Autonomous Package Delivery Robot

College of EECS - Capstone Project
Fall 2021 - Spring 2022

Drew Gehrke
Nicholas McBee
Andrew Pehrson
Nathan Searles
Tyrone Stagner

Oregon State University

Contents

1 Overview

This section will highlight the foundation of our project's goals and our team dynamic for collaboration. It will discuss the summary of the Autonomous Package Delivery Robot, set standards for organizing our collaborative process, and postulate the specific area of technology that this project will influence. Another goal is to establish a general and specific timeline of completion deadlines for the course of this year.

1.1 Executive Summary

The purpose of this project is to create a robotic package delivery system operating in the context of an environment with well-developed pedestrian-tailored infrastructure, such as a college campus. The Autonomous Package Delivery Robot (APDR) will be capable of carrying packages while autonomously navigating along sidewalks and avoiding obstacles to reach its final destination. The scope of this project also contains a user interface in the form of a website that will allow individuals to initiate and receive deliveries at a specified destination.

The goal of this project is to join the increasing number of autonomous delivery robots that provide contactless deliveries of food and goods to customers. This project will also introduce a solution to the rising issue of electronic waste, which will be achieved by using recycled electronics such as the base, motors and batteries of an electric wheelchair.

This project was inherited from a previous Oregon State University EECS Capstone group (2020-2021). This team will be working with Hanna Anderson, project sponsor, and previous team member on this project. In its current state, the robot is capable of movement under manual control, avoidance of stationary obstacles, and waypoint creation using GPS. The technical goals for the team inheriting this project are developing a secure package delivery system and increasing the capability for autonomous outdoor travel of the APDR. The developed product will be incredibly aware of stationary objects and dynamically moving pedestrians and vehicles, as well as provide an intuitive and reliable courier service to distributors and customers alike.

Many changes and improvements have been made to the system by the new team. The APDR system now has a way to store packages in its lockbox mounted right on top of the electric wheelchair base. Other hardware changes include the addition of a team made PCB and new circuitry to properly distribute power to all the various electronic components of the system. Another key change is that many of the sensor modules have been off-boarded from the Raspberry Pi and are now processed on an ESP32 microcontroller. This allows for more processing speed on the Raspberry Pi. The inertial measurement unit, or IMU, and global positioning system, or GPS, aid in the navigation of the robot by sending the data first to the ESP32 to be processed, which the Raspberry Pi then receives and sends to the various topics which require the data. Speaking of topics, the entire system has been migrated from the first version of the Robot Operating System, or ROS, into the newer version, ROS2 Galactic. Many custom topics have been developed to get the APDR system working, including a MCU (microcontroller unit) driver, USB-to-Serial driver, motor controller driver, and many more. Several other topics have been utilized to aid in traversal, such as the navigation stack built into ROS2 and the robot translocation topic.

1.2 Team Protocols and Standard

Table 1: Team Protocols and Standards table.

Topic	Protocol	Standard
On-time Deliverables and Team Collaboration	Drafts of all individuals' contributions to teamwork artifacts / submissions should be fully complete by setting a pre-deadline (such as the time/date of a team meeting to review the final submission) so that the team can look it over and make final revisions together.	Work judged as complete will include all necessary content and formatting requirements listed in Canvas and will be nearly error-free.
Task Management	Team will use Trello for task assignment and record of completion.	During team meetings, the team will review tasks to be completed and assign out cards that represent these tasks in Trello. When a task is complete, individuals responsible will move it to the "completed" stack.
Communication	Group messaging and weekly meetings will be hosted on the Discord server for this project.	Team members will be expected to give notice as soon as possible if they will not be attending a meeting. Discourse should be had with a professional mindset. Jokes are encouraged to aid in building team camaraderie, but must be respectful and not at the expense of others.
Logging of Time Spent	Team will use Trello to keep track of time spent on specific tasks.	Logging of time spent should happen on a task basis. Before submitting a ticket as completed, team members should make a comment on the task highlighting the scope of the work completed, problems encountered, and give an estimate of total time invested in the task.
Interpersonal Conflict	If a situation arises where there is a team conflict, this hierarchy of actions should be followed.	Address the concern directly with the individual (1 on 1). If that is not possible, address the concern with another team member to get a second opinion. If the member is not comfortable working it out within the team, they will reach out to one of the Course Staff to resolve the issue.

Topic	Protocol	Standard
Documentation Standard	Finalized documentation should be compiled in LaTeX using Overleaf. Intermediate documentation should be written in a Google or Microsoft Document.	Before closing out a Trello ticket, work done should be properly documented. Minimum of 1 sentence per 30 minutes spent working. Team members should use best judgement to make sure adequate information is written.
Coding Standard	Code will be properly commented and actively synchronized with GitHub.	Coding comments: expectation is that comments should be thorough enough for the team to be able to follow along at a medium-high level of what is happening within the scripts. Individual files should contain a header with the purpose clearly described, this should be a paragraph. All functions should also contain a sentence or two description and logical processes within the function should be briefly described within reason. File names should match class names, each class should have it's own file. Naming conventions: Pieces of code should follow the below conventions.
Expenses And Purchases	Expenses will be tracked in this spreadsheet.	All project expenses should be approved by the project sponsor as well as all team members prior to purchase. Once approved, expenses will be tracked in the linked spreadsheet. All expenses exceeding the project budget will be asked to be financed by the project sponsor. If financing is not provided, they will be evenly split amongst team members.
Hardware Design Standard	Hardware designs and all 3D modeling for this project will be created and stored within a shared project inside of Fusion360. All PCB design and schematics will be created with Altium.	To maintain collaborative transparency, Fusion360 should be used, because of the cloud storage capabilities. We will use Fusion360 for 3D Modeling. Altium will allow us to collaboratively work on PCB design and schematics. All modules should have proper schematics starting with a block diagram and working up to specific component schematics (Low Level $->$ High Level).

Topic	Protocol	Standard
Team Meeting Standard	Team meetings will be conducted twice a week either in-person (as needed) or via Discord. Meeting agendas will be developed through a meeting notes document on the shared Google Drive	All team members are expected to attend each meeting during the week unless they have communicated that they will be absent. The first meeting of the week is expected to be a longer meeting for collaboration and planning out the week. The second meeting will be a check-in on progress done for the week.
Project Partner Communication	The project partner will have official updates via email. Questions, concerns, and meeting arrangements will be made via Discord.	The project partner is a member of our Discord server and encourages us to reach out with questions and concerns. This will be the primary way for getting immediate feedback from the project partner. For official updates on the project, an email will be drafted, checked over by the whole team, and sent.

Table 2: Team Contact Information

Contact Information	
Name	Email
Andrew Pehrson	pehrsona@oregonstate.edu
Nathan Searles	searlesn@oregonstate.edu
Nicholas McBee	mcbeen@oregonstate.edu
Drew Gehrke	gehrkean@oregonstate.edu
Tyrone Stagner	stagnert@oregonstate.edu

1.2.1 Communication Analysis

Table 1.2 shows all of the protocols in which the team will convey any sort of communication. Deliverables which require a group submission must be approved by all members of the team prior to being submitted to Canvas. These approvals will be sought out using the team's primary communication form of Discord. Task management, time logging, and meeting agenda's will be documented and tracked through the team's Trello Board. Final documentation any team member produces much be compiled using LaTeX in Overleaf. Once finalized and compiled, the file can be downloaded in the necessary format and upload to the team's shared Google Drive. In this Drive, other documents tracking expenses and meeting notes will be stored.

1.3 Gap Analysis

The global autonomous delivery robots market is currently almost 25 million dollars, and is set to grow to 237 million dollars by 2027 [1]. Currently, autonomous delivery robots mainly deliver food and packages. Although they claim to be autonomous, many companies still require human monitoring to track their movements. The team plans to enhance and expand the opportunities already available within the autonomous delivery robots market in two ways: incorporating recycled technology and striving for full autonomy. In order to incorporate recycled technology, the team will be using technologies which have supposedly reached their end-of-life cycle. These include wheelchair frames, batteries, motors, and wheels. With the huge amount of waste accumulated already, the team hopes to mitigate it as much as possible. When it comes to full autonomy, this project aims to develop a robot able to operate without monitoring, including when crossing intersections and avoiding obstacles. To help with this existing technology such as Tesla's Autopilot [2] and Ford's Enhanced Park Assist [3] will be used in reference.

At this time, the robot is intended to assist with "last mile" delivery services, but can always be expanded on in the future. The on-campus mailing service at Oregon State could use this system to automate the process of delivering packages to various buildings on campus. With the amount of obstacles, such as people, prevalent in-between buildings this would be a good goal to reach towards in our project timeline. Potential future customers of this project include mailing services such as Amazon and USPS. This technology could specifically be useful in large cities with congestion where pedestrian safety may be a concern.

1.4 Timeline

The infographic below is the project overview summary. It closely follows the course timeline and summarizes what the project plan will look like over the course of the year.

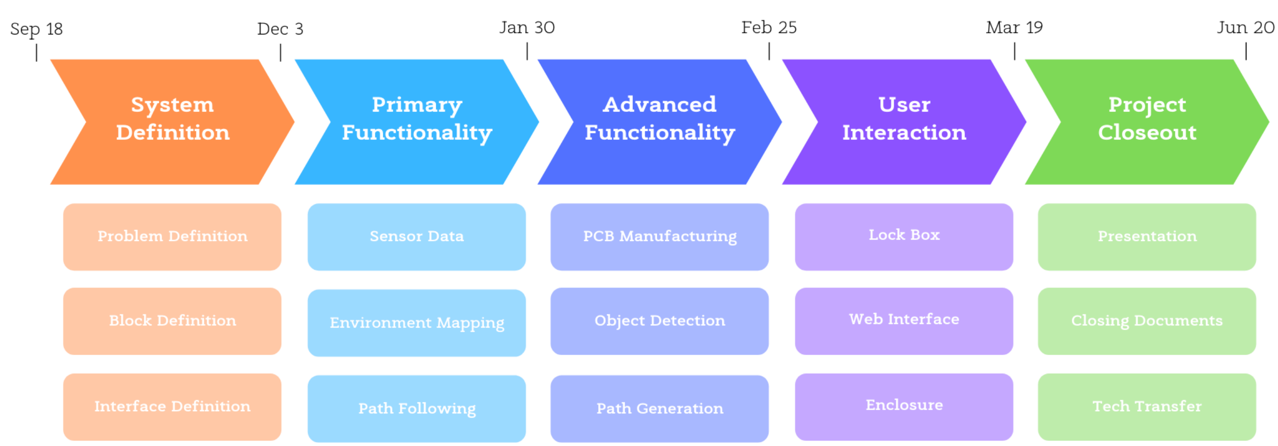

Figure 1: External Project Timeline.

The Gantt charts below are split by term and outline the specific tasks which will be completed by different members and what dependencies these tasks entail.

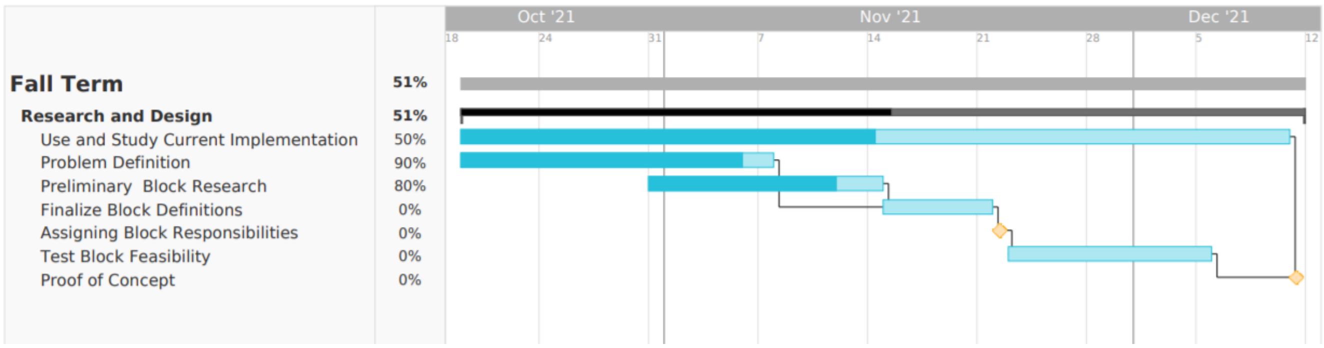

Figure 2: Gantt Chart timeline for Fall.

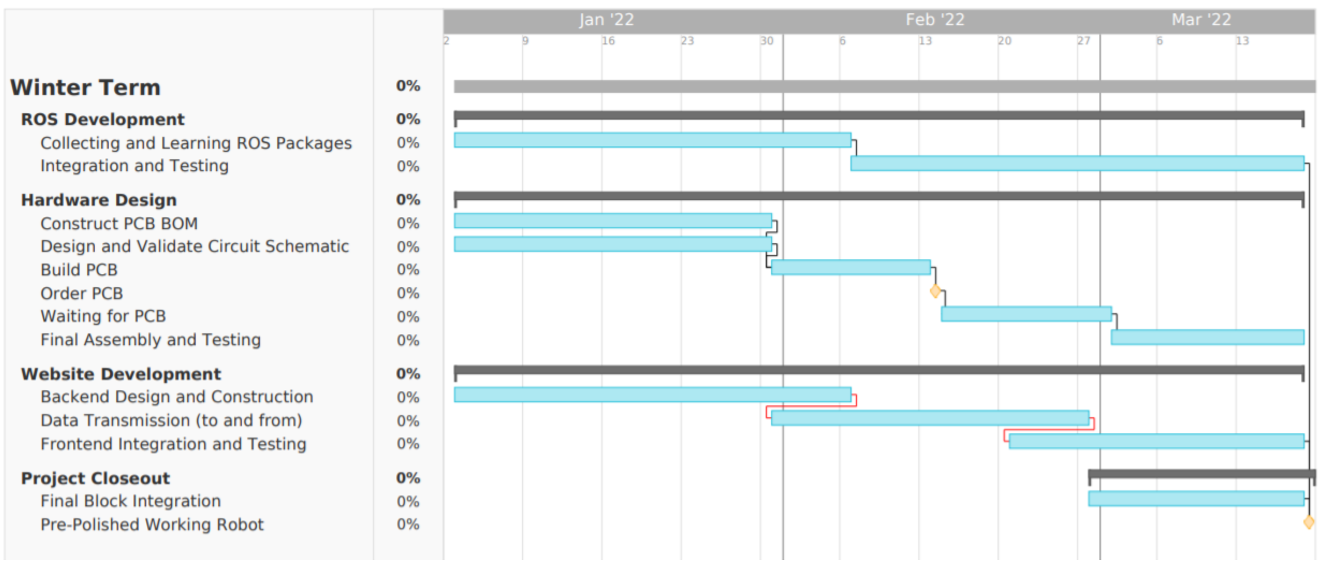

Figure 3: Gantt Chart timeline for Winter.

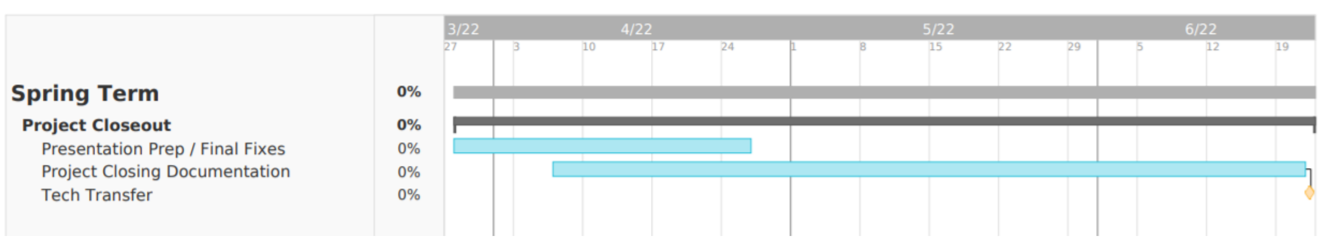

Figure 4: Gantt Chart timeline for Spring.

1.5 References

[1] Verified Market Research. "Autonomous Delivery Robots Market Size Worth $ 236.59 Million, Globally, by 2027 at 34.30 % CAGR: Verified Market Research®." GlobeNewswire News Room, Verified Market Research, 4 Oct. 2021, https://www.globenewswire.com/news-release/2021/10/04/

2308122/0/en/Autonomous-Delivery-Robots-Market-size-worth-236-59-Million-Globally-by-2027-at-34-30-CAGR-Verified-Market-Research.html.

[2] "Autopilot," Tesla. [Online]. Available: https://www.tesla.com/autopilot. [Accessed: 18-Oct-2021].

[3] "Enhanced Active Park Assist: Ford Co-Pilot 360™ technology," Ford Motor Company. [Online]. Available: https://www.ford.com/technology/driver-assist-technology/enhanced-active-park-assist/. [Accessed: 18-Oct-2021].

1.6 Revision Table

Section 1 - Overview Revisions	
Date	**Revision**
10/22/2021	Nick McBee: Review and grammar corrections prior to first submission.
10/16/2021	Nick McBee: Initial draft of Executive Summary created.
10/17/2021	Drew Gehrke: Initial draft of Gap Analysis.
10/18/2021	Nathan Searles: Team Standards and protocols agreed to and appended.
10/25/2021	Drew Gehrke: Added to Gap Analysis and new references.
10/28/2021	Nathan Searles: Added External Timeline.
11/02/2021	Drew Gehrke: Fixed figure / table captions, added Communication Analysis section, fixed text for the Timeline section
11/09/2021	Nathan Searles: Revised the Executive Summary
11/10/2021	Drew Gehrke: Revised sections in team protocol and standards table, added two new sections.
11/12/2021	Tyrone Stagner: Revised sections for Gap Analysis. Added another reference and adjusted the numbers for them.
12/02/2021	Nick McBee: Added contact table to Section 1.2 and current progress to Executive Summary.
12/03/2021	Drew Gehrke: Revised Gap Analysis to not use first person.
05/05/2022	Drew Gehrke: Modified Executive Summary to reflect final product.
05/05/2022	Nick McBee: Specified ROS2 version.

2 Project Scope

In this section, a more detailed outline of the project is described. Highlighted below are the list of project requirements derived from the Project Partner's expectations and acceptance criteria. There is also a risk assessment to guide mitigation tactics and action plans for a variety of potential situations that could arise during the course of this project. Additionally, a very extensive look into the potential and likely impacts of an Autonomous Package Delivery Robot is taken; how this would effect cultural norms, public health, safety, environmental, and economic impacts.

2.1 Requirements

2.1.1 Lock Box

Project Partner Requirement: The robot will have a lock box for transporting the package.

Engineering Requirement: The system will transport a package in a secure container unlocked via input from authorized users.

Verification Method: Test

Test Process:

1. Container will be opened by the user while the robot is not moving.
2. Package will be inserted into the container and closed.
3. The robot then moves to its destination with package in tow.
4. Once arrived, an authorized user will unlock the robot.
5. Once unlocked, container can be opened and package can be accessed.

Test Pass Condition: The package inside of the container will not be damaged upon arrival. The container cannot be opened unless unlocked via authorized user without breaking the container or mechanism. If the lock is unlocked via user input and the package is in tact the condition is met.

2.1.2 Emergency Stop

Project Partner Requirement: The robot should have an easy to access button to stop the robot in case of an emergency and to assist with testing procedures. The robot must also stop if a collision is detected.

Engineering Requirement: The system will shut down within 500ms after the emergency button or collision sensors activate.

Verification Method: Test

Test Process:

1. Begin a recording of the robot.
2. Command the robot to move forward at its standard operating speed with no obstacle in it's path.
3. Have someone push the emergency stop button to stop the robot.
4. Review the footage to ensure the robot stops within 500ms of the button being pushed.

5. Repeat the above steps with an obstacle in the path of the bump sensors.
6. Upon collision, ensure the robot stops within 500ms of the bump sensors being activated.

Test Pass Condition: The system has stopped moving within 500ms of the stop button being pressed as evidenced by the timer and camera recording. The system also stops 500ms after the collision has occurred.

2.1.3 Battery Monitoring

Project Partner Requirement: The robot should have a means of monitoring the voltage of its onboard batteries.

Engineering Requirement: The robot will measure the series voltage of its two lead acid batteries within an accuracy of 100mV.

Verification Method: Test

Test Process:

1. With the robot turned on and not moving, use a voltmeter to directly measure the battery voltage.
2. SSH into the Raspberry Pi and navigate to the working directory of the project.
3. Run the command "ros2 topic echo /battery".
4. After a brief delay, the battery voltage will be output to the terminal window.
5. Verify that the reported battery voltage is within 100mV of the read voltage.

Test Pass Condition: The reported battery voltage is within 100mV of the value measured with the voltmeter.

2.1.4 Edge Detection

Project Partner Requirement: The robot should stay on the sidewalk.

Engineering Requirement: The system will determine the bounds of pathways and maintain a minimum distance of 15cm from the edge of said pathway.

Verification Method: Demonstration

Test Process:

1. Place the robot so that it has a wall on either its left or right.
2. Set a waypoint to where the robot would collide with a wall if it went along a direct path to the waypoint.
3. Command the robot to traverse to the waypoint.
4. Observe and ensure the robot is capable staying at least 15cm away from the wall during its traversal.

Test Pass Condition: The robot maintained a minimum distance of 15cm from the pathway's edge. The robot did not travel off the bounds of the pathway.

2.1.5 Path Following

Project Partner Requirement: The robot should be able to make across campus deliveries.

Engineering Requirement: The system will follow a predefined path to its destination and deviate from that path by no more than 1 meter.

Verification Method: Test

Test Process:

1. Place a straight strip of tape down on the floor.
2. Align the robot along the strip.
3. Send the command to the robot to drive in a straight line.
4. Stop the robot once it has reached the end of the strip.
5. Identify the starting position from the end of the strip and ensure it is less than 1 meter.

Test Pass Condition: The robot did not deviate more than 1 meter from its path when it arrives at its destination.

2.1.6 Object Reaction

Project Partner Requirement: The robot should be able to go around stationary objects in its path.

Engineering Requirement: The system will traverse around stationary objects in its path and not get closer than 15cm to said object.

Verification Method: Test

Test Process:

1. Place an obstacle at least 5ft in front of the robot. obstacle should be at least 30cm wide and 90cm tall.
2. Start the robot along a straight path, towards a waypoint, with the obstacle in its path.
3. Leave enough space on at least one side of the obstacle for the robot to pass

Test Pass Condition: The system traversed around a stationary object in its path and did not get closer than 15cm to said object.

2.1.7 Data transferred from system to website

Project Partner Requirement: The system receives and transfers data to the website.

Engineering Requirement: The system will transfer IMU data to the website for users to view.

Verification Method: Test

Test Process:

1. Ensure robot system is online and IMU data is being loaded.

2. Access the web-page via the IP address of the website.
3. Input IP address and port number into ROS URL input field.
4. Hit "Toggle Connect" button on website.
5. Ensure all topics are showing on the website.
6. From the console log of the browser, look at the messages being displayed from the IMU data topic from the system.

Test Pass Condition: The system transferred IMU data to the website for users to view then the test has passed.

2.1.8 Data transferred from website to system

Project Partner Requirement: The system will receive data from the website.

Engineering Requirement: The system will receive data from website.

Verification Method: Test

Test Process:

1. Ensure robot system is online.
2. Access the web-page via the IP address of the website.
3. Input IP address and port number into ROS URL input field.
4. Hit "Toggle Connect" button on website.
5. Wait for lockbox to unlock

Test Pass Condition: The system received data from website then the test has passed.

2.2 Design Impact Statement

2.2.1 Public Health, Safety, and Welfare

Safety risks are a major concern with a robot that will be frequently interacting with pedestrians and vehicle traffic. By introducing more traffic to sidewalks and crosswalks the chance of collisions increases. Cases have been reported of individuals with accessibility needs being blocked by an Autonomous Delivery Robot (ADR) [2]. It is crucial that mobile delivery robots be equipped with the caution and capability to prepare for and react to a multiplicity of unforeseen environmental factors. That is not to say weather or road conditions, but rather an awareness of physical objects and especially people (both mobile and stationary) within the robots immediate surroundings. The real world can provide challenging obstacles to overcome, particularly for machines that fundamentally perceive the world in black and white (1 or 0). There are a few design considerations the team should abide by to mitigate these risks. One is that the robot should move in a predictable manner at all times. This means reducing the amount of stopping and readjustment of course. This allows for people in the robot's surroundings to react with ample time. Another design addition will be the inclusion of safety sensors to ensure the robot immediately stops its movement if any physical contact is made. Next, it should never position itself in a place that would prevent someone, especially a person in a wheelchair, from continuing along their path of travel. This aids to reduce any risk of other pedestrians having to navigate into dangerous situations in which the robot is an obstacle.

Another safety and welfare concern is in the use of lead acid batteries. Lead is a very toxic element and has the potential to cause many health issues for the public if not used properly. Lead poisoning is a prevalent issue for those who work with lead acid batteries. In Bangladesh, many motor vehicle companies have seen a rise in lead poisoning in their workers. Roughly 97% of the lead acid batteries used in these facilities comes from recycled batteries and scrap metal [3]. This continued use of the same batteries leads to lead exposure to these workers and eventual lead poisoning. In this project, the lead acid batteries used are recycled batteries, but these batteries are the same batteries already equipped in the motorized chair. This lowers the risk of lead exposure for any users of the robot. To mitigate the need for new batteries and the potential for lead exposure, a battery monitoring system will be developed to monitor the health of the batteries and ensure there are no problems with them during continual use of the robot. This will lower the chance of any lead poisoning from this project as the batteries will be monitored continuously.

2.2.2 Cultural and Social

Autonomous delivery robot technology is becoming a staple of daily life. With Oregon State University as anecdotal evidence to this, students were originally surprised by the fleet of "Starships" roaming around the sidewalks on their own accord. However, this system was quick to integrate with the culture at OSU. In a larger study [3] where consumer acceptance was directly assessed, it was found that consumer's attitudes towards ADRs are growing more positive. This is in correlation with the increase in online shopping and delivering goods for physical travel to retail centers. More individuals are ordering groceries to their homes every year. Amazon alone has seen at least a 3% growth in retail market share per year, for the last five years [4]. This shift in two consumer attitudes towards delivered goods shows no immediate signs of decreasing or reversing direction. Amazon has even shifted their delivery services to an entirely in house operation, as seen with numerous Prime delivery vans in the area.

2.2.3 Environmental

A major environmental impact of this project is to reduce greenhouse gas emissions using an electrically powered robot. Last mile delivery accounts for a small percentage of greenhouse gas emissions in the large aspect of package delivery, but is still a prevalent concern. One study found that electric delivery trucks making frequent stops in a large city emitted up to 61% fewer greenhouse gases and at least a third less energy overall compared to their traditional diesel counterparts [6]. However, these statistics are for delivery truck-sized vehicles which carry and deliver dozens of packages before returning to a centralized distribution location. With a small robot that only fits one package at a time it will need to navigate back and forth many times to achieve the same throughput of a delivery truck. The use of a small robot aids in the battle against increased congestion of roadways which in turn leads to increased emissions of carbon and pollutants. This project will reduce this risk by eliminating the need for a vehicle which produces these emissions. The robot is powered via two lead acid batteries and will not emit any greenhouse gases. The robots will be used in small range applications and will not have long distances to travel in order to receive a new package.

2.2.4 Economic

Robots displaced 670,000 manufacturing workers between 1990 and 2007 in the U.S alone [7]. This rate has only increased in the last decade, inspiring political candidates like Andrew Yang to hold this amongst his top issues to run for office on. Anecdotally we have seen major waves in technology that extends outside of manufacturing. Self-checkout is a new expectation for any chain grocery retailer. Tesla has built many of its manufacturing processes upon robotics, seeking to nearly eliminate the need for people to work on the assembly lines. Likewise, their growing investment in autonomously driving vehicles will soon displace taxi drivers or even public transportation operators. Dedicated secretaries to answer phone calls have been displaced by intelligent answering machines and call routing services. With the latest developments from American company, Boston Dynamics, and their variety of bipedal, quadrupedal, and track based robots, warehouse jobs and distribution center positions are also on the way out [8]. However, many of these concerns are greatly exaggerated, yet it is still estimated that "9% of all workers in the US face a risk of automation that exceeds 70%". This discrepancy is due to the fact that most occupations have niche tasks which cannot be automated away, meaning most workers cannot be entirely replaced [9]. However, certain labor categories are at higher risk than others, with delivery drivers falling within one of those categories. Therefore, it is likely that a commercialized version of this delivery robot system, alongside other autonomous vehicle systems will contribute to the elimination of delivery driver positions.

2.3 Risks

Table 3: Risk assessment table

ID	Description	Category	Probability	Impact	Performance Indicator	Responsible Party	Action Plan
R1	Vendor Delays	Timeline	80%	Medium-High	Lead times, in-stock quantity	Andrew	Reduce number of orders
R2	Data Loss	Technical	15%	Medium-High	Repo version checking	Tyrone	Avoid large gaps in commits
R3	Hardware physically damaged	Cost/ Technical	20%	Low-Medium	Inspection of components	Drew	Reduce potential for breaking
R4	Reckless Endangerment Liability	Legal	30%	High	Situational safety testing	Nathan	Avoid dangerous interactions with robot
R5	Computational Limitations	Technical	7%	Medium	Process runtime	Tyrone	Retain
R6	Hardware damage by improper connections/ use	Cost / Technical	15%	Medium	Component failure rate	Nick	Avoid improper connections
R7	Procrastination	Timeline	35%	High	Due dates not being met	Full Team	Avoid overwhelming teammates
R8	Communication disconnect	Technical	65%	Medium	Loss of connection to robot	Andrew	Retain

2.4 References

[1] "Risk Managment," Pace University. [Online]. Available: http://csis.pace.edu/ marchese/SE616/L11New/
[Accessed: 25-Oct-2021].

[2] E. Ackerman, "My Fight With a Sidewalk Robot," NCDJ, 19-Nov-2019. [Online]. Available: https://ncdj.org/2019/11/my-fight-with-a-sidewalk-robot/. [Accessed: 06-Dec-2021].

[3] S. A. Ahmad, M. H. Khan, S. Khandker, A. F. Sarwar, N. Yasmin, M. H. Faruquee, and R. Yasmin, "Blood lead levels and health problems of lead acid battery workers in Bangladesh," The

Scientific World Journal, vol. 2014, pp. 1–7, Feb. 2014.

[4] A. Pani, S. Mishra, M. Golias, and M. Figliozzi, "Evaluating public acceptance of autonomous delivery robots during COVID-19 pandemic," Transportation Research Part D: Transport and Environment, vol. 89, p. 102600, Dec. 2020.

[5] S. Chevalier, "U.S. Amazon Market Share 2021," Statista, 13-Oct-2021. [Online]. Available: https://www.statista.com/statistics/788109/amazon-retail-market-share-usa/. [Accessed: 30-Oct-2021].

[6] D.-Y. Lee, V. Thomas, and M. Brown, "Electric Urban Delivery Trucks: Energy Use, Greenhouse Gas Emissions, and Cost-Effectiveness," ACS Publications, Jun-2013. [Online]. Available: https://pubs.acs.org/doi/full/10.1021/es400179w. [Accessed: 29-Oct-2021].

[7] C. C. Miller, "Evidence that robots are winning the race for American Jobs," The New York Times, 28-Mar-2017. [Online]. Available: https://www.nytimes.com/2017/03/28/upshot/evidence-that-robots-are-winning-the-race-for-american-jobs.html. [Accessed: 30-Oct-2021].

[8] O. Garmash, "(PDF) the electronic scientifically and practical journal 'Intellectualization of Logistics and Supply Chain Management', v.1 (2020) ISSN 2708-3195," ResearchGate. [Online]. Available: https://www.researchgate.net/publication/343724234 The electronic scientifically and practical journal INTELLECTUALIZATION OF LOGISTICS AND SUPPLY CHAIN MANAGE MENT v1 2020 ISSN 2708-3195. [Accessed: 30-Oct-2021].

[9] M. Arntz, T. Gregory, and U. Zierahn, "Revisiting the risk of automation," Economics Letters, 15-Jul-2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165176517302811?casa_token=vK_XNoLq-2AAAAAA%3A0hDs7hCFyx1ar8gVh66FOr27ukmrPgIwONqPLK0FKIqM3Hd0JM_qBYwSOlvu5SMEiDt5JOm7Yg#fig1. [Accessed: 30-Oct-2021].

2.5 Revision Table

Section 2 - Project Scope Revisions	
Date	Revision
10/29/2021	Nathan Searles: Added engineering tasks
10/29/2021	Tyrone Stagner: Entered data into risk table
10/29/2021	Drew Gehrke: Added Lock Box requirement and formatting for other requirements
11/02/2021	Drew Gehrke: Added coloration to the risks table
11/12/2021	Nicholas McBee: Overhauled formatting, assigned individual responsible parties, and revised some indicators.
11/18/2021	Nicholas McBee: Revised Emergency Stop specifications based on project partner feedback
11/29/2021	Drew Gehrke: Added Probability column to Risk Table
11/30/2021	Drew Gehrke: Reformatted engineering requirements section
12/03/2021	Drew Gehrke: Updated Risk Table action plans
05/04/2022	Drew Gehrke: Updated all requirements to reflect proper testing procedure and test pass conditions.
05/05/2022	Drew Gehrke: Updated edge detection requirement to reflect proper testing procedure and test pass condition.
05/06/2022	Drew Gehrke: Added Design Impact Statement section

3 Top Level Architecture

3.1 Block Diagram

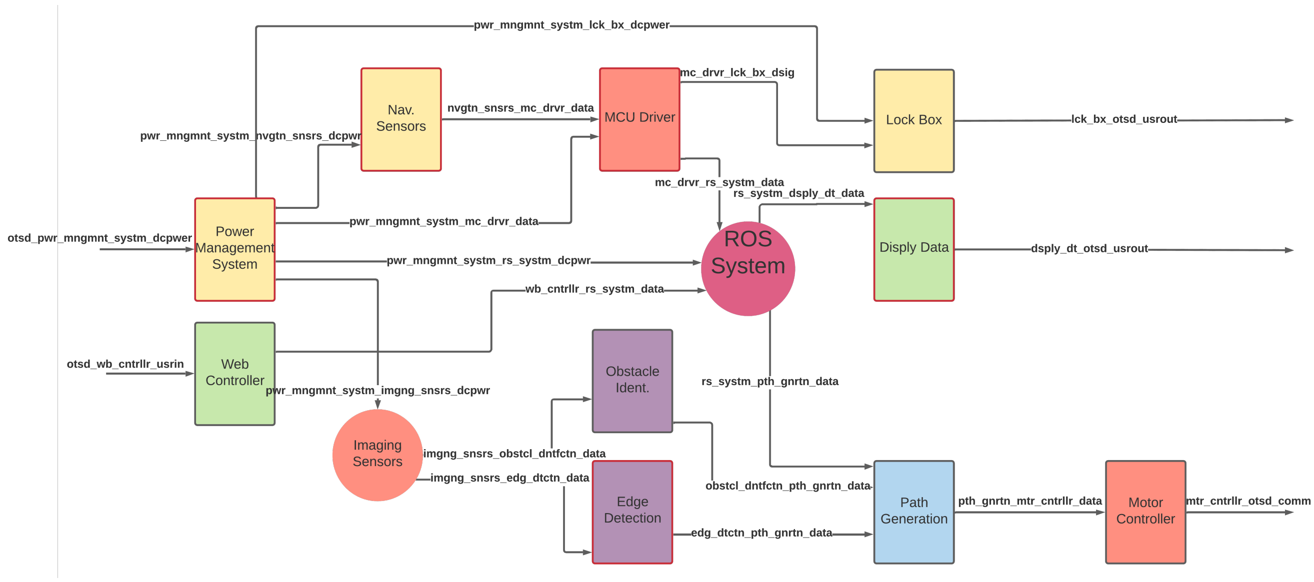

Figure 5: Block diagram of the final APDR system

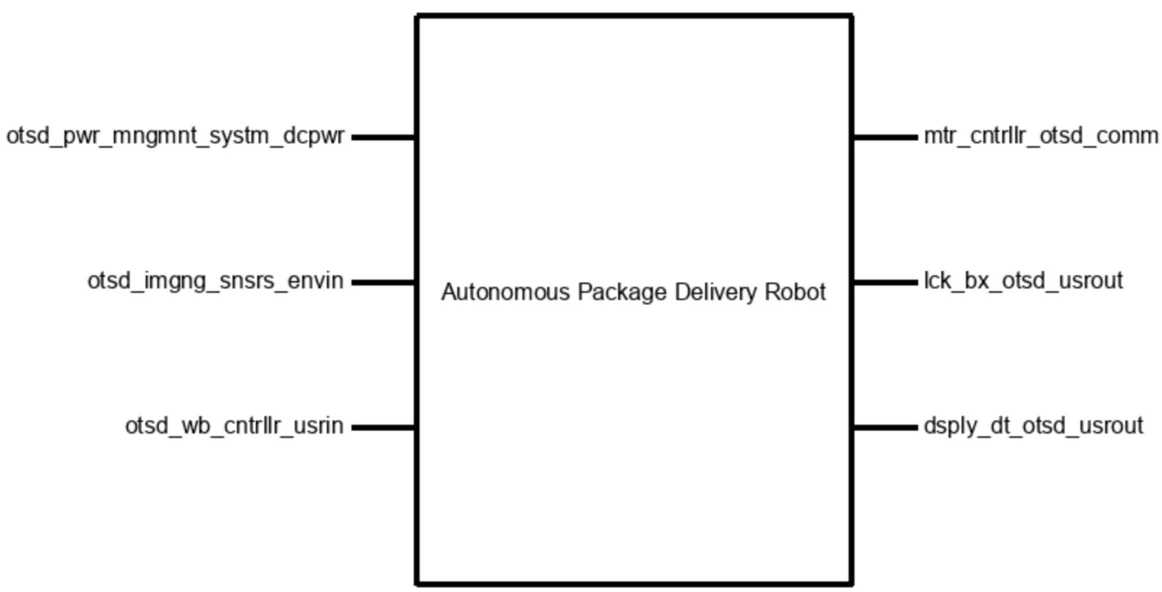

Figure 6: Black Box Diagram of APDR system

3.2 Block Descriptions

3.2.1 Display Data

Champion: Tyrone Stagner
The Display Data block will be the web application that displays system diagnostics. Some of the system diagnostics include the battery status, position, and error codes. The position will be displayed on a map. Battery status will be displayed showing numbers, and error codes will be displayed as a number and brief description of the error.

3.2.2 Edge Detection

Champion: Nathan Searles
The Edge Detection will serve as a precursor to the Path Generation block within the system. It will rely on optical sensors to receive environmental data. Using computer vision, this block will run the Canny Edge detection algorithm [1] to identify edges within the frames. This array will be filtered to only identify edges of interest. From here they will be passed through a transform matrix to convert the vectors into an aerial view coordinate system.

3.2.3 Imaging Sensors*

Champion: Andrew Pehrson
The The Imaging Sensors block is a sensor block which contains sensors used for taking in the environment. A Light Detection and Ranging unit, or LiDAR, will assist in obstacle detection and a camera will be used for edge detection, as well as inspection of surroundings.

3.2.4 Lock Box

Champion: Drew Gehrke
The Lock Box block is a mechanical and enclosure block which will contain a physical box with an electronically manipulated lock. It will receive a control signal from the microcontroller which will disengage the locking mechanism. Upon receiving the signal, an indicator light will show that the box is ready to be opened. When the lid is lifted, a signal will be sent back to the microcontroller to indicate the lid is still open and will turn off once the lid is closed.

3.2.5 MCU Driver

Champion: Andrew Pehrson
The microcontroller driver block's main purpose is to process and pass on sensor data from the navigation sensors block to the ROS system. This block consists of code on an ESP32, focused around the Rosserial Arduino library. The Rosserial Arduino library handles all serial communication and allows the microcontroller to be treated as a node within the ROS architecture. This allows the ESP32 to post and subscribe straight to ROS topics.

3.2.6 Motor Controller

Champion: Andrew Pehrson
The motor controller block is a software block that will pull movement commands and then transmit the data over serial to the motor controller board. The code for this block will act as a node in the ROS topic transport protocol and send data over a UART connection to the systems motor controller board the SmartDriveDuo. The movement commands will be pulled from the cmd_vel topic built into the ros navigation stack.

3.2.7 Navigation Sensors

Champion: Drew Gehrke
The Navigation Sensors block is a sensor block used to aid in the navigation and trajectory of the robot. Output data from various different sensors will act as inputs for the MCU driver block in order to gather and translate the data. Sensors include the GPS used for current position of the robot and the path generation, an inertial measurement unit, or IMU, used for current orientation and path correction, and emergency bump sensors being used in case the robot

hits an obstacle that the avoidance did not account for.

3.2.8 Obstacle Identification

Champion: Nathan Searles
The Obstacle Identification block is a code block which adjusts the robot's current course should an obstacle be identified. Based on a cost map created from the LiDAR data, an obstacle will be identified and a path will be charted to avoid said obstacle in the Path Generation block. The cost map will be created from a Robot Operating System, or RoS, package.

3.2.9 Path Generation

Champion: Nick McBee
The Path Generation block is a code block which will determine the path of traversal the robot will take. Based on a set of way points, the robot will navigate to the specified end goal, adjusting for obstacles when they have been identified. Path planning between way points and obstacle avoidance will be handled by the algorithms included in the ROS navigation stack.

3.2.10 Power Management System

Champion: Nick McBee
The Power Management System block is an electrical block which reduces the 24VDC battery voltage down to 12V, 5V, and 3.3V for various sensors and electronics to use. The battery voltage is also monitored via a voltage divider so the value can be read by an ADC to track battery state of charge.

3.2.11 ROS System*

Champion: Andrew Pehrson
The ROS system block represents the set of software libraries and tools we are using from the ROS2 galactic suite. This includes the ROS topic style Inter-process communication, package management, and navigation algorithms. Nothing within the ROS system block was made by us, it is included since it is central to the work we did on the robot.

3.2.12 Web Controller

Champion: Tyrone Stagner
The web controller block will be the web application that controls the system. The web controller will send data to the ROS system for the lockbox, approval, destination, and stop features. The lockbox will have the ability to be unlocked. The approval will send a code to the ROS system to tell the ROS system it is ok to proceed. The destination will send a code that tells the ROS system to go to a predetermined waypoint. The stop will make send a code to stop the ROS system.

***Note:** These blocks were not validated as they were a part of other blocks within the system. These serve to show the flow of the system as a whole.

3.3 Interface Definitions

The following tables will define the specific properties associated with the interfaces defined in Figure 5. These definitions will provide a specific profile of the interactions between user input and output and define the scope of intermediate steps required to fulfill those I/O specifications.

Table 4: Input Interface Definitions

Interface	Name	Properties
otsd_pwr_mngmnt_systm_acpwr	Wall Power	• V_{nom}: 120VAC • I_{peak}: 15A • $I_{nominal}$: 500mA • Other: NEMA 5-15R
otsd_wb_cntrllr_usrin	Website User Input	• Other: User can set Approval state • Other: User can set Stop state • Other: User can set Lockbox state • Other: User can set Destination state

Table 5: Internal Interface Definitions

Interface	Name	Properties
edg_dtctn_pth_gnrtn_data	Path Boundary	• Datarate: Minimum of 10 images per second • Other: Minimum range of 3 meters • Other: Image mapping of XY Plane
imgng_snsrs_obstcl_dntfctn_data	Point Cloud	• Other: Minimum range of 7 meters • Other: Angular resolution of 2pt-s/degree • Other: Rotation Frequency of 10Hz
imgng_snsrs_edg_dtctn_data	Video Signal	• Video stream • Datarate: 60 frames per second • Messages: 720p (5MP)
mc_drvr_lck_bx_dsig	Unlock Signal	• Logic-Level: Active high for unlock • Other: dsig returned • Vnominal: 3.3V

Table 6: Internal Interface Definitions, contd.

Interface	Name	Properties
mc_drvr_rs_systm_data	Sensor Data	• Messages: Battery voltage • Messages: Orientation (IMU) • Messages: Bool (bump sensors) • Messages: Position (GPS) • Protocol: ROS Topic
nvgtn_snsrs_mc_drvr_data	Positioning and Orientation Data	• Messages: Position (GPS) • Messages: Orientation (IMU) • Other: Dsig (bump sensors)
obstcl_dntfctn_pth_gnrtn_data	Costmap	• Other: XY plane image mapping, internal ROS coord. system • Other: Localization for robot position within 5cm • Protocol: 2D point cloud of obstacle locations
pth_gnrtn_mtr_cntrllr_data	Vector Motor Code	• Messages: Linear velocity • Messages: Angular Velocity • Protocol: ROS Topic cmd_vel message
pwr_mngmnt_systm_imgng_snsrs_dcpwr	Imaging Sensor Power	• Vmax: 5.2V • Vmin: 4.8V • Ipeak: 3A • Inominal: 2A
pwr_mngmnt_systm_lck_bx_dcpwr	Lock Power	• Vmax: 12.2V • Vmin: 11.8V • Vnominal: 12V • Ipeak: 2.1A • Inominal: 2A
pwr_mngmnt_systm_mc_drvr_data	Battery Status	• Other: Vmin of 0.15V • Other: Vmax of 2.45V • Protocol: Analog voltage signal
pwr_mngmnt_systm_nvgtn_snsrs_dcpwr	Nav. Sensor Power	• Vmax: 3.6V • Vmin: 2.7V • Ipeak: 100mA • Inominal: 50mA

Table 7: Internal Interface Definitions, contd.

Interface	Name	Properties
pwr_mngmnt_systm_rs_systm_dcpwr	R. Pi Power	• Vmax: 5.2V • Vmin: 4.8V • Ipeak: 3A • Inominal: 2A
wb_cntrllr_rs_systm_data	Web-System Interface	• Messages: Lockbox • Messages: Approval • Messages: Destination • Messages: Stop
rs_systm_pth_gnrtn_data	Waypoint Generation	• Messages: Integer specifying destination point • Other: Only send when current routing is complete • Protocol: ROS action goal
rs_systm_dsply_dt_data	System-Web Data	• Messages: Error data • Messages: Battery data • Messages: GPS data • Protocol: .json file

Table 8: Output Interface Definitions

Interface	Name	Properties
dsply_dt_otsd_usrout	Website User Interface	• Type: string • Type: numbers • Usability: 9 out of 10 people are able to sign in within 5 minutes
lck_bx_otsd_usrout	Lock Box User Interface	• Type: Switch • Type: Lid • Type: Light
mtr_cntrllr_otsd_comm	Motor Controller Output	• Messages: Left and right motor speeds • Protocol: SmartDriveDuo Serial simplified • Protocol: Reverse

3.4 References and File Links

3.4.1 References (IEEE)

[1] “Canny edge detection,” OpenCV. [Online]. Available: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html. [Accessed: 06-May-2022].

3.4.2 File Links

3.5 Revision Table

Section 3 - Top Level Architecture Revisions	
Date	Revision
11/18/2021	Nathan Searles: Added block diagram
11/19/2021	Drew Gehrke: Added black box diagram and block descriptions.
11/19/2021	Nathan Searles: Added Interface Definitions
11/29/2021	Drew Gehrke: Revised block descriptions
12/1/2021	Nathan Searles: Reformatted Interface Definition table
12/3/2021	Drew Gehrke: Changed interface definition names and added to properties
12/3/2021	Tyrone Stagner: Updated Block diagram
05/05/2022	Drew Gehrke: Updated block diagram picture, updated all block definitions, updated all interface definitions.
05/06/2022	Nathan Searles: Update edge detection block description.
05/06/2022	Nick McBee: Updated Power Management System block description.

4 Block Validations

4.1 Motor Controller

4.1.1 Block Overview

The motor controller block is a software block that will be championed by Andrew Pehrson.The main function of the motor controller block is to pull movement commands and then transmit the data over serial to the motor controller board. The code for this block will act as a node in the ROS topic transport protocol and send data over a UART connection to the systems motor controller board the SmartDriveDuo. The movement commands will be pulled from the cmd_vel topic built into the ros navigation stack. These movement commands will then be transformed into a left motor speed and right motor speed. Once these speeds are found they will be packaged according to the serial packetized instructions expected by the SmartDriveDuo. This data will then be sent over a uart connection to the SmartDriveDuo.

4.1.2 Block Design

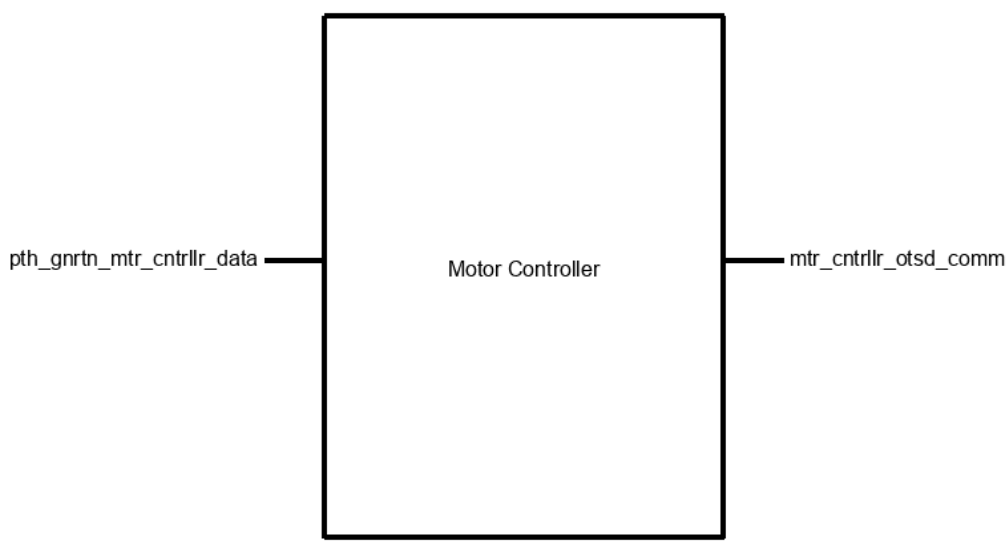

Figure 7: Black box schematic of Motor Controller block.

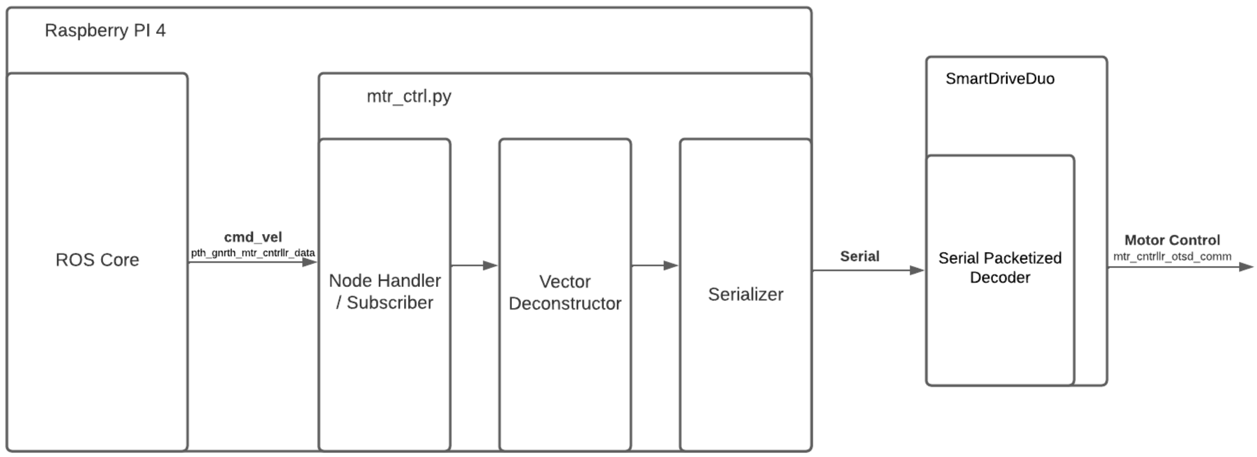

Figure 8: Flow Diagram for the Motor Controller block.

4.1.3 Block General Validation

The motor controller code passed down from the previous team is usable but for multiple design changes and possible areas of improvement we are programming our own. The biggest reason for us making our own motor controllers is our decision to migrate to ROS2. By migrating to ROS2 we get more tools, better pathing algorithms, and better drivers for cameras over ROS1. This does however mean that none of the previous team's code can be used. The code can however be adapted and I expect to reference their work but another change in design we are making is using the SmartDriveDuo's serial packetized protocol instead of the serial simplified protocol the preexisting code uses. The previous team had decided to use serial communication over PWM since it wont randomly drive the motors if connection is lost therefore making the robot safer. One issue they had run into however was that when using serial simplified, which lets you stream command data to the SmartDriveDuo, is that on start up the robot would follow whatever behaviors the connection had which in one occasion sent the robot spinning in nonstop circuals. Using serial packetized will require each command to need a proper header and checksum practically removing the concern for unexpected behaviors.

4.1.4 Block Interface Validation

Table 9: pth_gnrth_mtr_cntrllr_data

Protocol: ROS Topic	The ROS navigation stack exports its movement needs into a topic called cmd_vel	The motor controller code will be a node within ROS that can subscribe and post to the needed topics
Messages: Angular velocity	An angular velocity vector is expected to be given from the cmd$_v$*eltopic*	The motor controller code will take the angular velocity vector and use to calculate the difference in left, right motor speed
Messages: Linear velocity	An linear velocity vector is expected to be given from the cmd_vel topic	The motor controller code will take the linear velocity vector and use it to calculate the summed forward speed of the robot.

Table 10: mtr_cntrllr_otsd_comm

Protocol: Startup procedure	The SmartDriveDuo needs a 1 second delay on startup and then a dummy byte of 0x80 to auto fetch the baud rate	The motor controller code will run a function that initializes the SmartDriveDuo connection before handling velocity commands.
Protocol: SmartDriveDuo Serial Packetized	The SmartDriveDuo has multiple ways to communicate with it. A PWM is dangerous if the board were to disconnect. Serial simplified can also drive the motors randomly when garbage values are given.	Communicating with the SmartDriveDuo using serial packetized ensures that garbage data is unlikely to drive the motors since it needs an acceptable header to take velocity inputs.
Messages: Left, Right Motor speeds	These are the expected command value meanings where a bit is set for which motor is being driven and a value from 0 - 255 is used to give a throttle percentage	The motor controller code will process the angular and linear velocity vectors into a left and right motor speed.

4.1.5 Block Testing Process

A Raspberry Pi running ROS 2 will be connected to the motor controller over uart serial. The motor controller code will be saved on the Raspberry Pi

1. Power on the Raspberry Pi

2. Run the roscore command to start up ROS
3. Run the motor controller python script
4. Run the teleop_twist_keyboard script to use manual controls
5. Display the cmd_vel topic
6. Drive the robot forward, backwards, left, and right.

4.1.6 References and File Links

References

[1] hannabanana96, "Home · Hannabanana96/mpdr_masters wiki," GitHub. [Online]. Available: https://github.com/hannabanana96/MPDR_Masters/wiki. [Accessed: 05-Feb-2022].

[2] "Teleop_twist_keyboard," ROS Index. [Online]. Available: https://index.ros.org/p/teleop_twist_keyboard/ ros2-teleop_twist_keyboard/. [Accessed: 05-Feb-2022].

4.1.7 Revision Table

Section 4.1 - Motor Controller Revisions	
Date	Revision
02/04/2022	Andrew Pehrson: Draft Written
02/18/2022	Andrew Pehrson: 4.1.1 Condensed run-ons, 4.1.2 Better label input and output to main block, 4.1.3 Expand on why ROS2, 4.1.4 'protical' corrected to protocol, 4.1.4 Made startup procedure better quantifiable, 4.1.5 explain roscore, 4.1.5 link code referenced, 4.1.6 more links added

4.2 Navigation Sensors

4.2.1 Block Overview

The Navigation Sensors block is a sensor block used to aid in the navigation and trajectory of the robot. Output data from various different sensors will act as inputs for the MCU driver block in order to gather and translate the data. Sensors include the GPS used for current position of the robot and the path generation, an inertial measurement unit, or IMU, used for current orientation and path correction, and emergency bump sensors being used in case the robot hits an obstacle that the avoidance did not account for. The block champion for this block is Drew Gehrke.

4.2.2 Block Design

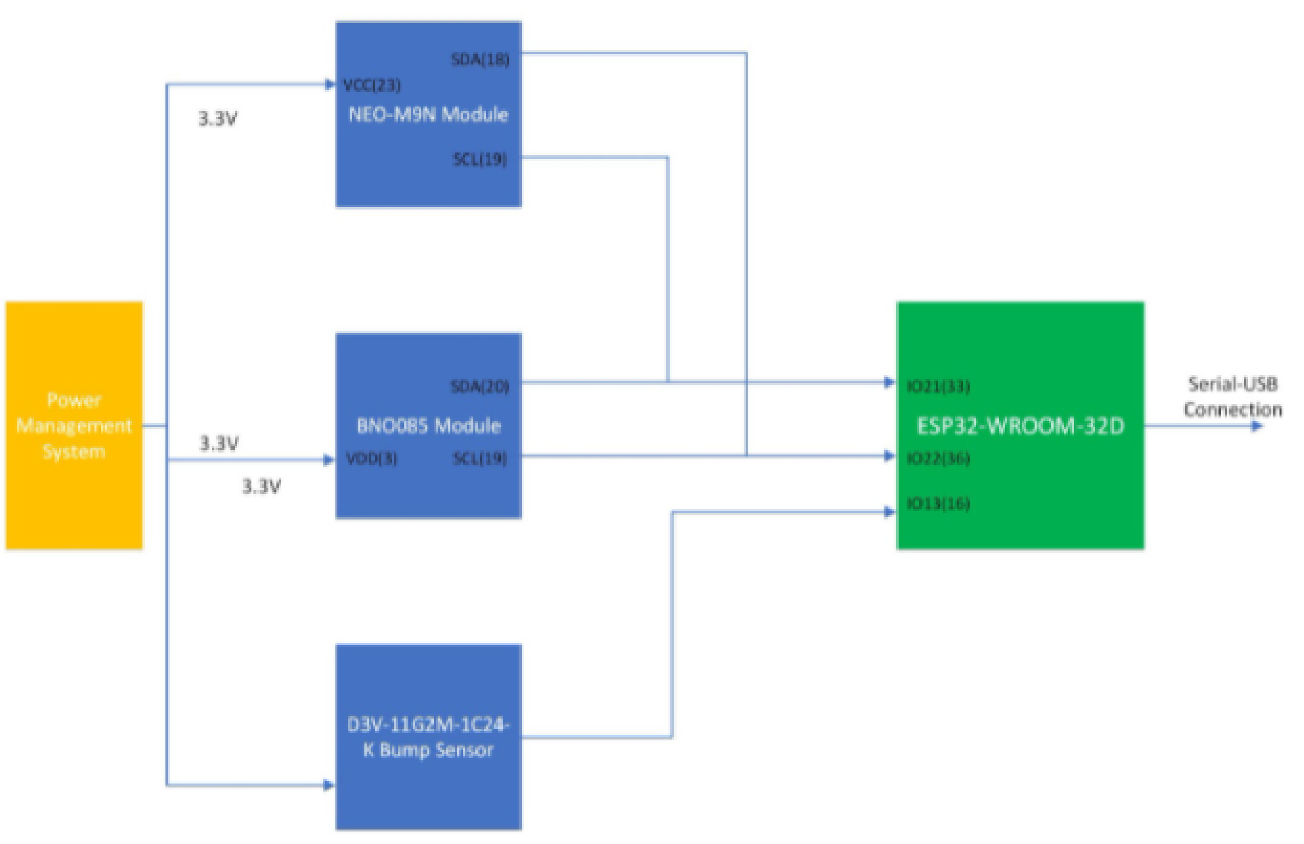

Figure 9: Electrical schematic for the Navigation Sensors block.

Communication between the GPS and IMU sensors will be done over I^2C to reduce the number of wires being used. The ESP32 will pull the information from those sensors and then parse it to be sent off to the computer to process and eventually RoS will use that data in the various topics.

Note: The ESP32 microcontroller is used in the MCU Driver block. The code used on it will be developed by myself and another member of the team.

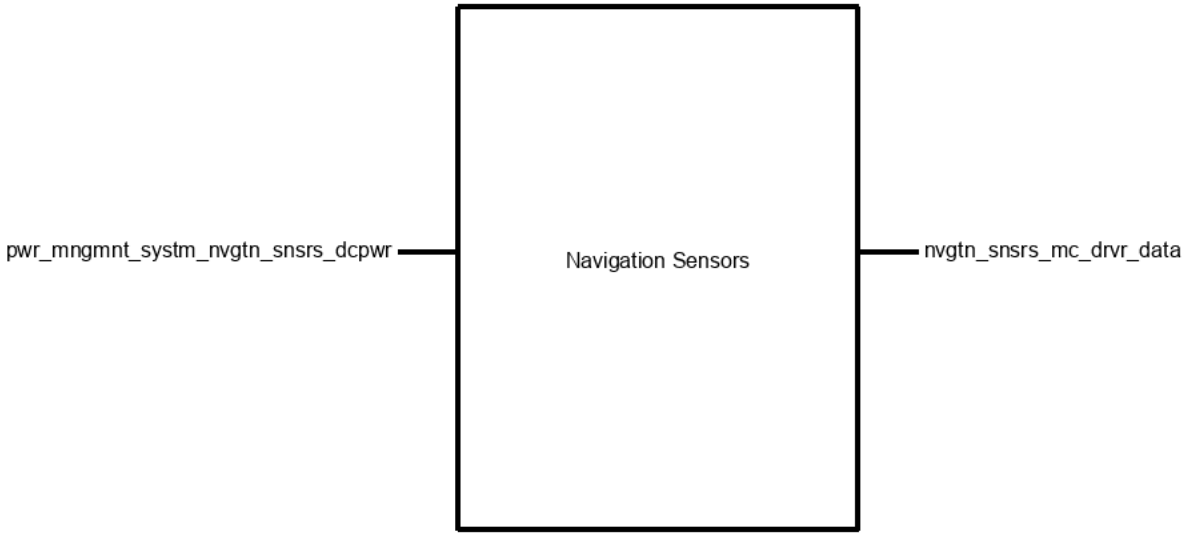

Figure 10: Black box schematic of Navigation Sensors block.

4.2.3 Block General Validation

Given many of the components from the work the Project Partner had previously done, these components will be utilized to create this block. The GPS and IMU sensors were not fully incorporated into the system prior to the team inheriting this project. The goal is to establish these components as core pieces of the overall system to improve the system as a whole. The IMU will be utilized to aid in correcting drift from the motors during traversal. The GPS will assist in traversal as well as sending information to the webpage updating the current position of the robot. This sensor has a 2 meter tolerance which will be observed and used to determine if the outputs are correct. The bump sensors were also to be used on the project previously, but were not implemented. The team will utilize these sensors to aid in traversal by providing a digital signal in case of hitting any obstacles which were not accounted for from the Obstacle Identification block.

4.2.4 Block Interface Validation

Table 11: Interface Validation for pwr_mngmnt_systm_nvgtn_snsrs_dcpwr - Power Input

Vmax: 3.6V	The IMU and NEO-M9N have a maximum voltage rating of this value. The optimal / typical value is 3.3V.	For the BNO080 IMU: Maximum voltage rating of 3.63V (Figure 6-1, pg 45) For the NEO-M9N in SMA: Maximum voltage rating of 3.6V (Table 11, pg 10)
Vmin: 2.7V	The minimum voltage for both the IMU and the NEO-M9N.	For the BNO080 IMU: Minimum voltage rating of 1.7V for power supply (Figure 6-2, pg 45) For the NEO-M9N in SMA: Minimum voltage rating of 2.7V for power supply (Table 11, pg 10)
Ipeak: 100 mA	The NEO-M9N has a peak current rating of 100 mA during acquisition of position.	For the NEO-M9N in SMA: Peak current rating of 100 mA (Table 12, pg 11)
Inominal: 50 mA	The NEO-M9N has a nominal current rating of 50 mA during acquisition of position.	For the NEO-M9N in SMA: Peak current rating of 50 mA (Table 12, pg 11)

4.2.5 Block Testing Process

1. Apply power to all the sensors
2. Wait for start up configurations to process
   - Wait for connection confirmation message from GPS to satellite, allow IMU to gather initial position.
3. Check for GPS coordinates from serial output
   - Compare longitude and latitude values to those from a phone GPS within tolerance.
4. Check for IMU position data from serial output
   - Compare values seen to the orientation of the robot in the current position.
5. Check bumpers for digital signal
   - Push down switches and see if anything reads in serial output.
6. Power off sensors

Table 12: Interface Validation for nvgtn_snsrs_mc_drvr_data - Data Management Output

Messages - Orientation (IMU)	The IMU will return information about the current orientation of the robot using I^2C. This will be used to determine path generation and potential drift.	For the BNO080 IMU: The message is described as being a series of hex values indicating the index, yaw, pitch, roll, and X- Y-, and Z- accelerations (Section 1.2.5.2, pg 11)
Messages - Position (GPS)	These messages will come from the GPS unit and be sent to the MCU Driver block via I2C to determine current position.	Using a predefined library (as seen in this example), the values for longitude, latitude, and number of satellites.
Message - 3.3V DSIG (Bump Sensors)	A digital signal from the bump sensors will be triggered when it is hit.	The sensors will be connected to a pull-down resistor GPIO to provide this functionality. The sensors will be connected between GND and the GPIO and will act as a digital boolean value.

4.2.6 References and File Links

References

[1] B. Siepert, "Adafruit 9-DOF orientation IMU Fusion Breakout - BNO085," Adafruit Learning System.[Online]. Available: https://learn.adafruit.com/adafruit-9-dof-orientation-imu-fusion-breakout-bno085. [Accessed: 21-Jan-2022].

[2] E. the Sparkiest, "SparkFun GPS NEO-M9N Hookup Guide," SparkFun. [Online]. Available: https://learn.sparkfun.com/tutorials/sparkfun-gps-neo-m9n-hookup-guide/all. [Accessed: 21-Jan-2022].

Files

- NEO M9N GPS Datasheet
- NEO M9N GPS Integration Manual
- BNO080 IMU Datasheet
- Bump Sensor Datasheet
- Project Partner GitHub

4.2.7 Revision Table

Section 4.2 - Nav. Sensors Revisions	
Date	Revision
01/04/2022	Drew Gehrke: Initial page made, added block overview
01/06/2022	Drew Gehrke: Started Interface Definition section, added references
01/07/2022	Drew Gehrke: Added more to interface tables, added more to other sections
01/08/2022	Drew Gehrke: Finalized draft sections
01/09/2022	Drew Gehrke: Revised sections to remove LiDAR from block
01/17/2022	Drew Gehrke: Revised interface names with new blocks
01/21/2022	Drew Gehrke: Revised interface properties to include bump sensors, images for black box and schematic updated. Finalized the interface properties.
03/05/2022	Drew Gehrke: Added all sections to project document

4.3 Edge Detection

4.3.1 Block Overview

The Edge Detection will serve as an important precursor to the Path Generation block in the Autonomous Package Delivery Robot. It will be able to determine the bounds of travel for the robot by using optical sensors and computer vision algorithms to identify the edges of sidewalks, roadways, and paths. The output interface will be a cost map that the robot will interpret for its path generation algorithms.

Since it is unknown what other obstructions may be present in a pre-designated path it is important that the output offer flexibility. This is why a cost map will be used. This will provide the path generation algorithm with variability in it's final path. Actual edges of a given path will be determined as absolute boundaries. The Edge Detection block should avoid letting the robot travel even near to these boundaries at all costs, and thus variable padding from these edges will be introduced. This block is championed by Nathan Searles

4.3.2 Block Design

The following figures will highlight how the Edge Detection block will function as a part of the larger project. The black box diagram below shows the block as well as the input and output interfaces associated with this block.

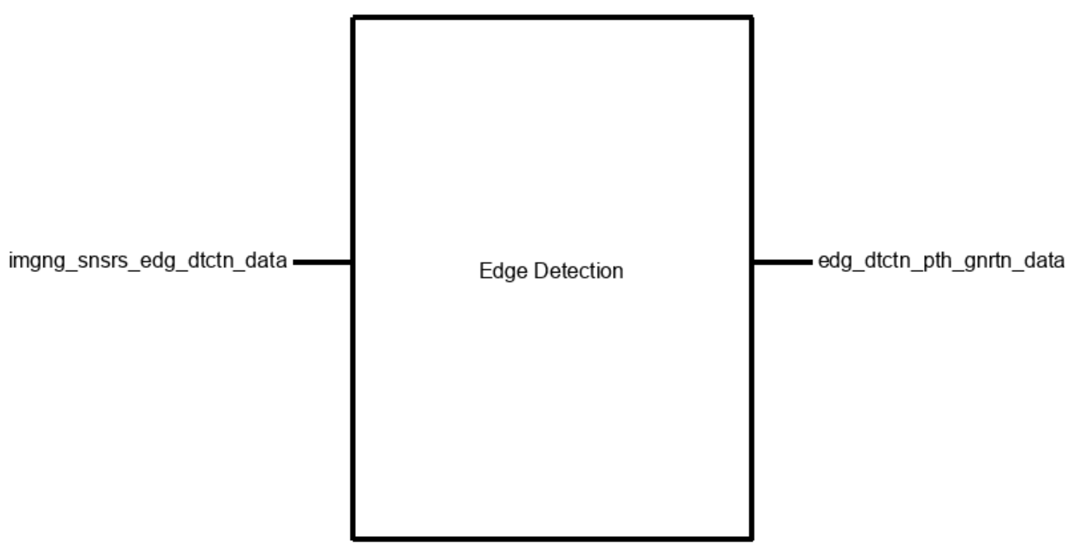

Figure 11: Edge Detection Black Box Diagram.

The flowchart in Figure 2 displays the internal behavior of the Edge Detection Blocks. Each of these elements will be described below.

Image Preprocessing: Once a frame from the camera is streamed into the Edge Detection program, there are a few processes that must happen before the edge detection can be processed. These include resizing the individual frame to maximize performance as well as converting the image to grayscale and also piping the image into the GPU for all subsequent calculations to be processed on.

Contour Detection: This is the heart of the program. Using OpenCV, there are multiple options for edge detection algorithms; the Sobel algorithm and the Canny algorithm. From preliminary testing the Canny algorithm will be better for the scope of this block and thus it will be utilized. A grayscale image will be passed into this algorithm and with a couple adjustments, a black and white image will be output, where all edges in the original image are converted to white curves.

Curve Connection: Next, any segmented curves without the area of interest must be joined together to create a cleaner image. With large edges spanning the length of the image, the Canny algorithm will output a series of segmented lines that represent a single edge, so it will be useful to join these lines together. This can be done by projecting lines from the endpoints of the existing ones and where two projects closely align in angle or intersection, the endpoints of these lines can then be assumed to be a part of the same edge.

Nearest Edge Algorithm: In practicality, the robot only needs to be aware of the edges within 2 - 4 meters from its body, so the area of interest should be determined in this step.

Cost Map Generation: As mentioned in the Block Overview, to provide the Path Generation Block with flexibility a cost map should be created from the edges. The absolute position of the edges should be treated as a wall, where the robot is able to approach within 10 - 20 centimeters of it but not touch or pass over this line. The Cost Map Generation step will also project the image into the horizontal plane for seamless interpretation in the path generation algorithm.

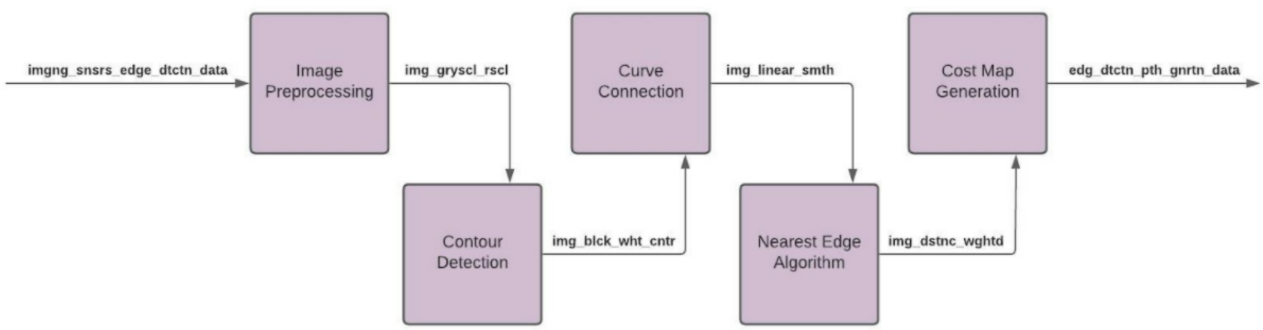

Figure 12: Edge Detection Internal Diagram.

4.3.3 Block General Validation

This block will be receiving individual frames from the Imaging module at a rate of 30 frames per second. The function of the Edge Detection block is to interpret the camera data and stream a set of curves of interest to the edge detection to the Path Detection block. This block will successfully do this by performing an edge detection algorithm on each individual frame and then subsequently isolating the curves of interest that have been determined to correspond with sidewalk edges.

The Edge Detection block will translate these sets of 2D curves into a point cloud consistent with an overhead view (XY Plane) [2]. The edges are treated as solid impermeable objects in this case and when used to create a cost map consistent with the performance of the Object Identification block, the Path Generation module will be able to interpret both of these datasets simultaneously. The output of this block will happen at frequency no lower than one sixth that of the camera input.

Since our camera will be outputting images at 60 frames per second [1], the output of the Edge Detection block will be no less than 10 frames per second. This will allow the data to be validated against the previous and following frames to ensure proper detection and minimize errors.

4.3.4 Block Interface Validation

Table 13: Interface Validation for imgng_snsrs_edg_dtctn_data: Input

Resolution: 720p	This resolution is consistent with the included camera module in the Jetson TX2 Devkit that the team is employing.	The Jetson TX2 that the team is employing for usage on the robot has a specific interface port for the camera in use.
Framerate: 60fps.	This framerate is consistent with the included camera module in the Jetson TX2 Devkit.	The Jetson TX2 that the team is employing for usage on the robot has a specific interface port for the camera in use.

Table 14: Interface Validation for edg_dtctn_pth_gnrtn_data: Output

Frame Orientation: XY Plane	The output must be in the form of an aerial view of the detected edges	The bottom of any input will have a known location relative to the robot's position. To translate the original image to an XY planar view (aerial) a simple calculation will be done by analyzing the convergence of the perceived sidewalk edges.
Data rate: 10 datasets / sec	This data rate will supply the path generation block enough datasets to avoid interrupting other processes.	After dividing the input frame rate to account for image processing delays, the Jetson TX2 will be able process and output 10 sets of edge detected and oriented data per second.
Minimum Range: 3 meters	Generated data beyond this range will become unreliable. The Path Generation block will only require this much advance notice on necessary corrections.	Data from an entire frame may be used to determine parallel convergence for mapping the frame to the XY plane. After these computations, data points that exceed this range will be masked out from the output data set.

4.3.5 Block Testing Process

1. Connect the camera module to the Raspberry Pi and connect to the Pi through SSH.
2. Apply power with the supply.
3. Launch the first test script (Range and Mapping).
4. Allow the camera to capture a still image.
5. Verify that the output image is properly mapped to a plane parallel with the Earth's surface.
6. Verify that the output edges are within the 3 meter range.
7. Launch the second test script (Data rate)
8. Holding the camera 0.5 meters above the ground at an angle of 15 degrees beneath the horizon moves the camera in a horizontal path.
9. The test script will run for 5 seconds.

10. Verify that 50 data sets were output from the module (60 fps * 5 sec / 6).

4.3.6 References and File Links

[1] "E-CAM52A_MI5640_MOD - 5MP OV5640 MIPI camera module," 5MP MIPI Camera — CSI-2 Camera Module. [Online]. Available: https://www.e-consystems.com/5MP-MIPI-camera-module.aspkey-features. [Accessed: 22-Jan-2022].

[2] "Using calibration to translate video data to the Real World," NVIDIA Developer Blog, 25-Aug-2020. [Online]. Available: https://developer.nvidia.com/blog/calibration-translate-video-data/. [Accessed: 22-Jan-2022].

4.3.7 Revision Table

Section 4.4 - Power Management Revisions	
Date	Revision
01/07/2022	Nathan Searles: Initial draft created.
01/19/2022	Nathan Searles: Revised block testing process and general validation
01/21/2022	Nathan Searles: Further revisions based on peer feedback
03/06/2022	Nathan Searles: Merged into Project Document.

4.4 Power Management System

4.4.1 Block Overview

The Power Management System block steps down and regulates the robot's 24V battery power for use in other subsystem's sensors and electronics, as well as a battery voltage monitor so that the admin can know when the robot needs to be recharged. To accomplish this, the block contains three buck converter power supplies; one is configured for outputting 5V, one for 3.3V, and the other for 19V. The battery voltage monitor utilizes a voltage divider to reduce the supply voltage to a safe level below 2.45V for it to be digitized by an ADC in the Motor Controller Driver. This block is championed by Nicholas McBee.

4.4.2 Block Design

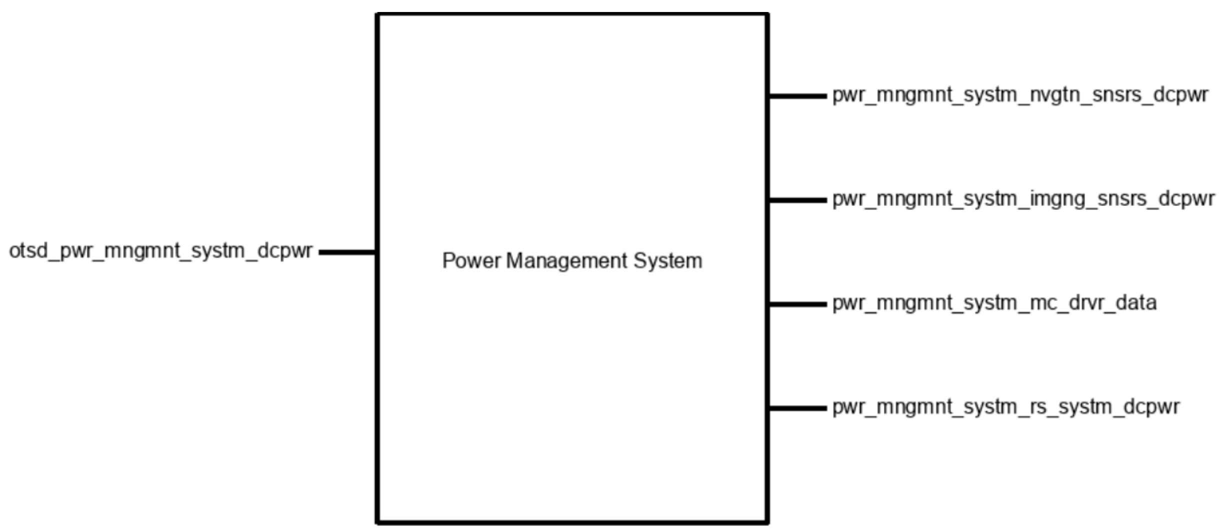

Figure 13: Black box schematic of Power Management System block.

4.4.3 Block General Validation

The voltage monitor used within the block meets the needs of the system by scaling the voltage down to a level of 2.0-2.8V, which is safe to measure with a microcontroller's internal ADC. The internal ADC is used to reduce the cost and part count of the system compared with a discrete ADC component. This monitor is also capable of achieving the 100mV accuracy range outlined in the Engineering Requirements.
The 3.3V and 5V buck converter circuitry was chosen for its relative simplicity and availability of premade modules. The controller IC is rated for the output currents necessary for the system's electronics, and a variable output voltage evaluation board is cheap and available, so its operation can be verified before the team has to commit to the creation of a PCB. The 19V buck converter was also chosen for simplicity and compatibility with other components. Its main purpose is to power the Jetson TX2 developer kit which functions as the project's main processing system. The Jetson included an AC adapter which outputs 19VDC at up to 4.74A. A step down DC-DC

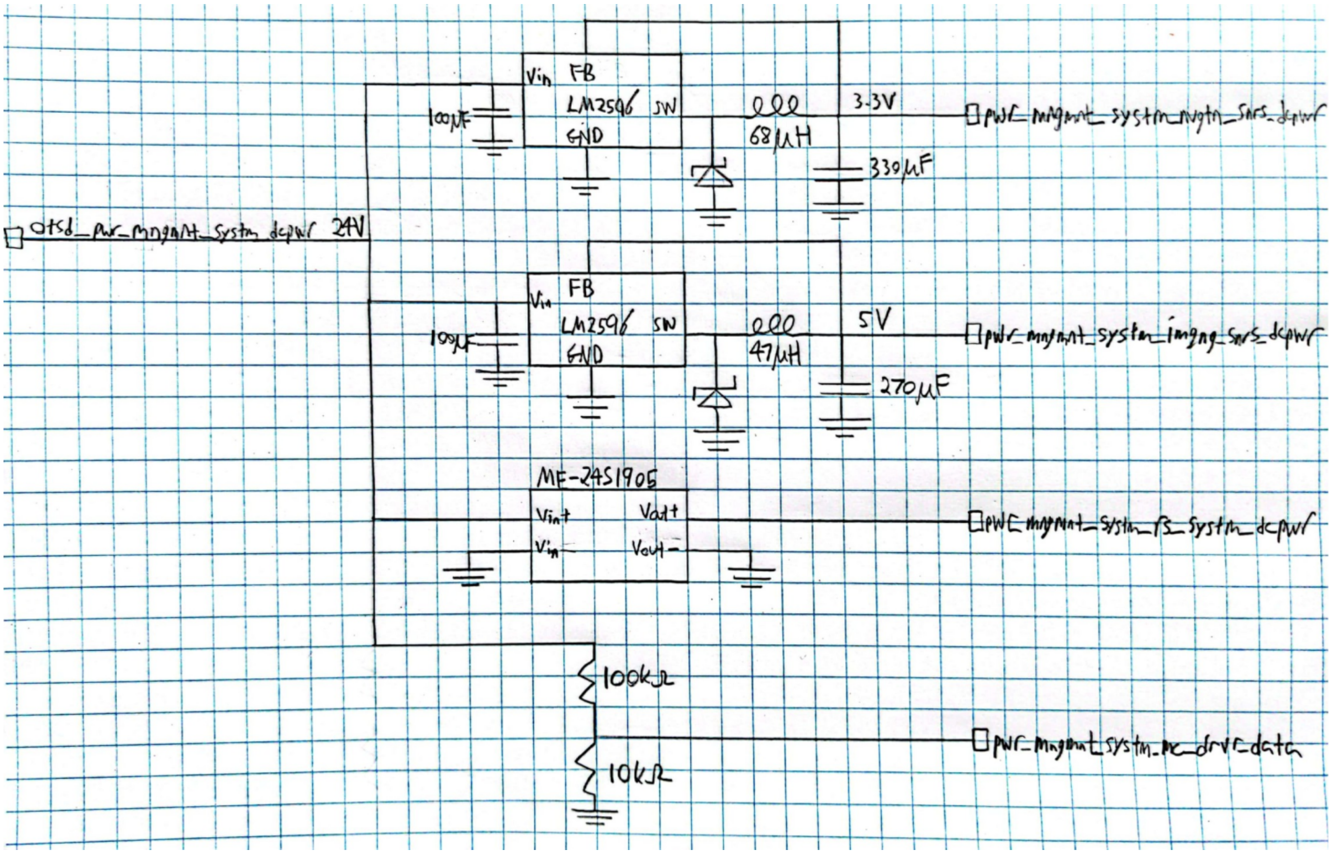

Figure 14: Electrical schematic for the Power Management System.

converter with the same output is necessary for the Jetson board to function on therobot's 24V batteries while mobile. Designing such a high power supply was determined to be difficult and time consuming, so the team elected to purchase a module with the necessary specifications. The manufacturing and shipment of these components will cause some environmental harm as discussed in the Design Impact Statement, but this risk has been considered necessary for the project to progress.

4.4.4 Block Interface Validation

Table 15: Interface Validation for otsd_pwr_mngmnt_systm_dcpwr: Input

Inominal: 1.3A	The combined expected nominal power output of the 5V and 19V supplies is 29W, which corresponds to an input current of 1.3A.	The 24V lead acid battery source is capable of supplying large amounts of current to operate the wheelchair batteries. The current requirements for the other electronics is quite small in comparison.
Ipeak: 4.4A	The combined expected peak power output for the 5V and 19V supplies is 105W, which corresponds to an input current of 4.4A.	The peak load is more considerable than it is for nominal, but this peak current is only expected to occur at startup when the motors are not moving and there is no other load on the batteries.
Vmax: 26V	The nominal lead acid battery voltage is 12V, and this value should never exceed about 12.6V when fully charged. With two batteries in series, this gives a voltage of 25.2V, which is rounded up to 26V	The onboard lead acid batteries for the system follow the described characteristics, which are explain in detail in [1].
Vmin: 22V	A lead acid battery with a voltage of 11V is significantly discharged and should be recharged before continued use to maximize its lifespan. This is the minimum battery voltage the system is therefore expected to operate at before recharging.	The onboard lead acid batteries follow the typical characteristics described and will need charging at 22V.

Table 16: Interface Validation for pwr_mngmnt_systm_mc_drvr_data: Output

Protocol: Analog voltage signal	The Motor Driver block includes an ADC that will digitize an analog voltage corresponding to battery voltage.	The voltage divider input is connected directly to the batteries, and so its output voltage will be a function of battery voltage.
Vmin: 0.15V	The Motor Driver block includes an ADC that will digitize an analog voltage corresponding to battery voltage.	At the minimum battery voltage of 22V, the output voltage signal will be 2.0V.
Vmax: 2.45V	This is the highest voltage which the Motor Driver ADC can reliably read from.	At the maximum battery voltage of 26V, the output voltage signal will be 2.36V.

Table 17: Interface Validation for pwr_mngmnt_systm_imgng_snsrs_dcpwr: Output

Inominal: 2A	The sum of the nominal current draw of all the 5V sensors and electronics is expected to be below 2A.	The LM2596 is capable of a nominal output current of 3A (7.6 Electrical Characteristics, pg6).
Ipeak: 3A	The sum of all the peak current draws of all 5V sensors and electronics is not expected to exceed 3A.	The LM2596's is capable of up to 3A of constant current output, and at least 3.6A of peak current (7.6 Electrical Characteristics, pg6).
Vmax: 5.2V	The supply voltage for the sensors cannot exceed 5.5V; 5.2V was chosen to provide safety margin.	LM2596 5V output mode will not exceed 5.2V (7.6 Electrical Characteristics, pg6).
Vmin: 4.8V	LiDAR sensor unit requires at least 4.8V to function properly.	LM2596 5V output mode will not drop below 4.8V (7.6 Electrical Characteristics, pg6).

4.4.5 Block Testing Process

Voltage Monitor Test Procedure:

1. Use a lab bench power supply to provide input otsd_pwr_mngmnt_systm_dcpwr with 22V.
2. Use a voltmeter to verify pwr_mngmnt_systm_mc_drvr_data is within 0.15V-2.45V.
3. Set the input voltage to 26V.
4. Use the voltmeter to verify pwr_mngmnt_systm_mc_drvr_data is still within 0.15V-2.45V

Table 18: Interface Validation for pwr _mngmnt_systm_rs_systm_dcpwr: Output

Inominal: 2.5A	Little information on the Jetson TX2's power requirements is available; this is a rough estimate of its typical current draw at 19V.	ME-24S1905 power converter has a nominal current output up to 5A (Uxcell product page, Specifications).
Ipeak: 4.72A	This is the current rating of the Jetson's supplied AC adapter, which presumably accounts for peak current draw + safety margin.	ME-24S1905 power converter's nominal current exceeds current rating of original supply.
Vmax: 19.4V	Conservative estimate of Jetson TX2 supply voltage range.	ME-24S1905 power converter has voltage regulation below 1%, corresponding to an output shift of 190mV (Uxcell product page, Specifications).
Vmin: 18.8V	Conservative estimate of Jetson TX2 supply voltage range.	ME-24S1905 power converter has voltage regulation below 1%, corresponding to an output shift of 190mV (ME-24S1905 product page, Specifications).

Table 19: Interface Validation for pwr_mngmnt_systm_nvgtn_snsrs_dcpwr: Output

Inominal: 50mA	This is the expected current draw of all navigation sensors using 3.3V.	The LM2596 is capable of a nominal output current of 3A (7.6 Electrical Characteristics, pg6).
Ipeak: 100mA	This is the peak expected draw of all navigation sensors using 3.3V.	The LM2596's is capable of up to 3A of constant current output, and at least 3.6A of peak current (7.6 Electrical Characteristics, pg6).
Vmax: 3.6V	This is the maximum recommended supply voltage of the navigation sensors.	LM2596 3.3V output mode will not exceed 3.432V (7.6 Electrical Characteristics, pg6).
Vmin: 2.7V	This is the minimum recommended supply voltage of the navigation sensors.	LM2596 3.3V output mode will not drop below 3.168V (7.6 Electrical Characteristics, pg6).

Power Source Voltage Test Procedure:

1. Use a lab bench power supply to provide input otsd_pwr_mngmnt_systm_dcpwr with 22V.
2. Use a voltmeter to verify that pwr_mngmnt_systm_nvgtn_snsrs_dcpwr is within 2.7-3.6V
3. Use a voltmeter to verify that pwr_mngmnt_systm_imgng_snsrs_dcpwr is within 4.8-5.2V.
4. Use a voltmeter to verify that pwr_mngmnt_systm_rs_systm_dcpwr is within 18.8-19.2V.
5. Repeat steps 2 through 4 with otsd_pwr_mngmnt_systm_dcpwr set to 26V.

Power Source Current Test Procedure:

1. Connect a 50mA load to pwr_mngmnt_systm_nvgtn_snsrs_dcpwr.
2. Connect a 2A load to pwr_mngmnt_systm_imgng_snsrs_dcpwr.
3. Connect a 2.5A load to pwr_mngmnt_systm_rs_systm_dcpwr.
4. Supply otsd_pwr_mngmnt_systm_dcpwr with 22V.
5. Let system run for at least 30 seconds. Verify all current loads are satisfied.
6. Repeat steps 1 through 5 with otsd_pwr_mngmnt_systm_dcpwr set to 26V.

Power Source Peak Current Test Procedure:

1. Connect a 100mA load to pwr_mngmnt_systm_nvgtn_snsrs_dcpwr.
2. Connect a 3A load to pwr_mngmnt_systm_imgng_snsrs_dcpwr.
3. Connect a 4.72A load to pwr_mngmnt_systm_rs_systm_dcpwr.
4. Supply otsd_pwr_mngmnt_systm_dcpwr with 22V.
5. Let system run for at least 3 seconds. Verify all current loads are satisfied.
6. Repeat steps 1 through 5 with otsd_pwr_mngmnt_systm_dcpwr set to 26V.

4.4.6 References and File Links

References

[1] R. Perez, "Batteries lead-acid battery state of charge vs. voltage," 1993. [Online]. Available: . [Accessed: 07-Jan-2022].

Files

- LM2596 datasheet: https://www.ti.com/lit/ds/symlink/lm2596.pdf?ts=1641572364177
- Premade LM2596 module: https://www.amazon.com/Valefod-Efficiency-Voltage-Regulator-Converter/dp/B076H3XHXP
- Uxcell (Eccanixity) ME-24S1905 power module: https://www.amazon.com/uxcell-Converter-Regulator-Waterproof-Transformer/dp/B01H97ETVM

4.4.7 Revision Table

Section 4.4 - Power Management Revisions	
Date	Revision
01/08/2022	Nick McBee: Initial draft created.
01/20/2022	Nick McBee: Modified General Validation.
01/21/2022	Nick McBee: Revised test procedure.
03/06/2022	Nick McBee: Merged into Project Document.

4.5 Web Controller

4.5.1 Block Overview

The web controller block will be the web application that controls the system. The web controller will send data to the ROS system for the lockbox, approval, destination, and stop features. The lockbox will have the ability to be unlocked. The approval will send a code to the ROS system to tell the ROS system it is ok to proceed. The destination will send a code that tells the ROS system to go to a predetermined waypoint. The stop will make send a code to stop the ROS system. This block is championed by Tyrone Stagner.

4.5.2 Block Design

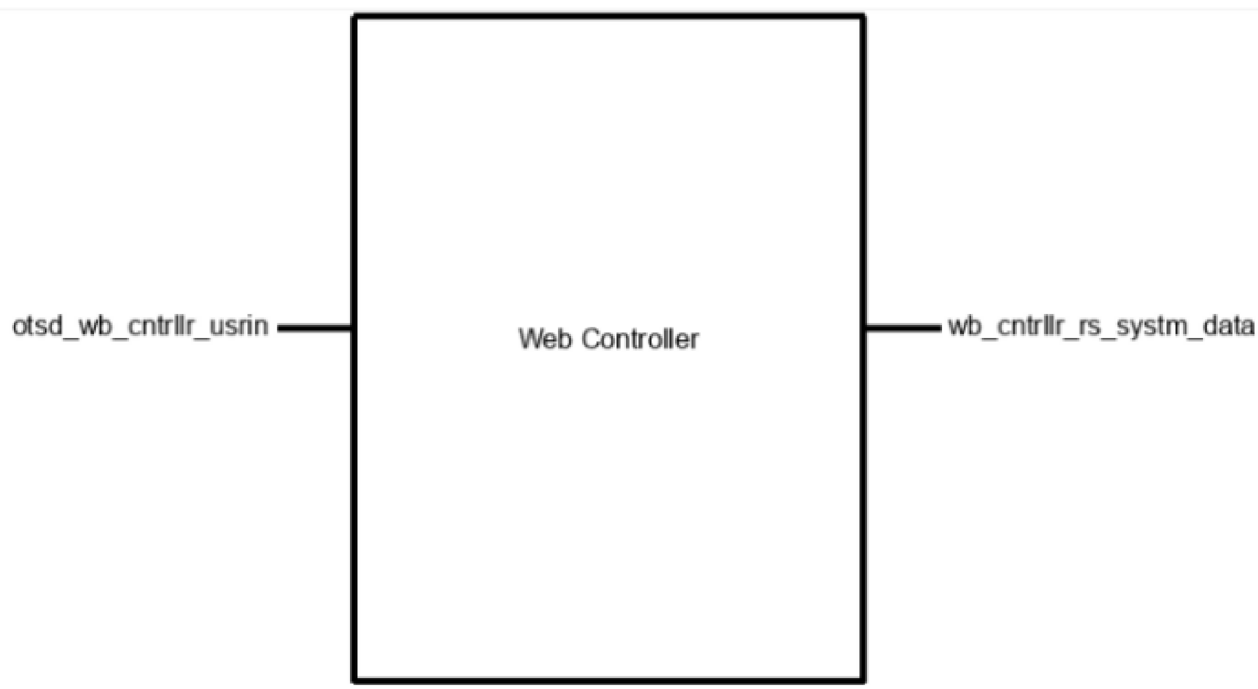

Figure 15: Black box schematic of Web Controller Block.

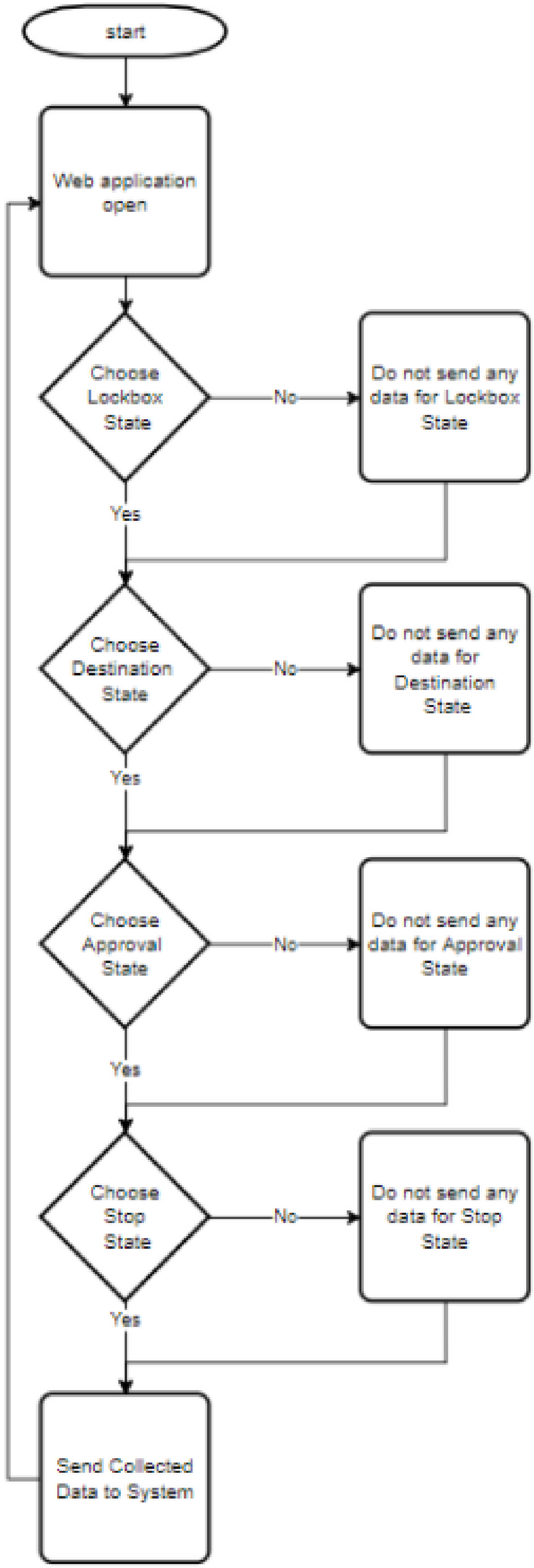

Figure 16: Design flowchart for the Web Controller Block.

4.5.3 Block General Validation

Robot Operating System which is known as ROS is used in conjunction with a module rosbridge_suite for the web application to meet the block's needs for data communication to the system. The rosbridge_suite module helps the website to send ROS.Message which will be destination, stop, lockbox, and approval data [1]. Utilizing the rosbridge_suite module will provide a WebSocket interface to ROS from the web application. Using a WebSocket interface will help lower latency by keeping a persistent connection to the ROS system. After a connection is made it will communicate with ROS topics on the system to transfer data between the system and web application.

4.5.4 Block Interface Validation

Table 20: Interface Validation for otsd_wb_cntrllr_usrin: Input

Other: User can set Lockbox state	The user needs to be able to unlock the system.	The code for this block will allow the user to choose from a dropdown menu the state of the Lockbox.
Other: User can set Destination state.	The user needs to be able to choose a destination.	The code for this block will allow the user to choose from a dropdown menu predefined numbers for the destination state.
Other: User can set Approval state.	The user needs to be able to approve the system to proceed.	The code for this block will allow the user to choose from a dropdown menu the approval state.
Other: User can set Stop state.	The user needs to be able to stop the system.	The code for this block will allow the user to choose from a dropdown menu the stop state.

Table 21: Interface Validation for wb_cntrllr_rs_systm_data: Output

Messages: Destination	The web application needs to send a destination message back to the system once the user provides where the ROS system need it to go.	The code for this block will send a destination using ROSLIB.Message to the ROS system.
Message: Stop	The web application needs to send a stop message back to the ROS system once the user provides that command.	The code for this block will send a stop ROSLIB.Message to the ROS system.
Message: Lockbox	The web application needs to send a lockbox message back to the ROS system once the user provides that command.	The code for this block will send a lockbox ROSLIB.Message to the ROS system.
Message: Approval	The web application needs to send an approval message back to the ROS system once the user provides that command.	The code for this block will send an approval ROSLIB.Message to the ROS system.

4.5.5 Block Testing Process

1. Open the web application on a computer.
2. Enter a username and user password.
3. Click on button that reads “Sign in”.
4. Click on a drop-down box and select a number from destination box.
5. Click on a drop-down box and select yes from the stop box.
6. Click on a drop-down box and select yes from the approval box.
7. Click on a drop-down box and select open from the lockbox box.
8. User will click on a submit button.
9. On the Raspberry Pi, open the ROS topics on the corresponding to the destination, stop, approval, and lockbox to view activity.

4.5.6 References and File Links

[1] “rosbridge_suite?,” GitHub, 04-Apr-2019. [Online]. Available: https://github.com/RobotWebTools/rosbridge_suite. [Accessed: 18-Feb-2022].

4.5.7 Revision Table

Section 4.4 - Power Management Revisions	
Date	Revision
01/30/2022	Tyrone Stagner: created block 2.
02/01/2022	Tyrone Stagner: made some revisions to the file in section 4.13, 4.14, and 4.15
02/12/2022	Tyrone researched ROS2-Bridge more and made revisions to 4.13
02/18/2022	Tyrone made some revisions to the file in section 4.13, 4.14, and 4.15
02/19/2022	Tyrone made some revisions to the file in section 4.15 , from the feedback given in the block 2 reviews.
03/06/2022	Tyrone Stagner: Merged into Project Document

4.6 Lock Box

4.6.1 Block Overview

The Lock Box block is a mechanical and enclosure block which will contain a physical box with an electronically manipulated lock. It will receive a control signal from the microcontroller which will disengage the locking mechanism. Upon receiving the signal, an indicator light will show that the box is ready to be opened. When the lid is lifted, a signal will be sent back to the microcontroller to indicate the lid is still open and will turn off once the lid is closed. The block champion for this block is Drew Gehrke.

4.6.2 Block Design

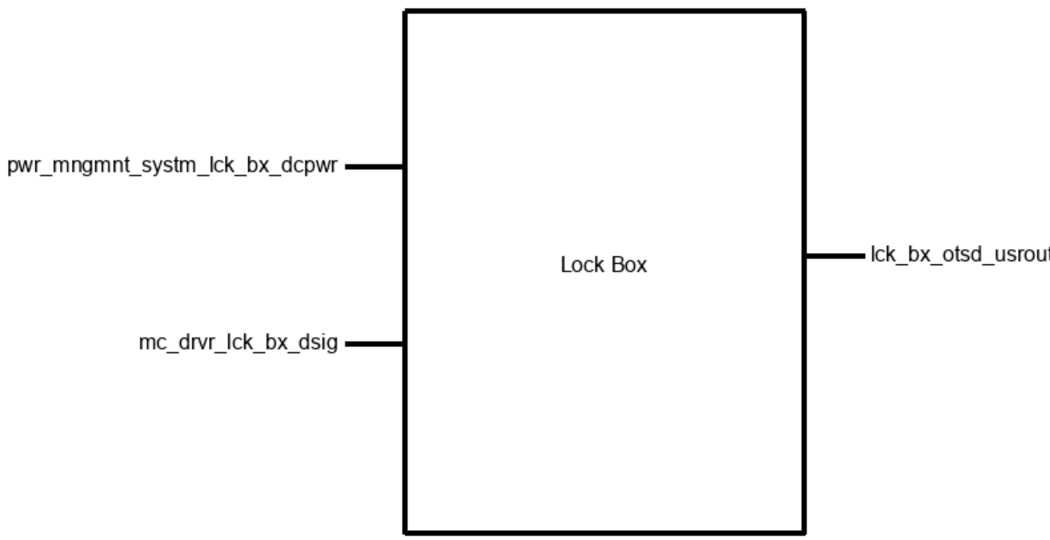

Figure 17: Black box schematic of Lock Box block.

4.6.3 Block General Validation

This block will allow for safe travel of any sort of payload or package the robot will transport. The components being used for the locking mechanism will be low cost as they are all found on Amazon. The box itself will be an ice box which will be customized to house the components necessary for the locking mechanism and the other hardware to meet the interfaces. The locking mechanism requires 12V in order to be powered. To do this, the microcontroller will need to control a relay. This relay will be able to handle the higher current and potential circuit needed for the lock itself. Other indicators will be used to allow for the user to know when the lid is opened still so they can fully close it.

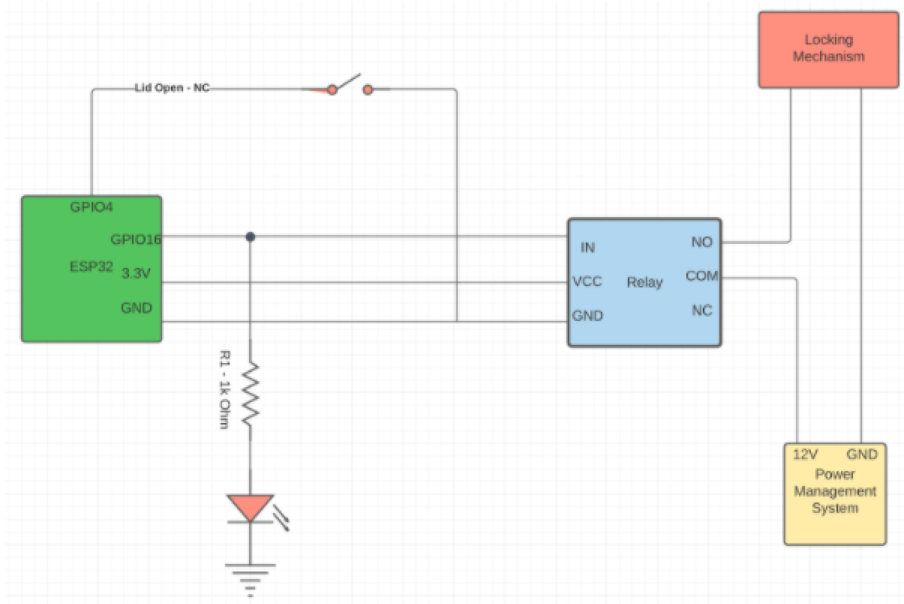

Figure 18: Electrical schematic for the Lock Box block.

4.6.4 Block Interface Validation

Table 22: Interface Validation for mc_drvr_lck_bx_dsig - Relay control signal input

Logic Level: Active high	The relay will enable when a logic HIGH signal is applied and will disengage the locking mechanism.	A GPIO pin on the ESP32 will provide the necessary digital signal to trigger the relay.
Other: dsig returned	A switch will be triggered to indicate the lid has been opened and will be returned to the microcontroller.	Connecting the switch in a normally closed configuration will allow for the functionality needed. So when the lid is opened, the circuit will close, driving the GPIO pin low as an indication.
Vnominal: 3.3V	The locking mechanism is expecting 12V but will be controlled via a relay which will take in 3.3V.	The GPIO pin on an ESP32 is capable of outputting 3.3V nominal, which is enough to trigger the relay.

Table 23: Interface Validation for pwr_mngmnt_systm_lck_bx_dcpwr - 12V input

Vmax: 12.2V	The rated voltage of the locking mechanism is 12V. It can most likely handle 2% more than that nominal voltage.	The power management system will be capable of providing the required voltage of 12V, and the lock will operate if it were to go up slightly.
Vnominal: 12V	The locking mechanism is rated to operate at 12V.	According to the Amazon description, the control signal must provide 12V in order to unlock the system.
Vmin: 11.8V	The rated voltage of the locking mechanism is 12V. It can most likely handle 2% less of that nominal voltage.	The power management system will be capable of providing the required voltage of 12V, and the lock will operate at slightly less.
Ipeak: 2.1A	The locking mechanism is rated to pull 2A nominally. The lock is capable of pulling slightly more current and thus must be able to handle this slight difference.	The PMS system is able to output 2A nominally and can handle slightly more.
Inominal: 2A	The locking mechanism is rated to pull 2A of current.	The PMS system will be able to very briefly (0.2s for the lock) allow the current draw to unlock the lock.

Table 24: Interface Validation for lck_bx_otsd_usrout - Box Indicators output

Type: Light	Indicator light will flash on when the lock is disengaged and the lid can be opened.	An LED will be wired to the GPIO pin and will flash on when the signal goes high, indicating the lock has disengaged.
Type: Switch	Switch is used to indicate the lid being opened or closed.	The switch will be connected in a normally closed configuration and will indicate when the lid is open.
Type: Lid	The lid is the way to access the package which the robot is delivering.	The robot requires a way to securely hold a package and the lid will the only way to access the package.

4.6.5 Block Testing Process

1. Start with lid closed and lock engaged
2. Trigger the unlock signal (apply 3.3V to the relay to close relay)
   - Check for light turned on to indicate unlocked
   - Open the lid and check the switch has disengaged
   - Look for indicator to see the lid is opened
3. Close lid and ensure switch has engaged
   - Check that the lid open indicator is gone
   - Ensure indicator light is off to show lock is engaged

4.6.6 References and File Links

References

[1] "ESP32 pinout reference: Which GPIO pins should you use?," Random Nerd Tutorials. [Online]. Available: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/. [Accessed: 18-Feb-2022].

Files

- Locking Mechanism
- Relay Module
- Ice Box

4.6.7 Revision Table

Section 4.6 - Lock Box Revisions	
Date	Revision
02/02/2022	Drew Gehrke: Initial document created, description added, black box figure added
02/03/2022	Drew Gehrke: Added content to multiple sections
02/16/2022	Drew Gehrke: Modified content for input signal, added links to parts
02/17/2022	Drew Gehrke: Added more content to each of the sections
02/18/2022	Drew Gehrke: Final touches added to interface definitions, pictures, and validation paragraphs
03/05/2022	Drew Gehrke: Added all sections to project document

4.7 Path Generation

4.7.1 Block Overview

The Path Generation block is a code block which will determine the path of traversal the robot will take. Based on a set of way points, the robot will navigate to the specified end goal, adjusting for obstacles when they have been identified. Path planning between way points and obstacle avoidance will be handled by the algorithms included in the ROS navigation stack. The block champion is Nicholas McBee

4.7.2 Block Design

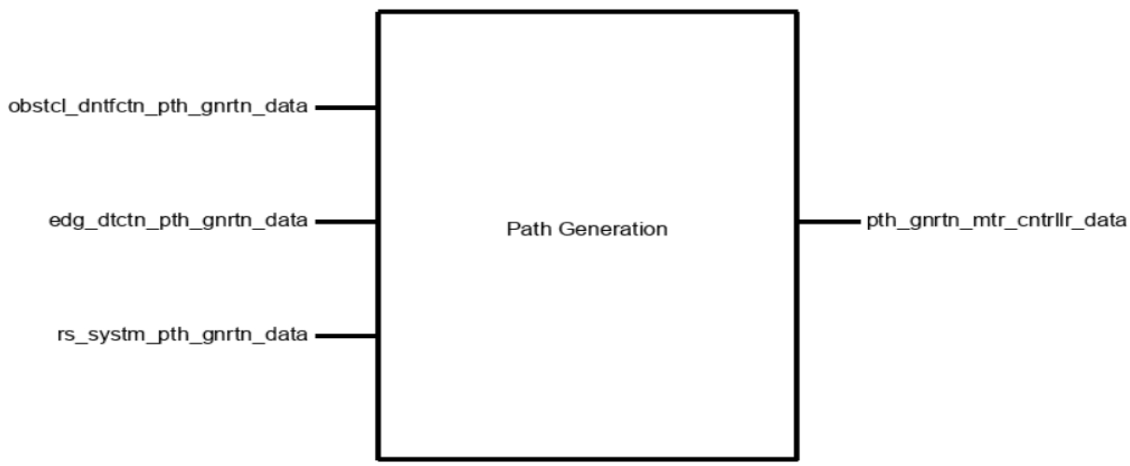

Figure 19: Black box schematic of Path Generation block.

The Path Generation block acts as an interface between the high level system controls, the imaging sensor outputs, and the ROS navigation algorithms. Therefore, its main purpose is to provide the ROS system with the sensor data and waypoint navigation goals as described below.

Peseudocode:

1. START: Wait for rs_systm_pth_gnrtn_data to get DELIVERY_DEST.

2. When DELIVERY_DEST received, send navigation system obstcl_dntfctn_pth_gnrtn_data and edg_dtctn_pth_gnrtn_data as sensor inputs. Set pth_gnrtn_mtr_cntrllr_data as motor output.

3. for i = num waypoints in DELIVERY_DEST:
    (a) NAV_GOAL = DELIVERY_DEST_Waypoint[i]
    (b) while[!nav_to_waypoint_done()]
    (c) i++

4. Navigation to destination complete. Repeat process at step one.

4.7.3 Block General Validation

Each delivery destination route is split into intermediate waypoints to simplify the processing required and provide direction to the autonomous path generation. Specifically, the waypoints can be placed at regular intervals on sidewalks so that the path planner algorithms will not inadvertently choose to generate paths along roads, and requires it to properly cross intersections. This also guarantees more consistency in routes, making testing and debugging easier.

4.7.4 Block Interface Validation

Table 25: Interface Validation for obstcl_dntfctn_pth_gnrtn$_data : Input$

Other: Minimum Range: 3 meters	This is roughly the minimum scanning distance needed for the robot to see upcoming obstacles and safely react.	Experience from the previous team suggests this range is sufficient for navigating around obstacles.
Other: Image mapping: XY plane, internal ROS coordinate system	The robot only needs to navigate along a 2D plane, and an internal coordinate system simplifies the interaction between all sensors and navigation code.	This is the format used by the ROS navigation system, and has already been confirmed through simulation.
Protocol: 2D point cloud of obstacle locations	Standard method of obtaining and representing wall and obstacle locations in ROS systems.	ROS navigation stack accepts the input of point clouds from sensors [1].

4.7.5 Block Testing Process

1. Load Gazebo simulation software and Rviz visualization tools.
2. Launch ROS navigation system and interface code. Provide simulated obstcl_dntfctn_pth_gnrtn_data and edg_dtctn_pth_gnrtn_data to map simulated environment. Verify that the test data is within interface property expectations.
3. Manually send destination value to rs_systm_pth_gnrtn_data.
4. Observe velocity values being sent to pth_gnrtn_mtr_cntrllr_data and the simulated robot navigating towards the destination. The block is performing as expected when the simulated robot navigates to the sequence of waypoints specified in the interface code.

Table 26: Interface Validation for edg_dtctn_pth_gnrtn_data : Input

Datarate: Minimum: 10 images / second	This is considered a fair compromise value between required performance and system processing resources throughput.	10 updates/second of the sidewalk edge location should be more than fast enough to keep pace with the robot's max speed of 0.5m/s.
Other: Image Mapping: XY Plane	The robot only needs to navigate along a 2D plane, and an internal coordinate system simplifies the interaction between all sensors and navigation code.	This is the format used by the ROS navigation system, and has already been confirmed through simulation.
Other: Minimum Range: 3 meters	This is roughly the minimum scanning distance needed for the robot to see upcoming obstacles and safely react.	Experience from the previous team suggests this range is sufficient for navigating around obstacles.

Table 27: Interface Validation for pth_gnrtn_mtr_cntrllr_data : Output

Messages: Floating point velocity values	Floating point values provide ample precision and range needed to represent	ROS Geometry Twist message type uses floating point data types [2].
Other: Contains X, Y and rotational values	Velocity needs to be expressed as its X and Y components, and the angle the robot is facing also needs to be described.	ROS Geometry Twist message type stores linear velocity components and angular position [2].
Protocol: ROS Topic cmd_vel message	This is the standard message format expected by the motor controller.	ROS navigation system uses the cmd_vel topic for motor control outputs [3].

4.7.6 References and File Links

References

[1] "Setting up sensors," Setting Up Sensors - Navigation 2 1.0.0 documentation. [Online]. Available: `https://navigation.ros.org/setup_guides/sensors/setup_sensors.html`. [Accessed: 05-Feb-2022].

[2] "Understanding Ros 2 topics," Understanding ROS 2 topics - ROS 2 Documentation: Galactic documentation. [Online]. Available: `https://docs.ros.org/en/galactic/Tutorials/Topics/Understanding-ROS2-Topics.html?highlight=velocity+topic#ros2-interface-show`. [Accessed: 05-Feb-2022].

Table 28: Interface Validation for rs_systm_pth_gnrtn_data : Input

Messages: Integer specifying destination point	The possible delivery destination locations is a finite list of spots around buildings on campus; each spot can simply be enumerated for referencing.	DELIVERY_DEST variable in pseudocode stores the desired destination point
Other: Will only send when current routing is complete	It does not make sense to start a new delivery while another one is still in progress, so the main code is not expected to give a new delivery route until the current one is complete.	Code only gets new delivery destination after arriving at the current one.
Protocol: ROS topic message	Internal communication between ROS nodes is mostly commonly accomplished through topic messages.	Code gets the new delivery location from ROS message.

[3] "Nav2," Nav2 - Navigation 2 1.0.0 documentation. [Online]. Available: https://navigation.ros.org/index.html?highlight=motor. [Accessed: 05-Feb-2022].

4.7.7 Revision Table

Section 4.7 - Path Generation Revisions	
Date	Revision
02/04/2022	Nick McBee: Initial draft created.
02/18/2022	Nick McBee: Added additional details to Block Testing Plan.
03/06/2022	Nick McBee: Merged with Project Document.

4.8 Obstacle Identification

4.8.1 Block Overview

The Object Identification block will serve a precursor to the Path Generation block in the Autonomous Package Delivery Robot. LiDAR is the primary hardware component associated with this block. This block will be able to determine the bounds of travel by using the onboard LiDAR sensor to create a point cloud based on physical objects in the vicinity of the robot. The output interface will be a cost map that the robot will interpret for its path generation algorithms. The LiDAR sensor will be mounted on the robot and analyze a planar slice of the world around the robot. This planar slice will be level to the ground and provide information about obstacles within line-of-sight of the LiDAR module. For redundancy the data collected from the LiDAR will be cross referenced with the visual data collected by the camera module in the Path Generation block. The program will be written with Python and run in real-time within the ROS Navigation stack. This block is championed by Nathan Searles.

4.8.2 Block Design

The following figure will highlight how the Object Identification block will function as a part of the larger project. The black box diagram in the figure below shows the block as well as the input and output interfaces associated with this block. In the next figure, the primary functions of this

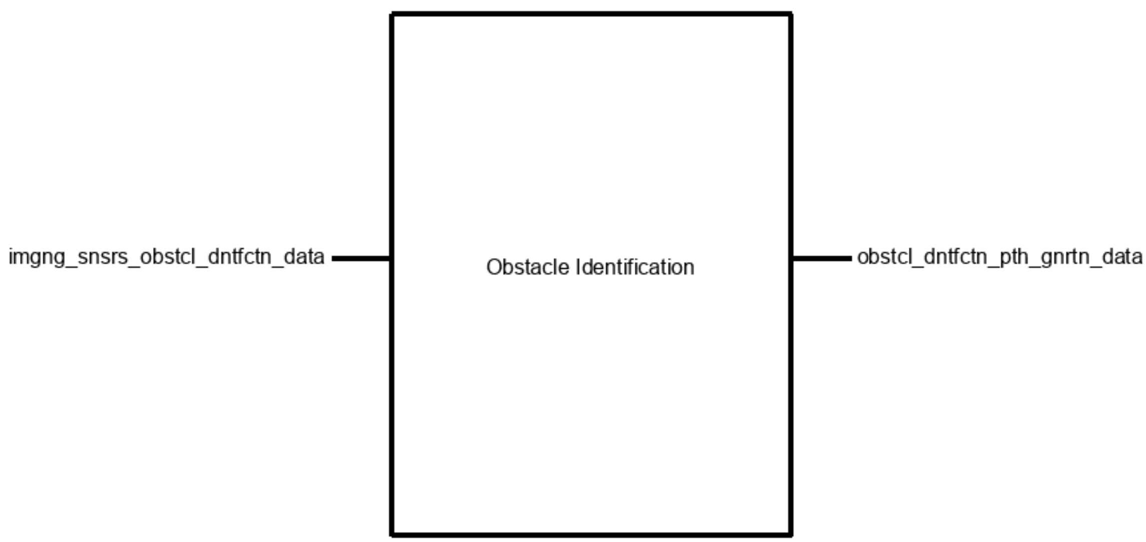

Figure 20: Black box schematic of Obstacle Identification Block.

block are described. The goal is to provide the robot with information to keep it from colliding with objects in its surroundings. The LiDAR will provide a set of points to the Python program where the interpretation will occur. Filtering out noisy data will be important for creating the curves bordering objects.

When creating the curves, there are a few important considerations. Given the height of the LiDAR and its limited ability to only scan a planar slice of its environment, some pieces of data will need to be projected to a model of potential objects. If the LiDAR were to scan 4 discrete "packets" of points that could predictably form a connected shape, that shape must be used for the cost map creation. Examples of this would include chair legs, mail dropoff boxes, and tables.

After object curves have been created, the cost map must be generated. This will be done internally with the ROS packages for the LiDAR [1]. A cost map depth of 15cm has been determined to be adequate for this application. A larger range could prevent usable paths from being generated on a standard 4ft sidewalk, while a smaller range could lead to collisions.

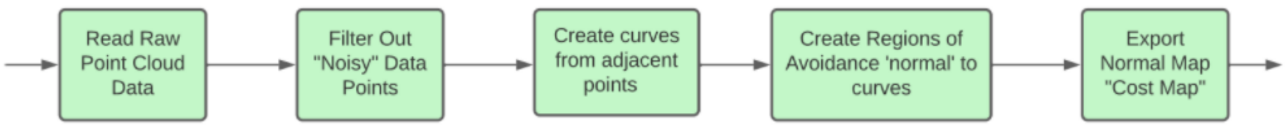

Figure 21: Object Identification Design.

4.8.3 Block General Validation

This block will receive a set of points from the LiDAR sensor. These individual points will be compiled into a single data set known as a point cloud. For this application, the point cloud will not take the form of a cloud, but rather a surface view of the robot's surroundings.

LiDAR Orientation: This aspect of the design required much deliberation from the team to determine the best implementation for the LiDAR module. The LiDAR module that will be present on the robot does not offer variable angles, and thus the LiDAR can only detect a planar slice of 3D points in its point cloud. In order to maximize the usable range of this data, the LiDAR will be mounted level to the base of the robot. This will return a 2D point cloud around the robot

LiDAR Height: Since the LiDAR will only collect data on the XY Panar slice of its surroundings, LiDAR height is an important variable that will determine what information can be accessed. The minimum mounting height available on the robot is approximately 30cm, as everything beneath that is chassis. Obstacles come in a variety of forms, but the critical points of interest are pedestrians, large animals, and walls. By mounting the LiDAR close to this minimum mounting height, the robot will have an awareness of most obstacles that exist at its height. Objects out of range of the robot's height can be disregarded as they will not cause a collision. The major concern is objects that are in the robots path, yet lower than the LiDAR's perspective. These will be handled in combination with the visual imaging sensors (cameras).

Data Integrity: Given the limited scope of data that will be detected by the LiDAR element, noise will present itself obviously and can be disregarded. This "noise" will be in the form of lonesome points in the cloud. Any significant objects will have multiple points associated with them, and these points can be analyzed across frames to ensure they are not noise.

Software Implementation: The analysis of this data is equally important to the collection of it. The SLAMTEC RPLIDAR S1 that will be used in this project has support drivers available [1] for ROS2.

Data Interpretation: The data being read into this block should be relatively clean. The main function of this block will be the generation of a cost map to be utilized by the path generation block. This primarily involves analyzing the point cloud and registering which points correlate to an object, using these points to create a curve, and outputting a set of ranges from each object, that the robot can feasibly travel. Touching the object should be interpreted as forbidden, while maintaining a distance of ¿15cm should be preferred, but flexible depending on other inputs.

4.8.4 Block Interface Validation

Table 29: Interface Validation for imgng_snsrs_obstcl_dntfctn_data : Input

Rotation Frequency: 10Hz	The cost map must be frequently updated during operation and travel of the robot.	10Hz is the median range for the LiDAR module that is used in this project. The range spans 5-15Hz. Using the middle of that range will improve stability.
Minimum Range: 10m	The data must be of adequate range for predictive modeling in the path generation block.	The LiDAR module has a range of 40m for pure white surfaces and 10m for black surfaces.
Minimum Angular Resolution: 2 points / degree	The collected data must contain enough useful points to extrapolate curves.	The LiDAR samples at 9.2kHz in 10Hz rotation mode: producing 1 pt / 0.391°

Table 30: Interface Validation for imgng_snsrs_obstcl_dntfctn_data : Input

Data rate: 5Hz min	The cost map must be frequently updated during operation and travel of the robot.	The full rotation sampling frequency for the LiDAR will be 10Hz. An output frequency of 5Hz accounts for lag in cost map generation
Minimum Radius: 10m	The data must be of adequate range for predictive modeling in the path generation block.	The LiDAR module has a range of 40m for pure white surfaces and 10m for black surfaces.
Field of View: 180°	The robot will reverse in manually guided emergency scenarios. For autonomous travel, the robot will travel forward and perform stationary turns. The robot must be aware of its surroundings before traversing.	The LiDAR is capable of 360° data capture. The limiting variable is the mounting position on the robot. Being a primary sensor, the LiDAR will be positioned to provide 180° minimum viewing angle.

4.8.5 Block Testing Process

1. Connect the LiDAR to the Raspberry Pi 4 with USB.

2. SSH into the Raspberry Pi and open the ROS Vizualization GUI.
3. Place the LiDAR 10 meters away from a 10cm wide object in low lighting conditions.
4. Observing the ROS Gui, verify that the object has been detected with a minimum of 2 data points. This verifies Angular Resolution and Radius.
5. Slide the LiDAR across a table-top surface for 10 seconds, traversing more than 2 feet from its original location.
6. Using the ROS Vizualization GUI, verify that the cost map updates with a frequency greater than 5Hz (50 frames minimum).
7. Placing the LiDAR in a stationary position on a table-top, place a flat obstruction object (notebook, piece of paper, etc.) flush against 1 edge of the LiDAR enclosure.
8. Verify that more than 180° of cost map data is being returned.

4.8.6 References and File Links

[1] A. H, "Allenh1/rplidar_ros," GitHub, 27-May-2021. [Online]. Available: https://github.com/allenh1/rplidar_ros. [Accessed: 05-Feb-2022].

[2] T. Huang, "RPLIDAR S1 portable TOF Laser Range Scanner parameters," SLAMTEC. [Online]. Available: https://www.slamtec.com/en/Lidar/S1Spec. [Accessed: 05-Feb-2022].

4.8.7 Revision Table

Section 4.8 - Obstacle Identification Revisions	
Date	Revision
02/04/2022	Nathan Searles: Initial Document creation
02/15/2022	Nathan Searles: Added figures, design revisions according to feedback
02/18/2022	Nathan Searles: Interface property revisions and testing process revisions
03/06/2022	Nathan Searles: Merged into project document

4.9 Display Data

4.9.1 Block Overview

The Display Data block will be the web application that displays system diagnostics. Some of the system diagnostics include the battery status, position, and error codes. The position will be displayed on a map. Battery status will be displayed showing numbers, and error codes will be displayed as a number and brief description of the error.

4.9.2 Block Design

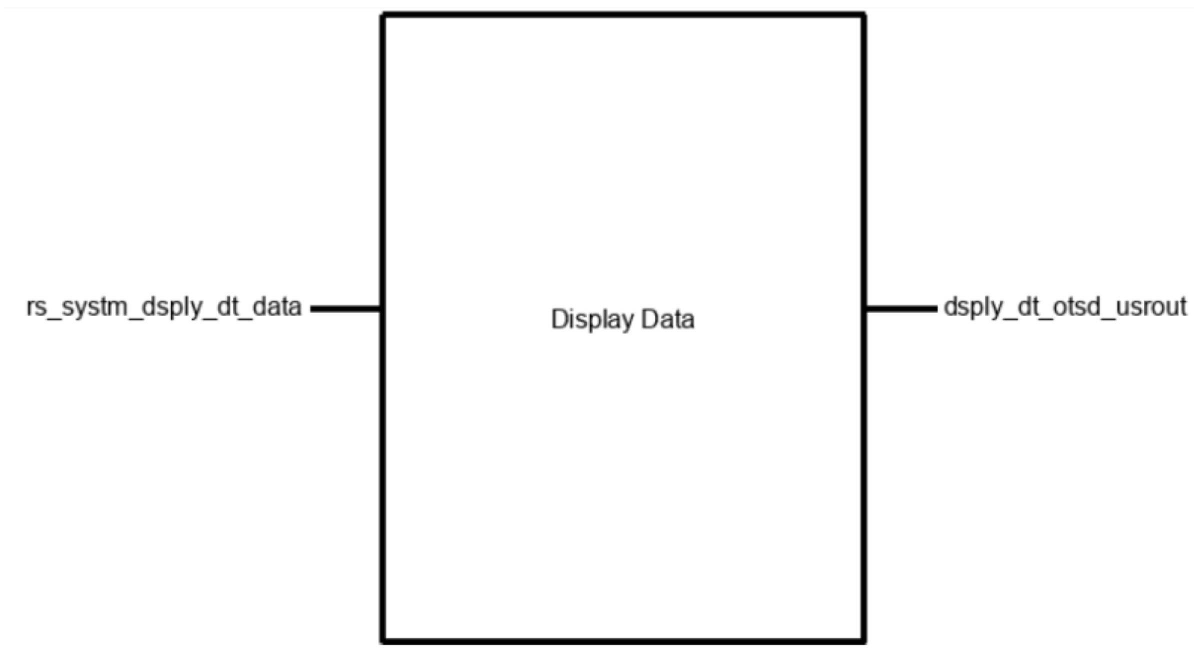

Figure 22: High level diagram of display data block.

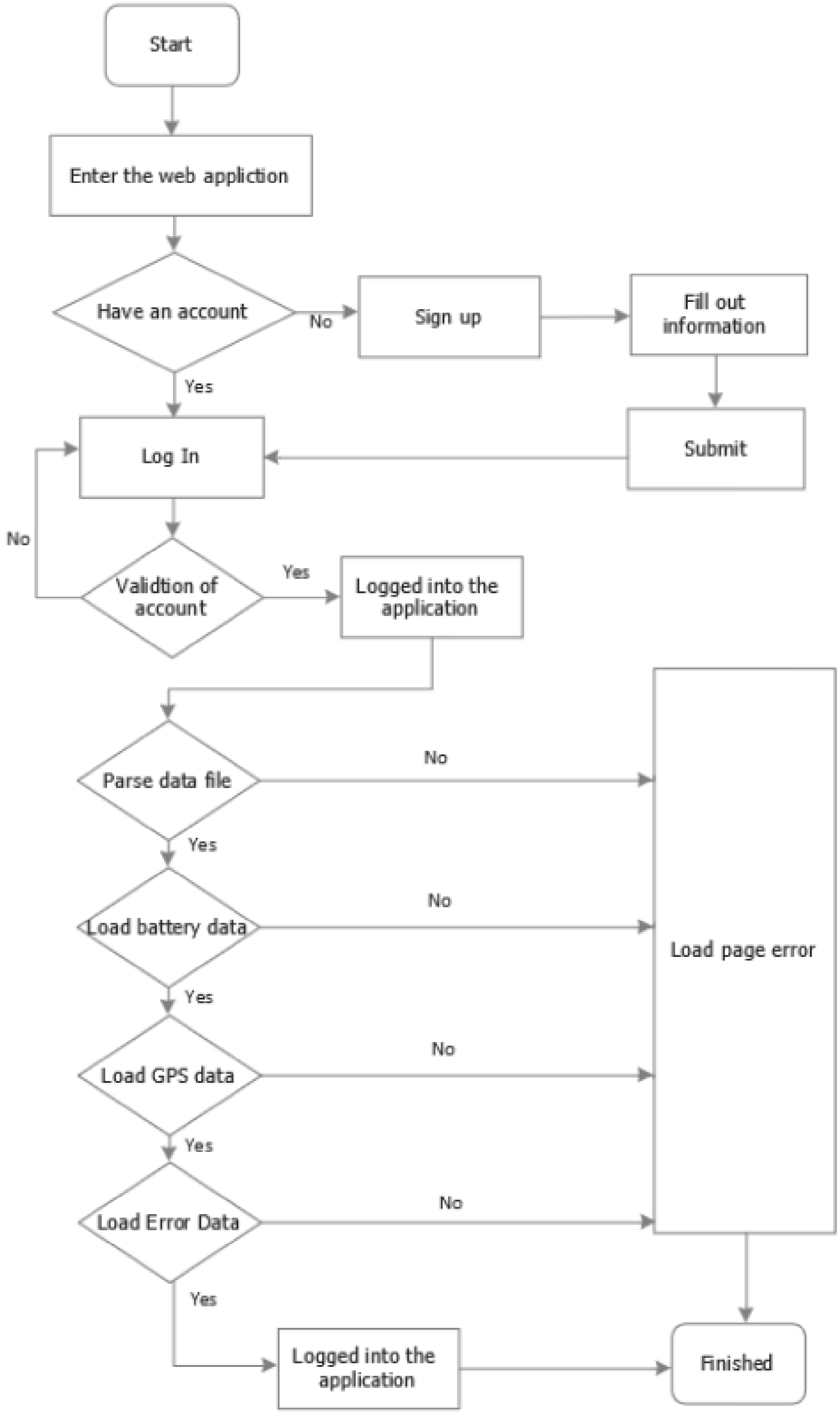

Figure 23: Flow chart of the web application.

4.9.3 Block General Validation

This block, shown as a high level block diagram in Figure 1, will display the battery data, GPS data, and error data to the user after logging into the web application. The web application will display the stored data that is gathered from the file. After logging in the application parses the data file then tries to load each portion of information to display to the user. If data from the file is unable to load an error will be displayed to the user. This is shown in Figure 2 as a flow chart. The web application will be hosted on Amazon web servers. Amazon guarantees a 99.99 percent up-time reliability for EC2 instances during a given month[1]. If Amazon web servers were to go down, we have the ability to host the web application on a home server.

4.9.4 Block Interface Validation

Table 31: Interface Validation for Dsply_dt_otsd_usrout

Type: numbers	The block will receive numerical data for displaying to the user.	The code for this block will take in numerical data and parse the data for display.
Type: string	The block will receive string data for displaying to the user.	The code for this block will take string data and parse the data for display.
Usability: 9 out of 10 people are able to sign in within 5 minutes	Ninety percent was used because it is what is required for the usability test to pass.	The system will be evaluated by at least 10 users and ninety percent of the users will be able to create an account and see data displayed on a computer within 5 minutes.

Table 32: Interface Validation for Rs_system_dsply_dt_data

Message: Battery data	The user needs to be able to see the status of the battery data.	The code for this block will display the battery data to the user.
Message: GPS data	The user needs to be able to see the GPS location of the system.	The code for this block will display GPS data to the user.
Message: error data	The user needs to be able to see the error codes and a description of the error code.	The code for this block will display error code to the user.
Protocol: file	The block data needs to be parse from a file to display data to the user.	The file will store data that will be parsed for display.

4.9.5 Block Testing Process

1. User will open the web application on a computer.
2. User will enter a username and user password.
3. User will click on button that reads “Sign in“.
4. User will be able to see the battery data, and IMU data.

4.9.6 References and File Links

References

[1] “Is the amazon web services cloud reliable?,” Today I Learned Cloud, 08-Jul-2020. [Online]. Available: `https://www.todayilearnedcloud.com/Is-The-Amazon-Web-Services-Cloud-Reliable/`. [Accessed: 22-Jan-2022].

4.9.7 Revision Table

Section 4.9 - Display Data Revisions	
Date	Revision
01/07/2022	Tyrone Stagner: created block 1 revision
01/15/2022	Tyrone Stagner: made adjustment via feed back
01/19/2022	Tyrone Stagner: made adjustments to 4.9.4 via feed back
01/20/2022	Tyrone Stagner: Tyrone made adjustments to sections 4.9.1 through 4.9.3. The block has changed names and the interface properties. The block diagram and description was updated to reflect the changes.
01/21/2022	Tyrone Stagner: made changes to sections 4.9.4 though 4.4.6 to update interface definitions and why they exist. The test steps were updated to verify the interface definitions
03/06/2022	Tyrone Stagner: Added to main document and made some changes to the sections 4.9.1 through 4.9.6
03/06/2022	Tyrone Stagner: Adjusted layout of section so it would be cleaner looking

4.10 Micro-controller Driver

4.10.1 Block Overview

The microcontroller driver block is championed by Andrew Pehrson and its main purpose is to process and pass on sensor data from the navigation sensors block to the ROS system. This block consists of code on an ESP32, focused around the Rosserial_arduino library. The Rosserial Arduino library handles all serial communication and allows the microcontroller to be treated as a node within the ROS architecture. This allows the ESP32 to post and subscribe straight to ROS topics. Some code that will be used from the navigation sensors block are some structs to hold sensor data, a function to initialize the sensors, and some functions to populate the structs with current sensor data. This block will also have some code that reads in a voltage from the power management block with the ESP32's ADS. The last component of the block is that there will be a GPIO pin programmed to unlock the lock box block via a relay within the block

4.10.2 Block Design

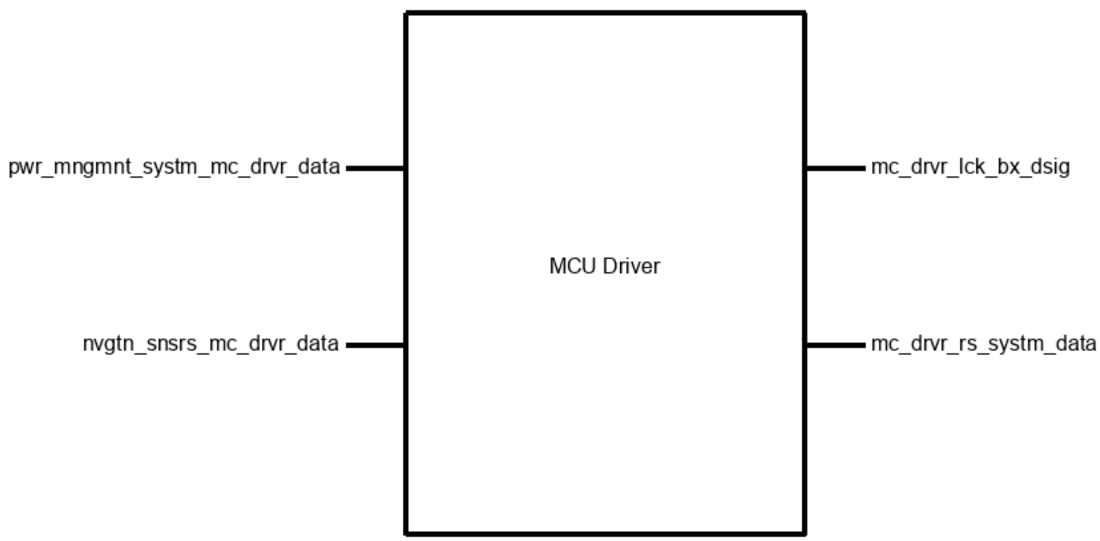

Figure 24: Black box schematic of Micro-controller Driver block.

Steps for posting data:
void setup()

1. Initialize Sensors

void loop()

1. Read data from nav sensors
2. Read ADC from power management block
3. Turn ADC into voltage
4. Package data

5. Post to relevant ROS topics

Steps for controlling lock box:
void setup()

1. Create handler function for topic message
2. Create subscriber for the lock box controller topic
3. Attache subscriber to node hander

void loop()

1. spin the node handler once

4.10.3 Block General Validation

The data transmission method being used for this block meets the needs of the system by utilizing ROS's capabilities to its fullest. Previous designs of this block were going to have a second script on the ROS system side that would act as the ROS node to post and subscribe to topics. Using the Rosserial Arduino library allows us to reduce the time to program this block, the amount of time data is passed around, and also integrates the ESP32 seamlessly into ROS's architecture.

The ADC reading of voltage from the power management block has been designed to require very little parts (just a voltage divider) and using the ESP32's 12 bit ADC we can get an accurate reading of the batteries voltage level.

The GPIO control of the lock box will simplify the amount of hardware needed on the robot. This also means that the ESP32 will be the robots main interface with the hardware such as sensors, actuators, lights, and any other addition the robot might see in the future. This allows the main computer being used for ROS to be upgraded without needing a change in ROS drivers.

4.10.4 Block Interface Validation

Table 33: pwr_mngmnt_systm_mc_drvr_data

Protical: Analog Voltage Signal	This method of reading battery voltage was chosen since a microcontrollers ADC can be used to read changes in voltage	The ESP32 has a 12 bit ADC that has variable attenuation modes to get the most resolution along with the ability to calibrate.
Vmin: 0.15V	This is the ESP32's minimum readable voltage with the largest attenuation being used	Since a voltage divider inside the power management block is being designed to wear as long as the batteries are connected the voltage should never get this low let alone below this value
Vmax: 2.45V	This is the ESP32's max readable voltage with the largest attenuation being used	It is being assumed that the voltage divider inside the power management block is being designed to output a voltage lower then this level when the battery is fully charged

Table 34: nvgtn_snsrs_mc_drvr_data

Messages:Orientation (IMU)	The IMU inside the navigation sensor block will be giving data relevant to the robots orientation	The code will take in IMU readings from a function being provided by the navigation sensors block and store this data in a struct
Messages: Position (GPS)	The GPS inside the navigation sensor block will be giving data relevant to the robots position	The code will take in GPS readings from a function being provided by the navigation sensors block and store this data in a struct
Other: Dsig (bump sensors)	The bump sensors inside the navigation sensor block will give a digital signal when it has collide with something	The ESP32 will read in this digital signal using a GPIO pin and be able to read if a collision has occurred

Table 35: mc_drvr_lck_bx_dsig

Logic-Level: Active High for unlock	This is the signal that will be needed to activate a relay that will unlock the lock box	The ESP32 will be able to output this signal through one of its GPIO pins
Vnominal: 3.3v	This is the voltage needed to activate a relay that will unlock the lock box	The ESP32 will be able to supply this voltage through one of its GPIO pins
Other: Dsig returned (lid closed)	This is a return signal from the lock box block and will tell the microcontroller whether the box is locked or not	The ESP32 will be able to receive this signal through one of its GPIO pins

Table 36: mc_drvr_rs_systm_data

Protical: ROS Topic	This is the communication protocol used throughout ROS and will allow the microcontroller to share data with the whole system	The Rosserial_arduino library will allow the ESP32 to directly subscribe and post to topics over serial with ROS
Messages: Orientation (IMU)	The orientation data from the IMU is needed by the ROS system for the navigation stack	The ESP32 will be able to pull data from the struct of IMU data and post it to the necessary ROS topic
Messages: Position (GPS)	The position data from the GPS is needed by the ROS system for the navigation stack and webpage	The ESP32 will be able to pull data from the struct of GPS data and post it to the necessary ROS topic
Messages: Battery Voltage	The battery voltage data is needed by the webpage to display battery percentage	The ESP32 will be able to pull data from the ADC and post it to the necessary ROS topic
Messages: Bool (Bump sensors)	The bump sensor data is needed by the ROS system to tell if there has been a collision	The ESP32 will be able to pull the data from the GPIO pin and post it to the necessary ROS topic

4.10.5 Block Testing Process

The ESP32 will be plugged into a Raspberry Pi running ROS. Test functions will be used that are the same format as the functions being provided by the navigation sensors block but will provide predetermined incrementing data.

1. Power on the Raspberry Pi
2. Open the navigation sensor data test topic on the Raspberry Pi to view activity

3. Check if data is being posted to the topic and match the data expected out of the test functions
4. Now open the battery voltage data test topic on the Raspberry Pi to view activity
5. Connect a DC power supply to the ADC that will be used by the power management block
6. Very the voltage from the supply between 1-2v
7. Check is change in voltage is reflected by topic
8. Finally connect DMM to GPIO that will be used by the lock box block
9. Manually change the lock box test topic on the Raspberry Pi
10. Check if change is reflect on the GPIO output

4.10.6 References and File Links

References

[1] "Analog to digital converter (ADC)," ESP. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc.html. [Accessed: 22-Jan-2022].

[2] "Wiki," ros.org. [Online]. Available: http://wiki.ros.org/rosserial_arduino/Tutorials/Blink. [Accessed: 22-Jan-2022].

4.10.7 Revision Table

Section 4.1 - Motor Controller Revisions	
Date	Revision
1/07/2022	Andrew Pehrson: added interfaces to motor controller block
1/09/2022	Andrew Pehrson: Block changed to emergency sensors
1/17/2022	Andrew Pehrson: Block changed to database
1/19/2022	Andrew Pehrson: Block changed to data management
1/21/2022	Andrew Pehrson: Block changed to microcontroller driver

5 System Verification Evidence

5.1 Universal Requirements

5.1.1 The system may not include a breadboard

The system does not contain a breadboard. A custom PCB was developed for this project and all module connections are made JST connectors, USB, threaded coaxial cable or similar connection types. All connections will be reliable for extended periods of time. No video or picture currently available (link will be here when it is).

5.1.2 The final system must contain both of the following: a student designed PCB and a custom Android/PC/Cloud application

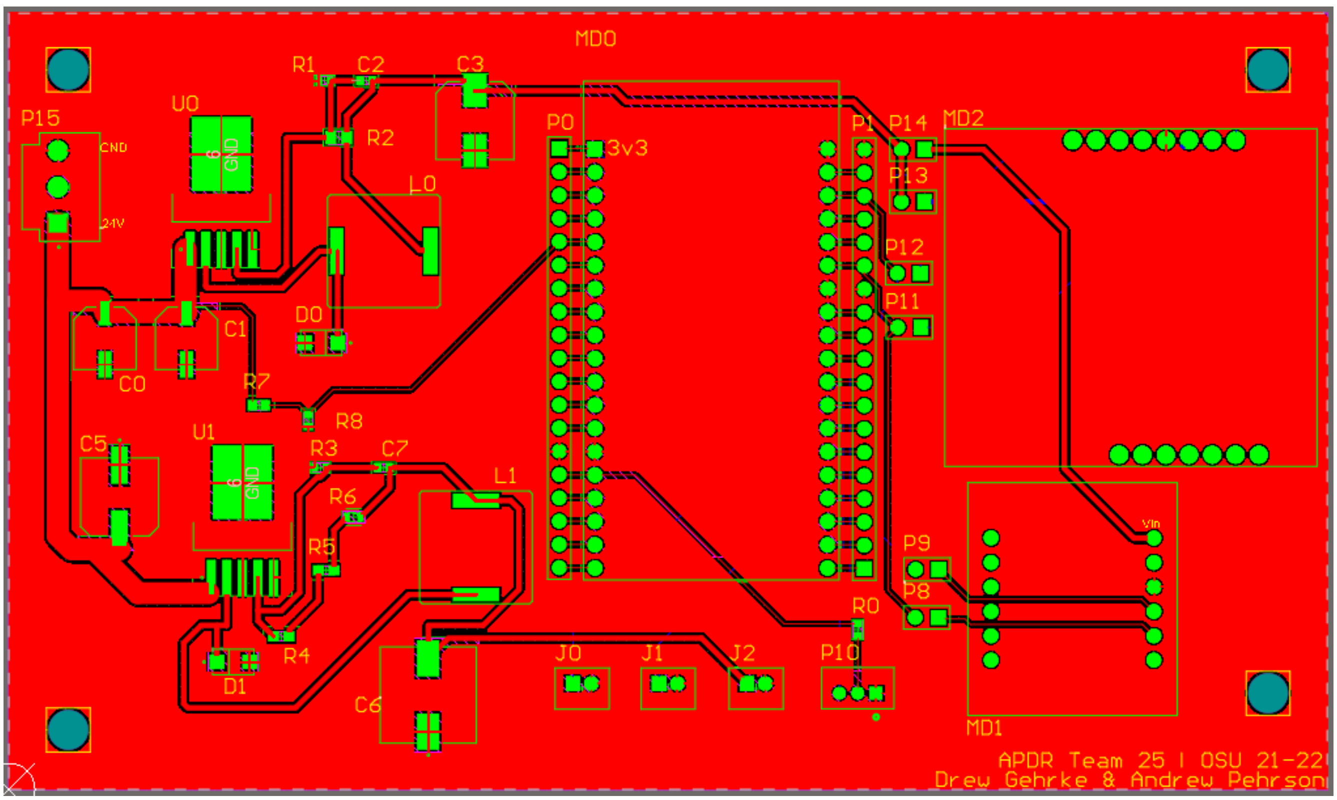

Figure 25: APDR PCB designed by Drew Gehrke and Andrew Pehrson

Figure 25 shows the footprint of the PCB designed for this project. It combines multiple blocks from the entire project into a consolidated package. The Power Management System block is present here with circuits utilizing the LM2596 chips to provide a 3.3V and 12V voltage output as well as a battery monitoring spot on the ESP32. The modules from the Navigation Sensors block (the ESP32, GPS, and IMU) are attached to the PCB as well via female headers and traces to each of the wires for the I^{2}C communication. Also, there are several JST connectors at the bottom for the various GPIO connections, including the bump sensors, the relay to power the locking mechanism on the lock box, and the lock box switch which indicates the open lid being

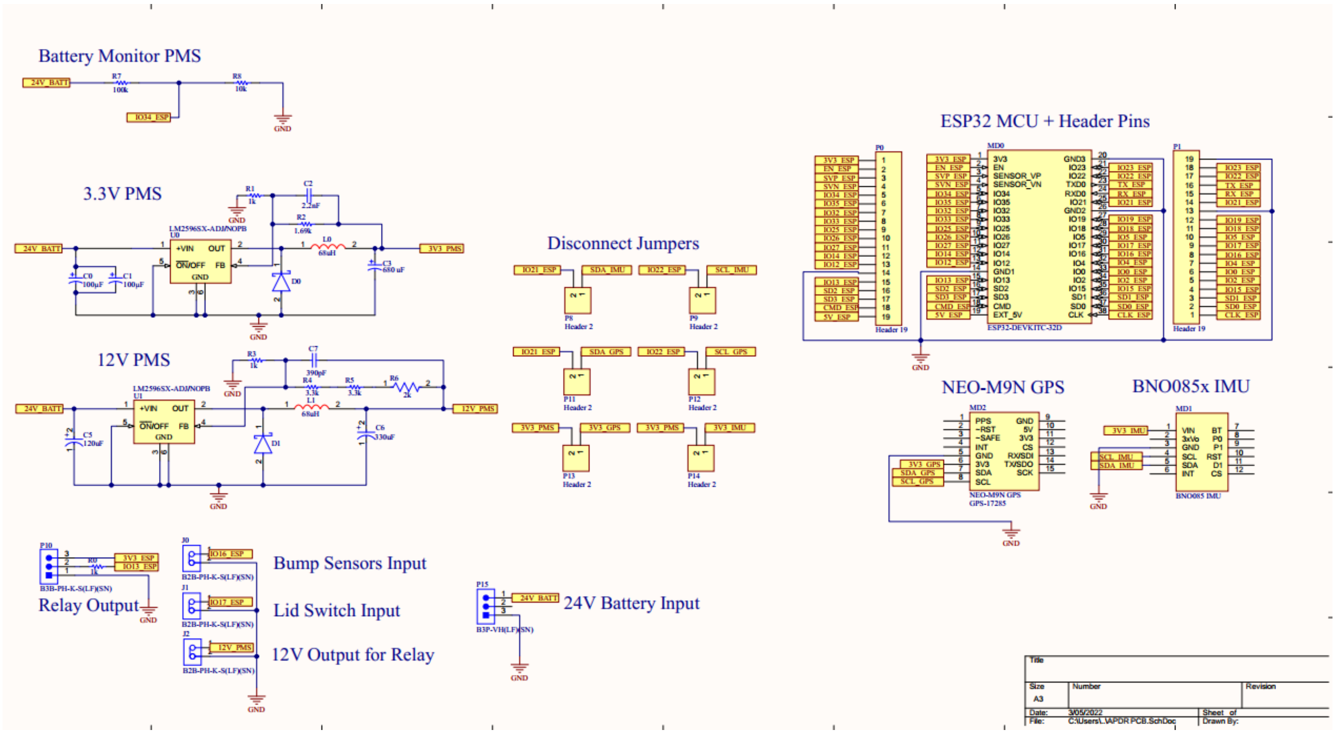

Figure 26: APDR PCB schematic

open. The schematic for the PCB can be seen in figure 26.

The Cloud application is using Amazon Web Servers to host a website. The Server is running Node.js and React. The website will connect to the system via a Web-Socket and communicate to different services that are running on the system.

5.1.3 If an enclosure is present, the contents must be ruggedly enclosed/mounted as evaluated by the course instructor

All electronics are ruggedly enclosed within a 3d printed enclosure located on the underneath of the lock box. This enclosure is secured onto the frame of the robot. This enclosure will contain the PCB and Raspberry Pi. It is built so that if the robot is shaken, no electronics will dismount and fall off the robot.

5.1.4 If present, all wire connections to PCBs and going through an enclosure (entering or leaving) must use connectors

Bellow is a table that shows each connection and how this connection meets this requirement...

Connection	Justification
GPS Antenna	This connection will be straight to the GPS module that is mounted against the side wall of the enclosure and allows access from the outside
PI4 to ESP32	This connection will be internal to the enclosure and be done with a USB cable
Battery to PCB	This connection will be done using a JST-VH on the inside of the enclosure and a JST-SM connector on the outside of the enclosure
Bump Stop	These connections will be done with JST-PH connectors at the wall of the enclosure and on the PCB
Lock Box	This connections will be done with JST-PH connectors at the wall of the enclosure and on the PCB
LIDAR	This connection will be done with a JST connector into the enclosure and then a USB to the PI

5.1.5 All power supplies in the system must be at least 65% efficient

All of the power supplies used by the system and implemented on the PCB are contained within the Power Management System block described in Section 4.4 of this document. All of these power supplies are implemented with the LM2596 switch-mode buck converter controller with varying external inductor values to achieve different output voltages.
One of the greatest advantages to switching power supplies is their high conversion efficiency ratings, with ratings exceeding 90% being relatively common, although this efficiency tends to drop off as the difference between the input and output voltage increases.
Three LM2596 circuits are used to produce outputs of 3.3V, 5V, and 12V from a nominal input of 24V. The LM2596 datasheet (Section 4.4.6) states there is a typical output efficiency of 73% for the 3.3V supply, 80% for 5V, and 90% for 12V.
These results have been confirmed through testing, with efficiencies never falling below%. Therefore, all power supplies are considered to be at least 65% efficient based on these specifications and evidence.

5.1.6 The system may be no more than 50% built from purchased modules

Differentiating between what was feasible to construct from scratch in the given time frame and what needed to be sourced externally took some deliberation at the start of this project. The APDR system is extremely complex project, the justification for the authorship of each block will be described in the table below.

Table 37: Team Authorship Verification

Project Block	**Justification of Team Authorship**
Motor Controller	This block consists of motors, wheel encoders, and a microcontroller to interpret the input from the wheel encoders and output a control signal to the motors. The protocol for communicating with the motors from the ESP32 needed to be created from scratch. The wheel encoders also required a custom mounting setup. The code to interface the input received from the encoders with the output being sent to the motors in order to prevent drift required custom development. The motors are an inherent component to the recycled electric wheelchair base that is a staple to this project, so they shall be discounted. Given the software interfacing with both modules and the custom mounting of the wheel encoders, **this block can be said to contain greater than 50% custom developed modules**
Navigation Sensors	The GPS is a purchased module. The software to interpret the coordinate data in the intended fashion was developed by the team. The IMU sensor is a purchased module. The code to read this IMU data and extract what we needed to interface with ROS was developed within the team. The bumpers were fully developed within the team. The mounting mechanism was designed internally and the software to interpret the inputs from the buttons was developed internally. **This block contains less than 50% purchased modules.**
Edge Detection	The camera module used as an input to this block is purchased. All software to convert the camera data to a useful coordinate system, filter images, isolate edges, and output data was developed internally. **This block contains more than 50% internally developed modules**
Power Management System	The power management system was developed to take an input of 22-26V and output 3.3V, 5V, and 20V across 3 different rails. The additional functionality of being able to determine charge level of the batteries is output on a different rail. All circuitry was developed internally and placed on the custom PCB. **This block was 100% developed internally** and contains no purchased modules.
Web Controller	This software-exclusive block was designed to fit the exact needs of our project. **This block contains no purchased modules**

The tables show that blocks vary greatly in the amount of purchased modules that were implemented in their design. It can be safely inferred that the project in its entirety contains no more than 40% purchased modules.

Table 38: Team Authorship Verification Cont.

Project Block	**Justification of Team Authorship**
Lock Box	A custom enclosure and interface with an electronically triggered actuator were developed for this block. The magnetic actuator was purchased. All software for interfacing with this actuator was developed internally. The physical lockbox was constructed from a premade box with enough custom mounting development to be considered not a purchased module. **This block contains no more than 50% purchased modules.**
Path Generation	This software-exclusive block was built to interface the team's custom mixture of sensors and design goals. This includes the ability to create waypoints from predefined destinations and using GPS, IMU and LiDAR data to accurately localize the APDR within it's environment and bring it to its destination. **This block contains no purchased modules.**
Obstacle Identification	This block utilizes the RPLIDAR S1. Developing the software to interface with this device and produce the desired costmaps and environment awareness was done internally. **This block contains 50% purchased modules.**

5.2 Lock Box

5.2.1 Lock Box

Project Partner Requirement: The robot will have a lock box for transporting the package.

Engineering Requirement: The system will transport a package in a secure container unlocked via input from authorized users.

5.2.2 Testing Process

1. Container will be opened by the user while the robot is not moving.
2. Package will be inserted into the container and closed.
3. The robot then moves to its destination with package in tow.
4. Once arrived, an authorized user will unlock the robot.
5. Once unlocked, container can be opened and package can be accessed.

5.2.3 Testing Evidence

When the user sends an input to the robot, the locking mechanism disengages and the box is capable of being opened. Evidence of this can be found here.

5.3 Emergency Stop

5.3.1 Emergency Stop

Project Partner Requirement: The robot should have an easy to access button to stop the robot in case of an emergency and to assist with testing procedures. The robot must also stop if a collision is detected.

Engineering Requirement: The system will shut down within 500ms after the emergency button or collision sensors activate.

5.3.2 Test Process:

1. Begin a recording of the robot.
2. Command the robot to move forward at its standard operating speed with no obstacle in it's path.
3. Have someone push the emergency stop button to stop the robot.
4. Review the footage to ensure the robot stops within 500ms of the button being pushed.
5. Repeat the above steps with an obstacle in the path of the bump sensors.
6. Upon collision, ensure the robot stops within 500ms of the bump sensors being activated.

5.3.3 Testing Evidence:

The system is capable of stopping movement within 500ms of the stop button or bump sensors being pressed as evidenced by the recordings found here and here.

5.4 Battery Monitoring

5.4.1 Battery Monitoring

Project Partner Requirement: The robot should have a means of monitoring the voltage of its onboard batteries.

Engineering Requirement: The robot will measure the series voltage of its two lead acid batteries within an accuracy of 100mV.

5.4.2 Test Process:

1. With the robot turned on and not moving, use a voltmeter to directly measure the battery voltage.
2. SSH into the Raspberry Pi and navigate to the working directory of the project.
3. Run the command ”ros2 topic echo /battery”.
4. After a brief delay, the battery voltage will be output to the terminal window.
5. Verify that the reported battery voltage is within 100mV of the read voltage.

5.4.3 Testing Evidence:

The reported battery voltage is within 100mV of the value measured with the voltmeter. Link to evidence can be found here.

5.5 Edge Detection

5.5.1 Edge Detection

Project Partner Requirement: The robot should stay on the sidewalk.

Engineering Requirement: The system will determine the bounds of pathways and maintain a minimum distance of 15cm from the edge of said pathway.

5.5.2 Test Process:

1. Place the robot so that it has a wall on either its left or right.
2. Set a waypoint to where the robot would collide with a wall if it went along a direct path to the waypoint.
3. Command the robot to traverse to the waypoint.
4. Observe and ensure the robot is capable staying at least 15cm away from the wall during its traversal.

5.5.3 Testing Evidence:

The team was not able to implement this feature into the final system. However, much of the support for this requirement can be implemented by future teams. Link to a simulation showing the requirement can be found here.

5.6 Path Following

5.6.1 Path Following

Project Partner Requirement: The robot should be able to make across campus deliveries.

Engineering Requirement: The system will follow a predefined path to its destination and deviate from that path by no more than 1 meter.

5.6.2 Test Process:

1. Place a straight strip of tape down on the floor.
2. Align the robot along the strip.
3. Send the command to the robot to drive in a straight line.
4. Stop the robot once it has reached the end of the strip.
5. Ensure that the robot did not deviate from its path by 1 meter.

5.6.3 Testing Evidence:

The robot did not deviate more than 1 meter from its path when sent forward. Link to the evidence can be found here.

5.7 Object Reaction

5.7.1 Object Reaction

Project Partner Requirement: The robot should be able to go around stationary objects in its path.

Engineering Requirement: The system will traverse around stationary objects in its path and not get closer than 15cm to said object.

5.7.2 Test Process:

1. Place an obstacle in front of the robot.
2. Mark a 15cm radius around the obstacle.
3. Command the robot straight towards the object and navigate around the obstacle.
4. Command the robot to be back on course for its straight path.
5. Ensure the robot was able to avoid the object outside of the 15cm radius and continue its path.

5.7.3 Testing Evidence:

The system traversed around a stationary object in its path and did not get closer than 15cm to said object. Link to the evidence can be found here.

5.8 Data transferred from website

5.8.1 Data transferred from website to system

Project Partner Requirement: The system will receive data from the website.

Engineering Requirement: The system will receive data from website.

5.8.2 Test Process:

1. Ensure robot system is online.
2. Access the web-page via the IP address of the website.
3. Input IP address and port number into ROS URL input field.
4. Hit "Toggle Connect" button on website.
5. Wait for lockbox to unlock.

5.8.3 Testing Evidence:

The system received data from website then the test has passed. Link to evidence can be found here.

5.9 Data Transferred from System

5.9.1 Data transferred from system to website

Project Partner Requirement: The system receives and transfers data to the website.

Engineering Requirement: The system will transfer IMU data to the website for users to view.

5.9.2 Test Process:

1. Ensure robot system is online and IMU data is being loaded.
2. Access the web-page via the IP address of the website.
3. Input IP address and port number into ROS URL input field.
4. Hit "Toggle Connect" button on website.
5. Ensure all topics are showing on the website.
6. From the console log of the browser, look at the messages being displayed from the IMU data topic from the system.

5.9.3 Testing Evidence:

The website is capable of showing all currently running topics and IMU message data is seen in the console log. Link to evidence can be found here.

5.10 References and File Links

5.10.1 References (IEEE)

5.10.2 File Links

5.11 Revision Table

Section 3 - Top Level Architecture Revisions	
Date	Revision
3/11/2022	Drew Gehrke: First two requirements added and section 5.1 updated
4/18/2022	Drew Gehrke: Added all other requirements and formatted for sections
4/21/2022	Drew Gehrke: Removed one requirement which shouldn't have been added.
5/4/2022	Drew Gehrke: Revised test procedures and testing evidence for all requirements.
5/6/2022	Nick McBee: Updated Power Supply Efficiency testing evidence.

6 Project Closing

6.1 Future Recommendations

6.1.1 Technical Recommendations

One problem the team faced was the lack of computing power from the onboard computer selection. The Raspberry Pi 4 has a base clock 1.5 GHz [1] which is substantial for most situations. Off-boarding calculations onto the ESP32 microcontroller proved to be very beneficial, but still did not solve all the issues. ROS itself requires lots of computational power and is better served on computers with faster capabilities and much more processing power, especially those with multiple cores to take advantage of the parallelism of ROS. We had attempted to use the NVIDIA Jetson TX2, but sadly were not able to get it running with ROS2. The newer NVIDIA Jetson Nano could be a viable option, once prices go down due to the ongoing chip shortage. It has been proven to work with ROS2 by various people and as such could be a valuable addition to the project. Take a look at reference [2] for more information.

One of the goals of the team was to implement a camera in order to incorporate image processing capabilities. However, this was not able to be implemented due to time, power, and processing constraints we found through the process of the project. If a new onboard computer with more processing power is implemented, then the inclusion of a camera would be more than ideal for the system. This would allow for obstacle identification to be reinforced, edge detection to work much better, and would greatly contribute to traversal of the robot.

Another technical recommendation we have is to learn the ROS2 navigation systems very early on and implement them quickly. Many simulation tutorials exist and should be performed on team member's personal laptops running Linux or a virtual machine to get insight on how ROS works. However, software integration on hardware should be performed as early as possible to allow time for debugging and configuring parameters. The navigation stack in particular requires considerable modification to obtain proper transforms, sensor fusions, and other tweaks to obtain good performance.

One final technical recommendation we have is to try moving to a more robust communication system than the OSU network. As good as the internet is at OSU, it sadly does not allow for peer-to-peer communication very easily. The only way to attempt to use peer-to-peer communication with the system is to get all devices registered on the OSU secure, hidden robotics network. This is a timely process and led to many issues when attempting to do web communications. As such, we recommend moving the system to something outside of the scope of WiFi. This could include cellular networks or other forms of peer-to-peer communications.

6.1.2 Global Impact Recommendations

One global impact recommendation we have is to incorporate more safety features for the robot. As it stands, the primary sources of safety come in the form of the LiDAR unit for obstacle avoidance, the bump sensors at the very front of the robot, and an emergency button for disconnecting power. These are a good foundation, but more can be implemented to ensure proper safety of the environment around the robot and the robot itself. Some options include other sensors (more LiDAR, more bumpers, etc.) around the robot to have a full 360 degrees of detection, some sort of manual sleep mode on the robot to allow an administrator user to temporarily stop the robot if something were to occur, image processing to allow the robot to identify potential obstacles sooner, or an alarm system which would warn people in the vicinity of the robot that they could be in the way of the robot. Look into more about LiDAR with ROS at reference [3].

Another global impact recommendation is to try and use environmentally friendly materials moving forward. A sort of baseline has been developed at the conclusion our part of the project, and now it's time for a new rendition / improvements. Some of the parts used – including the plastics, electrical components, and other materials – could be substituted with for eco-friendly materials very easily. As such, to reduce global and environmental impacts of this project even more, we highly recommend sourcing eco-friendly materials if anything new is to be added to the project, or replacing parts with environmentally-friendly materials.

6.1.3 Teamwork Recommendations

One very big recommendation is to integrate blocks early and often. While developing blocks, many of the inputs and outputs of the blocks are simulated values which are set by the block champion. While the design may have the proper outputs intended, things don't always work that way. When the time comes to integrate, oftentimes it is not as smooth as planned. As such, we highly recommend that future teams integrate their blocks with one another as soon as possible. This will help to build some communication amongst team members as you can discuss inputs and outputs of each block and be able to effectively come to a solution. This will also save time when system verification rolls around as you won't have to spend as much time the third term debugging and have less stress as the deadlines get closer.

Another big teamwork recommendation is to review each other's work frequently. Whether during the process of development or reviewing a final product, looking over another teammate's work can help to mitigate bugs which could occur in software or hardware. This will aid in team cohesiveness and communication due to the constructive criticism from other members of the team. Pairing up on certain parts of the project makes this an effective strategy. For instance, the team this year had an electrical team, ROS team, and website team. Two members of the team were on the each of the electrical and ROS teams, while only one was on the website team. As such, many members had knowledge of the components of other members' blocks and could aid in debugging or reviewing of work. This was a very effective strategy when it came crunch time to meet the deadlines and multiple members of the team were capable of understanding the parts that other members were struggling with. As such, we highly recommend future teams implement this strategy.

6.2 Project Artifact Summary

2020-2021 MPDR GitHub
2021-2022 APDR GitHub
Espressif Datasheet Documentation
ESP32 I^2C Communication
SparkFun GPS NEO-M9N Tutorial
ESP32 Serial Communication with Raspberry Pi Tutorial
Automatic Addison ROS2 Navigation Stack Setup Tutorial
ROS2 Galactic Documentation
Navigation 2 Documentation

6.3 Presentation Materials

2022 APDR Engineering Expo Poster

6.4 References

[1] Raspberry Pi, “Raspberry pi 4 model B specifications,” Raspberry Pi. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. [Accessed: 04-May-2022].

[2] A. Piltch, “Nvidia Jetson Nano: The raspberry pi of ai?,” Tom’s Hardware, 19-Mar-2019. [Online]. Available: https://www.tomshardware.com/news/jetson-nano-features-price,38856.html. [Accessed: 06-May-2022].

[3] V. Mazzari, “Lidar integration with ROS: Quickstart Guide and Projects Ideas,” Génération Robots - Blog, 06-Jul-2021. [Online]. Available: https://www.generationrobots.com/blog/en/lidar-integration-with-ros-quickstart-guide-and-projects-ideas/. [Accessed: 06-May-2022].