BeachBots 2020

Members:

Dennis Chavez Romero

Mia DiBattista

Spencer Gregg

Yossef Naim

Cameron Walsh

Beach Swarm: Phase 2

Beachbots Major Qualifying Project

A Major Qualifying Project

submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfilment of the requirements for the

degree of Bachelor of Science

By: Dennis Chavez Romero, Mia DiBattista, Spencer Gregg, Yossef Naim, Cameron Walsh

Date: May 6, 2021

Report Submitted to: Professor Nicholas Bertozzi, Professor Brad Miller and Professor Huang Xinming

Abstract

Beaches are becoming more littered by trash that is discarded by humans worldwide. Currently, there are methods for cleaning trash from beaches that usually entail large, expensive machines that may not be accessible for all types of beaches. This project aims to develop a multi-robot solution for cleaning beaches in an effective manner. Using the engineering design process, our team gathered information about beaches to design prototypes, test designs, and build a functioning proof of concept, multi-robot solution.

Acknowledgements

Our project would not have been completed without the help and generosity of many people. Our team would like to thank Professor Bertozzi and Professor Miller for their continuous guidance, wisdom, and endless support throughout the duration of this project. In addition, we would like to thank Grant Perkins for aiding our team in the training of a custom TensorFlow Lite (TFLite) model with the use of his own intellectual property.

Authorship

All team members contributed to the writing and editing of the report. The distribution of involvement amongst team members is detailed in the table below.

Section	Primary Author(s)	Reviewed By
1 Introduction		
1.1 The Ecosystem and its Importance	Yossef	Dennis/Spencer/Yossef
1.2 The Problem	Yossef	Dennis/Spencer/Yossef
1.3 Our Solution	Spencer	Dennis/Spencer/Yossef
2 Background		
2.1 Causes of Litter in Coastal Areas	Dennis	Dennis/Spencer/Yossef
2.2 Types of Beach Litter	Mia	Dennis/Spencer/Yossef
2.3 Impact on the Ecosystem	Mia	Dennis/Spencer/Yossef
2.4 Current Beach Clean-Up Methods	Cam	Dennis/Spencer/Yossef
3 Methodology		
3.1 Overview of the System	Spencer	Dennis/Spencer/Yossef
3.1.1 Analysis of Previous System	Spencer	Dennis/Spencer/Yossef
3.1.2 Overview of Current System	Spencer	Dennis/Spencer/Yossef
3.2 Smallbot System	Dennis	Dennis/Spencer/Yossef
3.2.1 Mechanical Design	Spencer	Dennis/Spencer/Yossef
3.2.1.1 Drivetrain Selection and Design	Spencer	Dennis/Spencer/Yossef
3.2.1.2 Gripper Mechanism	Spencer/ Cam	Dennis/Spencer/Yossef
3.2.1.3 Two Degree of Freedom Arm	Spencer/Dennis/Cam	Dennis/Spencer/Yossef

3.2.1.4 Trash Storage Design	Spencer	Dennis/Spencer/Yossef
3.2.1.5 Camera Mount	Spencer	Dennis/Spencer/Yossef
3.2.2 Electrical Design	Yossef	Dennis/Spencer/Yossef
3.2.2.2 Microprocessor Selection and Hardware Accelerator	Dennis/Yossef	Dennis/Spencer/Yossef
3.2.2.3 Sensor Selection	Spencer	Dennis/Spencer/Yossef
3.2.2.4 Battery Selection	Spencer	Dennis/Spencer/Yossef
3.2.3 Software Architecture	Dennis	Dennis/Spencer/Yossef
3.2.3.1 Chassis Alignment	Dennis	Dennis/Spencer/Yossef
3.2.3.2 Object Detection	Yossef	Dennis/Spencer/Yossef
3.2.3.3 Communication Between the Basebot and the Smallbot	Spencer	Dennis/Spencer/Yossef
3.3 Basebot System	Dennis	Dennis/Spencer/Yossef
3.3.1 Mechanical Design	Spencer	Dennis/Spencer/Yossef
3.3.2 Microprocessor Selection	Spencer	Dennis/Spencer/Yossef
3.3.3 Software Architecture	Dennis	Dennis/Spencer/Yossef
3.3.3.1 State Machine Path Planning	Dennis	Dennis/Spencer/Yossef
3.3.3.2 Localization with AprilTags	Spencer	Dennis/Spencer/Yossef
4 Results		
4 Results (Introduction)	Spencer	Dennis/Spencer/Yossef
4.1 Prototyping	Spencer	Dennis/Spencer/Yossef
4.1.1 Smallbot Prototyping	Mia	Dennis/Spencer/Yossef
4.1.1.1 Mechanical Prototyping	Spencer	Dennis/Spencer/Yossef
4.1.1.2 Electrical Prototyping	Yossef/Dennis	Dennis/Spencer/Yossef
4.1.1.3 Software Prototyping	Dennis	Dennis/Spencer/Yossef

4.1.1.4 Object Detection Prototyping	Yossef	Dennis/Spencer/Yossef
4.2 Testing	Spencer	Dennis/Spencer/Yossef
4.2.1 Smallbot Testing	Mia	Dennis/Spencer/Yossef
4.2.1.1 Mechanical Testing	Spencer/ Dennis/Cam	Dennis/Spencer/Yossef
4.2.1.2 Electrical Testing	Yossef/ Dennis	Dennis/Spencer/Yossef
4.2.1.3 Software Testing	Dennis/ Spencer/Yossef	Dennis/Spencer/Yossef
4.2.2 Basebot Testing	Spencer	Dennis/Spencer/Yossef
4.2.2.3 Software Testing	Dennis	Dennis/Spencer/Yossef
4.2.3 Integration Testing	Dennis	Dennis/Spencer/Yossef
5 Conclusion and Recommendations		
5.1 Reflection of Our Project	Mia	Dennis/Spencer/Yossef
5.1.1 Mechanical Reflection	Spencer	Dennis/Spencer/Yossef
5.1.2 Electrical Reflection	Yossef	Dennis/Spencer/Yossef
5.1.3 Software Reflection	Dennis	Dennis/Spencer/Yossef
5.2 Final Conclusion	Spencer	Dennis/Spencer/Yossef
Addendum	Mia	
Bibliography	Everyone	Dennis/Spencer/Yossef
Appendix	Spencer/Dennis/Mia	

Table of Contents

Table of Figures

Table of Tables

Section 1: Introduction

1.1 The Ecosystem and Its Importance

The natural world is composed of many ecosystems: biological environments, containing communities of interacting organisms that communicate with each other as well as with their physical environment. Ecosystems range from the Amazon Rain Forrest to the Himalayan Mountains and beyond. Many of these ecosystems are close to human civilization and are even common ground between the organisms that naturally live within and their human counterparts (National Geographic Society, 2012). A prime example of these environments are beaches. Beaches provide a range of both animal and plant life that contribute to the wellness of the area itself as well as its surroundings (US Army Corps of Engineers, n.d.).

Ecosystems play a major role in the health and stability of life on Earth. Specifically, one ecosystem has enough influence to affect its surrounding areas, as the changes experienced within a given ecosystem may influence its neighboring counterparts. Ecosystems do this by taking the resources available in a region and producing habitable areas for organisms to survive (National Geographic Society, 2012). For example, areas around beaches are littered with immobile mussels —a similar species to the clam—, which rely on changing tides to flow microscopic sea creatures through them for feeding purposes. These mussels in turn depollute the surrounding water and make the environment safer for other life forms (Wikimedia Foundation, 2021).

Therefore, ecosystems are built on the following concept: every component of an ecosystem is tasked with contributing to the health and stability of the system and its inhabitants.

Healthy ecosystems further benefit other ecosystems and ensure they remain in good condition. Ultimately, the organisms present around the globe are components to the grander ecosystem that is planet Earth.

1.2 The Problem

As previously mentioned, ecosystems have methods of impacting the surrounding areas. However, human beings are increasingly intruding into the natural flow of ecosystems. The action of being intrusive does not necessarily mean that humans have a decided negative impact on habitats. Scientists are constantly exploring ecosystems and aiding in the longevity of many different natural environments to produce a healthier planet. However, humans also have an extensive history of disrupting ecosystems and aiding in their ultimate demise (US Army Corps of Engineers, n.d.). It is important that steps are taken to help reduce the amount of harm that people are causing to different natural environments. Our team will be taking a step to making a cleaner planet.

The beaches surrounding heavily populated areas are constantly visited by tourists and locals year-round. These locations serve mostly as vacation getaways. However, it is important to note that all saltwater beaches serve as fully functional ecosystems; supporting wildlife and plant life, which aids nature in maintaining balance. One of the largest problems facing beaches is the litter that tourists and locals leave behind, without any regard for how their actions may affect the ecosystem. For example, the mussels previously discussed cannot differentiate between microplastics and microorganisms that flow into their shells; one of which feeds and

supports them while the other kills them. While mussel beds are constantly being destroyed due to the decomposition of the surrounding litter, water quality is declining, which heavily impacts the surrounding wildlife and ecosystems.

Microplastics, or micro-litter, come from manmade litter that has been decomposed in the saltwater. The litter gets washed into the ocean because of rising and falling tides. While there are countless examples of what beach litter does to the oceanic ecosystem, it is important to realize that this problem must be dealt with as soon as possible. Litter needs to be removed from beaches in order so sustain a healthy, functioning ecosystem (US Army Corps of Engineers, n.d.).

1.3 Our Solution

There are many ways that harmful litter can be cleared from the world's beaches. This project emphasizes a robotic approach to the solution, where an autonomous system is used to efficiently clear a beach with minimal human interaction. The robotic system designed in this project consists of two robots in an autonomous swarm-like approach: a Basebot and a Smallbot. The Basebot acts as the parent robot in the system and one or more Smallbots act as the children. The Basebot sits higher up on the shore and it includes a camera that overlooks and maps out a portion of beach. The camera on the Basebot tracks the Smallbots maneuvering within the mapped area to designate sections of said area to specific Smallbots.

For this project only one Smallbot will be present in the system. The Smallbot features a four-wheel, rocker-bogie drivetrain with grip wheels to move across the sand easily without

getting stuck. The Smallbot will drive within its designated area defined by the Basebot until its onboard camera recognizes a piece of litter based on a machine learning model. After detecting a trash object, a two degree of freedom (DOF) arm will move to the position of the object, grab the object with its gripper mechanism, and place the object in the onboard bucket. Once the Smallbot completes clearing the area, the Smallbot will return to the Basebot and dump the trash.

This approach to rid beaches of litter is efficient and practical for most coastal areas. Beaches are not always occupied all days of the year and therefore, can be cleared of any leftover litter that may have been left behind by visitors. An autonomous robotic system is efficient for clearing unoccupied beaches, as it will eliminate human error that could develop from volunteer projects that may not clear all parts of a beach. A robotic system can also run for extended periods of time without breaks and will therefore be able to clear a larger area than a group of humans.

Our team is achieving the solution to the problem by splitting the workload between four terms at Worcester Polytechnic Institute (WPI). The first term consisted of reviewing the progress made during the previous iteration of the project and determining areas that could be improved. In addition, the first term also consisted of forming preliminary designs and performing tests on mechanisms that could be improved. The second term involved making improvements to the prototypes from the first term, ordering parts/materials/electronics, and implementing a machine learning model for litter detection along with the necessary control systems. The third term included testing the final design and making final improvements to the dual-robot system. Finally, the fourth term consisted of further testing and documenting the progress made during this iteration of the project.

The objectives and goals of our team's iteration of the project were the following:

- The Smallbot needs to be able to maneuver on sand.
    - The robot should drive on sand at a speed of $0.5\,\frac{m}{s}$.
    - The robot's drivetrain must not get jammed or stuck from driving on sand.
    - The robot's battery life must last an hour at least.
- Improve upon the Smallbot's mechanical features.
    - The bottom half of the Smallbot must be water resistant.
    - The gripper should collect trash such as 473ml bottles and 355ml cans.
    - The storage bucket on the Smallbot needs to store at least one soda can (355ml 5.38cm diameter, 12.07cm tall) and one plastic bottle (473ml 6.1cm diameter x 19.69cm tall).
- Soda cans and plastic bottles must be detected as trash by a machine learning model. The Smallbot must be equipped with a variety of sensing components to ensure proper maneuvering and localization throughout the cleaning routine: an Inertial Measuring Unit (IMU), AprilTags and a camera.

Reach Goals:

- Implement communication between the Smallbot and the Basebot.
- When the Smallbot is full, it must return to its starting position.
- Make the Smallbot's onboard electronics waterproof.

Section 2: Background

2.1 Causes of Litter in Coastal Areas

Beaches are commonly used as locations for social gatherings and events, which has caused anthropogenic litter to become a byproduct of coastal development (Araújo, et al., 2018). Anthropogenic litter is defined as the waste that results from the influence of human beings on any given natural environment (Merriam-Webster, n.d.). The cause of this type of litter on beaches can be linked to a variety of factors, like "location and morphology of the beach, the presence of rivers and streams, and winds" (Araújo, et al., 2018). However, a large contributor to this issue is the commercial activity that has developed in urban beaches, where sales of food and beverages often take place (Araújo, et al., 2018). Therefore, an elevated number of social events and activities is linked to an increase in the anthropogenic litter found in densely populated coastal areas.

Marine debris is another major cause of anthropogenic litter in coastal areas. The types of marine debris that are commonly found range from five-millimeter microplastics, to fishing gear and abandoned vessels (Ocean Pollution, n.d.). This specific type of litter poses a major threat to marine animals and their ecosystems, as these creatures can get entangled in such waste, and their natural habitats become polluted (Ocean Pollution, n.d.). Despite marine debris originating on land, poor waste management practices, weather conditions and extreme natural events are gateways for this land-based litter to enter to ocean (Ocean Pollution, n.d.). Nonetheless, this debris is a major concern, as ocean currents cannot only create "garbage patches" — "large areas of the ocean where trash, fishing gear, and other marine debris collects"— but they can also

cause for some of the debris to sink into the depths of the ocean or be washed ashore with the waves and tides (Ocean Pollution, n.d.).

Ocean dumping is defined as the activity of depositing waste materials directly into the ocean (Mambra, 2020). This practice is commonly performed by factories and industries who dispose of their "tankers and ships and sewerage waste materials into the oceans and seas" (Mambra, 2020). While most of these waste products do not come in the form of solid, physical litter, a small percentage of this waste presents itself as medical, hazardous, or toxic materials, which once washed up on beaches, can cause accidents and injuries as a result of the sheer exposure to these substances (Bortman, 2020). Therefore, the closure of beaches is not unknow to the shorelines where this litter lands, as the pollution then needs to be removed, in order to ensure the well-being of beachgoers and animals in the area (Bortman, 2020).

2.2 Types of Beach Litter

It has been estimated that over 300 million tons of plastic is manufactured each year, half of which is made into single use products like plastic bags and cosmetic scrubs (Lebreton, et al, 2017). More than 8 million tons of this plastic ends up in the ocean where it seemingly "goes away" (Lebreton, et al., 2017). However, discarded plastic and other waste such as plastic bags, toothbrushes, plastic packaging, straws, and plastic bottles all break down when dispersed in salt water (Barry, 2009). This results in the plastic pollution spreading throughout the sea, being ingested by sea life, and particles of it ending up on our beloved beaches (Parker, 2019).

In addition to plastic, many other items are found along the coastlines and within the marine environment. Some other common forms of litter are cigarette butts, disposable face masks, bottle caps, food wrappers, and foam takeout containers and cups (Chow, 2019 & Reddy, 2018). In 2015, “it was recorded that there was 5.25 trillion pounds” of litter within the marine ecosystem and the beaches that surround it, and it has only exponentially grown from then (Weinhardt, 2019). In addition, according to the Ocean Conservancy, “8 million metric tons” of trash is dumped into our oceans each year (Leonard, George, et al. 2020). From the oceans, the litter ends up washing ashore, polluting beaches, and creating a danger for seaside life, as well as beachgoers (Environmental Protection Agency, 2020).

2.3 Impact on the Ecosystem

Marine litter has a very serious and severe impact on marine life in today’s waters. All around the globe, marine pollution is found both in bodies of water, as well as inside the species that inhabit said environments. Plastics and other litter can be consumed by marine animals, resulting in debris filling their stomachs in place of food. Plastics can easily be broken down by the seawater, creating microplastics. These microplastics can also be found in the bloodstream and tissue of marine animals (Sharma, et al., 2017). This is how the plastic debris then enters the bloodstream and inner tissues of marine organisms.

Though the impact of ocean pollution on marine life alone is extensive, it affects more than just the ocean and its ecosystem. Fish and other organisms that live in the world’s bodies of water are consumed by both land animals and humans. Some of these land animals will also be

eaten by humans, which therefore completes the food web (Stromberg, 2013). Due to the food web, both land animals and humans can be endangered by the impacts that litter in the sea have on marine environments (Sharma, et al., 2017). Through consumption of marine life, ocean litter also becomes harmful to humans. Human ingestion of these pollutants, as well as their resulting chemicals, causes serious health problems such as "alteration in chromosomes which lead to infertility, obesity, and cancer" (Sharma, et al., 2017).

In addition to the tragic effects of consuming marine litter, for both ocean life and humans, there are negative impacts of this pollution on land. One of the most prominent issues is the inconvenience for beachgoers (Environmental Protection Agency, 2019). People who frequently go to the beach are more likely to encounter large amounts of pollution along the shoreline (Environmental Protection Agency, 2020). Not only can beach litter be bothersome, but it can also be harmful. Oftentimes, both people and wildlife can get caught in it. For humans, this is a minor inconvenience and a possible trip to the doctor. Meanwhile, for wildlife within the ecosystem, it could mean the end of their lives or lifelong injuries (Werner, Stefanie, et al, 2016). According to National Geographic, a common example of this is when seabirds accidentally consume colorful plastics and other litter because "it smells like food" and eventually pass away due to the chemicals that enter their bodies (Parker, 2016). Since there are large amounts of litter and pollution within the marine ecosystem, it is necessary to create a plan of action to do our best by removing these harmful substances that we, as humans, put there.

2.4 Current Beach Clean-Up Methods

A variety of approaches have been developed to help negate the impact of pollution on the environment, specifically on beaches. Common beach cleaning methods revolve around volunteerism. Volunteers are able to clean beaches by hand and with common tools such as rakes, sand sifters, and shovels. However, there are companies that assist with the beach cleaning process. One example is a Greek environmental organization named iSea. iSea strives to innovate on current beach-cleaning methods. This specific company runs volunteer-based beach clean ups and educational programs, while also working with the community through projects, such as ‘Fishing for Litter’ (iSea, 2020). Other companies such as 4Ocean, only conduct beach cleaning operations, and are supported by selling merchandise. 4Ocean utilizes boats, fishing nets, and mobile ocean skimmers to expand their efforts to produce a clean environment (4Ocean, 2020). Ultimately, despite there being beach-cleaning robots that currently exist, like the Dronyx Solarino, they are too large and expensive to be used in most places (Dronyx, 2016).

Section 3: Methodology

3.1 Overview of System

The following two sections give a brief overview of the accomplishments of the previous iteration of the project and the current design of the system.

3.1.1 Analysis of Previous System

The previous implementation of the project was a proof of concept that a swarm system could be used for cleaning trash from a beach. The previous system was able to identify cans on a beach using a custom TFLite model and a Google Coral Edge Tensor Processing Unit (TPU). However, because the TFLite model was not quantized to run on the Coral, the inference could only be performed at a maximum of 2 frames per second (FPS) on the ELP camera. The Basebot could communicate and manage multiple Smallbots in simulation software (Gazebo) with Robot Operating System (ROS) and was also able to perform terrain mapping. However, the Basebot was never tested in practice. Regardless, beach areas were able to be divided into workable zones and assigned to the simulated Smallbots for cleanup.

A physical Smallbot was designed and tested in sand, though it proved to be unreliable. This is due to the treads clogging up with sand after a few minutes of runtime. Along with the drivetrain, a collection mechanism that was mounted on the Smallbot's drivetrain could lift empty cans. The Smallbot was successful in collecting trash from the ground and dumping the trash from the storage bucket. Overall, this past iteration of the project created a foundation for future teams to build upon.

3.1.2 Overview of Current System

To accomplish the requirements of the project and to efficiently clean trash from beaches, the idea of one Basebot and multiple Smallbots was maintained from the previous team. The single Basebot would act as the parent robot over the Smallbots (children) which maneuver about a specified area in the Basebot's camera view. To clear large portions of a beach, the Basebot would eventually move parallel to shore to allow to allow for more areas to be cleaned.

For the scope of this project the Basebot will be stationary and one Smallbot will maneuver in the Basebot's camera view. This swarm design is both efficient and cost effective versus a single robot that would take a longer period time to clean an entire beach. The current approach of one stationary Basebot and a single Smallbot can be easily scaled for future iterations of the project.

3.2 Smallbot System

As previously mentioned, the conceptual swarm would be comprised of multiple Smallbots working together under the direction of a single Basebot. The mechanical design of the Smallbot and the selection of sensors was determined based on the list of tasks that it needed to achieve as an independent system. These tasks include driving and turning on sand reliably, as well as being able to detect and grip litter in a consistent manner. Therefore, the mechanical, electrical. and software aspects of the robot had to undergo careful planning and consideration to achieve the designated goals.

3.2.1 Mechanical Design

The Smallbot was observed and tested based on the previous MQP's implementation. Once completed, it was apparent that there were a variety of mechanical issues that needed to be improved to meet the requirements for this iteration of the project. Our team was told that the previous iteration of the Smallbot was unable to successfully drive on sand for longer than two minutes without stalling. The sand clogged the treads causing the drive speed to decrease because there was not enough torque in the drive motors. Correcting this problem was a primary mechanical requirement for the Smallbot. Secondly, the gripper on the end of the 2-DOF arm was only able to pick up cans that were standing up vertically. This was an issue because litter can be positioned in multiple orientations. Furthermore, the onboard bucket could only hold up to two, 355ml cans. To meet the requirement of holding a maximum of three, 473ml bottles, the bucket needed to be enlarged and redesigned to fit within the new chassis design. Taking these considerations into account, and to meet the requirements that our team set for this project, a redesign of the chassis and the majority of the mechanisms was deemed necessary.

3.2.1.1 Drivetrain Selection and Design

After testing the previous iteration of the Smallbot's chassis, a redesign was necessary to meet the requirements that our team designated for the chassis. The previous chassis could only reach a top speed of $0.1\frac{m}{s}$, which was lower than the top speed requirement our team had made. The new speed requirement is approximately $0.5\frac{m}{s}$, or around walking speed. Furthermore, the

chassis from the previous team did not meet the requirement of driving on sand effectively as the tread drivetrain design would get clogged with sand after a short period of time.

To meet the requirements that our team had set for our project, initial design concepts were made for various drivetrain implementations and the correct drive motor speed/torque for the application. Calculations were conducted to determine the necessary drive motor torque for a four-wheel drivetrain driving or turning on sand. To calculate the necessary drive motor torque, a free body diagram (FBD) was drawn to designate the forces and torques involved for the four-wheel drivetrain. In this estimate calculation, it was measured that the robot weight was 11.3kg (W_r), the width and length of the base was 0.42m (T_R, B), the wheel radius was 0.076m (R_w), and a coefficient of friction of 0.35 (μ) was referenced from the "Table of Ultimate Friction for Dissimilar Materials" (Fine Software, n.d.). Figure 1 and the associated equations show the calculation for the motor torque for one of the four wheels of the drivetrain. The wheels are pictured in red, and the chassis frame is in blue. From these initial calculations we determined that around 1Nm of torque was required for each side of the robot's drivetrain to maneuver on sand.

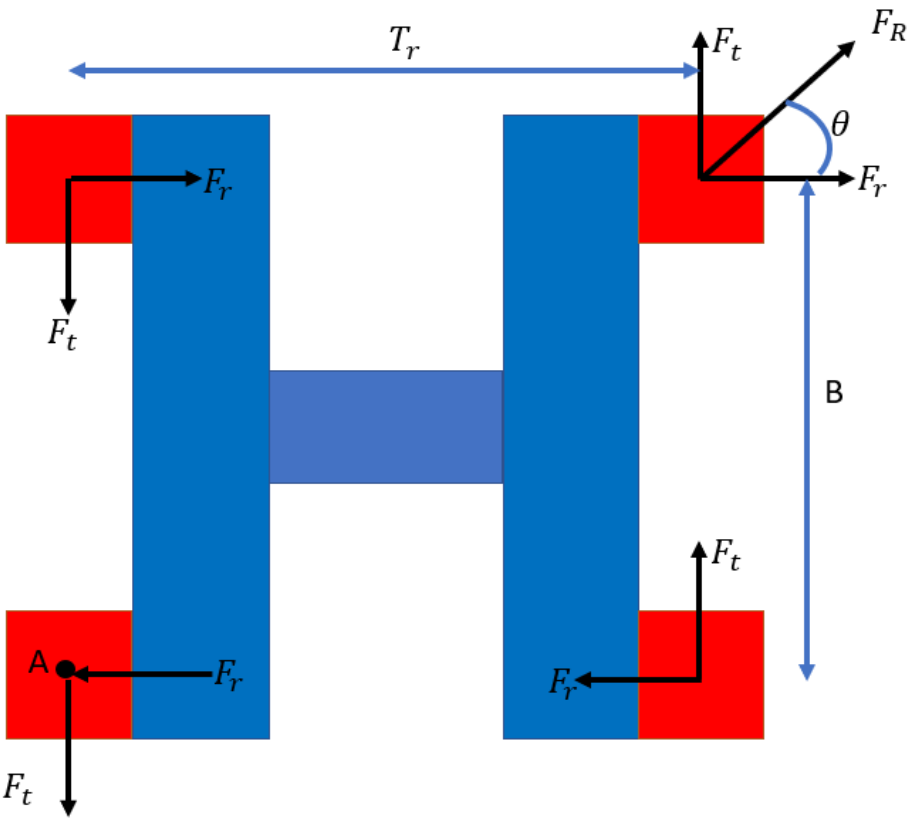


Figure 1: FBD Top View of Drive Train Chassis

$$\sum M_A = \sum Moments_A = 0 = \ 2 * (-F_r(B)) + 2 * (F_t(T_r))$$

$$F_R = \left(\frac{W_r * g}{4}\right)\mu = \left(\frac{11.3kg * 9.8\frac{m}{s^2}}{4}\right)0.35 = 9.73N$$

$$\theta = tan^{-1}\left(\frac{B}{T_r}\right) = tan^{-1}\left(\frac{0.42m}{0.42m}\right) = 45°$$

$$F_t = F_R \sin(\theta) = 9.73N * \sin(45) = 6.88N$$

$$F_r = F_t\left(\frac{B}{T_r}\right) = 6.88N\left(\frac{0.42m}{0.42m}\right) = 6.88N$$

$$\tau_w = Torque_{wheel} = F_t(R_w) = 6.88N * 0.076m = 0.524Nm$$

$$Torque_{one\ side} = \tau_w * 2 = 2 * 0.524Nm = 1.05Nm$$

From initial research into sand vehicles, our group considered two designs as potential options for the Smallbot's drivetrain: a belt-tread design or a four-wheel rocker-bogie drivetrain. The belt drive design idea was similar to the previous iteration of the project, except the treads would be a single belt to help prevent sand from clogging them. As mentioned, the previous iteration of the Smallbot's chassis used tread links that would clog from sand getting trapped between each link. Some advantages of a single belt design are that it would prevent the robot from sinking into the sand and provide the drivetrain with sufficient traction. However, the single belt system was more difficult to design and required additional testing for the correct spacing of the tread pulleys.

The four-wheel, rocker-bogie drivetrain option was a simple design that worked well for other sand vehicles, like the NASA Mars rover (NASA, 2019). This design was simple, but required testing, as sinking into the sand while turning was an inherit concern of the wheel design. Fortunately, our group received some of the components of a four-wheel, rocker-bogie kit from one of our advisors, Professor Miller. The components that our group received from the GearsEd Surface Mobility Platform kit included: four DC motors, four 0.152m diameter wheels, and the anodized aluminum box extrusions for connecting the motors together for each side of the drivetrain. The four DC motors that were included in the components had a stall torque of

18Nm, which was greater than the 0.524Nm that was calculated from our initial estimates. At 0.524Nm the drive motors would operate at an efficiency of around 30% which is ideal for brushed DC motors. The specifications of the drive motors were plotted and evaluated to determine the optimal speed and torque, as seen below in Figure 2.

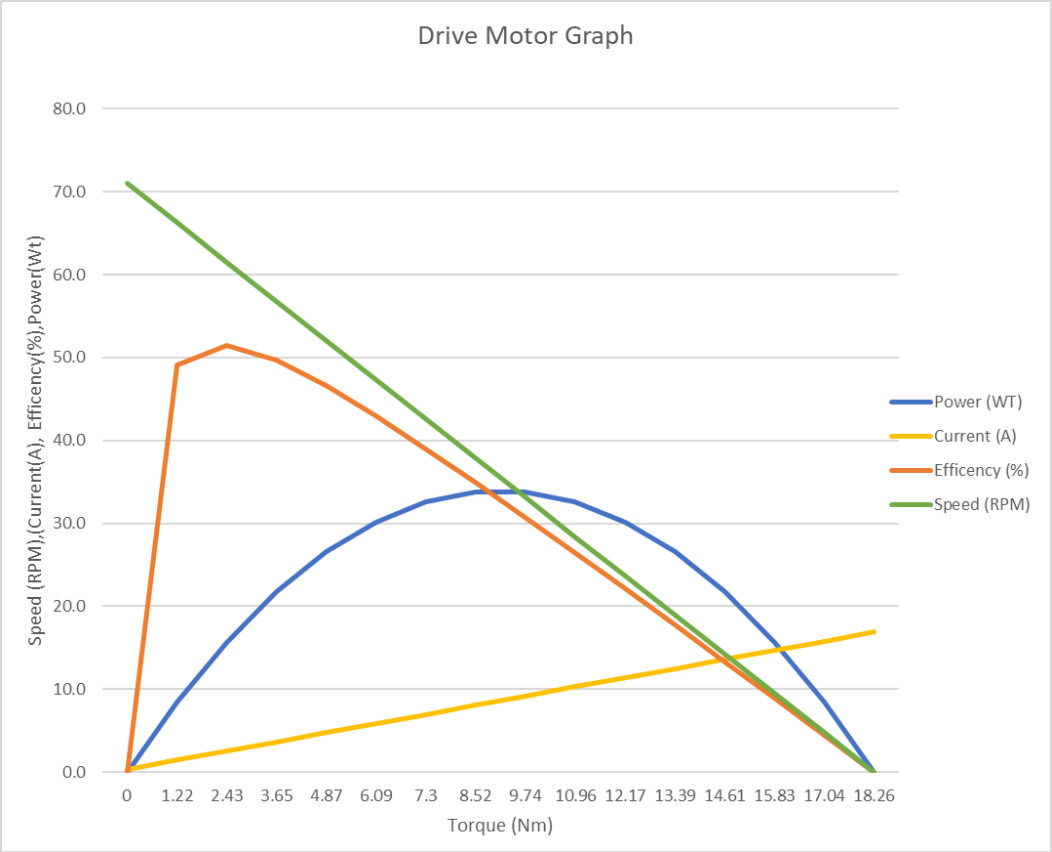


Figure 2: Drive Motor Specifications Graph at 12V

A testing prototype was created to test a functioning rocker bogie system. Using the components that were given to our group and by machining/building the parts necessary to connect both sides of the chassis together, a prototype of the drivetrain was built. The wheels, as seen in Figure 3, are 0.152m in diameter and have deep grooves in them, which help grip the

sand surface. In addition, these grooves do not allow for sand buildup, which made them the most efficient and cost-friendly option for driving on sand for a long duration of time. The 0.152m diameter wheels also helps prevent the robot from sinking into the sand while turning, since the larger surface area helps distribute contact with the ground.

Figure 3: Picture of Drive Wheel

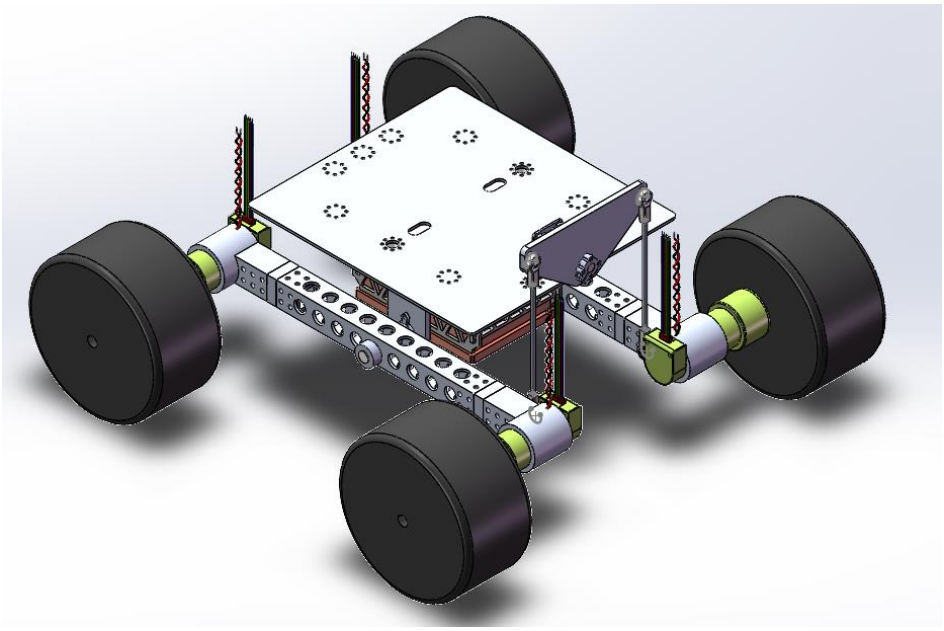


Figure 4: Isometric Back View of Chassis CAD

In addition to the four-wheel chassis, a rocker-bogie suspension system was implemented. The suspension system was crucial since there can be variances in the beach terrain, such as bumps, holes, or other debris. The main idea of implementing a rocker-bogie system was to maintain constant contact with the surface of the beach. The rocker bogie system is able to maintain contact with the surface by having a central shaft which allows each side of the chassis to rotate about its axis, as seen in Figure 4. Both sides of the chassis are connected by a 3-part linkage system which keeps the top plate parallel with the ground and prevents it from moving with the changing terrain. The C-channel attached to the top plate holds pillow blocks that allow a shaft that connects to the suspension bracket to rotate. The suspension system shown above attaches to this rod, so it can rotate freely as the wheelbase moves on terrain, while also keeping the top plate parallel to the ground.

Another requirement our team wanted to fulfill was to have the drivetrain be partially water resistant. To protect the electronics from water, the electronics were placed on top of the top plate as pictured in Figure 4. The top plate was measured to be approximately 0.203m off the ground, which is an improvement from the previous iteration of the robot which had the electronics mounted on the bottom of the Smallbot's chassis. Also, to help prevent sand or water from disrupting the drive motors, motor covers were mounted around the outside of each motor. The drive motor covers can be seen in the Figure 5.

Figure 5: Drive Motor Covers

3.2.1.2 Gripper Mechanism

To collect trash from the surface of the beach, it was decided that a claw-like gripper design was the best mechanism for the task. The previous iteration of the gripper had issues with stability, as there was too much backlash between the gears and the bolts that the gears rotated about. This first iteration of the design was reworked to maintain stability between the gears because it was important to retain a consistent position for grabbing and releasing a bottle or can. This was partially due to the 3mm thick gears which caused the bolts to loosen after use making

the gears skip. Our team attempted to correct the previous design by replacing the gears and fingers. However, the backlash of the gears was still unstable for consistent collection of the bottles and cans.

Since the previous design had mechanical issues that could not be fixed for consistent collection, a new design was created to meet the requirements of reliably collecting a bottle or a can. A two-gear, two-finger design was decided as the appropriate mechanism for the task; an outer plate was mounted onto the opposite side of where each gear with a finger attached to the inner plate and the servo horn. This support on both sides of the gripper helps with maintaining a constant center distance between the two gears. The gear and fingers were designed and 3D printed with 12.7mm thick gears to allow for a greater contact area between the teeth. The diametral pitch of the gears was lowered to 12 to help strengthen the teeth since they were 3D printed. The gripper was mounted parallel to the ground since bottles are more likely to be orientated parallel on a beach, as seen in Figure 8. This new gripper design can collect normal or crushed, 355ml cans and 473ml bottles, which meets and exceeds the requirement for this mechanism. The gripper uses a 30kg-cm (2.9Nm stall torque) servo that was directly mounted to one of the gears and finger parts. At 20% stall torque the gripper can exert a gripping force of about 4.57N. This force is exerted at the center of the finger as shown in Figure 6 and Figure 7, as well as in the equation of equilibrium below.

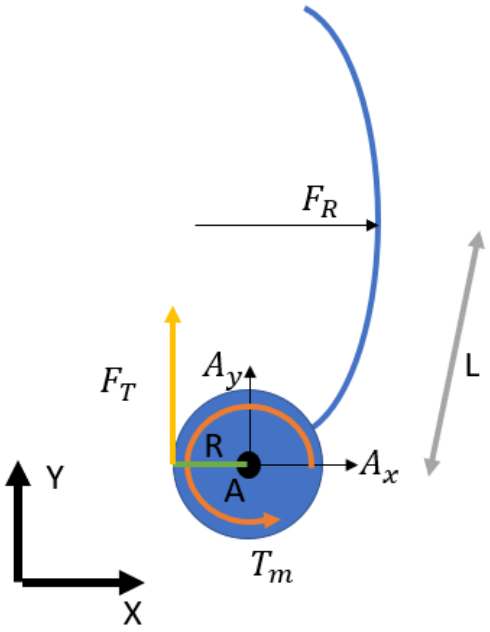


Figure 6: Right Gripper Finger FBD

Right Finger/Gear Parameters:

$$L = Distance\ to\ the\ middle\ of\ gripper\ finger = 0.0635m$$

$$F_R = Resistive\ Force\ at\ the\ middle\ of\ gripper$$

$$F_T = Force\ from\ other\ finger$$

$$R = Radius\ of\ Gear$$

$$A = Center\ of\ Servo\ Horn$$

$$A_X = Force\ on\ A\ in\ the\ X\ direction$$

$$A_Y = Force\ on\ A\ in\ the\ Y\ direction$$

Equations of Equilibrium for the Right Finger/Gear:

$$\sum M_A = 0 = T_m - F_T(R) - F_R(L)$$

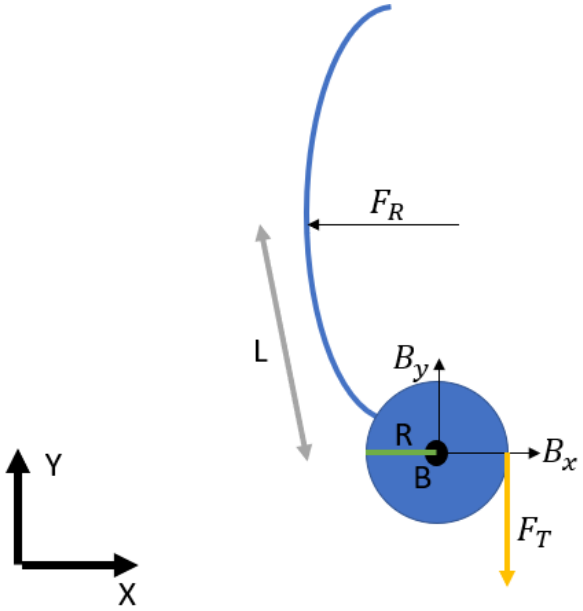


Figure 7: Left Gripper Finger FBD

Left Finger/Gear Parameters:

$$L = Distance\ to\ the\ middle\ of\ gripper\ finger = 0.0635m$$

$$F_R = Resistive\ Force\ at\ the\ middle\ of\ gripper$$

$$F_T = Force\ from\ other\ finger$$

$$R = Radius\ of\ Gear$$

$$B = Center\ of\ second\ gear$$

$$B_X = Force\ on\ B\ in\ the\ X\ direction$$

$$B_Y = Force\ on\ B\ in\ the\ Y\ direction$$

Equations of Equilibrium for the Left Finger/Gear:

$$\sum M_B = 0 = F_T(R) - F_R(L)$$

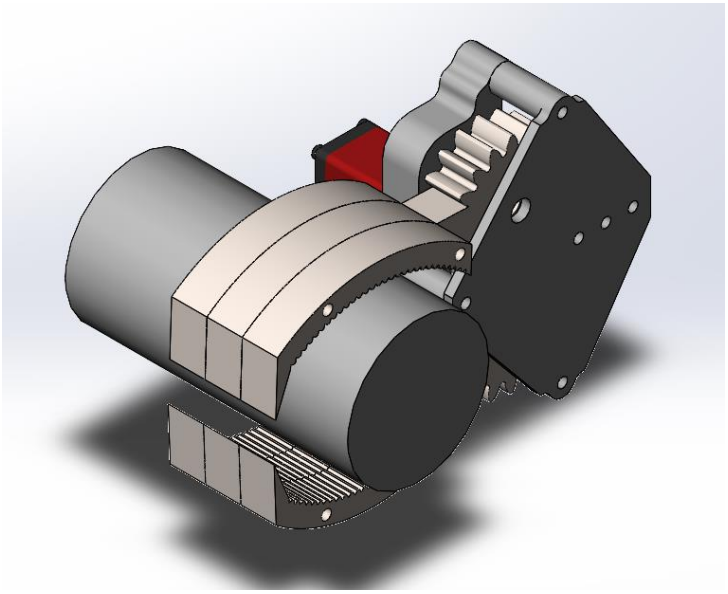


Figure 8: Isometric View of Gripper CAD

Figure 9: Picture of final gripper on the Smallbot

Another problem that had been noted on previous iterations of the gripper was that the motor stalled too often, which eventually lead to the original gripper servo burning out and having to be replaced. Additionally, the motor would struggle when returning the gears to their zeroed position while the gripper was fully closed and when the fingers of the gripper would be wrapped around a piece of litter. The motor would never be able to reach this position due to the litter restraining the gripper's range of motion. To alleviate this, a multiple-part solution was designed. Firstly, a gear-pawl mechanism attached to a 3.7g micro-servo motor was designed, in which a spring-loaded pawl would automatically lock the gears in place and prevent further rotation once a piece of litter was gripped. When the gripper moved again, the micro-servo would rotate the pawl away from the gears to allow them to move. This design is shown below in

Figure 10. However, this pawl design would require that the gripper servo be used to assist the locking tooth from clearing the gears and was scrapped in favor of a design that would allow the locking tooth to begin meshing with the gears by moving in a straight line.

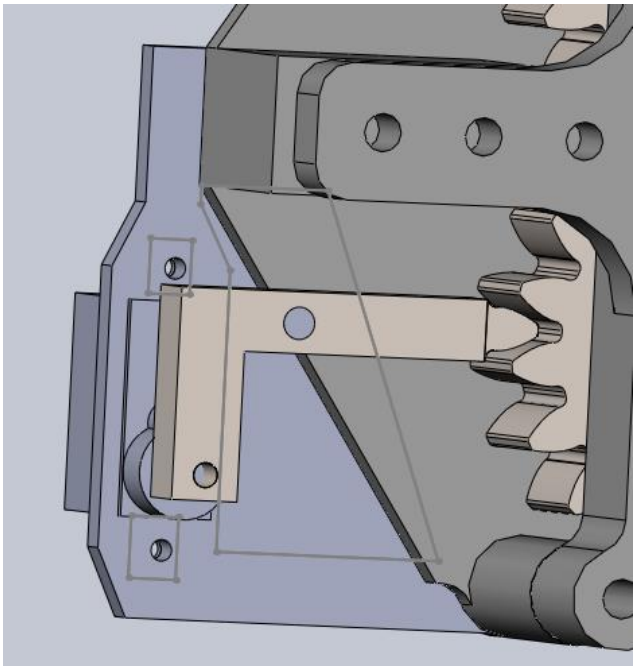


Figure 10: Initial Gear Locking Mechanism Design

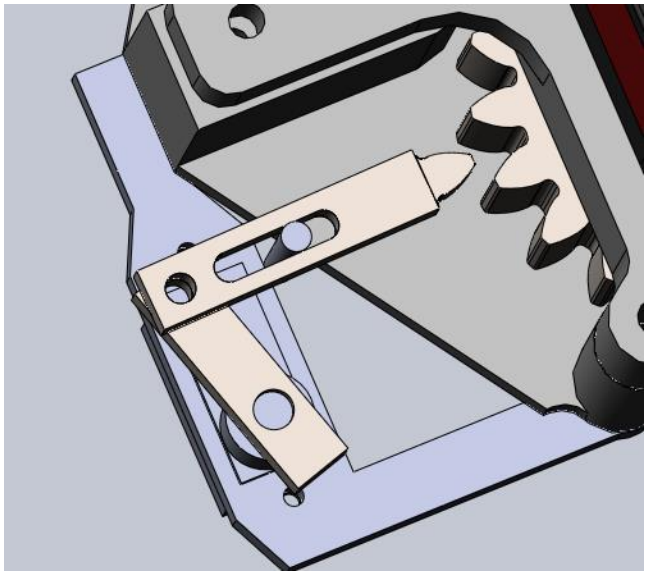


Figure 11: Second Gear Locking Mechanism Design

This was accomplished by designing a two-bar mechanism that used a pivot joint as well as a pin-slot joint, as shown above in Figure 11. The pivot link would be attached to the micro-servo and would push the link attached to the pin slot into the gears in order to lock them. This design was better than the first one, but still had some flaws. Mainly, the force that the gears would place on the pawl, and on the micro server horn was a concern. From further discussions and research, our team found a much simpler solution to the problem, which involved rubber bands. The main idea was that the rubber bands would increase the resistance acting on the gears as the gripper opened and they would then allow power to be cut to the servo when the gripper was closing. This would occur because the bands themselves could provide the necessary gripping force. Therefore, 3D-printed standoffs were added to the gripper fingers, between the grippers themselves and the gears. Rubber bands were then attached to these standoffs to provide ample gripping force to hold the trash without the servo having to be active, as shown below in Figure 12. The rubber bands chosen for this application were #64, VEX robotics rubber bands.

New and revised FBDs and calculations were completed to find the force that was necessary for these rubber bands to apply to the grippers in order to achieve a similar gripping force. The FBDs are shown below in Figures 13 and 14.

Figure 12: Rubber Bands Attached to the Gripper

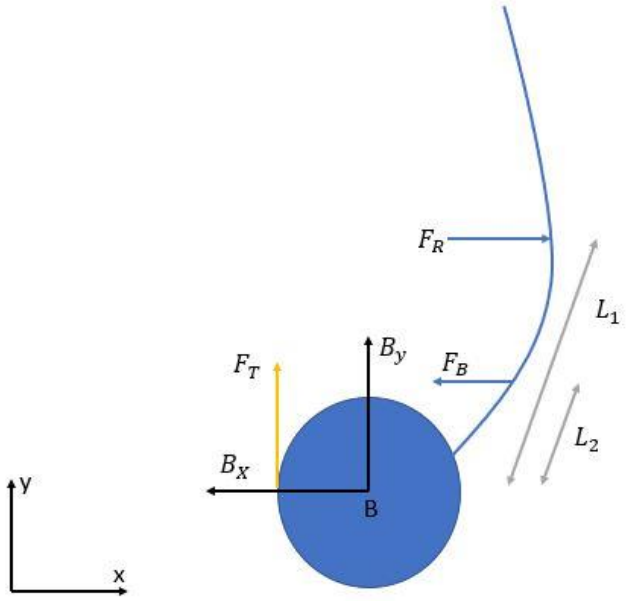


Figure 13: Right Finger FBD with Rubber Band

Right Finger/Gear Parameters:

$$F_R = \ Previously\ Calculated\ Gripping\ Force = 4.57\ N$$

$$F_B = Force\ Applied\ by\ Rubber\ Band$$

$$L_1 = Distance\ to\ the\ middle\ of\ \ gripper\ finger = 0.0635m$$

$$L_2 = Distance\ to\ Rubber\ Band\ Standoffs = 0.0216m$$

$$\mathrm{A} = \ Center\ of\ Right\ Finger\ Gear$$

$$\mathrm{A_X} = \ Force\ on\ A\ in\ the\ X\ Direction$$

$$\mathrm{A_Y} = \ Force\ on\ A\ in\ the\ Y\ Direction$$

$$F_T = \ Force\ from\ other\ finger$$

Equations of Equilibrium for Right Finger/Gear:

$$\sum M_A = 0 = F_B(L_2) - F_R(L_1)$$

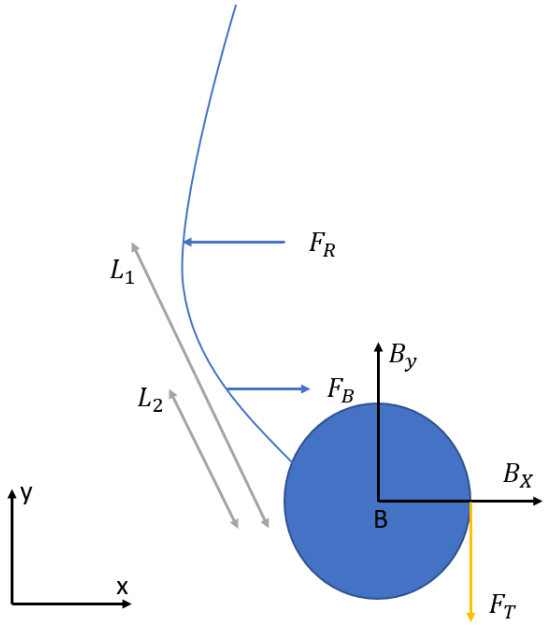


Figure 14: FBD of Arm for Stepper Motor or Shoulder Joint

Left Finger/Gear Parameters:

$$F_R = \ Previously\ Calculated\ Gripping\ Force = 4.57N$$

$$F_B = Force\ Applied\ by\ Rubber\ Band$$

$$L_1 = Distance\ to\ the\ middle\ of\ \ gripper\ finger = 0.0635m$$

$$L_2 = Distance\ to\ Rubber\ Band\ Standoffs = 0.0216m$$

$$\mathrm{B} = \ Center\ of\ Left\ Finger\ Gear$$

$$\mathrm{B_X} = \ Force\ on\ B\ in\ the\ X\ Direction$$

$$\mathrm{B_Y} = \ Force\ on\ B\ in\ the\ Y\ Direction$$

$$\mathrm{F_T} = \ Force\ from\ other\ finger$$

Equations of Equilibrium for Left Finger/Gear:

$$\sum M_B = 0 = F_B(L_2) - (F_R(0.2))(L_1)$$

Solving these equations for F_B indicated that the rubber bands need to apply a force of 26.8N in order to provide a similar gripping force as the servo motor at 20% stall torque. However, this is lowered due to the rubber band being layered over itself, which meant that each of the four layers of rubber band each must provide a force of 6.70N. It is also important to note that the force provided by the rubber band increases coincidentally with how open the gripper is, which therefore also increases the distance between the standoffs. Because of this, the gripper with the rubber bands cannot output the same gripping force as the gripper.

3.2.1.3 Two Degree of Freedom Arm

To adjust to varying terrain and positioning of trash, a 2-DOF arm was selected as the manipulator for collecting trash. The gripper mechanism was attached to the end of the 2-DOF arm to allow the trash to be picked up from the ground. The previous iteration of the project also

used a 2-DOF arm. However, this arm was made of 3D printed parts and was not robust enough to handle a significant amount of force. After analyzing the previous iteration of the arm, our team decided a redesign with more robust materials was necessary. The redesigned, 2-DOF arm features two aluminum links with the following lengths: 0.133m for the first link and 0.146m for the second link. The lengths of the links are set accordingly to allow the gripper to reach in front of the drive motors and to be in the view of the Smallbot's onboard camera. If the lengths of the two links were any shorter, the camera would not only have difficulties seeing the trash in front of it, but it would also be limited to the area that the arm can reach.

To determine the correct servo and stepper motors for the application, torque calculations were performed. The FBDs and the corresponding calculations can be seen below. The calculations were performed when the arm was straight, as seen in Figure 15 and Figure 16 below. The calculations assumed the arm to be lifting a half filled 473ml bottle of water or about 0.227kg. The required torque for the elbow and stepper motor turned out to have torque sufficiently below the stall torque for each joint.

Arm parameters:

$$Stepper\ torque_{stall} = 4Nm, Servo\ torque\ _{stall} = 2.94Nm$$

$$L1_{mass}\ and\ L2_{mass} = 0.023kg$$

$$Gripper_{mass} = 0.2kg, Elbow_{mass} = 0.1kg, Bottle_{mass} = 0.227kg$$

$$L1 = 0.133m, L2 = 0.146m, L3 = 0.051m$$

$$g = 9.81\frac{m}{s^2}$$

$$F_{B1} = L1_{mass} * g = 0.223N$$

$$F_{elbow} = Elbow_{mass} * g = 0.981N$$

$$F_{B2} = L2_{mass} * g = 0.223N$$

$$F_{gripper} = Gripper_{mass} * g = 1.96N$$

$$F_{bottle} = Bottle_{mass} * g \ = 2.22N$$

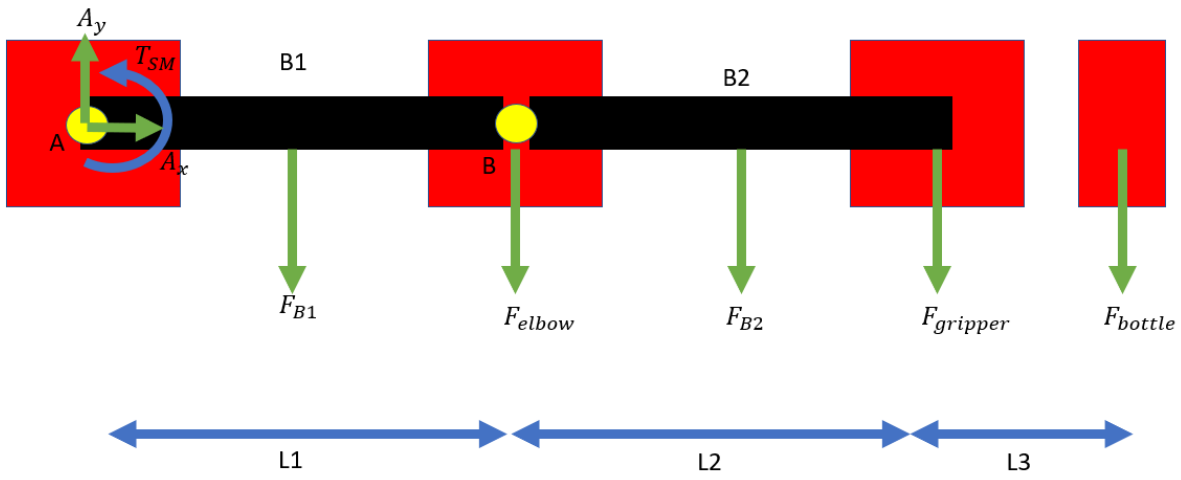


Figure 15: FBD of Arm for Stepper Motor or Shoulder Joint

$$\sum M_A = 0 = T_{SM} - (F_{B1})(\frac{L1}{2}) - (F_{elbow})(L1) - (F_{B2})\left(L1 + \frac{L2}{2}\right) - (F_{gripper})(L1 + L2)$$
$$- (F_{bottle})(L1 + L2 + L3)$$

Solving for T_{SM} the required torque for the shoulder joint is 1.47Nm which is below the stall torque for the stepper motor (4Nm).

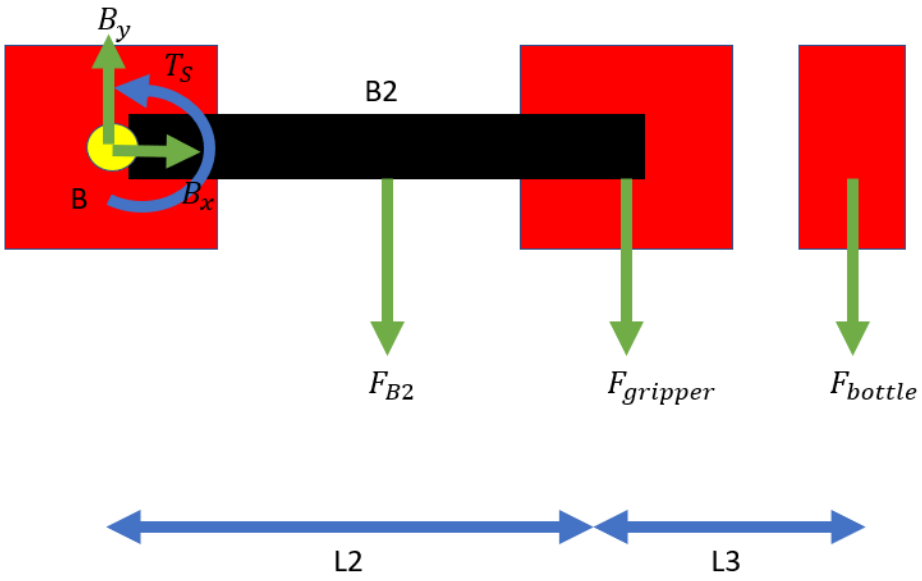


Figure 16: FBD of Arm for Servo or Elbow Joint

$$\sum M_B = 0 = T_S - F_{B2}(\frac{L2}{2}) - F_{gripper}(L2) - F_{bottle}(L2 + L3)$$

Solving for T_s the required torque for the elbow joint is approximately 0.741Nm, which is below the stall torque for the servo (2.942Nm).

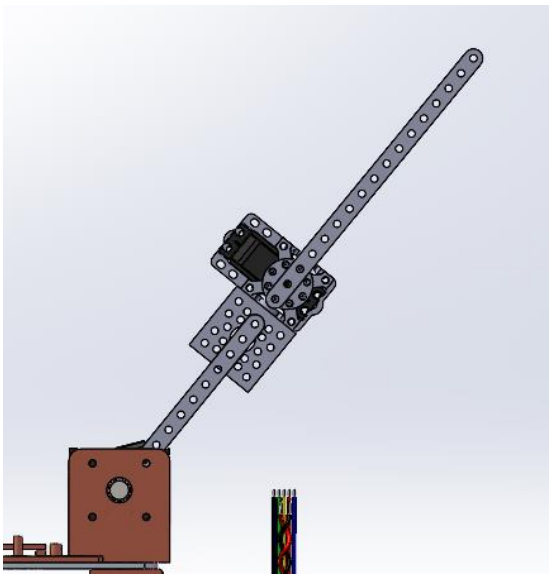


Figure 17: Side View CAD Assembly of Arm

Figure 18: Isometric View CAD Assembly of Arm

3.2.1.3.1 Kinematic Calculations

Denoting the Denavit-Hartenberg (D-H) parameters of the arm was an essential step in the early calculations of the 2-DOF arm's forward kinematics. To accomplish this, the D-H parameter table was filled out, as seen below in Table 1. The table was then used to derive the homogeneous transformation matrix from the arm's base frame to its end effector, as seen in Figure 19. A generic D-H matrix was calculated, which in turn allowed the forward kinematics of the arm to be computed for any given configuration.

	θ (degrees)	d (meters)	a (meters)	α (degrees)
T_0^1	θ_1	0	0.133	0
T_1^2	$-\theta_2$	0	0.197	0

Table 1: DH Parameters

$$\begin{pmatrix} \sigma_1 & \sigma_3 - \sigma_2 & 0 & \frac{133\cos(\theta_1)}{1000} + \frac{197\cos(\theta_1)\cos(\theta_2)}{1000} + \frac{197\sin(\theta_1)\sin(\theta_2)}{1000} \\ \sigma_2 - \sigma_3 & \sigma_1 & 0 & \frac{133\sin(\theta_1)}{1000} - \frac{197\cos(\theta_1)\sin(\theta_2)}{1000} + \frac{197\cos(\theta_2)\sin(\theta_1)}{1000} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$$\sigma_1 = \cos(\theta_1)\cos(\theta_2) + \sin(\theta_1)\sin(\theta_2)$$

$$\sigma_2 = \cos(\theta_2)\sin(\theta_1)$$

$$\sigma_3 = \cos(\theta_1)\sin(\theta_2)$$

The inverse kinematics of the arm were calculated through a geometric approach. By evaluating a simplified side-view representation of the arm, the equations needed to calculate the joint angles for any given set of end effector coordinates in the task space were computed, as seen below in Figure 19.

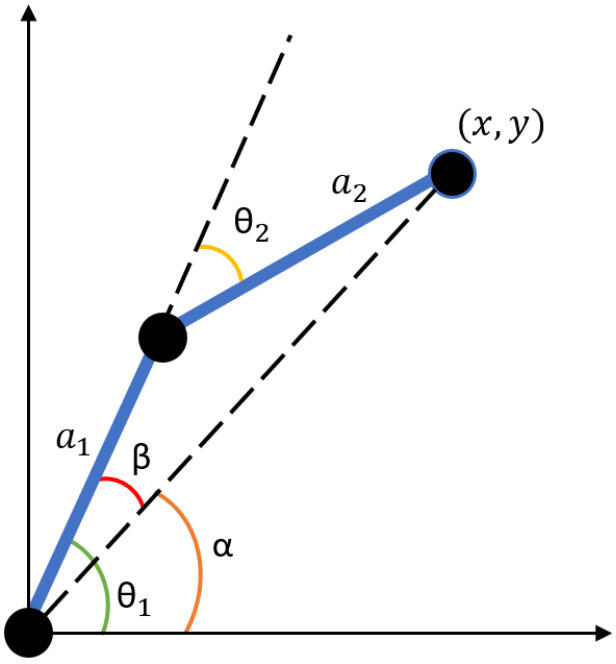


Figure 19: Inverse Kinematics

$$\alpha = tan^{-1}\left(\frac{y}{x}\right)$$

$$\beta = cos^{-1}\left(\frac{{a_1}^2 - {a_2}^2 + x^2 + y^2}{2a_1\sqrt{x^2 + y^2}}\right)$$

$$\theta_1 = \alpha + \beta$$

$$\theta_2 = 180 - cos^{-1}\left(\frac{{a_1}^2 + {a_2}^2 - (x^2 + y^2)}{2a_1a_2}\right)$$

3.2.1.3.2 Stress Analysis Calculations

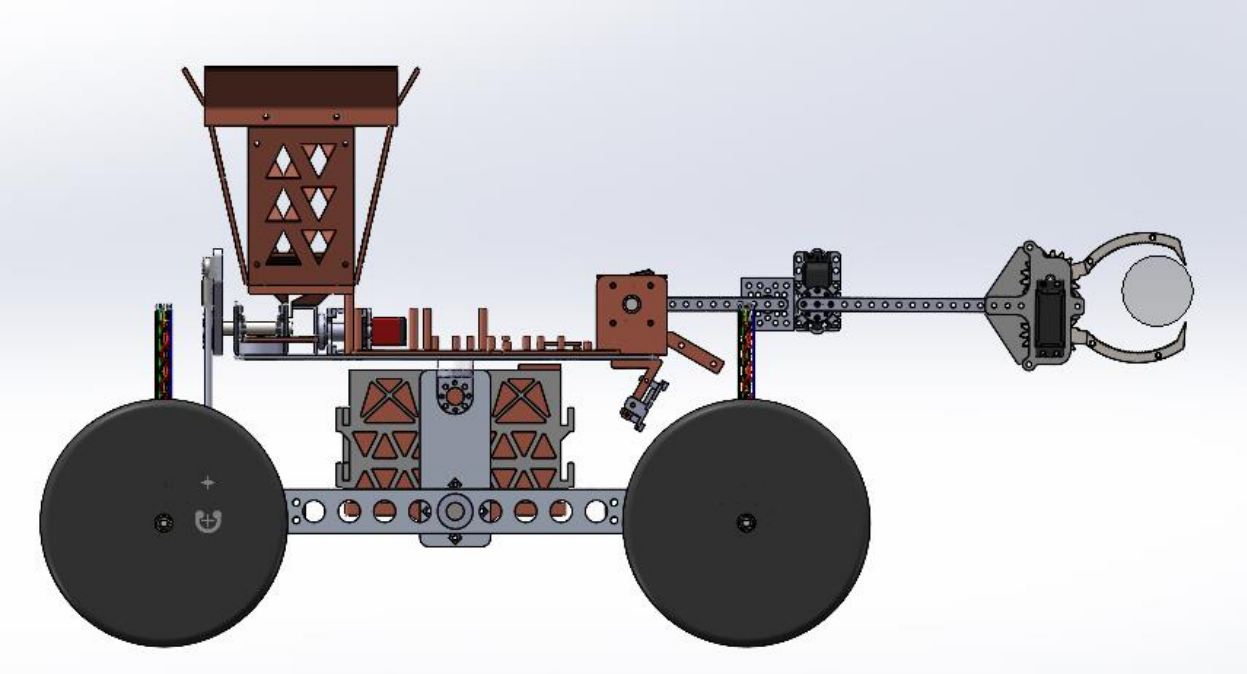


Figure 20: Depiction of the Arm at the Horizontal

It was imperative for the Smallbot's 2-DOF arm to undergo a stress analysis. This was done to ensure that the arm would not yield in the event that it was struck from the side when operating. These calculations were performed under the assumption that the 2-DOF arm was straight out horizontally, as this is the position in which the arm is most vulnerable, as seen above in Figure 20.

Assuming that the Smallbot would be collecting a half-filled, 473ml plastic bottle, weighing 0.227kg, calculations were conducted in the vertical direction to show that the stresses were low enough for the arm to withstand on its own. This weight is heavier than both empty plastic bottles and soda cans, which weigh 0.01kg and 0.0136kg on average, respectively. By

assuming a worst-case scenario, where the Smallbot collects a half-filled bottle, the feasibility of the system can be easily verified when picking up lighter objects.

The combined total length of the arm was 0.629m, and the moment about the shoulder corresponded to the previously calculated shoulder torque of 1.47Nm. The width of the examined cross section, W, is 0.00635m, while the height, H, is 0.00965m. The centroid for the cross section is located at a value of $c = \frac{H}{2}$, represented by the orange dotted line as the neutral axis. The cross section of the arm link can be seen below in Figure 21.

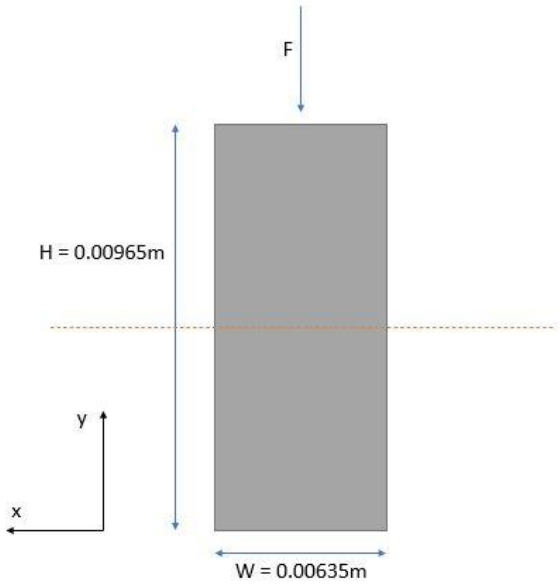


Figure 21: Cross Section of Arm Link

$$I = \frac{1}{12}(W)(H^3) = \frac{1}{12} * 0.00635m * 0.00965m^3 = 4.76 * 10^{-10}m^4$$

$$\sigma = \frac{M(c)}{I} = \frac{1.47Nm * \left(\frac{0.00965}{2}\right)}{4.76 * 10^{-10}m^4} = 15.9\ MPa$$

Here, the vertical stress in the arm is found to be 15.9 MPa. As the yield strength of 6061 Aluminum is between 124-290 MPa, reinforcing the arm in the vertical direction is not a concern since there is already such a large buffer.

3.2.1.3.2.1 Arm Reinforcement and Calculations

Figure 22: Two-Degree of Freedom Arm Reinforcement Piece Side View

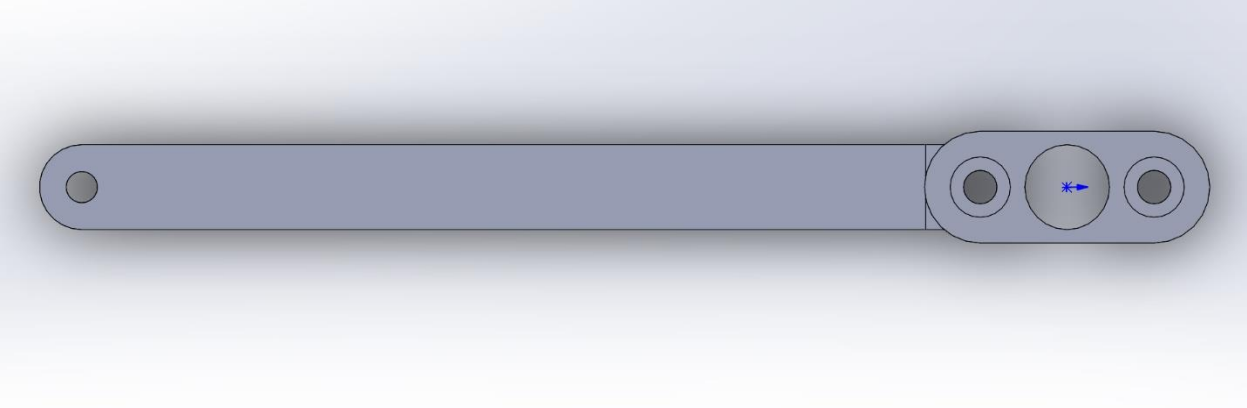


Figure 23: Two-Degree of Freedom Arm Reinforcement Piece Top View

While the arm might not need any support in the vertical direction, it is weaker in the horizontal direction due to the beam's cross-sectional area. To resolve this issue, a reinforcement

piece was added to the link of the arm between the stepper motor and the elbow joint, as depicted above in both Figure 22 and Figure 23. This reinforcement piece increased the moment of inertia, and therefore, decreased the magnitude of deflection and stress in the arm.

A stress analysis of the arm in the horizontal direction, both with and without the reinforcement link, was completed. The arm was examined under stress at a moment where, while the SmallBot would be turning, the arm would get caught and thus provide enough resistance for the wheels to begin to slip. At this moment, the stress in the arm is at its highest, so by examining the arm at this critical moment a further demonstration of the need for the reinforcement part is provided. The coefficient of friction, μ, was referenced to be 0.35 (from the "Table of Ultimate Friction Factors for Dissimilar Materials"), as for the purposes of this analysis, the SmallBot has not sunk into the sand (Fine Software, n.d.). As used previously, the width of the drivetrain, B, was 0.305m, while the weight of the robot, W_r, was 11.3kg. For this calculation, the total length of the arm, d, is 0.628m, which is the distance from the end of the arm to the center of the top plate. This is also the distance between where the arm experiences the force on the end and where the robot experiences the turning force of the drivetrain. First, the stress experienced by the arm without the reinforcement was examined. The width of the arm, W, is 0.00635m, while the height, H, of the arm is 0.00965m. To help with finding the Moment of Inertia, a diagram of a cross-section of the arm was created, shown in Figure 24. The orange, dotted line denotes the neutral axis, along which the centroid of the cross-section lies, at a location of $c = \frac{W}{2}$.

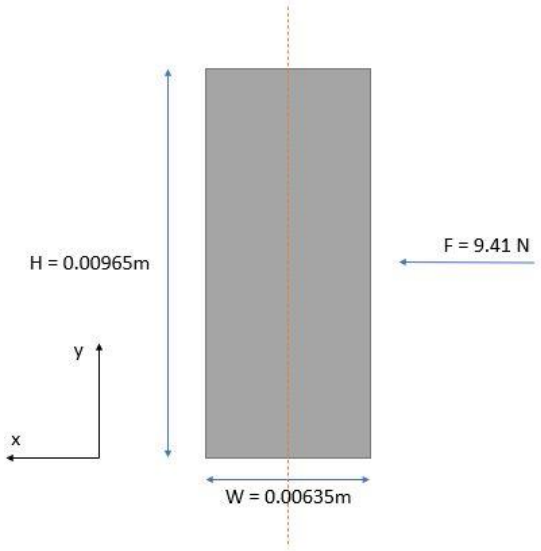


Figure 24: Cross-Section of Arm Without Reinforcement

Force Acting on End of Arm:

$$W_{bot} = W_r(g)$$

$$F = \frac{\mu(W_{bot}\frac{B}{2})}{d} = \frac{0.35 * 110.85N * \frac{0.305m}{2}}{0.628m} = 9.41N$$

Stress of Arm, without Reinforcement:

$$I = \frac{1}{12}H(W^3) = \frac{1}{12} * (0.00965m) * (0.00635m)^3 = 2.06 * 10^{-10}m^4$$

$$M = Moment = F(d) = 9.41N * 0.628m = 5.92\ Nm$$

$$\sigma = \frac{M\left(\frac{W}{2}\right)}{I} = \frac{5.92\ Nm * \left(\frac{0.00635m}{2}\right)}{2.06 * 10^{-10}m^4} = 91.3\ MPa$$

The arm, at this moment, experiences a stress of 91.3 MPa. While this is still less than the yield strength of 6061 Aluminum (as 6061 Aluminum begins to yield around 120 MPa), this can still be improved to further ensure the safety and longevity of the arm by adding the support. The reinforcement piece was made of 3D-Printed PLA, and was modeled as a combined, composite cross-section with the aluminum arm. This was accomplished by comparing the Young's Moduli of the two materials to find the equivalent dimensions. PLA has an average Young's Modulus of 7 GPa, 6061 Aluminum has an average Young's Modulus of 69 GPa. Therefore, the scaling ratio n is $\frac{7GPa}{69\ GPa}$, or 0.101. While the width of the reinforcement piece at the shoulder, W_R, does not scale, and remains 0.0127m as it acts as a spacer, the original height 0.00965m is multiplied by the scaling ratio to obtain a new height, H_R, of 0.000979m. This composite model is shown below in Figure 25. Once these were obtained, the new moment of inertia and maximum bending stress could be found. In order to find the location of the new centroid (marked by the orange line on Figure 25, which represents the neutral axis), the cross-sectional areas of the two parts had to be calculated. The cross-sectional area of the arm, A_A, is $H(W) = 0.00965m * 0.00635m = 6.13 * 10^{-5}\ m^2$, and the cross-sectional area of the reinforcement piece, A_R, is $H_r(W_r) = 0.000979m * 0.0127m = 1.24 * 10^{-5}\ m^2$. The centroid locations of the two parts were measured from the right side. The locations of the arm centroid and reinforcement centroid were calculated to be 0.0159m and 0.00635m, respectively, and are represented on Figure 25 as the blue dot (the arm) and the orange dot (the reinforcement). Using these area values and centroid locations, the new centroid, c, can be found using the below equation.

Centroid Location:

$$c = \frac{\sum x(A)}{\sum A} = \frac{(0.0159 * 6.13 * 10^{-5}) + (0.00635 * 1.24 * 10^{-5})}{(6.13 * 10^{-5}) + (1.24 * 10^{-5})} = 0.0143m$$

Now, the distances d_1 and d_2 can also be calculated. The distance between the arm centroid and the total centroid, d_1, is 0.00161m, while the distance between the reinforcement centroid and the total centroid, d_2, is 0.00792m.

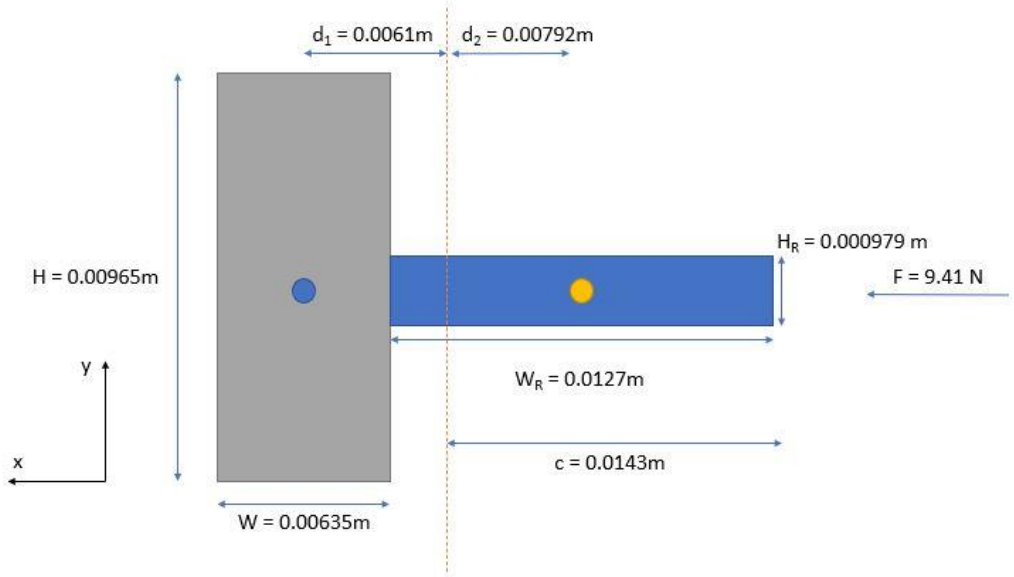


Figure 25: Cross-Section of Arm With Reinforcement Modeled as Composite

Stress of Arm, with Reinforcement:

$$I_R = \left(\frac{1}{12}H(W^3)\right) + (A_A d_1^2) + \left(\frac{1}{12}H_R(W_R^3)\right) + (A_R d_2^2)$$

$$= \left(\frac{1}{12} * 0.00965m * (0.00635m)^3\right) + (6.13 * 10^{-5}m^2 * (0.00161m)^2)$$

$$+ \left(\frac{1}{12} * 0.000979m * (0.0127m)^3\right) + (1.24 * 10^{-5}m^2 * (0.00792m)^2)$$

$$= 1.31 * 10^{-9}m^4$$

$$\sigma_R = \frac{M(c)}{I_R} = \frac{5.92\ Nm * 0.0143m}{1.31 * 10^{-9}m^4} = 64.4\ MPa$$

When the stress was analyzed at this position without the reinforcement part, the arm was found to undergo a maximum bending stress of 91.3 MPa. While this stress was below the yield strength of 6061 Aluminum, adding the reinforcement piece lowered the maximum bending stress at the shoulder to 64.4 MPa. This lower stress allows for a much safer range of operation for the arm and further expands the existing buffer in the event of a worst-case scenario.

After the manual calculations were completed, further stress analysis of the arm with and without the reinforcement was done. A wheatstone bridge and strain gauge were used to manually analyze the arm both with and without the reinforcement. These profiles are shown below in Figure 26 and Figure 27. Using Hooke's law to calculate the strain, we can find them to be:

$$\varepsilon_{\mathrm{A}} = \frac{91.3 * 10^6}{69 * 10^9} = 0.00132$$

$$\varepsilon_R = \frac{64.4 * 10^6}{69 * 10^9} = 0.00093$$

The orange lines drawn in Figures 26 and 27 below indicate where these strains would fall in the results, according to the trendline.

Figure 26: Stress & Strain of Arm without Reinforcement

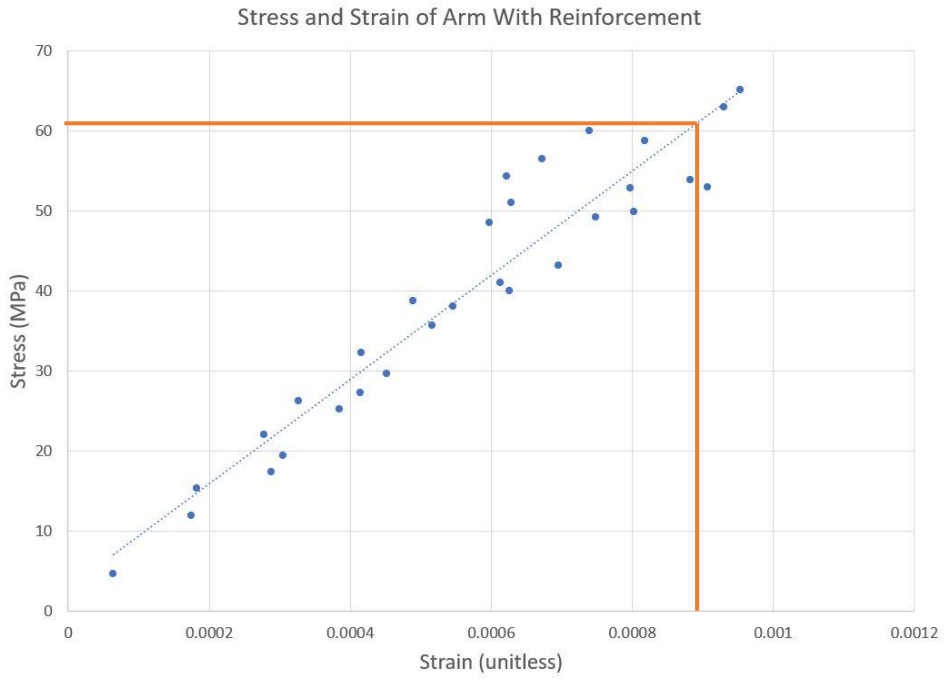


Figure 27: Stress & Strain of the Arm With Reinforcement

While the equivalent stresses shown with the orange lines in Figures 26 and 27 do not perfectly align with the calculated stresses, the maximum deviation of any given data point is around 20%, which is simply due to the margin of error in both the strain gauge, as well as the added human error that stemmed from approximating forces when using a spring force gauge. Therefore, it can be said that the measured data from the strain gauge agrees with the calculated stress.

Once this was analyzed, it was also found that the L-Bracket that holds the elbow servo onto the arm was another point of potential deflection, as a lot of force was put on the corner of the bracket. To resolve this, a reinforcement brace was made for the L-bracket as well, shown below in Figure 28. Calculations were also completed to demonstrate that this reinforcement would help improve the strength of the bracket. For the purposes of these calculations, the arm was assumed to extend straight out from the elbow, at a full length of about 0.330m. A collision force, labeled F_C, was applied at the end of the arm. The diagram shown in Figure 29 shows how this force was distributed and the torque it applied on the corner of the L-bracket while Figure 30 demonstrates how the forces are distributed when the reinforcement piece was applied.

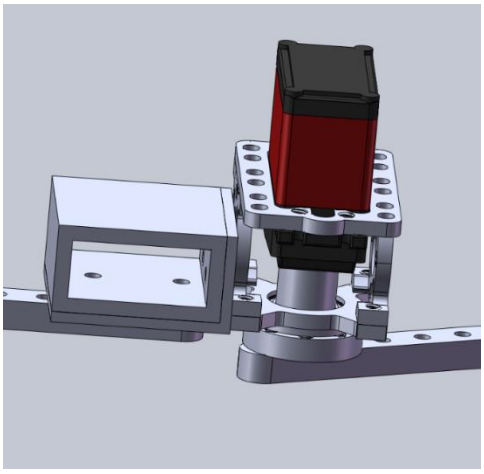


Figure 28: CAD of the Elbow L-Bracket Reinforcement

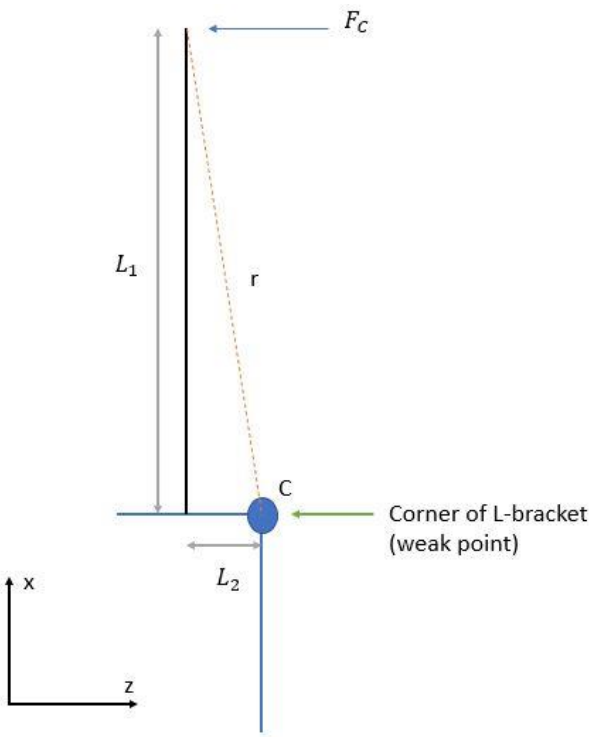


Figure 29: Force Distribution Diagram for L-Bracket

Parameters for L-Bracket Force Analysis:

$$C = Corner\ of\ L - Bracket$$

$$F_C = Collision\ Force, Assumed\ to\ be\ 10N\ for\ this\ analysis$$

$$L_1 = Length\ of\ Arm\ Link\ 2 = 0.330m$$

$$L_2 = Distance\ between\ Arm\ Connection\ and\ C = 0.00696m$$

$$r\ = distance\ between\ C\ and\ Collision\ Force = 0.330m$$

Equation for Torque acting on C:

$$T_C = r\,(F) = 0.330m * 10N = 3.30\,Nm$$

If the L-Bracket reinforcement piece shown above is added on, the force distribution diagram becomes:

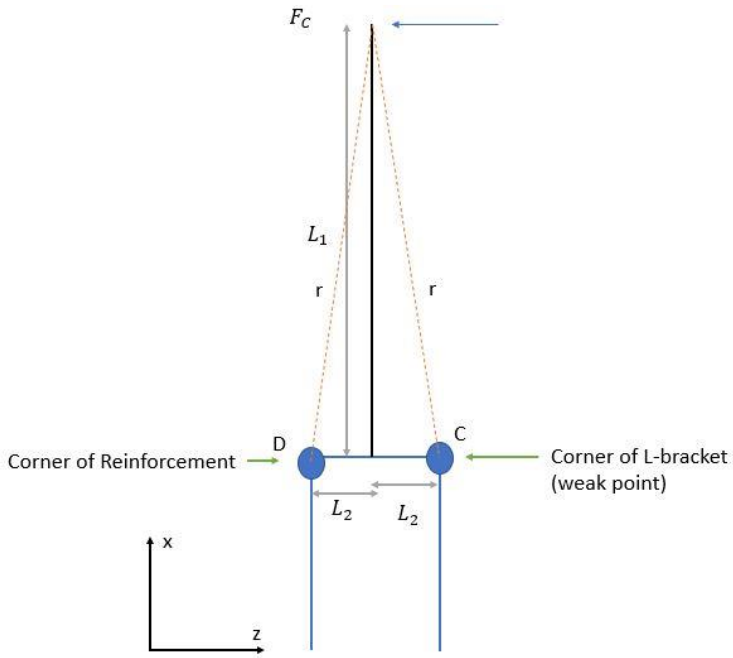


Figure 30: Force Distribution Diagram for L-Bracket with Reinforcement

Parameters for L-Bracket Reinforcement Force Analysis:

$$C = Corner\ of\ L - Bracket$$

$$D = Corner\ of\ Reinforcement$$

$$F_C = Collision\ Force, Assumed\ to\ be\ 10N\ for\ this\ analysis$$

$$L_1 = Length\ of\ Arm\ Link\ 2 = 0.330m$$

$$L_2 = Distance\ between\ Arm\ Connection\ and\ C/D = 0.007m$$

$$r\ = distance\ between\ C/D\ and\ Collision\ Force = 0.330m$$

$$\theta = Angle\ Between\ r\ and\ L_1 = \ cos^{-1}\left(\frac{L_1}{r}\right) = \ cos^{-1}\left(\frac{0.330}{0.330}\right) = 1.41°$$

Equations for Equilibrium at End of Arm:

$$\sum F_x = 0 = C_x - \ D_x$$

$$\sum F_z = 0 = \ F_C - \ C_z - D_z$$

Solving Equations for Torque acting on C:

$$C_x = D_x \rightarrow \ C_r * \cos(1.41°) = D_r * \cos(1.41°) \rightarrow C_r = D_r;\ C_z = D_z$$

$$0 = 10N - 2 * C_z$$

$$10N = 2 * C_z \rightarrow C_z = 5N$$

$$T_C = \ r * \ F = 0.330m * 5N = 1.65\ Nm$$

As shown via the above force calculations, when the reinforcement part was present on the L-Bracket, it greatly reduced the amount of force/torque on the corner of the bracket, which in turn reduced its stress and strain and made it more resistant to deflection.

3.2.1.4 Trash Storage Design

The dimensions of the bucket were imperative to the design, as the bucket needed to meet requirements to fit at least two, 473ml bottles and one 355ml can inside it. At the widest dimension, the bucket is 0.254m by 0.127m, and at the smallest dimension is 0.203m by 0.076m with 0.127m in depth. This allows for three 473ml bottles, as well as other combinations of bottles and cans to be stored. In addition, the bucket has cutouts throughout it, allowing for excess sand to exit upon rotation. The bucket is actuated by a servo at the end of the base plate. To determine the correct servo for the application, torque calculations for the servo were completed. These calculations considered the heaviest configuration of trash stored in the bucket, which was three half-filled 473ml bottles. In this calculation the center of mass of the bucket was assumed to be in the center of the bucket. The calculations can be seen below in Figure 31. From the calculations the required torque for the bucket and its contents was 1.25Nm, which was sufficient for the servos stall torque stated below.

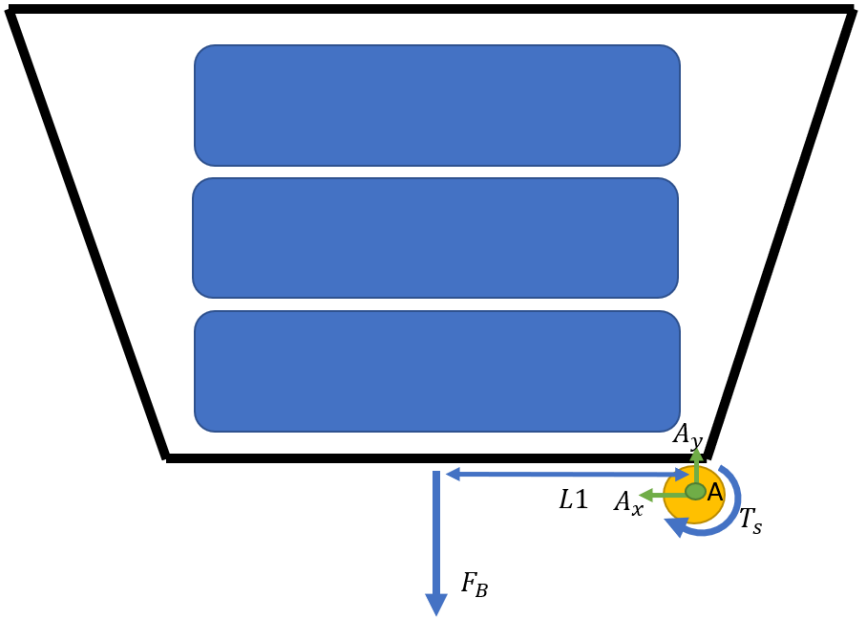


Figure 31: FBD of Bucket with Three Bottles

$$Servo\ Torque_{stall} = 2Nm$$

$$Weight\ of\ 3\ half\ filled\ 16oz\ bottles = 0.68kg$$

$$F_B\ (Weight\ of\ bucket\ and\ bottles) = 1.02kg$$

$$L1\ (Dist\ to\ Center\ of\ Bucket) = 0.125m$$

$$\sum M_A = 0 = F_B(L1) - T_s$$

Solving for T_S the required torque to rotate the bucket with three bottles was $Torque_{required} = 1.25Nm$. This required torque was below the servo stall torque which was

$2Nm$. The torque required is more than 60% of the stall torque. This mechanism will require future teams to make improvements to decrease the load on the servo.

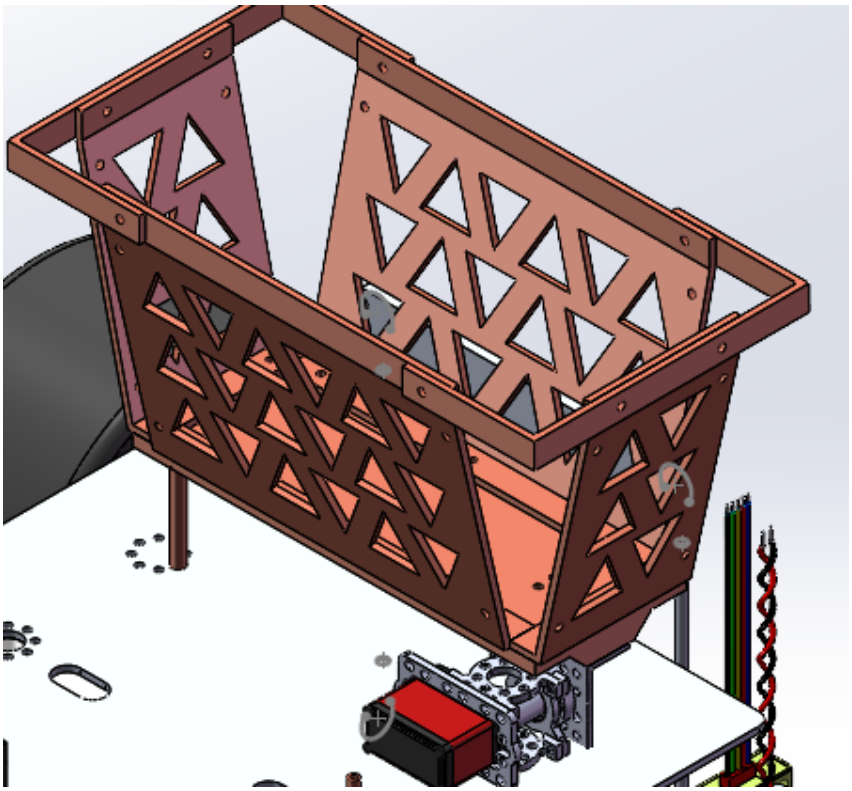


Figure 32: CAD Assembly Isometric View of Bucket

3.2.1.5 Camera Mount

To detect trash on a beach, it was important to have a camera mounted on the front of the Smallbot. The camera was mounted in the middle of the Smallbot and underneath the arm at a 30-degree angle below the horizontal. This angle was important to achieve the desired camera view of the surface. The desired camera view of the surface was where the bottom boundary of the camera was located, in front of the drive motors, and the top boundary of the camera was within reach of the gripper that was attached to the end of 2-DOF arm. The camera also needed

to be positioned in the center of the robot to allow the Smallbot to turn in either direction when a soda can or plastic bottle was detected off-center.

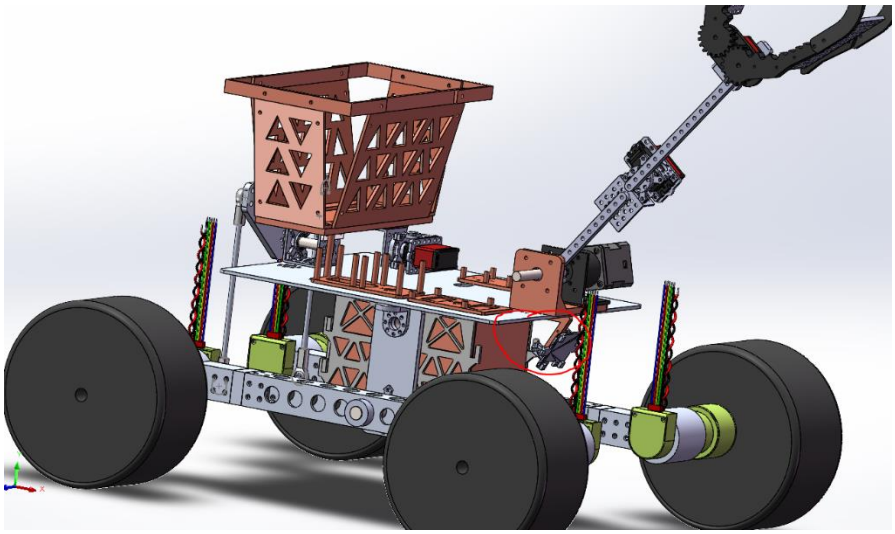


Figure 33: Isometric CAD view of Smallbot - Camera Mount is Circled in Red

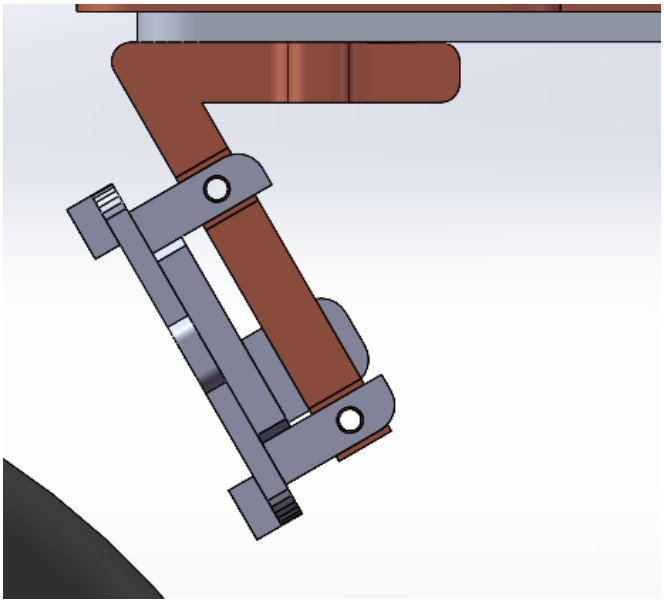


Figure 34: Detailed Side View of Smallbot Camera from CAD Assembly

3.2.2 Electrical Design

The electrical design of the first iteration of this project was evaluated and tested for functionality to meet this year's new requirements. From the previous iteration of the project, there were multiple problems with the electrical system and wiring. The issues observed are listed below:

- Cable management was unorganized.
- There were no wiring schematics, which made following the wiring and determining the types of components that were used difficult.
- Electrical components such as the IMU were still attached to a breadboard, which while helpful for testing, proved to have unreliable connections for a final product that may be deployed in the real world.
- The battery was duct-taped to the bottom of the chassis, which increased the risked of it falling off along with other electrical components.
- Most of the electrical components were exposed on the underside of the robot, which meant that sand and water could easily damage them.

To address the above issues discovered from the previous iteration and to meet the new requirements, the electrical system needed to be redesigned. In addition, to improve upon the issues listed, a battery to power all the components needed to last about one hour to clean the largest area possible without sacrificing weight or space on the robot.

3.2.2.1 Schematic

To improve upon the previous iteration and streamline the debugging process, a detailed schematic of the Smallbot was created. This schematic can be seen in Figure 35 below.

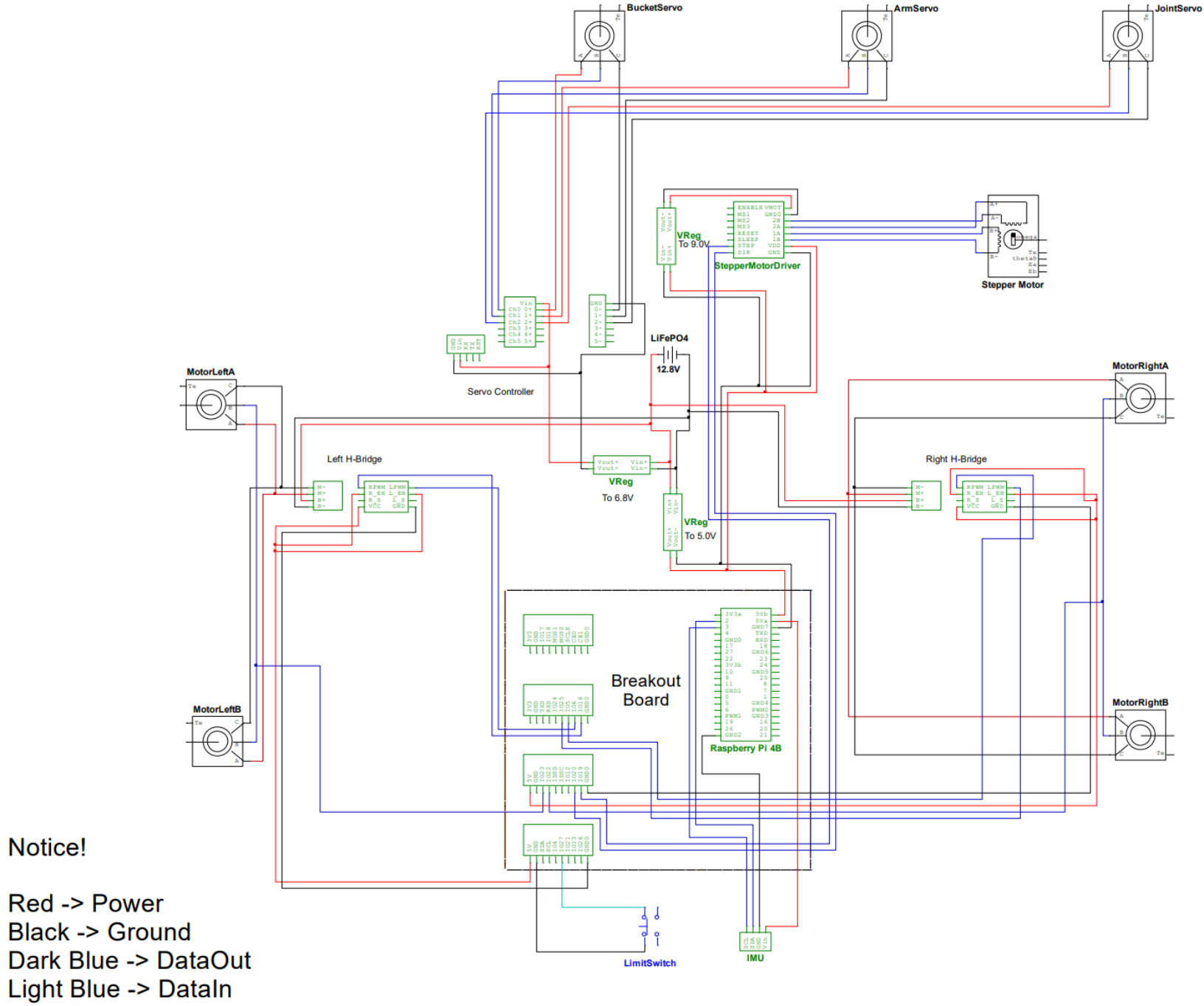


Figure 35: Schematic Diagram of the Smallbot

The Smallbot schematic was created for the user's convenience by following a similar layout to the mounted electronics on the Smallbot. By observing Figure 35, it can be seen that the Raspberry Pi and its breakout board are being centered in the middle of the schematic. The only exceptions are the H-Bridges, limit switch, and IMU. The H-Bridges are both located on the right side of the Smallbot, the limit switch was positioned under the arm servo, and the IMU was mounted under the bucket. The reason for this placement was to make the schematic as intuitive to follow as possible. For example, if the IMU was placed next to the bucket servo in the schematic, the viewer would most likely have trouble tracing its wires back across the whole schematic back to the breakout board GPIO pins to which they are attached. The Smallbot schematic was developed using NI Multisim.

3.2.2.2 Microprocessor Selection and Hardware Accelerator

Despite the last team's microprocessor selection being adequate for the scope of the project, it was decided to upgrade the system from a Raspberry Pi 3 Model B+ to a Raspberry Pi 4 Model B. This change was justified by the high computational load that was needed to run the system. The Raspberry Pi 4 not only has a high clock speed than its predecessor, but also features 4GB of RAM, as opposed to the Raspberry Pi 3 B+'s single 1GB of RAM memory. These improvements in the hardware have been reflected by a significant increase in performance, as seen in Figure 36 below (Hattersley, 2020).

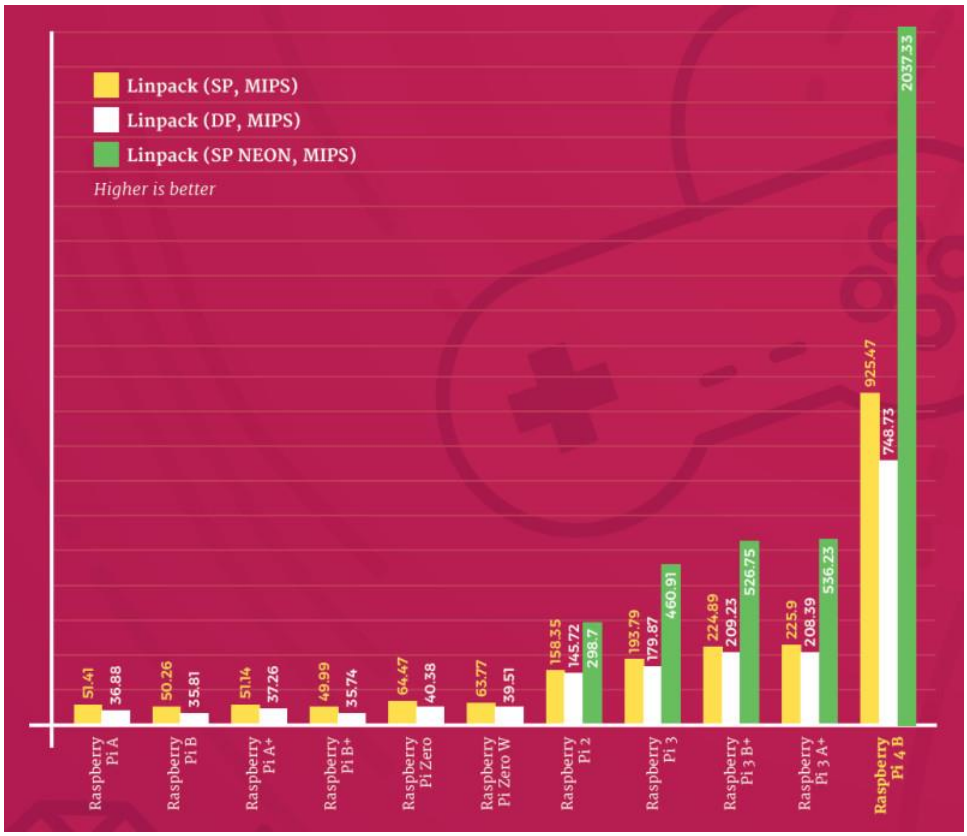


Figure 36: Raspberry Pi Performance Chart

Additionally, the Coral Edge TPU was an essential part of the system as it would be used to lighten the Raspberry Pi's workload when performing the inference on the video feed from the camera for object detection. The performance of running the TFLite model on the Raspberry Pi alone was approximately 2FPS: greatly draining the microprocessor's resources. However, with the aid the Edge TPU, the model performance was boosted to approximately 20FPS. An image of this TPU is provided in Figure 37.

Figure 37: Coral Edge TPU

3.2.2.3 Sensor Selection

A variety of sensors were used to maintain optimal control of the onboard mechanisms and for detecting visible trash. The main sensor that was used to detect these objects was the ELP 13-megapixel camera, as seen below in Figure 38. This USB camera was an important sensor to detect trash on the sand. The high-resolution camera sensor along with the wide 75-degree lens helped detect the litter scattered across the Smallbot's field of view.

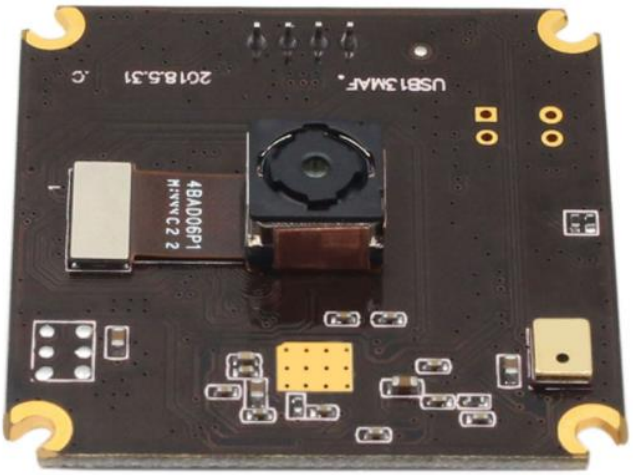


Figure 38: ELP 13MP Camera

Another sensor that was used on the Smallbot was the Adafruit BNO055 IMU. The IMU was a critical part of maintaining control of the Smallbot when driving through the sand. By using the IMU's heading, or Euler yaw angle, the Smallbot was able to maintain a consistent heading while driving on the sand. By maintaining its heading straight, the Smallbot was able to drive straight and turn regardless of changes in the terrain such as bumps, valleys, or sand inconsistencies.

Additionally, a limit switch sensor was used for control of the 2-DOF arm. As seen in Figure 39, the limit switch was mounted at the base of arm set 30 degrees below the horizontal. The limit switch allowed for the stepper motor on the arm to be calibrated from a known position. This happens when the limit switch was pressed or when the switch was set to ground. By resetting the arm's shoulder position, the Smallbot was able to maintain a consistent start position every time prior to object pick-up.

Figure 39: Picture of Limit Switch Mounted Below Shoulder Link of Arm

3.2.2.4 Battery Selection

A key requirement in our team's iteration of the Smallbot was to ensure that it could maneuver and drive on sand autonomously for an hour at a time. One hour of time was given as the requirement for the battery life because the Smallbot needed to clean the largest area possible without having a battery that would be too large or heavy for the chassis to handle. To determine the correct battery for the application, calculations were conducted with the necessary power requirements that would be included on the Smallbot. The maximum power for each of the Smallbot's components was found from their corresponding specification sheets. The total maximum power was determined by summing the maximum power of all the components. The power table in Table 2 below shows these calculations.

Component	Max Current Draw (Amps)	Idle Current Draw (Amps)	Max Power (Watts)	Idle Power (Watts)
Raspberry Pi 4B	1.25	0.6	6.4	2.7
Drive Motors (x4)	68	1.32	128	0
H-Bridge (x2)	-	-	-	-
Servo Controller	0.03	0.03	0.48	0.48
Bucket Servo	3.5	0.4	17.5	2
Elbow Servo	1.8	0.04	9	0.5
Gripper Servo	1.8	0.04	9	0.5
Stepper Motor Driver	0.04	0.04	0.2	0.2
Stepper Motor	1.68	0.4	8.4	2
Camera	0.4	0.4	0.22	0.22
IMU	0.0123	0.0004	0.04059	0.00132
Limit Switch	-	-	-	-
Total	78.5123	3.2704	179.24059	8.60132

Run Time	Hours
Max	1.071185941
Idle	22.3221552

Table 2: Power Calculations Table

From the power calculations, it was determined that a battery of 192.07Watt-Hours was necessary to run the Smallbot for 1 hour of time with all components running at maximum power. Although it would be unlikely for all components to simultaneously run at maximum power, our team wanted to consider the worse possible scenario, even if it would rarely happen. After researching different 12V battery types, our team was able to find the Miady LFP16AH 12V 16Ah Deep Cycle LiFePO4 Battery from Amazon. Not only did this battery have the necessary number of Watt-Hours, but it also was made from LiFePO4 chemistry which allowed for a smaller and lighter battery at just 1.8kg. An AGM battery with similar power specifications would have been slightly more inexpensive but would have weighed significantly more. The

lightweight battery helped keep the Smallbot's total weight below the initial 11.3kg calculated for the chassis.

3.2.3 Software Architecture

Having a robust codebase for the Smallbot was a priority, as it would not only allow for easier debugging, but would also influence the overall logic behind the robot's actions. The codebase for the Smallbot was written in Python. The reasoning behind choosing this language was due to the fact that Python is not only syntactically simple, but also very powerful when dealing with challenges that can be more easily managed when using a high-level programming language such as this one.

The functionality of each of the Smallbot's actuators and sensors was housed in designated classes, which were comprised of functions that carried out a set of defined actions that depended on their respective input parameters. The Chassis class introduced the methods needed to write any set of wheel efforts to the motors, drive straight with the use of the IMU, and perform point turns with the IMU. Furthermore, the Arm class allowed for individual movement of the joints in the arm, calibration of the arm shoulder joint with the use of the mounted limit switch, and the execution of a pre-set movements that allowed for pick-up of cans and bottles. The Vision class was used to perform inference with the TFLite model mentioned in section 3.2.3.1. This class included a set of methods that were used to perform the object detection and acquire the camera coordinates from a detected can or bottle through OpenCV. These

coordinates were then relayed to the DriveDetect class, which held the main logic for the Smallbot operations.

Once the OpenCV coordinates of an object were acquired, the methods within the DriveDetect class enabled the robot to align itself with a detected object. The DriveDetect methods allowed the Smallbot to turn towards the center of the object, drive forward to collect the object, and finally, drive back to the original position with a straight heading. Additionally, there were helper classes that were used to actuate the servos connected to the Maestro servo controller, obtain usable yaw data from the Adafruit IMU, stream a live video feed from the camera, and get up-to-date data from the Basebot. Ultimately, all these classes were used together within the main RunSmallBot class, which constantly requested data from the Basebot to perform a given action as can be seen in Figure 40 and Figure 41.

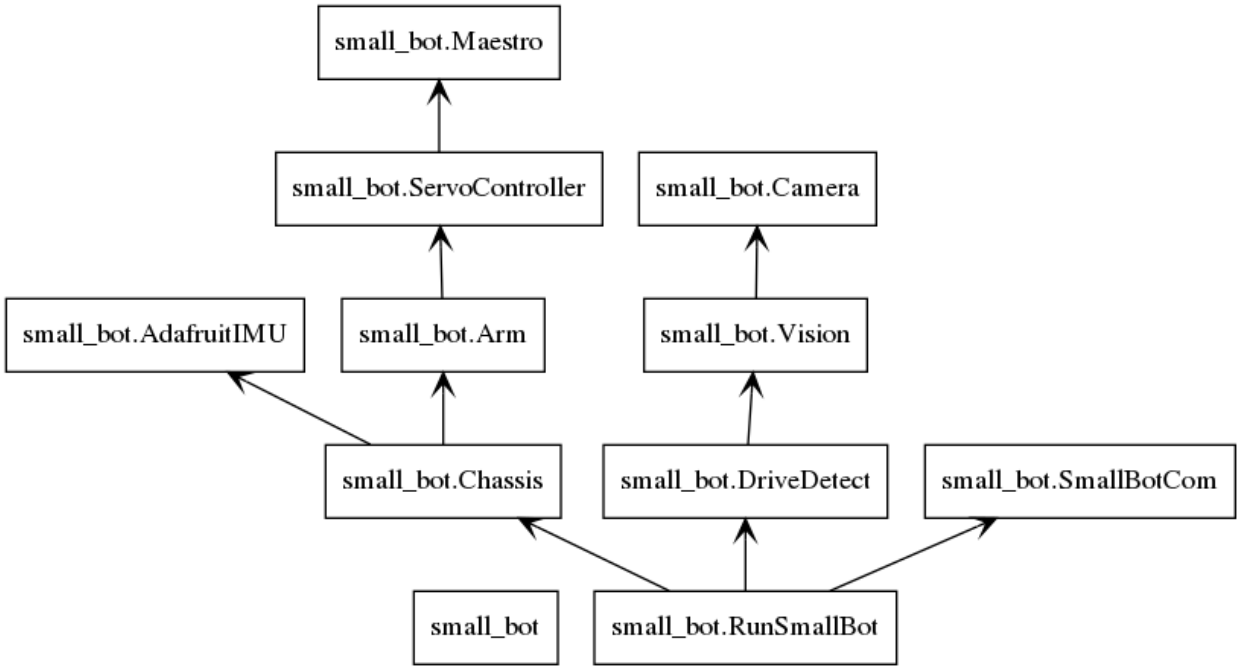


Figure 40: Smallbot Package Diagram

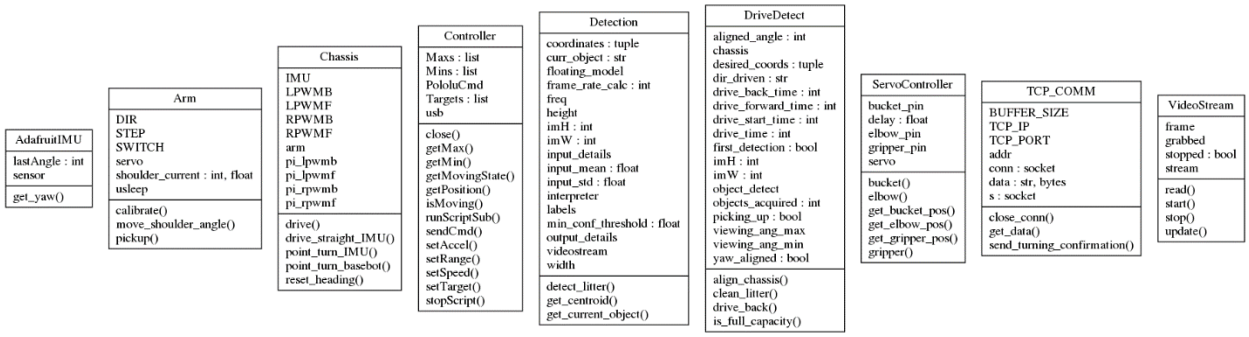


Figure 41: Smallbot UML Class Diagram

3.2.3.1 Chassis Alignment

As previously mentioned, in section 3.2.3, when a set of coordinates from a detected object were extracted from the live image through OpenCV, the Smallbot was commanded to align itself with it. To make this happen, it was necessary for a desired angle to be calculated for the Smallbot to successfully align its heading towards the detected object in a horizontal manner. This was accomplished by mapping the OpenCV x-values to the range of possible achievable headings from the IMU as represented in Figure 42.

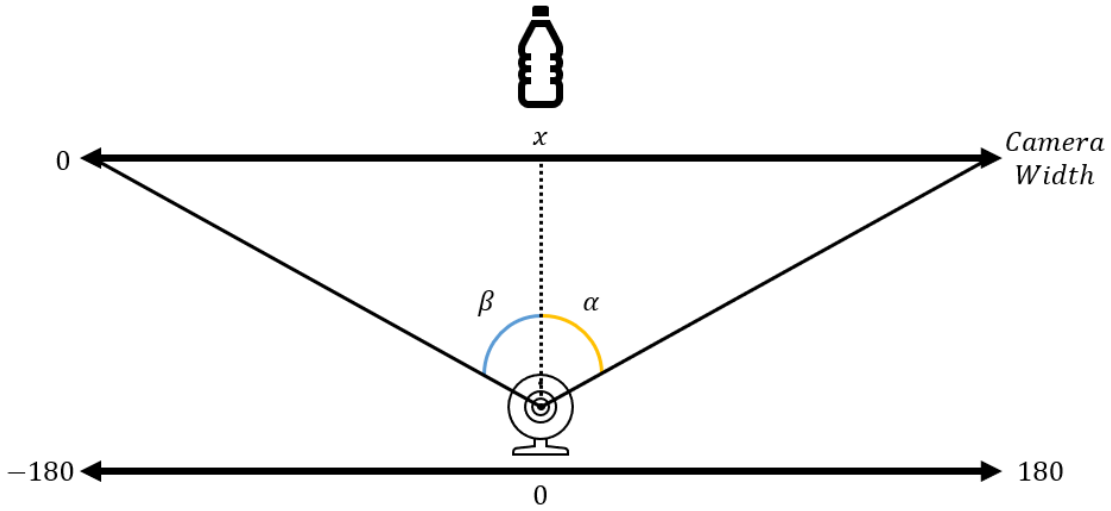


Figure 42: Visual Representation of the Yaw Alignment Process

To calculate the desired angle ($\theta_{desired}$) which the Smallbot would be required to turn towards, a ratio was found between the camera's width and the IMU's angle range. This ratio depended on the camera's viewing angle, as it ultimately limited the range for IMU angles from -180 to 180 to a more reasonable one that would ultimately yield a reliable $\theta_{desired}$ yaw angle.

$$\alpha = \frac{Camera\ angle}{2}$$

$$\beta = -\alpha$$

$$\theta_{desired} = \frac{x}{Camera\ Width\ (\alpha\ -\beta\ ) + \beta}$$

3.2.3.2 Object Detection

The previous iteration of the project only identified soda cans that were in a vertical position. To increase the functionality of the Smallbot, our team decided to improve upon the different can or bottle orientations that the camera could detect. Furthermore, our team decided it would be beneficial to train an entirely new model that would allow for the Smallbot to detect 473ml plastic bottles, alongside the 355ml cans from the previous implementation. The images used for machine learning training were based on standard, 473ml water bottles and 355ml soda cans.

It was essential to add other orientations of soda cans to improve the model's accuracy. This was achieved by adding more photos of cans, which were primarily taken from Google Images. To accomplish this task, a Google Chrome extension that allowed the user to automatically download all images within an open tab was used (Fatkun AI Downloader). This created an efficient and easy-to-use system to collect images for training. A vast array of pictures, consisting of bottles and cans, were collected to expand upon the previously used dataset. The can and bottle images also included crushed and partially visible cans or bottles to improve the model's accuracy. Ultimately, our team would need to label all of the images based

off what they represented and then train a TFLite model, which would also be quantized for the Google Coral Edge TPU.

3.2.3.3 Communication between the Basebot and the Smallbot

Transmission Control Protocol (TCP) wireless connection was used as the data transport protocol between the Basebot and the Smallbot. Although our team initially considered using User Datagram Protocol (UDP) as a means of data transmission, it was decided to use TCP due to a variety of reasons. Some key reasons for why TCP was chosen consist of the following: unlike UDP, TCP guarantees delivery of data to the destination router in the form of echo confirmation messages, and has a fixed order in which packets are sent and received (TCP vs UDP: What's the Difference?, n.d.). Having a defined order in which the data packets were being transmitted was imperative for not only debugging purposes, but also for the overall reliability of the dual-robot system.

An ESP32 microcontroller was used as a wireless access point (AP) between the Smallbot and the Basebot. By utilizing this AP as a hotspot, the conventional need of having a router as the AP was removed entirely. Once the framework for the communication was implemented, the Basebot (the client in this case) sent strings that were converted to bytes and then received by the Smallbot (in this case the server). These strings were then used as commands that directed the Smallbots actions based on where it was positioned relative to the Basebot's camera. The Basebot tracked the Smallbot's position from the mounted AprilTags as mentioned in section 3.3.3.2. This communication between the Basebot and the Smallbot was a

critical feature that can be expanded to a swarm of Smallbots collecting trash in a designated area.

3.3 Basebot System

The Basebot is the second robot in the beach cleanup system, which communicates with the Smallbot. The Basebot directs the Smallbot's actions based on where it is located with respect to its camera frame. The mechanical features of the Basebot and the camera selection were determined based on the list of tasks that it needed to achieve as an independent system. These tasks included: recognizing AprilTags and being able to communicate with the Smallbot via TCP communication.

3.3.1 Mechanical Design

As previously mentioned, the Basebot was stationary for this iteration of the project and therefore, a mechanical structure was not necessary. However, to test the full functionality between the Smallbot and Basebot, a camera stand was required. The camera stand pictured below in Figure 43 was used to test the Basebot's tracking of the Smallbot AprilTags. This stand was made to help with testing the correct height and angle that the Basebot camera needed to be set to.

Figure 43: Basebot Camera Stand for Testing

3.3.2 Microprocessor Selection

An Nvidia Jetson Nano microprocessor was used as the Basebot's electrical control board. The Jetson Nano was a low cost, yet effective system that included a GPU that was capable of handling intensive, graphics-processing tasks. The Jetson Nano's performance was especially helpful while running OpenCV for the AprilTag recognition. With the more capable GPU, the Jetson Nano could handle additional improvements to the mapping process of the beach, such as creating 3D meshes or recognizing/tracking multiple Smallbots in a swarm.

3.3.3 Software Architecture

The Basebot's main script used to execute the program was declared within the RunBaseBot class. This file contained the functions needed to send the necessary commands to the Smallbot via TCP communication, based on where it was positioned with respect to the Basebot's camera.

The ApriltagDetector class was used to perform the real-time AprilTag recognition and display it through OpenCV. This functionality was then utilized by the StateMachine class, which executed the set of calculated states for the Smallbot to carry out. Each defined state had an action attributed to it; The action was ultimately sent to the Smallbot by the BaseBotCom class, which introduced the functionality needed for the TCP communication between both systems. A graphical representation of the system can be observed in both Figure 44 and Figure 45.

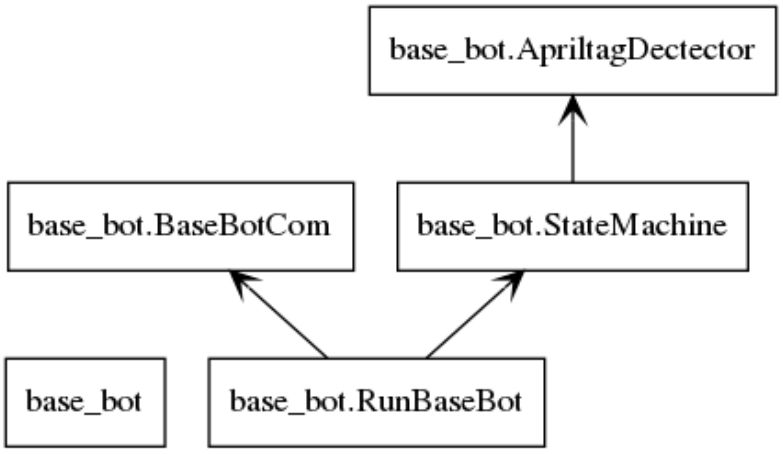


Figure 44: Basebot Package Diagram

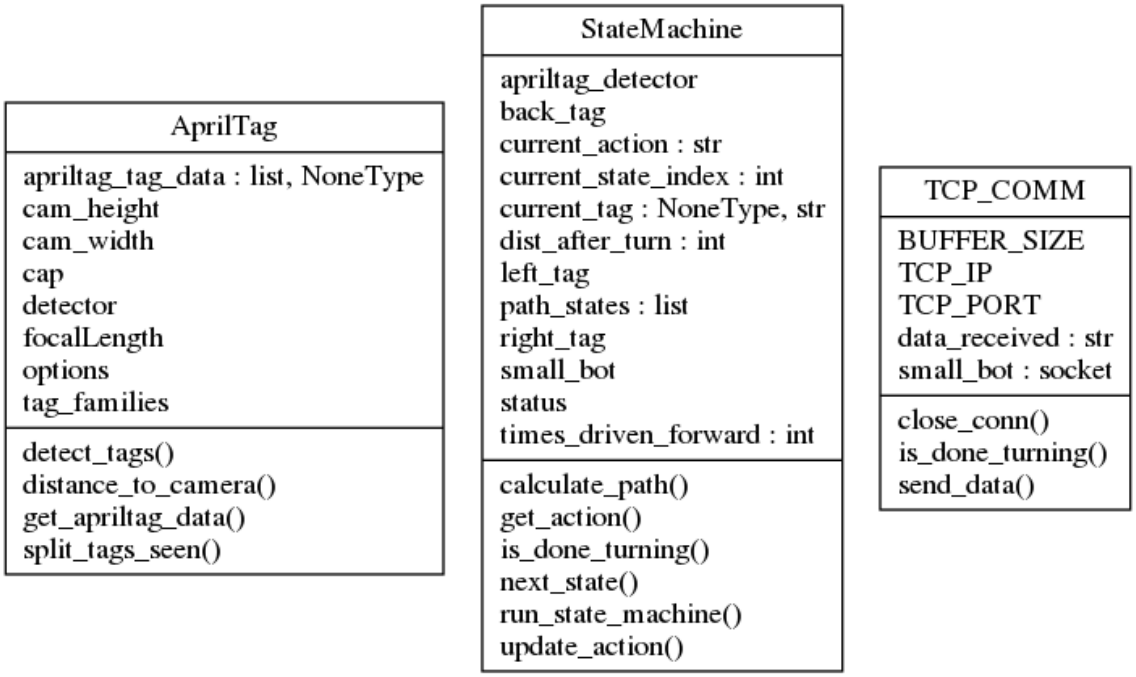


Figure 45: Basebot UML Class Diagram

3.3.3.1 State Machine Path Planning

A simple state machine was the chosen approach for determining the Smallbot's cleaning path. Though at first glance, a state machine may seem insufficient to fulfill the requirements for the task at hand, it happened to be an adequate technique for the purposes of this iteration of the Basebot. The main goal was to perform real-time AprilTag recognition and communicate with Smallbot.

The Smallbot's predetermined path was found in the form of a zig-zag pattern, where the Smallbot would drive from the right to the left side of the camera, drive forward for a given distance, and finally return to its starting position. This pattern was chosen because it would ensure full coverage of an area when driving through it. However, the number of laps was not

limited to the 2 seen in Figure 46, instead this number depended on the user's input parameter in the Constants support file. Ultimately, the Smallbot has the capability to perform an unlimited number of laps, as long as the AprilTags are visible by the Basebot's onboard camera.

Figure 46: Driving Pattern for Path Planning

3.3.3.2 Localization with AprilTags

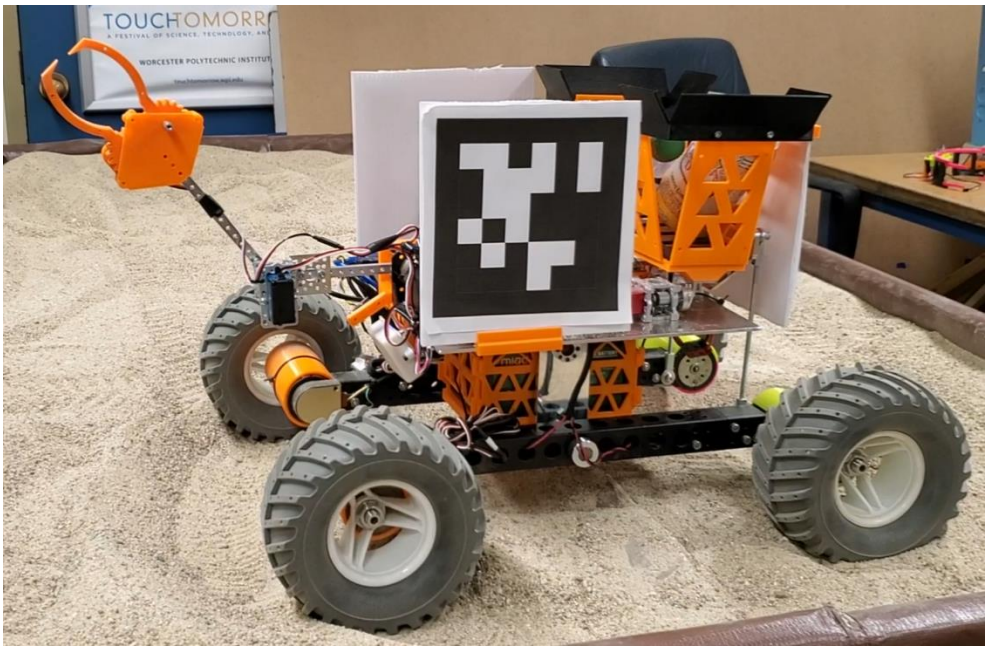


Figure 47: AprilTags on Smallbot

AprilTags were chosen as the marker to identify the Smallbots in the camera view of the Basebot. An AprilTag, which is a type of fiducial marker, is used a as reference object that is placed within the video frame of the camera view (Wikipedia, 2020). AprilTags resemble QR codes and consist of a pattern of black squares within a white background. This pattern of black squares makes them easy to be detected in a variety of sizes, angles, and lighting conditions. These AprilTags were mounted on both sides and the back of Smallbot, as seen above in Figure 47. By having AprilTags mounted on the right, left, and back sides of the Smallbot, the Basebot was able to track the position of Smallbot with respect to the camera position. By using the AprilTag package developed by the University of Michigan (Olson, 2010) in unison with

OpenCV, the Basebot was able to localize and send commands to the Smallbot according to where it was in its path.

To determine the distance between the Basebot's camera and the Smallbot, triangle similarity calculations of the AprilTag marks were derived as seen below.

$$F = Focal\ length\ of\ camera\ in\ pixels$$

$$D = Distance\ between\ camera\ and\ object$$

$$W = Width\ of\ object$$

Where the focal length was equal to:

$$F = \frac{P(D)}{W}$$

To calculate the focal length, an AprilTag was placed 0.6m away from the camera, and the pixel width was measured to be 170 pixels. The known width of the AprilTag was then calculated as 0.165m. With these values the focal length *F* was found:

$$F = \frac{170\ pixels * 0.6m}{0.165m} = 618\ pixels$$

Using this focal length, any distance between the camera and an AprilTag can be calculated by formatting the focal length equation to:

$$D' = \frac{W(F)}{P}$$

Where D' is the distance between the camera and AprilTag for a pixel width P that changes as the AprilTag was moved further away or closer to the camera.

Section 4: Results

To efficiently identify problems and establish solutions, our team followed the engineering design process, as seen in Figure 48 below. The first part of the engineering design process was identifying the problem. In this step our team identified the problem to be trash that was left on a beach. By identifying the problem, our team was able to research trash that was left on beaches and discuss ideas about potential solutions to tackle the issue at hand. From research and discussing ideas, initial designs were made as the solution to the problem. These designs were then used to create prototypes that could be tested. Testing these prototypes helped to quickly evaluate whether a design would serve as a proof of concept. If the testing was approved the design was improved to be used on the final product. However, if testing was unsuccessful our team would go back to the design and make the necessary changes to test it again or try a completely different design. The engineering design process was an important part of accomplishing the objectives for the project in a timely matter.

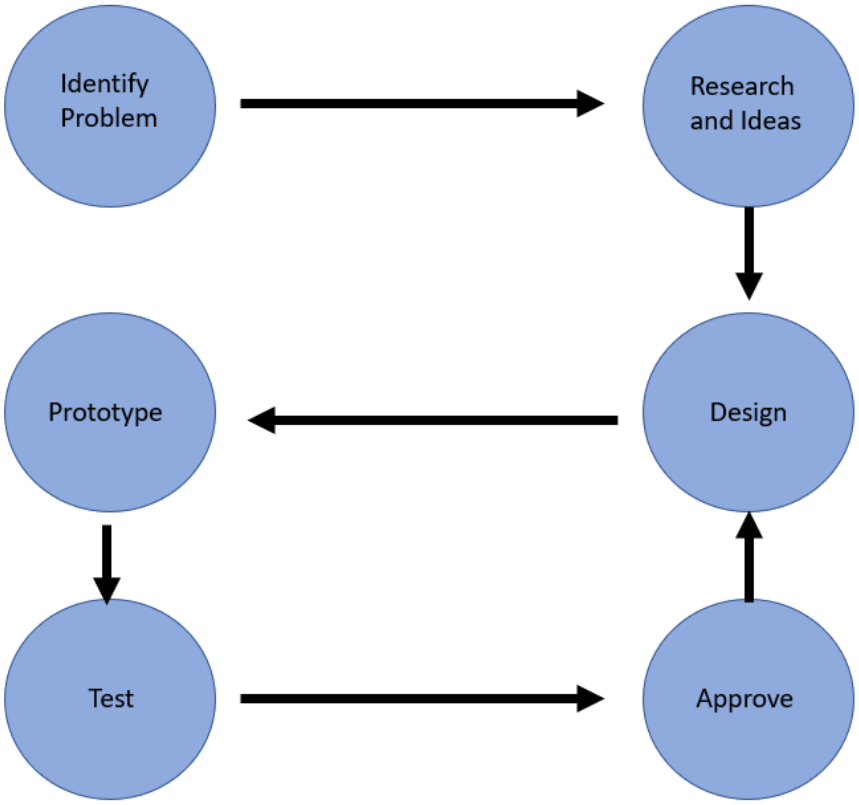


Figure 48: The Engineering Design Process

4.1 Prototyping

As discussed in the previous section, the following subsections discuss the prototyping that was completed for both the Smallbot and the Basebot.

4.1.1 Smallbot Prototyping

In order to test the new, four-wheel, rocker-bogie drivetrain for the Smallbot, a prototype first had to be created to hold the electronics.

4.1.1.1 Mechanical Prototyping

From the initial concepts of the drivetrain as presented in section 3.2.1.1, a drivetrain prototype was created. This prototype was designed to be a proof of concept of a four-wheel, rocker-bogie drivetrain. Some of the parts, including the four DC motors, the four 152mm wheels, and the anodized aluminum box extrusions for connecting the motors together for each side of the drivetrain had the potential to be used on the final design. However, the rocker-bogie assembly and the support parts to connect both sides together were made from plywood. The majority of the components were mounted by duct tape or zip ties. While working on this prototype, it was apparent that rigid and water-resistant materials would be necessary to meet our team's project goals.

4.1.1.2 Electrical Prototyping

The prototyping of the Smallbot electrical system was conducted in a linear fashion; from revising the previous team's electrical work, to optimizing and expanding upon what was available to work with for each given component. First, it was necessary for our team to analyze the previous team's work since it did not have proper documentation. Next, our team created a simple, Bluetooth Low Energy (BLE), prototype of the Smallbot using the new chassis and motors for a better insight into the drivetrain's functionality when driving on sand. This BLE-enabled prototype used the ESP32 microcontroller. After that, our team made the necessary changes needed to use the battery that would be in the final iteration of the Smallbot. Subsequently, each electrical component was individually tested with its corresponding mechanism on the Smallbot. For example, independently making the gripper servo and arm

function together. This was done to ensure that each joint in the system could be adequately actuated before soldering the wires to each electronic. Additionally, another aspect of the testing performed included: attaching each component to a power supply and a digital multi-meter to verify its power output, as well as determining how to properly implement servos and stepper motors while also testing their physical limits without damaging the components.

The final version of the prototype had all the electrical components soldered together; the one exception being any wires attached to GPIO ports on the Raspberry Pi. The final iteration of the electrical prototype was based off the early additions to the previous team's electrical work, new power calculations, and the routine testing of electrical components.

4.1.1.3 Software Prototyping

Several scripts were developed during the early prototyping stages of the Smallbot system. The overarching idea behind the creation of multiple small programs was to test each software component in a modular manner. By doing so, the debugging process was streamlined considerably, and the functionality of each aspect of the Smallbot could be tested in an effective and reliable way. These test programs included: scripts for speed controllers, encoder data recording, TCP communication with the ESP32, scripts for the stepper motor and servos, and developing a testbench for the TFLite model. After each one of these main components was tested and the functionality was verified, the final prototyping for the rest of the codebase on the Smallbot was finalized in a linear, and non-problematic manner.

4.1.1.4 Object Detection Prototyping

Once all the images of both soda cans and plastic bottles were collected, stored, and properly labelled, our team preceded to training models. To train a machine learning model for detecting bottles and cans, Google Cloud's AutoML training platform was utilized. AutoML was chosen for two main reasons: the previous team had written a brief explanation on how to properly utilize the platform, and when signing up with a new Gmail account, the user is granted $300 worth of credit towards Google's cloud services. This credit would be more than enough for our team to properly train two different models.

Preparing each model involved the steps listed below:

- Select multiple images to upload to Google Cloud project.
- When images are done uploading to cloud, create labels for the images.
    - In this project's case, the labels 'can' and 'bottle'.
- Go to the uploaded images and label them appropriately with bounding boxes; an example is provided in Figure 49.
- Navigate to "train" tab, and train new model.

Figure 49: Google Cloud Image Labeling

Our team often referenced Google's documentation for assistance on specific details when training the model (Google, n.d.). Once the model was trained, it could be downloaded for implementation with a variety of different frameworks; one of which being TFLite. The models could then be downloaded and stored on a personal computer for future use. Following this, our team would reuse the Google Coral Edge TPU Accelerator, which the previous team made use of. As previously explained on section 3.2.2.2 of the Methodology, the accelerator adds an Edge TPU co-processor to the Raspberry Pi, which takes the demanding computational workload accompanied with implementing a neural network off of the Pi.

Following model training, our team noticed that Google Cloud's services no longer supported the export of quantized, TFLite models for proper use on Edge TPU accelerators.

Ultimately, our team decided it was not optimal to use another model training service due to time constraints. However, example models were available from the Coral.ai website which our team made use of. These models were able to detect up to one-thousand common objects, including bottles. One downside of using these models was the high rate of false-positive detections that we experienced when testing.

Our team eventually reached out to a fellow WPI student (Grant Perkins) who had extensive experience with object detection and machine learning. At the time, Grant was developing software to train TFLite models which would be compatible with Edge TPU accelerators. Furthermore, he was able to recommend our team a different model labeling website, named Supervisely, which we utilized for a more efficient image-labelling process (Supervisely, n.d.). With Supervisely, our team was able to upload and label images simultaneously; in a similar fashion to how Google Docs works. Due to this, the time to prepare image data for training was exponentially reduced while also allowing for our team members to add more images of their choosing, which eventually increased the original image count from ~600 to ~1300. Supervisely, in terms of preparing image data, was vastly superior to Google Cloud for the following reasons: the user interface was smooth and reliable, newly labeled images were saved in the background while the user continued to work on different images, and most importantly, the Supervisely service that our team was utilizing was completely free of charge.

With this new software at our team's disposal, a new model was able to be trained that consisted of cans, bottles, and 'not' datasets. The 'not' dataset was comprised of bounding boxes that were used to identify everything in an image that is not a bottle or a can —essentially acting

as reject case when performing inference on the video feed from the camera. The addition of the ‘not’ dataset greatly improved the accuracy of the model to the point where there were no false positives on objects that were not bottles or cans, whereas previous iterations of the model would falsely identify almost everything including, sand mounds, as a false positive. Displayed to the left in Figure 50 is a labeled image from the Supervisely platform. To the right in Figure 50, the tan boxes represent ‘not’ while the green and turquoise bounding boxes represent ‘bottles’ and ‘cans’, respectively.

Figure 50: (From left to right) An image with ‘bottle’, ‘can’ and ‘not’ bounding boxes

The final steps to prototyping our team’s object detection, machine learning model consisted of making use of Grant’s software, named Axon, which allowed a TFLite model to be trained on a set of labeled images. Axon is written in Python and runs in a Docker environment.

Training with Axon was simple and intuitive as opposed to Google Cloud's training platform. When data preparation was completed in Supervisely, the newly edited image dataset was downloaded. This consisted of all the images which were originally uploaded, but with the addition of bounding boxes drawn on them. The download also included a very important .json file which consisted of the dataset's classes, representing their bounding boxes, respective IDs, and colors which were all needed to train the model successfully. Provided in Figure 51 is an image of Axon's training UI.

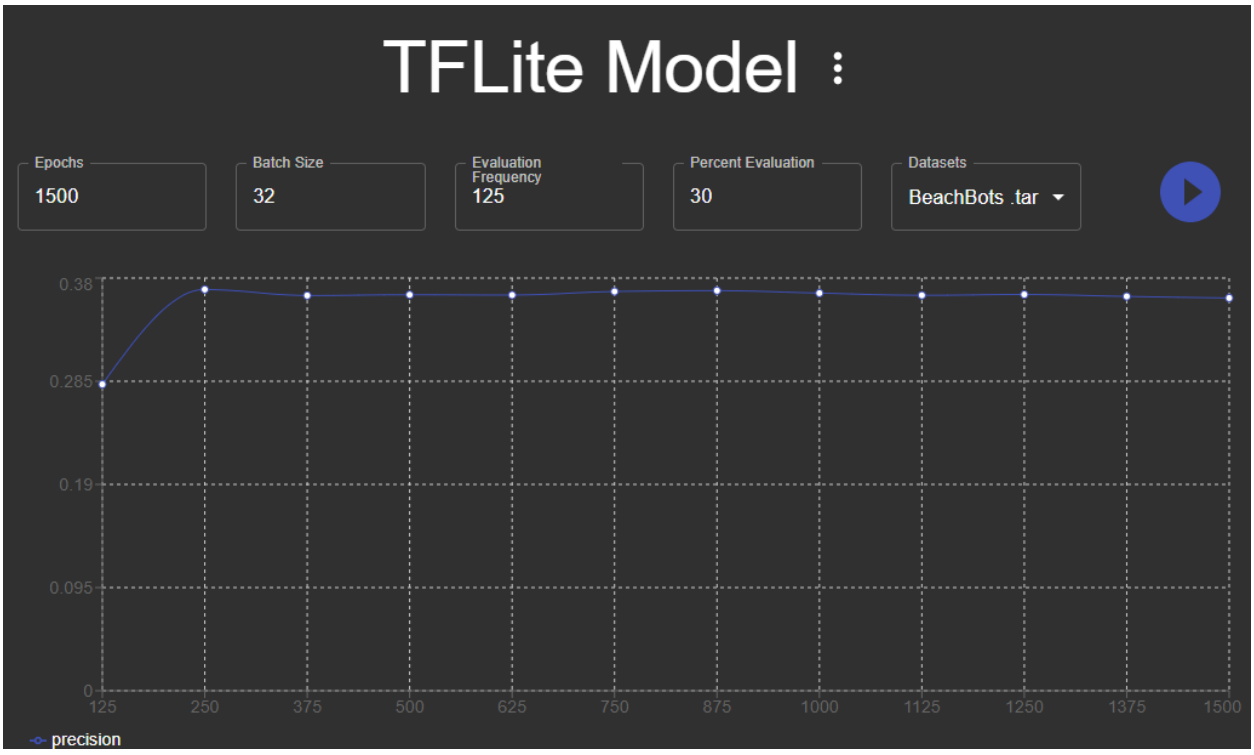


Figure 51: Axon User Interface

It can be seen in Figure 51 that training a TFLite model with Axon requires the user to input five arguments: Epochs, Batch Size, Evaluation Frequency, Percent Evaluation, and

Datasets. ‘Epochs’ represent how many passes the whole dataset will go through the neural network in one training session. ‘Batch Size’ indicates the amount of data that will be used for training the model per epoch. ‘Evaluation Frequency’ dictates how many times the neural network will make use of training examples to evaluate the accuracy of the model. ‘Percent Evaluation’ symbolizes how much training data will be used for accuracy improvement purposes. Finally, ‘Datasets’ is where the user inputs a .tar file containing the necessary dependencies and images; in our case, the files downloaded from Supervisely. Under the training settings in Figure 51, there is a graph depicting the accuracy of the model after each Evaluation. The user is able to select any node on this graph and download the TFLite model with that respective accuracy.

It can be seen in Figure 51 that the model training accuracy graph seems to plateau after the first accuracy evaluation. This was because while training the model, its accuracy will eventually stop improving. This was mostly based off what data the neural network was provided with. This trend is very common when training neural networks, however; the length of the plateau heavily depends on the training data as well as the type of neural network that is being used to train the model.

While observing the model accuracy graph in Figure 51, the best node to download an instance of the model from seems to only have an accuracy of ~37%. It is important to note that even though training accuracy is low, real world accuracy will usually be much greater since the model, in the case that it was being used by our team, will be provided with a high-quality camera feed with most of the frame being filled with the object that needs to be detected. In a

way, the accuracy improves because the Smallbot's environment was outputting the specific data that it required.

Finally, with a successfully trained TFLite model, that was also compatible with the Coral Edge TPU, our team was ready to move onto the testing phase for object detection.

4.2 Testing

The following sections discuss the testing that was completed with the Smallbot and the Basebot, as well as when the two robots were integrated together.

4.2.1 Smallbot Testing

To ensure that Smallbot was fully functional throughout this project, various testing had taken place. This occurred both at local beaches near the WPI campus and/or at our team's given laboratory space on a 1.22m by 2.44m table covered in sand. Testing was conducted in the lab space during the winter months due to inclement weather conditions that could be hazardous to the onboard electrical components. The purpose of these tests was to verify that our team's progress was significant, as well as to determine if the designs and prototypes that were developed would work for the full scope of the project.

4.2.1.1 Mechanical Testing

To ensure the mechanical redesign of the Smallbot was sufficient, various components were tested to determine whether they were able to withstand normal wear-and-tear, as well as any slight bumps that may arise.

4.2.1.1.1 Drivetrain Testing

The testing of Smallbot's prototype chassis was done at a local beach. The purpose of this test was to test how the four-wheel, rocker-bogie drivetrain operated on sand, dunes, and holes. The local beach had coarse sand that was somewhat damp, similar to what can be seen at the shore near the tide of an ocean. Therefore, these beach conditions were helpful in simulating what the Smallbot could encounter at an ocean. During testing, the Smallbot had no issues when driving over dunes or valleys up to 88.9mm on one side of the drivetrain. The drivetrain also had no issues with turning on the sand since it minimally dug into the sand, which was a concern of the wheeled design. In addition, the Smallbot's drivetrain prototype was able to reach a top speed of about $0.62\,\frac{m}{s}$ during testing, even with additional 3.63kg mounted on the robot. Overall, the prototype chassis was successful and was helpful to determine improvements that needed to make for the final design.

Figure 52: Chassis Prototype Design

The rocker-bogie drivetrain was also tested to find the experimental coefficient of friction while performing a 90-degree turn. To calculate the experimental coefficient of friction, encoders were used to determine the wheel speed while running the motors at 20% effort. Using this speed, the coefficient of friction was found by using the equations below.

$$t = Time\ of\ turn = 9sec$$

As observed in section 3.2.1.1 of the Methodology, the measured radius of the wheels was found to be approximately 0.076 meters (R_w). Thus, the wheel circumference was calculated with equation below.

$$C_w = Wheel\ circumference = 2\pi(R_w) = 2\pi * 0.076m = 0.477m$$

The number of encoder ticks (CPT) were recorded throughout the duration of the turn (t), and the resolution of the encoders (CPR) was found in the motor datasheet (PittmanExpress, 1994). The recorded values were the following:

$$CPT = 631\frac{counts}{turn}$$

$$CPR = 500\frac{counts}{rev}$$

Subsequently, the wheel speed (ω) was calculated in order to cross-reference the torque required to drive the motors at such RPM.

$$R_t = Revolutions\ per\ turn = \frac{CPT}{CPR} = \frac{631}{500} = 0.792\text{rev}$$

$$R_s = Revolutions\ per\ second = \frac{R_t}{t} = \frac{0.792\text{rev}}{9sec} = 0.088\frac{\text{rev}}{sec}$$

$$\omega = R_s(60) = 0.088\frac{\text{rev}}{sec} * 60sec = 5.28\text{RPM}$$

Once the RPM was found, a new motor graph like the one displayed on Figure 2, under section 3.2.1.1 of the Methodology, was generated to match our voltage output when driving the motors at 20% effort (2.4V).

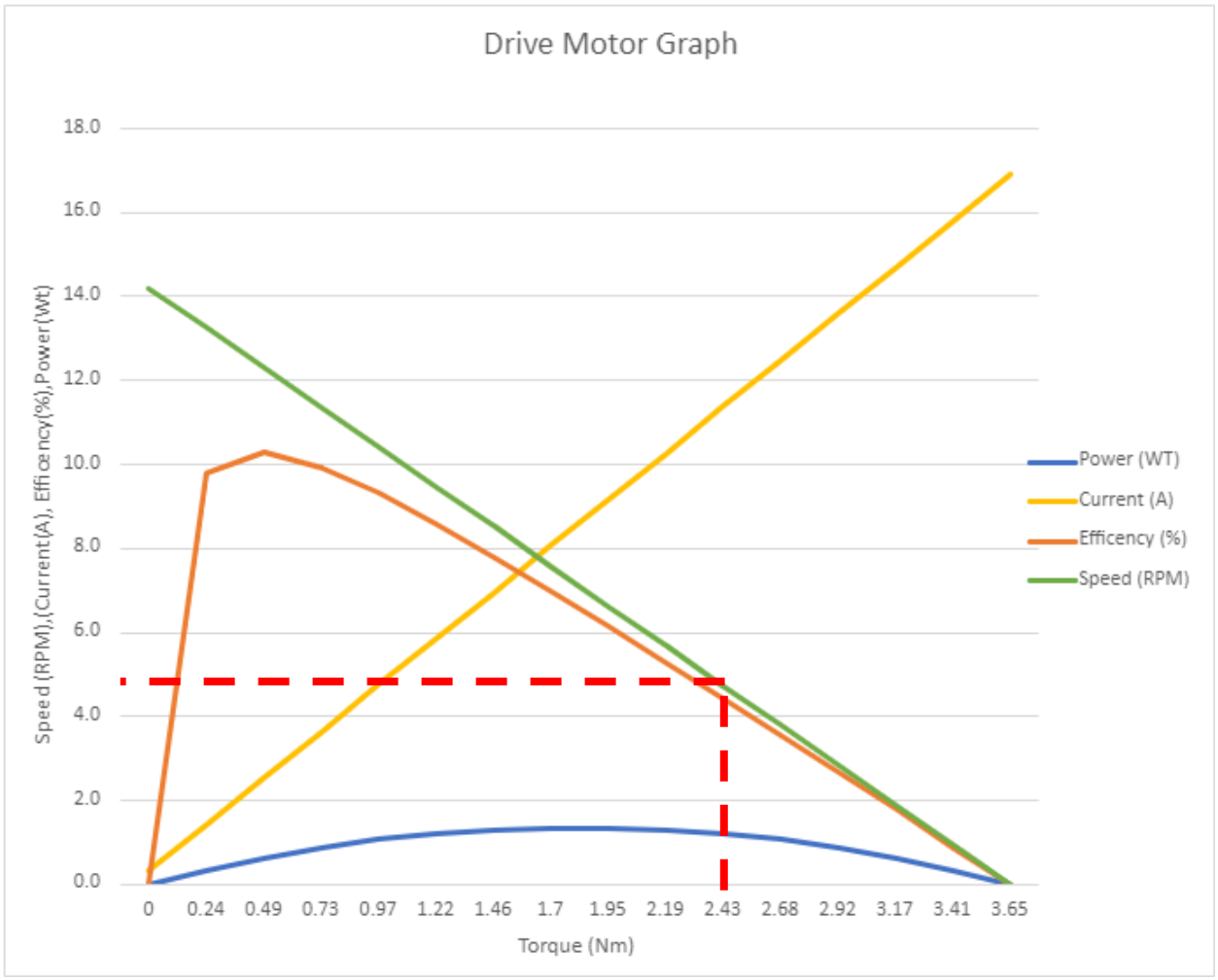


Figure 53: Drive Motor Specifications Graph at 2.4V

From the graph above, it can be deduced that the torque required to drive the motors at 5.28RPM is approximately 2.43Nm. This ultimately allowed us to calculate the experimental coefficient of friction, by reworking the equations used to calculate the theorical torque required per wheel with an assumed coefficient of friction of 0.35.

$$F_t = \frac{\tau_w}{R_w} = \frac{2.43Nm}{0.076m} = 32.0\text{N}$$

$$F_r = F_t = 32.0N$$

$$F_R = \sqrt{F_r^2 + F_t^2} = 45.3N$$

$$F_R = \left(\frac{W_t * g}{4}\right)\mu$$

$$\mu = \frac{F_{resultant}}{\left(\frac{W_r * g}{4}\right)} = \frac{45.3N}{\left(\frac{11.3kg * 9.8\frac{m}{s^2}}{4}\right)} = 1.64$$

The experimental coefficient of friction value of 1.64 seemed reasonable, despite the initial, referenced value of 0.35 in section 3.2.1.1 of the Methodology. This difference from our referenced coefficient of friction was not surprising, because it was expected that when the robot would turn on sand it would sink. This sinking in turn caused for the motors to exert a higher torque as the robot essentially had to dig itself out of the hole that it dug itself. Figure 54 is a picture of the Smallbot after completing a 90-degree turn. As seen in the Figure 54, the back-left wheel has sunk into the sand after completing a turn.

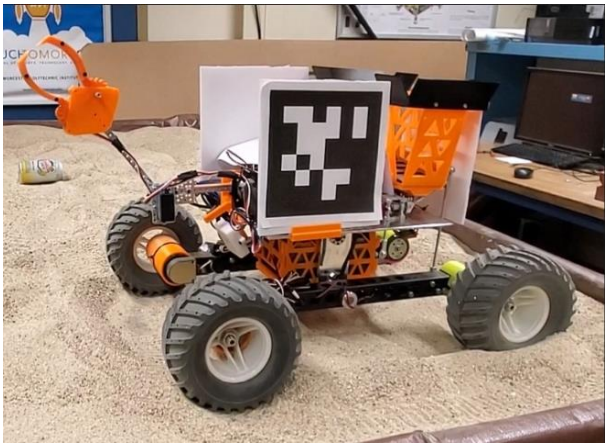


Figure 54: Smallbot After Completing a 90-degree turn

4.2.1.1.2 Gripper Testing

Our team first tested the previous iteration of the gripper before it underwent a total redesign. The previous iteration of the gripper featured two fingers that were manipulated by two gears. Both the driving gear and finger, which were mounted to the servo horn, allowed the driven gear to rotate in the opposite direction. This gripper was tested for collecting both bottles and cans in sand. A key feature of the gripper was that it was able to sift through sand well, however; over time the gears would separate and create too much backlash to the point where the gears would start to skip. This was mainly due to the thin teeth and improper placement of the nut and bolt holding the system together. As seen in the Figure 55 below, the gears became loose due to the continuous grabbing and releasing of either a can or bottle over time. This would cause inconsistencies when grabbing an object and therefore the gripper became unreliable.

Figure 55: Previous Iteration of the Gripper

After testing the previous iteration of the gripper, our team decided that a redesign was necessary for the gripper to reliably pickup both bottles and cans. This redesign of the gripper was an attempt to make the gears thicker, which could potentially help prevent backlash between the teeth. Once the thickness of the gears was increased, it was determined that there were still issues of the gears slipping and causing 10 degrees of backlash. After conducting further testing, it was found that the servo mount was the cause of the backlash between the two points of rotation for the gears due to it not being secure enough.

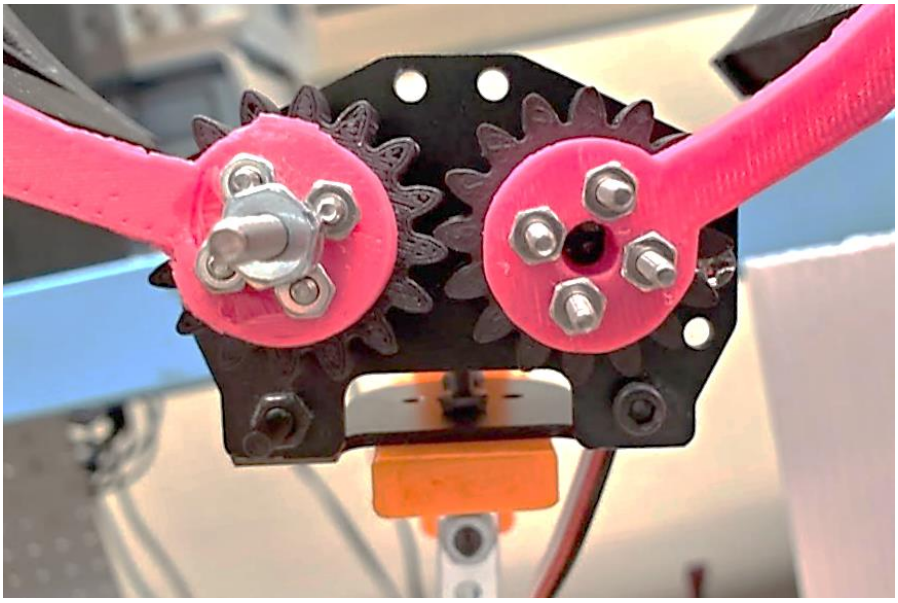


Figure 56: Gears from Redesign of Gripper

Following the first redesign of the gripper, a second redesign was deemed necessary for collecting bottles and cans reliably. This second redesign addressed the issues that were found

from the previous designs, which included the backlash in the gears as well as the lack of support on the servo mount. These issues were fixed in the second redesign. This was achieved by adding an external plate that provided support on both sides of the gears. Also, the diametral pitch and thickness of gears was changed to 12 teeth per inch and 12.7mm. The thicker gears and the extra support allowed for the gripper to consistently collect bottles and cans, as well as crushed cans. This third test was found to be the most successful gripper test since it proved to be reliable and prohibited any sort of backlash or movement between the gears and fingers.

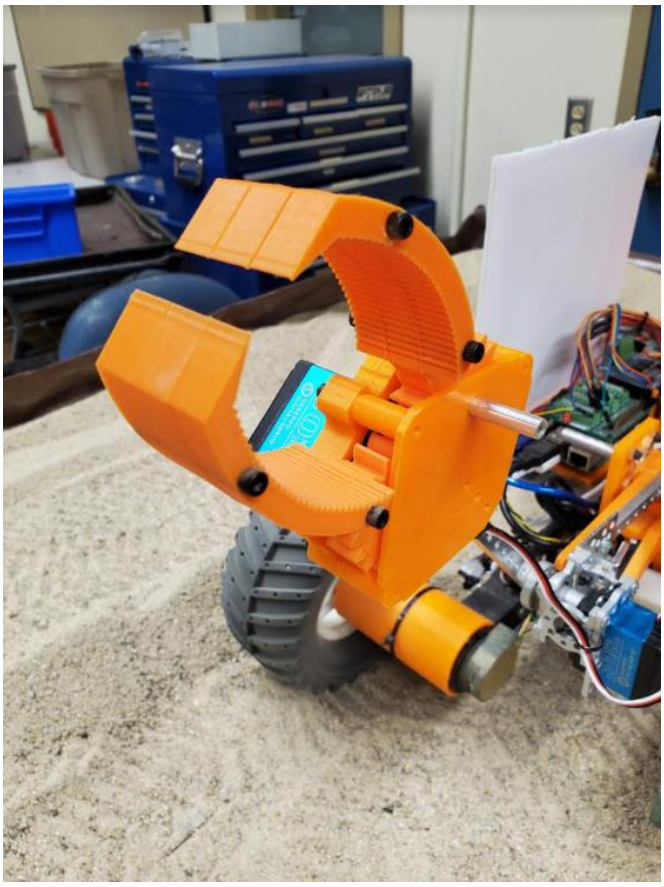


Figure 57: Second Redesign of Gripper

As discussed previously in Section 3.2.1.2, the gripper was fitted with rubber bands to maintain gripping force without power to the servo motor. New 3D-printed standoffs were added to the gripper's fingers and manually tested. This allowed our team to determine how many bands were needed to maintain sufficient gripping force on the bottles and cans. One rubber band doubled back on itself provided ample grip for holding litter in the gripper. The gripping force was further tested with a filled plastic water bottle. This test concluded that the bands were able to maintain a constant gripping force on objects that were heavier than the target litter.

Once the rubber bands were added to the gripper, it was discovered that the servo controller was unable to cut power to the servo. Thus, a new design had to be created to solve the servo motor burnout problem. Slots were added to the inside of the fingers to allow a pin on the servo to rotate edge of the slot to pull the jaws open. This design allows the servo to pull the jaws open to grip trash, while not having to return to position zero, as it can continue traveling through the slot even when the fingers have stopped moving. A version of this design, with the top plate removed can be seen, in Figure 58.

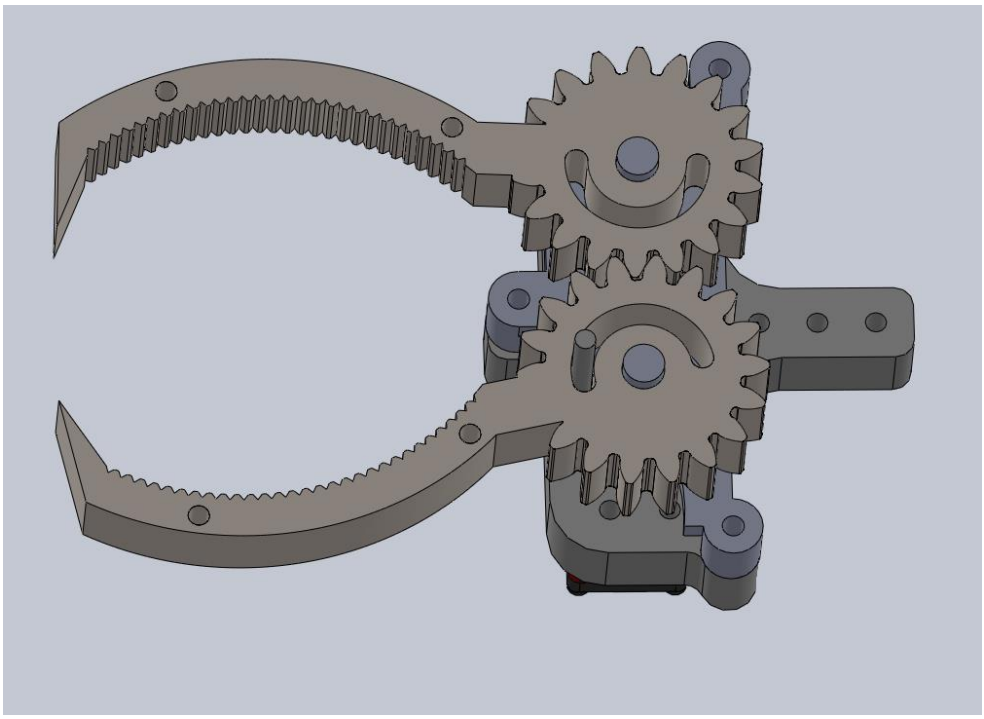


Figure 58 Pin Slot Mechanism on Gripper Fingers

4.2.1.1.3 Two DOF Arm Testing

As stated in section 3.2.1.3, testing was conducted to determine the strength and the correct distance required to collect trash with the 2-DOF arm. Although the arm was consistent at collecting objects and was a great improvement from the previous iteration, additional support was required for the arm to handle deflection from the side. To add additional support to the shoulder part of the arm an additional part was added to support the first link. The calculations for the deflection of the arm can be found in section 3.2.1.3 and 3.2.1.3.1.

The arm was also tested for the maximum lifting weight it could handle. For the scope of the project, the maximum weight out of the three trash options (a 473ml bottle, a 355ml can, and

a crushed can) was the 473ml bottle, which was measured to be 0.454kg. The arm successfully managed to lift all three objects separately, without any issues. The previous iteration of the arm experienced issues when lifting a 0.650kg bottle because of the lower torque servo used at the elbow joint. This servo with a stall torque of 1.96Nm was upgraded to a 2.96Nm servo, which ultimately increased the arm's lifting capacity.

4.2.1.2 Electrical Testing

To fully ensure that the selected electronics were functioning properly, there were a series of tests that occurred throughout the duration of the project. The motor controllers and battery were tested with an ESP32 microcontroller.

4.2.1.2.1 Arduino ESP32 Initial Testing

Initial testing consisted of sending wheel efforts through the GPIO pins of the ESP32. This process allowed for our team to fully understand how the new drivetrain would function in an outdoor environment. The setup for initial ESP32 testing is displayed in Figure 52 Our team drove to a local, freshwater beach in order to carry out the initial tests. The tests required the Smallbot to drive forwards and backwards and perform point/swing turns in both directions on uneven terrain. The tests were successful in showing that the chassis worked as intended and proved that the new drivetrain was superior for beach environments, when comparing it to the old chassis. Furthermore, the new chassis would allow for more space to mount the necessary electrical and mechanical components.

4.2.1.2.2 Motor Controller Testing

The initial testing for the motor controllers that were interfaced with the chassis began with a simple, single channel control board, which was manufactured by Anmbest and was used to modulate the individual wheel efforts of each side of the drivetrain. Since our preliminary testing did not require the motors be able to drive in both directions, a dual channel speed controller was not in immediate need. The purpose of this initial testing was to ensure that both sides of the drivetrain functioned appropriately and did not have any factory defects that could have had comprised any further development of this project.

Once full functionality of both sides of the drivetrain was confirmed, our team then purchased a pair of dual-channel H-Bridge motor drivers that were rated for the motors' voltage and current draw requirements. Each one of these H-Bridges controlled opposite ends of the drivetrain and were able to reliably send the PWM signals needed for any combination of wheel efforts that could be performed by the chassis.

4.2.1.3 Software Testing

To ensure that the software was robust, efficient, and effective enough to meet our team's requirements, various tests were conducted. These tests included Bluetooth, motor encoder, IMU, and object detection testing.

4.2.1.3.1 Bluetooth Testing

Once the functionality of the motor speed controllers was tested and verified, as discussed in section 4.2.1.2.2, our team needed to find a way to conduct driving tests with teleoperated commands. After a series of ideas were presented, Bluetooth communication was settled as a viable, temporary method. Bluetooth helped conduct tests that required teleoperation behaviors, such as driving the chassis outdoors on sand. To achieve this, a script that supported Bluetooth communication was written to run on the ESP32. This program enabled the microprocessor to be seen as a pairable device by other electronics, which in turn allowed our team to pair our smartphones with the ESP32. Furthermore, by using an Android App called "Serial Bluetooth Terminal", our team was then able to send individual characters to the ESP32 via Bluetooth, which could be interpreted as commands to drive forward, backwards, turn, etc. Ultimately, this temporary form to send data to the Smallbot was an effective way to conduct early testing of the drivetrain, outdoors on sand.

4.2.1.3.2 Drive Motor Encoder Testing

During the early stages of the project, testing was conducted with the drive motors encoders. The drive motors included built-in encoders that could potentially assist the localization of the Smallbot relative the Basebot's camera view. However, our team had initial concerns that the encoders would be inaccurate due to wheel slippage caused by the low traction of the sand. To address this concern, our team brought the original drivetrain prototype to a beach at a local pond in Worcester. This test also served as a way to check if there was a reduction in motor RPM when traveling in sand, due to wheel-sinkage or loss of traction. Our

team added an additional 3.6kg to the prototype to test for a possible worst-case scenario in the event that the Smallbot's final weight happened to be over 14.5kg. For this test, the Smallbot drove 1.52m over sandy terrain with varying bumps and valleys and was placed in three different configurations on the beach. These three different configurations include: the Smallbot facing parallel to the shore, the Smallbot facing uphill away from the shore, and the Smallbot facing down towards the shore. The results from this test can be seen below in Table 3.

			Dist in m:	1.52
Sideways	**Time (s)**	**Right Encoder**	**Left Encoder**	**RPM**
Test 1	2.5	2330	2290	65.5
Test 2	2.3	2080	1960	71.2
Uphill				
Test 1	2.3	2190	2330	71.2
Test 2	2.5	1670	1770	65.5
Downhill				
Test 1	2.1	2540	2190	78
Test 2	2.4	1980	1850	68.2
			Avg RPM:	68.3

Table 3: Encoder and RPM testing

As can be seen in the table above, no matter which orientation the Smallbot traveled, there was a significant difference between the left and right encoder values. With the encoder ticks having a maximum difference of 351 ticks over a span of 1.52m, our group decided encoders would not be effective for traveling larger distances as considerable error would accumulate over time. From this test, our team learned that an IMU and an external localization strategy (AprilTags), would be essential for a robust control solution.

4.2.1.3.3 IMU Testing

Testing of the Adafruit BNO055 IMU was conducted. First, our team needed to find which data transmission protocol would be best to obtain the most recent gyroscope values from the IMU. I^2C was the serial communication protocol of choice, as this would ensure that by

sampling the data synchronously with the Pi's master clock, our team would always request the most up-to-date information from the IMU's onboard sensors.

Once our team was able to obtain usable gyroscope values from the IMU, a controller to ensure that the chassis would drive straight needed to be implemented. This involved the integration of a simple proportional controller to the function that wrote the wheel efforts to both sides of the drivetrain. By calculating the difference between the actual heading and the desired heading, our team was able to interpret each wheel effort as a function of the computed delta in terms of a desired wheel effort.

$$\Delta\theta = \theta_{actual} - \theta_{desired}$$

$$Left\ wheel\ effort = wheel\ effort_{desired} - \Delta\theta$$

$$Right\ wheel\ effort = wheel\ effort_{desired} + \Delta\theta$$

Similarly, point turns were performed with the aid of the gyro data obtained from the IMU. By comparing the current heading and the desired heading of the chassis, our team was able calculate the difference in the yaw angle that the drivetrain needed to turn. After implementing a simple check to determine which direction the chassis needed to turn to, either right or left, our team was finally able to successfully perform point turns on sand with an accuracy of ~2 degrees.

4.2.1.3.4 Object Detection Testing

Furthering the work after prototyping the object detection, our team was ready to start the testing phase. The starting point for testing the object detection was to place either a bottle or can in front of the Smallbot camera to test if the model worked as intended. By using a program called “TeamViewer”, our team was able to remotely connect to the Raspberry Pi with a laptop and thus monitor the output without the need for an external monitor. Initially, with TeamViewer up and running, bottles and cans were individually placed in view of the Smallbot camera as seen in Figure 59.

However, as mentioned in the prototyping section for object detection, the model was falsely identifying many different areas of camera video feed as bottles or cans. In order to fix this problem, our team decided to retrain the entire model with a new dataset that was denoted by a collection of ‘not’ bounding boxes. The ‘not’ bounding boxes identified everything that was not a can or bottle. This allowed for the model to strictly identify bottles and cans, which dramatically increased the model’s practical accuracy. An updated image is provided in Figure 59, in which common areas of the frame that were previously falsely identified were labeled with ‘not’ bounding boxes.

Figure 59: Update Image with 'not' Bounding Boxes

As seen in Figure 60, the TFLite model was functioning as intended with a greater accuracy than that of what Axon had originally predicted.

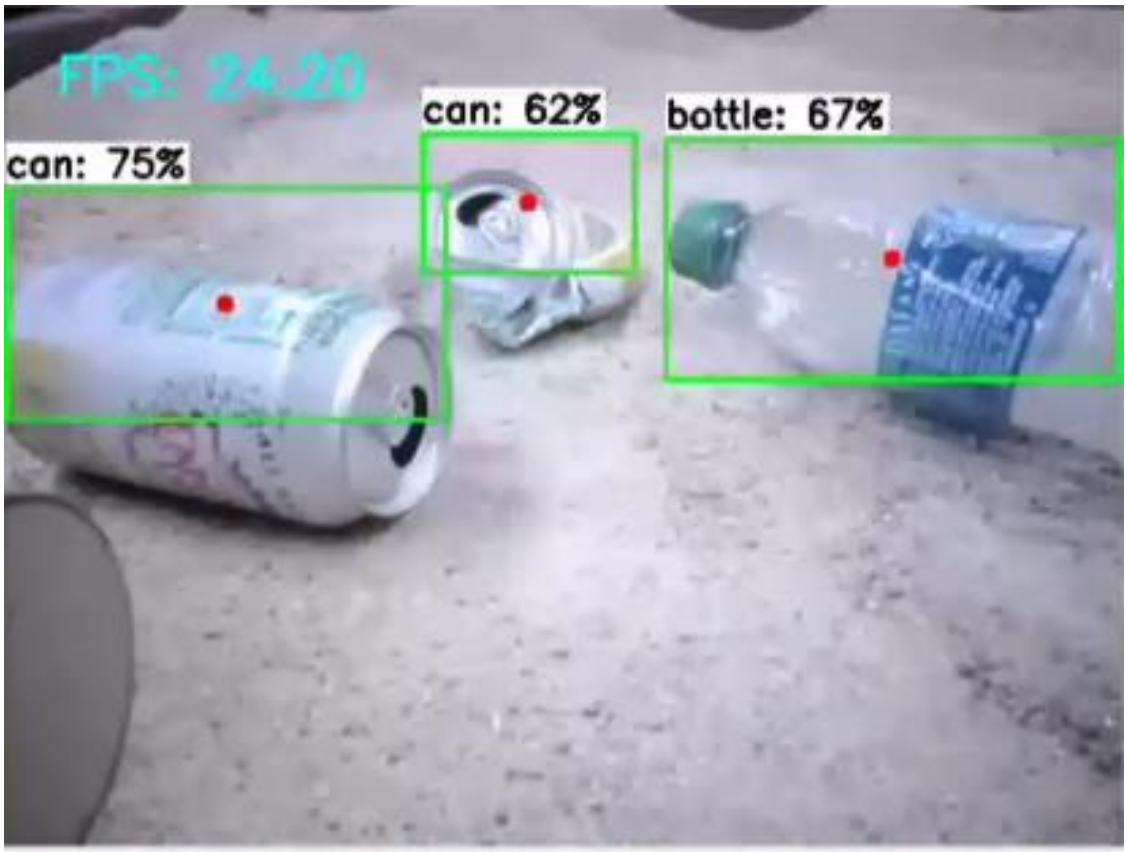


Figure 60: Model Running on Coral Edge TPU

Once our team verified that the model ran as intended on the Edge TPU, additional testing was conducted to determine how the model performed with the robot in motion while actively searching for objects to pick up. It is important to note that the Smallbot's speed played an important role, since the speed at which the camera was moving affected the accuracy and detection time of the model; hence, the speed at which the Smallbot would traverse its path was set to 20% of its maximum speed.

Our team found the optimal driving speed by testing different values which were above and below half of the maximum speed of the motors, and the results were as follow: the

difference between 20%, and any speed lower was negligible. However, there was a significant difference between 20% speed and above, as driving at higher speeds prevented the robot from reliably detecting litter on its path. Based off this series of tests, our team concluded that 20% of the maximum speed would allow the inference algorithm to have enough time to reliably detect objects while also maintaining a reasonable speed.

Once the optimal detection speed was determined, our team needed to perform tests to find the optimal position for the Smallbot to collect litter. This was accomplished by having the arm follow a set of pre-defined motions that would allow it to reliably pick up the trash that was in front of it. Therefore, the only calculation necessary was the time it would take for the Smallbot to drive forwards or backwards, depending on how far or close it was to the object of interest. To test this functionality, we calculated the equations needed to find the centroid of an object in the camera frame. The y-component of the centroid was then used to determine the amount of time that the Smallbot would need to drive to position itself in front of the object. The calculation to find the centroid of an object can be seen below:

$$centroid(x, y) = \frac{x_{min} + x_{max}}{2}, \frac{y_{min} + y_{max}}{2}$$

Furthermore, the centroid was used in a larger formula which would first find the angle at which the chassis needed to turn to face the object. The distance between the camera and the object was then detected. The Smallbot would finally either move backwards or forwards, based on an ideal distance value, with an offset applied, to position itself for successful collection.

After collection, the Smallbot would go return to its original position before it detected the object and continue to follow its path. The script for this part of the code is provided in the Appendix.

4.2.2 Basebot Testing

Testing was required to ensure that the Basebot was functioning properly and to meet the project requirements. The testing of the Basebot took place in our team's laboratory where the Basebot was placed near the table that the Smallbot operated on. The purpose of these tests was to verify that the project requirements were met for the Basebot prototypes.

4.2.2.1 Camera Stand Testing

To test the Basebot's ability to track AprilTags, a simple stand was built for the camera to be mounted onto. This stand used REV robotics 15mm extrusion to create the base and to allow the camera height to be adjusted. The camera could easily be adjusted by moving the L-bracket mount for the camera up or down, within the slotted extrusion. From testing, the optimal height was found to be about 0.914m from the top of the sand table where the Smallbot operates. This height difference was found to help with distinguishing a wide-angle view of the entire table that the Smallbot would be moving about on. This simple camera stand was used throughout the project and was helpful for maintaining a consistent height and position for the Basebot to observe the Smallbot's AprilTags.

Figure 61: Camera Stand Facing the Sand Table

4.2.2.3 Software Testing

In a similar manner to how the Smallbot's was tested, our team needed to ensure that the Basebot's codebase was robust, efficient, and effective enough to meet the set requirements. Therefore, various tests were conducted. These tests included AprilTag recognition testing and testing of the state machine, based on the camera's video feed.

4.2.2.3.1 AprilTag Testing

The early implementation of the AprilTag recognition software required our team to use the AprilTag package developed by Electrical Engineering and Computer Science department from the University of Michigan (Olson, 2010). Once the software was able to run on our local machines, a series of tests were performed to guarantee that the printed tags could be detected under a wide array of different conditions. Initially, our team needed to determine which type of AprilTag families the imported software was able to detect. After some research on the C++ implementation of the package, it was found that the detector was able to recognize the following series of AprilTag families: 16h5, 25h7, 25h9, and 36h11. Next, our team needed to test the maximum distance at which an AprilTag could be recognized —this number resulted to be approximately 5m, which meant that the detection range was not going to be an issue for the purposes of this iteration of the project. Finally, a wide variety of lighting conditions and angles were applied to the camera, to ensure that the AprilTag detection was reliable enough to be deployed during a real test run. Ultimately, the AprilTag package was able to be used reliably and proved to be robust, as it was able to effectively track multiple AprilTags even when external factors were applied.

4.2.2.3.2 State Machine Testing

The initial prototype of the state machine was developed according to the actions that the Smallbot needed to perform with respect to the Basebot's camera view. This was done by positioning the Smallbot's onboard AprilTags on different spots within the Basebot's camera's field of view. Each time after the AprilTags were displaced, our team needed to ensure that the

OpenCV coordinates were not only being extracted appropriately, but also that they were being attributed to their corresponding state. For instance, if an AprilTag was seen on the left side of the OpenCV coordinate frame, then the state machine needed to update the Smallbot's action to tell it to perform a point turn to the right to ensure that it would stay within the camera's field of view. This idea was applied to every state in the path, to verify that a state was constantly being outputted regardless of where the AprilTags were located in space.

4.2.3 Integration Testing

Once every software-related aspect of both the Smallbot and the Basebot was individually tested, our team had to ensure that all of these components worked in unison after both systems were combined and began communicating autonomously via TCP Communication.

4.2.3.1 TCP Communication Between the Smallbot and the Basebot Testing

As seen in Figure 62, the Basebot was tasked with sending field commands to the Smallbot via TCP. As previously discussed in section 3.2.3.4, the ESP32 acted as the wireless AP between both systems. Upon deployment of this implementation, it was found that there were some initial issues due to blocking code that stemmed from while loops being used to perform point turns on the Smallbot's end. These segments of blocking code caused a temporary halt on the Basebot's AprilTag detection's camera feed displayed on OpenCV. The root of this issue was traced down to be caused by the fact that the execution of the while loops delayed an echo command that needed to be sent back to Basebot, as a confirmation that the information was

being received by the Smallbot adequately. Nonetheless, this issue was solved by creating a new function that would perform point turns without the need of any loops. This new addition to the Smallbot's codebase fixed the issue and allowed for reliable back-and-forth communication between both systems.

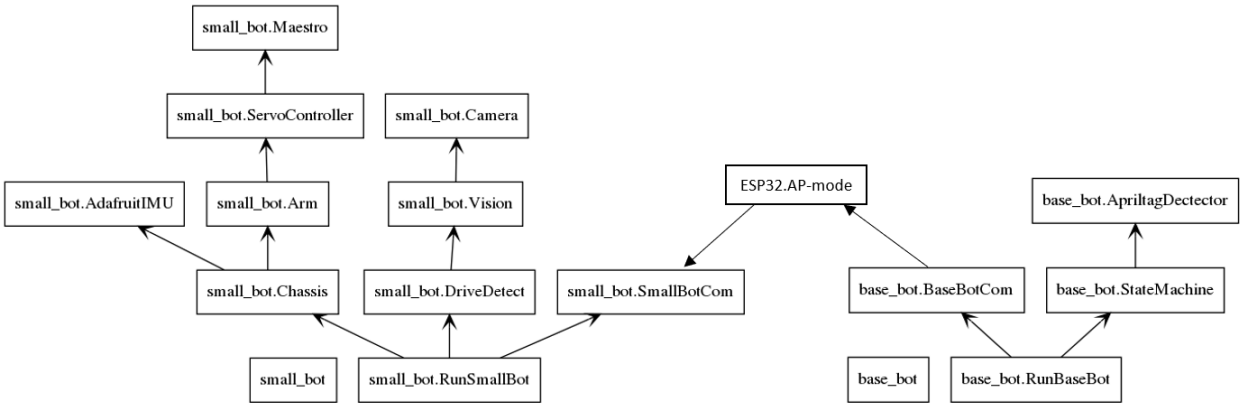


Figure 62: Package Diagram of the Entire System

4.2.3.2 Combined State Machine Testing

The final testing phases of the state machine involved running a series of tests runs to ensure that the information being sent from the Basebot adequately allowed the Smallbot to follow its path, while having it simultaneously scan for trash to pick up. This testing was accomplished by replicating scenarios where the Smallbot would detect an object in random locations at any given point throughout the path. After conducting numerous tests runs to ensure that this functionality did not interfere with the Basebot's commands, it was discerned that the system worked in a reliable manner and accomplished the required tasks.

Section 5: Conclusion and Recommendations

5.1 Reflection of our Project

This section provides more insight into the final product of the Smallbot and the Basebot. These reflections are broken down between mechanical, electrical, and software components.

5.1.1 Mechanical Reflection

Overall, the project's mechanical requirements were met and exceeded. The Smallbot's drivetrain successfully managed to drive on sand during early tests at a beach and later in the project when testing was conducted in the lab. The Smallbot was able to reach a top speed of $0.62\frac{m}{s}$ on sand. Although this speed had to be reduced to 20% for the object detection to function properly, the Smallbot has the capability to drive faster than $0.1\frac{m}{s}$, or around walking speed. Also, the Smallbot can drive without the drivetrain jamming or the robot getting stuck in the sand. Furthermore, the rocker-bogie system proved to be efficient and capable for maneuvering on varying beach terrain.

The gripper and arm proved to meet the requirements for collecting trash objects reliably from the beach surface. The gripper was designed and tested to be able to capture a full 473ml bottle, 355ml can, or even a crushed can. In addition, strain placed on the servo motor and chance of servo burnout can be reduced with the use of rubber bands to grip the cans and bottles. Also, the arm was capable of lifting and placing a full 473ml bottle into the bucket. The arm was reinforced to lessen the chance of breaking or deflecting if it runs into obstacles. The goals for

the bucket were exceeded since the final design of the bucket could store up to three 473ml bottles. The bucket was tested to hold all three bottles securely, and efficiently managed to dump all trash objects at the end of the Smallbot's path.

5.1.2 Electrical Reflection

The requirements for the electrical portion were met and exceeded. Starting a project of this caliber called for a solid electrical foundation, which was achieved through individual component testing, battery calculations, and the creation of a detailed, wiring schematic. Developing the electrical schematics and improving cable management were important first steps in developing this solid electrical foundation. Furthermore, our team was able to easily make additions to the Smallbot with the foundation that we had built.

Completing the electrical work in a timely manner allowed for our team to focus our efforts in other areas of the project which required more effort. All of the documentation that our team had created for the electrical work on the Smallbot will give the following team a better experience continuing this project.

5.1.3 Software Reflection

The software requirements for the entire system were exceeded. The Smallbot was able to use the gyroscope's data from the IMU, to not only perform accurate point turns on sand, but to also drive straight while scanning for trash. Furthermore, the dataset of detectable objects was successfully expanded from only cans to both bottles and cans. The new optimized TFLite model

for the Google Coral Edge TPU proved to be of great use in increasing the system's overall performance.

The implementation of TCP communication between both the Smallbot and the Basebot was a considerable improvement from the last iteration of the project, as this feature can be expanded to work with multiple Smallbot systems. Additionally, the use of AprilTags to perform visual localization and field commands allowed for both robots to work in unison. This also expanded the functionality of the system to a point where it can clean an entire area seen by the Basebot's camera. Finally, these requirements were further exceeded by having the Smallbot return to its starting position upon finalizing the clean-up of an area.

5.2 Final Conclusion

Currently, the multi-robot system is capable of detecting, navigating, and collecting trash from a beach. The Basebot can track and direct a single Smallbot that is navigating within the Basebot's camera view. With TCP communication, the Basebot can send instructions to the Smallbot while it is traveling on the shore. The Smallbot can drive efficiently and effectively on varying sand terrain without getting stuck or clogging the drive motors. Furthermore, the Smallbot can detect, maneuver to, and collect a variety of trash objects that are typically found on beaches. Some of the trash objects include empty or filled bottles and cans, as well as crushed cans. The second iteration of the project was another step towards the conceptual idea of having a swarm of beach clean robots.

Future teams of this project should look towards gradually adding additional Smallbots to the system, collecting different sizes of trash objects (i.e., liter sized bottles or beer bottles), and waterproof all of the Smallbot's electrical components. The Basebot should be able to move parallel to the Smallbot, and to the next section of the beach while still giving instructions to the Smallbots. After these improvements, the system will have a greater chance of accomplishing the overall goal for this project.

Addendum

Overview of Current Basebot:

Figure 63: Current Basebot

The current Basebot is an average garden cart, as seen above in Figure 63. This garden cart has not been utilized in any way, added on to, or altered from its initial form throughout the previous iteration of this project. Currently, the goal is to create a detailed design plan in which the following iteration of this project can rebuild the Basebot, in order to make it drive reliably and efficiently on sand.

Requirements

To allow the BaseBot to be fully drivable, the following objectives must be achievable through this design:

- The Basebot design must utilize the given Basebot, which is a garden cart.

- The Basebot must drive on sand.
- The Basebot must run at or above 0.25m/s.

Mechanical Design:

In order to fulfill the requirements for the Basebot's drive system, it was necessary to follow the engineering process and create an overall design for a drivable Basebot. This was done by taking measurements of the current garden cart, creating a model of it in SolidWorks, completing various calculations, and modifying the model to have a more effective and efficient design. Multiple aspects of the garden cart had to be reworked, including the steering mechanism, the addition of motors, and the addition of batteries, as well as other various electronics to power this autonomous vehicle.

Steering Mechanism:

Figure 64: Current Basebot Steering Mechanism View 1

Figure 65: Current Basebot Steering Mechanism View 2

Due to the difficulty of attaching drive motors to the original garden cart, as seen above in both Figure 64 and Figure 65, a new driving system was designed. First, it was decided that it was necessary for the front portion of the steering, as depicted in Figure 64, to be removed from the Basebot; this is a simple task that can be done with a wrench and nut driver. In addition, this removes the handle of the garden cart, which decreases the amount of interference it could have caused during operation. Once this was analyzed, it was apparent that the front part of the steering system would have to be completely redesigned. Furthermore, the back portion of the drivetrain would need to be powered by two motors and would nearly stay the same as the initial product, aside from slight changes to mount the motors on a new supporting beam and altering the axle. In terms of the front of the system, the wheels would not be powered themselves, however; a rack and pinion, as seen below in Figure 66, would be powered by a Pololu 37D Metal Gearmotor with a gear ratio of 131:1. This allows for easy steering, as well as a clean-cut

design, as seen below in Figure 67. In addition, the large wheels that the garden cart is equipped with allow for the Basebot to go over sand dunes and through dips in the surface with ease. This design was initially inspired by how cars drive and was redone in such a way that would be ideal for turning in sand.

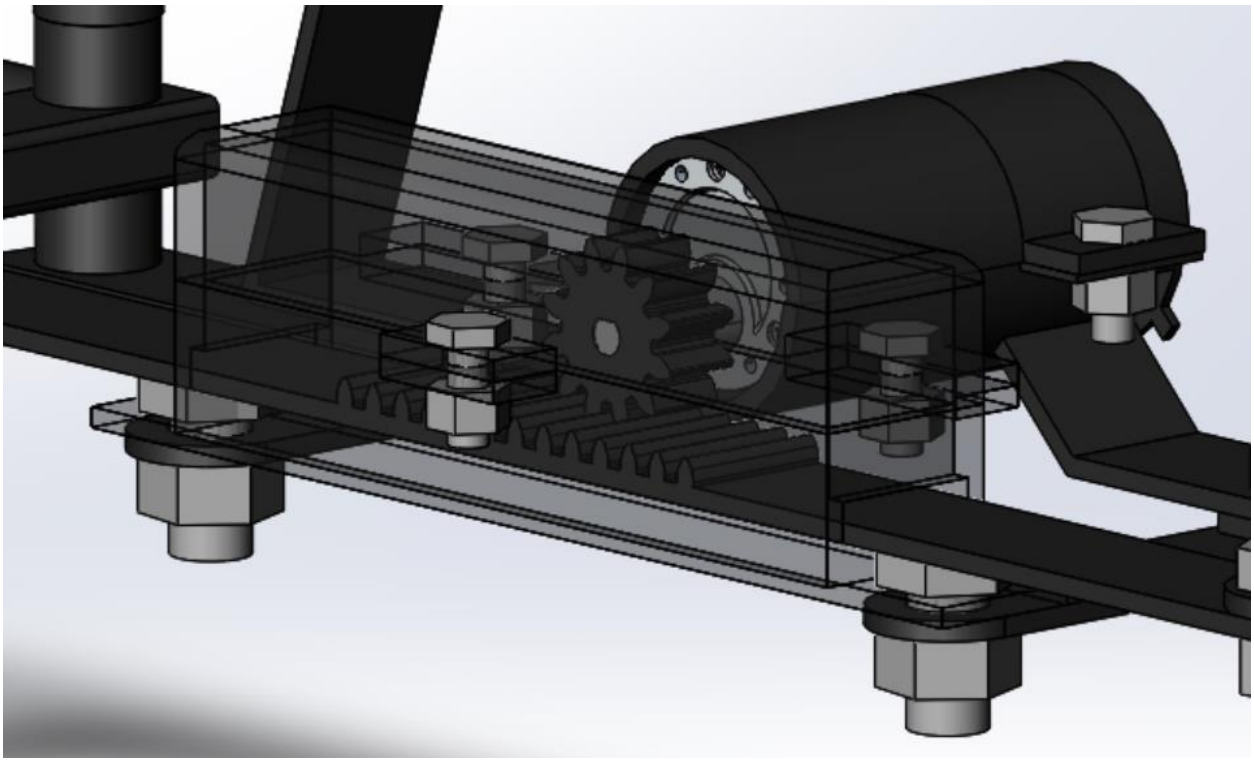


Figure 66: Rack and Pinion

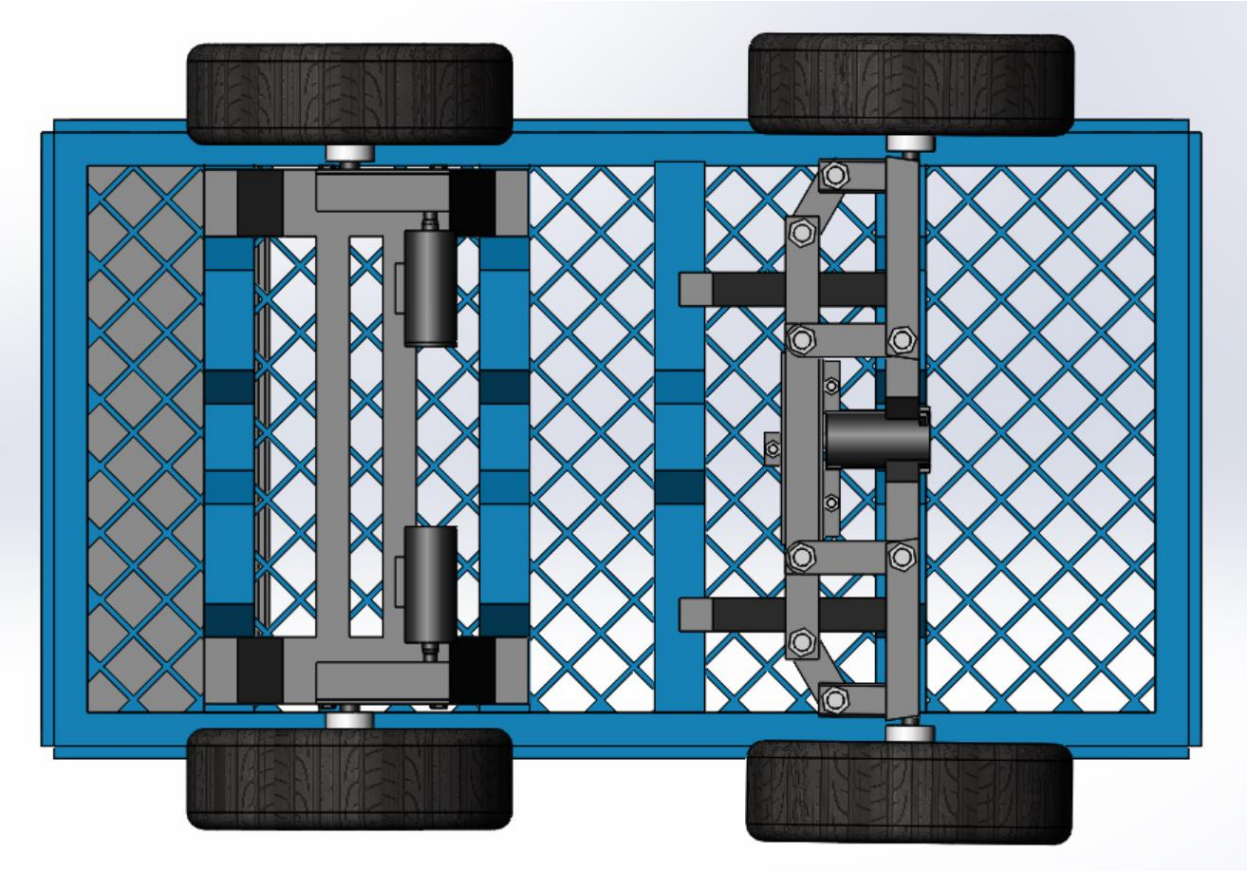


Figure 67: Steering Design and Bottom View of Basebot Final Design

Motor Selection:

In order to power both of the back wheels, it was necessary to choose a motor that would be reliable, efficient, and that could fulfill all of the requirements for the Basebot portion of this project. This was also completed in detail for the rack and pinion system.

Back Wheel Motor Calculations

To ensure that the chosen motor would function as intended, calculations were completed. These calculations were computed with the assumption that the Basebot would be

full of uncrushed, empty, 12oz aluminum soda cans. On average, an empty 12oz aluminum soda can weighs 0.0136kg, has a diameter of 0.0524m, and a height of 0.121m. In addition, it was also assumed that the Basebot would be completing a 45 degree turn for the overall robot calculations, as seen in the free body diagram depicted through Figure 68. For these calculations, the empty garden cart was found to be 16.6kg. Therefore, after a series of calculations were conducted, the total maximum mass of the Basebot was found to be 21.6kg. Due to the requirement for the Basebot's speed being ½ the average walking speed, as well as ½ of the Smallbot's speed requirement, it was found that the necessary minimum speed of the Basebot is 19.3 revolutions per minute. It was also assumed that the friction factor of clean sand is 0.3, this was also tested for, as described later in the report (Fine Software. (n.d.)).

As seen below through a free body diagram of one of the front, undriven wheels in Figure 69, there are effectively no forces in either the x-direction or the y-direction. This is due to the wheels not being directly driven by motors and therefore, there are no notable forces within that singular system. However, there would be a rolling resistance due to kinetic friction. This is computed to be 63.6N, as seen below.

Lastly, through the free body diagram depicted in Figure 70, of one of the back, driven, wheels and its moment calculations, the necessary torque for the motor was found. This torque was found to be 7.89Nm. In order to find the needed power for a chosen motor, calculations were also performed. The necessary power for both of the back, driven motors was found to be 15.9W. Therefore, the Pololu 37D Metal Gearmotor utilizes less than 80% of the maximum power potential and falls within the 85% of maximum efficiency curve for both driving the back wheels

and turning the rack and pinion mechanism. All of the free body diagrams and calculations for each of the steps previously mentioned can be found below.

Volume:

$$V_{Garden\ Cart} = L * W * H$$

$$V_{Garden\ Cart} = 0.469m * 0.851m * 0.241m$$

$$V_{Garden\ Cart} = 0.0961m^3$$

$$V_{Empty\ Soda\ Can} = \pi * r^2 * H$$

$$V_{Empty\ Soda\ Can} = \pi * (0.0262m)^2 * 0.121m$$

$$V_{Empty\ Soda\ Can} = 0.000261m^3$$

Maximum Soda Cans in the Garden Cart:

$$Max_{Soda\ Cans\ in\ Garden\ Cart} = (\frac{V_{Garden\ Cart}}{V_{Empty\ Soda\ Can}})$$

$$Max_{Soda\ Cans\ in\ Garden\ Cart} = (\frac{0.0961m^3}{0.000261m^3})$$

$$Max_{Soda\ Cans\ in\ Garden\ Cart} = 368\ cans$$

Total Mass:

$$m_{Empty\ Soda\ Can} = 0.0136kg$$

$$m_{Max\ Soda\ Cans\ in\ Garden\ Cart} = Max_{Soda\ Cans\ in\ Garden\ Cart} *\ m_{Empty\ Soda\ Can}$$

$$m_{Max\ Soda\ Cans\ in\ Garden\ Cart} = 368\ cans * 0.0136kg$$

$$m_{Max\ Soda\ Cans\ in\ Garden\ Cart} = 5.00kg$$

$$m_{Garden\ Cart} = 16.6kg$$

$$m_{Total} = m_{Garden\ Cart} + m_{Max\ Soda\ Cans\ in\ Garden\ Cart}$$

$$m_{Total} = 16.6kg + 5.00kg$$

$$m_{Total} = 21.6kg$$

Required Rotations per Minute:

$$Speed = 0.25\frac{m}{s} = 15\frac{m}{min}$$

$$d_{Wheel} = 0.248m$$

$$C_{Wheel} = d_{Wheel} * \pi$$

$$C_{Wheel} = 0.248m * \pi$$

$$C_{Wheel} = 0.779m$$

$$RPM = \frac{Speed}{C_{Wheel}}$$

$$RPM = \frac{15\frac{m}{min}}{0.770m}$$

$$RPM = 19.3\frac{rev}{min}$$

Forces:

$$\mathrm{F_g} = m_{Total} * -\mathrm{g}$$

$$\mathrm{F_g} = 21.6kg * -9.81\ \frac{m}{s^2}$$

$$\mathrm{F_g} = -212\mathrm{N}$$

$$\mathrm{F_N} = -\mathrm{F_g}$$

$$\mathrm{F_N} = 212\mathrm{N}$$

$$\mu_{Clean\ Sand} = 0.3$$

$$\mathrm{F_f} = \mathrm{F_N} * \mu_{Clean\ Sand}$$

$$\mathrm{F_f} = 212\mathrm{N} * 0.3$$

$$\mathrm{F_f} = 63.6\mathrm{N}$$

$$\theta = 45^{\circ}$$

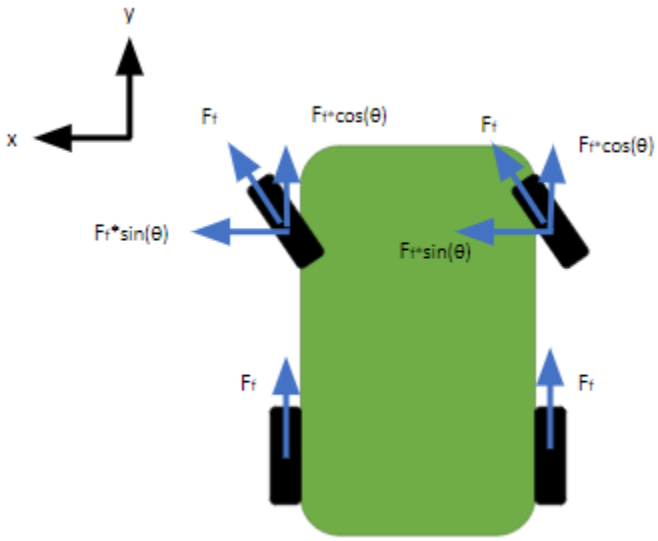


Figure 68: Whole Robot (Top View) FBD

Top View Calculations:

$$\Sigma F_x = 0 = 2 * F_f \cos(\theta)$$

$$\Sigma F_x = 0 = 2 * 63.6 \cos(45^{\circ})$$

$$\Sigma F_x = 0N$$

$$\Sigma \mathrm{F_y} = 0 = \ 2 * -\mathrm{F_f} \cos(\theta) + 2 * \mathrm{F_f}$$

$$\Sigma F_y \ = 0 = 2 * 63.6N * \cos(45^{\circ}) + 2 * 63.6N$$

$$\Sigma \mathrm{F_y} = 0N$$

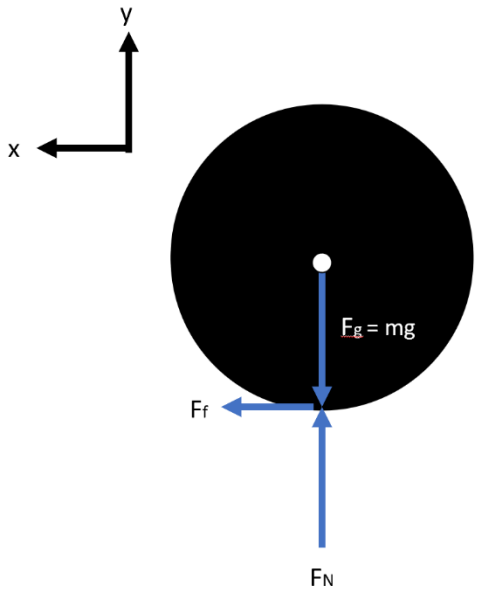


Figure 69: Front (Undriven) Wheel FBD

Front Wheel Calculations:

$$\Sigma F_x = 0 = F_f$$

$$\Sigma F_x = 0N$$

$$\Sigma F_y = 0 = F_N - F_g$$

$$\Sigma F_y = 0N$$

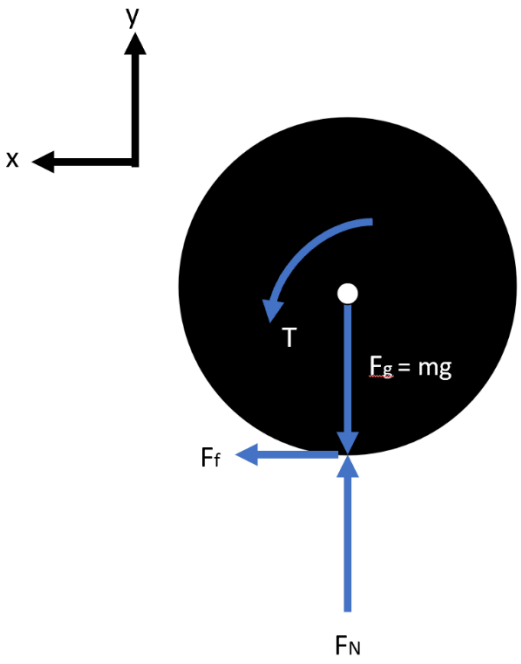


Figure 70: Back (Driven) Wheel FBD

Back Wheel Calculations:

$$\Sigma F_x = 0 = F_f$$

$$\Sigma F_x = 0N$$

$$\Sigma F_y = 0 = F_N - F_g$$

$$\Sigma F_y = 0N$$

$$\Sigma M_{axle} = 0 = \tau - r_{Wheel} * F_f$$

$$\Sigma M_{axle} = 0 = \tau - 0.124m * 63.6N$$

$$\tau_{Both\ Wheels} = 7.89\text{Nm}$$

$$\tau_{One\ Wheel} = 3.94\text{Nm}$$

Power Calculations:

$$Power = (\frac{(\tau * \text{RPM})}{9550})$$

$$Power = \frac{\left(7.89\text{Nm} * 19.3\frac{rev}{min}\right)}{9550}$$

$$Power = 0.0159kW = 15.9W$$

$$Power_{One\ Wheel} = 0.00796kW = 7.96W$$

Back Wheel Gear Train:

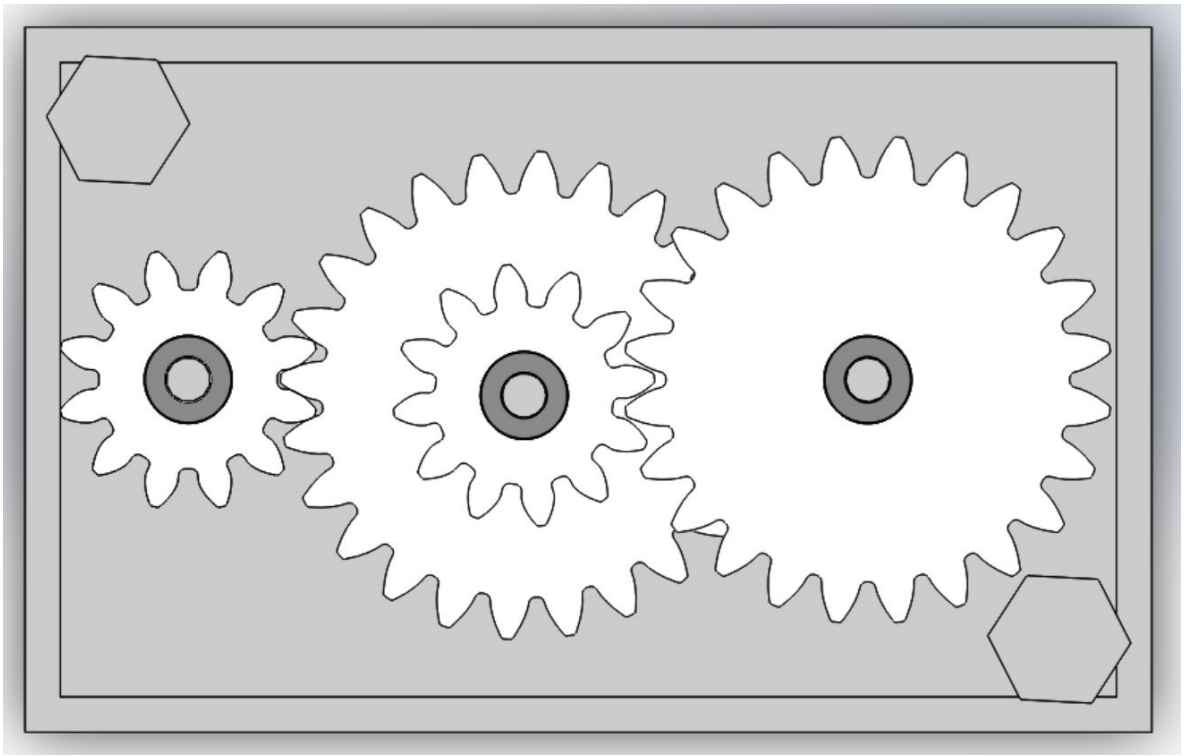


Figure 71: Back Wheel Gear Train

Due to the lower RPM, higher torque, and power requirements of the system being lower than 85% of the maximum power potential, it became apparent that the back wheel driven system would have to be significantly geared down for a motor to function within the Basebot system efficiently and reliably. The calculated gear ratio for this two-stage gear train was 3.04:1, as seen through the calculations below. This was achieved by first locating where the power is equal to 7.96W on the motor graph, which was 84% of the maximum power potential. At 7.96W, the torque was calculated to be 1.30Nm utilizing the motor graphs. This was then utilized with the desired torque of 3.94Nm to find the gear ratio. Next, the gear ratio was checked with the speed of 58RPM calculated at 7.96W utilizing the motor graph, in order to ensure that the gear

ratio was correct. This calculation was performed, as seen below, and it was equal to the necessary RPM for the system. Lastly, the gear train was calculated with an assumed 90% efficiency factor. With such, it was calculated that two 24 tooth gears and two 12 tooth gears were necessary for this two-stage gear train to reach the gear ratio of 3.2:1. The calculated gear ratio of 3.2:1, which is slightly slower due to gear limitations, but has more torque than the desired gear ratio of 3.04:1 would be. Thus, the system will be more accurate and precise.

The overall two-stage gear train design and housing can be seen in Figure 71 above. Each motor required a gear train and housing, and therefore, one was mounted on each side of the Basebot, as seen in Figure 81. Aside from this being a necessary feature for the motors to function, it also added stability to the back wheels by rigidly fixing the axles to the gear train housing. The gear train was rigidly attached at two mounting locations using both steel bolts and lock nuts. In addition, the input torque applied to by the gear train was computed to be 1.30Nm utilizing the gear ratio equation with 58RPM from the motor graphs, the actual rotational velocity and calculations at 3.94Nm, as seen in Back Wheel Calculations, and 19.3RPM as determined as the rotational velocity above. The power required to operate the gear box was also computed. This was done through utilizing the equation that converts from Newton-meters and RPM, divided by a unitless 9550, to power in kilowatts (Binsfield Engineering Inc. (n.d.)).

Calculating Gear Ratio:

$$\tau_{Desired} = 3.94Nm$$

$$\tau_{AtMaxPower} = 1.30Nm$$

$$Gear\ Ratio = \frac{\tau_{Desired}}{\tau_{AtMaxPower}}$$

$$Gear\ Ratio = \frac{3.94Nm}{1.30Nm}$$

$$Gear\ Ratio = 3.04:1$$

Checking Gear Ratio with Speed:

$$Gear\ Ratio = \frac{RPM_{Desired}}{RPM_{AtMaxPower}}$$

$$3.04:1 = \frac{58RPM}{RPM_{AtMaxPower}}$$

$$RPM_{AtMaxPower} = 19.3RPM$$

Gear Sizes:

$$Gear\ Ratio = \left(\frac{\omega_1}{\omega_2}\right) = \left(\frac{n_1}{n_2}\right) = \left(\frac{d_2}{d_1}\right) = \left(\frac{T_2}{T_1}\right) * \eta_{sys}$$

$$Gear\ Ratio = \left(\frac{T_2}{T_1}\right) * \eta_{sys}$$

$$Gear\ Ratio = \left(\left(\frac{24T}{12T}\right) * (0.90)\right) * \left(\left(\frac{24T}{12T}\right) * (0.90)\right)$$

$$Gear\ Ratio = 3.2:1$$

Torque:

$$\tau_{Desired} = 3.94Nm$$

$$\frac{RPM_{Actual}}{RPM_{Desired}} = \frac{\tau_{Desired}}{\tau_{Actual}}$$

$$\frac{58RPM}{19.3RPM} = \frac{3.94Nm}{\tau_{Actual}}$$

$$\tau_{Actual} = 1.31Nm$$

Power:

$$Power = \frac{RPM_{Actual} * \tau_{Actual}}{9550}$$

$$Power = \frac{58RPM * 1.30Nm}{9550}$$

$$Power = 0.00796 = 7.96W$$

Rack and Pinion System Motor Calculations:

To ensure that the rack and pinion steering system was properly motorized, a series of calculations were completed. The mass of the Basebot was 21.6kg, as previously calculated. In addition, the mass of the Basebot's front corner was calculated to be 4.86kg, and the mass of the rear corner of the Basebot was computed to be 5.94kg. The force of friction was then found to be 14.3N, utilizing gravity, mu, and the mass of the Basebot's front corner. Using both this found force of friction and measured distance to the king pin, as seen in Figure 72, the frictional torque was computed to be 1.77Nm. Furthermore, the force of the rack was found to be 26.1N by multiplying the force of the control arm by two. Lastly, the torque at the rack was found to be

0.726Nm, as seen through the free body diagrams in Figure 73 and Figure 74, as well as the computations below.

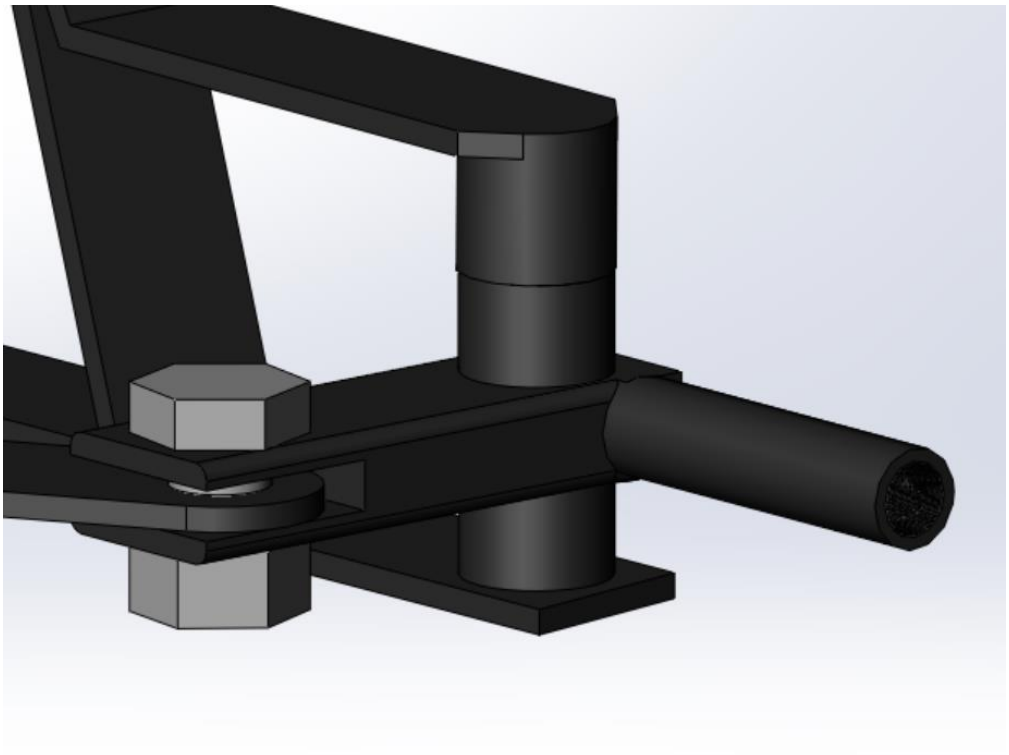


Figure 72: Basebot's Kingpin

Masses:

$$m_{Total} = 21.6kg$$

$$m_{Front} = \%_{Weight\ on\ Front\ Wheels} * m_{Total}$$

$$m_{Front} = 0.45 * 21.6kg$$

$$m_{Front} = 9.72kg$$

$$m_{Front\ Corner} = \frac{9.72kg}{2}$$

$$m_{Front\ Corner} = 4.86kg$$

$$m_{Rear} = 11.9kg$$

$$m_{Rear\ Corner} = \frac{m_{Rear}}{2}$$

$$m_{Rear\ Corner} = \frac{11.9kg}{2}$$

$$m_{Rear\ Corner} = 5.94kg$$

Force of Friction:

$$\mu = 0.3$$

$$g = 9.81\frac{m}{s^2}$$

$$F_{Friction} = m_{Front\ Corner} * g * \mu$$

$$F_{Friction} = 4.86kg * 9.81\frac{m}{s^2} * 0.3$$

$$F_{Friction} = 14.3N$$

Torque at Pinion:

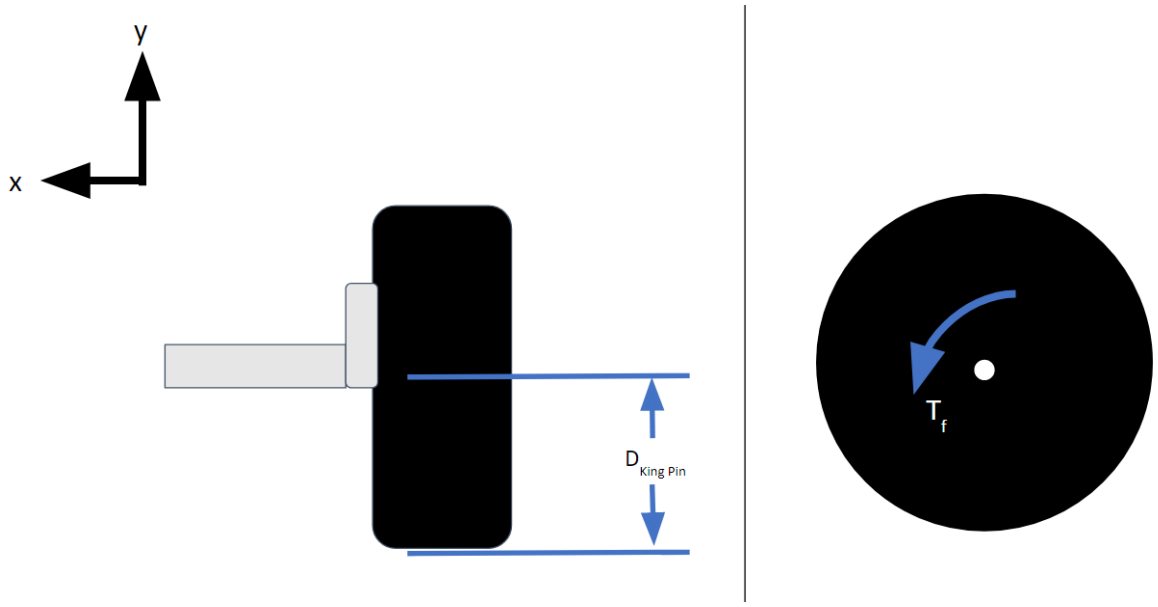


Figure 73: FBD of Frictional Torque

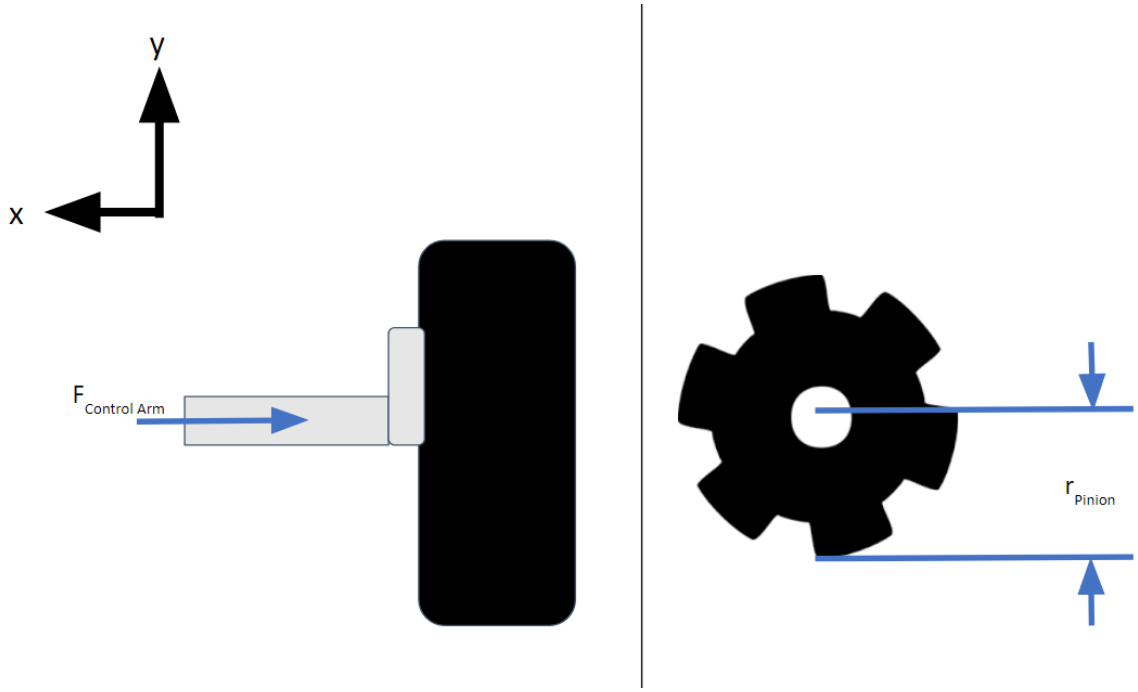


Figure 74: FBD of Pinion Torque

$$D_{King\ Pin} = 124mm = 0.124m$$

$$\tau_{Friction} = F_{Friction} * D_{King\ Pin}$$

$$\tau_{Friction} = 14.3N * 0.124m$$

$$\tau_{Friction} = 1.77Nm$$

$$F_{Control\ Arm} = 14.3N$$

$$F_{Rack} = F_{Control\ Arm} * 2$$

$$F_{Rack} = 14.3N * 2$$

$$F_{Rack} = 28.6N$$

$$r_{Pinion} = 25.4mm = 0.0254m$$

$$\tau_{Pinion} = F_{Rack} * r_{Pinion}$$

$$\tau_{Pinion} = 28.6N * 0.0254m$$

$$\tau_{Pinion} = 0.726Nm$$

Motor Selection:

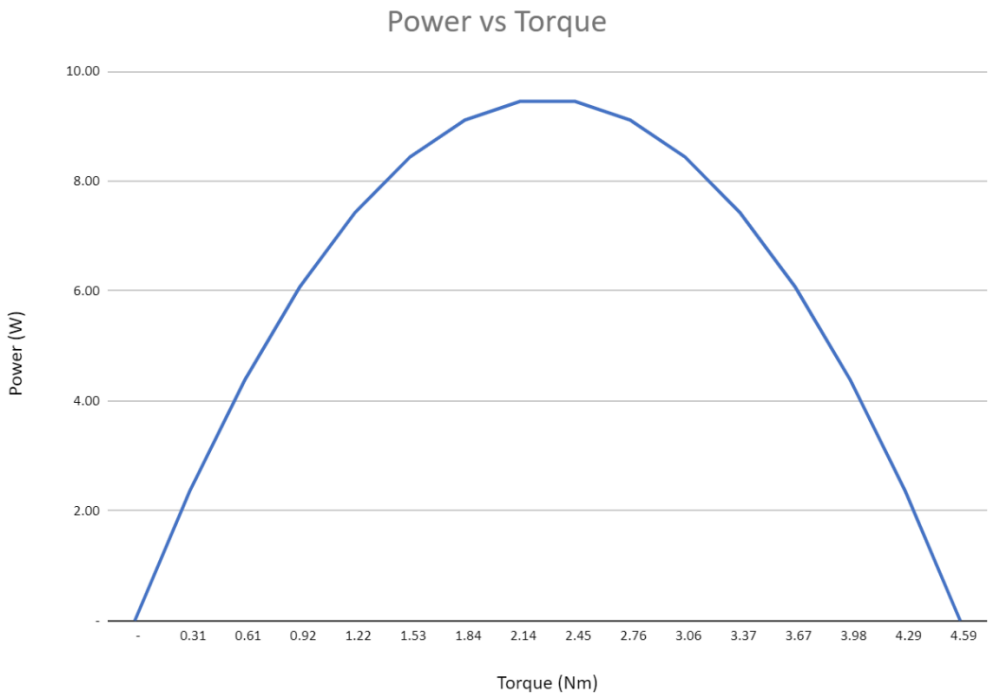


Figure 75: Power vs. Torque Graph for Motor

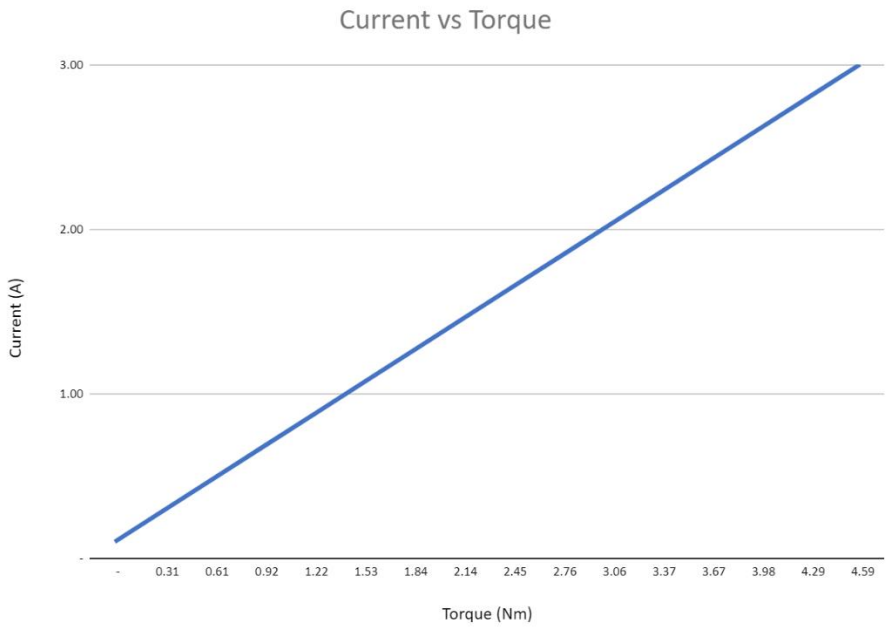


Figure 76: Current vs. Torque Graph for Motor

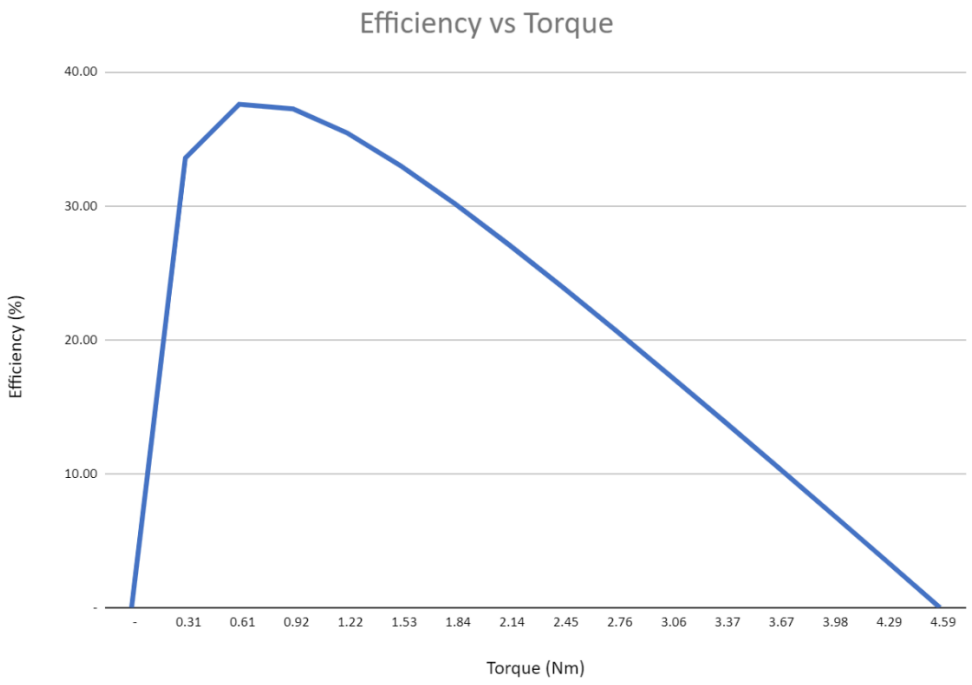


Figure 77: Efficiency vs. Torque Graph for Motor

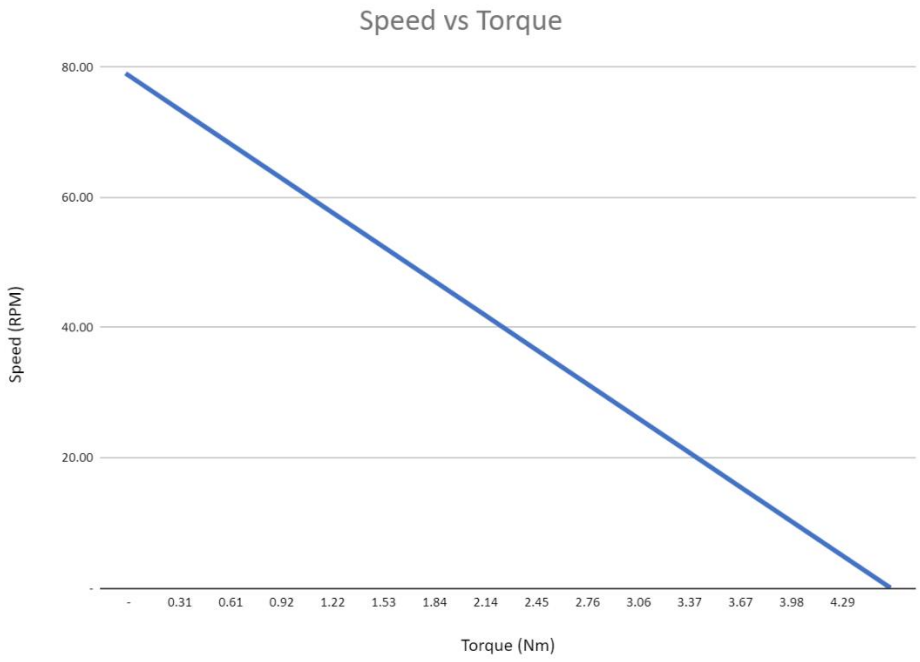


Figure 78: Speed vs. Torque Graph for Motor

Through careful consideration and research, the chosen motor to power each of the rear wheels of the Basebot, as well as the rack and pinion steering system, was the 131:1 Pololu 37D Metal Gearmotor, which can be seen below in Figure 79. In addition, this motor requires 24V to operate. The 131:1 Pololu 37D Metal Gearmotor is easily accessible through the Pololu website, and proves to be reliable through its affordable price, torque, and small make. Furthermore, the 131:1 Pololu 37D Metal Gearmotor has an attached encoder, which will aid in counting cycles of the motor when turning the pinion on the rack to either the left or the right. However, in order for these three motors to run, there must be at least 24V of supplied voltage.

More specifically, for the Basebot's back wheels individually, this motor proved to be sufficient. The 131:1 Pololu 37D Metal Gearmotor motor is one of the few available for an

affordable price, high torque, and small make. As seen above in Figure 75, this motor provides a large amount of power. Furthermore, this motor will never reach its maximum power, as the required amount of Watts for the Basebot to operate is 7.96W, which is less than 85%, in comparison to the maximum potential power output of the 131:1 Pololu 37D Metal Gearmotor. Additionally, as depicted through both Figure 78, it is apparent that the motor can reach and exceed the required 19.3 revolutions per minute. Therefore, this motor will allow the Basebot to move at the desired speed for extended periods of operation. Lastly, as seen in Figure 77, the Basebot motor requirements are located on the left side of the efficiency curve. This means that the 131:1 Pololu 37D Metal Gearmotor, as seen below in Figure 79, is within 80% of the maximum efficiency that is presented by the motor. Thus, the motor can handle the high amount of torque and power necessary to run the Basebot.

Furthermore, the 131:1 Pololu 37D Metal Gearmotor also was an efficient and effective motor choice for the rack and pinion steering mechanism. As seen above in Figure 75, this motor will never reach its maximum power, as the required amount of Watts for the rack and pinion system to operate is a mere 7.38W, which is less than 80% in comparison to the maximum power output of the 131:1 Pololu 37D Metal Gearmotor. Additionally, as depicted through both Figure 78, it is apparent that the motor can reach and exceed the required 68 revolutions per minute and therefore, this motor will allow the Basebot to move at the desired speed for extended periods of operation. Lastly, as seen in Figure 77, the Basebot pinion motor requirements are located on the left side of the efficiency curve. This means that the 131:1 Pololu 37D Metal Gearmotor is within 80% of the maximum efficiency that is presented by the motor. Thus, the motor can

handle the amount of torque and power necessary to run the Basebot, while being efficient and reaching the speed requirement.

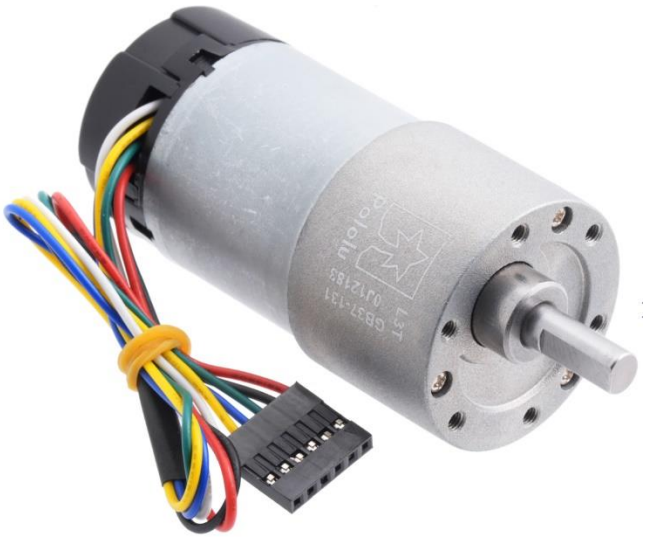


Figure 79: 131:1 Pololu 37D Metal Gearmotor

Battery Selection:

For the motors to run efficiently and correctly, at least 24V of power was required. To fulfill this requirement, it was found that running two 12V batteries in series would be the best option. This is because two 12V batteries would not only take up less space than a single 24V battery, but they would also weigh less and be much more cost efficient. The chosen battery was the Mighty Max Battery 12V 18AH.

Component:	Maximum Current Draw (A)	Idle Current Draw (A)	Maximum Power (W)	Idle Power (W)	Source
IMU	0.0123	0.0004	0.04059	0.00132	Link
H Bridge (x2)	-	-	-	-	Link
Raspberry Pi 4B	1.25	0.6	6.4	2.7	Link
Pololu Motors (x3)	16.5	0.6	28.2	0	Link
Total					
	17.7623	1.2004	34.64059	2.70132	
Battery Amp-Hours	Battery Voltage (V)	Battery Watt-Hours	Source		
36	24	864	Link		

Figure 80: Battery Power Calculations

As seen in Figure 80 above, battery calculations were computed. This was a necessary step in ensuring that this battery would be the correct fit for the Basebot system. All of the onboard electronics, including the AdaFruit BNO055 IMU, two H-Bridges, a Raspberry Pi 4B, and three 131:1 Pololu 37D Metal Gearmotors were analyzed. Each of these component's maximum current draw in amps, idle current draw in amps, maximum power in watts, and idle power in watts were recorded and multiplied for the amount of each in the table above. The same was done for the battery, analyzing the voltage in volts and battery watt-hours, multiplied by two for both of the batteries utilized. This proved that together, these two batteries are more than reliable to power the Basebot system and fulfill all of the set requirements.

Basebot Housings

To ensure that none of the electrical components are impacted by sand or other debris during cycles, housings for each component were designed.

Motor and Encoder Housings:

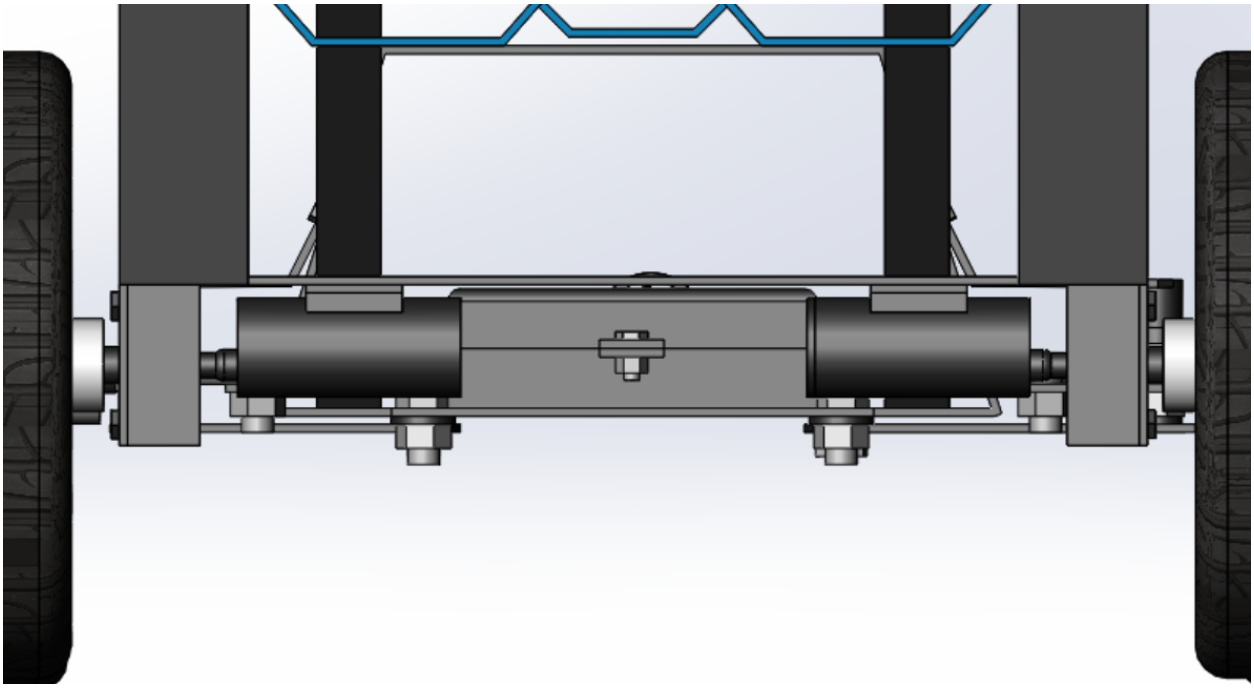


Figure 81: Housing for Back (Driven) Motors

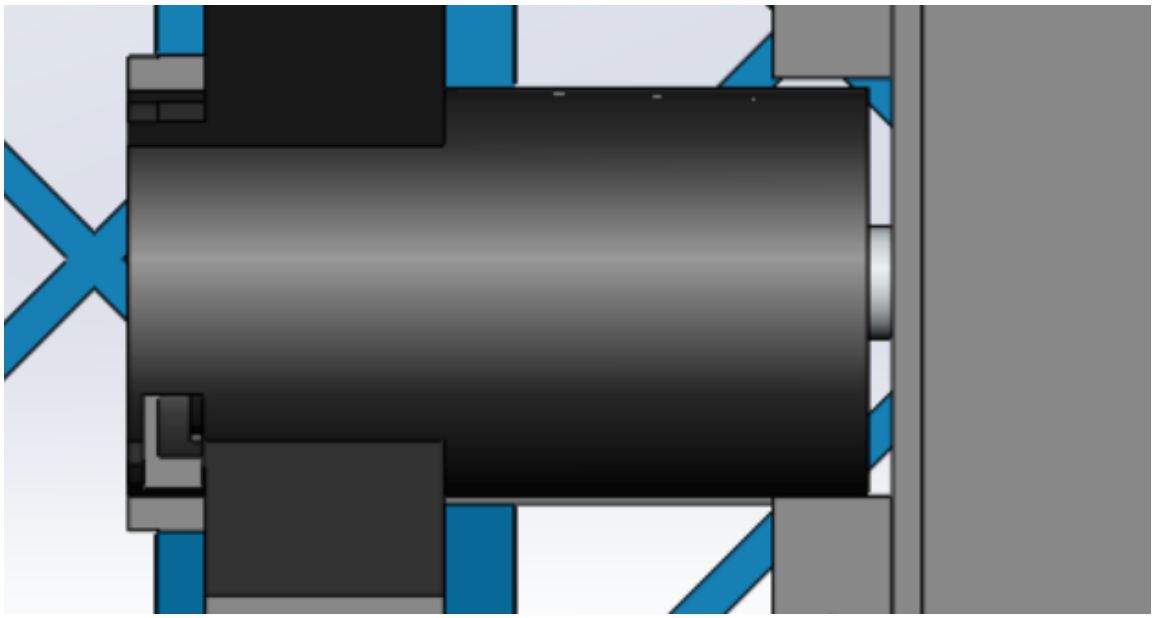


Figure 82: Housing for Front Motor on Rack and Pinion

Once the sandy terrain of beaches that the Basebot would be completing its run time on was analyzed, it became apparent that the drive motors needed to be encased. This was necessary in case of an event where sand got kicked up by the wheels or the Basebot was driving over or through any uneven spaces where sand and other debris were present. Figure 83 depicts the housing that was created for the two back motors on the Basebot. This cover encapsules both of the motors into one cohesive unit. In addition, there are holes for wires within the casing to allow for ease of access for wiring. Furthermore, Figure 81 above depicts the rack and pinion motor cover. This housing serves as both a bracket and protection from sand. Similarly to the back motor housing, there are holes for wires to flow through, for simple and stress-free wiring

Rack and Pinion Housing:

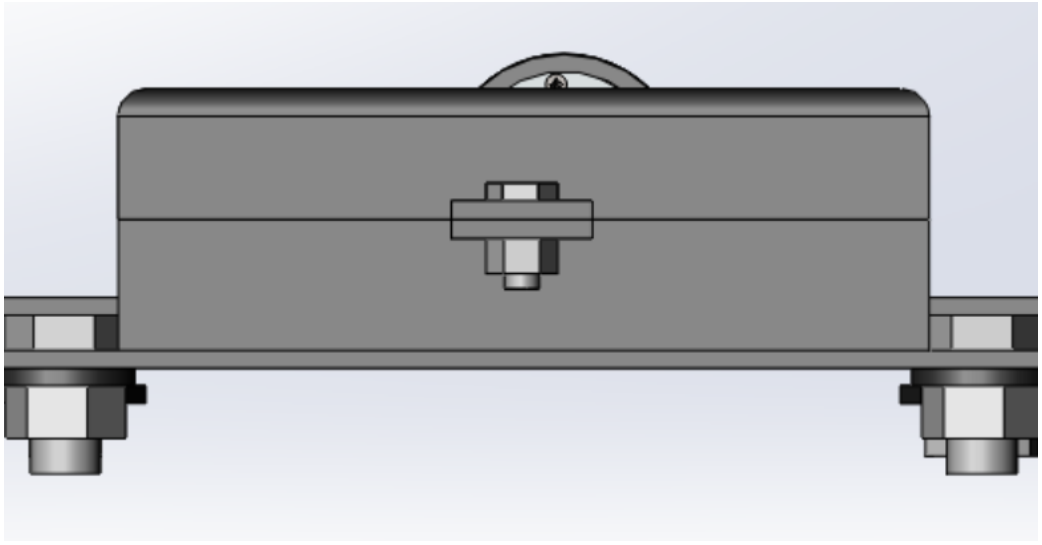


Figure 83: Housing for Rack and Pinion

It was a top priority to encase the rack and pinion portion of the Basebot. This was necessary due to the nature of the mechanism. For example, if sand or other debris were to enter

the teeth of the pinion on the rack, the gears could slip and lead to failure. Overall, this housing fully encloses both the rack and the pinion, as seen above in Figure 83.

Battery and Electronics Housing:

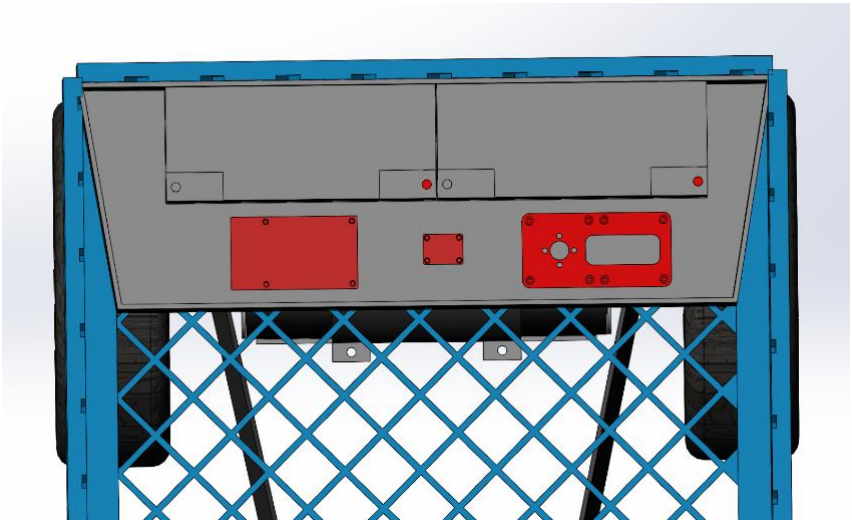


Figure 84: Top View of the Battery and Electronics Housing without the Cover

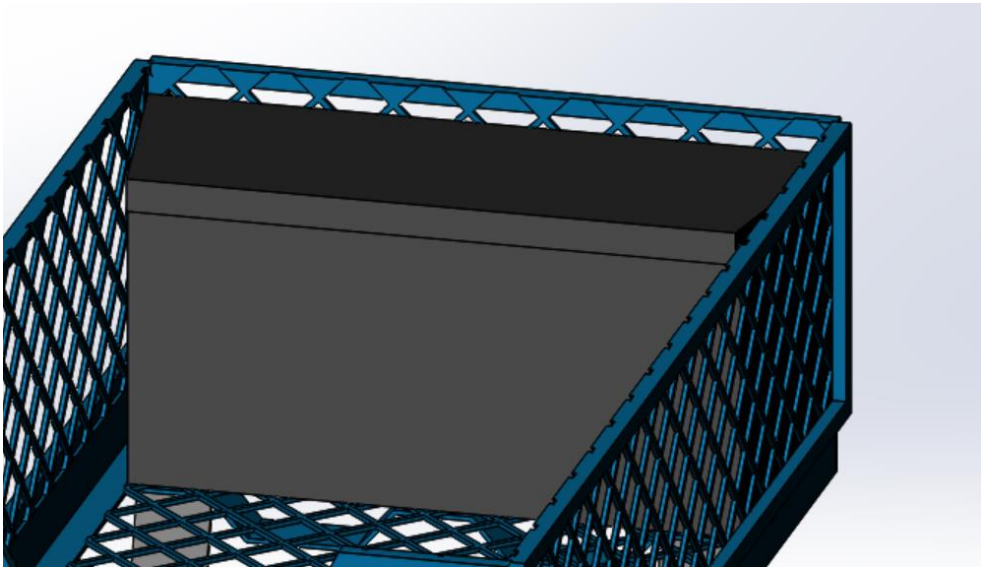


Figure 85: Fully Enclosed of the Battery and Electronics Housing

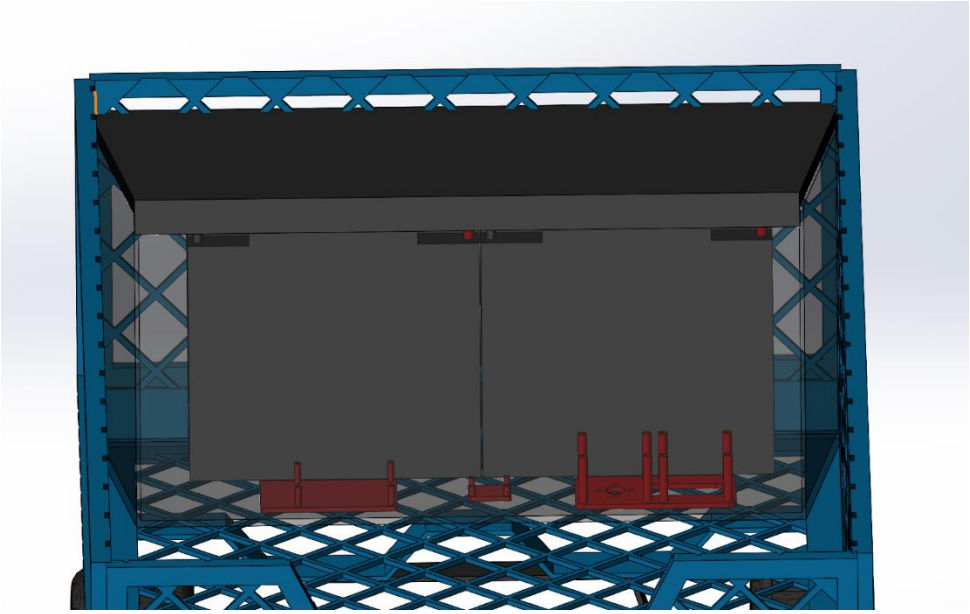


Figure 86: Front View of the Battery and Electronics Housing with Transparent Front Plate

To ensure that the electronics and batteries were unphased by water and sand, a casing was designed. This housing sits in the back of the Basebot and allows for the electronics to sit at the bottom, to ensure easy access for wiring and cable management. The batteries sit on top and are easily accessible from the top of the housing when the cover is opened. This design, as seen above in Figure 84, Figure 85, and Figure 86, is extremely user-friendly, as well as minimalistic and practical for the environment the Basebot will be operating in.

Electronics:

In order to allow the BaseBot to successfully drive on sand, electronics were researched and chosen based on the required tasks the Basebot needed to complete.

IMU Selection:

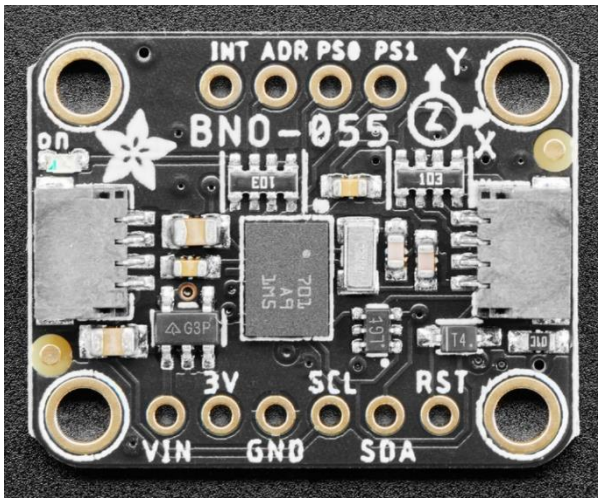


Figure 87: AdaFruit BNO055 IMU

There were many factors that were considered while choosing an IMU to work with for the Basebot. Ultimately, the AdaFruit BNO055 IMU was selected, as seen above in Figure 87. This particular IMU was chosen because of its capabilities, as well as the fact that this is the same sensor that is utilized on the SmallBot. One of the main reasons behind choosing AdaFruit BNO055 was that it has the ability to sense nine degrees of freedom. This is due to the fact that it has an accelerometer, a magnetometer, and a gyroscope in each of the three axes: x, y, and z, also known as pitch, roll, and yaw respectively. In addition, the BNO055 can output the following sensor data: the absolute orientation with respect to the Euler vector, the absolute orientation with respect to the quaternion, the angular velocity vector, the magnetic field strength

vector, the linear acceleration vector, the gravity vector, and the temperature (Adafruit. (n.d.)). Thus, through the use of the AdaFruit BNO055, the Basebot can be programmed for navigation, such as waypoint following, with a specified speed and orientation with ease, while having the orientation and velocity constantly tracked, which is an essential aspect of the Basebot's overall purpose.

Hardware Selection:

Figure 88: H-Bridge Motor Driver Selection

In order for the Basebot to have both forward and backward driving capabilities, it was essential to incorporate H-Bridges into the design. H-Bridges are utilized to reverse the polarity of a voltage that is applied to a motor. In this design with the 131:1 Pololu 37D Metal Gearmotor which is an electromagnetic DC brushed motor, it is crucial to complete the Basebot's required tasks. Furthermore, if H-Bridges were not included within this design, the Basebot would not be able to efficiently complete a run, as it would have to fully turn around 90 degrees while driving, as opposed to simply switching the direction that the motor is spinning in through polarity. The

chosen H-Bridge for this application is the BTS7960 Motor Driver by Handson Technology, as seen above in Figure 88. There is one H-Bridge for both of the back motors, as well as an H-Bridge for the rack and pinion system, which means that there are two of the BTS7960 Motor Driver H-Bridges onboard the Basebot.

Microprocessor Selection:

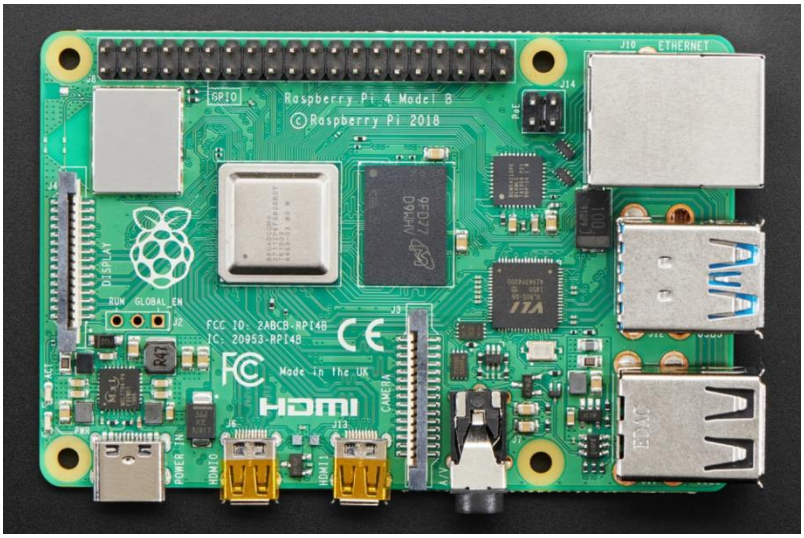


Figure 89: Raspberry Pi 4 Model B

As depicted above in Figure 89, the Raspberry Pi 4 Model B was chosen for a variety of reasons, including that the team had successfully integrated electronics and software on the Smallbot using the same microprocessor. In addition, the Raspberry Pi 4 Model B was chosen due to its ability to effectively communicate with other devices. This is an essential function of the Basebot, as it must communicate with the Smallbot, and in the future, with multiple Smallbots. Furthermore, the Raspberry Pi 4 Model B is able to handle camera capabilities. This is quintessential to the Basebot's operation, as its entire purpose is to locate new regions for the Smallbot to clean up on beaches, as well as to keep track of the Smallbot. Lastly, this particular

microprocessor was chosen because it is readily upgradable, due to its 40 GPIO pins, 4 GB of RAM, and its 64-bit quad core processor (RaspberryPi.org. (n.d.)). This is a necessary function for the Basebot to acquire, as this project is ever-expanding.

Testing:

In order to ensure correct values were being assumed throughout the duration of the designing process of the Basebot, a few tests were performed.

Coefficient of Friction of Sand Testing:

Figure 90: Coefficient of Friction of Sand Testing

Normal Force:

$$m_{cart} = 16.6kg$$

$$m_{Brick} = 1.90\ kg$$

$$m_{Total} = m_{cart} + 4 * m_{Brick}$$

$$m_{Total} = 24.2kg$$

$$F_N = m_{Total} * g$$

$$F_N = 24.2kg * 9.81\frac{m}{s^2}$$

$$F_N = 237N$$

Force of the Spring Balance:

$$F_{Spring\ Balance} = m_{Pulling\ Force} * g$$

$$F_{Spring\ Balance} = 7.14kg * 9.81\frac{m}{s^2}$$

$$F_{Spring\ Balance} = 70.0N$$

Coefficient of Friction:

$$\mu = \frac{F_{Spring\ Balance}}{F_N}$$

$$\mu = \frac{70.0N}{237N}$$

$$\mu = 0.295$$

Digital scales and spring balances measure weight, which is also known as the force acting on a mass equal to the object's mass times the acceleration due to gravity. Therefore, when utilizing such scales and balances, it was determined that the output force would be measured in kg but need to be multiplied by 9.81m/s^2 (O'Driscoll, A. (2018, July 17)). Utilizing

this knowledge, an experiment was conducted to confirm that the assumed coefficient of friction of clean sand is equal to 0.3, a simple test was conducted.

This test was performed on the table of sand in the team's laboratory space in Atwater Kent, on Worcester Polytechnic Institute's campus. This was achieved by first weighing a brick, which was 1.9kg. In addition, four bricks were utilized for this test weighing a total of 7.6kg. This weight was coupled with the measured weight of the Basebot, 16.6kg, to get a total weight of the Basebot 24.2kg. In order to get the total force, the total weight was multiplied by 9.81m/s^2 to achieve a normal force equal to 237N.

Next, the spring balance was attached to the front of the Basebot system, seen in Figure 90, and dragged through the sand by the spring balance. The observed maximum force was determined to be equal to 7.14kg. Similar to the normal force, the maximal force was multiplied by 9.81m/s^2 due to the spring gauge measuring the force acting on a mass equal to the object's mass times the acceleration due to gravity. The final maximal force was determined to be 70.0N. Furthermore, using the two forces, the coefficient of friction was calculated to be 0.27. This value is within a reasonable tolerance to the assumed value of 0.3, and therefore, the calculated values prove to be correct.

Turning Force Testing:

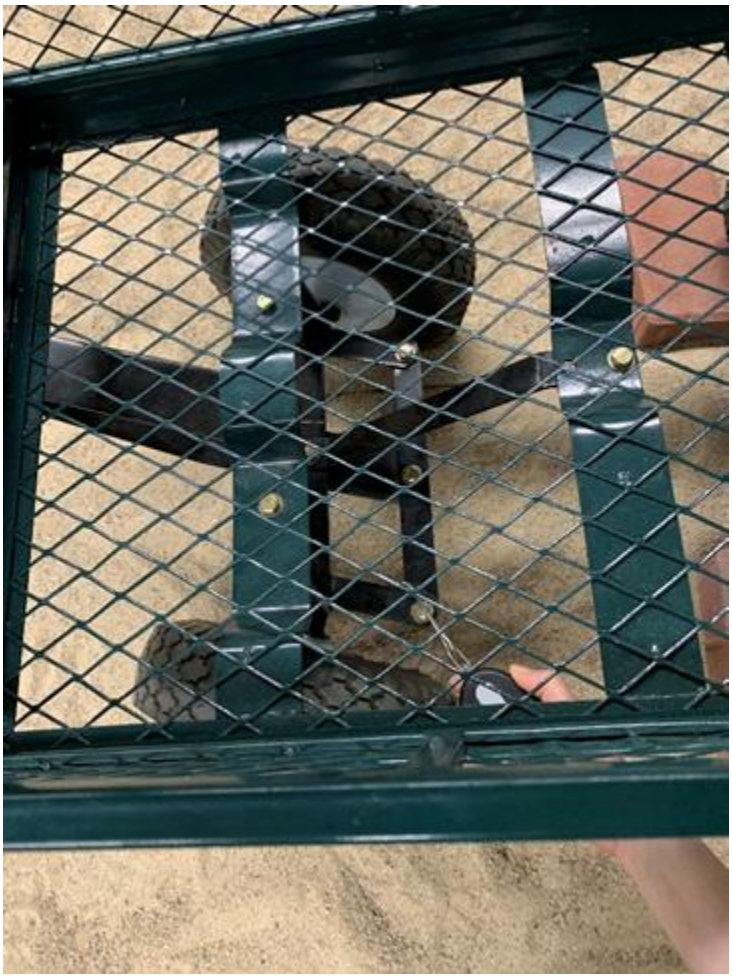


Figure 91: Turning Force Testing Part 1

Figure 92: Turning Force Testing Part 2

Turning Force:

$$F_{Turning} = m_{Pulling\ Force} * g$$

$$F_{Turning} = 1.44kg * 9.81\frac{m}{s^2}$$

$$F_{Turning} = 14.1N$$

In order to ensure that the turning force of the steering mechanism was correct, a simple test was performed on the table of sand in the team's laboratory space in Atwater Kent, on Worcester Polytechnic Institute's campus. The test was conducted by first rigidly hindering movement of the Basebot's rear wheels with two bricks utilized as a tire chock, as seen in Figure 92 above. The spring balance was then attached to the rear corner of the front steering, as seen

above in Figure 91. The spring balance was then pulled co-linearly with the steering, as also seen in Figure 91, and the maximal observed force was recorded to be 1.44kg. That was then multiplied by the acceleration of gravity, 9.81m/s^2, which was measured similarly to the coefficient of friction experiment in section 1.4.1 above, in which the spring balance was pulled. The value of the observed turning force was 14.1N, as seen below in the calculations. Comparatively with the theoretical calculated force on a singular control arm, the force was within a reasonable tolerance with a variation of 0.17N. Therefore, confirming the force required to turn the front wheels of the Basebot system using the rack and pinion steering system.

Basebot Results:

Figure 93: Front View with Wheels Turned Basebot Final Design

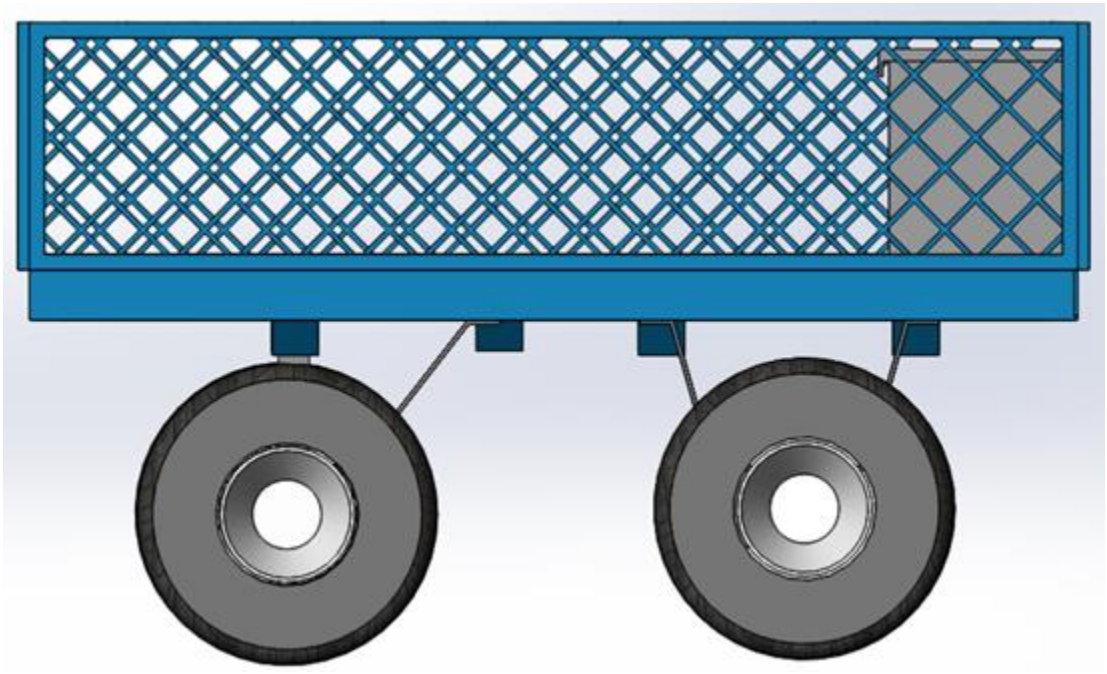


Figure 94: Side View of Final Basebot Design

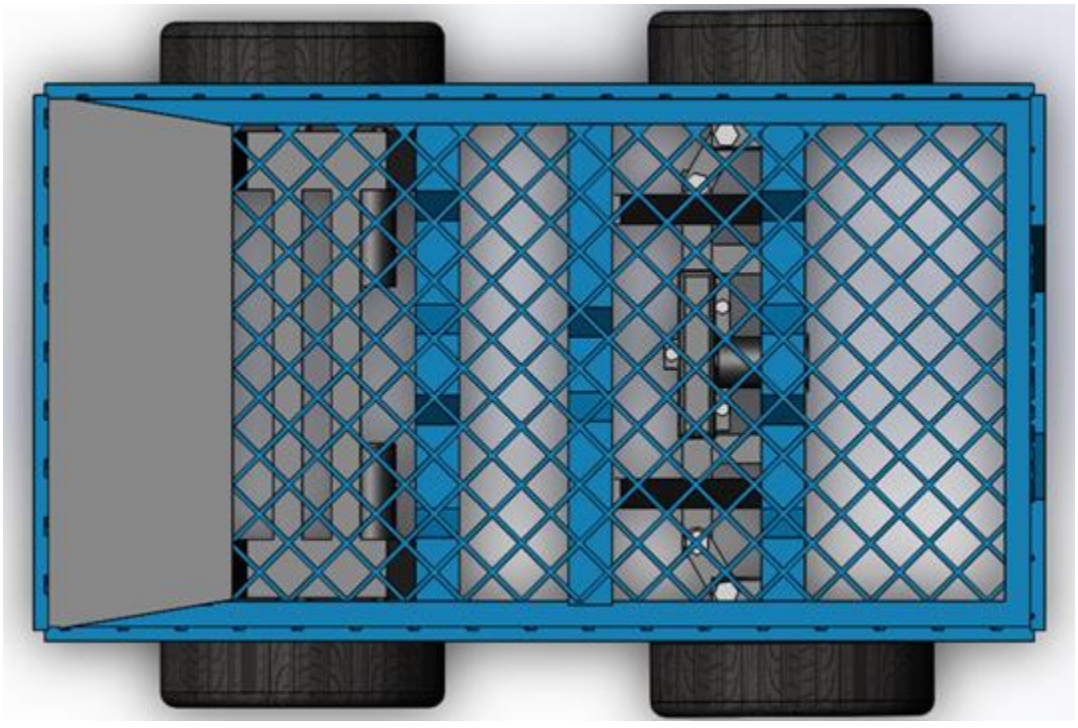


Figure 95: Top View of Final Basebot Design

Overall, the Basebot drive design was carefully crafted, while meeting and exceeding all requirements. Figure 95, Figure 96, and Figure 97 above all show the simulated Basebot design. The exploded view of both the steering mechanism and the gearbox design can be found in Figure 98 and Figure 99 in the Appendix below. In addition, the bill of materials for the overall Basebot can be seen in the Appendix through Figure 100. The Basebot was constructed utilizing many of the garden cart's original components, as well as additional structural parts and newly designed mechanisms to become a fully autonomous vehicle. In addition, the Basebot will be able to meet and surpass the 15 meter per minute speed requirement. Furthermore, the Basebot will be able to drive on sand effectively and reliably, due to the rack and pinion steering system, thick and wide wheels, and the provided motor torque significantly within the system's power requirements. With this design, the Basebot will be able to be constructed and executed early within the next iteration of this project. The SolidWorks model, calculations, and report will cohesively act as a guide for the next project team to bring this design to life.

Bibliography

"ACT: At the Beach." *EPA*, Environmental Protection Agency, 26 Mar. 2019, www.epa.gov/beaches/act-beach

Adafruit. (n.d.). *Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055*. Sensors. https://www.adafruit.com/product/2472#description.

Araújo, M., Silva-Cavalcanti, J., & Costa, M. (2018, June 15). Anthropogenic Litter on Beaches with Different Levels of Development and Use: A Snapshot of a Coast in Pernambuco (Brazil). Retrieved November 12, 2020, from https://www.frontiersin.org/articles/10.3389/fmars.2018.00233/full

Barry, Carolyn. "Plastic Breaks Down in the Ocean, After All-and Fast." *National Geographic*, National Geographic, 20 Aug. 2009, www.nationalgeographic.com/news/2009/8/plastic-breaks-down-in-ocean-after-all-and-fast/

Binsfield Engineering Inc. (n.d.). *Horsepower vs Torque: How Both Provide Insight into Engine Performance*. Torque Measurement. https://binsfeld.com/horsepower-torque/.

Bortman, M. L. (2020, November 4). Ocean Dumping. Retrieved November 12, 2020, from https://www.encyclopedia.com/earth-and-environment/geology-and-oceanography/geology-and-oceanography/ocean-dumping

Chow, Lorraine. *10 Most Common Types of Beach Litter Are All Plastic*. 18 Dec. 2019, www.ecowatch.com/beach-litter-plastics-ocean-conservancy-2581760475.html

Dronyx. (2016). Solarino Sand Beach Cleaner Robot. https://www.dronyx.com/solarino-beach-cleaner/

Fine Software. (n.d.). Table of Ultimate Friction Factors for Dissimilar Materials.

https://www.finesoftware.eu/help/geo5/en/table-of-ultimate-friction-factors-for-dissimilar-materials-01/.

Google. (n.d.). *Cloud AutoML Documentation* |. Google Cloud. Retrieved July 1, 2020, from https://cloud.google.com/automl/docs?hl=en_US

Hattersley, L. (2020). *Raspberry Pi 4 vs Raspberry pi 3b+*. https://magpi.raspberrypi.org/articles/raspberry-pi-4-vs-raspberry-pi-3b-plusHuman Health *Environmental Science and Pollution Research*. Retrieved from https://link-springer-com.ezproxy.wpi.edu/article/10.1007/s11356-017-9910-8

"Impacts of Mismanaged Trash." *EPA*, Environmental Protection Agency, 30 July 2020, www.epa.gov/trash-free-waters/impacts-mismanaged-trash

iSea. *About Us*. 2020. Retrieved from: https://isea.com.gr/about-us/our-goals/?lang=en.

Lebreton, L. C. M., Zwet, J. V. D., Damsteeg, J.-W., Slat, B., Andrady, A., & Reisser, J. (2017).

Mambra, S. (2020, October 30). Ocean Pollution: 6 Things That Make It Worse. Retrieved November 12, 2020, from https://www.marineinsight.com/environment/causes-and-effects-of-ocean-dumping/

National Geographic Society. "Ecosystem." *National Geographic Society*, 9 Oct. 2012, www.nationalgeographic.org/encyclopedia/ecosystem/. Accessed 20 Nov. 2020.

River plastic emissions to the world's oceans. *Nature Communications*, *8*(1). Doi: 10.1038/ncomms15611

Leonard, George, et al. "Plastics in the Ocean." *Ocean Conservancy*, 30 Oct. 2020, oceanconservancy.org/trash-free-seas/plastics-in-the-ocean/

"Marine Debris Impacts." *U.S. Department of the Interior*, Office of Congressional and

Legislative Affairs, 19 Sept. 2019, www.doi.gov/ocl/marine-debris-impacts

Merriam-Webster. (n.d.). Anthropogenic. In *Merriam-Webster.com dictionary*. Retrieved November 16, 2020, from https://www.merriam-webster.com/dictionary/anthropogenic

Nasa. (2019, August 20). Wheels. Retrieved March 23, 2021, from https://mars.nasa.gov/msl/spacecraft/rover/wheels/

Ocean pollution. (n.d.). Retrieved November 11, 2020, from

https://www.noaa.gov/education/resource-collections/ocean-coasts/ocean-pollution

O'Driscoll, A. (2018, July 17). The Differences Between Balances and Scales.

https://labbalances.net/blogs/blog/differences-between-balances-and-scales.

Olson, E. (2010). *AprilTag*. https://april.eecs.umich.edu/software/apriltag

Parker, Laura. “Animals Eat Ocean Plastic Because It Smells Like Food.” , *Thanks to*

DMS-Producing Algae, National Geographic, 9 Nov. 2016,

www.nationalgeographic.com/news/2016/11/animals-eat-ocean-plastic-because-of-smell-dms-algae-seabirds-fish/

Parker, Laura. “The World's Plastic Pollution Crisis Explained.” *Plastic Pollution Facts and*

Information, National Geographic, 7 June 2019, www.nationalgeographic.com/environment/habitats/plastic-pollution/

PittmanExpress. (1994). *GM9236S025*. http://courses.csail.mit.edu/6.141/spring2013/pub/labs/GeneralInfo/component-datasheets/Motor_GM9236S025.pdf

RaspberryPi.org. (n.d.). *Products*. Raspberry Pi 4.

https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

Reddy, Simon. “Plastic Pollution Affects Sea Life Throughout the Ocean.” *The Pew Charitable Trusts*, PEW, 24 Sept. 2018, www.pewtrusts.org/en/research-and-analysis/articles/2018/09/24/plastic-pollution-affects-sea-life-throughout-the-ocean

Sharma, S., & Chatterjee, S. (2017). Microplastic pollution, a threat to marine ecosystem and human health: a short review. Environmental Science and Pollution Research, 24(27), 21530-21547.

Stromberg, Joseph. “Where Do Humans Really Rank on the Food Chain?” *Smithsonian.com*, Smithsonian Institution, 2 Dec. 2013, www.smithsonianmag.com/science-nature/where-do-humans-really-rank-on-the-food-chain-180948053/

Supervisely. (n.d.). *Home*. Retrieved March 10, 2021, from https://supervise.ly/

Table of Ultimate Friction Factors for Dissimilar Materials | Influence of Friction between Soil and back of the Structure | GEO5 | Online Help. (n.d.). https://www.finesoftware.eu/help/geo5/en/table-of-ultimate-friction-factors-for-dissimilar-materials-01/.

TCP vs UDP: What's the Difference? (n.d.). https://www.guru99.com/tcp-vs-udp-understanding-the-difference.html#2

US Army Corps of Engineers. "Beach Ecosystem Restoration." *U.S. Army Engineer Institute for Water Resources (IWR)*, www.iwr.usace.army.mil/Missions/Coasts/Tales-of-the-Coast/Corps-and-the-Coast/Environmental-Restoration/Beach-Ecosystem. Accessed 20 Nov. 2020.

Weinhardt, Eric. “How Much Litter Is In Our Ocean?” *Adopt A Beach*, Adopt A Beach, 2 Apr. 2019, adoptabeach.com/blog/2019/4/1/how-much-litter-is-in-our-ocean

Werner, Stefanie, et al. “Harm caused by Marine Litter”. 2016. PDF File.

Wikipedia. Fiducial marker. (2020, December 06). Retrieved March 23, 2021, from https://en.wikipedia.org/wiki/Fiducial_marker

Wikimedia Foundation. (2021, March 31). *Mussel*. Wikipedia. https://en.wikipedia.org/wiki/Mussel.

Appendix

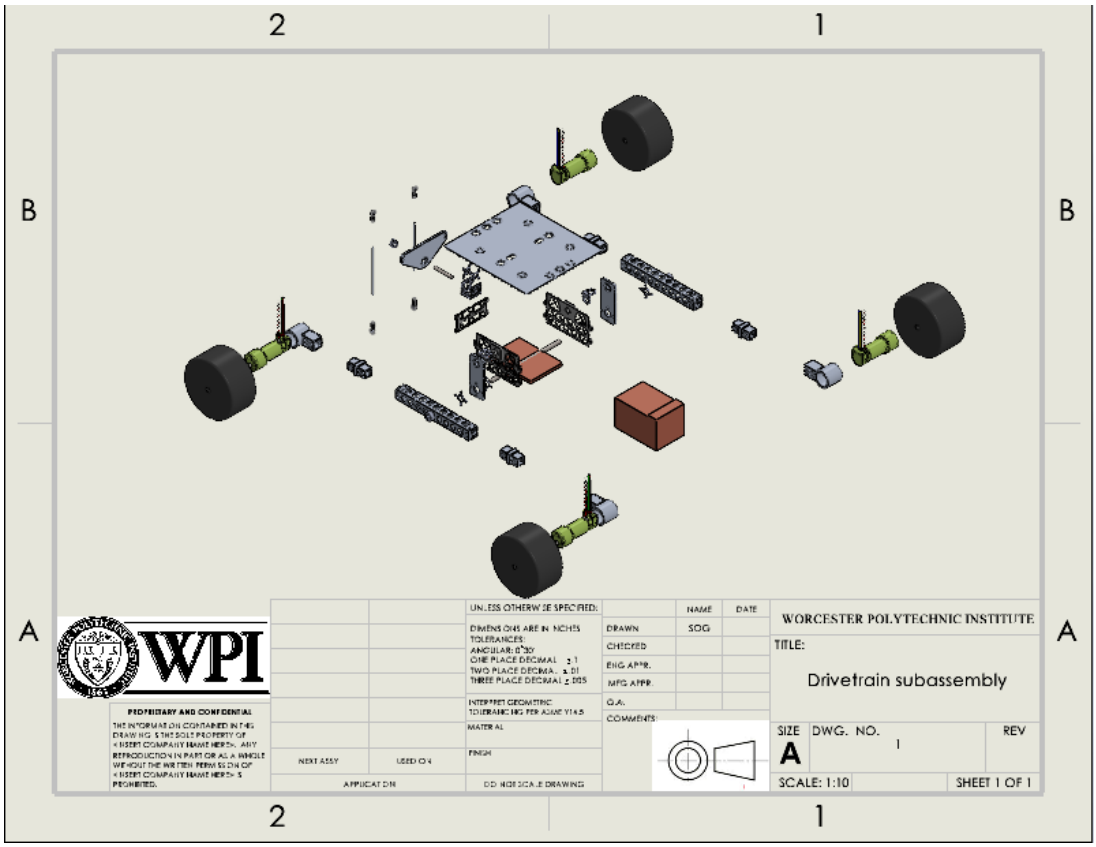


Figure 96: Solidworks CAD Drivetrain Sub Assembly Exploded View Drawing

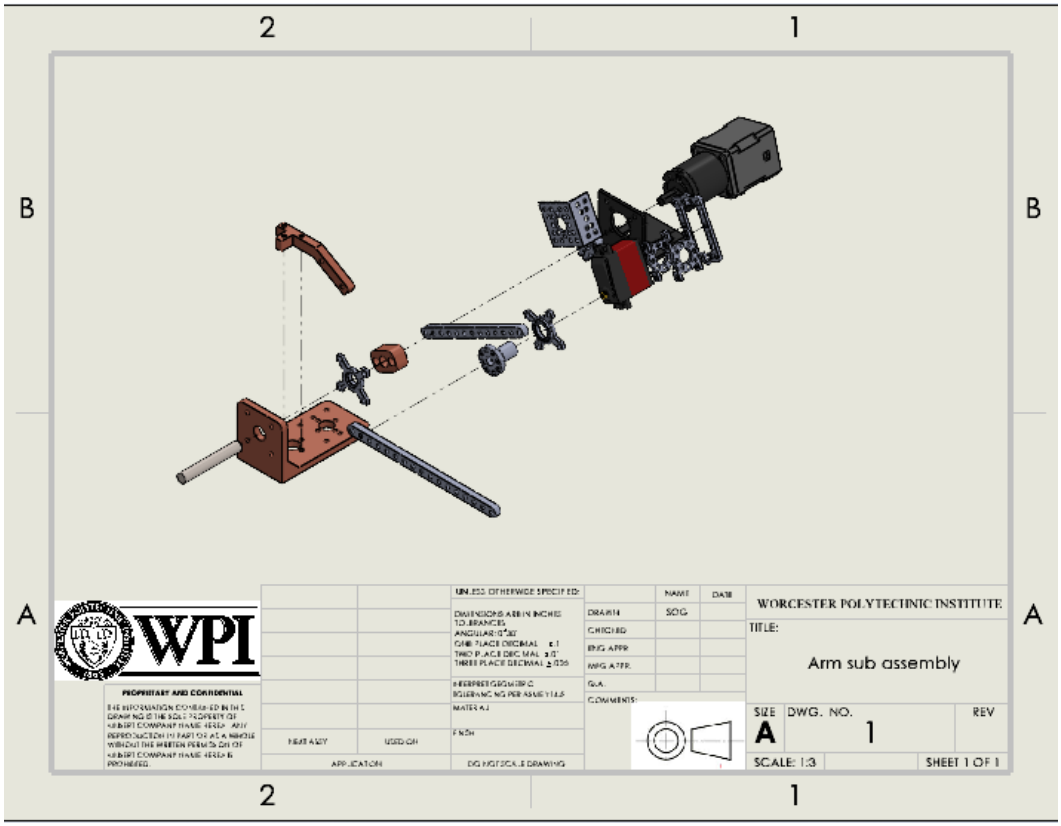
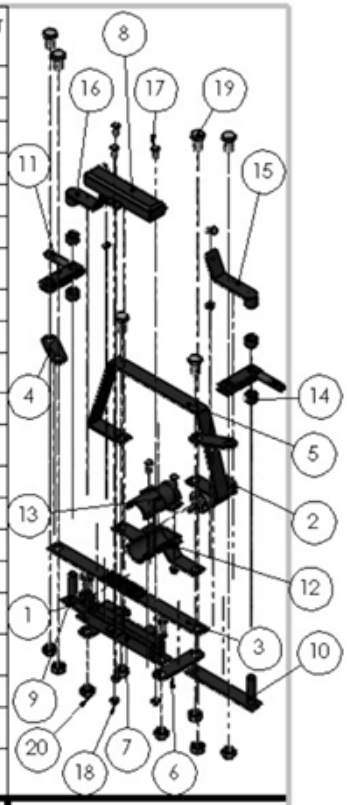


Figure 97: Solidworks CAD Arm Sub Assembly Drawing

ITEM NO.	PART NUMBER	QTY.	MATERIAL	UNIT COST	EXT. COST	UNIT WEIGHT (LBS)	EXT. WEIGHT (LBS)
1	PINION GEAR	1	STAINLESS STEEL	$4.00	$4.00	0.1	0.1
2	POLOLU 37D MOTOR	1	MOTOR	$24.95	$24.95	0.39	0.39
3	RACK	1	STAINLESS STEEL	$14.70	$14.70	4.08	4.08
4	CONTROL ARM	2	STAINLESS STEEL	$8.10	$16.20	1.9	3.8
5	FRONT BRACKET	1	STAINLESS STEEL	$0	$0	3.83	3.83
6	SHORT RACK SUPPORT	2	STAINLESS STEEL	$8.40	$16.79	2.3	4.6
7	RACK HOUSING (BOTTOM)	1	STAINLESS STEEL	$12.59	$12.59	3.45	3.45
8	RACK HOUSING (TOP)	1	STAINLESS STEEL	$9.79	$9.79	2.68	2.68
9	RIGHT WHEEL SUPPORT	1	STAINLESS STEEL	$7.86	$7.86	1.86	1.86
10	LEFT WHEEL SUPPORT	1	STAINLESS STEEL	$7.86	$7.86	1.86	1.86
11	FRONT AXLE	2	STAINLESS STEEL	$9.30	$18.58	1.95	3.9
12	MOTOR HOUSING (BOTTOM)	1	STAINLESS STEEL	$19.59	$19.59	5.36	5.36
13	MOTOR HOUSING (TOP)	1	STAINLESS STEEL	$16.79	$16.79	4.6	4.6
14	SPACER	4	STAINLESS STEEL	$3.11	$12.44	0.02	0.08
15	RIGHT KING PIN	1	STAINLESS STEEL	$13.78	$13.78	3.5	3.5
16	LEFT KING PIN	1	STAINLESS STEEL	$13.78	$13.78	3.5	3.5
17	0.25IN HEX BOLT	7	CHROME PLATED STEEL	$1.42	$9.94	0.02	0.14
18	0.25IN NUT	7	CHROME PLATED STEEL	$0.88	$6.16	0.02	0.14
19	0.5IN HEX BOLT	8	STAINLESS STEEL	$1.01	$8.08	2.09	16.72
20	0.5IN NUT	8	ZINC PLATED STEEL	$0.22	$1.76	0.03	0.24
				ASSEMBLY COST	$235.64	ASSEMBLY WEIGHT	61.33



BEACHBOTS MQP	SCALE 1:7	STEERING MECHANISM EXPLODED VIEW	4/25/2021

Figure 98: Exploded View of the Basebot Steering Mechanism

ITEM NO.	PART NUMBER	QTY.	MATERIAL	UNIT COST	EXT. COST	UNIT WEIGHT (LBS)	EXT. WEIGHT (LBS)
1	GEARBOX BOTTOM	1	STAINLESS STEEL	$38.22	$38.22	5.75	1.91
2	12 TOOTH GEAR	2	STAINLESS STEEL	$4.00	$8.00	0.1	0.2
3	24 TOOTH GEAR	2	STAINLESS STEEL	$6.00	$12.00	0.1	0.2
4	R155-2RS MINI BALL BEARING	8	PRESSED STEEL AND RUBBER	$0.38	$3.04	0.05	0.4
5	2IN AXLE	2	STAINLESS STEEL	$1.19	$2.38	0.03	0.06
6	1IN AXLE	1	STAINLESS STEEL	$1.19	$1.19	0.02	0.02
7	GEARBOX LID	1	STAINLESS STEEL	$12.74	$12.74	1.91	1.91
8	0.25IN HEX BOLT	2	CHROME PLATED STEEL	$1.42	$2.84	0.02	0.04
				ASSEMBLY COST	$80.41	ASSEMBLY WEIGHT	4.74

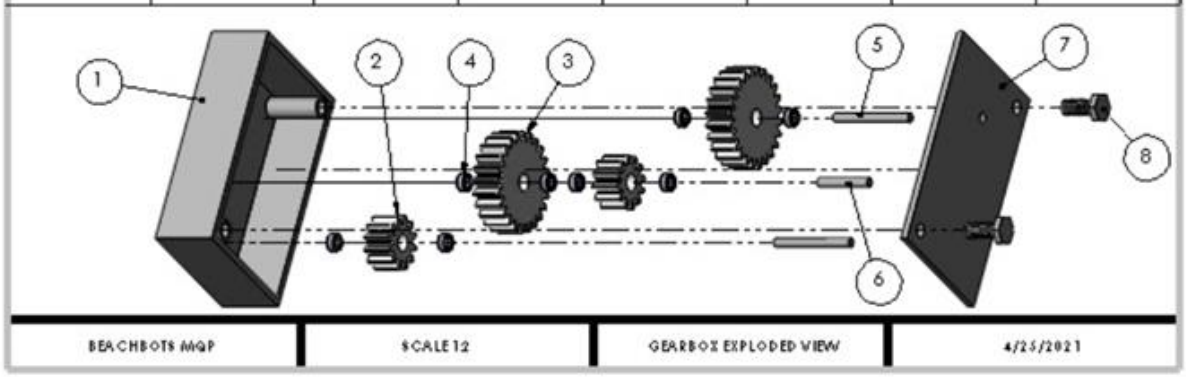


Figure 99: Exploded View of the Basebot Gearbox

Name of Item	Quantity	Material	Unit Cost	Total Cost	Link
R155-2RS Mini Ball Bearing 5/32x5/16x1/8 Sealed	8	Pressed Steel & Rubber	$0.38	$3.04	Link
Radial Ball Bearing, Double Sealed, 17mm Bore Dia., 38mm Outside	4	Pressed Steel & Rubber	$15.98	$63.92	Link
Everbilt 1/4in.-20 x 3/4in Chrome Hex Bolt (3-Pack)	9	Chrome Plated Steel	$1.42	$12.78	Link
Everbilt 1/4in.-20 Chrome Nylon Lock Nut (4-Pack)	7	Chrome Plated Steel	$0.88	$6.16	Link
Prime-Line 1/2in.-13x1in. Grade 304 Stainless Steel Hex Bolts (25-Pack)	8	Stainless Steel	$1.01	$8.08	Link
Everbilt 1/2in.-13 Zinc Plated Hex Nut (50-Pack)	8	Zinc Plated Steel	$0.22	$1.76	Link
			Total Cost	$95.74	

Figure 100: Basebot Bill of Materials

Final Release from Github: https://github.com/SpencerGregg/beachbots2020/releases/tag/final

Github link to repo: https://github.com/SpencerGregg/beachbots2020

PDD & Demo videos:

https://www.youtube.com/playlist?list=PLyf91onamMM4yoIhD3SHlfeA9mGNKwy6-

https://www.youtube.com/watch?v=z2DAhZM5XK0