# UCF Senior Design I: Final Draft

*MAC: Modular Autonomous Cart that Assists with Intra-home Object Transportation*

*Department of Electrical and Computer Engineering*
*University of Central Florida*
*Dr. Lei Wei, Dr. Samuel Richie*

## Group 8



| | | |
|---|---|---|
| Brandon Silva | Computer Engineer | brandon.silva209@knights.ucf.edu |
| Justin TenEyck | Computer Engineer | jteneyck@knights.ucf.edu |
| Lisian Shehu | Computer Engineer | lisianshehu08@knights.ucf.edu |
| Trevor Taulien | Electrical Engineer | trevortaulien@knights.ucf.edu |

# 1) Executive Summary

Modern homes are now filled with smart-devices intended to automate and improve the day-to-day life of people. Robots are also making headway in the smart home market with products such as the Roomba being able to traverse through homes as it vacuums. With an ever growing and competitive market, we believe there is an opportunity to build a robotic system to improve daily home life for the elderly and people with disabilities. As of 2014, approximately 26% of eldlerly live alone and as of 2016 1 in 4 of adults in the US live alone with a disability including mobility, cognition, hearing, vision, independent living and self-care according to the CDC. A disability that inhibits mobility is actually most common affecting 1 in 7 adults. This project aims to build a robotic system that will be geared for easing the transportation of home objects such as laundry, groceries or small furniture items within the home.

The two key components of the MAC are that it is to be modular and autonomous. The modular component of the robot will allow the user to interchange different storage structures such as a rack for smaller items for groceries or a larger storage basket for bigger items that can include laundry or small furniture. The MAC will also be able to follow the user to their desired destination so that it is available to help at any time the user sees fit. Since the MAC is geared towards improving daily living of the elderly and people with disabilities, there is a focus on making it easy to use, durable and robust. Ease of use requires having an interface that is simple such as a remote control with minimal buttons to command the system. The system needs to be robust so that it is able to handle loads of various weights and shapes. Finally, the system will be built to be durable to take any hits or drops as it is navigating through a household. These are the main project constraints that will drive both the hardware and software design of the system.

The MAC will use a camera as well as a variety of sensors such as LiDAR, ultrasonic and weight sensors. The built-in camera paired with computer vision will be used to help the MAC follow the user that will be walking in-front of them. The LiDAR and ultrasonic sensors will also aid in navigation and path planning. The MAC will contain a 4-wheel standard drivetrain, allowing the robot to turn in-place and keep high traction on surfaces without damaging the surface itself. This provides increased stability while carrying a load, and keeps the design compact in terms of the implementation of the wheels, motors and gears. The MAC will be battery powered and chargeable.

The MAC will be designed and built to ease the stress of transporting heavy cargo in the house, especially for eldelery users and users with disabilities. It was of the utmost importance to consider not only how the system will operate but also how it can operate in a manner that is safe, reliable and easy to use. A user will be able to remotely command the MAC to follow the user to their destination, place a load using the various storage methods and finally call on the MAC to follow them to their final destination where they can unload whatever they decided to transport. The MAC will offer a solution that will improve daily living as well as prevent any serious injuries in heavy lifting.

# 2) Project Description

## 2.1) Project Motivation, Goals, and Objectives

Disabled and elderly people have always had a tough time living on their own. Through many different technologies, it has become easier and easier for these people to live relatively normal lives without assistance from others, like in a nursing home or assisted living facility. There are devices to help people into their showers, up and down stairs, and make food preparation and other household chores easier. With this, robotics in the home has slowly but steadily been increasing in popularity, with one the most common applications being robot vacuums for automated cleaning of household floors. However, automatic carts or robots to move objects have little market for the average consumer, let alone someone with specific disabilities or needs. For such a common task of needing to lift or transport large bulky objects or many small objects, there needs to be a device that can reliably transport cargo from one part of a home to another, thus the need for a device like the MAC. Currently similar technologies exist for business or warehouse applications, like in hospitals to transport supplies around automatically, or moving packages around in an Amazon warehouse. These carts are big, bulky, and expensive, while needing advanced knowledge in operating the device. These are not viable for home use, or even affordable to our target demographic. According to the AARP, the average income of someone aged 65 or older is $31,742 per year, with half of the people falling below the median income of $19,604. For disabled people, the average yearly income for 2021 is around 16,000 dollars, about 1300 dollars a month. The MAC will be made with this in mind, keeping the cost within reason of someone on this type of income, unlike what is currently available.

Our team's goal is to apply the knowledge we have learned to a complex system of hardware and software to help those in need with a common task that usually can be hard to complete, or requires much time and effort from someone who may not be able to exert themselves as needed.

The main goal of the MAC is to automate transportation of objects around the household. Whether it be large bulky objects like laundry baskets, chairs, small tables, a container, or many small objects like groceries, cleaning supplies, folded clothes, electronics, the MAC will alleviate the need to get outside help or require many trips to move these everyday objects. Not only for those who may not be able to walk, but for those who are weakened and can not exert themselves to do these tasks. Because we are targeting an older population, the other goals of the MAC is to have an easy to use interface, along with a reliable robot that won't require frequent maintenance or recalibration. Whether young, old, technology competent or not, the MAC should be easy to use by all. Furthermore, because of this accessibility requirement the MAC needs to be reliable enough where it won't require constant maintenance, calibrations, resets, tinkering etc to keep it operational. If the MAC can't reliably offer it's services, then it becomes useless compared to manually doing the task it is trying to automate.
Besides accessibility, the MAC is designed also with safety in mind, especially with software restrictions on operation to ensure the MAC does not damage itself or its environment. The MAC needs to ensure the safety of the user by maintaining safe following distances, give extra

clearance to obstacles, and operate slow enough that it can stop or avoid obstacles without failure. This is so the MAC gives clearance to eldery people who may have walkers or wheelchairs, and disabled people who may be in an electric scooter. All of these motivations combine into our main object: to improve the lives of disabled and eldery people by making living on their own more convenient and possible, to avoid the need for assisted living.

## 2.3) Requirements and Specifications

The below table lists the requirements for the MAC.

- "Robot" refers to the MAC system.

| Requirement | Specification | Value | Unit |
|---|---|---|---|
| Size | Robot needs to fit inside and maneuver in standard homes / apartments | 2.5' x 3' x 3' | Feet^3 |
| Durability | Robot needs to take hits, bumps, spills, items being dropped on it, and possibly being knocked over | Handle 150 pounds dropped on it, IP54 for electronics compartments | Weight, IP |
| Ease of Use | Robot has a remote with clearly labeled and easy to press buttons, including an onboard panel with readouts on power, battery and efficiency | One remote or single onboard panel | Controllers |
| Cost | Robot should be priced so middle income users can afford it | $1,000 | USD |
| Battery Life | Robot should be able to last through normal use without dying | 15 minutes at 100 pounds, 30 minutes at 50 pounds, 1 hour for 25 pounds, and 2 hours for 0 pounds | Time to load |
| Charging Time | Robot should fully charge in a reasonable amount of time | 6 hours | Time |
| Weight Load | Robot should handle a max load that can handle most use cases | 100 pounds | Weight |
| Speed | Robot needs to be slow to ensure saftey of user and environment | 3 feet / second when traveling, 2 feet / second when following user<br>Lifts at 2 inch / second | Feet per second<br>Inch per second |
| Range of Remote Control | Robot needs to communicate with user anywhere in the home | 200 feet from wireless access point (depending on power of WAP) | Feet |
| Stopping Distance | Robot won't ever travel fast, but it needs to be able to stop quickly for safety | 6 | Inch |
| Intelligence | Robot will utilize multiple sensors to plan paths when traveling, replan when necessary, and process everything on board | Path planning done in 5 seconds, replanning done in less than one second, responds to obstacles in less than 100 milliseconds | Time |

Table 1: Requirement Specifications

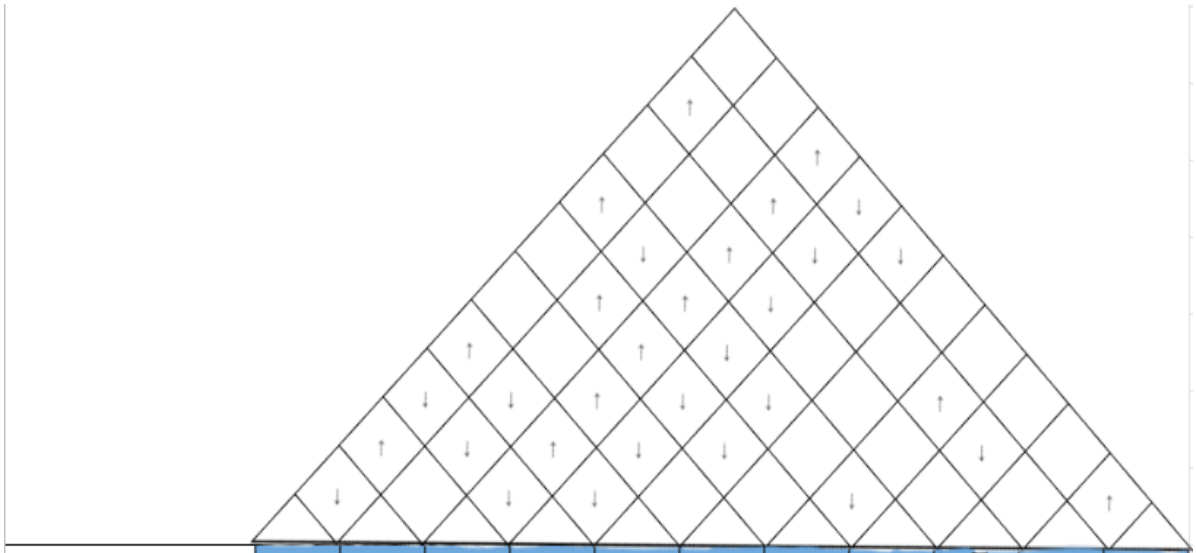
## 2.4) House of Quality Analysis

Our House of Quality portrays marketing requirements (row) versus engineering requirements (column), the targets for those requirements listed at the bottom of the chart (Table 2). Our marketing requirements are that the MAC should be easy to use for people of all ages and abilities, tough enough to withstand the wear and tear of daily, continuous use, last for multiple years requiring little maintenance, and be affordable to a larger portion of our target market.

Our biggest considerations are ease of use, intelligence, and durability. Not only should the MAC be smart enough to avoid obstacles, plan paths that leave large enough clearance in the environment, and stop far away from obstacles, it needs to be durable enough such that regular maintenance is not needed. It would be hard for our target demographic to perform maintenance on a highly technical device, so the device needs to be robust enough to only require minimal maintenance after large spans of time (6 months - 1 year). This will ensure the MAC won't cause more trouble than it can help its users.

As shown in the house of quality, the ease of use is a huge priority for both engineering and marketing of the MAC. Users of any technical ability or any disability should be able to operate the MAC with minimal effort, as simple as operating a landline phone or TV. This will be done through a combination of a remote control, touch screen display on the MAC, and a smart phone companion app. The main way to use the MAC will be the provided remote controller, which will have some buttons to control the various functions. The touch screen display on the MAC itself is provided for convenience in case the remote is missing or unavailable, and also to display errors for diagnostic purposes to make it easier to repair, or know when it needs a reboot. The smart phone companion app is a stretch goal for us to provide the functionality of remotely calling the MAC to a specific area of the home, instead of having to go to the MAC to have it follow using the line of sight following feature. On the smartphone app the user will see the MAC's currently mapped out environment using the SLAM algorithm, where they can select where the MAC is needed. When this selection is made the MAC will plan a path and go there, without needing a remote, additional beacons or other hardware. The smartphone app does make it a bit harder to operate the MAC, especially for less technical users, so all functionality except the remote calling is available in the remote itself.

The intelligence of the MAC is the other priority for our design. This will mainly come from the software the MAC will use. The MAC will be employing SLAM mapping using LiDAR, ultrasonic, and encoder sensors, Dynamic Window Approach path planning, Yolo V3 deep learning model to detect people and objects using just a front facing camera, and ROS to tie everything together into a fully functioning robot. This will allow our MAC to be responsive to its environment, which will be helpful not only in safety but how efficient it is in navigating someone's home. This will be important to allow us to meet the requirements set forth in Table 1 and focus on the priorities shown in Table 2.



| | | Size | Durability | Ease of Use | Cost | Battery Life | Charging Time | Weight Load | Speed | Range of Remote Control | Stopping Distance | Intelligence |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | - | + | + | - | + | - | + | + | + | - | + |
| 1) Ease of Use | + | ↑ | | ↑↑ | ↓ | ↑ | ↑ | ↑ | ↑ | ↑↑ | ↑ | ↑↑ |
| 2) Durability | + | | ↑↑ | | ↓ | ↓ | | ↑↑ | ↓ | | | |
| 3) Lifespan | + | | ↑↑ | | ↓↓ | | | ↑ | ↓ | | | ↑ |
| 4) Cost | - | ↑ | ↓ | ↓ | ↑↑ | ↓ | ↓ | ↓↓ | ↓ | ↓↓ | ↓ | ↓↓ |
| **Targets for Engineering Requirements** | | ≤ 2.5' x 3' x 3' | ≥ 150 lbs of vertical force, IP54 | 1 remote, 1 onboard display | ≤ $1000 | 15 min at 100 lbs, 30 min at 50 lbs, 60 min at 25 lbs, 120 min at 0 lbs | ≤ 6 hrs | ≤ 100 lbs | 3 ft when traveling, 2 ft when following, 2 in/sec lifting | ≤ 200 ft | ≤ 6 ft | Path planning ≤ 5 sec, replanning ≤ 1 sec, responds to obstacles ≤ 100 ms |

Table 2: House of Quality

# 3) Research Related to Project Definition

## 3.1) Existing Similar Projects or Products

### 3.1.1) Warehouse Applications

#### 3.1.1.1) Canvas Technology Carts

Canvas Technology's robotic carts have full vision from floor to ceiling, identifying objects and people as well as predicting their actions [1]. This behavior enables their carts to perform tasks such as object avoidance, making for efficient and non-invasive navigation in the factories that they are deployed in. Their carts handle equipment within factories and warehouses, reducing forklift traffic and the like. They are designed to handle long-term mileage autonomously in industrial facilities, stating that their autonomous system enables efficient operation in 80-90% of its intended environments. By use of a mix of high-performance sensors (notably LiDAR is not one of them) with simultaneous localization and mapping software, these carts build and update maps in real-time, storing that data [2]. Each cart shares its map with others in its fleet, allowing each cart to learn from each other about the environment they are in. Their carts feature a web application which allows users to set up waypoints, stopping points and routes at their facilities. Though not as beneficial for our project, these carts also integrate with enterprise warehouse management systems in order to track orders or scheduled loading and unloading of goods.

#### 3.1.1.2) Voith Panda

Considering this project is also exploring the relationships and operations of multiple autonomous systems, Voith's Panda robotic arm covers the necessary research to demonstrate the utility of this aspect [3]. Designed to function like a human arm, Panda is being used in manufacturing and sorting facilities to manipulate, assemble and move various products or packages. Given Panda's versatility, manufacturers and warehouses can configure the programming of Panda quickly through their integrated app, switching the cobot (computer-controlled robotic device) from assembly to inventory control on the fly. With IoT functionality, Panda communicates with other robots and warehouse management systems, loading supplies from a specific bin onto autonomous carts. In this specific case, Panda works hand in hand with Canvas Technology's autonomous carts. The integration of both autonomous systems removes the physical need for employees to handle heavy packages or assemblies, offloading the majority of the physical labor to these two autonomous systems. As a drawback, while Canvas Technology's carts can handle a heavier load, Panda robotic arms can only handle up to 3 kg. As this project is interested in the potential design of an accompanying robotic arm to handle packages or goods our users may interact with, such a product serves as a good example.

#### 3.1.1.3) Fetch Robotics CartConnnect AMR

CartConnect AMRs, more widely used, transport material within warehouse facilities by attaching itself to rolling shelves for boxes and other goods [4]. By design, these robots are sold with rolling carts that they can attach and detach from, making their application simple and efficient. These robots feature the use of 2D and 3Ds, perform dynamic obstacle avoidance, have occlusion detection, and make use of a tank drive system that allows them to turn in place. For vision support as well as indicators to help others notice the robot, the AMR also has floor-illuminating blue lights. AMR is managed through FetchCore, the company's cloud-based software, which integrates directly with a facility's existing management and IT systems. As far as functionality is concerned, AMRs pick up, transport and drop off the accompanying FetchCarts, which employees have loaded or unloaded. Pick up and drop off locations can be specified within their user application. Fundamentally, they are intended to reduce travel time and fatigue while increasing productivity, freeing employees to do more with their time while integrating into a pre-existing warehouse setting, requiring minimal changes to current processes.

#### 3.1.1.4) Twinny NarGo and TarGo

Twiny's NarGo and TarGo autonomous carts are currently being used by South Korean telecommunications company KT in their smartphone warehouses [5]. Making use of 5G technology, they were able to reduce staff movement by 47%. With a built-in monitoring system, the carts relay feedback and data on their movements and efficiency, allowing the company to make improvements based on that data. NarGo operates as a chain of multiple carts, with the front cart driving the rest of them (set up as a train). This allows for moving large amounts of cargo, with each individual cart carrying 100 kg of goods. TarGo on the other hand makes use of various technologies so that it can follow a person that it is designated to, helping staff with sorting goods. TarGo also has a carrying capacity of 100 kg. KT, the company which has implemented these carts, sought this solution in part to reduce human-to-human contact during the COVID-19 pandemic.

### 3.1.2) Delivery Applications

#### 3.1.2.1) Postmates Serve

Now owned by Postmates' spin-off company Serve Robotics, Serve was planned to be deployed across more than 550 cities in the United States which Postmates currently operates in [6]. Serve can carry up to 50 lbs of cargo and has a battery range of 30 miles, allowing it to handle a dozen deliveries per day. Making use of the sidewalks in urban as well as suburban areas, Serve is more time efficient by avoiding traffic (which would inevitably lead to traffic congestion). Postmates' original goal of introducing Serve is not just based on efficiency but also reducing emissions of cars and motorcycles which many of their 350,000 couriers use. Making use of Velodyne LiDAR sensors with an onboard Nvidia processor, Serve navigates and avoids obstacles while managing to make deliveries directly to people's homes. Serve is also equipped with indicator lights which act as turn signals and other features to suggest the direction in which it is heading. Given the scale at which these autonomous robots are deployed, a small touchscreen at the top of each bot is available if customers need to call for assistance. Remote supervision is also performed on

these robots as well. As of today, Serve Robotics plans to deploy a small fleet of these robots in Los Angeles of this year. By specifications, Serve is 30 inches long and 21 inches wide, with a height of 40 inches. Serve's maximum movement speed is 3 mph [7].

#### 3.1.2.2) Amazon Scout

As a 6-wheeled robot, Amazon Scout delivers packages to customers currently in Washington and California state [8]. The robot moves on sidewalks at a walking pace, carrying packages which are stored in a sealed container onboard and delivered directly to the customer. As a fully electric robot, Amazon created Scout to move towards net-zero carbon emission by 2040. For Amazon, the use of this autonomous delivery robot has helped speed up the average shipping and delivery time for the products they sell. Similar to the rest of the autonomous robots in this research section, Amazon Scout uses objection avoidance and collision detection methods for obstacles such as pets and pedestrians. This is done via a variety of sensors and the incorporation of Machine Learning. Notably, Scout does not use GPS as it is neither reliable nor detailed enough to track the location of the robot. Scout is built to be weather-resistant, facing heavy rains, winds, and snowfall during field testing as well as normal operations. The interior compartment which stores packages is also protected completely from the weather [9]. Additionally, federal and local regulations are limiting the current scalability of the robot, as they "limit the use of fully automated devices in public space." Similar to Serve, Scout has a carrying capacity of 50 lb, with dimensions 30in x 24in x 29in. The robot itself weighs 100 lb and has a maximum speed of 15 mph.

#### 3.1.2.3) FedEx Roxo

FedEx' Roxo makes use of many similar technologies to Serve and Scout, such as artificial intelligence and various sensors for obstacle detection and avoidance [10]. Uniquely, the system makes use of six wheels, with two on the front to climb stairs. By lifting up the front wheels and subsequently applying downward force, Roxo can hoist itself up a set of stairs, albeit slowly. Roxo can also navigate unpaved surfaces and curbs aside from stairs. The robot itself weighs 200 lbs and can carry up to 100 lbs, a step up from Amazon and Postmates' autonomous delivery robots [11]. Roxo makes use of software from iBot that is designed to be pedestrian-safe, while making use of LiDar and multiple cameras. Once again, as this system is battery-powered, it is a zero-emission system targeted to save money and be environmentally-friendly for FedEx [12]. The robot is planned for its first international application in Dubai, United Arab Emirates. By dimensions, the robot is 36 inches x 28 inches x 58 inches with a top speed of 10 mph. FedEx has partnered with other companies such as Pizza Hut to make efficient use of the delivery robot when not delivering packages ordered through FedEx.

#### 3.1.2.4) Starship Robot

Starship's autonomous delivery robots can carry items within a 4-mile radius of their charging stations, where the company has networked with cities so that their robots can make deliveries on groceries [13]. Parcels, one of the companies partnered with Starship, allows users to order from their mobile app and have their groceries delivered using Startship's robot. Users can then monitor the robots' journey from their phone. The cargo bay which contains goods transported is

mechanically locked and opened by users using the smartphone app. The idea of app integration for tracking and interacting with the robot will be seen in a later section of our research. The robot unloaded weighs 55 lbs, and can carry a max payload of 22 lb. In the event that autonomous operation fails, the robot can be remote-controlled by the company (making the system only viable for short-range local delivery) [14]. The dimensions of this robot are 26.7 inches x 21.8 inches x 49.1 inches and travels at a maximum speed of 3.7 mph.

### 3.1.3) Retail Applications

#### 3.1.3.1) E-mart Eli

Given the detailed functionality of MAC, our project found it necessary to explore products that handle detailed and intimate user experiences. Located in Korea, E-Mart's warehouse-style supermarket Traders plans to make use of the autonomous grocery cart Eli [15]. Eli recognizes human voices and avoids obstacles, following specific consumers that it is paired with via face as well as voice detection. The robot is designed to guide shoppers to find items they desire, displaying the locations on a built-in map of the store. Goods can then be paid for directly from the cart, scanning barcodes of products at the end of the customer's shopping experience. Using a mobile application, users can then pay for the goods they've scanned [16]. The cart itself compares the weight of the items in it in order to determine whether or not the items within it have been paid. After the user is done with the cart, Eli automatically queues at a charging station, returning automatically. Additionally, Eli has functionality to search for a car's location in a parking lot, check for discount coupons, and display the duration spent shopping.

#### 3.1.3.2) Amazon Dash Cart

Using a combination of computer vision algorithms and sensor fusion, Amazon's Dash Cart identifies items you put in the cart. Items are automatically identified in the cart when you leave the store and your payment is processed to the credit card on file with your Amazon account [17]. A screen at the top of the cart allows you to access your Alexa shopping list, checking items off and viewing your current total charge. Coupons can be scanned from the cart, allowing you to quickly apply them on the go. You can also perform price look-ups for items before you place them in your cart. The robot is neatly integrated with Amazon's other products such as Amazon's account services and Alexa, relying on accompanying technologies for advanced functionality. QR codes in the Amazon app allow you to sign in and begin using the cart, and upon finishing, an email of your receipt is sent to you [18]. By design, Dash Cart only supports small-to-medium sized grocery trips, fitting up to two grocery bags. As such, the system doesn't support large warehouse-style shopping, but instead a trip for a week's worth of groceries. A special lane at the checkout registers the cart, notifying it to process payment of all the goods you have in your cart upon leaving.

#### 3.1.3.3) Caper Cart

Similar to Eli and the Dash Cart, Caper Cart uses a series of sensors, including a weight sensor, to determine the item's value as sold by weight [19]. A touchscreen is featured on the cart's handle which displays a running tally of items selected to be purchased, and a card reader to

allow customers to pay for their purchases without going up to the register. Similar to Eli but unlike Dash Cart, Caper Cart has you pay before you leave the store, something which is not automatic. Uniquely, Caper Cart provides shopping list recommendations and promotional offers based on previous purchases. Like Eli, Caper Cart also has wayfinding abilities, directing shoppers verbally to their destination versus visually on a map [20]. The company's inspiration for developing this technology stems from contactless shopping experiences, especially during the pandemic, but also ease of implementation (which doesn't require an entire overhaul of the shopping experience).

### 3.1.4) Medical Applications

#### 3.1.4.1) Aethon TUG

TUG autonomous robots are designed to operate for long periods of time in hospitals, where 24/7 care is necessary [21]. It is designed to move materials and clinical supplies, allowing staff to focus more on patients versus logistics, overall improving the efficiency of hospitals. Nurses can put in "orders" for specific medicines, meals, supplies and tests, tracking when TUG will arrive with those goods. TUG is integrated with medical inventory systems, with waypoints mapping to supply rooms where it retrieves and delivers medications through the hospital and directly to nursing units. TUG can handle controlled substances in sealed containers, and given its modular design, can be outfitted to handle trash as well, with an automatic dumping system (removing the need for employee assistance). Our project has taken a lot of inspiration from this particular design (with modularity in mind), as TUG's back platform can house sealed containers, trash cans, tiered trays and more. Through biometric security and pin codes, TUG confirms that only authorized medical personnel add or remove goods in the secured container/cabinet.

## 3.2) Hardware Research

### 3.2.1) General Materials

#### 3.2.1.1) Preliminary

The following materials below were evaluated on the basis of their utility in this project, relevant to the overall construction of MAC. Considerations were necessary as to what materials would be used, given that various weights of material would affect the overall center of mass [22]. Certain mechanisms or systems on the robot require either lighter or denser material, depending on its use or forces applied, so the sections below briefly evaluate the materials considered for use, focusing on what their general properties are and how they are commonly used.

#### 3.2.1.2) Aluminum

More expensive than steel, aluminum is highly malleable, making it easy to shape and bend. The material is considerably lighter, and protects interior materials used on the project from rusting as aluminum does not rust. Overall, aluminum is reasonably durable, often being used to protect more fragile parts [23]. Pure aluminum is quite soft and weak, so adding small amounts of

silicon and iron can help strengthen aluminum significantly. Additionally, aluminum is a great conductor of heat and electricity. It is often used for aircraft construction, building materials, appliances, and cooking utensils.

#### 3.2.1.3) Steel

Most commonly used to build full-scale, industrial robots, steel can be hardened between 100,000 and 300,000 psi. Aside from hardened steel variants, other types of steel deal well with frequent impact forces. While the overall durability of steel is high (low malleability), it is considerably heavier than aluminum based on density and as such has to be carefully considered when being added to MAC. It is also more challenging to work with than aluminum, requiring special drill bits and tools to work with it. This includes having to perform such tasks as machining, welding and bending. On the plus side, steel is typically cheaper than aluminum.

#### 3.2.1.4) PVC

PVC, or Polyvinyl Chloride, is considered for this project given the material is very lightweight and enables the creation of modular components that are still strong. Given that there are two different forms of PVC, rigid and flexible, rigid PVC includes pipes that can be useful in building various supports that are not especially load-bearing. In many applications, PVC is an alternative to rubber, and also is used in the production of canvas. Plasticizers are used to make the material softer and more flexible.

#### 3.2.1.5) Plastics

Various plastics such as Lexan are useful in creating sealed enclosures for electronics on MAC. In some cases, visibility through transparent plastics is useful for prototyping as well as producing the final product. Given how lightweight plastics generally are, using this material can make the overall weight of the robot less. Plastics can also functionally be used to achieve some simple, mechanical actions.

#### 3.2.1.6) 3D Printing

For non structural components that require complex geometry 3D printing is the best option available. These properties are desirable because the only tools required are a CAD program, and a printer, and you can make almost anything you can model.This is especially useful for enclosures of things that users interact with like remotes. 3D printing comes in many different shapes and sizes depending on the goal of the print and the components requirements.

### 3.2.2) Drivetrains

#### 3.2.2.1) Preliminary

Below is a running list of various drivetrain designs we considered and did research on. When considering drive trains, our team looked at a number of design choices that would be directly impactful to the performance and efficiency of MAC [24]. These choices include traversing around obstacles, moving up inclined planes, traction to the ground (but avoiding causing

damage to a variety of surfaces when traversing), overall stability of the robot (stability over agility), center of mass, maximum speed and torque required for moving loads. Given that there were no particular external design constraints imposed by the project outline, we explored each of the following drive trains with respect to the design choices listed above.

#### 3.2.2.2) Standard Drive

Also known as a skid steer drive, standard drivetrain is usually powered by two motors, applying the power directly to the drive wheels or via a gear train (supporting multiple drive wheels, i.e. 2-wheel, 4-wheel drive or 6-wheel drive). As a disadvantage, standard drive is not omnidirectional, so strafing is not an option here. By comparison, standard drive is most akin to how modern motor vehicles operate. This means that you have to factor in a turning radius when traversing an environment, as returning to the same location when turning requires you to make a complete loop.

#### 3.2.2.3) Tank Drive

Unlike a standard drivetrain, tank drive robots do not traverse their environment as a motor vehicle would but instead turn in place, where the wheels on one side are driven opposite of the other side. Since tank drive does not make use of treads (track drive), a motor and gear train on each wheel is usually necessary to develop enough torque to turn. Stability for this style drivetrain is typically just as good as a standard drivetrain, with the added advantage of allowing the robot to reorient itself in a tight space. Similar to standard drivetrain, tank drive is also not omnidirectional.

#### 3.2.2.4) Standard Omnidirectional

For a standard omnidirectional drivetrain, 4 or 6 wheels are driven by one motor each (or a gear train is used to drive multiple wheels from one motor). Given that the wheels have horizontal rollers, the robot can strafe from side to side, producing horizontal motion despite being oriented in the same direction. In this case, stability is sacrificed for increased mobility, and collision with other objects can easily move the robot depending on its weight. In the event that a robot is working in a closed or small space, omnidirectional drive makes it easy for the robot to enter and traverse the environment without necessarily having to turn.

#### 3.2.2.5) H Drive

Unlike standard omnidirectional, H Drive features a cross section in the chassis that braces both sides of the drive train together, with an omnidirectional wheel set perpendicular in the center. Typically, H Drive has five omnidirectional wheels, driven by three or five motors. H Drive is built to utilize 2.75 inch, 3.25 inch or 4 inch omnidirectional wheels. Similar to standard drive, while strafing or sideways movement is an option (increasing mobility), stability is sacrificed given the fact that collision with other objects may cause the entire robot to slide. Additionally, the fifth center wheel could be caught by an obstacle as the robot traverses over it, putting off center of balance with the risk of high-centering on the obstacle, causing the robot to get stuck.

#### 3.2.2.6) Mecanum

Utilizing Mecanum wheels, mecanum drivetrains feature angled rollers allowing for omnidirectional movement. When the wheels rotate opposite of each other, the orientation of the rollers on the mecanum wheel cause the drivetrain to move sideways. An added advantage that mecanum drivetrains have over omnidirectional is that thanks to their angled rollers, powering two diagonal but opposite wheels at one time allow for diagonal motion as well, allowing the robot to have a finer range of motion (maneuvering tight spaces and the like). As a consequence, angled rollers require more torque from the motors in order to drive the wheels. More complex programming is also necessary to produce the motion desired from the robot than a standard drivetrain.

#### 3.2.2.7) Holonomic

Holonomic drivetrains are omnidirectional, where either three or four omnidirectional wheels and motors are seen (one wheel to motor, respectively). Again, wheel sizes can be 2.75 inches, 3.25 inches, or 4 inches. In a 3-wheel drive holonomic system, each wheel and drive motor are set at 120 degrees to each other. In a 4-wheel system, each wheel and drive motor are set at 90 degrees to each other, with each wheel facing perpendicular to the center from each side of the robot. 4-wheel drivetrains are typically more stable than 3-wheel drivetrains, but both variants are more complex to program. The advantage of this system is that its configuration allows for similar traversal as a mecanum drive, with the added step of requiring the robot to turn itself in the direction it wants to strafe.

#### 3.2.2.8) Track Drive

As another variation of the tank drivetrain, track drive makes use of tank treads instead of wheels. As a result, the drivetrain lacks the ability to be omnidirectional. Instead, these drivetrains typically have good traction and can easily traverse over obstacles in its path. Tank drive can come in 4 to 6-wheel variants, with each side usually driven by one motor (which can also feature a gear train).

### 3.2.3) Lifting Mechanisms

#### 3.2.3.1) Preliminary

Below is a running list of various lifting mechanisms we considered and did research on. When considering different ways of lifting, our team looked at a number of design choices that would be important to support the specified weight load in our requirements [25]. These choices include stability, center of mass, speed, torque, and durability [26]. Given that there were no particular external design constraints imposed by the project outline, we explored each of the following lifting mechanisms with respect to the design choices listed above. In our particular case and application, our team was less concerned with degrees of freedom as our project scope doesn't include manipulators to pick up or delicately handle objects.

#### 3.2.3.2) Rotating Joints

For this design, the lifting mechanism moves in an arc, changing distance from the robot base and orientation of the objects relative to their environment as it moves on the arc. This is usually done by constructing two standard base pillars (stationary channel or metal to support the arm) with a gear (or two) set in between the two pillars. These gears would then be driven by a motor in order to create rotational motion.. From there, a bar would be attached to the gear(s) and extend out from there, acting as the arm. In this particular design, another manipulator on the end is featured, such as a claw or scoop, powered by another independent motor. In its simplest form, rotating joint lifting mechanisms have two degrees of freedom (unless more joints are introduced). The advantage is that such a design is relatively simple to build and test (when evaluating speed and torque), but as a drawback struggles to support heavier objects when lifting them in comparison to other lifting mechanism designs. This design is best suited for objects that need to be handled delicately or need to be picked up and placed down via the robot.

#### 3.2.3.3) Elevator

Less common but similar to rotating joints, elevators feature a manipulator on the end (claw, scoop, etc.) but move that manipulator in a vertical motion (versus angular). Driven by an independent motor, a small gear usually meshes with a meshed track, moving up and down when the teeth make contact with the track. A brace holds the assembly together, restricting any type of rotation on the track or shearing force. This design style is known as rack and pinion, but chain drive is also an option too (where a chain and sprocket set, either one or two sets, drives the manipulator up and down). The advantage with this design is that it is typically very stable and reliable if built correctly, where the speed of the lift is determined by the gear ratio and weight which the manipulator supports. The disadvantage is that if the rack and pinion are not perfectly meshed, sections of the track may be skipped causing the lift to break down. For chain drive, making sure that the chains are identical and that the sprockets are spaced evenly apart on each side (if driven by two independent motors) is necessary, otherwise binding will happen in the chains.

#### 3.2.3.4) Linkages/Bar Lift

Linkages consist of a series of rigid bodies (bars, metal, etc.) called links, connected by rotating joints. In this system, there is usually a fixed link and driven link. The fixed link is directly attached to a gear producing rotational motion (rotating joint), where another link connected below, the driven link, shifts as the gear rotates. This allows for input motion to be converted into different types of output motion, such as a rotating joint to elevator output. The simplest form of a linkage is a four-bar lift, but the height can be increased by making it a 6-bar or 8-bar lift. At the end of your linkage is typically a driven manipulator or fixed platform, depending on the application. These systems usually only require one motor to drive the input and another to drive the output, but the systems can be more complex depending on the design or application (e.g. reverse bar lift, typically requiring three independent motors).

#### 3.2.3.5) Scissor Lift

This design, being a specific type of linkage, allows the robot to efficiently reach up to higher areas. The system is built using a series of two c-channels attached together at the center, known as stages. The stages are attached at each end of each c-channel, stacked on top of each other. From here, a cross-section piece (shaped like an X), is mounted in the c-channel with one end being fixed in the channel. The other end is driven by a motor using a screw drive (or, in some instances, the fixed end is driven by a motor and gear train, causing the free side to move), lifting the stages apart. As this is done, the cross section collapses upon itself, lifting the free stage upwards. This advantage of this system is that it supports heavy loads efficiently, acting as a very stable lifting mechanism. A platform can be mounted on top of the scissor lift if objects are to be lifted on top of it, or other manipulators can be mounted if the goal is to grab hold of an object high up off the ground. The disadvantage is that the system has less rotational freedom (unless accompanied by an independent, rotating base), meaning that in order to grab with a manipulator attached to a scissor lift, rotating your robot is necessary.

### 3.2.4) Batteries

#### 3.2.4.1) Preliminary

Below is a list of various types of batteries we considered and did research on. Our team looked at a number of design choices that would be important to the overall operational time of the robot and ease of charging. These choices include battery life, speed of charging, size, type and volatility [27]. Given that there were no particular external design constraints imposed by the project outline, we explored each of the following batteries with respect to the design choices listed above. Given our project requirements, we were more concerned with battery life and speed of charging.

#### 3.2.4.2) Lead Acid

Typically good for systems that require a large capacity of power, where 12V lead acid batteries are commonly seen in projects. These batteries are usually safe if you avoid wet-cell and flooded batteries, instead using gel, AGM or any “non-spillable” styles. On the downside, lead acid batteries typically weigh considerably more, and only come in 6V or 12V increments. The smallest of these batteries are several amp-hours. These batteries also take a long time to charge, but are reliable as long as they are kept fully charged. Otherwise, these batteries will lose capacity over time if not full. Lead acid batteries can be found in small vehicles like scooters or golf carts, or any systems that require a large energy capacity, such as solar power systems or universal power supplies (UPS).

#### 3.2.4.3) Nickel Metal Hydride

Nickel Metal Hydride (NiMH) batteries are on the much cheaper side, and lighter than lead acid batteries. Chargers for these batteries are typically cheaper since they can be trickle-charged. NiMH batteries are also hard to completely lose the capacity on. These batteries do see voltage drops throughout discharge, and are typically harder to find as Lithium Polymer batteries have

mostly replaced them. They also don't work for as many charge cycles as lithium, and similarly to lead acid, loses charge sitting on a shelf. These batteries are seen in airsoft guns, RC cars, replacements for AA or AAA batteries and some cheaper power tools.

#### 3.2.4.4) Lithium-ion

Lithium-ion batteries require a battery management system in a pack in order for them to be relatively safe. The advantage is that these batteries are very lightweight and durable, with stable voltage when it drains. These batteries also have a high energy density, so the capacity for these smaller batteries is significantly better than its counterparts. On the other hand, these batteries can be significantly more expensive given how efficient they are, and requires a more expensive charger. Although they have a large capacity for their size, they would need to be set up in a pack in order to be used for larger robotics projects. Often they're seen in flashlights, low current electrical equipment or as large battery packs, powering smart cars to other smaller electric vehicles.

#### 3.2.4.5) Lithium Polymer

Lithium Polymer or LiPo packs are very lightweight in comparison to other battery types, with a simple design to them. They are cheap at smaller sizes and competitively priced at larger sizes, ranging from voltage of 3.7 V to 22.2 V, from 1 to 6 cells respectively. They can also be seen up to 44.4V, or 12 cells in series, for projects that require a lot of power like electric longboards or skateboards. These batteries are of the best for current capacity and energy density. The disadvantages are that these batteries can be dangerous if not handled properly, and requires a smart charger. These batteries equally have to be stored and maintained as specified. LiPo batteries can be seen in smaller robots, RC airplanes and drones. A single cell is seen for Arduino projects or mobile devices as well as modern laptops.

### 3.2.5) Sensors

#### 3.2.5.1) Preliminary

The following exploration of sensors below are all of the sensors we deemed pertinent to our project, given that their use is necessary to achieve the operations our team desires from MAC. As such, other unrelated sensors are not included here in this research section, but may be referenced later.

#### 3.2.5.2) LiDAR

Light Detection and Ranging, or LiDAR for short, emits a pulsed light wave into the surrounding environment, pulses of which bounce off surrounding objects and return to the sensor in the shape of those objects [28]. By tracking the time it takes for each pulse to return to the sensor, LiDAR can calculate the distance traveled and subsequently create a map of the environment. This process of pulsing is repeated millions of times per second, creating a precise, real-time 3-D map of the environment. This map can then be used with various software algorithms, keeping track of the robot's location and change in environment.

LiDAR is commonly used to make high-resolution maps, seen in surveying, geodesy, geomatics, archaeology, laser guidance, autonomous vehicles and more [29]. Light is typically reflected via backscattering, or the reflection of waves, particles or signals back to the direction from which they came. This is unlike a pure reflection one would see with mirrors. There are two different detection schemes, incoherent and coherent energy detection. Coherent schemes are usually more sensitive than incoherent (direct detection), allowing them to operate at a much lower power. Various lasers are used, where 600-1000 nm lasers are common for non-scientific applications and 1660 nm lasers are used for longer ranges with lower accuracy. Two main photodetector technologies are used in LiDAR, those being solid state photodetectors or photomultipliers.

#### 3.2.5.3) Ultrasonic

Ultrasonic sensors measure distance using ultrasonic waves, where the sensor head emits the wave and receives the wave reflected back from the target. The sensors themselves measure the distance to the target by measuring the time between the emission of the wave and the reception of it upon return [30]. In comparison to optical sensors, ultrasonic sensors use a single ultrasonic element for both emission and reception where a normal optical sensor has a separate transmitter and receiver. For a reflective model, a single oscillator emits and receives the waves alternately, allowing for the sensor head to be very small. Distance is calculated by the formula $D = ½ \times T \times C$, where D is the distance, T is the time it takes for the wave to return upon initially leaving, and C is the sonic speed [31].

Typical characteristics of ultrasonic sensors include the fact that the waves can be reflected off of a glass, liquid or transparent surface (allowing for detection), detection is not affected by dust or dirt, and targets such as mesh trays or springs can also be detected. Primarily used for proximity sensors, ultrasonic sensors are seen on automobiles and anti-collision systems. Relating to our project's use, they help detect obstacles. The advantage of these sensors is that they are not susceptible to interference from smoke, gas or other particles, but on the downside are affected by heat.

#### 3.2.5.4) Limit Switch

Limit switches, a type of contact or proximity sensor, is an electromechanical device that has an actuator and a set of contacts [32]. When the actuator is triggered upon contact with an object, the limit switch makes contact with the contacts, allowing the sensor to trigger something when the contacts make or break the electrical connection. These types of sensors are often used because of their ease of installation and reliability, determining the presence or absence of an object against the switch. Standard styles of limit switches include levers, roller plunger and whisker type. These switches can be mechanically controlled by the motion of a lever or be triggered by a magnet. Some examples include photocopiers, printers, microwave ovens (detecting if the door is opened or closed) and garage door openers.

#### 3.2.5.5) Encoder

Encoders at a base turns position into an electronic signal, the two forms being absolute and incremental encoders. Absolute encoders return an absolute position value where incremental encoders measure or count distance traveled rather than return the absolute position (which can be derived after the fact). Position can be measured based on linear or angular position, where linear encoders convert linear position and rotary encoders convert rotary (angular) position [33]. In the case of absolute encoders, position information is maintained when power is removed from the sensor, and immediately retrievable once power is restored. This enables a system to power off and upon restoring itself know which location it was previously in (given it hasn't moved from that position). In this case, multiple code rings with various binary weightings return a data word which represents the absolute position within one revolution [34]. Incremental encoders, on the other hand, immediately report changes in position, but do not report or keep track of absolute position. Due to this fact, the system may have to be moved to a fixed reference point to initialize the absolute position measurements.

#### 3.2.5.6) Gyroscope

Gyroscope sensors measure as well as maintain orientation and angular velocity. A spinning disc on a free axis of rotation orients itself in a standard, balanced position, its orientation remaining the same despite tilting or rotating. This disc can rotate freely in either two or three axes, the independent rings (the each ring, including the outer frame known as gimbals of which the spin axis and rotor/disc is attached to) allowing for this. Orientation of the gimbals relays feedback about how the rotor or disc is oriented in the x, y and z axes. Gyroscopes are used in telescopes, submarines, motor vehicles, compasses and overall assist in stability of systems where inertial guidance is necessary [35]. They are also used by various sensors, such as accelerometers. For example, smartphones use gyroscopes to determine the orientation of the phone, making use of that feedback to note when certain programmed gestures have been made by the user.

### 3.2.6) Remote Control

#### 3.2.6.1) Preliminary

The following research below focuses on various forms of remote control (RC) communications that are possible. Given our project outline, developing an RC device is necessary in order to both test MAC as well as manually control it if necessary. Several factors were considered while doing research on this section, such as range, interference and reliability.

#### 3.2.6.2) Wi-Fi

Wi-Fi is commonly used for local area networking of devices, allowing communication or exchange of data between nearby devices via radio waves. A receiver is usually necessary to capture the data sent from one wired or wireless device to another, the strength and reliability of which varies depending on the receiver used. Wi-Fi most commonly operates at 2.4 GHz and 5 GHz frequencies, the radio bands of which are subdivided into multiple channels [36]. Given that the wavebands of WiFi have relatively high absorption, it often works best with line-of-sight use.

Otherwise, signal strength is lost as the wavebands pass through solid obstacles such as walls and other objects. Access points known as hotspots often are used to extend coverage, commonly extending the range up to 20 meters ndoors. Transfer of data using Wi-Fi is relatively fast, with some hardware at close range supporting 1 Gbit/s transfer speeds.

#### 3.2.6.3) Bluetooth

Bluetooth operates wirelessly at a short range, exchanging data between fixed and mobile devices. Similar to Wi-Fi, this is also done using radio waves, operating from 2.402 GHz to 2.48GHz. Though operating similarly, Bluetooth establishes personal area networks (PANs) versus local area networks (LANs). Transmitted data is divided into packets, where each packet transmits on one of 70 designated Bluetooth channels, each channel having a bandwidth of 1MHz. Bluetooth uses packet-based protocols, featuring a master-slave architecture. One master may communicate with up to seven slaves at once. At any given time, data can be transferred between the master and one of the seven other devices, typically switching rapidly in a round-robin fashion [37]. One slave can be paired to multiple masters as well. Considering the range is very short, the reliability and connection between remote control and robot may be unstable. Interruptions could occur, or the connection could be easily lost if out of range depending on how the environment is laid out or what obstructions exist between the RC and paired device. In some cases, the range can be extended (in rare cases reaching up to 1 km), but usually devices have a range of around 10 meters or so. Bluetooth is intended to be used for portable equipment and applications, whereas Wi-Fi is designed to replace high-speed cabling for local area network access. Applications of bluetooth feature phones, speakers, robotics systems, laptops and other networked accessories.

#### 3.2.6.4) Radio Control

Radio Control (RC) makes use of control signals which are transmitted from radios in order to remotely control a device. Modern examples of such systems include garage door openers and keyless entry systems which make use of small radio transmitters [38]. Radio control is also used to control model RC cars, some vehicles and drones. Useful for teleoperation or machine to machine control, the receiver checks the radio signal sent by the transmitter for the correct frequency as well as confirms the security codes match. Verification then establishes a connection, and the receiver sends an instruction to an activated relay. In some cases, multiple relays are used to propagate the system or enable a wider range of control. 2.4 GHz RC control systems are now commonly used to control model vehicles and aircraft [39]. Lower frequency ranges (those in the MHz) support channels for model vehicles and radio stations that transmit various channels for music.

#### 3.2.6.5) Cellular

Cellular or mobile networks are set up so that the last link is wireless. The network is distributed over areas called “cells,” with at least one fixed-location transceiver, but more are normally seen. Base stations provide the cell with network coverage supporting transmission of voice, data and other types of content. Each cell typically uses a different set of frequencies from neighboring cells as to avoid interference [40]. Cellular networks offer more capacity than a single large

transmitter, less power drawn than with a single transmitter or satellite since cell towers are placed closer, and larger area coverage based on additional cell towers constructed.

### 3.2.7) MCU and Higher-level processing boards

#### 3.2.7.1) MSP430

Designed for use as low powered embedded devices, they can idle at an almost negligible amperage (1μA). Top speed for the CPU is 25MHz, with wake-up times from low-power modes below 1 microsecond. The processor family comes with peripherals such as internal oscillators, timers (PWM, watchdog, USART, SPI, $I^2C$, etc.), reset circuitry, comparators, DAC, and LCD driver, USB and more. All of the processors are programmable via JTAG, a built-in bootstrap loader (using UART, for most MSP430 families) or USB [41]. Given that the MSP430 does not have an external memory bus, on-chip memory is limited, making it not useful for applications that require large buffers or tables for data.

#### 3.2.7.2) Arduino Uno

In comparison to the MSP430, the Arduino Uno has large community support, making this an easier option in terms of overcoming challenges or seeking advice on usage. The Uno also has stackable shields, allowing for much greater feature expansion versus the MSP430 [42]. With 40mA source/sink per pin, the LEDs are much brighter than the MSP430 (~4mA). Where the MSP430 can only run up to 3.6V, the Uno can run at 3.3V or 5V. Additionally, there are 20 IO pins versus 16 on the MSP430. Considering our project's budget is quite high, the extra $15 for the Arduino isn't considerable. Concerning realistic downsides to the Arduino Uno, the MSP430 does have an onboard button, more flexible PWM controls, and features for a much longer battery life.

#### 3.2.7.3) Raspberry Pi

Stepping up from microcontrollers such as the MSP430 and Arduino Uno, Raspberry Pis have significantly more processing power as they come with 64-bit microprocessors (essentially a minicomputer, or SBC). On top of all the features that the MSP430 or Arduino Uno may come with, Pis have all the hardware a computer would have such as a processor, memory, storage, graphics drivers and on-board connectors. It also requires an operating system to run unlike microcontrollers, but comes with a pre-made OS (Raspberry Pi OS, a Linux-based operating system). It also has a significantly faster clock speed (1.2 GHz versus 16MHz seen on the Arduino Uno). Despite similarly being powered by a USB, the power draw is significantly greater for a Raspberry Pi. Generally, Raspberry Pis are better for multi-application use, especially applications that are more demanding and computationally expensive [43]. Despite the added power draw and complexity, Raspberry Pis have a lot more to give in terms of functionality.

#### 3.2.7.4) Jetson Nano

Jetson Nanos like Raspberry Pis are minicomputers, and by comparison will be evaluated to the most current iteration of the Raspberry Pi, Raspberry Pi 4. Additionally, the evaluation will be based on the 4 GB memory variant of the Jetson Nano. Both use quad-core 64-bit ARM processors, but the Nano runs at a lower clock frequency (1.42 GHz versus 1.5 GHz). The Nano has four USB 3.0 ports, unlike the Pi 4, with two 3.0 ports and two 2.0 ports [44]. The Nano also has one USB 2.0 micro port, an HDMI 2.0 port and a display port whereas the Pi 4 only has two micro HDMI ports. As a downside, the Nano only supports single display while the Pi 4 supports dual display. Both support Gigabit Ethernet connection. The Nano's official OS is Linux4Tegra, based on Ubuntu 18.04, designed to run NVIDIA hardware. The biggest distinction between the two boards is the GPU, where the Nano has a 128-core Maxwell GPU, which is significantly more powerful than the Raspberry Pi 4. This makes the Nano more useful for graphically-intensive uses such as AI or ML applications. Factoring in neural computer sticks at an extra cost, the Pi 4 can compete with the GPU capabilities of the Nano.

### 3.2.8) Button Matrix

The following section discusses a novel method of implementing multiple buttons that will communicate with a microcontroller.

MAC's remote is planning to use buttons that the user presses in order to tell MAC to do various tasks. Because of all the things that MAC will be capable of there may be the need for a large number of buttons. The easiest way to accomplish this is by simply having each button be assigned to a different GPIO pin on the remotes controller. The problem with this method is that there may not be enough GPIO pins required. So there are two options from this point, use a controller that has more GPIO pins, or find a way to reduce the number of necessary GPIO pins without reducing the number of buttons. The first option is undesirable because more GPIO pins means a different controller that is larger, more expensive, and often over qualified for the job that we are trying to accomplish which is just reading a button push. The second method, if it can be accomplished is more desirable, but how can this be done? The solution is to use what is called a button matrix. A button matrix blends hardware and software to allow for less GPIO pins needed than the number of buttons, as shown in the figure below [45].

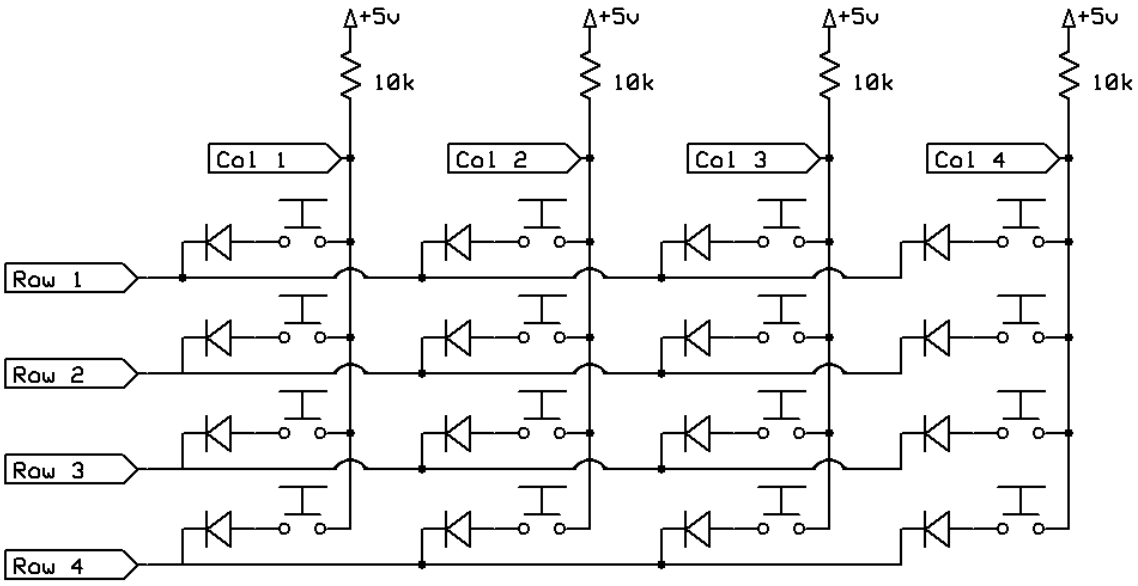



Figure 1: Basic Button Matrix

The idea behind a button matrix is that each GPIO pin is either connected to a row or a column as can be seen in the figure above. The column GPIO pins are configured as inputs and because of the pull resistors will read high when a button is not depressed. To test and see if a button is pushed software configures one row at a time to output low. Then the software scans to see the state of the column inputs, if any of the column pins read low then the button is being pushed. After the controller has scanned the first row the row is reconfigured as an input and then the process is repeated for the other rows. The reason we only have one row on at a time is so that no false positives occur as if more than one row is configured to output low any button press on a single column will cause the column GPIO to think a button is being pressed. It is important to note that this reconfiguring and scanning needs to happen in a timely manner because users like to send inputs in a timely manner.

The reason for the diodes is to prevent “ghost presses”. Without the presence of the diodes and if multiple buttons are pushed at the same time it is possible for the voltage of a column to unintentionally be brought low. This is undesirable and the possibility of this is eliminated by the inclusion of the diodes.

So, a button matrix is useful because it allows for C*R buttons to be used while only requiring a controller that needs C+R GPIO pins. Where C is the number of columns or the matrix and R is the number of rows in the matrix. As you can see as the number of buttons increases the amount of buttons that can be gained goes up.

## 3.3) Software Research

### 3.3.1) OS Versus RTOS (Real-time Operating Systems)

The Jetson Nano that will be used for both higher-level processing of sensor data and lower-level control such as controlling the motors via motor controllers will be running an Linux operating system. The OS will be responsible for installing any hardware drivers necessary for the WiFi module, USB to CAN adapters and any other other hardware that might be required to be connected to the Jetson Nano. The OS will be primarily used to run Python software that can capture sensor data, process it and use it to control the robots path and movement. Since there will be computer vision algorithms necessary for line-of-sight following from data captured from a mounted camera as well as SLAM mapping the Jetson Nano was chosen over a Raspberry Pi or lower-level microcontrollers. The need for an operating system in this case is clear as there will be several functionalities that the Jetson Nano will accomplish via the software that is running.

There is also the discussion of using a real-time operating system that is found in many embedded systems. An operating system such as Linux provides a non-deterministic and soft real-time response to external events caused by the user. There is no guarantee that a task or process running on the OS will finish before a specific deadline or time period. There are still scheduling algorithms present on an OS such as First In First Out (FIFO), priority-based algorithms, and round-robin based algorithms to name a few. Many of these scheduling algorithms are implemented in a real-time operating system (RTOS) as well. The key difference between an OS and RTOS is that an RTOS is capable of responding to external real-time events by providing a hard real-time response. This can be accomplished through various scheduling algorithms that schedule tasks in a real-time system in different ways and ensure that each task's deadline is met within a set.

A design decision that was made was to use a simple OS that can run on the Jetson Nano such as Linux Ubuntu without a RTOS due to the software requirements that were discussed. Though an RTOS would not be deficient or ineffective to implement or improve the design due to the time constraint and the current goals of the MAC we believed that an OS will be sufficient in running the software rather than having to split the software into multiple tasks that need to be assessed in terms of their priority, period, deadline and execution time (CPU time). This would only further complicate the already complicated software that is required to run computer vision algorithms and SLAM mapping. The MAC is not intended to respond critically to real-time events whether they are periodic or sporadic as it will be used for the purpose of simply carrying an object from one location to the next as specified by the user. Also the household environment the MAC will operate in though has its hazards will not pose an increased safety or security threat such as an outdoor environment would. In deciding to solely use an OS with a Jetson Nano for both higher-level processing and control, we recognize that there will need to be additional safety requirements that need to be implemented since there is no guarantee that the MAC will respond to a real-time event as fast as it would with a RTOS. Additional safety requirements include managing the speed of the MAC to be low especially when carrying a load, having many sensors that can collect data and steer the MAC away from any obstacles or objects that might

cause an injury and finally developing a robust software that can monitor the physical state of the MAC to alert the user if there is too much weight or if its tilting with a potential to fall over.

### 3.3.2) ROS

ROS stands for Robot Operating System that provides a framework or set of libraries that aid in the development of robotic applications. Since the initial release of ROS, there has been recent development of ROS2 which is meant to be a complete bottom-up improvement of the original ROS now called ROS1. A few group members have experience using ROS1 through robotic classes and projects thus understanding the limitations and down-sides of the ROS1 framework. This led to the exploration of ROS2 as the core library to be used instead of ROS1 for the robotic software [46]. This section attempts to give an overview of the differences between the two frameworks giving an insight also on the features included in ROS, how that affects our design decisions and finally which ROS distribution will be used for the MAC.

Initially, there are a few differences between how the ROS API is layered in ROS1 and ROS2. In general, ROS2 provides a core library written in C called **rcl** in which there are client libraries built on top in different languages such as Python [47]. This is different from ROS1 that had completely independent libraries for different languages such as C++ and Python. This feature to ROS2 makes it easier to integrate and use other client libraries in different languages as they essentially utilize the same core API. For us, this has an impact as in developing our mobile application as a stretch feature(explored later in the document) this will provide flexibility in making it easier to use a different programming language such as Java in the **rcljava** ROS2 client library with the **rclpython** ROS2 client library. In ROS, there is a core concept of a node that is basically just a process that performs some sort of computation. In ROS1 and ROS2, implementing a node differs as in ROS1 there is no structured method of writing a node while in ROS2 there is a convention defined where a class inherits from a **Node** object. The **Node** object then contains all the functionalities of ROS. This is another improvement that the ROS2 distribution made where there is a clearer way to implement nodes that results in better code in terms of modularity. Two other improvements in terms of the node concept in ROS2 is that there is the possibility of running multiple nodes in the same executable file and that there is the concept of lifecycled nodes. Regarding the former improvement, a node in ROS1 is bound to the executable and thus required the creation of “Nodelets” for intra-process communication. In ROS2, this became easier by introducing the concept of “components” which is a simplified version of a node that allows for execution in the same executable and thus makes interprocess communication easier to implement. The lifecycle node as mentioned was another feature that affected our design decision. A lifecycled node includes different states for a **Node** object where a developer can configure the **Node** before it being active in the system [48]. For example, we will have many sensors on-board the MAC and thus these sensors will most likely need some sort of configuration before the sensor node in ROS will be able to collect, manipulate and communicate data to other processes. Since we will have many sensors that will be relied on for navigation and motor control, the ease of intra-process communication and the introduction of a lifecycle node in ROS2 will allow for a faster development process as well as a robust software product.

There are also differences in communication interface between ROS1 and ROS2 that caused us to lean towards the decision of using ROS2 as the distribution for the software. In ROS1, there was a requirement of a “master” that initiated and handled communication between nodes. Without this “master” running there is no way nodes would be able to communicate with each other. This was relieved in ROS2 by having no centralized “master” ; instead each node is able to initiate and communicate with other nodes without the need for a “master”. This was an important feature as there is a possibility of having multiple machines in the future such as a mobile application that can be paired with the MAC. In this way, there is a distributed system possible where each of the nodes are independent requiring less set-up for the developers. A major feature of ROS2 compared to ROS1 is the fact that services are now asynchronous. Services in ROS are a part of a node that handles the request/reply interactions between a client that is requesting some piece of data or action. Since the MAC will be using a Jetson Nano as the central hub for all necessary processing, asynchronous services will reduce the likelihood of a failure and increase the responsiveness as the MAC will not be blocked from performing other processes while a service client is requesting data then waiting for a response from a service server. Actions are also a concept that has been made part of the ROS2 core that weren’t present in the ROS1 core. Actions are similar to services in that they are asynchronous and deal with communication but they also provide regular feedback while performing an “action” such as navigation and are also cancellable unlike services.

Finally, another important difference in terms of core features between ROS1 and ROS2 is the introduction of Quality of Service (QoS). QoS allows the developer to configure how nodes handle communication in terms of how often they receive messages, which messages they receive, message queues for nodes, etc. This is an awesome feature that gives us the possibility of having acute control over the nodes defined in software. Overall, the choice of ROS2 for the MAC software will offer increased flexibility as well as speed-up the development process since ROS2 offers a more complete core than ROS1.

### 3.3.3) SLAM Mapping

Simultaneous localization and mapping or SLAM involves creating and updating a map of an environment while also keeping track of the robot's location within the map. There are several different flavors or approximations for the SLAM algorithm that will be explored in the sections below such as Kalman Filters, RatSLAM, LiDAR, gmapping and others. In general, these methods attempt to compute the state of the robot and the map of the environment through probability [49]. There are two equations that are required to be solved or approximated in order to estimate the state of the robot and map of the environment. The equations are included below for reference. Before continuing, we will define some terminology that will be present in the upcoming sections. A term used frequently to describe the location and movement of a robot is the pose. The pose can tell the location and orientation of a robot in either 2D or 3D as well as the orientation of the robot [50]. SLAM is meant to create the map as mentioned before as well as update the pose of the robot within the map.

$$P(x_t|o_{1:t}, u_{1:t}, m_t) = \sum_{m_{t-1}} P(o_t|x_t, m_t, u_{1:t}) \sum_{x_{t-1}} P(x_t|x_{t-1}) P(x_{t-1}|m_t, o_{1:t-1}, u_{1:t})/Z$$

Figure 2: SLAM mapping approximation equation #1

$$P(m_t|x_t, o_{1:t}, u_{1:t}) = \sum_{x_t} \sum_{m_t} P(m_t|x_t, m_{t-1}, o_t, u_{1:t}) P(m_{t-1}, x_t|o_{1:t-1}, m_{t-1}, u_{1:t})$$

Figure 3: SLAM mapping approximation equation #2

#### 3.3.3.1) LiDAR-SLAM and Visual-SLAM

The two different types of approaches to algorithms that solve the SLAM problem are LiDAR-SLAM and Visual-SLAM. LiDAR-SLAM as the name suggests uses LiDAR as the main sensor for the collection of data that can be used to build out a map of the robots environment. LiDAR can detect the distance of objects from the robot and thus is able to construct a grid map of the robot's environment. For the methods that include the use of LiDAR as the primary sensor for data collection we have algorithms such as Kalman filters and gmapping that are Monte Carlo or probabilistic based approaches. There are also optimization based approaches such as KartoSLAM and HectorSLAM that use LiDAR. Visual-SLAM on the other hand is a more complex method of approximating SLAM algorithms as it uses images for input data such as the RatSLAM algorithm that is discussed below. Images contain more data than a laser scan from a LiDAR sensor does resulting in new challenges on extracting the important features and using those features to generate a map. The only Visual-SLAM algorithm we explored is the RatSLAM algorithm. For the generation of a household map where the MAC will operate in, we will using a LiDAR based approach due to the wide variety of accessible algorithms that use LiDAR that are also present in ROS as well as it is the best approach for building out grid-maps that are sufficient for navigating household environments [51].

#### 3.3.3.2) Filter-based (Kalman Filters)

A statistical filter-based approach to approximate the above equations is using Kalman Filters. Kalman filtering is an algorithm that makes use of several measurements taken over time and produces estimates of any unknown variables despite any statistical noise that might be present in the measurements [52]. This is done by using joint probability distributions over all variables for each moment of time in a two-step process. First, the Kalman filter generates estimates of the current variables as well as the uncertainty associated with each. Afterwards, once the next measurement is taken the estimates are updated using a weighted average that is dependent on the amount of certainty present in the measurements. Therefore, there is a prediction made based on previous estimates, the correction is calculated based on the previous prediction and the current measurement, and then the prediction is updated to reflect the correction [53]. This algorithm can run in real-time and thus in approximating the SLAM mapping problem be able to generate a map of an environment as well as updating the robots location in the map as the robot traverses through the location and taking measurements.

#### 3.3.3.3) RatSLAM (CNN SLAM)

RatSLAM is an appearance based approach to solving the SLAM system inspired by the hippocampus of the brain. According to OpenSLAM, RatSLAM uses visual scene matching, competitive attractor networks and a semi-metric topological map representation [54]. This allows for real-time SLAM by integrating odometric information and landmark sensing for generating a map of the environment. A quick overview of the architecture is shown below. [55]. For simplicity, the RatSLAM algorithm uses external vision sensing and internal sensing as inputs to establishing the local view and path integration. The local view and path integration are then injected into what they call pose cells that represent a competitive attractor network. This is a type of neural network that is meant to converge to a stable pattern across the different units[reference]. ROS does not include a package for implementing RatSLAM and thus in order to reduce the complexity of integrating a third-party library we will avoid using this approach for approximating the SLAM algorithm.

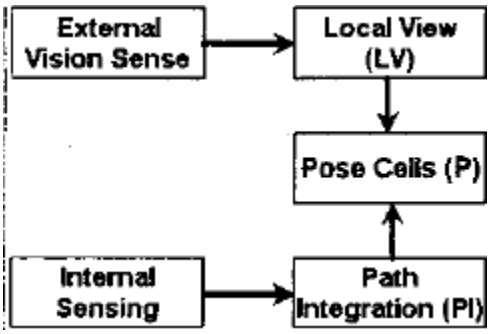



Figure 4: RatSLAM High-level architecture

#### 3.3.3.4) Gmapping

Gmapping is another algorithm based on particle filters that is able to learn grid-like maps from laser range data [56]. For reference, particle filters are similar to the Kalman filters described above in that it is a Monte Carlo approach updating predictions using statistics. Like Kalman Filters this can also be used in approximating the SLAM problem equations [57]. In Gmapping, each particle in the particle filter has an approximation for the map of the environment. In ROS, there is a wrapper for OpenSlam's Gmapping package where it can be represented as a ROS node. The mobile robot can collect laser data and pose data as an input to the **slam_gmapping** package that builds a 2-D grid map for the environment [58]. The localization of the robot is achieved through using an Adaptive Monte Carlo Localization algorithm that once again uses a particle filter to keep track of the position of the robot. The particles are updated by the odometry data measured from the sensors [59]. Since this method requires laser data the robot needs to be able to measure odometry data as well as be equipped with a laser range-finder such as a LiDAR sensor. Once this data is transformed appropriately by the algorithm it will generate an occupancy grid of the environment in relation to the position of the robot.

#### 3.3.3.5) RRT

Rapid-Exploring Random Trees or RRT is a probabilistic data structure that is meant to search high-dimensional spaces by building a space-filling tree. For reference, a space-filling tree is a geometric construction that is built through many iterations where each point in a space is connected to every other point in the space by a path of finite length [60]. In an RRT, we have a tree rooted at a starting point that is configured by taking random samples from the space we are searching. There is then a connection made between the sample taken and the closest state to the sample in the tree, if this connection is accurate and possible then a new state is added to the tree. Thus the tree grows and grows in each iteration. There is much more mathematically to this algorithm as there is for all other methods discussed briefly in these sections. These details will be left out as they are out-of-the-scope of the purpose for this research where the focus is more on implementation for our specific use case. In ROS, there is a package that implements a RRT-based map algorithm called **rrt_exporation** Like the other approaches the robot needs to have an onboard laser scanner or other type of sensor that is able to provide sensor data to the algorithm. This package contains 5 different ROS nodes that communicate with each other in order to update the map and command the robot. For simplification, there are nodes for global frontier detection and local frontier detection. These two nodes are simple as they take in the map as input and find frontier points that might be of interest to the robot. These points are then inputted to the filter node that filters out old, incorrect and redundant points. Once filtered the points are sent to the assigner node that commands the robot according to the frontier points received from the filter.

#### 3.3.3.6) Hector Slam

Hector SLAM is another approach to SLAM mapping that ROS contains a wrapper for to be used as a node in software. The algorithm does not require odometry and can be used on a robot that performs a roll or pitch like motion. This method is useful for the mapping of unstructured environments. The Hector SLAM method uses the high-update rate of LiDAR systems and provides 2D pose estimates at whatever the scan rate of the sensors is. In ROS, the package is called **hector_mapping** and requires a source of laser-scan data that can be provided by a LiDAR scanner [61]. In the ROS documentation it uses a Hokuyo UTM-30LX LiDAR sensor that is priced at $3500. This is far outside of the budget for the project and thus this method of SLAM approximation would not be possible to implement and seems to be overkill for the mapping of a household environment.

#### 3.3.3.7) KartoSLAM

KartoSLAM is a different approach to the probabilistic approaches that is used by Kalman filtering and particle filtering. KartoSLAM is an optimization based approach that uses a sparse pose adjustment for solving the problem of a matrix direct solution in non-linear optimization. Pose graphs are a set of robot poses that are connected by non-linear constraints that are collected from the observation of nearby features in the robots environment [62]. In KartoSLAM this optimization is done by using sparse pose adjustment that results in a better approximation of the SLAM problem thus a generation of a more accurate map and positional data for the robot. Again, like the other methods examined there is much more mathematical detail that can be

found in the sources referenced above. For the implementation of KartoSLAM, there is a ROS package called **slam_karto** that is a wrapper for the Karto mapping library [63]. In order to use **slam_karto** the robot needs to be equipped with sensors that provide odometry data as well as a horizontally-mounted laser range finder that can provide the algorithm with laser data. The **slam_karto** node then uses this data to generate an occupancy grid of the environment and define the pose of the robot similar to the **slam_gmapping** package discussed above.

### 3.3.4) Line of Sight (LOS) Following

Line of Sight (LOS) Following is a key feature that will be implemented where the MAC will be able to follow a user to a destination with the load it is carrying. There are several options that can be used to achieve this including an AI-based approach with computer vision, the usage of infrared beacons or using ultrasonic pulses. We concluded the best approach would be a camera/AI approach though these following sections outline the research done for each method and their viability in terms of implementation.

#### 3.3.4.1) Camera/AI-based

Computer vision is an AI field that continues to make progress and grow in popularity as the use cases for accurate computer vision increase. In CV applications a camera is often used that will feed images to software running a neural network model to be able to recognize and process the data from the image. Computer vision has recently been improved through the use of deep-learning neural networks more specifically convolutional neural networks that allow for applications of facial recognition and self-driving cars to be possible. In this approach, the computer vision will be powered by YOLO which is a state-of-the-art real-time object detection system that leverages a single deep-learning neural network to divide the image into several regions where there will then be bounding boxes as well as probabilities calculated for each region. An advantage of YOLO using only a single deep neural-network is that it makes much faster than other neural networks used for computer vision that require thousands of network evaluations [64]. With this approach, a model will need to be trained in order to be able to detect people for line of sight of following and inputted into the YOLO system for object detection so that the MAC can follow the correct person as well as increase the safety by preventing the MAC from following too close or hitting the person in front of them.

#### 3.3.4.2) Infrared Beacons

Infrared beacons are another approach that was discussed that requires more hardware and set-up in order to work accurately and correctly. Since GPS performs poorly inside which is where the MAC would operate the beacons would provide communication between the MAC and some sort of remote with an on-board beacon. This method would require both the remote and the robot to have an infrared beacon that contains infrared emitters shining in multiple directions. These infrared emitters are then received by IR receivers on the other beacon allowing for detection between two robots or in our use case detection between a remote held by the user and the MAC. Each beacon takes turns in transmitting and receiving so their infrared transmissions do not collide and cause reflections. A microcontroller is used to monitor the four receiving detectors in order to locate the direction of the transmitting beacon. The transmit and detect

cycles have a high frequency of about 1000 cycles per second or 1000 Hz [65]. The infrared beacons also have a range of 15 feet indoors which would satisfy the requirements for LOS following.

#### 3.3.4.3) Ultrasonic Pulse

The final approach considered for LOS following is in a similar vein to infrared beacons except through the use of ultrasonic pulses. This is an area that is still being heavily researched and there are several recent papers detailing different methods. In a more mathematical approach, there is an ultrasonic receiver placed on a robot with multiple ultrasonic beacons located in several different positions in a household. Through trilateration, the robot is able to calculate its position by calculating the distance between itself and two other receivers [66]. This requires even more additional hardware than the Infrared Beacons approach as there needs to be multiple ultrasonic sensors with additional hardware for clock synchronization between the ultrasonic sensors. A microcontroller is also needed in order to compute all the calculations for trilateration. Overall this method is too mathematically intensive and too complicated to implement with the time constraint. Also a computer vision based approach and infrared beacon approach both seem to be better alternatives with easier implementations.

### 3.3.5) Auto-following / Remote Return

An additional advanced feature that will act as a stretch goal is to include an auto-following or remote return feature for the MAC. An auto-following and remote return feature would allow for the MAC to automatically follow or go to the user based on other hardware, not just line of sight. As detailed in the section above, the line-of-sight approaches rely on the fact that the user is visible to the robot and thus it is able to detect the users position and follow them to the destination. Auto-following would improve this functionality by allowing the MAC to detect a user's location and follow them without the need to have the user visible. During research, there were three approaches to implement this feature and the following sections offer a brief explanation into the technology and viability of each.

#### 3.3.5.1) RFID

Radio-frequency identification(RFID) uses radio waves in order to detect the identity of an object and that is being extended to being able to also detect the position of the object. By placing multiple RFID tags in the household that act as transponders and a central reader called a locator node is able to receive these signals which allows for localization of the receiver. In this case, the MAC will be able to know the location of the transceivers in real-time [67]. This method is still one that is being currently developed and thus there are many research papers on the subject but would require too many external hardware components and complicated logistics software wise for the MAC to be able to triangulate the location of itself and the transceivers. Thus this method is not viable for our requirements.

#### 3.3.5.2) Wi-Fi Positioning

Wi-Fi positioning also known as the Wi-Fi positioning system (WPS) is a system that uses other Wi-Fi hotspots and access points in order to locate the intended device. In this case there are multiple different approaches that can be taken in order to determine the position of a client device with the respect of the position of access points being received signal strength indication (RSSI), fingerprinting, angle of arrival (AoA) and time of flight (ToF). In this case, the client device can be a docking station for the MAC where the MAC will need to know the position of the dock through Wi-Fi in order to traverse to its location. Also the client devices can be other access points or devices such as phones that the MAC can locate through Wi-Fi to traverse their location. Starting with a signal strength based approach, the RSSI localization method involves measuring the signal strength from a client device to several access points that can help determine the distance between the devices and through trilateration estimate the location of the client device [68]. This method is easy to implement compared to the other methodologies but there are accuracy and reliability concerns as signals strengths can be volatile. Another viable methodology is an AoA based technique that uses intensive math to apply triangulation on the multipath signals received at antennas in the access points. The MUSIC algorithm is a popular and commonly used algorithm for the computation of AoA. The final approach is a ToF based method that utilizes the timestamps taken at the wireless devices in order to calculate the time it takes for these signals to arrive from one place to another. This information then can be used to estimate the distance and therefore the position of a client device in respect to the access points. This provides similar drawbacks to the RSSI localization in terms of reliability and accuracy as these measurements can introduce large errors. ToF is harder to implement than RSSI even though it may be more accurate due to the clock synchronization difficulties and noise.

### 3.3.6) Voice Control

Voice control can provide huge accessibility benefits to the MAC. Voice control can be used to run various commands for the MAC via devices the user already may have, whether it be an Amazon Echo, their smartphone's Google Assistant, or a Google Home device. These devices can be accessed from fairly long distances or from the user carrying their phone, which can provide utility instead of needing to carry a remote or other device to operate the MAC. Currently, Amazon Alex and the Google Assistant are the two most viable options since they have options for external devices to run voice commands through.

Both of these options provide Python libraries to access their API's to send audio to, and then get a result from what was spoken by the user in the audio. Using this we can add functionality into our code to run a command based on what the user said. It is also pretty easy to include context and have different phrases respond the same way (like what is the weather today vs how is the weather right now given the same responses).

#### 3.3.6.1) Google Assistant

The Google Assistant provides many functionalities to the user in their day to day life, and through the Python library we can add new functionalities for the user to add and control their MAC with ease [69]. The library allows you to add commands that when the Google Assistant

recognizes certain keywords, to then send the command to the MAC for use. This can be used to start our functionalities like LOS following, remote call, docking, and even lifting. These are done through what are called services. These services are modular and can be added and removed when needed. It is easy to set up and free to use the API for small scale projects like this one, and since it integrates as an input it would require little extra programming to be able to use a feature like this.

#### 3.3.6.2) Amazon Alexa

The Alex Python API works quite similarly to the Google ASsistant. It provides functionalities by creating what are called Skills that provide new functionality, based on key phrases asked to the Alex assistant [70]. Since Alexa is very popular with the integrating into Amazon Echos and the Amazon firestick, it can be used to control the MAC from nearly anywhere in a home, without needing to remember buttons or other more complex control mechanisms. Since nearly 70% of users who have some type of home assistant device use a variation of the Amazon Echo, we can reach more users in our target audience with a feature like this if we support Amazon Alexa vs Google Assistant [71]. However, since both systems are fairly easy to integrate, given time constraints both could be implemented.

# 4) Related Standards and Realistic Design Constraints

Standards and design constraints are meant to guide the building as well as manufacturing of a product. This section intends to address any standards that are applicable to the design and building of the MAC as well as the design constraints present.

## 4.1) Related Standards

There are a few hardware and software standards that should be followed when designing and building the MAC. As an overview, the following subsections will address enclosure standards through the International Protection Code classifications, existing IEEE robotic standards and software standards such as PEP8 for Python code.

### 4.1.1) IEC Standard 60529: International Protection Code

The *International Electrotechnical Commission (IEC) Standard 60529* contains ingress protection (IP) ratings that numerically represents an enclosure's resistance to penetration of dust and liquids. The use of this standard was essential in our decision of the type of enclosure that is required to house the electronics that will power the MAC. The rating of the enclosure follows the following standard. The first number indicates the level of resistance to dust and the second number indicates the level of resistance to water and other liquids. For our enclosure, we are seeking to use a IP54 rated enclosure therefore our electronics will be dust protected so that dust will not interfere with any operation and the electronics will be protected from the splashing of water. We chose the following rating to follow as since the MAC will be made to operate in an everyday household, it is prone to any dust that may be present in the environment. Also on the

same note, any splashing of water is possible in a household since spills happen frequently. The IP rating of 4 for water protection was specifically chosen as the MAC will not be an environment where it will be submerged in water nor be in an environment where water can be projected at it powerfully. Therefore based on how the MAC will be designed and the environment it will operate in, we concluded that the IP rating of 54 will offer an enclosure that can protect the electronics and thus elongate the lifetime of the MAC.

*IEC Standard 60529* also outlines the testing for the numerical value pertaining to water protection. The enclosure should be tested with an oscillating spray for a minimum of 10 minutes where there should be a limited amount of protrusion and no harmful effect on the electronics.

### 4.1.2) ISO 13482: Safety requirements for personal care robots

Standards for robotics are still in progress as the technology has continued to progress exponentially over the last few years. The area of robotics is also vast and thus currently there are different sets of standards for the different types of robotics. The *ISO 13482* standard specifies requirements and safety guidelines for personal care robots which include mobile-servant robots, physical-assistant robots and person carrier robots. This standard was not available publicly for free-of charge therefore we used an IEEE paper that covered and evaluated the contents of the document. The MAC qualifies as a mobile-servant robot under the standard as it is a mobile standard intended to complete fetch and carry tasks in household environments [72]. The standard presents a 3-step methodology for risk-reduction for various hazards including mechanically-based hazards, hazards related to incorrect autonomous decision making as well as hazards of human interaction with the robot [72]. The steps for risk reduction are presented below.

1. Adopt an inherently safe design through mechanical adjustments such as non-sharp corners, and improving stability while still and moving.
2. Reduce risk by taking safeguards and protective measures using sensors such as LiDAR, cameras, ultrasonic, weight sensors, etc where the robot will stop within a close range of a human or other objects.
3. Requirements for documenting residual risk in extreme cases where there is no other option for risk mitigation. This will not apply to the design and implementation of the MAC as we will be able to mitigate any risks using the steps above.

*ISO 13482* outlines the safety measures of intended contact between the human and robot. This primarily deals with physical assistance robots or person carrier robots that enact a force on the human. The MAC is not classified as one of these and thus these safety guidelines are not directly applicable to our product.

### 4.1.3) IEEE 830 Software Requirements Specification

The *IEEE 830 Recommended Practice for Software Requirements Specifications* describes proven approaches for specifying software requirements when building a piece of software or

software product. The MAC will utilize complex software that aims to control the robotic hardware components and make the robot autonomous through the data collected by the various sensors that will be onboard the MAC. In order to effectively strategize the building of the software, a software requirements specification of some sort will be helpful. The need for a SRS becomes even more apparent when there is a team working on one piece of software that contains many components. The *IEEE 830* standard will help guide us in building out the requirements for the software so during the engineering process there is a reference to what needs to be accomplished. Not only does this allow the team members to be on the same page in terms of what is required for the software but also paves the way to develop a roadmap for the software. The document contains many sections on creating and using an effective SRS but for the building of the MAC we will focus on what the document defines as good characteristics of the SRS.

The *IEEE 830* standards lists several characteristics of a good SRS with elaboration that will be helpful for creating the requirements. The list is included below with a brief description as per the *IEEE 830* [73].

A. Correct: Each requirement included should be met by the software
B. Unambiguous: Each requirement has a clear meaning and a single interpretation
C. Complete: All requirements relating to functionality, performance, design constraints, attributes or interfaces. The responses of the system to both valid and invalid input data should be defined.
D. Consistent: Requirements should be consistent with other pieces of documentation
E. Ranked for importance and/or stability: Each requirement should be ranked or identified in terms of the importance
F. Verifiable: Each requirement should be able to be verified through testing of the software
G. Modifiable: The requirements should be able to be modified while keeping the consistency and structure of the document.
H. Traceable: The origin of each requirement is clear as well that it meets backward traceability and/or forward traceability. The software requirements for the MAC will be oriented to meet backward traceability so that there is a clear breadcrumb trail throughout the documentation.

The characteristics defined above will help in creation of each software requirement so that we are able to have a SRS that correctly and clearly defines what the software of the MAC is required to meet. Additionally, by designing the SRS with the mentioned characteristics in mind the testing of the MAC will prove to be easier as there is the potential of creating test cases related to each software requirement. Thus the SRS will not only be helpful in establishing the goals and scope of what is required by the software during the designing phase but also help guide the creation of test cases during the testing phase.

### 4.1.4) PEP8 Python Standard

*PEP8* is the widely accepted and official style guide for coding conventions in Python. Python will be the primary programming language used in engineering the software for the MAC. As multiple persons will be working on the code for the project the *PEP8* standard will be used to guide basic coding conventions to keep code organized, consistent and clean. This improves the quality of code making it easier to debug, improve and make general changes. The usage of *PEP8* will also help in keeping the software readable for third-party readers.

### 4.1.5) ISM Radio Band

Put simply, ISM radio bands are portions of the radio wave spectrum reserved internationally for purposes other than telecommunications, those areas being industrial, scientific and medical (ISM). Emissions from devices created in these fields can cause electromagnetic interference, leading to disruptions in communication when operated at the same frequency, so bands of frequencies are set up as a standard to limit this type of interference. Despite being a standard, there is no regulatory protection from ISM device operation, meaning that communication equipment must handle this type of interference. Many short-range, low power communication devices such as cordless phones, bluetooth devices, NFC, garage openers, and WiFI make use of the same ISM frequencies as their low power transmitters are not considered to be ISM devices and limit the range of the interference.

ISM bands are defined by the ITU Radio Regulations, and allow for unlicensed operations using these bands since communication devices must tolerate interference while using them. Due to the high likelihood of interference, licensed use of ISM bands is quite low. In the US, the FCC (Federal Communications Commission) governs the use of these bands and contains rules for unlicensed communication devices. Allocation of various ISM bands is provided by the ITU Radio Regulations and managed by national administrations, those allocations being primary, secondary, exclusive or shared. Concerning our applications for this project, the focus of this standard is on 802.11 wireless 2.4 GHz and 5.7 GHz ISM bands. The 2.4 GHz band is also used in Bluetooth communications, where the Bluetooth protocol divides the bands into 80 channels, each 1 MHz wide; channel 0-79). To guarantee no interference, Wi-Fi protocol requires (for the 2.4 GHz ISM band) 16.25-22 MHz channel separation, with the remaining 2 MHz gap used as a guard band [74].

## 4.2) Design Constraints

During the designing and development of the MAC there will be several design constraints that will need to be taken in consideration. The main design constraints that will impact our design will be the social, health and safety constraints since the MAC will be a household robot intended to offer a personal service geared towards the elderly and disabled population. This introduces several constraints that will dictate the direction of our hardware and software design. The following sections attempt to elaborate on the different types of constraints present and how that will affect the design or operation of the robot.

### 4.2.1) Environmental

There are several environmental constraints that are present in the type of location the MAC will operate that affects how it should be designed in terms of the chassis, electronic components, enclosure, and software. There are several hardware design constraints presented by the household environment. The dimensions of the MAC should be small enough to be able to fit in generic household doorways therefore our chassis with all components shall not exceed a 2.5 foot by 3 foot by 3 foot design as the standard for household doorways is approximately 32 inches in width, though they often can exceed this dimension. There are also several hazards in a household that the MAC will need to navigate through software and be able to withstand hardware wise. With the several sensors on-board the MAC the software will be able to develop a SLAM map of the household that will aid in navigation as well as be able to navigate away from any unexpected obstacles it might encounter on its path. As mentioned in the standards section above, the electronics will be housed in an IP54 rated enclosure in order to protect against any dust or water ingress. The decision for the IP54 rating was heavily based on the type -of dust and water hazards we expect operating in a household environment. Furthermore, in a household environment there was the design constraint of how fast the MAC should operate at as well as the durability of the MAC in terms of any drops or knocks it may take. The MAC will be limited to a speed of a max speed of 3 feet/second to ensure the safety of the user. Further details on speed are specified in *Table. 2 Requirements Specification*. Finally, the MAC will be designed to handle any drops of heavy objects up to 150 pounds on its chassis as there could be potential for a user to drop a heavy item on the robot or even a piece of furniture falling on the chassis if it is knocked over.

### 4.2.2) Ethical, Social and Political

As for any project there are ethical constraints that should be of the utmost priority that will be considered at every design step and during the implementation. We will be adhering to the *IEEE Code of Ethics* that will hold the team accountable in promoting ethical design that will not compromise the health and well-being of the customer as well as not break any code that deals with treating other team members fairly and with respect [75].

The MAC aims to improve the daily lives of the elderly and people with disabilities by offering a solution for carrying heavy objects in the household to reduce the chance of an injury or fall. With this in mind, the MAC will be designed with safety and ease of use at the forefront of design requirements. The use of sensors such as ultrasonic sensors and LiDAR as well safety precautions such as a limit switch for the scissor lift mechanism will be key in fulfilling the safety requirement for the design. Furthermore, the mechanical design of the MAC needs to make the loading, unloading and carrying of objects in the household safe for the user. With this in mind, there will be guard rails on the platform and grip tape to hold the objects in place without sliding off the MAC. This will prevent any object from falling off the MAC potentially damaging the payload or worse causing some sort of injury to the user. Overall, we want the MAC to be socially accepted as a robot built for the safety of the user that will actually make the household safer for the user. The MAC should also be easy-to-use since it is being geared to aid the elderly population and people with various disabilities. These populations may not have the

same experience in the use of technology and therefore our design choices will have to take this into consideration. To fulfill this requirement, there will be a simple remote used to command the MAC to follow you, to lower/lift the scissor lift and command the MAC to stop manually. A remote with a simple three buttons will be easy to use for anyone and not cause any confusion on how to command the MAC. Though, in order to implement more advanced features such as auto-following or remote return a potential smartphone app with a map of the robot's environment would be necessary. This is a nice-to-have feature but if implemented will follow the design constraint of the app being easy to use for diverse populations. Politically, there are no major design constraints that need to be taken into account as of now. Potentially, if the MAC is further advanced to handle larger objects in more difficult environments such as a military base then there would be several political design constraints that should be taken into consideration regarding communication and data colleciton.

### 4.2.3) Economic and Time

The economic and time constraints for this project are arguably the ones that will have the greatest impact on our design decisions in both hardware and software. Economically, there are two major factors that were considered. One, that the MAC shall be affordable to the average middle-class consumer therefore the mechanical and hardware parts should be chosen carefully in order to meet this requirement. The MAC will if sold to a consumer be valued at a price of $1000 which will make it affordable for middle-class families. Two, there is the economic constraint of keeping the project affordable for the designers and engineers due to the fact of being students with limited monetary resources therefore a total approximate budget of $1625 was calculated that includes *Must Have* and *Nice to Have* requirements. In accordance with these constraints, there will be an attempt to reuse any mechanical or robotic parts from previous projects as well as choosing appropriate sensors and batteries that fulfill the technological requirements but are also within the budget. Since there are a plethora of sensors and batteries at various price ranges we will be focusing on purchasing cheaper alternatives such as $150 LiDAR sensor over LiDAR sensors that venture into hundreds even thousands of dollars. Cameras, ultrasonic, and weight sensors will also be carefully chosen as they can meet the technological requirements while being within the budget set. The economic constraint present here is that we are unable to choose the most advanced battery or sensor technology to build our product rather we have to make careful decisions to select components that are within the budget and are functional technologically.

There is also a strict time constraint on the project due to the imposed deadlines that are required to be met. According to the project milestones mapped out, research and design is required to be finalized by April 1, 2021. Once that milestone is met, there will be a roadmap developed for prototyping, testing and finalizing of the project. For an approximation, the project should be finalized by the end of July 2021. This time constraint was a key design limitation when choosing the must-have features of our product. With unlimited time as with unlimited economic resources we would be able to build a robot with a plethora of features but this is not the reality and thus we have to make tradeoffs of the features the MAC needs to have in order to meet our project goals. For example, due to the time constraint imposed a remote return to dock feature

and teleoperated mode were both shifted to *Nice to Have* features and instead the features that will be focused on first will be SLAM mapping using LiDAR and the ability for the MAC to follow a user in its line-of-sight. Another important decision on the design that was affected by the time constraint was the decision on which sensors are key to the functionality that need to be implemented. Ideally, we would like to implement both weight and tilt sensors in order to protect the MAC from tipping over or for placing a weight exceeding the limitations of 100 pounds. These were moved to *Nice to Have* requirements as due to the time the implementation and processing of sensors such as ultrasonic, LiDAR and a camera were more important to the functionality of the robot. We expect to make further lower-level decisions in terms of which parts of the software can be implemented in the time constraint imposed.

### 4.2.4) Health and Safety

The health and safety of the MAC as mentioned in the Environmental and Social constraints section is essential through every design aspect in both hardware and software. Taking into consideration the social constraints mentioned in terms of the required safety the MAC must adhere to for the targeted population and the environmental safety concerns posed by operating a home environment there are general safety requirements that need to be considered during design. In selection of electronic parts and materials, there will be a focus on selecting Restrictive of Hazardous Substances (RoHS) compliant components [76]. RoHS restricts the use of certain hazardous and damaging substances in electronic and electrical components. RoHS gives restrictions on the amount allowed to use several substances such as Cadmium, Lead, Mercury as well as others. This will be important in minimizing the environmental footprint thus improving the healthy and safety of the product.

### 4.2.5) Manufacturability

The manufacturability of a product specifies its ability to be mass-produced with the resources available. The manufacturability of the MAC is limited by the resources and materials that can be acquired that are within the budget. Overall, the product will be designed with the goal of having the manufacturing done completely in house. The major manufacturing constraints that come into play are the building of the chassis and drive-train. Due to the mechanical nature of these tasks there will be a greater focus on the electrical hardware and software therefore we aim to build the chassis from used material such as aluminum. The scissor lift mechanism that will be built into the MAC will follow a similar principle and use a combination of aluminum and steel for fabrication. The chassis and scissor lift are expected to be the toughest pieces of manufacturing to deal with and we plan on using any shops in and around the UCF area that will provide us with a space to work in and tools to work with. Robotic chassis including the drivetrain will be salvaged from previous dismantled robotic projects and reused to save costs. There will still be the need to acquire materials to build the rest of the chassis and of course the integration of the electrical components such as sensors and batteries. These electrical components such as the battery, sensors, and PCB boards will be ordered from several common third-party vendors that manufacture these components as a part of their business. As mentioned in the economic constraint section, these parts will be chosen carefully so that they meet our budget constraints and with several vendors available, we are confident that we can select the

components that meet the technological requirements, economic requirements and manufacturing requirements. Finally, these components need to meet an overall manufacturability constraint during the integration of the chassis, hardware and software of the robot. The sensors, battery PCB boards and scissor lift will be required to fit within the chassis of the robot providing a design challenge and with the manufacturing constraint of limited materials, size constraints, etc there will be an emphasis on 3D modeling to validate the design before manufacturing begins.

#### 4.2.6) Sustainability

Sustainability is a major design constraint in every single engineering project as the product that is being built needs to be able to operate well within the normal operating conditions intended for the product. As detailed in the environmental constraints, the MAC will primarily operate in a household environment that even though does not provide issues with unpredictable outdoor weather conditions, there are still unpredictable hazards such as spills, and drops that need to be taken into consideration of the design. This design constraint will be achieved through a IP54 rated enclosure for the electronics to protect against dust and splashing of water. Also the chassis and scissor lift will be built out of durable materials such as steel and aluminum that can withstand heavy drops of objects as well as knocks so that the chassis of the robot is not damaged during operation.

In addition, there is a sustainability constraint on being able to conduct maintenance on the MAC that will be especially helpful in prototyping and testing. During the integration of the electrical hardware and mechanical components, there will be a design goal to be able to easily access these wiring and hardware through the chassis for any tweaks or maintenance. Apart from the hardware sustainability constraints, there are also software sustainability constraints that need to be met in order for seamless integration, ease of testing and future maintenance that might be done. Software will be kept sustainable by using a code repository such as Github with version control with several branches such as a main, develop and then the feature branches. The main branch will contain the working code with the develop branch containing the current features that are being developed. Also through pull-requests and code reviews we will be able to validate that the code is being maintained and consistent thus being able to be sustainable for a longer period of time.

# 5) Project Hardware and Software Design Details

## 5.1) Hardware Design Overview

This section discusses the electrical and mechanical hardware design design details of MAC. The hardware is split up into several sections that will be discussed individually. The following diagram shows how all of the components are connected to each other.

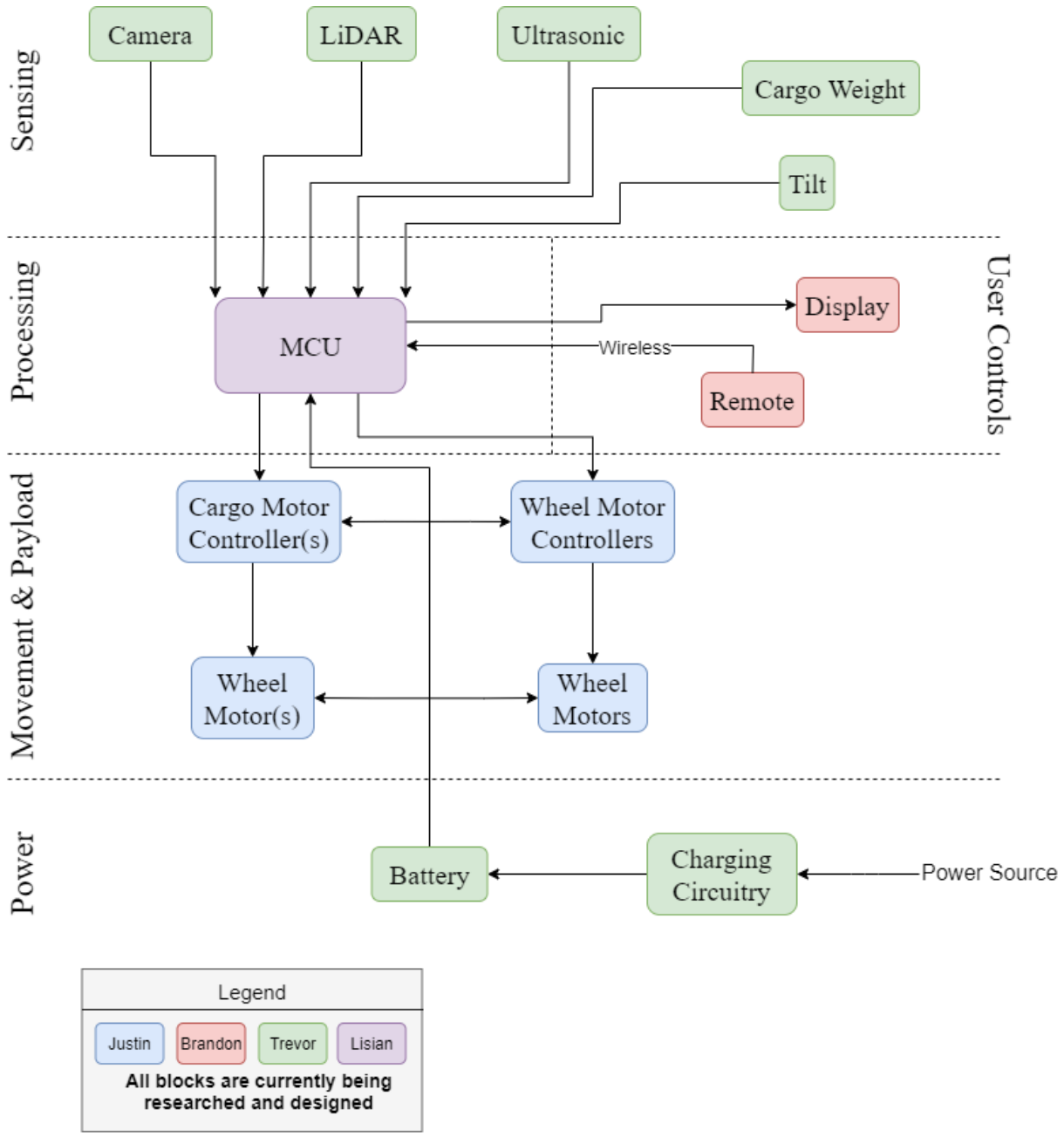



Figure 5: Hardware Block Diagram

The hardware design draws inspiration from commercial robots that move objects around warehouses and manufacturing plants, and applies relevant features to home use. MAC's size is limited by the average home environment as to properly navigate the home MAC must be able to travel through doorways, hallways, around furniture, etc. With this in mind MAC's footprint will be similar to a folding shopping cart, less than or equal to 2.5 feet wide and less than or equal to 3 feet long. MAC also needs to raise and lower cargo to an appropriate height for individuals that

do not have the strength to lift the cargo themselves. The relevant height for most individuals in their daily life is table height and/or countertop height. With this in mind MAC's load floor should be able to lift cargo up to 3 feet.

Movement is important in accomplishing MAC's purpose. MAC therefore needs a chassis that is strong, reliable, and inexpensive. A strong chassis is important because when loaded with cargo we do not want stress in the chassis to inhibit MAC's movement, or we face the risk of unreliable movement and possible dangerous situations if MAC is loaded to full capacity. Additionally, our primary focus of this project is to develop the electronic controls, not spend unnecessary resources developing the mechanical aspects of MAC. So we decided to use a pre-built robot chassis drivetrain, this allows us to focus on the electrical aspects of our system and perhaps most importantly our connections allowed us to acquire one for very low cost.

MAC will be driven using conventional wheels in a tank drive configuration. The reason for wheels is that wheels are cheap, easy to acquire and incorporate into the chassis when compared to other styles like omni-wheels or tracks, additionally unconventional wheels can easily damage flooring often found in household environments. The reason for tank drive is with tank drive the need for complicated steering mechanisms is completely eliminated because all possible movements can be accomplished by just spinning the wheels on each side the appropriate direction.

Also related to movement is how the wheels (and the rest of the onboard electronics) will be powered. MAC needs to travel around a house uninterrupted and free of any umbilicals used for either power or communications. MAC should also be quiet as to not disturb users. With all of these considerations in mind we decided that MAC will be battery powered and electric motors will handle all movement needs. Batteries exist that store enough energy to meet our runtime and load requirements and  can easily be mounted to the chassis. Additionally, batteries, with the proper circuitry, can easily be charged using mains power available in homes. Electric motors were selected because they can meet our torque requirement needs all while being small and lightweight, allowing for flexible packaging.

Being, battery powered MAC needs some way to recharge after use. Batteries are DC devices and almost all homes today have a reliable source of 120VAC 60Hz power that is also cheap to utilize. This AC power is the perfect candidate to recharge MAC, the only problem is it needs to be converted from AC to an acceptable DC value (depending on our battery configuration). From here we determined that a rectifier charging circuit would be required. Rectifier charging is the best decision for our goals because they are commonly used in all different types of devices and are therefore readily available and cheap to acquire.

In order for MAC to be recharged there must be some way to connect to mains power in the users home as previously discussed. MAC could have a standard US Type A outlet connector that the user connects to a wall output everytime that MAC needs to recharge. MAC could also have a dock that when the battery voltage drops to low MAC goes to automatically and connects with to recharge, from there several different charging methods could be pursued. One of MAC's

goals is to be autonomous so requiring the user to plug/unplug MAC every time the battery is low is not preferred.

Several different types of electric motors can be used for driving the wheels and the lifting mechanism. MAC required electric motors that provide enough torque to traverse the home environment and lift/lower cargo at a reasonable speed (we are not building a racecar). The purpose of the electric motors is straightforward so it reasons to have straightforward motors that are simple and cheap, as opposed to motors that require complex control such as brushless motors. With all of this in mind we decided to use brushed DC motors, these motors will be controlled using off the shelf DC-DC motor controllers as they are low cost and simple to implement.

In order to raise and lower cargo several types of mechanisms were considered. We desired a mechanism that is inexpensive and simple to manufacture, and will maximize the load floor height delta, that is the vertical distance of the load floor in its highest position to the load floor in its lowest position. After extensive research into different types of lifting mechanisms we determined that a scissor lift style mechanism would best accomplish our goals.

Lifting the scissor mechanism will be done with an electric motor similar to the ones used to drive the wheels as previously discussed. Attached to the scissor lift mechanism there must be a way to determine that the scissor lift has reached its maximum height. This is important because if MAC has no way of determining that max height has been achieved MAC will continue to drive the motor pushing the scissor lift past its mechanical capabilities and/or damaging the lift motor. In order to prevent this we determined that the simple solution of placing a limit switch on the scissor lift. Once the scissor lift reaches max travel the switch will be tripped and MAC's processor will know that max height has been achieved (the details of this will be discussed in the software design section). Limit switch was chosen because of its simple and cheap implementation when compared to other height determining schemes like counting lift motor revolutions or a linear/rotary potentiometer connected to the scissor arms.

MAC's processor will be responsible for managing multiple data streams while simultaneously making decisions and acting on those decisions by controlling power to the drivetrain motors. This will require a substantial amount of process power as a continuous stream of LiDAR and camera data will be present in addition to the decision making algorithm used for navigation. Furthermore, the processor will need to communicate with motor controllers for movement and lift operation. With all these considerations in mind several processors were considered both as single units and as a combination of multiple, and we determined that a single Nvidia Jetson Nano would meet our goals.

Users need a way to communicate with MAC in order to tell MAC what to do (for example: follow, stay, lift cargo, lower cargo, etc.). There are multiple ways to accomplish this goal. The user should be able to easily and intuitively communicate with MAC. The simplest way would be controls on MAC itself, this requires little hardware and software development as only a few buttons would need to be connected to a MAC's movement processor. Another method that we

considered was a smartphone app that wirelessly communicates to MAC through some protocol that is commonly found on smartphones. The final method we considered is a dedicated wireless remote, it would have a similar interface as on board controls just in a handheld package. After much consideration we determined that a dedicated wireless remote best fit our goals when compared to the other options for several reasons. A wireless remote is better than a onboard controls because it allows the user to operate MAC from a distance and removes any problems with the controls being hard to access because of the direction that MAC positioned itself, additionally onboard controls can possibly be unsafe if a user attempts to use them while MAC is moving. A wireless remote is better than a phone app because phone apps require more development to ensure that they work with multiple phone operating systems and the wireless communication scheme is limited to the hardware that the phone has onboard. Furthermore, and perhaps the most important reason for not using a phone app is that MAC's main user will be of the older demographic and a remote with buttons is easier to and more intuitive to operate than a smartphone app.

There are several wireless communication schemes that can be used for the remote communication between MAC and a user. The communication needs to be reliable and require little processing time. Additionally, the communication system should not require much power as users do not want to constantly have to charge their remote. Furthermore, a power hungry remote requires a bigger, heavier, more expensive battery to get similar runtime performances. With all of these considerations in mind we determined that Wifi communication would best accomplish our goals.

As MAC navigates from point A to point B MAC needs a way to scan the environment. The scanning must be done actively because as MAC moves the relative position of surrounding obstacles changes and we do not want any collisions to occur; this is especially important when MAC is carrying cargo. Furthermore, MAC must be able to scan all directions, in the event that reverse is required to best navigate to a destination MAC must know if it is safe to reverse. With all of these considerations we determined that a combination of a single 360° LiDAR and multiple ultrasonic sensors will suffice. A 360° LiDAR is a LiDAR that has all the sensing components built into a rotating assembly that spins with a specific frequency, a laser beam is sent out while the assembly is spinning and the data can be used to construct a 2D map of the immediate environment. A 360° LiDAR is more simple than using multiple sensors that look in different directions and then try to stitch all the information together in software. The reason for the ultrasonic sensors is that, although the LiDAR is 360°, the mechanical design of MAC will block some viewing angles. When this block occurs ultrasonic sensors will be used.

Following a user is one of the defining features of MAC. In order for MAC to know what to follow MAC needs to be capable of seeing its target. Multiple following schemes were considered and we settled on using a camera for following when MAC has line of sight on the follow target and after careful consideration the decision to use a camera and AI processing was chosen. The camera and AI combo is best for our application because it is self contained onboard MAC, no external hardware needs to be placed in the home environment. Additionally, high

resolution and high FPS cameras are cheap and easy to acquire when compared to other mechanisms like infrared beacons or ultrasonic pulses.

MAC's software will require additional sensors to keep maximize the software potential. Data that the software needs for MAC to perform the best is the distance traveled and the orientation. There are several different ways to achieve both of these measurements. After careful consideration of multiple ways to measure distance traveled we determined that hall effect wheel encoders would achieve our goal in a manner that is both cost effective and simple to implement. As for determining orientation there are several methods that all different types of devices use to determine orientation from cars to phone to planes. For our purposes we only need to know orientation for two reasons. Reason one is so MAC's lift will be facing the correct direction when the user wishes to load cargo by sliding it from the table/counter onto MAC. Reason two is in the case that MAC is attempting to drive up/down an incline we can ensure that the incline is not too steep and potentially dangerous. With these reasons in mind we determined that a gyroscope will meet our needs while not having extra features that will not be utilized, such as the additional sensors that are built into an IMU.

## 5.2) Hardware Technology

### 5.2.1) Lift

There are several different types of scissor lift mechanisms that exist and are used for different purposes on the market today the main two being hydraulic and electric. Large scale scissor lifts for industrial applications where scaling large heights and/or heavy loads is required use hydraulics as pumping a cylinder to a higher pressure is generally easier and cheaper than electric counterparts.

The downsides of hydraulic lift mechanisms is that they are inherently heavy and power inefficient. The reasons for this is because the fluid required to move hydraulic cylinders is inherently heavy compared to the fluidless electric system, and in order to achieve the high working pressures that hydraulics operate at lots of material must be present to ensure safe operation. All of these trade offs lead to a lifting system that is strong, but heavy, and for our purposes MAC's lifting system should aim to be lightweight so that we can maximize runtime on a single battery charge. So we decided to pursue an electric scissor lift.

There are two types of electric scissor lifts that are commonly used belt driven and spindle driven [77]. Belt driven provides the benefit of being fast and cheap, however there are several downsides that come along with those advantages. Belt driven scissor lifts are complex and provide a nonlinear load on the lifting motor which results in inconsistent lifting speed, which if inconsistent enough can cause cargo to become unsteady and in the worst case fall, which is unacceptable. Furthermore, there is little not a straightforward way to implement a mechanical backup in the event that a belt breaks, which in the worst case can cause the load floor to fall down and drop the cargo, or crush anything that is below. On the other hand, spindle driven scissor lifts suffer no issues from nonlinear loading or catastrophic failure modes. In the event

that the motor breaks, the spindle splits, or either of them seize the lift will just stay stationary. This is an acceptable failure mode as there are no safety concerns if a user is in the immediate vicinity.

So with all of these things in mind we set out to find an electric spindle scissor lift or a mechanical spindle scissor lift that we could incorporate a motor into. After extensive searching we were not able to find one that met our needs while remaining within our budget, and size requirements. There exist several mechanical spindle scissor lift tables on the market however they were sized for industry use not home use, and are too large for the home environment. Furthermore, the industry sized lifts were too expensive for a practical home robot like MAC. The closest we came to finding an off the shelf solution was a motorcycle scissor lift that was designed for individuals to use in their personal spaces, which can be seen in the figure below. This scissor lift was good in several ways but unacceptable in others.

Figure 6: Example of a Motorcycle Style Scissor Lift.

These motorcycle lifts are cheap and readily available [78]. They can easily be found rated for loads greater than 1000 lbs which is far more than MAC's planned payload rating. It would be straightforward to mount an electric motor to the input shaft and because the top and bottom plates are made of solid metal it would be simple to bolt this lift to MAC's chassis. All of this seems good, unfortunately there is one fatal flaw: the maximum lifting height of this style of scissor lift is generally around 1foot which is unacceptably low considering our goal is to have a load floor that raises to 3 feet. This led us to decide to design and construct our own scissor lift.

Designing our own scissor lift allows us to acquire all the lift features that we need and will make use of and disregard all of the features that we do not need, for example 10x load rating. We used the motorcycle lift for inspiration because, as previously discussed, the design almost meets all of our needs. We began by copying the motorcycle lift but extending the runner length this is the most straightforward method and would be simple to manufacture. Extending the runner length only works if you can extend the length of the lift longer than the desired height, because with a single set of scissors the maximum height will be limited by how long the

scissors are when they are oriented vertically. The following figure demonstrations our initial conceptual design of the lift.

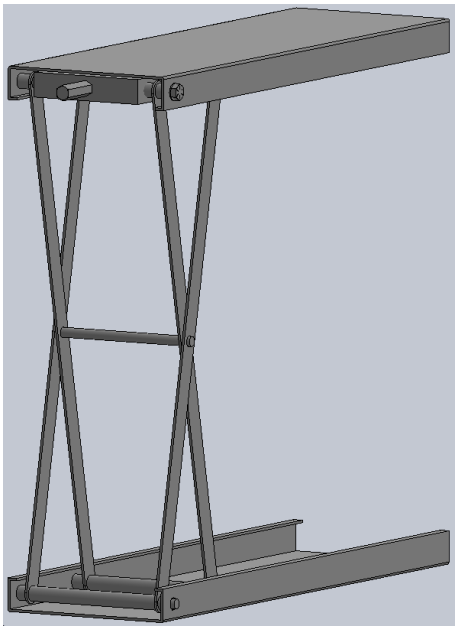


Figure 7: Scissor Lift Design Iteration

This iteration was 30 inches long and only able to achieve a height of 27 inches. With a lift this long we approach the maximum length that was allocated for MAC. Additionally, from the side profile with such a long loading floor if weight is loaded toward the rear there will be a large bending moment which could cause failure of the structure and/or cause MAC to fall over, which is unacceptable. So we decided that multiple sets of scissors will be necessary.

Finally, after extensive time spent modeling and considering the manufacturability and cost we arrived at the final design of MAC's lift, the figure for which is directly below.

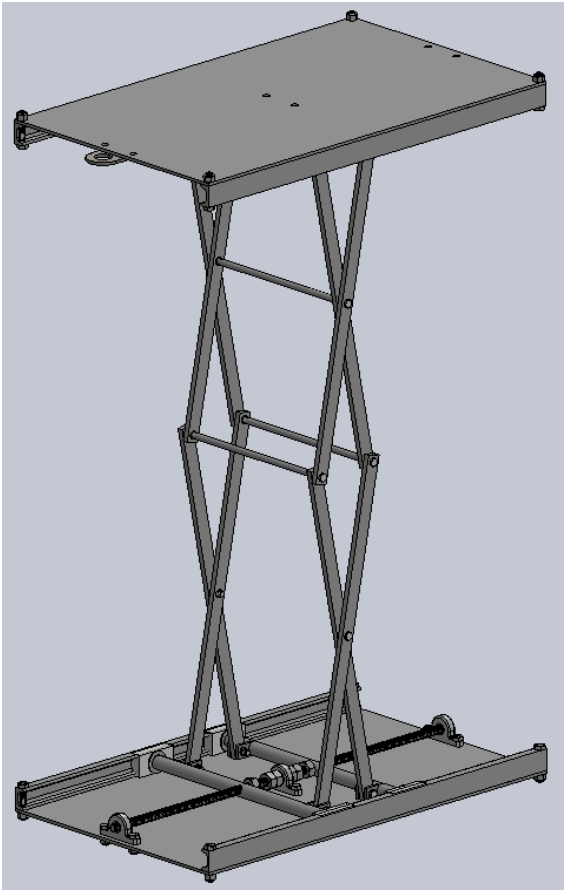


Figure 8: Final Scissor Lift Design

This scissor lift design is simple to manufacture using basic tools and a small amount of welding, and the materials are cheap to acquire consisting of mostly steel, aluminum, and plastic. With this design the overall length is 20 inches and the maximum extended height is 32 inches, which meets all of our goals. This design works by having a nut attached to the runner cross pieces that move closer together or farther away from each other depending on the direction of rotation of the motor, which in turn raises or lowers the height of the lift at a constant speed in a while applying a constant load on the motor. Furthermore, in their fully extended positions the scissors are centered longitudinally. This results in minimal bending moments if the cargo were to be loading the lift from either the front or back. The runners will be made from Delrin plastic because it is easy to cut and form into an appropriate shape for the steel runners and more importantly because Delrin on steel has a low frictional coefficient meaning less torque required from the motor. The spindle will be made from ACME threads because the purpose of ACME threads is for load bearing machinery. At every rotational join there are bronze bushings to reduce friction and on the bottom plate there are bearing housings to the ends of the spindle and keep the spindle from bending.

### 5.2.2) Ramp

Ramp will be incorporated into the chassis because with our current time, budget, and mechanical design experience we cannot get the load floor low enough in the down position so for loading stuff from the ground a ramp will be utilized.

### 5.2.3) Chassis and Drivetrain

The following section will discuss MAC's drivetrain and chassis. The selection of these components was limited because all of them were supplied by a sponsor for no cost. Although the selection was limited, the components the sponsor generously provided are capable of accomplishing MAC's goals, and in some cases were the components we would have selected if we were purchasing ourselves.

#### 5.2.4.1) Chassis

In order to minimize cost we will be using a chassis and gear train from a sponsor. The chassis that the sponsor was able to supply is the AndyMark AM14U4. This is a 6 wheel drop center chassis that is configurable to meet our needs for size and capabilities. The chassis is capable of mounting 4 CIM motors and has belt driven wheels. This chassis will work well for MAC as it is of appropriate size and is capable of supporting the weights that we plan to carry. Additionally, this chassis is able to be configured in a tank drive which is our preferred method of navigating. Also the chassis is able to accommodate the batteries that our sponsor is also supplying which makes for efficient packaging as heavy objects low to the ground is preferable to prevent possibility of tilting.

#### 5.2.4.1) Motors

Our sponsor is able to provide us with access to three different types of motors. Each of them has different capabilities, of which these three are compared in the table below.

| Model: | CIM | RS775-5 | 775 Redline |
|---|---|---|---|
| Stall Torque (in-oz): | 343.4 | 35 | 6.2 |
| Stall Current (Amps): | 133 | 20 | 130 |
| No Load Speed (RPM): | 5310 | 5700 | 21020 |
| Max Power (Watts): | 337 | 240 | 380 |
| Voltage (Volts): | 12 | 12 | 12 |
| Weight (lbs): | 2.8 | 0.86 | 0.8 |
| Miscellaneous: | N/A | Encoder | N/A |

Table 3: Motor Comparison

For our applications where MAC will be carrying a lot of weight we will require the relatively huge amount of torque out of the CIM motors. Additionally, the CIM motors mount to the AM14U4 chassis with no modification necessary. We may choose to use one RS775-5 in

addition to the CIM's because the RS775-5 has a built in encoder which reduces both cost and complexity of us having to add one ourselves.

### 5.2.4) Sensors

This section will discuss the selection of specific sensors that were chosen to be used on MAC.

#### 5.2.4.1) LiDAR

In order for MAC to map out the environment it is navigating a LiDAR will be required. A 360° LiDAR is critical because MAC will be required to turn, and back up to best navigate. Below is a comparison of two different LiDAR's that could be used from mapping the environment (seen in the figure below), the key features we considered when selecting the LiDAR and why they are important, and finally the LiDAR that we ended up selecting.

| Model: | YDLIDAR X2 | RPLidar A1M8 |
|---|---|---|
| Manufacturer: | EAI | Slamtec |
| Range Frequency (Hz): | 5000 | 8000 |
| Scan Frequency Min (Hz): | 6 | - |
| Scan Frequency Max (Hz): | 12 | 5.5 |
| Range Min (m): | 0.12 | 0.15 |
| Range Max (m): | 10 | 12 |
| Scan Angle (degrees): | 360 | 360 |
| Cost ($): | 79 | 99 |

Table 4: LiDAR Comparison

**Range Frequency:**
The range frequency is how many times per second the LiDAR sends out a beam to scan the environment. The higher the frequency the more samples of the environment are generated which means there is more data that can be used to construct the most accurate representation of the environment as possible.

**Scan Frequency:**
The scan frequency of the LiDAR is how fast the housing contains the sensing element. Depending on the range frequency, how fast the data can be sent to the processor, and the maximum scanning range one is trying to achieve would determine the optimal scan frequency.

**Range:**
The range is how far/close the LiDAR can detect objects in its environment. Depending on the application a large max or low min range will be important.

**Scan Angle:**

The scan angle tells us the angle that the LiDAR can see in its environment. Both of the LiDAR's have a 360° scan angle meaning the LiDAR can see everything around it that is in the same plane as the sensor.

**Cost:**
Cost is important to consider because a single LiDAR is an expensive single component and we do not want to overpay for something that has features that are not important to MAC's operation. At the same time we do not want to underpay and have MAC not be able to accomplish its goals.

Considering all of these factors we determined that the YDLiDAR X2 would suffice for MAC. The variable scan frequency along with the short minimum scanning range and cheaper cost offer a LiDAR that MAC will be able to take advantage of to its full advantage. The slightly shorter maximum scanning range of the X2 is not of concern because the use of the LiDAR is for navigating the immediate environment.

#### 5.2.4.2) Ultrasonic Sensors

Ultrasonic range sensors are required for where MAC's chassis interrupt the LiDAR's vision. There will be multiple ultrasonic sensors present. It is important that the ultrasonic sensors be simple to implement and can easily communicate to MAC's processor. Additionally, they must have a range measurement that is appropriate for home use. The following are two ultrasonic sensors that were considered for use in MAC.

| Model: | URM13 | MB1010 |
|---|---|---|
| Manufacturer: | DFRobot | MaxBotix |
| Operating Frequency (kHz): | 40 | 42 |
| Range Min (cm): | 15 | 15 |
| Range Max (cm): | 900 | 350 |
| Measuring Frequency (Hz): | 10 | 20 |
| Communication Interfaces: | I2C, UART, Analog | Serial, Analog, Pulse Width |
| Cost per ($): | 24 | 25 |

Table 5: Ultrasonic Sensor Comparison

**Operating Frequency:**
The operating frequency is the frequency of the pulse that the ultrasonic sensor emits when it is performing a scanning measurement.

**Range:**
The range is how far the ultrasonic sensor can detect objects inside its acoustics range. A longer range allows MAC to see objects farther away. It should be understood that a larger range for sensors that have the same input voltage will result in a lower resolution overall [79].

**Measuring Frequency:**
The measuring frequency is the frequency that the ultrasonic can be sampled for a range measurement. A high measuring frequency means that the sensor can tell the range of objects inside its acoustics range more often, which is desirable.

**Communication Interfaces:**
The communication interfaces are the protocols that the sensor will use to communicate with MAC's main processor. The interface will be used to read measurements from the sensor and to configure the sensor if needed.

With all of these characteristics taken into consideration we have chosen to use the MB1010 because of its higher resolution and simple implementation [80]. High resolution is of critical importance because the ultrasonic sensor is filling in the gaps that the LiDAR cannot see, which is critical to safe and reliable function of MAC especially when loaded with cargo.

#### 5.2.4.2) Limit Switch

A robust limit switch is critical to the safe operation of the MAC's lift. If the lift tries to travel too high stability of the lift could be compromised resulting in the cargo being dropped or in the worse case MAC falling over which is unacceptable in both situations. With these important characteristics in mind we considered the following limit switches. All of the following limit switches have a single pole single throw circuitry and are configured for ON-Momentary functionality. Additionally, all of the following limit switches have mounting holes so that they can be easily mounted to the MAC's chassis and/or a part of the scissor lift. The following switches were considered.

| Model: | MS0850506F035P1A | D2HWBL261H | ZMCJM9L0T |
|---|---|---|---|
| Manufacturer: | E-Switch | Omron | C&K |
| Operating Lifetime (cycles): | 50000 | 100000 | 10000 |
| Operating Force (grams-force): | 60 | 75 | 150 |
| Insulation Resistance (Mohms): | 100 | 100 | 100 |
| Contact Rating Current (A): | 0.4 | 1 | 0.2 |
| Contact Rating Voltage (VDC): | 20 | 24 | 60 |
| Cost per ($): | 1.26 | 3.42 | 2.79 |

Table 6: Limit Switch Comparison

**Operating Lifetime:**
The operating lifetime of a switch is how many times a switch can reliably be pushed and released (1 cycle) and have no failures. A high operating lifetime is important for longevity of a product [81].

**Operating Force:**
The operating force is the force required to activate the switch from its unpressed position to its pressed position [82]. A light operating force can be beneficial for fast detection. A heavy

operating force can be beneficial for ensuring that false positives do not occur under vibration or impacts, although this is of little concern to MAC.

**Insulation Resistance:**
The insulation resistance is the resistance of the internal electrical components to conducts that are touching the plastic body of the limit switch. A high insulation resistance is important because if any voltage is present on the part that the limit switch is connected to then there could possibly be false positives of activation which are undesirable and potentially dangerous.

**Contact Current Rating:**
The contact current rating is how much current the switches contacts can withstand when the switch is activated. A higher contact rating generally indicates a more durable switch and is half of the equation when calculating the contact power that the switch can handle.

**Contact Voltage Rating:**
The contact voltage rating is the other half of the equation for determining the amount of power that contacts can handle. Contact voltage rating describes the voltage that a switch can reliably operate at while the switch is activated.

**Cost:**
While the cost per unit is low it is still important to consider because multiple switches will need to be used for redundant operation to ensure safety.

With all of the following qualities considered we determined that the E-Switch MS0850506F035P1A will best fit our needs [83]. The low operating force in conjunction with the low price is the perfect match for MAC's lift and will ensure safe operation of the scissor lift.

#### 5.2.4.2) Camera

For MAC to be able to follow things it needs to be able to see what it is trying to follow. As discussed before MAC will use a camera to follow its target. Therefore it is imperative that a quality camera is selected. Several cameras with different features were considered based on specifications in the table below.

| Model: | C922x | N60 |
|---|---|---|
| Manufacturer: | Logitech | Nexigo |
| Resolution: | 1080p | 1080p |
| Field of View: | 78° | 110° |
| Frame Rate: | 30fps | 30fps |
| Price: | 99.99 | 39.99 |

Table 7: Camera Comparison

**Resolution:**
The resolution of a camera is a measure of how small a camera can see. A higher resolution means that the camera can more clearly see the environment. It also means that there is more data for MAC to process. Both of the cameras we are considering have a 1920 x 1080 resolution [84].

**Field of View:**
The field of view of the camera is how wide the camera can see. A wider field of view is more desirable because it allows more of the environment to be seen. The N60 has a field of view that is 20° wider than the C922x which is a significant increase in performance.

**Frame Rate:**
The frame rate of the camera is how many pictures the camera takes every second. A higher frame rate means that the camera can see how things in the environment are changing more easily. A higher frame comes at a cost though because more frames means more data that MAC has to process. Both cameras capture at 30 frames be second so the performance will be similar for both.

**Price:**
The price of cameras is important because one of the goals of MAC is to be affordable. The C922x is 2.5 times the cost of the N60 [85].

After consideration we determined that MAC will be using the N60 camera because of its adequate performance features while being significantly cheaper than other cameras on the market.

### 5.2.5) Power

MAC will be battery powered. The reason for battery power is that it is relatively cheap, quiet, and can be easily recharged using the mains power that is readily available inside the home environment. The batteries that MAC will be using were determined by the batteries that our sponsor was able to provide. The batteries that we received are Interstate SLA 1116. This is a 12V lead acid battery that is rated for 18AH. Although lead acid has its disadvantages it will suffice for MAC which has a goal runtime of 15 minutes fully loaded or 2 hours unloaded.

Furthermore, and perhaps the biggest advantage, is that the charging of lead acid batteries is simple and low cost. When compared to more modern battery chemistries such as lithium ion the circuitry needed to safely charge a lead acid battery is simple and consequently cheaper to implement.

### 5.2.6) Controls

MAC will respond to users inputs. There are several methods that a user can do to tell MAC what the user desires. The main three that we considered are as follows: buttons built into MAC's chassis, voice activation, wireless remote control.

The simplest method is buttons somewhere on MAC. With these buttons the user can command MAC to do various tasks, such as moving in a specific direction, raising/lowering the lift, returning home, etc. Buttons on MAC are cheap and simple to implement, requiring only the buttons and a little bit of wire to implement. However, buttons will not be an adequate solution for several reasons. The main reason being that the user has to be standing/walking next to MAC while operating the buttons. Depending on the orientation of MAC in the environment, operation of the buttons might be difficult or not possible. Furthermore, if the user is trying to command MAC to move in a certain direction the user will be standing right next to MAC and this can create issues with MAC's obstacle avoidance software creating false positives. To implement this would introduce additional complexity to the system for very little return because the reasons previously described.

The second and most complicated option is to use a voice activation system. This system would listen to the user and respond based on spoken commands. With voice activation the user can remain at distance from MAC and not interrupt MAC's navigation through an environment. Also, voice activation does not require the user to keep track of any other components or ensure that the components have power. Unfortunately, this method, while novel, is not appropriate for MAC for several reasons. The main reason is that in order for a reliable and safe system the execution of the system must be robust. For a user to tell MAC to turn a specific distance or move forward a specific distance the system but be robust or MAC risks collision, and when cargo is loaded onto MAC the possibility of collision must be as close to non-existent as possible. Additionally, proper voice activation would require more processing power that is already going towards navigation and path planning of the environment. Finally, MAC will have several functions that the user will be able to control, with voice activation the user will need to learn commands to speak to MAC which is not always preferable depending on the users preferences.

The third solution, and the solution that we will be adopting, is a wireless remote. A wireless remote would have buttons that the user interacts with similar to the first solution discussed earlier. The wireless remote is selected because it has all of the advantages of the previous two options and almost none of the downsides. Wireless remotes allow the user to send reliable and clear communications to MAC from a distance, and the remote can be labeled in a way so that anyone can use MAC at any time. Additionally, a wireless remote can be operated from entirely different rooms if desired. The only downside is the remote has to be powered in some way but this tradeoff is acceptable given the advantages we acquire.

There are several sources already on the market that sell wireless remotes that fill a role similar to the role we desire when we discuss a remote for MAC. We examined several options that

already exist on the market and while attractive there are some roadblocks. The second major roadblock is that the engineering documentation for the off the shelf products is limited at best as most of the documentation for the remotes internals are confidential to company employees only. So we are presented with the problem, do we purchase an off the shelf product and attempt to reverse engineer it to get it to accomplish our goals? Or do we build one ourselves and control the development from the ground up? After some consideration we determined that building our own wireless remote would be most appropriate for MAC. Also, one other reason worth noting is that developing our own remote allows us to completely control the design of button layout and placement to suit our needs perfectly, we do not have to try and force another design into our application. Next we will talk about the high level design of the wireless remote.

The wireless remote should be cheap and easy for the user to operate. There will be two components to the wireless remote; the first is the remote itself and the second is the receiver that is located on MAC and connects to MAC's main processor. The remote will gather inputs information from the user then package and wirelessly send them to the receiver. The receiver will receive the information and then convert it so that MAC's processor can accept the information.

The first question to ask is what type of wireless communication is best suitable to MAC's goals? So we examined possible wireless communication systems, and after much research we determined that using the 2.4GHz ISM band with the main reasons being that there is very little cost and no regulation of the band. Additionally, several modules already exist that are low cost and simple to integrate and come in a very small form factor while still being capable of reliable data transfer for the ranges that we expect the user to encounter in the home environment.

The next question is how the receiver will communicate with MAC's processor, the Jetson Nano. After much research into the capabilities of the Jetson we determined that the simplest method was to use one of the Jetson's USB ports as the Jetson already has USB controllers already onboard. This will greatly reduce the complexity of the receiver design and BOM.

Next we must consider the architecture of the remote. In order to accomplish our goals of a low cost and simple to operate remote it makes sense to incorporate only the minimum amount of components. The minimum required components include buttons, a wireless transceiver, a simple processor for input collection and controlling the transceiver, and a battery.

Following this same logic the minimum required components for the receiver are a transceiver to receive information from the remote, and a USB converter to translate incoming information into USB so that the Jetson can receive and process the data. Another benefit of having the receiver plug into a USB port on the Jetson is that the USB port can power the receiver so that no extra power source is needed, which simplifies the design of the receiver PCB significantly and also lowers the cost of the entire wireless remote.

Next we will go over the block diagram and the components selected for the remote and the receiver, the organization of which can be seen in the figure below.

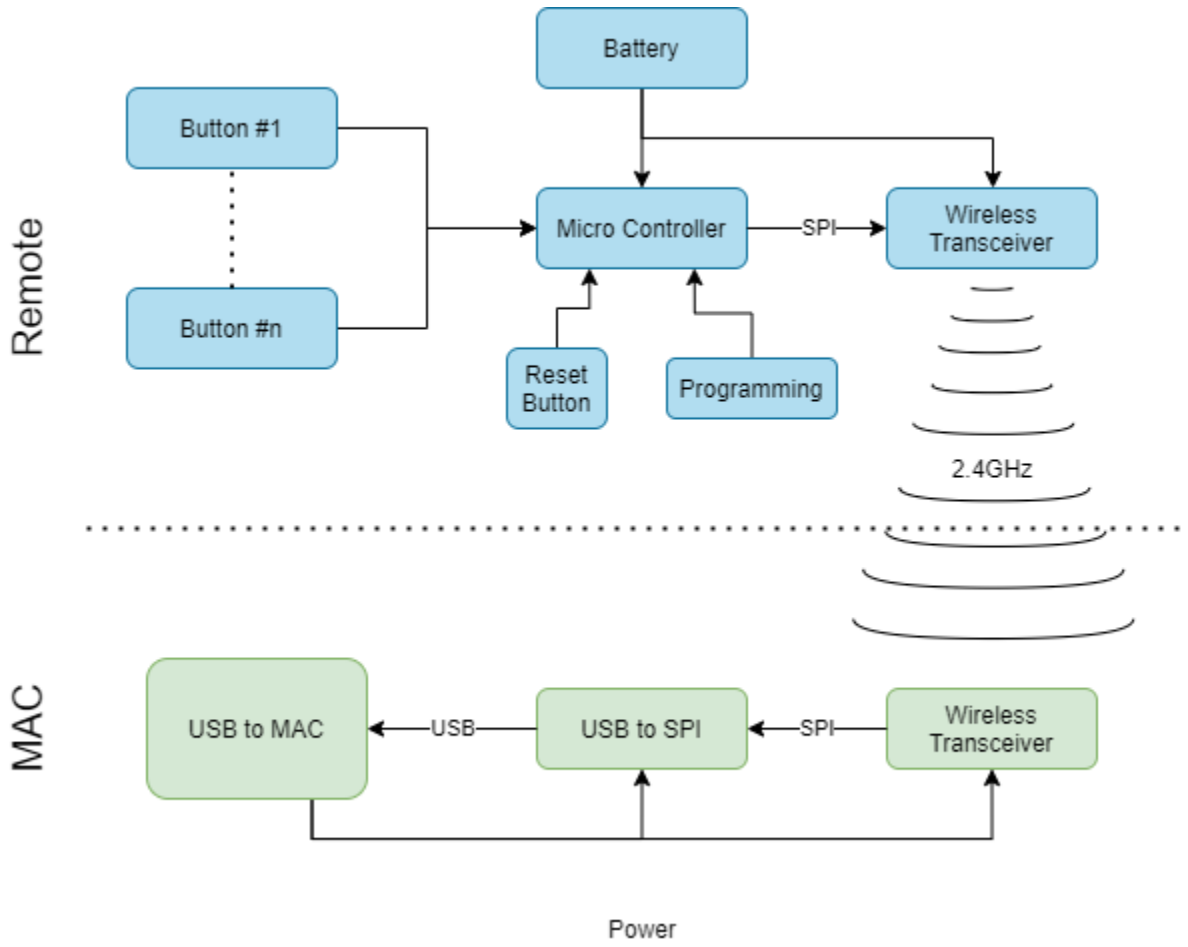



Figure 9: Remote Block Diagram

#### 5.2.6.1) Wireless Transceiver

The main component of the wireless remote is the wireless transceivers. They are what allow the remote to be wireless so selection of a robust wireless transceiver is paramount. The decision to use a wireless module was chosen over developing an integrated transceiver into the PCB we are developing. The reason for this is because there are several off the shelf transceiver modules that are small and low cost which suit our needs perfectly. We considered multiple wireless transceiver modules and had to make the decision between two which were readily available. It is important to note the nRF24L01 is the standard wireless transceiver that operates in the 2.4GHz ISM band so that is why both of the selections use this IC for the base of their product. Two manufacturers of the same style receivers, and their manufacturers, are compared below.

| Model: | nRF24L01+ | nRF24L01+ |
|---|---|---|
| Manufacturer: | Sparkfun | RobotDyn |
| Communication Protocol: | SPI | SPI |
| Transfer Speed Max (Mbps): | 2 | 2 |
| Range (m) ideal: | 100 | 80 |
| Size (mm): | 20.3x22.8 | 30x15 |
| Price ($) per: | 20 | 3 |

Table 8: Transceiver Module Comparison

**Communication Protocol:**
The communication protocol is the scheme that the transceiver uses to communicate to the controller so that the transceiver can be configured and be sent data that it will send. Both of the modules use the same transceiver so the communication protocol is the same which is Serial Peripheral Interface [86].

**Transfer Speed:**
Transfer speed is how fast data can be sent/received by the transceiver. This transceiver offers adjustable data transfer speeds which can be useful depending on our design goals. A lower transfer speed often leads to longer range and more reliable transmission because the signal to noise ratio is higher. Both modules use the same transceiver which offers a max transfer speed of 2Mbps.

**Range:**
The range of the module is how far the module manufacturer predicts their module can transmit data in ideal conditions. The Sparkfun module has a higher range because the antenna chosen for the module is a discrete ceramic. Whereas the RobotDyn uses a trace antenna which is not able to reach as far.

**Size:**
The size of the module is the length and width dimension in millimeters. A small module is important because it is easier to package into a remote, and it is easier to package for the receiver that is on MAC. If we compare the areas the Sparkfun is 463sqmm and the RobotDyn is 450sqmm so slightly smaller.

**Price:**
The price of the component is important because the goal of the remote package is a low price. As can be seen the price of the Sparkfun module is over four times greater per than the RobotDyn which is not insignificant.

Considering all of these options we decided to go with the RobotDyn module because of the greatly reduced price [87]. The range of the Sparkfun module is clearly superior however the RobotDyn modules range will suffice for the home environment.

#### 5.2.6.2) Remote Microcontroller

In between the buttons that the user interfaces with and the transceiver there will need to be some sort of microcontroller. This microcontroller will be responsible for recording and interpreting the button presses from the user and configuring and communicating with the transceiver the data that the transceiver needs to send to the receiver on MAC. Several microcontrollers were considered for the job. The goals of the microcontroller are to be cheap and simple to configure. Several different microcontrollers were researched and two were decided upon (seen below).

| Model: | MSP430FR2000 | STM8L101F1 |
|---|---|---|
| Manufacturer: | Texas Instruments | STMicroelectronics |
| Processor Size: | 16bit | 8bit |
| Operating Voltage: | 3.6V | 3.6V |
| GPIO: | 12 | 18 |
| Frequency: | 16MHz | 16MHz |
| Flash: | 0.5 kB | 2 kB |
| Communication: | SPI \| UART | SPI \| I2C \| USART |
| Price per ($): | 0.22 | 0.99 |

Table 9: Considered Microcontrollers

**Processor Size:**
The processor size is the size of data that a processor is designed to use. A larger processor size means that a processor can compute more data for each cycle. The MSP430 has a 16bit processor compared to the STM8's 8bit processor.

**Operating Voltage:**
The operating voltage is the voltage that the microcontroller needs in order for reliable operation, also the power supply for the microcontroller cannot exceed this value too much or we risk the chance of damaging the microcontroller. Both the MSP430 and the STM8 have an operating voltage of 3.6V so we do not need to worry about power supply considerations between the microcontrollers [88].

**GPIO:**
The GPIO number tells us how many general purpose input/output (GPIO) pins are present on the microcontroller. The MSP430 has 12 and the STM8 has 18, generally more GPIO pins are better because it allows for more flexibility but this is not always necessary depending on the requirement of the project. For our purposes both of these microcontrollers offer enough GPIO pins.

**Frequency:**
Frequency is the frequency that the microcontroller operates at. A higher frequency is generally better because it allows for the microcontroller to process data faster, provided everything else is equal. Although there is generally a tradeoff of increased power consumption because more work

is being done by the controller. Both the MSP430 and the STM8 run at 16MHz so the same frequency.

**Flash:**
Flash size is the size of non-volatile memory that is within the microcontroller. This size dictates how large the programs that we create for the remote. A larger flash size means that we can have a larger program running on the microcontroller. Larger flash size is nice to have but is not always necessary and does increase cost. The MSP430 comes with 0.5kB of onboard flash memory and the STM8 comes with 2kB of onboard flash. For our purposes 0.5kB should suffice.

**Communication:**
The communication describes what types of communication protocols the microcontroller supports. SPI is the protocol that we are concerned with as that is the protocol that is required for our transceiver module. In addition to SPI, the MSP430 also comes with UART, and the STM8 comes with I2C and USART (universal synchronous/asynchronous receiver transmitter). There are no plans as of right now to utilize the other communication protocols that both of these microcontrollers offer.

**Price:**
The price of these controllers is important, the price of the STM8 is almost five times larger than the price of the MSP430. One of the main goals of the remote is to be low cost.

In the end we determined that we will be selecting the MSP430 microcontroller [89]. The reasons are multiple. First, although the absolute price increase is small in comparison to other components, a key goal of the remote is low cost. The other specs of the MSP430 will suffice so why should we pay more for features that we will not make full use of? Another practical consideration is that we do not have access to a programmer for STM microcontroller while almost all of our team has access to programmers for TI microcontrollers.

#### 5.2.6.3) USB to SPI

The receiver only serves the goal of wireless communication and the wireless communication we receive from the remote is only useful if the Jetson can see the information. The receiver can only communicate in SPI so we need some method of converting the SPI of the receiver into USB to SPI converter becomes necessary. There exist several types of USB to SPI converters on the market. Both the converters that we considered can be programmed over the USB connection that will be made with the Jetson. This is advantageous because it means that the PCB can be simpler and in turn cost less. We considered two converters, the specifications of which are available in the table below.

| Model: | MCP2210 | MAX3421E |
|---|---|---|
| Manufacturer: | Microchip | Maxim |
| Data Rate: | 12Mbps | 12Mbps |
| Operating Voltage: | 5V | 3.3V |
| USB Speed: | Full | Full |
| EEPROM Size: | 256B | 256B |
| Buffer Size: | 64B | 128B |
| Oscillator Requirement: | External | External |
| Price ($): | 2.04 | 4.23 |

Table 10 : USB to SPI Converter Comparison

**Data Rate:**
This data rate is in reference to the transfer rate of data from the SPI device to the USB controller. A higher data rate means more data can be transferred to the controller in the same amount of time. Both of the controllers have the same data rate which is 12Mbps which is the speed limit of USB 2.0.

**Operating Voltage:**
The operating voltage is the input voltage that the converter needs to run properly. The MCP2210 can accept 5V inputs. 5V is what USB runs on so this makes powering the MCP2210 easy and no extra components are necessary. The MAX3421E can only accept 3.3V inputs so a regulator must be present if we expect to draw power from the USB connection from the Jetson Nano. This requirement raises both cost and part count. Also both of these converters have internal voltage regulators so that they can properly run SPI communications.

**USB Speed:**
This speed is the speed that the converters can communicate with a host device be it a PC, phone, or controller such as a Jetson nano. Both of these converters can operate at full USB 2.0 speed which is 12Mbps. This speed is plenty fast for the communications that we plan to do with our remote receiver.

**EEPROM Size:**
The size of the EEPROM for each controller is how much storage exists on the device for storing configurations of the controller and device descriptions depending on the type of SPI device that is connected to the controller. Both devices EEPROM are the same size.

**Buffer Size:**
The buffer size is a measure of how much data the converter can fit into its buffer before the data is sent/received from the SPI device. A larger buffer size is advantageous because it means that more data can be placed into the buffer. The MAX3421E has a buffer that is double the size of the MCP2210 [90]. This size difference will be of marginal significance as most of the time the SPI bus will not be pushed too heavy.

**Oscillator Requirement:**
In order for the converters to send data through USB they need a 12MHz oscillator so that the data can be sent at the proper frequency. Both of the converters selected require external oscillators.

**Price:**
The price of the converters is important because one of the main goals of the wireless remote system is to be low cost. The MCP is half of the price of the MAX.

After careful consideration we have decided to go with the MCP2210. The main reason being the price of the MCP2210 is cheaper. The reason for this is that the MCP2210 has an internal 5V regulator which eliminates the cost and size of the extra required components [91]. This works towards the goals we set out when we considered the idea of a wireless remote.

#### 5.2.6.4) Buttons

Buttons are the primary way that the user interacts with MAC. Reliable, simple, and low cost buttons are necessary for users to have a positive experience with MAC. The buttons will be simple by being momentary switches that short their two terminals. The buttons will be configured in a matrix array so that we can reduce the number of GPIO pins that are necessary for the microcontroller.

### 5.2.7) Dock and Charging

One of the nice to have things that complements MAC's goals is a docking charging station. The goal of this station is that when MAC detects that its battery power is low MAC will automatically navigate "home" and connect to the power that the dock power. The dock would receive power from the house's mains and convert it so that it can recharge MAC's batteries. MAC is using lead acid 12V batteries, compact chargers are readily available and cheap to acquire because of how common lead acid 12V batteries are in automobiles today.

The construction of the dock can be made out of low cost materials like wood and PVC because the dock will not need to be load bearing in any way. The challenging part of implementing the docking system is how MAC establishes electrical connection to the dock in a safe and hands free manner. One solution is to rely on MAC's navigation to line up perfectly. While possible this solution is not very robust and heavily dependent that MAC's positional data encounters zero problems for the duration that MAC is in use. Any problems with sensor error/drift, program miscalculation, mechanical compliance, user shifting MAC's position, etc. can corrupt MAC's data cause MAC to not be able to return to the dock.

One solution that is being considered is an alignment track that guides MAC into place as it drives into the dock. Attached to MAC is the female end of track and the dock contains the male end. The track would be similar to a triangular configuration. The idea is that as MAC drives towards the dock there is a lot of room to miss but, because of the track, as MAC moves closer the track helps MAC find its way home. This solution allows for some error in MAC's

navigational data while not docked  as long as MAC is close enough it should be able to successfully dock.

The final problem of the automatic docking system is how to establish electrical connection to MAC once successful docking occurs. We could do inductive charging as it requires no wires. There are two problems with inductive charging. The first problem is the low efficiency which can be expensive when you consider the amount of energy that MAC needs. The second reason, and the critical one, is that inductive charging is slow especially considering the size of the batteries we are attempting to charge.

So, we have determined that a wired electrical connection is necessary. There are several challenges with creating a wired electrical connection. The first challenge is ensuring proper connection is established. The second reason is designing the connection in a way so that while MAC is not docked there are no bare electrical contacts that can potentially injure someone if they were to be touched or get carelessly shorted out. A solution to the first problem is the use of a magnetic connector. Magnetic connectors have zero insertion force and depending on the design can be forgiving of slight misalignment issues. There exist several magnetic connectors on the market but real world testing of different types would be required to determine which one is appropriate for our purposes. A solution to the second problem would be a trapdoor system that moves out of the way and exposes the connector when MAC is docked. Ideally this system would use a few mechanical linkages built into the dock that get actuated when MAC docks, this design is advantageous because it requires no electrical communication between MAC and the dock. Another method to solve this problem is to have some form of communication between the dock and MAC and upon docking the dock opens the trapdoor and exposes the connector with one or several servos that actuate all the necessary components.

As we have stated this feature is nice to have and budgets and timelines will determine if the docking station will be constructed. This section is just to consider what elements would constitute a good charging dock and a general idea of what technologies we would employ to make the dock a reality.

## 5.3) Hardware Architecture

The following section will discuss the architecture of the hardware that we designed for MAC. This will include the remote and the remote receiver.

### 5.3.1) Remote

We have previously discussed the component selection that was done for the main components that make up MAC’s remote. Now we will discuss the design of the remotes schematic shown below.

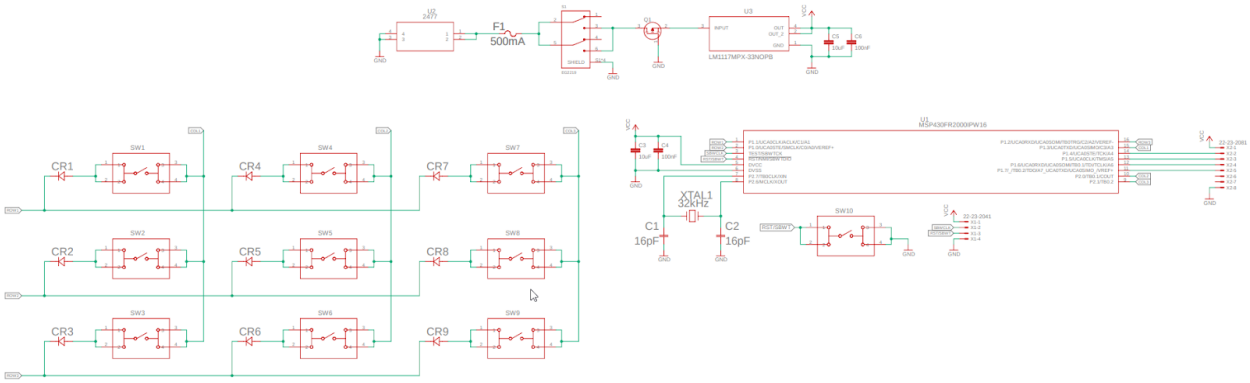



Figure 10: Remote Schematic

As we decided earlier the heart of the remote will be the MSP430FR2000. We will be utilizing all of the available pins for different functions. The schematic is grouped into three sections top, bottom left, and bottom right. The top section is responsible for power, the bottom left is responsible for input, and the bottom right is responsible for control.

Let's take a look closer at the power section shown in the following figure.

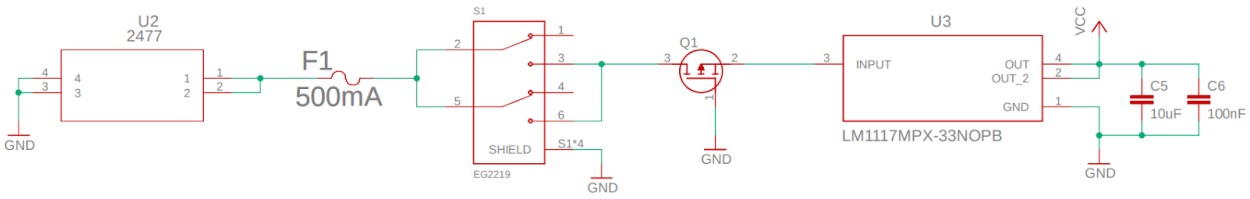



Figure 11: Remote Schematic Power

U2 is the battery housing which will be soldered directly to the board. The board will be powered by 4 off the shelf Alkaline batteries configured in series. The batteries, configured in series, will output 6VDC and provide around 15Wh in total. A 500mA fuse directly follows the batteries for over current protection. Then a single pole switch for turning the remote on/off to conserve battery power when not in use. AA batteries are reversible and although the change of a user installing all 4 batteries backwards is unlikely it is easily possible. To address this an enhancement mode PNP MOSFET (Q1) is used configured such that if all the batteries are installed incorrectly the MOSFET will not conduct and protect the rest of the circuit. Following the MOSFET is a LM1117 linear voltage regulator that outputs 3.3VDC which is the operating voltage of both the MSP430 and the transceiver. At the output of the regulator are a few capacitors of varying frequencies for noise filtering.

Now let's take a look at the input section shown in figure below.

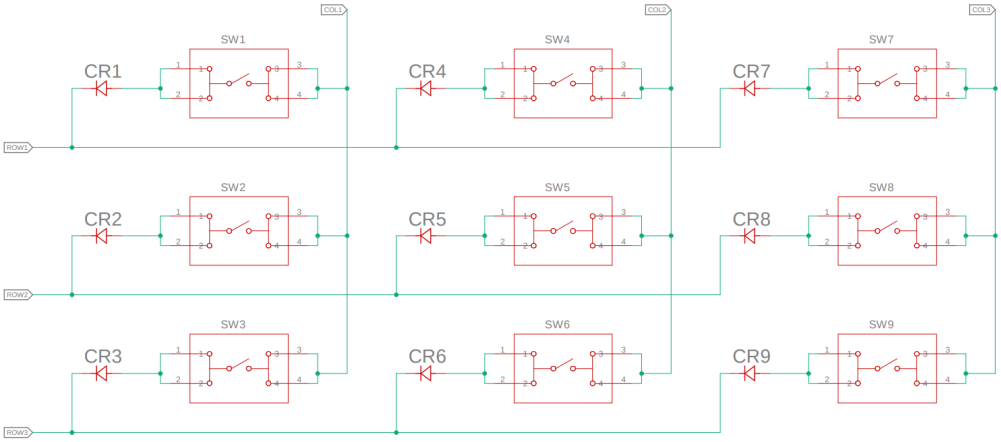



Figure 12: Remote Schematic Inputs

This section details the connections of the button matrix that we will be implementing. This matrix is advantageous because it allows us to have 9 buttons while only needing 6 GPIO pins. If this matrix is not implemented we would have to use a different microcontroller with more GPIO pins and would probably result in a higher cost. The diodes are placed to prevent "ghost presses" as explained in the section detailing the operation of button matrices.

Finally, let's take a look at the control section shown in the next figure.

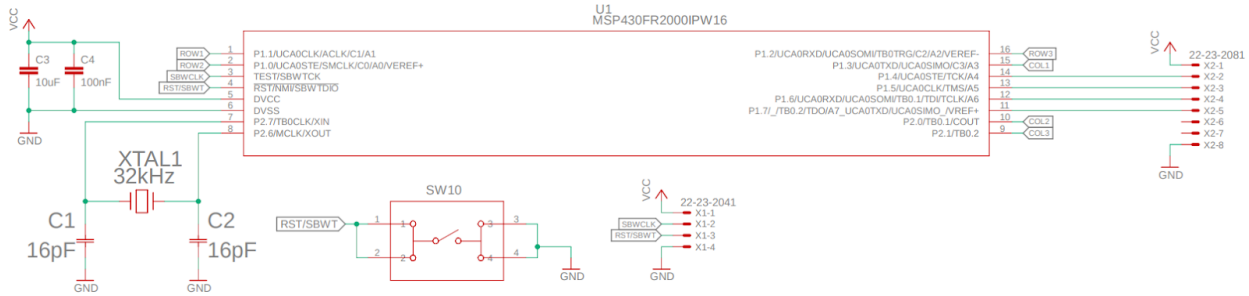



Figure 13 : Remote Schematic Control

This section details the connection of the MSP430 to other critical components. The MSP430FR2000 is the brain of the remote and controls all of the inputs and outputs that the remote makes. The capacitors in between the MSP430's VDD and VSS pins are for noise filtering and are recommended by the datasheet. The crystal and its load capacitors connect so precise timing of program events can take place accurately. The button is for resetting the controller if needed. The bottom 4 pin port is for programming the controller and is connected to VCC and ground because the recommended programming procedure of our programmer says that the microcontroller and programmer should share the same power when being programmed to prevent voltage level inconsistencies between the board and programmers power supplies. Finally, we have the 8 pin connector for the nRF24L01+ module. The module will get power and

ground from the batteries and all of the SPI communications will be done directly with the microcontroller.

### 5.3.1) Remote Receiver

We have previously discussed the component selection for the components that make up MAC's remote receiver. Now we will discuss the design of the remotes receiver schematic shown below.

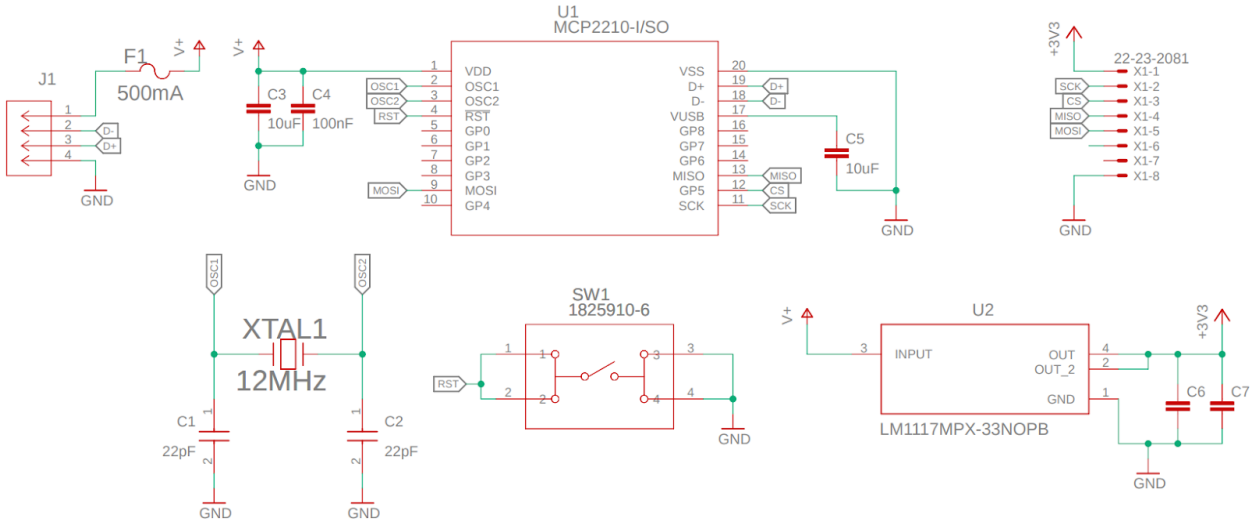



Figure 14: Remote Receiver Schematic

The USB connection both provides the power for all of the components and will allow for communication with the Jetson Nano. The brain of the Receiver is the MCP2210, it is responsible for receiving messages from the nRF24L01+ and then passing the messages to the USB port. There is a 500mA fuse for protection of both the components on the receiver and protection of the Jetson Nano's USB port. The crystal and its load capacitors are there because the MCP2210 does not have an onboard crystal for USB 2.0 communication. The button is for reset of the MCP2210 if needed. The LM1117 is needed because the nRF24L01+ operates at 3.3V and the USB will be providing 5V. The 5V is not a problem for the MCP2210 it contains all of the internal circuitry to properly accept a 5V power. The VUSB pin is the output of the MCP2210's internal 3.3V LDO it powers all of the 3.3V internals. The datasheet recommends that when the MCP2210 is powered by USB bus power to the pin to a filtering capacitor. The pullup resistors required for proper USB communication are internal to the MCP2210.

## 5.4) Software Design Overview

The software for the MAC will run on a Jetson Nano using Python. Our design will be based around ROS (Robot Operating System), which is very robust for building autonomous robots for almost any application. ROS follows a subscriber-publisher model, where sensors and other inputs will publish their results to different functions subscribed to receive those results, such as modules for navigation, planning, following, and various other functions. This model makes it

relatively easy to add new functionality such as new sensors or features, since these modules can simply be added on top of existing code as publishers for sensors and other input, and subscribers for the features, making the MAC extendible to do more then it's base functionality. ROS also has modules already made for navigation, planning, and sensor combination, meaning less time we need to spend on writing these functionalities from scratch. For our line out sight following feature, we will be using computer vision powered by deep learning to detect the person in the camera's field of view and follow them to their destination from a safe distance. This will be powered by Yolo V3 (you only look once). This model is a fully convolutional model that performs object detection and localization, it is very robust and widely used for various tasks involving some type of detection. A fully convolutional model is a model with only convolutional layers, more is explained in the Following section. This model will be trained using the Pytorch machine learning library, and also deployed on the Jetson Nano using the same library. This will feed in bounding boxes of the detect people in its field of view when following is enabled, which will then be used to calculate a distance and direction for the planning module to start moving to. We also have stretch goals to have the robot automatically dock, which will have the robot return to it's starting location and dock to a charger, although this will depend on the dock's technology and how easy it is to connect to charge automatically. We will also try to implement a remote call feature, where the robot can be called to the user via the robot. Typically, this is done by use of a smartphone companion app for something like the Roomba where the location is set on the SLAM map, however we want to avoid using external apps as much as possible, so we are researching some different technologies to allow the robot to get the position of the robot and go to it, no matter where it is in the house. The motor controllers control the various functions of the robot like driving, lifting, and the ramp will be controlled via CAN for the SPX Victor motor controllers. Python has a library for connecting devices over can, and the software will send commands to the motor controllers to either set them to a value of -1 to 1, which is full reverse power and full forward power, with a value of 0 indicating no power.

The figure below demonstrates the general flow of our software. The MAC will start in an idle state, waiting for instructions. The user will press a button on the robot's display or a provided remote to do some task. There will be buttons to follow, put the lift up and down, stop the robot, and return to dock. Commands will only be run one out a time, i.e lifting will not work unless the robot is stopped and is idling. The user interface is designed to be as simple as possible, no complicated controls or button presses will be needed, so no matter the technical ability of the user they can operate it as simply as a TV.

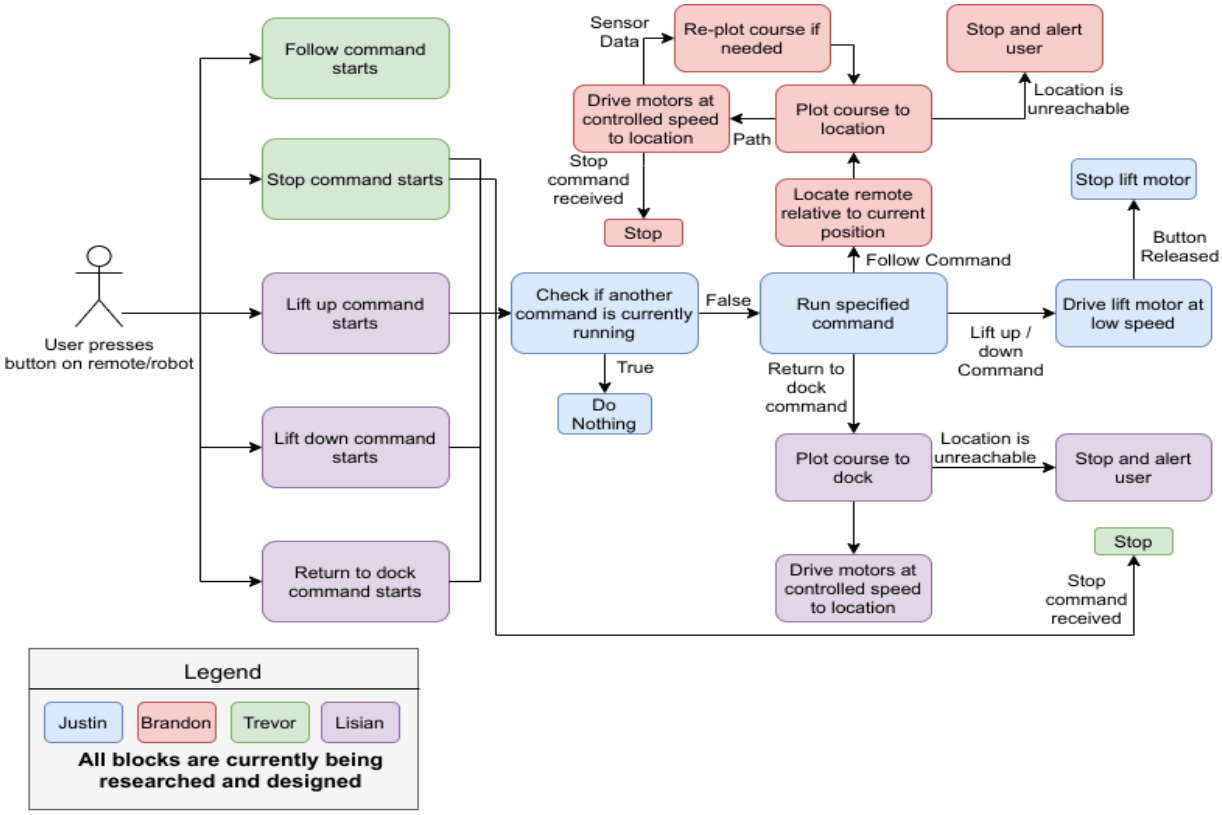



Figure 15: Software Block Diagram

## 5.5) Software Technology

### 5.5.1) ROS

Robot Operating System, known as ROS, is used widely for automated and teleoperated robot control. ROS has many include packages and user created packages to do a variety of tasks, from mapping and planning to motor control and sensor integration. Users are able to contribute these packages since ROS is fully open source, meaning anyone can contribute to the development of ROS. ROS will be used for all robot control, including SLAM algorithm for mapping, A* path planning, getting user inputs, docking, controlling motors via motor controllers over CAN bus, and safety features, ROS is written using Python 3, using a subscriber-publisher model for taking input, processing, and output control.

Despite its name, it is not its own operating system. ROS will run on any system that has Python 3 installed, in which case our Jetson Nano will be running Ubuntu 20.04 with our software running in it. This gives us much flexibility and also portability, since we can move our software over to any compatible computer system to run the robot, we are not required to stay with the Jetson Nano should we decide to change it.

We will be using ROS Noetic Ninjemys, the latest version for ROS that is supported until May 2025 on the Ubuntu 20.04 operation system. This version of ROS is also ROS 2, which is a complete rewrite from scratch of the original ROS system, with many new added features and bug fixes. From ROS 1, ROS 2 now has a convention from writing your nodes (which are the base object where classes are derived from to access the ROS functionalities), unlike in ROS 1 where nodes can be made in many different ways. This helps standardize implementations and makes it easier to code and troubleshoot. ROS 2 also has a new feature, called components, which allows creating many nodes in a single program, to control a variety of processes for the robot and do computation without needing to run to multiple separate programs. Nodes also now have the functionality to call setup functions before running, so that sensors and other components can be initialized before the program can use them. A big change is that services (which handles requests between server and nodes) are now asynchronous, which means that the robot can handle multiple tasks simultaneously for computation speed up. Overall, ROS 2 is more robust and will be able to handle control of our robot easily and allow us to focus on the logic rather than boilerplate code to make the low level communication and control work properly.

The ROS 2 subscriber-publisher method breaks down between a few different modules. Nodes are the base elements of a ROS robot. A node contains algorithms to perform some task, whether it be communication, reading sensors, processing data, navigation, or user input, everything is run in a Node. Each Node in rose is responsible for a single module purpose. For example, one node to control the drivetrain, one for the lift, one for controlling navigation sensors, etc. Each Node sends data to other Nodes via topics, services, actions, and parameters. Services are another way besides topics (explained below) for Nodes to communicate. Services use a call-and-response model versus the topic's publisher-subscriber model. Mainly, they only provide data when specifically called by a client, instead of continuously regardless of who is listening. They can also serve many clients, but can only be one server running for a specific communication channel.

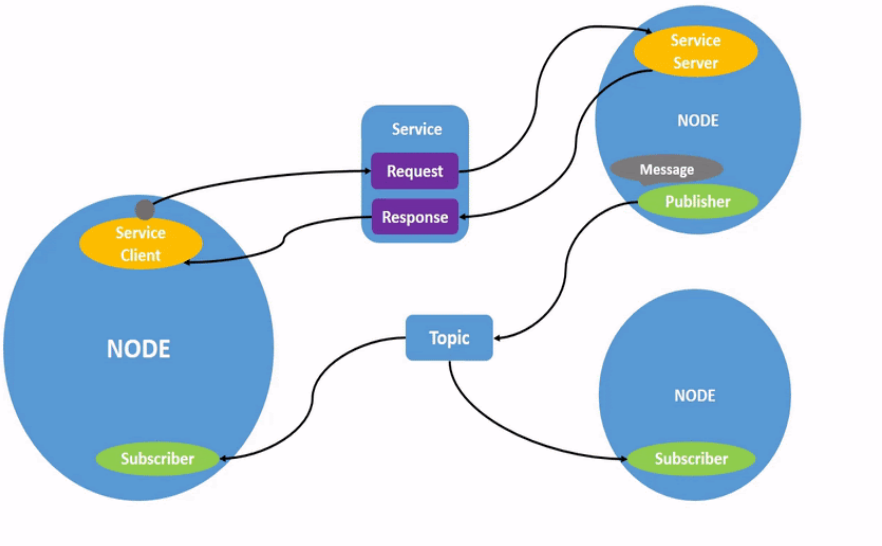


Figure 16: ROS Node Topology Breakdown

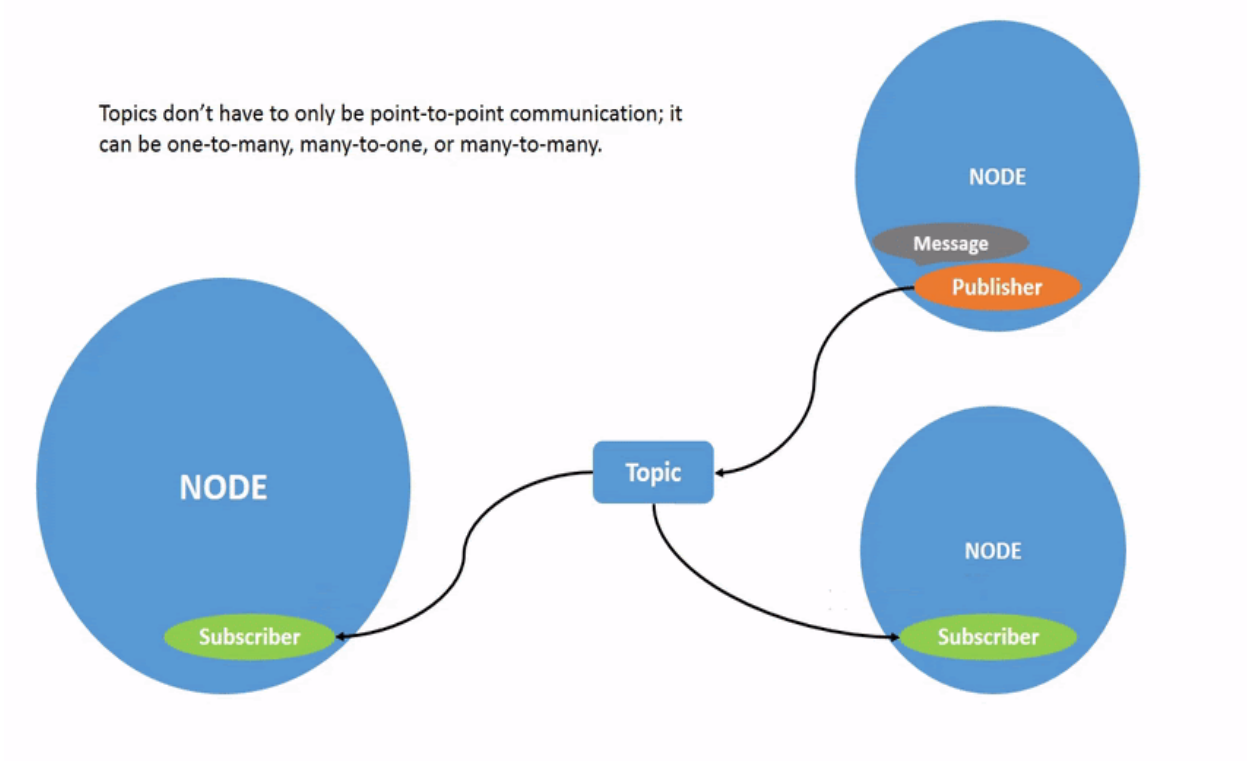



Figure 17: ROS Topic Topology Breakdown

A topic handles data flow between nodes. Multiple nodes can either publish or subscribe to a topic, so for example a topic to handle navigation may have publishers for LiDAR and ultrasonic data, a subscriber that performs SLAM mapping using this data, and finally a path planner to plan a path using the data provided. This makes communication easy and robust with ROS, allowing each functionality of the robot to be modularized to different nodes and communicate easily between multiple nodes asynchronously.

Finally, actions are a core communication type for long running tasks. They have 3 parts: a goal, a result, and feedback. They are built on top of topics and services, and are preemptable. This means that they can be cancelled while executing. They also provide steady feedback, unlike services which only return a single response. These use a client-server model, which is similar to topic's publisher-subscriber model. The client sends a goal to a server, which acknowledges the goal and returns a stream of feedback and a result from the action. The main point of actions is the continuous feedback, so the action clients can constantly update and change their output so their next result from a request is the most up to date available to ensure reliability of the entire robot software.

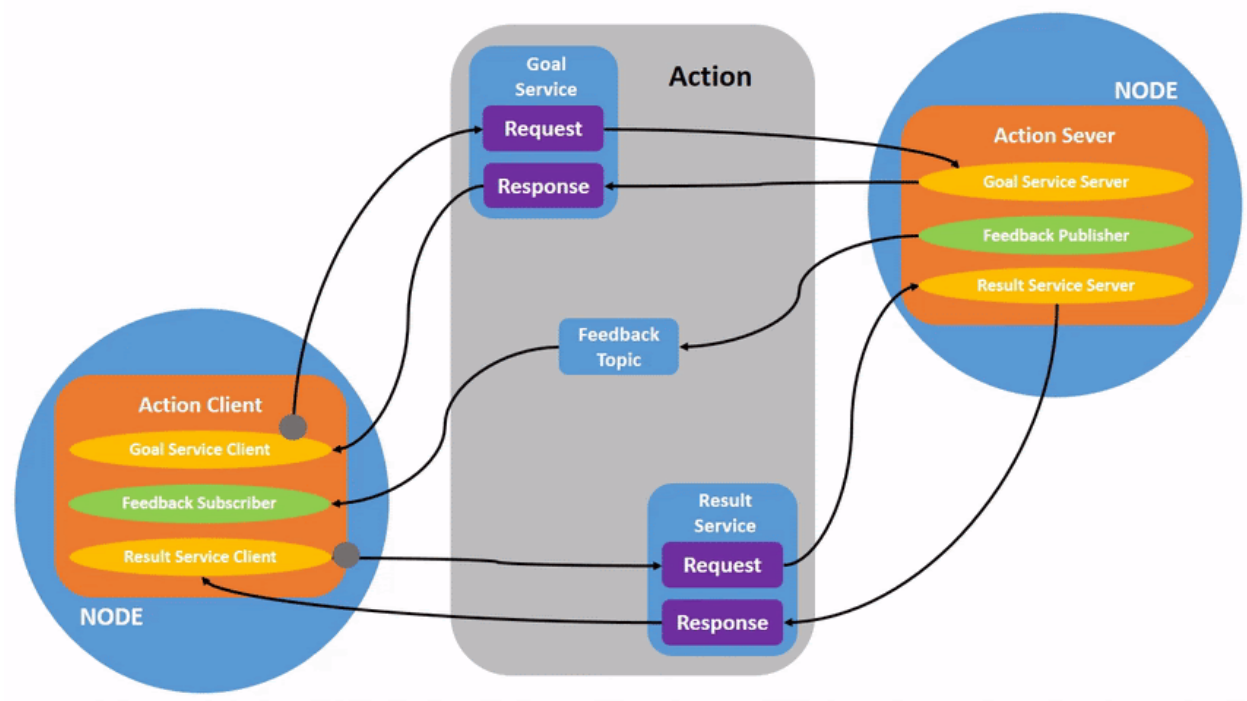



Figure 18: ROS Action Topology Breakdown

These models will be the basis for our robot, which all of our logic will be built under. Since nodes can use external libraries alongside ROS extensions, ROS can directly control motors using an external CAN library for communication over the CAN bus to our Victor SPX motor controllers. Each of these functionalities are broken down below.

### 5.5.2) SLAM Mapping

ROS has a built in package called **navigation** that handles mapping out the home with a LiDAR, encoders, and ultrasonic sensors using the SLAM (Simultaneous Localisation and Mapping) algorithm. Encoders are used for odometry for the robot's current velocity and helps with positioning the robot in the environment, while the LiDAR and ultrasonic sensors are used to build a map of the environment [92].

SLAM, as the name says, localizes obstacles in the environment while mapping it out. There are quite a number of ways to implement this concept, it is not a set algorithm or function. In our case we will be using 2D Autonomous SLAM for mapping out unknown areas using the Karto SLAM algorithm since it has better real world performance compared to other common algorithms like Hector SLAM, Gmapping, and Largo SLAM. Typically, SLAM has two sides to solving the problem. The first side is the SLAM front-end which handles the problem of associating data to the environment, mainly sensor collection and combination. This is handled by the standard ROS navigation package from combining sensor input for SLAM. The second part is the SLAM back-end, which is where Karto SLAM falls. The back-end handles pose correction, cost mapping, environment creation, and optimization of the cost map by maximizing

the likelihood of measurements taken from the LiDAR, ultrasonic, and encoder sensors to create a cost map of the environment.

The figure below shows generally how the SLAM algorithm works. When the robot's position changes, landmarks are extracted from the new position along with odometry data to determine the robot's location in the environment. For Karto SLAM this is done with sparse pose estimation, where only a subset of the environment and map landmarks are used to determine robot position before updating, instead of using all the data which is computationally expensive.

SLAM will be one of the core functionalities of our software, since proper navigation is required for our system to operate as expected. By using Karto SLAM in ROS, we will have precise navigation and robot localization for the MAC to traverse any home.

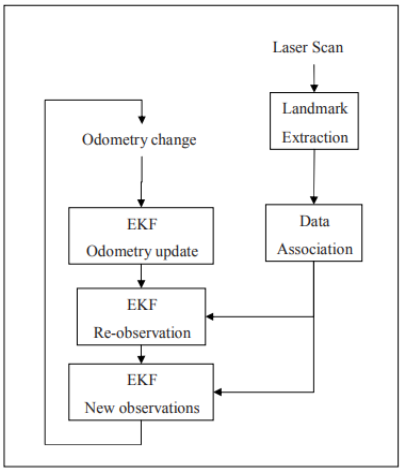



Figure 19: General SLAM Algorithm

### 5.5.3) Path Planning

The ROS navigation stack using the Trajectory Rollout and Dynamic Window approach to path planning in an environment. The cost map generated by the SLAM algorithm and the position to reach, the algorithm will plan out a kinematic trajectory for the robot and give velocity commands to the drivetrain to reach the location. Along the path, the planner creates a value function around the MAC represented as a grid map. This value function will encode the costs of traversing through each grid cell, these values determine dx, dy, and dtheta velocities to send to the drivetrain of the MAC.

The Trajectory Rollout and Dynamic Window Approach (DWA) algorithms follows:

1. Discretely sample robot's control space for dx, dy, and dtheta
2. For each sampled velocity, perform forward simulation from robot's current state to predict what would happen if the velocity was applied for a short duration

3. Evaluate the score of each trajectory resulting from forward simulation, using a metric based on proximity to obstacles, proximity to goal, proximity to global path, and speed, and remove illegal trajectories (such as obstacle collision).
4. Pick highest scoring trajectory and update velocity to drivetrain
5. Repeat steps 1-4

DWA typically is more efficient then Trajectory Rollout, but has issues with low acceleration limits since DWA does the forward simulate constant accelerations. Typically though, both perform reliability and the ROS package recommends DWA, but both can be used interchangeably. The main difference between the two is how the MAC's control space is sampled. Trajectory Rollout samples from the set of achievable velocities over the entire forward simulation period given acceleration limits of the robot, while DWA samples from the set of achievable velocities for just one simulation step.

Since this package is built into the ROS navigation stack, implementing and testing which algorithm performs base will be easy, allowing us to rapidly test multiple approaches to choose the best one for the MAC to map and navigate its environment. It also ensures we do not need to program these algorithms from scratch, allowing us to focus more on the design of the robot and software itself.

### 5.5.4) LOS Following

The MAC will feature a forward facing camera for detecting a single person to follow for our LOS (line of sight) following feature. The camera will be connected directly to our Jetson Nano which will be running a deep learning model called Yolo V3 ("you only look once"). Yolo V3 is a fast and accurate fully convolutional model for object detection and localization, outperforming other models to perform the same task by a factor of 4. Typically, it is used for object detection of everyday household items, but we will be training it just to detect people in order to follow them. The network will output bounding boxes of the detections, which will then be transformed into a distance and angle for input into the path planning node to follow the user throughout the home.

Neural Networks at their core is a system of linear matrices that are combined using a dot product then passed through some type of activation function (like sigmoid, ReLU, tanh, softmax), which are then passed to the next layer. Each path from one node to the next has an associated weight and bias, which is what is used to calculate the output of the next node using this formula, where input is the values of the nodes in the previous layer, w is the weights for each connection, and b is a constant bias added: $activation(input * w + b) = output$ .

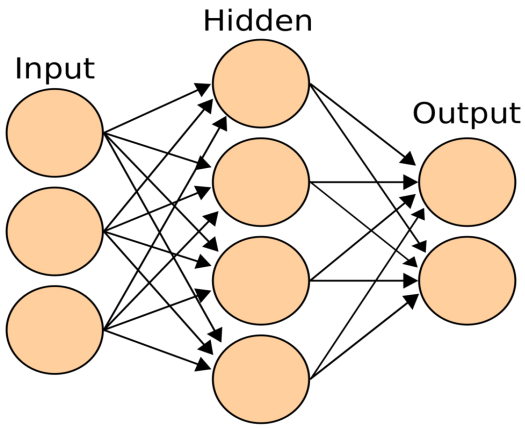



Figure 20: Neural Network Architecture

The neural network is then told how wrong the output is, the error. This error is then propagated backwards throughout the model, where every weight is updated such that the next time the model sees the input again, it will predict the right value. This process is called backpropagation, and it is how models are trained to generalize some function that defines the relationship between the input data and the expected output. Convolutional Neural Networks takes this a step further by introducing convolutional layers. These convolutional layers excel at feature extraction from the input data. For images, this can be high level features like lines, shapes, corners, gradients, etc, to low level features more specific to the task, like a tire, a face, an ear, etc.

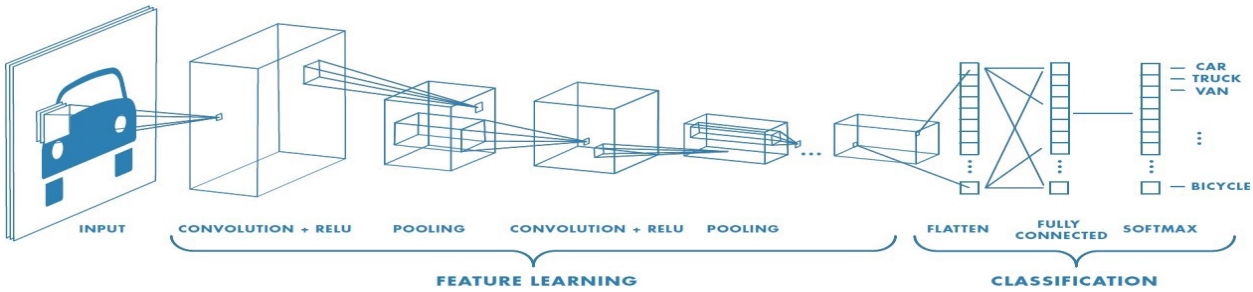



Figure 21: Convolutional Neural Network

These layers work by using a sliding window (like a 3x3 matrix) that goes across and down an image, convolving the features in the window with a kernel. These feature maps will contain the extracted data. The kernels are the learnable parameters that learn what features to extract to lower the error of the model. This process not only extracts features but localizes them to different parts of the image, meaning features can appear anywhere in an image and still be detected, which is critical for accurate object detection.

For Yolo V3, the model does not rely on any classifiers or localizers to perform the detection, hence it being fully convolutional. The network is applied to the full image by dividing the

image into regions and predicts bounding boxes and probabilities for each region. Each bounding box is weighted by the predicted probabilities. This means predictions take the entire context of the image which helps with accuracy, and is also much faster since it only looks at the image once.

Since the Jetson Nano is built with running neural networks in mind, it can perform reliably at 15-20 FPS for a video feed in order to accomplish this task. The model will only need to run while this feature is being used, otherwise it can be suspended when other functions are currently being used. This will help with resource management and making sure our Jetson Nano can handle all of the processing. When following, the MAC will wait until a person is detected in its camera. Then, the bounding box will be used to calculate the distance to the person using the center coordinate of the box, the resolution, and the camera's FOV angle. Once this is calculated an angle can then be calculated from the center of the camera, and given these two values a path can be planned to this location. The planner will replan the path as the person moves in the home. If the MAC loses sight of the person (like going through a doorway), it will travel to their last known location and wait to find the user again. Depending on how well this works, this can be further enhanced to predict where the person may have gone (like added an extra distance to get the MAC through a doorway or around an obstacle) such that the user doesn't have to constantly check the MAC detects them, they can travel normally. This feature is important as it ensures a user doesn't have to predefine a location for the MAC to go to, they press a button on the remote and the MAC will follow them to their destination. For those who are in wheelchairs, the model can be trained to also detect wheelchairs and other scooters so that it can follow them without issue as well as someone walking.

### 5.5.5) User inputs

There will be multiple ways to interact with the MAC robot. The main control will be a provided remote that connects directly to the MAC via 2.4 GHz wireless signal. The remote will contain buttons to lift the MAC's platform up and down, start the LOS following function, stop the robot, and return to dock. Each of these buttons can only be used one at a time, meaning that if the robot is currently lifting, following won't work, and vice versa. The robot must be stopped before giving it another command.

There will also be an on board display on the MAC to control it if the remote is unavailable. The display will be a 7 inch touch screen containing software buttons to do all the functions described above. They will also work the same way, where only one function can be used at a time. This is provided for sake of convenience, typical use will involve the remote.

The last part of the user interface will be a smart phone companion app for the remote call feature. The app will display the map created by SLAM on the phone, where the user can tap on the map to call the MAC to that location. The cell phone app can also call the same commands like on the remote and the on board display, but its main use is for those who want to use the remote call feature of the MAC.

### 5.5.6) Docking

Docking is a nice to have feature we are planning to incorporate into the MAC. There will be a physical dock with a charge port that the MAC will automatically return to and connect such that it can charge when not in use. This will work with a single command from any of the user input methods. The location of the dock on the map will be marked where the MAC initially recharges. This is set the first time the MAC docks after mapping out the home environment. If the dock moves, then the MAC will have to be manually docked to update the dock to the new location on the SLAM map. The dock will have a snap connector and a physical guide, where the MAC will line up too and connect with when docking.

The docking procedure will be as follows:

1. Plan path back to the dock using the path planner node
2. Traverse back to the dock
3. Once in front of dock, orient the robot to line up with the dock using the on board gyroscope, such that it is at the same angle as the dock
4. Back up until connection is made with the charger and the MAC is charging

The docking function will use the functionality that other functions use like following and such, using the planner and drivetrain functions to control the MAC back to the dock. Docking is important to make sure the MAC is always available for use, and not have to rely on the user to make sure the MAC is plugged in and charging when not in use. The MAC will also have an idle mode when idling and not charging, where some functionalities are shut down to preserve power like the planner, SLAM mapper, sensor data collection, etc and only have communications open to wake back up. Docking is something that will rely more on the hardware being built properly compared to the software being available, since it uses the functionalities already present in the software.

This feature is a nice to have feature, meaning that we may not be able to complete it for the final design. If the physical dock is not completed or we do not have enough time to implement this feature, then the MAC will still have a docking functionality. This docking functionality will simply remember the location where the MAC was last charged and when the MAC is docking it will return to this location for easy charging and storing use.

### 5.5.7) Safety

The MAC will have a number of features in order to ensure the safety of not only the user, but the user's home, other people in the home, and any other pets or obstacles. These features will use the existing sensors to impose constraints on the MAC's movement including tilt, speed, distance to obstacles, and operation of multiple functions at once.

Here is a list of safety features we plan to implement:

- Maximum tilt: If the MAC is going up an incline, it will stop and reverse to get off the include if the tilt reaches too steep of an angle, which we plan to be any angle above 45 degrees off the floor

- Speed: a maximum traveling speed will be 3 feet per second when traveling, and a slower 2 feet per second when following the user using the LOS following function. Similarly, the lift will only travel at 2 inches per second to make sure any load will fall out or be displaced by the lift.
- Distance to obstacles: using the LiDAR and ultrasonics, the MAC will maintain at least 6 inches from any obstacle on all sides of the MAC, if something gets too close it will back up and avoid it, or stop to wait for the obstacle (say a pet) to move if it can't avoid it.
- Multiple functionalities: Only one function of the MAC can run at a time, to avoid any safety risks. For example, the lift can't be operated while the MAC is moving, and vice versa.
- Following Distance: when performing LOS following on a user, the MAC will stay at least 2 feet back from the user to ensure there are no collisions and the user has room to maneuver, especially if they are in a wheelchair or have a walker. It also makes sure that if they happen to fall they won't hit the MAC.

Safety is something that is very important with the MAC, since it will reside in the homes of those who are more easily susceptible to injuries, especially from automated robots roaming about. The MAC will take a more cautious approach to movement, so it may stop or avoid obstacles with extra clearance just to make sure it avoids hurting or damaging anything or anyone. Since we are using a tank drive it's mobility is pretty high, allowing it to turn in place and make fine adjustments when traveling to avoid obstacles easily. All of this combined is to ensure the safety of both the MAC and its environment.

### 5.5.8) Companion App / Remote Calling

An advanced feature that will increase the autonomy of the MAC is to develop and implement a mobile companion application that can communicate with the MAC directly offering extended features that are not present with a physical remote. The mobile application will have a simple user interface that displays the loaded map generated using SLAM by the MAC. For details on SLAM mapping please refer to *section 5.6.2* above. Once the SLAM map has been generated via the ultrasonic sensors, LiDAR and encoders, the map can simply be loaded on the mobile application with areas of the map that can be clicked in order to command the robot to traverse to the intended location in the household by sending the coordinates of the location to the MAC for processing, planning, and traversal.

#### 5.5.8.1) Android Application

In choosing between the two different platforms of iOS and Android, we chose to develop the mobile application on Android for several reasons. Android applications are developed in Kotlin which can fully interlope with Java and is the preferred language for developing Android applications. Since it is similar to Java, the team has more experience in Java through course work rather than Swift which is the preferred language for developing iOS applications. This was an important factor in our decision making as it will be easier for development and meet the time constraints of the project rather where developing an iOS application would require learning of a new language from scratch as well as the platform. In addition, Kotlin is easy to learn especially

with Java experience already and has community support in the form of forums and other resources to help beginners learn and develop their applications.

In addition, the choice of Android and Kotlin for mobile development was influenced as there is now ROS support for Java and Android applications. ROSJAVA is an implementation of ROS in pure Java with Android support. These libraries will be helpful for integrating the companion app with the SLAM map generated by ROS on board the MAC. There is no current support for ROS on iOS devices and thus in order to decrease development time by using an already supported library for ROS on Android we will be able to focus on the implementation and WiFi communication between the companion app and the MAC. Another factor in deciding to create an Android application over an iOS application is the fact that you need a computer running Mac OSx in order to use Xcode which is used for developing iOS applications. There will be multiple people on the team presumably working on the application and not every member of the team has a Mac computer to be able to develop an iOS application. On the same note, there are Android emulators available that also contain virtual hardware for Wi-Fi communication which is how the application will communicate with the MAC. An important caveat is that the emulator contains virtual hardware for Wi-Fi starting from an API level of 25 [93]. Therefore, an emulator will be useful being able to test rapidly without the need of having a physical mobile device that will need to have the APK(Android application package) uploaded to it in order to test the application.

#### 5.5.8.2) Wi-Fi Communication

The primary form of communication between the mobile application and the MAC will be through Wi-Fi. There were other options to consider here such as various versions of Bluetooth and Bluetooth Low Energy that are both used frequently in IoT applications. The common problem with Bluetooth is that it can be inconsistent and the range of communication is limited and the MAC should be remote-called from anywhere in the household. The range of Bluetooth is approximately 10 meters while the range of Wi-Fi can be up to 100 meters and thus in a household, WiFi communication is a more viable option as we want the user to be able to remote-call or direct the MAC from anywhere in the house reliably [94]. A Jetson Nano will be used on board the MAC for higher-level processing, and thus it requires a Wi-Fi module as it does not come with Wi-Fi functionality in its default state. Apart from this, Wi-Fi also provides better security which is always a plus over Bluetooth though it does use more power. Also through Wi-Fi communication we believe it will provide less reliability issues for the user in terms of the Bluetooth connection getting severed as happens frequently with Bluetooth operated devices.

In terms of implementation of Wi-Fi communication, it was briefly mentioned that a Wi-Fi module will need to be purchased and added to the Jetson Nano ports. Furthermore, there will be additional configuration necessary through the Linux operating system running on the Jetson Nano. Depending on the Wi-Fi module purchased, there will be drivers that need to be installed for that specific Wi-Fi module through the command line interface. Once the drivers are installed, the connection between the Nano and the network will be established via several additional commands inputted via the command line interface. Once the Nano is set up for

having WiFi enabled on the Nano is set up, there can be WiFi communication between the mobile application and Nano through TCP sockets allowing data to be sent back in forth by having the Jetson Nano listen for any data that is being sent from the mobile application. Therefore, through asynchronous services provided by ROS, the Nano will act as the server listening for TCP connections from the mobile application. At a high-level the mobile application will be the client but will still be able to retrieve data from the server in this case the Jetson Nano. For simplicity, the mobile application will have the map generated by SLAM through the **navigation** package in ROS. The map of the home will be generated by the MAC on the Nano and periodically send updates to the mobile application so the map is constantly up-to-date for the user. The user on the mobile application will be able to select a location on the map and through the ROSJAVA library be able to send that corresponding coordinate data back to the MAC for navigation. This can be achieved through the help of the **map_server** ROS node that offers map data as a ROS service. The map data that is stored will be a YAML file that contains metadata about the map and the actual .pgm image file that contains the map generated by ROS.

### 5.5.9) Controlling Motors

The Jetson Nano will be used for all higher-level sensor processing such as path-planning, SLAM map generation and as TCP server for a mobile application. Furthermore, the Jetson Nano will be used for controlling the motors installed on the MAC and through the ROS publish/subscribe model. In this case, the motor controller will be encapsulated as a ROS node that can communicate via the path-planning node through a ROS topic. The motor controller will subscribe to the path-planning topic that will publish the necessary velocities via ROS messages. This publisher/subscriber architecture can be seen in Figures 3 and 4 above. Once the velocities are retrieved by the motor controller it will control the motors via CAN commands. The MAC will have two motors and thus two motor controllers to achieve independent motor control for precise movement throughout the household. Additionally, the Jetson Nano does not have out-of-the-box support for CAN and thus a USB to CAN bus converter adapter will be purchased in order for the Jetson Nano to be able to communicate to the motor controllers. The motor controllers will operate on the same CAN bus as two separate devices as well as the Jetson Nano. In this case, the Jetson Nano will be the sole node that transmits data to the motor controllers.

#### 5.5.9.1) Python and CAN

The Controller Area Network (CAN) communication protocol is a reliable and robust message-based protocol that is used heavily in the automobile industry [95]. With the use of CAN there is no need of having a host instead all devices can receive and send frames between each other on the CAN bus. CAN contains four different types of frames that can be sent between devices including a data, remote error and overload frames. For the case of controlling motors, we will be using the data frame containing the necessary data to control motors according to the velocities published by the path-planning ROS node. The data frame's data field can transmit between 0-8 bytes of data which is more than enough for transmitting the necessary data to the motor controllers. Similar to the rest of the software, the motor controller code will be written in Python and use the python-can library for CAN support. The python-can library

exposes a CAN **Bus** object containing several parameters describing the CAN bus such as the bustype, channel and bitrate. The configuration necessary is unknown currently until the parts are purchased and there is further research done on the type of CAN interface necessary for the purchased USB to CAN adapter for the Nano. A simple diagram of a typical CAN node on a CAN bus is shown in The figure below where there is a separate CAN node representing the Jetson Nano and two motor controllers in the case of our robot.

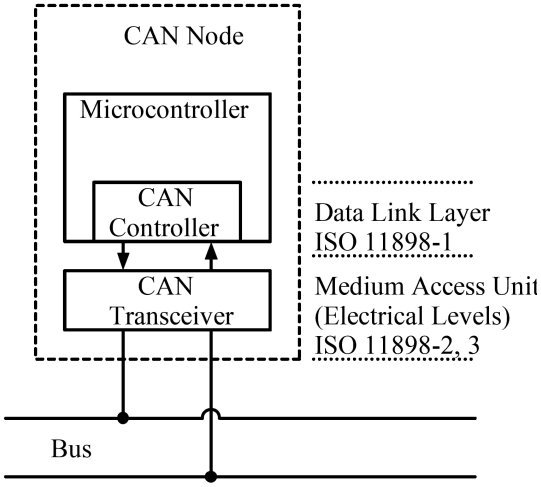



Figure 22: CAN Node

Once the CAN **Bus** object is configured correctly the **Message** object will be used in order to transmit messages as data frames to the motor controllers. The **Message** object contains several fields including the arbitration ID, boolean flags indicating the type of frame being sent, data parameter and several others [96]. For the case of controlling motors, we will be using the **Message** object to send a data frame with the data retrieved from the path-planning ROS node. The data field is exposed as a byte array in Python with a length between 0 and 8 since there can only be 8 bytes sent through a CAN data frame. The array of data can contain integers as well as bytes and the format of this data will depend on how the velocity data is processed in the motor controller ROS node. Since there are multiple devices on the CAN bus that can all receive and transmit data theoretically each will be assigned an ID for differentiation. The Jetson Nano will be assigned an ID of 0, and the motor controllers will be assigned IDs of 1 and 2. These IDs are important for arbitration on the CAN bus and help indicate with which node should proceed(has arbitration) on the CAN bus thus when setting up the devices they should be given different unique IDs so the data transmitted can be received by the correct node. The arbitration ID also naturally results in establishing priority of which node should take arbitration on the bus with a lower ID resulting in a higher priority. In the case of the motor controllers, there is not an issue of which one should take priority as the data communication should be fast enough so that any overhead where one motor controller has to wait for another to receive its data will not cause any issues with the movement. Thus, in software we will specify an arbitration ID for each **Message** object created depending on which motor controller has data available that needs to be delivered.

### 5.5.10) Operating System

The Jetson Nano we will be using Ubuntu 20.04, since it is officially supported by the version of ROS we are using, is open sourced, and free. Instead of using an embedded system board like an Arduino, it has the flexibility to run a variety of programs, servers, and other needed software for the MAC. In this case, it will be using Python 3 to run ROS and all of our software. We can also modify code and run tests right from the Jetson itself using SSH, which means testing and prototyping will be extremely quick instead of having to connect to a board and upload code using a cable or other device.

It also provides much needed flexibility, in case we need to drastically change the software or even to another language. In this case, whatever we can run on Ubuntu 20.04 is fair game, meaning that in the worse case scenarios we do not have to worry about being stuck with a specific language, package, paradigm, or even the operating system itself. We can further connect to it with a full desktop interface using VNC to directly see things like the SLAM map, sensor output, and other graphical debugging tools that come with ROS while the MAC is running in real time, again further enhancing our prototyping and testing capabilities.

Once we are ready for the final release version of the MAC, we will install Ubuntu Server 20.04 to eliminate many of the unneeded packages and software that take up resources, mainly the GUI. This will free up as much resources as possible for the MAC's software to run at maximum performance. For debugging purposes, we will have a computer monitor the MAC via an SSH connection to get diagnostic info before final demonstrations.

## 5.6) Software Architecture

Our software will be entirely composed within the ROS architecture. Controlling motors, getting sensor input, receiving input from the display and remote, all of this will be done within ROS nodes, actions, and topics. ROS is very flexible, and it allows us to use outside libraries as needed, such as the Python CAN library for controlling our motor controllers, which means that everything can be built within ROS and has access to the functionalities of ROS nodes, topics, services, etc.

Generally, each functionality of the MAC given in the software block diagram, which can be put into its own ROS Node or a group of related nodes. In our case, we will have a node for:

- Drivetrain Motor Control
- Lift Motor Control
- LiDAR Sensor Collection
- Ultrasonic Sensor Collection
- Camera Sensor Collection
- Encoder Sensor Collection
- Limit Switch Sensor Collection
- Gyroscope Sensor Collection
- Touchscreen Display Input

- Remote Input
- Smartphone app navigation input
- SLAM Mapping using KartoSLAM
- Path planning using SLAM cost map
- Line of Sight Following using YoloV3
- Charging / Docking
- Safety (to stop the MAC if some error occurs)

These nodes are independent, containing the code to run their specific tasks. These nodes will be linked together using topics, actions, and services to communicate. We will also have some global topics that contain current robot states, like what command is currently running, current path plan, and whether to stop the MAC completely when the stop command is received. The robot state is updated by each of the nodes independently, and every node will subscribe to the current robot state topic in order to follow any changes in the state automatically. There will also be a command manager that responds to inputs from the user and runs specific commands. Some of the topics we will use will be:

- Sensor Data
- Inputs (from remote, app, or display)
- Robot State
- SLAM Maps
- Motor Commands
- Path Points (for giving the path planner instructions on where to go)

One of the bigger design challenges we will face is making sure the nodes work together and obey each others states, so that the MAC will only perform one action at a time and can be properly interrupted (such as receiving a new command to go somewhere else, stop completely, or making sure the lift is disabled while moving). The path planning and SLAM mapping may also be split up into more nodes if needed, as the navigation and SLAM packages that come with ROS have their own nodes and structure that we work off of. While this makes our code base more complex since our code is spread out across many nodes, it makes it modular and robust (quite like the MAC itself) such that we can extend the functionality easily. For example, if we wanted to add another node that requires the sensor data, instead of having to write extra code to allow it to access the data, it simply needs to subscribe to the sensor data topic and get the information it needs. This will help in prototyping and testing since our need for nodes and topics will change as we figure out what works and what does not. We also do not need to worry much about communication since ROS will handle communication between all of our nodes. This leaves us the most time for creating the logic of the MAC.

### 5.6.1) ROS Diagrams

The figure below illustrates the structure of our nodes and topics, while also displaying the subscriber and publisher connections between them. The functions of the MAC are controlled by the Inputs topic and the Command Manager node. Based on the Inputs, the Robot State is updated to reflect what the MAC should be doing. If the input is to make the lift go up and down,

the Robot State would be checked first, and as long as the MAC is idle then the Command Manager will send velocity commands to the lift motors. The Command Manager will also change the Robot State to activate the LOS Following, Docking, or Path Planning modules by having the MAC follow the requested function. The inputs can either be to start LOS following, to go back to the dock, or in the case of the Path Planning a direct position for the MAC to go to (from the smartphone app). During this, the sensors are always collecting information, regardless of the state of the MAC.

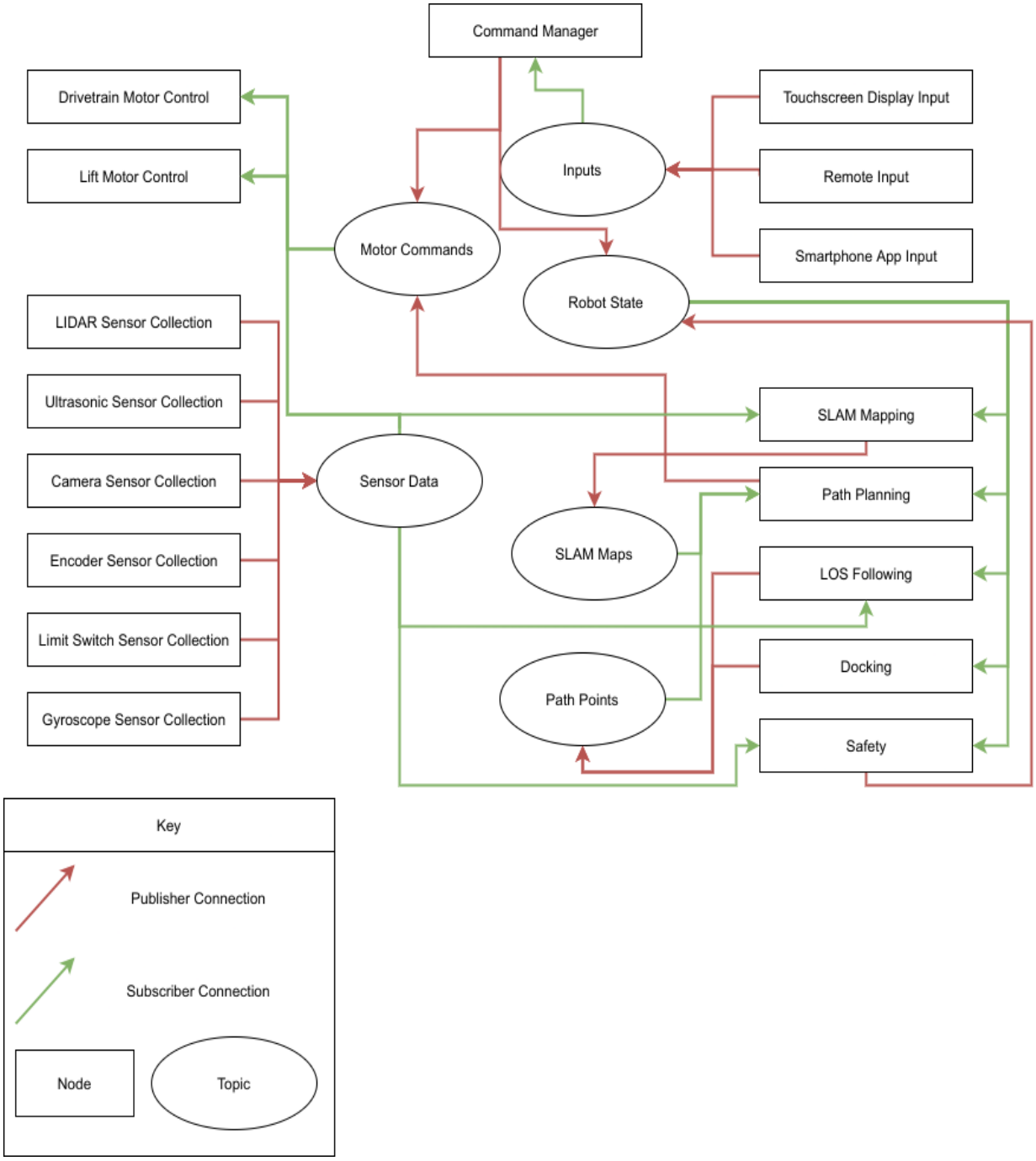



Figure 23: ROS Software Architecture Diagram

The SLAM map is only updated when moving to save resources, however. There is another node that modifies the Robot State, which is the Safety node. The Safety node takes in sensor input like encoder, ultrasonic, limit switch and gyroscope to make sure the MAC is operating within safe parameters at all times. The gyroscope is used to determine yaw and pitch of the MAC, and

will stop the MAC if it tilts too steep (like going up a steep incline) in which the load its carrying may fall over. Typically, this will be somewhere around 30 to 45 degrees. The Safety node will also measure speed using the encoders, and force the MAC to slow down or stop if it is going too fast (which may happen from going down an incline). Finally, the Safety node will ensure that the MAC does not run into anything using the ultrasonic sensors. While the ultrasonics will help with path planning, the Safety node will be in charge of abrupt obstacles appearing in the MAC's path, like people walking, pets, or other moving objects, and then stopping the MAC to wait for the obstacle to move or give the path planner time to adjust it's path around the new obstacle (since path planning isn't as quick as responding to ultrasonic sensors. We may also incorporate the camera for helping with this task, but that would require using much more resources and be more complex.

The motor controllers are controlled through either velocity commands or raw power commands to send to the motors. Velocities will be used alongside the encoders to maintain a desired speed, or if the speed doesn't need to be specific it can accept a number between -1 and 1 as how much power to send to the motor. 0 would be no power, and -1 and 1 would be reverse full power and forward full power respectively. Different nodes will be controlling the motor controllers for both the drivetrain and the lift. The drivetrain will be controlled either directly from user input, or by the Path Planning node which will send velocity commands for the MAC to follow a path to a specific position. The lift will be controlled by the user directly through the various input methods, or by the Path Planning node as well, since the lift must be down all the way before the MAC can move, which the module will ensure using the limit switch placed underneath the lift.

For the MAC to move around the home, the Path Planning node will be responsible for mapping a safe path using the SLAM Map generated by the SLAM Mapping node. The end position of the path can be set in the Path Points topic either by the Docking node (to return to the dock), the LOS Following node with the position of the detected person, or directly set by the user using the companion smartphone app. The Path Planning node is constantly checking the updated map as well, since as the MAC moves the SLAM Maps topic is constantly updated with a new map containing any new obstacles or absence of previous obstacles, in which case the Path Planning node will remap a new path based on these changes to the end goal.

## 5.7) Summary of Design

The figure below captures both the major electrical and mechanical components of our design, including their relative placement. While the specific mounting points for sensors, motors and electronics, location and size of MAC may be subject to change, we've included this section to summarize the general design of our robot. The major mechanical sub-assemblies include our chassis, scissor lift, electrical board as well as enclosure, and our interchangeable parts. Sensors found on the robot include the limit switch for our scissor lift, ultrasonic sensors at each corner of the chassis, a LiDAR sensor suspended from the front, a camera mounted at the top of the electrical board, and a gyroscope within the chassis. Electronics included at the front of the robot, which are mounted to the electrical board, include the power distribution board, three Talon SRX motor controllers, a voltage regulator module, a Jetson Nano and a kill switch. Other

notable components include the touchscreen display, which is suspended off the top of the electrical board, and a 12V battery located at the front of the chassis, below the electrical board. While other products such as our charging dock and remote are important elements which accompany the MAC, we've elected to only summarize the primary product of this project.

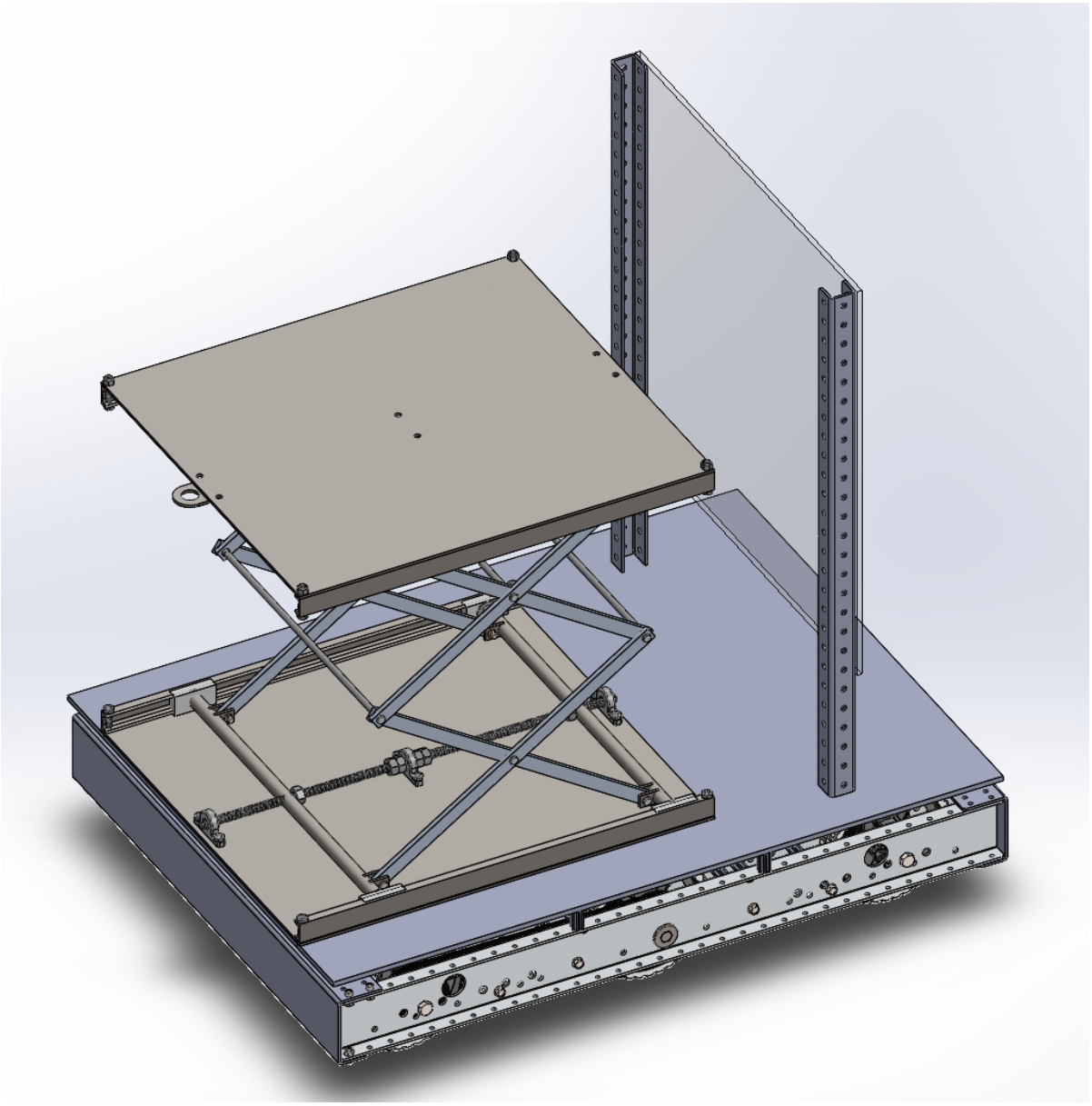


Figure 24: General Design for MAC

# 6) Project Prototype Construction and Coding

## 6.1) Needed Materials

In order to prototype various functionalities and modules of the MAC, we will require extra materials on top of the final design materials listed in section 5.1. As we are building the drivetrain, lift, and electronics storage areas we will use extra materials to prototype our designs to ensure they work before making a final iteration for the MAC. In order to do this, we need materials such as:

- Wood
    - 2 feet x 4 feet x 8 feet lumber
    - 11/32 inch x 4 feet x 8 feet of pine plywood
    - 1/2 inch x 2 feet x 4 feet sande plywood
- Aluminum Tube
    - 0.25 inches in internal diameter with a 0.014 wall thickness
- Aluminum Sheets
    - 1 inch thick
- Aluminum Bars
    - 1/2 to 1 inch thick
- Steel Bars
    - 1 inch thick
- Steel Sheets
    - 1/2 inch thick
- Steel Tube
    - 1/2 inch in diameter, wall thickness of 0.049 inch
- PVC
    - Elbow, T, Cross, and Straight connectors
    - Tubing with 1/2 inch to 1 inch diameters
- Screw / Nuts / Bolts / Washers
    - From 1/4 inch to 1/2 inch in diameter, 1/2 inch to 2 inch usable length
- Nails / Adhesives
    - Nails at 14 gauge and 1 ½ inches in length
    - Electrical Tape
    - Duct Tape
    - Scotch Tape
- Aluminum Brackets
    - Elbow, T, A, Cross, and Straight connector brackets
- Steel U Channels
    - 1 inch width, 1/2 inch height
- Bearings
    - 3/8 shaft diameter
    - 1/2 inch housing
    - Housings for bearings

- Belts
    - Rubber, 120 tooth 15 mm wide
    - Rubber, 160 tooth 15 mm wide
- Gears
    - 14 - 100 tooth
- Steel Hinges
- Lexan Sheets
    - 1/4 inch thick, 6 feet x 6 feet
- Powerpole Wire Connectors
- Wiring
    - 8 gauge black and red
    - 12 gauge black and red
    - 20 gauge green and yellow

## 6.2) Needed Facilities and Equipment

Our facilities we will be using will be both professional shops for CNC and welding, and one of our member's homes for building, prototyping, and testing. For equipment we will need a variety of power and electrical tools, as well as our laptops for programming and interfacing with the Jetson Nano. Summary:

### 6.2.1) Facilities:

- Garage of Brandon Silva's home
    - For building, prototyping, testing
- T&T CNC Design Studio LLC
    - Welding
    - CNC for wood, lexan, and aluminum

### 6.2.2) Equipment:

Since our project will require a lot of fabrication and tools to put together, we need a plethora of tools and equipment to be able to complete our design, prototype, and test. For power tools:

- Portable drills
    - 2 count, each strong enough to drill through aluminum and steel.
- Dremel
    - For grinding, sanding, and cutting PVC
- Drill press
    - Allows drilling accurate and straight holes through material much more reliability then by hand or with a portable drill
- CNC Machine
    - Manufacturing parts of our design out of wood, aluminum, and steel.
- Soldering Iron
    - For attaching components to our PCB and wiring devices together

We will also need many basic tools as we prototype and build the MAC. Some of the more important tools we will need are saws, wrenches, wire strippers/cutters/crimpers, and a variety of pliers. The MAC will need much work in all three aspects; software, electrical, and mechanical. We will be using a large garage with many tools available to be able to complete each stage of our project and testing. Some of this equipment will need to be purchased separately or rented, like the drill press, since our facilities don't have any access to this device easily.

## 6.3) PCB Vendor and Assembly

For the purposes of prototyping and final production any PCB that we create will be manufactured by JLCPCB. The reason we opt for JLC is that they offer reasonable production times at a competitive price. The build time would be around 24 hours for a 2 layer design or 4 days for up to a 6 layer design, and they are capable of trace widths as small as 5 mil. Additionally, JLCPCB can produce the stencils for any PCB that they create, which will be very useful when applying solder paste to our boards which will contain many SMT devices.

For our PCB CAD design, we will be using Eagle. Eagle is an industry standard PCB design, with a rich set of tools and options that will be more than capable of designing our PCB. Including schematic design, board layout, BOM generation, and a rich built in parts library.

Electrical components that are required for our PCB will be purchased from either Digi-Key Electronics or Mouser Electronics depending on price, availability, and shipping time. Care was taken when selecting components that the component was in stock on one of the above distributors and that a reasonable quantity of the component was available.

For the final assembly of our PCB we will be using a reflow that is available for groups to use at the Texas Instruments Innovation Lab on UCF's campus. The access to a reflow oven is critical as the majority of the components that we will be using are SMT devices that cannot be easily soldered by hand.

## 6.4) Electronic Component Packaging

The following below is a diagram and discussion of how the electrical components will be packaged within MAC. There are three main areas where all of the electronic components will be packaged. The first area is the control tower. The control tower is the front vertical structure of MAC that will have components fixed to it vertically. The components are the Jetson Nano, LiDAR, camera, gyroscope, and remote receiver. The reason for vertical is because we can are restricted in our footprint size in order to properly navigate the home environment, so the last place to go is up. The second is the base chassis. The base chassis is the metal structure that makes up the chassis and is what everything mounts too, the base chassis is low to the ground and the heaviest component of MAC. The bulky and heavy components were mounted to the base chassis which is the batteries, motor controllers, and motors. The reason for mounting the heavy components low is so that MAC is stable and the chance of tilting over is not possible.

Also the ultrasonic sensors that cover the LiDAR's blind spot will be mounted to the base chassis because it provides the best "view". The final place where electrical components will be packaged is the cargo lift. The cargo lift will have a motor and a sensor connected to it so that it can know when the lift is completely raised/lowered.

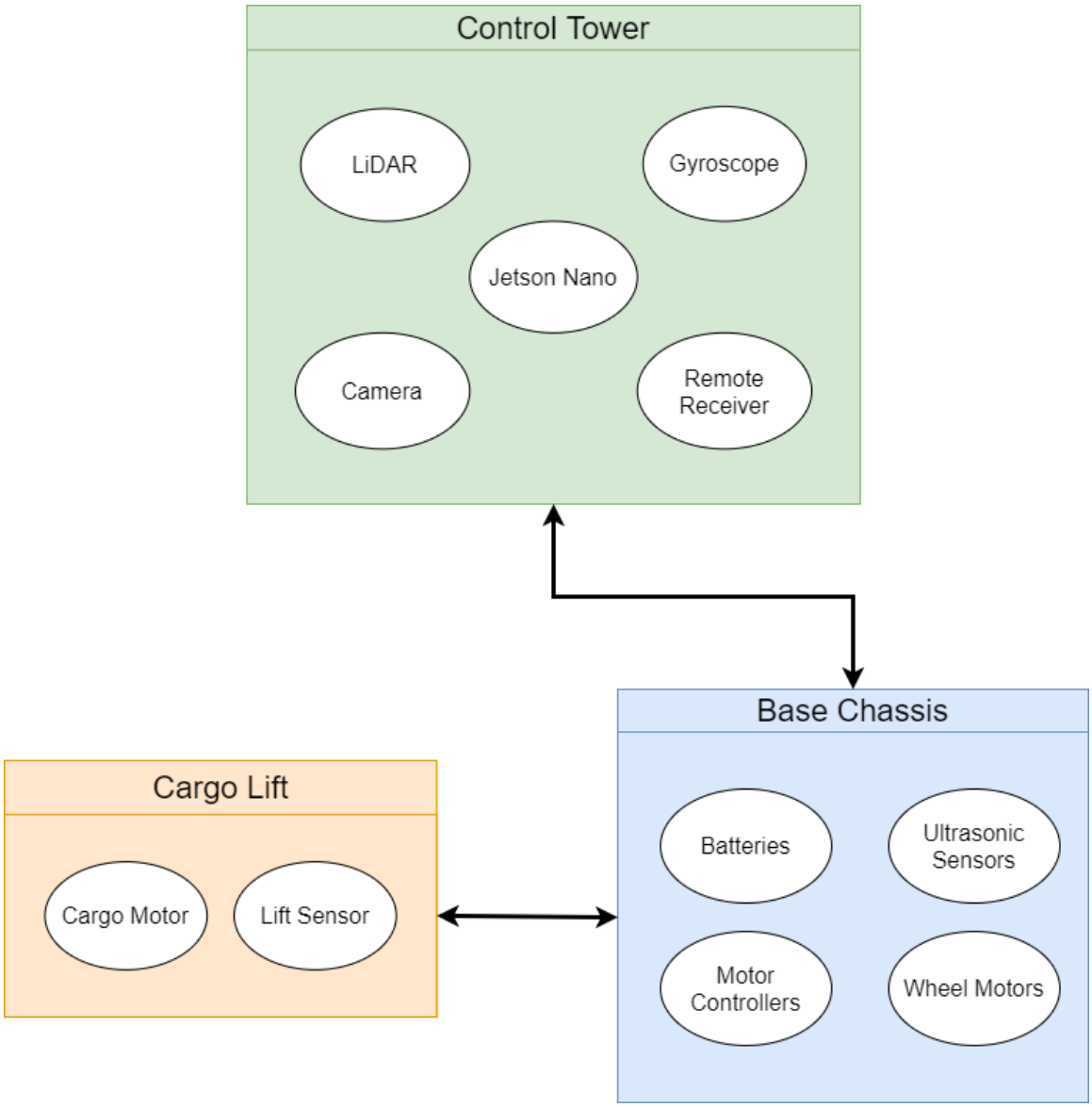



Figure 25: Electronics Component Packaging Block Diagram

## 6.5) Prototyping Platform

When prototyping we need a platform to be able to hook up motors, sensors, and various other components to run tests quickly and effectively. This platform will consist of basic electronic

components and a laptop to be able to control motor controllers and get sensor input for prototyping various functions of the MAC. We will also use a microcontroller for connecting and testing some of the sensors which will not be used on the final design, in case our Jetson Nano is unavailable or mounted in the MAC already.

The prototyping platform will be a sheet of lexan with the electronic components zipped tied to the sheet, using drilled holes to wrap the zip ties through. This will make it highly configurable, since the parts can be moved around as needed to accommodate a prototype test. It also leaves room to test out positioning of the various components, so the final board schematic for the MAC can be tested out before being mounted onto the finished frame of the MAC. These various electronic configurations will be crucial in order to fit the electronics in the smallest form factor as possible to leave room for changes in the overall design sizes, shapes, and positioning of mechanical components.

Separate test modes will be incorporated in our software to test components manually and individually as necessary, since each node is independent it can be turned on and tested without needing the others nodes hooked up or running. This will allow us to start testing our software quickly, since as nodes are created and hooked together via the topics testing can begin on those nodes without needing all the other nodes to be necessarily complete. We can also test various communication methods to make sure they will work as we expect. For example, we can test ranges of 2.4 GHz wireless signals, RF signals, Bluetooth, and connection to a WiFi access point through various obstructions, to help us see how signal strength holds up against various home configurations and possible obstructions or interruptions (like from a microwave being used).

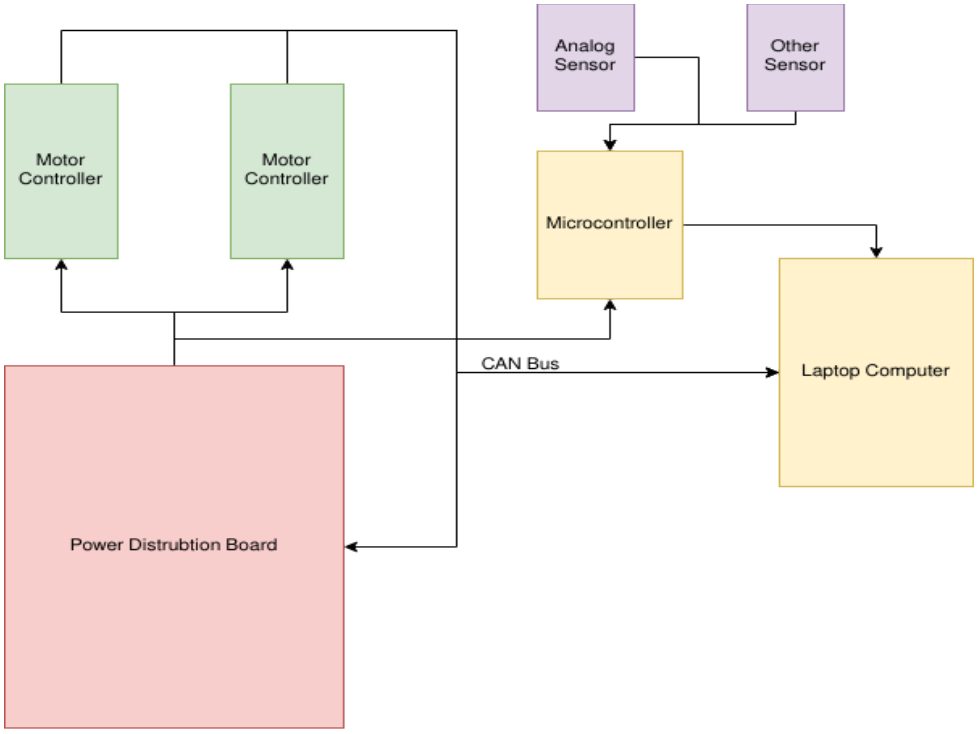


Figure 26: Prototyping Platform

The power distribution board will power two motor controllers and a microcontroller. The motor controllers can be hooked up to any motors for testing, whether it be the drivetrain, lift, or other component. The microcontroller will handle connection to sensors that do not have USB connections, like the ultrasonic sensors, if needed. The motor controllers will be wired up via CAN, which is connected to the laptop computer using a USB to CAN adapter, and terminates at a designated terminal in the power distribution board. The laptop computer will simulate the Jetson Nano in the final MAC build. The laptop computer will manually control the motor controllers, take in input from the sensors, and test out various functionalities like SLAM mapping, path planning, input from a remote (hooked directly into the laptop computer using USB), communication with the smartphone app, and any other software functionalities.

Not only will this help in prototyping, but also testing that each individual component is working to specification. Here the platform can be used to easily make sure ultrasonic, gyroscopic, LiDAR, and encoder sensors are reading properly. Motors can be tested to make sure they are working to their specified power and speed ratings, and the computer vision software can be tested to make sure it's properly detecting people and giving proper distance coordinates to a person. The prototyping platform is separate from the MAC electronics board, using spare parts we have. This is so testing and prototyping can be done in parallel to building of the MAC, where parts do not have to be removed or building has to halt in order to get the necessary parts and equipment to test some prototype or device.

Furthermore, the prototyping platform can be hooked up into a prototype drive train or MAC design to be able to test functions of multiple versions of the MAC if needed. The Jetson Nano can be used to test one build of the MAC, while in parallel another MAC prototype build can be tested with ease, given enough spare parts are available. Also, the prototyping platform itself is easily expandable. Since it is just a lexan board with zip tied electronics, the configuration of the board can be changed quickly to fit a specific prototype test.

## 6.6) Final Coding Plan

Our software can be built in an optimal way such that it is easier to test each iteration as it is being built. As mentioned before, since we are using nodes within ROS to contain much of our code, and since each node is independent, we can test node’s functionality almost independently from each other to make sure they are working as expected. In order to effectively achieve this, certain parts of the software should be done first, while others can be done in parallel at any time.

Our independent components that can be created at any point in parallel to building the main software package for the MAC, includes the YoloV3 model training and evaluation code, the LOS following node, SLAM Mapping, smartphone application, and touch screen GUI display. The YoloV3 model will need to be fine tuned to detect only people, since we do not care to detect other objects as come standard with the pre trained model. We can fine tune to just detect people, or train from scratch on only 1 class, which is people. Whichever one performs better,

this can be done outside of the main software development, in which once the model is trained the file containing the weights and structure can be dropped into the main code for deployment use. The LOS following node can also be created independently, since it can use the downloadable pretrained model from the YoloV3 website while the custom model is being trained. The LOS following node will take in camera input, calculate bounding boxes for people, then calculate the distance to the person, and transform this into an x and y coordinate for the path planner to process. This can all be developed independently of the robot, since it only relies on a camera to perform the rest of the calculations. The only thing needed from the other nodes is the current MAC's position in order to turn the distance to the person into an x and y coordinate that the Path Planner node can use. The smartphone application is completely separate, since it takes the SLAM map generated from the MAC, displays it, and allows selecting coordinates to move the MAC too. Since we can use example maps to test functionality, this can be done independently of the main software development. Finally, the touchscreen display will require some software to build it and provide functionality to each of the buttons and menus. Since the GUI itself doesn't rely on the main software, it can be built independently. Once the main software is complete the GUI can be linked to call specific functions when buttons are pressed on the touch screen.

For the main software package, it should be built in a way to maximize testing potential and extendibility for other nodes to build on top of our base code. The first nodes that will be built are all of the nodes to collect data from sensors, including the LiDAR, ultrasonics, camera, encoder, limit switch, and gyroscope. These will then all publish to a sensor topic for other nodes to subscribe to.

Once this is done, the command mangage can be built to manage robot states, but can just manually change states instead of relying on input in the early iterations. Then once this is done the drivetrain and lift motor controller nodes will be created to provide control to those components. Both of these will be extremely similar in structure. With these the motor commands and robot state topics can be created to publish and subscribe to these nodes. Once that is done, testing can ensue to make sure the sensors are publishing properly, motors can receive commands, and the robot state can be adjusted.

Once the base sensor collection and control are built, the SLAM mapping node should be done. With this, the path planning node can be built and tested to make sure it can probably plot a path and move to the path when manually given a location. This can be done in Gazebo with a simulation, and on the MAC prototyping bed if available. Then the path points topic can be created to allow other nodes to send positions to the Path Planning node. With this created, the LOS following can be finished in parallel and tested to make sure it is outputting correct positions to the planner. Once this is done then the final nodes can be added since the core functionality of the MAC is completed.

Once the core functionality is completed, the input nodes can be created along with the Inputs topic to be able to send commands to the command manager instead of sending them manually using a terminal. With the command manager being able to accept commands from these various

sources successfully, the safety node can be created as an extra layer of protection for the user and their home from the MAC. The Safety node will hook into the core functionalities and monitor sensor input, and stop when needed to ensure safety of the user and the MAC. It will run in the background constantly, so it is only needed once everything else is working. If we have enough time to build the dock, then the Docking node will be the final node that is created as a part of our main software package. The Docking node will use the path planning infrastructure alongside detection of when the MAC is charging. When the MAC is charging at it's dock, the Docking node will remember the position of the dock. When the user enters the command to dock the MAC, the MAC will plot a path to the dock by use of the Path Points topic and Path Planner, powered by the Dock node. Then the Dock node will take control when the dock is reached, following a predefined set of commands to line up the MAC with the dock using the gyroscope, encoders, and ultrasonics. Once lined up, the MAC will back up until it detects it is charging, meaning that the docking was successful. In the following figure below, the coding plan is laid out.

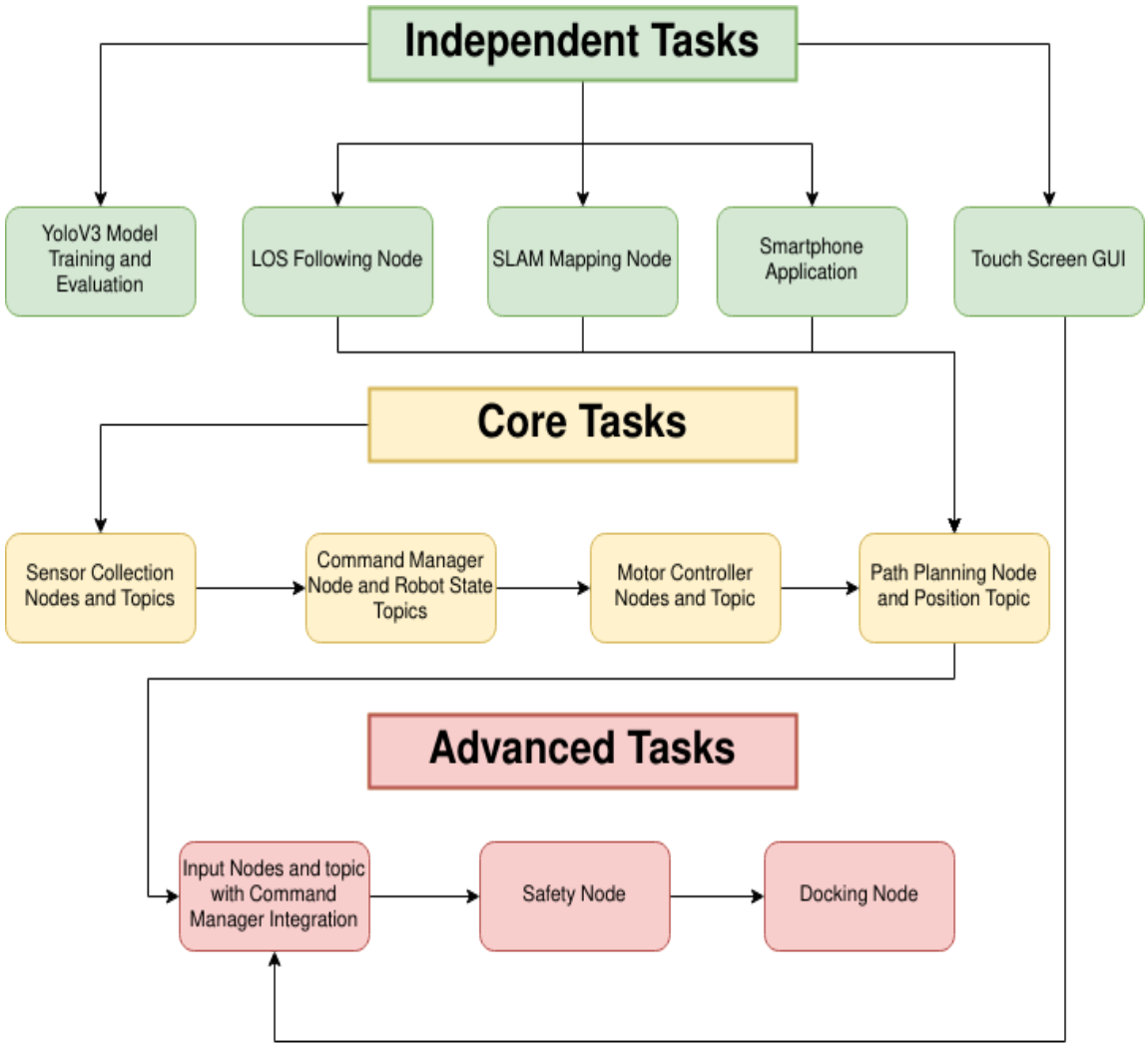



Figure 27: Final Code Plan

# 6.7) Prototyping Plan

## 6.7.1) MAC

### 6.7.1.1) Chassis

Two chassis will be used for prototyping of the MAC: the Nanotube and the AM14U4. A rough model based on the dimension range specified will be built prior to working with either of the mostly pre-built chassis, made out of wood. This will be done to confirm that measurements on length and width are appropriate for the space needed by both the scissor lift/usable space and the electronics. Necessary adjustments will be made on the wooden model as we evaluate how much space it takes up as well as how much space it has to work with in various-sized coordinators and entryways.

Following finalization of the perimeter dimensions, we will modify both the Nanotube and AM14U4 chassis so that they match the desired dimensions. The Nanotube chassis will act as a test bed to test electronics and sensors on, whereas the AM14U4 chassis will continue on as the chassis for the final product. A piece of plywood will be cut to shape, large enough to hold the necessary power distribution board, voltage regulator, Jetson Nano, and other electronic components to control as well as drive the robot. For the AM14U4, several factors need to be tested during prototyping: clearance on ramp angles, battery placement and number of motors.

Several ramps with differing degrees of incline will be used, each of which will reflect standards for how steep ramps for disabled or wheel-chair bound individuals are. The ramps will be made out of wood, reinforced by triangular wedges on the top and bottom. The chassis will be tested to see if the front u-channel (front face of the robot) will clear the ground as it changes from a flat surface to an incline. In the event that the chassis can not clear the immediate change from flat surface to incline, the chassis will need to be modified and reinforced so that a front u-channel piece is not necessary to hold the chassis together.

For the battery placement, two mounts will be made within the prototype AM14U4 chassis for both the front and back spaces within the chassis frame, each identical and designed to hold the 12V battery. A solid weight of 30lb will be placed on the back half of the chassis, along the top, in order to imitate the weight of the scissor lift. The prototype electronics board will be mounted on top of the chassis, with the system being teleoperated. Tests will be conducted to see how well the robot traverses inclines and maneuvers with the battery placed either in the front or back, with evaluations focused on center of mass. Based on the results, the battery will be mounted either front or back going forward.

Lastly, the MAC must fall within specifications for the desired speed we set, where the number of CIM motors used to drive the chassis will affect how fast it can go. A determination will be made on whether one or two CIM motors on each drive train side will be necessary based on speed tests up inclines, maneuvering around objects, and traversing opening areas. We will

further test the speed of the robot by adding up to 100lbs of extra weight (not including the 30lb weight representing the scissor lift or weight of the battery).

After testing these factors, an upper plate made out of aluminum will be machined and mounted as a foundation for the electronics, lift and interchangeable parts to be mounted to. This upper plate must also take into account the cut-out necessary to easily replace or remove the 12V battery. Other tests on durability, speed and functionality of the prototype chassis may be performed, but upon completion will yield the project's final-product chassis. The chassis upon completion of prototyping will act as the base for other mechanical and electrical subsystems to be tested on.

#### 6.7.1.2) Scissor Lift

Given that the scissor lift design has been chosen for our mechanism to lift, the initial prototype will be built out of aluminum scrap rather than wood. Upon assembly of the initial prototype, the design and details of which can be found earlier in the document, testing will be evaluating the robustness of the design, concerning factors such as: load tolerance, horizontal sway, and speed of lifting.

In order to evaluate load tolerance, our screw-drive will be driven via manual crank, testing the physical limitations of the lift using various weights, ranging from 10lbs to 100lbs. Tests will study and determine if the conceptual prototype can support the weight at various heights over certain durations of time, from being completely extended to completely collapsed. Given the performance of the scissor lift under various loads, we will either reinforce the cross section which is driven by the screw, or use a sturdier material (thicker aluminum or steel).

The initial prototype for the lift will be mounted to the completed chassis for further testing. From here, horizontal forces will be applied to test how much horizontal sway or give takes place on the scissor lift. Depending on the amount of movement in the horizontal direction, we may use sturdier material for the cross sections or reduce the tolerances at various mounting points for the lift. Tests will continue until minimal horizontal movement occurs, both while the scissor lift is extended and the robot is driving but also while loaded under the same conditions.

After testing load tolerance and horizontal sway, a gearbox with Mini-CIM motor assembly will be attached to the scissor lift which is now mounted to the robot chassis. A motor controller will be added to the prototype electronics board in order to control the motor. Various gear ratios within the gearbox will be tested in order to find the proper torque to speed ratio based on design requirements that we set forth. Once the ratio has been determined, the gearbox will be mounted to the upper plate of the chassis. Upon passing the tests above, a final product will be ready to be built.

#### 6.7.1.3) Interchangeable Parts

Wooden models for the tiered tray and basket will be built, using PVC as the legs for each of the parts. Prototyping for these two components will consist of finding the proper dimensions for

these two parts, identifying the best places to mount them, and establishing how to easily remove the pieces.

The tiered tray will be built by using multiple sheets of plywood with surrounding 2x4 to act as the frame. PVC will then be measured and cut to the desired height, the restrictions of which are specified in the design requirements. Similarly, the basket will be made using wood, forming a bucket to place the goods in, surrounded by 2x4 with legs made out of PVC. Slots for the PVC will be made with larger PVC pipes, fitted so that the PVC legs snuggly fit but are loose enough to slide out if pulled up on. The interchangeable parts should be easily removed, requiring an upward force to be applied to lift them out of the slots. The spacing of the legs and subsequent slots for the legs to sit in should be measured with respect to the chassis' upper plate, leaving clearance for the scissor lift assembly and electronics section.

Once the dimensions and shapes for the interchangeable parts and mounting points are determined, the load capacity will need to be evaluated. Various distributions of weight from 10lbs to 100lbs will be tested on each part. In the event that the tiered tray or basket can't support the weight, additional reinforcements will be made using aluminum or other wooden supports. Once all the necessary supports have been determined, final interchangeable parts will be ready to be made out of plastic and PVC. These parts will be machined from the CNC table to exact specifications.

#### 6.7.1.4) Electronics Layout

The prototype electronics board will now be replaced with a newly prototyped board, which will mount vertically at the front of the robot, on top of the chassis' upper plate. 2x4 pieces of wood will be used and act as the stands for the electronics board, a new larger piece of plywood cut to mount across the vertical pieces of 2x4. Measurements will be taken prior in order to determine how much clearance is necessary for the battery to be removed. This may either result in shortening or increasing the length of the vertical support pillars to lower or raise the electronics board for clearance. This new board will then have the Jetson Nano, voltage regulation module, power distribution board, motor controllers and emergency stop switch mounted to it.

Fasteners for wiring of electronic components will be installed onto the wooden prototype to confirm the measurements and organization of the wires themselves, with modifications to the board layout being made as necessary. All wiring at this point will be redone with wiring cut to the proper length for the final design. Before determining the final mounting position of the board onto the robot, testing will be done to confirm that the interchangeable parts and lift clear the electronics board. Once the dimensions have been confirmed and the components are spaced out properly, the entire board with supports will temporarily be mounted onto the robot.

#### 6.7.1.5) Sensor and Display Placement

Building off the finalized chassis, prototyped scissor lift and prototyped electronics board, the following sensors will need to have their mounting positions confirmed:

- LiDAR (1)

- Ultrasonics (4)
- Camera (1)
- Encoders (3)
- Gyroscope (1)
- Limit Switch (1)
- Touchscreen Display (1)

Starting with LiDAR, the sensor must be mounted at the front of the chassis, slightly offset from the robot so that it isn't flush with the robot's frame. This requires the sensor to be both suspended and protected ahead of the electronics board. To accomplish this, an initial prototype will consist of a wooden foundation which the LiDAR can mount to, sticking out approximately 2-4 inches from the front of the chassis, temporarily mounted to the top plate of the chassis. In order to protect the LiDAR sensor, which detects at the horizontal range which the sensor is placed, another piece of wood will be placed above it in order to act as a bumper or barrier so that the sensor doesn't directly collide with other objects in the environment. It is important in the prototype that the horizontal 360 degree view range is not obstructed, so supports/bumpers on the top and the bottom will be the only points at which the LiDAR sensor mounts to. From here, tests will be run to confirm that the LiDAR sensor can detect the majority of its environment, adjustments to the mounting position being made as necessary, either suspending the LiDAR sensor further off the front of the robot or moving it closer into the frame. Several tests will be performed to test the durability and robustness of the physical barriers preventing the LiDAR from coming into contact with the environment directly (as to avoid damaging the sensor). Changes to the mounting points may be made with respect to protecting the sensor. Once tests and evaluations have confirmed that the sensor is both protected and maximally effective in its position, official mounts will be machined for the final design.

The ultrasonic sensors will be mounted as low as possible, two positions being viable given the design of the robot: suspended below the frame of chassis but off the ground by only a couple inches or above the top of the chassis frame but below the top plate on the chassis. In either position, four sensors will be required in order to detect the perimeter effectively around the robot, placed at each corner facing outward at a 45 degree angle. The sensors will be flush with the frame of the chassis to protect them from coming into contact with the environment. Ideally, the sensors would be mounted beneath the chassis, but tests will be run to determine if mounting the ultrasonic sensors is actually viable, such as clearance tests for the robot going up inclines. If these tests fail, then the only viable position will be on top of the chassis. From here, temporary mounts will be made and the sensors will be put under various tests in order to guarantee that the robot doesn't come into contact with the environment, stopping before making contact at the distances we specify via code.

One camera will be mounted above the electronics board, facing out from the front of the robot. It is necessary for the camera to be mounted as high as possible, with its cone of vision unobstructed from any other systems on the robot. Prototyping here will consist of various mounting brackets, with stability and reduction of vibration being the primary concerns. The camera will be encased and secured so that as the robot moves, the camera is stationary. Too

much vibration or movement could throw off the computer vision algorithms which detect objects in the environment, reducing the effectiveness and overall usefulness of the camera. If a mounting style proves to be ineffective, it will be iterated upon until the camera is secure, where minimal to no disruptions to object detection occur while the robot is in movement. Once the mount for the camera has been finalized, a final mount will be machined with proper mounting points to the electronics board.

Three encoders will be necessary to calculate the absolute displacement of the robot from its starting or previous position. Two will be attached to the gearboxes of the drive trains, where the gearbox assemblies include an exposed shaft from the motor to attach them to. From here, the encoders will take measurements of each side of the drivetrain. The third encoder will be mounted to the exposed shaft of the gearbox which drives the scissor lift. In the event that a drop in power supplied happens, speed of the robot and the scissor lift can be maintained. Tests will be performed to make sure that the encoders work as intended when attached to these gearboxes. Our programs will confirm that the absolute position is being calculated correctly, and once confirmed, no final adjustments will be necessary for the finished project aside from their communication or relationship to other sensor input. Considering there is no other place to mount these encoders, no additional locations for mounting are necessary during the prototyping phase.

The gyroscope will be mounted at the center of the robot, within the frame of the chassis. During prototyping, a mount will have to be made to stabilize and secure the gyroscope between the CIM motors of the drivetrain. Testing of the gyroscope will then determine if the gyroscope is mounted securely, with adjustments being made based on feedback received by it. Various tests will be performed on inclines and while the robot is in motion to confirm that it is working properly, using it to perform hard stops on the robot's motion based on exceeding a certain incline and the like. Once the gyroscope is located and secured in the ideal location, an actual mount out of aluminum will be machined for it.

A limit switch will be used underneath of the scissor lift as a physical, mechanical stop when the lift has fully retracted back to its starting position. This will ensure that the motor doesn't continue to run despite the scissor lift reaching its desired return point. Tests will be performed to determine if the placement of the limit switch will subject it to damage, such as whether or not additional weight on the scissor lift will cause added pressure onto the mechanical limit switch. A number of places would be viable for the limit switch, so its final position through prototyping may be determined based on the efficiency of how it is wired (where the wires will be pathed back to the electronics board) as well as where it is the least exposed to any external forces other than contact from the scissor lift itself. In order to guarantee this, tests will be performed to see if it is possible to make contact with the scissor lift by other objects being placed on the robot's usable space, causing unintended behavior (the switch being pressed when it shouldn't be, causing the scissor lift motor to stop running prematurely). As such, a small wooden enclosure may be built during prototyping to prevent outside contact other than the scissor lift. If an enclosure such as this is necessary, a final piece may be machined from aluminum for the final design.

The touchscreen display, a necessary part of the robot's design that should be accessible to the user, will be mounted above the vertical electrical board at a similar height to the camera, but not in a position where it may obstruct the view of the camera. Several positions may prove to be viable, such as off either side of the robot. Prototyping of the functionality of the display may be extensive, but most importantly it must be secured so that any force applied to it would not cause it to move. Similar to the sensors discussed above, various mounting styles will be tested to determine which is most secure.

Though not discussed in this section, extensive testing of the functionality as well as communication of the various sensors and display will be performed as well during prototyping. Results of these tests may affect the placement of these sensors as well as how they are layered programmatically, but the overall concern during the mechanical prototyping is that the sensors and display are unobstructed, secured properly and perform as intended.

### 6.7.2) Charging Dock

The charging dock, essentially a station for MAC to return to and charge once reaching low power, includes several features to test during prototyping: load tolerance to support the weight of MAC, ability to line up with the charging plug, and the reliability of the transmitter so that MAC knows the location of the charging dock. The structure of the charging dock will be built from wood, whereas the final product may feature some additional material such as plastics or aluminum for extra security and aesthetic.

Starting with the structure, the dock will be a slightly raised plywood sheet, braced underneath by full lengths of 2x4 wood. The added support underneath is so that upwards of 250lbs can be supported by the dock, which includes the weight of MAC plus any goods it may be carrying. Surrounding the dock will be wooden guard rails, the entire structure not any taller than two to three feet. The front of the dock will be exposed, with an inclined ramp leading up into it. Additional guard rails may be placed at the sides of the ramp so that MAC will not fall off the side or miss the entrance. The inclined ramp will not be very steep, only extending in length to about six to eight inches to the ground. The inner dimensions of the dock must be enough to contain the robot, but not so large that it has significant flexibility of movement within it (to be discussed on lining up the charging plug below). Tests will be performed so that all of these specifications are met, and that it can support the weight of MAC overall.

The plug for charging must be positioned in such a way that MAC is able to easily line up and connect without intervention from the user. As such, an automatic feature to disconnect MAC's battery from the rest of the robot and connect solely to the charger must be available. Using a splitter and a switch, the circuit can be cut off and the battery can be disconnected. To focus on the design of the charging port, an Anderson connector may be used as the plug which MAC lines up with. Prototyping may prove that making this connection may be more difficult than anticipated, so other plugs will be experimented with. The placement of this plug must line up with the external plug for charging on the robot, which will be mounted at the front of the

chassis. The plug on the dock must be secured so that force applied from the robot would not cause it to come loose, so mounting strategies for the plug are subject to iteration.

Prototyping to test MAC's ability to identify the location of the dock must be tested, which is to be done after the completion of prototyping for the entire robot. Failure to identify the dock or accurately locate it so that it may traverse into the dock will require additional testing or a rethink of how to detect the charging dock altogether. More explanation of this feature can be found below in the testing and evaluation plans.

### 6.7.3) Remote

The initial remote enclosure during prototyping will be built out of wood, with two viable design options: nine cut-outs for buttons or five cut-outs for buttons and one for a joystick. Prototyping here will be concerned with the placement of the specific button functions (based on how intuitive the design is) and the connection of the remote. The following functions are represented via the remote: lift up, lift down, forward, reverse, turn left, turn right, go home, LOS follow and stop.

For the first of the two options, nine buttons would be laid out along the top of the remote. Forward, reverse, turn left and turn right would be grouped on the right side of the remote, in a plus-formation. Lift up and lift down would be to the left, lift down being directly under lift up. Go home, LOS follow and stop would all be to the leftmost side of the controller. For the other design option, the only major difference would be that the plus-formation for movement of the robot would be replaced with a joystick, allowing for more natural control of the robot. In both cases, the buttons would be labeled clearly on the final remote, but for the prototype this is not necessary. Some of the button placements may be subject to change based on how each control the robot is, and related testing will be done to determine whether or not there is a better layout.

Additional testing will be performed to test the wireless communication between the remote and the receiver on the Jetson Nano, which is connected into one of the available USBs. The wireless connection should reach well up to 80 meters in various environments, so any issues presented with this during prototyping may require re-evaluation of the method used for communication between the remote and the robot. Once the remote functionality has been tested and we have determined that the button placement is efficient, a final remote will be ready to be made.

### 6.7.4) Schedule

The following table and figure below capture the Gantt chart as a detailed list (specifying details such as start date, end date, duration, and assignment) and the accompanying visual representation of the data, respectively. In order to effectively prototype all of the various mechanical subsystems discussed prior, it was necessary to break them down by category, specifying how long each section would take. Notably, some of the subsystems will end up taking more time based on size, complexity and effort to construct. Additionally, construction and testing phases for each subsystem have been divided into their own sections, the granularization of which will be beneficial when measuring our progress. Some of our

subsystems will be constructed and tested either at the same time or at overlapping times with others, to which we've made these decisions based on factors such as how coupled or dependent a subsystem is on another, or whose available during those time periods to work on another component of the project.

## Prototyping Gantt Chart

| TASK NAME | START DATE | DAY | END DATE | DURATION (WORK DAYS) | DAYS COMPLETE | DAYS REMAINING | TEAM MEMBER | PERCENT COMPLETE |
|---|---|---|---|---|---|---|---|---|
| **MAC** | | | | | | | | |
| Chassis Construction | 5/17 | 0 | 5/20 | 4 | 0 | 4 | Justin, Brandon | 0% |
| Chassis Testing | 5/21 | 4 | 5/24 | 4 | 0 | 4 | Justin, Brandon | 0% |
| Scissor Lift Construction | 5/25 | 8 | 5/27 | 3 | 0 | 3 | Justin, Brandon | 0% |
| Scissor Lift Testing | 5/28 | 11 | 5/30 | 3 | 0 | 3 | Justin, Brandon | 0% |
| Electrical Board Creation | 5/17 | 0 | 5/19 | 3 | 0 | 3 | Brandon, Trevor | 0% |
| Electrical Board Testing | 5/20 | 3 | 5/23 | 4 | 0 | 4 | Brandon, Trevor | 0% |
| Interchangeable Parts Construction | 5/31 | 14 | 6/2 | 3 | 0 | 3 | Justin | 0% |
| Interchangeable Parts Testing | 6/3 | 17 | 6/4 | 2 | 0 | 2 | Justin | 0% |
| On-Board Electronics Construction | 6/5 | 19 | 6/7 | 3 | 0 | 3 | Justin, Brandon, Trevor | 0% |
| On-Board Electronics Testing | 6/8 | 22 | 6/10 | 3 | 0 | 3 | Justin, Brandon, Trevor | 0% |
| Sensor and Display Installation | 6/11 | 25 | 6/14 | 4 | 0 | 4 | Justin, Brandon, Trevor, Lisi | 0% |
| Sensor and Display Testing | 6/15 | 29 | 6/28 | 14 | 0 | 14 | Justin, Brandon, Trevor, Lisi | 0% |
| Final Prototype Testing | 6/30 | 43 | 7/2 | 3 | 0 | 3 | Justin, Brandon, Trevor, Lisi | 0% |
| **Charging Dock** | | | | | | | | |
| Dock Construction | 5/24 | 6 | 5/28 | 5 | 0 | 5 | Justin, Trevor | 0% |
| Dock Testing | 6/15 | 29 | 6/17 | 3 | 0 | 3 | Justin, Trevor | 0% |
| **Remote** | | | | | | | | |
| Remote Construction | 5/17 | 0 | 5/28 | 12 | 0 | 12 | Trevor, Brandon, Lisi | 0% |
| Remote Testing | 6/15 | 29 | 6/21 | 7 | 0 | 7 | Trevor, Brandon, Lisi | 0% |

Table 11: Prototyping Gantt Chart Data

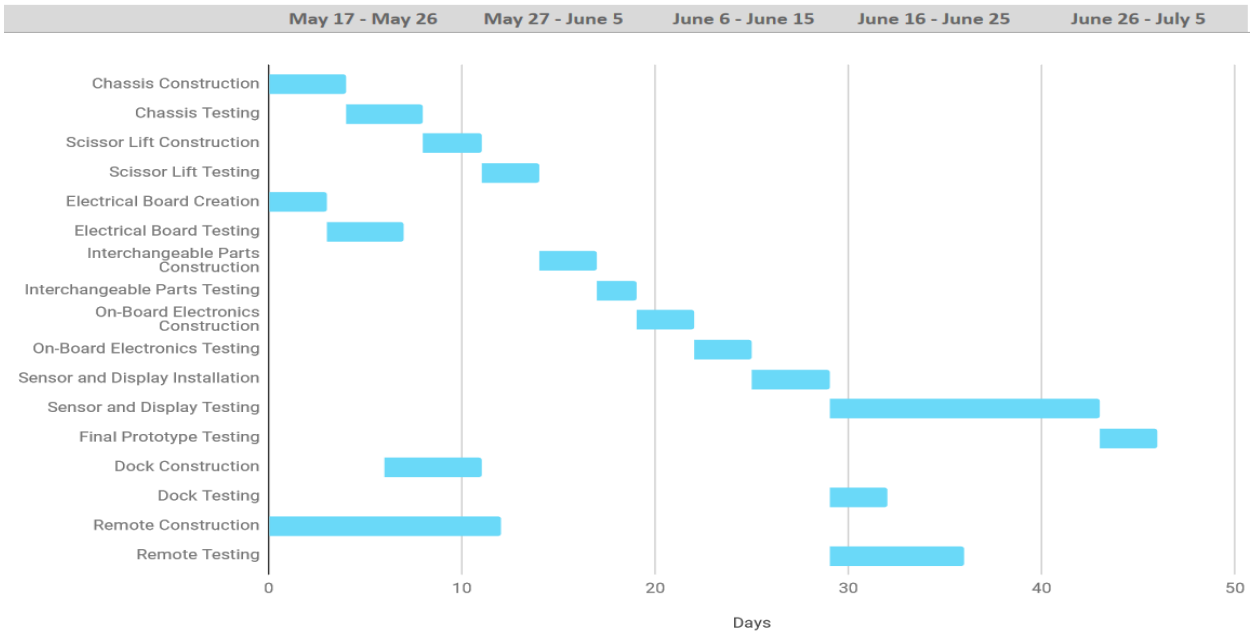



Figure 28: Prototyping Gantt Chart

# 6.8) Build Plan

## 6.8.1) MAC

The following section details, in order, how the mechanical subsystems of the robot will be integrated, with any finishing touches or additions being made as determined by the prototyping section.

Based on the prototyped AM14U4 chassis, several supporting plates, including the top chassis plate, must be machined with specific mounting holes for sensors such as the ultrasonics and gyroscope. Mounting points must also be machined for the vertical supports holding the electrical board as well as the gearbox/motor assembly and scissor lift assembly. Some churros which act as stands for the top chassis plate may need to be lengthened in order to provide clearance for the ultrasonic sensors, resulting in new churros needing to be cut and installed. Once these holes are machined and the appropriate adjustments to the churros are made (if necessary), the chassis can be completely assembled.

Moving on from the base of the robot, the scissor lift will be built using a combination of aluminum and steel, iterating on the prototyped model by adding extra structural integrity. The additional weight of these materials will be accounted for during testing in the prototype section so that the gearbox and motor assembly can support it. Otherwise, the final scissor lift assembly will be a close replica to the prototype. Additional traction material may be placed along the top

platform of the scissor lift to better grip materials and objects it supports. Upon assembly of the final lift, the entire subassembly will be mounted onto the chassis' top plate.

Final interchangeable parts, specifically the tiered trays and basket, will be made out of appropriate plastics, reinforced with aluminum. Additional weight testing will be performed to confirm that the new materials (replaced from the original wooden ones) can support the same as what was tested for during prototyping. The same PVC legs will be used and fastened to these new parts, with the accompanying slots for easy swapping mounted to the specified locations on the chassis' top plate.

The vertical supports for the electrical board and accompanying mounts will be installed onto the robot next, the supports made out of c-channel aluminum. The electrical board itself will be made from lexan, machined with the specific mounting holes for all of the necessary electronics as well as for fasteners to hold down the wires. The back of the electrical board will be reinforced with thin aluminum strips in order to make the board rigid (avoiding any flexing that may come with movement). Based on the measurements from prototyping, the electrical board will be mounted at the appropriate height to allow for clearance of the battery below.

The LiDAR sensor mounts will be made out of aluminum, providing the protection necessary on the top and bottom of the sensor. Based on the wooden mounts, the same mounting points will be made. The ultrasonic sensors will be installed close to the same as they were in the prototyped version. The camera mount will also be remade with aluminum, with a horizontal aluminum plate providing cross-sectional support between the two vertical supports for the electrical board. The camera mount will be mounted at the center of this aluminum plate, securing it above the electronics. As the encoders were already installed during prototyping, no modifications will be necessary for the final build. The machined gyroscope mount will be attached via the cross-sectional supports within the chassis, suspending the gyroscope between the motors of the drivetrain. The limit switch mounting points and machined aluminum enclosure will be added after to the appropriate spot designated via testing. Lastly, the touchscreen display with mount will be mounted to the location determined via prototype testing. Once all of the sensors are in place, all wiring of the sensors will be measured and cut to exact length, with additional fasteners being installed to clean up the wiring and secure them.

### 6.8.2) Charging Dock

Based on the prototype model, parts of the final charging dock may be rebuilt, but the entire enclosure will be made mainly with wood and spray painted for aesthetic effect. Minor adjustments and reinforcements will be made, including aluminum reinforcements and plastic guard rails to support the weight of the MAC better and protect the robot as well as its sensors from damage, respectively. A final mounting bracket for the charging plug will be made out of aluminum, machined neatly as to guarantee the charging plug is secure.

### 6.8.3) Remote

For the final remote, the original wooden enclosure in the prototype model will be 3D printed, with the appropriate holes and cut-outs for each of the buttons/joysticks necessary for all nine operations specified. The enclosure will be 3D printed so that the front and back may be separated, allowing for the installation of the buttons/electronics necessary to make the remote work. These components will be recycled from the prototype model, saving the cost of having to buy two sets of buttons and controllers for the device.

### 6.8.4) Schedule

The following table and figure below capture the Gantt chart as a detailed list (specifying details such as start date, end date, duration, and assignment) and the accompanying visual representation of the data, respectively. In order to control our build schedule, which is necessarily quick in order to complete the project before the end of July, it was necessary to break each of the final mechanical subsystems down by category, specifying how long each section would take. Notably, some of the subsystems will end up taking more time based on size, complexity and effort to construct. Some of our subsystems will be constructed either at the same time or at overlapping times with others, to which we've made these decisions based on factors such as how coupled or dependent a subsystem is on another, or whose available during those time periods to work on another component of the project. In the event that any last problems or issues are noticed at the end of the build cycle, time has been allocated for any additional modification as well as final testing of MAC. We anticipate that an additional design cycle will take no more than a week's time, as the majority of any mechanical or physical issues will be addressed during the prototyping phase.

Build Gantt Chart

| TASK NAME | START DATE | DAY | END DATE | DURATION (WORK DAYS) | DAYS COMPLETE | DAYS REMAINING | TEAM MEMBER | PERCENT COMPLETE |
|---|---|---|---|---|---|---|---|---|
| **MAC** | | | | | | | | |
| Chassis Construction | 7/3 | 0 | 7/4 | 2 | 0 | 2 | Justin, Brandon | 0% |
| Scissor Lift Construction | 7/5 | 2 | 7/8 | 4 | 0 | 4 | Justin, Brandon | 0% |
| Interchangeable Parts Construction | 7/9 | 5 | 7/11 | 3 | 0 | 3 | Justin | 0% |
| On-Board Electronics Construction | 7/5 | 2 | 7/7 | 3 | 0 | 3 | Justin, Brandon, Trevor | 0% |
| Sensor and Display Installation | 7/8 | 5 | 7/9 | 2 | 0 | 2 | Justin, Brandon, Trevor, Lisi | 0% |
| Build Testing | 7/10 | 7 | 7/12 | 3 | 0 | 3 | Justin, Brandon, Trevor, Lisi | 0% |
| Build Modification | 7/13 | 10 | 7/16 | 4 | 0 | 4 | Justin, Brandon, Trevor, Lisi | 0% |
| Final Build Testing | 7/17 | 14 | 7/19 | 3 | 0 | 3 | Justin, Brandon, Trevor, Lisi | 0% |
| **Charging Dock** | | | | | | | | |
| Final Dock Construction | 7/9 | 5 | 7/10 | 2 | 0 | 2 | Justin, Trevor | 0% |
| **Remote** | | | | | | | | |
| Final Remote Construction | 7/3 | 0 | 7/7 | 5 | 0 | 5 | Trevor, Brandon, Lisi | 0% |

Table 12: Build Gantt Chart Data

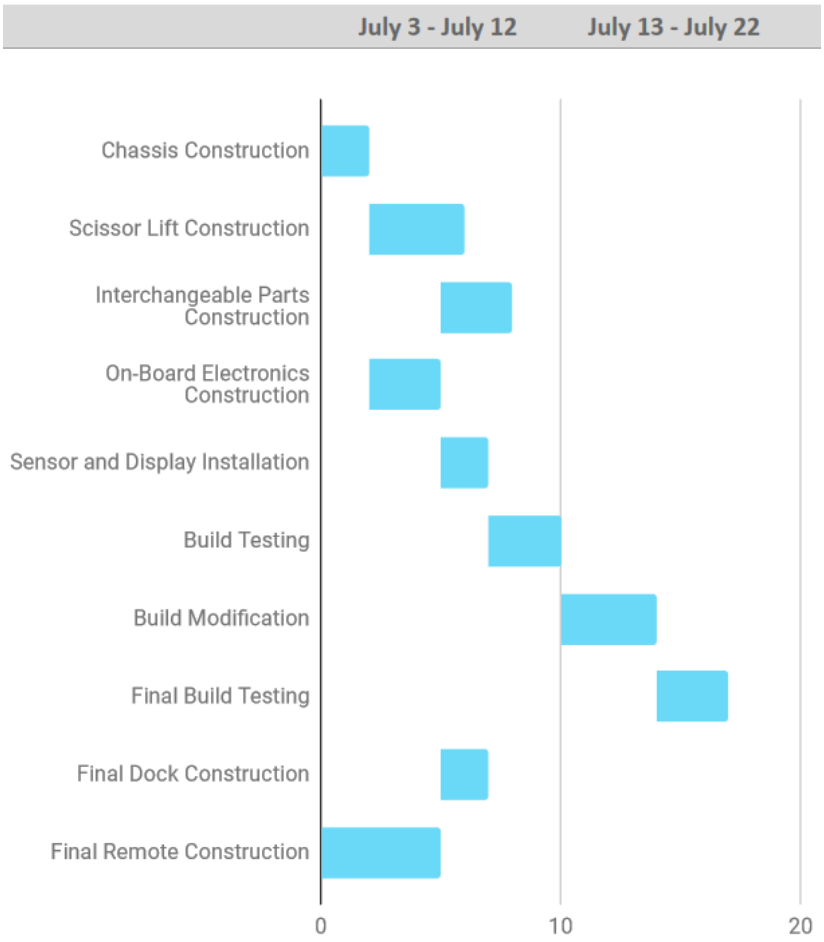


Figure 29: Build Gantt Chart

# 6.9) Consultants, Subcontractors, and Suppliers

## 6.9.1) Consultants

While there are no official consultants for the official planning, design, prototyping or construction of the project, we did elicit the technical expertise for equipment as well as materials selection from the following individuals:

1. Bartholomew Nash
2. Mark Thibodeau

Bartholomew Nash offered recommendations for certain materials, sensors and electronics to use on MAC, being a current coach for FIRST (For Inspiration and Recognition of Science and Technology) Robotics and previous coach for other robotics programs such as VEX. Through his advice, we solidified some of our choices for equipment such as motor controllers, voltage regulator modules, power distribution boards, encoders, limit switches, chassis hardware and gearboxes. Mark Thibodeau offered us expertise on making use of TIG welders and CNC tables,

both of which he has extensive experience with. Given that these are tools we are making use of throughout the construction of the project, his guidance for using those tools safely and effectively is highly valuable.

### 6.9.2) Subcontractors

No subcontractors were made use of for this project, as all of the mechanical and electrical subsystems were already bought as a packaged, easily-assembled system or designed as well as built by us.

### 6.9.3) Suppliers

The following suppliers for materials, equipment, tools and services were used for this project:

1. Lake Minneola High School Robotics Team - Bartholomew Nash
    a. 101 N Hancock Rd, Minneola, FL 34715
2. T&T CNC Design Studio LLC - Mark Thibodeau
    a. 3020 County Rd 48, Groveland, FL 34736

Lake Minneola High School (LMHS) Robotics Team offered the following materials to be used for free throughout the duration of our project, to be returned upon completion of the project:

- Nanotube Chassis, with full assembly (wheels, gear train, gear boxes, etc.) [97]
- AM14U4 Chassis, with full assembly [98]
- Talon SRX motor controllers (6)
- Additional gear trains and boxes with varying ratios
- 2.5 in CIM motors
- Mini-CIM motors
- PG71 motors
- 12V lead acid batteries (2)
- 12V battery charger (1)
- NI roboRIO (1) [99]
- Voltage regulator module (2)
- Power distribution board w/ fuses (2)
- Emergency stop switch (1)
- Limit switches
- Magnetic encoders
- Aluminum (c-channel, plates, churros, etc.)

T&T CNC Design Studio LLC offered the following tools for use at their facilities throughout the duration of the project:

- 6’ x 8’ CNC Table
- TIG Welder

# 7) Project Prototype Testing Plan

As the MAC contains both hardware and software components our testing plan will need to fully evaluate both the individual hardware and software components as well as the system as a whole. Our test plan will consist of unit testing, integration testing and system testing components. In order to keep track of test cases across the different testing methodologies we will be using a project management tool. In general, many individual hardware components purchased from a manufacturer should be operational such as resistors, capacitors, buttons, etc. as they have already been tested by the supplier during manufacturing. Hardware components such as sensors will need to be tested before integration in order to validate their functionality as they are key components in our design. Along with the sensors the motor controllers, power distribution board, voltage regulator module, and touchscreen display will all be tested before integration. There are several components of our software design that will need to be tested. Since we will be using Python we have several options in software testing through the **unittest** framework or through **pytest** another popular and flexible testing framework. Generally, the **unittest** framework is preferred in ROS along with the use of the **rostest** package. In the following sections, the testing environment and plan for both hardware and software is discussed.

## 7.1) Hardware Test Environment

For hardware testing, our testing environment will be the same environment in which we are building the robot. In terms of equipment, we will be using a Digilent Ultra Analog Discovery 2 Bundle and a Hantek 2D42 3 in 1 for the testing of hardware components. The Digilent Ultra Analog Discovery 2 is an oscilloscope and logic analyzer. The Hantek 2D42 is a handheld, portable oscilloscope, multimeter and waveform generator. The multimeter will be used for the testing of passive and active circuit components such as any resistors, capacitors, and the battery source. Even though as mentioned above these should be tested by the manufacturer, it will be a sanity test that will allow us to narrow down that any issues are due to the circuit design and not the individual components. The oscilloscope and logic analyzer will be used for testing the CAN network we will be building that will consist of the Jetson Nano,and motor controllers. We will also have analog communication and Serial UART communication in our system. CAN data will be sent to the devices using the **pycan** library as described in the software section thus in order to verify that CAN frames are being delivered correctly. The analog data can be converted using the Jetson Nano ADC converter and the serial UART data can be processed using the **pyserial** library. Further details on the testing plan will be given in the section below.

## 7.2) Hardware Testing Plan

The following sections will expand upon the different types of hardware testing that will be conducted on the hardware components being used to create the MAC. We will be performing functionality tests on the individual hardware components. Functionality testing will focus on conducting tests on hardware components such as the battery, resistors, capacitors, LiDAR sensor, motor controllers, ultrasonic sensors, gyroscope, power distribution board, Jetson Nano, voltage regulation module, camera, and kill switches. The testing of these components is

important in order to verify functionality before beginning the prototype building of the robot. Any components that do pass the tests will be kept track of in a table and reordered if necessary before the prototype building process begins. The hardware testing plan is meant to give an overview of the type of testing that can be conducted but this can be expanded or narrowed as seen fit by the engineers.

### 7.2.1) Battery & circuit components

The battery and electronic circuit components such as resistors, capacitors, and push-buttons will be tested through the use of a multimeter. The lead-acid battery being used comes in a set of two thus we will have a replacement if one of the batteries malfunctions. The battery is 12V and thus when testing with the multimeter at the battery terminals we expect to receive a reading of approximately 12V for both batteries. The resistors and capacitors will be tested in a similar way by verifying the value of each component using the multimeter on the ohm and farad setting respectively. The voltage regulator module (VRM) is a buck converter that is used commonly for DC-DC step down conversions. The Jetson Nano contains a power jack for a 5V power supply and thus the voltage regulator will be used to step-down the 12V output from the battery supply in order to power the Jetson Nano. Providing a 12V input to the Jetson Nano that requires only 5V has the potential to damage the board and thus it is important we use a multimeter to test the voltage at the output of the VRM with the battery supply connected before connecting the Jetson on the receiving side. The power distribution board (PDB) will be used to distribute power from the battery to the sensors, motor controllers and other components. The PDB will also contain a CAN termination bus therefore we will not have to terminate the CAN bus with a resistor as needed when developing a CAN network. A multimeter can be used to measure the output at each channel that is being used on the PDB as well as measure the resistance at the CAN channel to verify that the resistance is 120 ohms which is the standard value for proper CAN termination.

### 7.2.2) Sensors

The sensors included on the MAC can be tested in various ways. We can test each sensor using an oscilloscope and multimeter or we can use some barebones software. Our testing procedure for the sensors will thus include both bare hardware testing and afterwards sensors will be tested again with software to verify that the hardware can be controlled properly through the various libraries we will be using for interfacing. The software testing procedure will be outlined in the software testing dedicated section later in the document. For powering the sensors, we will be using the power supply in this case the battery through the PDB then test the output data of each sensor through the oscilloscope and logic analyzer if possible.

#### 7.2.2.1) Ultrasonic Sensor

The ultrasonic sensor we will be using is the LV-MaxSonar EZ1 that contains a GND, 5V, TX, RX, AN, PW and BW pins. This sensor has three different types of output formats including serial digital output, analog output and pulse width output. The three different types of outputs will all be tested as the method that will be used for outputting the data for software processing has not been determined therefore it will be assuring to test all three types of output to remain

flexible. Analog and pulse width output can be tested via an oscilloscope. Both these methods have their own translation between voltage and pulse width specified by the manufacturer. The serial output can be tested using the logic analyzer to see if the appropriate data is being sent. The sensor should be tested at various ranges to see if the output is changing correctly at the different ranges. Therefore, we should actively place the sensor at varying easily measured ranges, convert the data output appropriately using the specified conversions for analog or pulse width and verify the range output is correct. Additionally, the three results from the three different methods should all match accordingly. In terms of testing distances, this can be chosen arbitrarily as we want to verify the sensor is functional. More specific range testing will be done through software.

#### 7.2.2.2) LiDAR Sensor

The LiDAR sensor we will be using is a RPLiDAR A1M8 from Slamtec. This LiDAR sensor can be tested similarly to the ultrasonic sensor with the use of a logic analyzer. Since the LiDAR sensor uses UART for data communication and output, we should verify we are getting correct UART output measurements for the distance and heading of the LiDAR sensor compared to the sampling point. The LiDAR sensor requires power to both its scanning and motor system. The scanning system should have 5V applied for power with a maximum of 5.5V. If this is exceeded, the scanning system can potentially be damaged. For the motor system, can have a range of 5V to 10V applied where a higher voltage results in higher motor speed. Once again we will use the 12V battery tested before hand with the PDB and voltage regulator to step-down the voltage as needed. Again the testing of the voltage regulator will be important to verify the output is 5V to not damage the sensor as it is an expensive component. We will measure at a few different ranges in order to verify that the output varies correctly. These ranges can be chosen arbitrarily for the initial functional testing of the hardware. More detailed testing will be done through software to verify that the LiDAR sensor works for chosen ranges that will be common use cases when the MAC is operational.

### 7.2.3) Jetson Nano, Camera, Limit Switch & Kill Switch

The camera will be the simplest device to test as all we need to verify is that it is capturing images. The camera will be connected via USB to the Jetson Nano where it will be verified that it is capturing images. Since the Jetson Nano will be running the Ubuntu Linux distribution, it will be trivial to connect the Jetson to a monitor and view the feed from the camera. The limit switch and kill switch will be used for the scissor lift and battery respectively. The limit switch will be used to prevent the scissor lift so that it doesn't fold on top of itself and damage the chassis. The kill switch will act as an emergency stop that will completely cut power from the battery from the rest of the MAC in case there is an emergency where the MAC needs to be shut down completely. The limit switch will be tested by simply having an object make contact with it and verify that there is no electrical current being conducted. The scissor lift will be powered by motors and thus the limit switch will shut off the motors thus stopping the scissor lift. The kill switch will be tested in a similar way except that it will be connected to the battery and by pressing down it will result in an open circuit so no electrical current will be generated. The Jetson Nano will be easy to test for functionality as we will verify that the Ubuntu Linux

distribution is installed correctly and displaying the GUI interface. We can also verify that the USB ports included on-board are functioning properly.

### 7.2.4) Motor Controllers

The motor controllers that will be used is the Talon SRX which will be used to control each motor for finer and more precise movement. These will be controlled via CAN and thus we will need to test that it is able to communicate through CAN. There will be more robust testing through software using the **pycan** library where we can easily transmit CAN frames. For this testing, it is sufficient to connect the motor controller to the CAN bus and send some manual data frames through a CAN software.

## 7.3) Software Test Environment

The software testing environment will differ slightly from the hardware test environment. Ideally, we will be writing code on our personal machines then sending the code to the Jetson Nano for operation on the MAC. Therefore, our software testing will be done in two different environments. In order to keep the testing environment consistent across the machines of the software engineers, the same version of Python3 will be used, all code will be pushed to a Github repository and reviewed. In addition, there should be a consistent coding style established according to the ROS and PEP8 standard as defined in the *Standards* section of this document. The software will be interfacing with several hardware components as defined in the sections above. The software testing of the hardware components will be more comprehensive in order to verify the functionality of sensors at precise ranges. Another piece of the software that will need to be tested is the Android companion application that is a stretch-goal for the product. This environment will be on the software engineers machine consisting of an Android phone emulator with the capability of WiFi simulation. The development and testing of this application will be done after there has been validation of functionality for the MAC system as a whole since it is outlined as a reachable stretch goal.

Once the hardware has been verified for functionality via software, the code will be tested on the Jetson once major changes are made that affect the performance of the robot. There will be an established method of testing that will be done on Jetson Nano based on milestones defined but if necessary there will be ad-hoc testing in for any bug fixes or performance improvements. This testing environment will differ as the code will be pulled by the Jetson Nano and most likely multiple hardware components will be connected thus it will not be an isolated test of one specific hardware component. Both these tests will be important in verifying the functionality of the hardware component with basic controlling software as well as verifying the functionality of the requirements outlined. The software will be responsible for path-planning based on sensor data from the LiDAR and ultrasonic sensors as well driving the motors at the appropriate speeds based on the path-planning data. Also the software will be controlling the motor speeds of the lift motors as well. Our environment will attempt to isolate these components when connected to the Jetson Nano but it will also be necessary to test the functionality of these components together which will be detailed at the system testing plan.

## 7.4) Software Testing Plan

The software test plan will be outlined below and shown in a table for each software component. The procedure and plan outlined for each software component has the possibility of being repeated several times in order to achieve the desired result. The outline is meant to be a guide to the software engineer in order to iterate through testing fast and efficiently to find bugs and improve performance. The testing included in this section will follow most closely to unit testing where the “units” in this case are the individual hardware components that are being controlled by their respective ROS nodes. Therefore as mentioned before, we will be using **rostest** and **unittest** libraries for Python to conduct these unit tests. The ROS wiki specifies different levels of testing using the **rostest** and **unittest** libraries. The unit tests are defined as Level 1 where the testing is done at a function level that includes various inputs, conditions and outputs. Level 2 testing involves the use of the **rostest** library where each ROS node will be tested with the external API such as actions, topics and services. Finally, Level 3 testing is an integration test across all ROS nodes. The integration test will validate that multiple ROS nodes that may or may not communicate with each other are all interacting with each other and functioning correctly. This level of testing is meant to uncover any deadlock, race conditions or any other bugs to the developer. Level 3 testing in our case will be handled in the system testing plan where the individual subsystems will be tested. Since the software testing environment will differ from the software engineers machine and the Jetson Nano on-board the MAC the test code should be able to handle configuration for the different environments through environment variables and instance variables. The tester should be able to test their code in both environments. The following sections will thus be split into Level 1 and Level 2 testing each containing its own generalized test plan.

### 7.4.1) Level 1 Unit Testing

This unit testing will consist mainly of library unit testing where the core functions and libraries being used will be tested. Since there are several sensors, motors and motor controllers that will need to be tested with various communication protocols such as CAN, UART and analog communication there is the possibility that several functions that will be created that are dedicated to each. It is important to note that for this testing, we will not be testing any ROS nodes with the publish-subscribe architecture as that comes in Level 2 testing rather this testing will focus on the functionality of the basic code with hardware interfacing.

#### 7.4.1.1) LiDAR Sensor

The LiDAR sensor will be powered through the battery and PDB as specified in the hardware testing and hooked up to a computer running the basic hardware interfacing code via USB. The LiDAR communicates via Serial UART and thus we will require the **pyserial** library. The port number will vary between the Jetson Nano and the software engineer's computer therefore this should be handled in code during the configuration part of the testing. The LiDAR sensor will be its own test suite consisting of several test cases meant to test the functionality of the hardware. The test suite will create an instance of the LiDAR sensor object and contain multiple test cases including serial UART connection, and sending data. It is not possible for the LiDAR sensor to

move on its own and thus the tester will have to place the LiDAR sensor at different distances from a fixed sampling point. The unit testing will involve receiving and logging the UART data with verification that the range it is approximately correct. We do not need to test the LiDAR sensor at this stage at long ranges up to 12 meters in radius as this will be done during level 2 ROS node testing and system testing. It will simply be easier to test the LiDAR at longer distances by having it on-board the MAC where it is easily mobile. The test cases are shown in the table below.

| Test Steps | Test Cases | Test Criteria |
|---|---|---|
| 01 | Connect through correct serial port and verify that data is being received through the serial port | **Pass**: If no error message was shown by **pyserial** and some sort of data was received<br>**Fail**: Otherwise |
| 02 | Log data received from LiDAR sensor at various ranges using a consistent sampling point that can be any reasonable object | **Pass:** If data is successfully received and is approximately close to the constant range specified in the test case<br>**Fail:** Any error messages or range measured is not within a 3-inch margin |

Table 13: Test Cases for LiDAR Sensor

#### 7.4.1.2) Ultrasonic Sensor

The ultrasonic sensor we will be using has 3 different output formats including serial, analog and PWM. On the Jetson Nano we will be using the analog output format for the sensor with the specified conversion value of Vcc/512 per inch where Vcc is the supply voltage of 5V which results in approximately 9.8mV/in. The Jetson Nano has an ADC converter and thus through the GPIO pins the analog voltage output will be converted to a digital value that can then be used for processing. Therefore, the ultrasonic sensor will be unable to be tested without using the Jetson Nano and thus we will conduct testing through there instead of a software engineers machine as was done for the LiDAR sensor. Despite this the ultrasonic sensor will have a similar approach to testing as the LiDAR sensor above. We will test to make sure we receive range data and that the received data is within a margin of error. The ultrasonic sensors will also be chained as there will be multiple of them on board the MAC unlike the single LiDAR sensor we will be using. In order to avoid crosstalk interference, the manufacturer outlines several methods of how chaining can be done thus we will need to verify range data with the 5 sensors chained together. The data from these sensors should be clear and there should not be any interference or cross-talk. The test cases for the ultrasonic sensor are listed below.

| Test Steps | Test Case | Test Criteria |
|---|---|---|
| 01 | Connect the sensor through AN pinout and GPIO on Jetson Nano. Verify that the sensor gives a range reading during the calibration cycle | **Pass:** LV-MaxSonar is powered, gives a range reading that the Jetson Nano is able to convert to a digital output.<br>**Fail:** The sensor does not provide any data, or the Jetson Nano does not display the range data correctly |
| 02 | Log and verify the range data from the sensor from a few different ranges such as the minimum 6 inches, 24 inches and 36 inches. (These can be modified as needed during testing) | **Pass:** Range data is approximately accurate to the specified test range by a margin of error of 0.25 inches.<br>**Fail:** The sensor does provide accurate data or doesn't provide data at all. The Jetson Nano does not correctly convert analog data to digital. |
| 03 | Chain the sensors in the desired way according to the LV-MaxSonar manual. Verify that the sensors are triggered as stated in the manual and report accurate values. | **Pass:** The sensors are triggered in a chain according to the method described in the manual. There is no crosstalk and there should be 5 clear sensor readings logged and verified.<br>**Fail:** Sensors interfere with each other resulting in inaccurate readings. Jetson Nano does not log or verify the 5 data from sensors properly. |

Table 14: Test Cases for Ultrasonic Sensors

#### 7.4.1.3) Motors and Motor Controllers

The motors and motor controllers will be tested together as they will be operating in tandem on the MAC. Each CIM motor will be controlled via a motor controller and thus both motor controllers will need to be tested with their respective motors. The Talon SRX motor controller has capability of communication through CAN which will be preferred over PWM. As stated in hardware testing, the motor controllers will have already been tested that they are able to be commanded over CAN. This unit testing will mainly be testing the interfacing of software with the motors via the **pycan** library. This library is more complex than the **pyserial** library since the CAN bus requires more configuration than a connection over UART. This testing will be critical in verifying that the use of the **pycan** library is capable of sending CAN frames to the motor controllers. We will also need to verify that the motors are moving according to the CAN data being sent. It is currently unknown the type of data that should be sent over CAN for the controlling of the motors but this will be researched and a method of the type of CAN data to be sent. We can continue by defining the testing that is currently known that will verify that the

motor controllers are properly communicating over CAN. The motor controllers contain LEDs with a LED blink code table defined in the manual for when the CAN bus is detected or if there is an error. If connected properly to the PDB CAN termination bus and machine via the USB to CAN converter the LED on the motor controller alternates between being off and orange. If the CAN bus is not detected it will alternate between being off and a slow red. The next stage of testing is to use the **pycan** library to send CAN data frames on the bus that the motor controller will receive and drive the motor according to the data sent. Once the type of CAN data sent is known it will be easy to validate as the motor will either drive slower or faster in a specific direction. The test cases are shown in the table below and this table will be updated as we move forward with proper CAN data frames.

| Test Steps | Test Case | Test Criteria |
|---|---|---|
| 01 | Connect each motor controller via CAN using the PDB and USB to CAN converter. Verify the LEDs alternate between off and orange. | **Pass:** LEDs blink in correctly according to the table in manual (off/orange)<br>**Fail:** No LED blinking or LED blinks between off and slow red. |
| 02 | Connect motors to motor controllers and send CAN data commands that should reflect in the speed and rotation of the motors. | **Pass:** The motor controllers receive the CAN commands and the motors respond accordingly<br>**Fail:** No CAN commands received by the controllers or the motors do not move correctly based on commands |

Table 15: Test Cases for Motor Controllers and Motors

#### 7.4.1.4) Camera and CV

The camera on-board the MAC will be forward facing and will be able to capture images that will be used for LOS following when commanded by the user. The camera will feed image data to a Yolo V3 convolutional neural network that is a power neural network used for image detection and processing. As the case for any machine learning model, the model will have to be trained correctly so it is able to detect people, wheelchairs and other objects are used commonly for mobility in the household. Once the model is trained, we will test the model by feeding real image data from the camera and measure how accurate the model is. The test results here will define whether the model is able to accurately detect the human, wheelchair or scooter it needs to follow. We have not included a table of test cases for this testing as it will be highly dynamic and dependent on how the model is trained.

### 7.4.2) Level 2: ROS Node Testing

As mentioned previously the next level of testing will focus on unit testing on the ROS node itself instead of just library testing as done above. We will be using the **rostest** package that requires the creation of a ROS node that will be a designated test node. Therefore, we will have a test node created for each respective node that requires testing. This will be testing the operation of the ROS node on the Jetson Nano with hardware components and thus can include ROS packages such as the **slam_gmapping** package that will be used for the SLAM mapping and the **navigation** package that will be used for path planning. There are several topics that will be created that the nodes can publish and subscribe to. The test cases are meant to start-up a node and then test the external API of the topics the node needs to subscribe to or publish to. In order to keep the testing timely and condensed similar nodes that require similar testing will be paired together such as the sensor nodes. Also testing will be conducted so that it primarily follows the general flow of data in software which will provide flexibility if we use sample data or real data collected by sensors. This testing will be based on the key nodes shown in the ROS software architecture diagram included in *section 5.7.1*.

#### 7.4.2.1) Sensor Nodes

There are 6 different sensor nodes that will be deployed to be responsible for interfacing with hardware to collect sensor data and publish to the sensor data collection topic in order to be accessible by other nodes. This will be a natural transition that follows the unit hardware testing done above as we will be mainly testing the ability for ROS nodes to capture the necessary sensor data. Also this will be the first point of testing as it will allow us to have options for testing the nodes that require sensor data by giving real sensor data or we revert to using simulated data available from online resources. At this stage of testing, we will verify that the hardware components are functioning as well as that the ROS nodes are able to collect data from the sensors correctly thus giving a foundation for the rest of the testing plan. The test case will be similar for all sensor nodes except that they differ in the type of data they are expecting to acquire. For example, the LiDAR sensor node should expect laser scan data and not any other type of data. The ultrasonic sensor will expect to receive range data that can then be transformed and used to measure the distance of objects from the MAC. Other sensors such as the limit switch operate slightly different as they break the electrical current when triggered. The limit switch node needs to be able to read that the electrical connection has been broken and relay this to the MAC. The limit switch is a safety critical component for the scissor lift in order to verify that the lift does not reach an unstable height for the load to potentially fall or for the MAC to potentially tip. The encoders will provide rotational angle data and positional data for the motors which will provide the MAC with a feedback loop that will be useful for testing and sanity checks within the code. This testing may need to be performed several times in order to verify that all the sensors are operating as they should in order to not encounter any issues with the rest of the development and testing. The general test cases for each sensor node are shown in the table below.

| Test Steps | Test Cases | Test Criteria |
|---|---|---|
| 01-LiDAR | Verify that the LiDAR sensor node is able to acquire laser data as it is streamed by the sensor and is able to publish to the sensor data collection topic | **Pass:** All data coming from the sensor is collected and handled appropriately in the node.<br>**Fail:** Data is missed or any error messages |
| 02-Ultrasonic | Verify that the ultrasonic sensor node is able to acquire range data as it is streamed by the sensor and is able to publish to the sensor data collection topic | **Pass:** All data coming from the sensor is collected and handled appropriately in the node<br>**Fail:** Data is missed or any error messages |
| 03-Camera | Verify that the camera sensor node is able to acquire image data as it is streamed by the sensor and is able to publish to the sensor data collection topic | **Pass:** All data coming from the sensor is collected and handled appropriately in the node<br>**Fail:** Data is missed or any error messages |
| 04-Encoder Sensor | Verify that the encoder sensor node is able to acquire encoder data as it is streamed by the sensor and is able to publish to the sensor data collection topic. This data should be transformed to odometry data | **Pass:** All data coming from the sensor is collected and handled appropriately in the node<br>**Fail:** Data is missed or any error messages |
| 05-Limit Switch | Verify that the limit switch sensor node is able to acquire the electrical signal sent by the switch when the switch is triggered and is able to publish to the sensor data collection topic | **Pass:** All data coming from the sensor is collected and handled appropriately in the node<br>**Fail:** Data is missed or any error messages |
| 06-Gyroscope | Verify that the gyroscope sensor node is able to acquire odometry data as it is streamed by the sensor and is able to publish to the sensor data collection topic | **Pass:** All data coming from the sensor is collected and handled appropriately in the node<br>**Fail:** Data is missed or any error messages |

Table 16: Test Cases for Sensor Nodes

#### 7.4.2.2) SLAM Mapping Node

This node will use the **slam_gmapping** package to generate a SLAM map using the sensor data collected primarily from the LiDAR sensor. Therefore, our testing environment in this case will be on the Jetson Nano with the LiDAR connected communicating through UART. The laser scan data will be collected by the SLAM mapping node by subscribing to a sensor collection topic and transformed into odometry data where the map will be generated along with map metadata. The node will publish the map data to the SLAM mapping ROS topic so that other ROS nodes such

as path planning are able to use the map. This testing will focus on the SLAM mapping node receiving data from the necessary sensors through the subscribed topic and then publishing the data to the SLAM map. We will create a ROS test node that will be responsible for conducting an automated test for this. The ROS mapping test node will start-up the SLAM mapping node and conduct the necessary tests. It is recommended to not use hardware for testing ROS nodes in many cases as it is necessary to isolate the software with the hardware so we can use sample laser scan data that is readily made available by ROS instead of real data from the LiDAR sensor. In our case, if the hardware components are not accessible the SLAM mapping node can be tested via sample data. Ideally, we want to test both cases with the actual LiDAR hardware component as well before conducting any system testing. The test cases are defined in the table below that are meant to be software based so that both real or sample laser scan data can be used.

| Test Steps | Test Cases | Test Criteria |
|---|---|---|
| 01 | Verify that the SLAM mapping node is able to subscribe to laser scan data provided by sensor collection topic | **Pass:** The necessary laser scan data is acquired by the node<br>**Fail:** Laser scan data is not read by the ROS node or sensor topic. |
| 02 | Verify that the SLAM mapping node converts laser scan data to odometry data and generates a map. | **Pass:** The odometry data and map is accurate with the sensor data provided<br>**Fail:** If there is no map generated or map is severely inaccurate |
| 03 | Verify that the map topic receives the published map from the SLAM mapping node. | **Pass:** We are able to access map data from the map topic that reflects the generated map from the node<br>**Fail:** There is no map made available |

Table 17: Test Cases for SLAM Mapping ROS Node

#### 7.4.2.3) LOS Following

The LOS following node will run the processing for a key feature of the MAC. If commanded by the user the LOS node will be spun-up by being subscribed to the robot state topic. The robot state topic will contain the state of the MAC as a result of user commands or in the case of a safety precaution. If LOS following is activated, the node will process the location of the human by using computer vision and publish the resulting location to the path points topic. This will then be used by the path planning node to generate a safe path for the MAC and relay that information to the motor controller. This testing is crucial for verifying the overall functionality of the MAC as that is the primary node that will be activated directly by the user. The test cases here will simulate a user command that in practice will be coming from primarily a remote or touchscreen display. The user input devices and the respective nodes will be tested separately in order to isolate testing properly before integration. The SLAM map data can be a map generated

using real laser-scan data or simulated data as described in SLAM map node testing above. The LOS following will be utilizing camera data to locate the human that the MAC needs to follow and thus we will have to provide simulated CV data or capture real data using the camera that will be on-board the MAC. If real image data is being used, the YoloV3 model will need to be trained and ready to receive input data. Since we have tested hardware components before the software testing either method will suffice for testing. The table shown below lays out the test cases.

| **Test Steps** | **Test Cases** | **Test Criteria** |
|---|---|---|
| 01 | Verify the LOS following node is spun-up upon activation by the simulated user command | **Pass:** The simulated command successfully starts-up the LOS following node<br>**Fail:** LOS following does not start, or any error messages show |
| 02 | Verify that the LOS following node uses camera sensor data to generate a path point based on the location of the user. | **Pass:** A path point is successfully created and published using CV from camera sensor data.<br>**Fail:** The path point is inaccurate, not published or any error messages |

Table 18: Test Cases for LOS Following

#### 7.4.2.4) Path Planning

The path planning node follows naturally from the data processed and the map generated by the SLAM mapping node tested above. The path planning node will be subscribed to the map topic along with others such as the path points node and robot states node. The path planning node is responsible for mapping a path for the MAC to take based on the SLAM map and in case of LOS following, the position of the user stored in the path points node. Since the path planning node is subscribed to multiple sources of data, the testing will be more complex. For clarity, this testing will focus on solely acquiring data from sources that aid in path planning and publishing the resulting data to topics such as the motor commands topic. The LOS following node should provide the location of the user as a path point. We will test the path planning node to verify it is able to plan a safe path for the MAC by receiving data from the SLAM maps topic and path points topic. Also the path planning node should then publish the velocities to the motor commands topic that will be used by the motor controllers. The test cases for the path planning node are shown in the table below.

| Test Steps | Test Cases | Test Criteria |
|---|---|---|
| 01 | Verify that the path node acquires data from the SLAM map topic and path points node | **Pass:** Data from both sources is received correctly<br>**Fail:** Data is not received or any error messages |
| 02 | Plan a sample path using the SLAM map and path points data for the MAC. This should generate velocity commands for the motor controllers | **Pass:** The path is planned correctly considering both sources and produces velocity commands<br>**Fail:** Velocity commands are not produced, any error messages. |
| 03 | Verify the path node publishes velocity commands to motor commands topic | **Pass:** The path node publishes the velocity commands to the motor commands topic as they are produced<br>**Fail:** Velocity commands are not published in its entirety, any error messages |

Table 19: Test Cases for Path Planning Node

#### 7.4.2.5) Drivetrain Motor Control & Lift Motor Control

The drivetrain motor control and lift motor control will be responsible for controlling the motors of the MAC and the motors of the scissor lift on-board respectively. The drivetrain motor controllers will subscribe and thus accept data from the path planning node primarily that will provide the velocity commands that the motor controllers will use to control the motors. Also the drivetrain motor control node will accept commands directly from the user thus this will need to be tested as well. The lift motor control is similar except the primary commands for the motors to lift the scissor lift up or down will come from the user. The path planning node will be responsible for making sure the lift is all the way down before moving. This results in a fairly complicated testing scheme as we have two sources of inputs for both motor controlling nodes. The user input will primarily come from a remote input or touchscreen display. The smartphone app input is a stretch goal that if reachable will be a way for the user to communicate with the MAC as well. This testing will focus on the node itself and thus the motor commands will be produced by the path planning node. Testing with user input from any of three sources above will be done separately.

Our testing process for the motor controllers will be similar to the SLAM mapping testing in the sense that we will focus on the nodes itself and their external API to other nodes, topics or services. We will need to test the controllers so that they are able to receive data from the motor commands topic and then use that data to control the motors. The motor commands topic is used

by the path planning node to publish the velocity commands and to make them available for the motor control nodes. We can test this in a couple ways where we can use simulated sample data to use for motor control or we can use real data collected from the MAC. Since our ROS software architecture is a flow of data from one node to another, the testing of the nodes is done in the way the data is meant to flow thus the SLAM mapping node tested above will have produced some type of data that can be used by the path planning node which can then provide the necessary velocity commands. Either way will be acceptable for the testing at this stage. The test cases are outlined in the table below.

| **Test Steps** | **Test Cases** | **Test Criteria** |
|---|---|---|
| 01 | Verify that both the **drivetrain motor control** and **lift motor control** nodes are able to receive data from the **motor commands** topic when velocity data is published. | **Pass:** Both nodes are able to receive data from the **motor commands** topic.<br>**Fail:** Data is not received by either or both the motor control nodes. |
| 02 | Verify that the data received from the **drivetrain motor control** node properly commands the CIM motors on board the MAC | **Pass:** Both drivetrain motors are controlled appropriately via commands sent<br>**Fail:** Motors fail to operate or fail to operate correctly based on commands |
| 03 | Verify that the data received from the **lift motor control** node properly commands the lift motor in terms of increasing or decreasing the height of the lift | **Pass:** Lift is lowered or raised appropriately to the commands passed in<br>**Fail:** Motors fail to operate or fail to operate correctly based on commands |

Table 20: Test Cases for Motor Controller Nodes

## 7.5) System Testing Plan

At this stage of testing we will focus on evaluating the requirements that were outlined in the beginning of the document and during the project proposal. The MAC should be fully operational at this stage after making necessary adjustments after the hardware testing and software testing iterations are completed. The system testing plan will verify that the MAC meets the performance metrics as well requirements outlined in the requirements table below. The following subsections will outline the test plan for key subsystems. During this system testing, we will focus on treating the specific subsystem as a blackbox where the output will be evaluated

due to a specific input. To keep track of the specific subsystem, we will be using a simple excel sheet with a testing schedule in order to have enough time to make adjustments as needed for the final product. The subsystems that we will test include the scissor lift, LOS following, and input sources. The test cases for each subsystem and the final requirements/testing matrix for the whole system are detailed in the following subsections.

### 7.5.1) Scissor Lift

The scissor lift is a key subsystem of our design that involves both mechanical and software complexity. In this testing, we will focus on the controlling software for the scissor lift and verify that the scissor lift operates safely from a user perspective. The scissor lift was testing manually in the hardware section for load tolerance, horizontal sway and speed of the lift. In terms of operation, we will use the remote for the primary input device. The buttons on the remote that will be used for testing is lift up and lift down. When the lift-up button is clicked the mini-CIM motors should safely and slowly lift the platform with a maximum height of 27 inches. When the lift-down button is clicked then the lift should descend to its minimum height. There will be a limit switch that makes sure that the lift does not exceed its maximum height as that can lead to safety issues of objects falling and even the MAC tipping over. Thus this testing will also verify that the limit switch is operating correctly within the core operation of the MAC. The table below outlines the test cases and metrics the scissor lift will meet based on the requirements.

| **Requirement** | **Test Number** | **Test Case** | **Test Status** |
|---|---|---|---|
| Speed | SL-01 | Press lift-up button on remote and verify the lift ascends with proper height management with the limit switch | **Pass:** Meets requirement metrics on lift speed and height<br>**Fail:** Otherwise |
| Speed | SL-02 | Press lift-down button on remote and verify the left descends correctly | **Pass:** Meets requirement metrics on lift speed and height<br>**Fail:** Otherwise |

Table 21: Trace of Requirements to Test Cases

### 7.5.2) LOS Following

This will be a key feature for the MAC and the testing done here will effectively evaluate the MAC in its most important operational areas. The LOS following will use CV and the SLAM map generated in order navigate through the household environment while following the user. The remote will be used for input and by clicking the LOS following button the MAC will conduct the necessary steps to follow the user correctly. This will evaluate speed, stopping distance, and intelligence requirements. The speed the MAC travels at needs to be 3 feet / second when traveling and 2 feet / second when following the user. The stopping distance should be 6

inches away from the user. Finally, the intelligence requirements will be evaluated in order to verify the MAC computes its path in 5 seconds, replanning is done in less than a second and responds to obstacles in less than 100 milliseconds. This can be measured by simple observations and by measuring the time elapsed in the software test cases. We will also test the basic functionality that the MAC is able to reach the end destination by following the user. The trace of requirements to the test cases table is shown below.

| **Requirement** | **Test Number** | **Test Case** | **Test Status** |
|---|---|---|---|
| - | LOS-01 | Verify that the MAC is able to follow the user from the starting location to the destination. | **Pass:** The MAC will traverse its environment and follow the user to the end destination correctly.<br>**Fail:** The MAC does not reach its end destination or follow the user properly. |
| Speed | LOS-02 | Verify that the MAC is traveling at approximately 2 feet/second when following the user. | **Pass:** The MAC travels at the speed requirements with a margin of 0.5 feet.<br>**Fail:** The MAC drives outside the range of acceptable speed. |
| Stopping Distance | LOS-03 | Verify that the MAC stops within 6 inches of the user when either the user stops moving or gets too close to the MAC | **Pass:** The MAC stops within the stopping requirements.<br>**Fail:** The MAC fails to stop when needed and/or fails to stop within the distance with a margin of 1 inch. |
| Intelligence | LOS-04 | Verify that the MAC is able to compute path planning in 5 seconds | **Pass:** The path is planned correctly with a margin of 1 second<br>**Fail:** The path is not planned at all or not within the time range specified |
| Intelligence | LOS-05 | Verify that the MAC is able to replan its path in less than 1 second | **Pass:** The path is replanned correctly within the time specs and margin of 1 seconds<br>**Fail:** The path is not replanned at all or not within the time range specified |
| Intelligence | LOS-06 | Verify that the MAC is responds to obstacles in less than 100 milliseconds | **Pass:** The MAC responsiveness has an upper bound of 100 milliseconds<br>**Fail:** The MAC does not respond to obstacles or does not meet the time constraint |

Table 22: Trace of Requirements to Test Cases for LOS Following

### 7.5.3) Input devices and general testing

The testing in this section will be around the input devices of the MAC and other general testing in order to evaluate the requirements. This includes verifying that the MAC is easy to use using the remote and touchscreen display, the buttons on the MAC that have not been tested yet are functioning correctly, the speed when the MAC is remotely commanded is within the correct specifications of 3 feet per second, the range of the remote will be verified and the battery life will be verified as well according to the requirements table. The buttons that will be tested here are the move forward, reverse, turn left, turn right and stop. The lift and LOS buttons will have already been tested above therefore these do not have to be tested. The range of the remote will be validated by attempting to command the MAC from various ranges with a maximum of 200 feet from the wireless access point since we will be using WiFi communication. Battery life will be tested by operating the MAC at the different battery life to weight categories specified in the requirements table. The trace of requirements to the test cases table is shown below.

| **Requirement** | **Test Number** | **Test Case** | **Test Status** |
|---|---|---|---|
| Ease of use | GT-01 | Verify that all the buttons on the MAC are functioning correctly, are easy to press and clearly labeled | **Pass:** All buttons are functioning correctly<br>**Fail:** A button does not operate as intended |
| Battery Life | GT-02 | Verify that the battery life at various weights (100 pounds, 50 pounds, 25 pounds and 0 pounds) lasts within the time specified | **Pass:** The battery life lasts as specified in the requirements table at the various weights with a margin of 2 minutes.<br>**Fail:** Otherwise |
| Speed | GT-03 | Verify that the MAC travels at 3 feet / seconds when being remotely operated | **Pass:** The MAC travels at the speed intended within a margin of 0.5 feet<br>**Fail:** Otherwise |
| Stopping Distance | GT-04 | Verify that the MAC stops within 6 inches of where the MAC was when the stop button on the remote is pressed | **Pass:** The MAC stops within 6 inches with a margin of 1 inch.<br>**Fail:** Otherwise |
| Touchscreen Display | GT-05 | Verify that the touchscreen display on the MAC operates correctly and is easy to use for the user | **Pass:** The touchscreen has unambiguous labels and responds to the user input reliably<br>**Fail:** The touchscreen display is unclear, or user input commands do not process correctly. |

Table 23: Trace of Requirements to Test Cases for General Testing

# 8) MAC Operations

## 8.1) Setup

This section describes the initial setup for the MAC to operate properly in your home. Please follow all steps accordingly and refer to the Troubleshooting section for fixing common issues.

### 8.1.1) Powering On

The MAC has a physical button located on the top, next to the charging port and on board display. This is a full cut off of power to the MAC, meaning turning this switch on and off will completely shut down the MAC and all of its functionalities. It is recommended to leave this on, as the MAC has sleep functionalities for saving power and performs processes in the background while charging and idling, like battery level, remaining charge time, current status, and the ability to readily accept commands. The MAC should only be fully powered off because of some malfunction or issue that requires a full reboot of the MAC. Beforing charging, please turn on the MAC.

### 8.1.2) Charging

Before first use, fully charge the MAC for best performance. The MAC comes with a battery charger that can plug into any standard 120V AC wall outlet. It is recommended to pick a charging location that is easily accessible for you, as the MAC will remember the charging location in your home and return to it when charging is required. The MAC has a charging port on the top part of the front structure for easy access, simply plug in the provided charger into a wall outlet and plug in the MAC using the top charging port. The MAC's display will show the status of charging while it is powered on, as will the charger itself indicate the charging status. Unless the MAC is malfunctioning, the power should be kept on at all times to ensure proper operation.

The charger will display a red light while charging, and a green light when the battery is fully charged. If the MAC is powered, the on board display will also display current battery power, charging status, and time to a full charge. Once fully charged, the charger can be disconnected. The charger can be disconnected at any time during charging as well. When charging, the MAC will not respond to any commands or input from the remote or on board display, it must be off the charger before normal operation can occur.

When ready to charge the MAC or have it return to it's charging location, simply press the dock button on either the remote or the MAC's on board display. It will return and park itself next to the charging relocation remembered in the previous step. If you change the charging location, you can use the Teleoperated mode to manually drive the MAC to the new charging location,

where once the charger is connected to the top port the location will automatically update in the MAC so the dock command will return to the new charging location. If Teleoperated mode is not available, the MAC can be manually pushed to the new charging location as well, but this is not recommended.

If you are planning to not use the MAC for a long time, it is recommended to connect the charger and power off the MAC. This ensures the lifetime of the battery does not degrade, and that the MAC is not wasting resources or reducing the lifetime of the components on the MAC by idling while powered on.

### 8.1.3) Mapping Out the Home

Once the MAC is fully charged and powered on, you need to do an initial mapping of your home for best operational performance. The MAC constantly learns your home as it is used, but an initial mapping can help with reducing errors and malfunctions the MAC can run into. The map is used to navigate around your home quickly and avoid any obstacles. You can also skip this step, but may run into issues on first operations with the MAC.

In order to start this process, disconnect the charger and make sure the MAC is powered on and ready for operation. Then, around your home make sure to open any doors such that the MAC can enter and exit the room. On the on board display, click the gear icon, then click the button that says **"Map Home"**. The MAC will then start traversing your home, building an internal map for navigation. You can cancel this operation at any time by pressing the **"Stop"** button on the remote or the **"Dock"** button to have it return to the charging location. These operations are described in detail in section 8.2.

Once completed, the MAC will then return to its starting location, where it was last charged at. At this point, the MAC is ready for normal operation as described in section 8.2. The next section describes the modules that MAC offers, and how to add and remove them.

### 8.1.4) Adding and Removing Modules

The MAC comes with a few lightweight modules that can be added or removed with ease to provide extra functionality to the MAC. Included with your first generation MAC is a small ramp to allow the MAC to go up and down small inclines, and a plastic shelf that slots into the bed of the MAC to allow the MAC to transfer small objects on a multi-tiered shelf.

The ramp is something that does not attach directly to the MAC, but is used if you have doorways or other small steps too big for the MAC to normally traverse over. Simply set the ramp down such that the gap is completely bridged by the ramp. The other module that comes with the MAC is the shelf. On the platform of the MAC where items are stored, there is a circular cutout in each corner. Take the included shelf, which should be light enough to pick up easily, and line up the four poles of the shelf with the circular cutouts on the MAC. Then slot the shelf so that it goes completely into each cutout of each corner. Now your MAC can carry items such as groceries or other small objects on the shelf!

## 8.2) Operation

During any of the operations listed below, the MAC can be stopped and put back into an idle state at any time using the **“Stop”** button on the provided remote, or the on screen display. Any of the below operations as well can be done either from the provided remote, or the onboard touch screen display located at the top of the MAC. The onboard display also has extra settings to run setup tasks described in the previous section, or edit various other settings. For convenience, the remote is the recommended input to use with the MAC.

Also, only one operation mode can occur at any given time. For example, if the MAC is currently in “Follow” mode, the lift functionalities will not work. Same with trying to use the Teleoperated mode while the MAC is docking. In order to use a different mode, make sure to click the **“Stop”** button before using another mode. The onboard display will show what mode is currently being run, or if the MAC is idling, in which case you can use any functionality.

### 8.2.1) Following Mode

The following mode is mainly how the MAC moves around. The following mode will have the MAC follow you using a front mounted camera. The MAC will follow 1-3 feet behind you as you move to your destination. The MAC recognizes the closest person and follows them, so others can walk around you without confusing the MAC. This mode requires you to be in front of the MAC and close enough such that the MAC can detect you. Click the **“Follow”** button on the remote or on the onboard display to start this mode. Before this mode can start, if the lift is not completely down, the lift will automatically go fully down before moving as a safety precaution, so please wait for the life to return to the starting position before the mode begins operation.

If the MAC does not recognize anyone to follow, it will stop in place and wait for you to walk in front of it, with a beeping noise going off signifying it lost sight. However, it will move to your last location it was following you to, so when going through doorways or around corners the MAC won’t stop completely, so it is recommended to walk your normal pace throughout the home and the MAC will follow. To stop the MAC once you reach your destination, click the **“Stop**” button or the **“Follow”** button again. The MAC will then move closer and position itself so that you may unload it easily. If the MAC did not position itself optimally for you to unload it, you can use the Teleoperated mode as described in section 8.2.2 to manually position it however you need easily.

The MAC’s camera is only used for this feature and for object detection to avoid collisions with the environment. None of the data is recorded or sent to any third party service, it is all done on the MAC itself, so your privacy is maintained by the MAC. The camera is also only on when the MAC is moving, while it is idling or lifting the camera remains off.

### 8.2.2) Teleoperated Mode

This mode can only be accomplished using the remote, the onboard display **cannot** be used to manually control the MAC. Please make sure to maintain a safe distance of at least 1-2 feet when in this mode to avoid accidently hitting yourself or others with the MAC, although in this mode the MAC will travel much slower then in its autonomous modes, for easier control. Also make sure the MAC is not near other people, pets, or obstacles that might cause interference with the MAC. The lift also has to be down completely before this mode can be activated. If the lift is not down completely, the MAC will automatically return the lift to the starting position before you can control it, please wait for this finish before continuing with the Teleoperated mode.

The remote has four buttons under the Teleoperated section. These buttons are labeled "**Forward", "Reverse", "Left", and "Right".** These will move the MAC in those referenced directions. The front of the MAC is signified by where the camera points, meaning that the tall structure in the front that houses the onboard display, charging port, and power switch signifies the front of the MAC. These buttons will move the MAC with this reference. This mode should not be used for driving the MAC around your home, it should only be used for positioning the MAC for easier offloading when it does not position itself optimally after following or docking modes are completed.

During teleoperated mode, some of the safety features and collision avoidance is suspended for easier use, so you can hit other obstacles and possibly hurt or damage them, so use this mode with caution and only while you are near the MAC to easily guide it as you use this mode. Once you are done using this mode, press the **"Stop"** button, or after 1 minute without moving the MAC will automatically go back to idling, allowing for operation mode input from the remote or the onboard display.

This mode is also useful in case the MAC is stuck on some obstacle or gets blocked in where it cannot move easily. You can activate this mode to move the MAC out from where it is stuck or turn it around such that the follow mode or other mode you are using can resume as normal. Do not be afraid of using this mode, as it is quite easy and can help when you need the MAC to perform a maneuver it cannot do easily on its own.

### 8.2.3) Lifting

The lifting mode is used solely for moving the lift on the MAC up and down for lifting its load up to 3 feet in the air. This is useful for lifting a load up to a counter or other surface for easy unloading. In order to use the other functions described in this section, the lift must be completely down. When another mode is activated, the lift will always return to the starting position before continuing.

There are two buttons for this mode under the lift section on the remote or onboard display. These are labeled **"Up" and "Down"** each of these will move the lift either up or down at a constant rate for safety precautions. The lift will stop moving once it reaches its starting position all the way down or its max height of 3 feet all the way up. Clicking the button in the

corresponding direction once it reaches its limit will not move the lift further to avoid damage to the MAC. These buttons are toggles, meaning that you only need to press the button once to have the lift constantly move in that direction. Press the same button again to stop the lift. For example, press the **“Up”** button and wait until the lift is at the desired position, then press the **“Up”** button again to stop the lift.

As a warning, **do not** put anything under the lift while it is extended, the lift can crush your hands, fingers, or other objects that may fall under it while it is extended. Take caution when moving the life up or down of children, pets, or other objects that may accidentally get caught underneath the lift of the MAC.

### 8.2.4) Docking

The docking mode returns the MAC to where it was last charged in your home such that you can store it safely and charge it without needing to manually move the MAC or the charger everytime the MAC requires charging or not in use. This mode may require you to do the final position of the MAC so that it can be easily charged using the Teleoperated mode, but typically the MAC should be able to properly position itself without user input.

While the MAC is idling and no other mode is currently being used, you can press the **“Dock”** button on the remote or on the onboard display to start the docking process. The MAC will then start traversing to the charging location. If the MAC is blocked and cannot reach the charging location by any obstructions, the MAC will stop and let out 3 beeps to indicate the docking failed. Remove any obstructions and press the **“Dock”** button again on the remote or onboard display to continue the docking operation.

As the MAC is moving, you or others may safely walk in front of it to go around the MAC, it will slow down and speed up as needed as to not hit you, pets, other people, or any other obstacles while moving.

## 8.3) Troubleshooting

### 8.3.1) Map Outdated or Corrupted

If the map is extremely outdated or not displayed properly, you can follow the section 8.1.3 Mapping out the home to have the MAC remap your home. This can be done anywhere in the home, but preferably from the charging location. The map may be corrupted because of abrupt power loss, changing to a new home, or heavy modification of your home layout (like furniture being moved around). Any time one of these occurs, it's better to do a remapping in order to avoid issues with mapping in the future.

# 9) Administrative Content

## 9.1) Financing and Budget

The following section will discuss the budget of MAC and how the construction of MAC was financed.

### 9.1.1) Financing

The main source of materials supply for MAC's construction is through Bartholomew Nash, as referenced in *section 6.9.3 Suppliers*. They are the head of a local robotics club that already has some supplies and are supportive of the project and willing to help through means of components that they already possess. Without their generous support, the construction of MAC would not be possible. For other items that the sponsor was not able to supply, we decided that we would fund the purchasing of the components and split the costs evenly.

### 9.1.2) Bill of Materials (BOM)

The MAC is the first robot of its kind being designed for home use, that being said we chose an arbitrary budget of $1250 based on current prices of the parts needed to accomplish our must have features. Over the course of constructing this report and the discovery of a sponsor we have determined that the must have components will be around $750 USD. And if we plan to implement all of the nice features, the total cost would be around $1900 USD. Prices are assuming MAC will be built to full size. Not all parts will be used on the final version of the MAC, since the design might change as it is being built to accommodate unforeseen issues or design flaws. The exact expected break down of costs is listed below.

| Importance | Subsystem | Description | Price Per Unit Estimate ($) | Quantity | Total ($) |
|---|---|---|---|---|---|
| Must Have | Chassis | Chassis AM14U4 | Sponsored | 1 | Sponsored |
| Must Have | Chassis | Chassis Hardware | 50 | 1 | 50 |
| Must Have | Chassis | Cargo Motors 2.5 CIM | Sponsored | 2 | Sponsored |
| Must Have | Chassis | Cargo Motor Controller TalonSRX | Sponsored | 1 | Sponsored |
| Must Have | Scissor Lift | ACME Lead Screw | 9 | 2 | 18 |
| Must Have | Scissor Lift | ACME Nuts | 3 | 6 | 18 |
| Must Have | Scissor Lift | Bearing Housings | 5 | 4 | 20 |
| Must Have | Scissor Lift | U Channel | 8 | 4 | 32 |
| Must Have | Scissor Lift | Sheet Steel | 30 | 2 | 60 |
| Must Have | Scissor Lift | Bushings | 0.5 | 20 | 10 |
| Must Have | Scissor Lift | Scissor Lift Hardware | 2 | 10 | 20 |
| Must Have | Chassis | Wheels | Sponsored | 4 | Sponsored |
| Must Have | Electrical Moven | Battery | Sponsored | 1 | Sponsored |
| Must Have | Electrical Moven | Wheel Motors 2.5 CIM | Sponsored | 2 | Sponsored |
| Must Have | Electrical Moven | Motor Controllers TalonSRX | Sponsored | 2 | Sponsored |
| Must Have | Computation | MCU | 50 | 1 | 50 |
| Must Have | Sensing | Camera | 40 | 1 | 40 |
| Must Have | Sensing | LIDAR | 150 | 1 | 150 |
| Must Have | Sensing | Ultrasonic Sensor | 40 | 5 | 200 |
| Must Have | Sensing | Cargo Weight Sensor | 30 | 1 | 30 |
| Must Have | Control | Remote Microcontroller | 1 | 1 | 1 |
| Must Have | Control | Remote Battery Holder | 2 | 1 | 2 |
| Must Have | Control | Remote Power Switch | 1 | 1 | 1 |
| Must Have | Control | Remote Buttons | 0.5 | 10 | 5 |
| Must Have | Control | Remote Oscillator | 0.75 | 1 | 0.75 |
| Must Have | Control | Remote Enclosure | 10 | 1 | 10 |
| Must Have | Control | Receiver SPI to USB Converter | 2 | 1 | 2 |
| Must Have | Control | Receiver USB Connector | 0.75 | 1 | 0.75 |
| Must Have | Control | RF Receiver | 3 | 2 | 6 |
| Must Have | Control | Remote Enclosure | 5 | 1 | 5 |
| Must Have | Control | Remote Receiver Enclosure | 5 | 1 | 5 |
| Must Have | Accessory | Ramp Materials and Hardware | 30 | 1 | 30 |
| Must Have | Accessory | Shelf Materials and Hardware | 30 | 1 | 30 |
| Must Have Total: | | | | | 766.5 |
| Nice to Have | Sensing | Tilt Sensors | 10 | 1 | 10 |
| Nice to Have | Chassis | Cargo Sorting Rails | 25 | 1 | 25 |
| Nice to Have | Dock | Docking Station Electronics | 30 | 1 | 30 |
| Nice to Have | Dock | Docking Station Enclosure | 20 | 1 | 20 |
| Nice to Have | Control | Onboard Display | 15 | 1 | 15 |
| Nice to Have | Chassis | Omnidirection Wheels | 30 | 4 | 120 |
| Nice to Have Total: | | | | | 1085 |
| Grand Total: | | | | | 1851.5 |

Table 24: MAC Bill of Materials

## 9.2) Milestone Discussion

Concerning this project's major milestones, they have been divided down by category, measured by the total number of days starting on May 5th to the completion of the project. With the exception of program development, testing, iteration and final testing, other categories such as electronics, prototype and build categories have been condensed, as the milestones for these

categories have been broken down extensively elsewhere in this document. Purchasing and acquisition has been included on the milestone chart below, which we intend to conclude prior to starting prototype development and testing. Categories such as programming and electronics will take place throughout purchasing, prototyping and the final construction of MAC, as the nature of these areas are iterative throughout the prototyping and construction phases of the project. It is estimated that the entire cycle, starting on May 5th, will take a total of 77 days to complete, concluding on July 19th. The final deadline for this project was set with the intentions of saving any necessary preparation time for the final demonstration of this project, which takes place at the end of July. The table and figure below reflect our overall milestones set.

## Milestone Gantt Chart

| TASK NAME | START DATE | DAY | END DATE | DURATION (WORK DAYS) | DAYS COMPLETE | DAYS REMAINING | PERCENT COMPLETE |
|---|---|---|---|---|---|---|---|
| **Milestones** | | | | | | | |
| Purchasing and Acquisition | 5/5 | 0 | 5/16 | 12 | 0 | 12 | 0% |
| Program Development | 5/5 | 0 | 5/31 | 27 | 0 | 27 | 0% |
| Program Testing | 5/17 | 12 | 6/21 | 36 | 0 | 36 | 0% |
| Program Iteration | 6/22 | 48 | 7/2 | 12 | 0 | 12 | 0% |
| Final Program Testing | 7/3 | 60 | 7/19 | 17 | 0 | 17 | 0% |
| Electronics Development and Testing | 5/5 | 0 | 7/2 | 59 | 0 | 59 | 0% |
| Prototype Development and Testing | 5/17 | 12 | 7/2 | 46 | 0 | 46 | 0% |
| Build Development and Testing | 7/3 | 60 | 7/19 | 17 | 0 | 17 | 0% |

Table 25: Milestone Gantt Chart Data

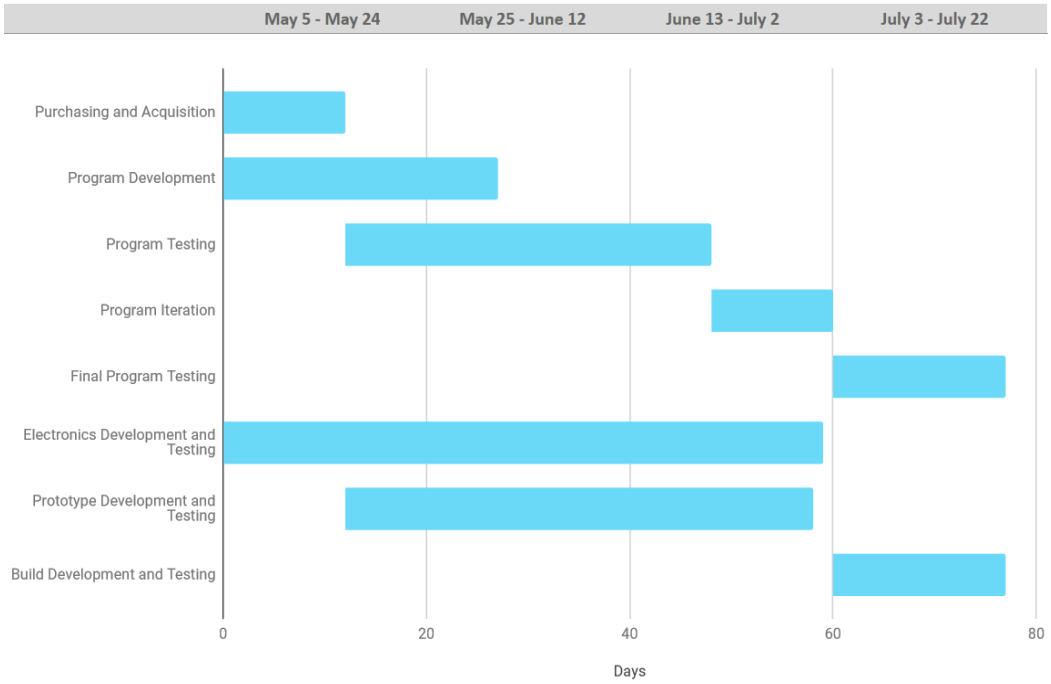



Figure 30: Milestone Gantt Chart

# 10) Project Summary and Conclusions

The field of robotics is constantly advancing with new technologies and new applications that aim to improve daily life. The MAC intends to improve the daily lives of in particular the disabled and elderly population by offering autonomous assistance in the household by helping carry daily household items from point A to point B. The MAC design is complex in both software and hardware that required many conversations about various tradeoffs that were critical in reaching the project goals. These discussions included how the user would interact with the MAC, how the MAC would traverse from one location in the house to another, the lift mechanisms and safety features. Other constraints such as time, budget and resources made an impact on which features we chose to implement and which ones were defined as stretch goals or features of the future. Despite these design constraints we believe that the MAC will be a unique and new product that solves a new problem compared to other similar household robotic products.

The MAC is a challenge to design and build due to the sheer complexity of the software and hardware design. The electrical hardware includes the drivetrain, the scissor lift, various sensors and remote that will be used to control the MAC. The integration of the hardware with several communication protocols used for the sensors and motor controllers such as WiFi, CAN, UART and analog will prove to be a design challenge in software. The MAC needs to be able to properly generate and update a map of its environment using SLAM and be able to follow a user using advanced CV with the help of a deep-learning model using the image data captured from the on-board camera. With LOS following, the MAC will be able to come to a user's location, the user can place the object on the chassis with the help of the scissor lift and then follow the user to the desired location. The MAC can also be operated remotely by the user if needed without the use of LOS following. The implementation of path planning and map generation using SLAM paves the way for future features such as the use of a companion app that can command the MAC to go to a specific location in the household through the map. Safety features such as obstacle detection and avoidance will also be key features that will ensure both the safety of the MAC and the safety of the user.

Overall, the MAC if designed and implemented correctly will meet the requirements outlined as a mobile carrier assistant robot and allow for future improvements to be made. We believe that our product has the potential to reach and greatly improve the lives of the disabled population and elderly population by not only making general everyday lifting tasks easier but also improving the quality of life and safety in their household. By using the MAC, there will be less chance of an injury due to the lifting and carrying of heavy objects such as laundry, any small furniture, and groceries.

# 11) Appendices

## 11.1) References


1. Francis, Sam. “Canvas Technology Launches 'World's First Self-Driving Carts' for Factories and Warehouses.” *Robotics & Automation News*, 20 Dec. 2017, roboticsandautomationnews.com/2017/09/08/canvas-technology-launches-worlds-first-self-driving-carts-for-factories-and-warehouse/13989/.
2. “CANVAS Technology.” *Robotics Business Review*, 15 June 2018, www.roboticsbusinessreview.com/robotic-company/directory/listings/canvas-technology/
3. Lampinen, Megan. “Voith Joins CANVAS Technology at IMTS 2018 to Showcase Intelligent Robotic Solutions Collaborating for Process Automation.” *Automotive World*, 6 Sept. 2018, www.automotiveworld.com/news-releases/voith-joins-canvas-technology-at-imts-2018-to-showcase-intelligent-robotic-solutions-collaborating-for-process-automation/.
4. 24/7, Material Handling. “CartConnect Autonomous Cart-Based Workflows - Material Handling 24/7.” *Recently Filed RSS*, 6 June 2018, www.materialhandling247.com/product/cartconnect_autonomous_cart_based_workflows/robotics.
5. Mu-Hyun, Cho. “KT Deploys 5G Autonomous Carts at Warehouses.” *ZDNet*, ZDNet, 24 May 2020, www.zdnet.com/article/kt-deploys-5g-autonomous-carts-at-warehouses/.
6. Porter, Jon. “Postmates Has Created a Robot to Speed up and Automate Its Deliveries.” *The Verge*, The Verge, 13 Dec. 2018, www.theverge.com/2018/12/13/18139394/postmates-serve-autonomous-delivery-robot-los-angeles.
7. “Database of Dimensioned Drawings.” Dimensions, www.dimensions.com/.
8. Scott, Sean. “What's next for Amazon Scout?” *About Amazon*, Amazon, 23 Sept. 2020, www.aboutamazon.com/news/transportation/whats-next-for-amazon-scout.
9. “Amazon Scout.” *Wikipedia*, Wikimedia Foundation, 20 Mar. 2021, en.wikipedia.org/wiki/Amazon_Scout.
10. “FedEx Welcomes Roxo™, the FedEx SameDay Bot to the U.A.E.” *FedEx Newsroom*, newsroom.fedex.com/newsroom/fedex-welcomes-roxo-the-fedex-sameday-bot-to-the-u-a-e/.
11. “Please Enable Cookies.” *StackPath*, www.fleetowner.com/technology/autonomous-vehicles/article/21704527/fedexs-roxo-delivery-bot-gets-chilly-reception-in-new-york-city.
12. “Roxo, the FedEx SameDay Bot.” *CSRWire*, www.csrwire.com/press_releases/718126-roxo-fedex-sameday-bot.
13. “A New Kind of Business.” *Starship*, www.starship.xyz/business/.
14. “Starship Technologies.” *Wikipedia*, Wikimedia Foundation, 14 Feb. 2021, en.wikipedia.org/wiki/Starship_Technologies.
15. Herald, The Korea. “E-Mart Unveils Autonomous Shopping Cart Eli for Test Run.” *The Korea Herald*, 17 Apr. 2018, www.koreaherald.com/view.php?ud=20180417000718.

16. Anna. “E-Mart Introduces Eli Autonomous Shopping Cart.” *Robotics & Automation News*, 4 May 2018, roboticsandautomationnews.com/2018/05/04/e-mart-introduces-eli-autonomous-shopping-cart/17113/.
17. Statt, Nick. “Amazon's New Smart Shopping Cart Lets You Check out without a Cashier.” *The Verge*, The Verge, 14 July 2020, www.theverge.com/2020/7/14/21323421/amazon-dash-cart-smart-grocery-shopping-woodland-hills-store-cashierless.
18. Amazon, 27 Apr. 2021, www.amazon.com/b?ie=UTF8&node=21289116011.
19. Russell Redman 1 | Jan 19. “Kroger Tests 'Smart' Shopping Cart from Caper.” *Supermarket News*, 19 Jan. 2021, www.supermarketnews.com/technology/kroger-tests-smart-shopping-cart-caper.
20. *Caper AI - Automated Checkout Smart Cart*, www.caper.ai/.
21. “Mobile Robots for Healthcare - Pharmacy, Laboratory, Nutrition and EVS.” *Aethon*, aethon.com/mobile-robots-for-healthcare/.
22. Matthews, Kayla, and Kayla Matthews. “Materials to Evaluate for Designing and Building Robust Robots.” The Robot Report, 19 Aug. 2019, www.therobotreport.com/materials-rugged-robot-design-building/.
23. “Aluminum.” Encyclopædia Britannica, Encyclopædia Britannica, Inc., www.britannica.com/science/aluminum.
24. “Creating a V5 Drivetrain.” KB Vex, kb.vex.com/hc/en-us/articles/360035952771-Creating-a-V5-Drivetrain.
25. “Robot Subsystem #2: Lift.” VEX ROBOTICS COMPETITION, cariwilliamzvex.weebly.com/robot-subsystem-2-lift.html.
26. “Lifting Mechanisms.” *Surrey Robotics Innovation Lab*, 14 July 2020, surreyroboticsinnovationlab.ca/lifting-mechanisms/.
27. Schaffer, Seth. “Types of Battery Systems for Robots: Custom.” Maker Pro, Maker Pro, 26 Apr. 2021, maker.pro/custom/tutorial/battery-systems-for-robots.
28. “What Is Lidar? Learn How Lidar Works.” Velodyne Lidar, 5 Jan. 2021, velodynelidar.com/what-is-lidar/.
29. “Lidar.” Wikipedia, Wikimedia Foundation, 20 Apr. 2021, en.wikipedia.org/wiki/Lidar.
30. “Detection Based on ‘Ultrasonic Waves‘ What Is an Ultrasonic Sensor?” KEYENCE, www.keyence.com/ss/products/sensor/s ensorbasics/ultrasonic/info/.
31. Jost, Danny. “What Is an Ultrasonic Sensor?” FierceElectronics, 7 Oct. 2019, www.fierceelectronics.com/sensors/what-ultrasonic-sensor.
32. “Limit Switch.” Wikipedia, Wikimedia Foundation, 21 Apr. 2021, en.wikipedia.org/wiki/Limit_switch.
33. “Encoder (Position).” Wikipedia, Wikimedia Foundation, 4 Jan. 2020, en.wikipedia.org/wiki/Encoder_(position).
34. “Rotary Encoder.” Wikipedia, Wikimedia Foundation, 24 Mar. 2021, en.wikipedia.org/wiki/Rotary_encoder#Absolute.
35. “Gyroscope.” Wikipedia, Wikimedia Foundation, 23 Apr. 2021, en.wikipedia.org/wiki/Gyroscope.
36. “Wi-Fi.” Wikipedia, Wikimedia Foundation, 22 Apr. 2021, en.wikipedia.org/wiki/Wi-Fi.

37. “Bluetooth.” Wikipedia, Wikimedia Foundation, 14 Apr. 2021, en.wikipedia.org/wiki/Bluetooth.
38. “Radio Control.” Wikipedia, Wikimedia Foundation, 25 Apr. 2021, en.wikipedia.org/wiki/Radio_control.
39. James, Michael. “Frequencies for RC Cars.” *LiveAbout*, www.liveabout.com/radio-frequencies-in-the-us-for-radio-controlled-vehicles-2862530.
40. “Cellular Network.” *Wikipedia*, Wikimedia Foundation, 22 Apr. 2021, en.wikipedia.org/wiki/Cellular_network.
41. “TI MSP430.” Wikipedia, Wikimedia Foundation, 20 Apr. 2021, en.wikipedia.org/wiki/TI_MSP430.
42. Unknown. Arduino Uno vs TI LaunchPad (MSP430 Edition), 1 Jan. 1970, humboldtmcu.blogspot.com/2014/06/arduino-uno-vs-ti-launchpad-msp430.html.
43. Engineer), Ravi Teja (Embedded. “What Are the Differences between Raspberry Pi and Arduino?” Electronics Hub, 5 Apr. 2021, www.electronicshub.org/raspberry-pi-vs-arduino/#Differences_between_Raspberry_Pi_and_Arduino.
44. “Raspberry Pi vs Jetson Nano: The Differences in 2021.” All3DP, 21 Jan. 2021, all3dp.com/2/raspberry-pi-vs-jetson-nano-differences/.
45. “Gammon Forum : Electronics : Microprocessors : Using a Keypad Matrix.” Written by Nick Gammon - 5K, www.gammon.com.au/forum/?id=14175.
46. “ROS1 Vs ROS2, Practical Overview For ROS Developers - The Robotics Back.” End, 28 Dec. 2020, roboticsbackend.com/ros1-vs-ros2-practical-overview/.
47. “Wiki.” Ros.org, wiki.ros.org/.
48. “Robot Operating System.” Wikipedia, Wikimedia Foundation, 20 Apr. 2021, en.wikipedia.org/wiki/Robot_Operating_System.
49. “Simultaneous Localization and Mapping.” *Wikipedia*, Wikimedia Foundation, 25 Apr. 2021, en.wikipedia.org/wiki/Simultaneous_localization_and_mapping#Mapping.
50. “2. The Pose of a Robot¶.” 2. The Pose of a Robot - Robotics Programming Study Guide, faculty.salina.k-state.edu/tim/robotics_sg/Pose/Pose.html.
51. Jiang, Guolai & Lei, Yin & Jin, Shaokun & Tian, Chaoran & Ma, Xinbo & Ou, Yongsheng. (2019). A Simultaneous Localization and Mapping (SLAM) Framework for 2.5D Map Building Based on Low-Cost LiDAR and Vision Fusion. Applied Sciences. 9. 2105. 10.3390/app9102105.
52. “Kalman Filter.” Wikipedia, Wikimedia Foundation, 26 Apr. 2021, en.wikipedia.org/wiki/Kalman_filter.
53. “Localization, Mapping, SLAM and The Kalman Filter According to George.” CMU Computer Science, Carnegie Mellon University, www.cs.cmu.edu/~motionplanning/lecture/Chap8-Kalman-Mapping_howie.pdf.
54. OpenSLAM.org, openslam-org.github.io/openratslam.html.
55. M. J. Milford, G. F. Wyeth and D. Prasser, "RatSLAM: a hippocampal model for simultaneous localization and mapping," IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004, New Orleans, LA, USA, 2004, pp. 403-408 Vol.1, doi: 10.1109/ROBOT.2004.1307183.

56. “Particle Filter.” Wikipedia, Wikimedia Foundation, 17 Mar. 2021, en.wikipedia.org/wiki/Particle_filter.
57. OpenSLAM.org, openslam-org.github.io/gmapping.html.
58. “Wiki.” *Ros.org*, wiki.ros.org/gmapping.
59. Grisetti, Giorgio. Introduction to Navigation Using ROS. Sapienza University of Rome, www.diag.uniroma1.it//~nardi/Didattica/CAI/matdid/robot-programming-ROS-introduction-to-navigation.pdf.
60. “Space-Filling Tree.” Wikipedia, Wikimedia Foundation, 1 Jan. 2018, en.wikipedia.org/wiki/Space-filling_tree.
61. “Wiki.” Ros.org, wiki.ros.org/hector_mapping.
62. Konolige, Kurt & Grisetti, Giorgio & Kümmerle, Rainer & Burgard, Wolfram & Limketkai, Benson & Vincent, Régis. (2010). Efficient Sparse Pose Adjustment for 2D mapping. Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems. 22-29. 10.1109/IROS.2010.5649043.
63. “Wiki.” Ros.org, wiki.ros.org/slam_karto.
64. Redmon, Joseph. *YOLO: Real-Time Object Detection*, pjreddie.com/darknet/yolo/.
65. “Pololu IR Beacon Transceiver.” Pololu Robotics & Electronics, www.pololu.com/product/701.
66. Bjerknes, Jan & Liu, Wenguo & Winfield, Alan & Melhuish, Chris. (2009). Low Cost Ultrasonic Positioning System for Mobile Robots.
67. “Technology.” Infsoft, www.infsoft.com/technology/positioning-technologies/rfid.
68. “Wi-Fi Positioning System.” Wikipedia, Wikimedia Foundation, 19 Apr. 2021, en.wikipedia.org/wiki/Wi-Fi_positioning_system.
69. “Introduction to the Google Assistant Service  |  Google Assistant SDK.” Google, Google, developers.google.com/assistant/sdk/guides/service/python.
70. Lamberterie, Charles de. “Les Adjectifs Grecs En -Us: sémantique Et Comparaison.” Amazon, Peeters, 1990, developer.amazon.com/en-US/docs/alexa/alexa-skills-kit-sdk-for-python/overview.html.
71. Perez, Sarah. “Nearly 70% of US Smart Speaker Owners Use Amazon Echo Devices.” TechCrunch, TechCrunch, 10 Feb. 2020, techcrunch.com/2020/02/10/nearly-70-of-u-s-smart-speaker-owners-use-amazon-echo-devices/.
72. T. Jacobs and G. S. Virk, "ISO 13482 - The new safety standard for personal care robots," ISR/Robotik 2014; 41st International Symposium on Robotics, Munich, Germany, 2014, pp. 1-6.
73. "IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.
74. “ISM Radio Band.” *Wikipedia*, Wikimedia Foundation, 26 Apr. 2021, en.wikipedia.org/wiki/ISM_radio_band.
75. “IEEE Code of Ethics.” IEEE, www.ieee.org/about/corporate/governance/p7-8.html.
76. “ RoHS Guide.” *2021 RoHS Compliance Guide: Regulations, 10 Substances, Exemptions*, www.rohsguide.com/.

77. *SIOS*, support.industry.siemens.com/cs/document/109482756/sinamics-g120-simogear-simatic%3A-scissor-lifting-table?dti=0&lc=en-WW.
78. “Extreme Max 5001.5044 1000 Lbs. Motorcycle Scissors Jack - Wide.” Amazon, www.amazon.com/Extreme-Max-5001-5044-Motorcycle-Scissor/dp/B0196PAZ1S/ref=sr_1_3?dchild=1&keywords=motorcycle scissor jack&qid=1617316282&sr=8-3 wiki.dfrobot.com/URM13_Ultrasonic_Sensor_SKU_SEN0352.
79. “URM13 Ultrasonic Sensor for Arduino/Raspberry Pi Wiki.” DFRobot, wiki.dfrobot.com/URM13_Ultrasonic_Sensor_SKU_SEN0352.
80. “LV-MaxSonar-EZ_Datasheet.” MaxBotix, MaxBotix, 2015, www.maxbotix.com/documents/LV-MaxSonar-EZ_Datasheet.pdf.
81. “ZM Series, Mini Size Snap-Acting Switches.” C&K, 11 Nov. 2020.
82. “D2HW Sealed Ultra Subminiature Basic Switch.” Omron, Omron Corporation, omronfs.omron.com/en_US/ecb/products/pdf/en-d2hw.pdf.
83. “Series MS Switches, Snap Action Switches - Miniature.” Amazon AWS.
84. Logitech C922x Pro Stream Webcam – Full 1080p HD Camera , Amazon, www.amazon.com/Logitech-C922x-Pro-Stream-Webcam/dp/B01LXCDPPK/ref=sr_1_3?dchild=1&keywords=webcam&qid=1619486524&sr=8-3.
85. 1080P Web Camera, HD Webcam with Microphone & Privacy Cover, 2021 NexiGo N60 USB Computer Camera, 110-Degree Wide Angle, Plug and Play, for Zoom/Skype/Teams/OBS, Conferencing and Video Calling . Amazon, 1080P Web Camera, HD Webcam with Microphone & Privacy Cover, 2021 NexiGo N60 USB Computer Camera, 110-degree Wide Angle, Plug and Play, for Zoom/Skype/Teams/OBS, Conferencing and Video Calling .
86. #393720, Member, et al. “SparkFun Transceiver Breakout - nRF24L01+.” WRL-00691 - SparkFun Electronics, www.sparkfun.com/products/691.
87. “RobotDyn nRF24L01+ 2.4GHz Wireless Transceiver.” Www.addicore.com, www.addicore.com/Robotdyn-nRF24L01-p/ad278.htm.
88. “STM8L101F1.” STMicroelectronics, www.st.com/en/microcontrollers-microprocessors/stm8l101f1.html.
89. “MSP430FR21xx, MSP430FR2000 Mixed-Signal Microcontrollers.” Texas Instruments, Dec. 2019.
90. “USB Peripheral/Host Controller with SPI Interface.” Maxim Integrated, 13 July 2020.
91. Microchip. (n.d.). USB-to-SPI Protocol Converter with GPIO (Master Mode).
92. “Wiki.” *Ros.org*, wiki.ros.org/navigation.
93. “Run Apps on the Android Emulator : Android Developers.” *Android Developers*, developer.android.com/studio/run/emulator.html#wi-fi.
94. says, Rosered, et al. “Difference Between Bluetooth and Wifi (with Comparison Chart).” *Tech Differences*, 28 Dec. 2019, techdifferences.com/difference-between-bluetooth-and-wifi.html#:~:text=The%20radio%20signal%20range%20provided,it%20is%20high%20in%20Wifi.
95. “CAN Bus.” *Wikipedia*, Wikimedia Foundation, 16 Apr. 2021, en.wikipedia.org/wiki/CAN_bus.

96. “Creative Commons License Deed.” *Creative Commons - Attribution-ShareAlike 4.0 International - CC BY-SA 4.0*, creativecommons.org/licenses/by-sa/4.0.
97. “AM14U4 - 6 Wheel Drop Center Robot Drive Base - FIRST Kit of Parts Chassis.” *AndyMark Inc*, www.andymark.com/products/am14u4-kit-of-parts-chassis?via=Z2lkOi8vYW5keW1hcmsvV29ya2FyZWE6OkNhdGFsb2c6OkNhdGVnb3J5LzViYjYzNDM5YmM2ZjZkNmRlOGU2YWNhNg.
98. “AM14U4 - 6 Wheel Drop Center Robot Drive Base - FIRST Kit of Parts Chassis.” *AndyMark Inc*, www.andymark.com/products/am14u4-kit-of-parts-chassis?via=Z2lkOi8vYW5keW1hcmsvV29ya2FyZWE6OkNhdGFsb2c6OkNhdGVnb3J5LzViYjYzNDM5YmM2ZjZkNmRlOGU2YWNhNg.
99. “NI RoboRIO.” AndyMark Inc, 16 Feb. 2020, www.andymark.com/products/ni-roborio?via=Z2lkOi8vYW5keW1hcmsvV29ya2FyZWE6Ok5hdmlnYXRpb246OlNlYXJjaFJlc3VsdHMvJTdCJTIyYnV0dG9uJTIyJTNBJTIyc2VhcmNoJTIyJTJDJTIycSUyMiUzQSUyMnJvYm9SSU8lMjIlMkMlMjJ1dGY4JTIyJTNBJTIyJUUyJTlDJTkzJTIyJTdE.