Shelf Buddy

Authors: Ludi Cao, Esther Jang, Bhumika Kapur: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— ShelfBuddy is an assistive robot that helps users grab objects from high shelves. When going grocery shopping, reaching objects on a shelf can be a very difficult task, especially for those with disabilities. ShelfBuddy makes grocery shopping an easier and a more accessible process by allowing users to focus on just looking for their items. A user can point to an object on a shelf using a laser pointer, and ShelfBuddy will be able to grab the object from the shelf and place it in the user's basket.**

***Index Terms*—AprilTags, Autonomous Navigation, Computer Vision, Control, Laser Point Detection, Linear Slides, Robotics**

1 INTRODUCTION

Grocery shopping is intended to provide people with the option of selecting their own goods and thus be a seamless and enjoyable experience. Unfortunately, it is still a difficult task for people with disabilities as many objects are located on higher shelves that are out of reach. Current solutions are also inconvenient or require other people's assistance.

ShelfBuddy is an autonomous robot system that can retrieve items on higher grocery shelves for users, ensuring that grocery shopping is a personalized experience for everyone. The user first selects items by pointing a laser on the object they want. Then, after detecting the laser point, ShelfBuddy will navigate to the object on the shelf, grab the object, and deliver the item to the user's basket. This solution allows users in need to have more control over their grocery shopping experience and improves their accessibility.

Our current solution applies to shopping medical containers typically found at a pharmacy. We identify that shoppers in this environment are more likely to require assistance. In addition, the testing inputs would be more controlled for us to test. The requirements for our solution were chosen with the intention of making ShelfBuddy a user-friendly system. Our primary requirements are the following:

1. The overall success rate of our system, which is the entire process of detecting the item being pointed at, retrieval of the object, and navigation to and from the shelf and basket, should be 97.5%

2. Requirement 1 depends on the success rate of three subsystems. The first requires the success rate of the navigation to and from the shelf and basket. We would like this value to be 98.5%.

3. The second subsystem in which we would like to measure its success rate is the detection of laser pointer. We would like the robot to correctly detect an object pointed with a laser with a success rate of 99.5%

4. The third subsystem to measure is the process of retrieving the item on the shelf, and holding on to the item when sending to the basket. We would like the success rate to be 99.5%

5. The robot should travel at a speed of 0.5 m/s.

6. The latency for grabbing an object when at the shelf should be 3 seconds.

7. The latency of processing a snapshot of the shelf for the laser point is 1 second.

8. The distance between items on the shelf is roughly 2 inches.

9. The dimension of the shelf is 3ft $\times$ 4ft $\times$ 1ft

10. The robot will retrieve items sized around 3 inches and weighing a maximum of 1 pound.

11. The robot should be user-friendly in terms of space occupancy within the grocery store and ease of use.

2 DESIGN REQUIREMENTS

Requirement 1: The average number of items a customer would purchase per visit at a popular pharmacy such as CVS is 4 [1]. Our robot would want to hit the benchmark of successfully meeting the user's shopping needs 9 times out of 10 visits. We believe this would meet user satisfaction levels. On average 10 visits at the store would require 40 runs, so at lest 39 of them should be successful. This corresponds to a success rate of $\frac{39}{40}$. To test this requirement, we plan to run 40 complete trials, where after executing the program the robot first follows the user, scans through the shelf to find the pointed object, successfully retrieves the object and places it to the grocery basket. We expect the entire process to succeed 39 times.

Requirement 2: Based on the overall success rate from *Requirement 1*, we determined that the success rate of the navigation process is 98.5%. This success rate is slightly lower than the expected success rate based on the overall success rate. We determined that navigation would have a poorer success rate because of the fact that compared to the other subsystems, navigation involves more moving components and has to deal with the complexity behind traversing across an area.

Requirement 3: Based on the overall success rate from *Requirement 1*, the success rate for laser detection is 99.5%. This is the anticipated success rate of the subsystem based on the overall rate, and we decided that this was reasonable given that laser detection is not as complex as navigation but also not straightforward enough to have a lower error rate.

Requirement 4: Based on the overall success rate from *Requirement 1*, the success rate for object retrieval is 99.5%. This is the highest accuracy among the three sub systems which contribute to the overall success rate. We believe that given successful navigation and item detection, the robot would be positioned in a straight angle that should be easy to grab the object, so there is little room left where error could happen.

Requirement 5: To provide user-friendly assistance, ShelfBuddy should travel at a speed that is close to the average speed a user can travel in a wheelchair, which is 0.79m/s. [2] This would ensure that using ShelfBuddy would take roughly the same amount of time as the user travels.

Requirement 6: The latency of grabbing an object is 3 seconds because while it is slightly slower than the average latency of the time taken for a human to grab an object, it is still not unreasonably long for the user's experience.

Requirement 7: The latency of processing a laser pointer on an object will be 1 second. Research shows that the average time a human takes to process visual stimuli is .25 seconds [3]. Thus, a one second laser processing time is within reason in comparison to the amount of time it would take a person to detect the laser.

Requirement 8: Typically, two items on a shelf are on average tightly packed on a grocery store. However, we would be operating under the assumption that the gap is 2 inches in order to feasibly achieve our grabbing requirements. This is still as minimum of an increase from the real world scenario.

Requirement 9: Our shelf mirrors a typical shelf found in a pharmacy store in most dimensions. However, because the actual height would unfortunately introduce too much complexity for our project to manage in the short time frame, we chose to limit our height to 3 feet. Our project is hence more of a proof of concept for this idea that would be blown into larger proportions in the real world.

Requirement 10: Our use case of the robot would be to retrieve medicine containers typically seen at a CVS pharmacy. The items are around 3 inches and weigh at most 1 pound.

Requirement 11: Our use case is within a grocery store with other shoppers and carts that the robot will have to share narrow aisle space with. The robot should hence cause minimal interference in order to succeed in our use case. It should also be simple to use since its intention is to improve a shopper's experience and hence not cause unnecessary inconvenience.

3 ARCHITECTURE OVERVIEW

A labeled figure of our robot (Figure 1, page 3) is shown at the top of the following page and an informational flow chart of our system (Figure 18, page 18) is shown in the appendix. The overall mechanical design has not changed significantly from our design review. The main difference for our block diagram is that we added an IMU to our system, where we record the angle of the robot to counteract against the drift it was experiencing. We replaced using two Arduino Uno boards to one Arduino Mega board, since the larger board is sufficient for our purpose and we ran into communication issues with multiple boards. We also made explicit how we utilize two buck converters to power the servo and the motor drivers with the same battery as the motors. Finally, we added a separate 12V power bank for powering the Xavier. Our system is roughly split into three sub systems: the drive train and navigation algorithm, the detection and item recognition algorithm, as well as the linear slide and retrieval system. We use a Jetson Xavier as the main computing board, and an Arduino Mega as the controller of the motors, servo, and IMU. The computer vision algorithms are run on the Jetson Xavier based on frames taken by the Intel D435 camera. Information is passed between the Xavier and the Arduino Mega board connected with a USB cable via sending/decoding messages on the serial terminal.

3.1 Wheel Chassis Design

The wheel chassis is driven by four omni-directional wheels placed at diagonal directions. These omni-directional wheels come in packs of two which give the robot much more stability and less drift. Each pair of wheels is controlled by its own DC motor, where it's mounted to aluminum extrusions that form the base of our system. The chassis base is an octagonal shape such that the four sides at the corners connecting to the wheels are shorter than the four sides on the side.

3.2 Computer Vision

We use an Intel RealSense D435 Depth Camera, which has high resolution and depth sensing technology. The camera allows our robot to travel throughout the grocery store, follow the user, and navigate to the shelf to retrieve the object.

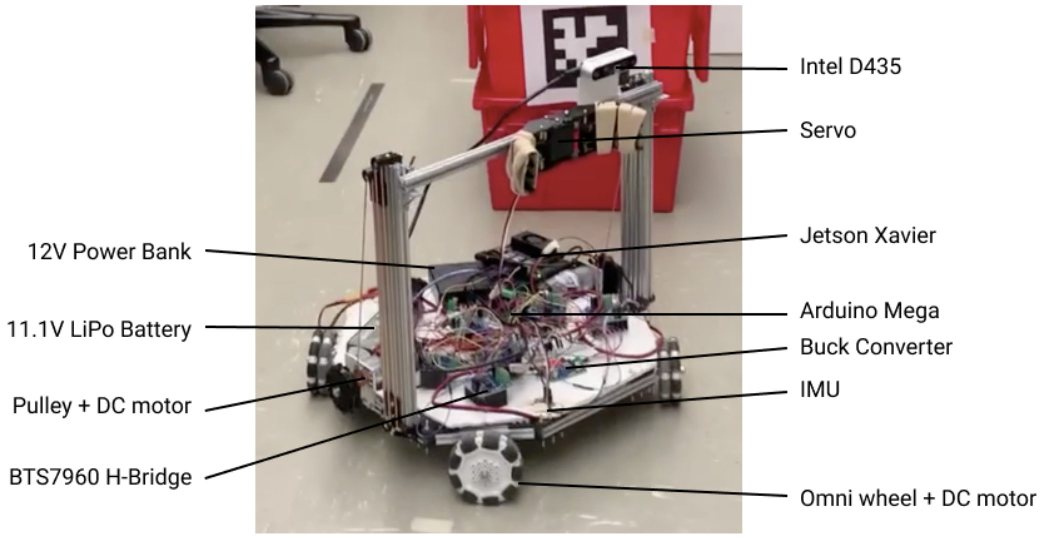


Figure 1: Labeled Robot

3.2.1 Navigation:

We use AprilTags, a visual fiducial system, Figure 2, for navigation. The tags were printed on paper and were attached to the shelf and the user's basket. Using the April-Tag detection software, our robot can follow the user's basket throughout the store. The AprilTag detection software outputs the tag information and its precise 3D location in relation to the camera, including the angle. The detection software is highly accurate, within 4 centimeters of the actual position when the camera is 2 meters away from the item [4]. This information allows our robot to navigate accurately and efficiently.

3.2.2 Item Detection:

After the robot has reached the shelf and extended the slides, it runs the edge detection function to draw bounding boxes around the items on the shelf. The bounding boxes provide the exact location of the items, and are used when grabbing the item and searching for the laser.

3.2.3 Laser Recognition:

Once the bounding boxes have been computed, the robot searches for the laser point inside each of the bounding boxes. If the laser point is detected, the robot travels to the object with the laser point, and proceeds to retrieve the item.

3.2.4 Changes from Design Review

The order of events in the final project was a little different from the design review. In the design review, we had planned to run the laser recognition algorithm first, and use the location of the laser to find the items. However, once we began testing we found that items in the background of the frame resulted in the laser being detected in the wrong location. To address this, we had to rearrange the two algorithms so that first the bounding boxes were calculated, and then the robot searched for the laser within the bounding boxes. This sequence is also better in terms of correctness, as we would not want the robot to travel to the laser if the user is not shining it on a product, such as on the back or sides of the shelf.

3.3 Retrieval System Design

3.3.1 Claw Gripper End Effector

We use a PWM servo-powered claw for our end effector. This servo was used selected because its torque load fit our use case specification.

3.3.2 Z-Axis Linear Slide System

We use a motorized pulley-powered linear slide system for the z-axis linear slide system. We have 2 mirrored, parallel linear slide systems mounted on both sides of the chassis and powered by 2 DC motors (1 each

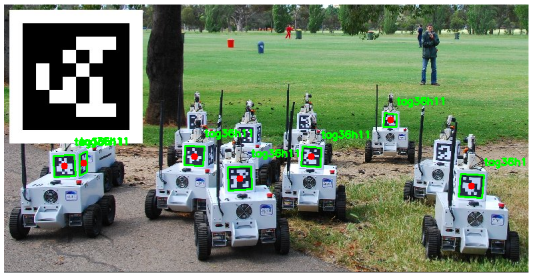


Figure 2: AprilTag example.

respectively). We built 2 sets of linear slides in order to keep the system stable during the extension process and while extended.

4 DESIGN TRADE STUDIES

4.1 Wheel Selection Trade-offs

We had a few options of wheel types for our system drive train. To meet *Requirement 2*, the robot had to move in different directions with high precision, since navigation involves following the user in one direction, and moving towards the shelf in a potentially perpendicular direction. The robot also needed to be able to position itself parallel towards the shelf from any angle, since the retrieval process required high precision from the direction of the wheel base. Therefore, the choice of using normal wheels would have been restricting, since the robot would then constantly perform turning which might accumulate high inaccuracy.

We then decided from two popular omni-directional wheel choices: omniwheels vs mecanum wheels. This required our system to have four DC motors for each wheel. Although this was incovenient, we believe this was more advantageous than using normal wheels connected to 2 DC motors, as it saved us from putting significant cost in directional control. Both wheels were configurable and programmable in a way that enabled it to travel in eight directions. The main advantage of mecanum wheels was that they had greater traction to the floor, which helped in terrains where the road conditions are less than ideal. However, this was not useful to our use case environment, and its cons of being more costly, heavier, and less accurate in directional control were far more problematic.

As for wheel size, there were 60 mm diameter choices and 90 mm diameter options that the vendor provides. Smaller wheel sizes provided more torque and large wheel sizes provided more speed. Since the robot was travelling on a smooth (low friction constant) and flat (no vector from weight acting against) floor, we anticipated that the torque inserted on the wheels would not be a main concern. However, to satisfy *Requirement 5*, we did want to optimize the speed of the robot as much as possible. Hence, we decided to use the 90 mm diameter wheels. As later verified in section 4.2, this decision would work well with our choice of motors and provided the speed that would meet the requirement.

4.2 Motor Selection Trade-offs

We selected the appropriate motor to satisfy *Requirement 5*. We referred to equations

$$w = 2\pi f \tag{1}$$

$$v = rw \tag{2}$$

where f represents the rotational frequency, w refers to angular velocity, v represents velocity of robot, and r represents radius on the wheels. Combining equations (1) and (2), we derived

$$\begin{aligned} v &= \frac{D}{2} \cdot 2\pi f \\ &= \pi D f \\ &= \pi \cdot 0.09m \cdot \frac{RPM}{60s} \end{aligned}$$

The weight of our system included sub components as follows: (Length of extrusions is justified in section 4.3, where we derive the weight)

Item	**Weight(kg)**
Wheel	$0.07 \times 8 = 0.56$
Extrusion	$9\text{m} \rightarrow 2$
Electrical components	1.5
Claw and object weight	1.5

Total: 5.56 kg + 6 × motor weight

Since our system incurs much weight, we estimated a 25% reduction from max speed of motor. Hence, the expected speed of our robot was

$$\begin{aligned} &\pi \cdot 0.09m \cdot \frac{RPM}{60s} > \frac{0.5m/s}{0.75} \\ &\Rightarrow RPM > 141.47\text{rpm} \end{aligned}$$

Hence, the robot required an RPM of at least 141.47 rpm, so we couldn't use another common standard of 100 rpm motors. We also wanted to limit the weight of the motors, as higher weight would cause greater reduction in max speed. This is because if the gain in rpm is small, then the overall speed requirement might not be met.

The vendor that we ordered from provided a couple of DC motor options, including a planetary gear motor with a 40:1 gear ratio, a planetary gear motor with a 20:1 gear ration, a spur gear motor with a 40:1 gear ratio, as well as a spur gear motor with a 20:1 gear ratio. The 40:1 gear ratio motors provide 150 rpm at stall torque of 4.2 Nm, and the 20:1 gear ratio motors provide 300 rpm at stall torque of 2.1 Nm. Although an rpm value of 150 rpm would theoretically meet the speed requirements according to calculations, the requirements would barely be met under the ideal case. We also used the maximum specs to compare, and maximum specs were not always achieved during experiments. The cost of using a 20:1 gear ratio motor is that its stall torque is reduced by half. However, given that each motor weighs at most 0.5 kg, the total weight would be at most 8 kg. With the assumption that the robot travels on a flat and friction-less surface, the robot would require the torque to provide a force such that $F = ma$ to start from the stopped state to the start state. Assume $a = 0.25m/s^2$. The output shaft length is 0.04 m, so the force that can be applied by one motor at least $2Nm/0.04m = 50N \approx 5kg$. We power 4 DC motors in total, so the max weight of our robot could be 20 kg. Note that most of the numbers we used to estimate the minimum torque load are worse than worst case situations. The conclusion is that the stall torque specs for either motor should work for our system.

Initially our team decided to choose the spur gear motors with 20:1 gear ratios followed from the calculations above. We also chose the spur gear motors over planetary motors because our system isn't complex enough to justify the usage of planetary motors, where the advantage is prominent during high loads. The spur gear motors are also lighter, cheaper, and perform equally if not better in efficiency. Unfortunately, the 20:1 gear ratio motors were out of stock. The only feasible option was to use the planetary gear motor with a 20:1 ratio, with a slight increase in 20 dollars for our budget.

4.3 Motor Driver Trade-offs

We noticed the Capstone Inventory had a L298N based motor controller, and the model is a popular choice among similar projects. Our motor requires 12 V and has a maximum output power of 15 W. This means that the motor needs at least 15 W / 12 V = 1.25 A. The maximum allowable current of an L298N is 2 A, which is too close to the lower bound that our motor needs. To err on the safe side, we decided to use the BTS7960 DC stepper motor drivers, as the maximum allowable current is 43 A

4.4 Wheel Chassis Trade-offs

We designed the separate mechanical components in SolidWorks to ensure the components were physically compatible with each other. Our initial design is shown in Figure 3. Compared to the current iteration, the main difference between the two designs is size. To satisfy *Requirement 9*, we were concerned that having a base size much smaller than the height of the reaching claw would cause the robot to be unbalanced, especially since the claw would extend horizontally further than the front of the robot. The previous size of the design was roughly 70 cm $\times$ 70 cm, where the 4 longer extrusions were 55 cm long. However, after preliminary calculations, we realized the initial concerns were redundant: The weight of the motors themselves are 0.45 kg $\times$ 6 = 2.7 kg. From *Requirement 1* our item weight is 0.45 kg. Additional components that reach in front of the robot include an Intel Realsense camera, weight around 0.072 kg, and the weight of the claw itself. Assuming the claw extends in front of the robot base for 20 cm, and according to lever theory in physics, we estimate there is

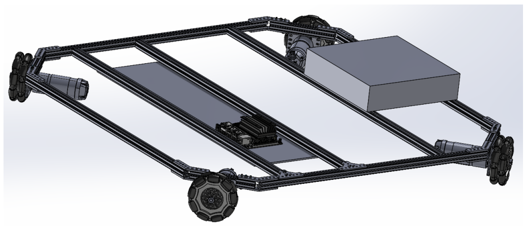


Figure 3: First chassis design

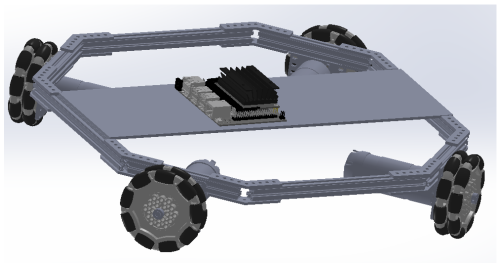


Figure 4: Current chassis design

sufficient normal force from the ground that would balance the robot and prevent it from "tipping" forward.

Therefore, our goal was to make the chassis base as small as possible, since a smaller size means smaller weight and more power efficiency. We wanted to mount the linear shaft on the side of the base, which is 1.5 cm $\times$ 4 = 6 cm in length. The side also included the length of the motor and the size of its mount. We also included the margins of connecting brackets, which take up roughly 8cm. Hence, we designed the four longer extrusions to be 20cm in length, which sufficiently fit these components without wasting too much extra space. We also wanted to fit all the electrical components on the board mounted on top of the extrusions, including a Jetson Xavier, one Arduino, a breadboard, a power supply, motor drivers, and other miscellaneous items. From rough calculations we found a board size of 8 cm and the length across the base was sufficient. Note that the extrusions at the corners which mount to the wheels were designed to be of length 12 cm, since the diameter of the wheels are 9 cm.

The top down design for the current chassis is shown above in Figure 4.

4.5 Battery Selection Trade-offs

The nominal voltage of our motors is 12V. We decided to use rechargeable batteries, which could be of material NiMH or LiPo.

We looked at two reasonable models correspondingly, the "Tenergy NiMH Battery Pack 12V 2000mAh High Capacity Rechargeable Battery", as well as the "Ovonic 11.1V 5000mAh 3S 50C-100C LiPo Battery", two of which most suited our needs. Including the charger, the NiMH battery is around 10 dollars more expensive than the other option.

We first calculated the energy needed for our robot. For power consumption using the NiMH battery, given the maximum rated power of the motors is 15 W, the maximum current each motor draws would be 1.25 A. The wheel chassis which includes 4 motors connected in parallel would draw 5 A current. With the assumption that the motors controlling the linear slide system are not consistently turning, we estimated an average of 6 A current drawn throughout one run. With a 2000 mA h capacity, this means that a single charge would only last for 2 / 6 h = 0.333 h ≈ 20 min, which is inconvenient for consistent testing. It also violated *Requirement 11*, as the robot would have to be constantly recharged, and a user-friendly system should not only operate for a short time frame. In addition, the specs state that a recharging rate of 1 A maximum is recommended. Therefore, the battery needs to charge itself for 2 hours, which worsens the problem of a short duration per charge.

We then considered the power consumption associated with the usage of the LiPo battery. Most LiPo rechargeable batteries come at a standard size of 11.1 V. We decided this 1 V difference would not significantly impact performance. Given the maximum rated power of the motors is 15 W, the maximum current each motor draws would be 15 W / 11.1 V = 1.35 A. The wheel chassis which includes 4 motors connected in parallel would draw 5.4 A current. In a similar fashion, we estimate the average current drawn through a run would be 6.5 A. With a battery of 5000 mA h capacity, a single charge would last for 5 / 6.5 h = 0.769 h ≈ 46 min. A charge rate of 1C is the recommended value for LiPo batteries [5], which corresponds to a current of 5 A. Therefore, it would take one hour to fully charge the battery. This is a significant improvement in timings compared to the NiMH battery. Therefore, we decided to use the 11.1 V LiPo batteries as our power source.

We also chose to use a separate 12V power bank for the Xavier since it requires a 9-20V power supply and we already were drawing a lot of current from our LiPo battery.

4.6 End Effector Trade-offs

We wanted to select an end effector that would be able accessible for us in terms of money and availability while still emulating modern solutions. At first, we explored using a vacuum suction gripper end effector as this was the most common solution being used in current industry for robots similar to our intended use case. Using a vacuum suction gripper would have also given us the ability to have a larger margin of error for getting bounds of an object and gripping items of various shapes and sizes. Unfortunately, such vacuum suction gripper parts were unavailable to us as they were either too costly for our budget or would not have shipped in our tight time frame. Hence, we pivoted our end effector design choice to using a claw gripper, as this was the second most popular end effector option we found in our research.

When researching claw gripper options, the most popular choice that was accessible for us and would meet our use case was a servo-powered claw. We found that we could either purchase an off-the-shelf part for out of the box use, or we could design and custom cut or print a claw. Based off our research, custom designing our own claw had the advantage of potentially improved gripping strength and the ability to grab larger objects. However, we determined that the time cost of this approach was significantly larger than its added value as there was not much existing information regarding it to work off of. As a result, we opted for purchasing an off-the-shelf servo claw part that would meet our requirements. Since choosing to purchase the part returned an extra week of time that would have been spent building and testing the claw part, we decided to allocate this time towards integration. This is a highly beneficial investment of time because our project has so many subsystems that are required to integrate completely for the system to work and thus was one of the most complicated, time-consuming, and failure-prone parts of our project.

To satisfy our *Requirement 10*, we had to find a claw gripper that has a maximum gripping width of at least 3 inches. We also needed a servo that would power the claw to provide a gripping force and have a coefficient of friction with the object as defined in the following:

$$\begin{aligned} F_{grip}\mu &\geq (0.4536\text{kg})(9.8\frac{\text{m}}{\text{s}^2}) \\ &\geq 4.4463\text{N} \end{aligned}$$

To find the part that would satisfy these requirements, we considered many vendors. We first looked at Robotshop.com as there were many claw gripper options from various brand-name robot part vendors. However, we had a lot of difficulty finding a part that could satisfy our width requirement and had a promising design. Hence, we looked to Amazon.com instead as even though the vendors may have been slightly more off-brand, we knew that the part would arrive with faster shipping time and allow us to spend more time testing to compensate for this concern.

We found 2 servo claws that seemed to meet our *Requirement 10.*

Both were from the same vendor, and one had a larger linear gripping block (claw A) while the other had a more rounded claw design (claw B). Based on its specifications, claw A has a link radius of 43mm. This means that for claw A to grab an object of width 3 inches, a single claw side must open $\theta = 62.38°$. Similarly, based on its approximate specifications, claw B has a link radius of 100mm. This means that for

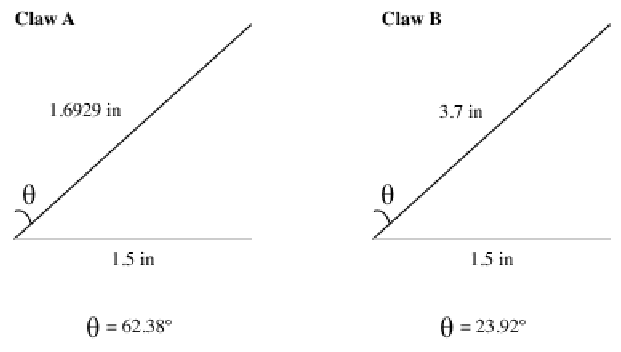


Figure 5: Claw Angle Calculation

claw B to grab an object of width 3 inches, a single claw side must open $\theta = 23.92°$.

Originally, we had assumed that the aluminum claw would be capable of gripping our materials, but the claw did not have a large enough coefficient of friction to properly hold onto our cardboard boxes. Hence, we wrapped rubber bands around the claw which gave us a static coefficient of friction (between rubber and cardboard) of at least 0.5 [6]. This means that we could find approximately find the torque, τ_{servo}, of the servo needed using following:

$$\begin{aligned} F_{grip} sin(\theta) r &= \tau_{servo} \\ F_{grip} &= \frac{\tau_{servo}}{sin(\theta) r} \\ \frac{\tau_{servo}}{sin(\theta) r} &\geq \frac{4.4463\text{N}}{\mu} \\ \tau_{servo} &\geq \frac{4.4463\text{N} sin(\theta) r}{0.5} \\ \tau_{servo} &\geq 8.893\text{N} sin(\theta) r \end{aligned}$$

This meant that claw A would require a servo with a torque of at least $8.893\text{N}sin(62.38°)(0.043\text{m}) = 0.339\text{Nm}$ Similarly, this meant that claw B would require a servo with a torque of at least $8.893\text{N}sin(23.92°)(0.1\text{m}) = 0.361\text{Nm}$ The claw kits were from the same vendor and came with a PWM servo of torque 1.96Nm which was more than sufficient for meeting our requirements.

We purchased both claw options and manually tested each claw to determine which shape was best suited for gripping objects in our use case, given that both otherwise satisfy our end effector requirements. In the end, we chose claw B because the shape of the claw was significantly better suited for grabbing our items due to its larger gripping range.

4.7 Linear Z-Axis Actuating System Trade-offs

For our linear z-axis actuating system, we made several alternative design considerations with a focus on meeting the height aspect of *Requirement 9.* The industry standard choice would have been to use a link arm robot, but we decided that this would introduce unnecessary complexity to our project since we only need actuate on the z-axis and would not use the extra degree of freedoms offered. Another alternative was to use a linear guide rail system, but we did not consider this choice as it would have occupied too much vertical space. This would have caused our robot to pose as a larger interference to others, especially when not in use, and hence did not meet our *Requirement 11.* Hence, we decided to use a cascading linear sliding system and ultimately ended up deciding between a pulley powered linear slide system or a cascading rack and pinion system. We ended up selecting the pulley powered linear slide system as we wanted to reach a large height which would require several sets of rack gears and would introduce unnecessary complexity and cost.

The total mass of the lifted system is $m_{extrusions} + m_{slideparts} + m_{camera} + m_{claw_system} = 0.44\text{kg} + 0.1\text{kg} + 0.3\text{kg} + 0.2\text{kg} = 1.04\text{kg}$. This means that the lift had to be able to handle a force of $(1.04\text{kg})(9.8\frac{\text{m}}{\text{s}^2}) = 10.19\text{N}$. We also wanted to have 2 parallel sliders to keep the system balanced. Hence, each motor and pulley pair that we chose to control the system had to be able to handle at least 5.095N of force. The motor we chose had $4.2Nm$ of stall torque. We also opted for a pulley with smaller radius of 60mm, which resulted in a force of $\frac{4.2\text{Nm}}{0.06\text{m}} = 70N$ which was more than enough force.

4.8 Computer Trade-offs:

When choosing the computer for our project we considered two main options, the Jetson Xavier and the Raspberry Pi. Both computers were in the ECE inventory, so price was not a concern. Some members of our group were familiar with the Raspberry Pi, while none of us had experience with the Xavier NX. In terms of specifications, the Raspberry Pi has a BroadCom VideoCore VI GPU, 2-4GB of RAM, a 40 GPIO pin header and a variety of ports. [7] In contrast, the Xavier NX has a 384-core NVIDIA Volta™ GPU, a 7-way visual processor, and 8GB of memory.[8] We chose to use the Jetson Xavier due to the extra memory, and the fact that all our research concluded that the Jetson Xaiver was superior when it came to AI applications. Since we planned to use Computer Vision in our project, we wanted to select a computer which would best support this.

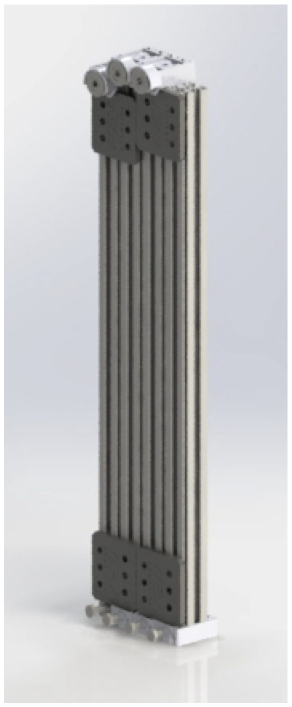


Figure 6: CAD of one side of linear slide part

4.9 Camera Trade-offs:

We mainly considered two cameras for our project, an Intel Realsense or the Arducam UC-698 along with a depth sensor. Both cameras were available to us at no cost, and the distance sensor would also be inexpensive. The Arducam has a 75 degree FOV, 4k@30FPS frame rate [9]. In comparison the Intel Realsense 435 has an 85 degree depth FOV and 90 FPS depth frame rate, 69 degree RGB FOV and 30 FPS RGB frame rate [10]. The FOV and frame rate of both cameras were similar, but the main difference between the two cameras was the depth sending capability of the RealSense. Given that our robot must approach the shelf and move the claw a precise amount to retrieve the object, we wanted to select the camera that would provide us with the most accurate depth information. Thus, we chose the Intel Realsense for our system.

5 SYSTEM DESCRIPTION

5.1 Hardware Connections

As explained in the design trade studies, we decide to use the BTS 7960 DC stepper motor drivers due to the current requirements from our motors. One disadvantage that comes along with this choice is that only one motor can be connected to this motor driver. Along with the two motors from the actuating system, we would require 6 motors in total, which require 6 motor drivers. Each motor driver consists of 2 PWM pins for controlling motor speed and 2 output signal pins for controlling motor direction. In total, we would need to connect to 12 PWM pins, at least 24 more digital pins, and an addition of a few more digital pin for the servo and IMU. The Jetson Xavier has 40 GPIO pins in total, but unfortuntely do not provide PWM pins. Hence, we decide to connect the pins to and Arduino Mega board, which has 15 PWM pins and 39 more pins can be used for digital input. Communication between the Jetson and the Arduino can be performed through the PySerial library. Hardware connections to the Jetson board include one intel realsense camera, one Arduino Mega board, and a battery source. The camera and the Arduino all require a usb-c connection, and a Jetson has 4 usb-c ports in total, which is much more than needed. Even though the Jetson requires 9-20V, and the LiPo battery provides 12V which falls into that range, due to the power senstivity of our system, we decide to power the Jetson through a different power source, the Talentcell 12V power bank, which is conveniently supplied by the inventory. The motor shield and the servo require 5V and 7V separately, which is lower than the 12V supplied by the power source. Hence, we added a buck converter. After experiments we realized that the servo would draw more power in comparison to the Arduino, so we used a buck converter to 5V for each of them separately, giving the system the most consistency. The hardware connection design is shown in figure 7.

5.2 Software Control

The entire process is broken in smaller steps that run sequentially in a python program. Since the Jetson runs the computer algorithm programs, whereas the Arduino Mega runs the program to control the motors, a communication bridge needs to be established between the two entities. Since all of us on the team are very familiar with Python, we decide to use Python as the language to send serial data between these models. There is a package called PySerial which is a convenient tool to read data to / from the Arduino.

We experiment and find the relationship between distance input from the sensor to the time needed to run the motors on the Arduino. There were some issues regarding the encoder on the motors, and rather than spending the effort on debugging the encoders, our test results show that running motors based on time measurements are accurate enough. We would apply this linear relationship conversion to the Arduino for it to run the specified amount of time.

Finally, during our experiments we realize that drift on the wheels are inevitable. Once the robot orients itself to the shelf from the basket, it would then move past the basket and towards the shelf. After the robot is in front of the shelf, the tag is off the viewpoint from the camera, so the robot could not utilize the april tag information to re-orient itself again. Hence, we added in IMU to record angle information, such that when the robot is in front of the shelf, the robot can re-orient itself such that it faces the parallel direction again.

5.3 Computer Vision

5.3.1 Navigation:

For navigation we used AprilTags. We chose to use AprilTags because the tags are free to generate and print, and easy to use. The detection software is originally written in Java, but we used a version written in Python for simplicity [11]. The detection software is able to detect all AprilTags that are fully in frame, and outputs the center and corner coordinates, along with the pose matrix. Using the pose matrix we are able to calculate the x,y and z location and the angle of the tag in relation to the camera. The detection software also outputs the tag family and id of the tag, allowing us to differentiate between different shelves and baskets. The detection software mostly worked out of the box, but we had to make a few changes to integrate it seamlessly with the camera stream, and with the navigation code. These changes included configuring the algorithm to work with our camera given its specifications, and extracting the output information. An example of the algorithm's output can be seen figure 8.

5.3.2 Edge Detection:

The edge detection algorithm draws a bounding box around each item on the shelf and outputs the location,

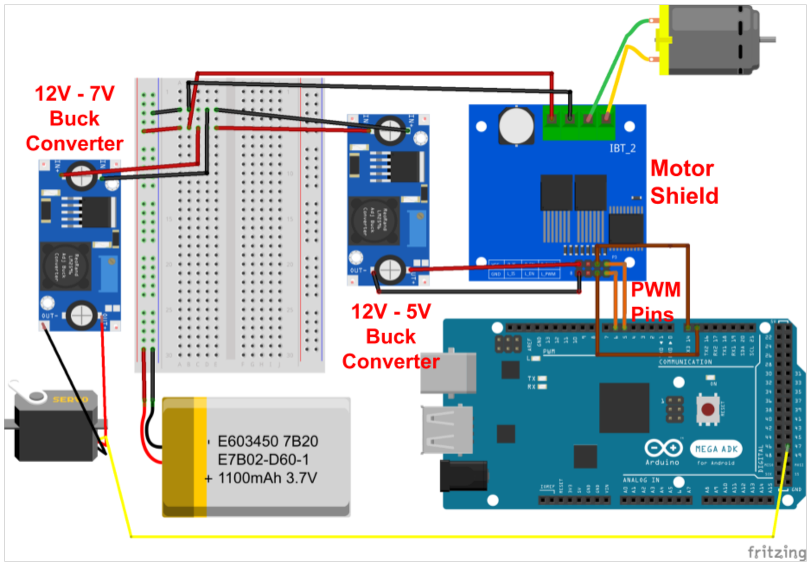
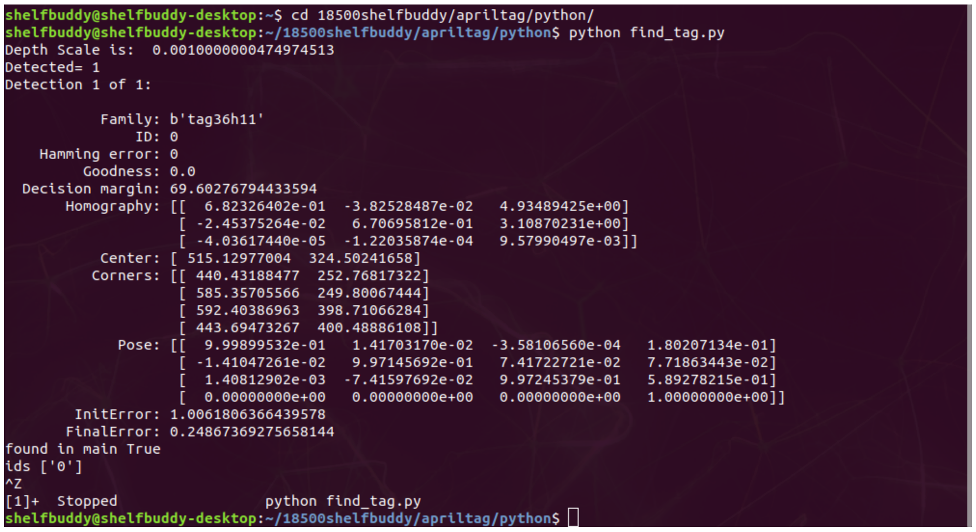


Figure 7: Hardware connections for Motor, Motor Drivers, and Servo



```
shelfbuddy@shelfbuddy-desktop:~$ cd 18500shelfbuddy/apriltag/python/
shelfbuddy@shelfbuddy-desktop:~/18500shelfbuddy/apriltag/python$ python find_tag.py
Depth Scale is:  0.0010000000474974513
Detected= 1
Detection 1 of 1:

           Family: b'tag36h11'
               ID: 0
    Hamming error: 0
         Goodness: 0.0
  Decision margin: 69.60276794433594
       Homography: [[  6.82326402e-01  -3.82528487e-02   4.93489425e+00]
                    [ -2.45375264e-02   6.70695812e-01   3.10870231e+00]
                    [ -4.03617440e-05  -1.22035874e-04   9.57990497e-03]]
           Center: [ 515.12977004  324.50241658]
          Corners: [[ 440.43188477  252.76817322]
                    [ 585.35705566  249.80067444]
                    [ 592.40386963  398.71066284]
                    [ 443.69473267  400.48886108]]
             Pose: [[  9.99899532e-01   1.41703170e-02  -3.58106560e-04   1.80207134e-01]
                    [ -1.41047261e-02   9.97145692e-01   7.41722721e-02   7.71863443e-02]
                    [  1.40812902e-03  -7.41597692e-02   9.97245379e-01   5.89278215e-01]
                    [  0.00000000e+00   0.00000000e+00   0.00000000e+00   1.00000000e+00]]
        InitError: 1.0061806366439578
       FinalError: 0.24867369275658144
found in main True
ids ['0']
^Z
[1]+  Stopped                 python find_tag.py
shelfbuddy@shelfbuddy-desktop:~/18500shelfbuddy/apriltag/python$ 
```

Figure 8: April Tag detection output

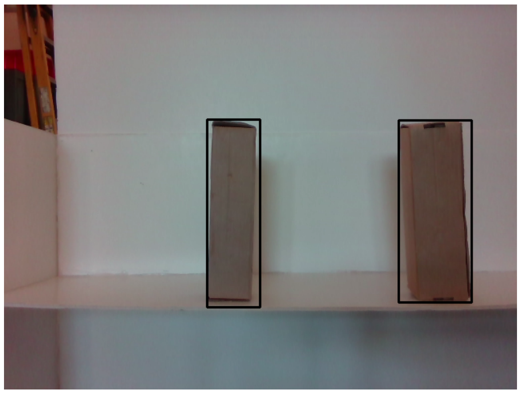


Figure 9: Box detection example output

width, and height of the item so that the robot is able to orient itself to retrieve the item. The steps of the algorithm are as follows:
1: The CV2 Gaussian blur function is applied on the current frame to smooth the image
2: The CV2 Canny edge detection function is applied, which identifies all the edges in the image
3: The CV2 find contour function is applied, using the rectangle option, which joins all points within a boundary.
4: The rectangles found are filtered based on width and height values to remove extraneous boxes
The algorithm outputs the top left and bottom right corners of each box found, and the width and height. An example of the detected boxes can seen in figure 9.

5.3.3 Laser Detection:

The laser detection algorithm detects the presence of the laser point in the frame. The algorithm is implemented in Python [12][13]. In order to account for noise, the robot only searches for the laser point within the boxes on the shelf. The robot first performs object detection, drawing bounding boxes around the items present on the shelf and then searches for the laser point within the bounding boxes. To detect the laser point, the image is first converted from BRG to HSV. Then, two masks are applied. The first mask keeps the brightest 10% of pixels in the image. The second mask filters out all pixels in the image that are not red. These two masks are then combined to preserve the pixels that are both very bright and red. An example of the algorithm's output is seen in figure 10, which has a white dot at the location of the laser and is black everywhere else.

6 TEST & VALIDATION

6.1 Overall Success Rate

The success rate of the entire procedure is 80%. We obtained this number by running the process 20 times, starting

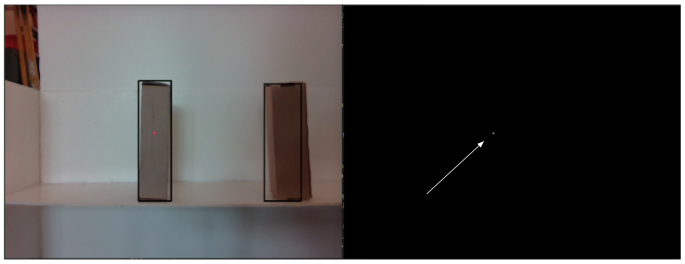
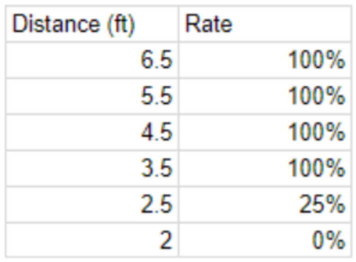


Figure 10: Laser detection output result

from navigation to the user's basket and ending at dropping the item to the basket. Since the entire process depends on the success rate of sub systems, error rates in the subsystems would affect the correctness of the overall run. In addition, failures might occur at points not accounted for in the requirement where we didn't envision in the design phase. For instance, consistent lifting and lowering of the linear slides, navigation to the correct distance range up to the shelf, and dropping the item in the region of the basket. After a long time of fine tuning, the current success rate still does not meet the design requirement of 97.5%. We believe this is due to many hardware inconsistencies including wheel drift and communication delay. However, we believe our current success rate still demonstrates a viable working product.

6.2 Navigation Success Rate

The success rate for traveling to and from the basket is 92.5%. We obtained this number by running the sub procedure 40 times, and 37 of the trials succeeded. Since the robot's navigation is based on detection of the AprilTag and accurately reading off its location information, the bottleneck is to ensure the AprilTag can be read when the robot is searching for its position, which depends on the position of the robot from the AprilTag. Given the far distance a Realsense camera is able to detect, the robot is easily able to see the tag from over 10 feet away. However, such a far distance does not apply to the use case, since the robot is likely to be closer to the user. Hence, we determine the closest distance which the robot can find the AprilTag instead. We found this value by incrementally moving the robot closer to the basket. We defined the success case as whether the robot can correctly find the AprilTag while rotating. The testing data and plot are seen below in figures 11 and 12.



Distance (ft)	Rate
6.5	100%
5.5	100%
4.5	100%
3.5	100%
2.5	25%
2	0%

Figure 11: Navigation Testing Data

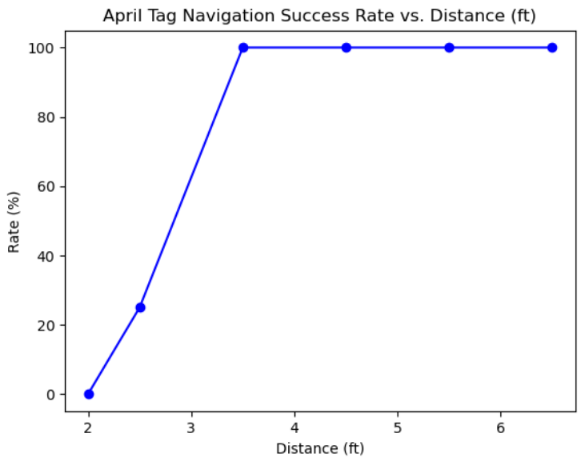
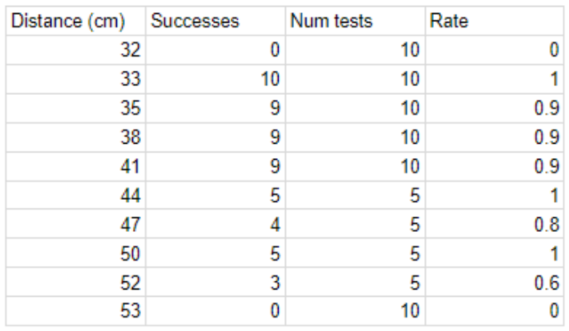


Figure 12: Navigation Testing Plot

The data shows that locating the AprilTag during rotation is relatively consistent from further distances, up until the robot is 2.5 feet away, where the success rate then drops significantly. This is due to the AprilTag not being fully in frame, as the camera is too close to capture the entire tag. As the robot moves even closer, the success rate drops to 0. Hence, for the navigation process to be successful, we have to position the robot at least 3 feet away from the user's basket.

6.3 Laser Detection Success Rate

To determine the optimal distance in which the camera can detect the laser, we ran a few tests at various distances and calculated the success rate at these distances separately. The corresponding testing data and plot are seen in figure 13 and 14.



Distance (cm)	Successes	Num tests	Rate
32	0	10	0
33	10	10	1
35	9	10	0.9
38	9	10	0.9
41	9	10	0.9
44	5	5	1
47	4	5	0.8
50	5	5	1
52	3	5	0.6
53	0	10	0

Figure 13: Laser Testing Data

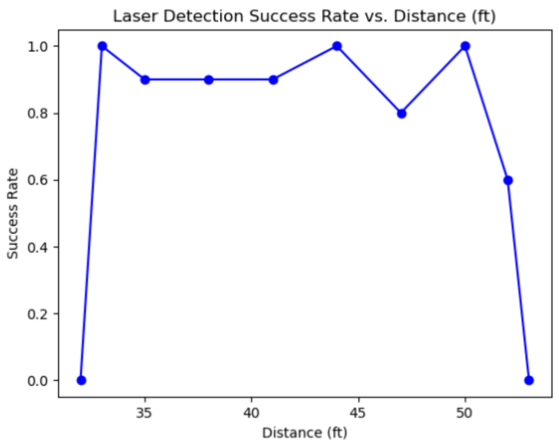
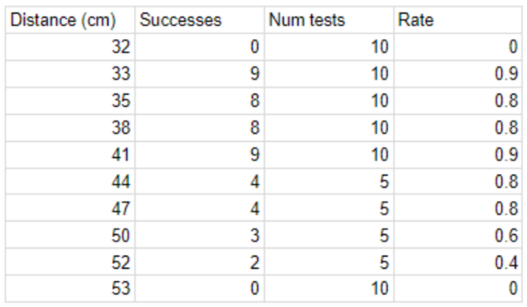


Figure 14: Navigation Testing Plot

From data we see that laser detection is accurate in the range of 33 cm to 50 cm from the shelf. At 52 cm, accuracy drops significantly, and the success rate drops to 0 at farther distances. This is due to hardware constraints in the camera not being able to detect the tiny laser point from a far distance, as well as the fine tuning implemented in the code to filter out objects of specific sizes. Within the optimal range, overall the success rate is 92.73%, which is calculated from 51 success trials out of 55 in total.

6.4 Grabbing Success Rate

Similar to laser detection testing, to determine the optimal distance in which the robot can grab the object, we ran a few tests at various distances and calculated the success rate at these distances. We define grabbing as the robot navigating and centering itself in front of the object and moving forward to grab the object. The corresponding testing data and plot are seen in figure 15 and 16.



Distance (cm)	Successes	Num tests	Rate
32	0	10	0
33	9	10	0.9
35	8	10	0.8
38	8	10	0.8
41	9	10	0.9
44	4	5	0.8
47	4	5	0.8
50	3	5	0.6
52	2	5	0.4
53	0	10	0

Figure 15: Laser Testing Data

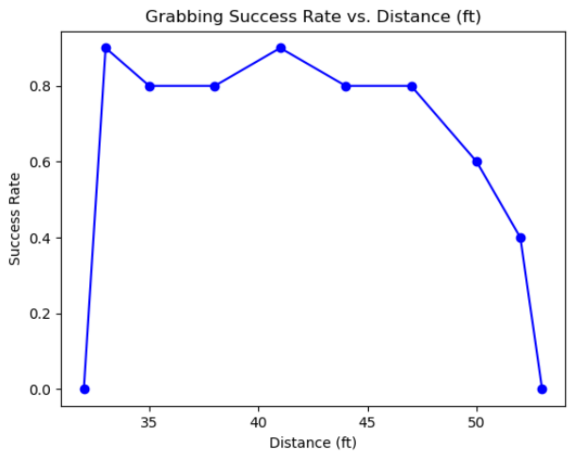


Figure 16: Navigation Testing Plot

From data we see that grabbing is most accurate when the robot starts from a distance of 33 cm to 47 cm from the shelf. After 47 cm, we see a sharp decreasing trend of grabbing accuracy. This is due to drift in wheels resulting in imprecise stopping in front of the object to grab it. Within the optimal range, overall grabbing success rate is 84%, which is 42 successful trials out of 50.

In conclusion, we should aim to have the robot stop at 33 cm to 47 cm from the shelf for grabbing and laser detection to work optimally.

6.5 Other Requirements

The average speed of our current implementation is 0.33 m/s. We performed the test by having the robot drive a measured distance, record the time taken, and calculate the speed. We didn't achieve the designed specification of travelling at 0.5 m/s. Since the motors are not running under the maximum rotational speed, our group in theory could increase that number in the Arduino code. However, after trials we found out that this might lead to other inconsistencies such as the the wheels "overshoot" a bit by traveling half a second more than it should have, which would lead to inaccurate results for the further procedures.

The current grabbing latency is currently 6.8 seconds, which is roughly 2.6 times slower than our design requirement. We had to increase the time for closing the claw to avoid accidentally tipping the object over.

The current laser point latency is 0.561 seconds. We define laser point latency as the time from the start of the laser detection computer algorithm, to identifying the laser point and its position. To test this we ran the algorithm and printed out timing information on the console, and took the average over a few runs. This meets our design specification of being under 1 second.

The current distance between items on shelf is 4 inches. This is greater than our design specification. This is due to our claw having a wide range when opening, and if the objects are closer together, there's a much higher chance of the claw knocking off one of the neighboring objects. 4 inches is the sweet spot we determined such that item retrieval can achieve sufficient accuracy, while minimizing distance between objects.

The items that we are using for test are 2 inches wide and weight 1 lb. This roughly meets our design specifications.

7 PROJECT MANAGEMENT

7.1 Schedule

See figure 19 on page 19 for the schedule.

7.2 Team Member Responsibilities

Ludi Cao was primarily responsible for building the drivetrain and implementing its navigation to and from the shelf.

Esther Jang was primarily responsible for building and programming the linear slide system and setting up the claw gripper.

Bhumika Kapur was primarily responsible for the computer vision aspect of the project.

All members worked together for integration and fine tuning parameters to achieve the most accurate result. The project schedule is seen in figure 19.

7.3 Budget

See page 16 for the bill of materials for our various subsystems. Note that we added the order of one switch battery, one 5.5x2.5mm male to DC 5.5x2.1mm female adapter, and one IMU after the design review. We also borrowed the power bank from the inventory, as well as replaced the usage of two Arduino Uno boards with one

Arduino Mega boards. The Arduino boards are all already previously owned.

7.4 Risk Management

In terms of the potential design risks, we have the concern that laser detection would not work as accurately as desired. This is due to the laser point being small, the presence of noise, and the distance from the camera to the laser needs to be within a certain range. The initial performance was indeed very poor. In order to mitigate the noise interference, we built our own shelf using white stryofoam board, so that no other red light would be in the frame. In addition, we ran edge detection to identify the boxes first, so that objects in the background are not mistakenly identified as the laser point. We also had imprecise results when running edge detection algorithms to find boxes, as miscellaneous edges in the camera frame would disturb what the algorithm would think as boxes. To mitigate the issue, our team did a lot of experiments to filter out the range of height width of boxes the camera should see in frame.

Another design risk that came up during testing, is that the wheels tend to drift when travelling through the low friction floor in the lab. The drift could also be caused by the unbalanced weight on the robot. In order to resolve this issue, we added an IMU sensor. This allows the robot to rotate to the angle it recorded previously to compensate the drift it encountered when driving.

Finally, during the design phase we were concerned the claw would not grab the object firmly enough to grasp it as it travels to the basket. We found that wrapping the claw with a material of high coefficient of friction mitigates the issue. We ended up wrapping rubber bands around edge of the claws, which increased it's gripping capability significantly.

In terms of schedule, we made sure to budget enough time for integration of our subsystems as there is a big risk of our entire system failing if integration does not succeed. Initially in our design review, we set aside 3 weeks in our time schedule and also had Esther spend a week starting on integration to allocate even more time to this task. In reality, we started integration from the week Nov 8 - Nov 13, which is three weeks before the final presentation, and five weeks before the final demo, which is on our proposed schedule plan.

Our project faced many budget constraints. The robot required many pieces of hardware that when added together, was almost over budget. The motors themselves cost nearly 200 dollars in total, and a lot of aluminum was required for the chassis since our robot has a wide frame and a great height when the linear slides are expanded. To mitigate these risks, we tried to borrow from the inventory or clubs so we did not exceed our budget. Thanks to Tao Jin, we were able to borrow an Intel Realsense camera. We also acquired the Jetson Xavier and the power bank from the inventory. To save up the cost of purchasing a shelf to test, we built a simple version of our own by using the stryofoam pieces in the lab. We also borrowed miscellaneous items from Roboclub, such as jumpwires, nuts and bolts, and the servo for the claw. In addition, we did careful calculations about the exact amount of pieces we needed for the robot when placing the order from RevRobotics, so we could minimize our shipping fees. Even though this put us about half a week behind in building the robot, eventually this process turned out successful as we used up all the pieces we ordered with barely any extra material left. We still exceeded the allocated budget by around 100 dollars, but we were able to minimize excess spending.

8 ETHICAL ISSUES

Our intended audience is people who have disabilities and are unable to grab items on high shelves, hindering them from grocery shopping. Hence, one main ethical concern is that our robot should not harm the vulnerable users under any circumstances. Since our robot includes integration of various hardware components, the failure of any one of them might lead to dangerous outcomes. For instance, if the sensor distance malfunctions and reads items farther than they actually appear, then the robot might accidentally run into the user, causing injuries. Other possible edge cases of harm include the linear slides breaking and falling down on the user when up at a certain height. Or a wire wearing down, which results in an electrical hazard due to potential short circuiting. The user would be adversely impacted in these situations, and we strive for zero tolerance on any edge cases that would risk the user's safety. However, it is impossible to guarantee all components would work seamlessly, and it is hard to predict what might go wrong in the system. Hence, the best way to mitigate these risks is to implement an "emergency-stop" system, such that once the user detects a problem and senses danger, the user can press a button and the robot would immediately stop all programs. We hope that the probability of both the robot and the emergency halt system not working at the same time is minimal, and the user can feel safe around the robot. Another safety procedure is to conduct an inspection on the robot every few months.

One other ethical issue relates to the misuse of computer vision technologies, where some users might feel uncomfortable about their privacy potentially being infiltrated. Edge cases happen when a malicious party gains illegal access to the computer vision information, and use this negatively against the the person captured. The most important solution is to ensure the robot's network is secure and safe from third parties breaking in. The computer vision should also only be turned on when the robot is in operation, to reduce the amount of time that individuals may be caught on camera.

The last ethical issue is the delegation of responsibility when the robot unintentionally cause damages (such as breaking an item in the store). The manufacturer of the robot, the store, and the user may all be asked to account for this damage, and it may be unclear who is responsible. Mitigation approaches include only allowing the robot to operate on items that are not fragile (avoid glass) and to avoid dangerous items (anything that is sharp or heavy). A legal statement about the delegation of responsibilities for all parties involved should be written and agreed upon by all. Once a mutual agreement is established, it should be clear who bears responsibility in different situations.

9 RELATED WORK

IAM Robotics Swift: An autonomous manipulation robot that can autonomously pick up items specified in a fulfillment order from shelves. A shelf can be up to 7 feet tall, and the robot can scan through an aisle of shelves. The robot uses a vacuum gripper end effector that can hold up to 15 lbs. It also uses 3D mapping-based computer vision for its autonomy control [14].

Amazon Picking Challenge: The company Amazon used to host a robotics competition which involved robots being able to retrieve objects off a shelf. The winning robot of 2017's competition used an end effector that was a combination of a suction gripper, claw, and sliding mechanism. It also trained itself on models of objects to guide its computer vision detection [15].

10 SUMMARY

Our project meets the functionality requirements set for our use case. The robot first locates and drives upwards towards the user's basket. It then orients itself towards the shelf, approaches it, then lifts the linear slides. The robot detects the object being pointed at by the laser tag through edge detection filtering. The robot retrieves the object and return back to the user. To meet the functionality requirements, we had to modify many of our accuracy requirements, due to the many hardware inconsistencies not addressed in the design review, including sensitive power requirements, and communication lag between the Arduino Mega and the Jetson. In hindsight, we were overly confident about our power analysis during the design review, and didn't take into account that the robot can only work at peak consistency if the battery is charged at full power. This requires a power source of much higher capacity for us to continuously test for some amount of time, so we should have chosen a battery that would better satisfy the requirements. In addition, we realized we could've potentially eliminated using an Arduino Mega board by using software pwm pins instead of hardware pwm pins, and the 40 GPIO pins on the Jetson would then be sufficient. This would result in a much faster run time.

10.1 Lessons Learned

One lesson our team learned is to conduct thorough research about various approaches during the design review stage, and be confident that the proposed solution meets the requirements and is reasonable for the team to implement. For instance, our team initially used two Arduino uno boards, since the motors require 6 pwm pins, but one Arduino Mega board turned out to be better. We also should have looked into software pwm pins, eliminating the need for an Arduino altogether. The other important takeaway is that hardware projects can have much more inconsistencies, especially compared to software projects. Hence, a sufficient amount of time should be budgeted for integration, as many tests need to be run to debug hidden errors and find tune parameters.

11 Bibliography

1. Reuter, Dominick. "Meet the Typical CVS Shopper: A White Gen X College-Educated City Dweller Earning a High Income." Business Insider, Business Insider, 9 Sept. 2021, https://www.businessinsider.com/typical-cvs-shopper-demographic-urban-genx-earning-high-income-2021-9.

2. Tolerico ML;Ding D;Cooper RA;Spaeth DM;Fitzgerald SG;Cooper R;Kelleher A;Boninger ML; "Assessing Mobility Characteristics and Activity Levels of Manual Wheelchair Users." Journal of Rehabilitation Research and Development, U.S. National Library of Medicine, https://pubmed.ncbi.nlm.nih.gov/18247253.

3. Experiment: How Fast Your Brain Reacts to Stimuli, https://backyardbrains.com/experiments/reactiontime.

4. "Apriltags." Robotics Knowledgebase, 30 Aug. 2018, https://roboticsknowledgebase.com/wiki/sensing/apriltags/.

5. "How Long to Charge a 3s 11.1V 5000mah Lipo?" RC Groups RSS, https://www.rcgroups.com/forums/showthread.php?2798973-How-long-to-charge-a-3s-11-1v-5000mah-Lipo.

6. "Friction - Friction Coefficients and Calculator." Engineering ToolBox, https://www.engineeringtoolbox-.com/friction-coefficients-d_778.html.

7. Zwetsloot, Rob, and Rob is amazing. He's also the Features Editor of The MagPi. "Raspberry Pi 4 Specs and Benchmarks." The MagPi Magazine, https://magpi.raspberrypi.com/articles/raspberry-pi-4-specs-benchmarks.

8. "Jetson Modules." NVIDIA Developer, 3 Dec. 2021, https://developer.nvidia.com/embedded/Jetson-modules.

9. "Arducam Mini High Quality Camera with M12 Mount Lens, 12.3MP 1/2.3 Inch IMX477 HQ Camera Module for Jetson Nano Xavier NX." Arducam, 31 Oct. 2021, https://www.arducam.com/product/arducam-high-quality-camera-for-jetson-nano-and-xavier-nx-12mp-m12-mount/.

10. "Depth Camera D435." Intel® RealSense™ Depth and Tracking Cameras, 17 June 2021, https://www.intelrealsense.com/depth-camera-d435/.

11. Swatbotics. "Apriltag/Apriltag.py at Master · Swatbotics/Apriltag." GitHub, https://github.com/

12. "Tracking a Laser Pointer with Python and Opencv." Brads Blog RSS, https://bradmontgomery.net/blog/tracking-a-laser-pointer-with-python-and-opencv/.

13. Chávez, F., et al. "Automatic Laser Pointer Detection Algorithm for Environment Control Device Systems Based on Template Matching and Genetic Tuning of Fuzzy Rule-Based Systems." International Journal of Computational Intelligence Systems, vol. 5, no. 2, 2012, p. 368., https://doi.org/10.1080/18756891.2012.685327.

14. "Swift Robot: Autonomous Material Picking Robot with Obstacle Detection." IAM Robotics, 10 May 2021, https://www.iamrobotics.com/products/swift/.

15. Leitner, Jürgen. "Picking the Right Robotics Challenge." Nature Machine Intelligence, vol. 1, no. 3, 2019, pp. 162–162., https://doi.org/10.1038/s42256-019-0031-6.

Table 1: Bill of Materials for Chassis

Description	Model #	Manufacturer	Quantity	Cost @	Total
HD Hex Motor 40:1 Spur Gearbox	REV-41-1301	Rev Robotics	4	\$30.00	\$120.00
15mm Extrusion - 1m - 90° Ends	REV-41-1017	Rev Robotics	2	\$12.00	\$24.00
90mm Omni Wheel - 2 Pack	REV-41-1190	Rev Robotics	4	\$25.00	\$100.00
Slim Shaft Collar - 10 Pack	REV-41-1629	Rev Robotics	1	\$8.00	\$8.00
15mm Plastic 135 Degree Bracket - 8 Pack	REV-41-1310	Rev Robotics	2	\$5.00	\$10.00
15mm Metal Flat HD Hex Motor Bracket V2 - 4 Pack	REV-41-1486	Rev Robotics	1	\$5.00	\$5.00
15mm Metal Bent HD Hex Motor Bracket V2 - 4 Pack	REV-41-1487	Rev Robotics	1	\$5.00	\$5.00
M3 Nut - 100 Pack	REV-41-1126	Rev Robotics	1	\$5.00	\$5.00
M3 x 16mm Hex Cap Screws - 100 Pack - 4 Pack	REV-41-1360	Rev Robotics	1	\$11.00	\$11.00
DC Stepper Motor Driver	BTS7960	DORHEA	1	\$27.55	\$27.55
Ovonic 11.1V 5000mAh 3S 50C-100C LiPo Battery	Lipo Battery	Ovonic	1	\$27.99	\$27.99
HTRC LiPo Battery Charger	B3AC Pro	HTRC	1	\$22.24	\$22.24
9-DOF Absolute Orientation IMU Fusion Breakout	BNO055	Adafruit	1	\$39.9	\$39.9
Switch Battery On/Off Connector Lead	B07NQGV78D	Dilwe	1	\$10.89	\$10.89
5.5x2.5mm Male to DC 5.5x2.1mm Female Adapter	B07L5GGW7Q	ZdyCGTime	1	\$5.99	\$5.99
12V power bank model	YB1206000-USB	Talentcell	1	\$26.09	Borrowed
					\$411.62

Table 2: Bill of Materials for Linear Slides and Claw

Description	Model #	Manufacturer	Quantity	Cost @	Total
HD Hex Motor 40:1 Spur Gearbox	REV-41-1301	Rev Robotics	2	\$30.00	\$60.00
15mm Metal Flat HD Hex Motor Bracket V2 - 4 Pack	REV-41-1486	Rev Robotics	1	\$5.00	\$5.00
15mm Metal Bent HD Hex Motor Bracket V2 - 4 Pack	REV-41-1487	Rev Robotics	1	\$5.00	\$5.00
15mm Linear Motion Kit V2	REV-45-1507	Rev Robotics	3	\$12.00	\$36.00
5.5mm Nut Driver	REV-41-1119	Rev Robotics	1	\$6.50	\$6.50
2mm Allen Wrench	REV-41-1377	Rev Robotics	1	\$1.00	\$1.00
15mm Extrusion - 1m - 90° Ends	REV-41-1017	Rev Robotics	2	\$12.00	\$24.00
Small Pulley Bearings - 10 Pack	REV-41-1368	Rev Robotics	1	\$11.00	\$11.00
1.2mm UHMWPE Cord - 10M	REV-41-1162	Rev Robotics	1	\$6.00	\$6.00
15mm Metal HD Inside Corner Bracket - 4 Pack	REV-41-1688	Rev Robotics	1	\$10.00	\$10.00
15mm Plastic Lap Corner Bracket - 8 Pack	REV-41-1321	Rev Robotics	1	\$5.00	\$5.00
60mm Pulley - 4 Pack	REV-41-1345	Rev Robotics	1	\$8.00	\$8.00
Mechanical Claw BigClaw Robot Gripper w/ 335MG Servo	B089KMK954	LewanSoul	1	\$33.00	\$33.00
LewanSoul Mechanical Claw BigClaw Robot Gripper	B08Q7XZVR4	LewanSoul	1	\$23.00	\$23.00
Adafruit 16-Channel 12-bit PWM/Servo Driver	PCA9685	Adafruit	1	\$15.00	\$15.00
DC Stepper Motor Driver	BTS7960	Teyleten Robot	1	\$10.50	\$10.50
					\$259

Table 3: Bill of Materials for Miscellaneous Items

Description	Model #	Manufacturer	Quantity	Cost @	Total
SanDisk 32GB MicroSDHC Memory Card	SDSDQ-032G-A11M	SanDisk	1	\$7.50	\$7.50
Shipping From Vendor			1	\$14.50	\$14.50
Intel RealSense	D435	Intel	1	\$299.00	Borrowed
Jetson Xavier NX		Nvidia	1	\$399.00	Borrowed
Arduino Mega	A000067	Arduino	1	\$38.05	Already Owned
					\$25

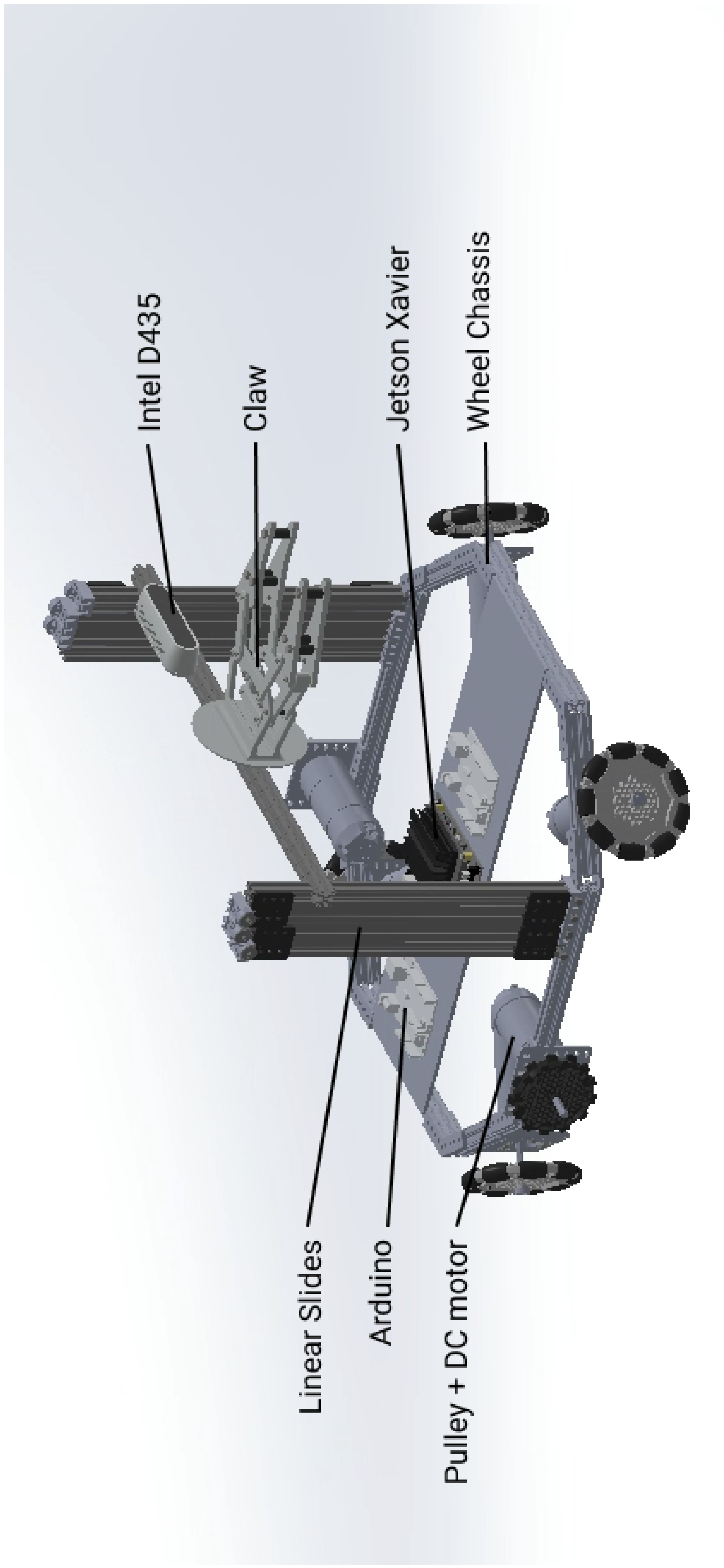


Figure 17: Full CAD Design of System

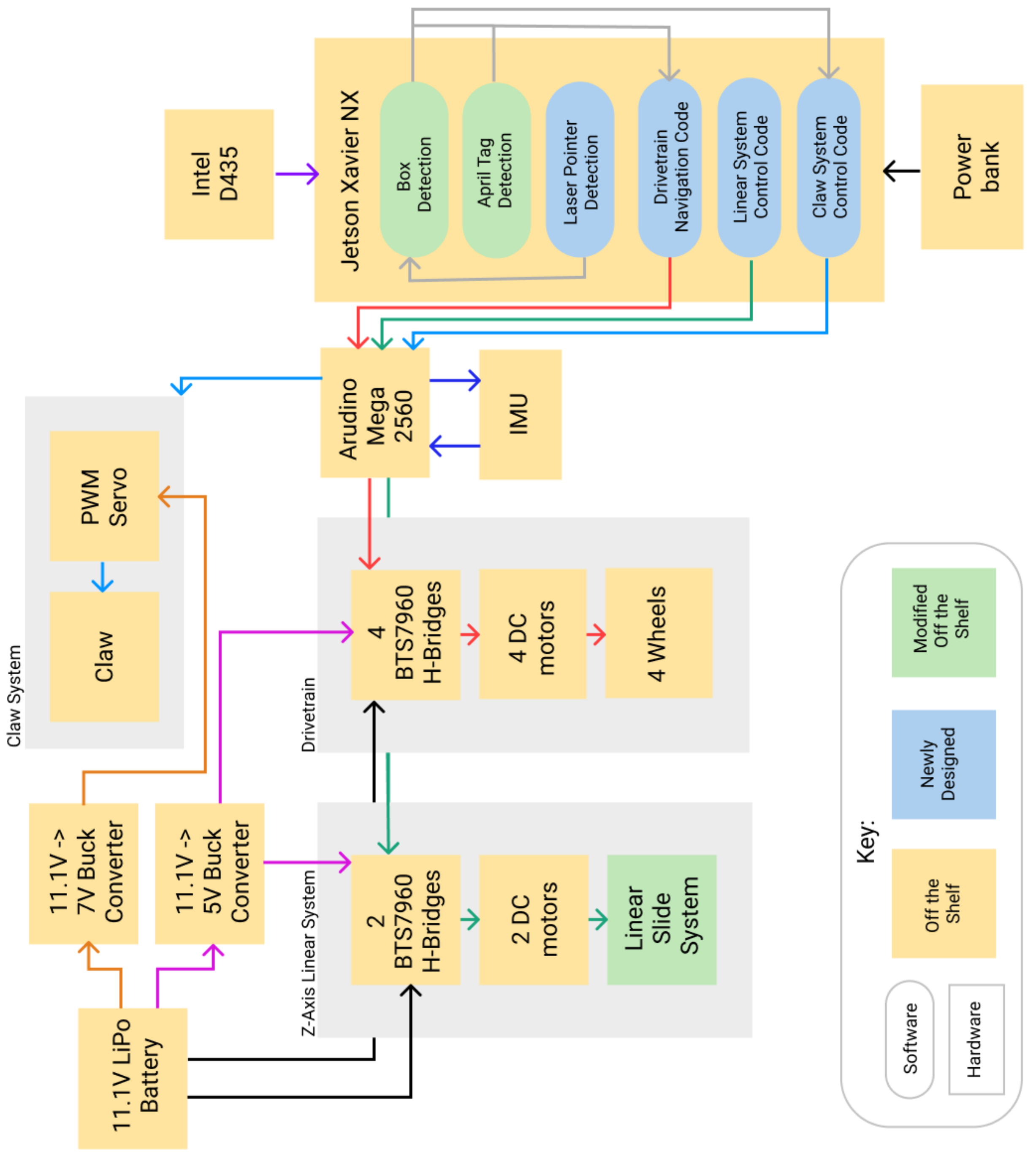


Figure 18: Block Diagram of System

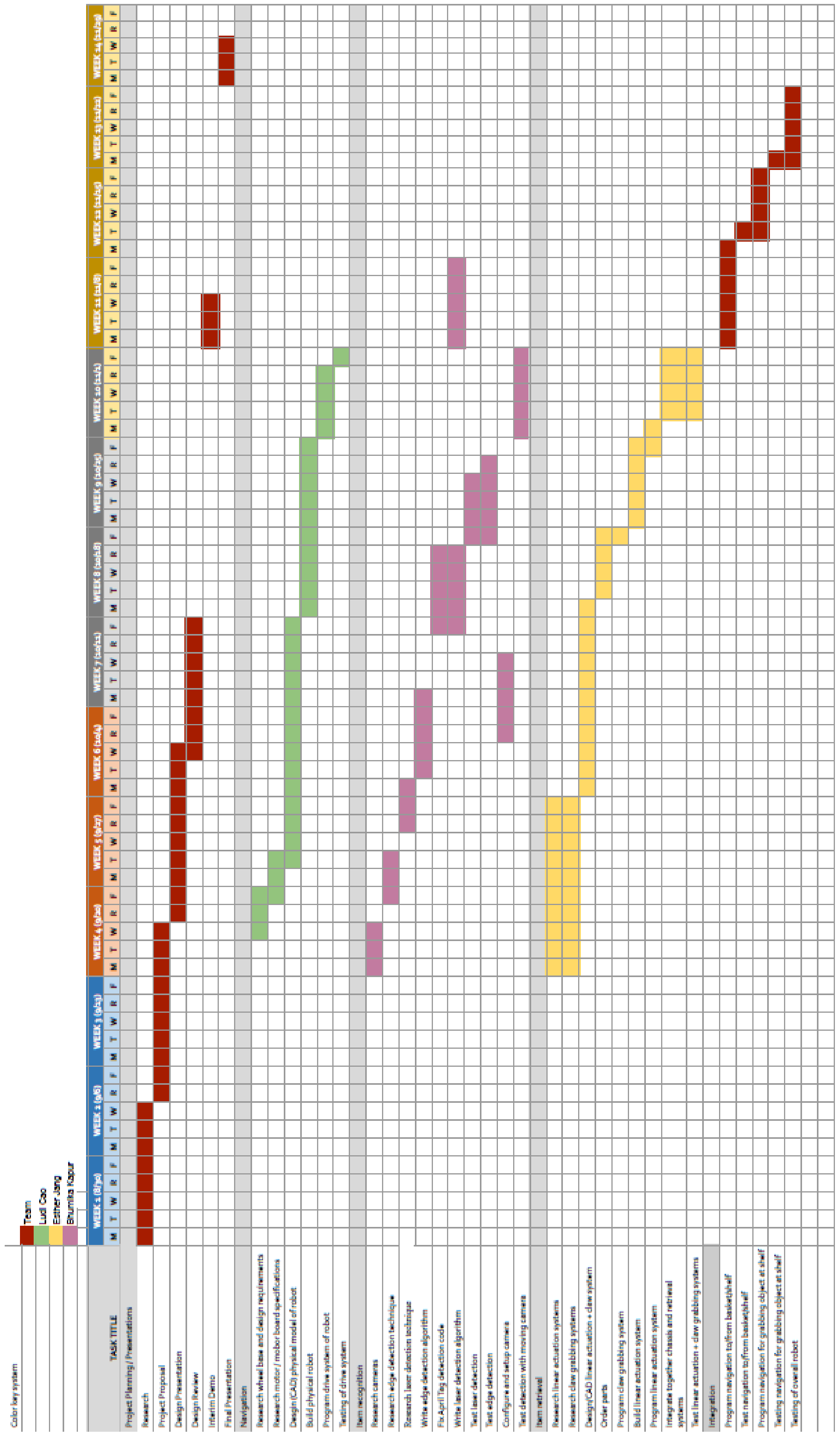


Figure 19: Gantt Chart