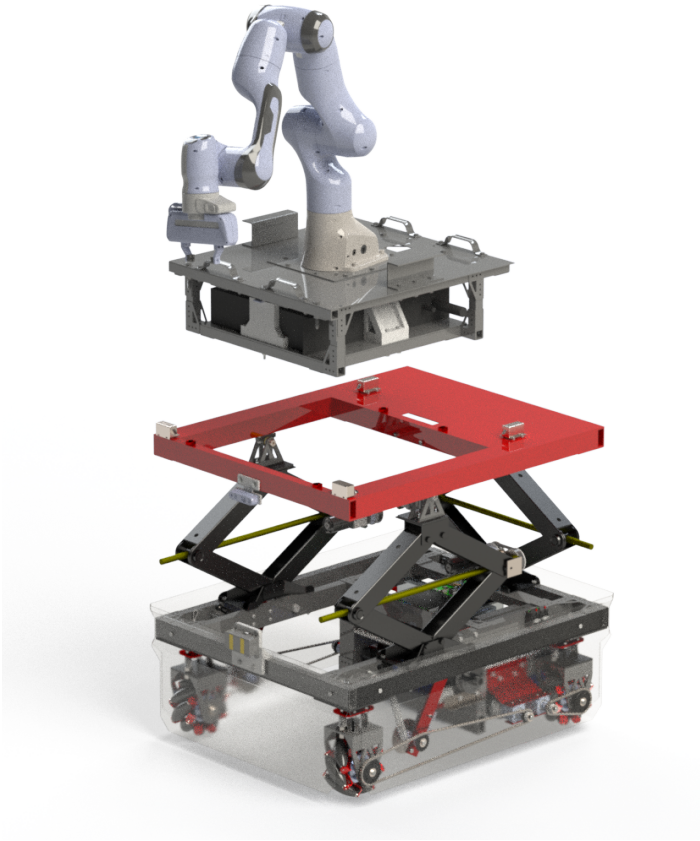
Design and Implementation of a Modular Mobile Robotic Base



WORCESTER POLYTECHNIC INSTITUTE

By: Kenneth Armijo, Alexander Corey, Braden Foley, Edward Jackson, & Timothy McCarthy

Date: April 28th, 2022

Report Submitted to:

Professor Berk Calli Professor William Michalson

Abstract

Mobile robotic bases provide mobility for robotic serial manipulators and are essential for service technologies, warehouse applications, and robotics research. However, current mobile bases are expensive and require significant effort to integrate with robotic manipulators. The goal of this MQP was to design and implement a modular, programmable, and affordable mobile platform for interfacing with various manipulators. The team created a chassis and elevator system to increase mobility and introduce effortless electrical and mechanical integration with various manipulators utilizing a modular design. Furthermore, abstraction between user input and manipulator-specific hardware drivers was developed for seamless control of the base and manipulator.

Table of Contents

Abstract **1**

Table of Contents **2**

Acknowledgements **5**

1 Introduction **6**

2 Background **7**
- 2.1 Hardware 7
  - 2.1.1 Robotic Manipulators 7
  - 2.1.2 Modular Hardware 11
- 2.2 Software 12
  - 2.2.1 Modular Frameworks 12
  - 2.2.2 ROS 12
- 2.3 Robotic Arms at WPI 13

3 Design **16**
- 3.1 Use Cases 16
- 3.2 Initial Parameters and Design Goal 16
- 3.3 Mechanical Design 17
  - 3.3.1 Initial Mechanical Design 17
    - 3.3.1.1 External Factors 17
    - 3.3.1.2 Mobile Design Factors 19
    - 3.3.1.3 Modular Design Factors 23
    - 3.3.1.4 Home Base 25
  - 3.3.2 Initial Mechanical Analysis for Advanced Design 26
    - 3.3.2.1 Center of Mass 31
    - 3.3.2.2 Tipping & Slipping 32
    - 3.3.2.3 Motor Selection 39
- 3.4 Electrical Design 51
  - 3.4.1 Proposed Electrical Design 52
  - 3.4.2 Initial Electrical Analysis 60
  - 3.4.3 Solenoid PCB Design 62
- 3.5 Software Design 66
  - 3.5.1 Package Design 66
  - 3.5.2 Abstraction 69
  - 3.5.3 State Machine Design 70

4 Implementation **72**

4.1 Segmentation of Tasks 72
4.1.1 Gantt Chart 72
4.1.2 Meeting Structure 73
4.1.3 DFMEA 73
4.2 Mechanical Implementation 73
4.2.1 Mobile Base 74
4.2.1.1 Chassis 75
4.2.1.2 Mecanum Wheel Subassembly 76
4.2.1.3 Mecanum Axle 77
4.2.1.4 Drive Train System 78
4.2.1.5 Elevation 81
4.2.1.6 Electronic Housing 86
4.2.2 Modular Insert 87
4.2.2.1 Mounting System 88
4.2.2.2 Universal Plug 90
4.3.3 Home Base Charging Station 93
4.3 Electrical Implementation 95
4.3.1 12V Converter 95
4.3.2 Inverter 96
4.3.3 5V Converter 97
4.3.4 BMS 97
4.3.5 Control Components 98
4.4 Software Implementation 100
4.4.3 Simulations 100
4.4.3.1 Base Simulation 100
4.4.3.2 Arm Simulations 101
4.4.3.3 Insert Simulations 102
4.4.4 State Machine 105
4.4.5 Distributed Processing 106
4.4.6 Robot Actuation 107
4.4.6.1 Mecanum Logic 107
4.4.6.2 Elevator Logic 107

5 Analysis & Validation 108
5.1 Mechanical Analysis 108
5.1.1 FEA Simulations 108
5.1.2 Hand Calculations 110
5.1.2.1 Elevator Alignment System Shoulder Bolt Calculation 110
5.1.2.2 Suspension Spring Calculation 116
5.1.3 Proposed Testing Plans 116

5.1.3.1 Slip/Tip Testing Procedure 117
5.1.3.2 Strain Gauge Testing Procedure 117
5.1.3.3 Suspension Testing Procedure 120
5.2 Electrical and Software Analysis 122

6 Discussion 123
6.1 Mechanical Discussion 123
6.1.1 Chassis 123
6.1.2 Suspension 123
6.1.3 Drivetrain 124
6.1.4 Elevation 125
6.1.5 Electronic Housing 126
6.1.6 Modular Insert 126
6.2 Electrical Discussion 127
6.2.1 Battery Management System 127
6.2.2 Converters 128
6.2.3 Motor Controllers and Motors 129
6.2.4 Sensors 129
6.2.5 Control System 130
6.2.5.1 Raspberry Pi 4B 130
6.2.5.2 PCA9685 PWM Board 130
6.2.5.3 Custom Solenoid Board 131
6.2.6 Modular Insert Wiring 131
6.3 Software Discussion 131
6.3.1 Gazebo Implementation 131
6.3.2 ROS Services 132
6.3.3 State Machine 132

7 Recommendations 134
7.1 Mechanical 134
7.2 Electrical 134
7.3 Software 136

8 Conclusion 137

9 Citations 138

10 Appendices 139
10.1 Use Cases 139
10.1.1 Initial Use Case Ideas 139
10.1.2 Final Use Cases 139
10.2 Bill Of Materials 141

10.3 DFMEA 144
10.3.1 Chassis 144
10.3.2 Elevator Alignment 145
10.4 Programming Examples 146
10.4.1 URDF Example 146
10.4.2 State Machine Example 148
10.4.3 Strain Gauge Code 150
10.5 Robotic Arm Data Sheets 151

Acknowledgements

This MQP would not be possible without the aid of the following sponsors or mentors. We sincerely thank them for their unyielding support and guidance.

- Professor Berk Calli & Professor William Michalson
- Tinkerbox
- FRC Team 190
- FRC Team 78
- MER Laboratory
- European Metal Recycling Group

It was a privilege to be able to work with these people and organizations.

1 Introduction

Robotic applications are growing extensively, recently including the service and logistics industries. Indeed, robotic arms can increase safety, cut costs, improve accuracy, and provide higher flexibility. Serial robotic manipulators have been extensively studied mathematically through control theory and linkage relationships. Devin Hartenberg Parameters were developed in 1955 to relate the end effector, the end point of the manipulator, to any configuration of the joints. More recently, the Product of Exponentials method has been implemented to solve forward and inverse kinematics.

With the rise of these relationships and computational competency, more advanced robotic arms became available. Three degrees of freedom (DoF) SCARAs (Selective Compliance Assembly Robot Arm) could now be replaced by seven DoF manipulators to perform vastly more complex tasks. Despite first finding themselves in research and manufacturing laboratories, service industries have made strides to include robotic arms. Boston Dynamics' Spot has made headlines in its wide range of uses, from exploring nuclear power plants to research settings from its sophisticated robotic arm and navigation usage.

However, robotic manipulators were designed to be stationary and confined to a workspace dictated by the physical length of their links and range of motion of each respective joint. Additionally, robotic manipulators paired with a mobile base, a platform capable of moving the robotic arm across the environment, such as Spot, are suited for one specific arm and require a significant effort to work with different robotic arms. Thus came the motivation behind this project: to develop and implement a modular and mobile robotic base for serial robotic manipulators. Three objectives were developed to satisfy this goal fully:

1. To provide modularity between different robot arms mechanically, electrically, and programmatically.
2. To add degrees of freedom to the attached manipulator, enhancing its mobility and expanding its workspace.
3. To ensure reliable and robust operation.

Modularity, Mobility, and Reliability will be central themes discussed in the various disciplines of Robotics Engineering, Mechanical Engineering, Electrical Engineering, and Computer Science explored in the Design and Implementation sections.

2 Background

The background section will explore hardware and software relevances necessary to understand the fundamental concepts of robotic arms and mobile bases. Additionally, examples of existing solutions and limitations will be discussed. The robotic arms available for experimentation and interfacing with are introduced as well.

2.1 Hardware

2.1.1 Robotic Manipulators

Serial robotic manipulators are a series of linkages and joints powered by motors. Prismatic joints translate motion linearly, while revolute joints utilize revolutionary motion. Revolute joints can even be superimposed on top of each other, allowing for rotational motion around two independent axes. Figure 1 demonstrates a simplified RRRRR (revolute, revolute, revolute, revolute, revolute) manipulator diagram.

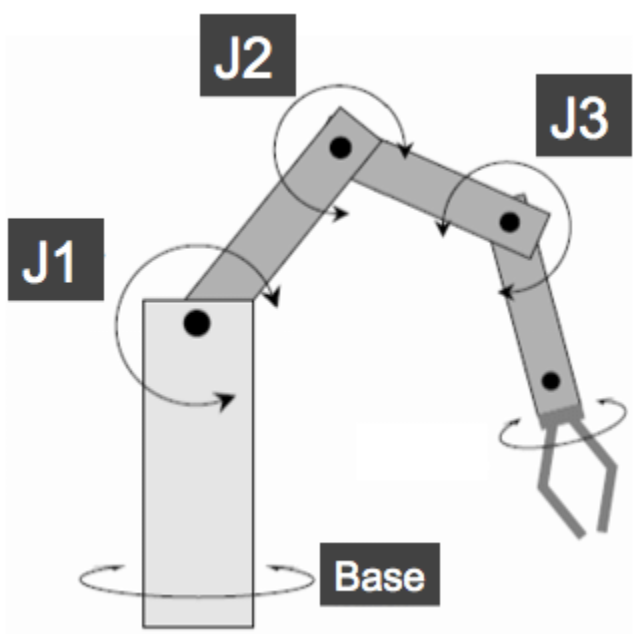


Figure 1: Simplified diagram of serial robotic manipulator[1]

[1] Revolute geometry - 21116. Robotpark ACADEMY. (2015, January 8). Retrieved April 28, 2022, from http://www.robotpark.com/academy/revolute-geometry-21116/

Here, three joints, J1, J2, and J3 rotate about the Z-axis (coming out of the page), while the base rotates around the Y-axis and the end effector (EE) rotates about the last linkage's direction. The EE, or gripper in this model, can be localized to the base frame through a series of transformational matrices such as Devenhen-Hartenberg (DH) parameters or Product of Exponentials. There is enormous benefit to understanding the relationship of the EE's location in reference to the base frame with the joint angles as input parameters. This relationship can be inverted to calculate the joint angles in relationship to a desired EE location. This forms the basis of forward and inverse kinematics and is the fundamental mathematical relationship that allows robotic manipulators to move to an expected location. Additionally, with each joint added to the robotic arm, another degree of freedom is achieved. Seven degrees of freedom have become the standard in the commercial robotic industry despite only six being necessary for complete travel in 3D space. The additional degree of freedom allows for more intricate positioning and increases the available workspace.

Most advanced manipulators will also require a controller to utilize the arm in a coordinated fashion. These controller boxes are specialized computers designed to work specifically with the robot arm. Figure 2 illustrates the controller associated with the Franka Emika Panda arm.

Figure 2: Franka Emika Panda with controller (right)[2]

[2] Franka Emika panda. Active Robots. (n.d.). Retrieved April 28, 2022, from https://www.active-robots.com/franka-emika-panda.html

It is standard for each controller to be provided with wall power (120V AC) and an ethernet connection to a host computer. Additionally, each controller generally supports a novice level of control through a direct connection with a specified IP link. The Franka Emika Panda, for example, is capable of creating and repeating simple user-defined tasks without ever having to program or understand complex mathematical relationships [8].

Mobile bases exist in a wide variety of designs to achieve various goals. All mobile bases, however, work to achieve mobility in some fashion. Some instances work to pass aggressive terrain, such as extraterrestrial bodies or deep underground mines, while others are designed to pass through warehouses and commercial residences. Indeed, some mobile bases work with a specific arm integrated, as seen in Figure 3, or with an empty top shelf with various mounting holes. Generic "one and done" bases were of particular interest as these bases were created with the intent of utilizing serial robotic manipulators.

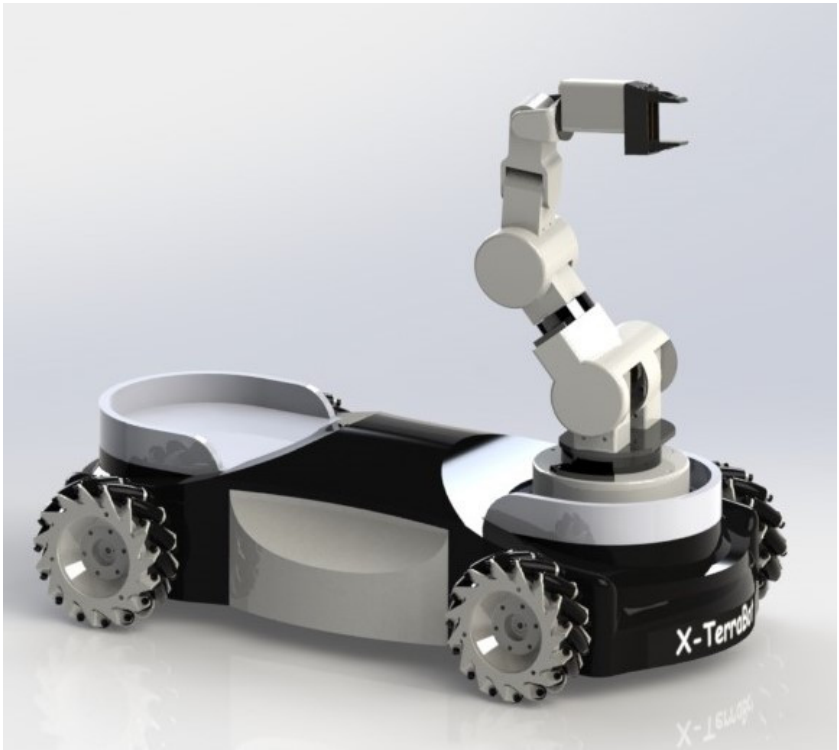


Figure 3: Generic "One and Done" mobile bases by X-Terrabot[3]

In these designs, common practices included a holonomic drive system fitted for indoor use only. Holonomic drive enables the most versatile and maneuverable drive system. The design above utilizes a

[3] *X-Terrabot*. robots.ros.org. (n.d.). Retrieved April 28, 2022, from https://robots.ros.org/x-terrabot/

mecanum wheel drive system where each individual wheel is composed of small rollers placed at a 45-degree angle. Moreso, the robot arm was typically placed asymmetrically closest to one end of the mobile base, enabling the arm to pick and place objects from the floor and reach objects it otherwise could not have had it been placed in the center.

These “One and Done” bases also range significantly in size, with the KUKA Moiros being some twenty feet long.

Figure 4: KUKA Moiros cleaning a commercial aircraft[4]

However, the general design remains the same as the X-Terrabot, as it utilizes mecanum wheels for holonomic drive, and the robot arm is positioned towards one end of the base for an increased reach. This particular model also must be heavy in the backend to counteract the moment induced by the robotic arm.

One major drawback to these models is that they were designed with the integration of a specific robotic manipulator in mind, so using a different robotic arm would not be natively supported or recommended by any company. Opening up the internal electronics necessary to house a nonstandard robotic arm would void the warranty and without guaranteeing success. Proprietary software inside would

[4] *Kuka moiros - YouTube*. (n.d.). Retrieved April 28, 2022, from https://www.youtube.com/watch?v=p6NwH3G0V6Y

also create a logistical challenge to interface without a natively supported manipulator. Thus, a modular interface would best solve these problems.

2.1.2 Modular Hardware

Modular designs for hardware interfaces are far from a new practice. CNC machining utilizes collets to house a variety of bits to perform their specialized task. Even simple tools, such as a hacksaw and various blades, can use different pieces to perform separate but related tasks. The common theme among all of these elements is a standard connection protocol; while each bit for the CNC machine is different, the collets that fit into the head of the machine are identical.

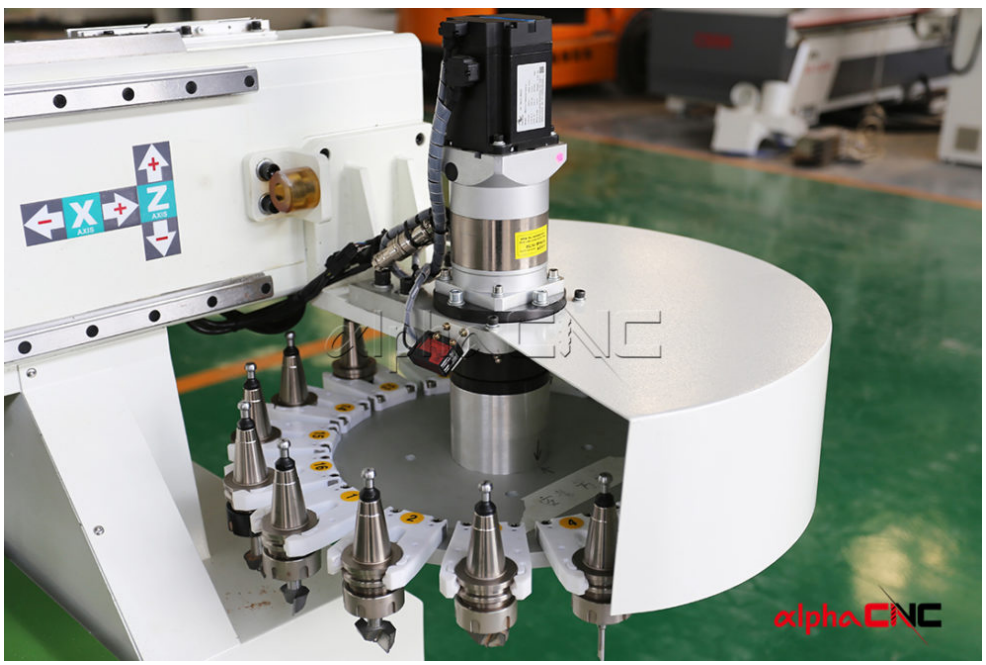


Figure 5: CNC carousel[5]

The idea of modular hardware interfaces was utilized as the basis for designing an interface for a modular base and various robotic arms. The carousel can swap independently functioning bits into the CNC mill seamlessly through the common hardware medium.

[5] ATC wood CNC router 1325 ATC woodworking auto tool changer with 16-position Rotary Tool Carousel SYNTEC 6MD bus communication - CNC routers, CNC lasers: CNC Router Machines, CNC Laser Machines: Alpha Cnc. CNC Routers, CNC Lasers | CNC Router Machines, CNC Laser Machines | Alpha CNC. (n.d.). Retrieved April 28, 2022, from https://alphacnc.com/product/atc-wood-cnc-router-1325-atc-woodworking-auto-tool-changer-16-position-rotary-tool-carousel-syntec-6md-bus-communication/

2.2 Software

The space for modular robotic arm software is currently minimal. Many companies that make robotic arms also include software for controlling their robot, but that software is limited to their specific arm. For example, the Franka Emika Desk is a web-based robot programming platform explicitly designed for products in the Franka Emika line [8]. Despite its inherent ability to program different Franka robot arms, it does not work with other companies' arms. In order to control multiple arms, some intermediate controller would have to exist and contain specialization for each respective arms protocol.

2.2.1 Modular Frameworks

Several attempts have been made at creating standards for interacting with robotics frameworks. One such attempt, RobotUI, describes an architecture for modularizing various robots' user interfaces and control frameworks [3] [9]. Its solution entails combining robotic-specific framework interface components and reusable UI components to present the user with a uniform view of robot information while hiding framework connection details under the hood. Another project that breaks up its software behaviors into modular components is OROCOS (Open RObot COntrol Software) [2] [11]. OROCOS's primary focus is to provide a toolchain that includes a Real-Time Toolkit for building real-time applications in C++ and the OROCOS Component Library for quickly setting up applications with prebuilt components. Unfortunately, these projects have been deprecated recently, as neither has received updates since 2015.

Furthermore, though these projects both present solutions for controlling different robots using the same interfaces, they don't incorporate solutions for robots that are modular themselves, i.e. controlling a base that can have different arms. In order to incorporate intra-robot modularity, the framework for controlling the robot itself had to be modularized directly. To accomplish this ROS was chosen as the main framework.

2.2.2 ROS

ROS, or the Robotic Operating System, is a collection of frameworks for building software for robots [1]. It is separate from most other software platforms in that its primary design centers around nodes, which are independent processes that are connected at runtime. Distributing processes into separate nodes allows developers to reuse code multiple times and organize their nodes how they want, usually into packages that connect to other packages. Another benefit of ROS is that it is extremely lightweight; it

requires very little extra processing power to connect nodes and packages together, so it saves on power consumption and scales up on more extensive projects very easily. Finally, ROS is language-independent, so two nodes written by different developers in C++ and Python can still exchange information seamlessly. These features combined make a very flexible and powerful tool for writing robot software.

As the industry standard for controlling robotic arms and the leading robot control software taught at WPI, ROS was the obvious choice for this project [12]. The Noetic distribution was utilized as it is the most up-to-date distro and includes support for Python 3, which is the most up-to-date version of Python [4].

2.3 Robotic Arms at WPI

This section provides a brief summary of the robotic arms that WPI has in their inventory, along with some of each arm's essential characteristics, which will be relevant to the design of the mobile base. It is important to note that while the robotic base is designed to interface with a wide variety of arms, it will be tailored to suit the arms that WPI currently owns. This will allow the robot to easily integrate with what is currently available to the team, provide a proof of concept for a modular robotic base, and eventually be extended to allow the base to accept more robotic arms.

WPI currently has three robotic arms in its inventory: the Franka Emika Panda, the Universal Robotics UR5e, and the Kinova Gen3. These three arms are medium-sized robotic arms with comparable sizes, masses, and capabilities. These arms are primarily used for assistance in labs, pick and place operations, and may be used as service robots to help people who require assistance picking up and interacting with objects. Figure 6 shows the three arms below.

Figure 6: Franka Emika Panda, UR5e, and Kinova Gen3 robot arms, respectively[6][7][8]

The characteristics of each arm are critical design parameters when creating a modular robotic base that fits all three. The complete set of characteristics for each arm is too long to include in the report, so the important characteristics relating to the mobile base design are highlighted below. Hyperlinks to arm specific datasheets are included in Appendix 10.5.

The crucial characteristics for each arm with regards to the design of the mobile robotic base can be summarized as follows:

1. Mass of the arm
2. Maximum reach of the arm
3. Degrees of freedom of the arm
4. Mass of the controller for each arm
5. Dimensions of the controller for each arm
6. Power consumption of the arm and controller
7. Maximum payload each arm can carry
8. Voltage and power consumption of each arm and controller
9. Communication protocols for each arm

[6] *Franka Emika Panda - Pomo Robotics*. POMO Robotics - Robotic Automation & Industry 4.0 Soluton Provider. (2022, January 26). Retrieved April 28, 2022, from https://www.pomorobotics.com/robots/frankpanda/

[7] *Universal Robots UR5E*. Unchained Robotics. (2021, September 20). Retrieved April 28, 2022, from https://unchainedrobotics.de/en/products/robot/cobot/universal-robots-ur5e/

[8] *Discover our gen3 robots*. Kinova. (n.d.). Retrieved April 28, 2022, from https://www.kinovarobotics.com/product/gen3-robots

Each arm comes with a controller, which translates commands from a computer into the necessary voltages to move the arm. The masses of the arm, controller, and maximum payload are essential for the mechanical design of the chassis, as they impact the geometric relationship of parts and weight distribution of the mobile base to prevent tipping. The number of degrees of freedom of the arm relates to how many movable joints the robot arms have. The maximum reach of the arms impacts how low the arms must be to reach objects on the ground, as well as where the center of mass must be to prevent tipping. The power consumption and required voltages for each arm will impact how the power management system of the base is designed to power each arm when it is connected to the base. Finally, the communication protocols for each arm and controller impact how the microcontroller onboard the base must communicate with the arms and their controllers. The values for each arm are documented in Table 1.

Arm	**Arm Mass**	**Arm Reach**	**DOF**	**Cont. Mass**	**Cont. Dims**	**Typical Power Consump.**	**Max Payload (full ext.)**	**Robot/ Control Power Method**	**Comms**
Kinova Gen3	7.2 kg	902 mm	6	Built into base	Built into base of arm	36W	2kg	24V DC/ 120Vrms 60Hz with adapter	Wifi/BT/USB / Ethernet
Panda	18 kg	855 mm	7	7 kg	355x 483x 89 mm	300W	3kg	120Vrms 60Hz	Ethernet
UR5e	20.6 kg	850 mm	6	13.6 kg	475x 423x 268 mm	200W	5kg	120Vrms 60Hz	Ethernet/USB

Table 1: Characteristics of each robotic manipulator in WPI's possession accessible for use

3 Design

3.1 Use Cases

Before designing the mobile base, now referred to as the robot, the team brainstormed potential use cases to inform some basic design requirements and goals. The most general use cases are as follows:

The robot is currently stationed at location A. The robot is commanded to pick and place an item at location B. It must traverse to location B and then find the item of interest. Finding the item of interest, it should pick it up and wait for further instruction.

The mobile base detects through an interrupt that it has a low battery. It stops its specific task at a safe time and location and proceeds to drive to a charging station.

The Kinova Gen3 arm is replaced with the Panda arm. The connector on the insert is plugged into the connector on the base. The robot waits for instruction.

These use cases outline 'pick and place' operations with a mobile base component, self-charging, and modularity components. With these elements in mind, a problem statement was generated as follows.

Current robotic mobile bases are too expensive and require significant effort to integrate multiple robotic manipulators. The Robotic Mobile Base MQP team needs to design and implement a mobile robotic base that is modular to various robotic manipulators.

3.2 Initial Parameters and Design Goal

With the above problem statement identified, three significant objectives were created to satisfy this goal:

1. To provide modularity between different robot arms mechanically, electrically, and programmatically.
2. Add degrees of freedom to the attached manipulator, enhancing its mobility and expanding its workspace.

3. To ensure reliable and robust operation.

Objective 1 illustrates the modularity component of the project by recognizing that a modular interface must exist in the three disciplines of robotics: mechanical, electrical, and programming. These elements, when combined, will create an interface that mechanically fits into place, is electrically powered, and programmatically moves in expected ways. Objective 2 enables movement of the mobile base, describing a foundation to increase maneuverability and versatility. Components such as elevation and a holonomic drive system were designed and integrated from this objective. Objective 3 works to promote a safe robot that works in a way that is repeatable, expected, and cautious in its actions. These objectives lay the groundwork for the decisions made in the following design sections.

3.3 Mechanical Design

The mechanical design objectives can be summarized as the following:

1. Mobility Objectives
    a. Create a holonomic drive system
    b. Be able to traverse a standard handicap ramp without slipping or tipping
    c. Minimizing tipping from the robotic arm in an extended position
    d. Be able to have a working height of a standard countertop
    e. Being able to fit through the minimum standard doorway
2. Modularity Objectives
    a. Create a system that serves as an intermediate station to marry the robotic arm and the robotic base
    b. Develop a system to lock and unlock the intermediate system from the mobile base
    c. Have alignment features to enable easy placement of the intermediate system with the mobile base

3.3.1 Initial Mechanical Design

3.3.1.1 External Factors

Some initial parameters must be outlined based on the working environment itself. One of the primary decisions was to limit the width of the robot to be able to fit through a standard doorway. This would allow for easy traversal in any building, regardless of application. Since the minimum opening for

a doorway must be no less than 30 inches in width, this made the design parameter a max width of 28 inches with clearance.

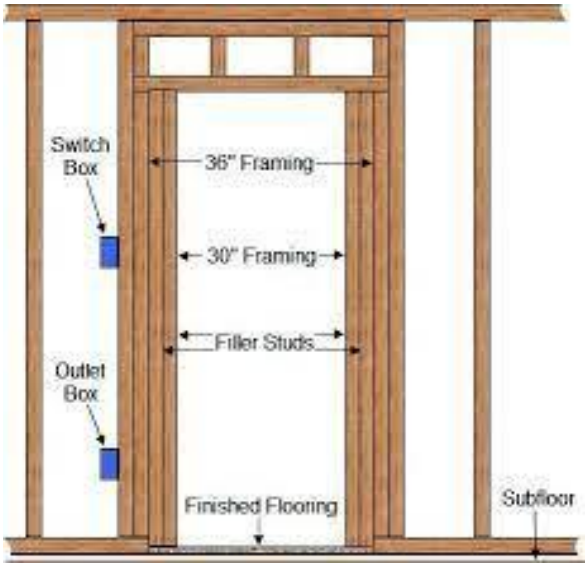


Figure 7: Standard doorway dimensions[9]

Another one of the environmental parameters that drove the mechanical design was the height at which the elevation of the robot could reach. For this, an assumption was made that the lowest point should allow the robot to pick up items from the floor, and the highest point should have the working height of a standard countertop. This determined the requirement to raise the height of the top surface of the robot to the height of a standard countertop, 36 inches high.

[9] Building for future accessibility - doorways. (n.d.). Retrieved April 28, 2022, from https://www.renovation-headquarters.com/accessibility-future.html

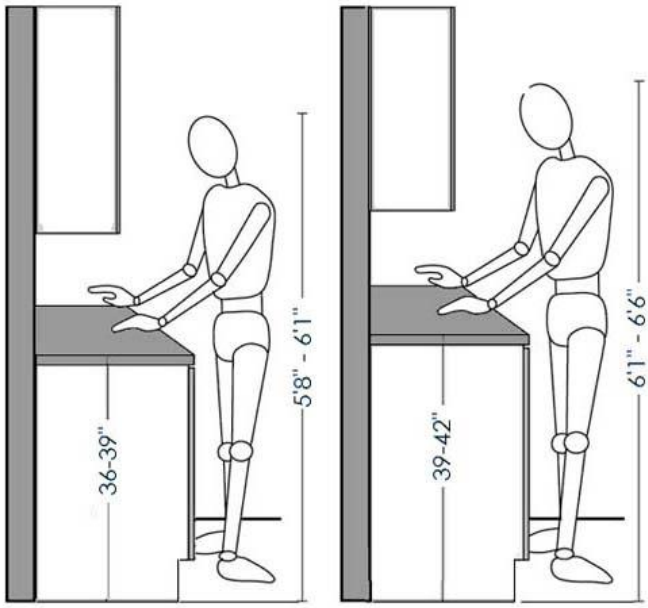


Figure 8: Standard countertop dimension[10]

3.3.1.2 Mobile Design Factors

There were several different form factors for chassis structure that were examined. They can each be described by their cross-sectional pattern. C-channel, Square tubing, Rectangular tubing, and aluminum extrusion (80/20 brand extrusion) were the primary forms considered as the general structure of the frame. Aluminum was chosen initially for its price and ease of use over steel. The aluminum extrusion was the initial choice for the frame due to the ease of manufacturability and assembly. However, due to the excessive cost of the hardware involved, such as proprietary t-nuts and brackets, this idea was not pursued. C-channel was also removed from the options as the assembly process would be complex, and the shape would cause increased bending under torque. This led to the decision between square and rectangular tubing.

[10]https://www.regattaexports.com/kitchen-ergonomic-justifying-36-inches-of-your-kitchen-countertop-height/

Figure 9: Aluminum rectangular tubing[11]

The two profiles were simulated with a rough approximation of the robot's final weight, which steered the decision to utilize 1/16th inch rectangular tubing. This allowed for space to mount all of the various components and increase rigidity in a vertical loading scenario. Rivets were utilized to fasten the box tubing for ease of assembly. A bolt and nut fastening strategy would be cumbersome and expensive, so it was avoided.

The drive system had to be determined to satisfy part of the mobility objective. Mecanum wheels were a free option given to the project and thus were utilized for the entire duration of this project. Additionally, other drive systems were analyzed in a decision matrix, as seen in Table 2 below. Mecanum wheels were still ranked as the ideal drive system.

	Holonomic				**Non-holonomic**		
Drive Criteria	**Mecanum**	**Kiwi**	**H-Drive**	**Swerve**	**Treads**	**Tank Drive**	**Ackerman**
Friction	4	3	4	4	1	2	3
Manufacturing	5	5	4	4	2	3	1
Accuracy/Repeatability	4	4	3	5	3	3	1
Price	2	3	3	1	2	4	4
Power Req.	4	2	4	1	2	4	4

[11] *6063 t52 aluminum rectangular tube*. Midwest Steel & Aluminum. (n.d.). Retrieved April 28, 2022, from https://www.midweststeelsupply.com/store/6063aluminumrectangulartubing

Stability	3	3	4	4	5	5	4
Reliability	4	4	4	1	2	5	5
Complexity	5	3	4	0	4	5	3
6	24	18	24	24	6	12	18
9	45	45	36	36	18	27	9
10	40	40	30	50	30	30	10
7	14	21	21	7	14	28	28
5	20	10	20	5	10	20	20
8	24	24	32	32	40	40	32
8	32	32	32	8	16	40	40
6	30	18	24	0	24	30	18
Score	229	208	219	162	158	227	175

Table 2: Drive ranking chart

Figure 10 portrays the holonomic drive capabilities of a four-wheel mecanum drive system. Each blue arrow represents a clockwise or counterclockwise rotation of an individual wheel to produce a final direction of travel, as seen with the red arrow. This motion is capable by mecanum wheels utilizing a series of passive rollers pitched at a 45-degree angle, as seen in Figure 11.

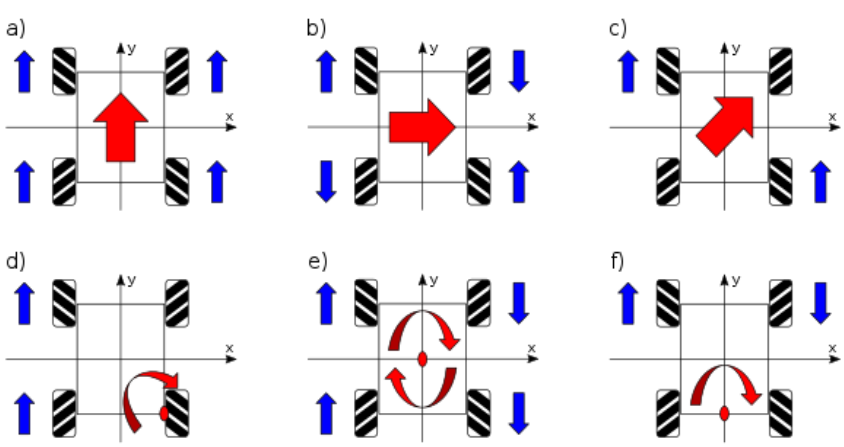


Figure 10: Mecanum wheel direction possibilities[12]

Figure 11: Mecanum wheel[13]

The drive system was also designed to utilize a braking mechanism actuated by a window motor and bike brake calipers. The window motor would compress the brake calipers around a rotor to provide a frictional force to stop the motion of the drive system. The intent was to be able to save power on inclines and promote safety.

The design of the elevation component went through several iterations before settling on a design. Initially, a chained elevator was investigated due to the simplicity of design and the team's prior experience with building it. However, through research into more cost-effective alternatives, automotive scissor jacks were simpler, stronger, and more accessible. They additionally provide excess lifting capability and would enable a simple motor implementation.

[12] Wikimedia Foundation. (2022, March 28). *Mecanum wheel*. Wikipedia. Retrieved April 28, 2022, from https://en.wikipedia.org/wiki/Mecanum_wheel

[13] *96mm mecanum wheel set (70A durometer bearing supported rollers)*. goBILDA. (n.d.). Retrieved April 28, 2022, from https://www.gobilda.com/96mm-mecanum-wheel-set-70a-durometer-bearing-supported-rollers/

The scissor jacks seen in Figure 12 are capable of vertical elevation by actuating the center lead screw causing the jack to elevate. The vertical load these jacks can handle is within the range of 500-10,000lbs, which is far in excess of what they would experience through general use.

Figure 12: Scissor Jack[14]

3.3.1.3 Modular Design Factors

The initial concept for a modular system was a modular insert. The modular insert combines the controller with the robotic arm. There would exist one modular insert for every type of arm such that every robotic arm would have a unique housing requirement. Combining these two systems as a singular unit enables an intermediate mechanical system to house them and interface with the mobile robotic base. This connection would be standard among all types of modular inserts, so any robotic arm would be able to fit within the mobile base. Indeed, the largest controllers would dictate the worst-case scenarios and be the tightest fit. In this project, the UR5e's controller was the largest, and thus the mobile base and modular insert were designed with this particular controller's dimensions in mind. Figure 13 shows this initial design concept.

[14] Murphy, G. (n.d.). *Leveling scissor jack - 24" lift - 5,000 lbs 1 single Jack*. RV Parts Nation. Retrieved April 28, 2022, from https://www.rvpartsnation.com/rv-towing-driving/rv-jacks/leveling-scissor-jack-24-lift-5-000-lbs-1-single-jack/

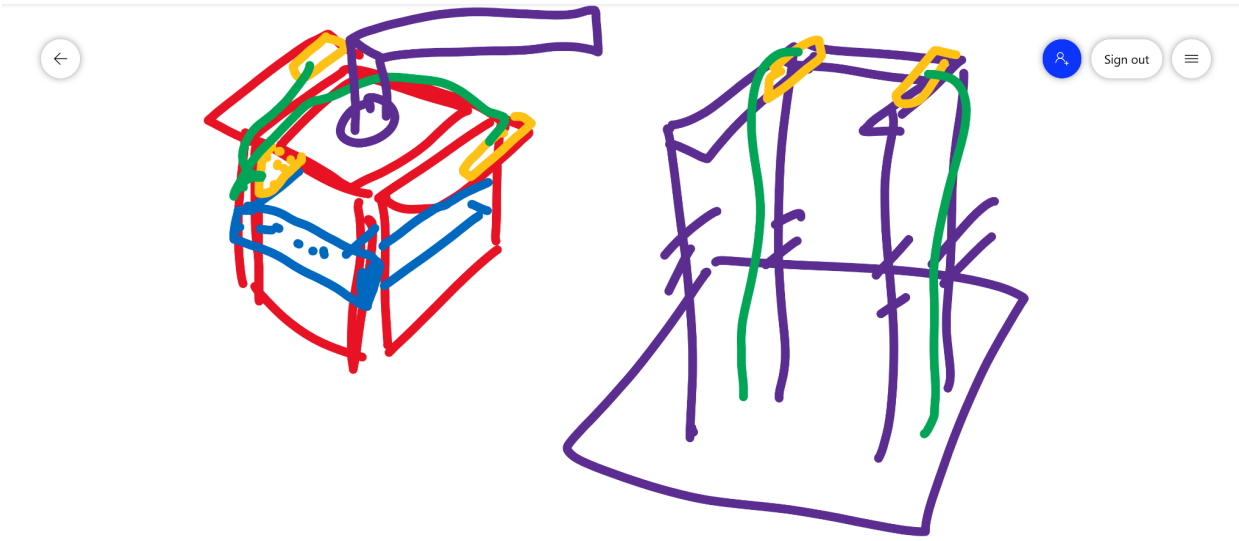


Figure 13: Initial design concept for the modular insert (left) and mobile base (right)

The Modular Insert consists of the blue controller, purple robotic arm, red physical housing, a yellow standard electrical interface, and green wires. The mobile base consists of a purple structure, yellow electrical interface and green wires. The hardware interface between the modular insert and the mobile base would be standard among all modular inserts such that they can be replaced with different robotic arms and controllers.

Grouping these concepts, an initial sketch, as seen below, was produced.

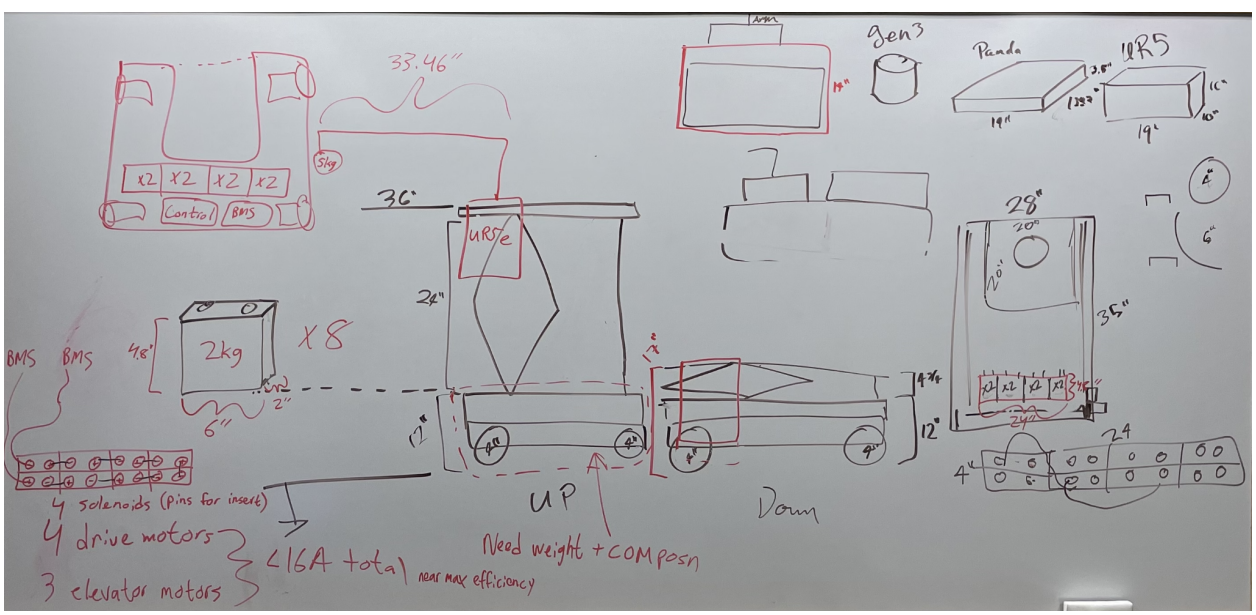


Figure 14: Initial sketch of combined ideas

A mixture of the design concepts for the insert, scissor jacks, and sizing of the base was introduced to help visualize how they may all fit together. Three applicable controllers are dimensioned in the upper right, which was used to determine the sizing requirement of the modular insert concept.

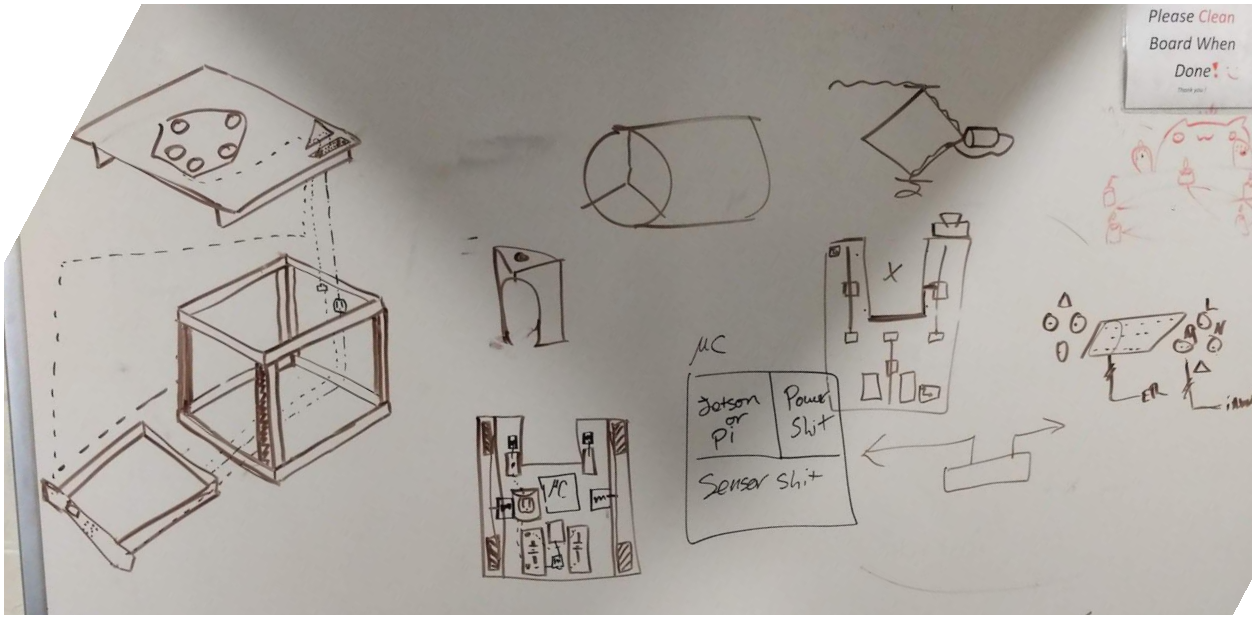


Figure 15: Hardware placement and modular insert sketches

Figure 15 shows a more in-depth modular insert with the potential to utilize server racks as the frame to build upon, as seen on the left. This also shows the mounting plate for the arm and the potential location for the plug. Basic layout principles are demonstrated on the bottom and right, where the scissor jacks reside relative to other components. In the center is a fundamental concept for a preliminary misalignment system.

3.3.1.4 Home Base

To aid the objective of swapping manipulators autonomously, the concept of a stationary home base was proposed. The reasoning behind this base was to provide a space for the modular insert to rest, which was still accessible to the robot-- leading to the potential ability for the robot to drop off an insert at a base and proceed to another with a different robotic arm. The concept is straightforward and allows for

an added benefit of a location for the robot's charging station. Due to the concept including a place for the manipulators to rest and a location for charging, the idea was dubbed the home charging station or the home base.

3.3.2 Initial Mechanical Analysis for Advanced Design

When designing a robot, it is important to first do a mathematical analysis of the proposed solution to both verify that the design is viable and to determine specifications for various subsystems such as motor selection, gearbox design, and current draw. This section will go into detail regarding the mathematically driven design of the robot. First, it will discuss the derivation of the center of mass of the robot and how the choice of center of mass will optimize robot performance. Secondly, it will discuss the operation of the robot at its extremes and verify that the robot will be able to safely transport the arms. Before beginning the calculations, it will be important to define some variables on the robot. Figure 16 and Table 3 below will summarize these definitions.

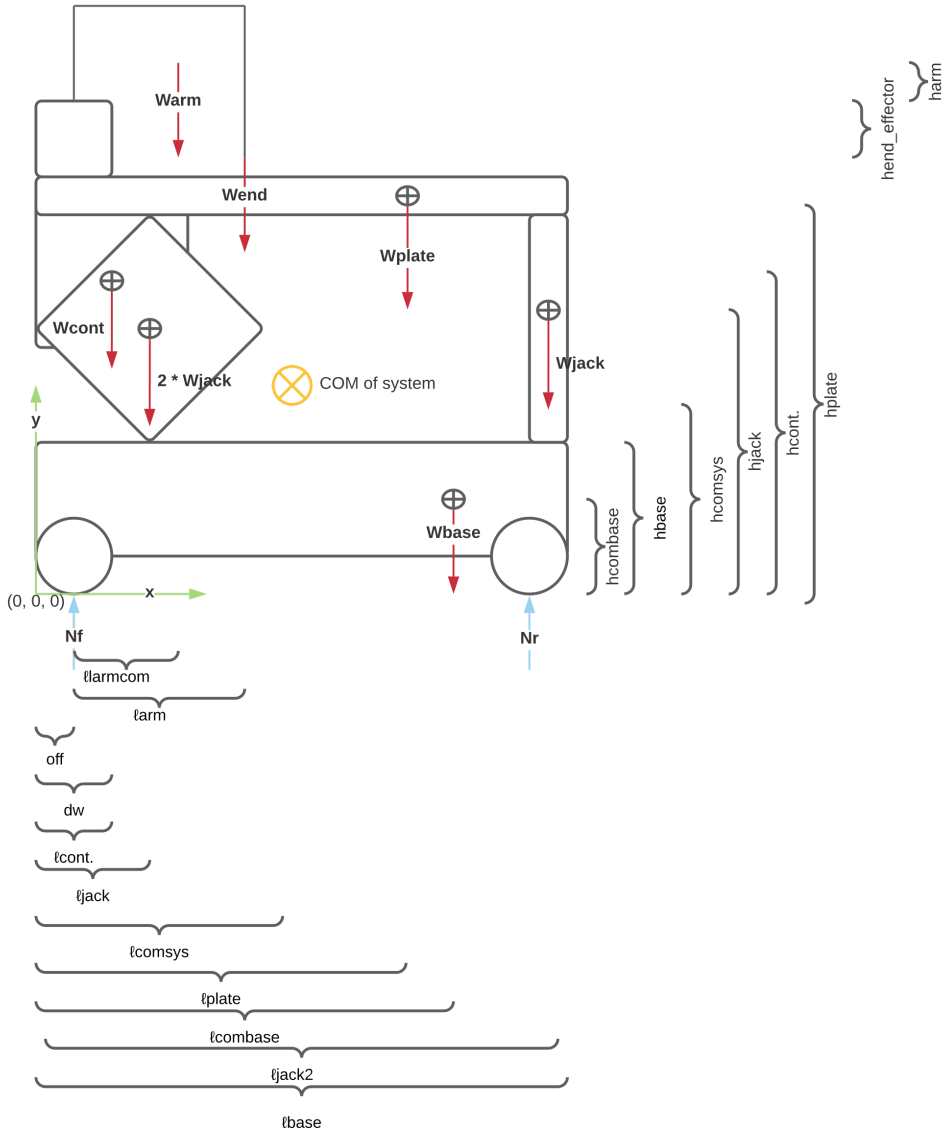


Figure 16: Mockup of robot with variables

Parameter	Description
(0,0,0)	Represents the origin of the system
larm	X coordinate distance from the base of the robot arm to the end effector

larmcom	X coordinate distance from the base of the robot arm to the center of mass of the arm
Off	X coordinate distance from the origin to the base of the arm
dw	Diameter of the wheels
lcont	X coordinate distance from the origin to the center of mass of the robot controller
ljack	X coordinate distance from the origin to the base of the pair of scissor jacks in the front
lplate	X coordinate distance from the origin to the center of mass of the plate on top of the jacks
lcombase	X coordinate distance from the origin to the center of mass of the mobile base
ljack2	X coordinate distance from the origin to the base of the rear scissor jack
lbase	X coordinate distance from the origin to the end of the mobile base, the length of the mobile base
lcomsys	X coordinate distance from the origin to the center of mass of the entire system
hcombase	Y coordinate distance from the origin to the center of mass of the mobile base
hcomsys	Y coordinate distance from the origin to the center of mass of the entire system
hbase	Y coordinate distance from the origin to the top of the mobile base, the height of the base
hjack	Y coordinate distance from the origin to the center of mass of the front scissor jacks
hcont	Y coordinate distance from the origin to the center of mass of the robot controller
hplate	Y coordinate distance from the origin to the center of mass of the plate on top of the jacks

harm	Y coordinate distance from the base of the robot arm to the center of mass of the robot arm
hend_effector	Y coordinate distance from the base of the robot arm to the end effector
zcombase	Z coordinate distance to the center of mass of the base
zcomsystem	Z coordinate distance to the center of mass of the entire system
tbase	Z coordinate distance from the origin to the other side of the mobile base, the width of the base
Wbase	Weight of the mobile base
Wjack	Weight of one scissor jack assembly
Wplate	Weight of the plate on top of the jacks
Wcont	Weight of the robot controller
Warm	Weight of the robot arm
Wend	Weight of object being picked up by the robot

Table 3: Mockup parameters and definitions

Additionally, it was assumed that the center of mass in the z direction for each component would be in the middle of each component, and that the mobile base and entire system would be symmetrically designed. This would put both zcomsystem and zcombase at a distance of tbase/2, greatly simplifying the mathematics. The mobile base would then be designed to achieve this center of mass component in the z direction.

Finally, it was important to determine the weights of the robot based on the hardware used. A summary of the components included in the robot's design and their weights based on CAD can be found in the table below. Everything below Jack Motors and Gearing contribute to the mass of the base.

Component	**Mass**
Kinova Gen3 Arm	7.2kg
Kinova Gen3 Controller	0kg
Kinova Gen3 Payload	2kg
Franka Emika Panda Arm	18kg

Franka Emika Panda Controller	7kg
Franka Emika Panda Payload	3kg
UR5e Arm	20.6kg
UR5e Controller	13.6kg
UR5e Payload	5kg
Insert (contributes to Wcont)	14kg
Plate	10.22kg
Alignment System (contributes to Wjack)	2.079kg
Scissor Jacks	20kg
Jack Motors & Gearing	2kg
Drive Motors	5.08kg
Chassis Tubing	33.13kg
Batteries	32.24kg
Wheels	10kg
12V Converter	2.2kg
5V Converter	0.65kg
BMS (Battery Management System)	0.25kg
Inverter	2.05kg
Raspberry Pi	0.025kg
Speed Controllers	0.5kg
Wiring	4kg
Fuses	0.5kg
Sprockets, Tensioner, Braking, Gearboxes	5kg
Base & Kinova Gen3 Mass	153.124kg
Base & Franks Emika Panda Mass	171.924
Base & UR5e Mass	183.124kg
Base Only Mass	95.625kg

Table 4: Masses of various subsystems

3.3.2.1 Center of Mass

The first step of designing the robot was to determine the parameters for the mechanical design of the base, namely, the center of mass. When designing this aspect of the robot, the team decided to make some design choices to both improve the ease of computing the math, and the overall operation of the robot itself. It was determined that during typical operation of the base while driving, the attached arm should be set to a home position. This home position would have the arm contained within the frame of the robot and pointed towards the rear of the robot. This home position would also attempt to minimize the y component of the center of mass by keeping the arm close to the base and keeping the elevator retracted to improve stability of the base. In addition, by doing this, the arm would be protected from hitting obstacles which may damage the arm or throw the base off course. To simplify the following calculations, it was assumed that the center of mass of the arm would be in line with the center of mass of the controller, and the height would be a quarter of a meter above the base of the robot.

After determining the home positions of the robot, the desired center of mass of the base could be calculated. Using the definitions for weights and lengths above, the desired center of mass of the base could be determined which would give the center of mass for the whole system under typical operating conditions. The desired x component center of mass under typical operating conditions was determined to be half of the length of the base, putting it directly in the middle of the base at 0.4445m. This distance was chosen so that the normal forces on the front and rear wheels would be equivalent, allowing the 4 drive motors to operate at the same efficiency and speed when geared correctly. In addition, putting the center of mass in the very center of the system would minimize the likelihood of the robot tipping over on both flat ground and an incline as well.

The formula for the center of mass for the system is $x_{com} = \frac{1}{\sum m}\sum_{i=1}^{n} m_i x_i$ where each m is the mass of a specific component and x is the corresponding x coordinate of its center of mass. For this system, the center of mass computation is given below.

$$l_{comsys} = \frac{1}{W_{system}}(W_{end} * \left(l_{arm} + off\right) + W_{arm} * \left(l_{armcom} + off\right) + W_{cont} * l_{cont} + 2W_{jack} * l_{jack} + W_{base} * l_{combase} + W_{plate} * l_{plate} + W_{jack} * l_{jack2})$$

Using this equation, the x component of the center of mass for the base can be solved for if we know the weight for that base, which was determined before by summing the masses of the components of the base. Once doing that, we can calculate the total weight with the following formula.

$$W_{system} = W_{end} + W_{arm} + W_{cont} + 3W_{jack} + W_{base} + W_{plate}$$

Using this result, the x component of the center of mass can be found with the following equation.

$$l_{combase} = l_{comsys} - \frac{1}{W_{system}}(W_{end} * (l_{arm} + off) + W_{arm} * (l_{armcom} + off) + W_{cont} * l_{cont} + 2W_{jack} * l_{jack} + W_{plate} * l_{plate} + W_{jack} * l_{jack2} - W_{end} * l_{comsys} - W_{arm} * l_{comsys} - W_{cont} * l_{comsys} - 3W_{jack} * l_{comsys} - W_{plate} * l_{comsys})$$

Solving this equation tells us where the center of mass of the mobile base should be designed so that the system's center of mass is directly in the center. Running the calculation for all three arms gave the following values, summarized in the Table 5 below.

Franka Emika Panda COM	.51m
Kinova Gen3 COM	.49m
UR5e COM	.53m

Table 5: Center of masses of mobile base with each arm attached

Choosing a value that was central to those three gave a desired x component of the base's center of mass to be .51m from the front. The team then designed the geometry of the base and laid out its electrical system in an attempt to get as close to this desired value as possible. Upon checking the CAD, it was determined that the actual center of mass was at .5035m from the front of the base which was close to the desired value.

3.3.2.2 Tipping & Slipping

The next step was then to verify that the base would not tip over while on flat ground with this center of mass. To calculate this, each arm was set to its worst-case scenario and the torques on the system were analyzed. At its worst, each arm would be extended straight out while carrying its maximum rated load, as shown in Figure 17 below.

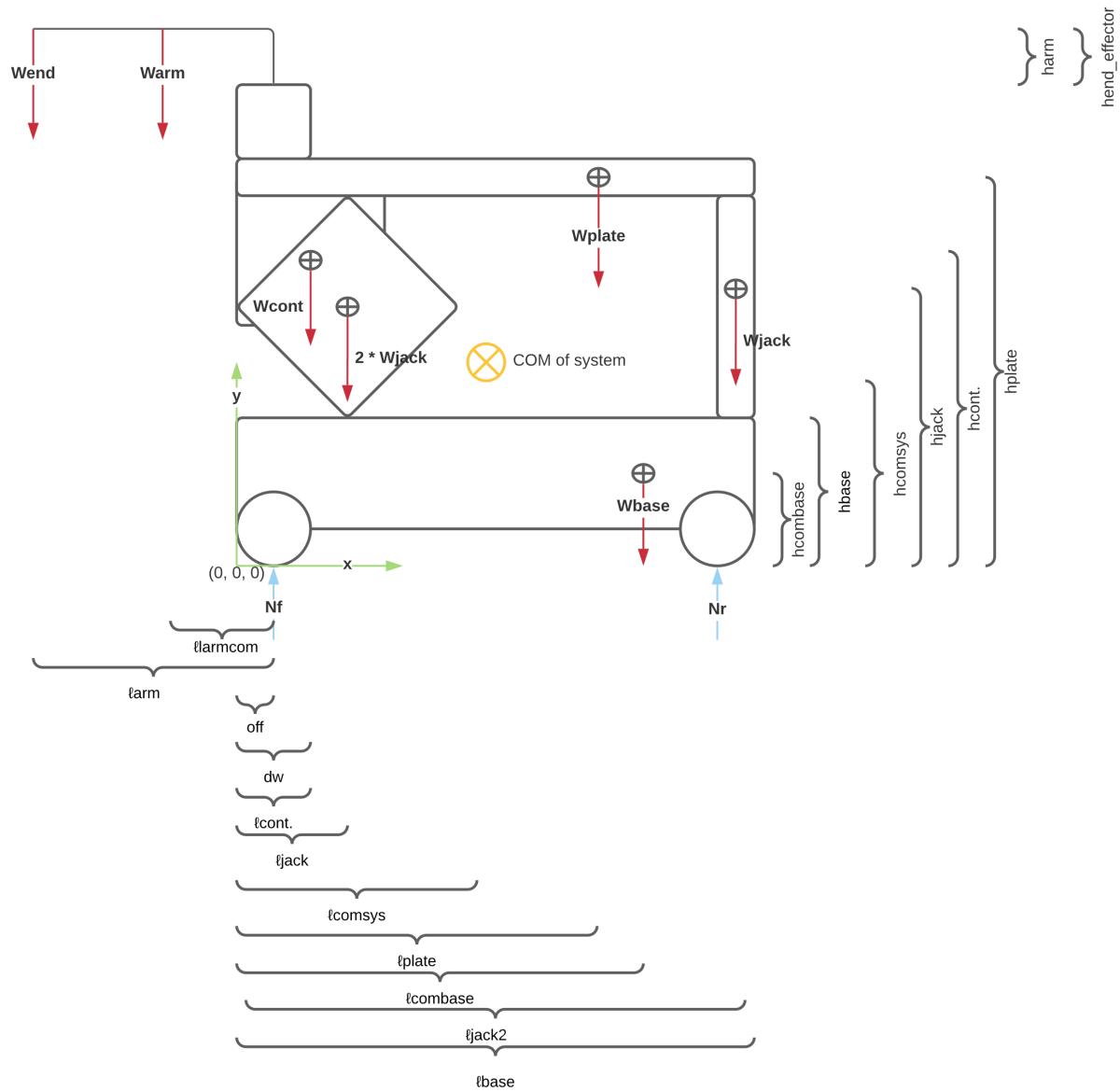


Figure 17: Free body diagram of mobile base under load

The equations of motion were set up as seen below and the moments were summed about the front wheel (which is the point around which the robot would tip). The robot would end up tipping when the normal force on the rear wheel was equivalent to 0.

$$\sum F_x = 0 = 0$$

$$\sum F_y = 0 = N_f + N_r - W_{end} + W_{arm} + W_{cont} + 3W_{jack} + W_{base} + W_{plate}$$

$$\sum M_{frontwheel} = 0 = W_{end} * \left(l_{arm} + off - \frac{d_w}{2}\right) + W_{arm} * \left(l_{arm} + off - \frac{d_w}{2}\right) - W_{cont} * \left(l_{cont} - \frac{d_w}{2}\right) - 2W_{jack} * (l_{jack} - d_w/2) - W_{base} * (l_{combase} - d_w/2) - W_{plate} * (l_{plate} - d_w/2) - W_{jack} * (l_{jack2} - d_w/2) + \tau_{external}$$

The robot would tip when the back wheel begins to leave the ground, so when the normal force of the back wheel was set to 0. In order to get to that point, an external torque would need to be applied and that is what was solved for. If the external torque was positive, the robot would remain stable. If it was negative, the robot would tip when the arm is extended. Solving the torque equation for each arm gave the following formula and results.

$$\tau_{external} = - W_{end} * \left(l_{arm} + off - \frac{d_w}{2}\right) - W_{arm} * \left(l_{arm} + off - \frac{d_w}{2}\right) + W_{cont} * \left(l_{cont} - \frac{d_w}{2}\right) + 2W_{jack} * (l_{jack} - d_w/2) + W_{base} * (l_{combase} - d_w/2) + W_{plate} * (l_{plate} - d_w/2) + W_{jack} * (l_{jack2} - d_w/2)$$

Franka Emika Panda External Torque to Tip	43Nm
Kinova Gen3 External Torque to Tip	47Nm
UR5e External Torque to Tip	42Nm

Table 6: External torque to tip mobile base

This can also be verified by calculating the x component of the center of mass of the system as done before. If the x component at the worst-case scenario remains between the points of contact of the two wheels, the robot will not tip. The results of this following calculation are shown in the table below as well. The center of mass must be between .0762m and .8128m from the front of the robot.

$$l_{comsys} = \frac{1}{W_{system}} (W_{end} * \left(l_{arm} + off\right) + W_{arm} * \left(l_{armcom} + off\right) + W_{cont} * l_{cont} + 2W_{jack} * l_{jack} + W_{base} * l_{combase} + W_{plate} * l_{plate} + W_{jack} * l_{jack2})$$

Franka Emika Panda X COM Arm Extended	.31m
Kinova Gen3 X COM Arm Extended	.38m
UR5e X COM Arm Extended	.29m

Table 7: Center of mass under external torque

The arms are within the required tolerances, so therefore the base will not tip.

When doing the calculations for tip angles and slip angles of the robot, it is easier to first calculate the x component and y component of the center of mass for different situations and then use those values to develop a simplified robot model to determine the angles at which the robot will slip while or tip while

driving up a ramp. It was decided that when driving, the robot would only be in its home position, so the centers of mass for that situation needed to be determined.

Using the following formulas for the x component and y component of the center of mass of the system, the values for each scenario with the three arms were calculated and summarized in the table below.

$$l_{comsys} = \frac{1}{W_{system}}(W_{end} * \left(l_{arm} + off\right) + W_{arm} * \left(l_{armcom} + off\right) + W_{cont} * l_{cont} + 2W_{jack} * l_{jack} + W_{base} * l_{combase} + W_{plate} * l_{plate} + W_{jack} * l_{jack2})$$

$$h_{comsys} = \frac{1}{W_{system}}(W_{end} * \left(h_{end} + h_{plate}\right) + W_{arm} * \left(h_{arm} + h_{plate}\right) + W_{cont} * h_{cont} + 2W_{jack} * h_{jack} + W_{base} * h_{combase} + W_{plate} * h_{plate} + W_{jack} * h_{jack})$$

Franka Emika Panda COM (x,y)	(.41m, .32m)
Kinova Gen3 COM (x,y)	(.42m, .31m)
UR5e COM (x,y)	(.41m, .32m)

Table 8: Center of masses of robotic arms

Now with these center of mass calculations, the slipping and tipping angles for each scenario can be calculated to verify the angles at which the robot would slip or tip while traversing an incline. It is desirable that the robot slip while driving up a steep ramp rather than tip, giving the arms better protection from falling and hitting the ground, damaging the arm. In addition, sensors could be added to the system and integrated with code to prevent the robot from driving up too steep an incline, but this mechanical design aspect would add redundancy to the system and better protect the arms. In order to calculate the angles for this design, the following free body diagram was used. Since the robot centers of mass had been calculated, the system could be represented in a much simpler manner. The origin is defined in the same manner as before where the x component is in line with the front of the base and the y component is in line with the wheels.

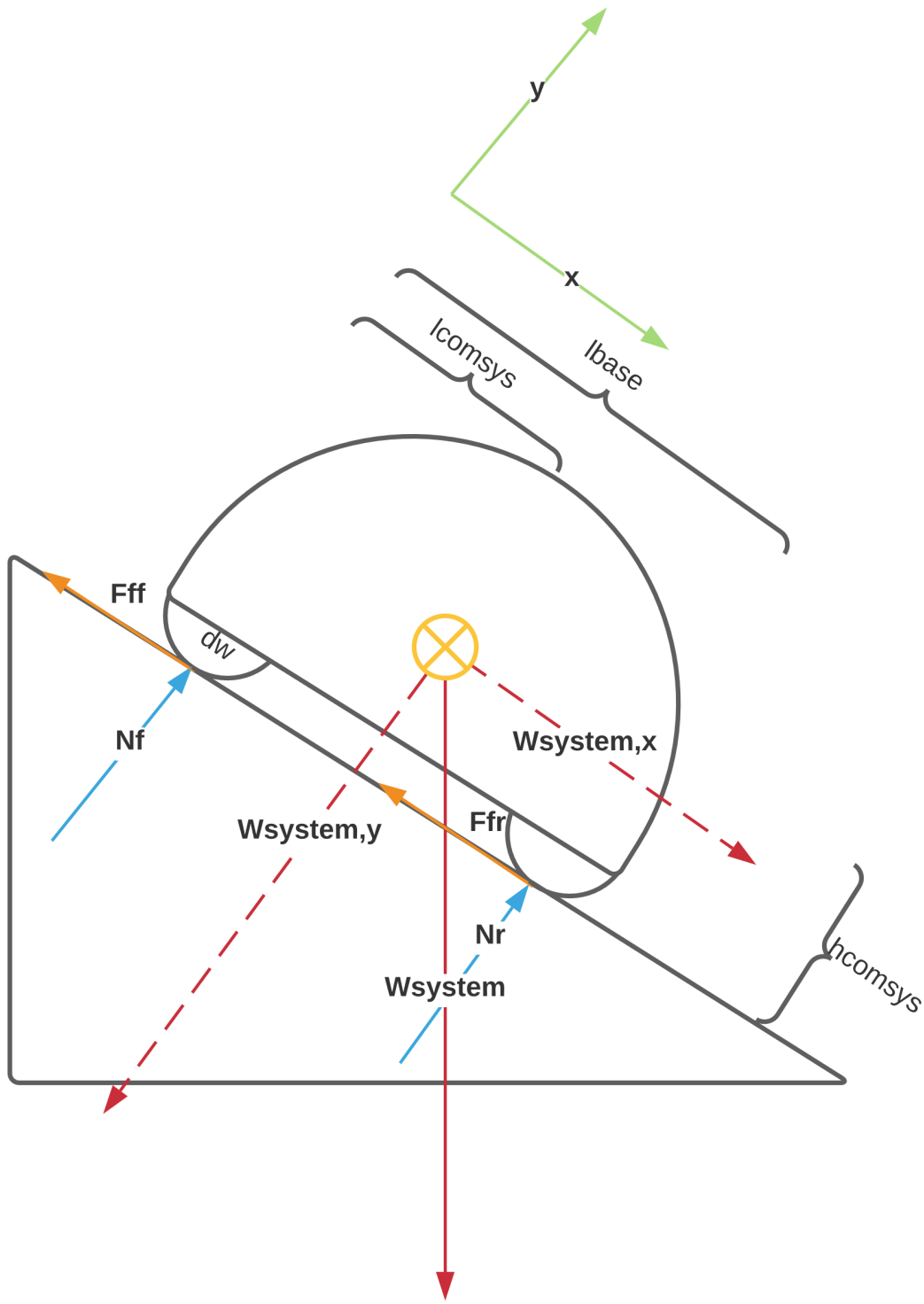


Figure 18: Mobile base on ramp free body diagram

The scenario where the robot tips is when the front wheel leaves the ramp, or when the normal force on the front wheel is equivalent to 0. With that in mind, the equations of motion for this scenario can be seen below. The value of theta is the angle of the ramp, and the moments will be summed around the point of contact of the rear wheel.

$$\sum F_x = 0 = F_{ff} + F_{fr} - W_{system} * \sin\theta$$

$$\sum F_y = 0 = N_f + N_r - W_{system} * \cos\theta$$

$$\sum M_{rearwheel} = 0 = -N_f\left(l_{base} - d_w\right) + W_{system} * \cos\theta * \left(l_{base} - l_{comsys} - \frac{d_w}{2}\right) - W_{system} * \sin\theta * h_{comsys}$$

Rearranging the equations gives the following formula for the tipping angle.

$$\theta_{tip} = \left(\frac{l_{base} - l_{comsys} - \frac{d_w}{2}}{h_{comsys}}\right)$$

The tipping angles for each arm in each orientation are summarized in the table below.

Franka Emika Panda Tip Angle	51 degrees
Kinova Gen3 COM Tip Angle	51 degrees
UR5e COM Tip Angle	51 degrees

Table 9: Tipping angles

The scenario where the robot slips is when the x component of the gravitational force is greater than or equal to the frictional forces applied by the wheels. Since the robot uses mecanum drive, each wheel has a friction force component in the x direction and the z direction, as depicted in Figure 19.

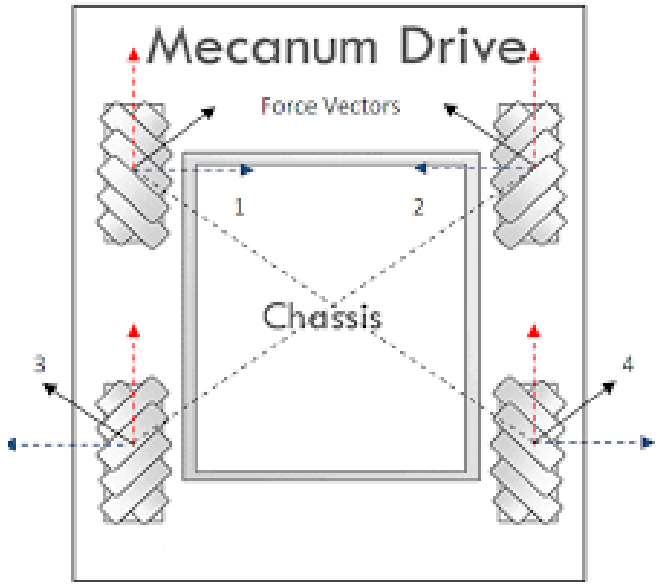


Figure 19: Mecanum drive forces[15]

When static or driving forwards, the z components end up canceling each other out, meaning that the frictional force applied by the front and rear wheels in the x direction to counteract the x component of the gravitational force is $\sqrt{2}$ times less than the total frictional force. The equations of motion for this scenario can be seen below.

$$\sum F_x = 0 = F_{ff} + F_{fr} - W_{system} * \sin\theta$$

$$\sum F_y = 0 = N_f + N_r - W_{system} * \cos\theta$$

$$\sum M_{rearwheel} = 0 = -N_f\left(l_{base} - d_w\right) + W_{system} * \cos\theta * \left(l_{base} - l_{comsys} - \frac{d_w}{2}\right) - W_{system} * \sin\theta * h_{comsys}$$

Recognizing that the frictional forces can be rewritten in terms of the normal force, the sum of x component forces becomes the following equation.

$$\sum F_x = 0 = \frac{\mu_{mecanums}}{\sqrt{2}} * N_f + \frac{\mu_{mecanums}}{\sqrt{2}} * N_r - W_{system} * \sin\theta$$

[15] *Programming Tutorial - Mecanum drivetrain*. Game Manual 0. (n.d.). Retrieved April 28, 2022, from https://gm0.org/en/latest/docs/software/mecanum-drive.html

In addition, recognizing that the robot will slip once the two frictional forces are less than or equal to the x component of the gravitational force and that the sum of the normal forces are equal to the y component of the gravitational force allows for the equation to be rearranged and give the following formula for the slipping angle.

$$\theta_{slip} = (\frac{\mu_{mecanums}}{\sqrt{2}})$$

The slipping angles for each arm in each orientation are summarized in Table 10.

Franka Emika Panda Tip Angle	20 degrees
Kinova Gen3 COM Tip Angle	20 degrees
UR5e COM Tip Angle	20 degrees

Table 10: Slipping angles

Finally, since the team wanted to design the robot so that it would slip before tipping over. To achieve this goal, the following relationship must be met

$$\frac{l_{base} - l_{comsys} - \frac{d_w}{2}}{h_{comsys}} > \frac{\mu_{mecanums}}{\sqrt{2}}$$

In order for this relationship to hold true, the center of mass of the robot should be kept as low as possible. For all three arms, the tip angle is 51 degrees and the slip angle is 20 degrees, meaning that the robot will slip before tipping, protecting the arms. Also, the slip angle is 20 degrees, which is greater than the angle of a handicapped ramp, allowing the robot to traverse that surface as well.

3.3.2.3 Motor Selection

In addition to calculating slipping and tipping angles, the center of mass calculations can be used to calculate the normal forces on each wheel and in turn calculate the required motor torques. The motors will be designed with the typical operation in mind, but the ability to operate at the extremes will be verified as well.

When calculating the motor torques required to drive the robot, it is important to note that it was assumed that the z component of the center of mass of the system would be designed such that it was directly in the middle of the base. In this case, it was 14in away from the origin. By doing this, it could be assumed that the normal force on the front two wheels would be evenly distributed between them, and the

rear two wheels would act in the same manner. Then, setting up the equations of motion for the system depicted below gave the following results for each arm.

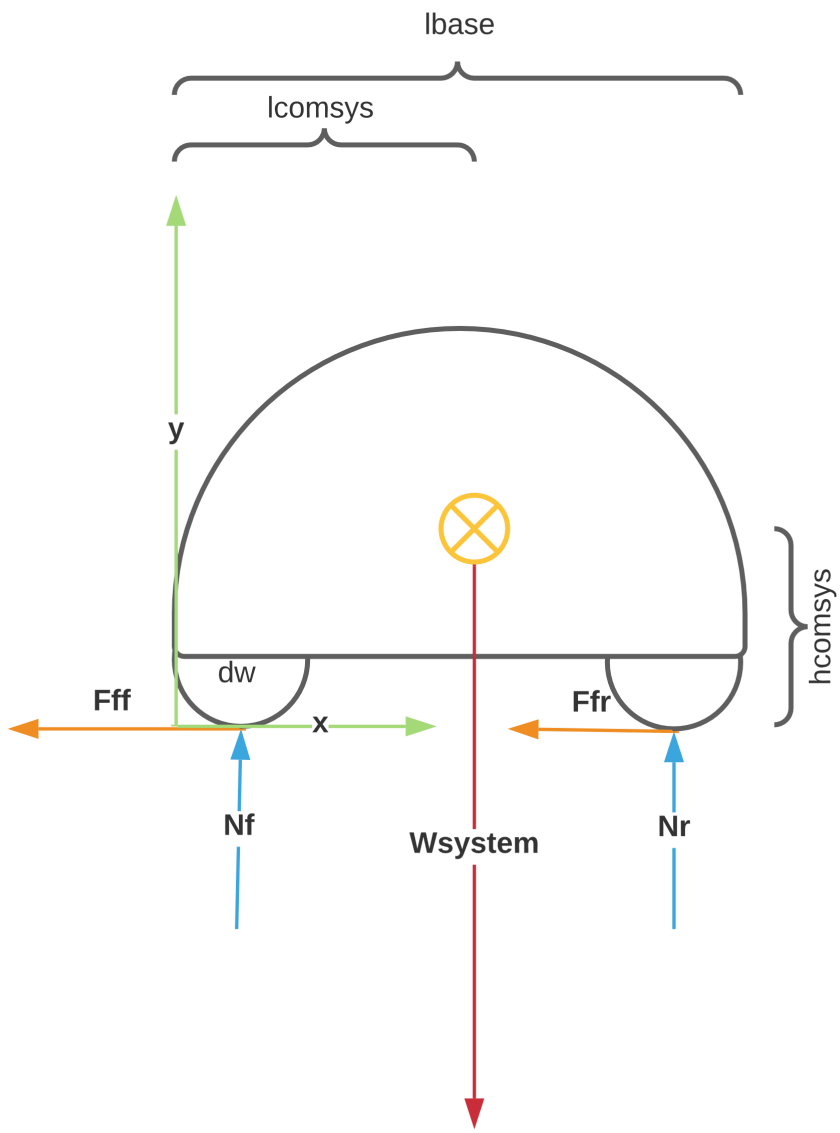


Figure 20: Motor free body diagram

$$\sum F_x = 0$$

$$\sum F_y = 0 = N_f + N_r - W_{system}$$

$$\sum M_{frontwheel} = 0 = - W_{system} * \left(l_{comsys} - \frac{d_w}{2}\right) + N_r * (l_{base} - d_w)$$

Rearranging the moment equation and solving for the normal force on the rear wheel gives the following equation.

$$N_r = \frac{W_{system} * (l_{comsys} - \frac{d_w}{2})}{l_{base} - d_w}$$

Substituting this value into the y forces equation and solving for the normal force at the front wheel gives the following equation.

$$N_f = W_{system} * (1 - \frac{\left(l_{comsys} - \frac{d_w}{2}\right)}{l_{base} - d_w})$$

Using the calculated normal forces, the required torques and gear ratios could be determined to drive the robot at a constant speed. Since this calculation looked at the torque required for one wheel, and it was assumed that the weight would be distributed symmetrically in the z dimension, the normal force at the front right wheel would be equivalent to the normal force at the front left wheel and the normal force at the rear right wheel would be equivalent to the normal force at the real left wheel. The following free body diagram was used to set up the following equations of motion. The torques were summed around the wheel axle, at the center of the wheel.

$$\sum F_x = 0 = \frac{F_{ff}}{\sqrt{2}} - F_{ax}$$

$$\sum F_y = 0 = \frac{N_f}{2} + F_{ay}$$

$$\sum M_{axle} = 0 = \tau_{cimx} - F_{ff} * \frac{d_w}{2}$$

Since the chosen wheels were mecanum wheels, the friction force developed at the base of the wheels would not be perpendicular to the axle; it would be at a 45-degree angle. Thus, the friction force component in the x direction while driving forward would be $\sqrt{2}$ times less than the overall frictional force. Rearranging the equations of motion would allow us to solve for the required output torque to drive the robot which is given by the following equation.

$$\tau_{out} = \frac{\mu_{mecanums}}{4} * N_f * d_w$$

The required motor torques for the three arms can be found in the table below.

Franka Emika Panda	39.7Nm
Kinova Gen3	35.3Nm
UR5e	42.3Nm

Table 11: Required motor torques for each arm

Next, using this torque calculation, the team could determine the required gearing to move the robot at a maximum desired speed. To keep the robot at a controllable speed, the team determined that the robot should drive no faster than 2 feet per second. It was determined that the robot would move the fastest when all four mecanum wheels were spun, meaning that some of the motor torques on each wheel would be translated into frictional forces with components that would cancel, as shown in Figure 19.

Moving at .6m/s means that the required motor power could be determined by multiplying the motor torque by the output speed. This power needs to be $\sqrt{2}$ times larger to account for the fact that the friction forces occur at a 45 degree angle from the wheel since the wheels are mecanum wheels, as seen in the diagram above. 2 feet per second corresponds to a rotational velocity of 8 radians per second. The equation for the power required at one of the wheels can be seen below.

$$P_{out} = \sqrt{2} * \tau_{out} * \frac{2v_{max}}{d_w}$$

Recognizing that the output power of a motor is related to the input power of the motor and the efficiency of the gearbox by the following equation, one can solve for the input power.

$$P_{in} = \frac{P_{out}}{\eta^n}$$

Here, the n corresponds to the number of gearbox stages, and eta corresponds to the gearbox efficiency for one stage. In this case, efficiency was assumed to be around .9. Also, for a given motor, the power curve can be given by the following equation.

$$P = \frac{-\omega_{free}}{\tau_{stall}} \tau_{motor}^{\ 2} - \omega_{free} \tau_{motor}$$

By setting the power in that equation to the input power, and replacing the input power with the Pout/efficiency, the motor torque can be determined by rearranging the previous equation into the following form and solving for the torque.

$$\tau_{motor} = \frac{\omega_{free} \pm \sqrt{\omega_{free}^2 - 4(\frac{\omega_{free}}{\tau_{stall}})(\frac{P_{out}}{\eta^n})}}{2(\frac{\omega_{free}}{\tau_{stall}})}$$

It should be noted that there are two possible torques that the motor can run at which would give the correct power. The smaller of the two values should be chosen because it will operate the motor at a higher efficiency, and should the motor need to apply more power (if the robot gets stuck and needs to push something or if the weight increases), the motor will be able to apply the necessary torque and power. After determining the input torque, the required maximum speed ratio of a gearbox can be determined by the following equation which relates this torque to the required torque at the wheel.

$$e_s = \frac{\tau_{motor}}{\tau_{out}}$$

Given these calculations, the motor gearbox reduction can be given by 1/speed ratio. The operating point of the motor can be verified again by looking at the motor graph and finding the efficiency corresponding to the motor's operating point. The main goal of this calculation is to ensure that the chosen motor can in fact provide the required power, and then make adjustments to the gearbox based on design goals. For example, if the motor is not operating at peak efficiency, the calculation can be reevaluated by choosing the motor input torque corresponding to the max efficiency for typical robot operation. Then the speed ratio can be recalculated using the desired torque at max efficiency and the calculated required output torque using the following equation.

$$e_{s,new} = \frac{\tau_{motor,maxefficiency}}{\tau_{out}}$$

From here, the maximum speed can be determined as well and verified to operate below the desired speed. It can be found with the following calculation, which depends on the angular speed of the motor at the chosen operating point.

$$v_{max,new} = e_{s,new} * \frac{2(\omega_{motor,chosen})}{d_w}$$

For the base, the team decided to use CIM motors. CIM motors were chosen because the team could get them at reduced cost and they provided the required power at a reasonable speed. The characteristics of the CIM motor can be seen in Figure 21.

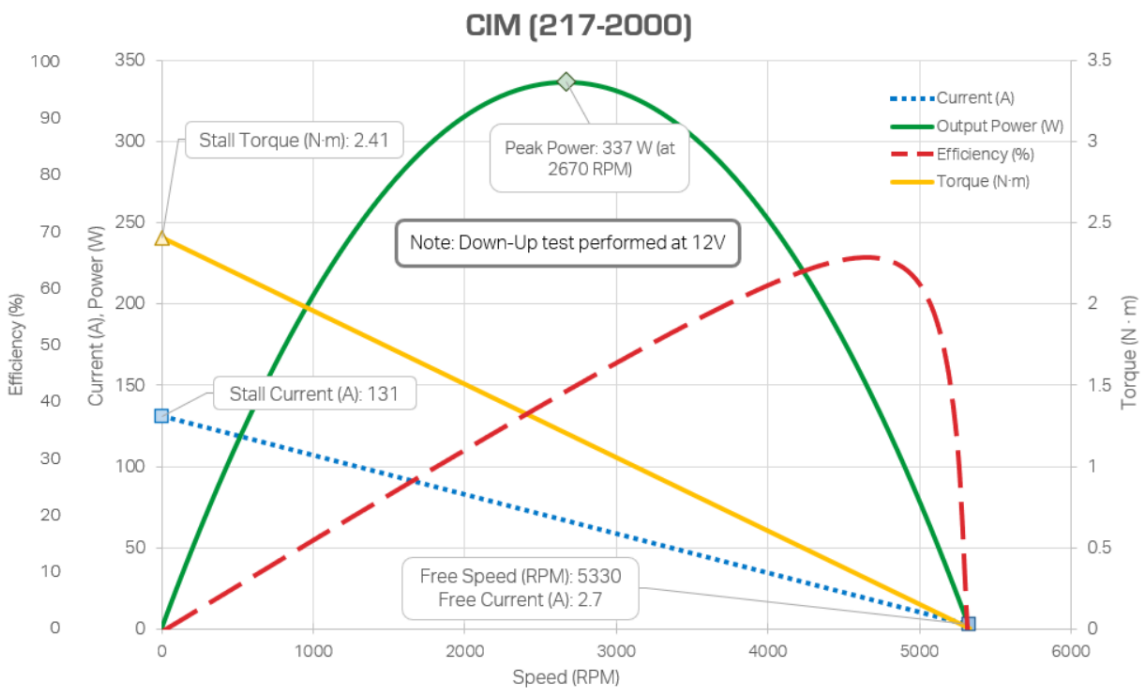


Figure 21: CIM motor curve[16]

Designing to run the motors at higher efficiency gave the following gearbox designs for each arm. Based on this data, a reduction of around 200:1 would allow for near-max efficiency operation for each arm. This value corresponded to the following max speeds, detailed in the table below. The desired design was to have the robot operate at less than 0.6m/s so that it would be less likely to veer out of control and damage the expensive robot arms. The torques used for this calculation were shown in the previous table and then using the 200:1 gear reduction, the CIM operating torque can be found, and corresponding motor speed determined. After doing unit conversions, the following was found for the robot speed.

Franka Emika Panda	.20m/s
Kinova Gen3	.21m/s
UR5e	.20m/s

Table 12: Mobile base speeds for each arm

[16] *Cim Motor - VEXPRO Motors - VEX robotics*. Motors. (n.d.). Retrieved April 28, 2022, from https://motors.vex.com/vexpro-motors/cim-motor

The robot should run at a speed of approximately .2m/s, well within the desired controllability of the arm. After completing the base calculations, it was important to calculate the elevator forces and torques as well in order to select gearboxes. The free body diagram of the elevator system can be seen in Figure 22.

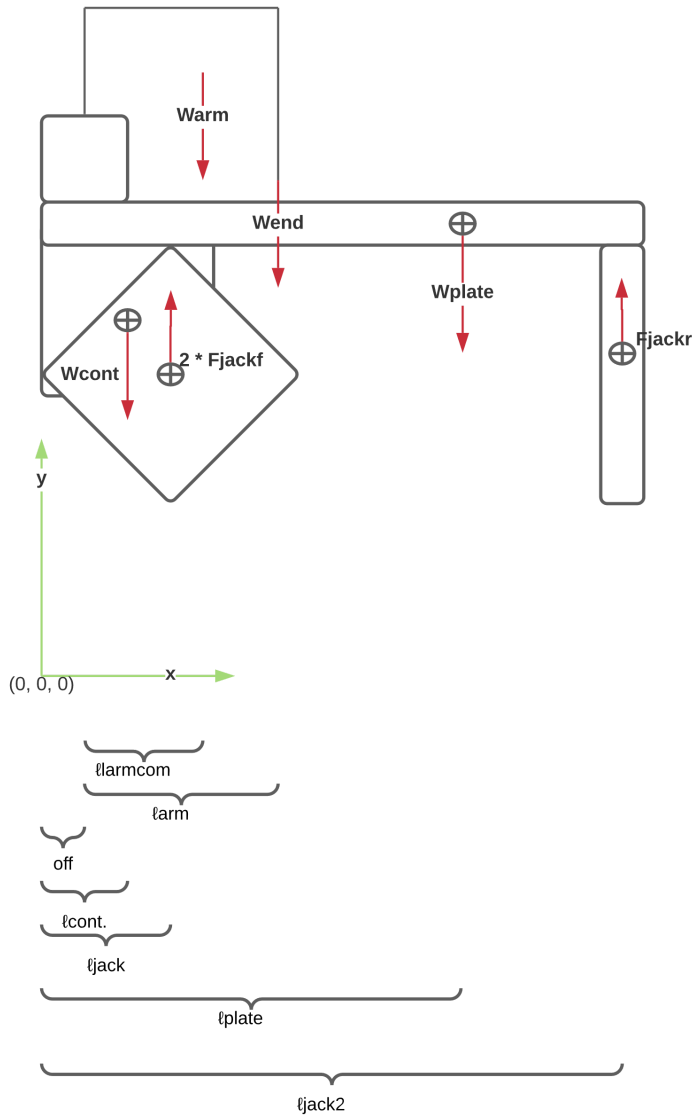


Figure 22: Elevator system free body diagram

From this diagram, the following equations of motion can be set up.

$$\sum F_x = 0$$

$$\sum F_y = 0 = 2F_{jackf} + F_{jackr} - W_{cont} - W_{plate} - W_{arm} - W_{end}$$

$$\sum M_{(0,0,0)} = 0 = 2F_{jackf} * l_{jack} + F_{jackr} * l_{jack2} - W_{cont} * l_{cont} - W_{plate} * l_{plate} - W_{arm} * \left(l_{armcom} + off\right) - W_{end} * (l_{arm} + off)$$

To simplify the calculations, Tweight was defined by the following equation and substituted back into the sum of moments.

$$\tau_{weight} = W_{cont} * l_{cont} + W_{plate} * l_{plate} + W_{arm} * \left(l_{armcom} + off\right) + W_{end} * (l_{arm} + off)$$

Then, the force at the rear jack could be solved with the moment equation. It was found to be described by the following equation.

$$F_{jackr} = \frac{\tau_{weight} - l_{jack} * 2F_{jackf}}{l_{jack2}}$$

Then, substituting back into the y forces equation gave the following result for the force on each of the front jacks if they both evenly held the weight, as was planned in the design specifications.

$$F_{jackf} = \frac{W_{cont} + W_{plate} + W_{arm} + W_{end} - \frac{\tau_{weight}}{l_{jack2}}}{2(1 - \frac{l_{jack}}{l_{jack2}})}$$

After figuring out the forces that the jacks need to produce, one can determine the amount of torque required to lift the jacks. In order to do this, a mapping between torque and force can be experimentally developed. By placing weights on each jack in increments of 5 pounds and measuring the torque required to start moving the jack, a linear regression could be developed. This was done in the gym with one 5 pound weight plate, two 10s, and a 25, and varying the weight from no weight to 40 pounds, which would account for more than the worst case scenario. The moment arm would be .18415m long (the length of the wrench used to twist the jacks) and the test setup can be seen in the tables below.

Force (N)		12in (Torque in Nm)	6.5in (Torque in Nm)	N 6.5in
0		0.66294	2.76225	15
22.24		0.69977	3.3147	18
44.48		0.88392	3.683	20
66.72		1.03124	4.0513	22
88.96		1.1049	4.4196	24
111.2		1.25222	4.7879	26
133.44		1.39954	5.1562	28
155.68		1.51003	5.5245	30
177.92		1.62052	5.8928	32

Table 13: Scissor jack 1 torque

Force (N)		12in (Torque in Nm)	6.5in (Torque in Nm)	N 6.5in
0		0.40513	0.25781	1.4
22.24		0.66294	0.77343	4.2
44.48		0.69977	1.1049	6
66.72		0.77343	1.32588	7.2
88.96		0.81026	1.69418	9.2
111.2		0.88392	1.8415	10
133.44		0.92075	2.2098	12
155.68		1.03124	2.5781	14
177.92		1.06807	2.9464	16

Table 14: Scissor jack 2 torque

Force (N)		12in (Torque in Nm)	6.5in (Torque in Nm)	N 6.5in
0		0.07366	0.55245	3
22.24		0.29464	0.7366	4
44.48		0.44196	0.99441	5.4
66.72		0.47879	1.196975	6.5
88.96		0.58928	1.39954	7.6
111.2		0.62611	1.69418	9.2
133.44		0.69977	2.2098	12
155.68		0.84709	2.9464	16
177.92		0.92075	3.3147	18

Table 15: Scissor jack 3 torque

Force (N)		12in (Torque in Nm)	6.5in (Torque in Nm)	N 6.5in
0		0.14732	0.40513	2.2
22.24		0.3683	0.55245	3
44.48		0.47879	0.84709	4.6
66.72		0.55245	1.03124	5.6
88.96		0.62611	1.25222	6.8
111.2		0.69977	1.58369	8.6
133.44		0.84709	1.8415	10
155.68		0.92075	2.2098	12
177.92		1.06807	2.5781	14

Table 16: Scissor jack 4 torque

Figure 23: Characterizing the torques required for each jack

After testing all four jacks (and rejecting the jack that required the most torque since it had a hard time turning), it was found that at the worst-case scenario the required torque was around 3Nm. The worst-case scenario was when the jack was in its lowest position. Then, from there the gear ratio could be determined. The chosen motors for the elevator were the 775pro since they were readily available and could easily provide the required torque once geared down to the necessary gearing. The motor curve of the 775pro can be found in Figure 24 below.

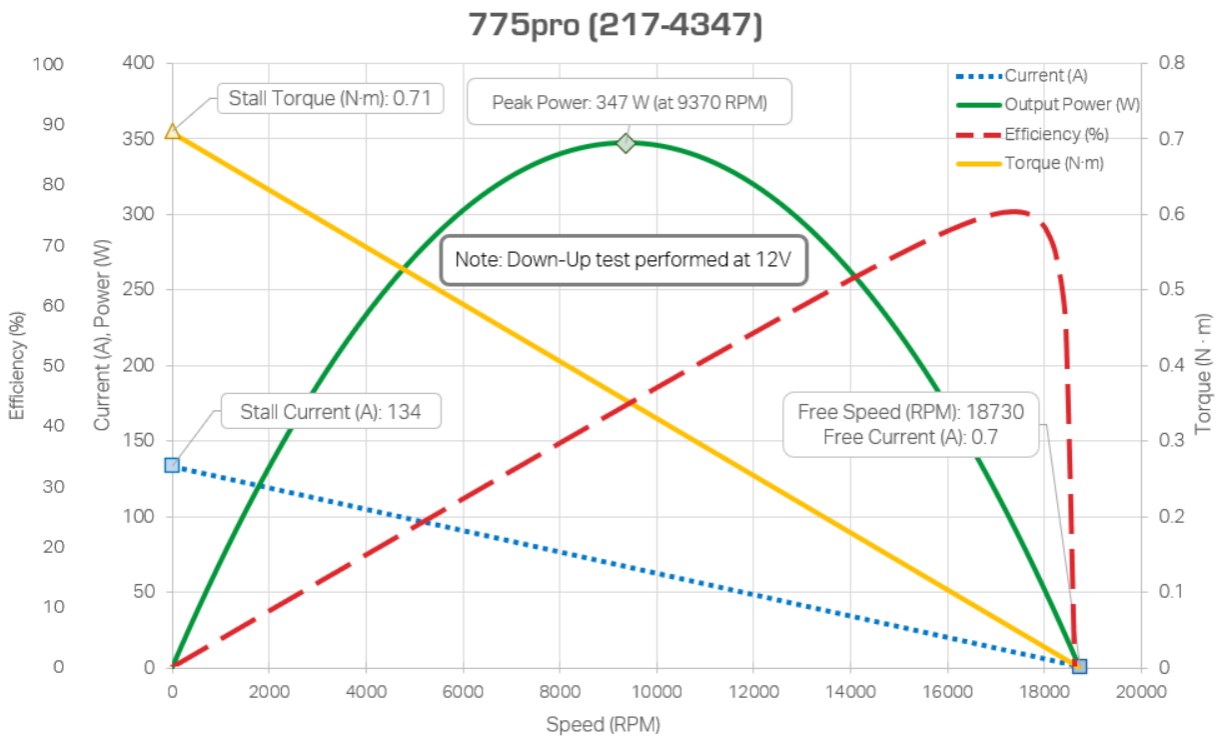


Figure 24: 775pro motor curve[17]

To run the motors near maximum efficiency, the input torque of 0.05Nm was chosen as the operating point, since the maximum torque was around 3Nm and the average typical operation was around 2Nm, choosing an output value of 2Nm would guarantee efficient operation of the jack near the bottom, which is a more desirable position due to a lower center of mass. Then, the gear ratio could be determined by dividing the output torque by the input torque multiplied by the efficiency (which would be two stages at .9 per stage, or n = 2), giving the following formula.

$$e_s = \frac{\tau_{out}}{\tau_{in} * \eta^n}$$

The gear ratio was thus found to be around 50:1. Wanting a slower speed, a gear ratio of 70:1 was chosen for the jacks. This meant that at maximum efficiency, the jacks would rotate at about 4.25 revolutions per second, when the minimum torque is required, about 4.5 revolutions per second, and at the maximum required torque of about 5Nm, at 3.8 revolutions per second.

[17] *775PRO - vexpro motors - VEX robotics*. Motors. (n.d.). Retrieved April 28, 2022, from https://motors.vex.com/vexpro-motors/775pro

VersaPlanetary gears were chosen to complete the gear reductions as they are readily available, easy to work with, and have favorable dimensions. Additionally, there is support for adding encoders to the motors.

Engineering is an iterative process. These calculations gave a rough estimate for the initial design of the entire system. After making more design decisions, such as the choice of motors and batteries, the process can be repeated with actual quantities instead of rough estimates. As such, after selecting the motors, gear reductions, and completing a CAD model of the frame of the mobile base, the actual weight of the system can be used to determine the desired center of mass and adjust the layout of the components inside. The new weight of the base affects the torque requirements of the motor, which then affects the gearing design and current draw of the system. When going back through this iteration, if the specifications remain similar, the system can proceed as planned. However, if the recalculated parameters are vastly different, a redesign would be necessary.

Upon completion of the base CAD, the calculations were verified, and small design changes were made if necessary. As it turned out, many of the preliminary calculations gave viable values for design specifications and the design could proceed as planned.

3.4 Electrical Design

The electrical design objectives strongly correlate to the mechanical design objectives as they enable the functionality of these hardware elements. Indeed, the electrical objectives can be summarized as follows:

1. Mobility Objectives
    a. Create a holonomic drive system
    b. Power the elevation system and drivetrain
    c. Have room for expansion of electronics
2. Modularity Objectives
    a. Create a system to connect the necessary connections to robotic arms universally
    b. Power the locking mechanism to the modular insert

One parameter to note is the average time of use for the base. This will give the average battery life to design for a near-full current draw. It was determined that an average lifespan of three hours with constant work would be acceptable while staying within the budgetary requirements. Should longer battery life be needed in reality, the capacity of the batteries can be upgraded while keeping the overall

electronic design the same. As the battery charge length is determined by use, it was also beneficial for the state of charge (SOC) sensing to be included in the robot. The SOC sensing allows the robot to protect the battery cells from overcharging and undercharging and allows for users to implement the ability to navigate to a charging station and self-charge. As the robot is a mobile base used for these different robotic arms, it is found necessary for the base to be able to navigate a space and avoid obstacles in its way. This allows the mobile base to be deployed in a non-controlled environment and still be able to navigate without crashing into the environment. The navigation also enhances the self-charging component and allows the robot to find its way to the charging station should the batteries discharge to a predetermined level.

One of the critical features determined to make the base modular was the ability to quickly and easily swap which arm the base is using. This is both physically, i.e., both mechanically and electrically, and programmatically implemented. This is an essential feature for true modularity. As mentioned earlier, if the arms become as modular as desired, a path to autonomous modularity opens when combined with the navigation component. The combination of the features will allow the base to navigate to a predetermined location with one of the arms and allow the robotic base to attach itself to the arm and permit the use of the arm without any assistance from a human moderator. These are just the external requirements and features which will drive the design of the robot and any specific specifications of the design will be discussed in the following sections.

3.4.1 Proposed Electrical Design

In order to meet the design specifications, an electrical system for the robot needed to be created. The following block diagram outlines the electrical system of the robot. The system components are discussed in depth in the section that follows. A breakdown of the system costs can be found in Appendix 10.2.

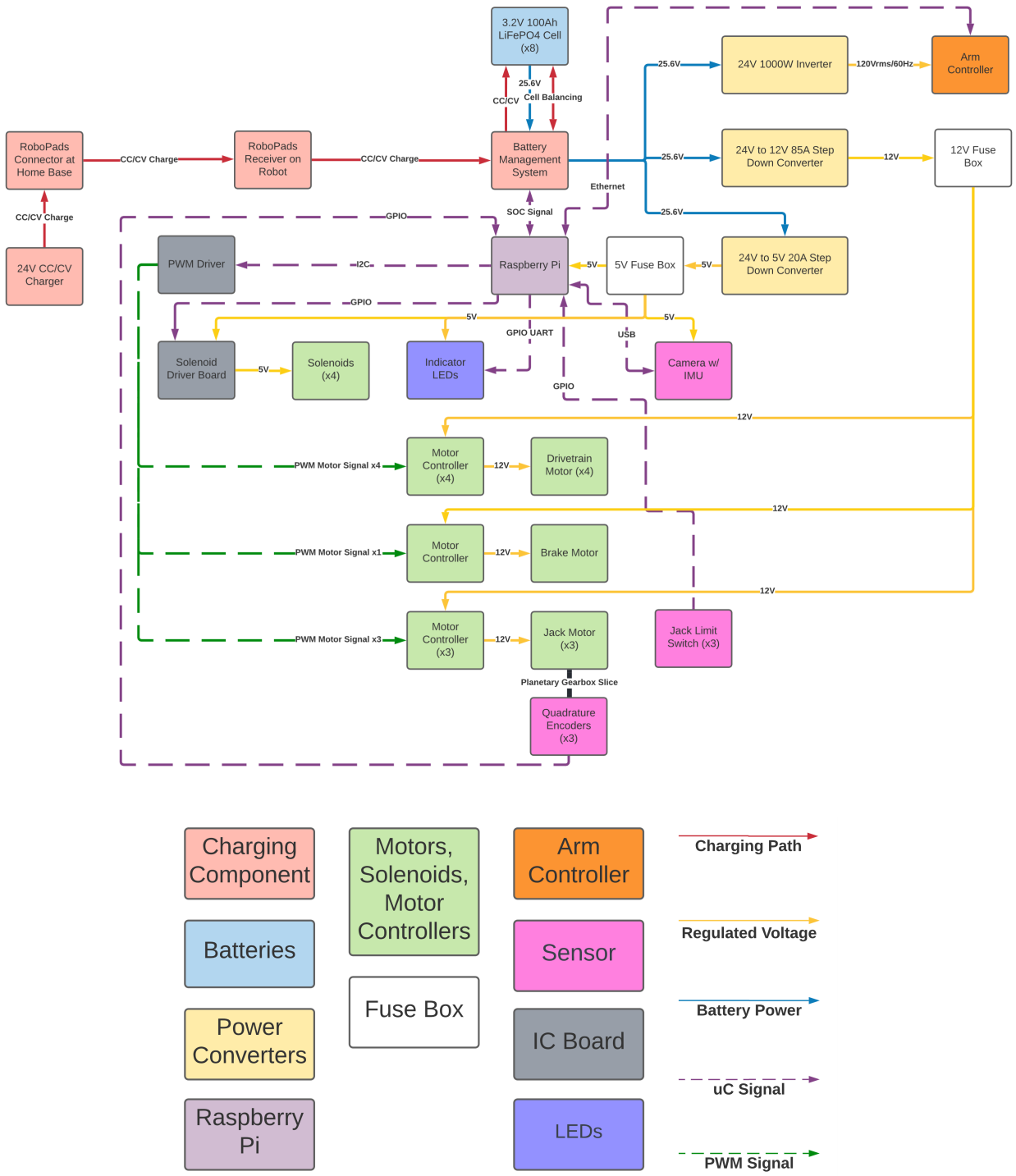


Figure 25: System block diagram

In order to run the system remotely and allow the robot to operate without a tether, the robot needed a set of batteries. In addition, the robot needed some piece of electrical hardware to control the charging and discharging of the batteries and monitor the batteries' health and state of charge. The team decided to use the Roboteq BMS1040ABT (BMS) battery management system. This BMS can monitor the charge of the batteries, control the charging and discharging of the batteries, relay statistics about the battery charge and health to a phone or computer via Bluetooth, and balance the cells so that they are all at the same amount of charge during charging and discharging. This is important because batteries must be charged and discharged at the same rate (whether in series or parallel) to maintain the health of the batteries and extend their life.

Figure 26: Roboteq BMS1040ABT[18]

Choosing a high-end BMS greatly influenced the rest of the decisions for the electrical system. As the robot would need an onboard microprocessor, a 5V power supply needed to be included in the robot. In addition, the CIM and 775pro motors chosen required 12V to run as desired, so a 12V power supply also needed to be included. Finally, a 120V inverter was required to power the controllers and arms of each robot. The actual components chosen for the power supplies will be covered later, but their

[18] *Search results for: '40V-100-AMPS-MANAGEMENT-SYSTEM-FOR-6-10-CELLS-LITHIUM-ION-BATTERIES-7786'*. Active Robots. (n.d.). Retrieved April 28, 2022, from https://www.active-robots.com/catalogsearch/result/?q=40v-100-amps-management-system-for-6-10-cells-lithium-ion-batteries-7786

necessity, along with BMS characteristics, dictated the choice of batteries. For starters, the Roboteq BMS required between 6 and 10 cells wired in series to effectively use the cell balancing system to preserve the robot's battery life. In addition, it is easier to find power converters that run off of standard voltages, such as 12V, 24V, or 48V inputs.

Similarly, high-capacity battery chemistries such as LiFe, LiFePO4, Lead Acid, and NiCd were all viable candidates for the chosen batteries. Researching their nominal charges and purchasing availability led the team to choose LiFePO4 batteries (typically used on fishing boats or to store excess energy generated by solar panels) because they were the most readily available battery for a fair price and had a nominal voltage of 3.2V. This meant that wiring eight cells in series would give 25.2V (within specifications for the majority of 24V input converters) and fulfill the specification of between six and ten batteries for the cell balancer in the BMS. Finally, LiFePO4 batteries could be easily sourced for a reasonable price, so the final decision was made to use LiFePO4 batteries. The exact capacity of the batteries chosen, as well as the charging circuitry, is discussed after completing current draw calculations based on the other components of the power system.

To charge the robot, an adequate charger needed to be chosen. The team decided to add Roboteq's Robopads to the robot to allow the robot to plug itself in and charge when the battery was low. The pads contain two strips of copper that correspond to the charging path and ground for the robot and connect to the BMS. When the robot approaches the complementary set of pads at the charging station, a magnet in the pads on the robot causes the pads to extend and make contact with the charging station, effectively plugging the robot into the charging controller without manual input. Finally, to charge, the team chose the charger recommended by the BMS, which was the Meanwell HEP-600C-24 charger. This charger could provide 20A of current in constant current mode to charge the batteries. The charger provides a constant current until the voltage of the batteries reaches 28.8V (the maximum for 8 LiFePO4 cells wired in series) and then maintains the voltage as the current drops in constant voltage mode. Upon completion of charging, the BMS will record the batteries as fully charged, and the robot will now be able to detect when it is low on charge and return to its home station.

Figure 27: Robopad charger[19]

The motor math from the previous section and robot arm specifications dictate the characteristics of the chosen power converters. It was found that under normal operation the Kinova Gen3 required 36W of power, the Franka Emika Panda required 300W, and the UR5e required 200W. The UR5e has a max draw of 570W, and the Panda has a max draw of 600W. Since all three run off 120V RMS power (typically plugged into a wall outlet), the chosen inverter must be able to continuously supply at least the maximum power to guarantee that the robot can continuously operate when plugged in. With the second most important factor being cost due to the limited budget of the project, the cheapest inverter that could supply around double the max power to guarantee optimum arm operation was the Aeliussine 1000W inverter. This inverter can supply 1000W of 120V RMS power from an input of 24V, which is more than enough for the robot arms. It cost $120, which was a cost-effective and readily available component on the market.

Since the CIMs and 775 pros are geared to run at their approximate maximum efficiency, the current draw at those operating points will dictate the specifications of the 12V converter. At max efficiency, each CIM motor will draw approximately 12.5A. Four motors running at max efficiency translated to 50A of continuous current. The 775pro motors draw around 10A each at max efficiency, equivalent to 30A of continuous current when three are running. If all seven motors were running

[19] *RoboPads Charge System*. Roboteq. (n.d.). Retrieved April 28, 2022, from https://www.roboteq.com/all-products/robopads-charge-system

simultaneously, the system would draw 80A of current. However, to limit current draw and keep the robot under control to better protect the robot arms, it was determined that the elevator and base would not be allowed to move at the same time. That means that the 50A of continuous current drawn by the CIM motors would be the maximum current drawn from the 12V regulator. Should the robot run into an object or need to carry a heavy load, more than 50A of current would be drawn because the required torque to move the robot would be more significant. Thus, a regulator that could source at least 50A was required, with 80-90A being ideal. With price being a primary concern, a cheap, readily available 24V to 12V converter that could source between 80A and 90A was chosen. The converter chosen was the Daygreen 24V to 12V step down converter because it could provide 85A of continuous current at 12V from a 24V input and cost $85.

A vast majority of microcontrollers and sensors require 5V to run. As the robot would need a microcontroller and multiple sensors, a 5V power converter was required to supply 5V to the system. The system had three major components that needed 5V: the indicator LEDs, the microcontroller, and the sensors (camera, limit switches, encoders, etc.). Examining the power requirements of each component would determine the exact 5V controller necessary for the system. First, a Raspberry Pi 4B (hereby referred to as the Pi) was chosen because they are cheap and readily available. They also contain 40 pins for input and output, which was plenty for the system and could be extended if necessary, and could build ROS environments, which were integral to the project. Under normal operating conditions, a Pi draws a little over 1A of current. The indicator LEDs draw 3.5A per meter of LED strip, and since the robot would use 3 meters, that maximum current would be 10.5A when all the lights were white. Finally, the limit switches would draw a negligible current because they would be connected to the Pi input with a high impedance, and the camera would draw 700mA, giving a total current of 12.2A. So, a 5V converter that could source at least 12.2A was required. Recognizing that having extra amperage for the addition of more sensors to improve the modularity of the robot, a 5V converter that could provide 20A was chosen. The chosen converter was the Meanwell SD-100-5B converter because it met all of the current and voltage specs while being the cheapest and most available component.

In addition to the 12V and 5V converters, fuse boxes were included to protect the robot in the case of a component short or high current draw. If a fuse blows, the robot gets shut down, and the problem can be identified and fixed, so the current draw from the 12V or 5V converters does not exceed its rating. Fuses were connected between the converter and each motor, as well as to any sensors.

Section 3.3.2.3 determined the selection of motors. However, the motor controllers still needed to be chosen. The team chose the Vex Victor SP motor controllers for several reasons. First, they were cheap and readily available, as the team acquired them for free from a partner. Secondly, they interface well with

the chosen motors. These CIM and 775pro motors are some of the most widely used in the First Robotics Competition, and the Victor SP controllers were specifically designed to be used with these types of motors. Finally, these motor controllers were easy to control because they were well documented and already had driver libraries.

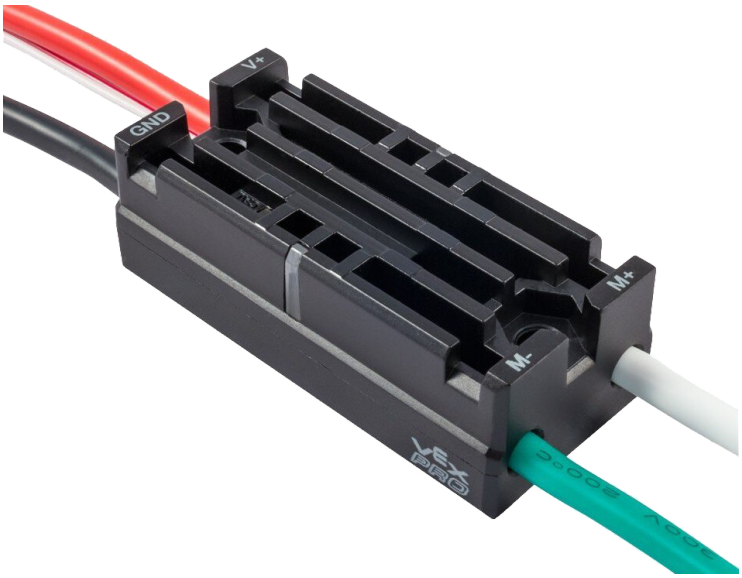


Figure 28: Victor SP motor controller[20]

To control the motors, a PWM signal was needed. Since the robot was running short on GPIO pins, the team decided to find an intermediate board that could be controlled via I2C. I2C (Inter-Integrated Circuit) uses two pins to talk to different peripherals from the Pi. One pin is the SCL (Serial Clock) line or the clock which generates a timing signal. The other is SDA (Serial Data) or the data line which transmits data between peripherals according to the I2C protocol. To send data, the address of the desired peripheral is sent across the line. The peripheral acknowledges, and communication can begin. The chosen board was the PCA9685 Adafruit 16 Channel PWM driver. With this board, up to sixteen motor controllers could be driven. Since the robot only had seven, this leaves room for other motors in the future, improving modularity. Additionally, this board was chosen because it had driver libraries available for easy system integration and was cheap.

In addition to the motor controllers, the 775pro motors needed quadrature encoders to ensure that the position of the jacks could be determined. The encoders chosen were the Versa Planetary encoders

[20] *Hardware component overview*. FIRST Robotics Competition Documentation. (n.d.). Retrieved April 28, 2022, from https://docs.wpilib.org/en/stable/docs/controls-overviews/control-system-hardware.html

because they interfaced easily with the gearboxes on the jacks and are widely used in FRC, making their integration into the system simple thanks to substantial documentation.

The robot needed a window motor to actuate the brakes. The brakes would help keep the robot in place when not driving and help to extend the battery life. The justification for adding this motor is shown in the following section.

The indicator lights chosen were the Adafruit Neopixels. These were chosen because they only required one GPIO pin to work through via a UART protocol, and were easy to control due to ample documentation.

Ideally, the mobile base would understand where it is in relation to the world through advanced sensors, such as the Intel Realsense d435. This RGBD camera generates a 3D point cloud which can be filtered into an occupancy grid to enable navigation. A USB-C line connects this camera to a USB3.0 in the Pi.

Figure 29: Realsense d435 camera[21]

The robot also needed solenoids to help lock/unlock the modular insert with the arm in place once deposited in the base. The chosen solenoids were the ROB-15324 from Sparkfun because they were cheap, readily available, and consumed around 1A each, but only when turned on.

[21] *Depth camera D435*. Intel® RealSense™ Depth and Tracking Cameras. (2021, June 17). Retrieved April 28, 2022, from https://www.intelrealsense.com/depth-camera-d435/

To drive the solenoids, a control PCB was designed to allow for control from the Pi or from an external button. An in-depth analysis of the PCB follows after the battery life math.

Finally, the chosen limit switches were the ME-8108 Momentary limit switch. These limit switches were necessary so that the robot could zero the jacks and also be able to tell the motors to stop lowering the jacks when activated so that the system did not break. These switches were chosen because the lever arm was easy to position to be actuated by the jacks and they were robust enough to withstand the force of the jacks pushing down on them.

3.4.2 Initial Electrical Analysis

In addition to the current draw math, which helped determine the necessary power converters, the battery life of the robot could be estimated by examining it under typical operating conditions. The current draw of the different components plus the duty cycle, the assumed average percentage of the time running, are seen in the tables below. Two scenarios are explored: one where the motors need constant power to keep the robot in place while the arm moves, and one where a braking mechanism is included to prevent the wheels from turning, saving power. It was assumed that the indicator lights would not be on all the time, so the current was halved, since each part of the RGB lights take 1/3 of that 10.5A and would not always be on at once. The Frank Emika Panda would be the arm mostly used with the base and also have the highest current draw, so its characteristics were used in these calculations. It was also assumed that the jacks would be moving around 5% of the time because they only move a couple of times to get the arm into position and then will stay stationary until the base needs to move again.

Component	**Current Draw**	**Duty Cycle**	**Average Current Draw**
Arm Controller & Arm	3.5A	100%	3.5A
CIM Motors & Controllers (4)	50A	100%	50A
775pro Motors & Controllers (3)	30A	5%	1.5A
RealSense Camera	700mA	100%	0.7A
Neopixel Indicator Lights	5A	100%	5A
Solenoids & Board	4A	0.5%	0.02A
Raspberry Pi	1.2A	100%	1.2A

Table 17: Current draw without braking

By adding up the average currents, it was calculated that the robot would draw an average of 61.92A. The battery capacity is 100Ah. This capacity was chosen because it would give the robot without brakes at least an hour of runtime, and the specific 100Ah LiFePO4 batteries found on AliBaba were the cheapest high capacity that could be found by far. By using the following formula, the average battery life was determined.

$$Life_{battery} = Capacity/Current_{avg}$$

In this case, the robot would average 1.61 hours of runtime on a full charge. Next, the braking senario was investigated. It was assumed that at worst, the robot would be driving for at most 1/3 of its runtime because it is a slow-moving robot. Also, the arm would likely not need to be spending more than that much time swapping its workspace. The brakes would be engaged the other 2/3 of the time. However, since the breaks would be using one window motor, it would only draw current as it actuated the brakes, as the motor cannot be back driven. Thus, the braking mechanism would only be active around 5% of the time.

Component	Current Draw	Duty Cycle	Average Current Draw
Arm Controller & Arm	3.5A	100%	3.5A
CIM Motors & Controllers (4)	50A	33%	16.7A
775pro Motors & Controllers (3)	30A	5%	1.5A
RealSense Camera	700mA	100%	0.7A
Neopixel Indicator Lights	5A	100%	5A
Solenoids & Board	4A	0.5%	0.02A
Raspberry Pi	1.2A	100%	1.2A
Braking Mechanism	7.5A	5%	0.375A

Table 18: Current draw with braking

Once again, by adding up the average currents, it was calculated that the robot would draw an average of 28.995A. The battery capacity is 100Ah. By using the following formula, the average battery life can be determined.

$$Life_{battery} = Capacity/Current_{avg}$$

With the addition of a braking mechanism, the battery life will be around 3.45 hours of runtime, which is an increase of 2.14 times over the non-braking scenario. Thus, the braking system should be added to the robot.

3.4.3 Solenoid PCB Design

Finally, the design of the PCB solenoid driver is discussed. The circuit needed to be able to take an input from the Raspberry Pi GPIO to actuate the solenoids. In addition, it needed a manual release button with a timer to be able to release the solenoids should the Pi be turned off. This determined that the circuit would need some way to take two inputs and still be able to actuate should one of them turn on. Using N-channel MOSFETS would allow the inputs to be connected in a MOSFET 'or' configuration. Then, an RC timing circuit could be determined. In order to first determine the RC time constant, the transistors had to be chosen. Knowing that the Pi can output 3.3V, a transistor with a 3V or less threshold voltage needed to be found. Also, the transistor needed to be able to handle at least 4A of current each. A search for cheap transistors with these characteristics yielded the PSM015-100B, 118 N-channel MOSFET. The threshold is 3V and the drain current is up to 75A. Now the RC time constant could be determined. The design required a time constant that would drop the voltage from 12V to 4V over a maximum of 10s of activation, as that was the maximum time that the solenoids should be on. The following formula was used to get the time constant.

$$RC = \frac{-10}{ln(\frac{1}{3})} = 9.10$$

Having found the time constant, now the R and C values could be determined. A 10uF capacitor was chosen because it was a readily available standard value, and led to the resistance value of 910k Ohms. Since 10s is the maximum time that the solenoids were rated for, choosing a resistance of around 732k Ohms would cause it to discharge in around 8s instead and thus was chosen. In addition, the same resistance was applied to the Pi side to make sure that the MOSFET gate could discharge. A series resistance between the Pi and MOSFET was included to limit current and prevent oscillation when the signal is sent. Finally, flyback diodes were placed in parallel with every solenoid in reverse bias to prevent large current spikes when the solenoids turn off. The schematic of the PCB is shown in Figure 30.

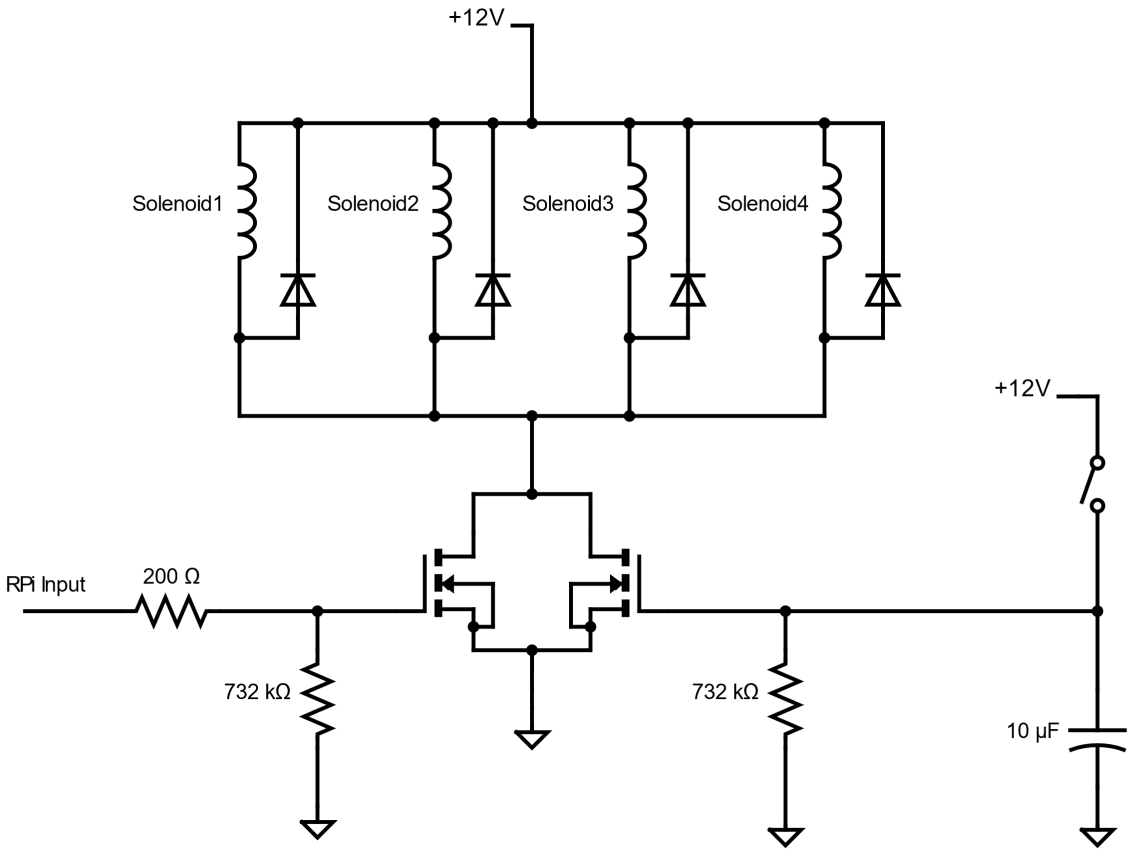


Figure 30: Circuit diagram

This schematic was then input into Altium so a PCB could be designed. The Altium schematic can be seen below. In addition to the schematic, this schematic contains a button with a light that turns on when pressed and off when the RC timer discharges, as well as a 15 pin header to attach every wire for the solenoids, button, control, and power. The schematic can be seen below in Figure 31.

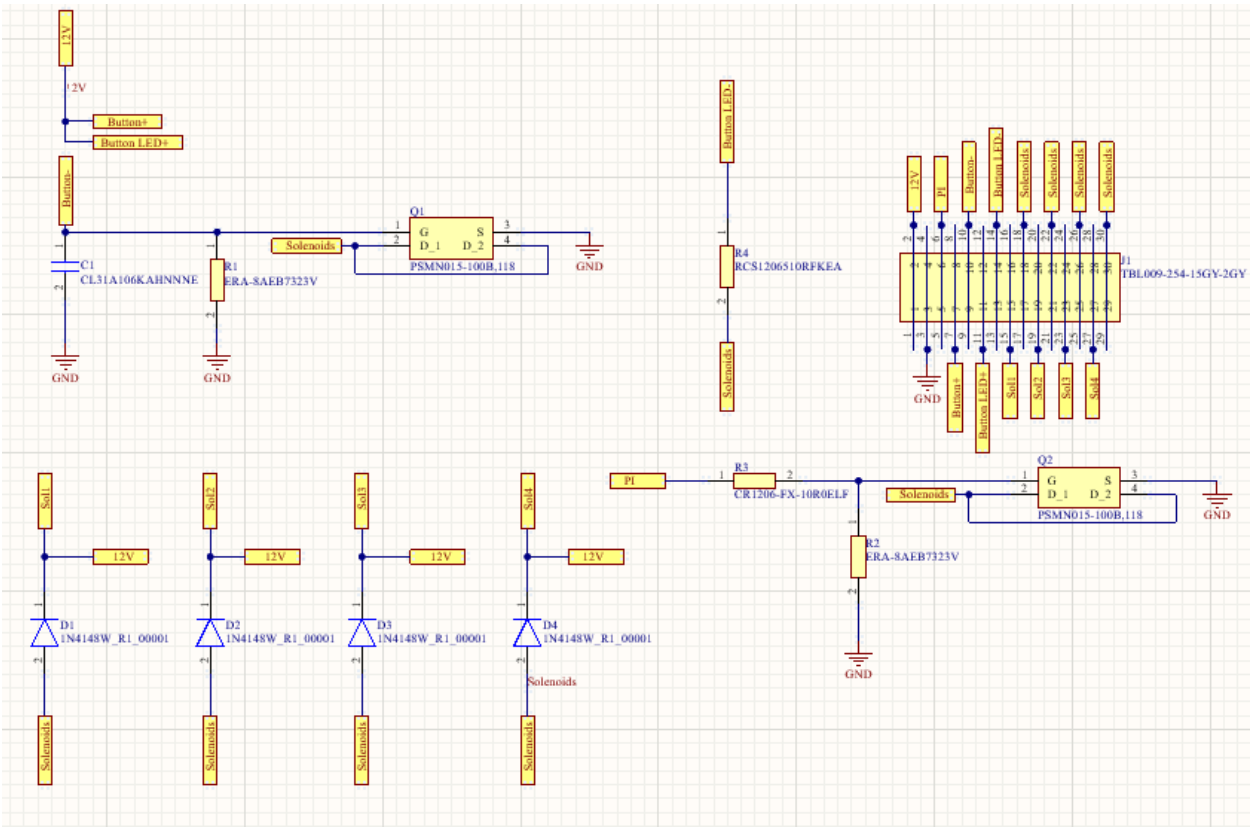


Figure 31: Altium schematic

By connecting all the components as shown in the schematic with traces, a PCB could be realized. The PCB was designed to be as small as possible to make it easy to mount. The board utilized four layers: two signals on the top and bottom, and a 12V and ground plane inside the board. This allowed for easier routing of components and improved the nodes; thermal characteristics of nodes, which took a significant amount of power. Finally, a copper pour was utilized on top of the board so that the high current solenoids could connect to the transistors and produce less heat. Images of the schematic and 3D board are shown in Figure 32 and Figure 33.

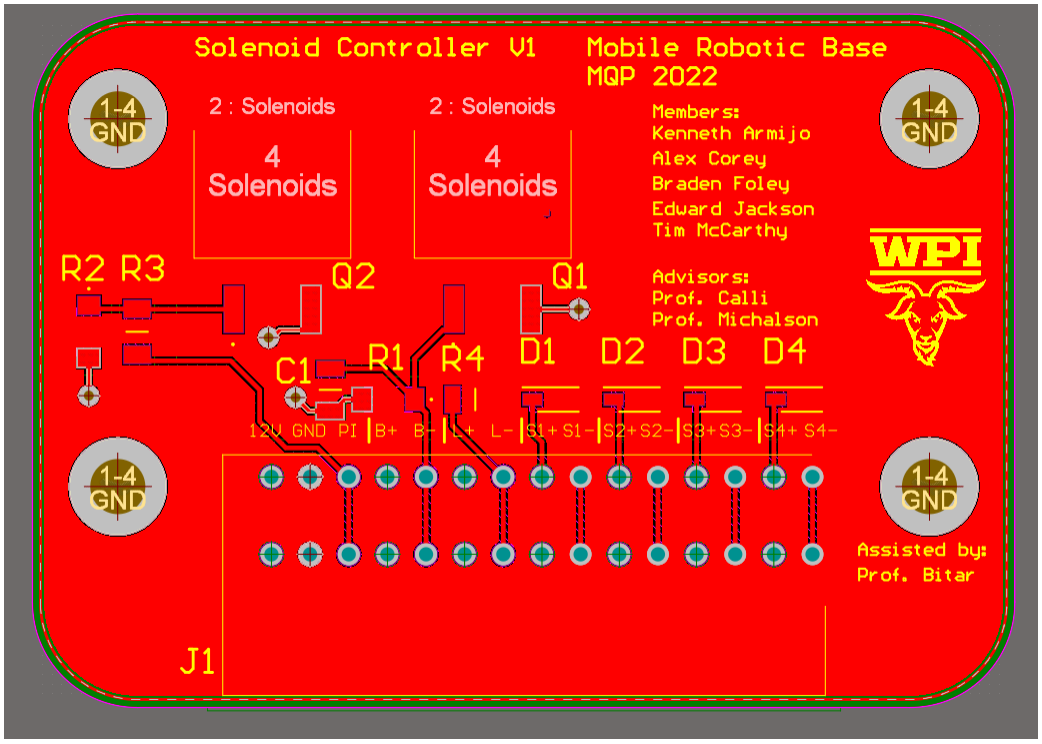


Figure 32: PCB Design

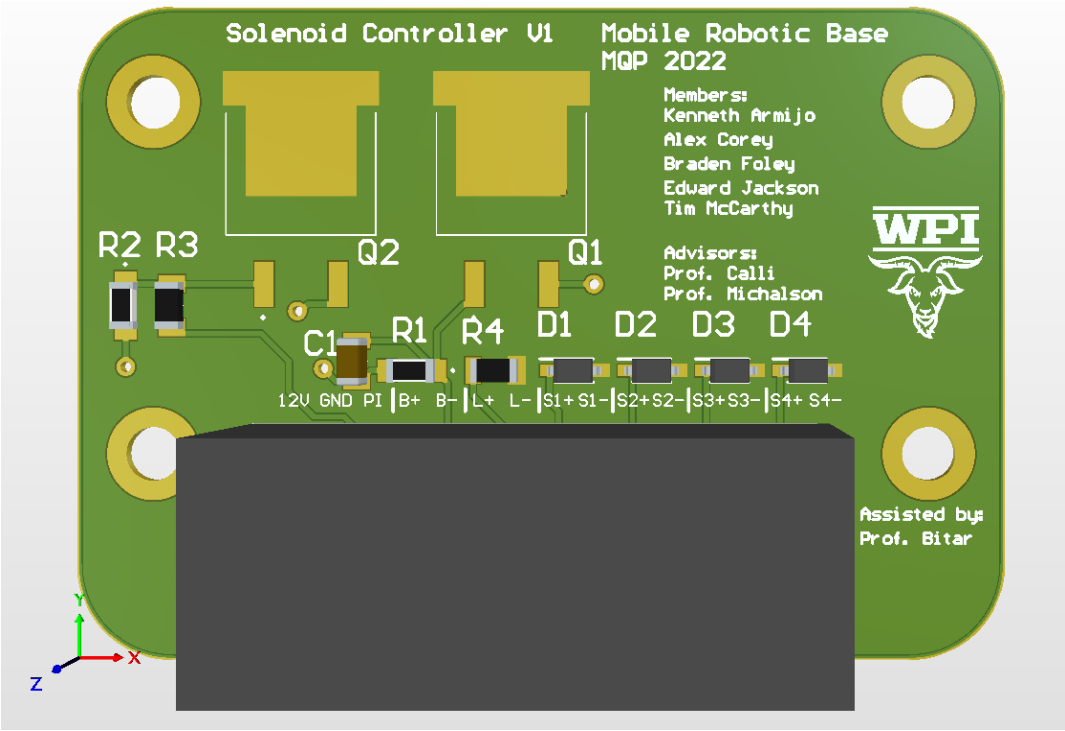


Figure 33: 3D PCB

3.5 Software Design

3.5.1 Package Design

The initial design for the ROS software was broken up into three packages: arm actuation, base actuation, and task manager. Inside each package were several nodes for performing various functions, though they were generalizations for unknown nodes that would need to be used. The arm package included drivers for each arm and an abstract node that would interface between the drivers and the central planner node to control the arm. A sensor from the modular insert would be passed to the abstraction node to determine which arm was plugged into the base. The base package originally included a driver for the elevator and an entire navigation stack for moving the base around. Encoders on the elevator jacks would be used to tell how high the elevator was. For the chassis, other sensors like a camera and Inertial Measurement Unit (IMU) would be used for SLAM, or Simultaneous Localization And Mapping of the environment. The task manager package included two nodes: one for scheduling tasks and delegating each part of the subtask to the other packages, and one for controlling the queue of tasks created by the user. Tasks could be submitted by the user and stored until the previous task had been completed.

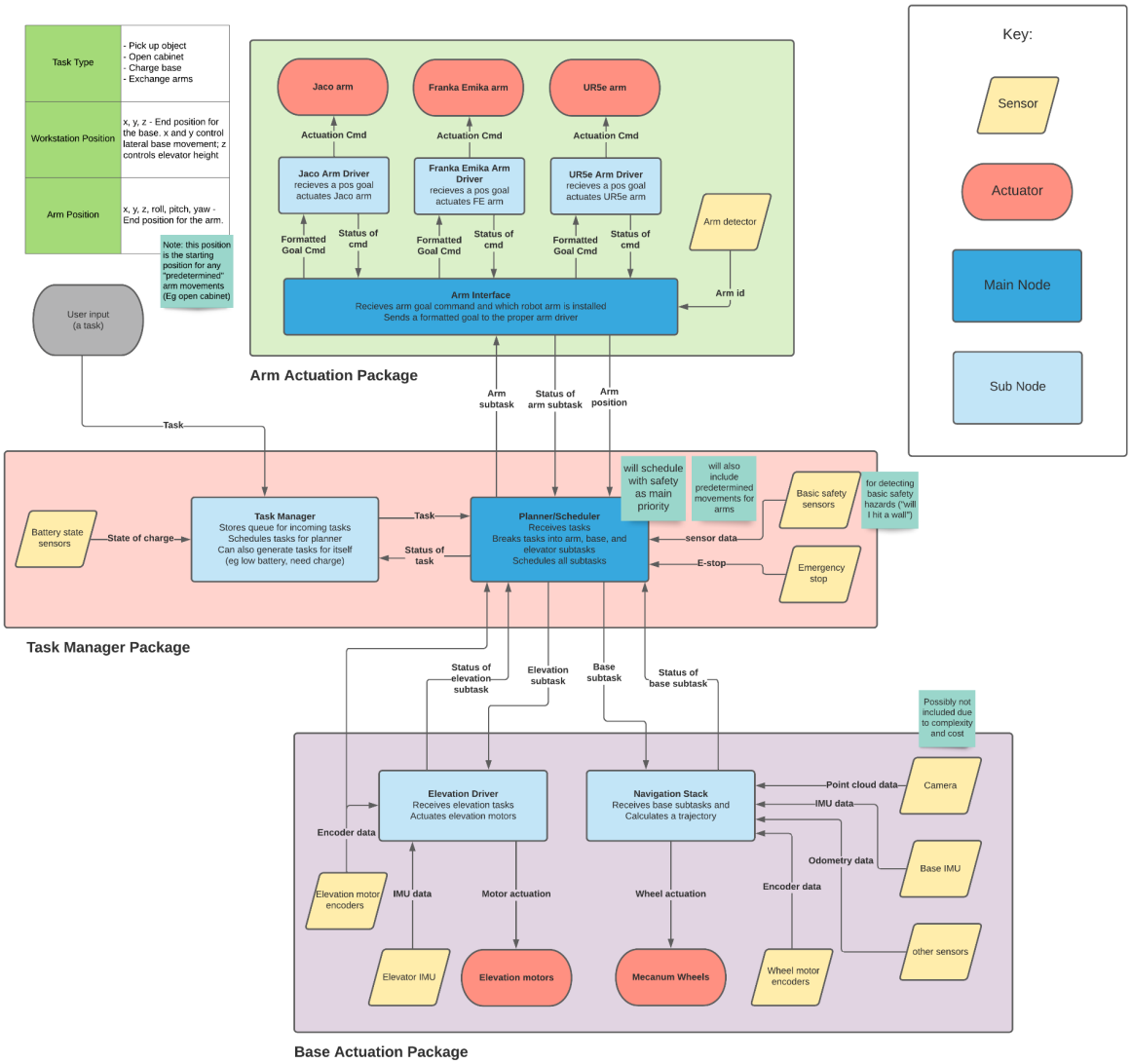


Figure 34: Initial package design specification

Over the course of development, the package design evolved to fit the actual implementation of the program. Primarily, the planning node was replaced by a fully fleshed-out state machine, which properly separated tasks into arm subtasks and base subtasks. After being split up, the subtasks were routed into a client's /server interface for a more clear division of labor. The node structure of the base package was more flushed out. Elevator jack encoders and limit switches received their own nodes and were routed to the elevator p-controller. Additionally, the task data type was more fully defined, adding in specific task types and data types for the base, elevator, arm, and gripper values. Finally, a distinction was

made between which nodes were launched on the host machine, which controlled most of the computationally intensive logic, and which nodes were on the remote machine, which operated the mobile base. The final design specification can be found in Figure 35.

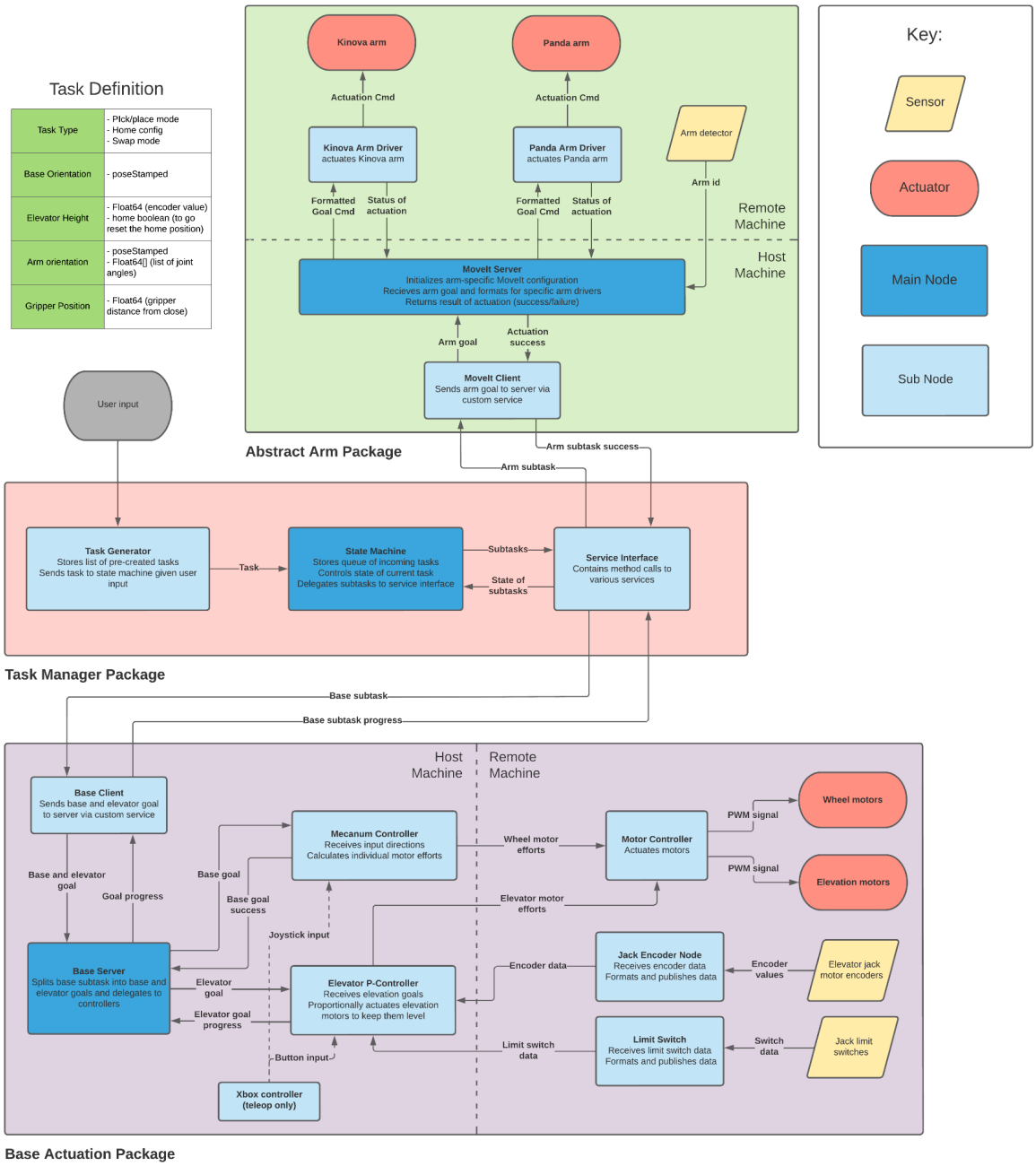


Figure 35: Final package design specification

3.5.2 Abstraction

Abstraction played a vital role in the development of the software system. ROS is a prime example of using abstraction. Instead of having to wire every node to every other node, ROS adds several abstraction layers for sending data and messages between nodes through launch file hierarchies, namespaces, and groupings.

A primary instance of abstraction in the program design is controlling the robotic arms. Each arm has its dedicated driver and MoveIt! configuration to make it move properly. However, as one of the objectives of this project was modularity, a solution was needed to control each arm using the same interface. This is where the MoveIt! server came in. Based on which arm was inserted into the robot on initialization, the server would launch the arm's associated driver and MoveIt! configuration. Then using the previously established control flow, the user wouldn't have to worry about setting up arm-specific drivers, as the abstraction layer would handle data routing automatically. Figure 36 describes a simplified data flow for controlling one of the robotic arms.

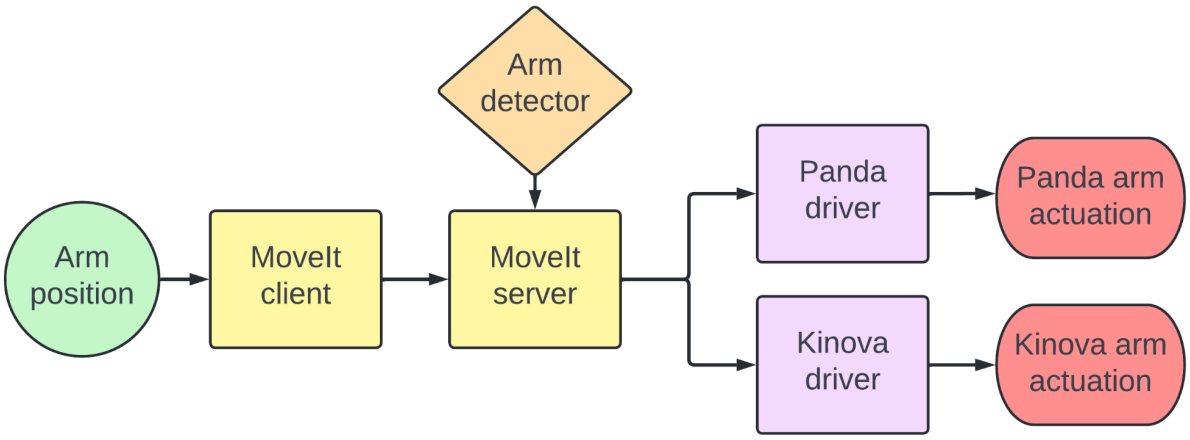


Figure 36: Simplified abstracted arm control

3.5.3 State Machine Design

The autonomous state machine had to be primarily designed with safety in mind. To this end, two constraints were identified for controlling the order in which parts of the robot should be active:

1. The base should not move when the elevators are extended.
2. The base and elevators should not move when the arm is not in its home position.

Specifically, the state machine was assembled such that neither the base nor the elevator would be able to move while the arm was not in its home position, as an outstretched arm on a moving base could potentially cause danger to either itself or its users. Similarly, the base was designed not to move when the elevator jacks were up, as an extended mobile base would be more prone to tipping. These two constraints lent themselves to a decision-based state machine, as can be seen below in Figure 37.

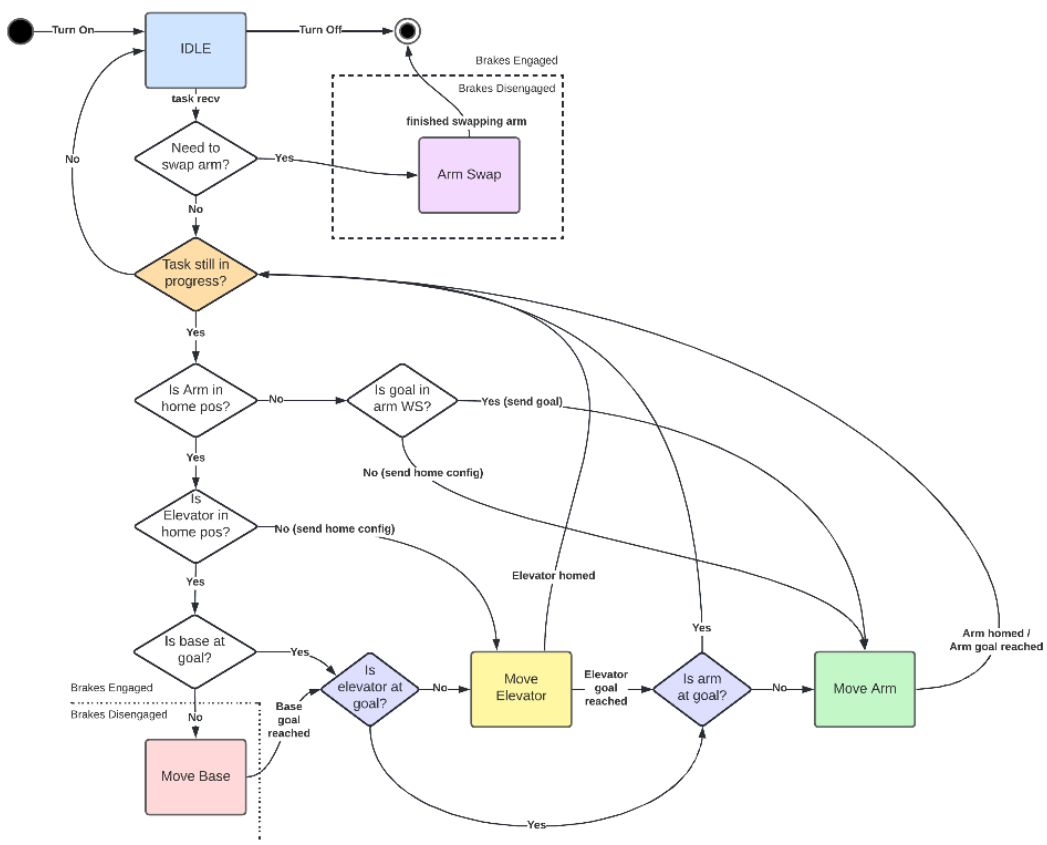


Figure 37: Autonomous mode state machine

While waiting for a task, the robot would sit in the Idle state. When a task was received, the first check was to determine if the new task was an arm swap. An arm swap required a different order of operations than normal autonomous operation, so it was given its own state with a different control flow. Regular tasks went on into the Task Progress state, where the two aforementioned constraints were addressed. If the arm was not in the home position, the state machine moved to the Arm state where the arm moved to its home position. This then looped back around to the task progress state. After the arm was addressed, then the elevators had to be checked. If the elevator was not low enough, the state machine moved to the Elevator state and the elevators moved to their home position. The loop then started once again.

Once these two constraints were satisfied, the robot could actually start performing the intended task. First, the base orientation was checked; if the base was not in its intended location, the base moved. Then the elevator was checked; if it was at the incorrect height, the elevators moved. Finally, the arm's position was checked; if it was out of place, the arm moved. If any of these subsystems didn't need to move, they were skipped over. After the arm was finished moving, the state machine again looped back to the Task Progress state. Once the base, elevator, and arm subtasks were achieved, the machine returned to the Idle state.

4 Implementation

4.1 Segmentation of Tasks

This section describes the team structure and planning outline utilized for the duration of this project. As a team of five, efficiency and specialization were key elements in bringing this project to the highest degree possible. It then discusses how the team implemented the above design into a fully functioning robot.

4.1.1 Gantt Chart

A Gantt Chart was created and frequently reflected upon to evaluate progress and highlight lacking areas.

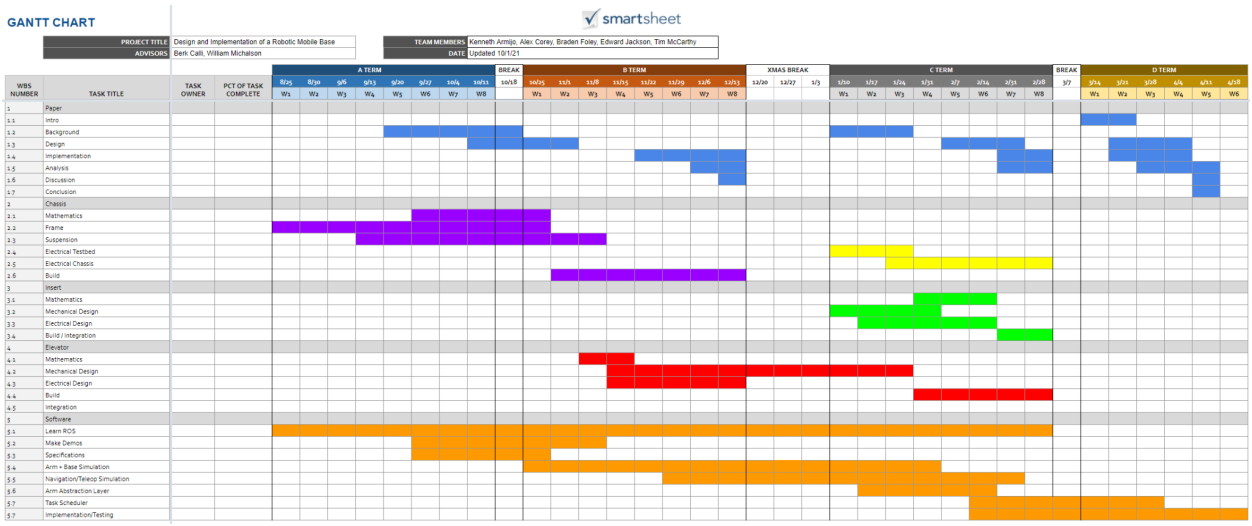


Figure 38: Gantt chart of project

The timeline was divided by WPI's quarterly terms and by key components of the project, such as the chassis, elevation, modular insert, and the software side. In many instances, work was being overlapped to promote diversity and maximize efficiency, however, the start paid a significant portion of time dedicated to just the chassis. This was to ensure the foundation was robust and reliable for future designs. This also enabled for a wide exploration of alternative design paths and ideas.

4.1.2 Meeting Structure

Meetings were designed to constantly coordinate, evaluate, and reevaluate progress while promoting design iteration and brainstorming. This was accomplished through two types of meetings: general body meetings and intra-disciplinary meetings. General body meetings were only twice a week but allowed the team to determine progress, highlight issues, and increase communication. Intra-discipline meetings specialized in their respective departments (mechanical, electrical, and computer science) to brainstorm, collaborate, and create procedures for the project.

4.1.3 DFMEA

Design Failure Mode and Effect Analysis (DFMEA) was a tool utilized in the Mechanical discipline to evaluate areas of weakness in the design. Indeed, the criteria deemed in need of attention were rectified through design choices or proven safe through mathematical analysis. An in-depth DFMEA of the elevation and chassis can be found in Appendix 10.3.

4.2 Mechanical Implementation

The mechanical implementation is divided into two subcategories: the mobile base design and the modular insert. Additionally, a comparison between the designed Computer Assisted Design (CAD) model and the physically constructed model is made.

To easily visualize the components to be further explored and discussed, the final design is portrayed below in an exploded view in Figure 39.

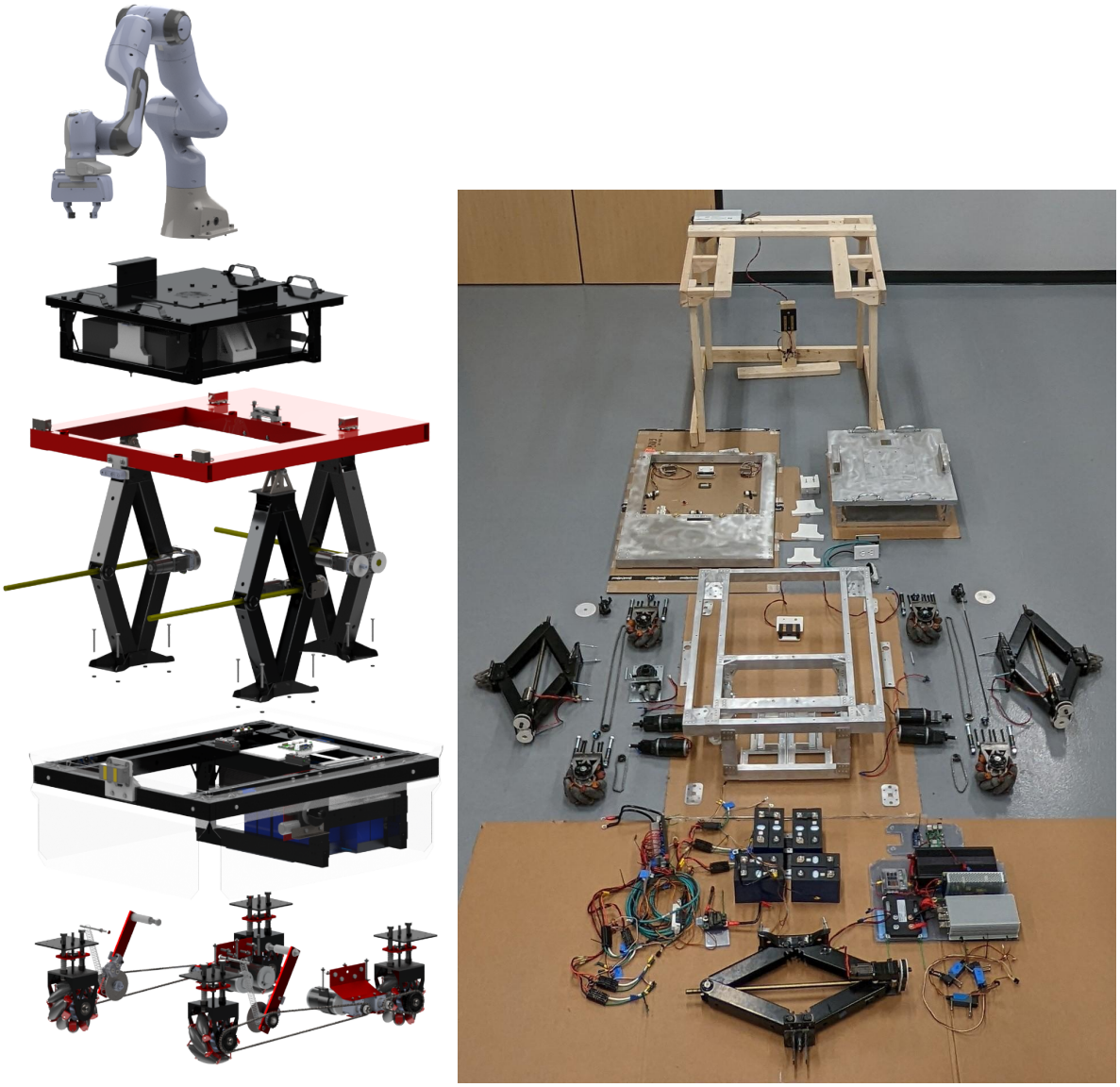


Figure 39: Exploded view of CAD and actual exploded assembly

4.2.1 Mobile Base

The mobile base consists of the chassis, drivetrain, elevation, and electronic housing. In short, everything below the red tabletop in the above figure is considered part of the mobile base.

4.2.1.1 Chassis

The chassis was designed to support the weight of the mobile base, fit the center of gravity and slipping/tipping requirements derived in 3.3.2.2, house the electronics adequately, and fit within a standard doorway.

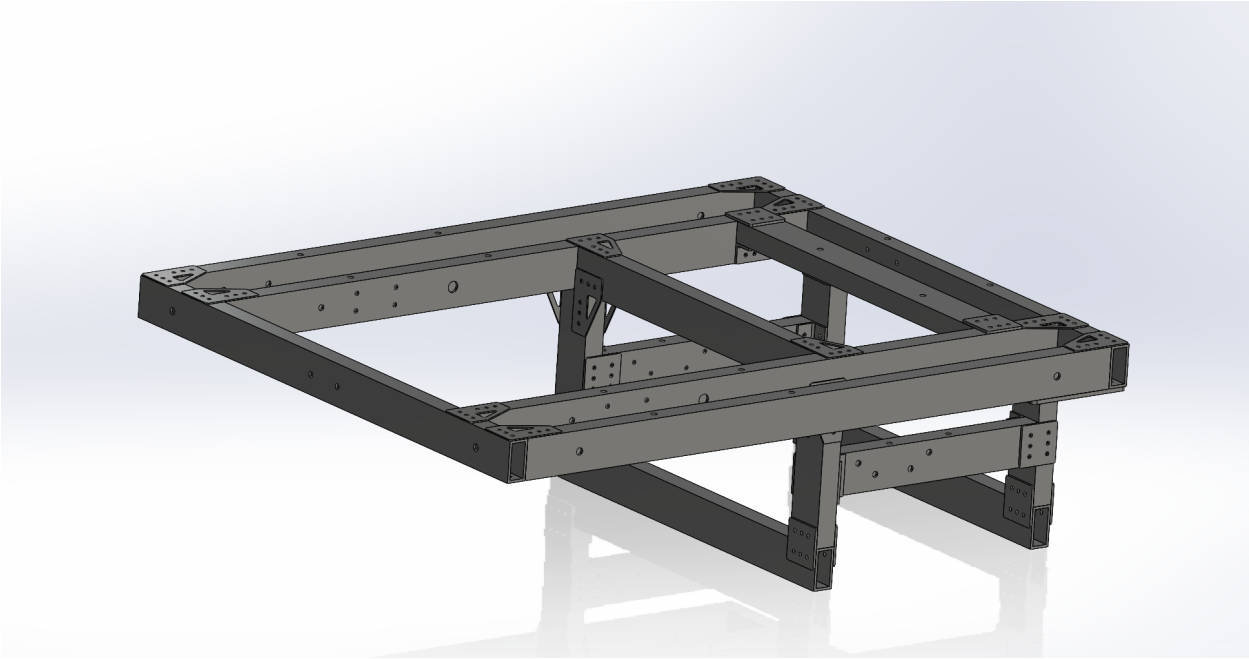


Figure 40: Chassis in CAD

The chassis was created with 1/16th in. box tubing ordered from Metal Supermarket and cut to length. Individual holes were drilled manually to ensure any inaccuracies from the CAD model and the assembly did not result in unexpected alignment. Additionally, only rivets were utilized in the assembly of the chassis along with custom brackets created from SendCutSend.

The large cutout in the center was designed to be able to house the largest controller of the UR5e despite most controllers being considerably smaller. This imbalance causes all of the electronics to be stored in the right deposit.

The width and length of the chassis were designed to be 28 inches and 35 inches, respectively, giving enough clearance to fit within a standard 30-inch doorway. However, the current implementation has sharp corners and could potentially damage the doorway if the 2-inch clearance is not sufficient.

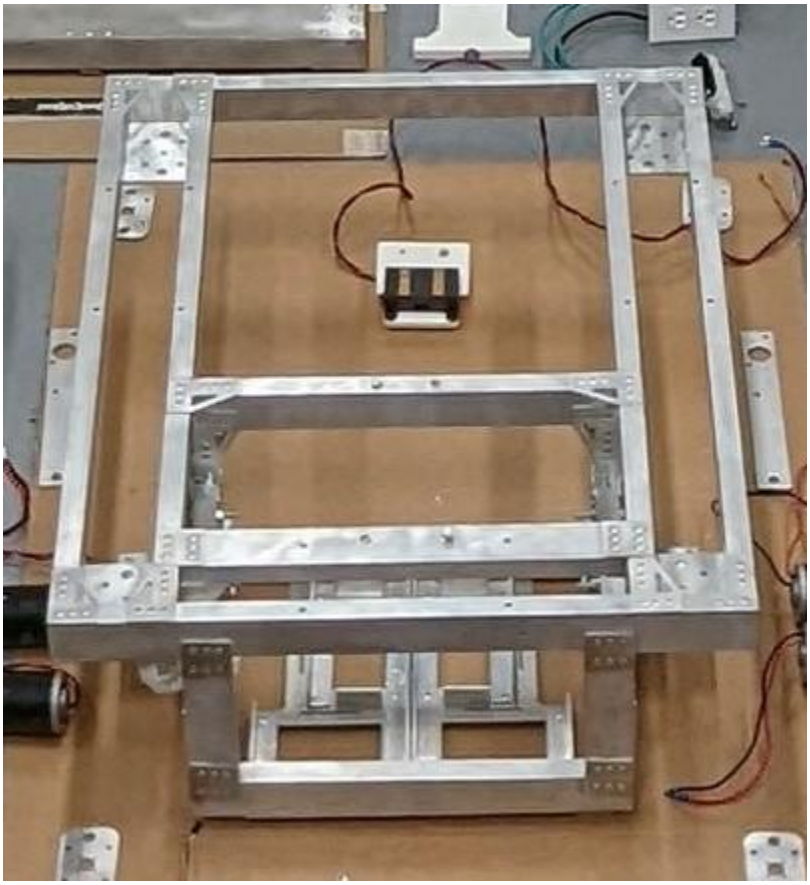


Figure 41: Constructed Chassis

4.2.1.2 Mecanum Wheel Subassembly

When a robotic base is equipped with mecanum wheels it is critical for a floor contact to be established between each wheel. In the scenario a wheel is not in contact with the floor, then the behavior of the mobile base will not be as expected and perform strangely. Thus, a spring based suspension was developed to ensure constant floor contact. Additionally, an intermediate block was utilized to guide the suspension with a linear bearing but also to tune the suspension's maximum allowable compression. Tuning the suspension enables misalignment between the individual subassemblies to be corrected and minimizes the difference in normal forces generated from the springs.

Figure 42: Spring based suspension with mecanum wheel and sprocket

The spring was chosen based on the inner diameter, spring constant, and maximum allowable compression. This particular spring is protected by hard stops to prevent plastic deformation and will bottom out during normal operation. This was specifically chosen so that the suspension would always have more room to extend if needed. About 0.5 inches can be extended before the spring can no longer extend.

The housing for the mecanum wheel was chosen to be steel such that the individual pieces could be sent to SendCutSend for proper dimensioning and then welded together.

4.2.1.3 Mecanum Axle

The mecanum axle was a custom machined piece of 0.5-inch steel cylindrical stock purchased at Home Depot. This stock has poor tolerances but performed as expected so a replacement was never ordered. This axle was machined to have two keyways and two threaded ends. The keyways are connected to ⅛ inch rectangular stock to interface with the mecanum wheel and the sprocket.

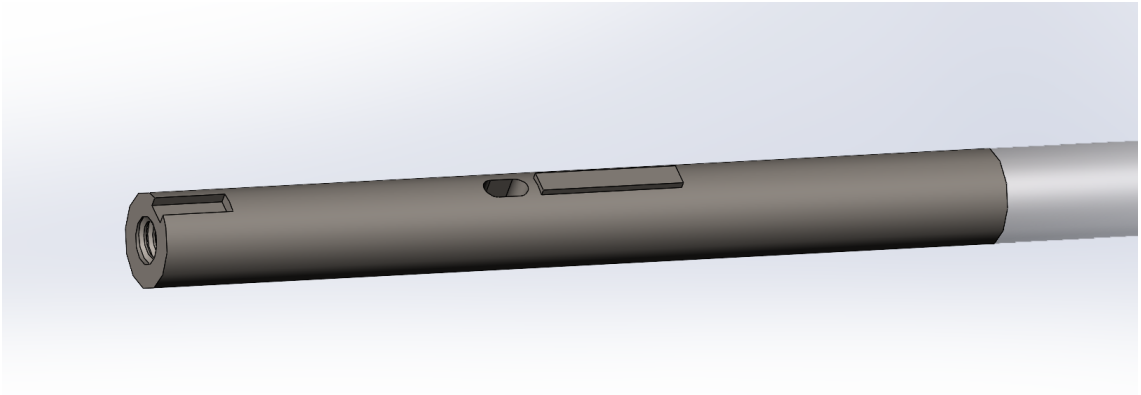


Figure 43: Mecanum wheel machined axle

This axle also has a slot for a pin to limit the depth the axle can fit within the mecanum wheel. Then, the bolts on the opposing ends will tighten everything in place. A spacer will be needed in the mecanum wheel assembly, however, to prevent the axle from sliding during strafing.

4.2.1.4 Drive Train System

The drive system consists of the four independent mecanum wheel subassemblies, a chain and sprocket system, and a tensioner / braking system.

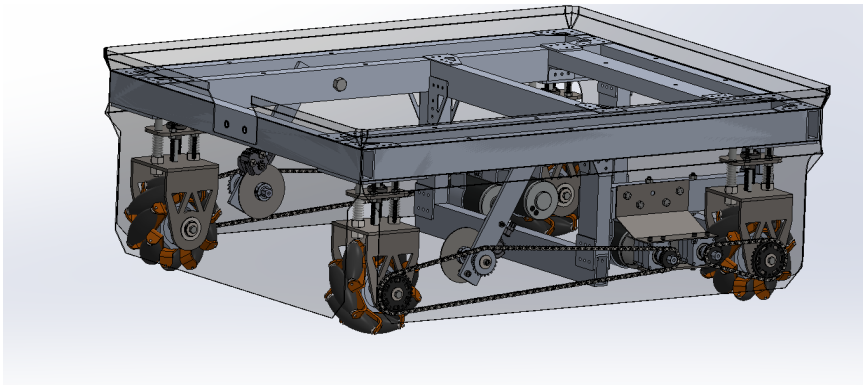


Figure 44: Drivetrain system

The chain and sprocket system was utilized to minimize width so as to leave as much possible space for the modular insert. A belt system would contain just enough additional width to interfere and a direct drive is mechanically complex to fit within the tight parameters allotted. Thus, the chain and sprocket system was opted for to reduce as much space while also maintaining a fairly easy level of implementation. The chain was chosen to face the outside of the mobile base so interacting and modifying

the setup would be easy. Additionally, the chain would need to be protected from the internal electronics, chassis, and Modular Insert had it been placed inwards which creates tight tolerances unrealistic to achieve within this timeframe. The solution was, therefore, to add polycarbonate paneling to prevent outside factors from interacting with the chain.

The CIM motors were mounted with a hardened steel bracket (red) and fence post brackets (underneath the motors). This solution was extremely cost-effective and durable. A piece of steel support was added to the underside of the VersaPlanetary gearboxes to assist in bending the steel.

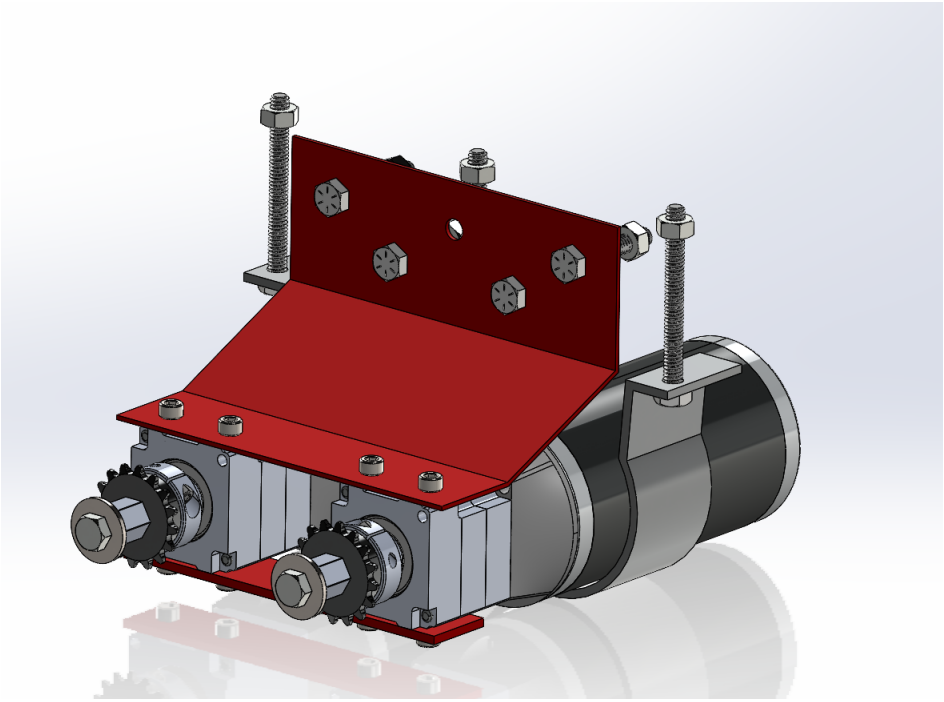


Figure 45: Motor mount system

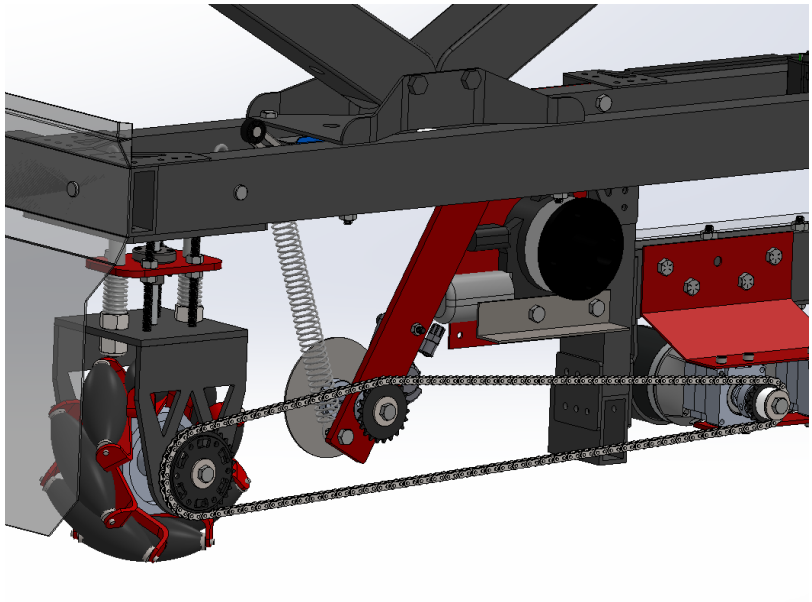


Figure 46: Tensioner and braking system

With a chain and sprocket system, a tensioner is essential to keep the chain taught. Chain is susceptible to stretching with use and a passive tensioner, such as a spring, will work to keep tension. For a mecanum base drive, adequate tension is necessary to perform complex and high friction movements, such as strafing. This tensioner was also designed with a brake caliper and brake rotor inline with the tensioner sprocket. This brake caliper is capable of clamping to the rotor and thus preventing the chain from moving. The brake caliper is activated by the window motor as seen to the immediate right of the tensioner lever arm. The window motor was never fully integrated electrically or programmatically.

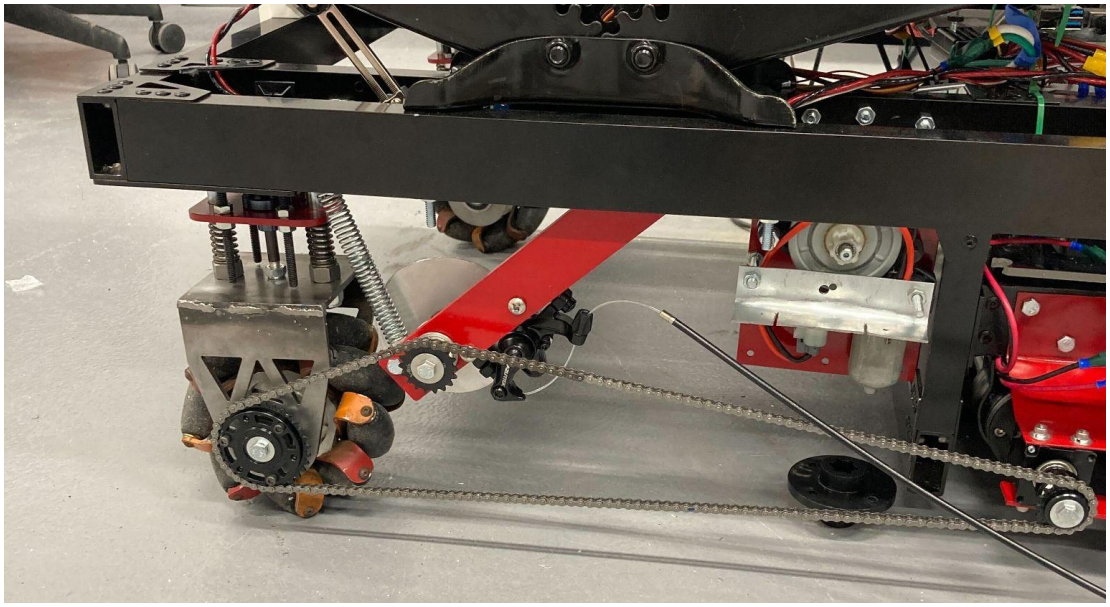


Figure 47: Constructed drivetrain system

Here the tensioner is keeping the chain taught with the braking system in place. The window motor is mechanically attached but not wired. Additionally, the mecanum wheel subassembly has been built.

4.2.1.5 Elevation

The elevation system consists of three individually actuated scissors jacks powered by 775pro motors. These motors were geared down appropriately and suited with a quadrature encoder. With each scissor jack independently actuated, this enables the red tabletop to not only vertically elevate, but also to pivot laterally.

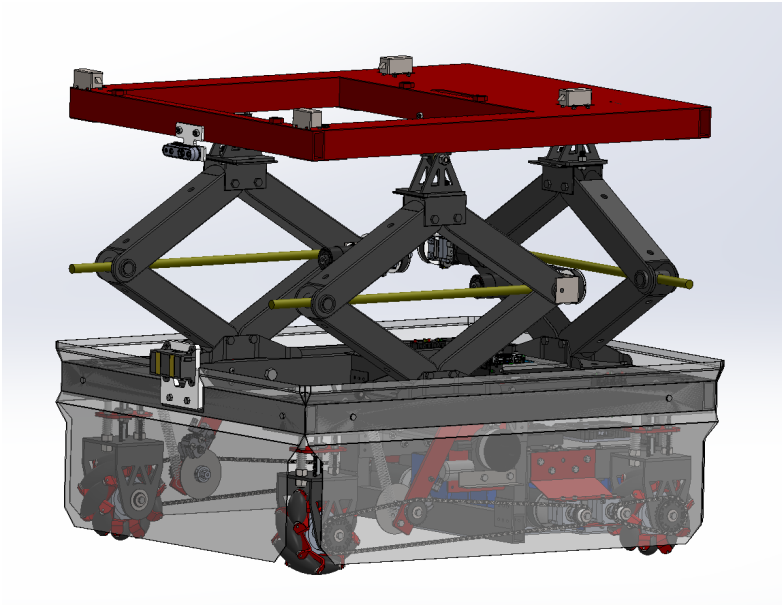


Figure 48: Elevation system

This feature is attributed to the elevator alignment system as well as the scissor jacks having a large tolerance of motion. Indeed, the scissor jacks were commercial standard and purchased from a retailer of Amazon and were not created with robotic level precision. So, the jacks do not need to slide across the tabletop to enable pivoting.

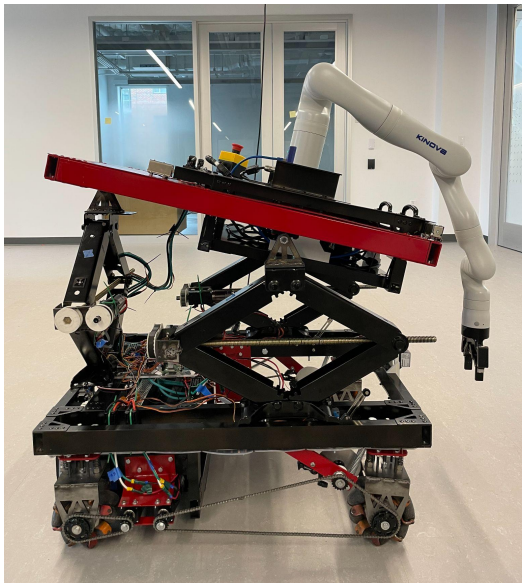


Figure 49: Laterally pivoting to reach low

The motors were attached to the scissor jacks through an intermediate bracket welded into place. The bracket could then mount to the 775pro motor. The connection between the 775pro and the scissor jack utilized a pulley belt setup to account for misalignment. Indeed, with the belt in tension, there will still be rotation as the belt is passively working to be aligned. Additionally, the pulleys were 3D printed to have custom mounting and to save cost. The scissor jack was also post manufactured to be able to fit within the Modular Insert's pathway. The top brackets were redirected and cut such that they were fit to size.

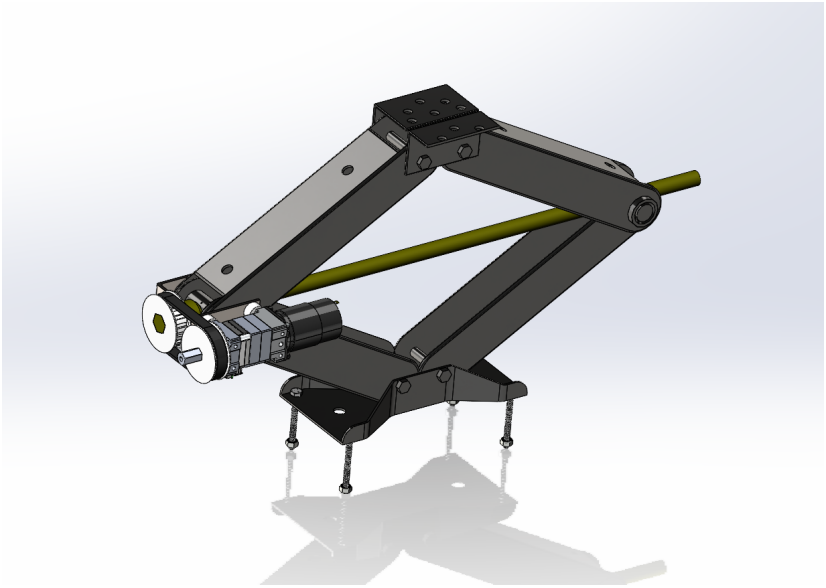


Figure 50: Individual scissor jack apparatus

The elevator alignment system, mounted directly above the scissor jack apparatus, interfaces with the tabletop through a pillow block.

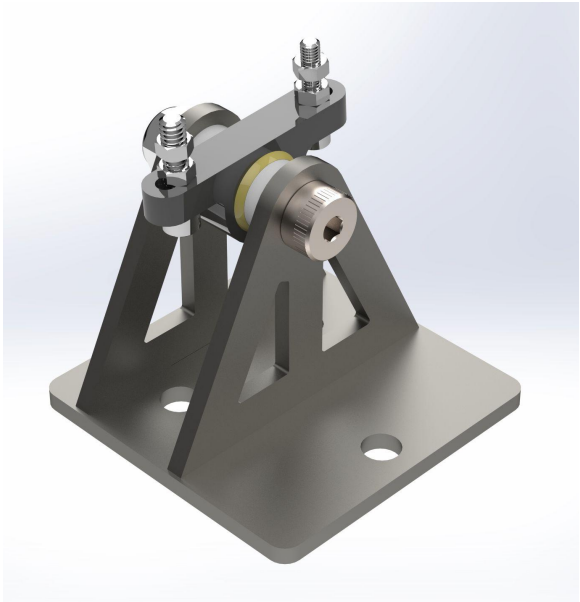


Figure 51: Elevator alignment system

The pillow block is capable of rotating about the shoulder bolt it is mounted to as well as around itself with its intermediate plastic bearing. This gives many additional degrees of misalignment that can be handled without fear of metal deforming. The pillow block is rated to handle 90lbs of load, so with three of them in place there should not be an issue with normal operation.

The elevator system results in an additional 24 inches of vertical workspace. However, the bottom 6 inches are restricted to prevent the scissor jacks from compressing too far and damaging themselves. Indeed, the lowest the jacks can reach before bottoming out is 5 inches, giving 1 inch of clearance for the signal to be delivered. The ME8108 limit switches were used as the interrupt to detect when the jacks are too low.

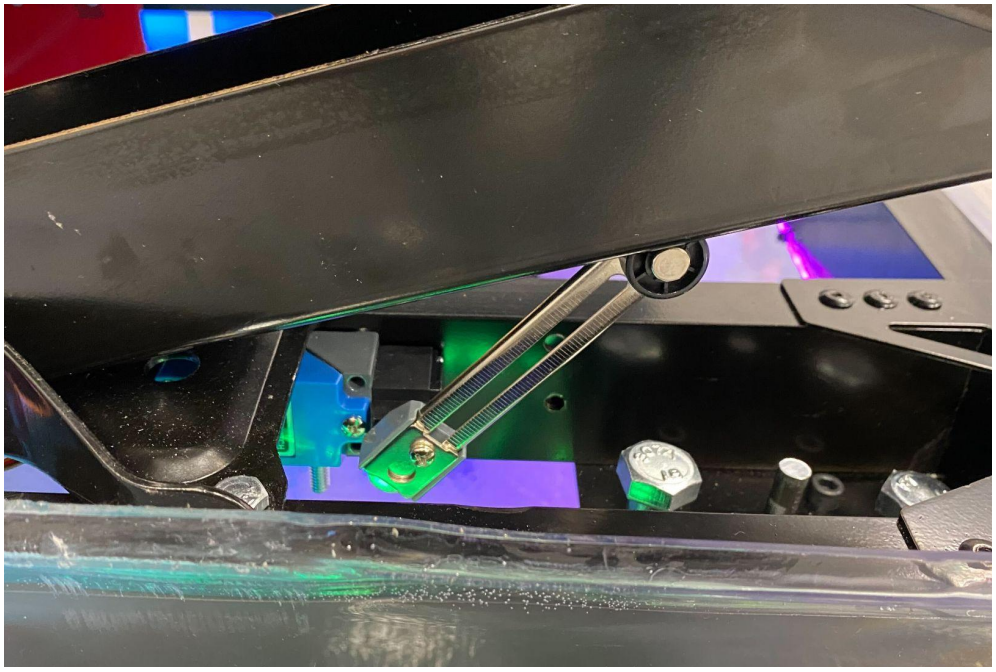


Figure 52: ME8108 limit switch

These are industrial grade limit switches capable of being adjusted to an ideal location to trigger, and are fitted with a roller end that rolls off the scissor jack when compressed.

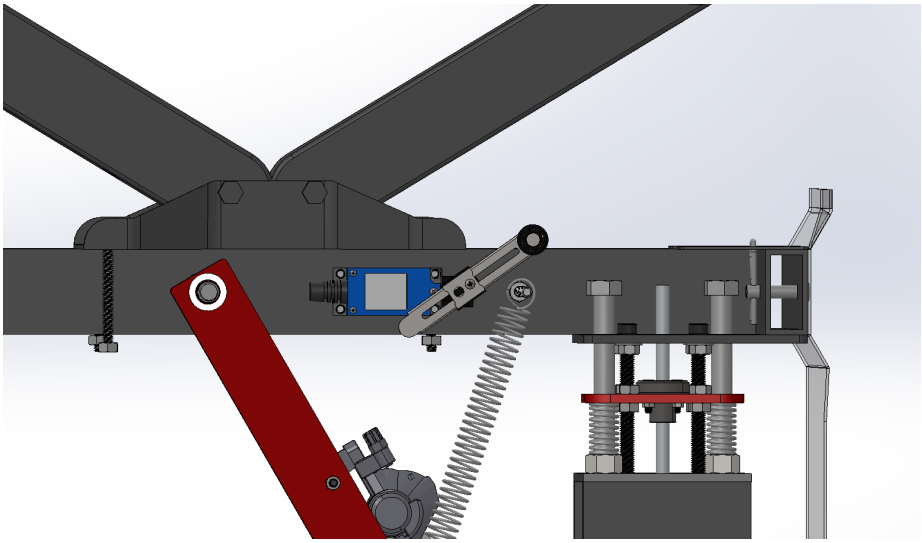
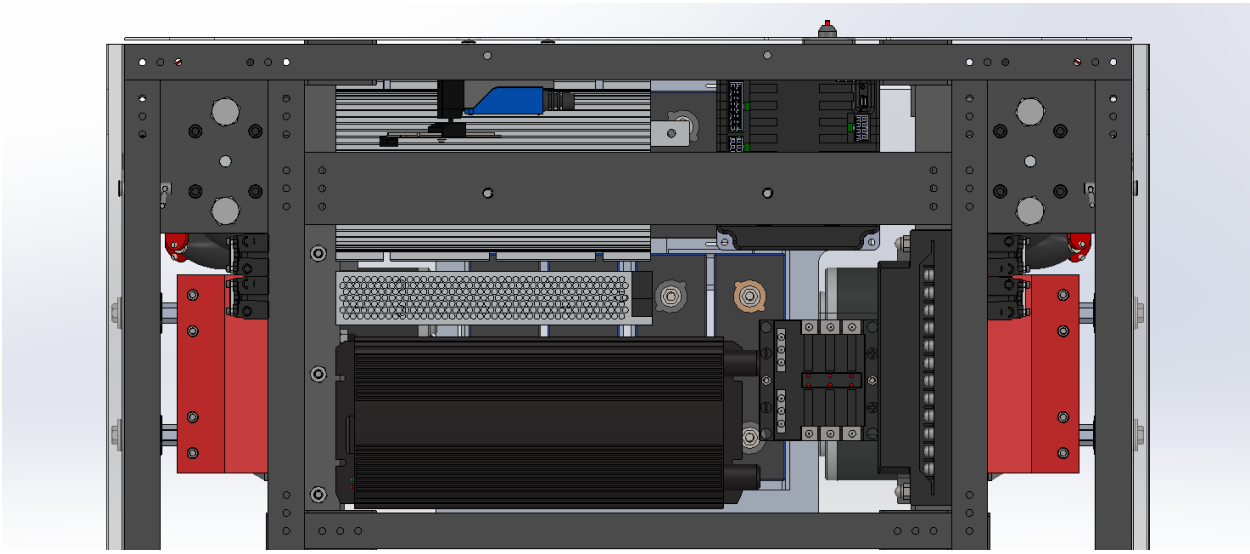


Figure 53: Cross-sectional view demonstrating limit switch

4.2.1.6 Electronic Housing

The electronic housing was mounted to the chassis through an acrylic panel with bolts, rivet nuts, and zip ties. The bare minimum electronics were placed on the lower panel of acrylic where there is more protection and room. The top panel was used to hold the Raspberry Pi and PCA PWM board for easy access and modification. Below, a top view and isometric view of the electronic housing area are illustrated in Figure 54.



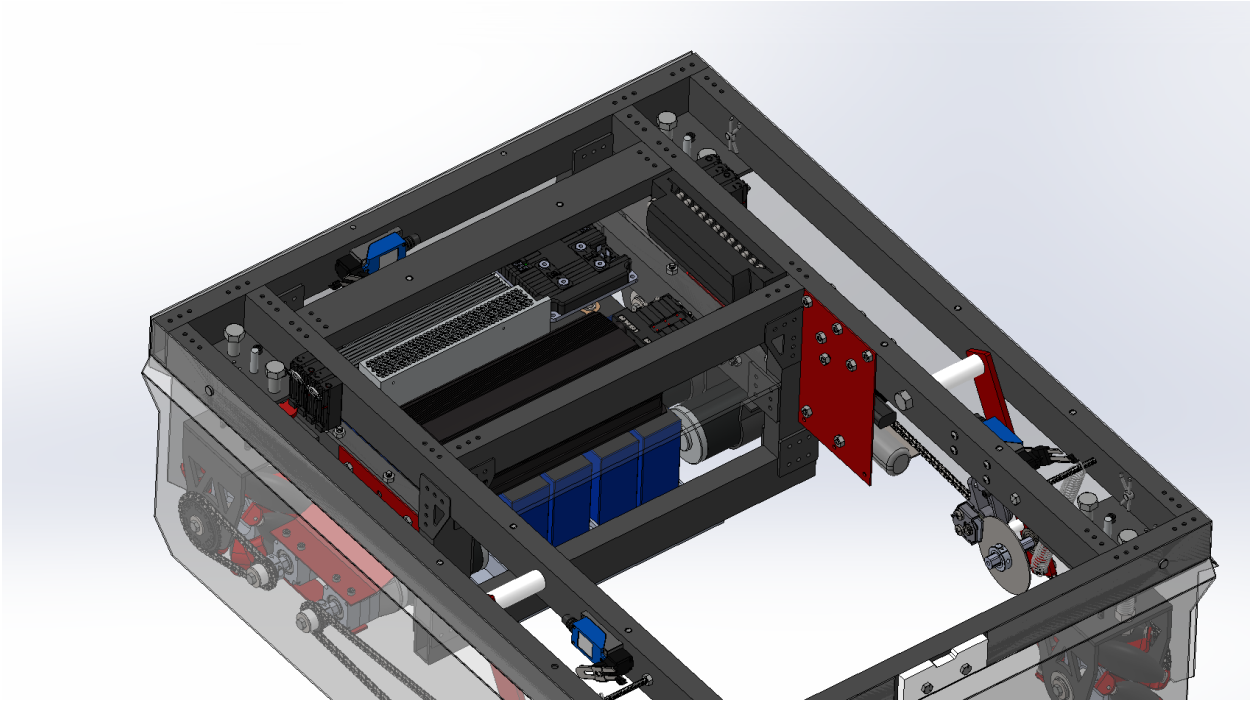


Figure 54: Electronic housing in chassis

Not shown is the top panel holding the Pi and PCA board as they interfere visually with the electronics on the bottom acrylic sheet. Below the bare minimum electronics are the batteries arranged in a fashion to wire in series easily while also giving clearance for the CIM motors. The housing is incredibly close-knit and thus has issues with electrical noise and modifying individual pieces once they have been put in place.

The accuracy of the electronic pieces was not necessary to create the CAD model but an effort was made to create them to be as realistic as possible to achieve high accuracy of the model.

4.2.2 Modular Insert

The Modular Insert is one of the flagship items of this project and innovated the concept of combining a single unit to house the controller and arm into a modular piece of hardware. Figure 55 demonstrates the final design.

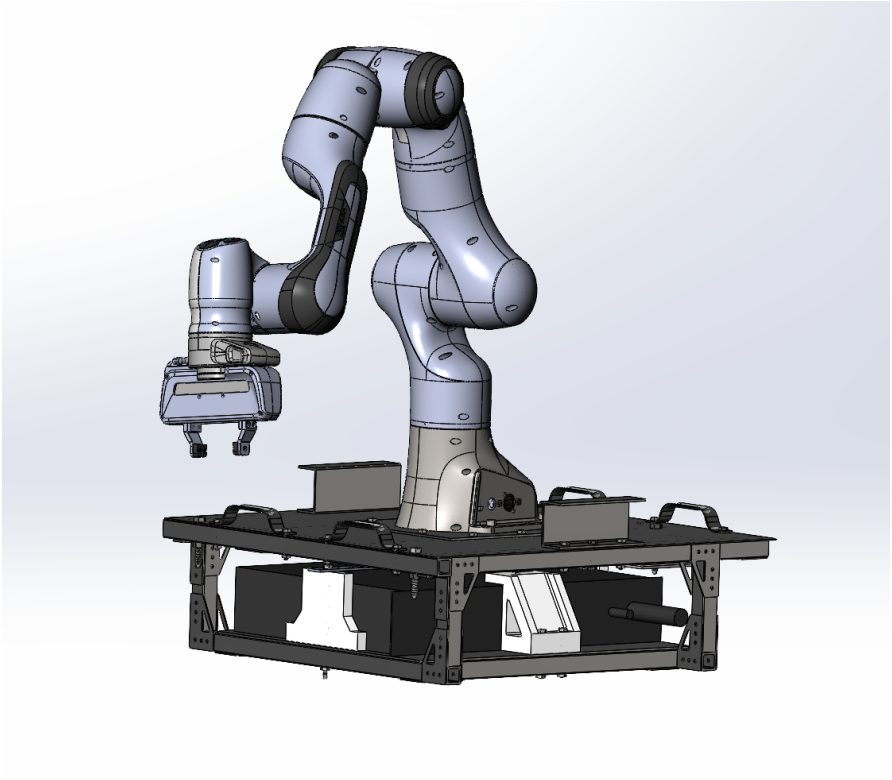


Figure 55: Modular insert

In this design the Franka Emika Panda manipulator is mounted to the modular insert along with the controller secured in place below. The controller is secured with 3D printed brackets that are bolted into place. The arm itself mounts with fitted hole placements in the top panel of the insert. This particular insert is fit for the Panda arm such that only the Panda's controller will fit with this design. In an ideal world, there would be a modular insert for every arm so each controller can be custom fit. Since the controller is required for the operation of these arms, it is critical the connection is secure and reliable which is why a custom and tailored fit is ideal.

There are also elements of mobility incorporated into this design such as handles for removing and placing the insert and C channel for sliding the insert into a home base.

4.2.2.1 Mounting System

The mounting system consists of the Modular Insert, the tabletop (red panel), and the solenoids. The solenoids are used to lock the modular insert in place on the tabletop.

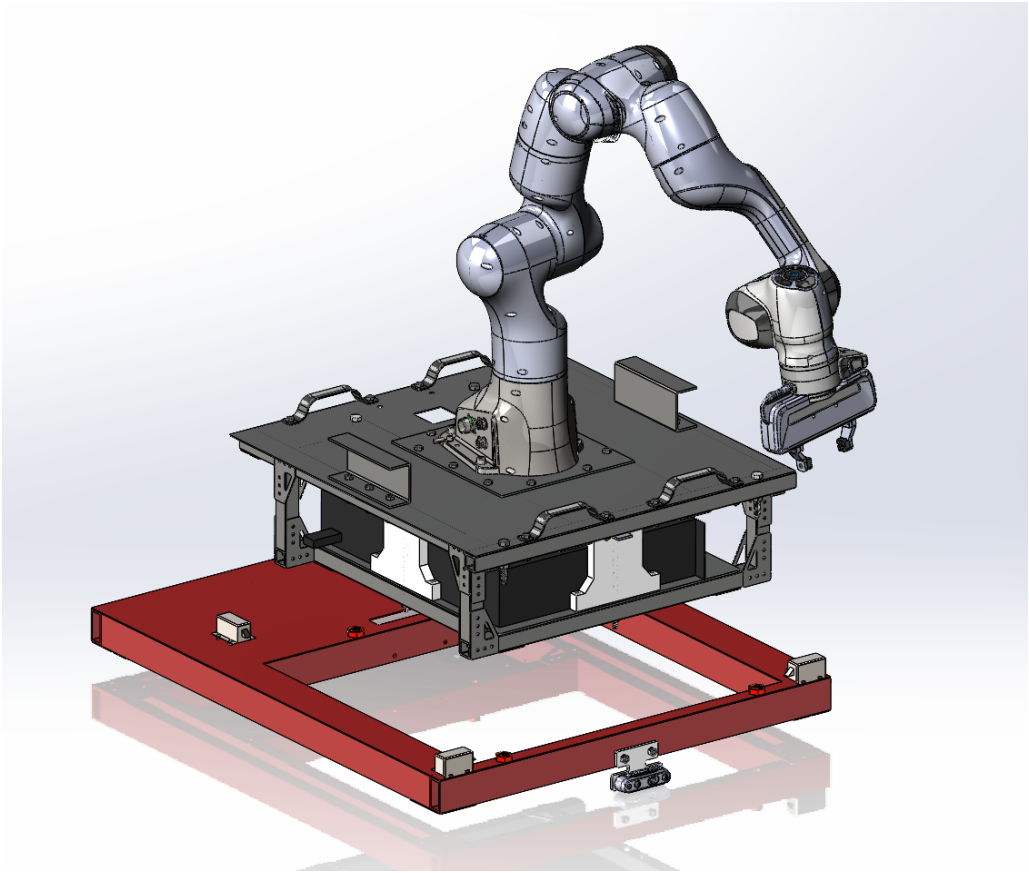


Figure 56: Tabletop and modular insert

Here, the modular insert will vertically drop into place with the tabletop with the assistance of guidance pins and bushings. The bushings, as seen below, are machined pipe fittings that screwed into the tabletop. The guidance pins were bolts that were machined into rounded ends.

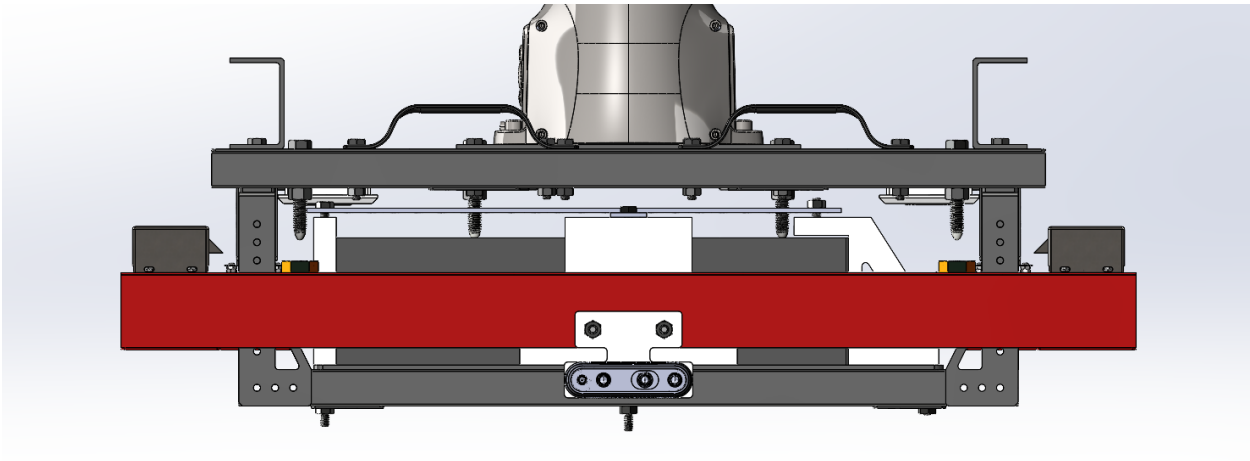


Figure 57: Guidance pins inline with bushings

The guidance pins are on the left and right sides of the modular insert and fit into the goldish red bushings. In practice, the guidance pins had to be slightly adjusted with a mallet to create a seamless connection with the tabletop.

The Intel Realsense d435 was also mounted to the tabletop to generate 3D point clouds. In this location, the camera would be able to observe the increased elevation and be capable of understanding that it is being elevated in its mapping generation. This spot was also chosen as it's directly above the robopads and could be used to align the mobile base into a charging station.

The solenoids interface with the modular insert through the box tubing. Indeed, since the box tubing is hollow, this enables the solenoids to have their unretracted end fit within the tubing. This will lock the modular insert in place unless the end is retracted. The solenoid ends are pitched at a 45-degree angle such that a bar vertically dropped on top will reactively cause them to close and subsequently extend when there is no load. The bottom face is flat, which prevents the modular insert from being lifted out of the tabletop without retracting the end of the solenoids. Figure 58 below demonstrate this process as well as highlight the guidance pins and bushings.

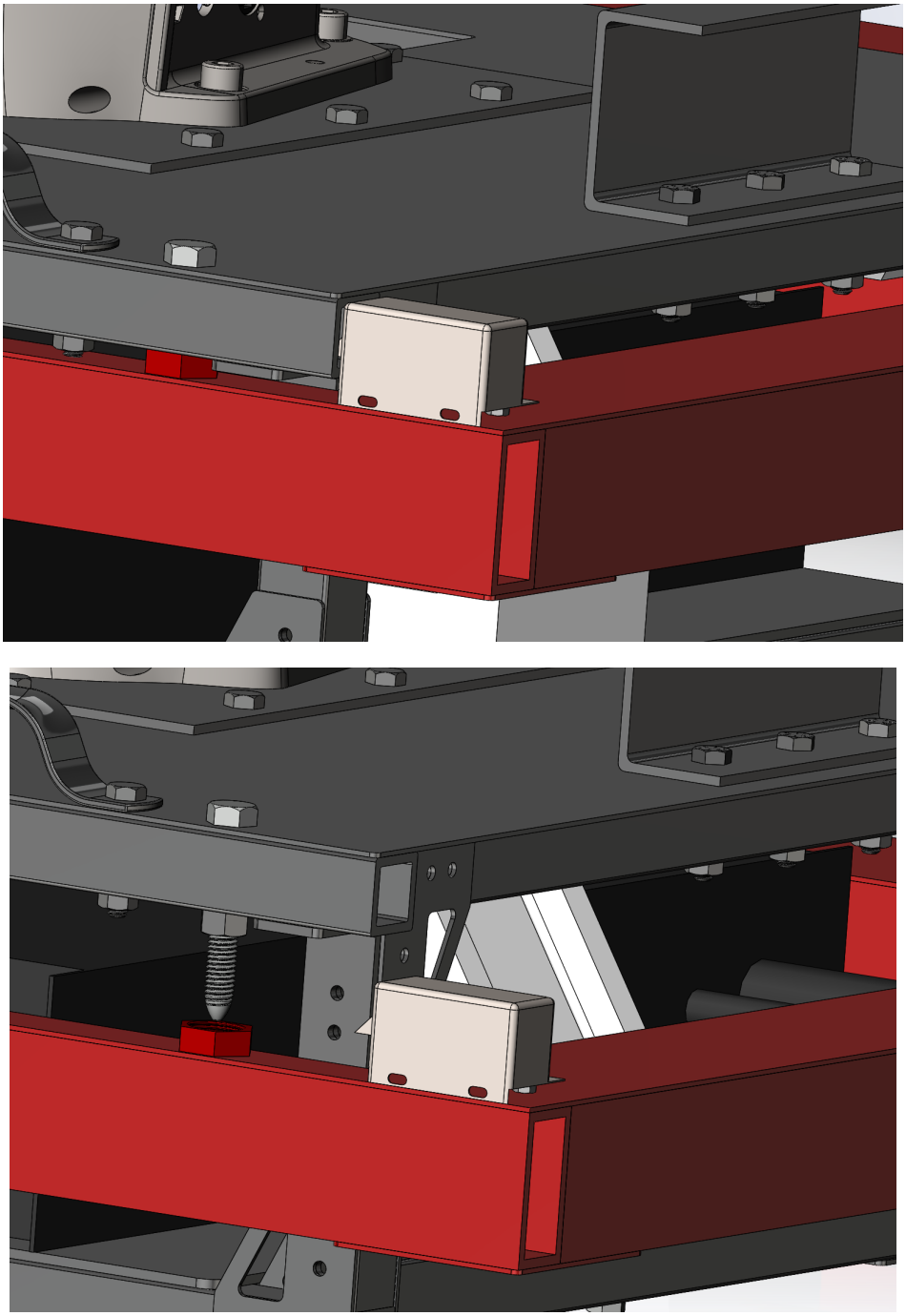


Figure 58: Solenoids interacting with modular insert box tubing

4.2.2.2 Universal Plug

The universal plug marries the necessary connections for the robotic arm (power, specific controller protocols) to a single plug that interfaces between the modular insert and the tabletop.

Figure 59: Universal plug

The alignment between the plugs is critical and is ensured by the accuracy of the bushings and the guidance pins. In practice, there was not a single instance of pin misalignment between the universal plug.

The bottom plug is secured in place with a custom 3D printed fastener in ABS. ABS is one of the most robust commercial plastics available so it was ideal for the application. The top plug connects to 3D printed Nylon which compresses when in contact with the bottom plug.

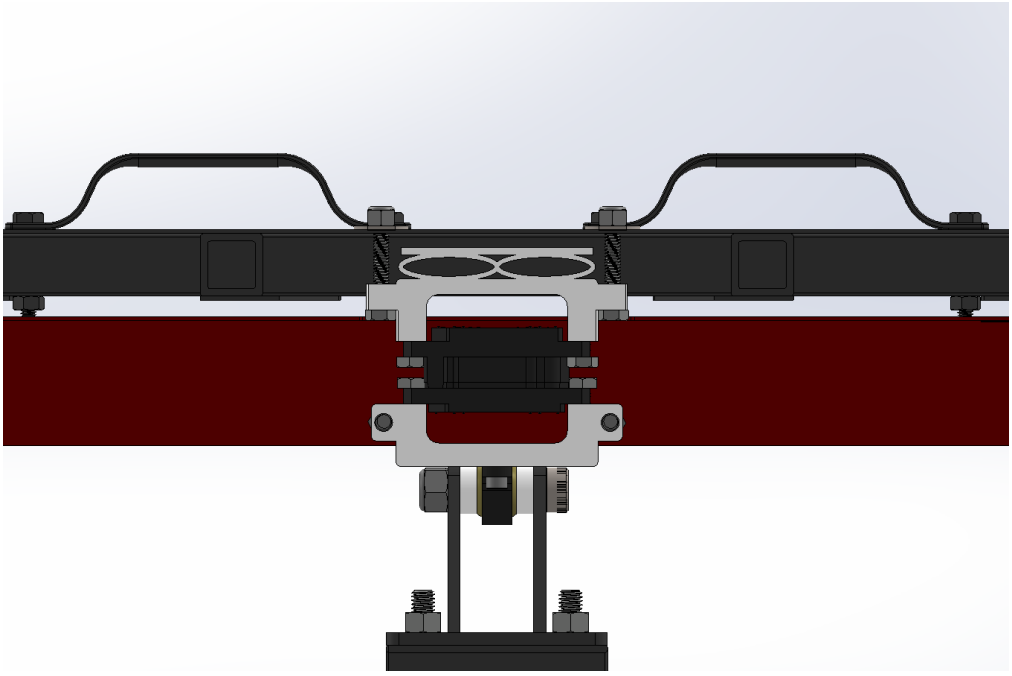


Figure 60: Universal plug & connections cross-section

Here, the top connection connects to the modular insert and extends slightly too far to make contact with the bottom connection. The top connection compresses under the weight of the modular insert and ensures a resistive force on the plug. This works to secure the pins together and have a solid connection.

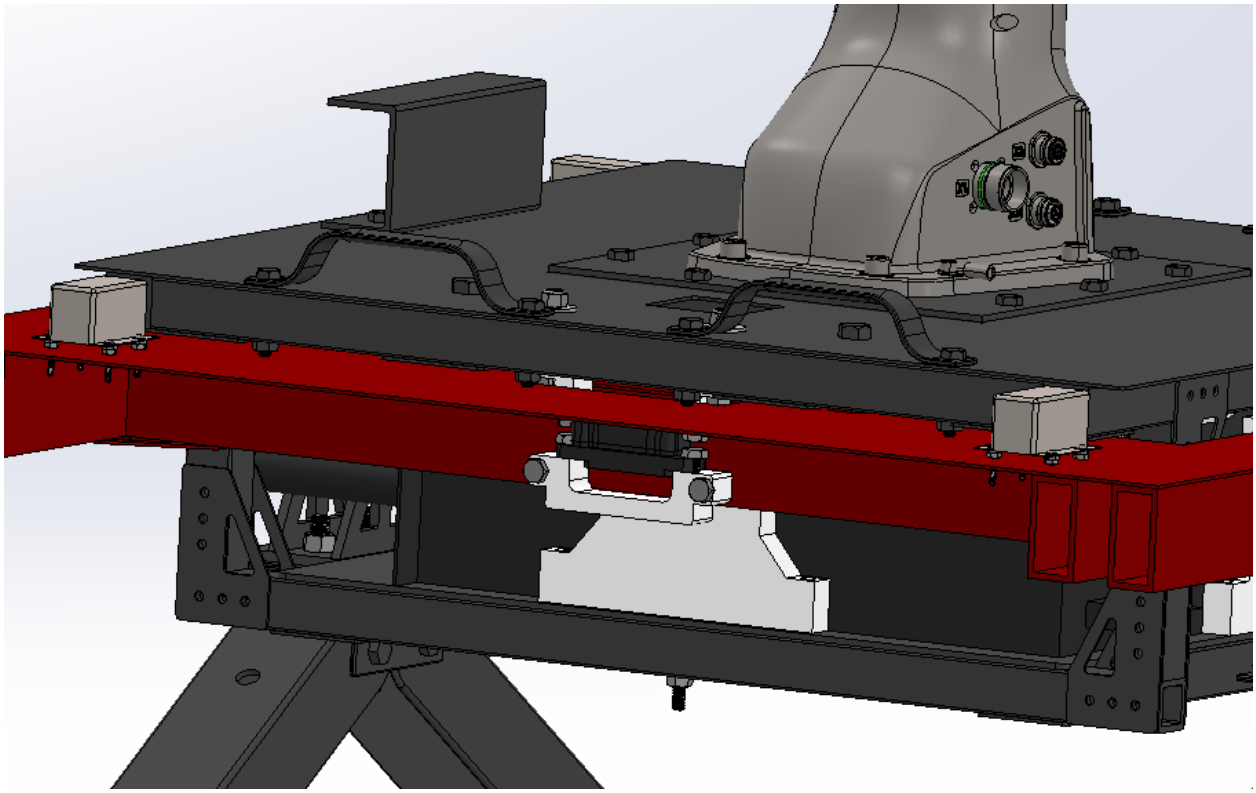


Figure 61: Isometric view of universal plug

4.3.3 Home Base Charging Station

The implementation of the home base led to some loose design constraints when referring to the actual makeup of the structure. The only necessary constraints to be included were a free space under the insert such that the robot could fit under and lift into a secure position, a sturdy design to steadily hold an insert with an attached arm, and a location for the charging pads to be accessible to the robot.

When determining the overall structure for the base, it was outlined that the shape would need to follow a forked pattern to allow the insert to rest with a free underneath. This drove the decision for C-channel wings on the insert for smooth placement and removal of the insert in the base.

The composition of the frame of the base was not imperative as long as critical dimensions were met, therefore a decision to use 2’x4’ nominal beams as the primary construction material as they were the most accessible and cost-effective option. The dimensions and layout of the base are outlined below in Figure 62 and a cut-list can be seen on the left based on a purchase of 5 2’x4’x8’ nominal beams.

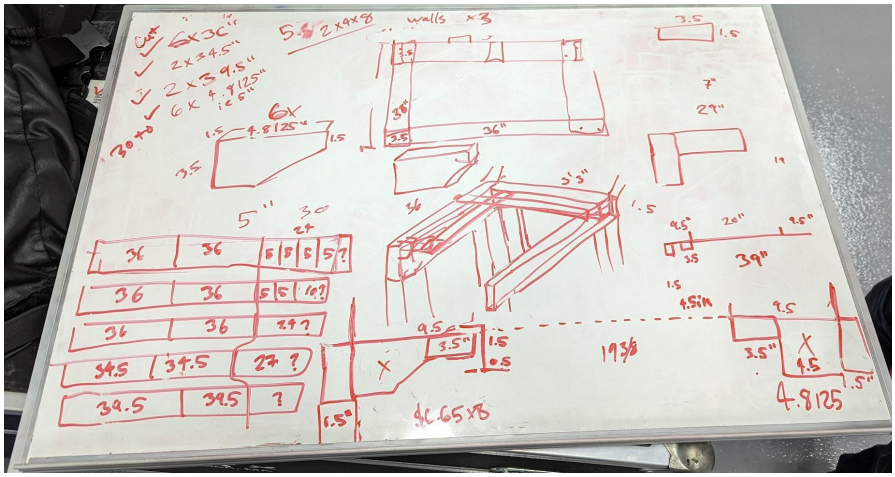


Figure 62: Home base charging station initial design

The home base was then constructed and painted as seen in the image below. The final revision of the base included the three base boards for rigidity as can be seen attaching the four legs of the base together. Excess lumber was then used to mount the charging pads to at the appropriate height for interfacing with the robot. The forked section in front was tested prior to placement of an insert to determine if the base was sturdy enough as to not drop the inserts with attached manipulators. This was done by supporting a team member at the location where the inserts' weight would be transferred. This test demonstrates the ability for the base to support in excess of 220lbs.

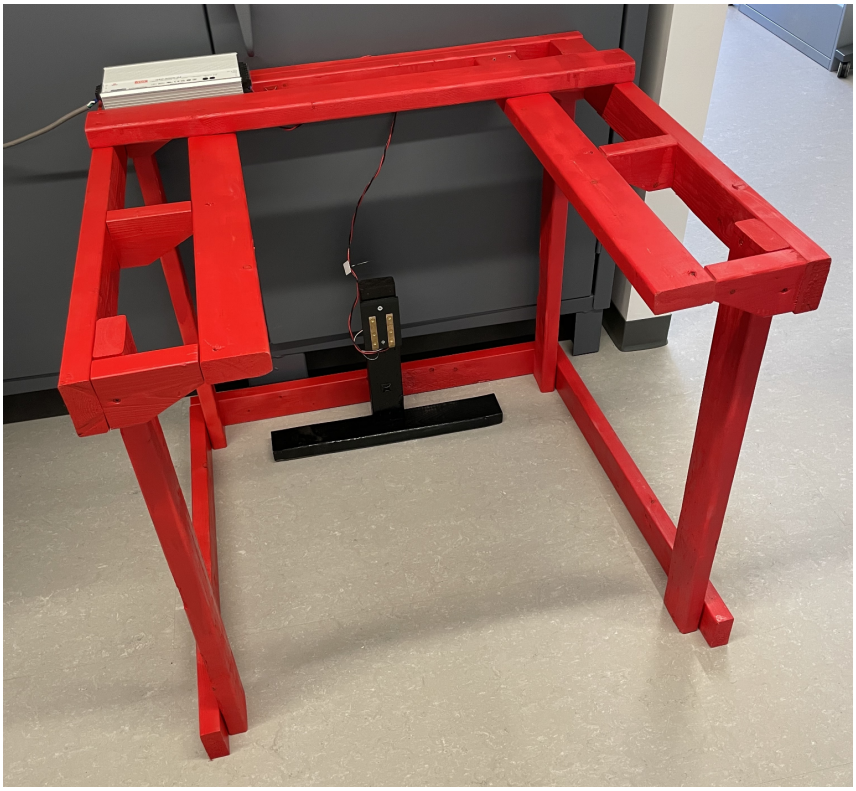


Figure 63: Home base charging station

4.3 Electrical Implementation

4.3.1 12V Converter

Implementing and testing the electrical system for the robot occurred by implementing each component in steps, verifying the system worked, and then slowly adding more components until the entire system was constructed. To start, the batteries were tested with the 12V converter and the CIM motors. The batteries were wired to the 12V converter which was then wired to the CIMs. The converter was switched on and once switched on, the CIMs spun as expected. Wiring the CIMs in reverse polarity spun them in the other direction. This verified that the 12V converter worked. It also doubled as a way to grease the gearboxes, so the test was run with the 775pro motors as well.

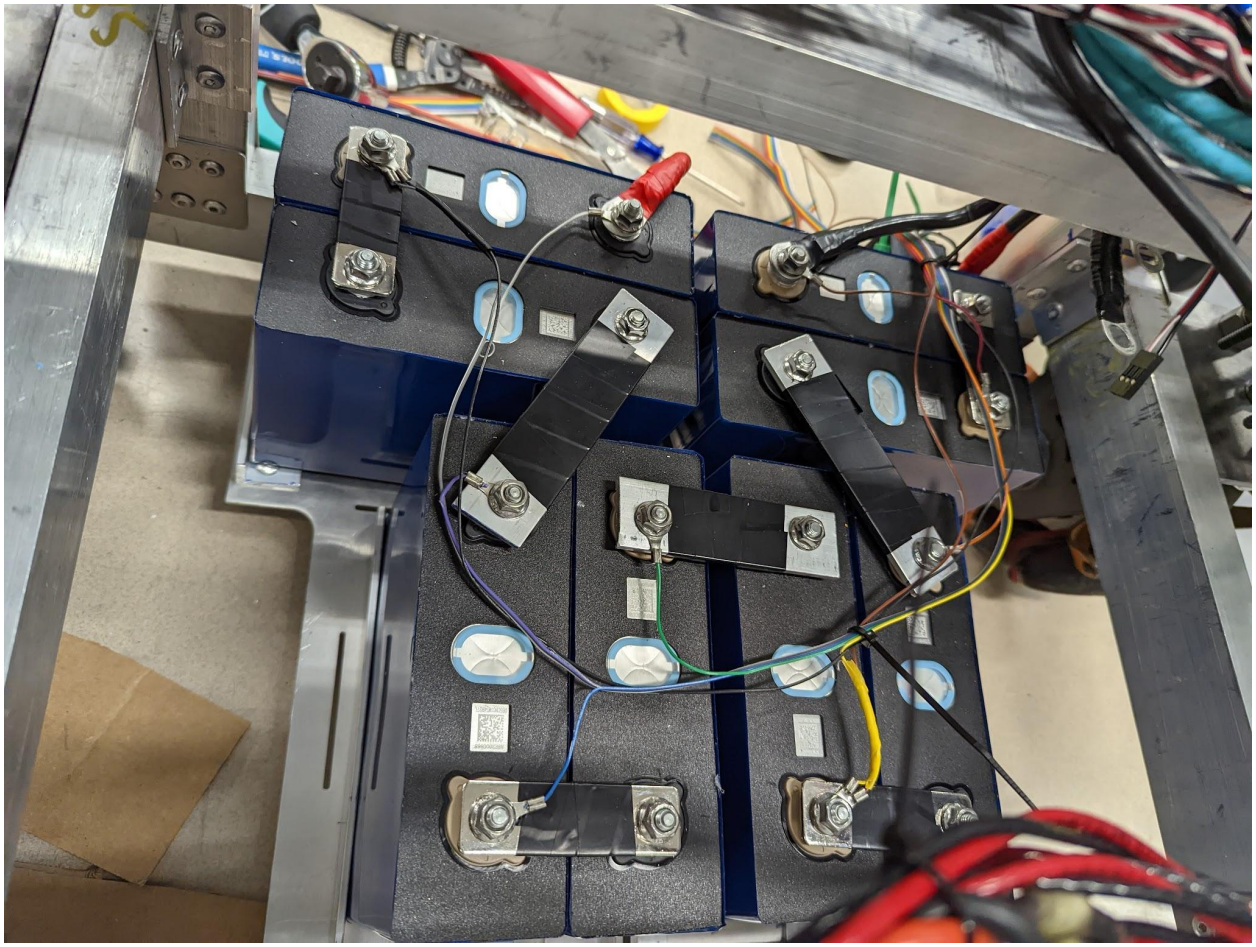


Figure 64: Wired up batteries

4.3.2 Inverter

After running the CIMs and 775pros, the inverter was tested. Not wanting to risk damaging the arm, the effectiveness of the inverter was tested using a laptop and seeing if the battery would charge while plugged in. The arms all have onboard AC/DC converters, so using a laptop would provide a similar feature. The inverter and laptop were added to the motors and the 12V converter and switched on. The laptop drew power as expected and the motors spun as expected, showing that the inverter worked as well.

4.3.3 5V Converter

Next was to verify that the 5V converter worked with the whole system. To do this, the test setup remained the same as before, but the 5V converter was wired and connected to the Raspberry Pi. Running the system this time ran the motors, charged the laptop, and turned on the Pi, which was able to connect to a monitor and keyboard and run as expected, thus proving that all the converters could run simultaneously.

4.3.4 BMS

Next, the BMS and charging circuitry were added. The BMS was wired in series with the batteries and the cell balancing wires from their respective pins to each cell. The converters were wired to the BMS as well. Also, the Robopads were wired to the charging port of the BMS. The test setup remained the same as before, but this time the charger was turned on and connected to the BMS. The converters and other electronics performed as before and the batteries charged at the predicted 20A that the charger could supply. Also, as the batteries charged, the cells were individually balanced so that their voltages stayed within 10mV of each other as specified in the BMS software.

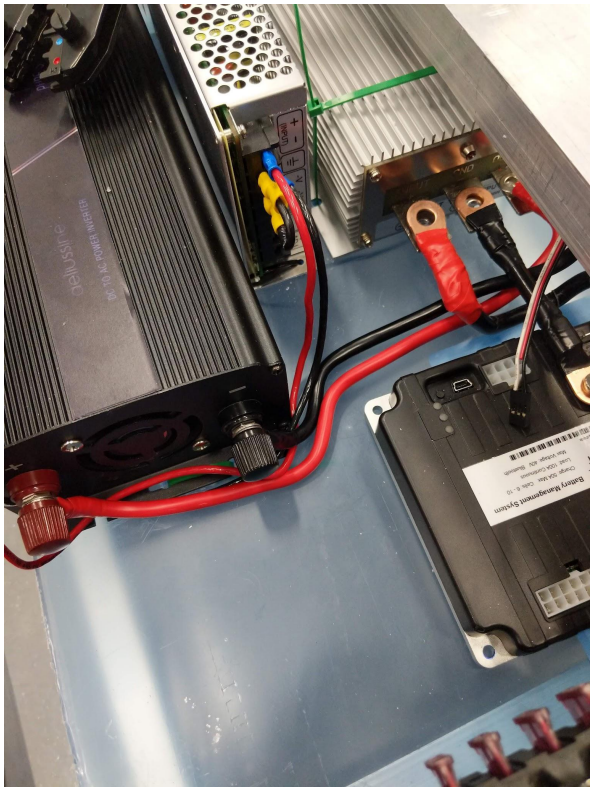


Figure 65: Initial control board wiring

4.3.5 Control Components

After that, the motor controllers were placed in series with the 12V converter and the motors to test if they could control the speed and direction of the motors. The PWM signal from the Pi was simulated using a function generator in the lab. By varying the output of the function generator, the speed and direction of the motors could be adjusted.

The next step was to get the Pi to control the motors. To do this, it needed the I2C PWM driver board. A script was written that would vary the motor effort between 0% and 100% for both forward and reverse. The Pi would communicate over I2C to tell that board to output the corresponding PWM signals which were then sent to the motor controllers. Running the script allowed for the motors to work as expected.

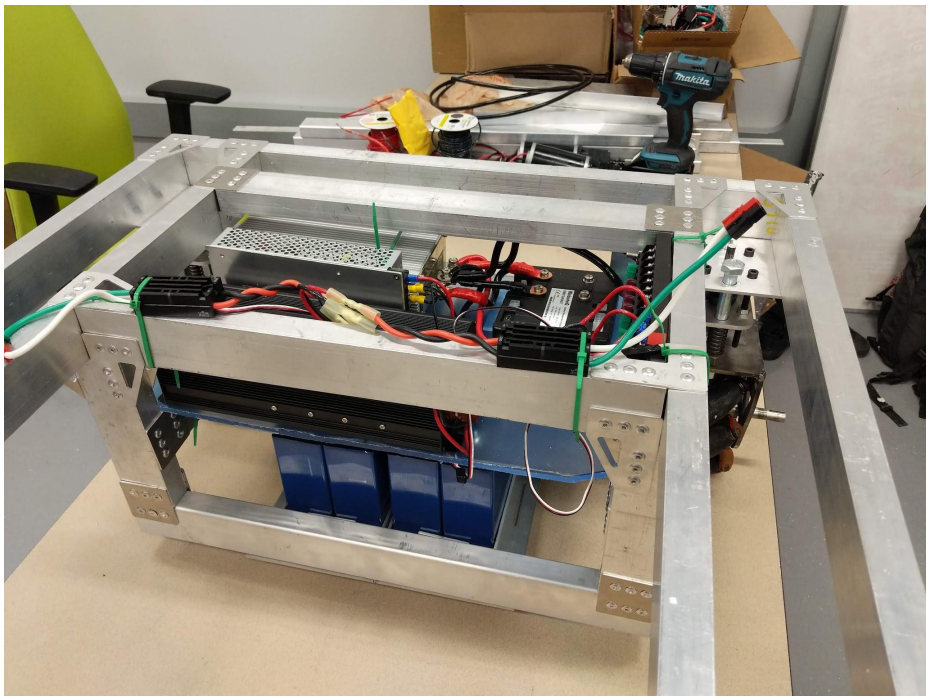


Figure 66: Wiring of motor controllers

To test the jack calibration limit switches and encoders, the elevator jack 775pro motors were run and the data was read by the Pi while the entire robot was on. The Pi was able to read the current number of encoder ticks as well as reset the count to zero when the limit switches were hit as expected.

Next, to test the solenoid board, it was wired to a power supply separate from the robot and connected to the solenoid boards. Giving the board a 5V signal to emulate the Pi or pressing the button caused the solenoids to activate. If the button was pressed, they held open for about 8 seconds before shutting, close to the 7 seconds desired in the design since the solenoids were rated for a maximum of 10s continuous current. However, implementing the solenoid board on the robot led to some problems. When the button was clicked, there would be some interference with the PWM signal sent to the jack motor caused by a ground loop created with the wiring of the board. This interference would cause the jack motors to twitch undesirably. By redoing the wiring of the board and placing RF ferrite chokes around the PWM wiring, this problem was eliminated.

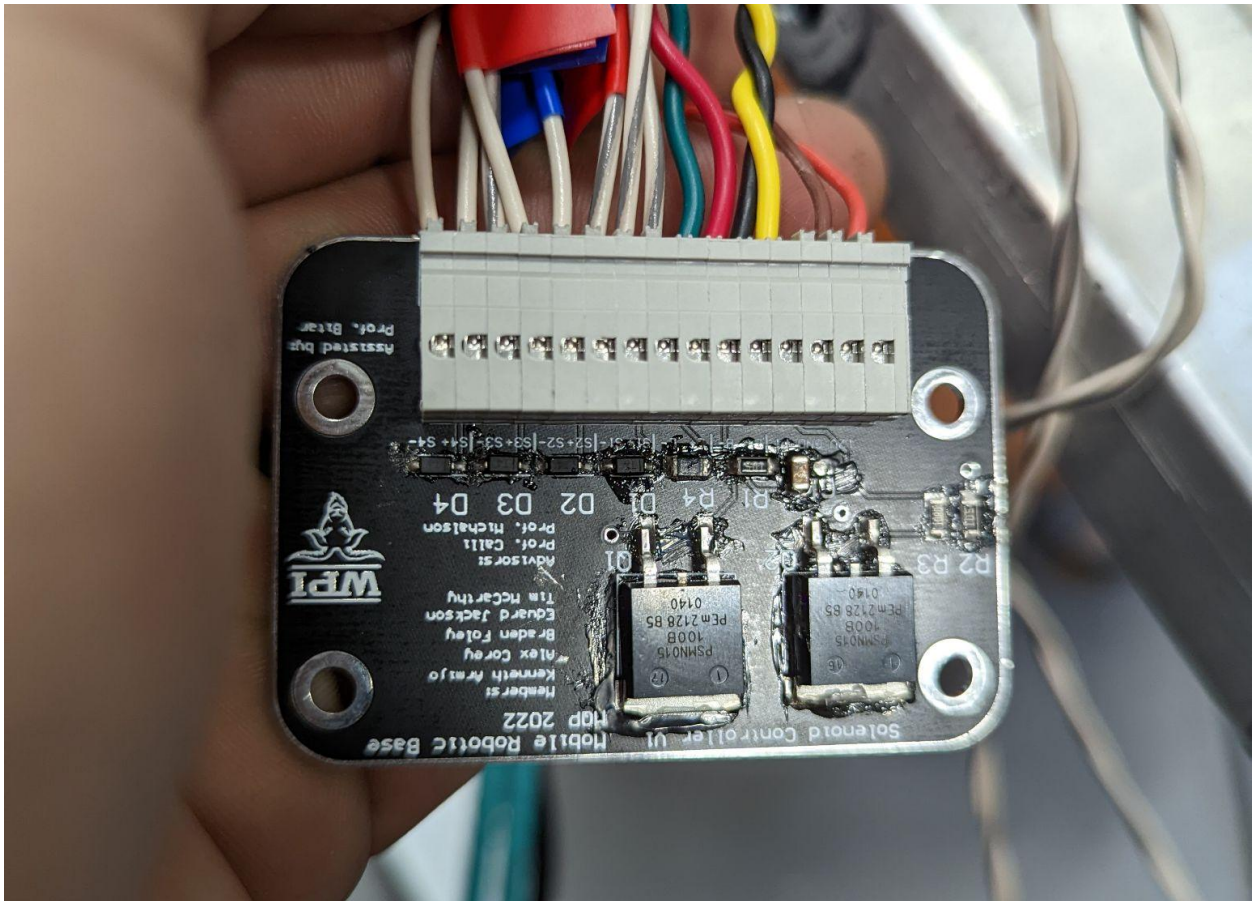


Figure 67: Soldered PCB

To test the indicator lights, a script that would make them turn rainbow colors and chase each other around the base was implemented. The lights were plugged into the 5V converter and the signal was routed to the Pi. Upon turning on the base, the lights behaved as predicted.

Finally, to test the arm and its controller, the whole robot was turned on and the arm and controller plugged into the Ethernet and Power ports on the insert. By using MoveIt!, the arms could be controlled remotely from the host computer, proving that the power and communication between base and arm worked.

4.4 Software Implementation

4.4.3 Simulations

Before the base was constructed, simulations were used to set up ROS environments, develop data and control flows, and evaluate system and subsystem functionality. To achieve this, the simulations were built-in Gazebo, an open-source simulation environment with an integrated physics engine [13]. There were three parts to building simulations of the whole robot: base, arm, and insert simulations. The following sections break down how each of these parts was implemented.

4.4.3.1 Base Simulation

The first and most simple simulation was that of the base. It needed to be able to translate in the X, Y space much like any robot would on a flat surface. Since mecanum wheels had been chosen for the drive train, they also needed to be included in the base simulation. However, mecanums are notoriously difficult to simulate due to their inherent diagonal roller design; to properly simulate one wheel, the coefficients of friction of all rollers must be accounted for and independently simulated. This hurdle was determined to be too difficult to implement due to the team's lack of familiarity with Gazebo, so another alternative was developed: using joints connected to the world link. The idea was to create three invisible joints that could be used to control the robot as if it was driven using mecanum wheels. Two prismatic joints would control the robot's position along the X and Y axes and one continuous joint would control the robot's Z rotation, effectively giving the simulation the same capabilities as a mecanum drive train. Furthermore, as the chassis had not been fully designed yet, a simple box served as the simulation's chassis. The definition of this robot went into its Unified Robot Description Format (URDF) and can be seen in Figure 68 below.

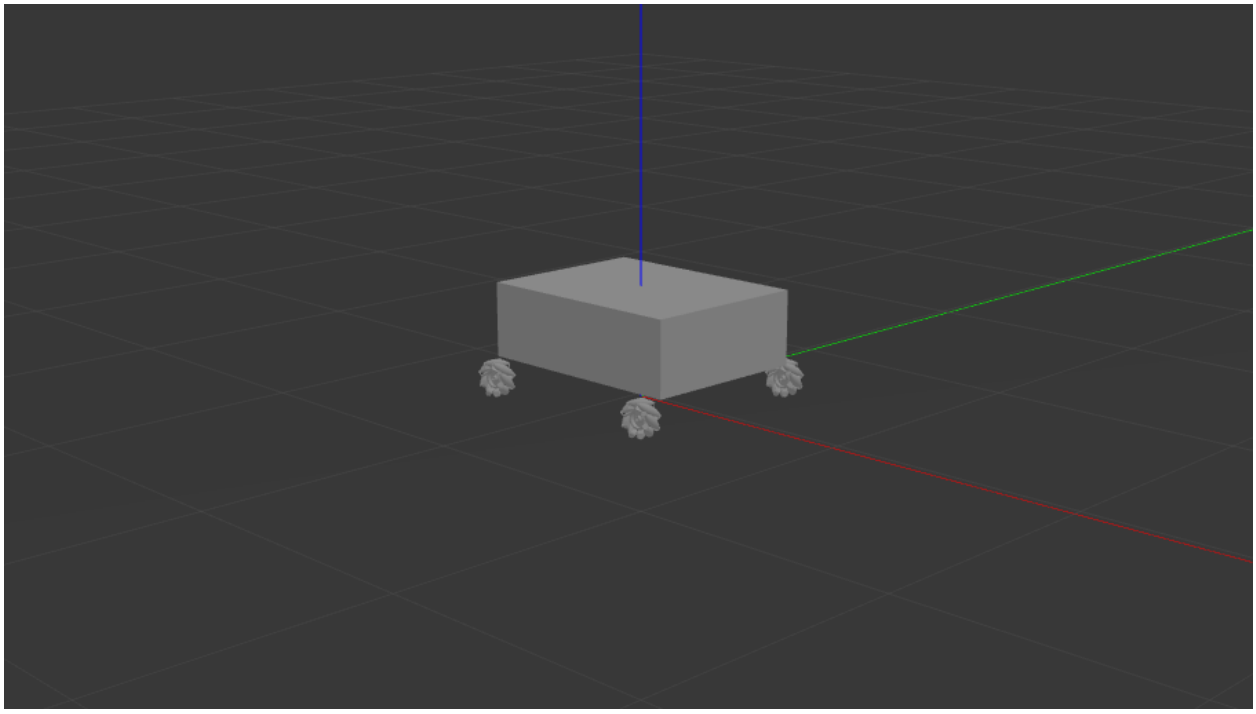


Figure 68: Early simulation of the base

This base served as a starting point to which a robotic arm could be attached. To control the base, a teleoperation (teleop) script was written using keyboard input that allowed the base to move forward and backward, strafe left and right, and rotate left and right.

4.4.3.2 Arm Simulations

Each robotic arm was simulated using a URDF and other native control elements and dependencies, with their official GitHub repository. These repositories were cloned and fit to the mobile base such that the robot description and world frame linkages were not disrupted.

Both the Panda and the Gen3 came with native support for MoveIt!, a library dedicated to handling the motion of robotic manipulators. MoveIt! initializes arms through a robot commander, a planning scene interface, and an arm group. The robot commander works to identify the robot description, the planning scene interface assists with path planning avoiding obstacles in the world, and the arm group contains the geometric relationships between the joints and linkages as defined by their respective URDF. Each arm will have a slightly different MoveIt! initialization from how the repository formatted and organized its software. Moreso, each arm will have a different arm group. MoveIt! is also capable of

interfacing with Robot Visualization (rviz), which allows for a GUI interface to move the end effector to a desired location, plan a path, and finally execute the path.

MoveIt! capabilities were fit with ROS services. Several services were created and abstracted to work with any arm. These services were:

1. Move to a desired EE geometric pose
2. Move to a desired set of joint angles
3. Open or close the gripper a specific distance

The service client structure enables the initialization and MoveIt! specific functionality to be handled in the service node without disrupting the flow of other nodes. The client node does not need to import MoveIt! message types or format the information in a particular manner other than obeying ROS message conventions. The services were custom-written and designed to return back a boolean depending on the success of the operation.

4.4.3.3 Insert Simulations

Finally, the arm and base simulations had to be combined to simulate the whole system. The first step was to attach a box that served as a placeholder for the modular insert. This was achieved by creating a new URDF that imported the arm's URDF and linking a box to the bottom of the arm, as shown in Figure 69. An example of the URDF can be found in Appendix 10.4.1.

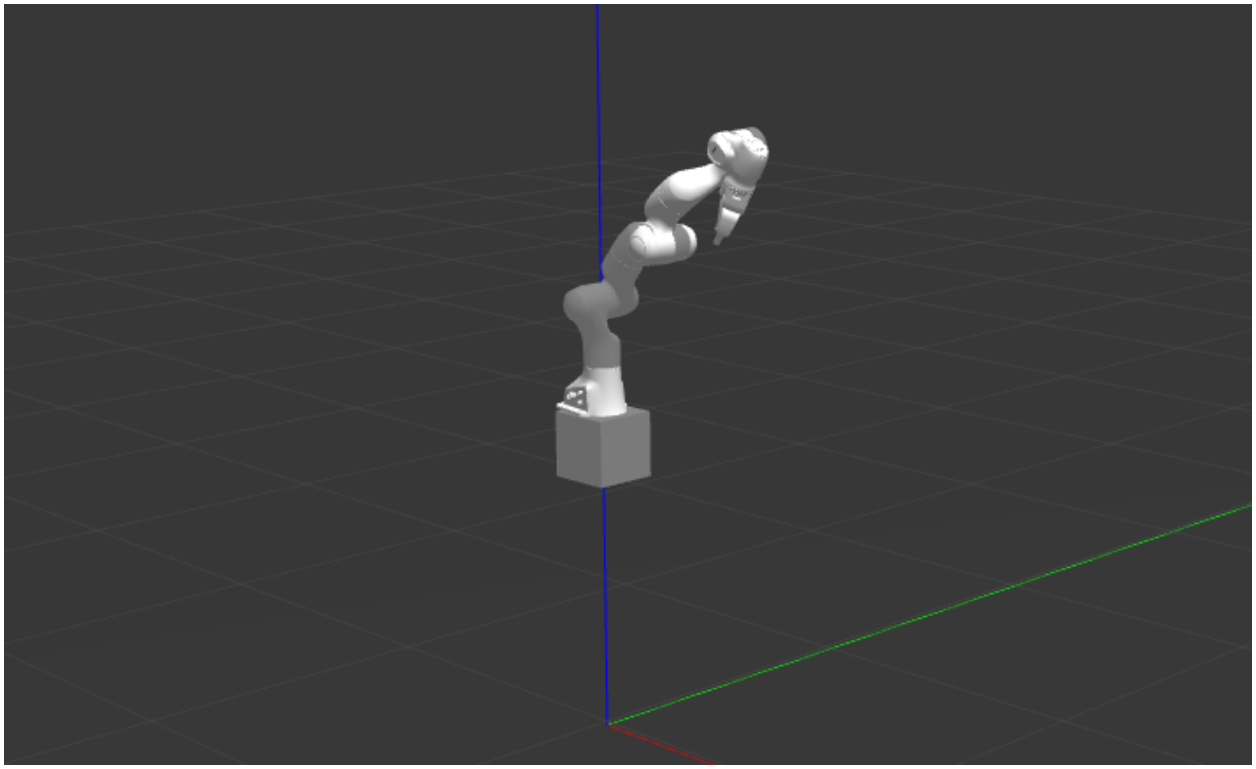


Figure 69: Simulation of the insert with the Panda arm attached

Since the base and insert were completely separate robots in Gazebo, a solution was needed to make the insert stay fixed on top of the base and follow its movements. To control the location of the insert, the world joint solution was replicated from the base simulations. This time, the insert received one rotational joint for its Z rotation and three prismatic joints to control its X, Y, and Z-axis positions, where the Z-axis capability served as a replacement for the elevator jack system. With both pieces of the robot controllable, they could be spawned into the same simulation and controlled together, as seen in Figure 70.

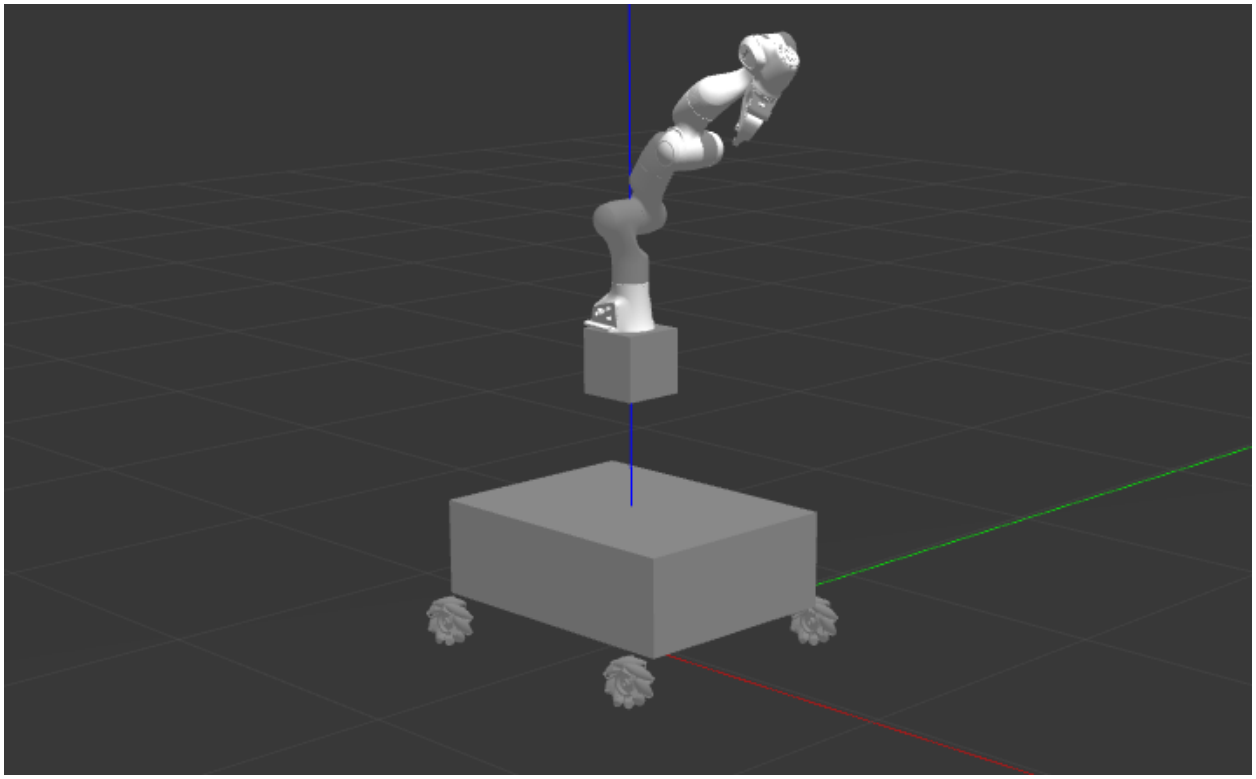


Figure 70: Simulation of the base and Panda insert together

In order to make the insert follow the base, a short script was written that took the base's x and y coordinates and fed them to the insert's x and y position controllers. Additionally, after the CAD models of the base and insert were completed, they were attached to their respective Gazebo URDFs for a complete looking simulation, shown in Figure 71.

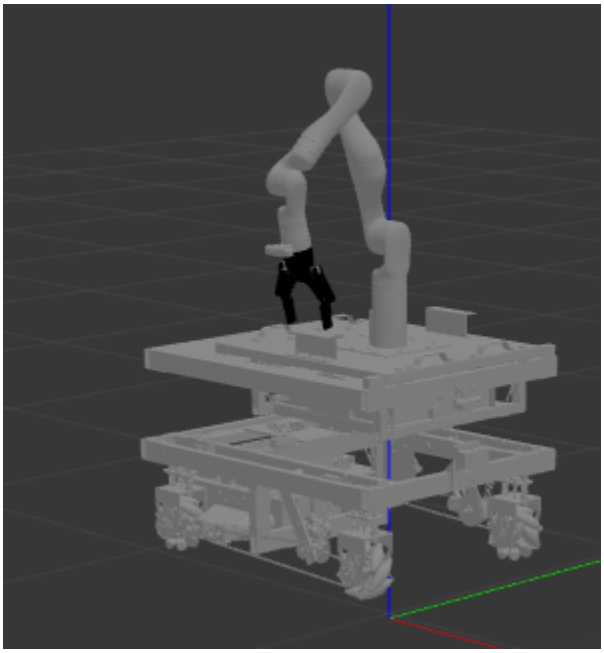


Figure 71: Completed simulation robot model

Once the simulation was able to be controlled, development of the state machine began, using the simulations to verify its functionality.

4.4.4 State Machine

The autonomous mode state machine was implemented using SMACH (short for State MACHine), a ROS-independent Python library for rapid state machine development [14]. Each state was encapsulated in its own class and had two major functions. The __init__ function initialized the state and specified which outcomes could be achieved after that state, and the execute function did the actual work of the state, returning an outcome when it was finished. These outcomes determined which state came next in the process loop. Additionally, task progress data was passed between states via input and output keys that were mapped together at runtime. An example of a state class can be found in Appendix 10.4.2.

Inside the execute function was where state machine logic was implemented. To reduce file complexity, a separate script called service_interface.py was written for delegating function calls to the arm, elevator, and base services. Due to the nature of services and their integration with python, these function calls were synchronous, so the state machine would always wait for a subtask to be completed before moving on to the next state.

4.4.5 Distributed Processing

ROS is a distributed computing environment, meaning any of its nodes can be run in different processes or across multiple machines. To save on space and power, instead of directly attaching a laptop to the robot, a Raspberry Pi 4 was chosen to serve as the local computer on the robot. The Pi would only run necessary drivers to move the robot, while the rest of the computationally intensive work, like MoveIt! calculations or the state machine loop would run on a stationary laptop.

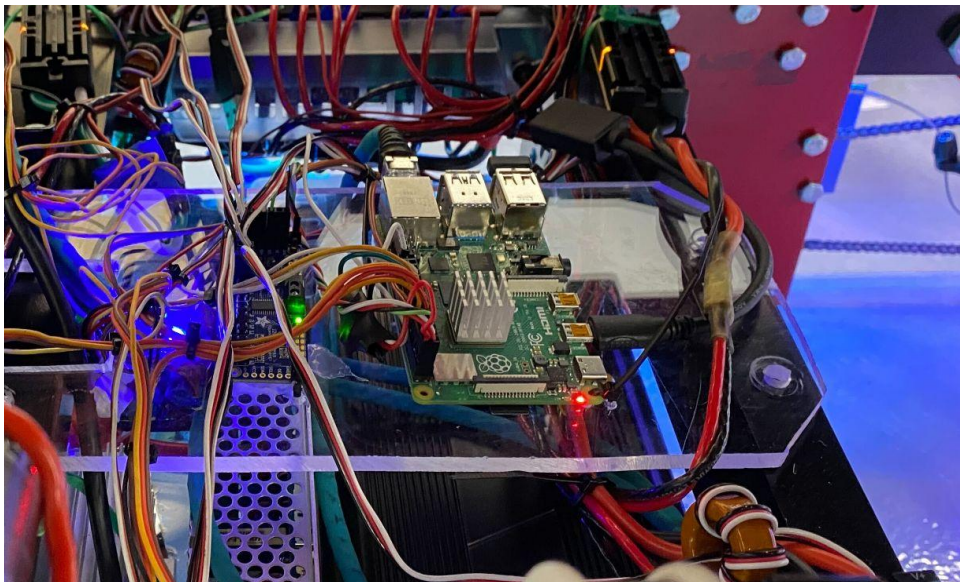


Figure 72: Raspberry Pi onboard the robot

To set up the Pi, first the Raspbian 64bit OS was installed for compatibility with ROS Noetic and robotic arm drivers. Later this OS was patched to become a fully preemptible real-time OS to be able to communicate with the Franka Emika Panda properly. The Pi was then connected to the WPI wifi and additional ROS environment variables were established to be able to exchange data between computers. Not only did the ROS_HOSTNAME variable have to match the Pi's hostname (which was simply "raspberrypi"), but on the laptop, the Pi's IP address also had to be mapped to a host called "raspberrypi". It was essential that these names matched up for ROS to communicate correctly between both computers. Additionally, the ROS_MASTER_URI variable had to be set to the laptop's IP:11311.

Once the Pi could subscribe and publish to the main ROS program running on the laptop, the driver nodes that had to be run on the Pi could be launched remotely via a separate launch file. This was usually done via SSH from the laptop, as it was a quick and easy method for running programs remotely.

4.4.6 Robot Actuation

The robot contained two separate motor-driven subsystems: the drive train and elevator jacks. To drive these motors, a motor controller script was written to translate input motor efforts into a PWM signal sent to the motor controllers.

4.4.6.1 Mecanum Logic

The drive train logic was simple. The node received joystick input from the XBOX controller, calculated the efforts of the four mecanum wheel motors, and sent those to the motor controller script running on the Pi.

4.4.6.2 Elevator Logic

The elevator logic was significantly more complex. Due to manufacturing differences, each jack required a different torque to spin the through-rod. That meant that even if all three jack motors were spinning with 100% effort, the jacks would raise at different rates. Thus it was necessary to develop a proportional controller or p-controller for the jacks to develop a feedback loop that would keep the jacks level with each other as the elevator went up and down. This was achieved using the motor encoders and limit switches to keep track of each jack's height. First, on initialization, the system slowly lowered the jacks until the limit switches were pressed, which served as both a zeroing location for the encoders and also as a software stop to prevent the jacks from depressing too low. Then the controller switched to proportional mode. On user input from the XBOX controller's D-pad, the *set_point* variable would increase. This served as the goal encoder value that each jack would try to achieve. Then, the effort of each jack was calculated using the distance it needed to travel to achieve that *set_point* value, also taking into account how far away from the other two jacks it was. For example, if all three jacks were lifting up, the lowest jack would receive the highest effort value of the three in order to try to catch up to the two ahead of it. This feedback loop kept the tabletop level when going up and down.

5 Analysis & Validation

5.1 Mechanical Analysis

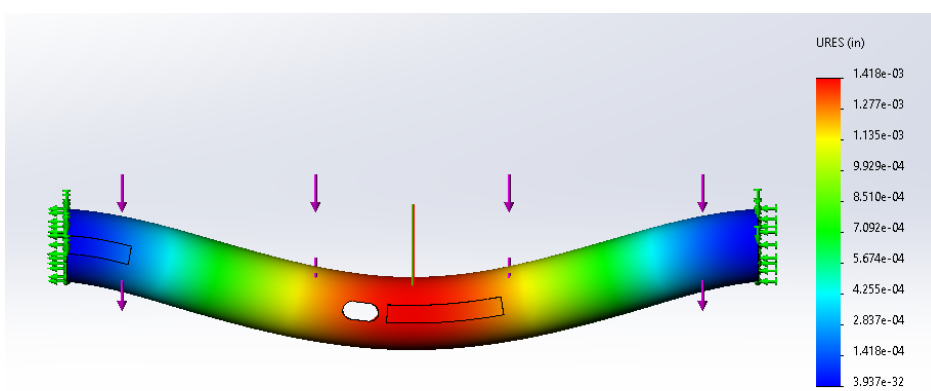
The mechanical analysis entails Finite Element Analysis (FEA) simulations, hand calculations to verify the FEA, and a proposed testing plan. In brief, this section will cover the expected forces on the most critical systems and their measured forces in practice.

The components chosen to be evaluated were generated from DFMEA and intuitive analysis. The full DFMEA can be found in Appendix 10.3. The components chosen to be evaluated were:

1. Chassis
2. Elevator Alignment System
3. Suspension
4. Arm Mounting Plate

5.1.1 FEA Simulations

FEA simulations were performed on the axle of the mecanum wheel to validate that it could comfortably perform under the total weight of the robotic mobile base. As seen below there is an exaggerated version of the stresses and deflections of the axle. These simulations were performed in SolidWorks before the specific weight of the robotic mobile base was known. The weight used in these simulations is approximately 1.5 the actual weight of the robot. Using 1.5 as a safety factor for the axle the robotic base would be able to withstand unexpected loads applied on top of the base without plastically deforming.



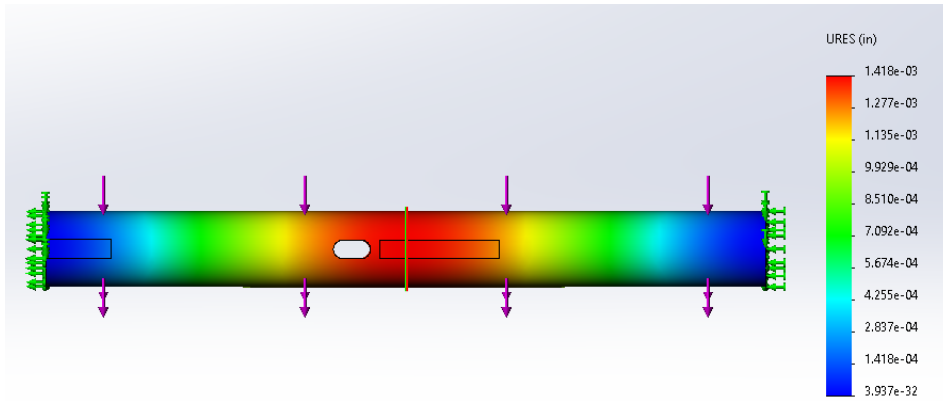
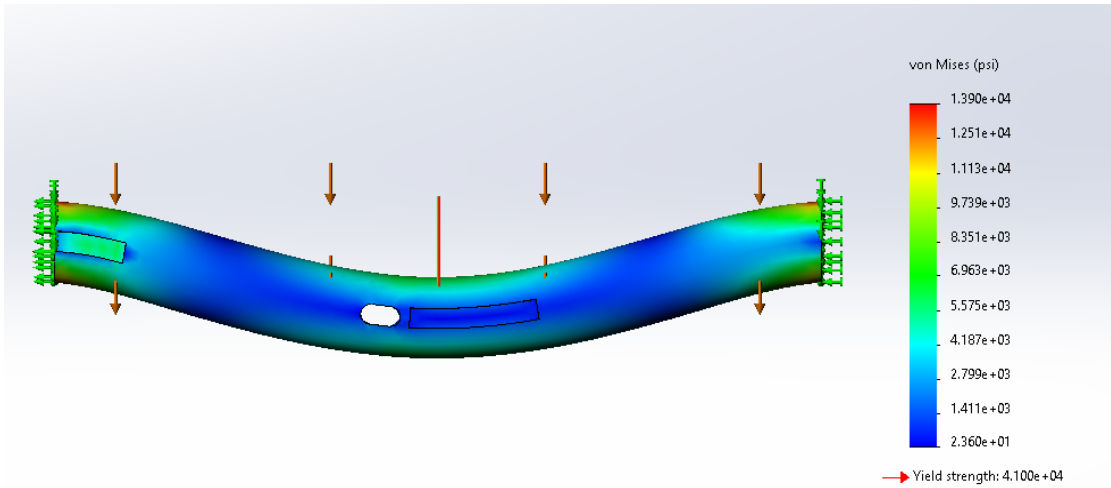


Figure 73: Axle Deflection Exaggerated (top) and Actual (bottom)

The axle deflection was determined to be most critical in the center of the axle. This is an expected outcome and reinforces the belief the FEA is accurate. Even at the maximum deflection, the steel axle will deform 1.418e-3 in. This is not enough for concern.

The von Mises stress was also simulated. Von Mises stresses evaluate areas that are the most likely to yield. The areas of interest were the ends of the axle connecting to the sprocket and bearings.



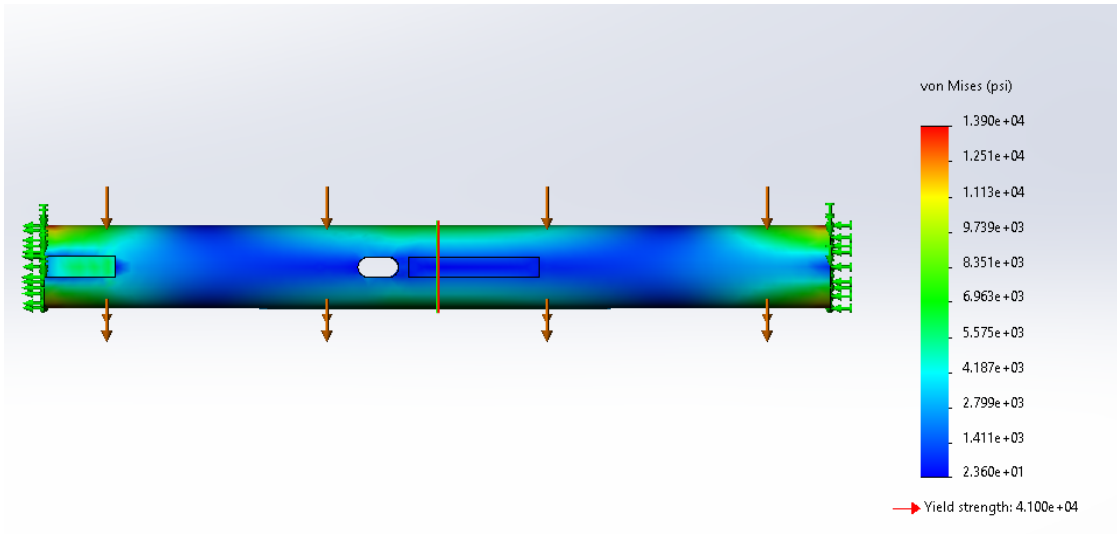


Figure 74: Von Mises stress of the axle exaggerated (top) and actual (bottom)

The simulation gives confidence that even with a conservative payload of 1.5x the maximum weight of the robot, the axles will not deform. This is critical to ensuring the axles will behave in an expected manner during normal operation.

5.1.2 Hand Calculations

5.1.2.1 Elevator Alignment System Shoulder Bolt Calculation

The most crucial component in the elevator alignment system were determined through the DFMEA located in Appendix 10.3.2. It was found to be the shoulder bolt located through the inner bearing. If this connection were to fail, the tabletop would no longer have a rigid connection to the elevator and would therefore lose the main goal of the project which was having an interface with different robotic manipulators. Hand calculations were performed to verify that this failure would not occur. Assumptions about the calculation are listed below.

- Use rates given by the manufacturer for the pillow blocks to ensure safety factors
- Statically determinate
- Weight of the shoulder bolt is negligible
- Gravity is equal to 9.8 m/s^2
- Maximum mass of the modular insert is 12 kg
- Maximum mass of the heaviest arm is 18 kg
- Maximum mass of the heaviest arm controller is 7 kg

- Maximum payload of the arm is 3 kg
- Maximum weight of the modular insert and heaviest arm (with maximum payload) is 392 N
- The applied load approximated as a point force 1/3 of the maximum total weight of the modular insert and the heaviest arm
- The shoulder bolt will be analyzed as a cylindrical beam fixed at two ends
- The material that the beam consists of is alloy steel

Shoulder Bolt		
1/3 of the total applied force	130.6666667	N
Shoulder Bolt Diameter	0.012	m
Length between fixed points	0.03	m
Alloy Steel Modulus of Elasticity (E)	200000000000	N/m^2
maximum distance form neutral axis (y)	0.006	m
Area moment of inertia (I)	0.000000001017	m^4
Cross-sectional area of cylindrical beam	0.00011309724	m^2

Table 19: Dimensions of Shoulder Bolt

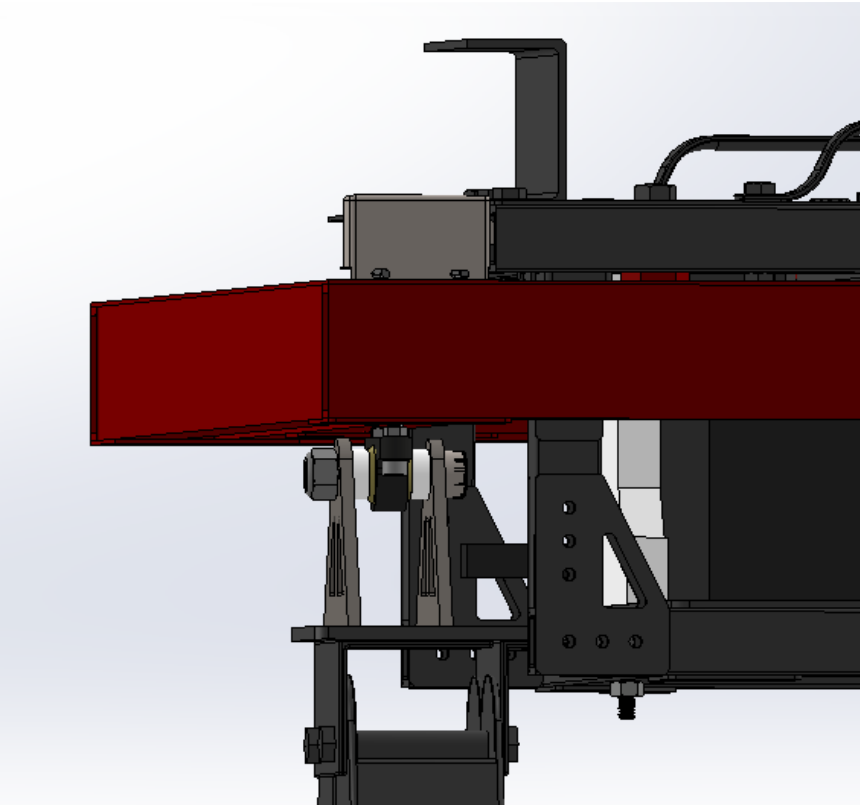


Figure 75: Elevator alignment system in full assembly

The elevator alignment system consists of a pillow block, shoulder bolt, and interfaces with the scissor jacks (bottom) and the tabletop (top). Under load, the shoulder bolt will experience a load under the inner bearing of the pillow block and reactionary forces from the vertical supports. This will induce a bending moment that could impair the functionality of the pillow block and dampen the mobile bases' ability to elevate. A detailed free body diagram is described below.

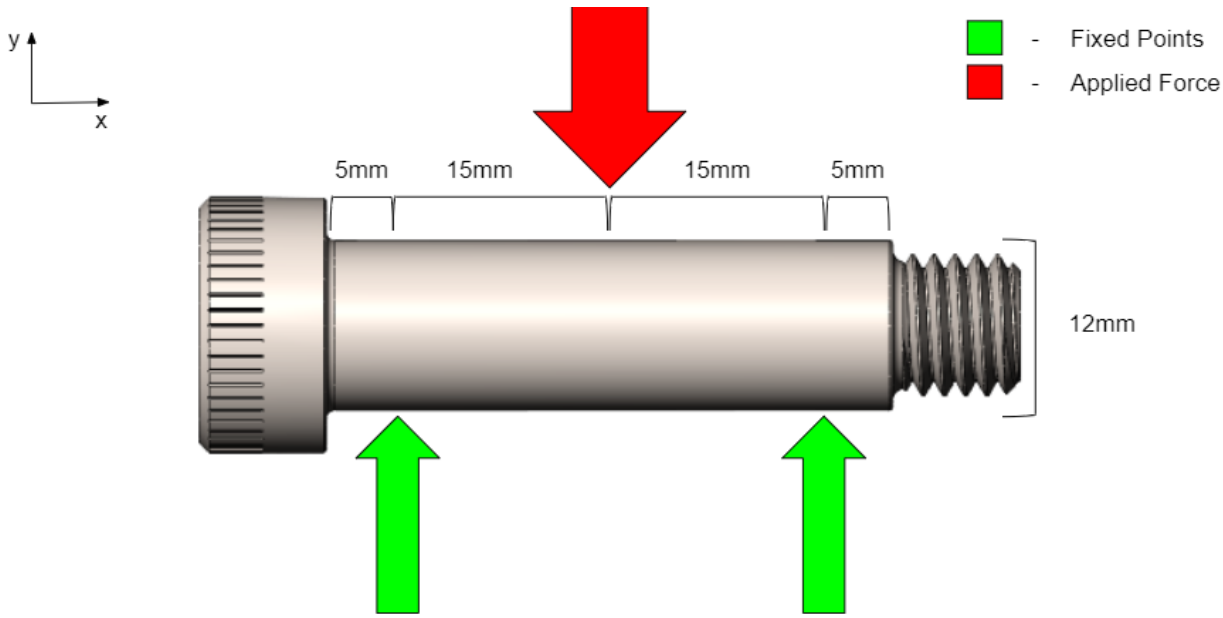


Figure 76: Shoulder bolt free body diagram

Here, the reactionary forces (green) counteract the load induced from the tabletop. The equations of stress and bending are widely known for a simple bending arm. The secondary moment of inertia and the maximum shear stress were calculated below.

$$I_{Shoulder\ Bolt} = \left(\frac{\pi d^4}{64}\right) = 0.102cm^4$$

$$V_{max} = \frac{F}{2} = 65.3N$$

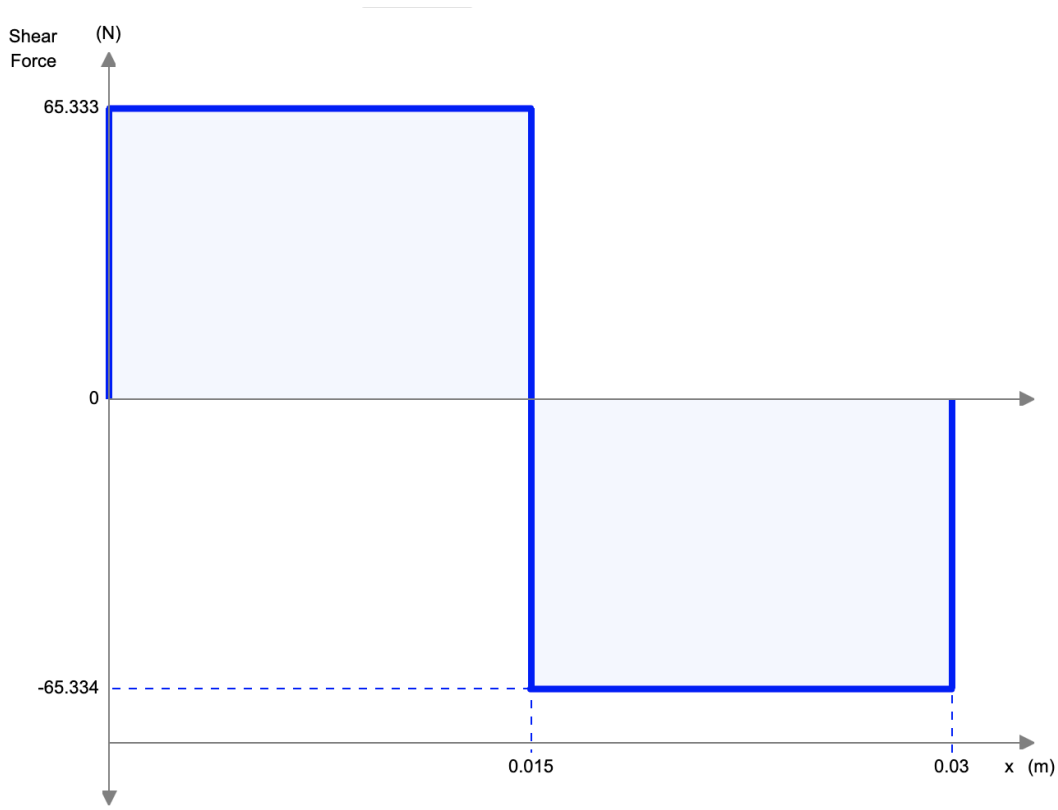


Figure 77: Shear force diagram of shoulder bolt

The shear force across the bolt is described in Figure 74. This relationship explains the forces experienced along the bolt. Indeed, there is not a shear stress in the center of the bolt.

$$M_{max} = \frac{FL}{4} = 0.98Nm$$

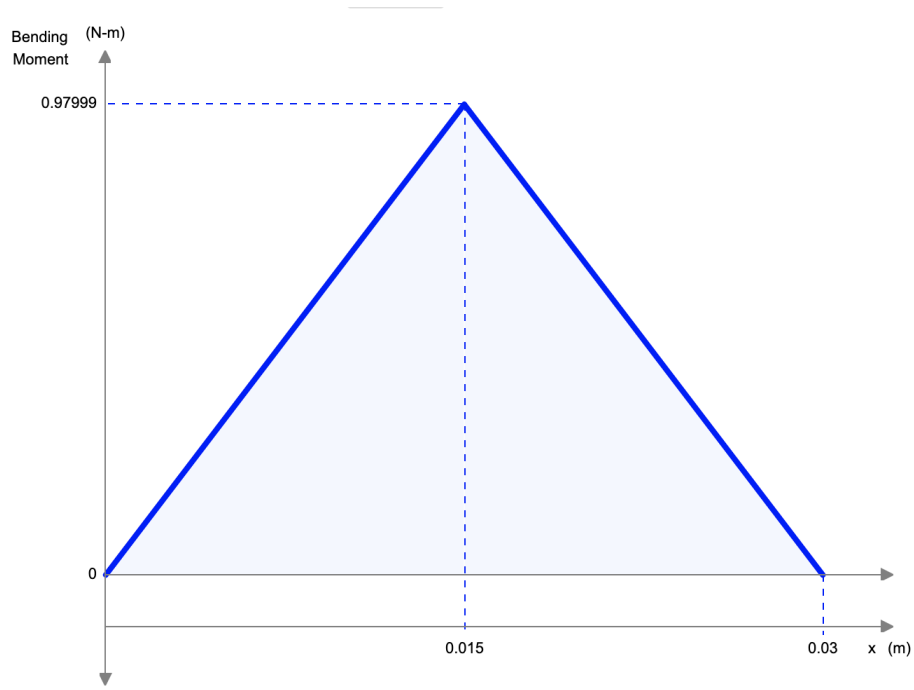


Figure 78: Bending moment diagram of shoulder bolt

This figure demonstrates the bending moment along the shoulder bolt. The bending moment is the integral of the shear stress relationship. This relationship is expected as there will be the maximum amount of bending at the center of the bolt where the load is applied. Below are key characteristics describing the maximum normal, shear, and deflection values.

$$\sigma_{max} = \frac{FLy_{max}}{4I_{Shoulder\ Bolt}} = 5776739.851\frac{N}{m^2}$$

$$\delta_{max} = \frac{FL^3}{48EI_{Shoulder\ Bolt}} = 0.000361mm$$

$$\tau_{max} = \frac{4V_{max}}{3A_{cross\ section}} = 770231.980\frac{N}{m^2}$$

$$\sigma_1 = \frac{\sigma_x+\sigma_y}{2} + \sqrt{\frac{(\sigma_x+\sigma_y)^2}{2} + \tau_{max}^2} = 7045125.746\frac{N}{m^2}$$

$$\sigma_{Von\ Mises} = \sqrt{\frac{(\sigma_1-\sigma_2)^2+\sigma_1^2+\sigma_2^2}{2}} = 7045125.746\frac{N}{m^2}$$

With the extremely minimal forces, the design goal to have at least a safety factor two was met with these stresses. A safety factor of two would enable double the weight applied onto it without plastic deformation. The maximum deflection of 0.0003 mm also gives a strong validation that the shoulder bolts will not deflect under load and will perform as expected and achieved the safety factor well over two.

5.1.2.2 Suspension Spring Calculation

The suspension spring calculation was based on several key factors and assumptions as listed below.

1. The entire system will weigh 150 lbs
2. There will be two springs per mecanum wheel (for a total of 8 springs)
3. The center of gravity of the robot will be in the geometric center
4. The springs will be designed to compress at most 1 inch
5. The springs will be designed to idle at half compression (0.5 inches)

Using the relationship F = -kx, where F is the force, k is the spring constant, and x is the displacement of the spring from the load, the spring constant can be solved as

$$\frac{150lbs}{8\,springs} = k * (0.5in)$$

Where k will be equal to 37.5lbs/inch to achieve this relationship. The spring selected has a spring rate of 30lbs/in which will result in a deeper compression, and also a maximum compression of 1.1inches before plastic deformation. Additionally, the inner diameter of the spring is 0.56 inches, wrapping nicely around the 1/2 inch hex bolts used as guides.

The entire robot does not weigh 150 lbs, however, as this was just an earlier assumption. The robot weighed a total of 287 lbs, causing the springs to be fully compressed in their idle state. This design comes with some benefits as now the entire 1-inch compression range will be used in extension as well as having a smoother operation. Indeed, if the springs were to be idle at halfway compression, this would create a bobbing system without a damper. The robot also does not have a center of gravity in the geometric center so the springs would be pitched if not fully compressed, which would cause the chassis to be pitched and thereby the elevation system, as well.

5.1.3 Proposed Testing Plans

The proposed testing plans consist of a goal, necessary materials, a setup, and a procedure. Three testing plans were developed to measure the slip/tip objective, measure a surface with a strain gauge, and evaluate the suspensions racking from the tensioner.

5.1.3.1 Slip/Tip Testing Procedure

GOAL

The goal of this is to outline the basic setup and testing procedure for the slip and tip tests that will be performed for the Robotic Mobile Base MQP. This procedure will test whether or not the Robotic Mobile Base will slip, tip, or neither while traveling up an inclined surface.

TOOL UTILIZATION

For this test, a standard handicap ramp on the WPI campus will be utilized.

MATERIALS

Item	Quantity	Purpose
Standard Handicap Ramp	1	Test Platform
Camera	1	Record findings used for further analysis later

Table 20: Slip/tip testing materials

SETUP

1. Bring mobile base to a location with a handicap ramp

STEPS (varies for each test performed)

1. Turn mobile base on
2. Utilize UI controls to traverse the mobile base up the ramp
3. Control arm to be extended off the side at its maximum range in the base's most vulnerable position
4. Record results with camera

5.1.3.2 Strain Gauge Testing Procedure

GOAL

The goal of this testing procedure is to outline the basic setup and testing procedures for the strain gauge tests that will be performed for the Robotic Mobile Base MQP. This procedure will test the actual stresses/deflection that occurs on the bracket of the Robotic Mobile Base during the crash test.

TOOL UTILIZATION

For these tests, a Wheatstone Bridge will be utilized because of the simplistic setup, the accuracy/consistency it can be measured to, and the proof of success from numerous experiments.

WHEATSTONE BRIDGE

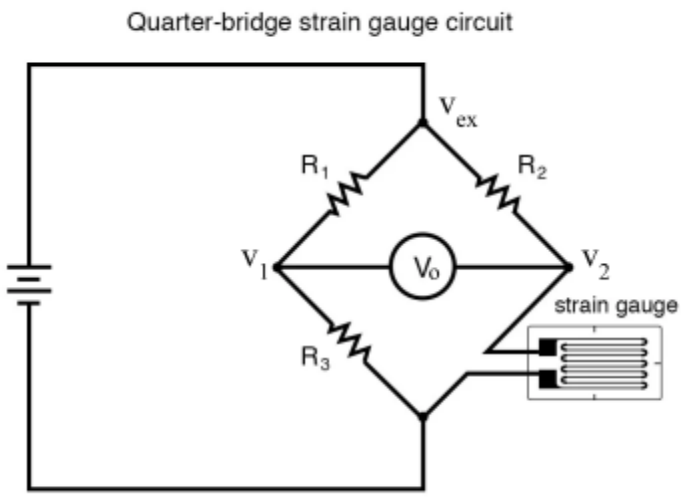


Figure 79: Wheatstone bridge diagram[22]

The Wheatstone bridge is an electrical circuit utilized to compare the resistances of resistors. Indeed, if R_1, R_2, R_3, and R_4 (strain gauge) are all equal resistance then the voltage reading across the terminals will be 0V. After the strain gauge is deflected the cross-sectional of the internal wire will deform resulting in a change in electrical resistance. The change in resistance will cause a measurable voltage difference across the terminals which can be mathematically manipulated into a relationship with the amount of stress or deflection the specimen experiences.

Symbol	Explanation	Units
$V_{O,ex}$	Voltage	V
$R_{1,2,3,4,g}$	Resistance	Ω

[22] *Strain gauges: Electrical Instrumentation Signals: Electronics textbook*. All About Circuits. (n.d.). Retrieved April 28, 2022, from https://www.allaboutcircuits.com/textbook/direct-current/chpt-9/strain-gauges/

ε	Strain	None
GF	Gauge Factor	None
E	Modulus of Elasticity	N*m^2
σ	Stress	N*m^2
δ	Deflection	m
L	Length	m

Table 21: Nomenclature

MATHEMATICAL RELATIONSHIPS

Resistance difference -

$$V_o = V_{ex}[\frac{R_1 \cdot R_4 - R_2 \cdot R_3}{(R_1+R_2)\cdot(R_3+R_4)}]$$

$$\Delta V_o = V_{ex}[\frac{\Delta R_1 + \Delta R_4 - \Delta R_2 - \Delta R_3}{4R_g}]$$

$$\frac{\Delta V_o}{V_{ex}} = \frac{\Delta R}{R}$$

Stress -

$$\varepsilon = \frac{\Delta R}{R \cdot GF}$$

$$\sigma = E \cdot \varepsilon$$

Where GF & E are known

Deflection -

$$\delta = \frac{\sigma \cdot L}{E}$$

Where L is known

MATERIALS

Item	**Quantity**	**Purpose**

Raspberry Pi 4 B	1	Microcontroller
ADS1115 ADC	1	Measure analog voltage
Uctronics board	1	To visualize pinouts
330-ohm strain gauge	1	Change resistance with deflection
10-ohm potentiometer	3	Fine-tune resistances
330 ohm 5% resistor	3	Resistors
Male to Male jumper wires	As needed	Electrical connection

Table 22: Strain testing materials

TESTING SETUP (varies for each test performed)

1. Place a hard surface against the wall (plywood or other sturdy material) for the robot to run into
2. Adhere both ends of the strain gauge to the inside of the box tubing front face
3. Start recording data snd then proceed with testing steps

TESTING STEPS

1. Turn mobile base on
2. Through the UI, control the robot to run into a wall at full speed (contacting the corner of the chassis)
3. Have this run into the wall for a total of 3 seconds
4. Reverse the robot
5. Read and plot all the values from the strain gauge
6. Compare to simulation and hand calculated results

5.1.3.3 Suspension Testing Procedure

GOAL

The goal of this is to outline the basic setup and testing procedures for the suspension tests that will be performed for the Robotic Mobile Base MQP. This procedure will test whether or not the suspension acts as intended and extends/compresses the expected length without the occurrence of racking.

TOOL UTILIZATION

For these tests, vernier calipers will be used to measure the distance the suspension springs extend/compress.

MATHEMATICAL RELATIONSHIPS

Force Acting on the Spring -

$$F = -kx$$

Summation of Forces/Moments Acting on Robot

$$\Sigma F_y = 0$$

$$\Sigma M_k = 0$$

MATERIALS

Item	Quantity	Purpose
2x4 Lumber	As Needed	Necessary to construct apparatus
Vernier Calipers	1	Measure the distances the springs extend/compress

Table 23: Suspension testing materials

SETUP (varies for each test performed)

Suspension designed to have a maximum flexion of 1 inch

1. Assemble the lumber apparatus by placing two 2x4s in line with each other, spaced the width of the mecanum wheels. Ensure one 2x4 is shorter than the other such that the mobile base will drive one side off of it.
2. Control the robot to drive one wheel over the gap
3. Record the extension of the springs

4. Repeat this task for each mecanum wheel

STEPS

1. Load robot onto the apparatus. Ensure all wheels are centered on the 2”x4”s.
2. Measure the compression of the springs for each wheel. It may be assumed that the pairs of springs per wheel are equally compressed.
3. Drive the robot forward until one front wheel has fully come off the 2”x4”.
4. Measure the compression of each wheel's springs.
5. Drive the robot forward until the wheel has climbed up the other 2”x4”. If the robot is not capable of making this climb, lift it until the wheel is now on the next 2”x4”.
6. Measure the compression of each wheel's springs.

The code necessary to read the strain gauge and record the data is in Appendix 10.4.3.

5.2 Electrical and Software Analysis

An electrical and software analysis was not performed. To read an evaluation of the testing and performance, see section 6.

6 Discussion

The discussion section is used to evaluate the performance of the mobile base. Due to the complexity and depth of the project, it will be divided into the mechanical, electrical, and software sections.

6.1 Mechanical Discussion

All mechanical elements behaved as expected and did not experience issues during normal operation. However, that does not mean there isn't room for improvement or small caveats worth detailing. The following sections will evaluate each individual subsystem's performance.

6.1.1 Chassis

The chassis was created with aluminum box tubing to create a rough outline first and mounting elements later. This procedure allowed the project to progress without being fully functional and proved to be an extremely successful strategy. Aluminum is soft and easy to drill through, so mounting elements at different times did not prove to be a challenge. The chassis also was easily able to handle the weight of the entire robot, despite weighing a total of 287 lbs when fully constructed. Lightning holes could have been designed to reduce weight while maintaining structural integrity, but this is an added challenge with no clear benefit to the objectives created. Lightning holes would’ve also prevented areas from being mounted, which would have proven to be a risky gamble without most of the design finished.

The rectilinear chassis design of 28 in x 35 in did not prove to be an issue with moving through doors. Indeed, the screw on the scissor jack would often protrude more outwards than the chassis itself. Nonetheless, the greatest concern for traveling through doors was damaging the door itself. The use of polycarbonate paneling helps alleviate this problem but does add additional width, which removes some clearance. It is unclear if the paneling will create too tight a tolerance in the future and requires additional testing.

6.1.2 Suspension

The spring-based suspension was designed to be fully compressed when working under normal compression. There are several advantages to having the suspension bottomed out, such as

- A maximum amount of spring leftover to correct contact imbalances

- Even positioning of the chassis
- No need for a damper

Since the suspension will only be working to create contact with the floor in a few instances, it is important to have the springs not be fully bottomed out during normal operation. This simplifies the design and also reduces cost.

The suspension was also designed to be able to be tuned for each specific wheel. As the chassis is prone to having misalignment with the bracket holes and riveting, precise tuning is critical. This was simply achieved by directly altering the position of components through tightening or loosening nuts. By doing such, the pieces moved slightly, enabling smooth guidance of the suspension and a tailored bottoming out range. In practice, this worked excellently and without issue, though it was time-consuming to set up.

6.1.3 Drivetrain

In practice the drivetrain faced several small-scale issues that did not impact functionality but could have performed better. Despite contact with the floor, the mecanum wheels are unable to strafe. This directly defeats the purpose of utilizing this drive system as its holonomic capabilities were very attractive. When attempting to strafe, the robot would maintain the same orientation, but translate in circles, indicating slipping was occurring. Since the mecanum wheels were manufactured in 2008, it is very likely the polyurethane rubber on the rollers dry rotted and lost many of its original frictional properties. Moreso, despite an appearance of less friction, the suspension was pulled towards the drive motors, affecting the alignment. Strafing was not used after these properties were observed.

Additionally, the tensioner spring was too aggressive and worked against the suspension spring. The suspension was implemented first with a loose idea of a tensioner to be added later, so the thought of them interfering with each other was not well thought out. Nonetheless, the tensioner pulls the chain up while the suspension pulls it down. The result is the suspension is mostly non-functional, being trapped in its bottomed-out position. In practice, this had little difference in expected vs actual outcome as the robot was only driven on flat floors.

The braking system was not electrically nor programmatically implemented and thus was not tested. However, the concept was tested and proven to be a successful strategy. The window motor provided more than adequate torque to compress the brake calipers around the brake disc. Mechanically what is left to complete setting up the brake window motor is as follows:

1. Create pathways for the brake cable with 3D printed supports
2. Connect the brake cable to the window motor

3. Potentially add spacers where necessary to reduce bending moments

The chain itself was found to be more ductile than anticipated and stretched frequently. This can most likely be attributed to moving the heavy 287 lb mobile base. For the front two tensioned chains this was not an issue, but the back two chains were often not fully taught. The current solution was to manually remove chain tension as adding a tensioner to such a small chain circuit is mechanically challenging. It is also worthwhile noting that the brake disc came surprisingly close to interacting with the steel suspension housing. If the chain stretches a significant amount or the suspension is pulled towards the motor from a strafe attempt, the brake disc can make contact. This is not ideal and will cause the tensioner to work less effectively and the frictional contact makes an undesirable scratching noise.

The motor mounts also suffered some damage during high frictional events such as strafing. Since cheap parts were utilized, they were susceptible to bending induced by the CIM motors. The motor mounts were reinforced with an additional bar of stock underneath the VersaPlanetary gearboxes which did give some noticeable aid but it is still unrecommended to perform these high frictional maneuvers. For low frictional operations, such as driving straight or even turning, there were no issues observed.

6.1.4 Elevation

The elevation system functionally worked as expected. Each jack is raised at a different speed from various torques required, but this was expected. The difference in torque is attributed to different spacing between the brackets, differences between the brackets themselves, individual scissor jack torque, and alignment of the pulleys. Moreso, the use of commercial scissor jacks saved development time and generated a solution that could withstand thousands of pounds without issue. While this will never happen in even the most conservative estimates, it is reassuring to have such a strong safety factor. However, the scissor jacks are not designed to elevate under load, which they will always be under in this system. Due to the modular insert, even when combined with the maximum payload of the robotic arm, being lightweight and distributed across three scissor jacks, there should not be an issue.

The elevation alignment system performed exceptionally well and enabled lateral pivoting. However, pivoting would not work with just the elevator alignment system. Indeed, the tolerance, or wiggle room, of the scissor jacks allows for considerable shifting, which allows for the individual scissor jack to match the necessary pose to achieve a certain tilt. This element was not directly planned for but was integrated as part of the functionality. The alignment system prevents longitudinal misalignment directly and gives many degrees of additional workspace to correct the imbalance.

It is also worthwhile to note that the scissor jacks follow a nonlinear relationship between rotation and altitude. That is, the scissor jacks are more susceptible to changes in height when they are most compressed. This relationship is ideal to understand when working to program the encoders reading with an associated height. The mathematical relationship was not derived during this project due to time constraints.

The physical height of the elevator alignment system is conservative and designed to be higher to prevent the tabletop from interfering with the scissor jacks. However, this additional height works against the robotic arm and makes it more challenging for it to reach the floor.

Additionally, the pillow blocks are mounted to the tabletop through rivet nuts which are an insecure connection. The back scissor jack is under tension, actively pulling out the rivet nuts. This can create an issue with an excessive load applied and create a bending moment that will inadvertently pull off the rivet nuts. This happened once during a mishandling of the robot, and new rivet nuts had to be inserted. These rivet nuts should not come undone during normal operation as the loads are too light, but mishandling could result in damage.

6.1.5 Electronic Housing

The electronic housing was confined to a small space to minimize the overall size of the mobile base. To not be in the way of the modular insert, it was placed next to it as opposed to under. This elongated the chassis causing the mobile base to be only able to fit through a door in one orientation, decreasing mobility. The housing could’ve contained more supports for electronic mounting and provided additional scaffolding to protect the electronics against electrically conductive particulates. However, the open-ended electronics were helpful in actively debugging and modifying the mobile base.

6.1.6 Modular Insert

The modular insert performed exceptionally well. The guide pins worked to create a seamless electrical connection with the universal plug, and the robotic arm was in the same location every time. The insert also was lightweight enough for two people to insert and remove comfortably. However, despite using two different robotic arms, only one modular insert was created. The hardware connections necessary for each arm were superimposed onto each other on the modular insert, and the arms were switched with each other. This is not the intended design but was the only way to test the system. The Gen3 does not have an external controller box, so this setup worked.

The solenoids did not appear to show any signs of tear-out despite significant stress placed on them during testing. Indeed, directly pulling on the modular insert out of the tabletop resulted in very little bending or showing signs of weakness. Once the modular insert is clicked into place, it will not fall out of the mobile base without the solenoids opening. The solenoids also did not interfere with the alignment of the modular insert and functioned exceptionally well. There was some slight misalignment generated from compound error measurements that caused the solenoids to be vertically higher, but this was simply achieved by mounting with nuts underneath the solenoids.

The compressible connection to the universal plug compressed ideally and was visibly inspected to be functioning as expected. Three attempts were made to satisfy an adequate amount of compression.

6.2 Electrical Discussion

The Power System, composed of the BMS, rails, converters, inverter, and batteries, behaved as expected. The performance of individual components is evaluated below.

6.2.1 Battery Management System

The BMS was capable of protecting the batteries from overcharge and undercharge scenarios within its normal use but was incapable of handling reverse charging currents. During charging, the monitoring of cells was key for ensuring batteries were operating within the same range of voltages. Balancing individual cells protects against possible damage to the batteries. The BMS was also capable of reading the real-time current draw, which was useful for detecting shorts and if the electrical system was behaving as expected. If a short was noticed, it would also disconnect the rest of the electronics to prevent damage. The three onboard LEDs were useful for indicating the health of the BMS and electrical system for events where a laptop could not be plugged in to monitor the system.

However, the BMS was accidentally connected in reverse through the charging pad, forcing roughly 20A of current backward. This resulted in unexpected behavior of the BMS, the breaking of some internal components, and a new BMS was subsequently purchased. The blown BMS indicated that reverse current was detected but was unable to properly handle such an extreme reverse load. The internals of the blown BMS can be seen in Figure 80.

The advanced software onboard the BMS was not utilized. This included creating scripts to detect and report low battery, damaged batteries, or other faults back to the Pi.

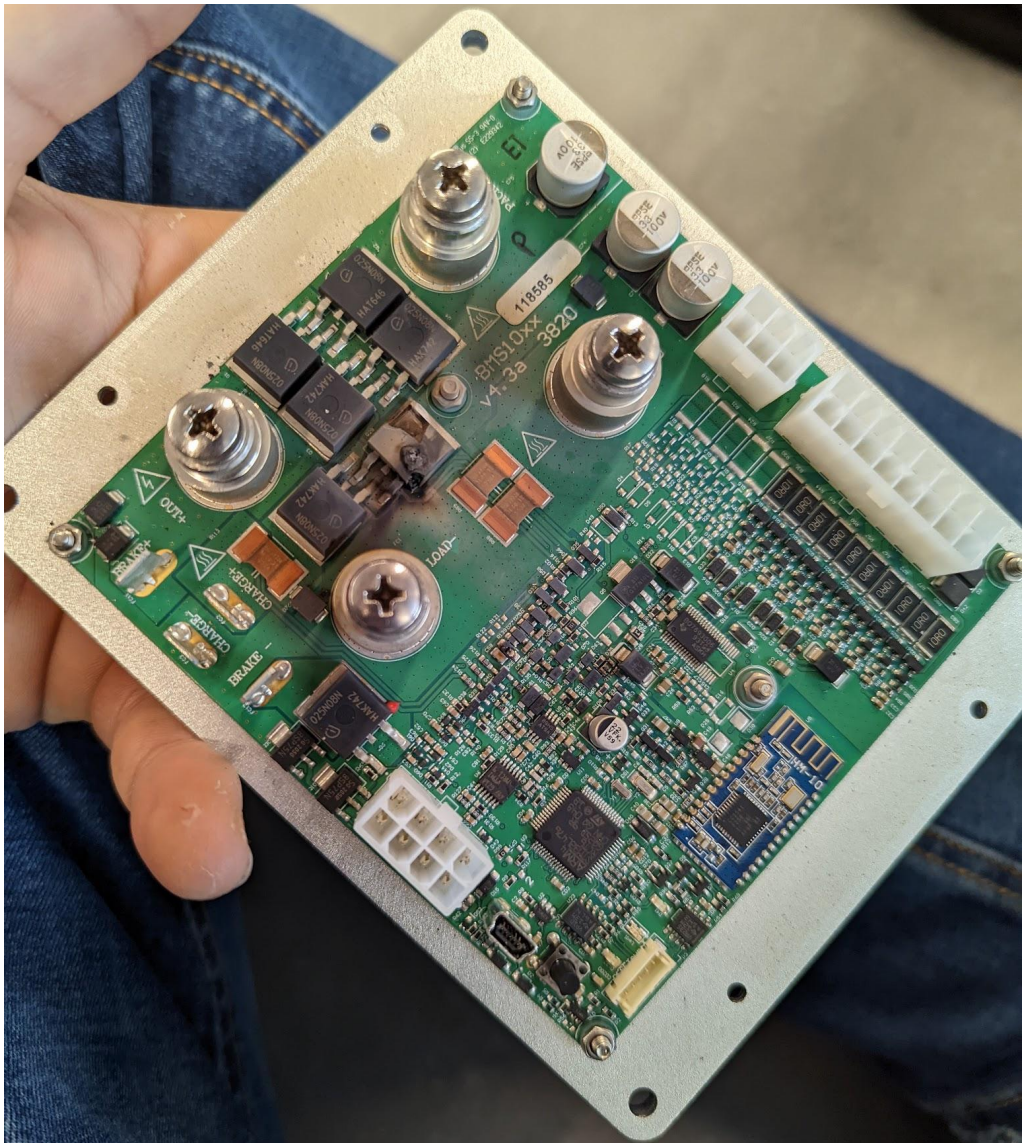


Figure 80: Blown BMS

6.2.2 Converters

The 5V and 12V converters also behaved as expected along with the associated fuses. However, the physical location of the 12V rail proved to be exceptionally challenging when modifying connections. Additionally, the 12V rail is mounted to the chassis itself, as opposed to the acrylic where the other power system components are secured.

The window motor for the braking mechanism will also have to be connected to the 12V rail through another motor controller in the future as it was not implemented within the time frame of this project.

6.2.3 Motor Controllers and Motors

The onboard LEDs on the motor controllers were essential for debugging and state indication of the motors. Indeed, it was key for testing to observe the sent direction of the motor through the motor controller before the motors were directly wired. This enabled for safe implementation of untested code and debugging of any unwanted signals without motors physically moving and damaging elements of the robot.

However, an additional motor controller will need to be put in place to interface with the window motor in the future. This window motor will be connected to the PCA9685 PWM board along with all the other motor controllers.

The CIM and 775 Pro motors behaved as expected drawing 13A per CIM and 10.5A per 775pro when 12.5A per CIM, and 10A per 775pro was expected. There were no recorded issues with power draw or torque requirements, suggesting the designed setup works adequately for sustained use. The braking window motor was not tested and it is not fully known how much current is drawn.

An issue with the PWM signal was observed with the motor controllers, however. A mixture of noise generated from the 12V converter and potentially also ground bounce induced from the solenoid controller wires generated a frequency that the motor controllers interpreted as valid PWM signals. This noise was amplified by a stray capacitance due to the chassis acting as a large capacitor. Ferrite chokes were used to reduce high-frequency noise in the wires just enough for the signals to no longer be registered as valid. They can still be measured with an oscilloscope, however.

6.2.4 Sensors

The sensors utilized were the limit switches and the quadrature mag encoders on the 775Pro motors. The limit switches behaved as expected by triggering an interrupt on the Raspberry Pi when the scissor jacks compressed to the home height elevation. Functionally, the switches behaved as expected without any issue, but a large hysteresis was observed. A possible fix to this would be smart software implementation to incorporate this hysteresis or to implement different switches but it was not a high priority as the elevation system behaved normally.

The quadrature encoders implemented on the 775 Pros were utilized to send a count to the Pi such that the elevation of each scissor jack is known along with the direction the jack is moving. This functionality behaved exactly as expected without issue. No noise was observed in the data being sent.

The Intel Realsense d435 was not wired to the Pi, but only a USB3.0 to USB-C connection of roughly 10ft is necessary. The wire can be routed to the other bundle of cables connecting the table to the base as long as there is adequate shielding in the cable.

6.2.5 Control System

The control system consists of three subsystems: the Pi, the PCA9685 PWM board, and the custom Solenoid driver board.

6.2.5.1 Raspberry Pi 4B

The Raspberry Pi worked well with the 5V rail and did not experience a lack of voltage. Each pin behaved as expected and without issue. However, the main issues can be summarized as follows:

1. Inadequate RAM for building ROS environments
2. I2C stopped functioning after a reverse current incident
3. The CPU becomes overheated when building ROS environments
4. The Pi is unprotected from dust and debris

It was observed that for building the environments of the Kinova Gen3 and the Panda arms, 2GB boards were not adequate. Indeed, a 4GB board was used in place without issue. The I2C channel also stopped properly communicating with the PCA9685 after a reverse current incident. This resulted in the Pi being replaced with a new board. The Pi was observed to be ~70℃ when building the ROS environments for the robotic arms. This is the most computationally expensive mode and thus the extreme heat should be taken as the maximum the Pi can reach. A heatsink was placed on the CPU to prevent overheating. Finally, the Pi was not protected, thus leaving it exposed to electrically conductive debris that could create unwanted shorts. This fault was deemed a low priority and measures were never taken to prevent this possible outcome.

6.2.5.2 PCA9685 PWM Board

The PCA6685 PWM Board behaved as expected with the ferrite chokes in series with the motor controllers. The ferrite chokes, however, caused the wires to the motor controllers to be incredibly taught, pulling on the pins. While they were not observed being pulled out, it is not impossible for this to occur. Moreso, much like the Pi, the PCA board is exposed and can become shorted from conductive debris.

There are 9 other PWM slots for expansion with this board. This leaves more than enough room for modification and expansion of the current system. The braking system will only use an additional slot, while a more complex drive system, such as a swerve drive, will use another four slots.

6.2.5.3 Custom Solenoid Board

The solenoid board behaved as expected with a fairly accurate timing circuit resulting in a 10-second retraction period for the solenoids. The analog timing circuit was controlled by the manual override button located on the table's surface. The wire which connects the Pi to the board was left unplugged as the circuit places a large capacitive load on the Pi's GPIO pin, which was undesirable. This can be added back for autonomous control with the addition of a lower capacitance driving circuit. It is also recommended to not hold the solenoids open in the active state longer than 10 seconds as they may begin to heat up.

6.2.6 Modular Insert Wiring

The electronics comprising the modular insert are the universal plug and the jumper pins chip. The universal plug supplied 120V AC power, the ethernet connection, and the connection for the configuration pins. There were not any problems observed; the robotic arms were fully powered and able to communicate with the Pi. The wire itself has the potential to interfere with the back scissor jack, however, this was not recorded during any experimentation. It is recommended that a wire wrap or cable chain be used to guide the wires from the table. The integrated alignment pins on the plug itself were sheared off from fatigue. There were no changes in functionality, as each time a secure connection was made using the insert's mechanical alignment pins, it performed as expected.

The configurable selector pins were not fully integrated with the Pi due to time constraints but the perf board setup with jumper pins was created and fed through the universal plug.

6.3 Software Discussion

6.3.1 Gazebo Implementation

Simulating mecanum wheels in Gazebo is an incredibly challenging task as each wheel contains individual rollers. While libraries were found to tackle this problem, they were not well documented and untrustworthy in their accuracy. The solution was to instead force Gazebo to translate the entire mobile base URDF across a prismatic joint defined in the world axis. Indeed, the mobile base is forced to

translate across the x and y axes, as well as to rotate about its respective Z-axis through this plugin. This behavior approximates holonomic drive from a mecanum wheel setup but could be improved significantly. For instances where the mobile base needs to strafe across a 45-degree angle (with respect to the X-axis), a velocity message would have to be sent to both the x and y prismatic joints as opposed to just a single velocity command sent to the X-axis on the mobile base.

The current simulation was more concerned with creating a collision obstacle of the mobile base so the MoveIt! planning scene would not have the robot arm execute a motion into the base. Additionally, translating the arm across the world with the mobile base was an exhaustive task that took extensive time to successfully implement. The modular insert follows the mobile base by updating its location in the world by constantly evaluating its transform distance to the base and moving to be above it; in short, following the base. This following behavior is visible through a slight lag in the modular insert. The tf2 transformation library does always work as intended and specific exceptions were written to prevent failure, but they were seldom used in practice.

Simulating the full mobile base and modular insert STL renders most computers inoperable due to the extremely heavy computer specifications necessary to compute the physics of 700MB worth of data. It is recommended to use the boxes for the modular insert and mobile base for adequate performance.

6.3.2 ROS Services

Utilizing services as the primary driver for actuation in the state machine and with MoveIt! will functionally work well but could be improved through the implementation of actions. Actions behave similarly to services but are capable of an additional third return message of real-time updates. This is particularly helpful in applications where a service implementation takes a long duration, and progressive feedback is helpful for either debugging or implementing a dynamic response. Here, the services do take some time to complete, and could benefit from this additional real-time feedback message. While it is not necessary with the current implementation, there is the potential to implement actions without a real-time return message in its place. This enables the possibility of real-time feedback for a dynamic response that can promote more advanced behavior in the state machine.

6.3.3 State Machine

The implemented state machine performed as designed. It was able to correctly traverse the state machine graph given the progress of the current task. At decision boundaries, the machine properly used

services to determine the position of the arm and elevators and whether it was safe to move the base. However, a service for raising the elevator to a specified height and moving the base to a specified location was not implemented due to time constraints.

7 Recommendations

7.1 Mechanical

The mechanical system performed well and as expected but could benefit from more specific implementations. They can be outlined as follows:

- A more robust drive system
- Specialized elevation jacks
- Smaller chassis
- Electrical noise reduction design

The drive system was the most lacking element of the mechanical design and prevented the mecanum wheels from being fully utilized as a holonomic drive. A swerve drive is incredibly complex to program, design, and wire, but the benefits are enormous to creating a robust drive. This design would be incredibly challenging to implement to fit within the chassis's tight design, but it would be possible if a project were dedicated to only implementing this drive.

The scissor jack elevation system worked well but is more of a proof of design system than a fully robust solution. The welds are questionable and can break with sufficient external load. It is recommended to create a custom elevation system that not only rigidly attaches the motor driver but also prevents the main screw from extending past the chassis of the mobile base when the elevators are at their highest. This is not a major concern as the base should only move with the elevation fully down, but the system could benefit from a more specialized elevation mechanism.

A smaller chassis would necessitate a complete redesign of every system in order to increase mobility. This is a risky trade-off and should only be completed with a very thoroughly designed solution. If a swerve drive is implemented, then the chassis should maintain its dimensions for as long as possible to prevent a redesign.

Finally, the chassis was not designed to reduce electrical noise from the necessary components. There are instances where additional pieces of scaffolding could've been created to isolate noisy electronics. However, the ferrite chokes reduce the noise issue so this is a minor recommendation.

7.2 Electrical

Even though the electrical system's performance exceeded expectations, a few changes could be made by future projects to make the system more robust. For starters, the motor controllers could be

updated to motor controllers that can run off of the battery system without the need for an intermediate 12V step-down converter. In order to do this, motor controllers that can run off of up to 28.8V are necessary since the maximum cell voltage of LiFePO4 batteries is 3.6 per cell, making 28.8V the voltage of all 8 in series. Then, to get a steady voltage output at the motors, the duty cycle could be varied as the batteries discharge. For example, the duty cycle to run the motors off of an average of 12V when the batteries are fully charged to 28.8V would be 42%, and when they have discharged down to their nominal 25.2V would be 47%. This would save space in the robot and allow the robot to move faster if necessary with duty cycles above 50%. This would also allow for the motors to draw more current than the maximum of 85A that the 12V step-down converter can source.

In addition, there were some issues with the charging system. For starters, the BMS should provide reverse charging protection and shut down the charging system if the charger is connected in reverse polarity. For some reason, the reverse charging system failed one time and reverse biased the batteries. This degraded the health of the batteries to the point where during the initialization of the robot, the batteries cannot source enough current to start the robot and it will throw a fault. In order to start the robot, one must connect the charger in the correct polarity, turn on the robot, and then remove the charger after about one minute once the robot finishes initializing. Also, it is believed that there is an internal short in the batteries, which is decreasing battery life and likely causing the initialization problem. To fix these problems, new batteries should be purchased. There are LiFePO4 batteries that are the exact same size as the ones currently on the robot but have a capacity of 400Ah instead of 100Ah. It is recommended that these higher capacity batteries be purchased so that the robot's operating life under typical conditions would increase from about 3.5 hours to around 14 hours. Additionally, the Robopads which are used to connect the charger to the robot should be mounted in such a way that the pads will never be able to make electrical contact unless they are in the right orientation.

Finally, cleaning up the wiring of the robot should be implemented in the near future. Cleaning up the wiring will make it easier to identify and rectify issues with the electrical system. It will also prevent the wiring from being caught in pinch points on the robot. This could be done by adding wire snakes throughout the robot as well as routing wires through the aluminum box tubing. In addition, the wires to the solenoids should be fed through the tabletop and not wrapped around the tubing through the hole for the modular insert where it could be pinched by the modular insert while being deposited.

7.3 Software

The final software had all the functionality needed for the robot to perform basic tasks. However, though each piece of the software worked on its own, it was not all integrated together. In the future, the first step should be to integrate the arm, elevator, and base services into the state machine for fully autonomous task completion. The LEDs should be programmed to reflect in which state the state machine is currently.

The simulations, while functional, ended up being very memory intensive and not accurate. The URDFs should not use CAD STLs as models in the simulations, as they increase computational complexity and are purely visual. Instead, simple boxes can be used. For driving accuracy, the prismatic joints for movement should be replaced with a gazebo plugin specifically designed for a holonomic drive system. To further increase precision, the Realsense camera should be incorporated to build a SLAM model. This would give the base a better sense of its location in the room. The elevators should also be simulated accurately; a function should be created that maps elevation height to encoder values and implemented to control the insert's Z position. Finally, the real base and simulation should be launched together such that they move simultaneously; the simulation should be as accurate to the real base as possible.

8 Conclusion

Despite being a first-year iteration of an MQP project, the robot was able to perform simple tasks like moving the base, picking up the arm from the home base, elevating and lowering the arm, and moving objects with the arm. It was able to combine all of these tasks into one, allowing for seamless teleoperation of the base and arm system. Throughout the project, many different elements of mobility, modularity, and reliability were incorporated into the design to make this possible.

For starters, the robot achieved its goal of modularity. It did this primarily with the modular insert. This insert allowed for a variety of robot arms to be connected mechanically and electrically to a piece of hardware allowing it to seamlessly integrate with the rest of the system. With the universal plug, the modular insert could be aligned, and power and signal connections from the chosen arm effortlessly sent back to the base. The arm selector pins on the insert allowed for the control system to automatically detect the type of arm mounted to the insert and use the respective software features to communicate with the arm. Some of these features included control abstraction of the robot arms using MoveIt! and the state machine code, which allowed for the safe and modular operation of the system as a whole. These components were contained in the insert and could easily be deposited at the home base, which could store different inserts with different arms throughout the robot's workspace.

In addition, the robot achieved its goal of mobility. The holonomic drive implemented with mecanum wheels allowed the robot to rotate, drive, and strafe, adding three degrees of freedom to the arm. The elevators, in conjunction with the tabletop alignment system, allowed the robot arm to elevate and tilt to different angles, adding two more degrees of freedom and vastly expanding the robot's workspace. These components of mobility also allowed the robot to effortlessly interface with the home base to swap out robot arms.

Finally, the robot met its goal of reliability. The battery management system was able to monitor the batteries' health and charge level and signal for the robot to return to the home base to self-charge when the batteries were almost out of charge. The limit switches were able to zero the encoder position of the elevator jacks as well as act as a soft stop, preventing the jacks from depressing too far. Said encoders allowed for easy control of the height of the elevators. Simulation of the system in Gazebo before implementation on the physical robot allowed for the safety and viability of the code to be determined.

This robot progressed from an idea to a fully designed and capable robot in its first iteration. The robot met its goals of modularity, mobility, and reliability. It will be exciting to see how this robot progresses in future MQP iterations.

9 Citations

1. Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R. & Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In ICRA Workshop on Open Source Software.
2. H. Bruyninckx, "Open robot control software: the OROCOS project," Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164), 2001, pp. 2523-2528 vol.3, doi: 10.1109/ROBOT.2001.933002.
3. Juan, S. H. and Tur, J. M. M. 2006. Generic Modular Framework for Robotic Arm Applications. International Conference on Computer Systems and Technologies, CompSysTech '06.
4. ROS Distro Usage. ROS Metrics. (n.d.). Retrieved April 28, 2022, from https://metrics.ros.org/packages_rosdistro.html
5. M. Henning and M. Spruiell. Distributed Programming with Ice. 2006.
6. Zhang, W. (n.d.). (PDF) a study on the static stability of scissor lift. ResearchGate. Retrieved from https://www.researchgate.net/publication/284175695_A_Study_on_the_Static_Stability_of_Scissor_Lift
7. Makarenko, A., & Brooks, A. (2006). Orca: Components for robotics. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06).
8. Franka Emika. (2017). Frank Control Interface Overview. Overview - Franka Control Interface (FCI) documentation. Retrieved April 28, 2022, from https://frankaemika.github.io/docs/overview.html
9. F. Poppa and U. Zimmer, "RobotUI - A software architecture for modular robotics user interface frameworks," 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 2571-2576, doi: 10.1109/IROS.2012.6385526.
10. Orca Project. Orca2. http://orca-robotics.sf.net
11. OROCOS, Open Robot Control software Open Robot Control Services, http://www.orocos.org
12. Hou, L., Zhou, F., Kim, K., & Zhang, L. (2021, March 5). Practical model for energy consumption analysis of omnidirectional mobile robot. MDPI, from https://www.mdpi.com/1424-8220/21/5/1800
13. Newans, J. (2021, November 14). Getting ready for Ros part 8: Simulating with Gazebo. Articulated Robotics. Retrieved April 28, 2022, from https://articulatedrobotics.xyz/ready-for-ros-8-gazebo/

14. Bohren, J. SMACH. http://wiki.ros.org/smach
15. Bogdon, C. (2019, July 31). *Clearpath brings Franka Emika Panda Robot to robotics research community*. Clearpath Robotics. Retrieved from https://clearpathrobotics.com/blog/2019/07/clearpath-brings-franka-emika-panda-robot-to-robotics-research-community/
16. Alves, S. (n.d.). Advances in robot navigation. Google Books. Retrieved April 28, 2022, from https://books.google.com/books?hl=en&lr=&id=d-GdDwAAQBAJ&oi=fnd&pg=PA3&dq=mobile%2Brobotic%2Bbases&ots=FKHsY2-CqJ&sig=DhrQO2FblCGYEzUvYGD9gMbk9OU#v=onepage&q=mobile%20robotic%20bases&f=false

10 Appendices

10.1 Use Cases

10.1.1 Initial Use Case Ideas

- I have a lab with a number of different robotic arms. I begin development with one arm for a time period. The arm is requested by another group so I am able to simply switch the arm mounted to the base and continue my development with few to no programming changes.
- I am developing a task with another lab and both labs have different arms. I would like to share code and work with the other lab with minimal changes to the system except for a new arm mounted to the base.
- I am developing a task for a customer (they need an arm to move around a store and collect or move objects) and they have yet to decide which arm they would like to purchase, I want to be able to develop the system with the arm I have on hand and be able to transfer to their new selected arm with minimal changes, hopefully just by changing hardware.
- I have a robotic arm that I would like to expand its area of influence by adding three degrees of freedom (move around the room and move height up and down)
- I have minimal table space and I need a base to operate whatever robot arm I am currently using and I would like switching between the arms to be easy
- Having the mobile base assist transportation of different arms through dirt, paved roads, and carpet
- An arm in a hospital to help a patient brush teeth (the base moves)
- Kenneth wants to transition from one arm to another arm to prototype with different arms
- I have 2 arms, and source code for each runs in different languages. I want to use them both seamlessly on the same base

10.1.2 Final Use Cases

- A researcher is trying to determine which of their two robot arms is better suited for assisting people with everyday tasks around the house. They have set up an experiment where the robot arms must retrieve a glass of water and bring it back to them. The researcher places the first arm and controller onto the mobile base and plugs the power jack in. The researcher initializes the

software controller for the specific arm they chose. The researcher has the robot base and arm complete the task by navigating to the glass of water while avoiding obstacles. The arm picks up the glass and the robot navigates back. The researcher unmounts the first arm and controller and replaces it with the second arm. After briefly initializing the software, the researcher runs the same script and the second robot arm completes the same task on the same mobile base. The researcher notes down which arm performed better.

- A homeowner is using a robot arm to help them perform chores around the house. They place their robot arm on the mobile base so it can follow them throughout the house. They start doing chores in the kitchen which has a hardwood floor. The robot traverses the tile floor without slipping and assists the homeowner. Next, the robot and homeowner go to the living room to continue chores. The robot base seamlessly transitions to the carpeted floor in the living room and navigates without slipping. Once complete, the robot returns to its docking port in the garage which has a tile floor. It docks at its charging port without slipping on the tile floor.
- The robot base has just completed its task. It returns to its docking port and goes to sleep. While awaiting its next command, it begins charging the battery. The robot charges the battery until either the battery is fully charged, or it is summoned to complete a new task. That battery power will be used to power both the base, and then inverted to provide 120Vrms 60Hz power for each arm controller and mounted arm.
- A homeowner is using a robot arm to help them reach a glass off a high shelf and place it on a requested location (countertop). The mobile base navigates in front of the shelf and raises the arm along the z-axis in order for the arm to retrieve the glass. The arm safely retrieves the glass above the mobile base and the mobile base retracts along the z-axis. The mobile base then returns the glass to the requested location by the homeowner (countertop).
- A homeowner is using a robot arm to help them lift a speaker from the ground and put it away on a shelf. The speaker is quite heavy and will require the robotic arm to fully extend in order to pick it up. The robotic arm will pick up the speaker while keeping the base firmly planted on the ground. The center of mass should be maintained by the base and no traction of any wheels will be lost even when the large load of the speaker is applied to the end of the fully extended arm. The arm will then retract and the base will navigate to the homeowner's desired shelf. The mobile base will then raise the robotic arm along the z-axis and the robotic arm will once again fully extend and place the speaker on the shelf. The robotic arm will place the speaker down on the shelf while keeping the base firmly planted on the ground even when the mobile base is raised along the z-axis. The center of mass should be maintained by the base and no traction of any

wheels will be lost even when the large load of the speaker is applied to the end of the fully extended arm.

- A homeowner needs the help of the robotic mobile base and arm to retrieve a children's toy from a tight space. The mobile base moves through the tight space and makes a 90-degree turn in order to navigate toward the toy. The robotic arm picks up the toy and makes another 90-degree turn once again and navigates back through the tight space. The robotic arm places the toy where the homeowner specifically told the mobile base to put it.
- A robot arm is being used to complete chores throughout the house. While performing the task, it needs to understand its orientation so that upon completion it can return to the charging dock. It keeps track of how far each wheel has turned. It also uses cameras and other sensors to build a model of its surroundings. However, since this robot is operating in a house, it needs to be able to detect obstacles that the homeowner may have placed in the path of the robot since the last use. The robot arm completes the task while avoiding obstacles and the base returns to the dock.
- In a warehouse, a worker needs to get a heavy package off of a shelf and put it on a table. From its base, the robot starts navigating to the shelf. In the path to the shelf is someone's lunch box. As the robot is moving, it detects that there is an object in its path and navigates around it. When it reaches the shelf, the base raises up so the package is within the grasp of the arm. The robot then uses computer vision to identify the shape of the box and determines how to properly grab it. It grabs the box, determines the path to the table, and navigates there. It then sets the package down and returns to its base.

10.2 Bill Of Materials

Item	**Money Account**	**Cost+Shipping**
LiFePo Batteries	TinkerBox	$456.60
RSP-2000-24	WPI	$454.16
SD-100B-5	WPI	$39.88
Daygreen 24v-12	WPI	$138.00
inverter	WPI	$119.00

Scissor Jacks	TinkerBox	$169.98
BMS	Tinkerbox	$425.00
Robopads	Tinkerbox	$419.25
chassis metal	WPI	$191.40
suspension (igus)	WPI	$155.81
chassisbrackets+suspension plates	TinkerBox	$276.95
suspension Hardware	TinkerBox	$188.66
fuse box+fuse	Ken	$41.44
insert electronics - mouser	Grant	$217.40
Vex Order	Grant	$1,154.20
table	Grant	$158.08
battery tray	Grant	$48.95
igus pillow blocks	Grant	$30.91
McMaster Axle + Belts	Grant	$75.01
Shoulder Bolts	Grant	$19.95
Bearings - pt1	Grant	$10.61
Bearings - pt2	Grant	$10.61
enc breakout	Grant	$58.71
Solenoid controller comp	Grant	$30.01
insert+table brackets	Grant	$111.07
charger+caps	Alex	$283.73
table plates	TinkerBox	$250.01
home base frame	Alex	$50.70
limit switches + dollies	TinkerBox	$50
vex idler sprockets	TinkerBox	$62.34

lexan siding	TinkerBox	$259.12
Total with free parts		$7305.46

Table 24: Purchased parts and total cost

Product Detail	Customer #	Order Qty.	Price (USD)	Ext. (USD)	Status	Date	Invoice #
Mouser #: 187-CL31A106KAHNNNE Mfr. #: CL31A106KAHNNNE Desc.: Multilayer Ceramic Capacitors MLCC - SMD/SMT Multilayer Ceramic Capacitors MLCC - SMD/SMT 10uF+/-10% 25V X5R 3 1206		2	$0.20	$0.40	2 Shipped	07-Feb-22	66024923
Mouser #: 490-TBL00925415GY2GY Mfr. #: TBL009-254-15GY-2GY Desc.: Fixed Terminal Blocks Fixed Terminal Blocks 2 24 Poles, Screwless, Horizontal, 2.54 Pitch, 26 18 (AWG), Terminal Block Connector		2	$3.35	$6.70	2 Shipped	07-Feb-22	66024923
Mouser #: 771-PSMN015-100B118 Mfr. #: PSMN015-100B,118 Desc.: MOSFET MOSFET TAPE13 PWR-MOS		4	$2.17	$8.68	4 Shipped	07-Feb-22	66024923
Mouser #: 667-ERA-8AEB7323V Mfr. #: ERA-8AEB7323V Desc.: Thin Film Resistors - SMD Thin Film Resistors - SMD 1206 732Kohm 25ppm 0.1% AEC-Q200		4	$0.66	$2.64	4 Shipped	07-Feb-22	66024923
Mouser #: 652-CR1206FX-10R0ELF Mfr. #: CR1206-FX-10R0ELF Desc.: Thick Film Resistors - SMD Thick Film Resistors - SMD 10ohm 1%		2	$0.10	$0.20	2 Shipped	07-Feb-22	66024923
Mouser #: 71-RCS1206510RFKEA Mfr. #: RCS1206510RFKEA Desc.: Thick Film Resistors - SMD Thick Film Resistors - SMD 0.5watt 510ohms 1% 100ppm		2	$0.33	$0.66	2 Shipped	07-Feb-22	66024923
Mouser #: 750-1N4148W-HF Mfr. #: 1N4148W-HF Desc.: Diodes - General Purpose, Power, Switching Diodes - General Purpose, Power, Switching DIODE SWITCHING 100V 150mA 500mW SOD-123		8	$0.18	$1.44	8 Shipped	07-Feb-22	66024923

Table 25: Solenoid driver board bill of materials

10.3 DFMEA

10.3.1 Chassis

DFMEA

Item	Function	Failure Mode
Mecanum Wheel	Allows for movement of robotic base	Mecanum wheel slips
		Roller shears off
Wheel Axle	provides rotational support for the mecanum wheel	Axle shears under weight
Flanged Bearings	Provides support for rotational motion of the mecanum wheels	Cannot support the weight of the robot and plastically deform
		Fall out of alignment
Triangle Bracket	Provides support for the suspension	Buckle from applied load
		Bending from normal operation causes the welds to shear
Wheel Bracket Plate	Provides support for the suspension	Bending from normal operation causes the welds to shear
		Bend from applied load
1/2" Bolt	Allows for springs to compress in an expected way	Bolts could shear or bend
1/2" Hex Nut	Secures the 1/2" bolt in place	Nut threads are tornout
Linear Rail	Guides the suspensions travel path	Excessive forces cause the linear rail to bend, causing misalignment
Linear Bearing	Works in conjuction with linear rail to provide translational guidar	Excessive forces cause the bearing rail to bend, causing misalignment
Linear Bearing Bolt	Locks linear bearing into place	Becomes loose from vibrations
Linear Bearing Nut	Locks linear bearing into place	Becomes loose from vibrations
Linear Rail Locknut	Locks linear bearing into place	Becomes loose from vibrations
1/2" Locknut	Secures the 1/2" bolt in place	Becomes loose from vibrations
Hard Stop Bolt	Provides a mechanical hardstop to prevent springs from pastically	Bend from applied load
Hard Stop Nut	Allow for placement of intermediate steel plate and securing harc	Becomes loose from vibrations
Hard Stop Washers	Disperses forces aquired from weight to prevent excessive pressu	Excessive pressure can plastically deform or crack the washer
Intermediate Steel Plate	Provides alignment for suspension and tensions springs	Material plastically deforms from bolts bending
Top Aluminum Mounting Plate	Connects to chassis and provides alignment for suspension	Material plastically deforms from bolts bending
Spring	Resists against the weight of the robot and forces mecanum whee	Plastically deform from excess weight

DFMEA

Item	Effects	Sev.	Causes	Occ.	Prevention/Detection Controls	Det.	RPN
Mecanum Wheel	mobile base does not move	8	Insufficient friction	3	Operate on expected surfaces	2	48
	mobile base does not move	8	Structual integrity (of the r	1	Perform tests with the maximum load applied	3	24
Wheel Axle	mobile base does not move	8	Weak axel materials	4	Ensure thorough structural integrity through simulations	2	64
Flanged Bearings	Robot becomes either unmoveable or beha	7	Poor material choice, high	2	Perform tests with the maximum load applied	2	28
	Robot moves in an unexpected way	9	Appropriate spacers were	1	ensure that spacers are of an approriate size	1	9
Triangle Bracket	Robot would behave slightly differently	5	poor material choice	1	Choose a material with sufficient material properties	1	5
	The robot is no longer able to move in a coc	9	poor material choice	1	Choose a material with sufficient material properties	1	9
Wheel Bracket Plate	The robot is no longer able to move in a coc	9	poor material choice	1	Choose a material with sufficient material properties	1	9
	Robot would behave slightly differently	5	poor material choice	1	Choose a material with sufficient material properties	1	5
1/2" Bolt	Robot would behave slightly differently	8	poor material choice	1	Choose a material with sufficient material properties	1	8
1/2" Hex Nut	Robot would behave slightly differently	8	poor material choice	1	Choose a material with sufficient material properties	1	8
Linear Rail	The robot is no longer able to move in a coc	8	Excessive bending forces	2	Ensure thorough structural integrity through simulations	3	48
Linear Bearing	The robot is no longer able to move in a coc	8	Excessive bending forces	2	Ensure thorough structural integrity through simulations	3	48
Linear Bearing Bolt	Linear bearing is no longer secured	3	Insignificant tightening	1	Ensure the bolts/nuts are sufficiently tightened	1	3
Linear Bearing Nut	Linear bearing is no longer secured	3	Insignificant tightening	1	Ensure the bolts/nuts are sufficiently tightened	1	3
Linear Rail Locknut	Linear bearing is no longer secured	3	Insignificant tightening	1	Ensure the bolts/nuts are sufficiently tightened	1	3
1/2" Locknut	1/2" bolt no longer secured	3	Insignificant tightening	1	Ensure the bolts/nuts are sufficiently tightened	1	3
Hard Stop Bolt	The robot is no longer able to move in a coc	8	Excessive bending forces	1	Choose a material with sufficient material properties	1	8
Hard Stop Nut	The hardstop is no locker secured	3	Insignificant tightening	1	Ensure the bolts/nuts are sufficiently tightened	1	3
Hard Stop Washers	Robot would behave slightly differently	1	Excessive bending forces	2	Choose appropriate washer/ mounting bracket material ar	2	4
Intermediate Steel Plate	Robot would behave slightly differently	7	Excessive bending forces	1	Choose a material with sufficient material properties	1	7
Top Aluminum Mounting Plate	Robot would behave slightly differently	7	Excessive bending forces	1	Choose a material with sufficient material properties	1	7
Spring	Mobile base does not move	8	Excessive forces	1	Ensure hardstops are in place	1	8

DFMEA		
Item	**Recommended Actions**	**Actions Taken**
Mecanum Wheel	**Perform validation testing**	Only traverse on known materials
	Perform validation testing	Perform validation testing
Wheel Axle	**Perform validation testing**	Perform FEA Simulations for part validation
Flanged Bearings	**Choose bearing that can withstand apprioriate static load**	**Acquired bearing that can withstand apprioriate static load**
	Provide assembly instructions to rectify errors in assembly	Ensure the manufacturer has knowledge of the assembly instructions
Triangle Bracket	**Choose strong enough material to support weight**	**Acquired strong enough material to support weight**
	Choose strong enough material to support torsion	**Acquired strong enough material to support weight**
Wheel Bracket Plate	**Choose strong enough material to support weight**	**Acquired strong enough material to support weight**
	Choose strong enough material to support weight	**Acquired strong enough material to support weight**
1/2" Bolt	**Choose strong enough material to support weight**	**Acquired strong enough material to support weight**
1/2" Hex Nut	**Choose strong enough material to support weight**	**Acquired strong enough material to support weight**
Linear Rail	**Choose strong enough material to support weight**	**Acquired strong enough material to support weight**
Linear Bearing	**Choose linear rail that can withstand apprioriate loads**	**Acquired linear rail that can withstand apprioriate static load**
Linear Bearing Bolt	**Provide assembly instructions to rectify errors in assembly**	Ensure the manufacturer has knowledge of the assembly instructions
Linear Bearing Nut	**Provide assembly instructions to rectify errors in assembly**	Ensure the manufacturer has knowledge of the assembly instructions
Linear Rail Locknut	**Provide assembly instructions to rectify errors in assembly**	Ensure the manufacturer has knowledge of the assembly instructions
1/2" Locknut	**Provide assembly instructions to rectify errors in assembly**	Ensure the manufacturer has knowledge of the assembly instructions
Hard Stop Bolt	**Utilize multiple hardstops to disperse forces**	**Acquire multiple hardstops to disperse forces**
Hard Stop Nut	**Provide assembly instructions to rectify errors in assembly**	Ensure the manufacturer has knowledge of the assembly instructions
Hard Stop Washers	**Choose strong enough material to support weight**	**Acquired strong enough material to support weight**
Intermediate Steel Plate	**Choose strong enough material to support weight**	**Acquired strong enough material to support weight**
Top Aluminum Mounting Plate	**Choose strong enough material to support weight**	Acquired strong enough material to support weight
Spring	**Ensure hardstops are suffient to withstand forces**	Design and acquire sufficient hardstops

10.3.2 Elevator Alignment

DFMEA		
Item	**Function**	**Failure Mode**
Vertical Mounting Brackets	Connects the jacks to the pillow block	Welds break from excessive twisting
		Brackets crack from exessive vertical loading
Shoulder Bolt	Where the pillow block pivotes about. Under direct load from tabletop	Excessive forces cause bending preventing the pillow block to rotate about itself
Lock Nut	Secures the vertical mounting brackets to jacks	Falls out of place from vibration
		Threads tearout from excessive twisting
Pillow Block	Allows for misalignment and pivoting of the tabletop	Plastically deforms from excessive weight
		Plastically deforms from moving past designated range of motion
Pillow Block Bolts	Holds the pillow block in place	Threads tearout from excessive twisting
Pillow Block Nuts	Holds the pillow block in place	Threads tearout from excessive twisting
3D Printed Spacers	Centers the pillow block inbetween the vertical mounting brackets	Plastically deform from excess forces

DFMEA			
Item	**Effects**	**Sev.**	**Causes**
Vertical Mounting Brackets	Elevation no longer functions optimally	7	Excessive twisting of jacks
	Elevation behaves slightly differently	7	Large payload on modular insert
Shoulder Bolt	Pillow block cannot rotate as expected, misalignment is weakened and pivoting is impossible	8	Bending from execessive normal forces
Lock Nut	Elevation no longer is functional	7	Excessive vibration from drivetrain or loose assembly
	Elevation no longer is functional	7	Excessive twisting of jacks
Pillow Block	Elevation behaves slightly differently	5	Large payload on modular insert
	Elevation no longer functions optimally	7	Bending past the physical limit of the pillow block
Pillow Block Bolts	Elevation behaves slightly differently	5	Excessive twisting of jacks
Pillow Block Nuts	Elevation behaves slightly differently	5	Table Top over extends while tilting
3D Printed Spacers	Elevation behaves slightly differently	3	Brackets compress the spacers while table is tilting

DFMEA

Item	Occ.	Prevention/Detection Controls	Det.	RPN
Vertical Mounting Brackets	4	Limit the amount of twisting the jacks can undergo from software prevention	3	84
	1	Choose a material with sufficient material properties	1	7
Shoulder Bolt	3	Choose a material with sufficient material properties	4	96
Lock Nut	3	Ensure tightness of locknut or use a locknut with sufficient tightness properties	2	42
	2	Choose a material with sufficient material properties	1	14
Pillow Block	3	Choose a material with sufficient material properties	5	75
	4	Software prevention to detect the tilt of the tabletop	3	84
Pillow Block Bolts	2	Verify that the part can handle the static loads through spec sheet	3	30
Pillow Block Nuts	2	Verify that the part can handle the static loads through spec sheet	3	30
3D Printed Spacers	2	Print with correct tolerancing and appropriate wall thickness/infill	5	30

Recommended Actions	Actions Taken
Perform validation testing	Written concrete code elements to prevent exessive twisting of jacks
Choose strong enough material to support weight	Acquired strong enough material
Choose strong enough material to support weight	Perform Hand Calculations to verify
Ensure proper assembly	Ensured proper assembly
Choose strong enough material to support weight	Acquired strong enough material
Ensure spec sheet meets requirements of conservative use	Ensure spec sheet meets requirements of conservative use
Read encoder values of jacks to determine tilt	Written concrete code elements to prevent exessive twisting of jacks
Choose that the part can handle the static loads through spec sheet	Chose part that can adequately hold weight according to spec sheet
Choose that the part can handle the static loads through spec sheet	Chose part that can adequately hold weight according to spec sheet
Print with correct tolerancing and appropriate wall thickness/infill	Print with correct tolerancing and appropriate wall thickness/infill

10.4 Programming Examples

The full repository for this project can be found at https://github.com/RoboticMobileBaseMQP/mobile-base. Some examples of various files are written below.

10.4.1 URDF Example

https://github.com/RoboticMobileBaseMQP/mobile-base/blob/main/panda_arm/panda_insert/panda_insert_description/urdf/panda_insert.xacro

```
<?xml version="1.0"?>
<robot name="panda_insert" xmlns:xacro="http://ros.org/wiki/xacro">

  <xacro:arg name="origin" default="0 0 1" />

  <xacro:include filename="$(find franka_description)/robots/panda_arm_hand.urdf.xacro" />
```

```
<link name="world" />

<joint name="insert_link_joint" type="fixed">
  <origin xyz="0 0 0" rpy="0 0 0" />
  <parent link="panda_link0"/>
  <child link="insert_link" />
</joint>

<link name="insert_link">
  <visual>
    <origin xyz="0 0 -.1" rpy="0 0 0" />
    <geometry>
      <box size=".2 .2 .2"/>
    </geometry>
  </visual>
  <collision>
    <origin xyz="0 0 -.1"/>
    <geometry>
      <box size=".2 .2 .2"/>
    </geometry>
  </collision>
  <inertial>
    <origin xyz="-0.01103 0 0.002" rpy="0 0 0"/>
    <mass value="2.844"/>
    <inertia
      ixx="0.018819942" ixy="0" ixz="-0.000101519"
      iyy="0.057333716" iyz="0"
      izz="0.074201740"/>
  </inertial>
</link>

 <!-- Load world prismatic joints-->
<xacro:macro name="prismatic_joint_macro" params="axis_name origin:='0 0 0' axis parent child">
  <joint name="panda_${axis_name}_joint" type="prismatic">
    <origin xyz="${origin}" rpy="0 0 0" />
    <parent link="${parent}"/>
    <child link="${child}" />
    <axis xyz="${axis}"/>
    <limit effort="1000" lower="-1000" upper="1000" velocity="2"/>
  </joint>

  <link name="arm_${axis_name}_link">
  <inertial>
  <origin xyz="0 0 0" rpy="0 0 0" />
  <mass value="1" />
  <inertia
      ixx="1.0" ixy="0.0" ixz="0.0"
      iyy="1.0" iyz="0.0"
      izz="1.0" />
  </inertial>
  </link>

  <transmission name="panda_${axis_name}_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="panda_${axis_name}_joint">
        <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
```

```
      <actuator name="panda_${axis_name}_joint_motor">
          <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>
  </xacro:macro>

  <xacro:prismatic_joint_macro axis_name="x" origin="$(arg origin)" axis="1 0 0" parent="world"
child="arm_x_link"/>
  <xacro:prismatic_joint_macro axis_name="y" axis="0 1 0" parent="arm_x_link" child="arm_y_link"/>
  <xacro:prismatic_joint_macro axis_name="z" axis="0 0 1" parent="arm_y_link" child="arm_z_link"/>

  <!-- Load world rotation joint -->
  <joint name="panda_z_rotation_joint" type="continuous">
    <origin xyz="0 0 0" rpy="0 0 0" />
    <parent link="arm_z_link"/>
    <child link="panda_link0" />
    <axis xyz="0 0 1"/>
    <limit effort="1000.0" velocity="2"/>
  </joint>

  <transmission name="panda_z_rotation_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="panda_z_rotation_joint">
        <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
    </joint>
    <actuator name="panda_z_rotation_joint_motor">
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

  <!-- Gazebo control plugin -->
  <gazebo>
    <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
      <robotNamespace>/panda</robotNamespace>
      <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
    </plugin>
  </gazebo>
</robot>
```

10.4.2 State Machine Example

https://github.com/RoboticMobileBaseMQP/mobile-base/blob/state-machine/task_manager/src/state_machine.py

```
import rospy
import smach
import time
import threading
import queue
from task_manager.msg import task
from service_interface import ServiceInterface
```

```
# define Idle State
class Idle(smach.State):
    def __init__(self):
        smach.State.__init__(self,
                             outcomes=['swap','task_progress', 'shutdown'],
                             output_keys=['task_progress_out'])

        self.mutex = threading.Lock()
        self.received_task = None
        self.task_queue = queue.Queue()

        task_queue_subscriber = rospy.Subscriber('/task_topic', task, self.add_to_queue)


    def add_to_queue(self, msg):
        self.mutex.acquire()
        self.task_queue.put(msg)
        rospy.loginfo('Added new ' + task_type_dict[msg.task_type] + ' task to queue')

        self.mutex.release()
        pass

    def execute(self, userdata):
        global current_task
        rospy.loginfo('Sitting in Idle state')
        waiting_for_task = True

        # spin while waiting for next task
        while waiting_for_task:
            self.mutex.acquire()
            if not self.task_queue.empty(): # if queue is not empty
                waiting_for_task = False

                current_task = self.task_queue.get() # pop from queue
                rospy.loginfo("--- Current task is " + task_type_dict[current_task.task_type] + " task
---")
            self.mutex.release()

        # after receiving task, all progress is reset
        userdata.task_progress_out = {
            'base_moved': False,
            'elevator_moved': False,
            'arm_moved': False,
        }

        if current_task.task_type != current_task.TASK_SHUTDOWN:
            if current_task.task_type == current_task.TASK_SWAP:
                return 'swap'
            else:
                return 'task_progress'
        else:
            return 'shutdown'
```

10.4.3 Strain Gauge Code

https://github.com/ejackson1/RoboticMobileBaseMQP_Testing/blob/master/ADS1x15/strain_gauge_Test.py

```
# Simple demo of reading the difference between channel 1 and 0 on an ADS1x15 ADC.
# Author: Tony DiCola
# License: Public Domain
import time
import math
import matplotlib.pyplot as plt
import logging


# Import the ADS1x15 module.
from ADS1x15 import ADS1115


# Create an ADS1115 ADC (16-bit) instance.
#adc = ADS1115()

# Or create an ADS1015 ADC (12-bit) instance.
adc = ADS1115()

# Note you can change the I2C address from its default (0x48), and/or the I2C
# bus by passing in these optional parameters:
#adc = ADS1015(address=0x49, busnum=1)

# Choose a gain of 1 for reading voltages from 0 to 4.09V.
# Or pick a different gain to change the range of voltages that are read:
#  - 2/3 = +/-6.144V
#  -   1 = +/-4.096V
#  -   2 = +/-2.048V
#  -   4 = +/-1.024V
#  -   8 = +/-0.512V
#  -  16 = +/-0.256V
# See table 3 in the ADS1015/ADS1115 datasheet for more info on gain.
GAIN = 16

voltList = []
stressList = []
timeList = []

logging.basicConfig(filename='data.log', filemode='w',level=logging.DEBUG)

Vex = 5
R = 350
GF = 123
E = 69 * 10 ** 9


print('Press Ctrl-C to quit...')
timeStart = time.time()
try:
    while True:
        # Read the difference between channel 0 and 1 (i.e. channel 0 minus channel 1).
```

```
        # Note you can change the differential value to the following:
        #  - 0 = Channel 0 minus channel 1
        #  - 1 = Channel 0 minus channel 3
        #  - 2 = Channel 1 minus channel 3
        #  - 3 = Channel 2 minus channel 3
        value = adc.read_adc_difference(0, gain=GAIN)
        # Note you can also pass an optional data_rate parameter above, see
        # simpletest.py and the read_adc function for more information.
        # Value will be a signed 12 or 16 bit integer value (depending on the ADC
        # precision, ADS1015 = 12-bit or ADS1115 = 16-bit).
        factor = value / 32767
        volts = factor * 4.096/GAIN
        voltList.append(volts)

        timerec = time.time()
        delta = timerec -timeStart

        timeList.append(delta)
        #print(volts)
        logging.info("volts {}".format(volts) + ", " + "time {}".format(delta))
        print('Channel 0 minus 1: {0}'.format(value))
        print('Voltage: {0}'.format(volts))


        # Pause for 0.01 seconds.
        time.sleep(0.01)

except KeyboardInterrupt:
    for volts in voltList:
        deltaR = volts / Vex * R
        strain = deltaR / (R * GF)
        stress = E * strain
        stressList.append(stress)




    plt.plot(timeList, stressList)
    plt.ylabel("Stress (N/m2)")
    plt.xlabel("Time(s)")
    plt.show()
except Exception as e:
    print(e)
```

10.5 Robotic Arm Data Sheets

Franka Emika Panda:

https://wiredworkers.io/wp-content/uploads/2019/12/Panda_FrankaEmika_ENG.pdf

UR5e: https://www.universal-robots.com/media/1807465/ur5e-rgb-fact-sheet-landscape-a4.pdf

Kinova Gen3:
https://artifactory.kinovaapps.com/artifactory/generic-documentation-public/Documentation/Gen3/Technical%20documentation/User%20Guide/User%20Guide%20Gen3%20-%20R06.pdf