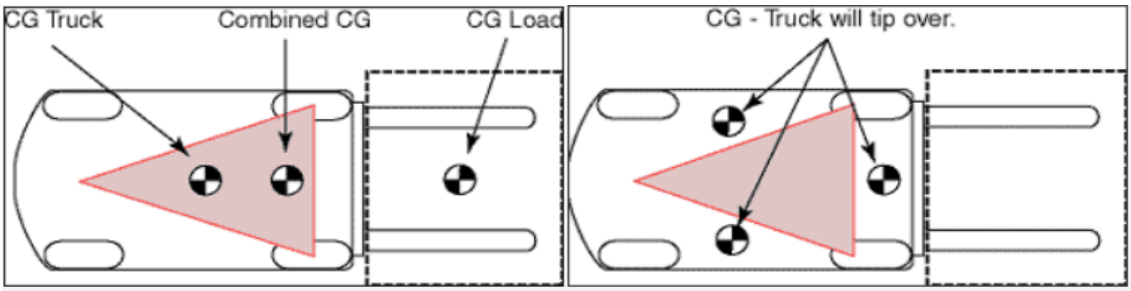


Figure 1 Schematic of Stability Triangle

Source: Retrieved from **[5]**

As the forklift is carrying a load, the final center of gravity location could be located. After knowing the location of the center of gravity, if the combined center of gravity location is lying out of the boundaries of the stability triangle, the system is exposed to overturning risks. In fact, the location of the center of gravity will change according to the location of the load, as well as the forklift motion such as making turns. At any time, the combined center of gravity should remain within the stability triangle, otherwise, it will turn.

Despite the advantages of hydraulic technology, hydraulic systems are usually very complex and heavy. Apart from that, the motions are usually slow. Therefore, hydraulic systems are not preferred in our application. As a result, the lifting mechanism and stability analysis methodology could be useful for designing the robot.

2.2 Robotic Arm

Robotic arms are widely utilized due to its high precision, repeatability and complex motion. It could replace manual handling at different fields of work that require high accuracy, as well as continuous production. For example, along with production lines of machinery, such as automobile vehicle, most of the processes are complicated and repeated to ensure continuous production. Companies of automobile production are generally having high standards regarding their product quality [6]. Therefore, precision and standardization are also key criteria for their performances. For manual fabrication, there would be a higher chance of having an error, or even worse, the products are out of quality. Therefore, in order to address the above requirement, the most efficient way is to replace workers with robotic arms along the production line. Robotic arms are programmable devices that can perform tasks just like a human arm. The application of robotic arm into the industries are relatively more cost-efficient, and more accurate with a streamlined process [6]. Robotic arms play an important role in these industries as it sustains the continuous production line. For example, Kuka robotic arms are implemented in an automobile production line as shown in Figure 2.

Figure 2 Robotic Arms in a Production Line

Source: Retrieved from **[6]**

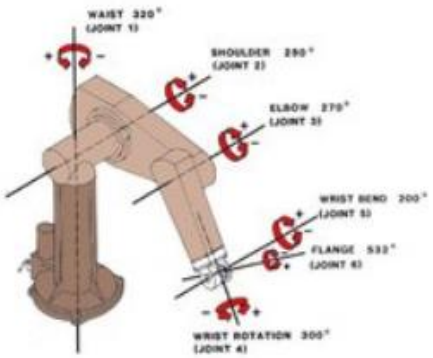


Figure 3 Robotic Arm Axes of Movement

Source: Retrieved from **[7]**

In a robotic arm, it is usually given many degrees of freedom, such that it is capable to mimic the movement of a human arm [7]. This implies that there will be more movable joints or connections due to the movement requirement. Moreover, the structural design of the robotic arm is inducting a huge moment due to its own weight. Therefore, the robotic arm components and joints will require a sturdy mechanical frame for supporting the arm.

Apart from that, controlling robotic arms are also demanding. In order to have such high accuracy and degree of freedom in motion, the programs of robotic arms are usually very complex and have many variables specifying its relative location. This will introduce more variables to be programmed. Therefore, programming for robotic arms will be tedious and require lots of effort.

Moreover, regarding the end effector of a robotic arm, the grippers are usually specified and streamlined for a particular task. Such specialization will reduce the diversity of robotic arm, narrowing down its capability of gripping objects with ranging sizes.

In conclusion, robotic arms are usually demanding a robust frame because of its degree of freedom, and that based on the degree of freedom, more variables will be introduced to program it properly. Therefore, it could be used as a reference for designing the robot with a fewer degree of freedom, so that the structural requirement will be met easier and that the programming is easier to develop. Moreover, the gripper could be designed to fit and grip a better range of objects.

2.3 Dump Cart

Apart from forklifts, another application for delivery purpose is dump carts. For dump carts, there are two major classifications. The first classification is powered with manual work, where the other is powered by hydraulic power as well. Some of the applications are construction sites where shoveling are required, as well as some gardening applications.

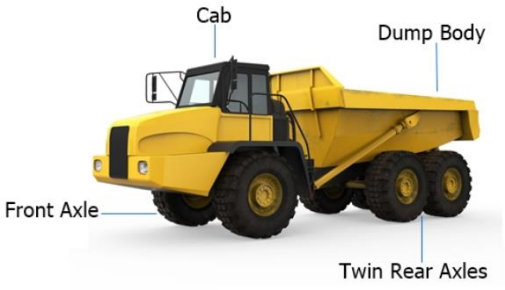


Figure 4 Example of hydraulic dump carts

Source: Retrieved from **[8]**

For the manual dump cart, it has a mechanism such that a removable box will be mounted onto a twin wheel axle. This twin wheel axle is also connected to a bar that allows the user to pull the dump cart for maneuverability. This design adopted a modular approach such that the box could be removed easily. This allows convenience when replacing the reservoir, as well as maintenance ease due to its modular approach.

As for the hydraulic dump cart, most of the applications have the reservoir mounted to the automobile directly and draws power from the automobile as shown in Figure 4. The unloading mechanism is relatively simple, by simply inclining the surface of the reservoir towards the gate or flipping over the whole reservoir by pivoting the body around the rear axles [8]. Then the objects or items inside will be dropped out of the reservoir, leading to successful unloading. However, one point to notice is that dump carts are usually used for transferring soil or items that are not fragile. As the unloading is simply dropping the items out of the container, the gravity force acting on the payload will induce an impact force as it reaches the ground. Therefore, it has a limited range in payload delivery.

From the review of dump carts, it is found to be beneficial to have a device that adopts a modular approach, which allows replacement and assembly to be performed at less difficulty. Moreover, the unloading mechanism implemented is also simple and the status of released payload could also be studied in order to make sure their condition. Therefore, it could be inspiring when designing the unloading mechanism for the robot.

2.4 ROARY (Sweden University)

In Figure 5, The ROAR Project, representing Robot based Autonomous Refuse Handling, is developed by the Volvo Group and Malardalen University in Sweden [9, 10]. It aims to develop a device for handling solid wastes such that the collection service could be streamlined and replaced by automation.

Figure 5 ROARY Robot in action

Source: Retrieved from **[12]**

The developed robot is capable of detecting the target bins, routing the pathway to the target, as well as delivering the target back and forth. Apart from that, it is also composed of various sensors, such that if something unexpected happened, the robot will immediately pause or even terminate its work and stay stationary. In order to know the environmental parameters and identify the target, the robot is integrated with a LiDAR system. This system emits laser pulses around the environment and evaluates the laser pulse feedback signals to collect environmental parameters, such that it can generate a 3D model and map of the surrounding and identify obstacles [11].

For the implementation of the robot, it is first deployed along with a drone from the rear of a refuse truck. The integrated LiDAR system, sensors on the robot and the

drone allow the robot to search and identify target refuse bins. The drone is responsible for taking an eagle view of the surrounding so that it can generate the real-time pathway and relay the path information back to the robot for following it [12]. When the LiDAR system detects an obstacle in front of the robot, it will then communicate with the drone to identify the relative location of the obstacle, and the drone will re-route a pathway for the robot to follow. As the robot reaches the target, it will then perform the "collection" stage and deliver the bin to the refuse truck.

As the robot has implemented a simple lifting mechanism, which is similar to a forklift, it can perform the collection stage easily. However, the robot structure is relatively simpler than that in a forklift. ROARY is mainly constructed with Aluminum profile and the profiles are having a high strength to weight ratio due to its special structure. The X-type structure of Aluminum profile can allow the beam to withstand more pressure, as well as more resistance to bending and twisting [13].

Once the robot has collected the refuse bin, it will then be transferred to the rear of the refuse truck as shown in Figure 6. Then, the refuse bin will be flipped over, pouring the content within the refuse truck.

Figure 6 Flipping of refuse bin

Source: Retrieved from **[12]**

This project is a very good example for us in designing the robot, due to its simple lifting mechanism, robot maneuver mechanism, as well as the material selection for the robot framework. The flipping mechanism also plays an important role in providing insights for the robot design.

Chapter 3. ASME Robot Hardware Design and Prototype Development

3.1 Engineering Design Process/Methodology

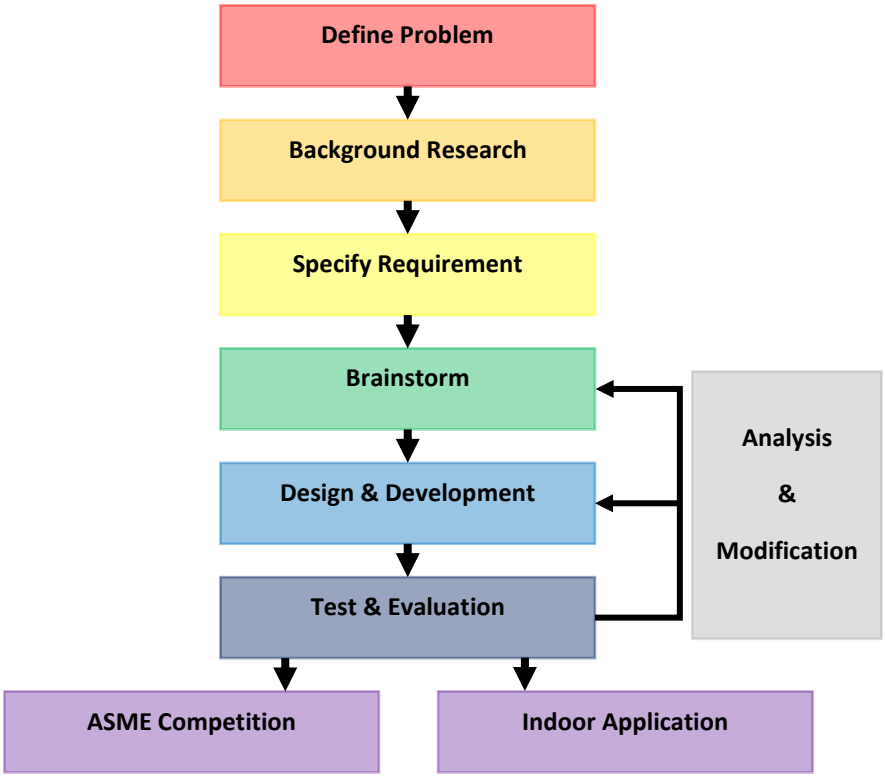


Figure 7 Engineering Design Process – Iteration Method

Before brainstorming the robot design and begin its development, the team has drawn a concrete design process and iteration methodology as their guideline so that they can always be aware of their progress throughout the project. Firstly, the 'Define Problem' stage was very obviously building a controllable mobile robot which is able to collect, store and transport different sizes of balls. 'Background Research' was also done in the literature review, where the team learned from others' experiences and existing relevant solutions from the market, to adopt appropriate knowledge, and at the same time, to avoid foreseeable mistakes in advance. Hence, the team could start from 'Specifying Requirement', and in this stage, students revised the ultimate objectives, and listed down the design criteria that should be considered in the following steps. Then, lots of ideas were brainstormed, designed, tested and evaluated. Test and evaluation process were

based on the robot's performance, efficiency, potential risks and finally whether it satisfied objectives or not. After analysis, the team may go back to earlier stages to modify the design and test the improved performance.

3.2 Design Criteria and Requirement

Below is the critical criteria and requirement should be considered for the team's robot design:

(1) All design objectives shall be considered, particularly the locomotion of the robot for multi-directional movement, and the end-effector, which is gripper in this case, for grasping various sizes of balls safely.

(2) Official regulation of the ASME competition shall be considered and must be followed strictly. For examples, the dimension of the overall robot has to be within 50*50*50cm. This regulation includes some specific cases such as the collected balls should not touch the ground unless the robot is unloading/deploying them at the starting zone.

(3) The robot shall have parts modularity for the effective development so that any modification in iteration stages will not lead to modification of entire robot, but just the essential parts shall be changed without heavily affecting others.

(4) Simple structure and mechanism with the least complexity shall be considered to reduce difficulty in control and minimize computational or human-control error.

3.3 Design Ver.1 Development – Iteration 1

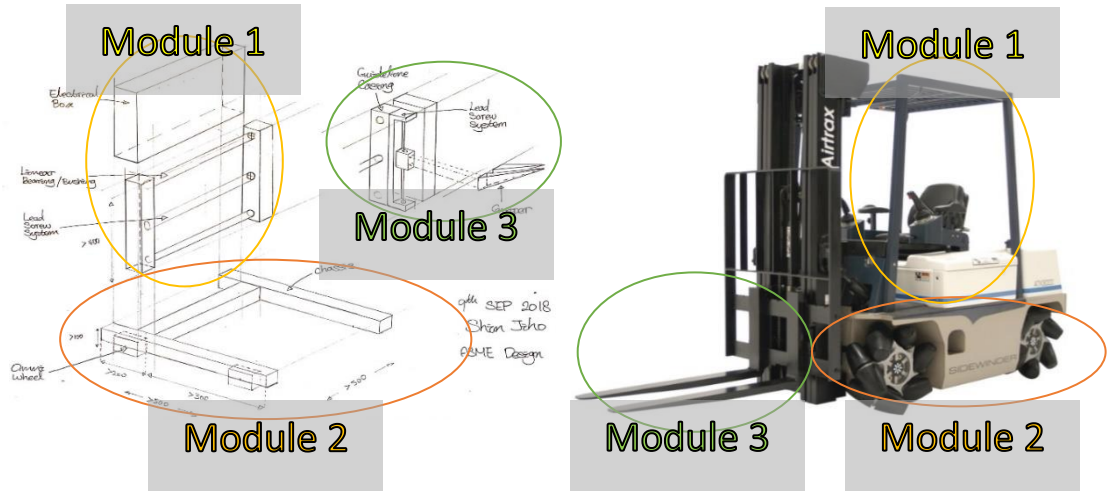


Figure 8 Modular Approach Design Concept

The very first design concept is basically derived from the forklift vehicle that the team has researched in literature review. While the majority of them are used to lift pallet in warehouses, this design was supposed to lift the ball. It had 3 modular parts such as Module 1 – Upper body, Module 2 – Lower body and Module 3 – End-effector.

For Module 1 – Upper body, the students had an idea of integrating the lead screw system, which moves two gripper parts horizontally, so that it can adjust the grasping width according to the sizes of the ball. To ensure its synchronized motion, either rack-and-pinion or lead screw system with twin threaded lead screw were considered, and since both have standardized parts available from the online market, students' concern was only focused on the implementation of it and its feasibility. For Module 2 – Lower body, different types of locomotion were studied for the robot's multi-directional movement without any complicated control required from the controller. For Module 3 – End-effector, again, the team has carefully researched on potential design options for the gripper parts for effective lifting motion.

3.3.1 Synchronized Horizontal Motion Research

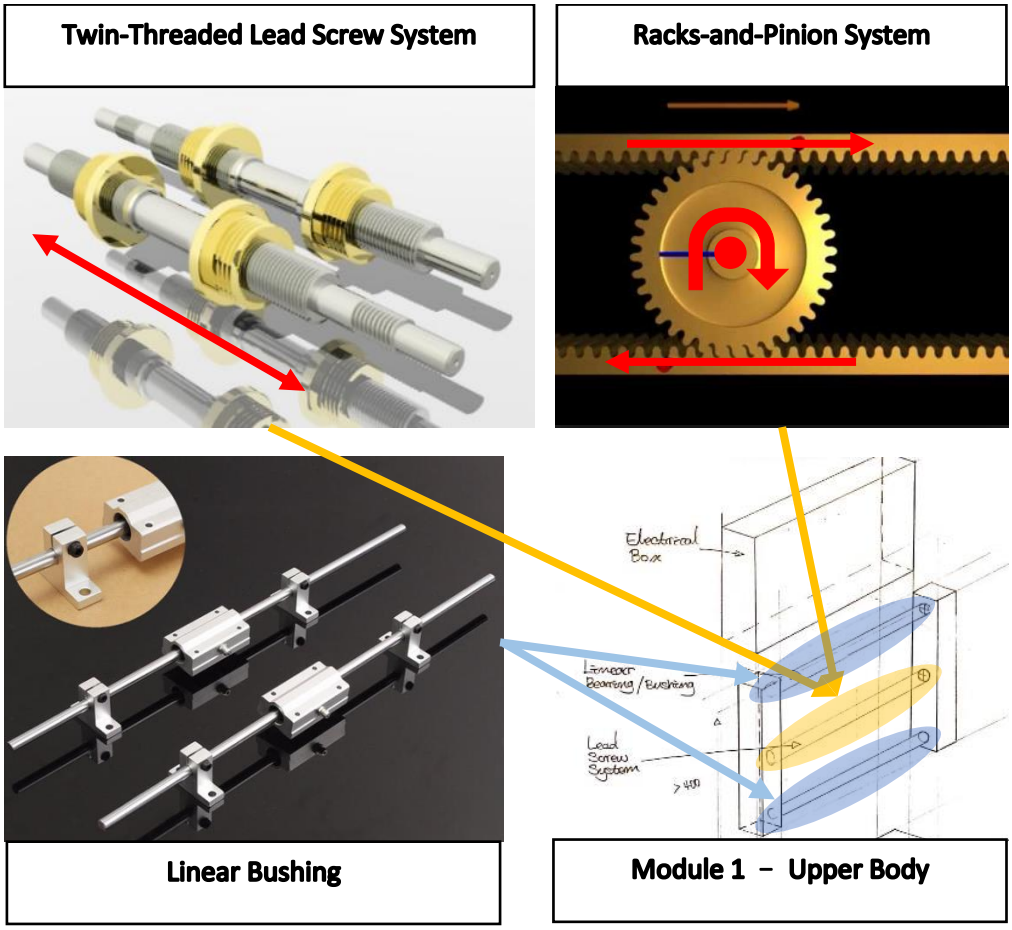


Figure 9 Horizontal Motion Element

Two different types of mechanism were briefly researched, and it indeed introduced the team with the feasibility of the synchronized horizontal motion for two separated gripper parts. With the movement element of either twin-threaded lead screw system or rack-and-pinion system, the team also considered having linear bushing as shown above, which critically assists for much smoother and robust movement.

3.3.2 Multi-directional Driving

To develop the module 2 – Lower body with multi-directional movement, students knew that differential driving method [14] will lead to difficulty in control and the time-

consuming performance. It is not only because its large turning radius in such a small playground has high potential to bump into other poles accidentally, but also it has less freedom in available motion, for examples, driving diagonally [15].

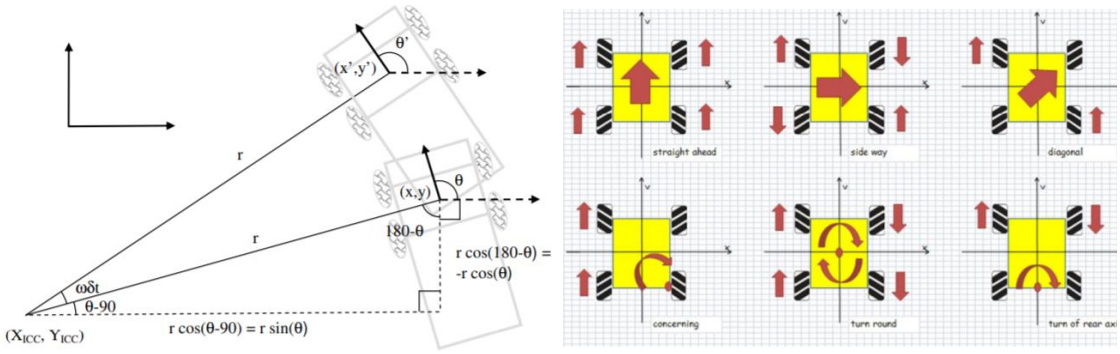


Figure 10 Differential Driving & Mecanum Driving

On the other hand, pre-programmed mecanum driving could allow the robot to drive in any direction freely without any technical control requirement. To adopt this technology in the robot, different types of wheels available for the locomotion were researched.

Locomotion is the action of a robot for transporting itself from one place to another. There are several ways in which a robot can move about a plane of motion like walking, rolling, and hopping; however, for the current application the team chooses to focus on the application of wheels. This application requires the robot to move with 3 Degrees of Freedom (DOF) on the plane of motion: 2 translational (XY plane) and 1 rotational (Z axis). Here, standard steering is automatically discarded since it would have a turn-radius and would require moving either forward or backward to change directions. Next, even though a differential drive has a 0-radius turning capability, it would also not meet the requirements because of its lack of lateral translation capability [16]. Thus, omni-directional wheels are considered for this project which can meet the requirement of 3 DOFs.

Scientists have developed several types of wheels over the past decade for robots to achieve omni-directional motion (Appendix III), and the two most widely used in research and commercial applications are omni-wheels and mecanum wheels. With the rise of the usage of these wheels in AGVs for industrial applications, modeling and control for such types of platforms have been extensively addressed [17] and will be the focus of analysis for this project as they best fit the project application.

3.3.2.1 Holonomic Drive

The holonomic drive uses omni-wheels, shown in Image 1, which are built using smaller wheels, or rollers, attached along the periphery of the main wheel. These rollers have their central axis perpendicular to the motor axis in most cases. Two or more omni-wheels are required to drive a robot, where each wheel provides traction in the direction perpendicular to the motor axis and parallel to the floor. The forces applied to the wheels by individual motors add up and provide a translational and/or a rotational motion for the robot. The kinematics and dynamics models of holonomic robots have been extensively studied and presented in the mentioned articles [18, 19].

Figure 11 Omni Wheel

Figure 12 Kiwi Drive

The most widely used models are the 3-wheeled, referred to as kiwi drive, and 4-wheeled, referred to as holonomic drive, models. Omni-wheels are attached to the periphery of the chassis as shown in Figure 12. The kiwi drive cannot be used in most medium to large scale applications due to the limited

stability it offers. Thus, holonomic drive is the predominant variation where the wheels, and respective motors, are mounted to the chassis at 45-degree angles to the axis of forward motion of the robot.

3.3.2.2 Mecanum Drive

Figure 13 End-mounted mecanum wheel

Figure 14 Central mounted mecanum wheel

The mecanum drive uses mecanum wheels, which, similar to the omni wheel, is a conventional wheel with a series of rollers attached to its circumference. These rollers have an axis of rotation at 45° to the plane of the wheel in a plane parallel to the axis of rotation of the wheel. It can have many types where two of the most common variations are i) Central mounted rollers and ii) End mount rollers, both shown in the images above. End mount rollers on mecanum wheels are sturdier, cheaper, and readily available in the market whereas centrally mounted rollers provide lower friction with the ground but are relatively expensive.

Modeling and control of mecanum wheels have been broadly address in [20, 21, 22], where the kinematics and dynamics models of such platforms are also included. Both, the mecanum and holonomic drives use 4 motors; one for each wheel, which implies that the control of both platforms is the same (Appendix IV). One of the major differences is

the orientation of the wheels and motors with respect to the chassis, which can be seen from the adjacent figure.

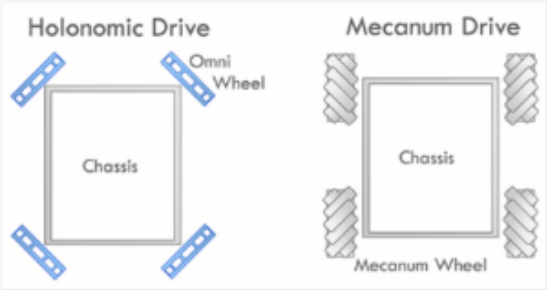


Figure 15 Comparison of orientation of wheels in holonomic and mecanum drives

There are however, some downfalls for the mecanum drive too. The rollers attached to the mecanum wheels can cause it to slip which consequently causes a decrease in positional accuracy [23]. Therefore, an additional vision-based control system must be implemented in the proposed robot if not controlled manually to provide correction for the slippage as well as misalignment. In addition, mecanum wheels cause both, horizontal and vertical vibrations in the machine because of the sequential contact points between the rollers and the ground. Bae and Kang [24] have elaborated the cause and effects of this problem by conducting several tests and also propose a modified version of the mecanum wheels to reduce this problem. Another simpler solution is the implementation of a shock-absorption system including springs or other cushioning material within the machine to reduce impact of the constituent parts. Another crucial consideration while assembling the mecanum wheels onto the robot chassis is its configuration. The left and right mecanum wheels are different due to their roller angles and must follow the O wheel configuration as shown in Figure 16 [25]. The X wheel configuration does not permit yaw rotation of the robot due to lack of moment.

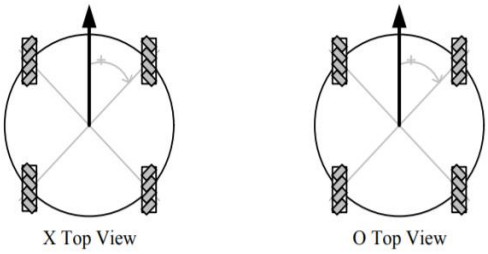


Figure 16 X and O configurations for mecanum drive

Furthermore, the mecanum drive can achieve top speed in the forward and backward direction whereas the holonomic drive can achieve maximum speed in the diagonal directions. Most of the robot motions will be in the forward and backward directions with translational motion acting as a support mechanism for our application; thus, mecanum wheels are preferred for this characteristic. Also, a mecanum wheel can generate more driving force than an omni wheel driven by the same torque and its design implementation is less intricate than the omni wheel. Overall, the mecanum wheels are the first choice for this project because of all the mentioned reasons summarized in the table below:

	Speed	Pushing Power	Climbing Ability	Build Complexity	Robustness	Motors Required	Control Complexity	Weight
Omni	Full Speed on Diagonals	Low Torque	None	Low	Low	4	Medium	Light
Mecanum	Full Speed Backward and Forward	Medium Torque	Low	Very Low	High	4	Medium	Moderate

Table 1 Comparison of Holonomic and Mecanum drives

3.3.3 End-effector Research

Grippers, the very commonly used Module 3 - End-effector, are the parts that directly interact with the environment, and they are essential for completing the desired duty of the controller. They are generally referred to as robot hands, and the types of the gripper to be chosen highly depends on different tasks, system methods, and object materials. They usually can be categorized into 4 major groups:

Gripper type	Description
1. Impactive gripper	‣ Mechanical gripper, such as jaws and claws, where the grasp is achieved by direct impact upon the objects. ‣ Most welcomed type in industrial robotic automation projects handling rigid objects.
2. Ingressive gripper	‣ Gripper penetrating the object surface (Permeation). ‣ Intrusive: Pin, needle and hackle. ‣ Non-intrusive: Hook and loop. ‣ Commonly used in textile industry.
3. Astrictive gripper	‣ Binding/attractive forces applied on the surface of objects in one direction. ‣ Vacuum suction, magnetism and electro-adhesion are commonly utilized.
4. Contigutive gripper	‣ Mechanical gripper but without producing impactive grasping forces. ‣ Some examples include using glue, and thermal adhesion (freezing or melting).

As our primary objective was collecting balls, any complex multi-fingered designs with independently actuated joints were not considered, but rather we focused on studying typical impactive grippers consisting of two rigid units with moving elements. Also, we researched several varieties of different kinematic systems (Appendix V) to choose a suitable mechanism.

To properly hold a spherical object, we first needed to secure stability of the ball in 6 velocities corresponding to translation (x, y, z) and rotation (n, o, a).

Table 2 [26] clearly shows the number of points of contact and its orientation directly affecting the objects movement. Especially for 2-finger gripper without any shape, the object has one rotational degree of freedom that may be useful to guide the ball to roll towards the direction of the collecting container itself. If we can design a gripper shape as shown in Figure 17, and assume that the ball will roll along the guideline and fall into the container by gravitational force, we can simplify our system via eliminating one mechanism of delivering the ball into the container.

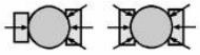
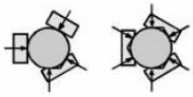
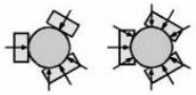
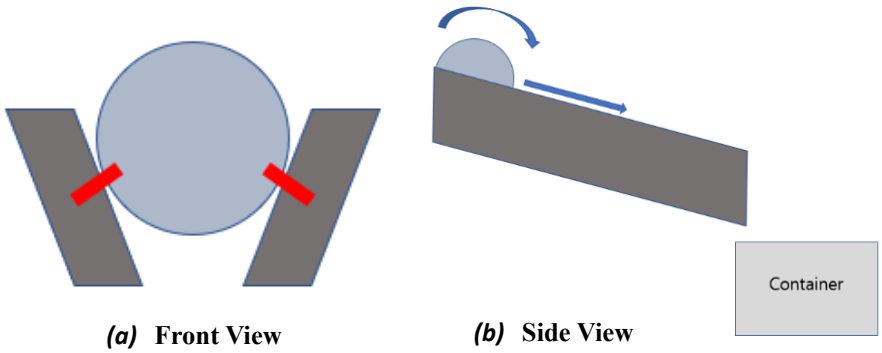
	single point contact	two point contact	multi-point contact
1 finger gripper			
2 finger gripper			
3 finger gripper			

Table 2 Number of Points of Contact



(a) **Front View** ***(b)*** **Side View**

Figure 17 Gripper Allowing Object with 1-DOF

Thus, students came up with following gripper design as shown below:

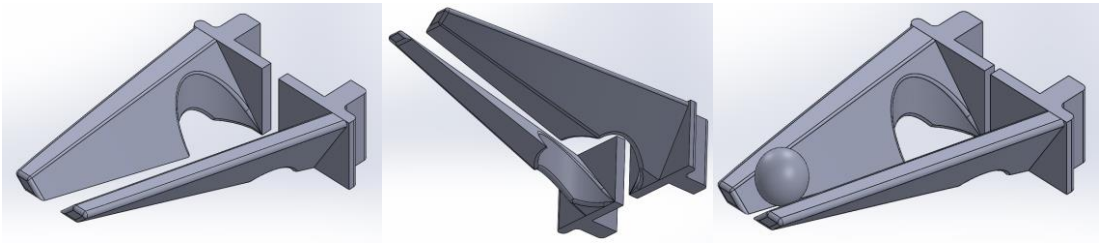
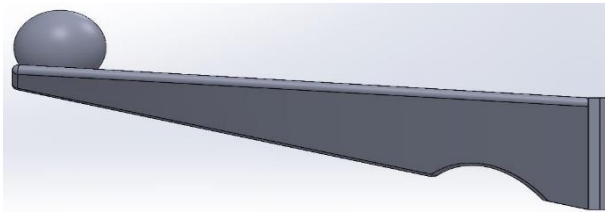


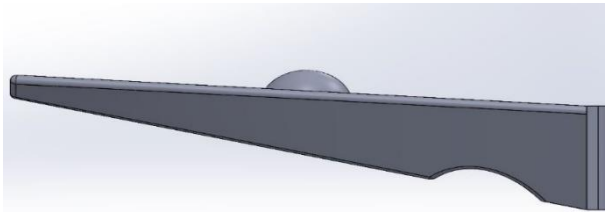
Figure 18 Gripper finger

The methodology of ball collection is clearly illustrated below in few steps:

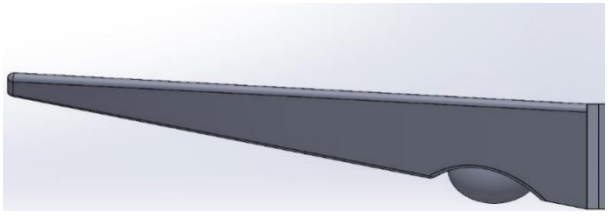
(1) Collect the ball



(2) Roll backward by gravity



(3) Fall through the hole



(4) Drop into the container

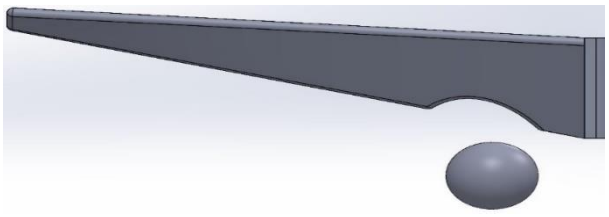


Figure 19 Collection Methodology

3.3.4 Design Ver.1 Fast Prototyping and Evaluation

To evaluate this mechanism, the team has built 3D model in SolidWorks and used scrap wooden material to rapid prototype the Module 1 - upper body.

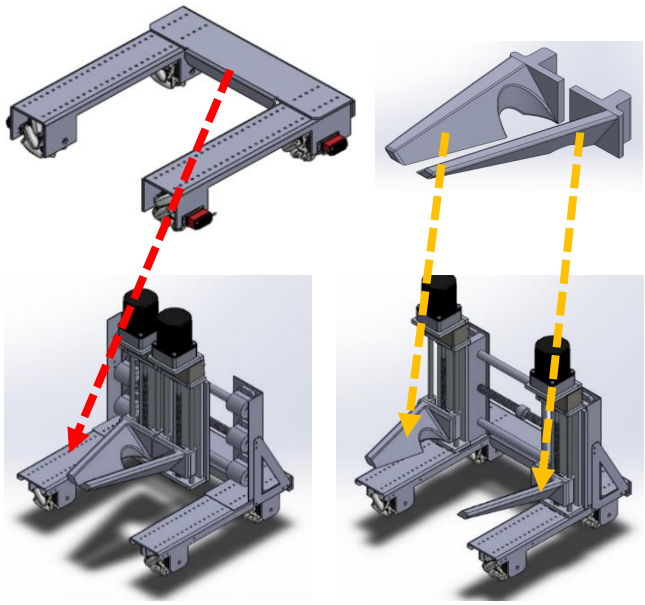


Figure 20 Design Ver.1

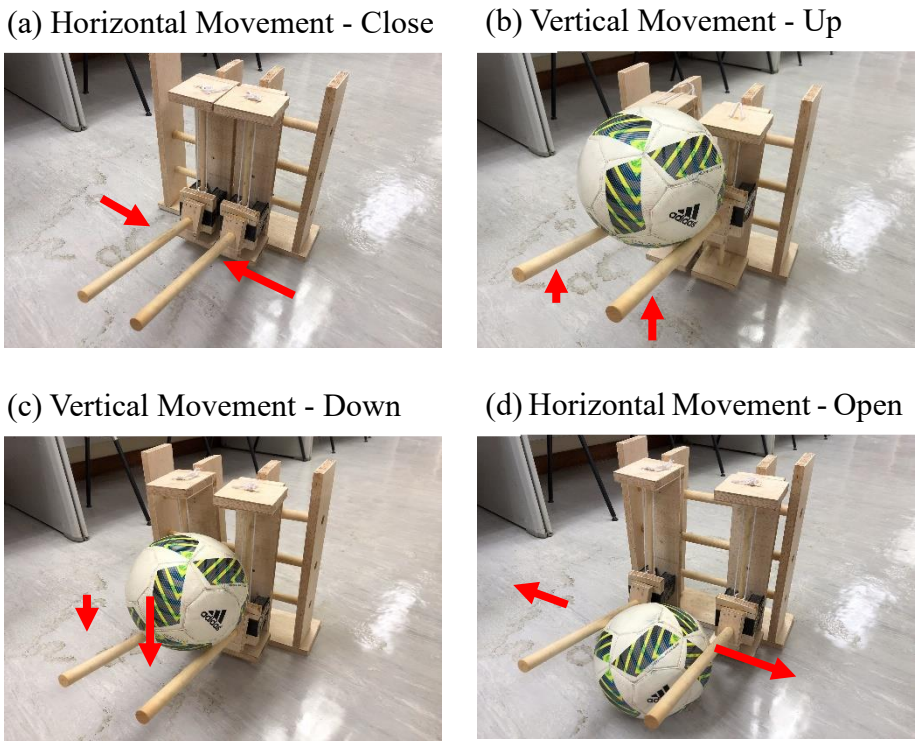


Figure 21 Design Ver.1 Prototype

Even though the football seems successfully be collected and stored with the Module 1 – upper body prototype testing, the team could find few potential problems with the design.

First, the grasping mechanism of collecting the ball was too complicated and the overall weight seemed to be heavy. The robot gripper has a horizontal and vertical movement that must be synchronized, and any minor error in this particular movement will be accumulated and lead to misalignment of gripper fingers which is very critical. Also, Module 1 has lots of motion elements and assisting parts including one twin-thread lead screw system or racks-and-pinion system for the synchronized horizontal movement, two lead screw system which moves gripper fingers vertically, and another two linear bearing mounted on the upper chassis for support. This complicated system indeed is not easy to achieve perfectly synchronized motion, and it may also result in heavy weight.

Second, it has no enough space for mounting the container for the ball-storage purpose, and so it must return to the starting zone for every single journey to place back the ball. This is indeed one of the most critical issues during the competition, where the robot will have to compete with the opponents.

Lastly, to have accurate locomotion of the robot, the mecanum wheels shall be always in the right position and angle. However, the Module 2 - lower body with only one linkage introduced a weak structure that may loosen over time, and thus the orientation of the mecaum wheels had high potential to be misaligned throughout the operation over time. At the same time, the proposed material of aluminum sheet has high potential to be bent over time, which again will lead to misalignment of the parts and even fracture failure.

Below is the summary of major issues analyzed in the Design Ver.1:

	Major issues with the Design Ver.1	Solution/Modification
1.	Complicated gripper mechanism (synchronization of gripper in both vertical and horizontal motion)	
2.	Not enough space to install the storage box (ball container)	
3.	Weak lower body/chassis may lead to inaccurate locomotion due to misalignment of mcanum wheels	
4.	Aluminum sheet metal tends to bend over time due to weight affecting overall functionality of the robot.	

3.4 Design Ver.2 Development – Iteration 2

Through evaluating the first design concept, the team was able to do second iteration brainstorming about some possible modification based on discussed issues to minimize the foreseeable risks and improve its performance.

3.4.1 Design Ver.1 Problems Analysis and Modification

To have a less complicated grasping mechanism, students have decided to eliminate horizontal movement of the gripper and use only one gripper part in bowl shape

which moves vertically to lift the ball. Via getting rid of horizontal movement and reducing the gripper parts by one, students were able to completely eliminate the error that may be caused by unsynchronized movement and reduce the overall robot weight significantly.

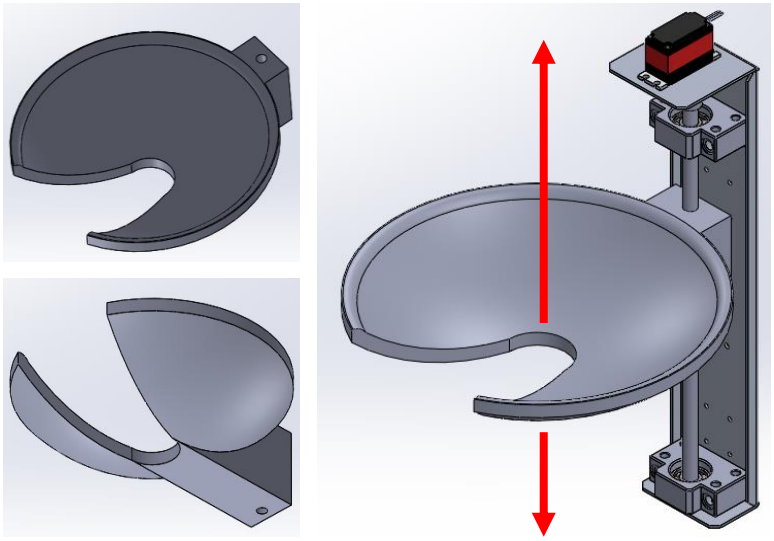


Figure 22 Bowl Shape Gripper Part

	Major issues with the Design Ver.1	**Solution/Modification**
1.	Complicated gripper mechanism (synchronization of gripper in both vertical and horizontal motion)	Eliminate horizontal motion, use single gripper part instead of having two separated parts.

To install a large container for the ball storage purpose, students relocated the Module 1 - upper body mounting position to the front of the robot. With enough space behind the upper chassis, a container, which is large enough to accommodate 2 big balls or 16 small balls, was mounted as shown below figure. With this modification, the robot was not only capable of storing more balls that can finish most of the job within one journey but also had a stronger chassis structure by having the container to assist as the major connection part of Module 2 - Lower body.

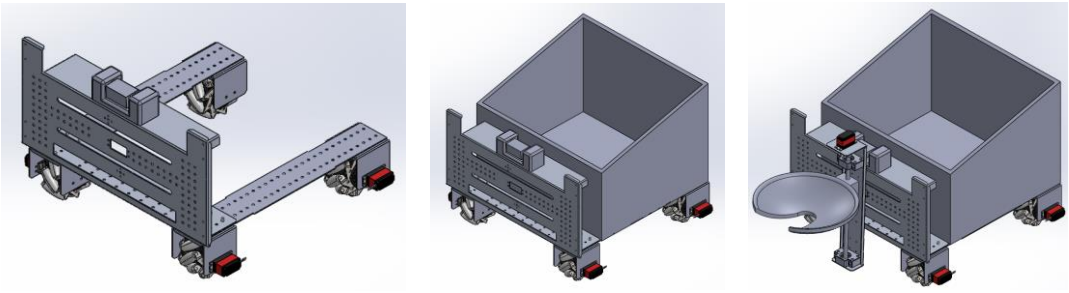


Figure 23 Modified Structure and Components

	Major issues with the Design Ver.1	**Solution/Modification**
2.	Not enough space to install the storage box (ball container)	Relocate the mounting position of the upper part.
3.	Weak lower body/chassis may lead to inaccurate locomotion due to misalignment of mcanum wheels	Additional mounting of the huge container assist as a connection bridge to hold the lower parts tightly.

Then, to learn about the material property of the aluminum sheet, students used SolidWorks 3D model to build a sheet sample and compared its simulated stress caused by the gravitational force of upper body to its yield strength.

Considering the weight of the lead screw system, upper chassis, container, battery, collected ball, microcontroller, and other electrical parts, students estimated the robot's center of gravity as shown below calculation and illustration:

All specification including weight and distance is reasonably estimated based on the researched standard parts and 3D model prototype.

Parts	Unit Weight (kg)	Quantity (Unit)	Total Weight (kg)
Lead Screw System	8	1	8
Upper Chassis (Aluminum Sheet)	1	1	1
Lower Chassis (Aluminum Sheet)	2	1	2
Container	1	1	1
Battery	1	2	2
Collected Ball (Overall)	1	1	1
Electrical Parts	1	1	1
			14.5

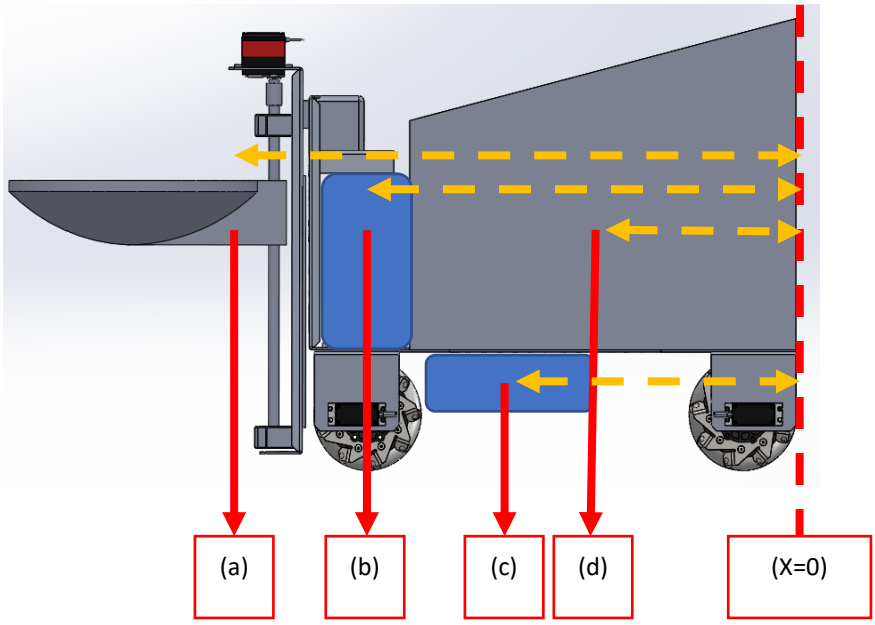


Figure 24 Estimated Weight Distribution

The estimated center of gravity can be calculated as shown below:

Parts	Weight (kg)	Gravitational Force (N)	Distance, X (mm)
(a) Lead Screw System + Upper Chassis	9	88.2	340
(b) Electrical Parts	1	9.8	280
(c) Battery	1.5	14.7	190
(d) Container + Collected Ball (Overall) + Lower Chassis	4	39.2	130

$$\sum m_i g_i = (88.2 + 9.8 + 14.7 + 39.2) = 151.9N$$

$$x_{cg} = \frac{\sum m_i g_i x_i}{\sum m_i g_i}$$

$$= \frac{(88.2 \times 340) + (9.8 \times 280) + (14.7 \times 190) + (39.2 \times 130)}{151.9} = 267.4mm$$

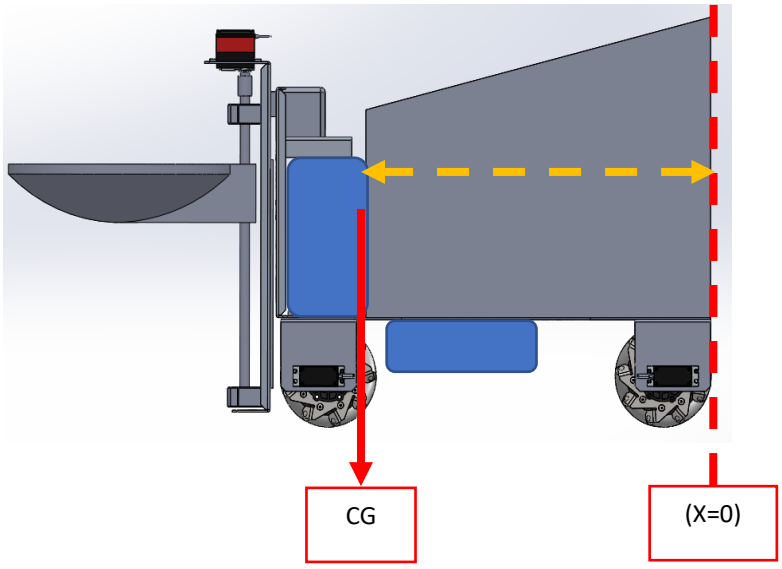


Figure 25 Estimated Center of Gravity

With the estimated result from the calculation, students did simple loading analysis on one of the essential lower chassis parts to check if there is any possibility of bending.

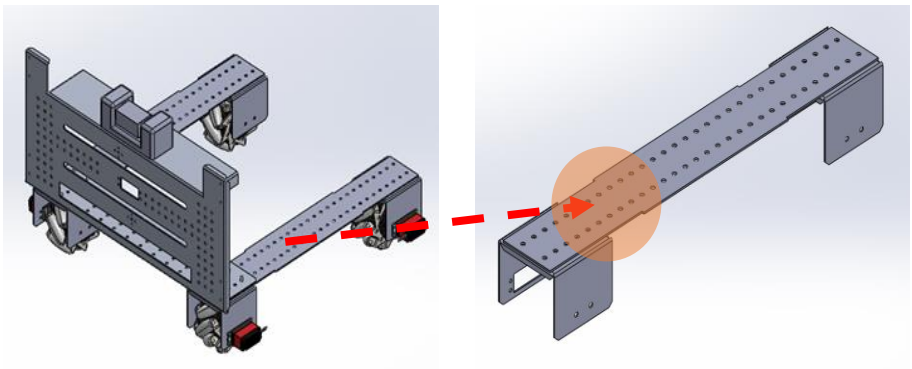


Figure 26 Loading Position

This workpiece thickness was set to 2mm, and the loading force was set to 75N, which is half of the overall weight estimated. Assuming that it is fairly distributed around the location that students calculated.

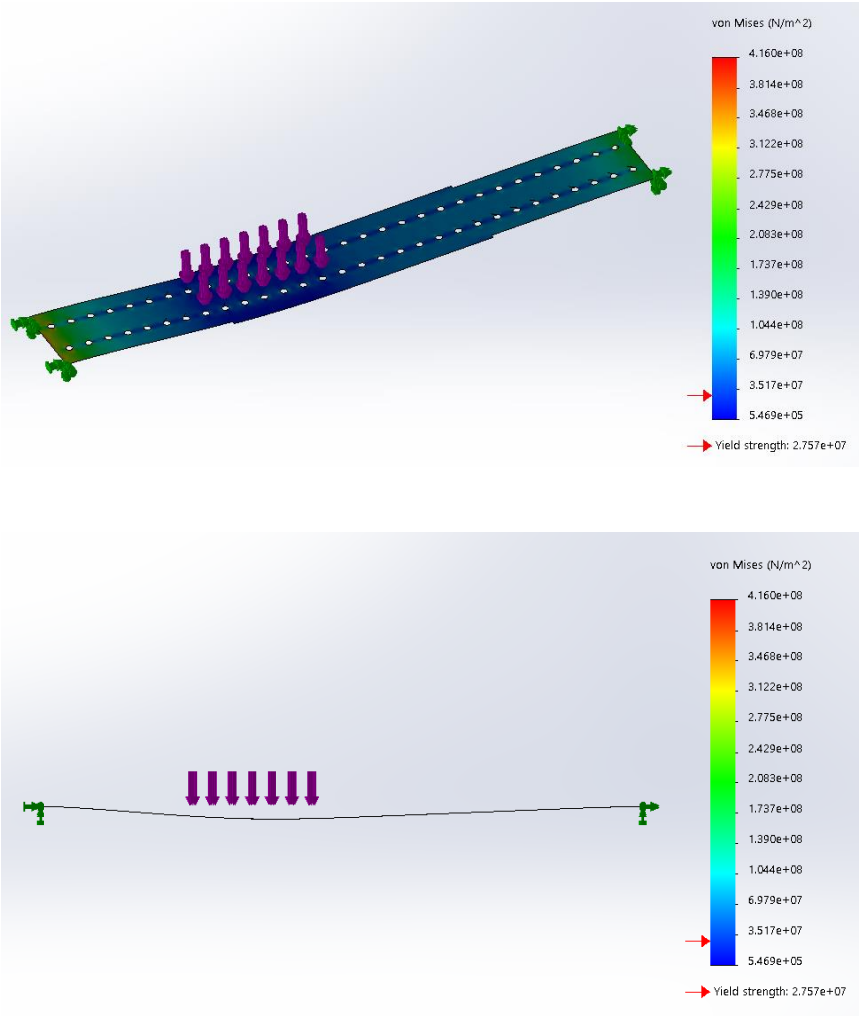


Figure 27 SolidWorks Compression Simulation Study

Comparing to the yielding strength of 2.757e+07, the developed strength almost reached 3.517e+07. This result clearly showed that there is potential risk of bending over time. Furthermore, the robot's inevitable vibration during operation or unexpected collision with opponent may lead to even worse scenario.

The team at first thought of simply increasing the thickness of sheet, and even adding some bended feature to prevent bending. Even though it may solve the problem, but it was not an optimum solution to deal with root cause. Instead students considered aluminum profile as one of the options, which was researched in Section 2.4 ROARY (Sweden University).

To compare the strength of aluminum profile to 3mm sheet, the team has conducted 3-point-bending test with both sample workpieces under below assumption and condition.

1. Separation of support point was set to 6cm for both work piece.
2. Test speed, which is the speed of compression, was set to 1mm/min.
3. Approximate pre-loading was set to 30N to ensure full contact with the tool.

Figure 28 Workpieces & Supporting Tool

The sample workpieces were scrap material, and the aluminum profile was 20*20 standard size with X-shape structure, while aluminum sheet thickness was 3mm with 10cm width. The length difference between the two workpieces was meaningless because the supporting separation was set to 6cm. The low speed of compression loading is to collect detailed data and the pre-loading of 30N is to make sure the compression tool fully touches the workpiece before starting the testing. Their individual result data was not important, and in fact, these values, especially the compression, highly depends on the supporting separation. For examples, the compression result may be greater or faster if the supporting separation is set to 15cm instead of 6cm. Thus, in this experiment, the main purpose was on the comparison of two workpieces' property, so that the team can learn about more suitable material for the robot chassis.

The procedure of the experiment is as listed below:

1. Replace the grasping tools from the tensile test machine to supporting tool and compression tool
2. Align two parts in parallel and horizontal, and screw tightly to prevent slip or turn
3. Turn on the program and set the test method to ‘MTS Simplified Compression’
4. Lower down the compression part to touch the workpiece, and slightly increase the load to ensure full contact between the tool and workpiece, then set the compression to 0mm.
5. Set the condition, such as loading speed of 1mm/min, and start experiment
6. Stop when the loading exceeds 500N.

Even though this machine has ability of testing up to 25,000N loading force, because the only contributor of this potential bending in the real case is purely the weight of robot which is only around 15kg, so the data was collected up to 500N which is double of the estimated robot’s weight. Below is the graph plotted with collected data of Loading and Compression.

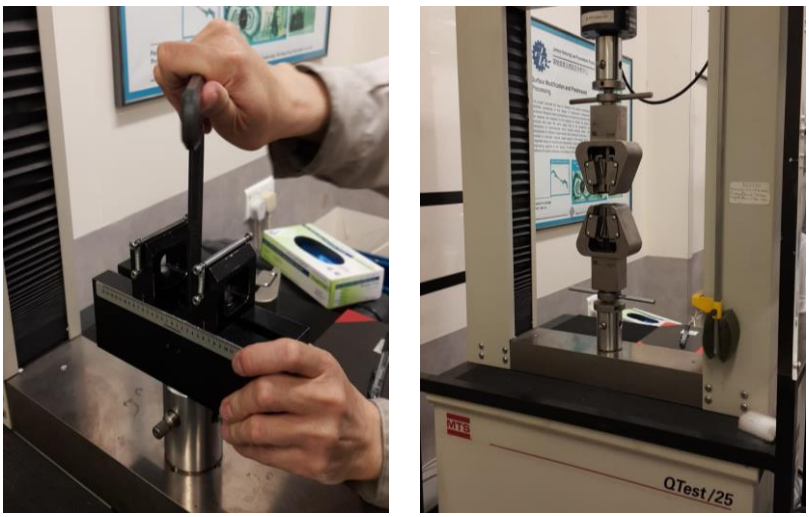


Figure 29 Bending Test Preparation

The graph is plotted with the data collected from 30 N to 300N, and it clearly shows the significant difference in compression resistance. For the aluminum sheet test, the compression at the 300N loading was around 1mm, while the profile at the same loading only be compressed by 0.05mm. Thus, it is assumed that aluminum profile has almost no compression with 300N and could withstand much larger loading.

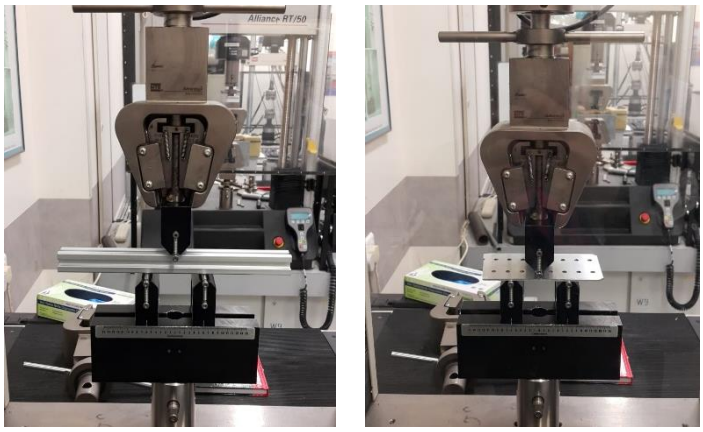
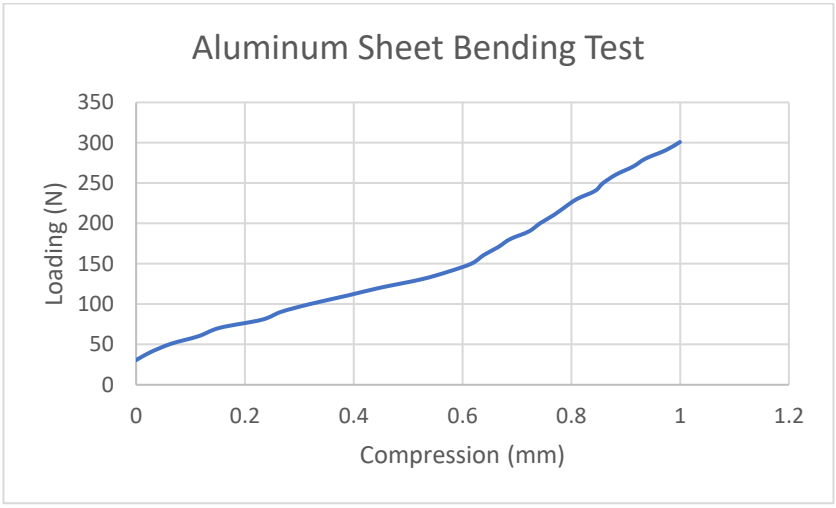
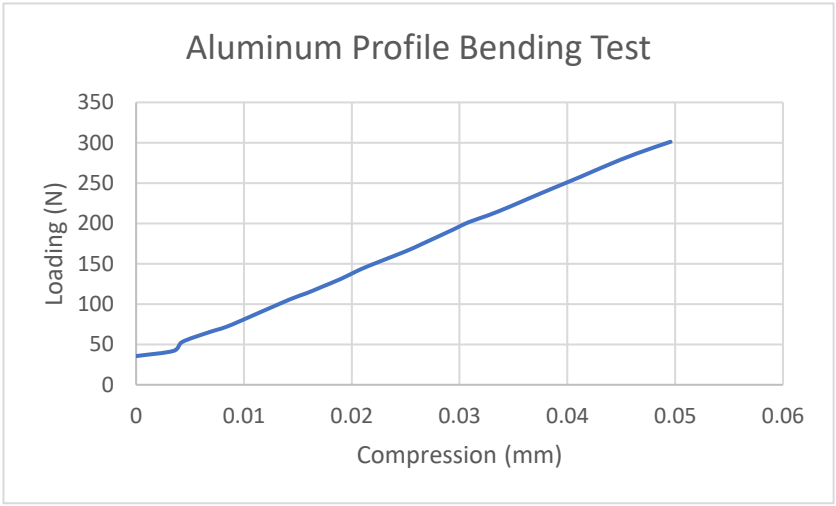


Figure 30 Bending Test - Aluminum Profile & Sheet





In the real scenario of this project, the robot's weight was estimated to be around 15kg, and it was reasonable to assume that this much force will not introduce any failure or even bending to profile due to the structural advantages. Hence, students decided to redesign the robot chassis replacing all aluminum sheet to profile.

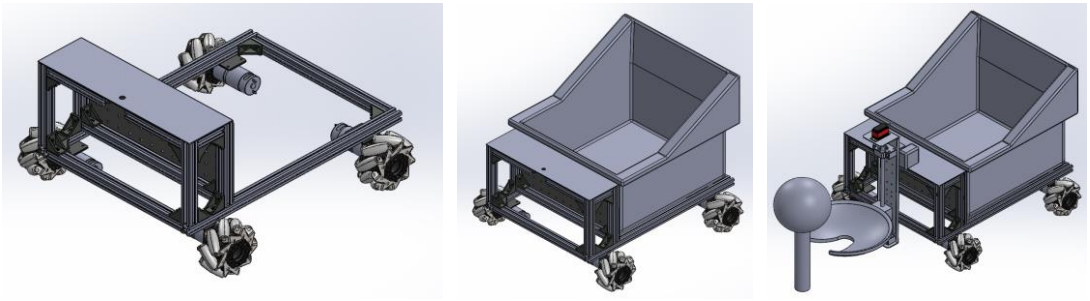


Figure 31 Modified Robot Chassis

	Major issues with the Design Ver.1	**Solution/Modification**
4.	Aluminum sheet metal tends to bend over time due to weight affecting overall functionality of the robot.	Redesign the chassis with aluminum profile.

Overall summary of the problems faced in the previous design and its solution is as shown below:

	Major issues with the Design Ver.1	**Solution/Modification**
1.	Complicated gripper mechanism (synchronization of gripper in both vertical and horizontal motion)	Eliminate horizontal motion, use single gripper part instead of having two separated parts.
2.	Not enough space to install the storage box (ball container)	Eliminate horizontal motion, use single gripper part instead of having two separated parts.
3.	Weak lower body/chassis may lead to inaccurate locomotion due to misalignment of mcanum wheels	Eliminate horizontal motion, use single gripper part instead of having two separated parts.
4.	Aluminum sheet metal tends to bend over time due to weight affecting overall functionality of the robot.	Redesign the chassis with aluminum profile.

3.4.2 Flipping Motion Development

To store the collected balls, the robot required more than just one vertical motion from lead screw system, because it can only lift up the ball but not placing it into the container. It was necessary to add one more mechanism which flips the ball into the box where this idea was based on the dump cart research done in the literature review.

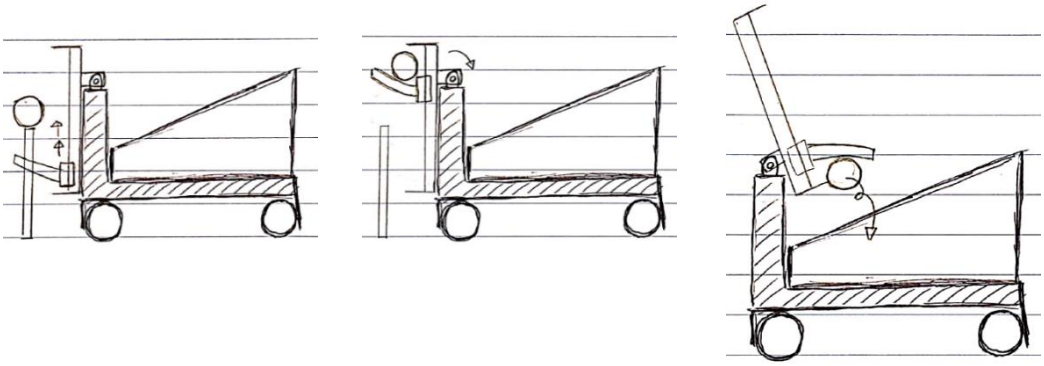


Figure 32 Primary Flipping Motion Brainstorming and Sketching

The methodology of collecting balls for Liter Ver.2 is as shown below:

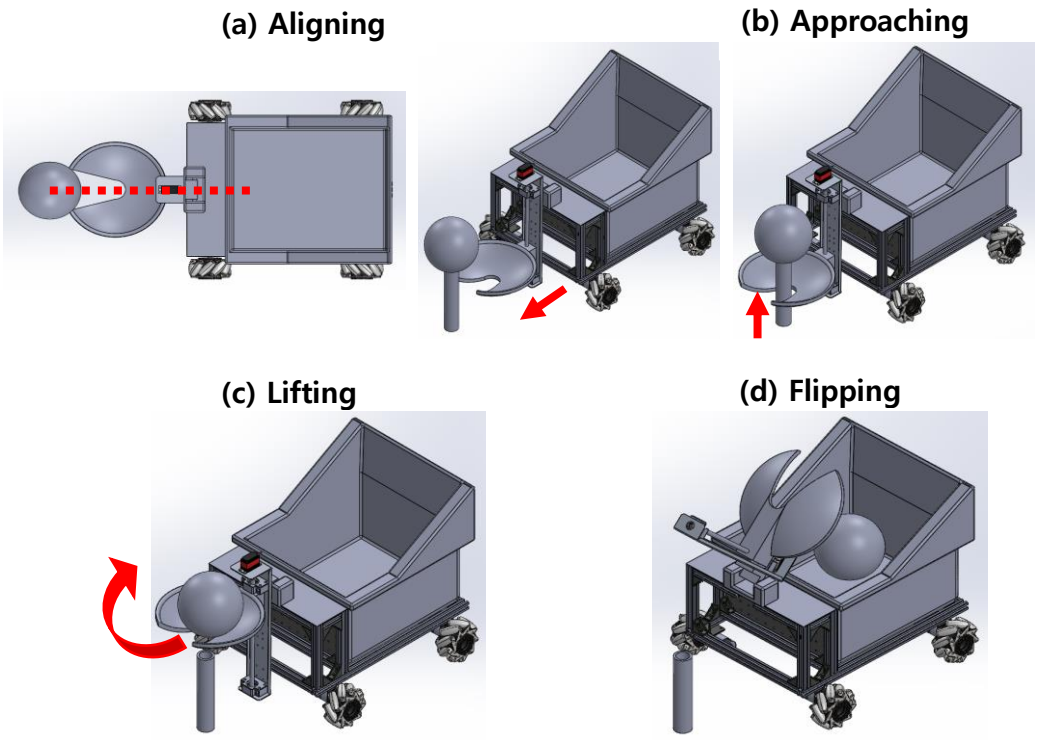


Figure 33 Design Ver.2 Methodology

In order to successfully flip the lead screw system with the collected ball, the most important specification that students needed to clarify was torque. To estimate the torque requirement, the team used previously researched information about the parts weight and made reasonable assumption for the calculation. As the weight of the lead screw system is around 4kg, and the weight of the basketball is around 0.65kg, the rest of the required information is assumed as follow:

1. All other necessary parts that may add more extra weight on lead screw system such as the connection to the flipping motor are estimated to weight 1kg.
2. The weight of the end-effecter, which students considered to 3D print with ABS material, could be estimated by SolidWorks, and it's around 0.62kg.
3. No friction is considered for simplify the calculation.
4. Only the starting torque is considered but not the holding torque, because it is the most extreme case that may overload the flipping motor.

	Weight (kg)	Gravitational Force (N)	Distance, X (m)
(a) Lead Screw System + Connection	4.5	44.1	0.06
(b) Gripper	0.62	0.62	0.16
(c) Basketball	0.65	6.37	0.18

To calculate the starting torque, τ_{req}

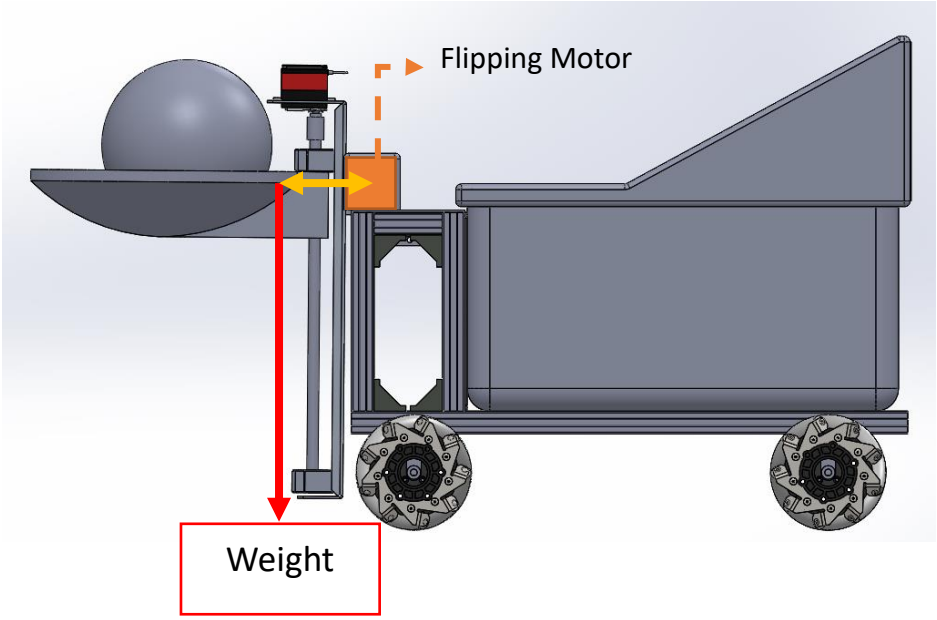


Figure 34 Required Flipping Torque Analysis

$$\tau_{req} = \sum m_i g_i x_i = (44.1 \times 0.06) + (0.62 \times 0.16) + (6.37 \times 0.18)$$

$$\tau_{req} = 3.89Nm$$

Considering safety factor of 2 to compensate some friction and efficiency loss:

$$\boldsymbol{\tau_{req} = 3.89(2) = 7.78Nm = 76.32kgcm}$$

Thus, students could roughly understand the lower boundary of the required torque for flipping motion in the case of collecting basketball.

3.4.3 Design Ver.2 Evaluation

The team evaluated the final design of the Design Ver.2, and again listed quite a few issues that must be solved and improved. First, the center of gravity of the robot itself without any ball stored was beyond the front two wheels as shown below

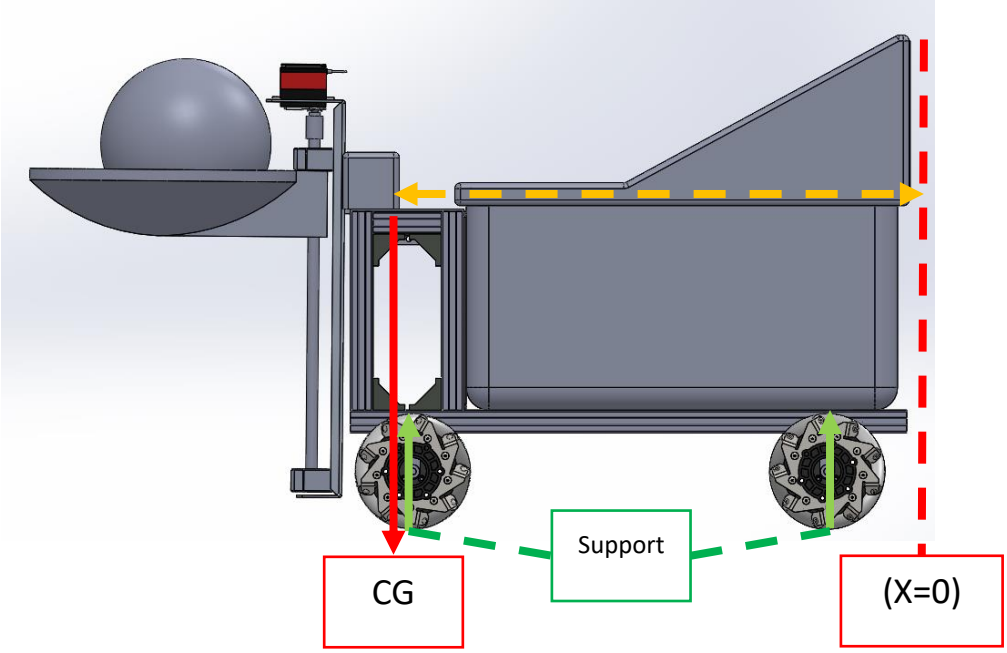


Figure 35 Robot Center of Gravity Study

$$x_{cg} = \frac{\sum m_i g_i x_i}{\sum m_i g_i} = \frac{(88.2 \times 340) + (9.8 \times 280) + (14.7 \times 190)}{112.7} = 315.22mm$$

This clearly shows that its heavy front module will make the front wheel acting as the tipping point, thus the robot will fall forward. Second, the gripper could not grasp or hold the ball tightly during flipping motion, and so it is possible that the ball, especially those small balls, will not go into the container but rather slip and fall to the ground.

Lastly, the estimated torque required for the flipping mechanism was too high that a really strong motor shall be equipped.

Below is summary of the problems that students evaluated from Design Ver.2:

	Major issues with the Design Ver.2	**Solution/Modification**
1.	Robot may fall forward due to heavy front module.	
2.	Gripper cannot hold the ball tightly and thus loss control of it during flipping.	
3.	Flipping mechanism requires extremely heavy torque.	

3.5 Design Ver.3 Development – Iteration 3

Again, based on the evaluation and results collected from the previous design, students tried their third iteration of brainstorming for the modification to solve all those critical issues and to improve its performance. Ultimately, the team finalized their Design Ver.3.

3.5.1 Design Ver.2 Problems Analysis and Modification

Analyzing the summarized problem list, the team realized that they were actually all sharing one common root cause, which was module 3 – End-effector parts. As summarized below, the main reason behind all these issues was due to the heavy weight of Module 3 and the gripper with a low universal level. The heavy weight of Module 3 is the major contributor of the robot falling forward, and also greatly increased the torque requirement for flipping that the motor may be overloaded easily. The gripper has very high potential to lose control of the ball during the flipping motion or even when it vibrates as it is not tightly grasping the object, and it led to ineffective grasping and flipping mechanism. Thus, students mainly concentrated on brainstorming the new Module 3 – End-effector design, where this time they did not consider having the standard lead screw system because it was almost more than half of the overall module 3 weight, and it also restricted implementing creative design ideas.

	Major issues with the Design Ver.2	**Reason**
1.	Robot may fall forward due to heavy front module.	Heavy Module 3 – End-effector part is the major contributor.
2.	Gripper cannot hold the ball tightly and thus loss control of it during flipping.	Module 3 – End-effector part not having universal gripper.
3.	Flipping mechanism requires extremely heavy torque.	Heavy Module 3 – End-effector part again is the major or even sole contributor.

3.5.1.1 Racks-and-Pinion System Development

As the standardized vertical motion element was excluded, considering the symmetrical movement of two fingers and wide grasping range of up to 250mm, students opt not to adopt angle gripper with toggle lever mechanics or parallel gripper with cam disk (Appendix V), which were researched previously in Design Ver.1. Instead, the team studied on a simple rack and pinion system that can provide synchronized motion of the fingers as well as introducing a wider range of workspace. This system also has an advantage of easier control and manipulation over other systems including a lead/ball screw and pneumatic linear actuation.

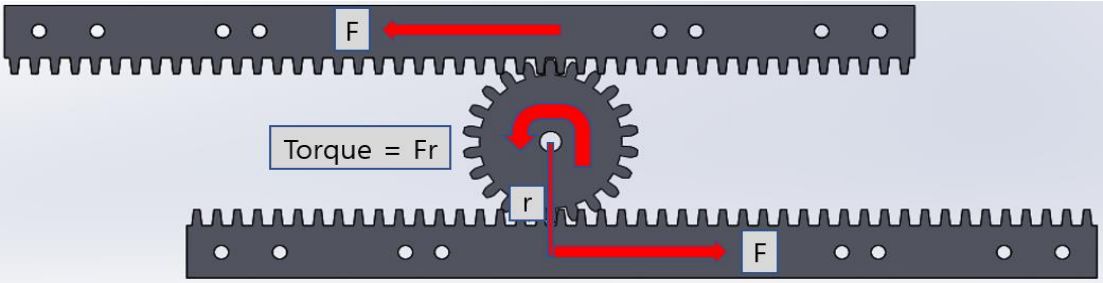


Figure 36 Rack and Pinion Gripper Mechanism

This customized racks-and-pinion allowed the perfectly synchronized horizontal movement of the gripper parks, which is fixed at the height of 20cm from the ground, and it does not allow any chance of misalignment as they are gear assembled.

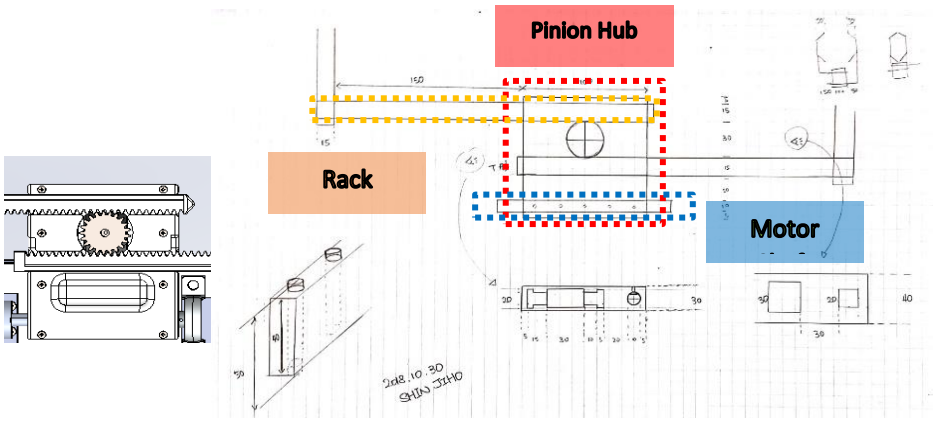


Figure 37 Racks-and-Pinion Brainstorming and Sketch

Referring to the last figure, the pinion hub could be directly connected to the motor shaft which enables it to flip around 160 degrees, so there will be no extra connector parts required. Assuming that the part is 3D printed, which actually is the only way of manufacturing due to its customized casing, it is expected to significantly reduce the overall weight of Module 3 and solve the majority of problems that students faced in Design Ver.2 which was caused by the heavyweight. However, a better universal gripper yet had to be brainstormed and developed. Thus, the team further researched on developing new gripper parts in Module 3 which can be easily mounted on the rack arm for horizontal motion and has universal shaped object grasping capability.

3.5.1.2 End-effector Development and Analysis

Through developing racks-and-pinion system instead of implementing the standard lead screw system, it significantly reduced the weight and led to acceptable torque requirement for the flipping. However, it indeed did not have the full control over handling the ball during flipping motion. There was still some possibility of losing the ball, and thus the team needed a new gripper which could tightly hold the ball all the time until it is finally released into the container. Hence, the team has researched about universal gripper development in robotic industry, particularly about soft robotics which grasps objects of various shapes.

Soft robotic gripper control mechanisms can be categorized into three groups: i) by simple actuation, ii) by controlled stiffness of material, and iii) by controlled interface mechanical adhesion or electro adhesion. The majority of modern soft gripping devices are indeed a combination of those mentioned, proving that this technology is promising.

According to 'Soft Robotic Grippers' research data (Figure 38) from EPFL – Ecole Polytechnique Federale de Lausanne [27], a university in Switzerland specialized in natural sciences and engineering, spherical objects are the simplest shape to be grasped among other complex ones regardless of the control technology integrated. At the same time, since mechanical properties of the object (e.g., fragility) as well as the operating environment (e.g., dry, wet) are not a concern, soft grippers are very much idealistic for the purpose of ASME competition at least in terms of functionality in collecting diverse sizes of balls.

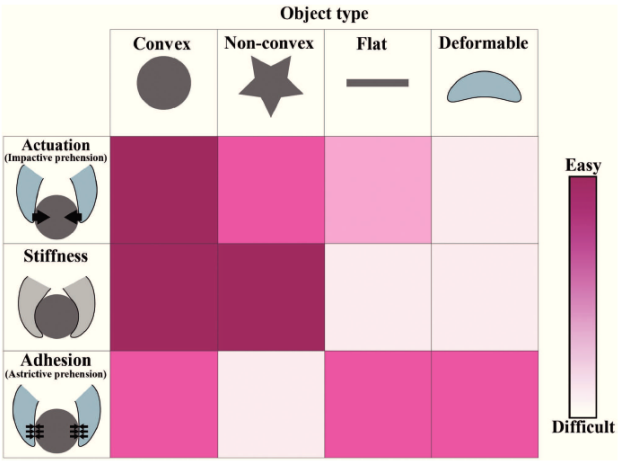


Figure 38 Different gripping technologies of soft grippers and object types

One of the typical universal soft robotic grippers, called jamming gripper as shown in Figure 39 [28], focuses mainly on control of interface mechanical adhesion. In fact, it is a simple passive gripper that can be easily built. It consists of granular material, for instance, sand and coffee grains, encased in an elastic membrane; via manipulating pressure input, the suction force generated will grip or release the objects. This device shows a strong competency in shape transformation of elastic membrane to match various complex object shapes. That means, with higher pressure input, the elastic membrane will be deformed allowing more grains to be displaced around the object, such that larger or heavier objects can be grasped relatively easier as compared to the capability of other rigid robotic grippers.

However, as seen in Figure 40 [28], where the blue line indicates normal jamming gripper with only negative pressure input for gripping, and the red line indicates results for an upgraded version with both, negative and positive pressure inputs for gripping and releasing, the success rate of grasping falls dramatically as the radius of object approaches about 85% of the gripper radius. Also, pressure force applied increases proportionally as the object size increases leading to strong input requirement.

Figure 39 Jamming Gripper

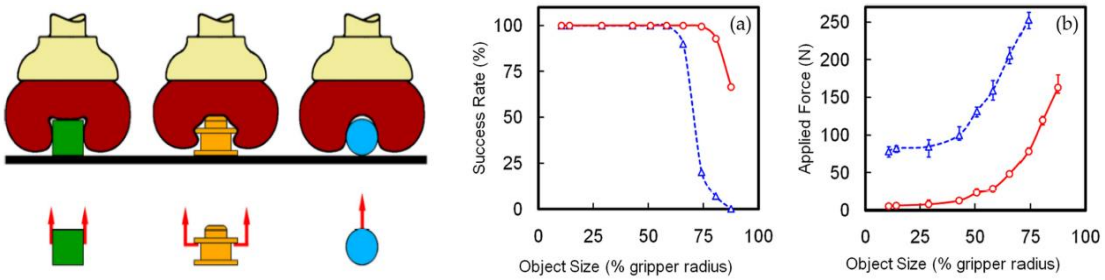


Figure 40 Jamming Gripper Grasping Ability

Since it is not favorable to have pneumatic actuation system in the robot due to its complexity and difficulty in precise control, the team has brainstormed about utilizing the elastic material, such as yoga band, and actuating by simple motion element which is the racks-and-pinion system.

With this background research done, the team was able to brainstorm of mounting yoga bands parallel to each other on the gripper and actuating the horizontal open and close motion by motor-driven racks-and-pinion system to allow the deformation of its shape according to the ball sizes. Though, its friction must be tested with trial-and-error method in later

Figure 41 Resistance Band

stages.

To achieve this, the team first designed the gripper skeleton where the yoga band should be mounted.

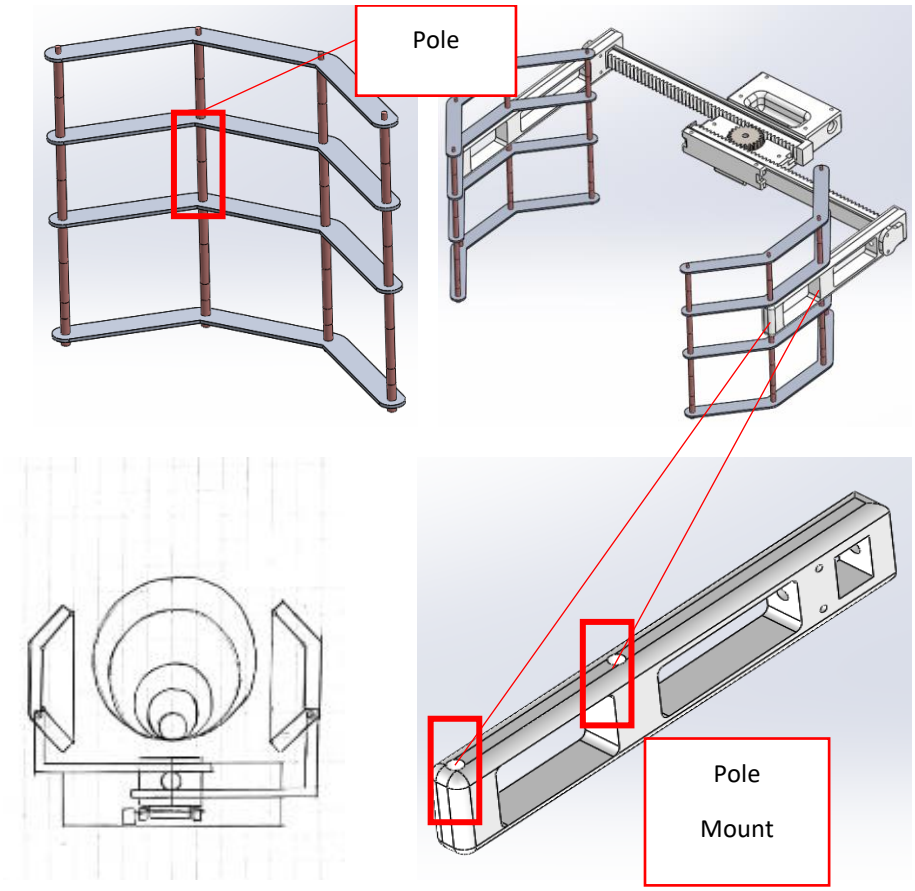


Figure 42 Design Ver.3 Gripper Skeleton

This unique design of gripper skeleton has two major advantages. First, the pole introduced an easier mounting method on the rack arm where the team do not need extra connector component for assembly. It not only reduced the complexity in design, but also prevented adding extra weight on Module 3. The second advantage is the yoga band, that the team is going to use as the elastic membrane for universal grasping, can be mounted on the pole directly, again without any extra connecter.

To understand and analyze how the force is acting on these parts, team first drew free body diagram with simple 3D model considering the extreme case of carrying a basketball, and calculated estimated force acting on the rack arm with assuming that the ball is directly contacting the ABS material.

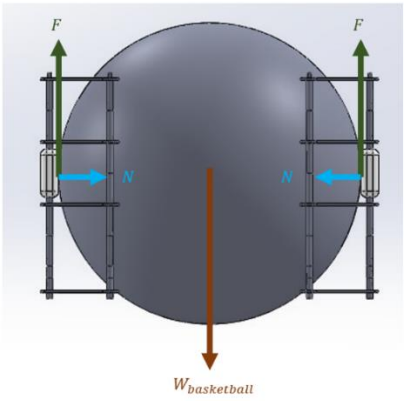


Figure 43 Free Body Diagram

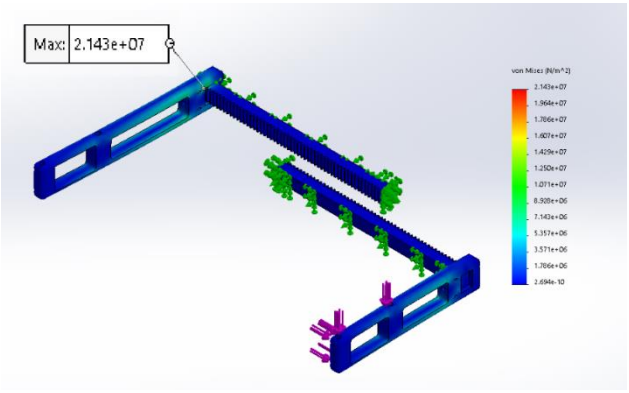


Figure 44 Stress analysis of ABS gripper

$$\sum F = F + F - W_{basketball} = 0$$

$$\therefore \boldsymbol{F = \frac{0.65(9.81)}{2} = 3.188\ N}$$

Refer to Figure 43 and according to mechanical properties of ABS [29],

assuming $\mu_s = 0.19$

$$F = \mu_s N$$

$$\therefore \boldsymbol{N = \frac{3.188}{0.19} = 16.78\ N} \qquad (1)$$

The maximum stress experienced is still lower than the yield strength of ABS, which is 27 MPa, therefore the **arm will not break**. However, more evaluation on fillet features and designs could be carried out for minimizing the stress concentration.

To improve the protection to bending and fracture at the edge, the stress concentrator was eliminated by adding smooth rib or fillet appropriately. Also, by splitting the rack and the arm parts as shown in the below figure, this unique cap design assisted the prevention of bending by pressing from the other side. Furthermore, with the screws connected through the rack and arm, it worked as the core material which improved the overall material property.

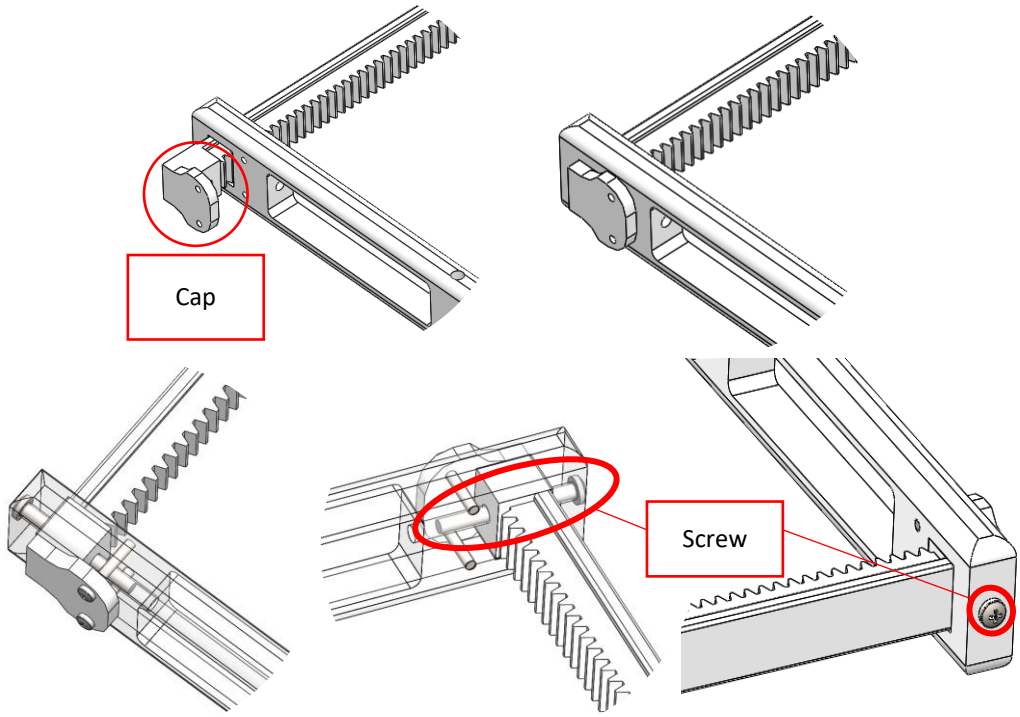


Figure 45 Improving Bending and Failure Protection

3.5.1.2 Flipping Mechanism Development and Analysis

To analyze flipping torque required to choose the right motor, the following assumptions are made to allow simple calculation and analysis:

1. The mass of the greatest payload, which is basketball is considered to validate maximum torque required.

2. The mass of Module 3 including the motor is considered to acquire a practical result.

3. The friction is not considered as the shaft and bearing is appropriately integrated.

4. A safety factor of 2 is included because vibration or shaking of the robot during flipping motion may instantly increase the payload experienced on Module 3.

Below is the flipping torque calculation and its analysis:

	Weight (kg)	Gravitational Force (N)	Distance, X (m)
(a) Gripper + Motor	0.792 + 0.2	9.73	0.1059
(b) Basketball	0.65	6.37	0.2278

Based on the measurements obtained from SolidWorks, the moment arm between the rotational axis and the forces could be calculated.

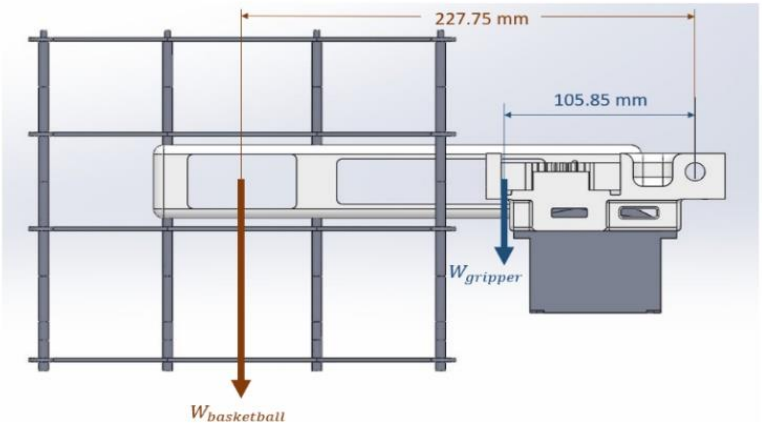


Figure 46 FBD and geometry of forces of gripper

By applying Newton's Law,

$$\tau_{req} = \sum m_i g_i x_i = 6.37(0.2278) + 9.73(0.1059)$$

$$\tau_{req} = 2.482 \text{ Nm}$$

Considering safety factor of 2 to compensate some friction and efficiency loss:

$$\boldsymbol{\tau_{req} = 2.627(2) = 4.963Nm = 48.69kgcm} \qquad (2)$$

Comparing to the torque in Design Ver.2 which its required flipping torque was around 7.78Nm, the torque required in Design Ver.3 was reduced by around 40%, which ultimately reduces the power consumption.The methodology of collecting the ball is demonstrated as below:

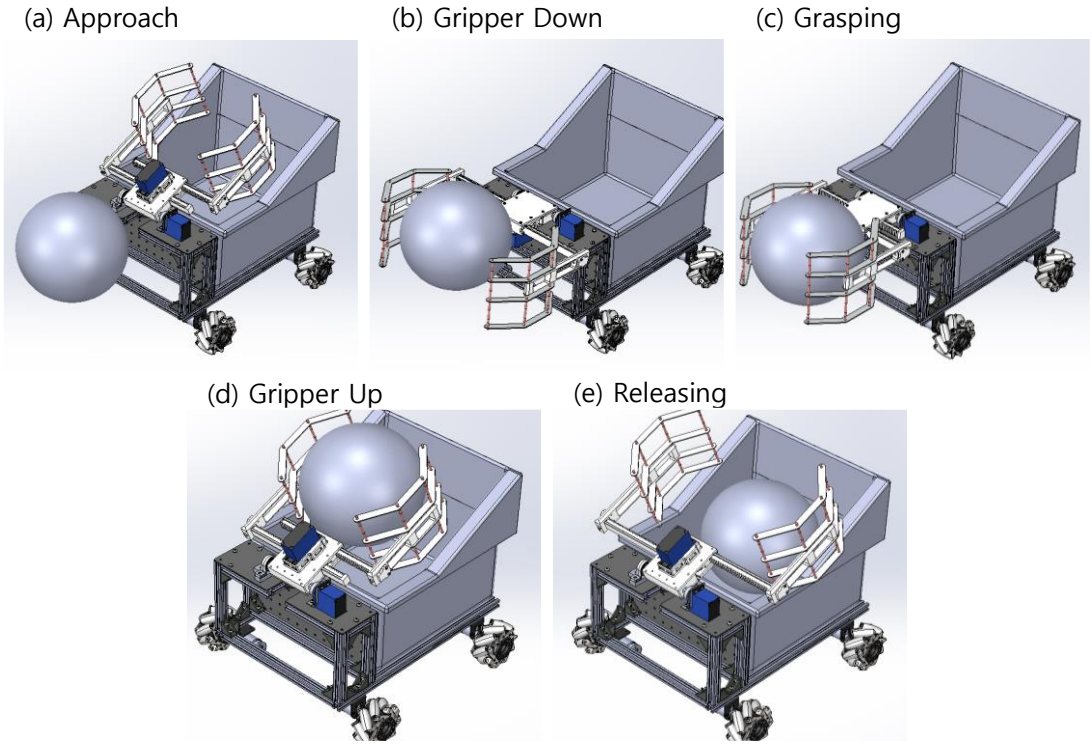


Figure 47 Design Ver.3 Methodology

The summarized solution to previous issues in Design Ver.2 is as shown below:

	Major issues with the Design Ver.2	Reason
1.	Robot may fall forward due to heavy front module.	Getting rid of heavy lead screw system but instead introduced 3D printed rack and pinion end-effector parts.
2.	Gripper cannot hold the ball tightly and thus loss control of it during flipping.	Soft robotics – use of yoga band to tightly grasp different sizes of balls.
3.	Flipping mechanism requires extremely heavy torque.	3D printed with ABS material and the skeleton structure of gripper parts significantly lower down the overall weight.

3.5.2 Unloading Mechanism Development and Analysis

Design Ver.3 allowed a simpler mechanism of collecting multiple balls comparing to the previous designs, as well as storing and transporting method for completing the mission. However, it was not equipped with unloading mechanism at Module 2 – upper body, where the collected balls could not be deployed to the ground. At first, the team had an idea of adopting some mechanism studied in dump cart literature review with an actuator lifting the container at certain gradient to let the balls roll down along the slope.

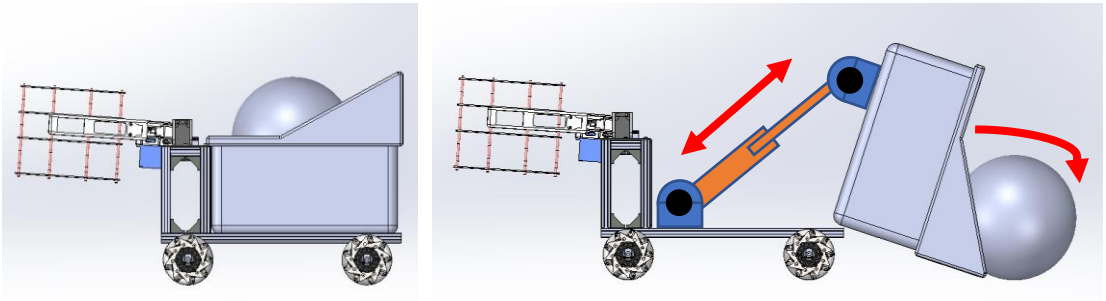


Figure 48 Linear Actuation Mechanism

However, this lifting system required a large space at the bottom of the container that may result in allowing less space to store the ball, and at the same time, this additional part may increase the overall weight of the robot affecting its performance such as lower speed. Most importantly, the robot had to make sure the ball is released and at rest inside the starting zone, which means the ball shall not roll out the restricted area after unloading.

In case of this mechanism, the unloaded ball has high potential to continue rolling and that was not acceptable in this competition. Hence, students could not directly adopt the dumping cart mechanism but was able to design a new mechanism with the idea learned from this brainstorming.

Instead of lifting the container with an actuator, the team decided to customize the container with a sloped bottom plate as shown in below figure.

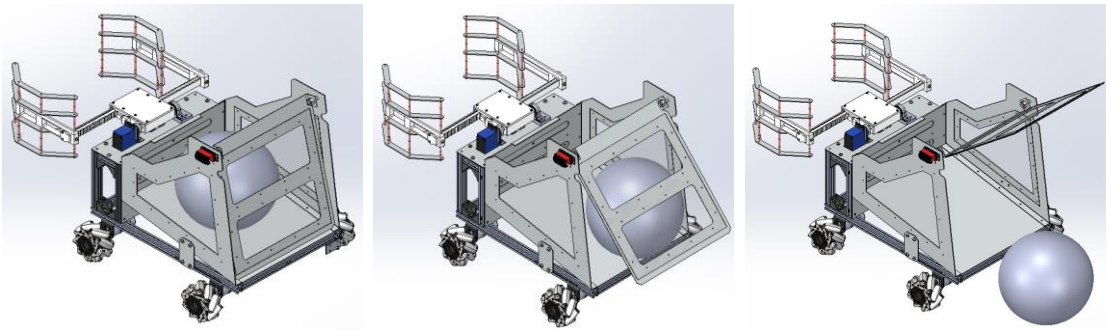


Figure 49 Unloading Mechanism

Even though the team had to give up a little space underneath the container to introduce some gradient at the bottom plate, it was very efficient idea because lifting the container itself would be much complicated and accommodate much more space. To keep the ball inside the container, students had to analyze the gravitational force of the balls acting on the gate and compare with the holding torque specification of the motor to ensure the gate will not be accidentally opened due to a heavy payload.

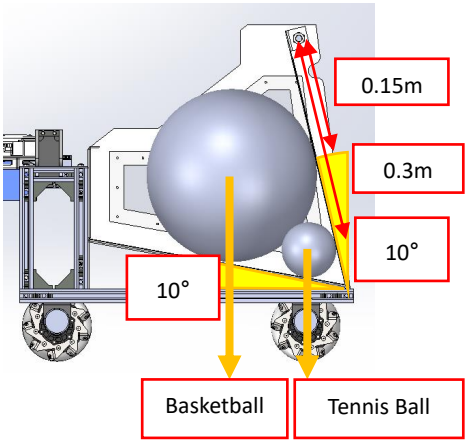


Figure 50 Design Ver.3 Cut View

Again, the team did not considered friction but only the weight of the ball for simple calculation and quick analysis. First, the free body diagram of the ball is as shown below:

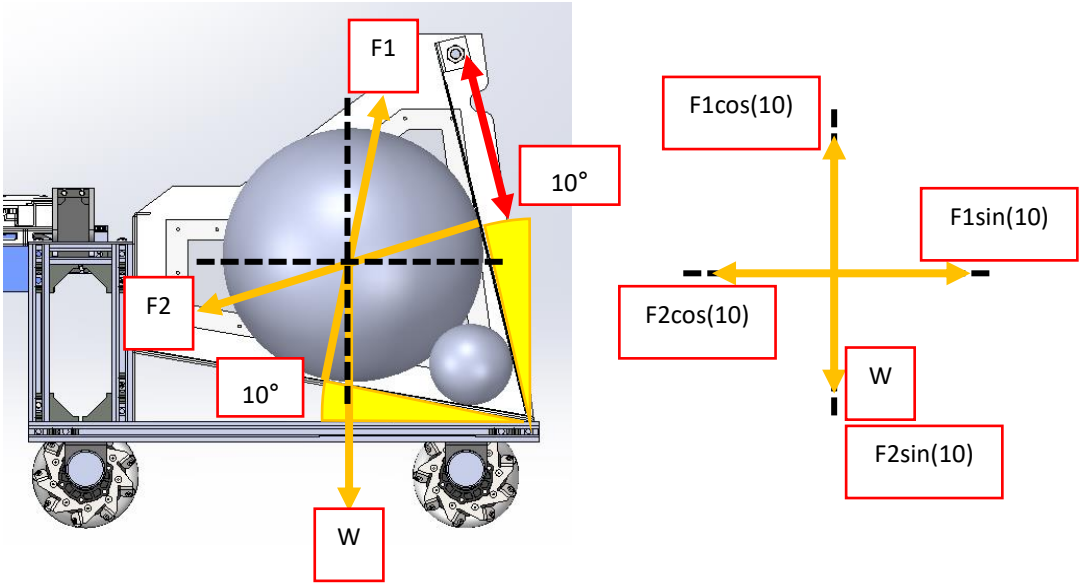


Figure 51 Basketball Free Body Diagram

$(\rightarrow)\ \mathrm{X-axis:}\ F_1 \sin(10) - F_2 \cos(10) = 0$

$(\uparrow)\ \mathrm{Y-axis:}\ F_1 \cos(10) - W - F_2 \sin(10) = 0$

$\because W = 0.65(9.81) = 6.3765\mathrm{N}$

$\therefore F_1 = 6.683\mathrm{N}$

$\therefore F_2 = 1.178\mathrm{N}$ *(3)*

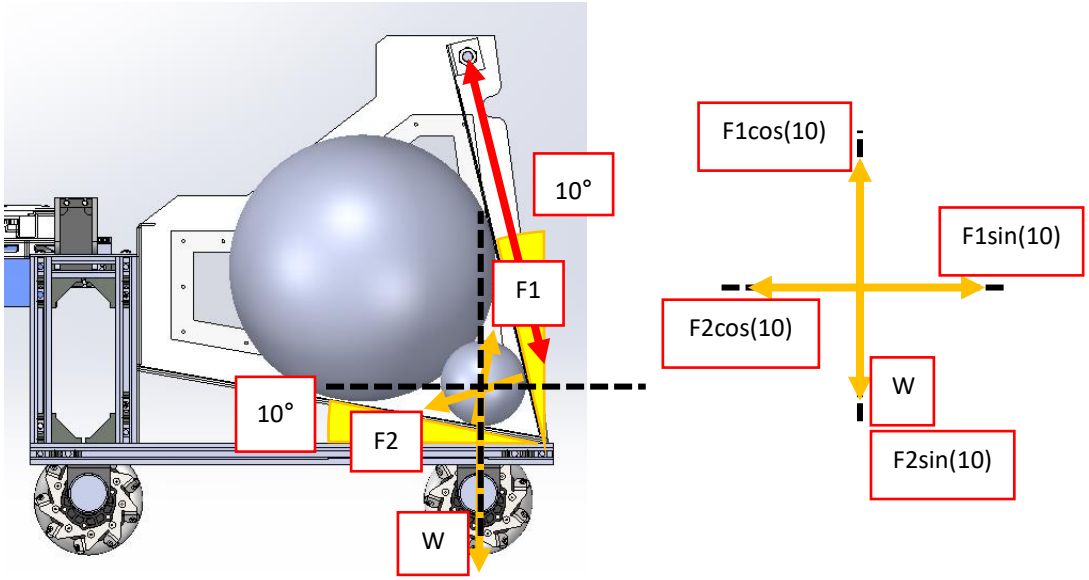


Figure 52 Tennis Ball Free Body Diagram

$(\rightarrow)\ \mathrm{X}-\text{axis: } \mathrm{F}_1 \sin(10) - \mathrm{F}_2 \cos(10) = 0$

$(\uparrow)\ \mathrm{Y}-\text{axis: } \mathrm{F}_1 \cos(10) - \mathrm{W} - \mathrm{F}_2 \sin(10) = 0$

$\because \mathrm{W} = 0.05(9.81) = 0.4905\mathrm{N}$

$\therefore \mathrm{F}_1 = 0.514\mathrm{N}$

$\therefore \mathrm{F}_2 = 0.09064\mathrm{N}$ *(4)*

Assuming there are 4 basketballs, 8 tennis balls and 4 ping pong balls on the playground, where the scenario is that the different sizes of balls are evenly distributed, the robot is able to complete the mission in two round-trips:

1. Collecting 8 tennis balls, 4 ping pong and 2 basketball

   (1 basketball is assumed to be held on gripper during unloading)

2. Collecting 2 basketballs

   (2 basketballs are all in the container)

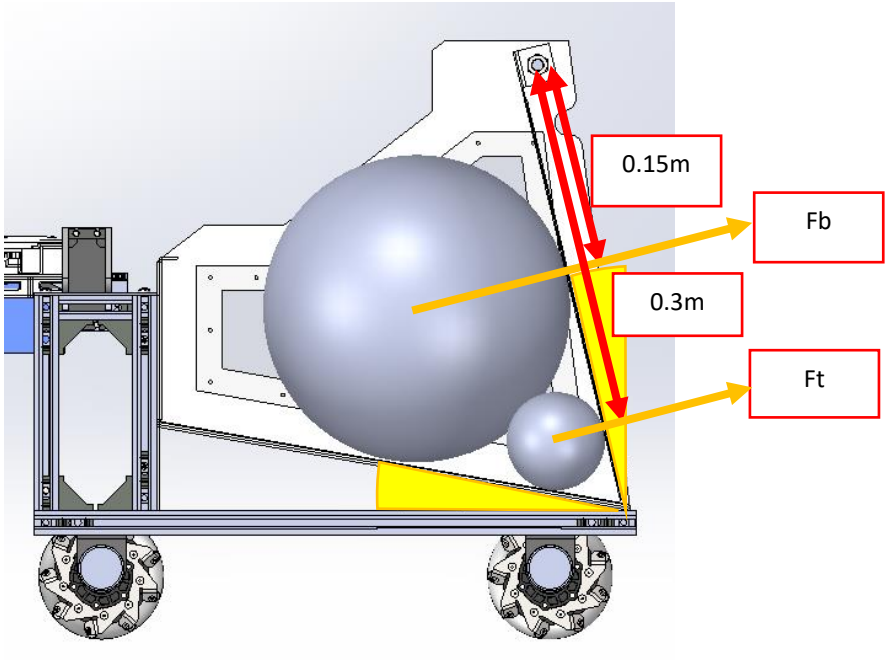


Figure 53 Container Free Body Diagram

To simplify the calculation, 8 tennis balls are assumed to contribute on 'Ft' together at the same location.

$(Clockwise)\ \tau_{req} - F_b(0.15) - F_t(0.3) = 0$

$\because \mathrm{F_b} = 1.178N$ *From (3)*

$\because \mathrm{F_t} = 0.09064(8) = 0.7251N$ *From (4)*

$\therefore \tau_{req} = F_b(0.15) + F_t(0.3) = 0.3942Nm$

Again, since the team did not include any friction that may be caused by shaft or even inside the motor, safety factor of 2 is taken into consideration to compensate some friction and efficiency loss.

$\therefore \tau_{req} = 0.3942(2) = 0.7885Nm = 7.734kgcm$ *(5)*

In the case of this assumption, the torque required to hold the gate is not an issue as it is indeed easy to find a suitable motor from the market and integrate it into the robot.

Then, the team took another extreme case, considering the case of collecting 2 basketballs at one single journey, and calculated estimated torque required. To simplify the calculation, 2 basketballs are assumed to contribute on 'Fb' together at the same location.

$(Clockwise)\ \tau_{req} - F_b(0.15) = 0$

$\because \mathrm{F_b} = 1.178(2) = 2.356N$ *From (3)*

$\therefore \tau_{req} = F_b(0.15) = 0.3534Nm$

With the safety factor of 2:

$\therefore \tau_{req} = 0.3534(2) = 0.7068Nm = 6.934kgcm$

The required torque is even lower than the previous case and thus it is acceptable.

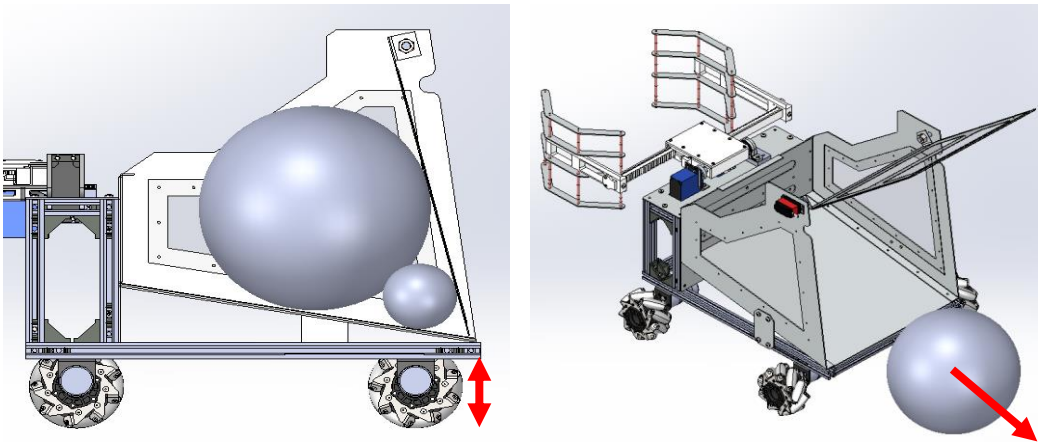


Figure 54 Unloading Mechanism Issues

As the robot releases the ball with this mechanism, the team found that there is still a high possibility of not securing the ball inside the starting zone. It is mainly because the sloped bottom plate accelerates the rolling of the ball, and the gap between the ground and the container at certain height introduce sudden acceleration downward. Thus, the team had to equip the robot with the capability of stopping the ball after unloading. Below figure clearly shows how these extra blocking parts are stopping the balls from rolling further.

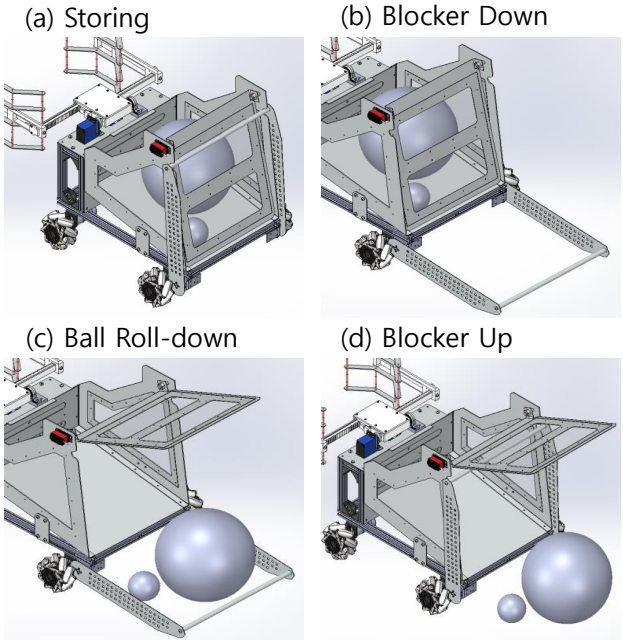


Figure 55 Arm Blocker Integration

3.5.3 Design Ver.3 Evaluation

After modifying the design concept through the Design Ver.1, Ver.2 to Ver.3, the team was able to improve the robot in every iteration of re-designing, and the Ver.3 was indeed conceptually flawless. Furthermore, the modular approach design allowed the team to quickly modify the problematic module in later stages without changing the overall design, which would be very time consuming and not efficient.

The only uncertainty in this Design Ver.3 was the yoga band integrated gripper parts, and the fastest way to check and find alternatives in case of the malfunction was by trial-and-error. It is because the material property of the resistance band, such as friction coefficient, was very dynamic, and the price of the yoga band is very cheap. Instead of wasting time on researching so many different types of yoga band material, the team decided to purchase a few options and try them on the gripper, which is more accurate and faster.

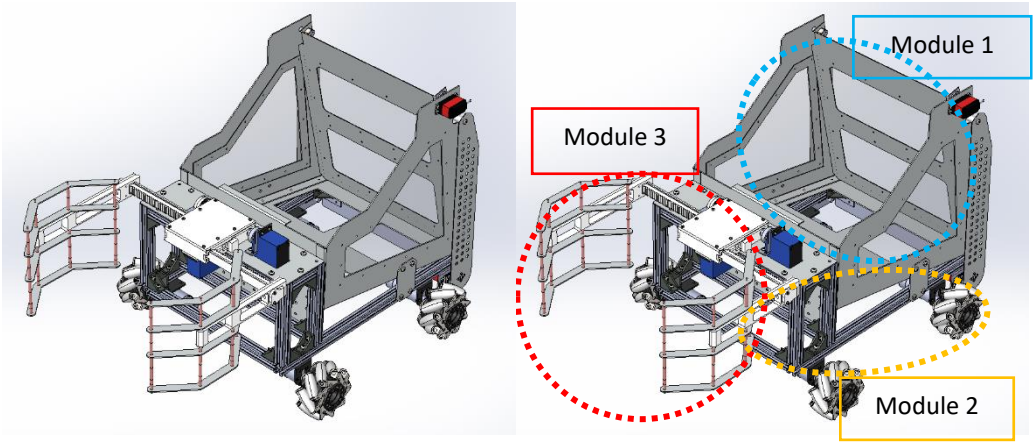


Figure 56 Finalized Design Ver.3

3.6 Overall Design Development Summary

Below is the summary of all design concept development stages to clearly illustrate the improvement from (a) Design Ver.1, (b) Design Ver.2 to (c) Design Ver.3.

(a) Design Ver. 1

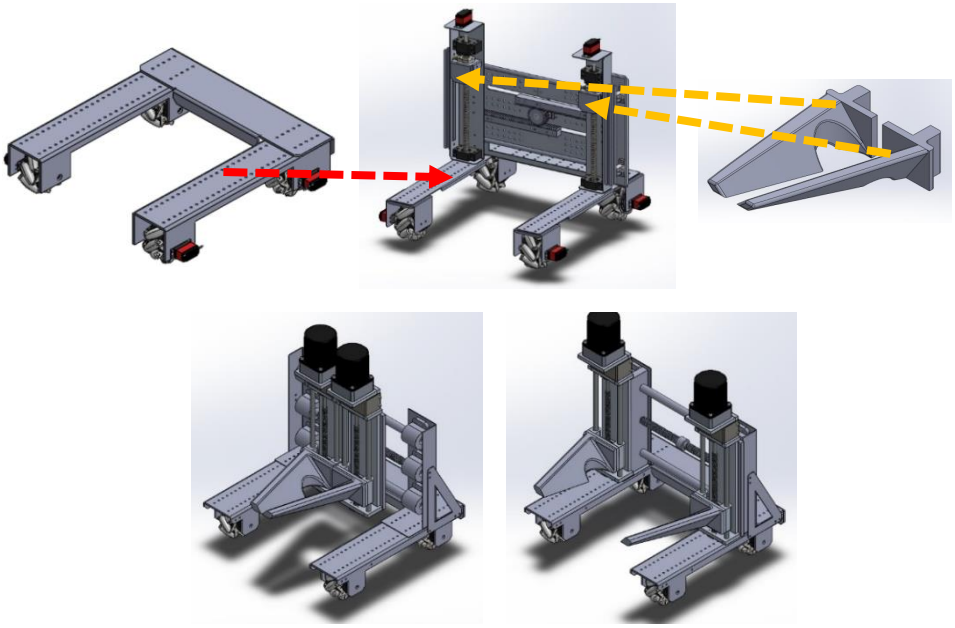


Figure 57 Design Ver.1 Summary

(b) Design Ver.2

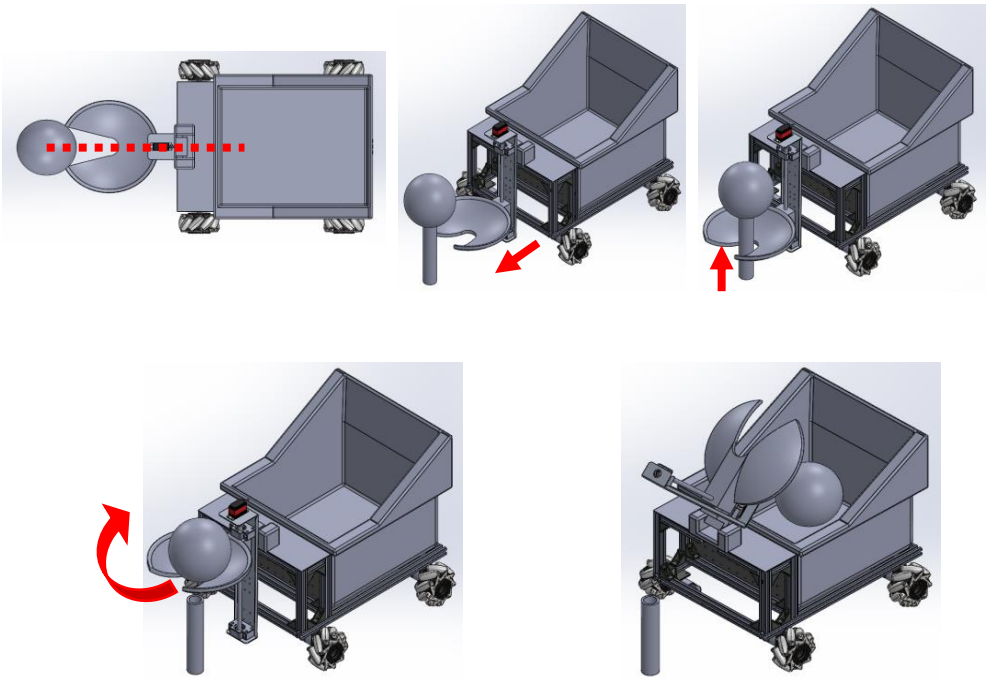


Figure 58 Design Ver.2 Summary

(c) Design Ver.3

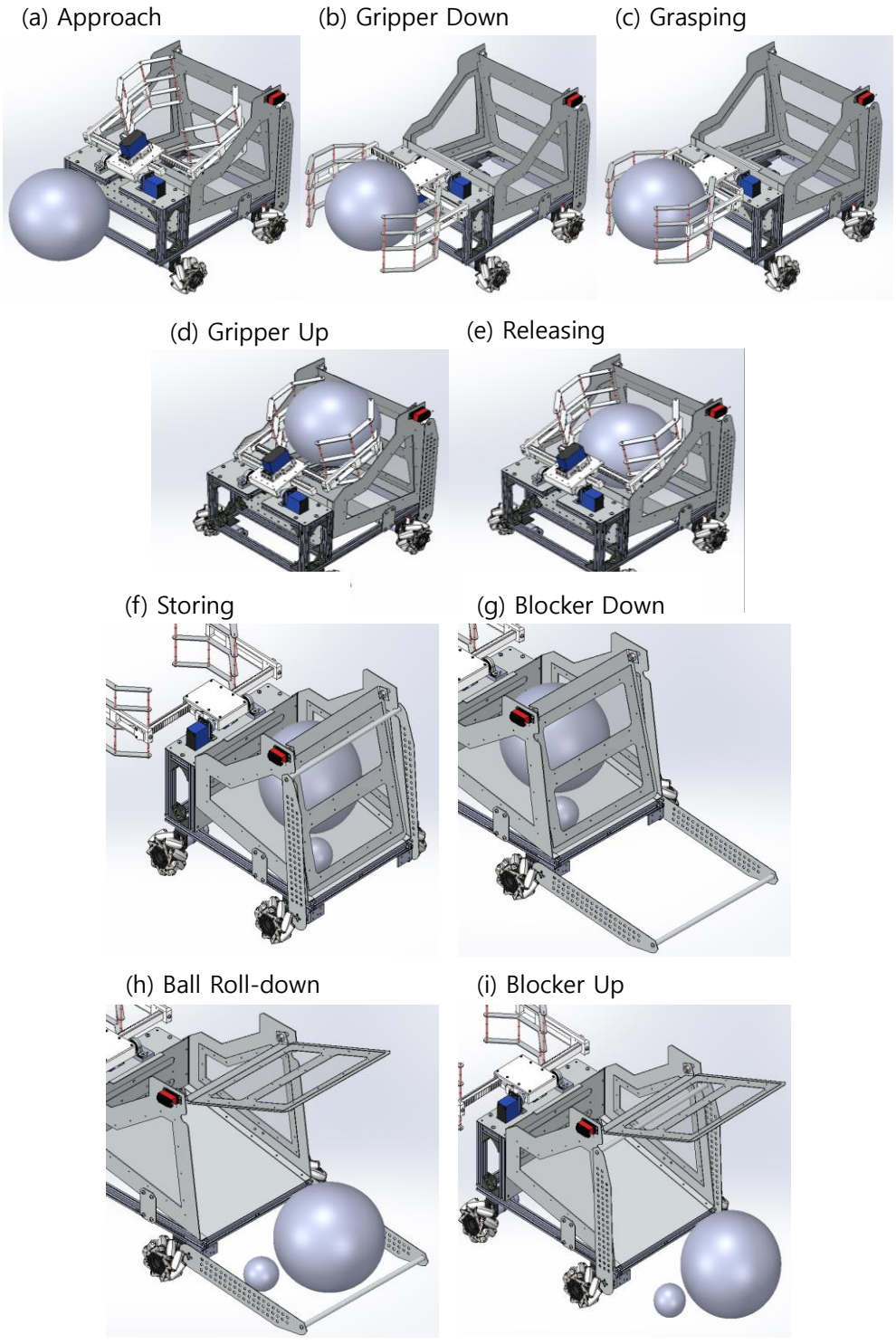


Figure 59 Design Ver.3 Summary

3.7 Prototype Manufacturing – Ver.3

As the Ver.3 design was finalized, the team already had some ideas of what material and parts should be purchased and fabricated, as well as the manufacturing method for certain customized parts. In this chapter, the material and parts selection will be discussed firstly, and the manufacturing processes of each part will be introduced briefly. Bill of Material (BOM) is attached in Appendix VI.

3.7.1 Standard Parts Selection

In this Ver.3 design, the standard parts considered were motors, mecanum wheels, microcontrollers, and battery which the team had to carefully analyze and research on their specification and choose the most appropriate options. Since the torque required is calculated for each mechanism during the design development stages, the team focused on following the simple selection graph designed by themselves for motors. Also, as the characteristics of different mecanum wheels were researched, the team directly refer to its dimension and maximum payload specification to choose the best option.

3.7.1.1 Motor Selection

The team first researched on the different types of motors available in the market that can be implemented in each mechanism, and below is the selection options graph designed to understand the characteristics of the stepper, servo, and standard DC motor.

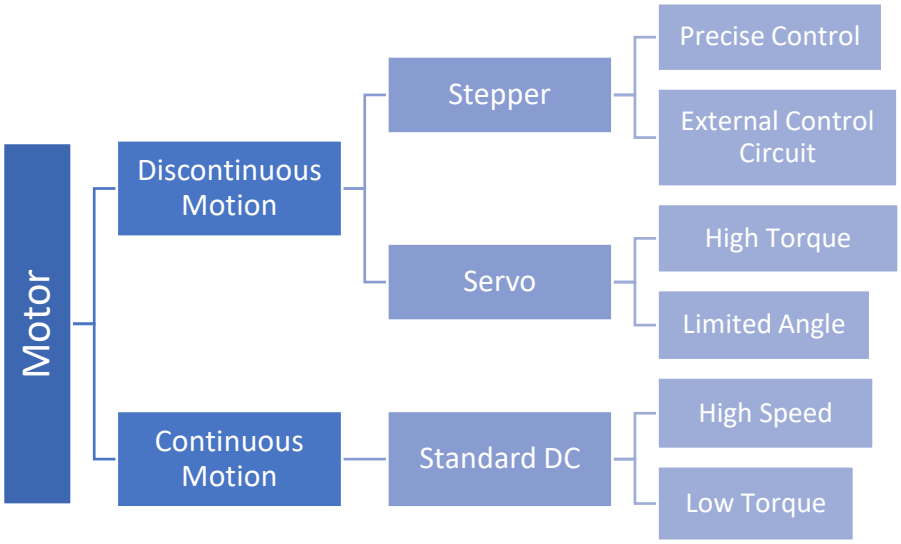


Figure 60 Motor Selection Criteria

The most common motor used in robotic projects is obviously standard DC motor, which is relatively cheap and easy to control. With adequate power supply, it can easily reach high rotation speed with constant torque across the speed range. Its control is as simple as a switch, the controller just needs to supply certain voltage for the operation, and it spins in the other direction when the voltage is reversed. Thus, it could be a good option for the application where this is used to drive the wheels. However, when this is applied to a system requiring a precise control on position, or high holding torque to hold a payload, it is not the suitable option because it does not have any closed-loop sensor to keep the track of its position and also has low torque at zero speed.

Servo motors that are widely used for small-scaled projects consists of four fundamental components including a DC motor, gearing set, control circuit and a position sensor like an encoder. It is applied to those systems where moderate-precise positioning and high torque at useful speed is required, instead of just high speed. Especially, it is capable of providing holding torque to withstand certain loading at a specific position. On the downside, they usually have limited rotational angle ranging from 0 to 180 degree

or up to 270 degrees. Hence, they are not welcomed in the application where infinite continuous rotation is required.

Stepper motors are similar to the servo motor but with different motorization method. Its motion seems to be continuous due to high pole counts but in fact, it is not. It energizes multiple poles, around 50 or even more, in sequence so that the rotor turns in a series of increments or steps. It may be the best option for precise positioning, speed control as well as in such application requiring high torque at low speed. It has the ability to rotate accurately and also in an extremely small step without the aid of encoder. However, it also has its disadvantages such as constant consumption of maximum current and relatively complicated circuit system than the other two. The summarized motor comparison on their functionality is as shown in below table:

	Advantages	Disadvantages
DC	High speed and extremely easy to control. Continuous motion.	Low torque at zero speed.
Servo	High torque at low speed and good precision. Closed loop system.	Limited angle of rotation usually ranging from 0 to 180 degree or up to 270 degree. Discontinuous motion.
Stepper	Precise control in positioning and speed. High torque at low speed.	High consumption or current and the need of external circuit system. Discontinuous motion.

Other than the different application and functionality of each motor types, since the team considers using Arduino to control them, the operation voltage must be ranged in 5V to 12V.

3.7.1.1.1 Unloading Mechanism

The unloading mechanism required certain holding torque at zero speed to hold the gate from opening itself due to the weight of the collected balls, and it does not need high-precision in control on positioning or speed but just enough rotation angle to fully open the gate during unloading. Thus, the team decided to find a suitable servo motor for this mechanism. Referring to the calculation for the unloading mechanism:

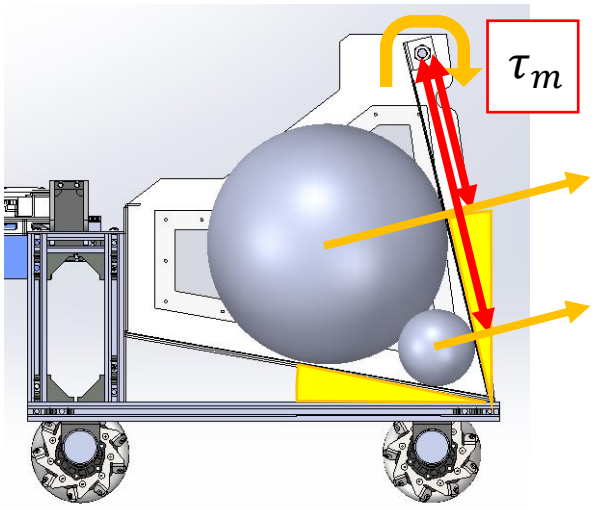


Figure 61 Motor Selection - Unloading Mechanism

Assuming that τ_m is maximum torque provided by motor, then it must be larger than required torque to hold the gate from opening due to the weight of collected balls.

$$\boldsymbol{\tau_m > \tau_{req} = 7.734 kgcm} \qquad \textit{From (4)}$$

Referring to stall torque specification provided in data sheet, the team was able to find **D-25HV** servo motor which its stall torque is **25kgcm** operated at 7.4V.

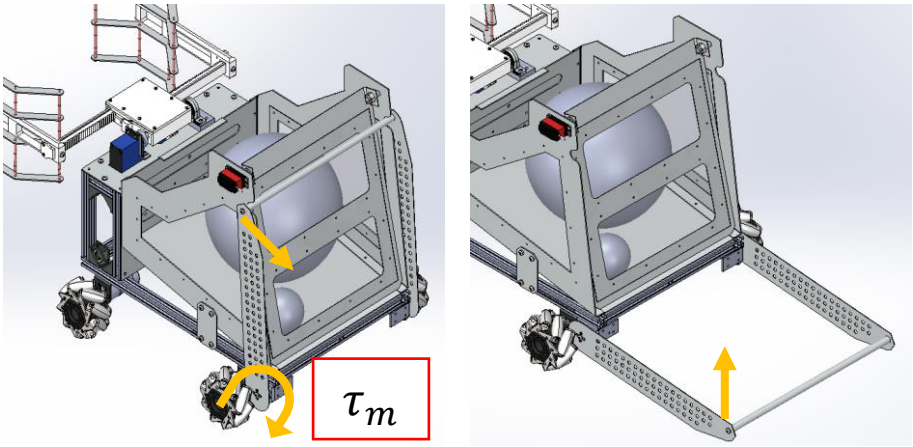


Figure 62 Motor Selection - Arm Blocker

To let down and up the blocker arm, this servo motor has no problem because its torque demand is much lower than the gate due to its light weight. The detailed motor specification is attached in Appendix VII [30].

3.7.1.1.2 Flipping Mechanism

For flipping mechanism, a strong motor with great torque is required to successfully flip the Module 3. It should also have capability to hold the position of the gripper at certain angle according to the command from the controller. Stepper motor could be one of the options, but since it requires additional motor driver units for each motor to control its function, the team find servo motor easier and simpler because it can fully satisfy the work without any extra external circuits. However, the team could not adopt the previous servo motor, which is used for unloading mechanism, and had to search for a motor with greater torque.

For this flipping motion, its required torque was indeed the highest among all mechanism.

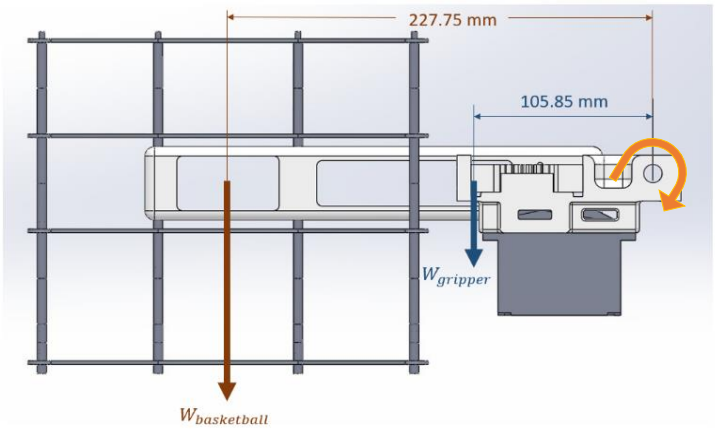


Figure 63 Motor Selection - Flipping Mechanism

Assume that it is holding a basketball and with safety factor of 2:

$$\boldsymbol{\tau_m > \tau_{req} = 2.627(2) = 4.963Nm = 48.69kgcm} \qquad \textit{From (2)}$$

The D-25HV servo motor used in unloading system is obviously not strong enough to overcome the extreme case of flipping basketball. Thus, the team could not use the same motor but instead used **XP-70HV** servo motor which provides **70.1kgcm** at 7.4V supply. In addition, its operation voltage was identical as D-25HV servo motor that allowed the robot to have identical battery through one Arduino. Again, its detailed specification is attached on Appendix VIII [31].

3.7.1.1.3 Racks-and-Pinion Mechanism

For the racks-and-pinion system, the team has already estimated required force to hold a basketball, and with this information, the required torque at the gear could be approximated as shown below:

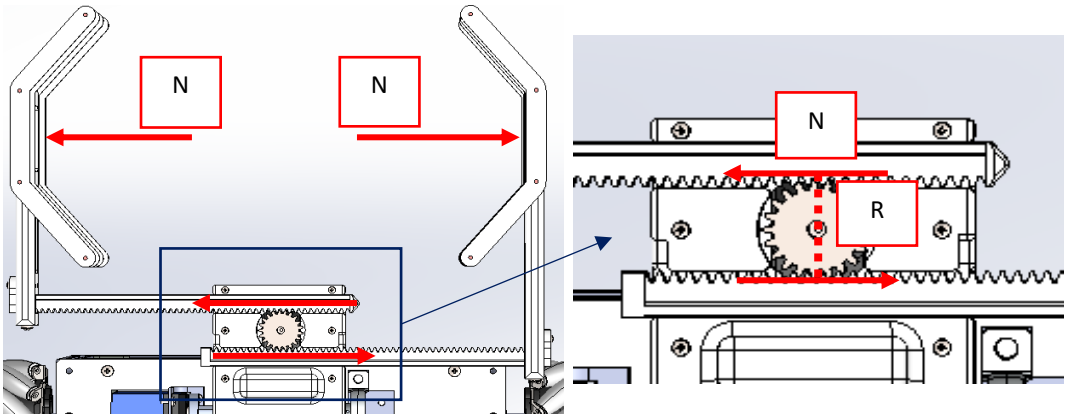


Figure 64 Motor Selection - Racks-and-Pinion System

$\tau_{req} = 2(N \times R)$

$\because N = 16.78 \approx 17N$ *From (1)*

$\because R = 0.02m$

$\therefore \tau_{req} = 2(17 \times 0.02) = 0.68Nm$

Because the friction between the pinion hub and the racks are extremely high, and the pinion itself also has friction with the top surface of the hub, the safety factor of 3 is given for the calculation.

$\therefore \boldsymbol{\tau_m > \tau_{req} = 3(0.68) = 2.04Nm = 20kgcm}$

The team did not consider driving this mechanism with DC motor even though it gives a smooth continuous motion. It is mainly because the torque requirement could be much higher than what the team has calculated and in that case the speed of DC motor could be really slow. In other words, the DC motor will provide not sufficient torque at

high speed while servo motor provides reasonable speed at greater torque.

Even though it is still acceptable for the previous servo motor with 25kgcm torque, as the team needs to conduct trial-and-error test with different resistance band for universal gripper application, students decided to have **XP-70HV** servo motor because it has much stronger torque to continuously move the racks to deform the elastic band in the shape of object.

3.7.1.1.3.1 Servo-to-DC Motor Modification

To allow the servo motor to rotate without any limitation to rotating angle like a standard DC motor, the team has done slight mechanical modification on this servo motor.

In order to change the servo motor into a D.C. motor, some of the components in the servo motor must be removed. The parts that require removal are the potentiometer and the stopper. The potentiometer is responsible for position control in the electronic perspective, where the stopper is a mechanical device that is implemented on the gear of servo motor that stops the servo motor from going further mechanically.

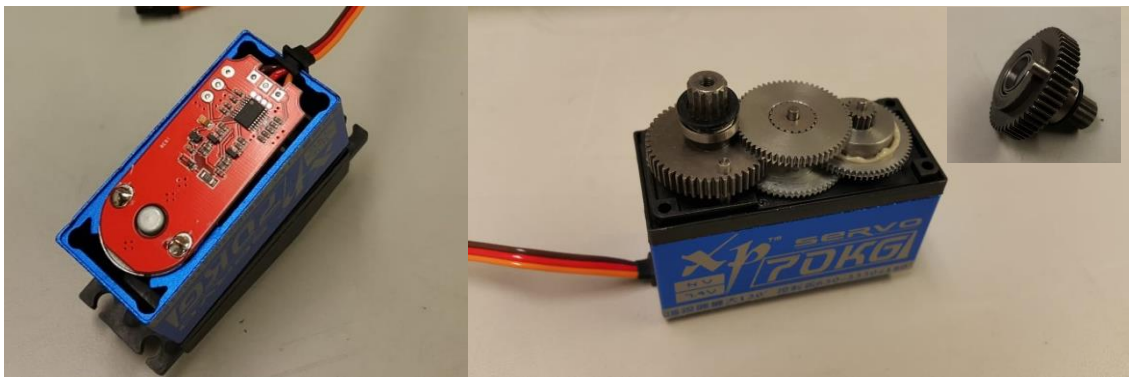


Figure 65 Potentiometer and stopper in servo motor

Figure 66 on the right shows the saw cutting modification of removing the stopper, such that it can mechanically behave like a normal D.C. motor. As for the potentiometer removal, it could be conducted by simply removing the PCB in the servo motor, and solder to extend the wires connecting the positive and negative terminal out from the servo motor.

Figure 66 Stopper removal

After completing the aforementioned modifications, the modified servo motor could simply be treated as a D.C. motor and used for any D.C. motor application normally. Therefore, it could be used for controlling the opening and closing motion of the gripper without any motor rotational limits.

3.7.1.1.4 Driving

The robot weight is the key element of choosing the right motor for driving, and the modular weight estimation are as shown below:

Module	Estimated Weight (kg)
(a) Module 1 (Lower body – Aluminum Profile Chassis, Wheel, etc)	5
(b) Module 2 (Upper body – Aluminum Sheet Container, Shaft, etc)	4
(c) Module 3 (End-effector – ABS 3D Printed Racks-and-Pinion, etc)	3
(d) Others (Battery, Motor, Bearing, Arduino, Collected Ball, etc)	3
Total Weight	15

Above the weight is not carefully measured but rather exaggeratedly estimated based on 3D design in SolidWorks, and it is reasonable to consider maximum robot weight to prepare for the extreme case.

Assuming that the estimated weight of the robot is evenly distributed to each wheel, then the reaction force, N, of this distributed weight on the wheel is:

$$N = \frac{\text{mg}}{4} = \frac{(15)(9.81)}{4} = 36.79 \qquad (6)$$

Assuming that the static friction coefficient between the wheel and ground is 1, which is in fact much smaller because of the slipping characteristics of the mecanum wheel. However, since the team do not know the floor surface material whether it is carpet, wood, cement or even bricks, the team decided to assume as above to ensure that it can deal with any kinds of ground surface.

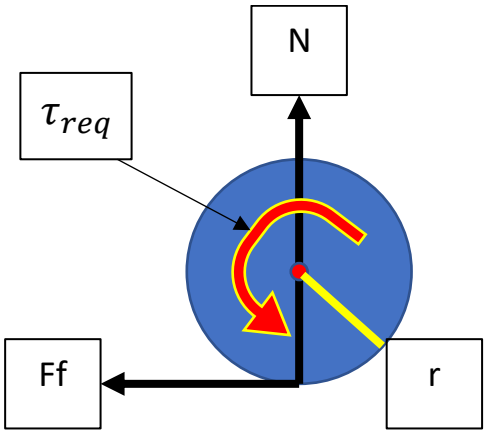


Figure 67 Motor Selection - Wheel Driving

$$F_f = \mu_s N = (1)(36.79) = 36.79 \qquad \textit{From (6)}$$

$$\tau_{req} = \text{rF}_\text{f} = (0.05)(36.79) = 1.839Nm = 18.04kgcm$$

This calculation shows torque required for each wheel motor to drive the entire robot in extreme case of having 15kg. Referring to the data sheet provided for DC motor XD-37GB520 as attached in Appendix IX [32], it rotates at the speed of 50rpm providing 7kgcm torque, and its stall torque is 26kgcm. To understand the robot's running speed, the team again assumed that there is no any attenuation due to friction or torque, but just calculated with the estimated maximum angular speed of 50rpm.

$$Angular\ speed = \ w = 50rpm = \frac{(50)(2\pi)}{60} = 5.236\theta/t$$

$$Linear\ speed = \ v = rw = (0.05)(5.236) = 0.262m/s = 15.7\text{m/min}$$

With this speed, it only takes around 19 seconds from one end of the playground to another, which its distance is 5m. Also, as this calculation is estimated with relatively high friction coefficient, where the team thinks should be much smaller due to slipping characteristics of mecanum wheel and the floor surface material itself, the torque required shall be smaller than our assumption and thus the speed may be faster than what we have calculated here.

3.7.1.2 Mecanum Wheel Selection

To choose appropriate mecanum wheel, the team had 2 candidates, which were 60mm diameter and 100mm diameter mecanum wheel. The major difference between them was payload that they can carry, and four 60mm mecanum wheels as a set were able to drive 10kg payload while 100mm set could drive up to 30kg. Thus, the team decided to choose 100mm.

3.7.1.3 Microcontroller Selection

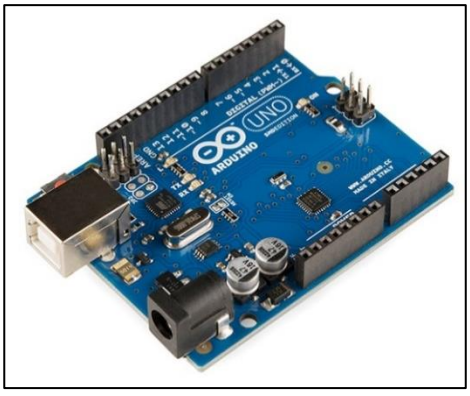


Figure 68 Arduino UNO

Arduino is an open source hardware and software organization that manufactures single-board microcontrollers and kits used in building digital and electronic devices. The most prominent model from this company is the Arduino Uno shown in Figure 68. Arduino Uno is a microcontroller board based on the ATmega328P chip developed by Atmel. It has 14 digital input/output pins, of which 6 can be used as PWM outputs, 6 analog inputs, a 16

MHz quartz crystal, a USB connection, a power jack, an In-circuit serial programming (ICSP) header and a reset button [33].

PWM or Pulse Width Modulation is a very important feature for this project and will be used to control the 4 servo motors of the machine. It refers to the technique where a microcontroller rapidly turns the power to a device on and off at a given frequency and a certain pulse width. By varying the width of the pulse, desired power is provided to the device connected. The term "duty cycle" describes the amount of time the power is on and is measured in percentage. A lower duty cycle corresponds to lower power whereas a higher duty cycle refers to higher power. This concept can be visualized in the adjacent figure [34]and will be useful for controlling the speed of the motors.

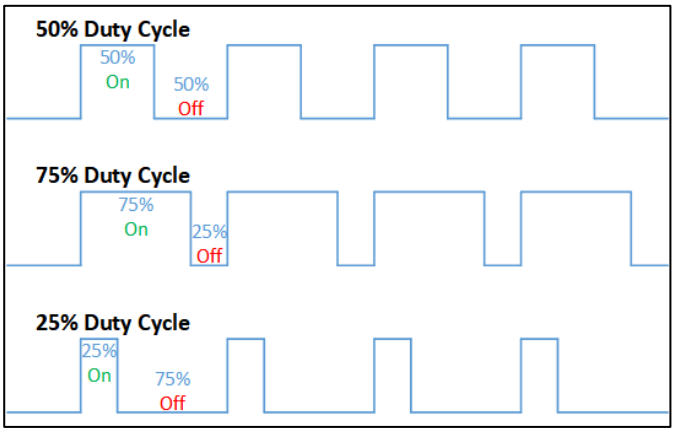


Figure 69 Duty Cycle variations

Arduino also has their own Integrated Development Environment (IDE) for programming purposes. This software is available across multiple platforms (Windows, Linux, macOS) and is based on Java whereas the programming languages supported are C and C++ with special structures / syntax. The interface is very user-friendly with a single click to compile and upload the code to the Arduino board. The microcontroller can be connected to any PC via a USB cable and it can be powered by a 5V battery. The technical specifications of Arduino Uno can be found in the Appendix X.

For the current project, the team has attached a wireless PS2 module on the Arduino Uno board to allow them to control the robot with a wireless PS2 controller; this module comes pre-equipped with an Adafruit motor-shield.

3.7.1.4 Battery Selection

The team researched battery types that are widely used for small scale robot projects and learned about rechargeable LiPo (Lithium Polymer) battery and NiMH (Nickel-Metal Hydride) battery. The biggest difference between these two is charging and discharging methodology, but what really matters and important is not about the technology, but about which one is more suitable for this robot operation. Thus, students focused on their properties to understand, analyze and decide.

Nickel-Metal Hydride battery is a round shape cell that usually can be found in toy RC cars. It is relatively cheap, safe and reliable because it does not explode upon overloading. Also, it has voltage increments of 1.2V that provide varieties of operating voltage. On the other hand, it has low energy density which means it will be used up very fast. This could be solved by having extra batteries connected together to supply more power, but in that case, another problem that may arise is extremely heavy weight and thus affecting its efficiency.

Lithium Polymer, or more correctly Lithium-Ion Polymer battery is a cell in flat design usually, instead of the round shape. It is suitable for those application consuming high energy as its energy density is very high compared to the NiMH battery. Also, its discharge rate is much higher and so able to supply power to multiple loads simultaneously. However, it must be used in an appropriate manner as it may explode if it is handled incorrectly. Also, its voltage increment is not as sensitive as NiMH but

increases with a 3.7V gap. The most commonly used ones are 7.4V and 11.1V, where they have 2 and 3 LiPo cells respectively. Below is the summary of advantages and disadvantages of LiPo and NiMH battery [35, 36].

Battery Type	+/-	Description
LiPo Battery	Advantages	- High Energy Density - High Discharge Rate - Flat Cell Design
	Disadvantages	- Explosive (especially during charging) - High Voltage Increment per Cell (3.7V/cell)
NiMH Battery	Advantages	- Safe and Reliable - Small Voltage Increment per Cell (1.2V/cell)
	Disadvantages	- Heavy and Bulky - Low Energy Density - Inefficient

Based on above information researched, the team first did a simple analysis on the minimum quantity of the battery cell required to power the robot.

Battery Type	Loads	Cell Quantity	Total Quantity
LiPo Battery	7.4V Motors	Two 3.7V cells to provide exactly 7.4V.	5 cells in total
	12V Motors	Three 3.7V cells to provide 11.1V.	
NiMH Battery	7.4V Motors	Six 1.2V cells to provide 7.2V	16 cells in total
	12V Motors	Ten 1.2V cells to provide exactly 12V	

Even though both are fairly able to supply power at a favorable voltage, there is a significant difference in the quantity of battery cell required between LiPo and NiMH. LiPo battery only needs 5 cells to operate at desired voltage while NiMH needs 16 cells. However, it is possible that the 16 NiMH cells have a lighter weight than LiPo but considering the much lower energy density of NiMH compare to LiPo, NiMH may need extra sets of cells while 1 set of LiPo is more than enough to operate for a long period. Also, having extra sets of NiMH cells may take much more space than LiPo which is undesirable in this project.

Hence, the team decided to use LiPo battery and to choose the product, the power consumption of overall loads is estimated to understand the required capacity of the battery, and some assumption is made as listed below:

1. The data of the current for servo motors are provided with minimum and maximum current required according to the torque, which is 5mA and 3800mA respectively, and the team took the average for this analysis.

2. No minor loads such as microprocessor, lights and no energy loss are considered in this analysis. Below table shows the estimated power consumption of major loads, which are motors:

Parts	Operating Voltage (V)	Current (A)	Unit	Power (W)
D-25HV Motor	7.4	1.875	2	27.75
XP-70HV Motor	7.4	1.875	2	27.75
XD37GB520 Motor	11.1	0.5	4	24

Thus, the 7.4V battery requires more than 55.5W to run 4 servo motors, and 11.1V battery needs at least 24W to run 4 standard DC motors. With this information, the team referred to ‘capacity’ and ‘watt-hour’ specification of the available battery in the market and chose 5000mAh 11.1V LiPo battery and 7600mAh 7.4V LiPo battery.

Below is the summarized specification of two batteries chosen:

Battery	Ampere/hour (Ah)	Watt/hour (Wh)	Requirement (W)	Time (Hr)
7.4V LiPo	7.6	**56.24**	55.5	**1.013**
11.1V LiPo	5	**55.5**	24	**2.313**

Each one of these batteries is theoretically suitable to run the robot for 1 hour. However, since the servo motor current is assumed to be average but not the maximum,

it may run out faster, and the team does not want that to happen during the competition day. Hence, the team added one more 7.4V LiPo battery to make sure that it can run for a longer time, which is extended to 2 hours approximately.

Battery	Ampere/hour (Ah)	Unit	Watt/hour (Wh)	Requirement (W)	Time (Hr)
7.4V LiPo	7.6	2	**112.48**	55.5	**2.027**
11.1V LiPo	5	1	**55.5**	24	**2.313**

One of the disadvantages mentioned above regarding the LiPo battery is safety, and to reduce damage to LiPo and everything near it, the team learned critical cautions such as always using LiPo balance charger on the correct setting, keeping it in LiPo safe bag while charging and storing, and storing at the correct storage voltage. The technical specifications of the selected batteries are attached in the Appendix XI.

3.7.2 Customized Parts, Material and Manufacturing Process

In this section, the material selection, manufacturing processes and methodologies will be discussed. Before getting into manufacturing processes, it is important to know that the material determination for different parts is critical. During the design stage, all parts are designed for a purpose and they should fulfill and deliver their function when they are integrated into the final product. Out of more than thousands of engineering material such as polymeric materials, metallic materials, ceramics or even composites, all materials have their own material properties. Therefore, choosing the most suitable material after considering its costs and property is very important to suit our part application.

Apart from the material selection, in the engineering industry, there are different

kinds of manufacturing processes such as metal casting, milling, center lathing, and sheet metal fabrication. All these manufacturing processes could aid and assist in producing desired parts and products. However, the resulting part mechanical properties might vary for different manufacturing processes. Moreover, some manufacturing processes require high-end equipment and high operational cost. Therefore, it is important to choose the correct manufacturing processes such that the produced parts can fit their desired function, and the production of robot components could be cost-efficient.

In our robot, the components could be classified into 4 categories. They are aluminum extrusion profiles, aluminum sheet metal, 3D printed parts and customized parts. These components are manufactured through different processes based on their design specifications.

3.7.2.1 Aluminum Extrusion Profile

As the robot is required to have high mobility, weight accumulation of the parts must be minimized as much as possible. At the same time, the structure of the robot must be reasonably robust to withstand certain loads and external impacts. As a result, for the body structure of the robot, aluminum is chosen as the material of the framework. Due to the material properties of aluminum, its characteristic of having a high strength to weight ratio is very beneficial to the purpose when comparing to another widely used material, stainless steel. As aluminum is popular in the industry due to its property in most of the application, there are various market available products available for purchase. Based on the concepts of structural mechanics and proven from the previous experiment, it is learned that instead of a solid metal block or metal sheet, a more efficient way in material utilization is to reshape it and form a complex shape. As a result, aluminum extrusion profiles are chosen at the market as the robot chassis. The model of extrusion profiles that

are suitable for our application is JT-6-2020 aluminum extrusion profile, which is also widely used in the aerospace industry. It has a square cross-sectional area, with the dimension of 20mm as shown in Figure 70.

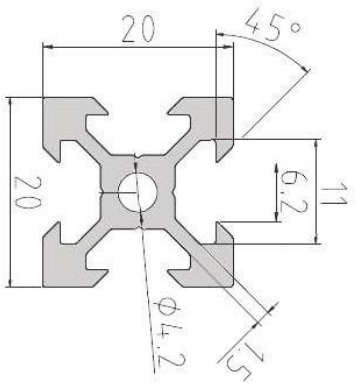


Figure 70 Cross section of JT-6-2020 Aluminum V-slot profile

One major customization required is that the profile length is longer than the requirement. Therefore, saw cutting will be applied to reduce the profile length, followed up by manual filing to ensure the burr induced at cutting edge are smoothened for safety purposes and to fine tune the profile length.

Another advantage of choosing aluminum extrusion profile is its compatibility for modular approach. Along the profile surfaces, there are tracks on it so that standardized part such as nuts and screw could be inserted. This allows mounting of additional parts onto the robot chassis to be easier, such that modular approach could be achieved.

3.7.2.2 Aluminum Sheet Metal

As mentioned previously, due to aluminum's material property advantage, it is also applied for fabricating some of the robot surfaces. These surfaces include the skeleton of the gripper, mounting sheet for the gripper, container box, gate, and the barrier arms.

In order to manufacture metal surfaces, sheet metal fabrication is chosen as the

methodology. This process utilizes computer aid and machinery for cutting the sheet metal to the specified dimension and requirement. At the industrial center of the university, there are various machinery available for sheet metal fabrication, namely Turret Press “AMADA AE2510NT” and the Press Brake “AMADA HDS8025NT”. The turret press contains a large variety of cutting tools within for punching holes onto the workpiece, and the press brake uses press brake tools and dies to bend the sheet metal parts. The machinery are shown below in Figure 71 and Figure 72.

Figure 71 Turret Press AMADA AE2510NT

Figure 72 Press Brake AMADA HDS8025NT

For the Turret Press to operate, path tool specification will be required in a program package provided by the machinery supplier AMADA called AP100.

Figure 73 AP100 program package from AMADA

SolidWorks part files are converted to DXF format, which is a 2D representation for the program to recognize. After importing the file, the cutting tools available in the press machine are selected, and the tool cutting path and order will then be specified as shown in Figure 74.

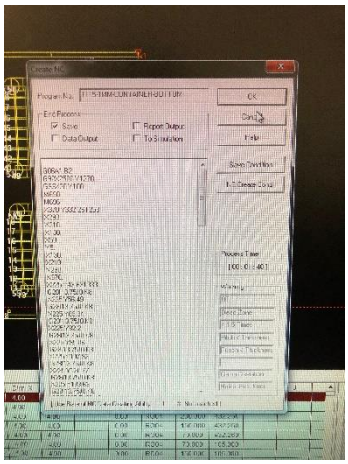


Figure 74 Turret press cutting tool path

One remark to take note is that after the generation of the tool cutting path, the corners or main support of the sheet metal are modified to leave small material connections with the raw metal sheet as shown in Figure 75. By doing so, the wanted parts will not be left behind in the press machine, and they could be removed afterward by pivoting a screwdriver near the metal connections. After specifying and importing the tool cutting path into the press machine, the raw metal sheet is mounted and coordinated

to be fed for cutting. After starting the program, the press machine will generate the sheet metal parts, and they could be taken for manual filing to remove the cutting burr generated.

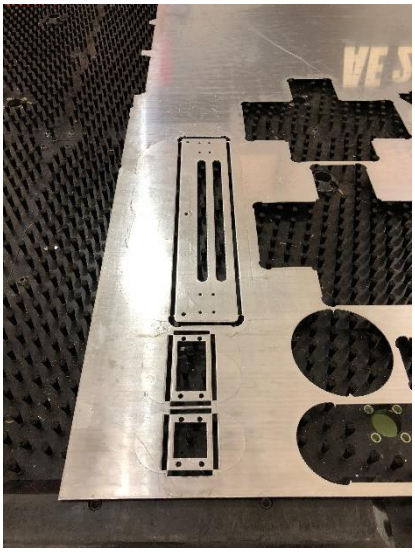


Figure 75 Sheet metal parts with linkage to raw material

After removing the burr on sheet metal parts, if the parts require further modifications such as bending, it will then be taken to the Press Brake. This machine consists of various press brake tools and a die. Before bending the sheet metal parts, programming is also required in AP100 to generate the bending procedures and tool selections. Since the metal sheet will tend to recoil due to elastic deformation, the press brake forces will be calculated by the program after specifying the sheet material and bend deduction.

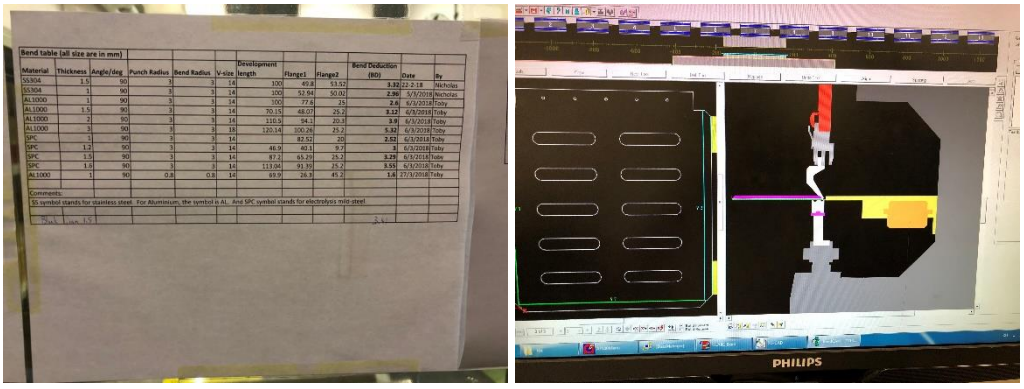


Figure 76 Bend deduction (left) and press brake programming (right)

After these processes, the sheet metal parts are measured for their dimension and bending angles as quality controls and evaluated whether they are exceeding the tolerance limit.

3.7.2.3 3D Printed Part

Some of the designed parts are not available on the market, and they are in relatively complex and detailed shape and structure. By applying conventional manufacturing methodologies, the parts will be expensive and time-consuming to be manufactured. Therefore, these parts would be 3D printed and be referred to as printed parts. These printed parts include the gripper body and the rack and pinion system.

The SolidWorks files are imported to the program GrabCAD Print, such that the printing procedures could be specified and printed by the printer Stratasys uPrint SE Plus. Moreover, the material selection is also specified. Since Acrylonitrile Butadiene Styrene (ABS) is having a lower weight and a reasonable strength for our application, among other thermoplastics, ABS is chosen.

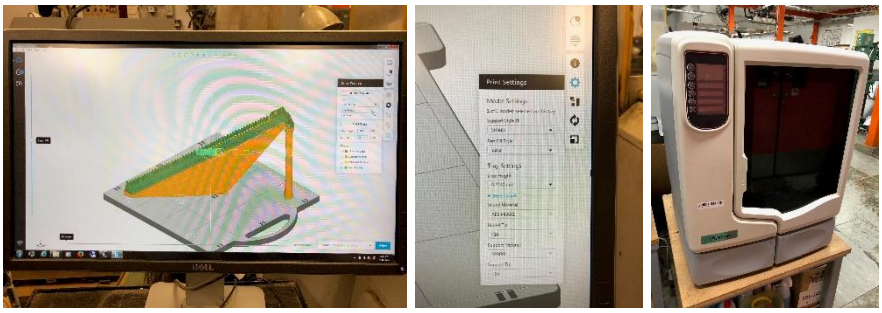


Figure 77 GrabCAD Print program, material specification, and Stratasys uPrint SE Plus

3.7.2.4 Customized Part

For some designed parts with less complex, they could be manufactured with conventional methods such as turning and milling. These parts are referred to customized

parts. These parts include the bearing support, some shafts and some addon modifications added to the robot.

For the shaft, it is manufactured through center lathing, then taken for milling and drilling to produce a flat surface and holes for tapping screws.

Figure 78 Shaft for Flipping

In order to provide support to the flipping shaft, a bearing is added for free rotation of the shaft. However, based on the alignment, an additional platform is required for the bearing to align with the shaft. Therefore, an aluminum block is added under the bearing for supporting the shaft. The aluminum block is manufactured through milling and drilling for holes such that it could be assembled together afterward.

Apart from these manufactured parts, additional modifications are also added to the robot. For example, window seals are purchased and added to the arms such that the balls are prevented from rolling out, and clips with rubber bands are mounted at the gate to prevent the gate from overextending.

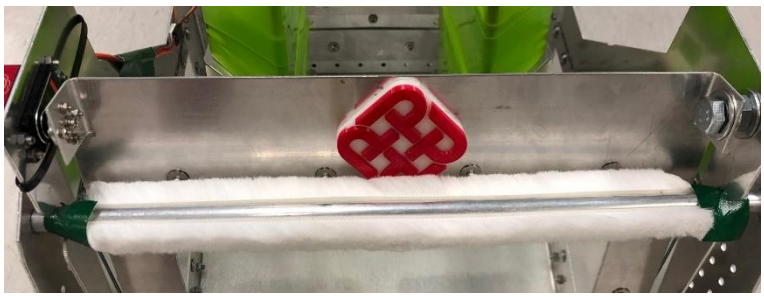


Figure 79 Window seals for the arm

Figure 80 Clips and bands for gate closing

3.8 Circuitry Research and Study

In this section, the connection of electronic parts of the robot will be discussed in detail. The electronic circuit design is important for the robot to be controlled and safety. It establishes a pathway for communication between the user and robot hardware, and among the hardware.

In our robot, 4 D.C. Motors and 4 Servo Motors are used as the actuators. In order to coordinate them and customize the program for controls, as discussed previously in Section 3.7.1.3 Microcontroller Selection, microcontrollers such as Arduino UNO, Arduino Motor Shield, and PS2 Remote Controller are used. First, the Arduino Motor Shield has to be connected to the Arduino UNO such that it could receive the signal information. The overview of the board connection and motor shield port configurations are shown below in Figure 81.

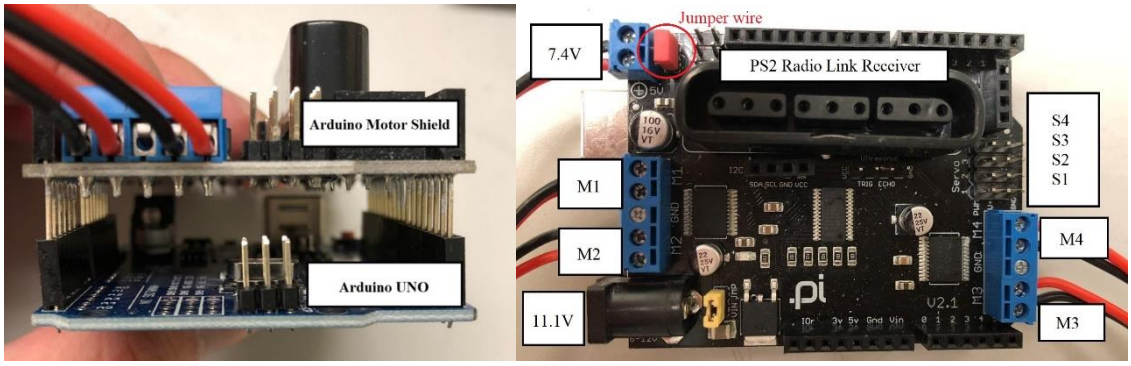


Figure 81 Board connection and motor shield port configuration

For the power supply, the D.C. motors are connected to the 11.1V LiPo battery, where the servo motors are connected to the 7.4V LiPo battery. Within the Arduino Motor Shield, the circuits for controlling D.C. motors and servo motors are connected with a jumper wire encircled in Figure 81. When connected, the two circuits can share the same power source. However, it would be removed to separate the power supplies for our application.

Figure 82 Battery mount

In order to ensure electrical safety, based on the specifications of the Arduino Motor Shield, the peak current is 3.2A. Since fuse with specified current values can short the circuit when the current surges over its limit, it can protect other components from current overloading. Therefore, 3A fuses are added to the circuit so that the circuit board will not be damaged. Moreover, the switches are added so that the power supply could be switched on and off with ease. The circuit is displayed in the circuit layout as shown in Figure 83.

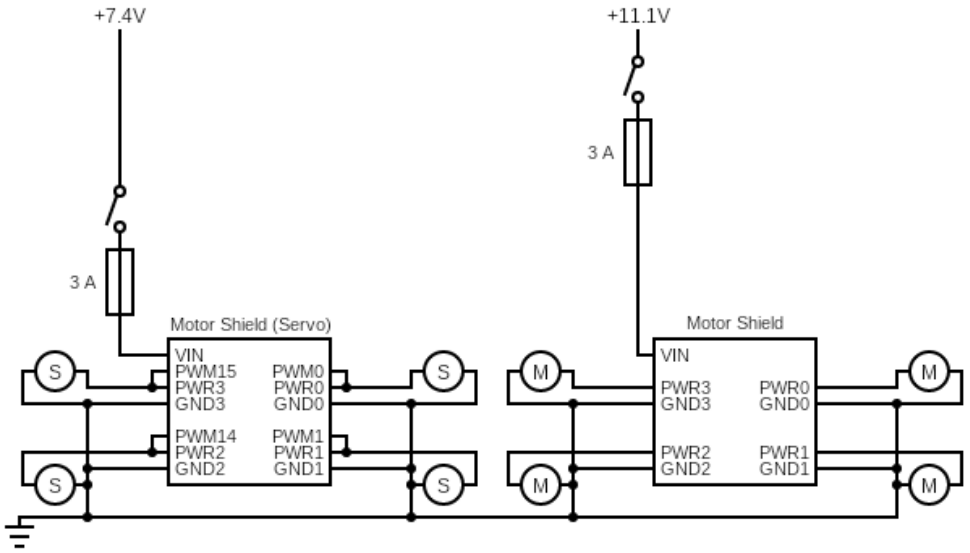


Figure 83 Circuit summary

As for the motor end connection, drivers are required for converting the signals into actuator movements. They are TB6612FNG D.C. motor driver and PCA9685 servo driver. A TB6612FNG driver can control up to 2 D.C. motors where PCA9685 driver can control up to 4 servo motors. Based on the 2 TB6612FNG drivers and PCA9685 driver on the Arduino Motor Shield, there are 4 sets of D.C. motor pins and 4 sets of servo motor pins. The connections are tabulated in the following Table 3 and illustrated in Figure 84 and Figure 85.

Arduino Motor Shield	TB6612FNG Drivers	M1	PWR0	Power source for anterior left D.C. motor
			GND0	Earth for anterior left D.C. motor
		M2	PWR1	Power source for anterior right D.C. motor
			GND1	Earth for anterior right D.C. motor
		M3	PWR2	Power source for posterior right D.C. motor
			GND2	Earth for posterior right D.C. motor
		M4	PWR3	Power source for posterior right D.C. motor
			GND3	Earth for posterior right D.C. motor
	PCA9685 Driver	S1	PWM0	Signal for servo motor that flips gripper
			PWR0	Power source for servo motor that flips gripper
			GND0	Earth for servo motor that flips gripper
		S2	PWM1	Signal for servo motor that opens/closes gripper
			PWR1	Power source for servo motor that opens/closes gripper
			GND1	Earth for servo motor that opens/closes gripper
		S3	PWM14	Signal for servo motor that deploys arm barrier

			PWR2	Power source for servo motor that deploys arm barrier
			GND2	Earth for servo motor that deploys arm barrier
		S4	PWM15	Signal for servo motor that opens/closes gate
			PWR3	Power source for servo motor that opens/closes gate
			GND3	Earth for servo motor that opens/closes gate

Table 3 Pin connections to actuators

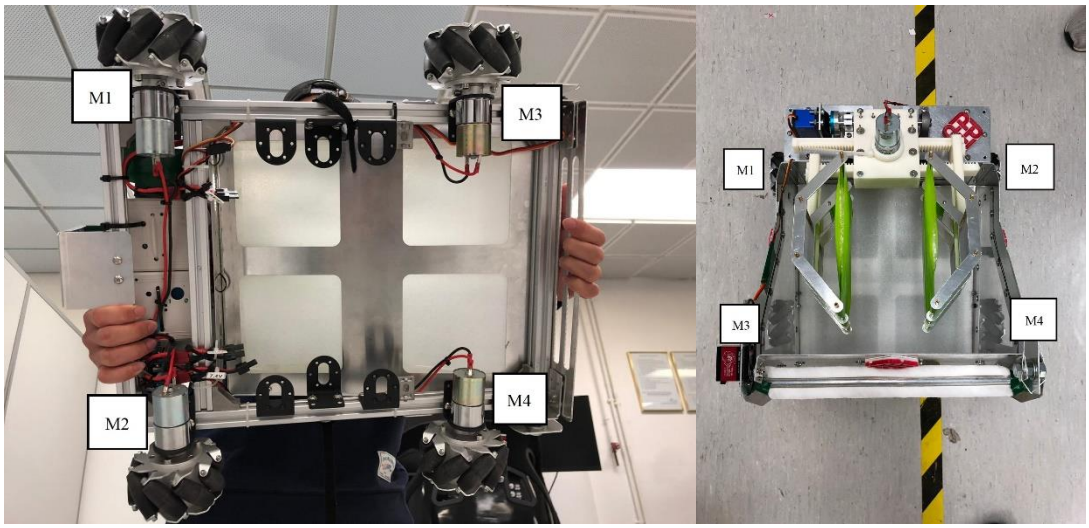


Figure 84 D.C. motor configuration

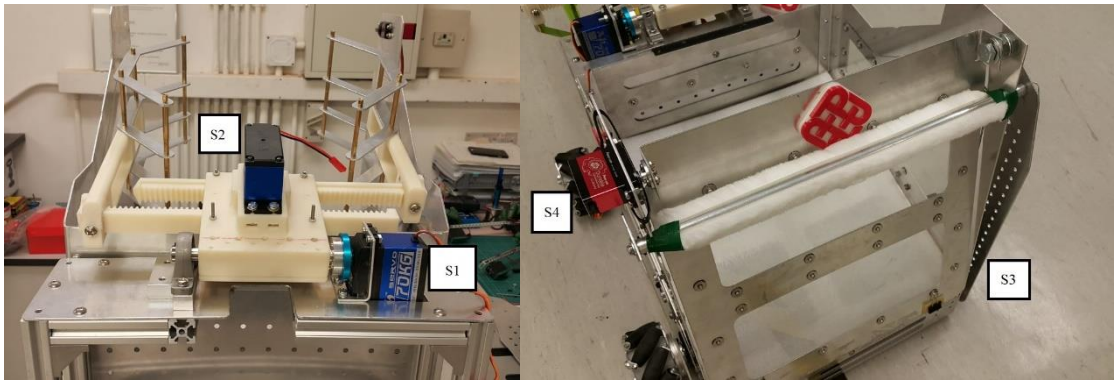


Figure 85 Servo motor configuration

3.8.1 D.C. Motor Driver TB6612FNG

The TB6612FNG driver has an integrated circuit layout as shown below in Figure 86.

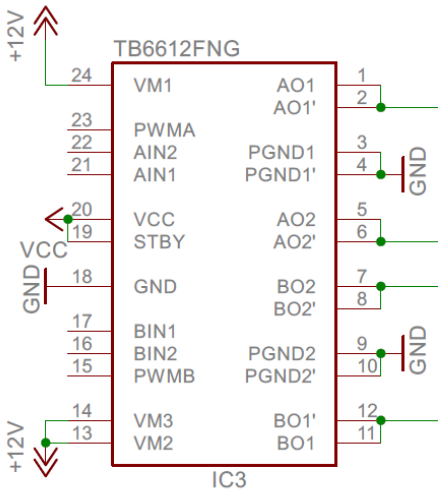


Figure 86 TB6612FNG integrated circuit module

For this TB6612FNG driver, the functions of the pins are evaluated as the following.

1	AO1	Channel A output 1, controlling one terminal of the D.C. motor
2	AO1’	
3	PGND1	Ground for motor power
4	PGND1’	
5	AO2	Channel A output 2, controlling another terminal of the D.C. motor
6	AO2’	
7	BO2	Channel B output 2, controlling another terminal of the D.C. motor
8	BO2’	
9	PGND2	Ground for motor power
10	PGND2’	
11	BO1	Channel B output 1, controlling one terminal of the D.C. motor
12	BO1’	
13	VM2	Motor power supply
14	VM3	
15	PWMB	Channel B PWM input
16	BIN2	Channel B input 2
17	BIN1	Channel B input 1
18	GND	Ground for signal power
19	STBY	Standby pin
20	VCC	Signal power supply
21	AIN1	Channel A input 1
22	AIN2	Channel A input 2
23	PWMA	Channel A PWM input
24	VM1	Motor power supply

Table 4 TB6612FNG pin functions

Pin 1-14 and 24 are the outputs from TB6612FNG driver to the motors, where pin 15-23 are the signal inputs into the TB6612FNG driver. This module will process the signals obtained and relay to the D.C. motors correspondingly. The controls on D.C. motors could be modeled in the circuit diagram illustrated below in Figure 87, along with the corresponding pin numbers from Table 4.

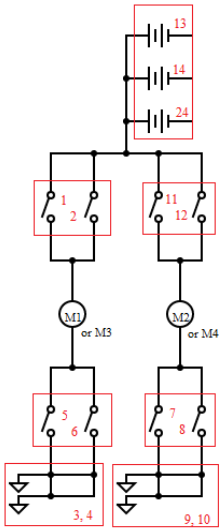


Figure 87 TB6612FNG driver control modeling

3.8.2 Servo Motor Driver PCA9685

As for the PCA9685 driver, it has an integrated circuit layout as shown below.

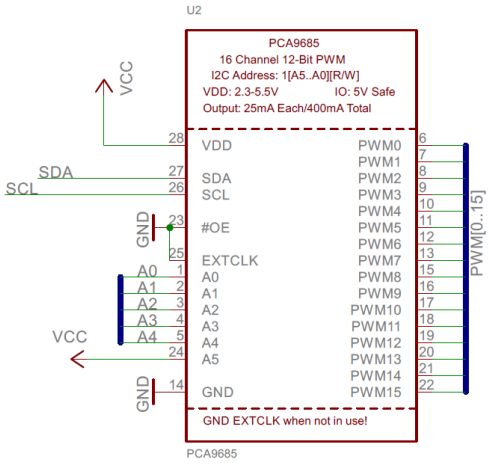


Figure 88 PCA9685 integrated circuit module

For this PCA9685 driver, the functions of the pins are evaluated as the following.

1	A0	Analog input 0
2	A1	Analog input 1
3	A2	Analog input 2
4	A3	Analog input 3
5	A4	Analog input 4
6	PWM0	Pulse width modulated power supply for motors
7	PWM1	Pulse width modulated power supply for motors
8	PWM2	Pulse width modulated power supply for motors
9	PWM3	Pulse width modulated power supply for motors
10	PWM4	Pulse width modulated power supply for motors
11	PWM5	Pulse width modulated power supply for motors
12	PWM6	Pulse width modulated power supply for motors
13	PWM7	Pulse width modulated power supply for motors
14	GND	Ground terminal for the motors
15	PWM8	Pulse width modulated power supply for motors
16	PWM9	Pulse width modulated power supply for motors
17	PWM10	Pulse width modulated power supply for motors
18	PWM11	Pulse width modulated power supply for motors
19	PWM12	Pulse width modulated power supply for motors
20	PWM13	Pulse width modulated power supply for motors
21	PWM14	Pulse width modulated power supply for motors
22	PWM15	Pulse width modulated power supply for motors
23	#OE	Low output voltage pin
24	A5	Analog input 5
25	EXTCLK	External clock input
26	SCL	Serial clock line, for time domain information
27	SDA	Serial data line, for package domain information
28	VDD	Power supply terminal for motors

Table 5 PCA9685 pin functions

This module will process the signals obtained and relay to the servo motors correspondingly. This driver differs from the D.C. motor drive by having additional pins for regulating the angular of rotation information.

3.8.3 Circuitry Evaluation and Modification

A short circuit problem happened during the operation as current surged. Therefore, the modified servo motor that is responsible for opening or closing the gripper is separated into an additional circuit board. However, the circuit board is designed for D.C. motors. Since the servo motor is already modified into a D.C. motor, it can function by simply connecting it to the power terminals only.

Similar to previous measures, through adding the switch and fuse to increase ease of control and safety, the new circuit is modified and displayed in the circuit layout as shown in Figure 89.

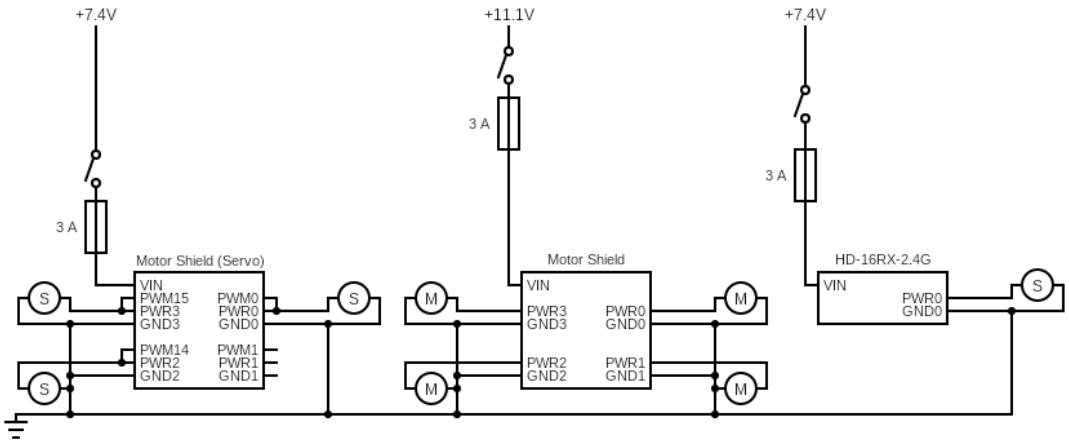


Figure 89 New circuit layout

3.9 Final Robot Assembly

As all the robot components and circuitry are well defined, the final robot is assembled together. The assembly flow of the three main modules to the chassis is illustrated in the following figure.

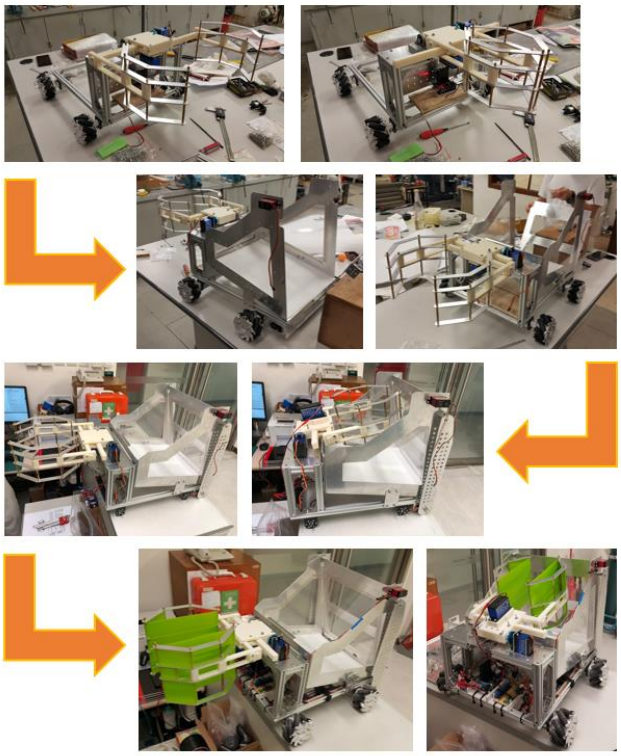


Figure 90 General Assembly Flow

The overall dimension of the robot meets the dimension requirement of the competition with 50cm width, 50cm length and 50cm height as shown in Figure 91, and its weight is approximately 10kg, which is less than the team' estimation. Apart from that, the modular approach adopted on the robot as shown in Figure 92.

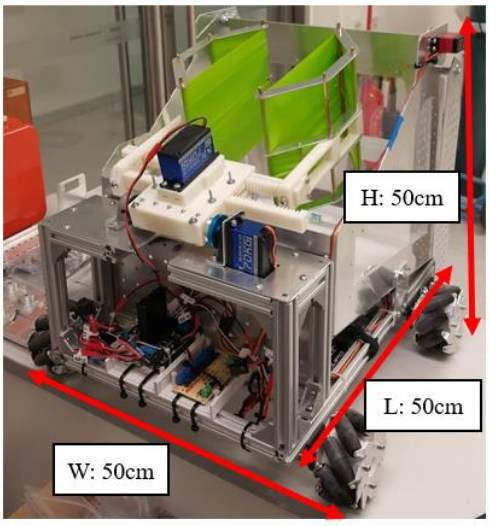


Figure 91 General Robot Dimension

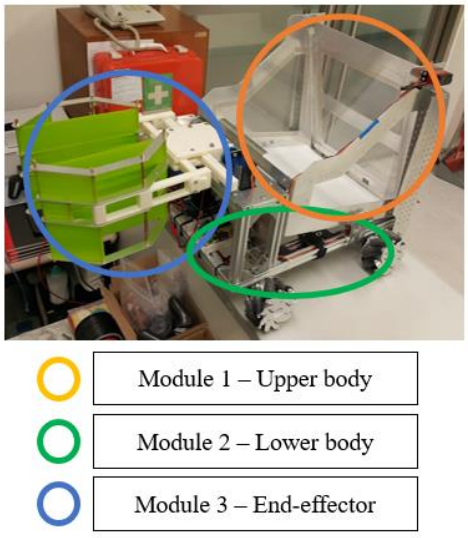


Figure 92 Modular Approach of Assembly

Some of the key components are illustrated in the following figure and table.

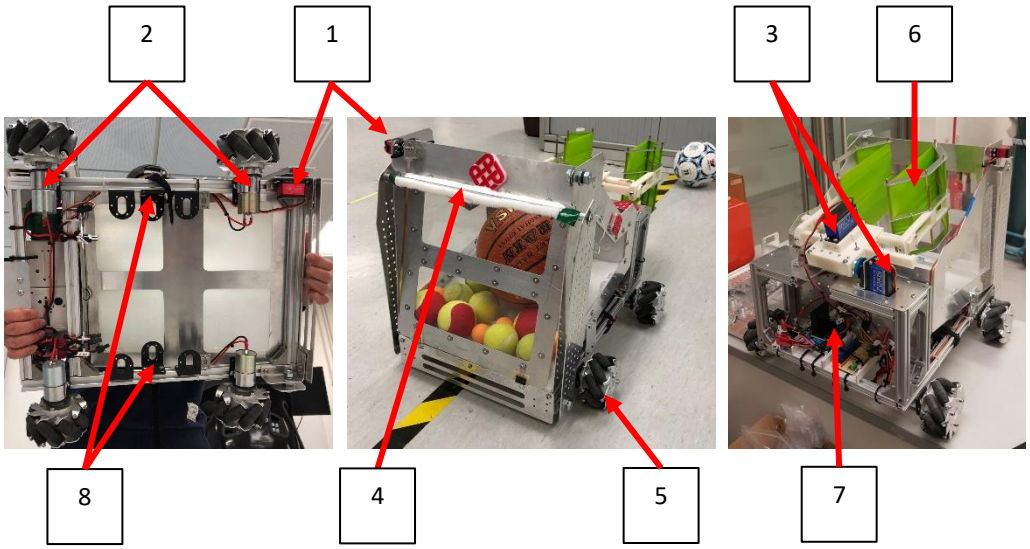


Figure 93 Design Ver. 3 Key Components

Part No.	Component	Part No.	Component
1	D-25HV Servo Motor	5	Mecanum Wheel
2	XD37GB520 DC Motor	6	Elastic Band
3	XP70HV Servo Motor	7	Microcontroller
4	Arm Blocker	8	Battery Holder

Chapter 4. ASME Robot Software Development

4.1 Microcontrollers

It is taught to everyone at a very young age that a CPU is the brain of the computer. Similarly, a robot also needs a brain to carry out the tasks it is built for accomplishing and this brain can be in the form of a microcontroller for small to medium scale applications like this project. A microcontroller is essentially a microprocessor core paired with memory and peripherals on a single integrated circuit (IC) that can be implemented to control and monitor electronic systems [37]. It is programmed to perform a certain task; in order to change its functionality new software must be installed. Today, the microcontrollers available are more powerful than the computers used 2 decades ago. The size of these ICs is reducing exponentially, and the storage capacity as well as computing power are ever-increasing. The most basic components of a microcontroller are a computer processor (CPU), Random Access Memory (RAM), Read Only Memory (ROM), Input/Output (I/O) interfaces, and other components like timers, interrupt controls, and so on. Variations in specifications of these core components of a microcontroller determine its uses and applications. There are many chips designated to be used for robotics and here the development environment is standard PC software, which allows creating and compiling programs, uploading programs to the microcontrollers and bridging in the programs during running in order to detect possible faults. Ease and comfort of usage of this software becomes decisive because during the development phase of the program, it will be the primary working area. Microcontrollers available today have the ability to add different kinds of modules and sensors (e.g. Wi-Fi modules, PS2 controller modules, light sensors) to customize the board based on the

application. For this project, 2 microcontrollers that are widely used and have a vast range of applications are utilized: i) Arduino Uno, and ii) Raspberry Pi

These two boards have been chosen mainly due to the availability of boards as well as modules in the market, availability of software resources online, and low costs.

4.1.1 Arduino Uno

The Arduino Uno board has been described in detail in Section 3.7.1.3 Microcontroller Selection. We can thus, explore the PS2 Module in detail in this section.

4.1.1.1 PS2 Module

The team purchased the wireless PS2 module (Figure 94) from an online store (Taobao) [38] after understanding the interfacing mechanism, software protocol, logic, and method of usage of this module. These areas have been discussed briefly to provide a basic understanding of this PS2 module to the reader.

4.1.1.1.1 Hardware Interface / Wiring Connections

The module has a transmitter and a receiver as seen in Figure 94 and the receiver has 9 pins on it which connect to the motor-shield on the Arduino Uno. For a wired controller,

there would be 9 wires connecting the controller to the Arduino, however, the wireless controller uses a 2.4 GHz frequency radio transmission to exchange data between the two components. The interfacing for both variations is the same and the functionality of the 9 wires and/or pins is mentioned below [39] (Only 6 out of these 9 pins are required for the communication which are marked with an asterisk):

Figure 94 Wireless PS2 Controller

1. Data* : Carries data from the Controller to the Arduino
2. Command* : Carries command data from the Arduino to the Controller
3. Vibration Motors Power : +7V 300mA power source for the vibration motors on the controller
4. Ground*
5. System Power* : +3.6V power source for the module directed from the Arduino
6. Attention* : This line must be pulled low before each group of bytes is sent / received, and then set high again afterwards. Some sources [39, 40] consider this a 'Chip Select' line that is used to address different controllers on the same bus.
7. Clock* : The clock rate is 250 kHz when the controller is connected from the Arduino
8. Unknown
9. Acknowledge : The Arduino will consider the Controller missing if it does not receive an Acknowledge signal within 100 µs.

4.1.1.1.2 Software Protocol

1. Low-Level : How bytes and packets are transferred

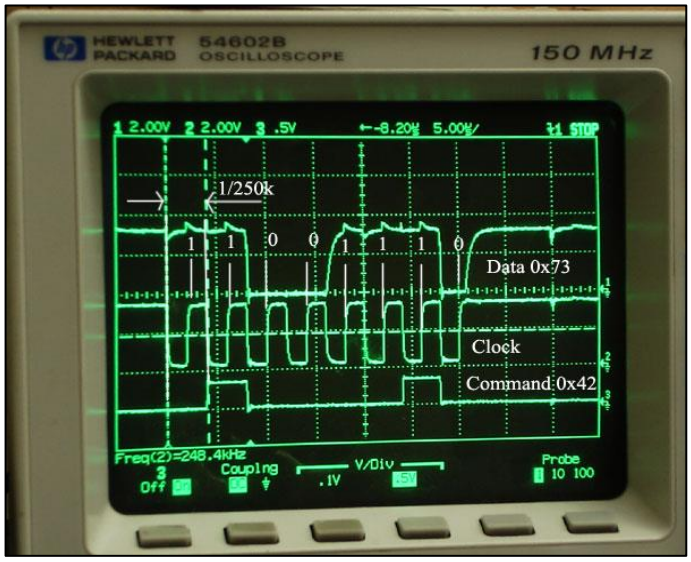


Figure 95 Oscilloscope displaying 3 different signals

The Arduino (or usually, the PlayStation) sends and receives a byte simultaneously via serial communication. The adjacent picture shows the data, clock, and command signals displayed on an oscilloscope between a PlayStation and guitar-hero controller configured in analog mode. The clock signal is maintained at a high value (1) and drops to a low value (0) when a byte is to be sent out. It thus starts 8 cycles of 1s and 0s during which data is transmitted and received at the same time. When the clock edge drops to low, the values of other signals start to change, and these values are read when the clock value goes from low to high. Bytes are transferred LSB (Least Significant Bit) first, so the bits on the left (earlier in time) are less significant.

2. High-Level : Packet Structure, Command, and Data Meanings

According to Dowty [39, 41], packets have a 3 byte header followed by an additional 2, 6 or 18 bytes of command and controller data like button states and vibration motor commands. For a simple example, the signal from a controller when it is first plugged in is analyzed and explained. The controller in this stage is in digital mode by default and only transmits the on and off status signals of the buttons pressed in the 4^{th} and 5^{th} bytes. It does not transmit any joystick data or motor vibration data at this stage

and does not transmit any more bytes. When no buttons are pressed, these are the command and data signals that it transmits:

Here, the first 3 bytes belong to the header, which is followed by 2 to 18 bytes depending on the mode of the controller. The Command and Data signals follow the given protocol.

byte #	1	2	3	4	5
Command	0x01	0x42	0x00	0x00	0x00
Data	0xFF	0x41	0x5A	0xFF	0xFF

Figure 96 Byte and packets

For the Command signal, new packets always start with 0x01 and thus byte #1 has the value 0x01. The second byte can hold a value ranging from 0x41 to 0x4F to configure or poll the controller. Here, the Command is 0x42 which is called the 'Main polling command'. It can get all the digital and analog button states, as well as control the vibration motors. Lastly, the third byte of the command signal is always 0x00. After this, the header has ended, and the remaining bytes can be configured to control any of the motors.

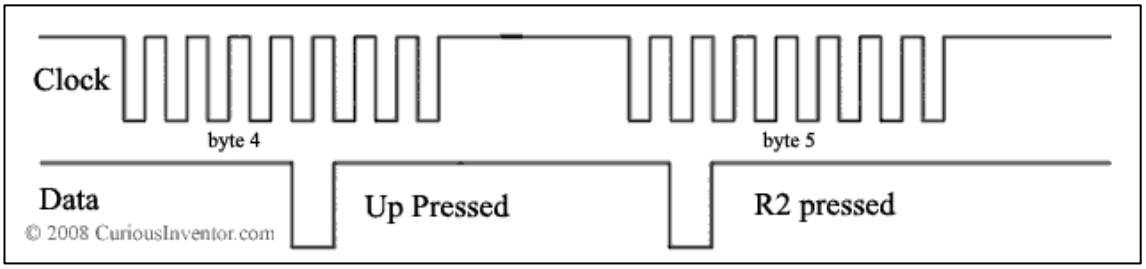
The Data signal always has its first byte with the value 0xFF. The next byte determines the Device Mode. The high nibble (4) indicates the mode where 0x4 is digital, 0x7 is analog, and 0xF is config and the lower nibble (1) is the number of 16-bit words following the header. And lastly, the third byte is always 0x5A. After the header is over, each digital (on/off) button state is mapped to one of the bits in the 4th and 5th byte. In the given example, 0xFF (or 1) means that all buttons are un-pressed.

4.1.1.1.3 Digital Button Mapping

Button	Select	L3	R3	Start	Up	Right	Down	Left
byte.bit	4.0	4.1	4.2	4.3	4.4	4.5	4.6	4.7
Button	L2	R2	L1	R1	△	O	X	□
byte.bit	5.0	5.1	5.2	5.3	5.4	5.5	5.6	5.7

Table 6 Buttons and corresponding byte

Table 6 clearly shows the individual bits assigned to each of the digital buttons on the PS2 controller. Thus, as an example, if the Clock and Data signals for the 4th and 5th byte look like the following graphs, then it implies that the 'Up' and 'R2' buttons are pressed since the Data signal dips at bits 4.4 and 5.1.



The entire list of commands can be found at [39, 42]. However, programming the Arduino to work with the PS2 controller with these rudimentary commands will make the task at hand much more difficult than it should be. Thus, there are libraries that are built to assist users like the project team to simplify the process of writing such programs to make the robot perform as required via the PS2 controller, which will be discussed in a later chapter.

4.1.1.2 Motor Shield

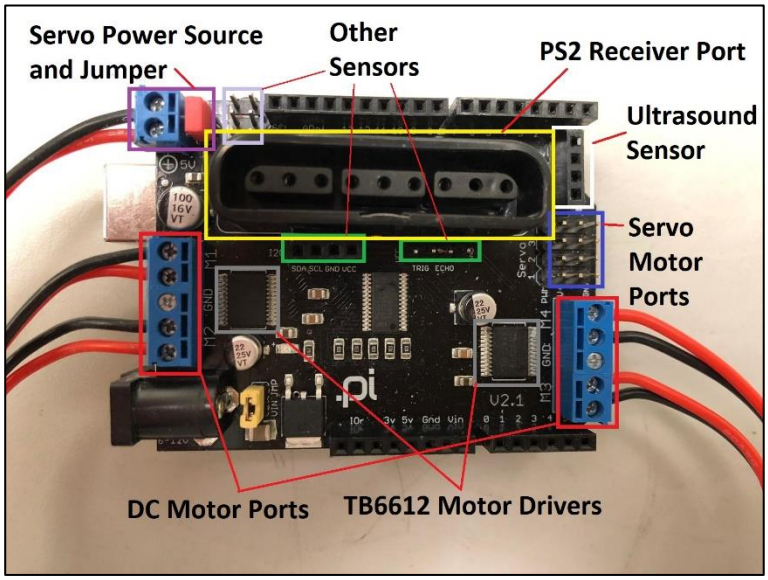


Figure 97 Adafruit motor-shield knock-off version

The motor shield (Figure 97) used for the robot to drive the 8 motors on the machine is attached on top of the Arduino Uno board and is a knock-off of the Adafruit motor-shield. Adafruit Industries is an open-source hardware company based in New York City but they also provide the software required for enthusiasts and students to use along with their chips. The PS2 module that was purchased by the team comes equipped with this shield which can control up to 4 bidirectional DC motors and 4 servo motors. The main difference from other modules to note is that instead of using the L293D motor driver circuit as usual, this motor-shield uses TB6612 MOSFET drivers from Toshiba which makes it possible to have much lower voltage drop across the motors during operation which further implies that the motor can provide more torque from the batteries as compared to the L293D drivers [43]. It can provide an average output current of 1.2A and peak current of 3.2A. The complete data-sheet with detailed technical specifications of this motor driver chip can be found on the official Adafruit website [43]. Adafruit has also provided a library to make programming this chip faster, and more user-friendly. Using this library will make the entire process of setting up motors for our Arduino Uno and also assigning parameters like speed and direction very easy.

4.1.2 Raspberry Pi

Figure 98 Raspberry Pi 3 Model B+

The Raspberry Pi is a series of small single board computers; the latest model is the Raspberry Pi 3 Model B+ (Figure 98). This board is much more powerful than the Arduino Uno and has numerous added I/O interfaces and chips. It uses the Broadcom BCM2837B0 SoC (System on a Chip) with a 1.4 GHz, 64 bit quad-core, ARM Cortex-A53 processor. It comes with 1GB RAM, in-built wireless LAN and Bluetooth capability, an ethernet port, an HDMI port, 4 USB ports, as well as a headphone jack. Unlike the Arduino boards, Raspberry Pi needs to have an operating system installed on its SD card. It supports a wide range of Operating Systems and the most commonly used are the 'Raspbian OS' and 'Ubuntu Mate' which is a lighter version of Linux made specifically for Raspberry Pi. It also includes a Camera Serial Interface (CSI) for connecting a vision sensor to it for video input and a Graphics Processing Unit (GPU) to run functions such as image processing. Technical Specifications of this model can be found in the Appendix XII as well.

In the first stage of the project, the team chooses to utilize an Arduino Uno because it is easier to use and serves the purpose of controlling the robot movement, and unlike a Raspberry Pi, it does not require to load an entire operating system before performing any task. Thus, the booting time is minimal. However, for the later stages where image processing is required, the team added a Raspberry Pi board to the robot.

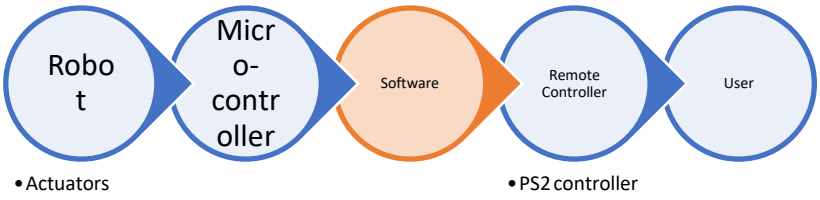
4.1.2.1 Raspberry Pi Camera Module V2

Figure 99 Raspberry Pi Camera Module V2

The team will need an additional camera module to go with the raspberry pi and the camera module V2 shown in Figure 99 is one of the many peripherals available on the official raspberry pi website [44]. It has a Sony IMX219 8-megapixel sensor, the technical details of which can be obtained from Sony's official website, that supports 1080p resolution at 30 fps, 720p resolution at 60 fps, and VGA90 video mode. It attaches via a 15cm ribbon cable to the Camera Serial Interface (CSI) port on the Raspberry Pi.

4.2 Introduction to Arduino Software

Software plays the role of a middleman between the user and the Robot. The movement of a robot as an entire system can be broken down into the movement of each individual actuator which is connected to the microcontroller via the circuitry mentioned in the previous parts. This microcontroller also runs a set of code (also referred to as program or software in this report) which helps this component to interact with the user via the remote-controller. This flow can be visualized from the diagram below representing how software is a bridge between the robot and its microcontroller, and the user and its remote-controller.



Now that all information regarding the hardware, electronic components, circuitry, and the microcontrollers and their modules relevant to this project has been established and explained in detail, it is easier to understand the software component of this project.

4.3 Problem Definition – Arduino

Based on the rules of the ASME student design competition (Appendix I), and the design of our robot which has been critically discussed in Chapter 3, a program is to be written which can aid the user to perfectly control the robot as a whole. The robot movements must be broken down into individual actuator movements (4 DC motors for controlling the mecanum wheels, and 4 servo motors for controlling the gripper motion, and gates), and the program must coordinate the movement of these actuators, and their speeds based on the user input from the wireless controller. This piece of code must be easy to alter in case the robot does not perform optimally, also be completely debugged as there is no room for errors while performing at the competition.

4.4 Methodology and Approach

As time was limited, the team needed to strategically tackle the problem instead of simply trying out solutions that may or may not work. Thus, the 2 main strategies used by the team to efficiently develop a program with minimal wastage of time, as well as their implementation in the project are discussed below:

4.4.1 Iterative Strategy

When it comes to automatic design, there are many elements to take into consideration. Firstly, a configurable or evolvable microcontroller is needed, which the team has already obtained, on which an algorithm that best fits the configuration of the remote-controller (PS2 in this case) is applied. A mapping can be defined for the remote controller based on the input from the person controlling the robot, about how frequently does a button need to be pressed, which correlates to how frequently a certain task is performed by the robot. There are numerous combinations of configuring a 16-button, 2 joystick gamepad to control a total of 8 motors. The team mapped these buttons based on the default controls of a video game while adding the extra tasks to the unused buttons. The operator then tries to control the robot and requests changes in the configuration or the speed of the motors to optimize the efficiency and speed of operation and minimize human errors.

Furthermore, the code is also tested, and the performance is evaluated by timing how fast the robot can pick up all 16 balls on the play-area and deploy them. Alterations are made if the robot misbehaves and tests keep on producing better results. The team decided that the robot control and code was fully optimized when the results became constant. The time-sheet for different trail runs can be found in Chapter 8.

4.4.2 Modular Strategy

The idea is to decompose the whole task into smaller primitives and take care of a simpler and/or partial version of the task and gradually achieving the goal. Furthermore, complexity can also exist in the machine itself when its actuators need to be controlled together in a specific way to achieve a primitive behavior [45]. Each task can be treated

as a block which can be replicated to achieve similar tasks. Blocks can be added, removed, or modified without affecting the overall performance and output of the program. For example, consider a task as simple as going from point A to point B; running a DC motor in the forward direction at a specific speed can be treated as one block, which is copied four times for controlling the 4 DC motors to achieve a forward motion of the robot. The entire program structure for the Arduino board is broken down into such blocks and is represented in the form of a flowchart in Section 5.1 Program Structure.

In Renzo De Nardi et al. [46] a similar modular structure is studied for a miniature helicopter where the sensor inputs values such as position, velocity, rotation and rotational speed along the three axes for a total of twelve values and the four actuators control the rotor speeds and blade inclinations. To stabilize the helicopter and make it able to move forward these actuator needs to be operated in a given, coordinated way, while the other configurations would make it unstable and unable to move along the path. One of the working approaches found in the paper was to split the control component of the machine into four simpler modular networks, each controlling one actuator and using only the needed subset of inputs. The ASME team thought this was a great method to tackle the given problem and decided to adopt this strategy.

4.5 Mecanum Wheels Vector Analysis

Before designing the flow of the program, it is important to understand the behavior of the robot as a whole when different inputs are provided for the DC Motors powering the mecanum wheels. This can be achieved by conducting vector analysis for the mecanum wheels.

By adopting the O-configuration for mecanum wheels as mentioned in Section 3.3.2.2 Mecanum Drive, the vectors produced by each wheel individually, if all motors rotate in the forward direction, is depicted in Figure 100.

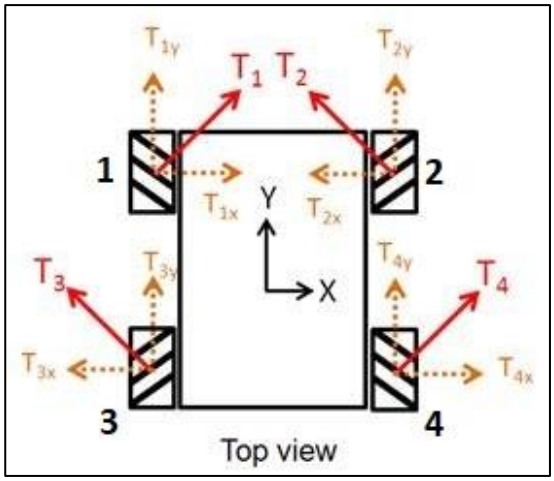


Figure 100 Mecanum wheel vectors

If the motors rotate in the backwards direction, then the vector components will have a negative value to indicate motion in the opposite direction.

The different combinations of wheel motions are analyzed to show the resultant platform motion direction.

(i) For forward translation:

Motor	Rotation Direction	Vector Components
1	Forward	T_{1x} (→) + T_{1y} (↑)
2	Forward	T_{2x} (←) + T_{2y} (↑)
3	Forward	T_{3x} (←) + T_{3y} (↑)
4	Forward	T_{4x} (→) + T_{4y} (↑)

Vector Addition:

$T_{1x} + T_{1y} + T_{2x} + T_{2y} + T_{3x} + T_{3y} + T_{4x} + T_{4y} = T_{1y}$ (↑) + T_{2y} (↑) + T_{3y} (↑) + T_{4y} (↑)

Similarly, when motors rotate in the opposite direction as compared to (i), the platform will translate in the backward direction.

(ii) For left translation:

Motor	Rotation Direction	Vector Components
1	Backward	$- T_{1x} (\leftarrow) - T_{1y} (\downarrow)$
2	Forward	$T_{2x} (\leftarrow) + T_{2y} (\uparrow)$
3	Forward	$T_{3x} (\leftarrow) + T_{3y} (\uparrow)$
4	Backward	$- T_{4x} (\leftarrow) - T_{4y} (\downarrow)$

Vector Addition:

$$- T_{1x} - T_{1y} + T_{2x} + T_{2y} + T_{3x} + T_{3y} - T_{4x} - T_{4y} = - T_{1x} (\leftarrow) + T_{2x} (\leftarrow) + T_{3x} (\leftarrow) - T_{4x} (\leftarrow)$$

Similarly, when motors rotate in the opposite direction as compared to (ii), the platform will translate in the right direction.

(iii) For clockwise rotation:

Motor	Rotation Direction	Vector Components
1	Forward	$T_{1x} (\rightarrow) + T_{1y} (\uparrow)$
2	Backward	$- T_{2x} (\rightarrow) - T_{2y} (\downarrow)$
3	Forward	$T_{3x} (\leftarrow) + T_{3y} (\uparrow)$
4	Backward	$- T_{4x} (\leftarrow) - T_{4y} (\downarrow)$

Vector Addition:

$$[(T_{1y} + T_{3y})(\uparrow) + (- T_{2y} - T_{4y})(\downarrow)] (\circlearrowright) + [(T_{1x} - T_{2x})(\rightarrow) + (T_{3x} - T_{4x})(\leftarrow)](\circlearrowright)$$

In this case, the opposite vectors do not cancel out as they are equally spaced from the center-point of the platform. Thus, they provide a rotational motion about its central axis.

Similarly, when motors rotate in the opposite direction as compared to (iii), the platform will rotate in the anti-clockwise direction.

Translation motion in the diagonal directions can also be achieved by using different combinations of motor control as shown in the Appendix IV.

Chapter 5. The Arduino Code

5.1 Program Structure

At this point, all the information regarding the Arduino Uno board as well as the mecanum wheels has been established and based on the methodology and approach to solve this problem discussed previously, the team can proceed to develop the Arduino code. A program structure in the form of a flowchart is displayed on the following pages. The flowchart has been split into sections for clarity and the purple nodes labelled with alphabets represent the connectors that maintain the flow.

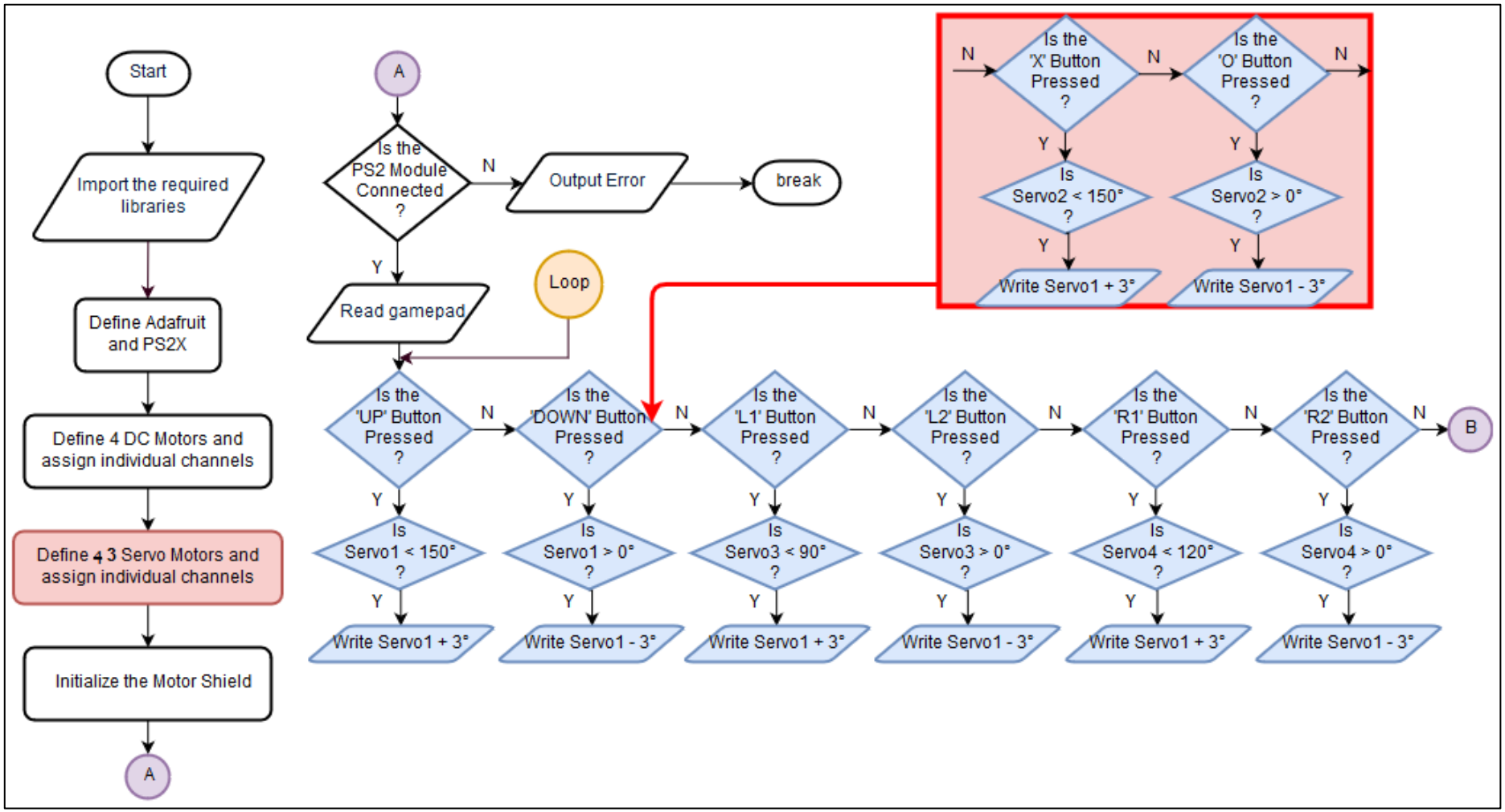


Figure 101 Arduino Flowchart

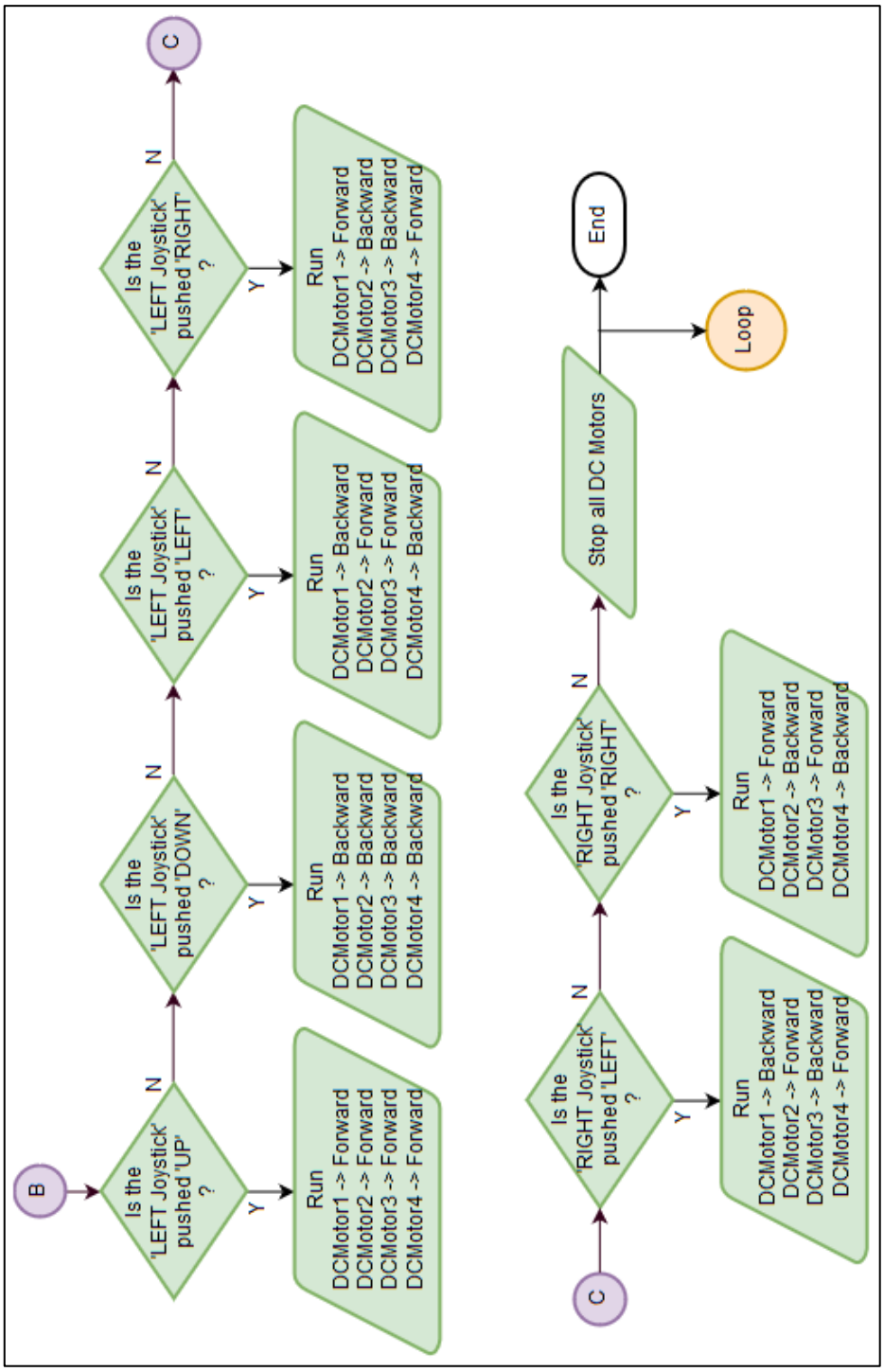


Figure 102 Arduino Flowchart Continued

This flowchart also demonstrates clearly the implementation of the modular strategy where the blue blocks undertake servo motor control whereas the green blocks relate to the DC motor control.

In addition, as discussed in the evaluation section 3.8.3 Circuitry Evaluation and Modification, instead of 4 servo motors connected to the Arduino Uno, one of them was removed and connected to a separate circuit. During the initial stages of development of the program, it was made to work with 4 servo motors. After making the decision to change the circuit, the blocks in the red box that represented Servo2 were simply removed and 3 motors were initialized instead of 4. This truly demonstrates the benefits of the Modular approach where every component of the solution is more independent and does not greatly affect the rest of the solution in case of its failure. Once the flow has been visualized, it is easier to write the program.

5.2 About the Program

5.2.1 Language

The program is written in C++ in the Arduino IDE (Version 1.8.8) which allows a one-click upload of the code to the Arduino board. Just like most programming platforms, the Arduino environment can also be extended through the use of libraries that provide added functionality for working with hardware or manipulating data. These libraries consist of functions which help to control the Arduino board and perform computations whenever required. Thus, the essential knowledge about C++ programming obtained in the ENG2002 course will be put to use here to kickstart this component of the project.

5.2.2 Libraries

There are 3 main libraries used in the program (apart from a few other secondary libraries that need to be included as well) and they are:

1. Adafruit Motor Shield V2 Arduino Library

The first library that we make use of in our program is made available for use to the public by Adafruit [47]. It contains classes and functions that let the motor-shield control and keep state of DC motors as well as Stepper motors. The original version of this library does not include support for servo motors, but the team uses a modified version with added capability to implement usage of servo motors as well. Using this library will make the entire process of setting parameters like speed and direction of the motors very easy.

2. Adafruit PWM Servo Driver Arduino Library

This library is imported as a support for the previous library. Being a modified version, it cannot control Servo motors all by itself and needs the Adafruit PWM Servo Driver Arduino Library [48] to be imported into the program as well. It lets the board store the state and function for interacting with the PWM chip and is mainly utilized for handling servo motors.

3. PS2X Arduino Library

This library is made available by Bill Porter [49], an engineer and a hobbyist, and it is used to interface a PlayStation 2 Controller with Arduino Uno. This library tackles the problem of interfacing and includes pre-defined button constants while assigning the respective pins to

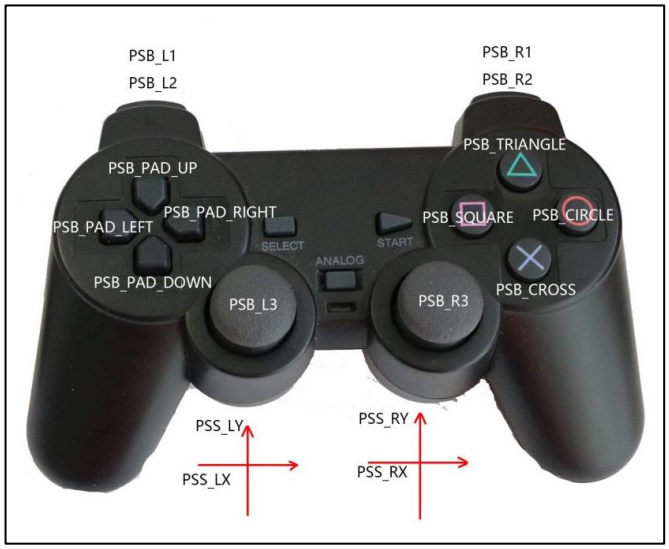


Figure 103 PS2X library button mapping

them. It also includes functions that let the user read and configure the gamepad and get an input of whether a button is being pressed or released, amongst many other features. The pre-defined button constants are depicted in Figure 103. The Joysticks can be configured using the x and y values as shown based on the established coordinate system that ranges from 0 to 255 on each axis.

Another library that is included in the code is the 'wire.h' Arduino library that supports libraries 1 and 2 and enables communication with I2C devices. This is not used first hand in the code but must be included in the header. All the library files mentioned above can be found in the Appendix XIII for the readers reference.

5.2.3 Code Breakdown

In this section, the code is analyzed step-by-step and details about the usage and behavior of important functions are provided [50, 51].

```
1 #include <Wire.h>
2 #include <PS2X_lib.h>
3 #include <Adafruit_MotorShield.h>
4 #include <Adafruit_MS_PWMServoDriver.h>
```

The first four lines of the code are importing the libraries that are discussed above.

```
8 Adafruit_MotorShield AFMS = Adafruit_MotorShield();
```

- **class Adafruit_MotorShield;**

The Adafruit_MotorShield class represents a motor shield and must be instantiated before any DC Motors or Stepper Motors can be used. User must declare one Adafruit_MotorShield for each shield in the system.

Here an Adafruit_MotorShield class has been declared called ‘AFMS’ to further use the functions inside this class by calling the ‘AFMS’ class.

- **Adafruit_MotorShield(uint8_t addr = 0x60);**

This constructor creates the motor-shield object and takes one optional parameter which is an unsigned integer of length 8 bits (uint8_t) to specify the I2C address of the shield. The default address of the constructor (0x60) matches the default address of the boards as shipped. In case there are more than one shield in the system, each address must be unique.

In the same line of code above, we use this constructor without any parameters as we only use one motor-shield and it is automatically assigned as the default shield.

```
9 PS2X ps2x;
```

Similarly, in Line 9, the ‘PS2X’ class has been declared as ‘ps2x’ to use its functions later in the code.

```
10 Adafruit_Servo *Servo1 = AFMS.getServo(1);
11 Adafruit_Servo *Servo3 = AFMS.getServo(3);
12 Adafruit_Servo *Servo4 = AFMS.getServo(4);
13 Adafruit_DCMotor *DCMotor_1 = AFMS.getMotor(1);
14 Adafruit_DCMotor *DCMotor_2 = AFMS.getMotor(2);
15 Adafruit_DCMotor *DCMotor_3 = AFMS.getMotor(3);
16 Adafruit_DCMotor *DCMotor_4 = AFMS.getMotor(4);
```

- **Adafruit_DCMotor *getMotor(uint8_t n);**

This function returns a pointer to one of 4 pre-defined DCMotor objects that are already allocated, controlled by the shield. It essentially initializes the DC Motor and

turns off all its pins. It accepts a ‘unit8_t’ parameter to specify the associated motor channel ranging from 1to 4.

In lines 13 to 16, 4 DC motor objects are defined, and it assigns 4 motor channels to those defined objects.

- **Adafruit_Servo *getServo(uint8_t n);**

The getServo function works comparably to the previous function, the only difference being that it initializes Servo motors instead of DC motors and in lines 10 to 12, three Servo motors are defined and assigned individual channels.

```
18 void setup()
```

```
35 void loop()
```

The Arduino sketch is broken down into 2 major functions, which contain the other functions or tasks. All code written within the setup() function will run only once upon powerup or reset of the Arduino board. It is usually used to initialize variables, pin modes, and start using libraries. In this case, an error functionality is also added here. On the other hand, all code written in the loop() function will continuously repeat in loops allowing the program to change and respond to actively control the Arduino board.

```
20   AFMS.begin(50);
```

- **void begin(uint16_t freq);**

Next, within the setup function, begin() must be called to initialize the I2C hardware and PWM driver on the shield and then turn off all pins. An optional frequency parameter can be used for speed control and for this code we set it at 50Hz. This is also where the wire.h library comes into play on the backend.

```
22  int error = 0;
23  do{
24    error = ps2x.config_gamepad(13,11,10,12, true, true);
25    if(error == 0){
26      break;
27    }else{
28      delay(100);
29    }
30  }while(1);
```

Further in the setup() function, an error variable is initialized to 0 and the algorithm to check whether or not the PS2 module is connected is placed inside a do-while loop. Line 30 indicates that it is a never-ending loop, and thus, continuously checks for the PS2 module connection at intervals of 100 milliseconds (as implied by Line 28). In Line 24, ps2x.config_gamepad puts it in analog mode and takes parameters which are the pin numbers for the clock, command, attention, and data pins that were discussed in Section 4.1.1.1.1 Hardware Interface / Wiring Connections. This changes the value of the error variable from 0 when the module is connected; thus, if its value goes back to 0, it implies that the module is no longer connected and there is an error. The program then ends to allow the user to reboot and/or reconnect.

```
39  if (ps2x.Button(PSB_PAD_UP)) {
40    if (Servo1->readDegrees() < 150) {
41      Servo1->writeServo((Servo1->readDegrees() + 3));delay(1);
42    }
43  }
```

The program now enters the main code in the loop function. The block of code above performs the task of rotating Servo1 by 3 degrees at each interval if the ‘UP’ button is pressed and the current angle is less than 150 degrees.

- **boolean Button(uint16_t);**

Button() takes in the concerned button keyword as a parameter and returns true if that button is being pressed. Here, the first condition is to check of the ‘UP’ button is being pressed. If true, then it moves on to the nested ‘if’ to check the second condition.

- **uint8_t readDegrees( );**

This function simply returns the current angle of the Servo that it is associated with. In Line 40 it is used to check the second condition if the angle of Servo1 is less than 150 degrees.

- **void writeServo(uint8_t angle);**

Lastly, if both conditions are met, the board executes the task of rotating the servo. writeServo() uses an integer as the input parameter and changes the current angle of the servo motor to the one specified. Here, it writes a new value to Servo1 which is its current value added to 3. This achieves motor rotation by 3 degrees as long as the button is being pressed.

Similarly, all the other buttons mapped for the 3 servo motors can be used to execute the tasks as mentioned in the flow (Figure 101). The same structure can be used and put in several different ‘else if’ statements with the second conditions for each case within nested ‘if’s.

```
70   } else if (ps2x.Analog(PSS_LY) < 10) {
71     DCMotor_1->setSpeed(100);
72     DCMotor_1->run(FORWARD);
73     DCMotor_2->setSpeed(100);
74     DCMotor_2->run(FORWARD);
75     DCMotor_3->setSpeed(100);
76     DCMotor_3->run(FORWARD);
77     DCMotor_4->setSpeed(100);
78     DCMotor_4->run(FORWARD);
```

Next, the DC Motor movements are programmed, and the example above shows that when the LEFT Joystick is pushed UP, all 4 DC Motors move forwards at a fixed speed.

- **byte Analog(byte);**

ps2x.Analog() accepts a parameter to know which Joystick and which axis the program requires. Left Joystick, Y-axis in this case. It returns the current value ranging from 0 to 255. Here, if it is less than 10, it means that the Joystick is pushed upwards and according to the controls, the robot must move forward.

- **void run(uint8_t);**

The run() function controls the motor state : direction, and action. It accepts one of 3 parameters:

- FORWARD - Rotate in a forward direction
- BACKWARD - Rotate in the reverse direction
- RELEASE - Stop rotation

The Forward and Backward directions are arbitrary and if the motors move in opposite directions than anticipated, one can simply switch the motor leads on the circuit.

- **void setSpeed(uint8_t);**

The setSpeed() function controls the DC motor speed/throttle by varying the power level delivered to the motor. The speed parameter is a value between 0 and 255 and it simply controls the power delivered and not the speed itself. The actual speed of the motor will depend on several factors, including: The motor, the power supply and the load. All the motors are assigned a speed value of 100 which the team found to be ideal

for the competition. Any value lower than 100 was too slow and the machine would not stand a chance in the competition where speed is crucial to win against the other robots. Also, any value higher than that made it extremely difficult for the team-member to control using the controller. It also resulted in negative points because of poor and hasty control of the robot.

Thus, according to the flow, based on different joystick inputs, the motors can be assigned to move in different directions as required by simply using the same structure in another 'else-if' block.

```
124    else {
125      DCMotor_1->setSpeed(0);
126      DCMotor_1->run(RELEASE);
127      DCMotor_2->setSpeed(0);
128      DCMotor_2->run(RELEASE);
129      DCMotor_3->setSpeed(0);
130      DCMotor_3->run(RELEASE);
131      DCMotor_4->setSpeed(0);
132      DCMotor_4->run(RELEASE);
133   }
```

After all the movements have been programmed, the last step in the loop() function is to release all motors. When the RELEASE parameter is passed to run(), it cuts all power to the motor and setSpeed(0) implies that the motor is 'off'. This is done so that the Robot stays still until and unless it receives a command from the PS2 Module.

Thus, the program structure, functions information and implementation, and tailoring it to the project requirements and application have been explained in detail and the entire code can be found in the Appendix XIV.

5.2.4 Debugging Process / Difficulties faced

The debugging process for the Arduino program was relatively easy since the code is not too complex and is easy to understand. Initially, the Servo motors would misbehave upon initialization or not work at all; the team traced this problem to the angle limits set on those motors in lines 40, 46, and so on. The motors were then re-calibrated and once the correct limiting angles were entered, the robot worked fine. Another problem faced was that sometimes the program would seemingly skip steps or malfunction. This was because the processing power of an Arduino board is not industrial grade level and thus, it needs some intervals between heavier steps. This problem was solved by adding a delay() function wherever it seemed appropriate in the program. It accepts an integer input which is the number of milliseconds the program must wait before moving on to the next step. Thus, while the program is “waiting”, the board finishes the previous steps and is ready to move further. Apart from such minor issues, the code functioned as required and did not cause many problems during the ASME competition or otherwise.

Chapter 6. Vision Feedback Control System Development

6.1 Introduction to Python Programming

After the ASME student design competition in Vellore, India, the students of the team had limited time to work towards the goal of achieving object detection through image processing for the garbage collection application. The need for automatizing the robot was apparent due to the failure and limitations of the robot in certain aspects due to the lack of a feedback system. At this stage, the robot merely depends on the user input to collect objects and this leaves room for error and inaccuracies. Furthermore, to control the 8 motors on the machine, the controller has an extra board attached to the PS2 gamepad (Figure 104) and it is time-consuming for the user to get familiarized with all the buttons and master the control of this robot which reduces its reproducibility. Automation is needed to reduce the effects of such anomalies in the robot and to make it function more efficiently. The next part of this report starts with exploring the significance and of feedback control systems as well as its implementation in our project.

Figure 104 Robot Remote Controller

6.2 Feedback Control System

A control system is modelled based on the concepts of linear system and feedback control analysis. Understanding signal input-to-output and error cause-to-effect

relationship of the process is critical for obtaining the desired system response. Usually control system can be categorized into two types: i) Open-loop, and ii) Closed-loop control system.

6.2.1 Open-Loop and Closed-Loop Control System

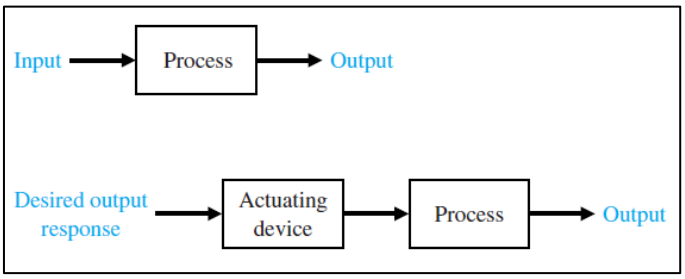


Figure 105 Open-loop Control System Block Diagram

An Open-loop control system can be described as a without-feedback system. It utilizes an actuating device (e.g., Motor) to control the overall robot mechanism and process without receiving or compensating feedback. As shown in the block diagram (Figure 105) [52], the controller simply inputs the signal of desired output response, and this signal will go through the block 'process' to provide output. However, this is not always an optimal control system model for robot manipulation because there are many intrinsic and extrinsic factors disturbing robot-output performance.

Today, modern control systems in both, robotics and various industrial processes including manufacturing and production focus not only on robust and stable control system qualities, but also self-organizing, adapting and learning qualities to reach higher level of automation. For example, some motor driven robots are affected by velocity

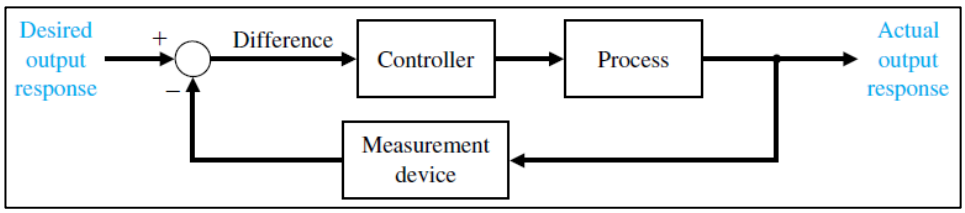


Figure 106 Closed-loop Feedback Control System Block Diagram

difference within each motor due to manufacturer's tolerance, and thus, may not travel in a straight line with forward command. To regulate and compensate the error between actual and desired output, it is necessary to equip the control system with feedback measurement components and controllers. A system dealing with feedback signal can be categorized into Closed-loop feedback control system. In contrast to the Open-loop block diagram, Closed-loop (Figure 106) [52] has additional blocks for 'measurement' (e.g., Output sensors), and 'Controller' that receive error feedback and provide compensation input.

6.2.2 Feedback Measurement Implementation

Integration of mecanum wheels on the robot chassis requires very precise orientation as a small tolerance due to misalignment may lead to malfunction. In the case of our robot and considering our final goal/objective of automation system, image processing with vision sensor is researched and a working proof of concept of a semi-automatic machine has been developed using python on the Raspberry Pi equipped with a camera module. In addition, infrared sensor, a type of range sensors, was researched as backup plan to deal with the potential issue of the original idea not working as planned. Of course, this has not been developed on the robot as the controller has no difficulties in position control as of writing this report. Both ideas have been discussed below to form a base for writing the python program for dynamic object detection.

6.2.2.1 Range Sensors

Range sensors typically have 3 ways of measuring distance i) Time of Flight (TOF) ii) Return Signal Strength iii) Triangulation. FC51 IR (Infrared) sensors consist of two components including a source emitting IR LED, which is invisible to human eyes, and a detector with photodiodes, receiving signals reflected from distanced object surfaces. The output signal, with the Return Signal Strength measuring theory, is defined based on the intensity of reflected signals. This IR sensor is normally used for short distance measurement depending on operating voltage. Figure 107 [53] illustrates the basic working principle of IR sensor.

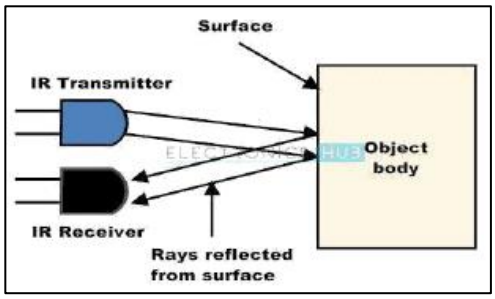
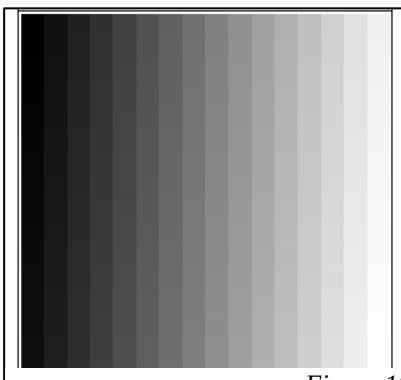


Figure 107 Working Principle of IR Sensor

6.2.2.2 Image Processing

To understand the research on image processing with vision sensors, fundamental understanding about image in mathematical representation is required. In ME42011 Fundamentals of Robotics, it was taught that an image captured and displayed on the computer is a mathematically describable matrix of either grayscale or BGR (3D matrix) intensities. This matrix size depends on performance quality (e.g., Pixel) of the image



0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245
6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251
12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252
13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253

Figure 108 Matrix Representation of a Grayscale Image

sensor used, and each number represents the intensity of color of light collected ranging from 1 (Dark) to 255 (Bright) as shown in Figure 108 [54]. With this information, programmers can design advanced software systems with numerous functions like object detection or even motion detection. In this project, as the purpose of using this vision sensor is to simply collect the feedback information, compare for error verification, and compensate for appropriate configuration between the robot and the container, a sticker with specific colour (fluorescent orange in our case) and shape (plus sign) can be pasted on each container as a reference, and a program thresholding the colour or detecting the shape edge can be developed. Eventually, the robot can learn its current location relative to the container and adjust itself without the need of any additional sensors, making the system simpler with fewer components performing the same task and avoiding any redundancies.

6.3 Problem Definition – Python

The very first foreseeable problem is positioning and orientation of the robot relative to the container to initiate grasping mechanism. In fact, this issue was also a minor problem that was experienced in the ASME competition where our robot position relative to the ball should be in a line to ensure safe grasping process without ball falling off from the 20cm-pole. If the garbage bin is misplaced by a small distance from the point of collection as illustrated in Figure 109, the robot

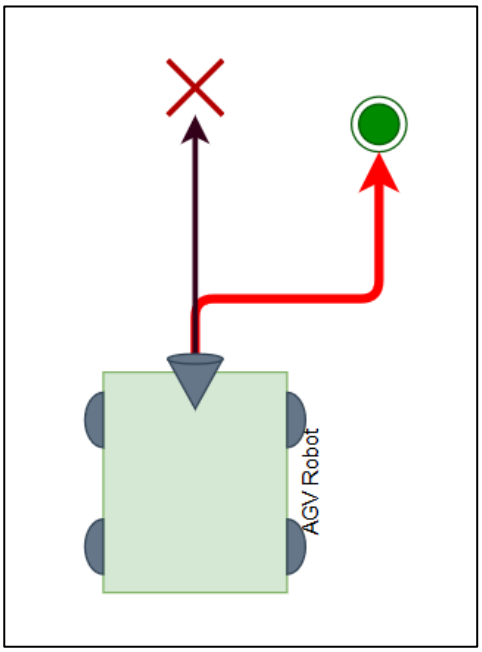


Figure 109 Rerouting with image

must be able to identify the target garbage bin (Green circle) and reroute to move to the new location of the bin, align itself with the target, and pick up the garbage. Without this function, the practicality of this project reduces greatly, and the robot can only function when there is zero error by the human while placing the garbage bin at the collection point, which will rarely be the case.

6.4 Methodology and Approach

The strategy used to approach this component is similar to the strategy mentioned in Section 4.4 Methodology and Approach. The modular strategy has proved to be beneficial in the first phase of this project and the team has experienced how that approach can save time and enhance the efficiency for coding a program. The python program is expected to be heavier than the previous one and thus, the importance of using this strategy only increases. First, a flowchart is designed based on the problem definition and knowledge attained about the various python libraries. This flowchart can be a very primitive version of the original flow, since unlike the Arduino program, this one cannot be entirely laid out in a chart form due to the fact that there are numerous ways to solve any problem using python and each person may use slightly different approaches to tackle the same problem. Thus, the base strategy/ideal to stick to is modularity. The Iterative approach mentioned is also employed here as several iterations of the same solution to any problem were tried and tested before the final version of the code was realized. This ensures that the program is the most efficient version which tackles more problems than previously envisioned. This will be made evident in future parts of this report.

Chapter 7. The Python Code

7.1 Program Structure

Based on the problem definition and the methodology discussed, the team can start visualizing the program structure in the form of a flowchart. This may not include the excruciating details of each step, but rather, gives the team a direction to move forward with the progress of the program. The program structure in the form of a flowchart is displayed on the following page.

The purple circles are connectors that maintain the continuity of the flow. The modularity of the code is shown by the blue blocks that are responsible for positioning the robot with its target right at the centre; and the teal blocks that make sure the robot is at the right distance from the target before executing further tasks. The program is within a continuous loop as shown. There will be modifications made to this flow further in the chapter and the new blocks will be added as indicated. The new code will also repeat the orange blocks again but with separate, new values.

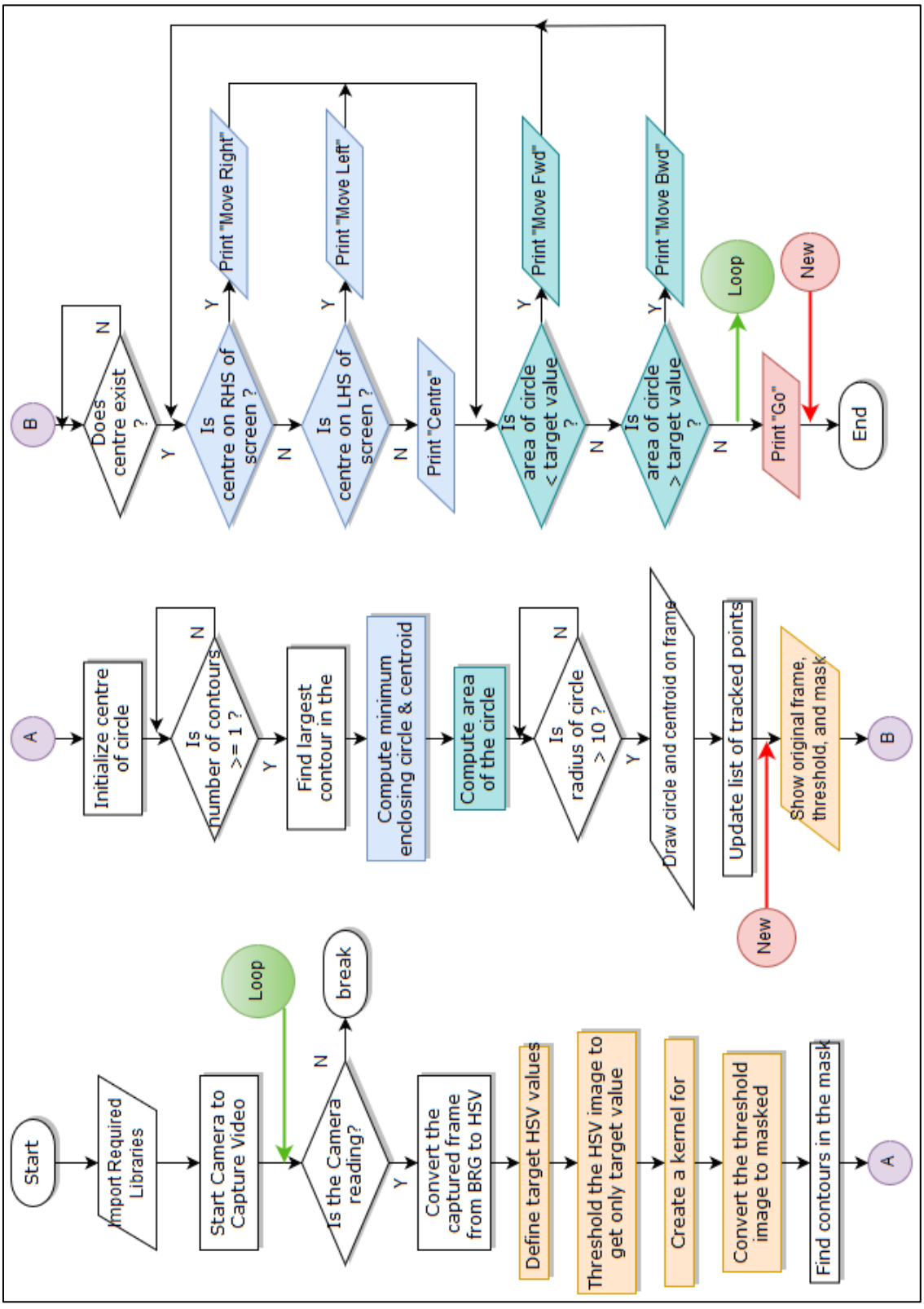


Figure 110 Program Structure for Python

Thus, the structure and flow of the python program has been established and we can move on to explaining the logic and the code.

7.2 About the Program

7.2.1 Libraries

A. OpenCV

OpenCV is one of the most commonly used tools for various computer vision experiments. It is an open source library consisting over 500 functions that can be ran with different programming languages including C++ and Python. the fundamentals of OpenCV library and colour thresholding detection theory as well as its methodology were researched and are explained to better understand the code.

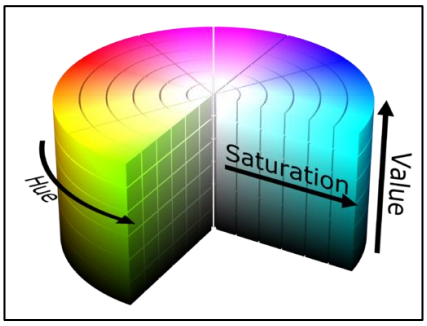


Figure 111 HSV Color Space Range

In order to threshold a specific colour from an image or video, BGR (Blue, Green and Red) colour space matrix should be converted to HSV colour space, which is more intuitive to how humans experience colour. HSV represents HUE (Colour, angle), SATURATION (Vibrancy, radius) and VALUE (Brightness, height); where each value ranges from 0 to 255 as shown in Figure 111 [55]. Some advantages of using HSV space in colour detection purpose are i) Easy to convert from BGR space ii) Simple HUE filtering enables thresholding of specified colour iii) Greatly decreased size of colour information of an image.

For the case of blue object detection with a laptop camera via simple HUE filtering method (Appendix XV), a typical HUE range for blue colour ’75 – 130’ was set. The SATURATION and VALUE depends on the room or environmental lightening condition as well as the surface of the object. For simulation in a bright room without sunshine and a cotton material object, these SATURATION and VALUE numbers were set to ‘200 – 255’ and ’60 – 255’ respectively by trial and error. The result is shown in Figure 112.

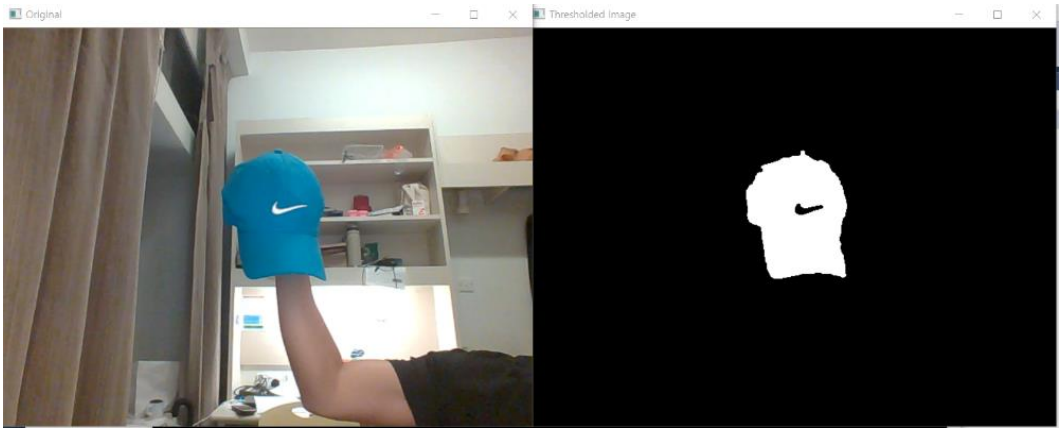


Figure 112 HUE filtering program for blue object detection

The result clearly shows the blue cap as a white patch in 'Threshold image' window while other colours, which are not essential, are converted into black. This HUE filtering method introduced positive potential in capturing the sticker on the container to develop a closed-loop control system.

B. NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object, sophisticated math functions, and useful linear algebra, Fourier transform, and random number capabilities. It is an open source library made available by [56]. We implement this library due to the requirement of using arrays as a kernel for masking operations on the pre-processed images.

7.2.2 Python Code Breakdown

In this section, the code is analyzed step-by-step and details about the usage and behavior of important functions are provided as well as explanations about any algorithms wherever required. The entire code is shown in Appendix

```
1    import cv2
2    import numpy as np
```

First, both the libraries used in this program are imported. Numpy is imported as ‘np’ simply for not having to type a long name every time we want to use it.

```
6    camera = cv2.VideoCapture(0)
```

To capture a video, we must create a VideoCapture object. Its argument can be either the device index or the name of a video file that needs to be read. The device index is just the number to specify which camera must the command fetch the information from. Normally when only one camera is connected (as in this case), simply pass 0 (or -1) as the argument and the camera is automatically set as the default camera. After that, we can capture any video frame-by-frame to further process these individual frames. At the end of the program, the code must also release the capture.

```
9        ret, image = camera.read()
10
11       if not ret:
12           break
```

camera.read() returns a boolean value where if the frame is read correctly, it will return true which enables us to check end of the video by checking this return value. Sometimes, the camera may not have initialized the capture, in which case, this code shows an error and breaks. It can easily be checked by calling the method camera.isOpened(). If it is true, then the code is good to move forward. Otherwise we need to open it using camera.open().

```
14          frame_to_thresh = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

There are more than 150 color-space conversion methods available in OpenCV. For color conversion, we use the function ‘cv2.cvtColor(input_image, flag)’ where ‘flag’ determines the type of conversion.

We are using the BGR to HSV color-space conversion with the flag ‘cv2.COLOR_BGR2HSV’ to convert the image from Blue-Green-Red to its equivalent Hue-Saturation-Value version and store it in ‘frame_to_thresh’.

```
16          v1_min=0
17          v2_min=144
18          v3_min=155
19          v1_max=18
20          v2_max=227
21          v3_max=220
```

Figure 113 Target Sticker

Here, we define the HSV range for our target, which is the orange sticker pasted on the garbage bin displayed in Figure 113. v1, v2, and v3 correspond to H, S, and V values respectively. This range depends on several factors including the camera quality, environment lighting conditions, as well as surface texture of the target. The team ran another python program found online to find these values for our target sticker and to fine tune it to reduce the noise.

```
31          thresh = cv2.inRange(frame_to_thresh, (v1_min, v2_min, v3_min), (v1_max, v2_max, v3_max))
```

This command changes the HSV image to apply a threshold to it and get only the target value colours. It is easier to understand if we look at the following syntax :

```
threshold = cv2.inRange(image, lower_value, upper_value)
```

Thus, on comparison, it is clear that in our command, we apply the threshold to 'frame_to_thresh' and store it in 'thresh' while provide the lower and upper limits that we defined previously. It converts Figure 113 to Figure 114.

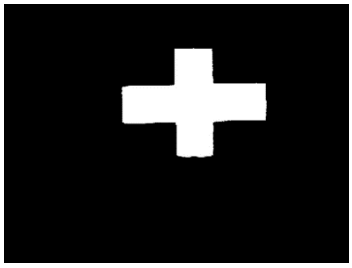


Figure 114 Threshold Image

```
34          kernel = np.ones((5, 5), np.uint8)
```

We need a kernel or a "mask" for our next step to further process the image. np.ones() returns a new array of given shape and type, filled with ones. The input parameters are 'shape' of the matrix (5x5 in our case), and the desired datatype for the array (8-bit integer in our case).

```
35          mask = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
36          mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
```

Next, we perform some morphological transformations on the thresh image to get rid of noise. Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation with its variant forms like Opening, Closing, Gradient etc. also coming into play. We will explore them in an orderly fashion with help of alongside image of the letter 'j' [57]:

Figure 115 Example image for morphological transformation

In Line 35, we have performed ‘Opening’ on the image which is just another name of erosion followed by dilation. It is useful in removing noise from the image and uses the function cv2.morphologyEx(). Here, the white dots, noise, surrounding the target element is eradicated by applying the above command.

Figure 117 Opening Example Results

Similarly, in Line 36, we perform closing on the image which we get as a result of opening. Closing is the reverse of Opening, which means that first the image is dilated followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object as seen in the alongside image.

Figure 116 Closing Example Results

```
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
```

Here, the program finds contours in the mask. The parameters accepted by the cv2.findContours function are the image, mode, and method, in order. The image input here is a copy of our masked image. The mode specifies the mode of the contour retrieval algorithm and cv2.RETR_EXTERNAL retrieves only the extreme outer contours. The method specifies the Contour approximation method. Here, cv2.CHAIN_APPROX_SIMPLE compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points according to the OpenCV documentation.

```
52                  c = max(cnts, key=cv2.contourArea)
53                  ((x, y), radius) = cv2.minEnclosingCircle(c)
```

First, the centre of the minimum enclosing circle is initialized, and if there was at least one contour found, then the program moves on to line 52. Here, the largest contour in the mask is located and stored in ‘c’. This ‘c’ is then used to compute the minimum enclosing circle of the contour which is a circle that completely covers the mask with minimum area. (x, y) are the coordinates of the centre of this circle. The area of this circle can easily be calculated by using the formula area = pi*radius*radius.

```
54                  M = cv2.moments(c)
55                  center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
```

The largest contour ‘c’ can also be used to compute the centroid of the masked object using its moments. The function cv2.moments() gives a list of all moment values calculated and then the centroid is given by the formula Cx=M10/M00 and Cy=M01/M00 which are the x and y coordinates of the centroid. This exact formula is written in python syntax in line 55 and the values are stored in ‘centre’.

If the radius meets a minimum size, then the centroid and minimum enclosing circle are drawn on the frame with some text output specifying the centroid coordinates as well.

```
86              cv2.imshow("Original", image)
87              cv2.imshow("Thresh", thresh)
88              cv2.imshow("Mask", mask)
```

The function cv2.imshow() is used to display an image in a window, which automatically fits to the image size. The first argument is a window name which is a string and the second argument is our image name. Multiple windows can be created as long as

they have different names. An important issue to note here is that after running several versions of this program on the Raspberry Pi, the team found that having more windows displayed for results slows down the board and affects the performance of the code.

```
100             if center[0] in range(360,640):
101                 print("Move Right")
102             elif center[0] in range(0,280):
103                 print("Move Left")
104             else:
105                 print("Centre")
106                 pass
```

The next step is to check whether the centroid of the object detected is on the left or right side relative to the robot, or right in front of it. This is done by comparing the x-coordinate of the centroid with a virtual central Y-axis. We know that the screen width is 640 pixels, which implies that the centre of the screen is at x = 320 pixels. After some trial and error with the robot, we decided to have a range of 80 pixels designated for the centre region. (40 pixels each side of the centre axis). Thus, if the x-coordinate of the centroid lies within x = (0, 280) then the program outputs “Move Left” on the screen telling the operator to move the robot towards its left side. Similarly, if the x-coordinate lies within the range x = (360, 640) then the program outputs “Move Right”. If it lies within the central range of 80 pixels width then the program outputs “Centre” and moves on to the next condition.

```
109             if 100<area<23000:
110                 print ("Move Forward")
111             elif 82000<area<90000:
112                 print ("Move Backward")
113             else :
114                 print("Distance OK")
```

The range of the area value ideal for the robot to continue with gripping the garbage bin was determined by conducting empirical evaluation discussed in detail in Section 8.3.3 Empirical Evaluation to quantify Area value. The logic here is simple: if the area if the minimum enclosing circle is less than the target range, then the program asks the robot to “Move Forward” and vice versa. If the area of minimum enclosing circle is within the target range then it outputs “Distance OK”, after which the program ends, meaning that the robot is at the correct position where the target object is right in front of the robot within its reach and it can go ahead and grip the garbage bin.

7.2.3 Shortfalls of this Version and modification made

The program works smoothly and without any errors, and the robot always reaches the correct position where it can grasp the bin. However, the team realized that there was one major element missing which must be incorporated even though it may not require it at this scale and level of production. The program so far can tell the robot the correct position; however, it cannot tell if the robot is facing the garbage bin head-on or if the garbage bin is tilted at an angle away from the robot. This information is essential at a higher scale of production and must be implemented to provide complete and full information of the robot’s state with respect to the garbage bin. In case the bin is tilted at an angle away, the robot can rotate around an axis in the middle of its front wheels since it is equipped with mecanum wheels. Upon doing so, the robot will finally be at the correct position with the correct orientation. To implement this, a new module is developed which is shown in the flowchart below. Once the orange blocks from ***Figure 110*** have been duplicated for this new module, it can advance to complete the mentioned tasks.

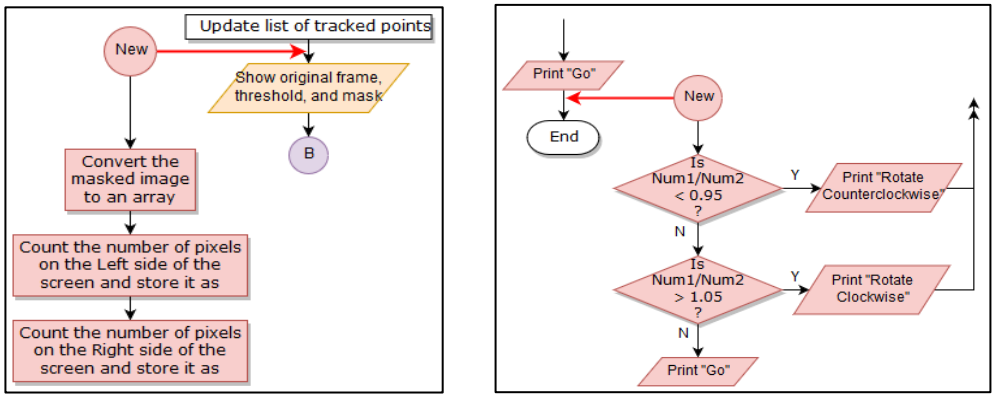


Figure 118 Additional module for the old flowchart

The logic here is that 2 additional stickers of a different colour (blue) are added to the garbage bin as displayed in the following image.

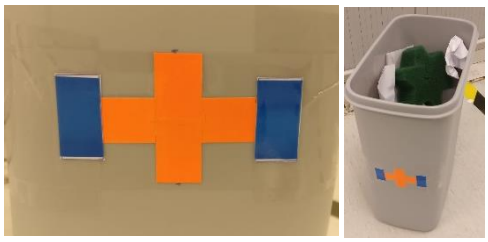


Figure 119 Modified bin sticker

These stickers are of the same size, and in theory, if the robot is perfectly aligned with the bin then the number of pixels on the left side of the screen should be equal to the number of pixels on the right side. If they are not equal then the robot must rotate in the direction which has lesser pixels.

```
70          npImg = np.asarray(mask1)  # No copying takes place      #new
71
72          coordList = np.argwhere(npImg == 255)                    #new
73          numWhitePoints = len(coordList)                          #new
74          num1 = 0                                                 #new
75          num2 = 0                                                 #new
76          for i in range(numWhitePoints):                          #new
77              if coordList[i][1] < 320:                            #new
78                  num1 = num1 + 1                                  #new
79              elif coordList[i][1] > 320:                          #new
80                  # else:                                          #new
81                  num2 = num2 + 1                                  #new
```

Thus, the newer module starts by applying a mask on the blue threshold and in Line 70, this image is stored in npImg in the form of an array. Next, we get a list of points from this array that are non-zero (255) by using the np.argwhere function. The length of this list equals the number of non-zero, or “white” points. After that, we initialize two variables to 0 which will be responsible for counting the number of non-zero points to the left (x-coordinate < 320) and to the right (x-coordinate > 320) where x = 320 is the virtual axis splitting the screen into two halves.

```
116                 if (num1/num2)<0.95:                    #new
117                     print("rotate counterclockwise")    #new
118
119                 elif (num1/num2)>1.05:                  #new
120                     print("rotate clockwise")           #new
121
122                 else:                                   #new
123                     print("GO")                         #new
```

Once we have obtained the values of Num1 and Num2, we can apply the logic shown in the block of code above. We allow a 5% error rate because it is difficult for both the numbers to be equal in practicality. Thus, the condition in Line 116 implies that if Num1 is lesser than Num2 by more than 5%, the robot should “rotate counterclockwise” and from Line 119, if Num1 is more than Num2 by more than 5% then the robot should “rotate clockwise”. However, if the number of pixels are equal on both sides of the screen with an error of +-5% then its GO time and the robot has achieved complete information of it’s state relative to the garbage bin.

Thus, some modifications were successfully made to overcome the shortcomings of the previous version of the code proving once again, that the iterative and modular strategies are very helpful, especially in a project like this one.

7.2.4 Debugging Process

For debugging this python code, the most used tactic was to enter print statements at different points of the code to see where it went wrong. This was especially helpful due to there being numerous loops, and conditions in this program. Thus, if a loop was not logically correct or accurate, then the print statement would should you how the code is being directed and it is much easier to catch a mistake. Another issue was that the similar functions like ‘break’, ‘pass’, and ‘continue’ were confused for one another many times which caused the code to misbehave. However, once the confusion was cleared and the differences know, the code was running as smoothly as the previous version. Other smaller bugs were found and fixed by constantly researching, reading, and learning more about python, OpenCV, and Numpy over the internet.

Chapter 8. Result and Discussion

In this chapter, the performance of the robot Design Ver.3 hardware and software will be evaluated and discussed based on the objectives that the team has set at the very beginning of this project, thus the team can clearly notice how far they have achieved and understand the factors or issues that might change the result in practice. Also, some minor problems that the team has faced before the competition was analyzed and modified to minimize the risk.

8.1 ASME Robot Functionality and Performance

There are 3 specific mechanisms that the team has targeted for this pick-and-place competition, which is Grasping and Flipping Mechanism, Multi-directional Driving and Unloading Mechanism. After careful research, calculation, analysis, and prototyping, the team finally operated their robot and checked its performance. First, the below figures clearly demonstrated the grasping and flipping ability of the Design Ver.3 prototype, handling the two extreme cases of ping pong and basketball.

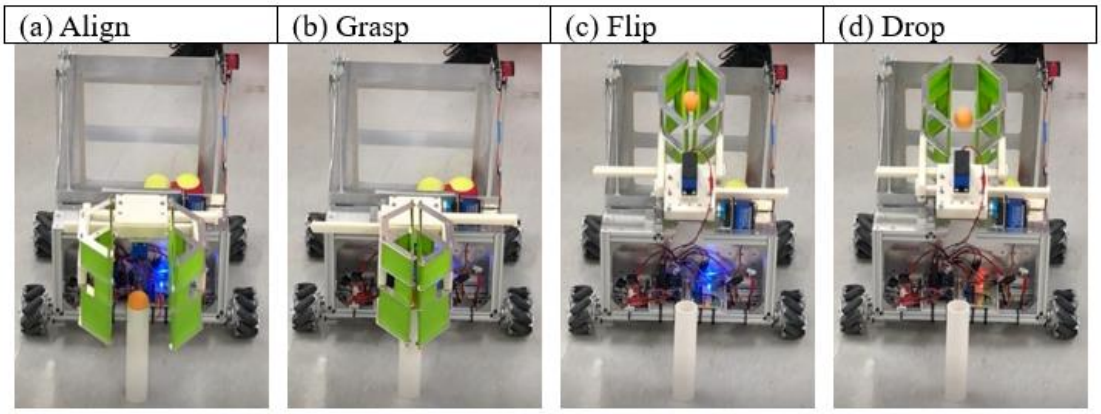


Figure 120 Grasping Ping-pong

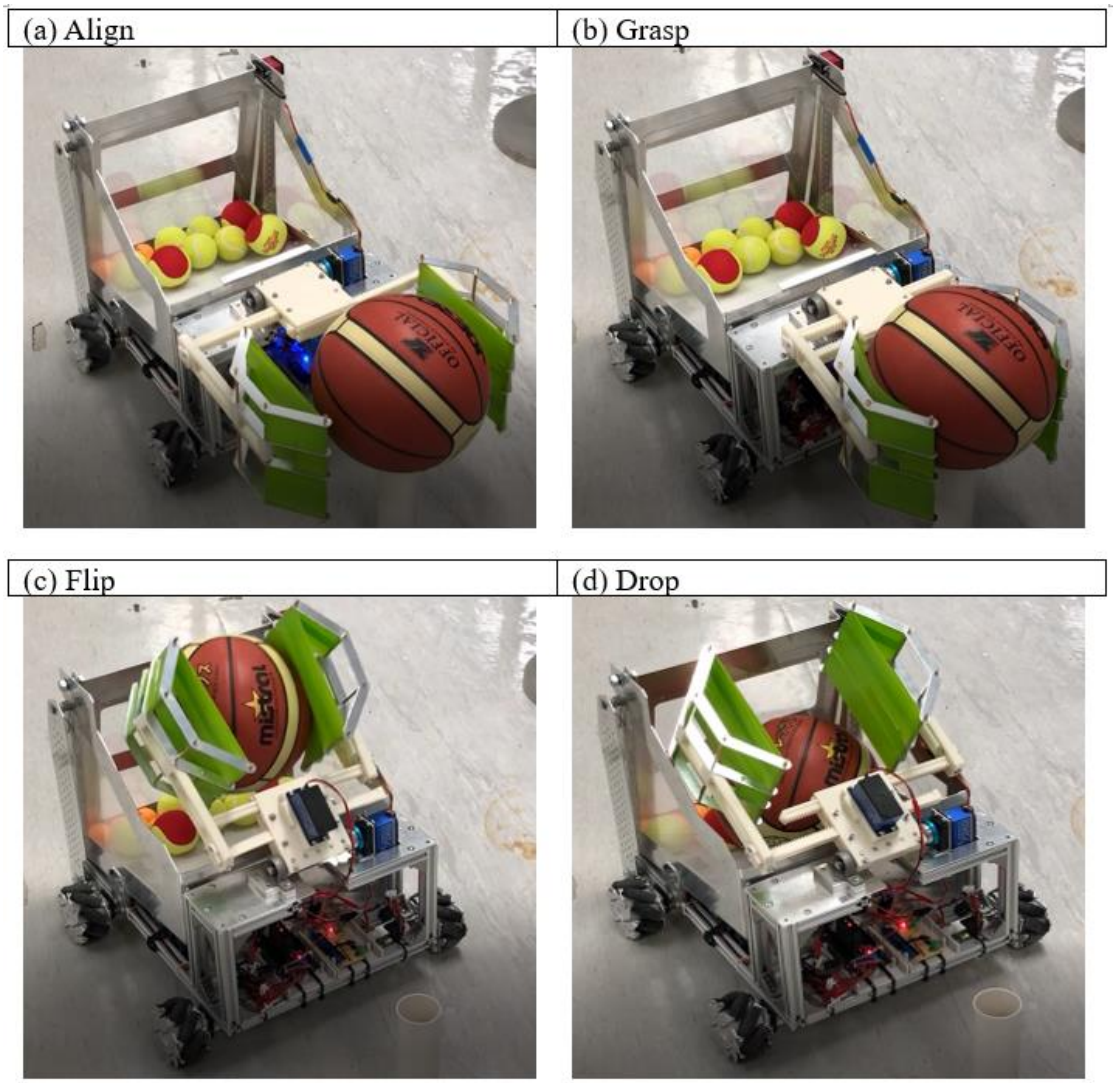


Figure 121 Grasping Basketball

Even though it introduced a slight bending on the arm while grasping tightly, but it worked perfectly without any failure mode, slippage, or overloading the motor. In addition, it was able to grasp the ball with a huge error in the respective position of the robot and the ball to the center. As the controller then do not need to pay too much attention to fine-tuning the position alignment, it indeed saved a lot of time on collecting the ball and enhanced the performance, especially during the competition against other team's robot.

As shown in below figures, even though the ping pong or the tennis ball is not near to the center but rather to the one gripper surface, it still secured the ball without dropping them on the ground.

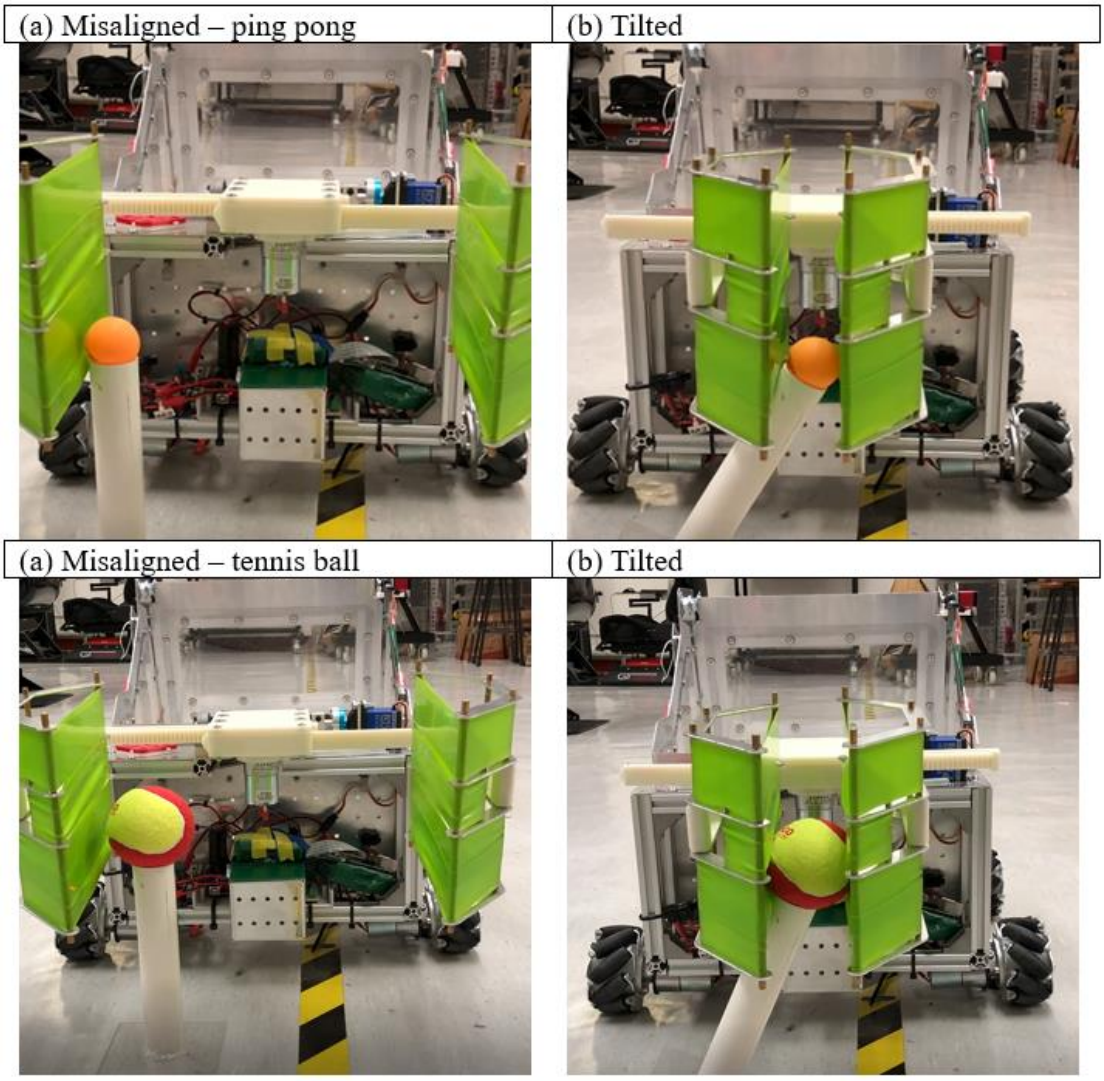


Figure 122 Flexible Grasping Capability

It is mainly because of the board that the PVC pipe is standing on that does not allow the pipe and ball to fall down but just tilt. As the team does not have any ideas on what material will be used for standing, they tried to fix the pipe with cardboard and acrylic to try, and it worked perfectly.

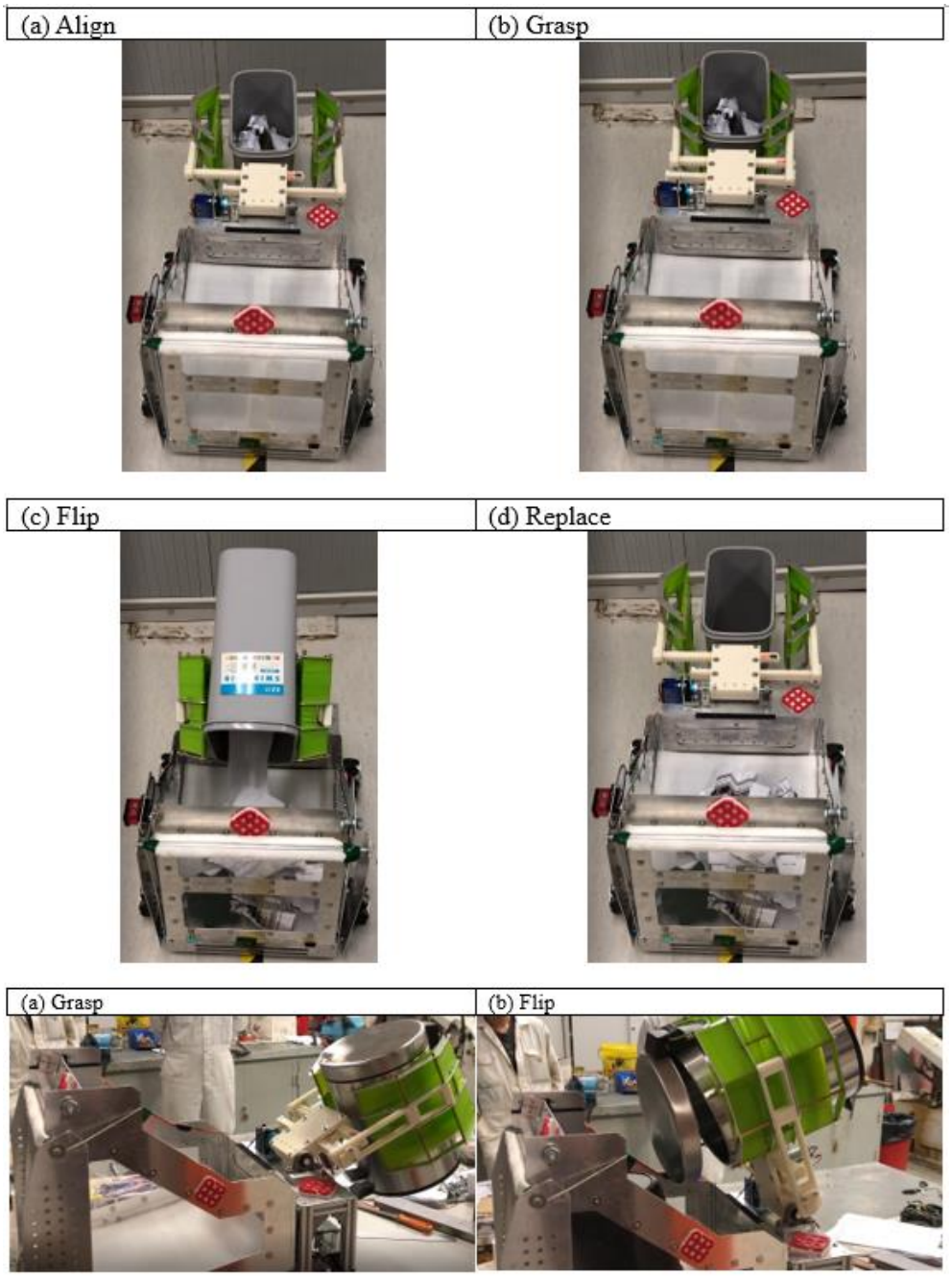


Figure 123 Grasping Bin

Different types of garbage bin are also tested but it indeed has size restriction such as the height of the bin which is higher than 40cm cannot be flipped to collect the garbage, and width also shall be within 25cm so that the robot can grasp it. The Weight of the maximum load that the robot can flip is calculated to be 2.7kg without considering any safety factor. The team tried and evaluated the maximum weight that this robot can flip

in practice is around 1.8 kg, which is 33.33% less than the expected calculation. Though, it is reasonable as there must be energy loss due to vibration, friction, etc.

To evaluate the Multi-directional driving ability, the team has done few simple trial-and-error tests such as driving forward or backward in a straight line, and driving left or right, pivot rotating as well as diagonal driving, and it was acceptable. Also, the driving speed is tested to understand how fast the robot can drive. From Section 3.7.1.1. 4 Driving, it was assumed that the driving speed will at least be around 0.262m/s. However, what the team has emphasized were, the friction coefficient was assumed to be 1, which was to ensure the worst possible case, and the robot weight was set to 15kg, which is actually around 10kg. In addition, the team expected the speed to be much faster not only due to these factors, but also due to special slippage characteristic of mecanum wheel. Thus, to test out the driving speed, the team has conducted a simple experiment to collect some data for evaluation.

In the experiment, the program delay will be changed from 0.75 seconds to 1.40 seconds with an increment of 0.05 seconds. For each delay, 3 sets of data are measured in order to ensure the reliability of the data sets. The experiment setup is shown in adjacent figure.

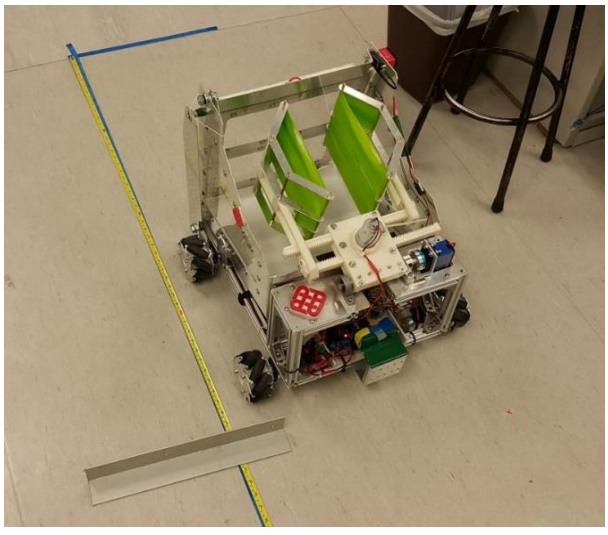


Figure 124 Speed Estimation Experiment

The data collected in this experiment are presented in Table 7 and Figure 125.

Time (s)	Travel Distance (cm)		
	1st	2nd	3rd
0.75	93.5	94.7	92.3
0.80	102.6	104.2	99.2
0.85	105.0	109.3	108.3
0.90	113.2	117.0	112.0
0.95	120.5	118.9	120.3
1.00	130.0	127.1	126.5
1.05	131.7	131.1	129.7
1.10	137.6	134.4	138.0
1.15	139.0	143.0	137.5
1.20	147.2	144.3	146.7
1.25	150.5	149.0	148.2
1.30	157.0	158.0	155.0
1.35	161.1	161.0	160.4
1.40	169.0	166.0	164.0

Table 7 Maneuverability calibration data

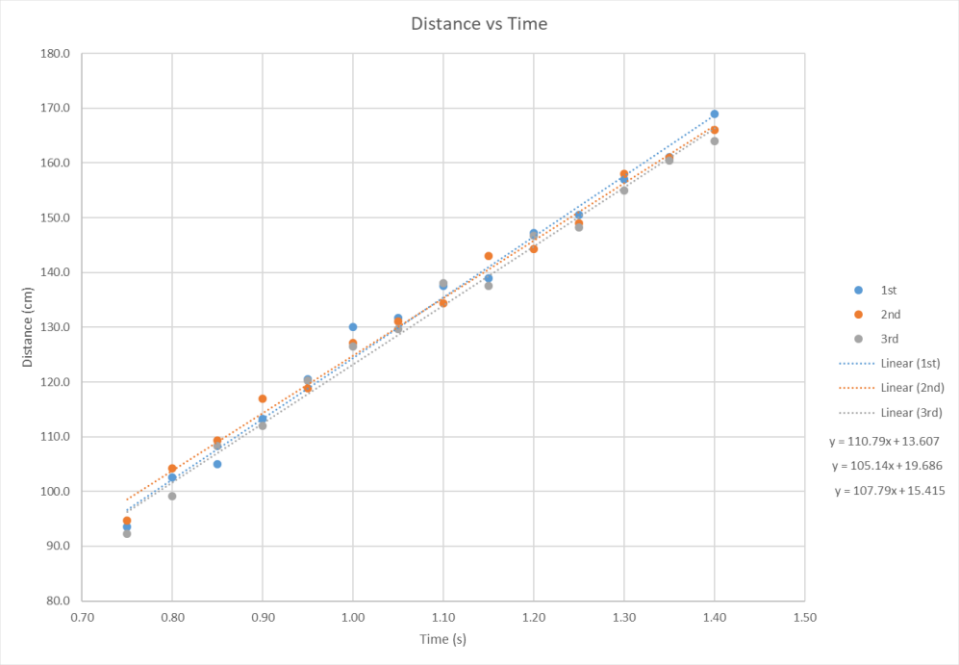


Figure 125 Maneuverability calibration graph

By using the computer aided numerical method in Excel, the following equations are obtained.

$$x_1 = 110.79t + 13.607$$

$$x_2 = 105.14t + 19.686$$

$$x_3 = 107.79t + 15.415$$

Since the equations are equating distance to time, the slope of the equations will be the corresponding robot speed.

$$v_1 = 1.1079\ m/s$$
$$v_2 = 1.0514 m/s$$
$$v_3 = 1.0779\ m/s$$

As a result, the maximum possible error of the system is calculated as follows.

$$\varepsilon_{max} = \frac{Maximum\ Difference}{Minimum\ Value} = \frac{1.1079 - 1.0514}{1.0514} = 5.374\%$$

Since the error value is approximately 5%, the speed could be assumed to be constant. As an average value, the equation will be simplified into the following.

$$\bar{x} = 107.91t + 16.236$$

Thus, the team concluded that the driving speed of the robot is approximately 1m/s, and it was indeed a satisfying result.

Unloading of collected ball safely at the starting area and preventing them from rolling out of the zone was also one of the design objectives that the team has set. As brainstormed and designed during the development stages, the team had equipped the robot with a special arm blocker to keep the ball inside the area. As shown below figures, it was very fast and efficient motion indeed, and was able to keep all of the collected balls without any trouble.

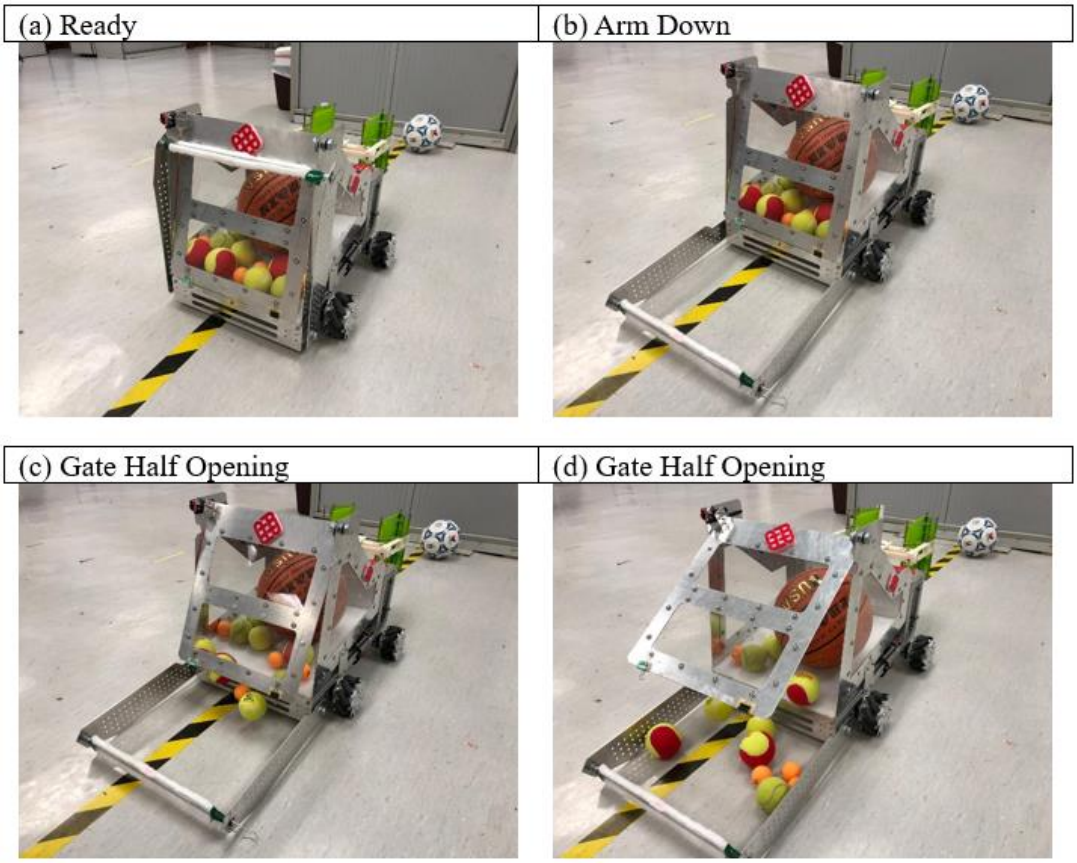


Figure 126 Unloading Mechanism Result

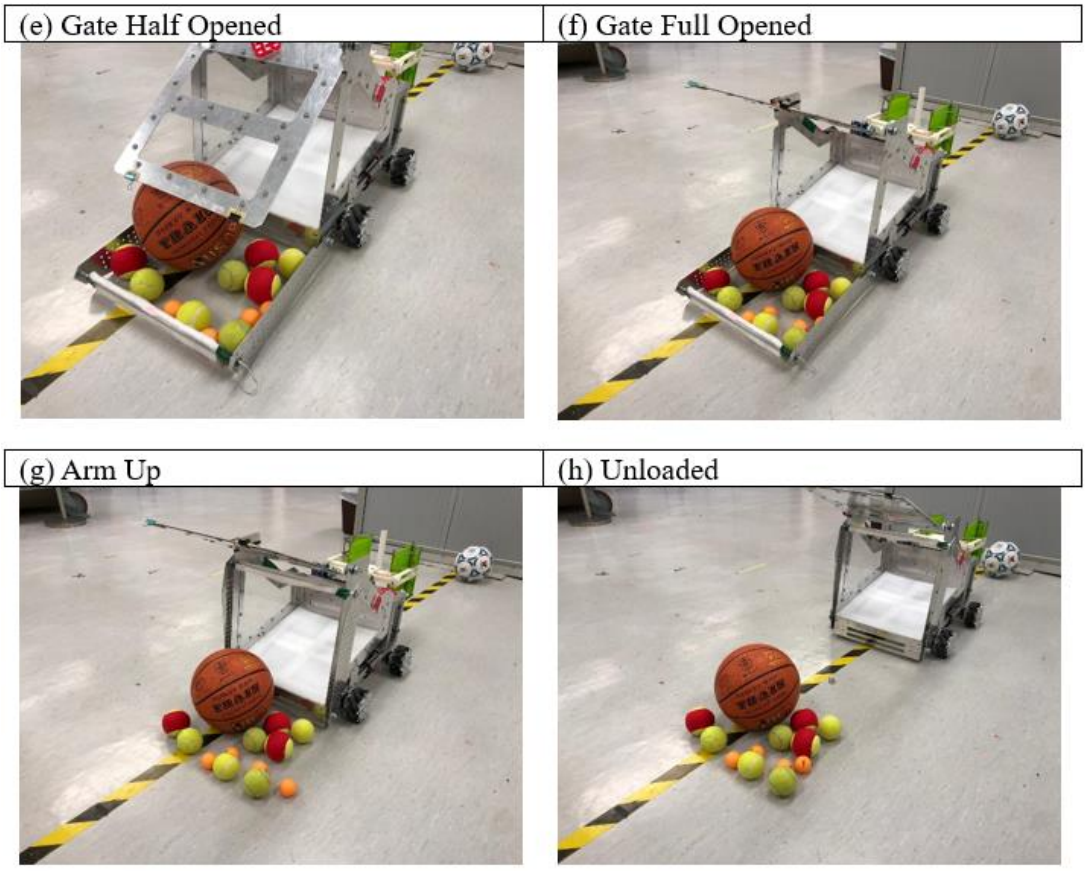


Figure 127 Arm Blocker Mechanism Result

All in all, 3 mechanism objectives such as grasping and flipping, multi-directional driving and unloading was achieved in satisfactory, and the robot was able to complete the mission as team's expectation.

8.2 ASME Competition Performance

In this section, the performances related to the ASME competition would be evaluated. For example, the practices during the preparation stage before going to India, some problem faced after these practices, as well as some strategic planning for the competition.

8.2.1 Practice Result and Strategy Evaluation

The team has studied and prepared for varieties of ball distribution scenarios, and compared different strategies based on the time consumed so that they can evaluate the one with higher efficiency. In below example figures, it is assumed that the arena set up will have evenly distributed balls of different sizes, including 4 basketballs, 8 tennis balls and 4 ping pong balls, and students have chosen two strategies for testing, which are based on (1) distance and (2) ball sizes. Number 15 and 16 in blue color are the balls that will be collected in second journey.

Distance priority strategy is basically collecting the balls which is near to the robot without considering the ball sizes. Though, since it can only collect 2 basketballs at most, it will have to come back and unload them first before collecting the last two. On the other hands, the Ball-size priority strategy is focused on collecting all small balls as well as 2 basketballs which are placed further away from the starting zone. This is to minimize the distance that the robot needs to travel again when they start their second journey after unloading. Each strategy was tested for 10 iterations, and data collected are illustrated in Table 8.

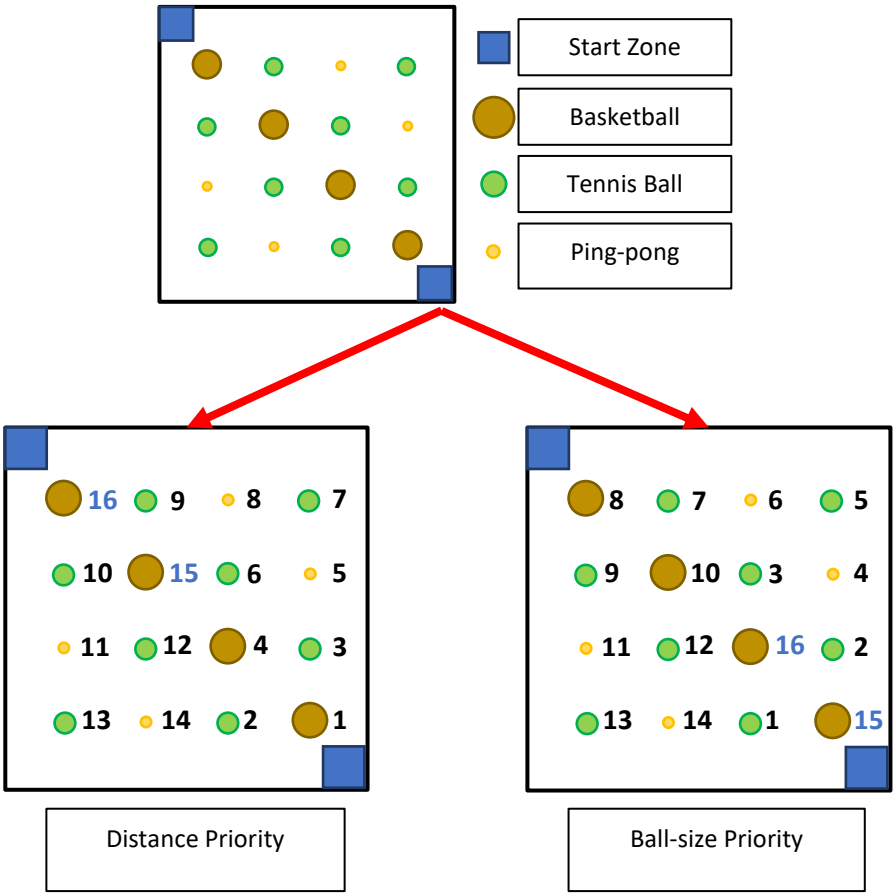


Figure 128 Strategy Evaluation Setup

Distance Priority Strategy	
#	Time taken
1	3.29
2	3.42
3	3.45
4	3.34
5	3.57
6	3.44
7	3.16
8	3.46
9	3.4
10	3.52
Avg	3.42

Ball-size Priority Strategy	
#	Time taken
1	3.24
2	3.29
3	3.14
4	3.23
5	3.39
6	3.14
7	3.16
8	3.23
9	3.19
10	3.4
Avg	3.24

Table 8 Comparison of time taken between collecting strategies

From experimental results, it could be seen that collecting small balls with priority is generally having a steadier performance, and the time taken for the robot to complete the process is generally quicker. Therefore, collecting smaller balls first will be the chosen strategy to be followed by controller.

Apart from that, the strategy is also evaluated for the knockout rounds. The scoring system is gaining 2 marks for balls secured and 3 marks for balls delivered back to starting area. Based on the rules, direct impact between robots are not allowed and the robot must be in contact with the arena. In this case, if our robot is blocking the whole return path of the opponent robot, they cannot push us away if we are stationary nor take an alternate path out of the arena. Then, the number of balls required to collect in a worst-case scenario could be calculated by the following assumptions.

(i) Assume that the opponents are having much higher collecting speed than us.

(ii) Assume that we can collect fewer balls and return them to starting area before the opponent collects the rest of the balls.

(iii) Assume that we block the opponent robot from starting area such that they only gain marks for securing balls, but not delivering balls to starting area.

Then, let x be the amount of balls that the robot is required to collect.

$$(2 + 3)x > 2(16 - x)$$

$$x > 4.57$$

Therefore, the minimum balls required to be collected for this strategy to work is 5. From this calculation, the team leader can decide whether this strategy will be adopted when the opponent is having a better performance.

8.2.2 Problem and Solution

The practice runs acts as a pressure test to the robot. This allows us to identify problems that are related to the time domain.

The first problem occurred after many practice runs is that the flipping motor has malfunctioned. After the diagnosis of the motor components, it is found that the gears worn out. Due to the vibration caused by locomotion, the servo motor gears are experiencing misalignment. This causes the components to collide with each other, causing the gear to eventually fracture as shown in Figure 129.

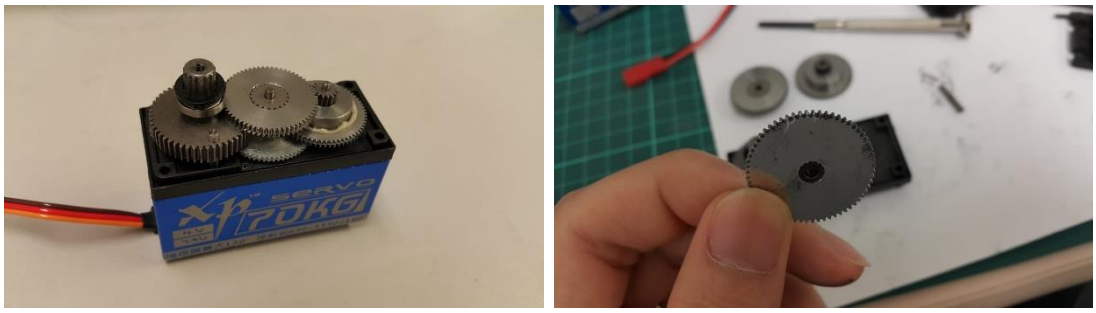


Figure 129 Motor diagnosis and gear fracture in servo motor

The measures taken to solve this problem is to add another bearing near the motor end as shown in Figure 130, such that the vibrational motion of the shaft is reduced. Apart from that, the fatigue parts are also replaced with new spare parts.

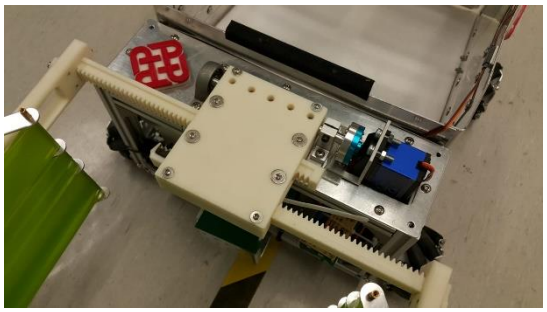


Figure 130 Two bearings for supporting the flipper shaft

The second problem occurred after the practice runs is that the wheels are slightly bending outwards. The reason for this phenomenon is that the motor mounts are bent outwards as shown in Figure 131.

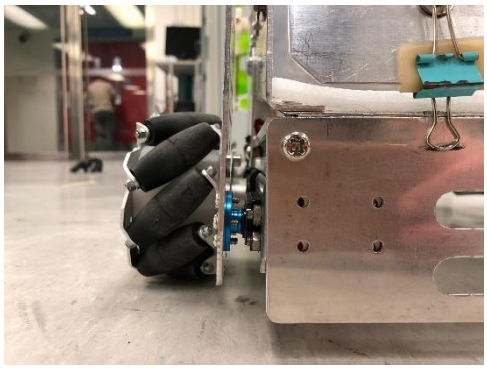


Figure 131 Motor mount bending outwards

One possible cause is that the vibration from locomotion caused fatigue deformation of the motor mounts. However, due to dimensional limitation of the competition, the motor mounts could not be further enforced with better design Therefore, the solution to this problem is also replacing the old parts with new parts before the actual competition.

8.2.3 ASME Competition

The competition is divided into 3 rounds. The setup for the competition in India is illustrated in Figure 132. For the preliminary round, the scores are used for ranking the teams in order, as well as setting a basis for the knockout round matches. Please refer to Appendix XVII for the team results.

Figure 132 ASME Setup in India

For the preliminary round, the performance is not as expected because the dimension of given pillars is not meeting the requirement specified in the rules. The diameter of the provided pillars from host university is more than the range stated in the rules. This causes the robot performance to be dropped because it was not designed for the scenario that we faced in India. However, we still managed to secure balls by gripping the balls with the very tip end.

After completing the competition on the first day, we modified the gripper part by adding more materials within the yoga band, such that the friction could be increased.

On the second day of the competition, the host changed back all the pillars into those meeting the requirement in rules. Therefore, the robot performed as expected and scored accordingly.

On the third day of the competition, some parts malfunctioned, and the robot did not respond to the controller. Therefore, we had to prepare for the worst-case scenario, and we adopted the blocking strategy to prevent opponents from delivering balls back to their starting area, and we managed to secure 2nd runner up of the whole competition.

8.3 Feedback Control System Evaluation and Performance

8.3.1 Arduino – The initialization feature

Based on the Arduino Uno program, the servo motors hardware must be calibrated precisely before running the software. The zero degree position must be placed at the angle where the motors are required to soft-initialize because upon startup of the board, when any button associated with the servo motors is pressed, the motors are initialized to zero degree position. For our robot, we were not aware of this fact before-hand and for the first trial-run, the gates flew open as the controller touched the corresponding button to test. At first, this anomaly was perceived as a bug / error, because in an industrial machine, this should most definitely not happen and can cause major injuries. However, for this small-scale robot and with the ASME competition in mind, this feature was retained as it could help the team to quickly test if all the motors connections were okay right before the competition. Recommendations to avoid this problem on an industrial machine are made in Chapter 10.

8.3.2 Python Program – Alternative solutions explored

For fixing the limitations of the first version of the python code mentioned in Chapter 7 the team came up with several ideas that worked to a certain extent but were not employed in the final version because the solution used there is the simplest and most effective. Some of these ideas are discussed in this section.

8.3.2.1 Harris Corner Detection

Harris Corner detection algorithm was developed to identify the internal corners of an image. The corners of an image are basically identified as the regions in which there

are variations in large intensity of the gradient in all possible dimensions and directions. Corners extracted can be a part of the image features, which can be matched with features of other images, and can be used to extract accurate information.

The mathematics behind the algorithm is such: For each pixel (x, y) it calculates a 2x2 gradient covariance matrix $M^{(x,y)}$ over a blocksize x blocksize neighbourhood. (blocksize is one of the input parameters of this function). Then, it computes the following characteristic: $\mathtt{dst}(x,y) = \det M^{(x,y)} - k \cdot \left(\mathrm{tr} M^{(x,y)}\right)^2$ Corners in the image can be found as the local maxima of this response map [58].

This algorithm was planned to be used to get a list of all the corners of the plus-sign sticker corners, and based on the geometry of the shape observed, and by simply calculating the difference in length of, for instance, two opposite sides of the sign, the orientation could be approximated. This idea sounds good in theory and also worked during the initial trial runs on a sample photo depicting a tilted plus-sign shown alongside here. However, when it was run on the Raspberry-Pi with the camera module, its performance was not at all impressive. It barely functioned in the real world. For a test, A similar plus sign was drawn onto a piece of paper and the code was run, Figure 134 only shows a few points not correct; but in real-time, the points seem to randomly appear and disappear like twinkling stars.

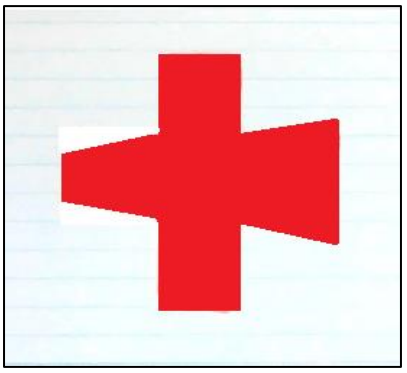


Figure 133 Sample for testing Harris Corner Detection

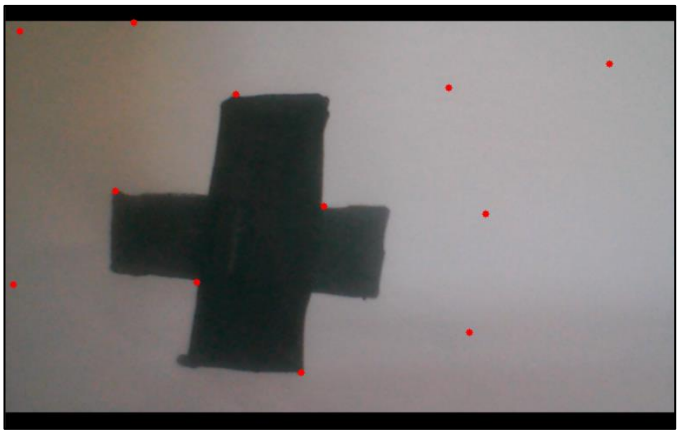


Figure 134 Real-world test for Harris Corner Detection

8.3.2.2 Contour Area Division

When nothing else was working, the team also tried to get the contour area of the plus-sign target sticker and split it at the center into 2 separate areas. Similar to the working solution, but much more complicated as we could not find any functions readily made to accomplish this task. It was, however, this approach that planted the idea to use the number of pixels to compare two sides instead of the area. Thus, the team kept trying various solutions to address an underperformance issue in the original version of the program till they found an apt solution.

8.3.3 Empirical Evaluation to quantify Area value

For this section, the goal is to determine the area threshold that indicates a suitable distance for gripping. Once the area range is obtained, the program could be further calibrated for better accuracy.

In the experiment, the minimum and maximum bounds of the area of the image detected will be recorded at ranges of distance from the target. The program will then

feedback the corresponding data on a screen. The values are observed over a period of time, and the maximum and minimum values of the area will be recorded. The experiment setup is shown in Figure 135 below.

Figure 135 Image area calibration setup

The data collected is presented in Table 9 and Figure 136.

Distance (cm)	Area (sq. pixel)	
	Lower Bound	Upper Bound
10	69500	71000
12	50000	51500
15	27000	30000
20	16100	16600
30	7100	7500
40	3700	4100
50	2400	2500
60	1530	1580
70	1110	1130

Table 9 Image processing calibration data

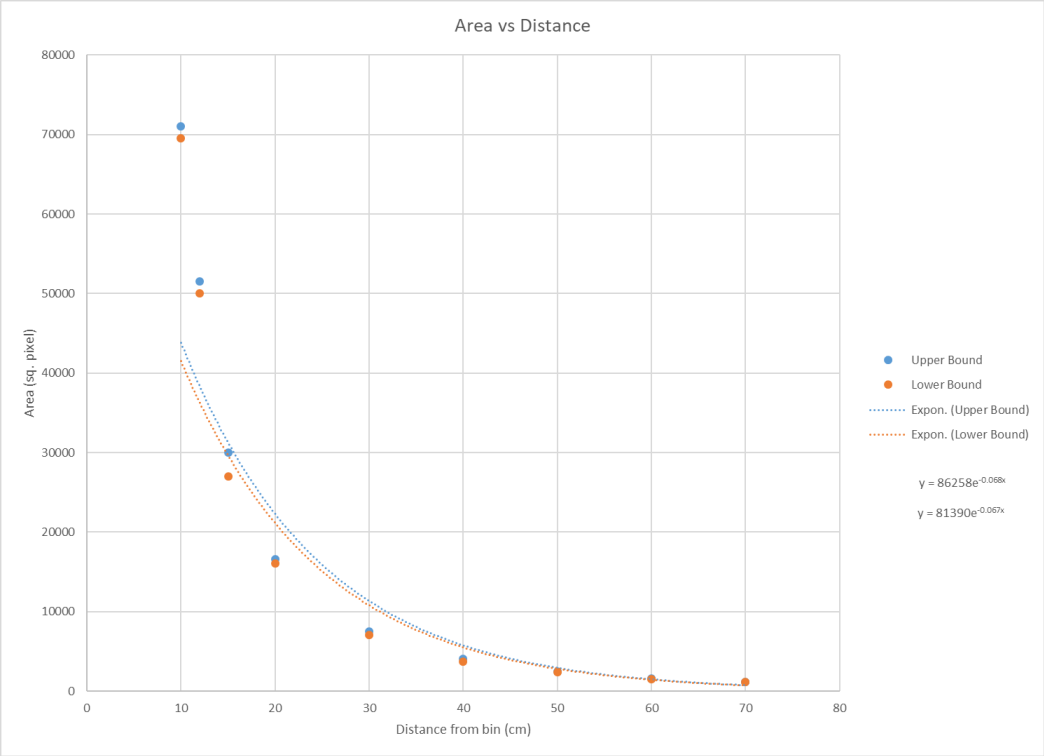


Figure 136 Image processing calibration graph

From the data sets and computer aided numerical method in Excel, the relationship between the image area and distance from the target are known.

$$A_{max} = 86258e^{-0.068x}$$

$$A_{min} = 81390e^{-0.067x}$$

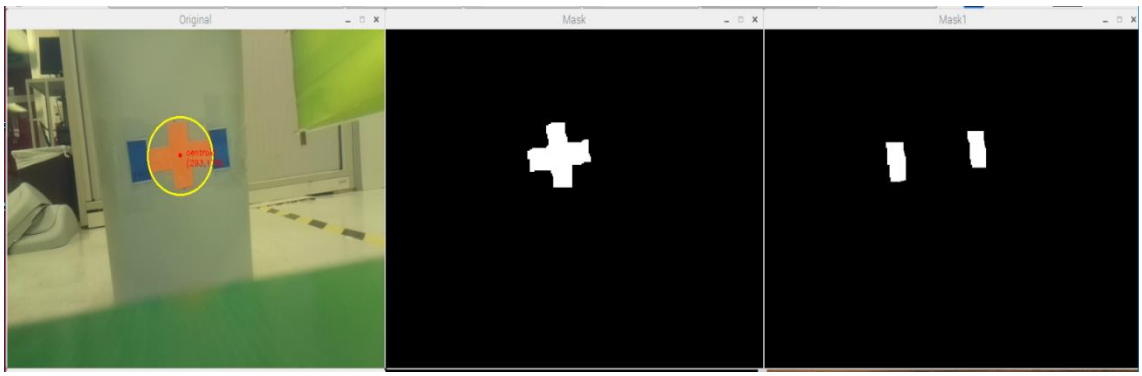
By inspection, the best location for gripping the bin is around 11cm. From the experimental results, this indicates that the area threshold of the program should range from 38949 sq. pixels to 40827 sq. pixels. This can allow the robot to achieve higher accuracy in its positioning.

For future development, since the robot can now know its distance from the target bin, it can automatically determine the required time to travel in order to reach the target.

8.4 Feedback Control System Results

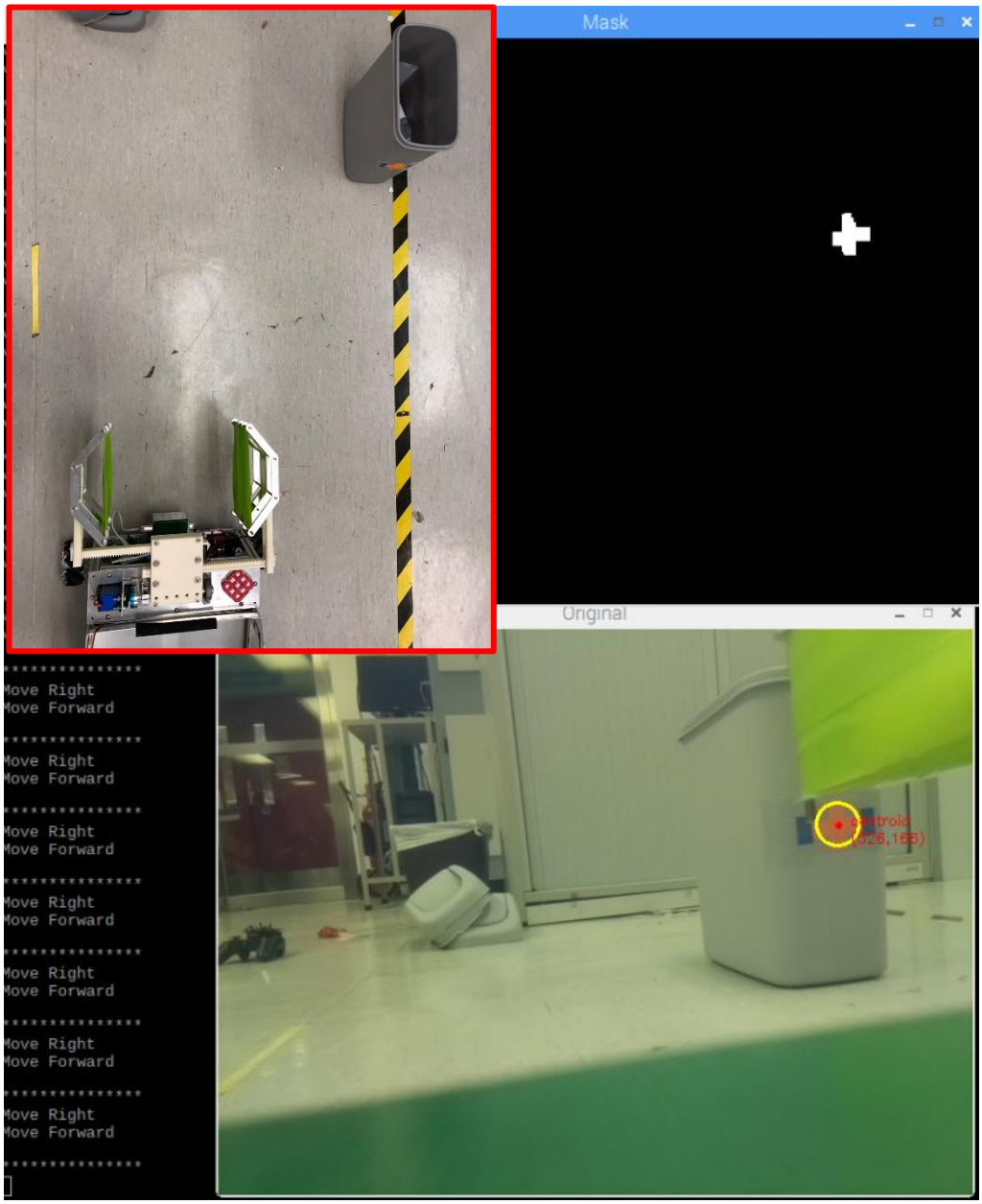
After several iterations of the python code, the team was satisfied with the results obtained for the dynamic object detection program using image processing from the Raspberry Pi. The board was connected to a team members personal computer via WiFi to view the results, which are displayed in this section.

The machine first thresholds the Orange colour on the sticker as shown in the window “Mask” below to compute and display the Left-Right and Forward-Backward movement of the robot. Next, the machine thresholds the Blue colour on the sticker shown in the window “Mask1” to compute and display the Clockwise-Anticlockwise rotational movement for the robot.

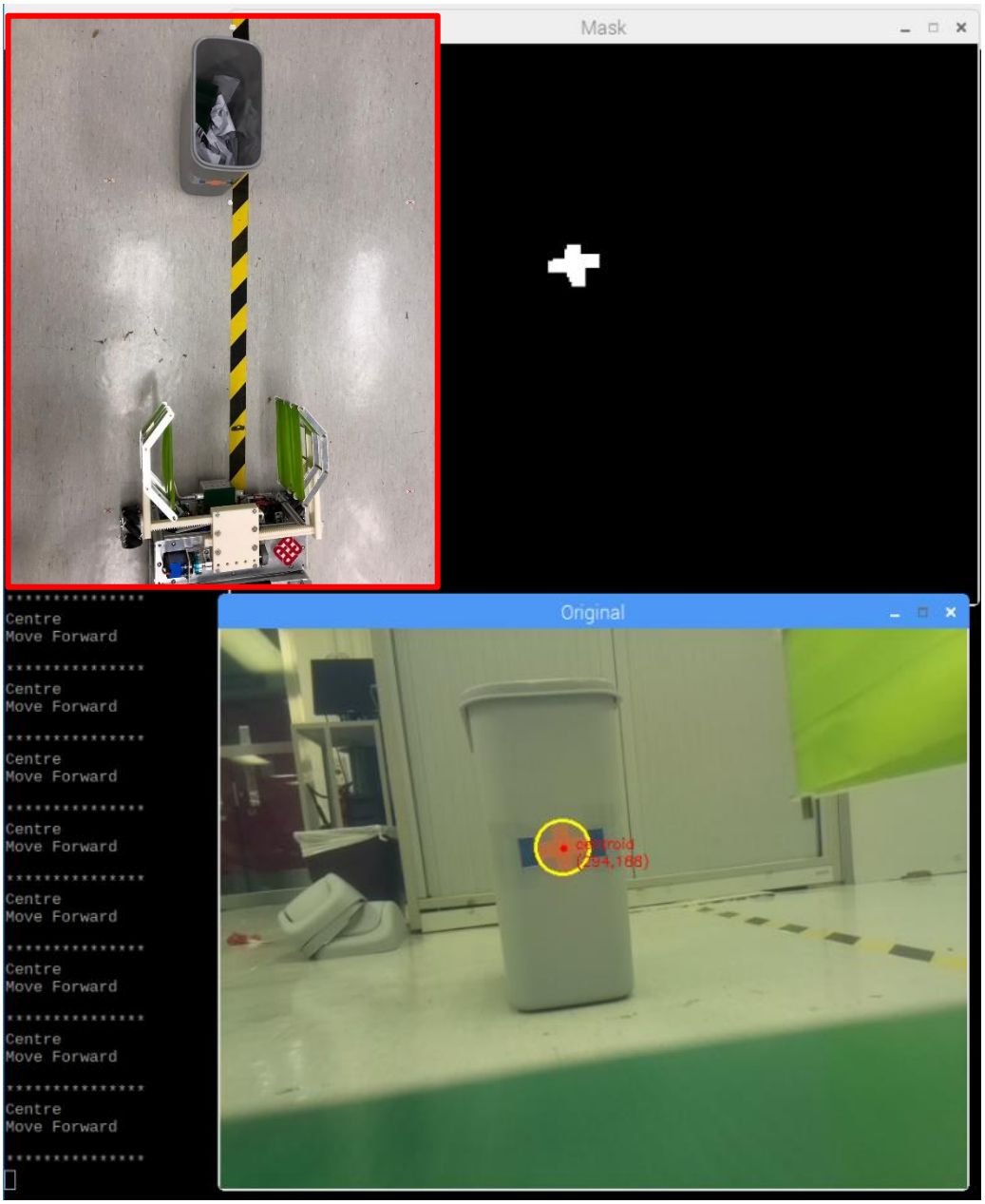


It is evident that the object detection component of the program works flawlessly from the results obtained above.

Now, to demonstrate its ability to align itself, the robot starts from a random point misaligned from the garbage bin and finds its way till the target position as shown with the following images. The output is displayed on the terminal of the Raspberry Pi.

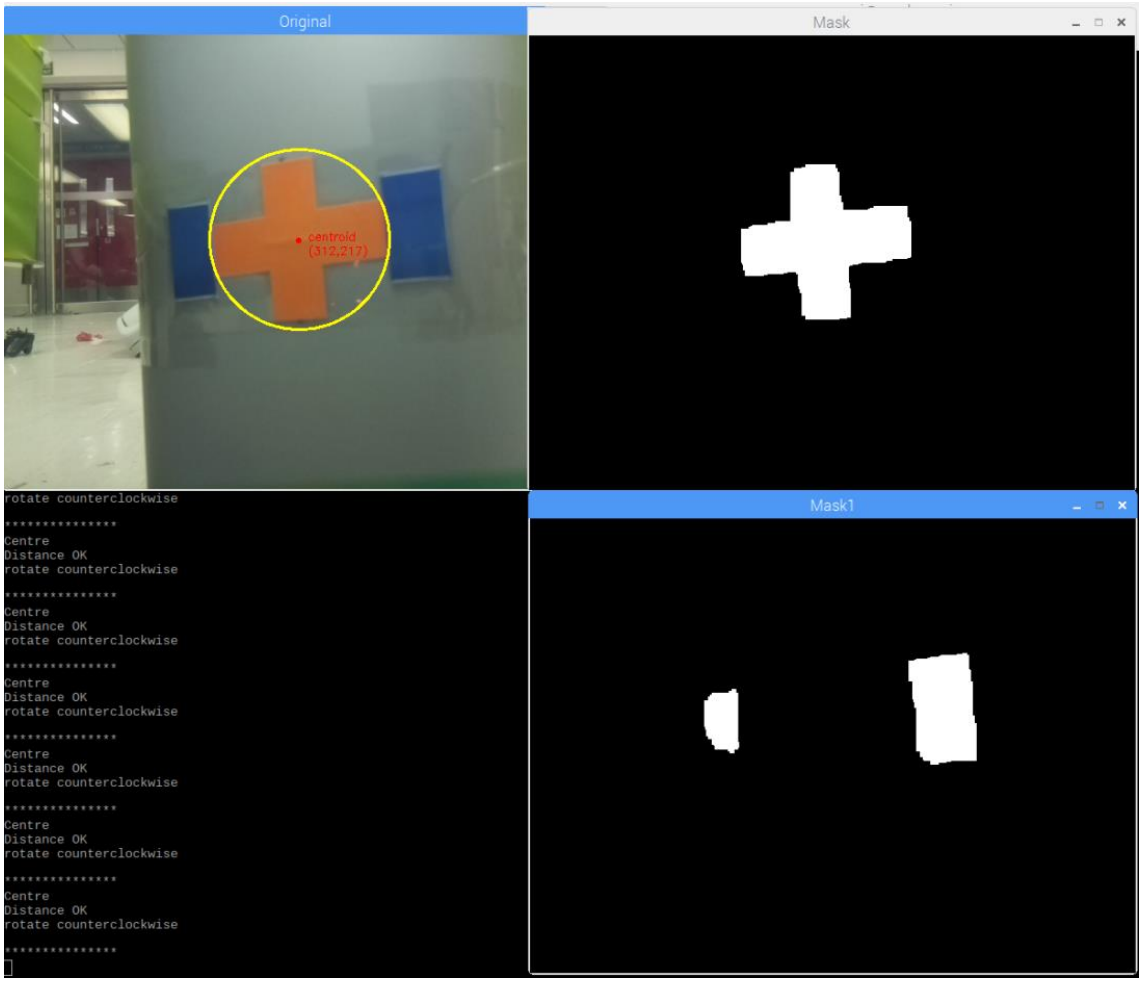


Here the robot starts from the random location shown and the output tells it to "Move Right" and "Move Forward" as expected.

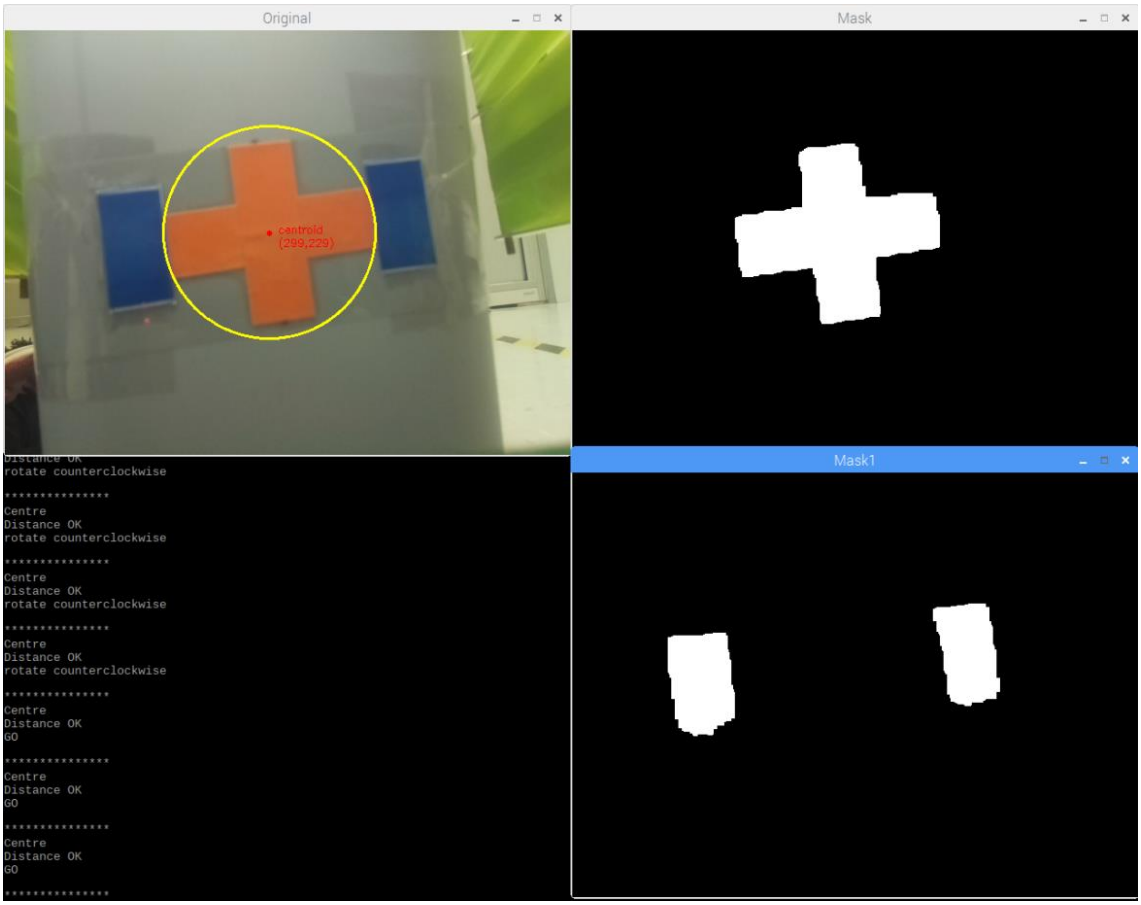


In the next step, once the robot has moved right and is aligned at the centre, the output displays “Centre” and tell the robot to now “Move Forward” as expected.

Now, once the robot has moved forward and the distance is within the error percentage set, if the bin is misaligned and tilted at an angle as shown in the image above, the robot must supply this information to the controller.



As displayed from the results obtained above, the code works as expected and displays “Centre” and “Distance OK” to indicate that the position of the robot is correct. It also displays “rotate counterclockwise” to indicate that the garbage bin must be rotated counter-clockwise, or the robot must rotate clockwise to accommodate for this misalignment of the target. Thus, this situation also reaps the results that are expected and confirms that the additional module in the final version of the code also works beautifully.



Finally, when the robot corrects itself and the correct position and orientation are achieved, the output from the terminal displays “Centre” “Distance OK” “GO” to indicate that all values are within the error ranges and the robot can GO and execute the next task of gripping the target bin. Thus, the results for the feedback control system match our expectations and the program works well.

Chapter 9. Conclusion

The team has achieved the majority of the objectives that they have targeted for this project not only restricted to ASME competition but also demonstrating the concept of proof for its possible application, which is garbage collection robot. They have applied their in-class knowledge as well as the techniques trained from the industrial center to tackle the problems defined, analyze the data collected, operate the machinery and optimize their solution. While the Gantt chart helped students to plan for overall project time schedule, they also set a clear design flow chart so that they can keep the project on track by following this as their development guideline and understand how far they have achieved.

Prior to begin designing and modeling, they have researched enough resources available from both online and local toy shops to understand the relevant technology, and even went to local electrical shops to learn about microcontrollers including Arduino and Raspberry Pi that the students did not really have any chance to deal with as mechanical engineers throughout their undergraduate study. Via evaluating the design concept iteratively, they could analyze the potential problems in advance and brainstorm for the solution and thus minimized undesirable modification in manufacturing stages. Also, their idea of adopting a modular approach design has indeed increased efficiency in each concept modification stages, because they could only focus on the problematic module instead of changing the whole design again and again.

After finalizing the design concept, the team conducted necessary analysis on the material property to understand if any failure mode will result in critical loading, and chose appropriate standard parts, such as motors, through carefully analyzing and checking their technical specification with the team’s requirement. In order to

manufacture those necessary customized parts, the team researched available machinery and tools on campus and consulted with the technicians to find the most appropriate fabrication method. Dimension tolerance due to fabrication as well as 3D printing error was taken into consideration during 3D modeling, and thus there was no trouble in the assembly stage.

The team also studied and demonstrated software skills including python, OpenCV, etc. As mechanical engineering students, even though they previously had a programming course learning C++, it was indeed the very first project that the students had to program a robot to complete a mission. Particularly, to develop a program for vision-sensor-based position alignment, the team never stopped challenging themselves on trying the code to optimize the outcome.

All in all, the students strongly believe that they have gained a solid knowledge and skills in robotics industries via managing both hardware and software problems. Also, the team always valued communication and cooperation within the members, and it ultimately motivated and inspired each other to achieve the team's goal and led to great team chemistry and synergy. Despite the devoted work for almost 8 months, the team has to admit that there are still lots of room for improvement for both ASME and its application of garbage collection.

Chapter 10. Further Development

10.1 ASME Competition

To win the championship in final rounds of ASME competition, which will be held this November, the first hardware problem that should be modified is Module 2 – Lower body. The team noticed that the motor mount was bending due to robot's own weight over time, and this should definitely be modified. One solution that the students have brainstormed was mounting the linear bushing on the other side of each wheel, so that the weight will not be concentrated on the mount, but evenly distributed. As for the dimension requirement, the distance between left and right wheel will have to be narrower since the linear bushing will increase the overall width, but the team believe that Module 2 can be modified to not affect the Module 1 – Upper body and even allow more room for the container because the aluminum profile chassis is very easy to be customized. This could be very critical in final rounds because it is expected to have more basketballs, and the robot should carry more payload, thus the mount will experience more loading, and the container must be big enough to accommodate as many basketballs as possible. One possible solution to enlarging the container is mounting Module 2 on top of wheel level as shown in **Error! Reference source not found.**.

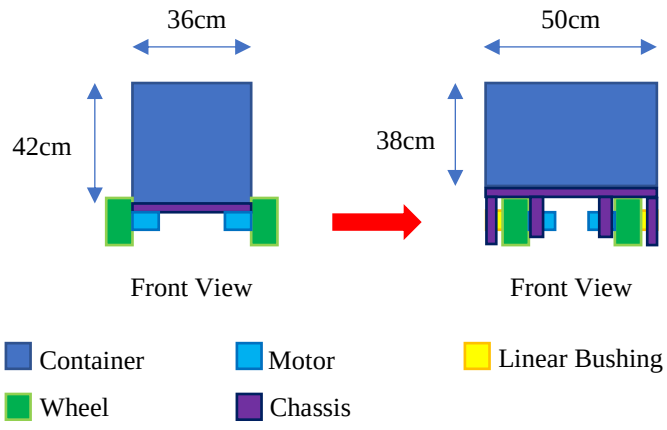


Figure 137 Preventing Mount Bending and Container Enlargement Plan

The team has done simple calculation to predict how much volume will be enlarged, assuming that the depth, or sometimes is referred as length, is identical but just changes in width and height.

$$\frac{V_{new} - V_{old}}{V_{old}} = \frac{(W_n)(H_n) - (W_o)(H_o)}{(W_o)(H_o)} = \frac{(38)(50) - (36)(42)}{(36)(42)} \times 100 = 25.66\%$$

As proven from above calculation, the container is expected to be enlarged by around 25 percent, and it should have enough space to accommodate 3 to 4 basketballs with the gripper holding one of them.

10.2 AGV Development with LiDAR Sensor

This project clearly demonstrates the proof-of-concept of a pick-and-place robot used for indoor garbage collection application. For a commercial-level machine, we cannot use microcontrollers such as Arduino or Raspberry Pi as even though they serve as excellent platforms for prototyping, they are not stable enough platforms for commercial use. Therefore, instead of the Arduino Uno, to control the motors, switches, and other devices, the team recommends using a PLC (Programmable Logic Control) Unit. For example, one of the most widely used PLC units for small to medium sized automation are developed by Siemens [59]. The Siemens S7-1200 is a compact controller and the SIMATIC HMI Basic are panels that can both be programmed with the TIA Portal engineering software. The ability to program both devices using the same engineering software significantly reduces development costs. It

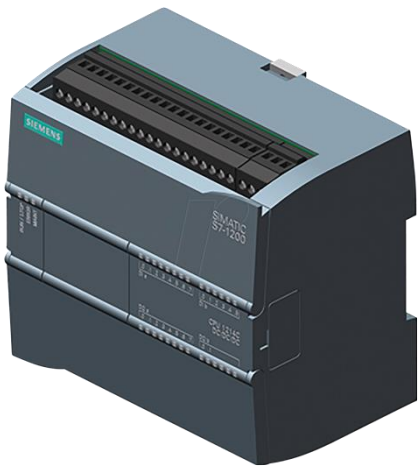


Figure 138 Siemens S7-1200 PLC controller

is very sturdy and has high-speed I/O capable of motion control, onboard analog inputs to minimize space requirements and the need for additional I/O, and 4 pulse generators for pulse-train and pulse-width applications. Additional modules can easily be added or removed to customize the automation solution to the client needs.

Furthermore, the Raspberry Pi is used to perform the image processing operations for object detection. While it performs beyond our expectations, it cannot handle such a load constantly. In addition, if we want to achieve complete automation of the robot, then it must be equipped with additional peripherals like the LiDAR sensor, which is an acronym for 'Light Detection and Ranging'. This sensor allows the robot to generate a 3d map with representations of its target as well which further allows the robot to traverse through various obstacles without the need for any human input. Other sensors like laser sensor, magnetic tape detector, and temperature sensors are also recommended to achieve complete automation. Most newer models of Automated Guided Vehicles (AGVs) also incorporate machine learning to enhance efficiency, speed, and ultimately user experience.

To run all these heavy tasks, the robot must be armed with a controller like the 'NVIDIA Jetson TX2' (Figure 139). This controller has enough RAM and a very powerful GPU for image processing and other applications mentioned above and can also easily control the PLC Board. These two controllers are proven to be a strong duo and are highly recommended for future use on this project.

Figure 139 NVIDIA Jetson TX2

10.3 Fully Automized Garbage Collecting AGV

To replace the monotonous labor job done by the janitors, which is collecting the garbage from the bin at each floor levels, the team could develop this robot to AGV as mentioned and build a futuristic garbage collection system. As shown in the below figures, this AGV will collect the bin and flip to drop garbage into the container, which is detachable from the robot. A main hub will be developed to collect the fully loaded garbage container from the AGV. With the help of SLAM technology, AGV will find its way to go around the floor and collect the garbage, then return to hub to replace its container with a new one. Via integrating with the infrared sensor at the back of the AGV, it will be able to reverse drive and park in the hub. Meanwhile, the AGV will stay inside the hub for auto-charging as well. It is a very conceptual idea without technical details at the moment, but as the team could prove the concept of collecting the garbage with the robot Design Ver.3, it is indeed feasible to develop garbage collection AGV.

The methodology of this garbage collection concept is clearly demonstrated below in detailed steps:

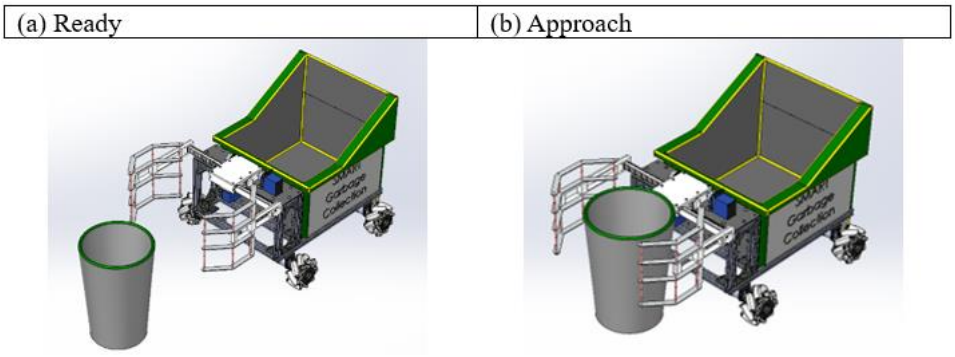


Figure 140 Application - Garbage Collection 1

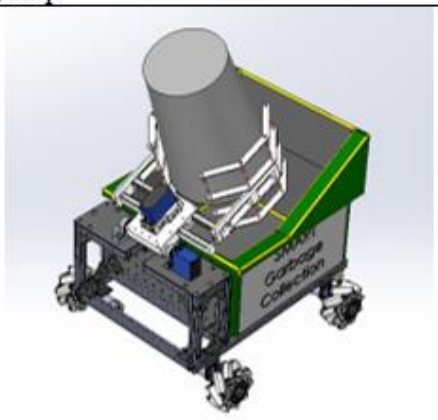
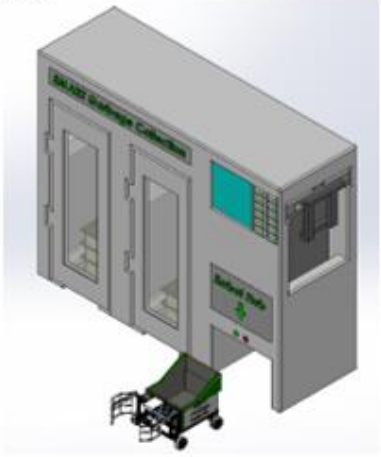
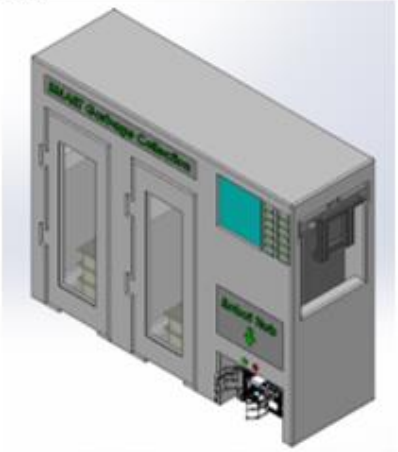
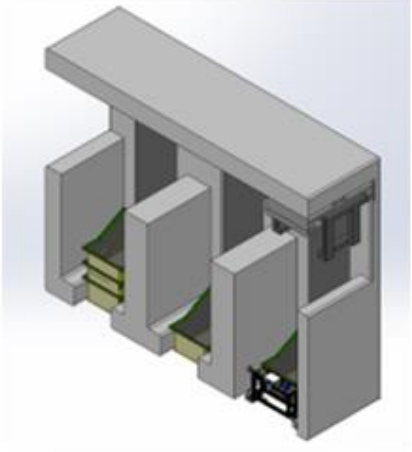
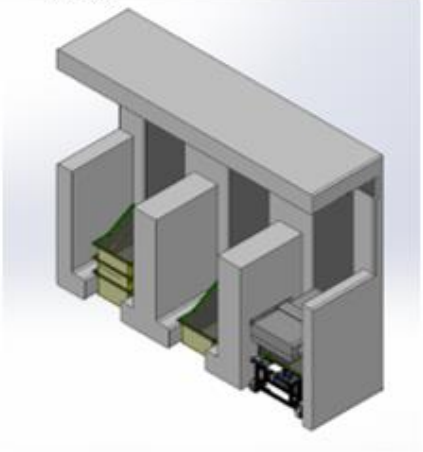
(c) Grasp	(d) Flip
	
(e) Hub	(f) Park
	
(g) Cut view	(h) Hub-gripper down
	

Figure 141 Application - Garbage Collection 2

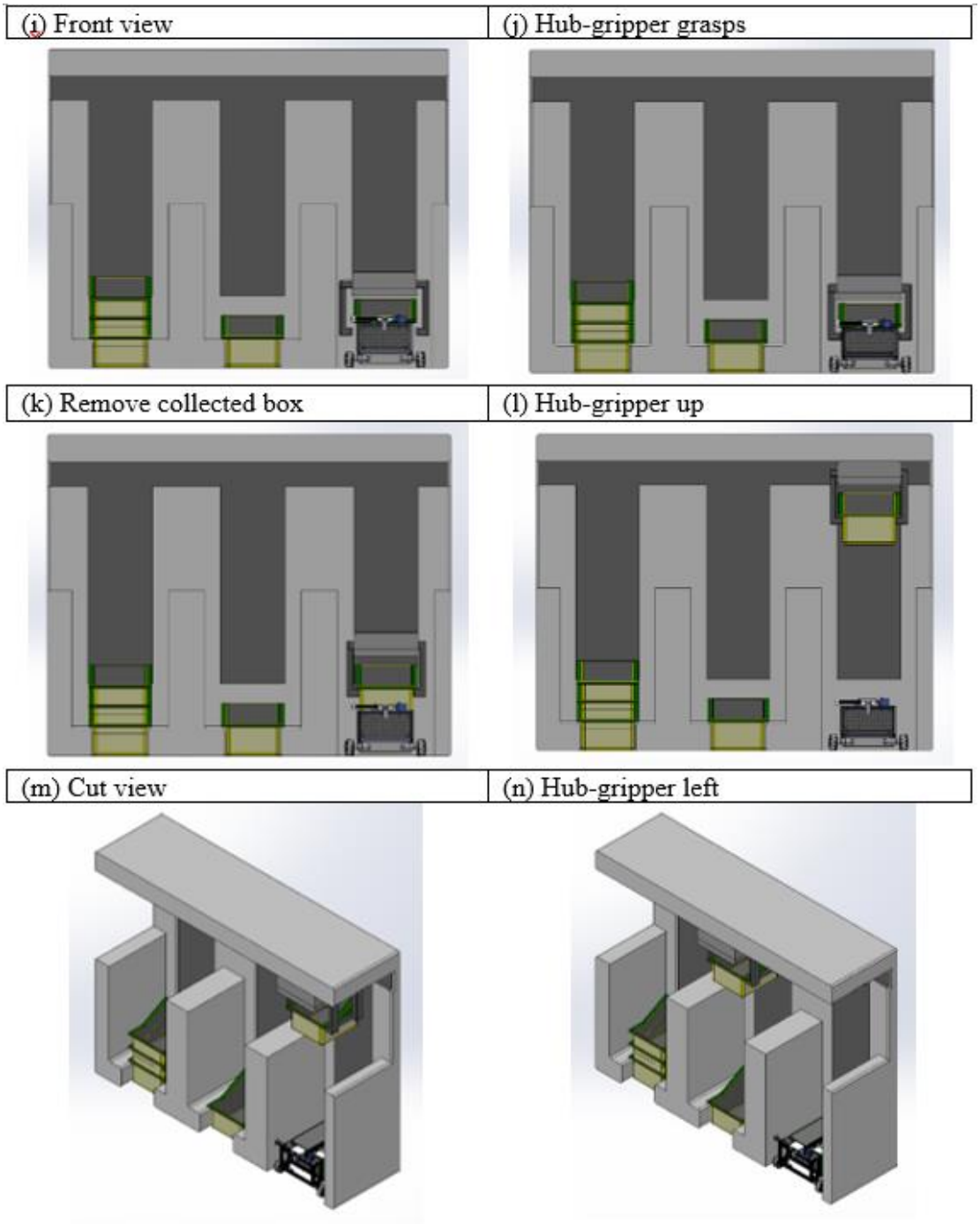


Figure 142 Application - Garbage Collection 3

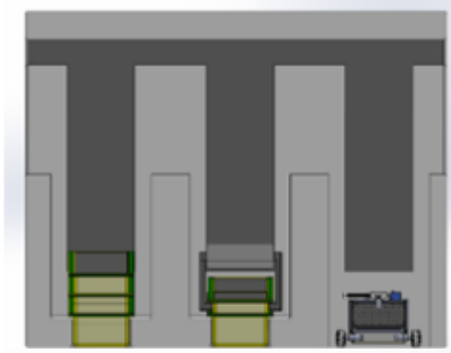
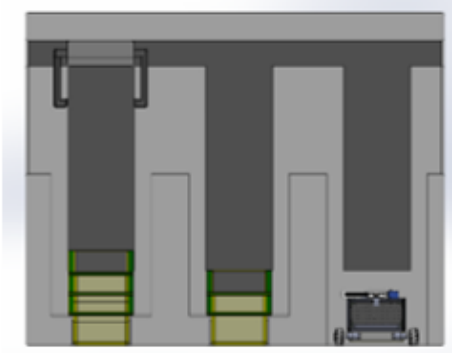
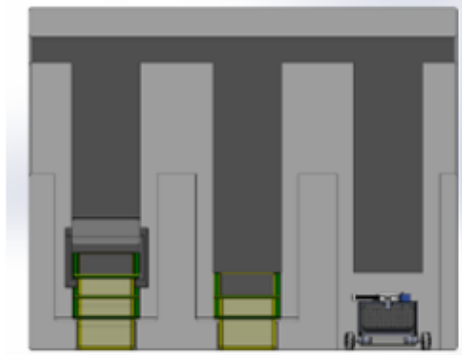
(o) Unload collected box	(p) Hub-gripper left
	
(q) Grasp new collection box	(r) Hub-gripper up
	
(s) Hub-gripper right	(t) Reload collection box
	

Figure 143 Application - Garbage Collection 4

Appendix

Appendix I. ASME SDC Competition Rules and Regulation

ASME Student Design Competition
2019 Contest

The Pick-and-Place Race

Design Problem Setup

The 2019 Student Design Competition challenges your imagination and technical design skills to create a device that can quickly but carefully secure a variety of different balls that will be balancing on tube stands in the middle of a flat playing surface. You must construct a single remotely controlled device to collect as many balls as possible, and place them in a collection area – you must do this quickly, but avoid having balls fall off their stand and hit the ground. The competition will have an initial round where your device will run without competition, and then devices will compete against each other in a knockout format.

The constraints and competition procedures for all devices are as follows:

Pre-Game Rules

1. Students participating in the competition must be undergraduate engineering students (any engineering discipline is allowed) and must be ASME members. There is no limit on the number of students on a team.
2. At the start of the competition, teams **must provide a sizing box for your device** and any tools your team would use to make minor repairs during the competition. Throughout the competition, your device, controls, any extra batteries, and any tools must fit within your rigid sizing box. This box must be no more than 50 cm x 50 cm x 50 cm (internal dimensions), but *teams should minimize actual box size, volume is a tie-breaker in the competition.*
3. Your device will be stored inside your sizing box throughout all of the rounds of the competition. Teams will have one minute to remove your device from the box to compete in each round. No modifications to the device are permitted during this setup.
4. All energy for the device must be provided by rechargeable batteries. No other forms of stored energy (such as pre-compressed springs or gas) are allowed unless the stored energy of this component is returned to the initial state (for example an initially compressed spring must be re-compressed using the energy from the battery).
5. Teams may replace batteries between rounds, however replacement batteries must be identical to the original, mounted in the same way to the device, and stored in the sizing box throughout the competition.
6. Your device must be controlled via remote control through a transmitter/receiver radio link. Transmitter/receiver radio links may be any commercially available model controller. Radio transmitter batteries do not have to be rechargeable. All radio controllers will be shut off/stored within the team's box during the competition unless the team is competing.
7. Communication between controller and device must be able to be secured to allow for at least 3 other teams simultaneously using live controllers at other games taking place in the same auditorium area.

8. Flying devices are not allowed. Devices must remain intact throughout the game – for example, the device may not split and retrieve multiple balls at once.

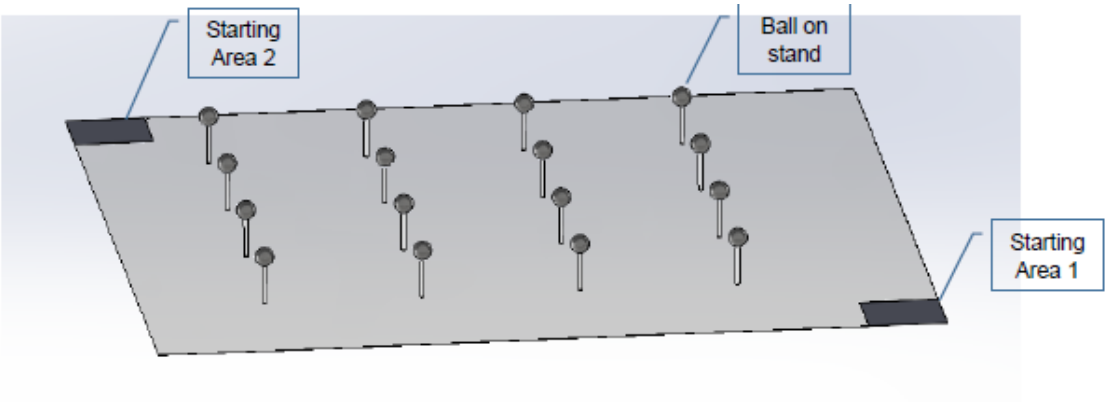
Game Rules

9. The playing surface dimensions are 5 meters x 5 meters, with boundaries marked by tape on the floor. The device will start each game in a 50 cm x 50 cm area in one corner of the playing surface, also marked by tape on the ground. See Figure 1 below with comments.
10. There will be sixteen balls each resting on top of a cylindrical stand at the start of each game. The balls will be at least 2.7 grams and 40 mm in diameter (a table tennis ball) and no larger than 650 grams and 250 mm in diameter (a basketball). Teams will not know the exact details of the balls (sizes or distribution of types of balls) until the competition – designs need to be flexible.
11. The sixteen cylinders will be approximately 3 – 5 cm in diameter (*probably PVC pipe*) and will be 20 cm tall. Judges may create a small base for each cylinder, but teams should expect that the stands will be easy to knock over if a device runs into them.
12. The playing surface will be level, and may be either hard surface or carpet typically found in public areas.
13. Just prior to the start of each game, judges will randomly place all the balls on the cylinders and immediately start the game. All games will last **5 minutes**.
14. Teams will earn/lose points as follows:
    - Plus two points for **collecting** a ball from its stand and securing it on their device
    - Plus three points for **placing** a ball in the team's starting area
    - Minus one point for every ball that is **knocked off** its stand and hits the ground
15. Once a ball has been scored (either secured, secured and placed, or knocked off stand) it can no longer be played during the rest of the game. Balls knocked off their stand will remain on the playing surface floor for the remainder of the game.
16. Teams earn two collecting points when a ball is secure within their device *for at least two seconds*; if the ball subsequently falls to the ground this does not change the scoring but placement points in the starting area cannot be earned for that ball.
17. Rules for earning the additional three placement points are as follows:
    - Only balls first secured within the device are eligible for placement points
    - Teams may either secure and then place one ball at a time, or may secure multiple balls and place them simultaneously
    - Placing points are earned when a ball is placed and remains stationary inside the starting area; if the ball is subsequently displaced this does not change the scoring.
    - The placed ball must rest on the ground within the scoring area, not touching the device
    - Balls that have been secured may be pushed on the ground into the scoring area, however any ball that leaves the 5m x 5m playing surface becomes ineligible.
18. If all sixteen balls are scored in less than 5 minutes, the game will end.
19. All teams will complete in one initial game with no other device present. The team score earned will be used to seed the first knockout round of the competition.

20. During the knockout stage, two teams will compete, attempting to earn the most points in the game. The team with the most points will advance to the next round. The following rules apply to team interactions during the knockout stage:
    - Only minimal, incidental contact between devices will be allowed. Reckless behavior will result in a time penalty. Devices should be robust to survive minor collisions.
    - When one team's device is attempting to secure a ball, the other team must not interfere. Striking another device while it is in the process of securing a ball will earn the interfering team the one point penalty if the ball falls to the ground.
    - If judges are not able to determine which device causes a ball to fall off the stand, then no penalty will be awarded either team.
    - Teams are allowed to play defense and block the other device from returning to their starting area or moving to a desired location, as long as there is no intentional contact.
    - Devices must stay within the outside boundaries of the 5m x 5m playing surface. If a device **entirely leaves** the playing surface (all device contact with the ground is outside the boundary lines) the team will receive an **official caution** – the team must then remain motionless for 30 seconds before being allowed to resume competing
    - Intentional *fouls* and overly aggressive behavior will be stopped by the judges. Excessive contact with the other device may result in an **official caution** – the team must then remain motionless for 30 seconds before being allowed to resume competing.

Competition Scoring Rules

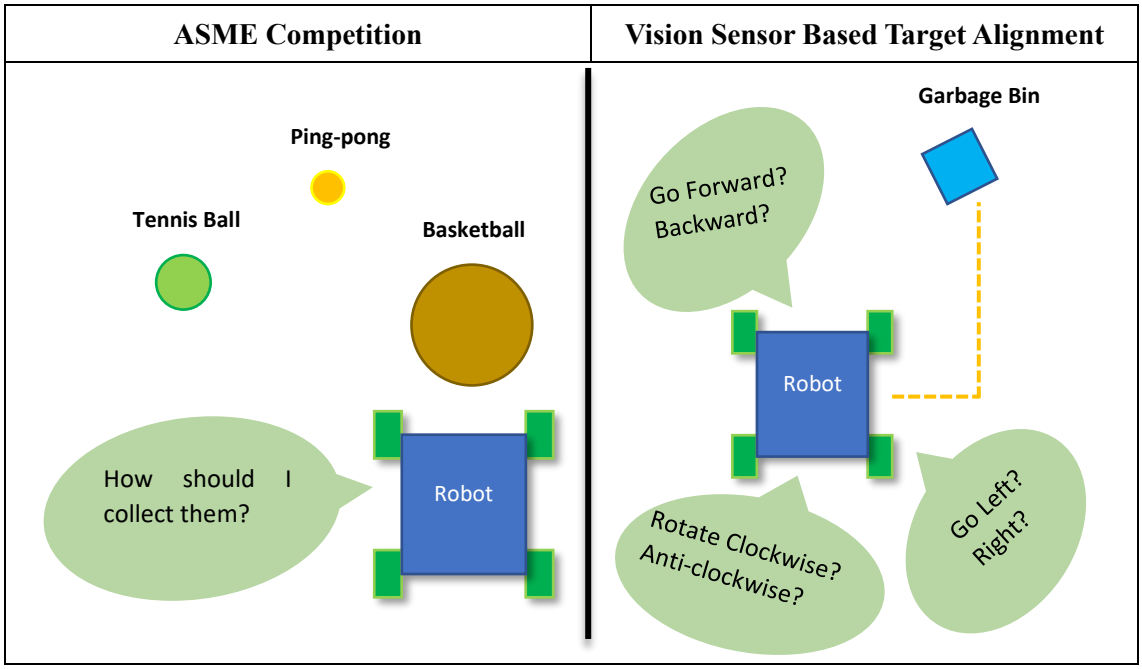
21. Teams will receive a score for their initial individual round – points for all balls collected/placed, minus deductions for balls knocked off stands.
22. The initial round score will be used to seed the knockout round. In the knockout round, highest scoring team will compete against lowest scoring team, 2nd highest against 2nd lowest, etc. For teams with the same score, the tiebreaker will be the volume of the team sizing box (smallest volume is best).
23. If there is an odd number of teams in any round, the two lowest scoring teams will compete against each other, and that winner will then compete against the highest scoring team in the prior round.
24. For all subsequent knockout rounds, each winning team's score earned will be used to seed the subsequent round. If necessary, the tiebreaker will be total number of points scored in the initial solo round (if still tied, box volume will be used). Again, highest score will compete against lowest score, 2nd highest against 2nd lowest, etc.
25. Knockout rounds will continue until the final match between the top two devices. The team with the most points is the champion. If the two teams are tied after 5 minutes, the course will be reset and the teams will compete in a one minute shootout. If still tied, shootouts will be repeated until there is a winner.



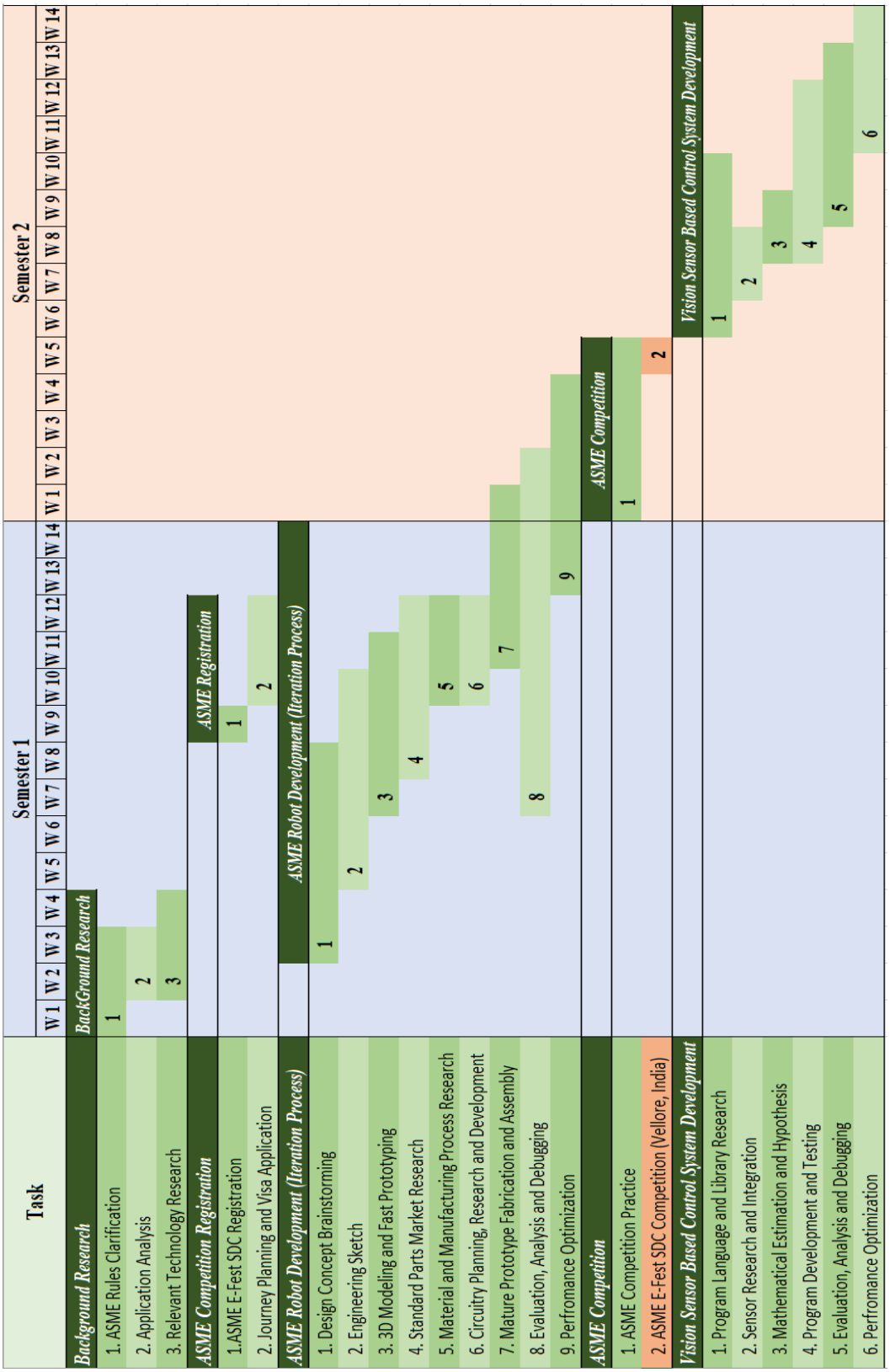
At the start of the 5 minute game, a team's device will begin in their designated starting area, a 50 cm x 50 cm square shown in two corners of the 5 m x 5 m playing surface. Teams will score points by collecting any of the 16 balls on stands, and attempt to bring the balls back to their own starting area. Teams are also trying to avoid knocking balls off the stands onto the ground.

The sixteen stands and balls are equally spaced 1 meter apart from each other and the overall game boundaries. The balls shown in Figure 1 are all the same size, this will not be the case for the actual competition. See the rules for the range of ball sizes. Distribution of the balls used will change throughout the competition.

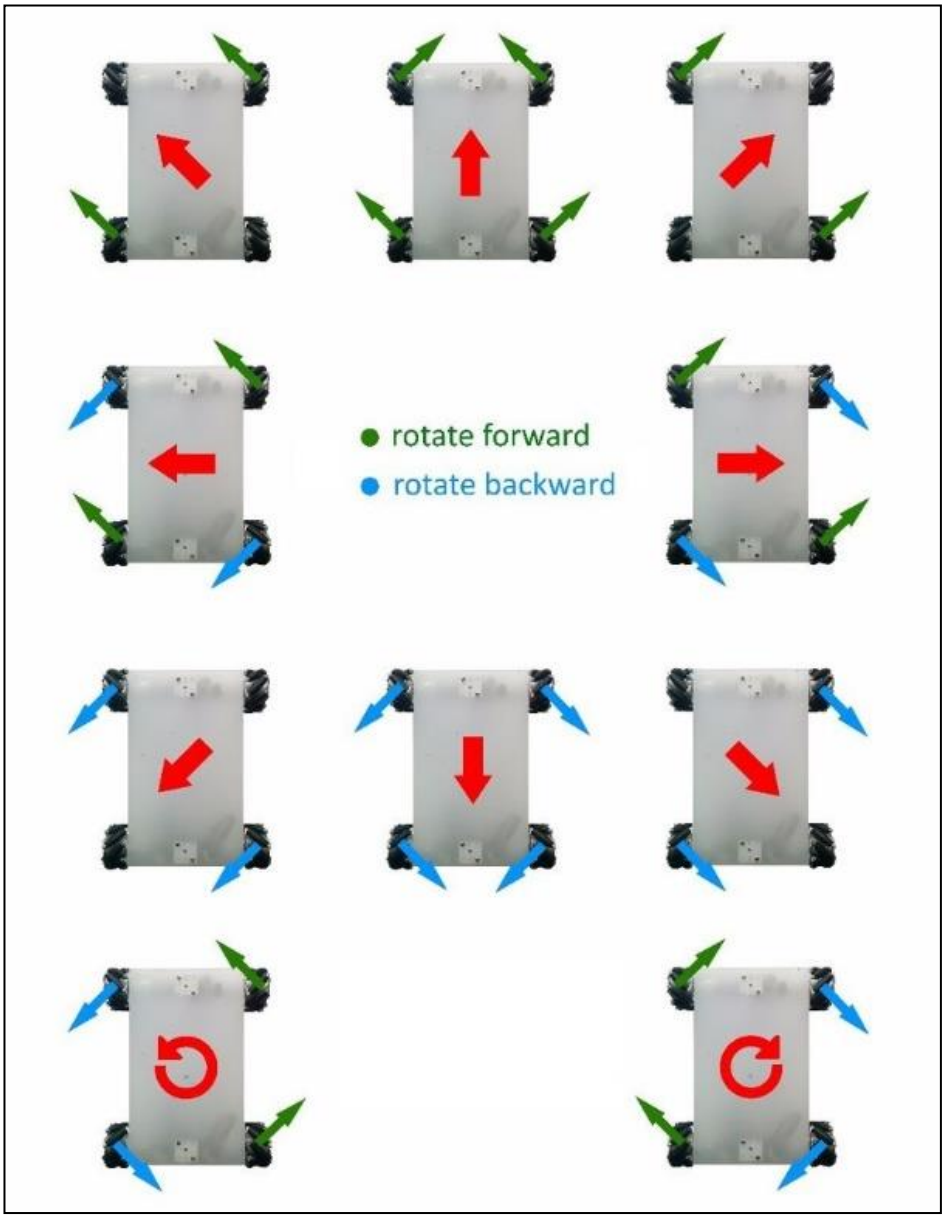
Appendix II. Project Scope Illustration



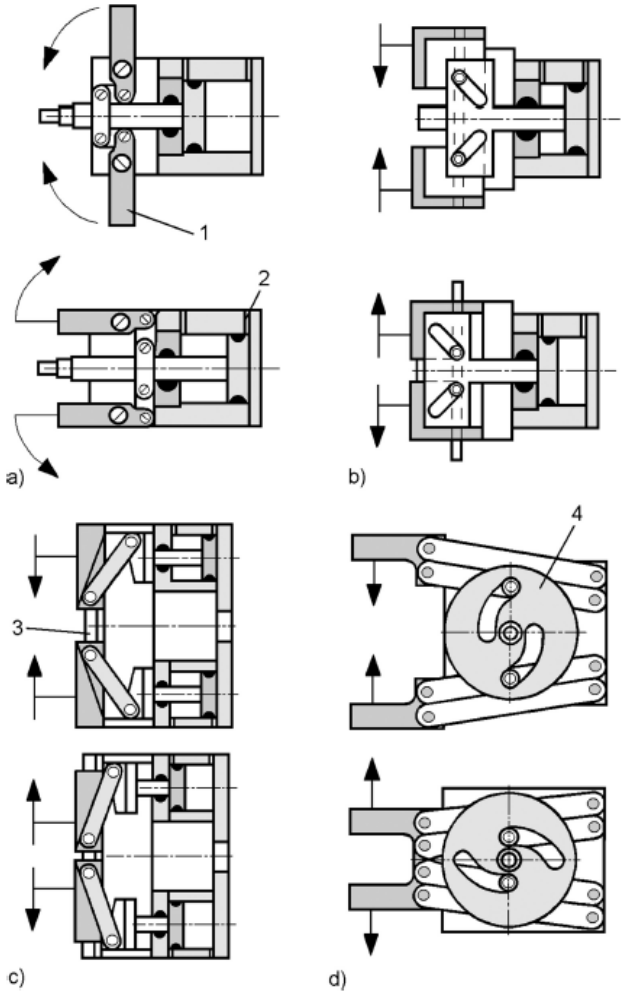
Appendix III. Gantt Chart



Appendix IV. Control of Mecanum Drive Platform Using 4 Motors

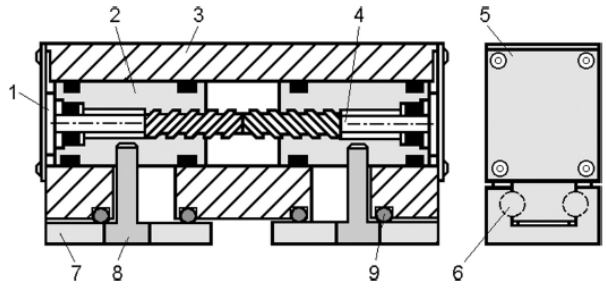


Appendix V. Kinematic System of Rigid Gripper

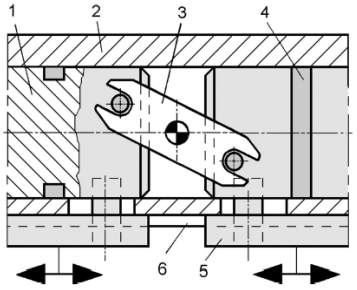


1 basic jaw or finger
2 pneumatic cylinder
3 straight guideway
4 cam disk

a) angle gripper with toggle lever mechanics
b) parallel gripper with roller link
c) parallel gripper with two pneumatic cylinders
d) parallel gripper with cam disk



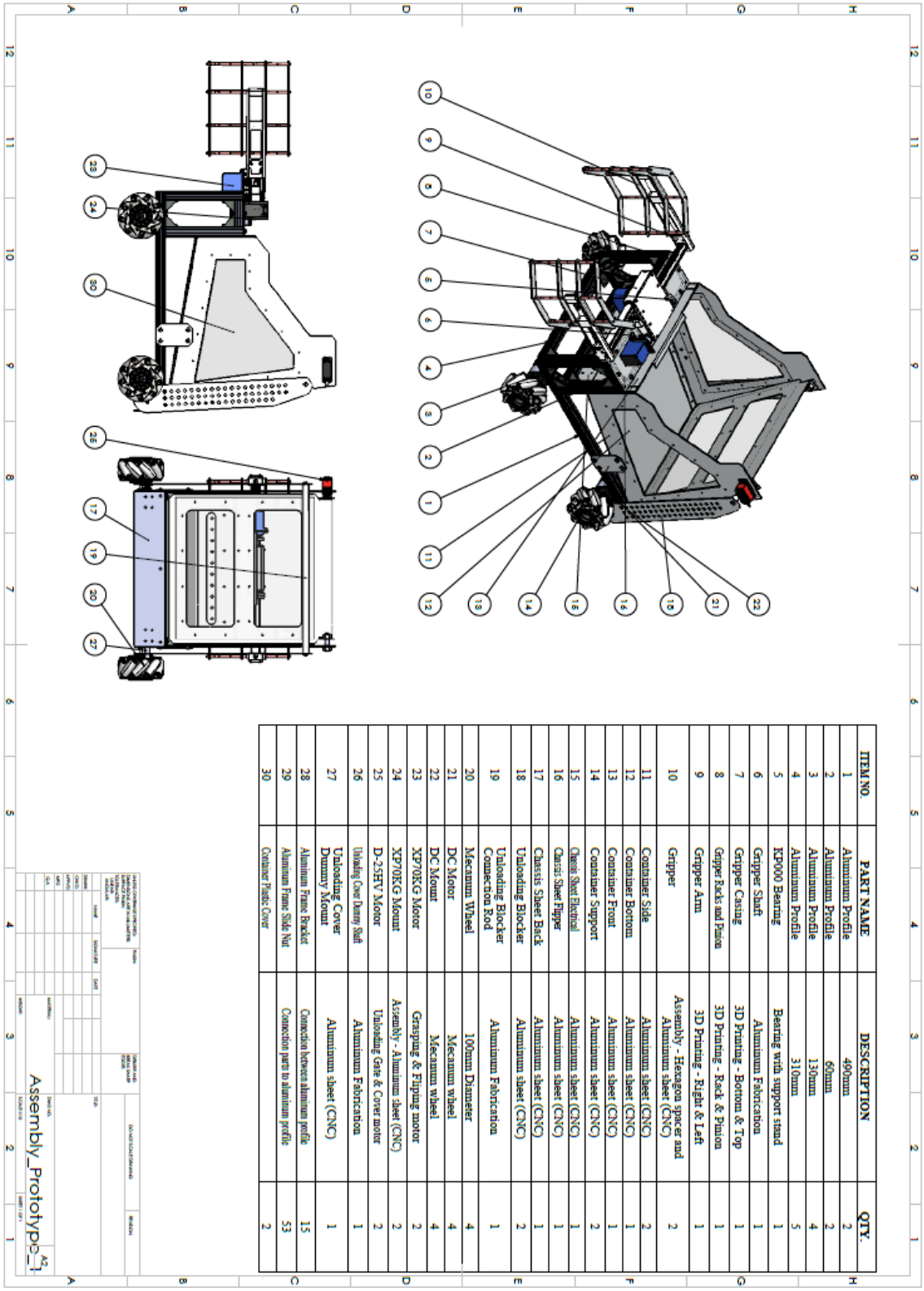
1 cover plate
2 pneumatic cylinders
3 housing
4 thread shaft
5 sealing plate
6 basic jaw guides
7 basic jaws
8 securing bolts
9 seals



1 pneumatic cylinder
2 housing or cylindrical boring
3 synchronous lever
4 sealing
5 basic jaw
6 jaw guides

Fig. 1.7: Synchronization by means of a double swing-yoke-drive (scotch-yoke drive)

Appendix VI. Bill of Material (BOM)



ITEM NO.	PART NAME	DESCRIPTION	QTY.
1	Aluminium Profile	490mm	2
2	Aluminium Profile	60mm	2
3	Aluminium Profile	130mm	4
4	Aluminium Profile	310mm	5
5	KP000 Bearing	Bearing with support stand	1
6	Gripper Shaft	Aluminium Fabrication	1
7	Gripper Casing	3D Printing - Bottom & Top	1
8	Gripper Rocks and Pinion	3D Printing - Rack & Pinion	1
9	Gripper Arm	3D Printing - Right & Left	1
10	Gripper	Assembly - Hexagon spacer and Aluminium sheet (CNC)	2
11	Container Side	Aluminium sheet (CNC)	2
12	Container Bottom	Aluminium sheet (CNC)	1
13	Container Front	Aluminium sheet (CNC)	1
14	Container Support	Aluminium sheet (CNC)	2
15	Chassis Sheet Electrical	Aluminium sheet (CNC)	1
16	Chassis Sheet Flipper	Aluminium sheet (CNC)	1
17	Chassis Sheet Back	Aluminium sheet (CNC)	1
18	Unloading Blocker	Aluminium sheet (CNC)	2
19	Unloading Blocker Connection Rod	Aluminium Fabrication	1
20	Mecanum Wheel	100mm Diameter	4
21	DC Motor	Mecanum wheel	4
22	DC Mount	Mecanum wheel	4
23	XP70JXG Motor	Grasping & Flipping motor	2
24	XP70JXG Mount	Assembly - Aluminum sheet (CNC)	2
25	D-25HV Motor	Unloading Gate & Cover motor	2
26	Unloading Cover Dummy Shaft	Aluminium Fabrication	1
27	Unloading Cover Dummy Mount	Aluminium sheet (CNC)	1
28	Aluminium Frame Bracket	Connection between aluminium profile	15
29	Aluminium Frame Slide Nut	Connection parts to aluminium profile	53
30	Container Plastic Cover		2

Appendix VII. D-25HV Servo Motor Specification

4. 電 氣 特 性
Electrical Specification (Function of the Performance) :

No.	項目 item	6.0V	7.4V
4-1	空載轉速 Operating speed (at no load)	0.18 sec/60°	0.16 sec/60°
4-2	空載電流(低速到快速) Running current at no load(Slow to Fast)	80~100 mA	100~120 mA
4-3	停止扭力 Stall torque (at locked)	20.0 kg-cm	25.0 kg-cm
4-4	停止電流 Stall current (at locked)	3100 mA	3800 mA
4-5	待機電流 Idle current (at stopped)	4 mA	5 mA

注：项目 4-2 定义平均值时，伺服器无负荷运行
Note: Item 4-2 definition is average value when the servo running with no load

5. 機 械 特 性
Mechanical Specification :

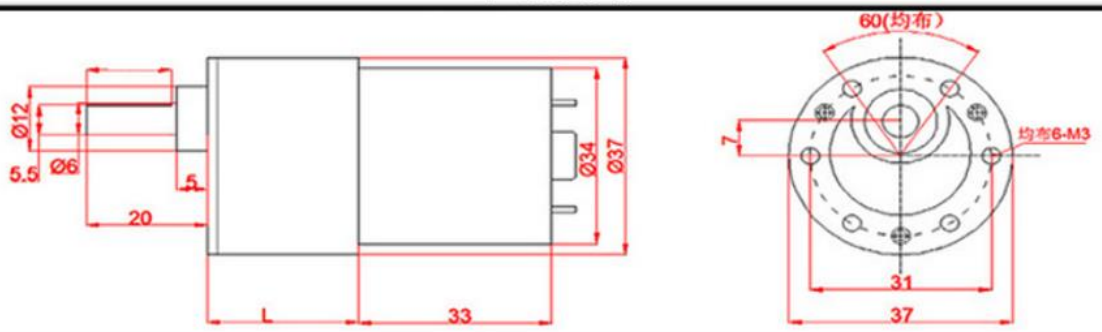
No.	項目 item	規格 standard
5-1	外觀尺寸 Overall Dimensions	见附件 See the drawing
5-2	機構極限角度 Limit angle	180° ± 10°
5-3	重量 Weight	75 ± 1g
5-4	導線規格 Connector wire gauge	#28 AWG 0.08*60
5-5	導線長度 Connector wire length	300 ± 5 mm
5-6	舵片規格 Horn gear spline	ψ5.8*25T
5-7	舵片種類 Horn type	一字、十字，圓盘，星型 Double Arm,Cross , Disk , Star
5-8	減速比 Reduction ratio	1/381
5-9	齿轮材质 Materials of the gear	镀钛齿轮 Titanium & 7075 Alu

Appendix VIII. XP-70HV Servo Motor

XP70HV DATA SHEET	
壳体Case	铝中壳+塑料上下盖 CNC aluminum middle case + plastic top and bottom shell
齿轮Gears	台湾进口碳钢齿 full steel gears made in Taiwan
遥控器角度Remote Angle	XP70HV; 90度到120度 90 to 120, 120 by the pwm 900-2100us
	XP70HV-320; 160度到180度 160 to 180, 180 by the pwm 900-2100us
控制板角度Servo MB	XP70HV; 180度 180 by pwm 650-2350us(NOT STANDARD PULSE WIDTH)
	XP70HV-320; 320度 320 by the pwm 500-2500us
死区dead	2 μs
工作频率 Frequnce	1520μs / 330hz
电机motor	高速有刷电机 high speed core motor
轴承bearing	两个滚珠轴承 2*ball bearing
输出轴shaft	15T
电压working Voltage	6.0v-7.4v
速度6.0v Speed	0.13 sec/60°
速度7.4v Speed	0.11 sec/60°
扭矩6.0v Torque	56.7 kg.cm
扭矩7.4v Torque	70.1 kg.cm
尺寸 Size	65.8*30*57.4mm
重量 Weight	202g
线长 Wire length	JR 260mm
接线定义 Wire for	橙色Orange S / 红色Red + / 棕色Brown GND
保修 Warranty	30 days天(主板/马达/电位器) for micro chip/motor/Resistance Potentiometer)

Appendix IX. XD-37GB520

产品图纸



电机参数表

电机特性表-List of Motof Characteristics

型号	电压(DCV)	空载转速(rpm/min)	空载电流(A)	负载转速(rpm/min)	负载电流(A)	额定力矩(Kg.cm)	堵转力矩(Kg.cm)	堵转电流(A)	功率(W)	减速比(1:00)	减速箱长(L约MM)	重量(约g)
XD-37GB520	12	5	0.1	4	0.59	15	55	2.19	7	700	31	215
XD-37GB520		10	0.1	9	0.59	15	55	2.19	7	500	31	215
XD-37GB520		15	0.1	13	0.59	12	40	2.19	7	333.3	31	215
XD-37GB520		20	0.1	22	0.59	10	35	2.19	7	200	31	215
XD-37GB520		30	0.17	31	0.68	8	28	2.19	7	142.8	28	210
XD-37GB520		50	0.17	45	0.68	7	26	2.19	7	100	28	210
XD-37GB520		87	0.17	75	0.68	5	18	2.19	7	57.5	28	200
XD-37GB520		100	0.17	93	0.68	3.5	12	2.19	7	50	26	200
XD-37GB520		150	0.17	128	0.68	3	10	2.19	7	33.3	26	200
XD-37GB520		200	0.17	175	0.68	2	6	2.19	7	25	24	195
XD-37GB520		300	0.17	245	0.68	1.8	5	2.19	7	16.6	24	195
XD-37GB520		400	0.17	335	0.68	1.5	4.5	2.19	7	12.5	24	195
XD-37GB520		500	0.17	465	0.68	1	3	2.19	7	10	24	195
XD-37GB520		600	0.17	552	0.68	0.5	1.8	2.19	7	8.3	24	195
XD-37GB520	24	5	0.09	4	0.55	15	90	2.19	10	700	31	215
XD-37GB520		10	0.09	9	0.55	15	85	2.19	10	500	31	215
XD-37GB520		15	0.09	13	0.55	13	78	2.19	10	333.3	31	215
XD-37GB520		20	0.09	18	0.55	12	73	2.19	10	250	31	215
XD-37GB520		30	0.13	28	0.55	9	70	2.19	10	166.6	28	210
XD-37GB520		50	0.13	45	0.62	8	50	2.19	10	100	28	210
XD-37GB520		100	0.13	93	0.62	4	42	2.19	10	50	26	200
XD-37GB520		150	0.13	128	0.62	3.5	35	2.19	10	33.3	26	200
XD-37GB520		200	0.13	175	0.62	2.5	28	2.19	10	25	24	195
XD-37GB520		300	0.13	245	0.62	2	20	2.19	10	16.6	24	195
XD-37GB520		400	0.13	350	0.62	1.8	10	2.19	10	12.5	24	195
XD-37GB520		500	0.13	435	0.62	1.2	5	2.19	10	10	24	195
XD-37GB520		600	0.13	535	0.62	0.6	2.8	2.19	10	8.3	24	195

Appendix X. Arduino UNO Specifications

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Appendix XI. Battery Specification [60]

5000mAh 11.1v 3-Cell 25C LiPo Battery

TRAXXAS CERTIFIED LIPO POWER! Traxxas Certified iD-Equipped LiPo batteries are engineered to provide the punch and on-demand power for reaching the top speeds that Traxxas models are built to achieve. Traxxas Power Cell iD-Equipped LiPo batteries are engineered specifically to fit Traxxas models and maximize their full performance potential. Only Traxxas gives you more of what you want most: simple installation, a great price, and the most speed and run time available.

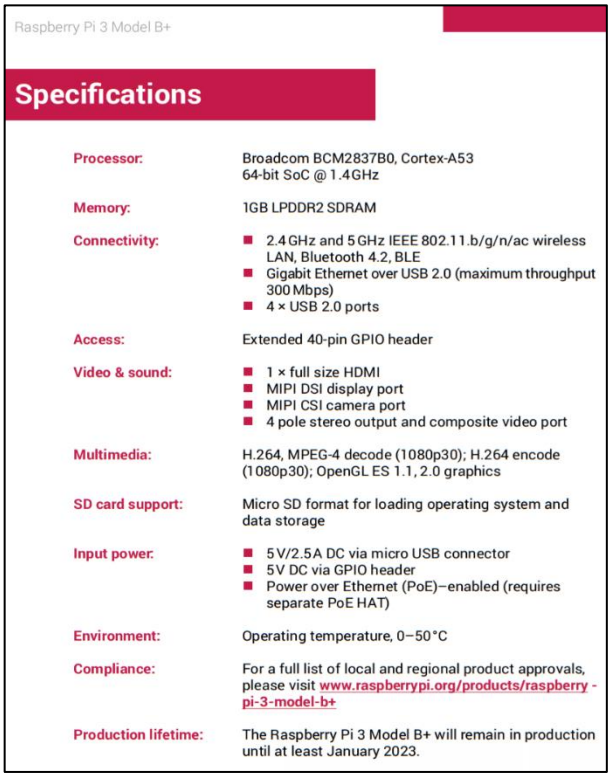
Type	Total Capacity	Voltage	C Rating	Length	Height	Width	Weight
LiPo	5000 mAh	11.1 V	25	135 mm	28.5 mm	44 mm	12.5 oz

7600mAh 7.4v 2-Cell 25C LiPo Battery

Traxxas iD Power Cell LiPo batteries offer the ultimate in power, "punch," and runtime with the highest possible capacity and voltage. 7600mAh 2-Cell LiPos provide 50% more runtime than 5000mAh, giving you more time for fast laps, jumps, and over-the-top RC fun.

Type	Total Capacity	Voltage	C Rating	Length	Height	Width	Weight
LiPo	7600 mAh	7.4 V	25	155 mm	25 mm	45 mm	13.2 oz

Appendix XII. Raspberry Pi 3 Model B+ Specifications



Raspberry Pi 3 Model B+

Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	■ 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ■ Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps) ■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	■ 1 × full size HDMI ■ MIPI DSI display port ■ MIPI CSI camera port ■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	■ 5V/2.5A DC via micro USB connector ■ 5V DC via GPIO header ■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50°C
Compliance:	For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+
Production lifetime:	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.

Appendix XIII. PS2X_lib.h

```
/******************************************************************
* Super amazing PS2 controller Arduino Library v1.8
*           details and example sketch:
*                 http://www.billporter.info/?p=240
*
* Original code by Shutter on Arduino Forums
*
* Revamped, made into lib by and supporting continued development:
* Bill Porter
* www.billporter.info
*
*      Contributers:
*           Eric Wetzel (thewetzel@gmail.com)
*           Kurt Eckhardt
*
* Lib version history
* 0.1 made into library, added analog stick support.
* 0.2 fixed config_gamepad miss-spelling
* added new functions:
* NewButtonState();
* NewButtonState(unsigned int);
* ButtonPressed(unsigned int);
* ButtonReleased(unsigned int);
* removed 'PS' from begining of ever function
* 1.0 found and fixed bug that wasn't configuring controller
* added ability to define pins
* added time checking to reconfigure controller if not polled enough
* Analog sticks and pressures all through 'ps2x.Analog()' function
```

```
* added:
* enableRumble();
* enablePressures();
* 1.1
* added some debug stuff for end user. Reports if no controller found
* added auto-increasing sentence delay to see if it helps
compatibility.
* 1.2
* found bad math by Shutter for original clock. Was running at 50kHz,
not the required 500kHz.
* fixed some of the debug reporting.
*     1.3
*      Changed clock back to 50kHz. CuriousInventor says it's suppose
to be 500kHz, but doesn't seem to work for everybody.
*     1.4
*           Removed redundant functions.
*           Fixed mode check to include two other possible modes the
controller could be in.
* Added debug code enabled by compiler directives. See below to enable
debug mode.
*           Added button definitions for shapes as well as colors.
*     1.41
*           Some simple bug fixes
*           Added Keywords.txt file
*     1.5
*           Added proper Guitar Hero compatibility
*           Fixed issue with DEBUG mode, had to send serial at once
instead of in bits
*     1.6
*           Changed config_gamepad() call to include rumble and
pressures options
*                 This was to fix controllers that will only go into
config mode once
*                 Old methods should still work for backwards
compatibility
* 1.7
*           Integrated Kurt's fixes for the interrupts messing with
servo signals
*           Reorganized directory so examples show up in Arduino IDE
menu
* 1.8
*           Added Arduino 1.0 compatibility.
* 1.9
* Kurt - Added detection and recovery from dropping from analog mode,
plus
* integreated Chipkit (pic32mx...) support
*
*
*
*This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or(at
your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
<http://www.gnu.org/licenses/>
*
****************************************************************/
```

```
// $$$$$$$$$$$$ DEBUG ENABLE SECTION $$$$$$$$$$$$$$$$
// to debug ps2 controller, uncomment these two lines to print out
debug to uart
//#define PS2X_DEBUG
//#define PS2X_COM_DEBUG

#ifndef PS2X_lib_h
 #define PS2X_lib_h

#if ARDUINO > 22
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

#include <math.h>
#include <stdio.h>
#include <stdint.h>
#ifdef __AVR__
 // AVR
 #include <avr/io.h>
 #define CTRL_CLK 4
 #define CTRL_BYTE_DELAY 3
#else
 // Pic32...
 #include <pins_arduino.h>
 #define CTRL_CLK 5
 #define CTRL_CLK_HIGH 5
 #define CTRL_BYTE_DELAY 4
#endif

//These are our button constants
#define PSB_SELECT 0x0001
#define PSB_L3 0x0002
#define PSB_R3 0x0004
#define PSB_START 0x0008
#define PSB_PAD_UP 0x0010
#define PSB_PAD_RIGHT 0x0020
#define PSB_PAD_DOWN 0x0040
#define PSB_PAD_LEFT 0x0080
#define PSB_L2 0x0100
#define PSB_R2 0x0200
#define PSB_L1 0x0400
#define PSB_R1 0x0800
#define PSB_GREEN 0x1000
#define PSB_RED 0x2000
#define PSB_BLUE 0x4000
#define PSB_PINK 0x8000
#define PSB_TRIANGLE 0x1000
#define PSB_CIRCLE 0x2000
#define PSB_CROSS 0x4000
#define PSB_SQUARE 0x8000

//Guitar button constants
#define UP_STRUM        0x0010
#define DOWN_STRUM            0x0040
#define STAR_POWER            0x0100
#define GREEN_FRET            0x0200
#define YELLOW_FRET           0x1000
```

```
#define RED_FRET        0x2000
#define BLUE_FRET       0x4000
#define ORANGE_FRET           0x8000
#define WHAMMY_BAR            8

//These are stick values
#define PSS_RX 5
#define PSS_RY 6
#define PSS_LX 7
#define PSS_LY 8

//These are analog buttons
#define PSAB_PAD_RIGHT 9
#define PSAB_PAD_UP 11
#define PSAB_PAD_DOWN 12
#define PSAB_PAD_LEFT 10
#define PSAB_L2 19
#define PSAB_R2 20
#define PSAB_L1 17
#define PSAB_R1 18
#define PSAB_GREEN 13
#define PSAB_RED 14
#define PSAB_BLUE 15
#define PSAB_PINK 16
#define PSAB_TRIANGLE 13
#define PSAB_CIRCLE 14
#define PSAB_CROSS 15
#define PSAB_SQUARE 16

#define SET(x,y) (x|=(1<<y))
#define CLR(x,y) (x&=(~(1<<y)))
#define CHK(x,y) (x & (1<<y))
#define TOG(x,y) (x^=(1<<y))

class PS2X {
 public:
 boolean Button(uint16_t); //will be TRUE if button is being pressed
 unsigned int ButtonDataByte();
 boolean NewButtonState();
 boolean NewButtonState(unsigned int); //will be TRUE if button was
JUST pressed OR released
 boolean ButtonPressed(unsigned int); //will be TRUE if button was
JUST pressed
 boolean ButtonReleased(unsigned int); //will be TRUE if button was
JUST released
 void read_gamepad();
 boolean read_gamepad(boolean, byte);
 byte readType();
 byte config_gamepad(uint8_t, uint8_t, uint8_t, uint8_t);
 byte config_gamepad(uint8_t, uint8_t, uint8_t, uint8_t, bool, bool);
 void enableRumble();
 bool enablePressures();
 byte Analog(byte);
 void reconfig_gamepad();

 private:
 inline void CLK_SET(void);
 inline void CLK_CLR(void);
 inline void CMD_SET(void);
 inline void CMD_CLR(void);
```

```
 inline void ATT_SET(void);
 inline void ATT_CLR(void);
 inline bool DAT_CHK(void);

 unsigned char _gamepad_shiftinout (char);
 unsigned char PS2data[21];
 void sendCommandString(byte*, byte);
 unsigned char i;
 unsigned int last_buttons;
 unsigned int buttons;

 #ifdef __AVR__
 uint8_t maskToBitNum(uint8_t);
 uint8_t _clk_mask;
 volatile uint8_t *_clk_oreg;
 uint8_t _cmd_mask;
 volatile uint8_t *_cmd_oreg;
 uint8_t _att_mask;
 volatile uint8_t *_att_oreg;
 uint8_t _dat_mask;
 volatile uint8_t *_dat_ireg;
 #else
 uint8_t maskToBitNum(uint8_t);
 uint16_t _clk_mask;
 volatile uint32_t *_clk_lport_set;
 volatile uint32_t *_clk_lport_clr;
 uint16_t _cmd_mask;
 volatile uint32_t *_cmd_lport_set;
 volatile uint32_t *_cmd_lport_clr;
 uint16_t _att_mask;
 volatile uint32_t *_att_lport_set;
 volatile uint32_t *_att_lport_clr;
 uint16_t _dat_mask;
 volatile uint32_t *_dat_lport;
 #endif

 unsigned long last_read;
 byte read_delay;
 byte controller_type;
 boolean en_Rumble;
 boolean en_Pressures;
};

#endif
```

Appendix XIII. Adafruit_MS_PWMServoDriver.h

```
/*************************************************
 This is a library for our Adafruit 16-channel PWM & Servo driver

 Pick one up today in the adafruit shop!
 ------> http://www.adafruit.com/products/815

 These displays use I2C to communicate, 2 pins are required to
 interface. For Arduino UNOs, thats SCL -> Analog 5, SDA -> Analog 4

 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!

 Written by Limor Fried/Ladyada for Adafruit Industries.
 BSD license, all text above must be included in any redistribution
 *************************************************/

#ifndef _Adafruit_MS_PWMServoDriver_H
#define _Adafruit_MS_PWMServoDriver_H

#if ARDUINO >= 100
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif


#define PCA9685_SUBADR1 0x2
#define PCA9685_SUBADR2 0x3
#define PCA9685_SUBADR3 0x4

#define PCA9685_MODE1 0x0
#define PCA9685_PRESCALE 0xFE

#define LED0_ON_L 0x6
#define LED0_ON_H 0x7
#define LED0_OFF_L 0x8
#define LED0_OFF_H 0x9

#define ALLLED_ON_L 0xFA
#define ALLLED_ON_H 0xFB
#define ALLLED_OFF_L 0xFC
#define ALLLED_OFF_H 0xFD


class Adafruit_MS_PWMServoDriver {
 public:
 Adafruit_MS_PWMServoDriver(uint8_t addr = 0x40);
 void begin(void);
 void reset(void);
 void setPWMFreq(float freq);
 void setPWM(uint8_t num, uint16_t on, uint16_t off);

 private:
 uint8_t _i2caddr;

 uint8_t read8(uint8_t addr);
```

```
 void write8(uint8_t addr, uint8_t d);
};

#endif
```

Appendix XIII. Adafruit_MotorShield.h

```
/*************************************************************
 This is the library for the Adafruit Motor Shield V2 for Arduino.
 It supports DC motors & Stepper motors with microstepping as well
 as stacking-support. It is *not* compatible with the V1 library!

 It will only work with https://www.adafruit.com/products/1483

 Adafruit invests time and resources providing this open
 source code, please support Adafruit and open-source hardware
 by purchasing products from Adafruit!

 Written by Limor Fried/Ladyada for Adafruit Industries.
 BSD license, check license.txt for more information.
 All text above must be included in any redistribution.
 *************************************************************/

#ifndef _Adafruit_MotorShield_h_
#define _Adafruit_MotorShield_h_

#include <inttypes.h>
#include <Wire.h>
#include "Adafruit_MS_PWMServoDriver.h"

//#define MOTORDEBUG

#define MICROSTEPS 16 // 8 or 16

#define MOTOR1_A 2
#define MOTOR1_B 3
#define MOTOR2_A 1
#define MOTOR2_B 4
#define MOTOR4_A 0
#define MOTOR4_B 6
#define MOTOR3_A 5
#define MOTOR3_B 7

#define FORWARD 1
#define BACKWARD 2
#define BRAKE 3
#define RELEASE 4

#define SINGLE 1
#define DOUBLE 2
#define INTERLEAVE 3
#define MICROSTEP 4

class Adafruit_MotorShield;

class Adafruit_DCMotor
{
 public:
 Adafruit_DCMotor(void);
```

```
 friend class Adafruit_MotorShield;
 void run(uint8_t);
 void setSpeed(uint8_t);

 private:
 uint8_t PWMpin, IN1pin, IN2pin;
 Adafruit_MotorShield *MC;
 uint8_t motornum;
};

class Adafruit_StepperMotor {
 public:
 Adafruit_StepperMotor(void);
 friend class Adafruit_MotorShield;

 void step(uint16_t steps, uint8_t dir, uint8_t style = SINGLE);
 void setSpeed(uint16_t);
 uint8_t onestep(uint8_t dir, uint8_t style);
 void release(void);
 uint32_t usperstep;

 private:
 uint8_t PWMApin, AIN1pin, AIN2pin;
 uint8_t PWMBpin, BIN1pin, BIN2pin;
 uint16_t revsteps; // # steps per revolution
 uint8_t currentstep;
 Adafruit_MotorShield *MC;
 uint8_t steppernum;
};

class Adafruit_Servo
{
 public:
 Adafruit_Servo(void);
 friend class Adafruit_MotorShield;
 void setServoPulse(double pulse);
 void writeServo(uint8_t angle);
 uint8_t readDegrees();

 private:
 uint8_t PWMpin;
 Adafruit_MotorShield *MC;
 uint8_t servonum,currentAngle;
};

class Adafruit_MotorShield
{
 public:
 Adafruit_MotorShield(uint8_t addr = 0x60);
 friend class Adafruit_DCMotor;
 void begin(uint16_t freq = 1600);

 void setPWM(uint8_t pin, uint16_t val);
 void setPin(uint8_t pin, boolean val);
 Adafruit_DCMotor *getMotor(uint8_t n);
 Adafruit_StepperMotor *getStepper(uint16_t steps, uint8_t n);
 Adafruit_Servo *getServo(uint8_t n);
 private:
 uint8_t _addr;
 uint16_t _freq;
```

```
 Adafruit_DCMotor dcmotors[4];
 Adafruit_StepperMotor steppers[2];
 Adafruit_MS_PWMServoDriver _pwm;
 Adafruit_Servo servos[4];
};

#endif
```

Appendix XIV. Our Arduino Code

```
#include <Wire.h>
#include <PS2X_lib.h>
#include <Adafruit_MotorShield.h>
#include <Adafruit_MS_PWMServoDriver.h>
volatile int i;
Adafruit_MotorShield AFMS =
Adafruit_MotorShield();
PS2X ps2x;
Adafruit_Servo *Servo1 = AFMS.getServo(1);
Adafruit_Servo *Servo3 = AFMS.getServo(3);
Adafruit_Servo *Servo4 = AFMS.getServo(4);
Adafruit_DCMotor *DCMotor_1 = AFMS.getMotor(1);
Adafruit_DCMotor *DCMotor_2 = AFMS.getMotor(2);
Adafruit_DCMotor *DCMotor_3 = AFMS.getMotor(3);
Adafruit_DCMotor *DCMotor_4 = AFMS.getMotor(4);
void setup()
{
AFMS.begin(50);
int error = 0;
do{
error = ps2x.config_gamepad(13,11,10,12,
true, true);
if(error == 0){
break;
}else{
delay(100);
}
}while(1);
i = 1;
}
void loop()
{
ps2x.read_gamepad(false, 0);
delay(30);
if (ps2x.Button(PSB_PAD_UP)) {
if (Servo1->readDegrees() < 150) {
Servo1->writeServo((Servo1->readDegrees() +
3));delay(1);
}
} else if (ps2x.Button(PSB_PAD_DOWN)) {
if (Servo1->readDegrees() > 0) {
Servo1->writeServo((Servo1->readDegrees() -
3));delay(1);
}
} else if (ps2x.Button(PSB_L1)) {
if (Servo3->readDegrees() < 90) {
Servo3->writeServo((Servo3->readDegrees() +
3));delay(1);
}
} else if (ps2x.Button(PSB_L2)) {
if (Servo3->readDegrees() > 0) {
Servo3->writeServo((Servo3->readDegrees() -
3));delay(1);
}
} else if (ps2x.Button(PSB_R1)) {
if (Servo4->readDegrees() < 120) {
Servo4->writeServo((Servo4->readDegrees() +
```

```
3));delay(1);
}
} else if (ps2x.Button(PSB_R2)) {
if (Servo4->readDegrees() > 0) {
Servo4->writeServo((Servo4->readDegrees() -
3));delay(1);
}
} else if (ps2x.Analog(PSS_LY) < 10) {
DCMotor_1->setSpeed(100);
DCMotor_1->run(FORWARD);
DCMotor_2->setSpeed(100);
DCMotor_2->run(FORWARD);
DCMotor_3->setSpeed(100);
DCMotor_3->run(FORWARD);
DCMotor_4->setSpeed(100);
DCMotor_4->run(FORWARD);
} else if (ps2x.Analog(PSS_LY) > 240) {
DCMotor_1->setSpeed(100);
DCMotor_1->run(BACKWARD);
DCMotor_2->setSpeed(100);
DCMotor_2->run(BACKWARD);
DCMotor_3->setSpeed(100);
DCMotor_3->run(BACKWARD);
DCMotor_4->setSpeed(100);
DCMotor_4->run(BACKWARD);
} else if (ps2x.Analog(PSS_LX) < 10) {
DCMotor_1->setSpeed(100);
DCMotor_1->run(BACKWARD);
DCMotor_2->setSpeed(100);
DCMotor_2->run(FORWARD);
DCMotor_3->setSpeed(100);
DCMotor_3->run(FORWARD);
DCMotor_4->setSpeed(100);
DCMotor_4->run(BACKWARD);
} else if (ps2x.Analog(PSS_LX) > 240) {
DCMotor_1->setSpeed(100);
DCMotor_1->run(FORWARD);
DCMotor_2->setSpeed(100);
DCMotor_2->run(BACKWARD);
DCMotor_3->setSpeed(100);
DCMotor_3->run(BACKWARD);
DCMotor_4->setSpeed(100);
DCMotor_4->run(FORWARD);
} else if (ps2x.Analog(PSS_RX) < 10) {
DCMotor_1->setSpeed(100);
DCMotor_1->run(BACKWARD);
DCMotor_2->setSpeed(100);
DCMotor_2->run(FORWARD);
DCMotor_3->setSpeed(100);
DCMotor_3->run(BACKWARD);
DCMotor_4->setSpeed(100);
DCMotor_4->run(FORWARD);
} else if (ps2x.Analog(PSS_RX) > 240) {
DCMotor_1->setSpeed(100);
DCMotor_1->run(FORWARD);
DCMotor_2->setSpeed(100);
DCMotor_2->run(BACKWARD);
DCMotor_3->setSpeed(100);
DCMotor_3->run(FORWARD);
DCMotor_4->setSpeed(100);
```

```
DCMotor_4->run(BACKWARD);
} else {
DCMotor_1->setSpeed(0);
DCMotor_1->run(RELEASE);
DCMotor_2->setSpeed(0);
DCMotor_2->run(RELEASE);
DCMotor_3->setSpeed(0);
DCMotor_3->run(RELEASE);
DCMotor_4->setSpeed(0);
DCMotor_4->run(RELEASE);
}
delay(10);
}void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
```

Appendix XV. HUE Filtering Program with Trackbar Controller [61]

```
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
	VideoCapture cap(0); //capture the video from web cam

	if (!cap.isOpened())  // if not success, exit program
	{
		cout << "Cannot open the web cam" << endl;
		return -1;
	}

	namedWindow("Control", CV_WINDOW_AUTOSIZE); //create a window ca
lled "Control"

	int iLowH = 0;
	int iHighH = 255;

	int iLowS = 0;
	int iHighS = 255;

	int iLowV = 0;
	int iHighV = 255;

	//Create trackbars in "Control" window

	//For our experiment, blue object was detected in the room with
HUE: 70~130, SATURATION: 200~255 and VALUEL: 60~255.

	cvCreateTrackbar("LowH", "Control", &iLowH, 255); //Hue (0 - 25
```

```
5)
	cvCreateTrackbar("HighH", "Control", &iHighH, 255);


	cvCreateTrackbar("LowS", "Control", &iLowS, 255); //Saturation
(0 - 255)
	cvCreateTrackbar("HighS", "Control", &iHighS, 255);


	cvCreateTrackbar("LowV", "Control", &iLowV, 255); //Value (0 - 2
55)
	cvCreateTrackbar("HighV", "Control", &iHighV, 255);


	while (true)
	{
		Mat imgOriginal;


		bool bSuccess = cap.read(imgOriginal); // read a new frame
 from video


		if (!bSuccess) //if not success, break loop
		{
			cout << "Cannot read a frame from video stream" << e
ndl;
			break;
		}


		Mat imgHSV;


		cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV); //Convert th
e captured frame from BGR to HSV


		Mat imgThresholded;


		inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHigh
H, iHighS, iHighV), imgThresholded); //Threshold the image



			//morphological opening (remove small objects from
 the foreground)
```

```
            erode(imgThresholded, imgThresholded, getStructuringElemen
t(MORPH_ELLIPSE, Size(5, 5)));

            dilate(imgThresholded, imgThresholded, getStructuringEleme
nt(MORPH_ELLIPSE, Size(5, 5)));


            //morphological closing (fill small holes in the foregroun
d)

            dilate(imgThresholded, imgThresholded, getStructuringEleme
nt(MORPH_ELLIPSE, Size(5, 5)));

            erode(imgThresholded, imgThresholded, getStructuringElemen
t(MORPH_ELLIPSE, Size(5, 5)));


            imshow("Thresholded Image", imgThresholded); //show the th
resholded image

            imshow("Original", imgOriginal); //show the original image


            if (waitKey(30) == 27) //wait for 'esc' key press for 30m
s. If 'esc' key is pressed, break loop

            {

                  cout << "esc key is pressed by user" << endl;

                  break;

            }

      }


      return 0;


}
```

Appendix XVI. Our Python Code

File - C:\Users\rudra\Desktop\FYP\codes\finalversion.py

```
import cv2
import numpy as np


def main():
    global area
    camera = cv2.VideoCapture(0)

    while True:
        ret, image = camera.read()

        if not ret:
            break

        frame_to_thresh = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

        v1_min=0
        v2_min=144
        v3_min=155
        v1_max=18
        v2_max=227
        v3_max=220

        w1_min = 87         #new
        w2_min = 76         #new
        w3_min = 92         #new
        w1_max = 138        #new
        w2_max = 232        #new
        w3_max = 164        #new

        #threshold the image to get only target value colours.
        thresh = cv2.inRange(frame_to_thresh, (v1_min, v2_min, v3_min), (v1_max, v2_max, v3_max))
        thresh1 = cv2.inRange(frame_to_thresh, (w1_min, w2_min, w3_min), (w1_max, w2_max, w3_max))  #new

        kernel = np.ones((5, 5), np.uint8)
        mask = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
        mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
        mask1 = cv2.morphologyEx(thresh1, cv2.MORPH_OPEN, kernel)       #new
```

```
39          mask1 = cv2.morphologyEx(mask1, cv2.MORPH_CLOSE,
    kernel)         #new
40
41
42
43          # find contours in the mask and initialize the
    current
44          # (x, y) center of the ball
45          cnts = cv2.findContours(mask.copy(), cv2.
    RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
46          center = None
47
48          # only proceed if at least one contour was found
49          if len(cnts) > 0:
50              # find the largest contour in the mask, then
    use
51              # it to compute the minimum enclosing circle
    and
52              # centroid
53              c = max(cnts, key=cv2.contourArea)
54              ((x, y), radius) = cv2.minEnclosingCircle(c)
55              M = cv2.moments(c)
56              center = (int(M["m10"] / M["m00"]), int(M["m01
    "] / M["m00"]))
57              area = np.pi*radius*radius
58
59              # only proceed if the radius meets a minimum
    size
60              if radius > 10:
61                  # draw the circle and centroid on the
    frame,
62                  # then update the list of tracked points
63                  cv2.circle(image, (int(x), int(y)), int(
    radius), (0, 255, 255), 2)
64                  cv2.circle(image, center, 3, (0, 0, 255),
    -1)
65                  cv2.putText(image, "centroid", (center[0]
    + 10, center[1]), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (0, 0,
    255),
66                                  1)
67                  cv2.putText(image, "(" + str(center[0]) +
    "," + str(center[1]) + ")", (center[0] + 10, center[1] +
    15),
68                                  cv2.FONT_HERSHEY_SIMPLEX, 0.4,
     (0, 0, 255), 1)
```

```
 69
 70          npImg = np.asarray(mask1)  # No copying takes
    place       #new
 71
 72          coordList = np.argwhere(npImg == 255)
             #new
 73          numWhitePoints = len(coordList)
             #new
 74          num1 = 0
             #new
 75          num2 = 0
             #new
 76          for i in range(numWhitePoints):
             #new
 77              if coordList[i][1] < 320:
             #new
 78                  num1 = num1 + 1
             #new
 79              elif coordList[i][1] > 320:
             #new
 80                  # else
    :                                                #new
 81                  num2 = num2 + 1
             #new
 82
 83          #print("num1: ", num1)
 84          #print("num2: ", num2)
 85
 86          # show the frame to our screen
 87          cv2.imshow("Original", image)
 88          cv2.imshow("Thresh", thresh)
 89          cv2.imshow("Mask", mask)
 90          cv2.imshow("Thresh1", thresh1)       #new
 91          cv2.imshow("Mask1", mask1)           #new
 92
 93
 94          if cv2.waitKey(400) & 0xFF is ord('q'):
 95              cv2.destroyAllWindows()
 96              break
 97
 98          if center:
 99              #print("area =", area)
100              #print("x= ", center[0],"y= ",center[1])
101              if center[0] in range(360,640):
102                  print("Move Right")
```

File - C:\Users\rudra\Desktop\FYP\codes\finalversion.py

```
103             elif center[0] in range(0,280):
104                 print("Move Left")
105             else:
106                 print("Centre")
107                 pass
108 
109             if 100<area<23000:
110                 print ("Move Forward")
111             elif 82000<area<90000:
112                 print ("Move Backward")
113             else :
114                 print("Distance OK")
115 
116                 if (num1/num2)<0.95:
    #new
117                     print("rotate counterclockwise")
    #new
118 
119                 elif (num1/num2)>1.05:
    #new
120                     print("rotate clockwise")
    #new
121 
122                 else:
    #new
123                     print("GO")
    #new
124                     #break
                                    #new
125             print (" \n***************")
    #new
126 
127         else:
128             continue
129 
130 if __name__ == '__main__':
131     main()
```

Appendix XVII. ASME Preliminary Round

Preliminary Round		Points Scored				Penalty		Game
Team		Collected	Points	Placed	Points	Balls Dropped	Points	Total
22 IIT	Avneesh Upadhayay	14	28	14	42	2	2	68
4 IIT INDORE	Himanshu verma	13	26	13	39	3	3	62
11 vit university	aabhas singhal	12	24	10	30	4	4	50
13 VEER SURENDRA SAI UNIV	ANEEK SINGHA ROY	11	22	11	33	5	5	50
6 IIT BOMBAY	Atharva Jaipurkar	10	20	9	27	5	5	42
2 The Hong Kong Polytechnic U	Hung Ming Roy CHOW	5	10	5	15		0	25
14 osmania university	krishna jadhav	4	8	3	9	6	6	11
3 Maulana Abdul Kalam Azad U	Simran Saha	2	4		0	1	1	3
7 National Institute of Technolo	Satish Ranjan Pradhan	1	2		0		0	2
5 National Institute of Technolo	Harshit Singh	2	4		0	2	2	2
12 Siksha O Anusandhan Univer	Suraj Kumar	3	6		0	4	4	2
15 Manipal University	Akshat Singhania		0		0		0	0
16 APJ Abdul Kalam Technologi	Niranjana M T		0		0		0	0
18 MPSTME NMIMS	Suyash Purohit		0		0		0	0
20 B.P.U.T., ROURKELA	SANKET KUMAR NAYAK		0		0		0	0
10 Kerala Technical University	Harish Gireesan		0		0	1	1	-1
19 SVKM's NMIMS	Mohit Jain		0		0	1	1	-1
1 APJ Abdul Kalam Technologi	Alan Kurian		0		0	2	2	-2
9 Shiv Nadar University	Rajat Naskar		0		0	6	6	-6
8 KTU	Tom Joseph		0		0	9	9	-9
17 National Institute of Technolo	Sanju Paul		0		0		0	DNS
21 Svkm's nmims	Aditya pujari		0		0		0	DNS
23 JNTU-H	Mohd Zainul abiduddin		0		0		0	DNS

Appendix XVII. ASME 1st Knockout Round

	Round of 16 Team			Points Scored			Penalty		Game	Position
			Collected	Points	Placed	Points	Balls Dropped	Points	Total	in Round
22	IIT	Avneesh Upadhayay	13	26	11	33	0	0	59	3
15	Manipal University	Akshat Singhania	1	2	0	0	2	2	0	
4	IIT INDORE	Himanshu verma	10	20	10	30	2	2	48	6
5	National Institute of Technology, Patna	Harshit Singh	2	4	0	0	1	1	3	
11	vit university	aabhas singhal	14	28	10	30	1	1	57	4
1	APJ Abdul Kalam Technological University	Alan Kurian	1	2	0	0	0	0	2	
13	VEER SURENDRA SAI UNIVERSITY OF TECHNOLOGY	ANEEK SINGHA ROY	11	22	11	33	2	2	53	5
8	KTU	Tom Joseph	0	0	0	0	3	3	-3	
6	IIT BOMBAY	Atharva Jaipurkar	13	26	13	39	0	0	65	1
10	Kerala Technical University	Harish Gireesan	1	2	0	0	2	2	0	
2	The Hong Kong Polytechnic University	Hung Ming Roy CHOW	12	24	12	36	0	0	60	2
12	Siksha O Anusandhan University	Suraj Kumar	1	2	1	3	3	3	2	
14	osmania university	krishna jadhav	7	14	7	21	0	0	35	7
9	Shiv Nadar University	Rajat Naskar	5	10	5	15	4	4	21	
3	Maulana Abdul Kalam Azad University of Technology	Simran Saha	2	4	2	6	3	3	7	8
7	National Institute of Technology, Rourkela	Satish Ranjan Pradhan	2	4	0	0	2	2	2	

Appendix XVII. ASME 2nd Knockout Round

Round of 8			Points Scored			Penalty		Game	Scores & Position of Round 2 for reference
	Team		Collected	Points	Placed	Points	alls Droppe	Points	Total	
6	IIT BOMBAY	Atharva Jaipurkar	0	0	0	0		0	65	1
3	Maulana Abdul Kalam Azad University of Technology	Simran Saha	0	0	0	0		0	7	8
2	The Hong Kong Polytechnic University	Hung Ming Roy CHOW	0	0	0	0		0	60	2
14	osmania university	krishna jadhav	0	0	0	0		0	35	7
22	IIT	Avneesh Upadhayay	0	0	0	0		0	59	3
4	IIT INDORE	Himanshu verma	0	0	0	0		0	48	6
11	vit university	aabhas singhal	0	0	0	0		0	57	4
13	VEER SURENDRA SAI UNIVERSITY OF TECHNOLOGY	ANEEK SINGHA ROY	0	0	0	0		0	53	5

Appendix XVIII. Summary of Contribution: CHOW Hung Ming Roy

Summary of Contribution

Project: ASME Competition Robot Development and Vision Based Target Alignment

Name of Student: CHOW Hung Ming Roy

Names of group-mate(s): SHIN Jiho, SOMESHWAR Rudra Ajay

This project involves the following tasks: (a) Background Research; (b) ASME Hardware Research & Design Development; (c) ASME Software Research & Design Development; (d) Iterative Evaluation, Analysis & Modification; (e) Standard Parts and Material Research & Selection; (f) Parts Fabrication, Manufacturing & Assembly; (g) Vision Sensor Integrated Image Processing System Development; (h) Report Writing.

I mainly took the responsibility in the literature review, manufacturing process of the robot, circuitry of the electronic parts and evaluation on the robot performance. Therefore, I mainly contributed in 'Literature Review' in **Chapter 2**, 'ASME Robot Hardware Design and Prototype Development' **Chapter 3. Section 3.7.2, 3.8, 3.9** and 'Result and Discussion' **Chapter 8. Section 8.2, 8.3.3**. These sections demonstrate how the team has accessed other similar products in the market and evaluate on them, manufacturing the prototype to a working robot and also the ability to reflect on the team's performance during the practice for ASME or additional functions added afterwards.

Also, I actively assisted in analysis of the standard part selection (**Section 3.7.1**) for the motor modification and some analysis on some of the calculations and SolidWorks analysis, which my group-mate took the lead as a hardware developer. I actively assisted my team in **Chapter 3** and **Chapter 4**, including the research on Arduino UNO and the

motor shield that we have utilized for better understanding in circuitry and its implementation in our robot.

Other than that, I was in charge in merging the format of the final report with all the image captions, table captions and references well established, such that presentation is better.

We worked closely together for background research, problem identification and setting the objectives, and thus all members contributed in writing **Chapter 1**.

Appendix XVIII. Summary of Contribution: SHIN Jiho

Summary of Contribution

Project: ASME Competition Robot Development and Vision Based Target Alignment

Name of Student: SHIN Jiho

Names of group-mate(s): Chow Hung Ming Roy, Someshwar Rudra Ajay

This project involves the following tasks: (a) Background Research; (b) ASME Hardware Research & Design Development; (c) ASME Software Research & Design Development; (d) Iterative Evaluation, Analysis & Modification; (e) Standard Parts and Material Research & Selection; (f) Parts Fabrication, Manufacturing & Assembly; (g) Vision Sensor Integrated Image Processing System Development; (h) Report Writing.

I mainly took the responsibility in 3D modeling of the robot, parts and material research and its prototype development, and thus I mainly contributed in ′ASME Robot Hardware Design and Prototype Development′ Chapter 3. Section 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.9. These sections demonstrate how the team has developed the robot from the sketch to a mature prototype, for examples, the iterative design evaluation, the BOM of final design, the bending test done on aluminum material to evaluate the strength and how they calculated technical requirement, such as motor torque and battery consumption, to choose appropriate parts for each mechanism.

Also, I actively assisted in vision-sensor-based control system development, which my group-mate took the lead as a software developer. I actively assisted my team in Chapter 6. and Chapter 7. including the research on open-loop and closed-loop system,

the methodology of HSV filtering that the team has ultimately implemented in the robot as well as the program structure and methodology brainstorm. For the reports, I assisted in writing Section 6.2 and Section 7.2.

Other than that, I was in charge of project planning and management (Ex. Gantt chart) as a team leader, and also contributed in ′Result and Discussion′ Chapter 8. Section 8.1, ′Conclusion′ Chapter 9. and ′Further Development′ Chapter 10. Section 10.1 as well as 10.3.

We worked closely together for background research, problem identification and setting the objectives, and thus all members contributed in writing Chapter 1.

Appendix XVIII. Summary of Contribution: Rudra Someshwar

Summary of Contribution

Project: ASME Competition Robot Development and Vision Based Target Alignment

Name of Student: Rudra Someshwar

Names of group-mate(s): Chow Hung Ming Roy, SHIN Jiho

This project involves the following tasks: (a) Background Research; (b) ASME Hardware Research & Design Development; (c) ASME Software Research & Design Development; (d) Iterative Evaluation, Analysis & Modification; (e) Standard Parts and Material Research & Selection; (f) Parts Fabrication, Manufacturing & Assembly; (g) Vision Sensor Integrated Image Processing System Development; (h) Report Writing.

In the first phase of the project before the competition, I contributed mainly in the area of Software Design of the Arduino Uno board and thus wrote Chapter 4 and Chapter 5. I also assisted in Section 3.7 including conducting research and developing the control system circuitry. All team members brainstormed design ideas of the robot together which are discussed in Chapter 3, but another team member took the lead for this part.

For the second phase of the project after the ASME competition, I took the lead in software development of the vision-based feedback control system and explored various algorithms and implemented the ones that best fit our application. I developed the program throughout the second semester and continued to optimize it for achieving the best results. Thus, related to this, I wrote Chapter 6 and Chapter 7.

Other than that, I also contributed in ‘Result and Discussion’ Chapter 8. Section 8.3 and Section 3.4, ‘Conclusion’ Chapter 9. and ‘Further Development’ Chapter 10. Section 10.2.

We worked closely together for background research, problem identification and setting the objectives, and thus all members contributed in writing Chapter 1.

Number of Chapters written does not correspond to the level of contribution. Thus, Although I have written more chapters in the Final Report, all team members have contributed equally.

References

[1] "Problem & Solution | Environmental Protection Department," Epd.gov.hk, 2005. [Online]. Available: https://www.epd.gov.hk/epd/english/environmentinhk/waste/prob_solutions/waste_problems.html. [Accessed 22 April 2019].

[2] G. E. Totten, Handbook of Hydraulic Fluid Technology, Boca Raton: CRC Press, 2011.

[3] "Fork-lift Truck Safe Operation," Occupational Safety and Health Council, 2019. [Online]. Available: http://www.oshc.org.hk/eng/main/hot/FLT/. [Accessed 31 March 2019].

[4] "Forklift Accident Statistics," McCue Corporation, [Online]. Available: https://www.mccue.com/content/forklift-accident-statistics. [Accessed 31 March 2019].

[5] "Load Handling: Load Composition," United States Department of Labor, [Online]. Available: https://www.osha.gov/SLTC/etools/pit/operations/loadcomposition.html. [Accessed 1 April 2019].

[6] "Auto robot maker Kuka to diversify target markets," The Star Online, 27 December 2017. [Online]. Available: https://www.thestar.com.my/business/business-news/2017/12/27/auto-robot-maker-kuka-to-diversify-target-markets/. [Accessed 5 April 2019].

[7] D. Napier, "Automated Assembly Line (Robotic Arms)," [Online]. Available: https://iptmajor.weebly.com/automated-assembly-line-robotic-arms.html. [Accessed 5 April 2019].

[8] "Articulated Dump Trucks," Iron Planet, [Online]. Available: https://www.ironplanet.com.au/jsp/marketing/landing-page.jsp?id=106773&canLinkManual=articulated-dump-truck-buying-guide&iprefoh=www.ironplanet.com. [Accessed 6 April 2019].

[9] School of Innovation, Design and Engineering, "Research project to develop refuse collection robot," Malardalen University Sweden, 16 September 2015. [Online]. Available: https://www.mdh.se/forskningsprojekt-utvecklar-sophamtningsrobot-1.81010?l=en_UK. [Accessed 31 March 2019].

[10] T. English, "Garbage Men are Being Replaced with Robots," Interesting Engineering, 7 March 2016. [Online]. Available: https://interestingengineering.com/garbage-men-are-being-replaced-with-robots. [Accessed 31 March 2019].

[11] "What is LiDAR and how does it work?," 3D Laser Mapping, [Online]. Available: https://www.3dlasermapping.com/what-is-lidar-and-how-does-it-work/. [Accessed 31 March 2019].

[12] A. Wiren, "The ROAR project - robot and drone in collaboration for autonomous refuse handling," 24 February 2016. [Online]. Available: https://www.youtube.com/watch?v=fNIV6Dcj29E. [Accessed 31 March 2019].

[13] H.-S. Kim, "New extruded multi-cell aluminum profile for maximum crash energy absorption and weight efficiency," *Thin-Walled Structures,* vol. 40, no. 4, pp. 311-327, 2002.

[14] T. Hellstrom, “Forward Kinematics,” UMEA University, 28 August 2011. [Online]. Available: http://www8.cs.umu.se/kurser/5DV122/HT13/material/Hellstrom-ForwardKinematics.pdf. [Accessed April 2019].

[15] "Driving Mecanum Wheels Omnidirectional Robots," RoboteQ, 25 October 2015. [Online]. Available: https://www.automation.com/pdf_articles/AN1543-DrivingMecanumWheels.pdf. [Accessed April 2019].

[16] S. Yu, C. Ye, H. Liu and J. Chen, "Development of an omnidirectional Automated Guided Vehicle with MY3 wheels," *Perspectives in Science,* vol. 7, pp. 364-368, 2016.

[17] G. Runge, G. Borchert and A. Raatz, "Design of a holonomic ball drive for mobile robots," in *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Senigallia, Italy, 2014.

[18] F. Tajti, G. Szayer, B. Kovacs and P. Korondi, "Robot base with holonomic drive," *IFAC Proceedings Volumes,* vol. 47, no. 3, pp. 5715-5720, 2014.

[19] R. Rojas and A. G. Forster, "Holonomic Control of a robot with an omnidirectional," *K¨unstliche Intelligenz,* 2006.

[20] N. Tlale and M. de Villiers, "Kinematics and Dynamics Modelling of a Mecanum Wheeled Mobile Platform," in *15th International conference on Mechatronics and Machine Vision in Practice*, Auckland, New Zealand, 2008.

[21] K. L. Han, O. K. Choi, I. Lee, I. Hwang J. S. Lee and S. Choi, "Design and Control of Omni-Directional Mobile Robot," in *International Conference on Control, Automation and Systems*, Seoul, Korea, 2008.

[22] A. Shimada, S. Yajima, P. Viboonchaicheep and K. Samura, "Mecanum-wheel Vehicle Systems Based on Position Corrective Control," *IEEE,* pp. 2077-2082, 2005.

[23] J. Kim, S. Woo, J. Kim, J. Do, S. Kim and S. Bae, "Inertial Navigation System for an Automatic Guided Vehicle with Mecanum Wheels," *INTERNATIONAL JOURNAL OF PRECISION ENGINEERING AND MANUFACTURING,* vol. 13, no. 3, pp. 379-386, 2012.

[24] B. &. K. N. Jong-Jin, "Design optimization of a mecanum wheel to reduce vertical vibrations by the consideration of equivalent stiffness," *Shock and Vibration,* vol. 2016, pp. 1-8, 2016.

[25] R. F. Roboticists, "Pros and cons for different types of drive selection," Robohub, 29 July 2016. [Online]. Available: https://robohub.org/pros-and-cons-for-different-types-of-drive-selection/. [Accessed 7 November 2018].

[26] G. J. Monkman, Robot Grippers, Wiley, 2007.

[27] J. Shintake, Soft Robotic Grippers, Wiley, 2018.

[28] J. R. Amend, "A Positive Pressure Universal Gripper Based on the Jamming of Granular Material," *IEEE Transactions on Robotics,* 2012.

[29] E. J. Brookes, "Engineered Plastics," TECARAN ABS, [Online]. Available: http://www.ejbplastics.com/product/89/TECARAN_ABS. [Accessed 2018 November 23].

[30] INC., HuiDa RC International, "Power HD," [Online]. Available: http://www.chd.hk/UploadFiles/Att/2013090514483372.pdf. [Accessed April 2019].

[31] dronehobby, "AliExpress," TAROT-RC, [Online]. Available: https://www.aliexpress.com/item/XP70HV-70kg-Digital-Servo-180-320-degrees-Large-Torque-Metal-Gear-Digital-Servo-70KG-com-for/32866982717.html. [Accessed April 2019].

[32] electronic, Shenzhen Integrity, "AliExpress," ZJMZYM, [Online]. Available: https://www.aliexpress.com/item/12V-24V-10W-XD-37GB520-miniature-DC-geared-motor-low-speed-high-torque-Can-be-positive/32775798003.html. [Accessed April 2019].

[33] G. Popiel, "Introduction to the Arduino Microcontroller," *QST,* vol. 99, no. 11, pp. 30-32, 2015.

[34] N. Schoenfeld, "PULSE WIDTH MODULATION," *Projection, Lights & Staging News,* vol. 18, no. 10, p. 106, 2017.

[35] A. McRitchie, "NiMH Vs LiPo Batteries - Which is better?," HOBBIESDIRECT, 5 October 2015. [Online]. Available: https://hobbiesdirect.com.au/blog/nimh-vs-lipo-rc-batteries. [Accessed April 2019].

[36] "Differences Between NiMH and LiPo Batteries," [Online]. Available:

https://www.eurorc.com/page/69/differences-between-nimh-and-lipo-batteries. [Accessed April 2019].

[37] A. Leoni, "MICROCONTROLLERS," *EEPN,* vol. 65, no. 6, p. 16, 2006.

[38] "PS2無綫遙控手柄套裝Arduino套件智能小車機器人遙控器送轉接板," World.taobao.com, 2019. [Online]. Available: https://world.taobao.com/item/579693764030.htm?spm=a21wu.10013406-tw.0.0.699a7d24ouxC8Y. [Accessed 10 March 2019].

[39] "Guides/PS2," curious-inventor, 2019. [Online].

[40] J. Walker, "Everything You Have Always Wanted to Know about the Playstation But Were Afraid to Ask," [Online].

[41] M. Dowty, "playstation-2-dual-shock-controller-protocol-notes.rest," GitHub, 2013. [Online]. Available: https://gist.github.com/scanlime/5042071. [Accessed 21 April 2019].

[42] Lynxmotion.com, [Online]. Available: http://www.lynxmotion.com/images/files/ps2cmd01.txt. [Accessed 21 April 2019].

[43] A. Industries, "Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit," Adafruit.com, 2019. [Online]. Available: https://www.adafruit.com/product/1438. [Accessed 21 April 2019].

[44] "Camera Module V2 - Raspberry Pi," Raspberry Pi, 2019. [Online]. Available: https://www.raspberrypi.org/products/camera-module-v2/. [Accessed 21 April 2019].

[45] Matteini, "An overview on automatic design of robot controllers for complex tasks," 2018.

[46] Julian Togelius Renzo De Nardi, Owen E Holland, and Simon M. Lucas, Evolution of neural networks for helicopter control: Why modularity matters, UK: Deaprtment of Computer Science University of Essex, 2006.

[47] "adafruit/Adafruit_Motor_Shield_V2_Library," GitHub, 2019. [Online]. Available: https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library. [Accessed 21 April 2019].

[48] "Adafruit PCA9685 PWM Servo Driver Arduino Library: Adafruit_PWMServo Driver Class Reference," Adafruit.github.io, 2019. [Online]. Available: http://adafruit.github.io/Adafruit-PWM-Servo-Driver-Library/html/class_adafruit___p_w_m_servo_driver.html. [Accessed 21 April 2019].

[49] B. Porter, "PlayStation 2 Controller Arduino Library v1.0 << The Mind of Bill Porter," Billporter.info, 2019. [Online]. Available: http://www.billporter.info/2010/06/05/playstation-2-controller-arduino-library-v1-0/. [Accessed 21 April 2019].

[50] "Library Reference | Adafruit Motor Shield V2 | Adafruit Learning System," Learn.adafruit.com, 2019. [Online]. Available: https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/library-reference#void-setpwm-uint8-t-pin-uint16-t-val-void-setpin-uint8-t-pin-boolean-val-5-3. [Accessed 21 April 2019].

[51] "Arduino Reference," Arduino.cc, 2019. [Online]. Available: https://www.arduino.cc/reference/en/.

[52] N. N. S., Control system engineering, Hoboken, NJ.: Wiley, 2015.

[53] P. B. G., "Characteristics of different sensors used for distance measurement," *International Research Journal of Engineering and Technology (IRJET),* 2017.

[54] Vesna, "Image and its matrix, matrix and its image," in *Belgrade*, 2008.

[55] M. Deswal, "A fast HSV image color and texture detection and image conversion algorithm," in *International Journal of Science and Research (IJSR)*, 2012.

[56] "NumPy - NumPy," Numpy.org, 2019. [Online]. Available: https://www.numpy.org/. [Accessed 21 April 2019].

[57] "Morphological Transformations - OpenCV 3.0.0-dev documentation," Docs.opencv.org, 2019. [Online]. Available: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html. [Accessed 21 April 2019].

[58] "Feature Detection," OpenCV, [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=cornerharris. [Accessed 21 April 2019].

[59] "Can be flexibly tailored to your requirements," SIEMENS, [Online]. Available: https://new.siemens.com/global/en/products/automation/systems/industrial/plc/s7-1200.html. [Accessed 21 April 2019].

[60] Traxxas, [Online]. Available: https://traxxas.com/. [Accessed April 2019].

[61] "Color Detection & Object Tracking," [Online]. Available: https://www.opencv-srf.com/2010/09/object-detection-using-color-seperation.html. [Accessed 23 April 2019].