

P.A.R.C.E.L

Payload Autonomous Robot for Coordinated, Expedition and Logistics

Midterm Report
EEL4914 - Senior Design 2
Group 24



University of Central Florida
Department of Electrical and Computer Engineering

Riley Newport, CpE
Melyia Ince-Ingram, CpE
Brandon Zabawa, CpE
Ibrahim Khater, EE

Reviewers: Dr. Mike Borowczak *ECE*
Dr. Avra Kundu *ECE*
Dr. Sonali Das *ECE*

Mentor: Chung Yong Chan

Table of Contents

Table of Contents.....	1
List of Figures.....	5
List of Tables.....	7
Chapter 1: Executive Summary.....	1
Chapter 2: Project Description.....	2
2.1 Project Background and Motivation.....	2
2.2 Current Technologies and Prior Work.....	3
2.2.1 Starship Technologies - Autonomous Robot Deliveries (ARD).....	3
2.2.2 Doll-E Automatic Tracking Platform.....	4
2.3 Goals and Objectives.....	4
2.3.1 Overview of Project Goals and Objectives.....	4
2.3.2 Project Features and Functionalities.....	5
2.4 Technical Specifications.....	6
2.5 House of Quality.....	9
2.6 Project Block Diagrams.....	10
Chapter 3: Research and Investigation.....	11
3.1 Identification with RFID.....	11
3.2 Distance Sensor.....	16
3.3 Payload Detector.....	20
3.4 Visual Sensors.....	22
3.4.1 Camera Types.....	22
3.4.1.1 Raspberry Pi V2.....	23
3.4.1.2 Arducam 2MP.....	23
3.4.1.5 Arducam 12MP IMX708.....	24
3.5 Inertial Measurement Units.....	25
3.5.1 6-Axis vs. 9-Axis IMUs.....	25
3.5.2 6-Axis IMU IC Comparison.....	26
3.6 Microcontroller.....	27
3.7 Motors.....	29
3.7.1 Brushed DC Motor Selection and Analysis.....	30
3.7.2 Brushed DC Motor Comparison and Selection.....	31
3.7.3 Driving the Motor.....	31
3.7.3.1 Motor Driver IC Selection.....	31
3.8 Single Board Computers.....	32
3.8.1 Raspberry Pi 5.....	33
3.8.2 Raspberry Pi 4B.....	33
3.8.3 Nvidia Jetson Nano.....	34

3.8.4 Arduino Mega.....	34
3.9 Power Supply.....	36
3.9.1 Battery Considerations.....	36
3.9.2 Power Distribution.....	36
3.9.3 Battery Types.....	37
3.9.3.1 Battery Selection.....	38
3.9.4 Voltage Regulation.....	39
3.9.4.1 Voltage Regulator Selection.....	40
3.10 Wifi/Bluetooth Connection.....	42
3.11 Communication.....	43
3.12 SLAM Protocols.....	45
3.12.1 SLAM.....	46
3.12.3 Visual SLAM.....	47
3.12.2 LiDAR SLAM.....	49
3.13 Visual SLAM Comparison.....	50
3.13.1 ORB-SLAM.....	50
3.13.2 RTAB-Map.....	51
3.14 Image Processor.....	52
3.14.1 OpenCV.....	52
3.14.2 MATLAB Image Processing Toolbox.....	53
3.14.3 Nvidia VisionWorks.....	53
3.15 Path Planning.....	55
3.15.1 DWA (Dynamic Window Approach).....	55
3.15.2 Navfn.....	56
3.16 Mobile App.....	57
3.16.1 Flutter (IOS/Android).....	57
3.16.2 Front and Back-end.....	57
Chapter 4: Standards and Design Constraints.....	58
4.1 Standards.....	58
4.1.1 I2C Protocol.....	58
4.1.2 UART Protocol.....	59
4.1.3 SPI Communications.....	60
4.1.4 RFID Tags ISO14443A/B.....	61
4.1.5 ISO 10218 - Safety Requirements for Robots.....	62
4.1.6 TCP/IP Protocol Standard.....	62
4.2 Design Constraints.....	63
4.2.1 Time Constraints.....	63
4.2.2 Economic Constraints.....	64

4.2.3 Power Constraints.....	65
4.2.4 Ethical and Social Constraints.....	66
4.2.5 Health and Safety Constraints.....	66
4.2.6 Physical Constraints.....	67
4.2.7 Manufacturability Constraints.....	67
4.2.8 Sustainability Constraints.....	68
4.2.9 Testing/Presentation Constraints.....	69
Chapter 5: Comparison of ChatGPT with other Similar Platforms.....	70
5.1 Riley’s AI Comparison.....	70
5.2 Melyia’s AI Comparison.....	71
5.3 Brandon’s AI Comparisons.....	73
5.4 Ibrahim’s AI Comparison.....	75
Chapter 6: Hardware Design.....	76
6.1 Subsystem Block Diagram.....	76
6.2 Schematic Diagram.....	78
6.2.1 Power Delivery.....	78
6.2.2 Motor Controls.....	81
6.2.3 Raspberry Pi 5 HAT.....	83
6.2.4 RFID Sensor.....	86
6.2.5 Main PCB.....	87
6.3 Architecture.....	90
6.4 Structural Illustration.....	91
Chapter 7: Software Design.....	93
7.1 Activity Flow Diagram.....	93
7.2 Use Case.....	93
7.3 System State Diagram.....	94
7.4 Connection & Communication to an Arduino or Raspberry Pi.....	97
7.5 ORB SLAM Algorithm.....	101
7.5.1 Yolov5 and OpenCV.....	101
7.6 Mobile App.....	102
7.6.1 Application Overview.....	102
7.6.2 User Interface.....	104
Chapter 8: System Fabrication and Prototyping Construction.....	105
8.1 Mainboard.....	105
8.2 Motor Controller.....	110
8.3 Raspberry Pi 5 HAT.....	113
8.4 RFID Reader Breakout Board.....	115
Chapter 9: System Testing and Evaluation.....	117

9.1 Hardware Testing.....	117
9.1.1 ESP32-WROOM-32E Microcontroller.....	117
9.1.2 Raspberry Pi 5.....	117
9.1.3 Optical ToF.....	118
9.1.4 Photodiode.....	118
9.1.5 Raspberry Pi V2 Camera.....	118
9.1.6 Motors.....	119
9.2 Software Testing.....	119
9.2.1 ESP32-WROOM-32E.....	119
9.2.2 RFID Module and Tags SPI Communications.....	119
9.2.3 ESP32 to Raspberry Pi UART Communications.....	120
9.2.4 I2C Communications.....	121
9.2.5 TCP Network Communications.....	121
9.2.6 ORB SLAM.....	122
9.2.7 Mobile App.....	122
Chapter 10: Administrative Content.....	123
10.1 Budget.....	123
10.2 Bill of Materials.....	124
10.3 Project Milestones.....	125
10.4 Work Distributions.....	126
Chapter 11: Conclusion.....	129
Appendices.....	131
Appendix A – References.....	131
Appendix B - Copyright Permission.....	139
Appendix C - Data Sheet.....	139
Appendix D - Software Code.....	143
D.1 RFID Read/Write and Print to Serial Monitor.....	143
D.2 Raspberry Pi Test Code for Communicating with ESP32.....	145
D.3 ESP32 Test Code for Communicating with Raspberry Pi.....	146
D.4 TOF Sensor Test Code [C.1].....	146
Appendix E – ChatGPT Prompts and Outcomes.....	148
E.1 Explain Paragraph on Inductive Coupling ChatGPT Prompt and Outcome.....	148
E.2 Explain Paragraph on Inductive Coupling DeepAI Prompt and Outcome....	149
E.3 Explain Capacitive Transducers ChatGPT Prompt and Outcome.....	150
E.4 Explain Capacitive Transducers DeepAI Prompt and Outcome.....	151
E.5 Ideas for Ethical Constraints.....	152
E.6 Which raspberry pi model would be recommended?.....	153
E.7 Make a comparison between Raspberry Pi 5 and other microcontrollers why	

is it best for this project proposal.....	155
E.8 Motor Torque Calculation - DeepSeek vs ChatGPT-4o.....	158
E.9 DC Motor Control MCU.....	161

List of Figures

Figure 1: House of Quality by Authors	9
Figure 2: Hardware Block Diagram by Authors	10
Figure 3: Software Flowchart by Authors	10
Figure 4: Visual SLAM Flowchart	48
Figure 5: Motor Control Subsystem Block Diagram	77
Figure 6: Sensor Subsystem Block Diagram	77
Figure 7: Power Delivery Subsystem Diagram	78
Figure 8: 3.3V Regulator Circuit	79
Figure 9: 5V Regulator Circuit	79
Figure 10: 5V Raspberry Pi 5 Regulator Circuit	80
Figure 11: Battery Input Terminal	80
Figure 12: Raspberry Pi Output Terminal	81
Figure 13: Motor Driver Circuit	81
Figure 14: Motor Driver Header	81
Figure 15: Motor Terminal	82
Figure 16: Motor Header on Main PCB	83
Figure 17: Raspberry Pi 5 HAT	83
Figure 18: TCA9548APWR 8:1 Multiplexer	84
Figure 19: ToF Sensor Plugs	85
Figure 20: IMU Sensor	85
Figure 21: RFID Module Breakout Board	86
Figure 22: RFID Header and Configuration on Main PCB	87
Figure 23: Microcontroller Circuit Schematic	87
Figure 24: Reset Button Configuration	88
Figure 25: USB-to-UART Interface	89
Figure 26: USB-to-UART Interface – Raspberry Pi 5 Connection	90
Figure 27: Top Down View of Robot Diagram	91
Figure 28: Side View of Robot Diagram	92
Figure 29: Front View of Robot Diagram	92
Figure 30: Activity Flow Diagram	93
Figure 31: Use Case Diagram	94
Figure 32: System State Diagram	96
Figure 33: Launch Screen	104
Figure 34: Sign In Screen	104
Figure 35: Dashboard Screen	105
Figure 36: Mainboard PCB Layout	106
Figure 37: 3.3V Voltage Regulator Layout	107
Figure 38: 5V Voltage Regulator Layout	107
Figure 39: 5V High Power Voltage Regulator Layout	108
Figure 40: USB-to-UART Connection to Micro-USB Programming Port	108
Figure 41: Peripheral and Sensor Connection Headers	109
Figure 42: Microcontroller and Status LEDs Layout	109
Figure 43: 3D Render of Mainboard - Front	110
Figure 44: 3D Render of Mainboard - Back	110

Figure 45: Motor Controller PCB Layout	111
Figure 46: Motor Controller Components Layout - Motor 1	111
Figure 47: Motor Controller Components Layout - Motor 2	112
Figure 48: Dual LED Direction Indicator	112
Figure 49: 3D Render of Motor Controller Board - Front	113
Figure 50: 3D Render of Motor Controller Board - Back	113
Figure 51: Raspberry Pi HAT PCB Layout	114
Figure 52: 3D Render of Raspberry Pi HAT - Front	114
Figure 53: 3D Render of Raspberry Pi HAT - Back	115
Figure 54: RFID Reader Breakout PCB Layout	116
Figure 55: 3D Render of RFID Reader Breakout Board - Front	116
Figure 56: 3D Render of RFID Reader Breakout Board - Back	117

List of Tables

Table 1: Specifications and Requirements	6
Table 2: Comparison of Barcode/QR Code Technology and RFID Technology	12
Table 3: Comparison of the Types of RFID Tag Technology	13
Table 4: Comparison of the Frequency Ranges of Passive Tags	13
Table 5: Comparison of ISO15693 and ISO1443A/B Standards	15
Table 6: Comparison of High Frequency RFID Reader Modules	15
Table 7: Comparison of Optical TOF and LiDAR Technologies for Clarification	18
Table 8: Comparison of Ultrasonic, Optical TOF, and LiDAR Technologies	18
Table 9: Comparison of Optical TOF Sensor Models	19
Table 10: Comparison of Payload Detection Sensors	21
Table 11: Comparison of Types of Photodiodes [A.87]	22
Table 12: Visual Sensor Comparison [C.20,C.21,C.22]	24
Table 13: Comparison of 6-Axis vs. 9-Axis IMU	26
Table 14: Comparison of 6-Axis IMUs [C.36, C.37, C.38]	26
Table 15: Comparison of Microcontrollers [C.8, C.9, C.10, C.11]	28
Table 16: Comparison of Motor Types	29
Table 17: Comparison of Geared Brushed DC Motor Models [A.33, A.34, C.12]	31
Table 18: Comparison of Geared Brushed DC Motor Models [C.13, C.14, C.15, C.16]	32
Table 19: Comparison of Single Board Computers [C.16, C.17, C.18,C.19]	35
Table 20: Power Distribution Table	36
Table 21: Comparison of Battery Types [A.35, A.37, A.40, A.41, A.42]	39
Table 22: Comparison of Low Power Voltage Regulators [C.26, C.27, C.28, C.29, C.30]	40
Table 23: Comparison of High Power Voltage Regulators [C.31, C.32, C.33, C.34, C.35]	41
Table 24: Comparison of Wifi and Bluetooth	42
Table 25: Comparison of TCP and UDP	43
Table 26: SLAM Comparison	49
Table 27: SLAM Algorithm Comparison	51
Table 28: Image processing Comparison [A.88, A.90, A.91]	54
Table 29: Path Planning Comparison [A.97, A.98]	56
Table 30: Physical Characteristics of Identification Cards [A.89]	61
Table 31: ESP32 VSPI Pins [C.8]	119
Table 32: Budget Estimation	123
Table 33: Bill of Materials	124
Table 34: Senior Design 1 Milestones	125
Table 35: Senior Design 2 Milestones	126
Table 36: Work Distributions	126

Chapter 1: Executive Summary

With the increase in major shipping and delivery services there is also an increase in warehouses to house, organize, and transport products. In an effort to improve the system of transporting products across a large warehouse floor with simplicity for the user in mind this project's autonomous delivery robot was designed.

The project discussed in the following report goes over the design of an autonomous smart delivery robot used to optimize payload transportation within warehouse environments. The robot's key features include obstacle avoidance, RFID-based payload identification, payload detection, path planning via the SLAM API, and a user-friendly mobile application for controlling and monitoring. Payloads are identified using RFID tags so the robot can then determine the payload's final destination automatically and accurately. After identifying the payload the robot plans the most efficient route to deliver the payload to its designated drop-off point. The navigation system combines real-time sensor data with pre-mapped coordinates to adapt to any obstacles found along the way to deliver the payload. SLAM will be our main focus for navigating within an indoor environment using localization and pre-mapping techniques. SLAM algorithms have to calculate the current position and orientation by using an environment map in order to function well. Specifically this project implements a Visual SLAM algorithm that uses the sensors to create a map of their surroundings to identify their specific location at a point in time. To provide a seamless user experience this project also includes a mobile application allowing users to track payload status and receive alerts or updates. The app will communicate wirelessly with the robot via Wi-Fi making interaction intuitive and efficient.

This document is a record of the design process for an autonomous delivery robot, including the fundamental theories and technologies, part selection, the implementation process, and limitations. The background motivations surrounding this robot are discussed first, followed by key technologies that enable its implementation. The parts that make up the robot and the design philosophy used to select these components are discussed next. After that, there are sections on the standards and constraints used within the design and a discussion on how AI has been used to assist the team members in this project. Then the schematics for how these components will be implemented into complete systems are provided and then next the software diagrams. Following that is the documentation regarding the system fabrication, prototyping, testing, and administrative content. Final Conclusions are presented, followed by appendixes containing all references, datasheets, copyright permissions, and AI prompts.

This project is a collaboration between three computer engineering students and one electrical engineering student. All students collaborated to make large-scale design decisions, and the details of the hardware and software were split between the electrical engineering students and computer engineering students, respectively.

The autonomous delivery robot designed in this report is a simple, user friendly and affordable product that could provide convenience to those working in a warehouse. It is a practical solution to transport items across a warehouse or factory floor.

Chapter 2: Project Description

2.1 Project Background and Motivation

The first Industrial Revolution in the 18th century marked the emergence of factories as a transformative force in manufacturing and production. Factories revolutionized how goods were created, enabling mass production, reducing costs, and fostering innovation across industries. Through the centuries, the world has witnessed multiple Industrial Revolutions as technology continuously developed. Today, with the rise of automation, the world of manufacturing is experiencing a so-called “Fourth Industrial Revolution” [1]. Advanced robotics, artificial intelligence, and other complex technologies are quickly becoming essential in making factories more efficient and flexible, allowing them to meet the global demands of the 21st century. Automation not only builds on the legacies of efficiency introduced during the previous revolution, but also paves the way for unparalleled advancements in productivity and precision.

From their inception as early prototypes, autonomous robots have undergone significant advancements. Over the past century, autonomous machinery has been continuously developed and improved. Autonomy can be implemented in both industrial and everyday life to improve efficiency and quality of life. In industries such as logistics, healthcare, manufacturing, and e-commerce, autonomous robots have become indispensable in increasing efficiency, reliability, and cost-effectiveness. The growing adoption of autonomous systems is driven by their potential to reduce human labor and streamline operations.

Despite the progress in this field, there are limitations that need to be addressed. Existing systems often lack affordability for small-scale use and fail to provide a simple and easy-to-use experience. This project aims to bridge these gaps, providing a flexible, cost-effective, and efficient system tailored for small and large-scale factory use.

The motivation for this project arises from the increasing demand for automation in both household and industrial applications. Businesses often face logistical challenges that could be lessened with the implementation of an adaptable autonomous delivery system. The project integrates different advanced hardware to achieve efficient autonomy and operation: a short range high frequency (HF) passive Radio Frequency Identification (RFID) system for payload identification, a LiDAR (Light Detection and Ranging) sensor for object detection, a two wheel differential drive system, among other hardware working in tandem with complex software that allows for autonomous navigation.

This project allows for the collaboration of different engineering disciplines to achieve a common goal to create a fully functioning autonomous delivery robot. Capable of object avoidance, payload identification and delivery, and floor map navigation, this project will

address real-world challenges and develop an optimal solution. The autonomous delivery robot project addresses the growing need for versatile and accessible delivery systems. By leveraging modern technology and focusing on practicality, this project provides a prime solution to inefficiencies and represents a step forward in the world of industrial autonomy.

2.2 Current Technologies and Prior Work

2.2.1 Starship Technologies - Autonomous Robot Deliveries (ARD)

Starship Technologies released the first and leading autonomous robot delivery that offers capabilities for deliveries of hot food, industrial supplies and groceries. Their goal was to address the limitations of effectively delivering goods without the added factor of human interference. It relies on artificial intelligent algorithms to properly navigate across a path that constantly operates throughout a 12 hour duration.

The *ARD* system itself utilizes a complex sensor suite containing ultrasonic sensors, time-of-flight cameras, stereo cameras, distance sensors, radar and an alarm system. In addition, various LEDs, the flag and luminous reflectors make them visible even in low-light conditions. This forms a type of awareness bubble for object detection and enhances navigation ability in a three-dimensional space.[2] This system is able to identify objects 200 ft away while reducing its speed when near an adjacent object not in front of it, and makes a full stop when an object is in front. Operating from sunrise to sunset, the robot uses both front and rear LEDs that illuminate across a distance of at least 500 ft.

In terms of navigation, They utilized an AI path planning algorithm that follows pre mapped routes with real time implementation in the case of road closures and construction. The ARD system is monitored by employees which can be taken over to control their autonomous driving features if an area is unable to be traversed. The robots are available to employees anytime and anywhere on site through a browser based platform. The user is able to use their mobile device to be able to track the ARD and receive notifications when the delivery has arrived.

With our project we want to align and fulfill our engineering requirements based on the Starship's AR design weighing in 35 kg. In our case we aim to make the mechanism more affordable as this will be on a lower budget making the body of the robot to be estimated to be less than 20 kg. With the load requirements to be about a 5 kg load however, this will be tentative based on the robots design. The biggest difference that our project will have compared to Starships ARD is the mode of navigation. Due to our time constraints and budget ability the robot will be using an open source route planning algorithm that will likely only utilize 2-3 sensors (LiDAR, ultrasonic and HF RFID sensor). This decision was made based on where the robot would primarily function within an indoor environment.

Another difference would be the energy efficiency and the speed at which it operates.

The starship's ARD use of a 1260 Wh battery approximates a 12hr drive time at a maximum speed of 6 kph. With the environmental difference, we have to adjust the speed to about 20 cm/s ideally consistent for all payloads operating on at least 12V battery.

Although the capabilities between both robots vary, we aim to bridge the gap and improve on affordability and versatility in indoor environments

2.2.2 Doll-E Automatic Tracking Platform

Doll-E is a previous senior design project that we have taken inspiration in regards to our project. Doll-E was built to have the same functionality as a traditional autonomous robot however their goal was to have the robot follow the user while also having to be able to carry various load capacities. Their aim was to target everyday consumers and were set on improving the traditionally dolly cart and eliminating having the user push the cart itself to have the platform to follow from a distance. Their first main specification was to have 20 lb load capacity that maintained a consistent speed of 2-3 mph. Doll-E employs a user-tracking system to follow a person around, while also utilizing ultrasonic sensors to stop within 1 cubic foot before colliding with an object. This is accompanied by the fact the platform must be able to detect the direction of the user within a 45° resolution at an 80% accuracy. We are keeping in mind that the Doll-E did come with its challenges when optimizing the motor and battery combination that increased the budget due to accompanying the 20lb capacity. We will primarily utilize 2 wheels to optimize cost but still keep the capabilities of traversing indoors.

Although our project shares similar functionalities for object detection and carrying capacity, there are various differences and improvements that can be made. Doll-E was designed for single-purpose tracking of the user within a 20lb load capacity, our robot will use the RFID tag to identify the payloads placement which will then trigger the pre mapped route. We will then have to ensure consistent moving speed in an indoor environment and reduce speed when nearing an object within 1m. System integration in Doll-E was limited pretty much to following and angular object avoidance, while our delivery robot will use RFID payload identification integrated with the payload alongside the mobile application that will access premade floorplans are the highlighted features of the project. Where Doll-E focused on tracking precision, our robot focuses on multi-destination support and a return to sender functionality.

Based on this, our work addresses other challenges of autonomous delivery in route planning for multi-destination routes and independence from any user operation. For our project, we hope to execute these improvements based on our autonomous robot predecessors.

2.3 Goals and Objectives

2.3.1 Overview of Project Goals and Objectives

This project's focus is to build a robot to aid in transporting payloads within a warehouse or other industrial environment. The three levels of goals are described below.

Basic Goals: The robot is able to safely transport payloads from one location to another and return to its home location while avoiding obstacles. Design a mobile application for the user to view the robot's status and indicate target location.

Advanced Goals: The robot is able to detect if a payload is on top of it and identify the payload correctly. The robot is able to create a new path to the target location if the original path is blocked.

Stretch Goals: The robot is able to deliver multiple payloads and is capable of creating a new floor map instead of the one given to it. The robot is able to pick and place payloads from a shelf and onto the robot.

To achieve the above goals, our team has certain objectives for the robot in order to perform as expected. Our robot will include a photoresistor to detect the presence of a payload on top of the robot, ensuring that the robot can confirm when an item is loaded or removed. A short range HF RFID sensor will be used to identify specific payloads, allowing the system to differentiate and track payloads effectively. To navigate safely, the robot will use an array of distance sensors to detect objects in its path and reroute its path accordingly. The robot's navigation system will be powered by an API/mapping toolbox running on an onboard Raspberry Pi. Additionally, a mobile application will allow the user to view target delivery locations and offer an intuitive interface for interaction and monitoring. The robot with the features and functionalities described in the following section will achieve our main goals of transporting payloads.

2.3.2 Project Features and Functionalities

The project will have several key features and functionalities. The robot will be able to avoid objects, detect whether payloads are present, create a path to the target location, create a path to its indicated home location, and a user interface in order to control.

Object Avoidance: The robot will be able to detect obstacles in its path and will then stop until the obstacle is removed. This functionality helps support our goal of being able to safely transport payload as it will not run into any person or damage itself or other property.

Payload Detection: The robot will be able to detect if a payload has been placed on its surface. This feature is important in achieving this project's goal of having a system that automatically returns to its home location. It will know that it is ready to return to its home location after the payload has been removed from its surface.

Payload Identification: The robot will have an RFID tag sensor on the surface to be able to correctly identify a payload placed. This feature will satisfy our Advanced Goal of payload identification.

Path to Target Location: The robot will already have a map of the permanent obstacles and structures of the room. Using this map the robot will be able to create a path to location that the user indicates. This path will be an efficient route that will have the robot then drive too. In our Advanced Goal, the robot will be able to create a new path if the original path is blocked by an unforeseen obstacle. In our first Stretch Goal, the robot will be able to deliver multiple payloads. In this case the robot will create a path to deliver the first payload, wait for its removal and then deliver the second payload. Our second stretch goal is for the robot to be able to have a new mode in which it creates its own map of obstacles instead of being provided one by a user ahead of time. The robot would go around the room and indicate the presence of these obstacles. Then when it is time to create a path to a target location it would use this map.

Path to Home Location: The robot will be able to autonomously determine an efficient path to its home location with no help from the user. It will know when the payload is removed and it is time to return to its home location.

User Interface: The robot will be accompanied with an app that will act as the user interface. The user will be able to upload a new map of the room the robot will be housed in. The user will also be able to enter a location on the app that they want the robot to deliver to. One of our Advanced Goals is to have the robot be able to automatically identify the payload. In this case then the user would not have to indicate where they want the robot to deliver to because the robot will have identified the payload and know where that payload is supposed to be delivered too.

2.4 Technical Specifications

The specifications and requirements that we will focus on meeting throughout this project are outlined in the following table. The yellow highlighted rows are the specifications that are demonstrable.

Table 1: Specifications and Requirements

Parameter	Specification	Engineering Requirement	Marketing Requirement
System Specifications			
Delivery System	Within 1m of destination	Accurately able to reach target location within 1 m	6
Object Avoidance	Detects object up to 1m away	Object Avoidance Sensor must detect objects within 1m	1, 6, 7
App Response Time	At most 5 seconds	Once a target	2, 3

		location has been selected on the app, app must show path to location within five seconds	
Identification Accuracy	90% accuracy	Identification must correctly identify a payload with RFID tag 90% of the time	8
Payload Detection	90% accuracy	Robot must correctly detect if a payload is present or absent 90% of the time	8
Payload Capacity	At least 25 kg	Robot's must be able to carry at least 25 kg payloads	4
Movement Speed	At least 20 cm/s	Robot's speed must be at least 20 cm/s	1, 3
Cost	Less than \$500	The Robot's complete construction will cost less than \$500	4
Dimensions	60cm x 60cm x 16cm	Robot's platform must be large enough to carry payloads at least 3600 cm ²	1, 3
Component Specifications			
Battery Voltage	12 V	Battery must be able to supply the motors, MCU, peripherals	9
Battery Capacity	3Ah	Battery must be able to meet current requirements of the motors, mcu,	9

		peripherals	
Battery Lifespan	At least 5 years	Battery must function successfully for at least five years	9
Wheel Radius	16 cm	Small wheel radius is ideal for smooth indoor and flat surfaces	5, 9
Marketing Requirements 1. The system is safe. 2. The system responds quickly to the user. 3. The system has a consistent movement speed. 4. The system should minimize cost. 5. The system is able to hold a high load capacity. 6. The system autonomously navigates from one place to another. 7. The system has object avoidance. 8. The system includes an easy to use user interface. 9. The system should be able to meet load requirements.			

2.5 House of Quality

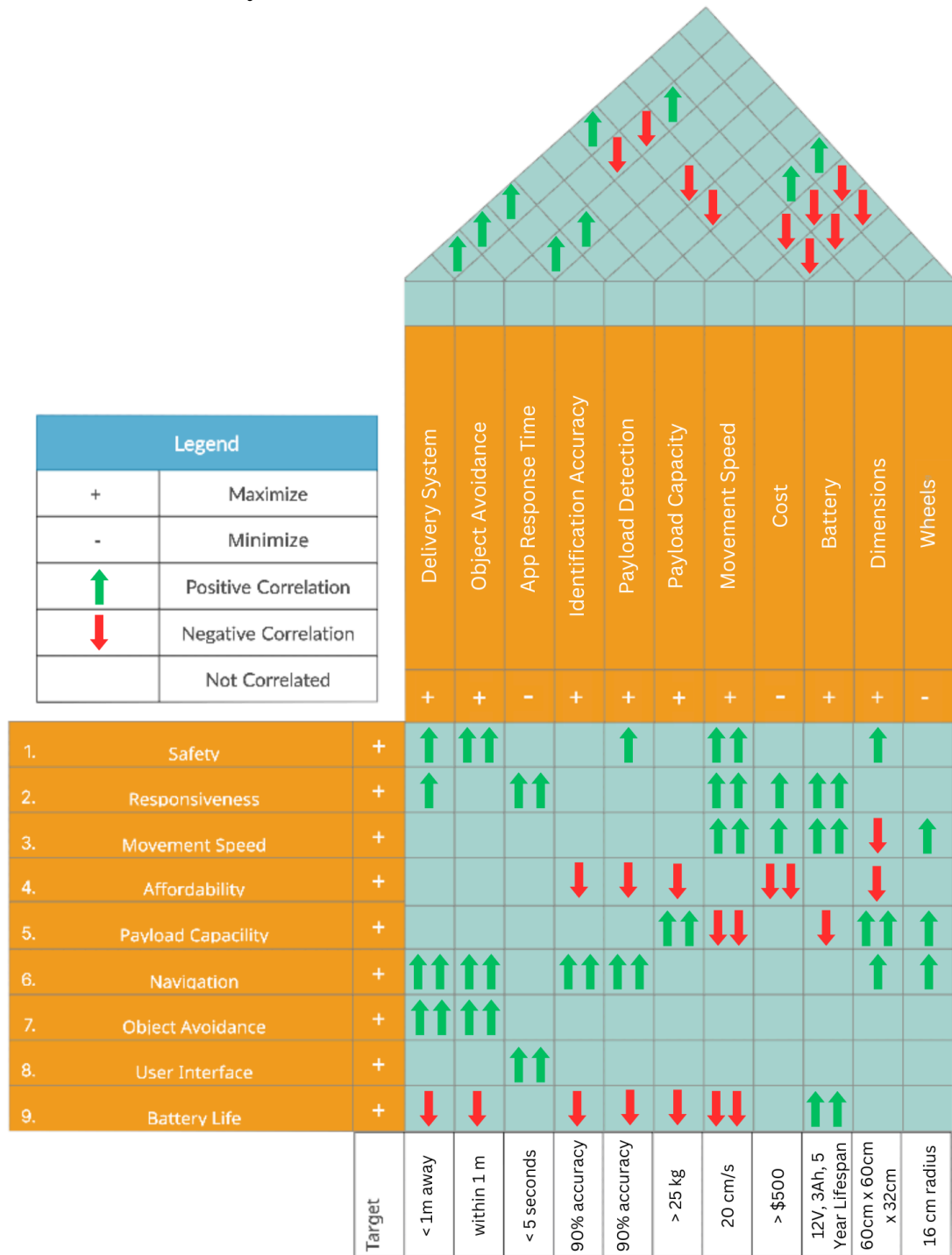


Figure 1: House of Quality by Authors

2.6 Project Block Diagrams

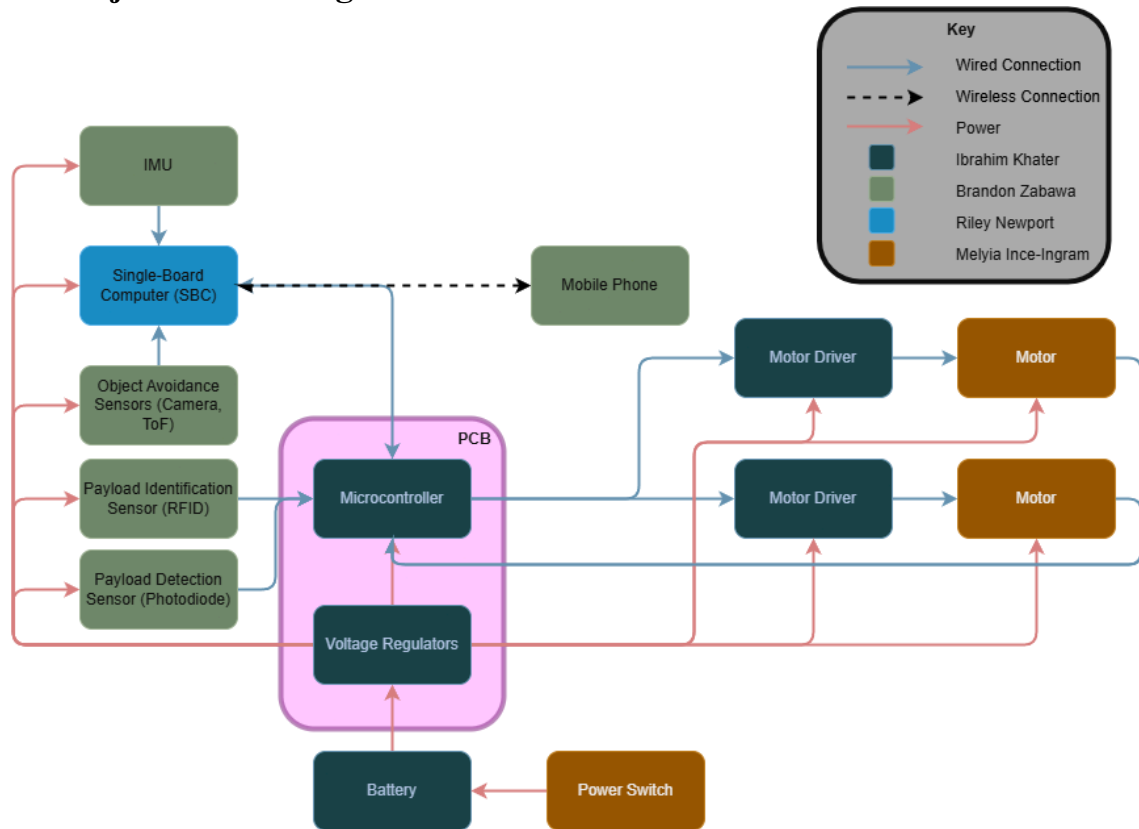


Figure 2: Hardware Block Diagram by Authors

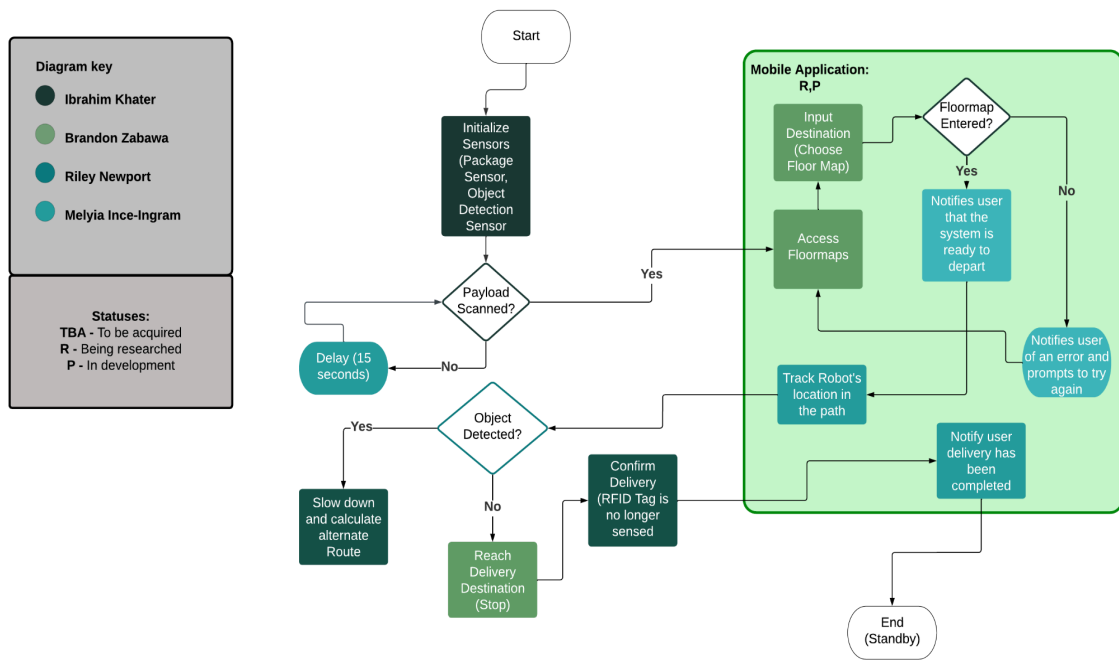


Figure 3: Software Flowchart by Authors

Chapter 3: Research and Investigation

3.1 Identification with RFID

The robot will include a mechanism to identify the name and location of each payload. There are several technologies that could help our robot achieve this including computer vision, barcode scanning, quick response (QR) codes, and Radio Frequency Identification (RFID).

Computer Vision is a type of computer science that uses a camera as the “eyes” and then derives information from Artificial Intelligence (AI), images and videos. It is used to help to identify and understand objects, gestures, or even expressions. Computer vision could be used to identify distinct objects. However, because of computer vision’s computational complexity and the fact that most payloads for the robot will look the same, it will not be a good fit for this project.

Barcodes can be found on almost every product we buy. They are a very inexpensive and simple way to label products. They operate by having a printed image of a barcode on a product or object. There are many different encoding systems but the most common is the Universal Product Code (UPC). In a UPC barcode the image contains 95 bars that are either black and white. The barcode is split into the first half and the second half with guard patterns, bars dedicated to indicating this separation. The remaining bars are grouped into 12 groups of 7. Each grouping of 7 will represent a digit 0 to 9. In the first half of the code the white bars on even spots and the second half of the code have white bars on odd spots. This is important for when the reader scans the barcode upside down it will be able to determine which way it is supposed to face. The first three numbers of the numerical code are associated with the country the company is located in, then the numbers indicate the company, then the product. How many numbers associated with the company and the product varies with the more expensive company codes consisting of less numbers to make room for more products [A.6]. The barcode reader is made of the scanner and the decoder. The scanner points a beam of light across the barcode and measures the amount of light that is reflected; it will then decode the pattern of black and white lines and send this information to a computer to process what information is associated with the code [A.6], [A.7].

QR codes can be found everywhere from restaurant menus, business cards to inventory management. They have their own encoding patterns with white and black blocks called data modules instead of bars but they use all the same principles as simple barcodes. Barcodes would be considered 1 dimensional because the scanners only have to oscillate the laser beam from left to right. QR codes would be considered 2 dimensional because the scanner has to oscillate up and down and left to right. Barcodes only encode numbers but QR codes can store numbers, letters, and symbols. Additionally QR codes can store significantly more data because there are more data modules in a QR code than bars in a barcode [A.8].

Radio Frequency Identification or for short RFID is a widely used technology that has been seamlessly integrated into our daily lives. It is used everywhere from preventing theft at retail stores to animal tracking to our government issued passports. The RFID system includes the RFID reader and the RFID tag that is placed on an object, person, or animal. It then uses radio waves sent from the reader to the tag. As long as the tag is in range of the reader it will activate and send a signal back to the reader which interprets the data. RFID readers can be found as handheld scanners or small stationary systems on doors like hotel rooms, or large readers at the doors of retail stores. Readers have an RF signal generator, a microcontroller, and a receiver/signal detector. The RF signal generator transmits radio waves via the antenna. The receiver/signal detector is there to receive the signal from the RFID tag. The microcontroller is there to process this incoming data from the tag. Tags consist of a transponder, rectifier circuit, controller, and memory. The transponder receives the signal from the reader and sends a signal back to the reader in response. The rectifier circuit allows for the power from the reader to be used as a supply for the controller and memory in the tag. The controller processes and manages the communication from reader to tag. And lastly, the memory stores the user-defined identification data. RFID does not require the reader to be in line of sight of the tag and instead only needs to be within range making this technology less human resource intensive than barcodes [A.9], [A.10].

RFID and Barcodes and QR codes seem like the best technologies to achieve the identification feature of the robot. The following table shows the major features, advantages and disadvantages of Barcodes and QR codes compared to RFID [A.6], [A.7], [A.8], [A.9], [A.10].

Table 2: Comparison of Barcode/QR Code Technology and RFID Technology

Barcodes/QR Codes vs. RFID	
Barcodes/QR Codes	RFID
Direct line of sight	Without direct line of sight and through objects
Closer proximity	Farther proximity
Data is read only	Data is read and write
Only identify the type of item	Can uniquely identify each item tagged
Read only one at a time	Read multiple tags simultaneously
Read time > 500ms per tag	Read time is < 100ms per tag
More subject to wear on payload	Less subject to wear on payload
Less costly than RFID	More costly than Barcodes/QR Codes

Both of these technologies could be used to identify payloads however RFID will be better for achieving an autonomous robot. RFID will not require a person to scan the payload before placing it on the robot like a barcode would. Other advantages of RFID is the read time is much less than the barcodes, data can be updated, and payloads can be uniquely marked. It will also lay a strong foundation for future improvements aimed at achieving the stretch goal of detecting multiple payloads simultaneously. Although barcodes are more inexpensive than RFID, RFID is still an affordable option that will fit within our budget constraints.

There are three different types of RFID tags: Passive, Active, and Semi Passive and the differences can be seen in the following table [A.9], [A.10], [A.11].

Table 3: Comparison of the Types of RFID Tag Technology

Types of RFID Tags		
Passive	Semi Passive	Active
Components: Integrated Circuit, Antenna	Components: Integrated Circuit, Antenna, Battery	Components: integrated circuit, antenna, battery, on-board transmitter
Receives power from the reader through inductive coupling	Powers its circuitry and processing capabilities, still relying on the reader to activate and transmit data	transmitter sends energy directly to the reader
Shortest range	Medium range	Longest range
Small size	Small size	Larger size
Typically least expensive	Medium price	Most expensive
Longest lifespan	Medium lifespan	Shortest lifespan

Of the three types of RFID tags passive tags will work best for this project. Passive tags will not require any battery and are the least expensive so they will be the most scalable option. The semi passive and active tags would provide a longer range but that is unnecessary because the tag will be put close to the reader when it is placed on the robot.

Passive RFID tags come in three frequency ranges: Low Frequency (LF), High Frequency (HF), and Ultra High Frequency (UHF). The key differences are outlined in the table below [A.10], [A.12].

Table 4: Comparison of the Frequency Ranges of Passive Tags

Frequency of Passive Tags

Low Frequency	High Frequency	Ultra High Frequency
Available in 125kHz, 134kHz	Available in 13.56 MHz	Available in 860–960 MHz
near-field inductive coupling	near-field inductive coupling	far-field radiative coupling also known as backscatter coupling
read range of a few inches	Read range less than 3 feet	Read range of 15 to 20 feet
lowest data transfer rate	Medium data transfer rate	Highest data transfer rate
Smaller memory in comparison to HF	Some can store up to 4K	Smaller memory in comparison to HF
No anti-collision → can't read multiple simultaneously	Some have anti-collision, most do not have anti-collision	All have anti-collision → allowing multiple tags to be read simultaneously
expensive to manufacture	lower in cost than LF tags due to simpler antenna design	UHF readers are more expensive than HF readers

This project will use a passive High Frequency RFID tag and reader system because it will be the most affordable option and has a sufficient read range needed for our robot.

High Frequency RFID readers use Near Field Coupling also known as Inductive Coupling as the method of transferring power from the reader to the tag. Inductive coupling works when the tag's coil comes within range of the magnetic field of the reader's coil. This creates a current through the tag's coil. The reader and tag are now coupled and when there is a change in the current flow of the reader it induces a current in the tag. The reader communicates with the tag through load modulation. The reader varies one aspect of the carrier wave (amplitude, frequency, or phase) and the tag detects the current changes in the tag's coil. The tag communicates with the reader by varying the impedance across the load. When the tag switches the load on and off it creates sideband frequencies which are coupled with the reader's antenna. The distance of a near field coupling zone can be approximated with $\lambda/2\pi$. So for a High Frequency RFID system operation at 13.56 MHz can be estimated to have an operating distance of 3.5 meters which can be seen by following allowing in Equation 1 and 2 [A.10], [A.13], [A.14].

$$\lambda = \frac{c}{f} = \frac{3 \cdot 10^8}{13.56 \cdot 10^6} = 22.1m$$

Equation 1: Finding Wavelength when Frequency is 13.56 MHz

$$\frac{\lambda}{2\pi} = \frac{22.1m}{2\pi} = 3.5m$$

Equation 2: Estimation of Operational Distance for High Frequency RFID Systems

There are a few standards that High Frequency RFID tags follow. The two main ones that operate at 13.56MHz and are widely available are the ISO15693 or ISO14443A/B standards. There are a few key differences that have to be taken into account before choosing a RFID module. ISO15693 is typically used for industrial applications such as access control, asset tracking and inventory control. In order to achieve a longer read range they have a slower baud rate. ISO14443A/B is particularly useful in financial transactions and focuses on very short ranges in order to achieve a safer transaction. The shorter range allows for higher baud rates. Table 5 below compares the main differences in these standards [A.26].

Table 5: Comparison of ISO15693 and ISO1443A/B Standards

Compare ISO RFID Tag Standards	
ISO15693	ISO1443A/B
Read Range: 1 m	Read Range: 7-15 cm
Data Transfer: 26k baud rate	Data Transfer: 106k baud rate
Frequency: 13.56MHz	Frequency: 13.56MHz

For this project ISO15693 or ISO1443A/B would both work. This project does not have a need for high security or fast data transfer rates that the ISO1443A/B, however it is helpful to have a longer range that ISO15693 has so the payloads do not have to be placed very specifically. It seems that ISO15693 would work better but ISO1442A/B would be sufficient for this project as well. In the next step of choosing an RFID module I looked at modules that supported both. Table 6 shows the Reader and Reader/Writer modules that we debated upon choosing for this project.

Table 6: Comparison of High Frequency RFID Reader Modules

High Frequency RFID Reader Modules			
CN0090 [C.4]	RYRR10S [C.5]	RYRR20I_DE [C.6]	Pepper C1 Module [C.7]
Read Only	Read Only	Read Only	Read/Write
Tags: ISO14443a	Tags: ISO15693, ISO14443A, ISO14443B	Tags: ISO14443A, ISO14443B, , ISO15693, FeliCa, ISO18092	Tags: ISO14443A, ISO15693
Range: few cms	Range: 4cm to 8cm	Range: 4cm	Range: up to a meter

Communication Protocols: I2C, Serial UART, SPI	Communication Protocols: UART, SPI	Communication Protocols: UART	Communication: UART, Wi-Fi
\$5.60	\$6.00	\$15.00	\$32.14

The Pepper C1 Module is a read/write module that supports ISO15693 and will have a read range up to a meter. The downside of this module is the price of \$32.14 which is significantly more than the other modules. In order to avoid this extra cost the other modules are read only. One of our group members already owns a module that has the functionality to write with ISO14443A/B. The robot's specifications only require that it can read the RFID tags. By using our group member's writer module this would allow us to save cost on the module for the robot. The CN0090 module would be the most cost effective choice however there are none in stock and there is a significant lead time. RYRR10S has a read range of 4 cm to 8 cm which is unideal but not outside of any specifications set. RYRR20I_DE provides no additional benefits over the RYRR10S but costs triple the price. The final choice for the RFID module is the RYRR10S.

3.2 Distance Sensor

The robot will include a distance sensor in order to implement the object avoidance requirement. The types of devices to consider for achieving object avoidance for this project are ultrasonic sensors and optical time of flight sensors or Light Detection and Ranging (LiDAR) technology.

Time of Flight (TOF) sensors measure distance between the sensor and an object with either light or sound waves. Ultrasonic sensors use the direct time of flight principle using sound waves. They measure the distance by measuring the time it takes for the sound wave to reach an object and return to the sensor. This is divided by 2 to find the time it took to go one way and then multiplied by the speed of sound. This can be found in Equation 3 below [A.17].

$$Distance = \frac{Time * Speed of Sound}{2}$$

Equation 3: Time of Flight Principle (Ultrasonic Sensors)

Ultrasonic sensors use sound frequencies in the ultrasonic range, from 20 kHz to 200 MHz which is above the frequency that humans can hear. Ultrasonic sensors are a form of transducers that convert sonic signals to electric signals. They can be either piezoelectric transducers or capacitive transducers. Piezoelectric transducers convert a mechanical energy of vibrations or pressure into electrical energy. In the case of ultrasonic sensors, they use a crystal substance that when a charge is introduced causes the particles to move and a diaphragm to vibrate. This vibration of the diaphragm generates the sound waves or vibrates in response to sound waves and converts it to electrical signals. A capacitive transducer measures the change in capacitance in response to the physical displacement of a diaphragm. They will consist of two conductive plates separated by a dielectric

material. Capacitance is calculated based on the distance between the two plates so when the sound waves move one of the plates and changes the distance between them it also changes the capacitance. The capacitance is converted to an electrical signal representing the sound waves. This process happens in reverse to emit the sound waves [A.17].

There are a few ways an ultrasonic sensor can fail to detect an object. The object must be within range of the ultrasonic sensor and must be large enough. If the sensor is mounted too low to the floor it might give off false positives. An object may also not be detected if it is in the blind zone, too close to the sensor. This happens with ultrasonic sensors that have one element that both transmits and receives. If the object is too close the sound waves reflected back will not be detected because the sensor is still transmitting and not receiving yet. Lastly, ultrasonic sensors will have less accuracy with objects that absorb sound waves such as fabrics or fluffy materials. Ultrasonic sensors are good because they are not sensitive to different colors or the amount of light in the room [A.17].

Optical TOF sensors have a transmitter, typically an LED or a laser diode, and a receiver, a photodiode. With direct TOF the light pulses are usually short with high intensity. The formula is the same as the one that one used with ultrasonic sensors; however in the case of optical TOF this equation is multiplied by the speed of light instead of sound and can be found below in Equation 4 [A.20].

$$Distance = \frac{Time * Speed\ of\ Light}{2}$$

Equation 4: Direct Time of Flight Principle (Optical TOF Sensors)

Indirect TOF uses low intensity modulated light pulses. The period of the pulses is determined by the modulation frequency. The phase delay between the transmitted light and received light is measured and represents the round trip [A.20].

$$Distance = \frac{Speed\ of\ Light * Phase\ Shift}{4 \cdot \pi \cdot (Moduled\ Frequency)}$$

Equation 5: Indirect Time of Flight Principle (Optical TOF Sensors)

TOF sensors typically operate at wavelengths in the near infrared range between 850nm to 940 nm [A.22].

LiDAR sensors can be seen as very similar to TOF sensors because they also use light to measure distance. LiDAR is a much more complex technology than optical TOF sensors. They emit rapid pulses of laser light and measure the time it takes for each pulse to return whereas optical TOF sensors emit a continuous light signal.

For LiDAR's optical distance measurement principle it can either use the indirect TOF or direct TOF principles as described above, or Frequency Modulated Continuous Wave (FMCW). Using TOF principles in LiDAR systems can be subject to interference with bright lights or other LiDAR systems. They typically operate in the near infrared range somewhere between 850nm and 905nm. This range can pose a concern for eye safety. A better range would be around 1550 nm because it would be safer for eyes and have less sunlight interference. The FMCW works with the laser source being a continuous wave

and the frequency is modulated with a “chirp” waveform. The received signal is mixed with a reference of the source frequency. This results in longer ranges, lower power emitted, and higher immunity to ambient light than using the time of flight principle. They mix the transmitted and received signal and then use a fourier transform to determine range. FMCW has a much higher detection and accuracy. They can be 10x more sensitive with 1000x lower power than LiDAR sensors that use TOF principles [A.18], [A.19].

For a LiDAR’s scanning architecture it can be a single emitter and detector array, emitter array and detector array, or scanning mirrors. The single emitter and detector array is the most common approach with the indirect TOF and direct TOF approach. Because there is a limit on the maximum optical power that can be transmitted, the range is limited. The emitter array and detector array also known as the True Solid State (TSS) LiDAR systems can overcome this problem by illuminating the field-of-view in sections. This means the available optical power can be more concentrated in each pulse. In the scanning mirrors architecture is used for the long-range LiDAR systems with a single source and detector for optimized range and precise timing. They can be supported with techniques like FMCW and various scanning methods, including a MEMS mirror and one dimensional rotating polygon mirrors [A.18].

To easily illustrate the main differences in optical TOF and LiDAR sensors refer to the table below [A.21].

Table 7: Comparison of Optical TOF and LiDAR Technologies for Clarification

Optical TOF vs. LiDAR	
Optical TOF	LiDAR
infrared light	powerful laser light sources
short to medium-range distance	Longer range and more detailed, high-resolution spatial data
Continuous light signal	Pulsed laser light
cost-effective and less complex than LiDAR systems	higher-resolution and more precise

The following table compares the major differences of ultrasonic sensors, optical TOF sensors, and LiDAR sensors with a focus on key characteristics that would come into play in this project [A.17].

Table 8: Comparison of Ultrasonic, Optical TOF, and LiDAR Technologies

Ultrasonic vs. Optical TOF vs. LiDAR		
Ultrasonic	Optical TOF	LiDAR

Less accurate with objects that absorb sound waves	Independent of object material	Independent of object material
Independent of target colors and reflectivity	Possibly can be affected by target colors and reflectivity	Better performance than optical TOF but can still be affected by target colors and reflectivity
Ambient immune	Ambient interference	Ambient interference, FMCW is ambient immune
few cm to several meters	few mm to several meters	several meters to hundreds of meters
Slower response time	Fast response time	Fastest response time
Moderate accuracy	High accuracy	Highest accuracy
Low power consumption	Low power consumption	Usually higher power consumption than Ultrasonic and TOF
Least expensive	More expensive by a little	Most expensive and by a large margin

From the information gathered about each of these sensors optical TOF sensors will be best for this project. Optical TOF will provide higher accuracy and faster response times than ultrasonic sensors. They will cost slightly more than ultrasonic but are still inexpensive and within our budget. Any LiDAR sensor would be outside of our budget but could possibly be implemented in a future iteration of this project. Many parts were looked at while choosing which part to choose, but the parts that seem to be able to accomplish our project specifications and goals are presented in Table 8 below [C.1], [C.2], [C.3].

Table 9: Comparison of Optical TOF Sensor Models

Optical TOF Models		
AST1024	SEN0245	410-392
Up to 2m	0.03-2m	Up to 5m
VL53L0X by STMicroelectronics	VL53L0X by STMicroelectronics	ISL29501 by Renesas
940 nm emitter	940 nm emitter	860 nm emitter
ROHS3 Compliant	ROHS3 Compliant	ROHS3 Compliant

Eye safe	Eye safe	Eye safe
I2C, Serial	Gravity-I2C	I2C
\$9.95	\$12.90	\$19.99

This project will use the AST1024 part. The part 410-392 is more expensive but can read up to 5m. For this project that distance is unnecessary as our design specifications only need the robot to identify obstacles within 1m of the robot. For these reasons part 410-392 is not a good choice. Part SEN0245 uses Gravity-I2C and is a part of the Gravity series by DFRobot. Up to 8 of any gravity modules can be plugged into the Gravity I2C HUB for an easy plug-and-play option. In order to use this part we would have to purchase this Gravity I2C HUB part in addition to the optical TOF module [C.2]. This part is already more expensive than the AST1024 so for those reasons this part is not a good choice. That leaves AST1024 which fits all our specifications. It uses I2C, is eye safe, and has a range up to 2m.

3.3 Payload Detector

The robot should be able to identify if there is a payload present on its surface. Some ideas for what type of sensor or detector that could be used are another optical TOF sensor, a pressure sensor, or a photoresistor.

We could use an optical TOF sensor like we use for the object avoidance to detect if there is a payload on the robot. An optical TOF would be able to measure how far away an object is and if this was mounted to the top of the robot it would be able to know if there is a payload there or not. However, we do not actually need to know how far away an object is just if there is an object present or not. Also optical TOF sensors are active and require more power than a passive sensor. For these reasons an optical TOF sensor is excess functionality and not worth the cost or power. A cheaper passive sensor would be preferable.

The second option for this detector would be to use a pressure sensor. If a pressure sensor is mounted to the top of the robot it would be able to detect the change in pressure from when there is nothing to when there is a payload. This option could run into trouble with extra lightweight payloads. If the payloads do not provide enough force onto the pressure sensor it would not detect properly. One type of pressure sensor that would work well for this would be force sensing resistors. These are a type of passive sensor using variable resistors. They measure the amount of resistance related to the applied force. When more force is applied to the sensor the resistance is lower. These sensors use conductive silver and carbon inks printed between two layers of thin, flexible material. The sensor sends a small electrical current through each of the two surfaces. When you press on the pad, it makes a connection between the layers and gives a signal. The higher the force, the more the conductive inks connect, and the electrical resistance lowers [A.24]. The larger the sensing pad is, the more expensive these are. So in this case we would use either one large pad or multiple smaller pads to achieve coverage of the entire robot.

The next option is to use photoresistors. Photoresistors are a type of passive sensor using variable resistors. The resistance will decrease when the intensity of light increases. The way this works is that when the light energy applied to the photoresistor increases, some of the valence electrons absorb energy from the light and break the bonding with the atoms. This increases the number of charge carriers generated in the photoresistor and as a result, the electric current flowing through the photoresistor increases. An increase in electric current means a decrease in resistance [A.23]. Photoresistors would have the same problem that force sensing resistors would have with coverage of the surface of the robot. Multiple photoresistors across the surface would be necessary to achieve full coverage.

The last option to use is a photodiode which is a semiconductor device with a P-N junction that converts light into electrical current. The P layer is the positive side with an abundance of holes, and the N layer is the negative side with an abundance of electrons. With this device when the brightness of light increases, so does the current in the device A[87].

Table 10: Comparison of Payload Detection Sensors

Comparison of Payload Detection Sensors			
Optical TOF	Force Sensing Resistor	Photoresistor	Photodiode
Measures distance	Measures change in resistance	Measures change in resistance	Measures change in current
Output: distance measurement	Output: voltage	Output: voltage	Output: current
High Cost	Medium cost	Lowest Cost	Low Cost

Force sensing resistors, photoresistors, and photodiodes are all good options for payload detection. Force sensing resistors and photoresistors both work by measuring the change in resistance in relation to force or light respectively. They all face the same challenge of needing to have multiple to cover the surface area of the robot. However photodiodes will be a better option because they offer faster response times and higher sensitivity compared to the other options. Additionally, photodiodes can operate in different modes for more precise and reliable detection. Their ability to work with amplifiers further enhances their accuracy, making them an ideal choice for real-time sensing.

There are three types of photodiodes that must be considered: P-N, PIN, and ADP. P-N diodes are the basic diode and work as described in the above photodiode paragraph. PIN diodes work like basic diodes but include an intrinsic layer placed between the P and N layers. This layer helps increase the strength of the electric field in the diode which

increases the depletion layer. PIN also has a lower capacitance at the junction which increases the speed of the photodiode.

Table 11: Comparison of Types of Photodiodes [A.87]

Types of Photodiodes		
P-N	PIN	APD
Basic photodiode	Intrinsic layer placed between P and N layers	impact ionization for a larger current generated per photon
Basic response speed	higher response speed from the increased depletion region	highest response speed
Good Reverse Bias	Best Reverse Bias	Good Reverse Bias
Poor in Low Light	Good in Low Light	Best in Low Light
Most inexpensive	inexpensive	Most expensive
Low noise	Least noise	Poor noise but has a higher signal-to-noise ratio

PIN diodes are best for this project because they have a higher response speed than the basic diode but are still inexpensive. APD diodes would not be a necessary expense or their functionality would not be fully utilized.

3.4 Visual Sensors

We decided toward a monocular camera as that would get us the basic feature analysis that we need while navigating. We thought this would be the most cost-effective way for us to still be able to use VSLAM without relying heavily on the camera functions. The other camera types we did consider were stereo and RGB-D cameras which did prove to be less cost-effective to go with and more unnecessary in terms of functionality.

3.4.1 Camera Types

For monocular cameras they are a singular lens that takes in 2D visual information typically used in low cost solutions alongside it being compact and lightweight. While it doesn't seem too complex to build, it still can achieve its purpose in terms of gathering information regarding depth estimation. Another camera is the stereo camera which uses two lenses that function as 2 sights that capture an image from slightly different angles that will create a 3D effect. Lastly the most advanced out of all of these cameras are the RGB-D camera which captures depth and merges the color data to output real time information.

All of these cameras use a lens to focus light onto an image sensor, which then converts the light into electrical signals. These signals are processed by the camera's software to create a digital image or video. For this project, we will be using a monocular camera since we will not be solely relying on the camera for sensor information. As opposed to using the stereo and RGB-D we wanted to stay as much in budget as possible. One of the key advantages of a monocular camera is its compact size and simplicity. It requires only one lens and sensor, making it more cost-effective and easier to integrate into various devices [A.60].

Monocular pure visual techniques have become attractive solutions for 3D reconstruction tasks due to their affordability, lightweight, easy deployment, and most of all availability [A.60]. When choosing the right camera, we also had to take into account how it would interface with the raspberry pi and it would be compatible with connecting to our other sensors. We have to consider factors such as the Lens type which would determine if we would be able to adjust the focus of the image. The shutter type that determines the speed or rate the video would be captured rolling shutter are mainly for slower motion capture and global are for high-speed motion capture. The resolution would be about 1080p or less in order to gain optimal feature extraction without compromising our budget. As well as the field of view the range at which the camera can cover and the Infrared Sensitivity (IR) for reacting to visible light.

3.4.1.1 Raspberry Pi V2

The Raspberry Pi V2 Camera is one of the leading lightweight, compact, and low-cost monocular camera that is specifically designed for the Raspberry Pi. It contains an 8-megapixel Sony IMX219 image sensor capable of capturing high-resolution still images at 3280 x 2464 pixels and video at up to HD video and is suitable for low-end imaging applications. The camera connects via a MIPI CSI-2 interface, which is the main interface of the Video Core GPU (Raspberry Pi's graphics component). It uses a fixed-focus lens and has a rolling shutter [A.21], which is generally sufficient for static or low-motion areas which is ideal since SLAM would operate mostly in an indoor environment.

It provides a cost-effective means of utilizing VSLAM for implementations such as ORB-SLAM3 which we will be using for this project. With a resolution of moderate quality, it provides a means of extracting sufficient features for operation under conditions of well-lit, textured environments to be best suited for tasks of simple navigation. However, it does not enable high-end capabilities like native HDR capability, low-light enhancement, or native infrared sensitivity, which can slow it down under low-light or high-contrast situations. Aside from these restrictions, its tiny size, low power consumption, and natively enabled Raspberry Pi ecosystem compatibility make it attractive for prototyping and the development of lean robotic systems.

3.4.1.2 Arducam 2MP

The Arducam 2MP was another option for monocular cameras that were widely available and cost-efficient for the features it was offering. It contains a slightly more updated Sony IMX image sensor with a lower resolution than the Raspberry Pi V2 camera which was seen as a drawback in terms of performance. It also contains a rolling shutter that has frame rates of 60fps when capturing slow motion video[A.22].

Additionally, the camera features a native Infrared sensitivity which allows it to fare decently in low light conditions but does not have the best recorded performance in terms of how we want the robot to fare in different lighting settings. In the context of the project, the Arducam 2MP was initially considered as a top choice since it has better performance under low-lighting and it will interface directly with the Raspberry pi 5. All of these attributes made it a contender when using it alongside ORB-SLAM however the low resolution, cost and other specifications deterred us from this option. The diminished resolution compared to the other cameras we explored can deteriorate the performance of feature detection and mapping for SLAM. The Arducam 2MP proved to be more costly relative to the other camera modules which was a substantial drawback.

3.4.1.5 Arducam 12MP IMX708

The Arducam 12MP is a successor to the Arducam 2MP. With the more advanced Sony IMX708 image sensor at 12MP resolution, a significantly higher resolution than the Pi V2 and the Arducam 2MP. It still provides the same HD image quality that supports around 60fps for smooth video capture.

It also possesses a up to 120 degrees of field of view, which is a major benefit to feature detection and environment mapping within VSLAM systems. Even though the Arducam is more advanced in comparison to the other camera modules, we did not end up going with this module due to the constraints and limitations set up for this robot.

Even though the high resolution and wide-angle provide detailed images, these are made at the cost of additional computational load that could stress the processing capability of the Raspberry Pi 5 during real-time ORB-SLAM operations. Higher resolution images require more processing and memory, potentially resulting in frame drops or delays within a real-time SLAM pipeline. Another factor is the higher cost of this camera compared to our other options and it did surpass our budget requirement. With these compromises between cost, complexity, and performance, we concluded that lighter-weight and SLAM-efficient camera solutions would be more appropriate to fulfill the functional and fiscal requirements of the project. Table 12 shows the overall comparison between these visual sensors and what we ultimately decided on.

Table 12: Visual Sensor Comparison [C.20,C.21,C.22]

Visual Sensors			
Specification	Raspberry Pi V2	Arducam 2MP	Arducam 12MP
Sensor Model	Sony IMX219	Sony IMX462	Sony IMX708

Shutter Type	Rolling Shutter	Rolling Shutter	Rolling Shutter
Focus Type	Adjustable Focus	Manual Focus	Fixed Focus
Resolution	8 MegaPixels	2 MegaPixels	12 MegaPixels
Video Quality	1080p47, 1640 × 1232p41 and 640 × 480p206	1920 × 1080	4608 x 2592
Pixel Size	1.12 μm × 1.12 μm	2.9 μm × 2.9 μm	1.4 μm x 1.4 μm
Frame Rate	Up to 30 fps (1080p)	Up to 60fps(1080p)	Up to 60fps, lower at higher resolutions
Field of View	62.2°x 48.8°	161°x141.4°x81.7°	120°x 102°x 67°
IR Sensitivity	NoIR variant, also works for low-light environments	Integral IR-cut Filter, visible light only	Integral IR-cut Filter, visible light only
Cost	15.00\$	35.99\$	39.49\$

3.5 Inertial Measurement Units

An Inertial measurement unit is necessary to provide motion tracking data for our VSLAM algorithm. It consists of a sensor suite of a gyroscope, accelerometer, and magnetometer sensors. The IMU in this way is can provide info to the the algorithm with regards to reference to the rate of angle (gyroscope) and acceleration (accelerometer) on the x-, y-, and z-axes and, also, the magnetic field around the device (magnetometer)[A.65]. The reasoning for using an Inertial Measurement Unit for this project was mainly for estimating structure and the current positioning for the robot. This would allow us to get even more accurate readings when the robot is traversing the environment.

3.5.1 6-Axis vs. 9-Axis IMUs

A 6-axis IMU consists of a 3-axis gyroscope and a 3-axis accelerometer. The gyroscope measures angular velocity, allowing the system to detect rotational movement, while the accelerometer measures linear acceleration, helping to track movement along the different axes. This combination enables the robot to estimate orientation changes and movement, which in turn aids in stabilizing the robot and detecting direction shifts. A 6-axis IMU would detect the robot's motion in all directions: forward, backward, left, and right. It also provides essential data for ORB-SLAM, ensuring smooth and accurate navigation.

A 9-axis IMU includes the same features of the 6-axis IMU but adds a 3-axis magnetometer. The magnetometer detects the Earth's magnetic field and enables absolute heading direction. The inclusion of the magnetometer is especially useful for precise navigation systems. A 9-axis IMU could provide the robot with additional heading information and more accurate movement.

While both IMUs would work in this project, the final decision must take into account factors such as complexity, cost, and the project's actual navigation needs. Given that the robot primarily relies on ORB-SLAM and ToF sensors for localization, the additional magnetometer included in the 9-axis IMU is not necessary. Adding a magnetometer also introduces more complexities and requires careful calibration. They are also prone to interference from nearby electronic components, so they must be designed and placed in such a way that it is not affected by interference. Overall, a 6-axis IMU was chosen to be used as the sensor for this project, as it meets the project's requirements without any added complexity.

Table 13: Comparison of 6-Axis vs. 9-Axis IMU

Feature	6-Axis IMU	9-Axis IMU
Sensors	3-axis gyroscope, 3-axis accelerometer	3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer
Complexity	Low	Moderate
Susceptibility to Interference	Low	High
Power Consumption	Low	Moderate
Cost	Low	Moderate

3.5.2 6-Axis IMU IC Comparison

After deciding to use a 6-axis IMU, three different 6-axis IMUs were further evaluated to determine the best fit for our project.

Table 14: Comparison of 6-Axis IMUs [C.36, C.37, C.38]

Feature	MPU6050	ICM-42688	LSM6DSO
Comm. Interface	I2C	I2C/I3C/SPI	I2C/I3C/SPI
Power Consumption	Low	Moderate	Low
Noise	High	Low	Low

Accelerometer Range (g)	$\pm 2, \pm 4, \pm 8, \pm 16$	$\pm 2, \pm 4, \pm 8, \pm 16$	$\pm 2, \pm 4, \pm 8, \pm 16$
Gyroscope Range ($^{\circ}/s$)	$\pm 250, \pm 500, \pm 1000, \pm 2000$	$\pm 15.625, \pm 31.25, \pm 62.6, \pm 125, \pm 250, \pm 500, \pm 1000, \pm 2000$	$\pm 125, \pm 250, \pm 500, \pm 1000, \pm 2000$
Cost	Obsolete	Moderate	Low

Before comparing the three IMUs, the MPU6050 can be immediately eliminated. The IC is no longer manufactured and is obsolete, meaning it cannot be bought easily. Thus, there is no need to continue considering it for the project. This now leaves the ICM-42688 and the LSM6DSO.

The ICM-42688 is a high performance IMU with good accuracy and low noise levels. It supports different communication protocols, which make it a versatile option. Similarly, the LSM6DSO is also high performance and has similar capabilities and characteristics with slightly lower power consumption. Even supporting the same communication protocols, the IMUs are very similar and either would work well for this project.

After fully comparing the three IMUs and their capabilities, the LSM6DSO was chosen for our project due to its overall balance of cost, performance, and power efficiency. While the ICM-42688 is also a viable option, it is more expensive and consumes more power than necessary for this application. The LSM6DSO offers a good balance between all of the desired features, maintaining a reasonable cost and power efficiency while still providing the needed precision.

3.6 Microcontroller

The microcontroller (MCU) is a fundamental component in any embedded system, providing essential control over hardware and enabling efficient automation. In this project, the MCU will serve as the core unit for handling real-time control tasks. Unlike a general purpose computer, an MCU is designed to efficiently manage low-level operations with precision timing, making it ideal for controlling the motors, interpreting the different sensor data, and ensuring smooth communication with the SBC. The MCU's primary role in this project is to interpret sensor data and provide control over the motors' movement, ensuring that the robot can navigate accurately and in real time.

Since the robot will need to be able to work autonomously during transport, the MCU will process real-time inputs from the sensors, such as the ToF sensors. It will need to work with motor drivers to control the motors and establish communication with the SBC. The selection of the MCU requires careful consideration of different criteria such as processing speed, available communication protocols, real-time processing capabilities, power consumption, functionalities, and cost.

These different criteria will directly impact the performance of the robot. The first consideration is the real-time processing capability. The MCU must be able to handle interrupts efficiently and execute code with minimal latency. Processing power is another major factor, as the MCU should be able to handle a variety of sensor inputs as well as motor control without significant delays.

Another important criteria is communication capabilities, as the MCU will need to communicate with different devices that have their own processor or MCU, such as the SBC and motor drivers. Protocols such as UART, SPI, I2C, and CAN are common in embedded systems and for motor applications. Selecting an MCU with multiple communication interfaces allows for more flexibility in system integration. Power consumption is another significant factor in MCU selection. The MCU must operate efficiently to maximize the robot's battery life. Additionally, there are other high-power draw components in the robot. A low-power MCU helps to extend operational time without compromising performance.

The different functionalities of the MCU are also crucial. For example, in order to properly control the motors, the MCU must have pulse width modulation (PWM) capabilities. The MCU must also have enough inputs and outputs (I/O) for the different sensors and other components. The I/Os should include both general purpose I/Os (GPIO) and pins with specific functions, such as PWM, analog-to-digital converter (ADC), and digital-to-analog (DAC). The final criteria is the cost of the MCU. An ideal MCU should be cost-effective while offering many different functionalities.

Table 15: Comparison of Microcontrollers [C.8, C.9, C.10, C.11]

Microcontroller	Processing Speed	Power Consumption	Communication Interfaces	Cost
ESP32-WROOM-32E	240 MHz	Medium (3-3.6 V, 0.5 A)	UART, SPI, I2C, Wi-Fi, Bluetooth	Low (\$2-\$8)
STM32F103C8T6	72 MHz	Low (2-3.6 V, ~50mA)	I2C, USART, SPI, CAN, USB	Medium (\$5-\$10)
ATmega328p	16 MHz	Low (2.7-5.5 V, ~20mA)	USART, SPI, I2C	Low (\$1-\$5)
PIC18F45K22	64 MHz	Very low (2.7-5.5 V, ~10mA)	EUSART, SPI, I2C	Low (\$3-\$5)

After evaluating the various MCU options, the ESP32 emerges as the most suitable option for this project. While the STM32F103C8T6 offers a robust real-time performance and low power consumption, the ESP32 provides significantly better processing speed, built-in wireless communication, such as Wi-Fi and Bluetooth, and a wide range of I/O capabilities at a lower cost. These advantages align well with the project's need for high responsiveness, communication capabilities with the SBC, and flexibility in system

integration. Its dual-core processor allows separate tasks and data to be run and processed at once. As a result, the ESP32-WROOM-32E is selected as the final choice for the project's MCU.

3.7 Motors

Motors play an important role in this project as the robot cannot function properly without the right ones, making their selection a critical aspect of the overall design. The motors must provide the necessary torque and speed to move the robot while ensuring precise control. Since the robot must be fully autonomous, precise control over the motors is a necessity. The MCU must be able to fine tune the motors and ensure they move the robot at a constant speed regardless of payload weight as well as precise direction changes.

The selection process for the motors includes criteria such as torque, power consumption, control precision, control complexity, and cost. The torque rating of the motors must be sufficient enough to carry the robot as well as any additional payload on top of it. Power consumption must be minimized to optimize the overall energy usage of the robot and prolong battery life. Precision control over the motors is needed to ensure smooth movement at a constant speed and allow for fine tuning. Control complexity also plays a role as some motors require more complicated drivers and feedback systems for operation. Finally, cost is a factor that ensures that the selected motors provide the best performance while staying within our limited budget.

Table 16: Comparison of Motor Types

Motor Type	Torque	Power Consumption	Control Precision	Control Complexity	Cost
Brushed DC Motor	High	High	Medium	Low	Low
Brushless DC Motor	High	Low	High	High	High
Stepper Motor	High	High	High	Moderate	Low
Servo Motor	High	Moderate	High	Moderate	Moderate-High

After evaluating the different types of motors, brushed DC motors were selected due to their ideal balance of criteria, namely their simplicity, cost-effectiveness, and ease of control. While brushless DC motors offer higher efficiency and longevity, they come with increased complexity and very high cost for this high torque application, making them impractical for this stage of the project. Brushed DC motors, on the other hand, provide

sufficient torque output and straightforward speed control through the use of PWM without requiring complex motor controllers.

Servo motors provide high precision and good torque control, especially in applications that require accurate positioning. However, their control complexity, power consumption, and price make them less appealing for the current phase of the project. They require more sophisticated control systems and separate feedback loops, which would increase development time and cost.

Although brushless DC motors would be ideal for a more advanced implementation, they require complex specialized drivers and precise control algorithms, significantly increasing both development time and cost. Stepper motors were also considered but were ultimately not selected due to their low efficiency and higher power consumption compared to brushed DC motors.

Overall, brushed DC motors offer a practical and cost-effective solution that meets the needs of the robot while minimizing design complexity. Their ease of implementation allows for faster and more efficient development and testing while allowing the team to focus on the more niche parts of the project, making them the suitable choice.

3.7.1 Brushed DC Motor Selection and Analysis

After deciding that brushed DC motors are the best option for this project, different types of brushed DC motors must also be compared before selecting a specific model. Four main types of brushed DC motors were considered: standard brushed DC motors, geared brushed DC motors, coreless brushed DC motors, and servo brushed DC motors. Each type offers its own unique advantages and disadvantages in terms of efficiency, torque, control complexity, and cost.

Standard brushed DC motors provide a simple and cost-effective solution with direct control over speed and torque using PWM. However, they lack the torque that would be needed for carrying payloads, making them less suitable for this project. Geared brushed DC motors offer increased torque in exchange for reduced speed. They are more efficient for load-bearing low speed applications such as this one. Gear motors are able to output very high torque in a compact size with a greater efficiency than spur gear motors.

Coreless brushed DC motors were considered for their lightweight design and precise operation. However, their lower torque output and higher cost make them impractical for this project. Servo brushed DC motors have the benefit of an integrated internal feedback system that allows for more precise position and speed control. However, their higher cost and complexity makes them less suitable for this project at this time.

Ultimately, geared brushed DC motors were selected as the optimal choice, offering a strong balance of torque, efficiency, cost, and ease of control.

3.7.2 Brushed DC Motor Comparison and Selection

After determining that geared brushed DC motors would be used in this project, the next step is to compare specific motors and to identify the most suitable option. The selection process involves key performance factors such as torque, speed, power consumption, control precision, and cost. By analyzing these criteria, the team will be able to select a motor that provides the necessary performance while maintaining efficiency and affordability.

Table 17: Comparison of Geared Brushed DC Motor Models [A.33, A.34, C.12]

Motor	Voltage (V)	Current (A)	Torque (Nm)	Speed (RPM)	Weight (g)	Control Precision	Cost (\$)
24PG-2430BL-1260-256	12	0.5-1.9	1-3	18.5-23	NA	None	19.69
Pololu 2828	12	0.2-5.5	0.5-5	67	210	Encoder	51.95
638266	12	0.54-1.2	2-4	20-26	380	None	39.99

After evaluating the different motor models, the team selected the Pololu 2828 as the most suitable option for this project. The motor has an ideal balance between performance, control precision, and cost. With a voltage rating of 12V and a current range of 0.2-5.5A, this motor is able to provide a wide torque range that can allow it to handle various payload weights. The speed also ensures that the motor will be able to move at a constant desired speed despite the payload size. The included encoder will also allow the MCU to receive feedback from the motors, which is essential for navigating and delivering payloads accurately.

While the Pololu 2828 is the most expensive of the three motors considered, the integrated encoder and the overall efficiency make it the best option. When taking into account its performance, the motor's weight is very reasonable. This in tandem with its fine control and high torque capabilities make it ideal for the robot. Thus, the Pololu 2828 brushed DC gearmotor provides the best combination of performance, control, and cost-effectiveness for the project's requirements.

3.7.3 Driving the Motor

To drive the motor selected in the previous section, a motor driver or an H-bridge integrated circuit (IC) is required to control the direction, speed, and power delivery to the motor. The motor operates at 12V and can draw varying currents based on the load, so it is important to select a motor driver capable of handling the required current without overheating or damage.

3.7.3.1 Motor Driver IC Selection

The motor driver IC should be able to handle the motor's voltage and maximum current draw. The Pololu 2828 has a voltage of 12V and a maximum current draw of approximately 5.5A. The IC must be capable of precise control using inputs from the microcontroller to control speed and direction.

Table 18: Comparison of Geared Brushed DC Motor Models [C.13, C.14, C.15, C.16]

IC	Voltage (V)	Output Current (A)	Protection Features	Control Interface	Price
TB9051FTG	4.5-28	5	Overcurrent, Overvoltage, Thermal Shutdown	PWM + DIR	Moderate
VNH5019	5.5-24	30	Overcurrent, Overvoltage, Thermal Shutdown	PWM + DIR	Moderate
DRV8871	6.5-45	4	Thermal Shutdown	PWM	Low
MC33926	5-28	5	Overcurrent, Thermal Shutdown	PWM + DIR	Moderate

Four different motor driver ICs were compared to determine the best fit for driving the motors on the robot. Starting with the DRV8871, it can immediately be eliminated due to its low output current and lack of directional control. MC33926 and TB9051FTG have similar characteristics, with the same voltage and current capabilities. However, the Pololu 2828 motors have a maximum current draw of 5.5A which might cause issues down the line when handling larger loads. Thus, this only leaves the VNH5019 as the remaining option.

The VNH5019 has superior current handling capabilities which well exceed the needs of the motors at a maximum current draw of 30A, ensuring that the motor will always receive sufficient power without the risk of overcurrenting the driver. Additionally, it includes multiple protection features that safeguard against any potential failures. The driver also allows for PWM control and directional control, allowing for precise movements and adjustments.

3.8 Single Board Computers

For our project, we needed to select which single board computer that will handle the computational capabilities of the planning and mapping algorithm that we will use. We

knew that a Raspberry Pi would be required to interface the mobile application with the user and be able to communicate with the robot via SLAM.

For any embedded system, our specifications require that we need to have a single-board computer that will use RTS and ROS for programming with SLAM. It needs to have an adequate amount of memory around 8GB for it to handle the storage information for the maps as well as the information gathered from our visual sensors. We had to consider the compatibility with the other components of our system such as IMU, TOF and the MCU. As for the mobile application we wanted it to be able to communicate over WiFi so there will be a stable connection with the single-board computer to provide real time updates via the mobile app. Below we compiled a list of components that we considered would be best in the context of autonomous navigation.

3.8.1 Raspberry Pi 5

The Raspberry Pi 5 is the most advanced model for the Raspberry Pi. It provides various capabilities that make it the top choice for our project.. We thought this model would be best mainly for taking on the computational load of VSLAM without compromising on performance. The Raspberry Pi 5 has a quad-core 64-bit Cortex-A76 processor at 2.4 GHz, offering a significant performance boost over prior models to the Pi,, along with up to 8GB of LPDDR4X RAM, a new VideoCore VII GPU, and faster USB 3.0 and microSD interfaces [17]. While these specifications make it well-suited for demanding applications like edge computing, media processing, or AI inference, they exceed the computational requirements of this project, which relies primarily on efficient implementation of ORB-SLAM3 and basic sensor data processing.

However, we do have to acknowledge the drawbacks which will be the increased power consumption and thermal output requiring us to consider the positioning of the robot as well as the capacity of the battery when in operation. Additionally, the higher cost of the Pi 5 compared to the Pi 4B can strain the project's strict \$500 budget, especially when paired with other required hardware components like motors, sensors, and the RFID system. Because the core functions of this project, such as SLAM navigation, RFID-based payload identification, and mobile app communications, can reasonably be accomplished with the Raspberry Pi 5, we opted in to using this in our main project to satisfy the requirements of our software components.

3.8.2 Raspberry Pi 4B

The Raspberry Pi 4B was one of the primary options when considering the computational load of the SLAM algorithm that we were going to use. The model B comes with additional features than the traditional model 4 that we felt was relevant to consider. It features a quad-core Cortex-A72 processor at 1.5 GHz, good enough to handle 2D SLAM algorithms and RAM up to 8GB LPDDR4[C.16]. The amount of memory is one of the bigger factors when considering feature extraction and image processing as the robot navigates around the environment. Moreover, VideoCore VI GPU supports OpenCV and hardware acceleration which would be required for interfacing with our camera module.

We also need to consider whether it is compatible with Wi-Fi to communicate with the robot and mobile application. The Raspberry Pi model 4B has dual-band Wi-Fi (2.4GHz) and 5ghz IEEE 802.11b/g/n/ac wireless LAN, with Bluetooth 5.0 capabilities as an alternative. This made it a leading contender for use in our project since we do require Wifi to communicate with our mobile application. In terms of input power, it will only need to operate on 5 V DC or any GPIO header with a minimum of 3A amps that we have to account for in combination with the motors and microcontrollers. Despite its optimal features regarding using alongside SLAM to navigate, we had to decide against it due to the fact that since we are trying to handle the computational load of a full sensor suite and a visual slam algorithm this was eliminated as an option for the project.

3.8.3 Nvidia Jetson Nano

The Jetson Nano is equipped with a quad-core ARM Cortex-A57 CPU and a 128-core Maxwell GPU, making it highly capable of running deep learning and computer vision algorithms, including complex SLAM systems and neural network-based object detection. It also supports 4GB of LPDDR4 RAM, HDMI output, USB 3.0, and GPIO expansion headers, which make it highly flexible for robotics applications. However, one of its primary limitations in the context of this project is its higher power consumption, typically around 5–10 watts, which is significantly more than that of a Raspberry Pi 4B, thus demanding a larger and more expensive power supply or battery system.

Since, the Jetson Nano lacks onboard Wi-Fi and Bluetooth, requiring additional peripherals that increase both the cost and complexity of the system. Based on our individual skill sets and the timeline we are on, It was necessary to reduce complexity as much as possible in order to achieve the goals of our project. The board is also significantly more expensive than a Raspberry Pi 5, and while its GPU offers advantages for AI applications, the project's SLAM implementation using ORB-SLAM would not need an interaction with AI and it would be unnecessary. Given the project's focus on affordability, energy efficiency, and ease of integration with lightweight sensors like ToF and IMU, the Jetson Nano introduces more hardware overhead than benefit, making it a less suitable option for a cost-effective, power-conscious indoor delivery robot system.

3.8.4 Arduino Mega

The Arduino Mega 2560 is a widely used microcontroller for basic embedded systems and sensor interfacing. The Arduino Mega features an 8-bit ATmega2560 microcontroller running at 16 MHz, with 256 KB of flash memory, 8 KB of SRAM for low-level control tasks like motor driving, simple sensor input, and digital communication. However, it lacks the processing power, memory capacity, and operating system support required to run complex visual algorithms like ORB-SLAM, image processing libraries like OpenCV, or middleware systems like ROS2.

The Arduino Mega does not support native networking or high-speed data interfaces, meaning that Wi-Fi or Bluetooth communication must be implemented via external modules, which introduces additional complexity and reduces reliability. The platform also lacks the ability to interface directly with cameras or process video data in real-time,

which is a fundamental requirement for SLAM-based navigation. While it is an excellent option for simple automation or control systems, it is not equipped to handle the computational demands, real-time decision-making, and high-bandwidth sensor fusion required for autonomous robotics, making it unsuitable as the primary processing unit for our project.

Table 19: Comparison of Single Board Computers [C.16, C.17, C.18,C.19]

Single Board Computers				
Specifications	Raspberry Pi 5	Raspberry Pi 4B	Nvidia Jetson Nano	Arduino Mega
CPU	2.4GHz quad-core 64-bit Arm Cortex-A76 CPU	Quad-core Cortex-A72 at 1.5GHz	ARM Cortex-A57 MPCore (Quad-Core)	Quad-core ARM Cortex-A57 at 1.43GHz
RAM	Up to 16GB LPDDR4X-4267 SDRAM	Up to 8GB LPDDR4	Dual channel 4GB at 1600MHz LPDDR4	8KB SRAM
Wireless Connectivity	Dual-band 802.11ac Wi-Fi,Bluetooth 5.0/Bluetooth Low Energy (BLE)	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE	No WiFi	No WiFi/Bluetooth
Input Power	5V/5A DC power via USB-C, with Power Delivery support	5V DC via USB-C connector and GPIO header (minimum 3A) Power over Ethernet (PoE)	5.0V DC input, supplies PMIC and other regulators	5V/200mA DC Input
GPIO	Standard 40-pin header	Standard 40-pin GPIO header	Standard 40-pin GPIO header	54 digital I/O pins
Communication	Same as Raspberry Pi 4	1 bus I2C,SPI, 1 UART config	4 ports UART,SPI,I2C	GPIO UART,SPI, I2C
GPU	VideoCore VII GPU	VideoCore VI	128-core GPU	None

Software Support	Linux, ROS2, OpenCV, AI models	Linux, ROS2, OpenCV	Linux, AI-based CUDA acceleration	Arduino
------------------	--------------------------------	---------------------	-----------------------------------	---------

3.9 Power Supply

Providing the appropriate power to the robot is crucial as it determines whether the system as a whole can operate properly. This section covers the different power supply requirements and options for the robot and how the power is implemented for the various components. The main power supply for the robot must be able to move with it and be wireless, isolated from any stationary power supply, such as a wall outlet. Therefore, the necessary power supply must be a type of battery. This section researches and considers different types of batteries as well as the voltage regulators needed to deliver the proper voltage to the different components on the robot.

3.9.1 Battery Considerations

Based on the electrical requirements of the robot, it was decided that a 12V battery would be ideal. This voltage allows for all the components to operate and can be easily stepped down using voltage regulators to the specific voltages needed. The battery must also be rechargeable, as the robot is not designed to be single use. A non-rechargeable battery would cause problems and lower usability and efficiency of the robot. An important consideration that cannot be overlooked is the safety of the battery. The battery must not be high risk and risk causing fires or harming anyone. Another consideration is the weight of the battery. Since the robot must carry heavy payloads, lowering the overall weight of the robot is important. Upon conducting extensive research and taking into account all the criteria, several battery types were considered.

3.9.2 Power Distribution

A well organized power distribution system is crucial to efficiently distribute power to all of the components and peripherals on the robot and operate properly. Efficient power distribution ensures stable and safe operation to maximize battery life. This section outlines the power distribution for the robot, detailing how the 12V battery will be distributed across the different subsystems.

Table 20: Power Distribution Table

Component	Voltage (V)	Current Draw	Qty	Power Consumption (W)	Total Power Consumption (W)
ESP32-WROOM-32E	2.0-3.6	~20 mA	1	0.04-0.072	0.04-0.072

Raspberry Pi 5	5	5 A	1	25	25
RYRR10S	3.3-5.5	~50 mA	1	0.165-0.275	0.165-0.275
AST1024	3.3-5.5	~100 mA	3	0.33-0.55	0.99-1.65
LSM6DSO	1.71-3.6	~0.55 mA	1	0.0009-0.002	0.0009-0.002
Pololu 2828	12	~1.5 A	2	18	36
Total System Power Consumption					62.2-63.12

3.9.3 Battery Types

Several suitable battery types include Lithium-Ion, Lithium Polymer (LiPo), Nickel-Metal Hydride (NiMH), Nickel Cadmium (NiCd), and Sealed Lead Acid (SLA) batteries. These batteries were considered based on the robot's power needs and the criteria discussed. Each type offers distinct advantages and disadvantages.

1. Lithium-Ion

Lithium-ion batteries are one of the most commonly used battery chemistries in modern devices due to their high energy density, light weight, and long life cycle. They have the ability to store a significant amount of energy relative to their size and weight, making them a good option for this robot, where space and weight are critical. They also offer relatively fast charging times and high discharge rates. Lithium-Ion batteries are commonly found in everyday devices, such as cell phones and laptops. Lithium-ion batteries offer protection against overcharging and require less maintenance. However, these batteries do come with significant safety concerns. With improper handling and usage, or a faulty battery management system (BMS), they are susceptible to overheating, fire, or even explosion. The batteries are very sensitive to damage as well, further increasing safety concerns. [A.35, A.36]

2. Lithium-Polymer (LiPo)

LiPo batteries are a type of lithium-ion battery that uses a polymer electrode instead of a liquid one, hence the name. LiPo batteries are commonly used in many hobbyist projects and industry applications, such as radio-controlled cars and drones, as well as electric vehicles. While LiPo batteries have a similar energy density to lithium-ion batteries, they are generally lighter and can be made smaller and thinner due to their polymer cells. Similar to lithium-ion batteries, LiPo batteries have high safety concerns and can be prone to damage, overheating, and fire. [A.35, A.36]

3. Nickel Cadmium (NiCd)

NiCd batteries are cost-effective and suitable for tools requiring relatively low power or voltage and are often used in smaller applications. However, NiCd are

prone to discharging issues and perform poorly in cold temperatures. [A.37] NiCd have the lowest capacity of any rechargeable battery as well as a high self-discharge rate, losing a significant portion of their charge even when not in use. NiCd also have some environmental safety concerns, as they contain toxic cadmium and cannot be disposed of in household trash, rather they have to be recycled in special facilities. [A.38]

4. Nickel Metal Hydride (NiMH)

NiMH batteries are a safer and more environmentally friendly option compared to lithium-based batteries. Similar to NiCd batteries, NiMH batteries suffer from the memory effect and high self-discharge rates [A.36]. This makes it difficult to have in the robot as the battery would continuously discharge when not in use and need to be constantly charged.

5. Lead-Acid

Lead-acid batteries are one of the oldest types of rechargeable batteries. Despite this, they remain a viable option for certain applications due to their reliability, affordability, and ease of maintenance. They are commonly used in automotive and industrial applications because of their ability to provide consistent power output and efficiently handle high current loads [A.40]. Despite having a low energy-to-weight ratio and low energy-to-volume ratio, they compensate by supplying high surge currents at a low cost [A.40]. Their durability and extremely low self-discharge rate make them stand out amongst the other batteries. Lead-acid batteries lose an average of 30-40% capacity after one year of storage [A.39]. Lead-acid batteries are still widely used in different small and large-scale applications [A.39].

3.9.3.1 Battery Selection

Among the battery types analyzed, each presented distinct advantages and disadvantages. Lithium-ion and LiPO batteries provide high energy density and are lightweight, but come with increased costs and safety risks, such as overheating and fire hazards. NiCd and NiMH, while offering good durability, have a lower energy density and suffer from memory effect and high self-discharge rates as well as having relatively high costs for the needed application. Lead-acid batteries, despite being heavier than the other options, emerge as the most suitable choice due to their combination of safety, reliability, cost-effectiveness, and ease of maintenance.

After weighing the various factors, lead-acid batteries were chosen as the best option for the robot. While they are heavier than the other batteries considered, their advantages outweigh their disadvantages. This added weight can also add to the stability of the robot when carrying the payloads. Additionally, lead-acid batteries are inherently safer, as they are not prone to thermal runaway or catching fire, unlike lithium-ion and LiPo batteries. Their low cost and wide availability also makes them a good choice, ensuring that the project remains within budget. Compared with the other batteries, lead-acid batteries are

a fraction of the price. Lead-acid batteries also have a very low self-discharge rate, meaning they can hold their charge for long periods of time when the robot is not in use.

Ultimately, while lithium based batteries offer advantages in terms of weight and energy density, the greater safety, affordability, and reliability of lead-acid batteries makes them the ideal power source for the robot. Due to this analysis, the team has selected sealed lead-acid batteries as the power supply.

Table 21: Comparison of Battery Types [A.35, A.37, A.40, A.41, A.42]

Feature	Lithium-Ion	LiPo	NiCd	NiMH	Lead Acid
Specific Energy (Wh/kg)	150-200	140-200	40-60	60-120	30-40
Self-Discharge Rate (%/month)	5-10	5-10	20	30	3-20
Memory Effect	No	No	Yes	Yes	No
Weight	Light	Very Light	Heavy	Medium	Heavy
Cost	\$75+	\$150+	\$100+	\$150+	\$20-30
Advantages	High energy density, low self-discharge, no memory effect	High energy density, low self-discharge, no memory effect, small and thin	Long life cycle	Inexpensive	High surge currents, very low cost, easy to use, low self-discharge
Disadvantages	Expensive, can overheat, safety concerns	Expensive, can overheat, safety concerns	Requires frequent discharging, heavy	Low capacity, high self-discharge, toxic materials	Low energy-to-weight ratio, corrosion, heavy

3.9.4 Voltage Regulation

Voltage regulators are essential components in power management systems, ensuring that all electronic devices, components, and sensors receive a stable voltage for proper operation. Regulators are used to prevent voltage fluctuations that could lead to instability and even damage sensitive components. In the case of this project, the 12V lead-acid battery supplies power to the entire system and every component, many of

which require smaller, more stable voltages. The voltage regulators will be used to provide the necessary voltages and currents to specific components.

There are three main power outputs needed for the robot: 3.3V, 5V, and another 5V for the Raspberry Pi 5. Since the Raspberry Pi 5 has such high power demands, it would be better to isolate it from the other power rails and provide it with its own voltage regulator that can supply it with the necessary current.

The first two voltage regulators, 3.3V and 5V, are needed to power the MCU and other peripherals and sensors. They do not need to have very high current capabilities as the MCU and all the components that will connect to them are not high powered devices.

Since the entire system is powered by the battery selected in the previous section, all of the voltage regulators must take a range of inputs that the battery can output from fully discharged to fully charged, approximately 10V to 13V. It would also be ideal to have a low heat generation to limit power waste and reduce risk of overheating on the PCB. Additionally, a higher efficiency is crucial as the robot is limited in uptime based on the capacity of the battery.

3.9.4.1 Voltage Regulator Selection

Table 22: Comparison of Low Power Voltage Regulators [C.26, C.27, C.28, C.29, C.30]

Parameter	AMS1117-3.3/5.0	LM2596-3.3/5.0	TPS563200	MP2315	LM317
Type	Linear	Switching	Switching	Switching	Linear
Input Voltage (V)	4.75-12	4.5-40	4.5-17	4.5-24	3-40
Output Voltage (V)	3.3 / 5	3.3 / 5	Adjustable	Adjustable	Adjustable
Max Output Current (A)	1	3	3	3	1.5
Efficiency	~50%	~80%	~92%	~90%	~40-60%
Quiescent Current	~5mA	~5mA	~150uA	~600uA	~1-10mA
Thermal Performance	High Heat	Moderate Heat	Low Heat	Low Heat	High Heat

One of the simpler choices was the ASM1117 series of linear voltage regulators, which provide a fixed output of either 3.3V or 5V and a maximum current capacity of 1A. While the design with these regulators is fairly straightforward and simple, they suffer

from extremely poor efficiency when stepping down from 12V, typically 50% or lower. This poor efficiency results in significant heat generation which in turn necessitates the use of heatsinks for proper thermal management, further increasing the complexity and potential problems with this regulator. Given the power restraints of the system and the regulator's poor performance, the ASM1117 is not an optimal choice.

Another linear regulator considered was the LM317. The LM317 is an adjustable voltage regulator capable of providing output voltages between 1.25V and 37V. Although it offers a wide range of flexibility with its output, it operates at a very low efficiency similar to the ASM1117, again leading to high heat production. This regulator has similar output performance to the ASM1117 and has a higher complexity in its design due to its adjustable nature. The high complexity on top of its poor and insufficient performance capabilities make the LM317 unsuitable for this project.

On the other hand, switching regulators offer much higher efficiencies than those of linear regulators. The LM2596 is a widely used buck converter that can efficiently step down voltages while supplying up to 3A of current. With an efficiency of approximately 80%, it shows a significant improvement over the linear regulators. However, this regulator has a quiescent current of approximately 5mA and a lower efficiency compared to the remaining regulators.

A slightly more efficient alternative is the MP2315. It offers an efficiency of approximately 90% and provides an adjustable output voltage from 0.8V to around 10V. It can provide up to 3A of current, making it a strong candidate. While MP2315 is a viable option for the regulators, it falls behind the TPS563200 in every category.

The TPS563200 was determined to be the optimal voltage regulator for the 3.3V and 5V rails on this project. It provides an adjustable output from 0.76V to 7V, a current of up to 3A, and boasts an efficiency of 92%, the highest amongst the options. Unlike the linear regulators, it generates minimal heat and negates the need for a heatsink and enhances system performance and reliability. Additionally, its extremely low quiescent current of 150uA ensures that minimal power is lost during idle operation, making it ideal for a battery power system like this one.

Table 23: Comparison of High Power Voltage Regulators [C.31, C.32, C.33, C.34, C.35]

Parameter	TPS566242 DRLR	LM2576	AP1501A	MP1584	LMR33630
Type	Switching	Switching	Switching	Switching	Switching
Input Voltage (V)	4.5-17	7-40	4.5-40	4.5-28	3.8-36
Output Voltage (V)	Adjustable	Adjustable	Adjustable	Adjustable	Adjustable

Max Output Current (A)	6	3	5	3	3
Efficiency	~95%	~75%	~85%	~90%	~92%
Quiescent Current	~12uA	~5mA	~5mA	~200uA	~26uA
Thermal Performance	Low Heat	Moderate Heat	Moderate Heat	Moderate Heat	Low Heat

This high power regulator will be used to power the Raspberry Pi 5, which has a voltage input of 5V and up to 5A. Thus, it is important to select a regulator capable of providing the necessary power if the Raspberry Pi demands it. Therefore, the options that provide less than 5A can automatically be eliminated as options. That removes the LM2576, MP1584, and LMR33630 regulators. That just leaves two options: TPS566242DRLR and XL4015.

The AP1501A can provide up to 5A with an efficiency of approximately 85%. With a potential load of 5A for the Raspberry Pi, the regulator may produce more heat and perform more poorly. However, the TPS566242DRLR offers an efficiency of approximately 95% and can provide up to 6A of current, above the maximum current requirements of the Raspberry Pi. Therefore, the TPS566242DRLR is the most ideal option for this.

3.10 Wifi/Bluetooth Connection

Table 24: Comparison of Wifi and Bluetooth

Wifi vs Bluetooth	
Wifi	Bluetooth
Consumes lots of Power	Consumes low amounts of Power, Great for Small or general battery-powered devices.
Free of charge and extends farther with local network connections (LAN) or to the internet.	Expensive for top-line products with great range, but options for a cost affordable option
Indoor & Outdoor range	Limited range (100 meters)
More users can connect to WiFi	Limited amount of users that connect to device
Large frequency band (2.4-5 GHz)	Low Frequency (2.4GHz) that has interference issues with other devices

Faster Speed: Data Transfer for WiFi 6 is 9.6GBps	Slower Speed: 50MBPS with Bluetooth 5.0
Greater security: WPA2 and WPA3	Bad Security: Attack prone for low-end Bluetooth devices.
Faster Bit-rate speed: 600MBPS	Slower Bit-rate speed: 2.1MBPS

The comparison shows the benefits and drawbacks between both options, and how it compliments the project. We understand there is no perfect option, we have to choose the best alternative. Naturally, the Internet Stack protocol (TCP/IP) uses WiFi to connect, and when developing an application that connects two devices over the Network it makes more sense we instead choose WiFi over Bluetooth. The table mainly demonstrates what each option provides, and what we must consider when implementing our design.

For example, using WiFi naturally consumes greater power, and requires better power-management or design for the system. If we decided to use Bluetooth this will not be an issue because it draws little power. Yet, has trouble maintaining high rates of bit-transactions between devices, connects few devices, and has the limited security (basic measures implemented) that makes it less reliable. Whereas, with WiFi it naturally has greater security implementations, can operate easily over dedicated hyperlinks (HTTP), and is naturally supported by the Internet Stack. Considering the project for our RTOS requires fast interactions between hardware components and mobile apps, speed becomes a necessary requirement. WiFi dominates here while working in multiple areas so we can reliably test it, and it proves to be the better alternative.

3.11 Communication

Table 25: Comparison of TCP and UDP

TCP vs UDP	
TCP	UDP
Manages the transfer of data in a set sequence between sender & receiver via measures like the 3-Way Handshake.	Has standard messenger protocol between sender and receiver. Nothing special and is why it is not very secure either unlike TCP.
Operations are independent on Operating System (OS)	Operations are independent of the Operating System (OS). However, it does its own work over this compared to TCP.
Multi-support routing protocols 1. Link State 2. DV	Multi-support routing protocols 1. RIP
Can reduce speed of data based off	Does not reduce speed to sacrifice

receivers speed.	performance, however does lose packets causing lower accuracy.
Greater amounts of recovery protocols, security protocols, and other measures taken to recover and manage packets over a network.	Low amounts of recovery protocols to ensure throughput is maintained.
HTTP1-2.0 is based off the TCP/IP stack protocol. 1. HTTP1-1.1 is the earliest version of browsers sending information over the internet via links but not reliably or as safe. 2. HTTP2.0 fixed many issues 1-1.1 has and remains the commonly used standard today.	Predominantly used HTTP3 with additional fixes to UDP to make it more reliable over this web encrypted hyperlink. So, it keeps the Speed and throughput of UDP while downsizing its flaws.
Has the ability to SSH (Secure Shell) into something via remote administration. This is great for remote SSHing into Pi, and continuing working on it from different locations. 1. Maintains a encrypted communication between CS (Client and Server).	Does not support UDP connections directly. Need additional tools to make this process work.
Common used sources (not particular to our project but interesting/important to know): 1. Netflix 2. YouTube 3. Spotify	The following are common uses: 1. Gaming 2. Remote Procedure Call (RPC) 3. DHCP 4. RIP 5. SNMP 6. DNS

The communication protocols mentioned are TCP and UDP. They are both designed with the intention of sending/managing packets between sender and receiver. The primary difference between them is their speed, reliability, and accuracy. For example, TCP known as the Transmission Control Protocol is considered to be the most reliable option. This is because it establishes a 3-Way Handshake, which manages package transmission between Sender and Receiver, and has greater security protocols to prevent package loss, corruption, and out-of-order delivery. In addition, TCP is what our modern hypertext link protocol (HTTP2.0) uses that manages sending packages over the network, and is considered one of the best versions for HTTP developed when dealing with package management. In fact, we would like to include that TCP's congestion control is a hybrid of several protocols that use different package management techniques to increase

reliability of these transactions happening over the Network. These techniques include: Sliding Window, Go-Back-N, SR (Selective Repeat), Sequence numbers, Timeout mechanisms, and other methods to ensure packets get recovered so that the data will reliably arrive at the final destination.

Moving on, TCP, despite having greater security measures and package management via SSL/TLS and the prior methods described, is not as fast as UDP. This is because the focus is reliability over speed. Granted, TCP is not slow to the extent that it becomes unusable, however, we find it important to note the speed difference between the two because it explains why our project is implementing TCP over UDP.

UDP is known as User Datagram Protocol and is much faster than TCP in nature. This speed comes at the price of accuracy because UDP is known for being unreliable unlike TCP. Which is where we discovered an alternative name used for UDP: Unreliable Datagram Protocol as it is genuinely unreliable. The unreliability comes from the increased speed to produce greater throughput. The idea was to decrease packet accuracy by having less reliability checks, security checks, and many components that TCP has to increase the overall throughput. This design decision makes UDP the preferred method for RTOS and IoT applications, since RTOS does not like delays between Sender and Receiver. The unreliability of UDP is the main reason we decided to not use this protocol. In addition, UDP is not the established standard unlike TCP, and despite the future implementation of HTTP3 which uses an improved version of UDP is still not considered the established standard.

Furthermore, this project requires information being transferred between robot and mobile app over the Network. Choosing a protocol that updates the user of the robots status, maintains reliable communications between the two, and focuses on accuracy as well speed made TCP remain the superior choice. Additionally, TCP supports encrypted protocols for messages sent between Client-Server (CS) architecture, it is used by reputable services demanding heavy use while managing network traffic like YouTube or Spotify. It supports standard routing protocols used to manage levels of traffic traveling through routers such as Link-State and Distance-Vector algorithms, and remains a relatively fast protocol to meet consumer use, all contributing reasons why we are using it over UDP. For example, when designing applications it is common to use HTTP2.0, and the reason is due to TCP's architecture and how well-designed HTTP2.0 for managing packets. The overall output in speed is 3-15 times higher than applications not using HTTP2.0. This means when designing the mobile application for our project, we can sacrifice some throughput to increase accuracy, the time-delay between Client-Server is tolerable, and the overall efficiency increase when developing the App with HTTP2.0 makes the choice of using TCP over UDP reasonable.

3.12 SLAM Protocols

Our robot will also need the implementation of an optimal routing algorithm that will allow it to navigate. For the context of our robot, ideally we want it to operate within an indoor setting without human involvement. We decided on implementing a path planning

algorithm with a premade map the robot will primarily be operating on. Path planning involves navigating a robot based on preexisting environmental data, which is loaded into the robot's planning system to compute a trajectory from the starting point to the destination [43]. This method generates a complete path before the robot begins its journey, essentially optimizing the route gradually.

Autonomous route planning comes with the problems of understanding where the robot is at a point on the path and the best approximation to the target location is.[44] When the user creates the path plan, the path generated by the planner must follow any imposed restrictions. These may come from the limitations in the adaptability of the robot to certain terrains.

We arrived at Simultaneous Localization and Mapping (SLAM) which allows the user to map out the environment around them using the data from various sensors. This will allow the use of an already established open source algorithm that we can utilize and save time for adjusting the indoor floor maps. Next we will be exploring the different SLAM protocols to identify the most optimal one for an indoor context.

3.12.1 SLAM

SLAM will be our main focus for navigating within an indoor environment using localization and pre-mapping techniques. Without the use of SLAM, it will just move randomly within a room and may not be able to detect certain obstacles along the path. In addition, this approach uses excessive power, so the battery will run out more quickly. Robots with a SLAM algorithm can use sensor information such as the number of wheel revolutions and data from cameras and other imaging sensors to determine the amount of movement needed.

SLAM algorithms have to calculate the current position and orientation by using an environment map in order to function well. The capacity to create devices that can function in complex surroundings solely using their on-board sensors and without the need for an external reference system like GPS is made possible by the availability of an accurate map.

The robot will use the sensor information from possibly a camera and other sensors to create a map of the obstacles in its surroundings and avoid cleaning the same area twice. This map is used to control the input and sensor readings. Since both data contain uncertainties from noises, it is necessary to apply an estimation theory to develop a map from this noisy information. Sensors such as wheel encoders, initial measurement units (IMU), RGB cameras for visual inputs, or LiDAR as remote sensing to measure ranges..

The foundation of SLAM includes the use of in using landmark correlations, data association, and loop-closure to reduce the uncertainties about its previously visited area and poses.[A.48] They primarily use Kalman Filter or particle filters for sequential state estimation and an optimal state estimation technique in a linear system with also calculating around the Gaussian noise. Bayes rule will be mainly used in the state

estimation as SLAM is creating the map. It states that a posterior probability of an event given an event is equal to the likelihood of given , multiplied with α , a normalization factor as shown in the equation below.

$$P(A|B) = \alpha P(B|A)P(A)$$

Equation 6: Bayes Rule

Bayes Rule encompasses the scenario that the robot would predict where it is at a moment in time and based on that probability here we predict what is the current state based on previous state estimate and current control input. Within the Bayes Rule algorithm, we have to account for x_k which is the correction step, where we incorporate the latest observation into the estimate of current state [A.48]. Within the context of our project we will be using Kalman Filter to mainly estimate an object's future location and detect changes in an indoor environment. Additionally, SLAM uses loop-closing to detect if a given keyframe or the pose given by the camera at a frame had been seen previously [A.48]. Loop Closure is formulated to optimize the state estimation that will compare the information that is stored in the past from generating the map and the point where the robot recognized a familiar area.

For our project, we took into account that we will be using TOF sensors, an Inertial Measurement Unit (IMU) and a monocular for gathering data inside our mapping environment to decide on the most optimal protocol. We will be going into the different methods for SLAM that will be dependent on the sensors we have chosen for our project.

3.12.3 Visual SLAM

Visual SLAM or VSLAM implements your traditional SLAM protocol however with the addition of visual sensors for a more straightforward way of identifying obstacles and objects. The sensors create a map of their surroundings to identify their specific location at a point in time. VSLAM primarily can use RGB-D, stereo and monocular cameras which can be relatively inexpensive when utilizing additional range sensors. This approach is widely popular since it is inexpensive, easy to calibrate, and has low power consumption in monocular cameras while also allowing depth estimation and high accuracy in RGB-D and stereo cameras.

It uses techniques such as computer vision to extract and match visual data for localization and mapping . It allows robots to map complex environments while performing tasks such as navigation in dynamic fields. It places a strong emphasis on accurate tracking of camera poses and estimating past trajectories of the robot during its work. This will be the method that we will use primarily due to its capabilities at a lower price point.

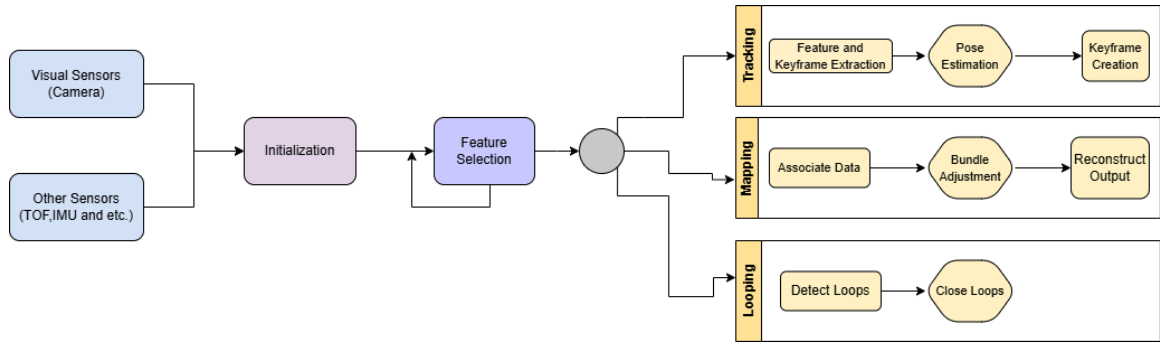


Figure 4: Visual SLAM Flowchart

The figure above outlines what we aimed to use within our project and how VSLAM will take in data from visual sensors for accurate mapping and localization. Some aspects that we need to consider are the two categories: direct and indirect (feature-based) methods. Indirect methods extract features obtained from textures by processing the scene and keeping track of them by matching their descriptors in sequential frames. We want to be able not only utilize these visual sensors but we want to be able to increase the robustness and accuracy for depth estimation that normal cameras lack. We thought of going with a visual sensor as a center point to understand a changing environment and then implement the TOF sensors to include depth estimation.

The way VSLAM works is by the camera receiving and understanding relevant features within the environment which get preprocessed into keyframes. Some approaches will perform a bundle adjustment after every keyframe to provide the basis for mapping the location the robot will be in. Filtering becomes extremely computationally expensive as the map model grows, however keyframes enable more feature points or larger maps, with a balanced tradeoff between accuracy and efficiency.

When we receive input data from the images, we should be able to see overlapping features between the images. These identical features can then be used to estimate the distance from the sensor. Which we will be needing to determine the destination of the robot when the payload is detected. We use this information to map pixels between the two images against a common plane. This is then used to determine the disparity of the common features between the two images. Disparity and distance are inversely related, such that as the distance from the camera increases, the disparity decreases[A.49]. Now, we can estimate the depth of each of the features using triangulation.

The color information obtained from the cameras contributes to a more vivid output map. The precision and resolution in mapping of VSLAM are in a way obsolete compared to LiDAR SLAM, however, the primary drawback of this method arises from its dependence on triangulating multi-view images' disparities, a computationally intensive task that is especially demanding of enormous computing resources in processing high-definition images and full traversal cases [A.52]. Therefore, hardware acceleration or server cluster support is usually required. The vision SLAM technique accurately computes the change in the carrier's position by matching consecutive frames of images. During this process, the carrier's position within the environment is accurately computed,

and the process is founded on the summation of positional changes. Unlike laser-based SLAM methods, visual SLAM boasts clear strengths that include lower cost, enhanced precision, and less volatile performance.

3.12.2 LiDAR SLAM

Light detection and ranging or LiDAR is a method that primarily uses a laser sensor to gather distance data. In comparison to using cameras, we use LiDAR to assess the environment. The value of intensity indicates the surface reflectance structure of barriers whereas the distance measurements are computed by time of flight [A.66]. There is the possibility of using LiDAR for applications with high-speed moving vehicles such as self-driving cars and drones. The output values from laser sensors are generally 2D (x, y) or 3D (x, y, z) point cloud data [A.48]. The laser sensor point cloud provides high-precision distance measurements and works effectively for map construction with SLAM algorithms. Movement is estimated sequentially registering the point clouds. Point clouds are essentially points in a 3D environment that indicates the surface of an object or landmark. This allows the robot to gather information from its environment based on where the laser sensor lands on.

From the point cloud, it then calculates the distance used for mobile robot localization. Laser scanners are important for obstacle detection and tracking, while cameras are often used to get a visual interpretation of the environment, but cannot work in bad lighting conditions. Its main drawback is the expensive search for point correspondences and its great sensitivity to the minimization starting point [A.67]. SLAM algorithms, including Google Cartographer, Hector SLAM, and more use these laser sensors to make precise and scalable maps in sparse-feature or dynamic environments. Based on the stance of SLAM algorithm developers, LiDAR uses SLAM that generates high mapping accuracy independent of lighting, even though VSLAM algorithms exist. LiDAR has utility in very outdoor or indoor environments and has the natural ability to function in low-light environments, whereas visual SLAM algorithms struggle to perform in those environments and obtain good results.

However, LiDAR has the main disadvantage of being too costly where the cheapest RPLiDAR A1 sensor is about \$100 which will take up a majority of our project budget. LiDAR units also have a larger size and weight compared to other commercial sensors which make it undesirable for our mobile delivery robot. We would have to account for the size within the chassis of our 60cmx60cm robot which will include the PCB and other components. Since the project is centered around our TOF sensors and visual camera we thought this would be more cost-efficient and still get the performance we need to accomplish the goals of this project.

Table 26: SLAM Comparison

Features	Visual SLAM	LiDAR SLAM
Technology Used	Visual sensors (Monocular,	LiDAR sensors (Laser based

	Stereo, RGB-D cameras)	sensors)
Cost	Moderate	Expensive
Power Consumption	Low (Cameras use less power)	High
Computational Load	High (including feature and frame processing)	Medium (point cloud processing)
Mapping Accuracy	Good but affected by lighting	High
Obstacle Detection	Moderate	High
Environment Conditions	Affected by lighting and visual obstructions	Not affected by conditions

3.13 Visual SLAM Comparison

3.13.1 ORB-SLAM

For VSLAM, there are many techniques that depend on the style of various cameras. One such technique is ORB-SLAM, a versatile and accurate SLAM solution for Monocular, Stereo and RGB-D cameras. It is able to compute in real-time the camera trajectory and a sparse 3D reconstruction of the scene in a wide variety of environments, ranging from small hand-held sequences of a desk to a car driven around several city blocks. It is able to close large loops and perform global relocalisation in real-time and from wide baselines. It includes an automatic and robust initialization from planar and non-planar scenes. ORB (Oriented FAST and Rotated BRIEF) uses binary features invariant to rotation and scale (in a certain range), resulting in a very fast recognizer with good invariance to viewpoint.

The sparse point cloud returned from the SLAM algorithm is compared with the area recognized by the OR network. Because each point from the 3D map has its counterpart in the current frame, therefore the filtration of points matching the area recognized by the Object Recognition algorithm is performed. The clustering algorithm determines areas in which points are densely distributed in order to detect spatial positions of objects detected. Images are converted to grayscale, and then the FAST algorithm is searching for corners. FAST consists of detecting a set of pixels N from the image within a certain radius that meet the threshold value. The ORB algorithm is using the intensity centroid method [A.53] and is able to determine the orientation of a corner. Assuming that the intensity is shifted from its center, the resulting vector will help to determine the orientation.

When ORB maps, it then uses bundle adjustment which identifies the equivalency of the mismatched points from the point cloud. The ORB points are searched from the remaining key frames to triangulate the new points [A.53]. The algorithm monitors the resulting points and eliminates the points so only the most accurate point data remains. We plan to use this ORB-SLAM algorithm for this project due to the lack of complexity in comparison to other VSLAM algorithms like RTAB-MAP and OPEN-VSLAM. The visual sensor that we select for this algorithm will allow us to use feature detection accurately without using a higher end camera making it the best choice.

3.13.2 RTAB-Map

RTAB-Map or Real-Time Appearance-Based Mapping is another alternative for visual SLAM however it primarily uses RGB-D, stereo cameras and LiDAR which deterred us from going this direction for visual SLAM. RTAB-Map is a graph based SLAM that uses a loop closure detector and uses additional memory management to determine how likely a new image comes from a previous location or a new location. When a loop closure is accepted, a new constraint is added to the map's graph, then a graph optimizer minimizes the errors in the map.

This approach is typically chosen for high end RGB-D cameras where low-drift odometry is warranted for map accuracy. It also allows for multiple sessions of mapping for the user to use multiple complex maps within their system. Some situations of VSLAM need the distinction of different objects to determine how the robot will navigate. Using RTAB-MAP will allow you to use the color data along with the visual feature data to determine how the robot will display objects on their generated map.

Sources have confirmed that when estimating odometry with sensors such as cameras and lidars can be very accurate when there are enough features in the environment, but only using one sensing modality can be problematic and prone to localization failures if their tracked features in the environment are no longer visible [A.64]. Which are part of the many drawbacks of SLAM and a part of the reason why we had to compromise using VSLAM. RTAB-Map is not only computationally intensive with the introduction of color data, it will also be more complex in learning how to calibrate accurate color data in terms of our robot. Since our project would not need the introduction of these colors as well as our limited skill set when working with computer vision we had opted out of using this algorithm. ORB-SLAM was more ideal in maintaining viability under the various constraints and costs.

Table 27: SLAM Algorithm Comparison

Feature	ORB-SLAM	RTAB-Map
Algorithm	Feature-based SLAM	Graph-based SLAM
Sensors Required	Works with Monocular, Stereo, and RGB-D cameras	Primarily used with RGB-D, stereo cameras and LiDAR

Mapping Type	Sparse point-based 3D map	3D point cloud and occupancy grid maps
Loop Closure Method	Bag-of-words approach	Appearance-based loop closure for relocalization
Real-Time Capability	Ideal for real-time tracking and feature extraction	Excellent Real-time tracking but with high hardware demand
Computational Load	Moderate efficiency	Highly Intensive
Use Cases	Indoor/outdoor navigation, object tracking and localization	Large-scale mapping, multi-session SLAM, long-term environment tracking
Cost	Relatively Cheap (depending on hardware)	Expensive

3.14 Image Processor

For this project, we need to utilize software that requires processing the visual data that we gather in the Raspberry Pi V2 camera and be able to interpret and manipulate the data for the needs of our SLAM navigation. With this technology, SLAM would have a better time filtering out what we consider obstacles and what should be accounted for in terms of mapping out our indoor environment. The techniques that image processing allows are image noise reduction, filtering, object detection and its ideal for real-time processing.

Since we will be using computer vision for this project, We have to be able to filter and enhance the image so the robot understands its destination. To properly utilize what the Raspberry Pi V2 camera captures, libraries and software are required to optimize what the robot can perceive and characterize certain objects as the robot is traversing the set path. In terms of criteria we were looking for, we wanted an application that will be intuitive to use, aligned with how we wanted ORB-SLAM the image, readily available and not hardware demanding.

3.14.1 OpenCV

OpenCV has many features used to perform image filtering, feature extraction and edge detection all of which are essential components of VSLAM implementation on Raspberry Pi V2. OpenCV is compact and optimized for ARM 'a' processor architecture that works on Raspberry Pi 5 for the purposes of processing camera frames in real-time without the need for dedicated GPU acceleration [A.88]. It is a universally available and open-sourced tool that mainly uses C++, Python, and Java that will allow us the flexibility we need to efficiently use the application. OpenCV contains a plethora of

algorithms they use that will be able to further optimize the navigation capabilities of our robot.

For OpenCV, it has the ability of feature extraction, image preprocessing, edge and object detection, motion and depth estimation [A.88]. OpenCV has all the components we need to preprocess image information for keypoint extraction. This will allow ORB-SLAM to accurately pinpoint the points with SLAM's mapping to determine the indicators for path rerouting. OpenCV contains optimized functionalities in comparison to other image processing methods that we have found. This is very important for our autonomous delivery robot, which depends on visual landmarks to make position estimates and navigate in real time. Using OpenCV's ORB feature detection algorithm, our robot can optimally extract key points from images taken from the Raspberry Pi V2 camera identified in the images regardless of environmental lighting and different camera angle positions. These key points give ORB-SLAM the ability to create a successful 3D map of the environment to track the movements of the robot to attempt to represent distance and accurately navigate and avoid obstacles during delivery.

Another important aspect that we will utilize is the object detection capabilities which will be essential in the context of using autonomous navigation. OpenCV's built in support for ORB object detection makes it a perfect match for ORB-SLAM. OpenCV is open source and free that is widely used in today's industry, which is also an important factor in comparison to many other image processing applications. OpenCV will be the optimal choice for working alongside SLAM especially through its ample algorithm support.

3.14.2 MATLAB Image Processing Toolbox

MATLAB's Image Processing Toolbox holds its own set of imaging algorithms and workflow apps for image processing, analysis, visualization, and algorithm development [A.91]. You can perform image segmentation, image enhancement, noise reduction, geometric transformations, and image registration using deep learning and traditional image processing techniques. The toolbox supports processing of 2D, 3D, and arbitrarily large images.

Image Processing Toolbox lets you automate common image processing workflows. You can interactively segment image data, compare image registration techniques, and batch-process large datasets. This is more catered to users who want to enhance static images and is not ideal for VSLAM. This also uses a paid license and lacks the features we would need to autonomously navigate through ORB-SLAM's functions since we will be using real-time video feed. We determined that this method would not be ideal in the concept of consistently being able process each frame of the video and output into a consumable dataset for ROS.

3.14.3 Nvidia VisionWorks

NVIDIA VisionWorks is a high-performance computer vision and image processing SDK to be run on NVIDIA GPUs, particularly Jetson and CUDA-enabled devices. VisionWorks, built on top of the OpenVX standard, provides a large set of functionality for real-time image processing, feature detection, motion tracking, and deep learning inference [A.90]. One of its greatest advantages is that it is able to leverage parallel computing through CUDA cores to accelerate vision-based applications like autonomous robots, augmented reality, object recognition, and SLAM.

VisionWorks supports key vision operations such as stereo depth estimation, optical flow tracking, edge detection, and geometric transformations, making it highly suitable for AI-based applications requiring precise environmental perception. Apart from that, its close integration with TensorRT and DeepStream SDK facilitates deep learning inference on video streams, which makes it particularly suitable for AI robots and autonomous vehicles. Nevertheless, despite its state-of-the-art features, VisionWorks is not an optimal choice for every embedded system. It has a high dependency on NVIDIA Jetson hardware, making it unsupported on non-CUDA systems like the Raspberry Pi, which is based on an ARM CPU without GPU acceleration [A.90]. Also, VisionWorks demands higher power consumption, which is less ideal for battery-powered, lightweight robotics projects. While it is more capable with more complex AI-based operations, for projects like ours where low power consumption and efficiency are favored. Meanwhile, OpenCV provides sufficient vision processing without necessarily needing dedicated GPU acceleration.

Table 28: Image processing Comparison [A.88, A.90, A.91]

Feature	OpenCV	MATLAB Image Processing Toolbox	Nvidia VisionWorks
Platform Capability	Linux, Raspberry Pi, Windows	Windows, Linux and MacOS	Nvidia Jetson (CUDA enabled)
Programming Language	C++, Java and Python	MATLAB	C++
Hardware Requirements	Relies on CPU processing however can use a GPU as well	Uses CPU and GPU acceleration with MATLAB's parallel computing	Relies on Nvidia hardware (GPU)
SLAM Capabilities	ORB-SLAM, RTAB-MAP, LiDAR SLAM	Limited SLAM support, only for static images	Compatible with all SLAM by Nvidia's SDK
Cost	Free/Open-Source	Requires paid license	Free
User Complexity	Moderate Ease of Use	User-friendly scripts and intuitive	High Complexity

Efficiency	High efficiency on standard hardware	Generally slower computation time	High Efficiency optimized with Nvidia GPUs
------------	--------------------------------------	-----------------------------------	--

3.15 Path Planning

As for navigation, we would need a way for the robot to properly navigate through the map that is generated through ORB-SLAM and adjust the route based on destination. When using a camera as a visual sensor for SLAM, we need a way for it to identify whether obstacles are an object or person and how the robot will navigate around once the map is made. There are various tools that allow us to calibrate the images processed in OpenCV or use distance sensors to adjust it so the robot can properly know what it will be avoiding. The main parameters that we need to determine an ideal path planning tool to use would be based on the input (camera or distance sensor based), the computational load, the compatibility with the ROS2 jazzy system and the type of algorithm we would need to be familiar with.

3.15.1 DWA (Dynamic Window Approach)

The Dynamic Window Approach (DWA) is an obstacle-avoidance practice which considers the dynamic and kinematic constraints of a mobile robot. The basic idea is as follows. First we find the admissible controls -- controls that allow the robot to stop before hitting an obstacle, in which that must conform to the above constraints. Then we optimize over those admissible controls to find which results in the highest utility in some prescribed sense [A.99]. It is classified as a local path planner which means it would adjust the path in real time accounting for unexpected obstacles that would not be accounted for in a pre-existing map.

Our robot would rely on the sensor data obtained from the TOF and Raspberry Pi V2 camera; this will be ideal when DWA is incorporated for its local path planning [A.100]. Since ORB-SLAM provides a well-formed position and sparse structure about the environment, DWA has a role to play in local navigation using real-time obstacle measurements of the real world in conjunction with ORB-SLAM. In terms of use case, this approach is ideal for indoor dynamic environments and functions under reactive obstacle avoidance. It is relatively light in computational load since it does not require full-depth data from the costmap and can operate under limited information.

There are some limitations to it since it would not use information gathered from the visual sensors and is based on what the robot's sensor can pick up. In particular, DWA's infers on the best path from desired goal position, distance to obstacles, and robot speed all based on their fixed weight coefficient. This can lead to selecting ineffective paths or even collisions in limited or congested areas since these coefficients can only account for so many unexpected encounters. Despite this, it still has a greater compatibility with the aspect of this project from ROS compatibility, simple integration and an ideal use case for our delivery robot.

3.15.2 Navfn

Navfn is a global path planner that uses a combination between Dijkstra's algorithm and A* characteristics to determine the robots trajectory in a large environment. It calculates the shortest (weighted) path to the goal using the Dijkstra algorithm. The points on the costmap which are obstacles are rated as high as possible. After calculating the path it is checked whether the goal is in an obstacle. If that is the case, the path is traversed backwards from the goal until a point is found which is not part of an obstacle [A.101]. It works on a global 2D costmap and produces a path consisting of waypoints that guide the robot toward its goal while avoiding known static obstacles.

This is another ideal approach to use when only focusing on a general destination without accounting for unforeseen variables. This approach includes being able to use the robots localization parameters then applying that to a generated 2D map. In comparison to the Dynamic Window Approach, it comes at the cost of only seeing the big picture versus addressing ever changing environments. For this project we viewed our robot being able to operate in a dense environment and navigate to its waypoint accommodating people and objects.

Navfn does not continuously run in real-time, preventing it from making reactive decisions for obstacle avoidance. Since it is not running all the time, you can accommodate an efficient computational load during operation at the cost of loss in dynamic environment interpretation. This makes it less than ideal in terms of operating in a local environment and working around active obstacles. Thus making DWA a more than ideal algorithm to use for SLAM and autonomous navigation.

Table 29: Path Planning Comparison [A.99,A.100,A.101]

Path Planning Algorithms		
Feature	Navfn	DWA
Algorithm Type	A* algorithm	Velocity-space sampling
Use Cases	Navigates large, known environments	Indoor real-time reactive obstacle avoidance
Input	Start & goal pose from SLAM	Current robot pose via SLAM, velocity, local costmap
Planning Type	Global Path Planning	Local Path Planning
Obstacle Avoidance	Avoids known static obstacles in global map	Dynamic obstacle avoidance
Computational Load	Efficient but Slightly	CPU-Accelerated/

	higher load	Lightweight
Real-Time Function	Runs occasionally, not frequent	Continuously running and scanning
Limitations	Doesn't account for robot dynamics; static environments	Needs a costmap; poor in congested areas
ROS	Integrated with the nav2 package	Integrated with ROS2's nav2 package
Ease of Use	Reliable on 2D maps; simple setup with Nav2 controller	Works as a package in the Nav2 plugin in ROS2

3.16 Mobile App

3.16.1 Flutter (IOS/Android)

Flutter, a UI framework developed by Google and designed for open-source use, allows applications built on a common codebase to run on various platforms. Dart programming language is used to build Flutter and Flutter is well-known for its fast rendering engine, enabling quick and responsive UI on both Android and iOS mobile devices [A.92].

We feel that Flutter is best suited for our robot, as it allows for a mobile app built with multiple platforms in mind to be built and deployed quickly, reducing overall development time, and providing a consistent user experience of the platform. Specifically, in our project, a mobile app built with Flutter will provide a simple and easy opportunity for the user to create an account, log into, and receive live updates from the robot. More specifically, in the case of our platform, the app will interface with the Raspberry Pi 5 over wireless networking protocols (HTTP), and the user will be notified as updates occur when the RFID is detected. The mobile application will also provide a status tracking feature that will show the robot's movements as they occur using ORB-SLAM3 and then notify the user when the delivery is successfully dispatched. Flutter allows an easy integration with Firebase or REST APIs, which makes it very ideal for managing account and user authentication, as well as exchanging data with the backend infrastructures.

Flutter also provides support for real-time user interface updates and animations which would allow us to visualize the robot's path in a friendly user manner. Flutter has a very active developer community and ample documentation, which provides assured support and resources so this is a good choice during this phase of the project.

3.16.2 Front and Back-end

The architecture of our mobile application will need to take into account a well-optimized and efficient plan for the front-end and back-end with factors such as cross-platform compatibility, real-time communication, and an efficient use of resources since the mobile app will only interface directly with the Raspberry Pi 5 that will take care of conducting all the robot's actions. The mobile app's front-end will be developed using Flutter—a user interface toolkit designed to make design easy, fast, and highly customizable for both Android and iOS applications from a single code base. Our plan for the front-end of the mobile application is to build a responsive user interface that goes through the uses of account login and generating real-time status updates, RFID notifications, and tracking deliveries. The front-end will connect to the back-end using either HTTP RESTful APIs or MQTT, a simple messaging protocol, which is also lightweight, suitable for real-time Internet-of-Things communication.

In terms of the back-end, even though mobile applications are often built from web-based stacks (i.e., Node.js and Firebase), we are going to utilize the C programming language on our Raspberry Pi 5 for our project's back-end. The C programming logic will manage core system logic, RFID processing, and SLAM data processing. The back-end logic written in C will provide structure for reading RFID tag scans, governing SLAM navigation decisions, and formatting the messages or status updates sent from the Raspberry Pi back to the app. Because of speed and post-processing, the Raspberry Pi and Flutter app will share information back and forth in either a lightweight object type (JSON) or binary packets.

For real-time updates (e.g., "payload identified" or "destination reached"), the backend can implement a publish-subscribe mechanism or use web sockets, depending on the network stability and scalability needs. Security measures such as token-based authentication or encrypted communication will be integrated where possible to ensure safe data transmission. Flutter for a visually rich front-end and C for performance-focused back-end communication—maximizes efficiency, responsiveness, and platform compatibility, all of which are critical for the user experience in our autonomous delivery robot system.

Chapter 4: Standards and Design Constraints

4.1 Standards

4.1.1 I2C Protocol

Inter-Integrated Circuit, or I2C, is a two-wire serial communication protocol using a serial data line (SDA) and a serial clock line (SCL) to provide simple and robust communication between peripheral devices and the microcontroller. In I2C, the peripheral devices are referred to as slaves or target devices and the microcontroller is referred to as the master or controller device.

To initiate communication the SDA and SCL must start in the idle state which means both lines are high. Only a controller device can initiate communication with the target

devices. It does this with a START condition that pulls down the SDA while SCL is high and then pulls the SCL down. This indicates that this controller device is going to use the I2C bus to communicate. The SCL is a serial clock controlled by the controller device. The SDA is used to transmit data to and from the target devices. Next comes the address to the target device that the controller wants to reach. Each target device has a unique 7-bit I2C address. The 8th bit sent on the bus is a read-write bit. If this bit is 1 the master is requesting to read from the target device. If it is 0, the master is requesting to write data to the target device. The extra 9th bit is an acknowledgement bit from the target device. The target device will pull the SDA line to indicate that the address has been received. This is an ACK or acknowledgement. If the address was not received or the communication was unsuccessful in any way it is left high to indicate a NACK or no acknowledgement. After the address frame the master will send one or more data frames each one byte at a time with an acknowledge bit at the end. If the master is reading information then it will send an ACK bit back to let the target device know the data has been received. If it is writing to the target device, then the target device will initiate the ACK or NACK to indicate if it has received the information successfully or not. When the master is done communicating it releases the SCL to high and then the SDA to high afterwards. This is the STOP condition. Until the controller device performs the STOP condition no other controller device on the shared I2C bus will be able to use the bus.

The I2C Protocol has 5 standard speed modes. These speed modes are standard-mode (operates up to 100kbps), fast-mode (operates up to 400kbps), fast-mode plus (operates up to 1Mbps), high-speed mode (operates up to 3.4Mbps), and ultra-fast mode (operates up to 5Mbps). The first three modes are relatively similar and are easiest to implement. In high-speed mode the controller device must first use a controller code to allow for high-speed data transfer to be enabled in the target device. Ultra-Fast mode is write-only and omits some other I2C features.

This protocol allows for multiple target devices and multiple controller devices. I2C is a half-duplex communication. This means only one device, controller or target device, can send data at a time. To remove communication conflicts between the controller device and the target device it uses START and STOP bits and ACK and NACK bits. To remove communication conflicts between different controller devices clock synchronization and arbitration. To synchronize the clocks there is wired-AND SCL. This wire-AND SCL is high only if both controller SCL's are high. If one of the controller's brings their SCL low then wire-AND SCL will also go low. Arbitration happens next and will also include a wire-AND but this time on the SDA lines. Both controller's monitor the SDA and SCL lines and they will notice when there is a difference in the SDA lines. The wired-AND SDA line will determine who wins the right to transmit. The other controller will stop transmitting.

4.1.2 UART Protocol

Universal Asynchronous Receiver/Transmitter or UART, is a two-wire serial hardware communication protocol. UART will be used as the communication protocol to send data from the ESP32 and the Raspberry Pi. UART is an asynchronous serial communication

which means that no clock signal is used to synchronize the outgoing transmission to the receiving device. However, the baud rate must be the same for the transmitting device and the receiving device. UART baud rates in bits per second could be: 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1500000. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off. The two signals are the transmitter and the receiver and a UART device; these are dedicated pins.

The data transmission is done in a packet consisting of a 1 bit start bit, 5-9 bit data frame, 0-1 bit parity bit, and 1-2 bit stop bits. The transmission line is held at high when it is not transmitting. The start bit is indicated by pulling down the transmission line for one clock cycle. Next, is the data frame. It can be 5 to 8 bits if the parity bit is used or 9 bits if the parity bit is not used. Next, is the parity bit. Parity bits are used to confirm whether the data transmitted was correctly received. There are multiple reasons as to why data is not correctly transmitting such as mismatched baud rates or long distance data transfers. Parity refers to an even or oddness. In this context, when the number of 1's in a data transmission are added together to get an even number the parity bit will be 0 and when the number of 1's in a data transmission are added together to get an odd number the parity bit will be 1. If the parity bit received correctly matches the parity but the receiver device calculates from the data received then it indicates the UART transmission was successful. If they do not match then the data received has errors. Lastly, the stop bit which just holds the transmission line high for 1 to 2 bits.

The steps for transmitting data across UART are as follows:

1. The transmitting UART receives data from the data bus. The data bus transmitted this data from another device such as a microcontroller or memory. The data bus transmits this data in parallel for which means it is transmitted all at once.
2. The transmitting UART adds the start bit, parity bit, and the stop bit(s) to the data frame to create the full data transmission packet.
3. The entire packet is sent serially, or 1 bit at a time, starting from start bit to stop bit from the transmitting UART to the receiving UART. The receiving UART device samples the data line at the preconfigured baud rate. The transmitting and receiving devices must be configured to the same baud rate.
4. The receiving UART discards the start bit, parity bit, and stop bit from the data frame before converting the data frame back back into parallel and transfers it to the data bus on the receiving end. [A.29]

4.1.3 SPI Communications

SPI or Serial Peripheral Interface is a synchronous communication protocol that is commonly used for sending data between microcontrollers and peripheral devices such as SD cards. In this system SPI will be used as the communication protocol between the ESP32 and the RFID module.

SPI uses separate lines for data and a clock to keep both devices in sync. There can only be one controller device but there could be multiple peripheral devices. In this project only one peripheral device will be used so this section will focus on that only. The device that generates the clock signal is the controller device and the other device is the peripheral device. Data is sent from the controller to a peripheral on the PICO data line (Peripheral In / Controller Out). And the peripheral's response will be on a third data line POCI (Peripheral Out / Controller In). The last line is the Chip Select that will indicate to the peripheral device that it needs to wake up and receive and send data now and is typically active low [A.94].

4.1.4 RFID Tags ISO14443A/B

ISO14443 is an international standard defining the physical characteristics and working interaction between contactless tags and devices operating at 13.56 MHz at up to 10 cm in distance. This standard is most often used for payment with contactless credit cards, mass transit systems, access control with ID badges, and identification in contactless ID cards and passports. ISO14443A/B will be used as the standard of RFID tags that are placed on the payloads and with the RFID reader. or a system to follow the ISO14443 standard it must meet requirements in 4 different parts: Physical Characteristics, Radio frequency power and signal interface, Initialization and anticollision, and Transmission protocol.

Physical Characteristics: Must follow ISO 7810 that outlines four standard card sizes shown in the table below.

Table 30: Physical Characteristics of Identification Cards [A.89]

Format	Dimensions (inches)
ID-1	3.370×2.125
ID-2	4.134×2.913
ID-3	4.921×3.465
ID-000	0.984×0.591

Radio Frequency and Signal Interface: This specifies the modulation methods, bit rates, and power rates for the communication. There are two different types of ISO14443. Type A and Type B differ in the modulation, coding, initialization procedures and other technical details but they have the same frequency of 13.56MHz [A.30]. Type A uses amplitude-shift keying and Type B uses binary phase-shift keying. Both use a half-duplex communication.

Initialization and Anticollision: This part of the standard documents outlines the protocols for detecting multiple cards within the reader's field and resolving conflicts using anti collision mechanisms. It ensures that only one card is selected at a time for

data exchange, preventing interference when multiple tags are present. In this project we will only be reading one RFID tag at a time, but a stretch goal for future iterations could include reading multiple payloads tags at once and this portion of the standard would be more relevant.

Transmission Protocol: This establishes the data transfer procedures, including frame structures, error detection, and recovery mechanisms. It supports secure and reliable communication between the reader and card, facilitating encrypted data exchange in security-critical applications.

4.1.5 ISO 10218 - Safety Requirements for Robots

ISO 10218 is an international standard that defines safety requirements for industrial robots. It provides general guidelines that ensure safe design, integration, and operation of robots in manufacturing and industrial environments. The standard is divided into two parts: ISO 10218-1 and ISO 10218-2.

The first part, ISO 10218-1, covers the safety requirements for the design and construction of the actual robot. It describes basic hazards that can be associated with the robots and provides requirements to reduce or eliminate the risks associated with those hazards. This standard highlights that there must be built-in limits on a robot's force, speed, and power to prevent human injury or damage with surrounding objects. It also includes guidelines for protective measures and failure protection such as emergency stop buttons and fail safes. [A.61, A.62]

The second part of the standard, ISO 10218-2, focuses on the installation of the robot in an industrial environment. It ensures a safe integration in the environment and with any surrounding equipment. It is intended for the workers and engineers responsible for setting up and operating the robot. This standard highlights assessing the environment and identifying any potential risks and hazards, such as crushing or entrapment. [A.63]

Overall, ISO 10218 is designed to build the foundation for safety when using robots in an industrial environment. As industrial robots become more advanced and widely used, complying with safety standards like ISO 10218 helps manufacturers, engineers, and operators implement established safety measures.

4.1.6 TCP/IP Protocol Standard

The traditional and national standard for Network communications is the TCP/IP stack protocol, and after additional research we determined it being the best framework to base our project off of. This has been the established standard for internet communication since the 1970s, and is one we predominantly use now (HTTP2.0 vs. 3.0 which is still in development), with more strides allowing faster communication between Sender & Receiver reliably. The standard was created by DARPA (Defence Advanced Research Projects). It was created after taking the theoretical OSI framework (7-layer stack) and

converting it into a 5-layer stack known as the internet stack, also known as TCP/IP Stack (also known as: **Internet Stack protocol**).

This is where the bimodal (2-part system) process became the standard approach for communicating over the Network stack and operating properly between sender and receiver. Then we looked into our decision further comparing if this model/protocol should be used. The importance of describing UDP & TCP helps explain why the **internet stack protocol** is considered the standard reliable network protocol. As seen above this framework is used because it is a reputedly more secure network communication model. When dissecting it further, we notice a 3-way system interacting between the upper layer where TCP handles transferring data, and IP handling the data itself, being sent between sender and receiver via the **3-way Handshake**. This level of security provides insurance that data being transferred over the network will reliably arrive.

Moving on, we mentioned HTTP3 versus HTTP2.0 because the new standard in development is HTTP3, yet we decided to use HTTP2.0 instead for our project. HTTP3 uses UDP instead of TCP, however, instead of being an unreliable network it is now reliable with its improvements, while keeping the overall throughput. In essence, the focus of the new standard being developed takes all the benefits of UDP, while mitigating the drawbacks. This resulted in UDP being recommended for RTOS projects; we still decided to use the current national standard.

Our project requires handling the communications interacting over an IP address to establish the link, connecting the hardware with our mobile app, sending proper updates between sender and receiver, while handling packets properly so the users can reliably receive the data, reinforcing trust between the client and P.A.R.C.L.E. The additional benefits of using HTTP in our mobile app was enabling faster debugging and interactions over our network web-servers.. Some sources claimed HTTP2.0 speed improvement is around 3-15 times faster than using HTTP1.1 (2nd generation). With these advancements in increasing the speed, TCP vs. UDP, where alone UDP dominates, TCP is fast and efficient enough to meet our standards for P.A.R.C.L.E.

4.2 Design Constraints

4.2.1 Time Constraints

The biggest constraint on this project will be the time frame this project needs to be accomplished in. This time constraint will put strain on the scope of the project, ordering parts, overall quality, and potentially result in inadequate testing and debugging. However, this team has put into place some strategies to help mitigate the stress of these issues.

The first aspect of the time constraint that causes strain is that senior design has hard deadlines. This means defining the scope of the project, conducting research, designing a prototype, ordering parts, building, testing, and redesigning will be limited to two

academic semesters of 16 weeks each. In order to pass senior design and graduate this project has a hard deadline we must meet at the end of senior design 2 and it can not be extended. This means that the scope of this project will be limited on what we think we can accomplish in this time frame. Our group is doing senior design 1 in the spring semester and senior design 2 in the fall semester. The summer break between semesters could possibly be used to our advantage.

The second aspect of the time constraint to consider is when ordering parts and the PCB. In order to ensure that all parts will arrive in time for enough building, testing, and redesigning, we will look for parts that are already in stock. If we are unable to get parts that are already in stock we will choose parts with low lead times. Regardless, all parts will be ordered before the end of senior design 1 so if anything is delayed there will be enough time for the parts to come in over the summer break. The first iteration of the PCB will be ordered and have enough time to be delivered before the start of senior design 2 so our group can begin the semester with a strong start.

A key part of the engineering design process is conducting research. Because of the hard deadlines and all the work that needs to be done it may result in rushed research and rush decision making on parts, design, and planning aspects of the project. Our group divided the research of each part of the project to different members of the group and dedicated 4 weeks to researching. With weekly in person meetings scheduled and access to view our group members research notes and writing we can keep each other accountable in staying on schedule so no part of the research is rushed.

Another key part of the engineering design process is testing and discovering issues and then fixing those issues. Because of the hard time frame we will have a limited time frame for testing and redesigning and debugging. This may lead to us not having complete coverage of all test cases and thus not discovering all issues. It may also be that we do not have enough time to address all issues before the deadline. This would impact the overall quality of the project. To mitigate the risk of this happening we are planning on working early and often in senior design 2.

In order to combat this major constraint our group has put in place milestone deadlines for each section and divided responsibility to each group member. To keep ourselves accountable for staying on track we have access to each other's work and weekly in person meetings to discuss progress, what each of us will be doing in the next week, and the blockers we are experiencing.

4.2.2 Economic Constraints

Our initial budget that we set will be capped at 500\$, which will be one of the biggest constraints when constructing this robot. This influences all the design decisions we made as well as functionalities that needed to take precedence over others. We are attempting to make an autonomous robot with routing capabilities that will allow it to successfully navigate based on the sensor data it picks up.

When deciding on the sensors that we would use, we thought at face value we could use a low-cost LiDAR sensor that would give us better performance at a 70\$ price point. However, it came at the cost that the lidar would not scan the full 360 degrees and we would need to provide redundancy for the sensors to scan the full environment. This would exceed our budget immensely and we would need to find an alternative that would not completely compromise the performance of the robot. Time-of-flight (TOF) sensors were the ideal candidate that were more cost-effective and used continuous infrared light to gather data from its surroundings. These will also be less complex to use and we can increase the amount of sensors that will be used for more accurate data at half the cost of the LiDAR sensors.

In terms of sensors such as the IMU and the camera, we went the low cost route since we do not need a strong camera to achieve what we want the delivery robot to do. The IMU that we opted in using would be a 6 axis MPU 6050 which is a more commonly used gyroscope to aid the sensor data for SLAM. This will only be about 11\$ for 3 sensors in comparison to a 9 axis IMU with the same functionality accounting for a greater scope at a higher price. For the camera, we are using a low cost Raspberry Pi V2 monocular camera that would perform basic image capturing to be used for our ORB SLAM.

These decisions were made based on the thought that the cost and quantity trade off showed that low cost sensors allowed us to have room to increase the quantity for better SLAM performance overall. Although with redundancy comes the consequences of getting a battery that will accommodate the voltage requirements of each sensor. We will be using a 45\$ 12V and 12A battery to power the motors, Raspberry Pi 5, camera, and PCB. This will be on the higher end since we are also getting a hub motor with encoder. For the majority of our other components, we have some readily available at no cost for example plywood for the main chassis of the robot as well as capacitors and resistors required for our PCB requirements. We also are utilizing the ORB SLAM libraries which will also be open source and free to use.

4.2.3 Power Constraints

Powering the robot is a critical design constraint as the system must efficiently operate on a limited energy supply. The robot will rely on a rechargeable battery, meaning factors such as battery capacity, power consumption of all the components, and energy efficiency will directly impact the performance, run time of the robot, and overall reliability. The robot must be able to complete its delivery tasks without unexpectedly running out of power. This prompts the team to work to select an adequate battery that is capable of powering all the components. Additionally, many onboard systems and components, such as motor controllers and sensors, require voltage regulation from the battery in order to provide stable power to the component without wasting any energy.

Many high power components and subsystems such as the motors, microcontroller, and single board computer can drain the battery quickly if not properly managed. Furthermore, the battery chemistry has to be taken into account too. While some batteries may be more ideal, for this project they may also pose a greater safety risk and thus not be used. Due to this, less efficiency batteries may be used and thus limit the design

capabilities of the robot.

4.2.4 Ethical and Social Constraints

This project as a senior design project has no major ethical or social constraints or impacts, however if this project were to be manufactured and sold as indented it could pose some issues. The ethical constraints and social constraints overlap and should be discussed as one instead of in two separate sections to avoid superfluous repetition.

The first ethical and social constraint is workforce displacement and job loss. If this project were to turn into a successful product that autonomously delivered packages in a warehouse then they would remove the need for a human to do this job. This is an ethical concern because the workers currently being employed will either lose their job. It will be the responsibility of the companies purchasing these robots to train and upskill their workers to learn how to operate, maintain, and oversee these robots rather than replace their workers altogether. This is a social constraint because many people who work in logistics may be resistant to a change that involves introducing robots as they may see it as their jobs being replaced. This will require educating these workers of how they will be taught how to work alongside the robots and how they can enhance the workers jobs and productivity.

The second ethical and social constraint about data privacy & security [E.5]. This project uses RFID on all payloads. Depending on how a company were to implement their system they could potentially be including sensitive information on each payload's RFID tags. If this were the case it would be an ethical concern if there was not properly secured communication between the robot, mobile app, and any cloud storage system most likely to be in place. It would also be a social concern if this information were not properly secured. It could put sensitive data at risk, leading to privacy concerns and potential misuse, and loss of trust among workers and customers. As a senior design project this will not be a concern as the only information on them will be simple information for testing purposes.

4.2.5 Health and Safety Constraints

The robot will be created around the standards that it would be safe to operate and have no impact on human health. The robot will mainly be operating in an indoor environment which will need to accommodate how it would navigate around objects and people. A significant constraint we face is the interaction between entities in the environment. By integrating the Time-of-Flight (ToF) sensors, IMU and camera, we can effectively use object avoidance to ensure the system does not bump into anything within 1m of it. Using the SLAM algorithm will also allow it to reroute to its destination to find its optimal path. This will further prevent any damages to the robot or any object around it during operation.

Another important consideration is safety to prevent electrical surges or outbursts. The robot will use a 12V battery-powered system to drive motors, sensors, and the Raspberry

Pi board. All electrical components must be properly enclosed and insulated to prevent short circuits, overheating, or accidental contact with live wires. Battery terminals will be secured within a protective casing, and appropriate fusing and voltage regulation circuits will be included to ensure stable operation and reduce fire or shock hazards. The battery itself must be safely charged and handled, following manufacturer guidelines to avoid thermal runaway or degradation over time. During the assembly and testing phase, safety procedures must be followed to protect team members from mechanical injuries, such as pinched fingers from moving parts or sharp edges on fabricated components. The robot design incorporates rounded edges and enclosed motor housings to minimize the risk of cuts or entanglement.

4.2.6 Physical Constraints

The physical constraints are warranted for our robot to navigate in an indoor environment. We have to account for robots being a reasonable size to navigate through hallways, obstacles, doorways, and around corners typically found in warehouses or laboratory spaces. From the design specifications, we decided that the robot needs to be about 60 cm x 60 cm x 16 cm, which can still allow it to maneuver across different spaces and will still be able to handle a standard sized payload. We also had to accommodate the components that will fit inside the robot which would be the RFID scanner, IMU, TOF, MCU and Battery.

Another physical constraint would be how the robot will be able maintain the weight of a given payload. Our requirements require the robot to be able to hold at least 25 kg without impacting the structural integrity of the system. We selected a wooden chassis to best handle this problem which would be able to handle the weight and be able to be sustained by our wheels and motors.

Considering the arrangement of the sensors will be a significant part of the robot that will have to work with these physical constraints. The TOF, IMU and the camera have to be mounted at specific angles and positions in order for VSLAM to work properly to navigate across the map. This will also have to be encased in the main chassis to avoid any exposure if the robot were to bump into something. When figuring out placement within the chassis, we need to be able to have an orientation that is easy to maintain and modality so that we can easily work inside the system. These physical constraints are directly tied to how we will build the system and accommodate the robot's ability to autonomously navigate.

4.2.7 Manufacturability Constraints

In terms of manufacturability, the main basis for this project is that we are able to obtain relatively available parts. Since this is not a sponsored project we are obtaining the different materials under our own means whether it is buying or borrowing materials that we already have. We want the robot to be able to carry the under 20 lb payload without exceeding the 60cm x 60cm x 16cm dimensions we have set for it. Throughout the

construction of the robot, we will need to address the design feasibility of getting the parts and being able to effectively construct it as well.

We have to account for manufacturability of the design of the main body that will use medium-density plywood. This will be obtained by one of our group members who already have it available from their home. This will offer significant structural strength without compromising on getting 3D printed parts or possibly an aluminum chassis. For mounting we will consider 3D printing using a PLA filament for the battery, PCB, camera, and wheels which will save time and be easily available within the engineering labs to develop. The Raspberry Pi and the ESP32-WROOM-32E are commercially available SBCs and MCUs that provide various libraries and support that will accommodate our functionality needs. The PCBs will be fabricated through JCL PCB who is a budget PCB manufacturer which will be initially tested on a breadboard to ensure that the board will work. The rubber wheels will be obtained from a hoverboard which will already have the motor connected to it that will be obtained from a third party site to make this project even more feasible. The DC motors attached to the wheels would be simple to integrate to adjust the mobility of the system. The main electronics of the system will be mounted under and inside the plywood body to facilitate heat dissipation and ease of replacement of parts.

The assembly process is designed to be straightforward and manageable using basic screwdrivers and drills, ensuring that all components can be easily assembled and maintained by us. Fastening methods will primarily include bolts and modular slotted designs, avoiding the use of adhesives or welding to allow for disassembly and iterative adjustments.

Given that we will not be constructing this in an industrial environment, tolerances must be designed to balance ease of assembly with structural integrity. The design will follow design for manufacturing (DFM) principles [25], including the use of standard components rather than custom made ones, minimizing the handling of positioning or orientation of the parts, and minimizing the number of parts to simplify construction. These are what we will abide by when constructing the system altogether.

4.2.8 Sustainability Constraints

When we were considering using VSLAM, we had to think about how this would affect the power being outputted by the raspberry pi and microcontroller. For this project, it is necessary to maximize energy efficiency for a system that can hold a 25kg load and run for more than 20 minutes. Sustainability was one of things we had to consider when we decided to go with a 12V battery. This will be able to power both the MCU and Raspberry Pi as well as the motors for the wheels that we are using. We want to optimize the power utilized by the system and still have enough time for the robot to go to its destination and back.

The construction of the robot also requires that we use materials that are able to be reused and environmentally safe. The use of wood and minimal 3D printing for the main chassis

and component mounts will help us accomplish this. In comparison to if we were to use metals or 3D print the whole robot, this was a better way to stay within these constraints.

Additionally, We designed it so the batteries and motors could be maintained or replaced without rebuilding the robot due to making the system more user friendly. This will reduce the need for rebuilding certain parts. Keeping the components in the body also enhances the overall durability and supports a longer lifespan and aids in the sustainability of the circuit components. We will also be using open source software libraries and programs that will help the longevity of the programming itself. Using these types of libraries will allow it to stay updated overtime and make it sustainable as you add more features as time goes on.

4.2.9 Testing/Presentation Constraints

For this project, the robot requires us to test and present its functionality within various constraints that are set. The foremost issue is connecting the Mobile app to the robot over a local wireless network (LAN). The reason is that several factors impact the reliability of a robot's ability to perform its tasks successfully. For example, take the network part of this project; several constraints include but are not limited to bandwidth, network security, load-balancing, scalability, managing traffic, and developing contingencies for desync scenarios. Furthermore, for mobile development, the battery life, intermittent network handling, reduced design complexity, and focus heavily on the hardware's constraints. Finally, the hardware components that make up the robot constraints are establishing an RTOS for Pi, reducing power consumption with network loads, and communicating between each part wirelessly to name a few.

While our robot is meant to travel autonomously within a controlled setting, in theory it can be used for greater challenges. For example, the robot could technically traverse busy streets during rush hour, and navigate I-4. However, given our inexperience in designing a 3-party system addressing autonomous payload delivery services, we are demoing a proof of concept within a controlled environment like UCF.

We affirm that the above constraints clarify why we chose specific tools for our project and help test the quality of service (QoS) that meets our standards. The tools include two categories: chosen languages and software applications used. The languages chosen are C, Python, and Flutter. We decided to focus on reducing complexity and increasing productivity by using what each of us has experienced and agreed on what will be deemed necessary. For example, all team members know C, many know Python, and none know Flutter. As stated before, our dominant obstacle is experience. That is why we opted to use what is necessary and what serves dedicated roles the best.

The primary constraint was our inexperience developing applications such as Mobile apps and Network interface. This is why we decided to use Flutter, as it was deemed an easy language for developing apps. The decision for Python was to design the Network interface. The two common languages used in Network applications are Python and Go. Python is easier to use and natively supported in Raspberry Pi, so for these reasons as

well, we decided to use Python. The C-language was chosen for low-level Microcontroller/other hardware components development, as given most experience developing low-level C-code for classes such as Embedded Systems. Given this experience, it became reasonable to integrate C-code into our overall hardware design.

The two software applications are AWS Amplify and IoT. The idea was to choose platforms that integrate well with each other, reduce complexity, and are compatible with Raspberry Pi 5. Both applications meet these requirements, so we decided to use them. Amplify is meant for App development, while the Internet of Things (IoT) was purpose-built for the Network portion. In pondering how to reduce the development cycle's complexity, finding the correct tools that are RP5 compatible was essential.

The testing portion becomes designing reliable test cases, ensuring each feature added is verified and proven to work on a theoretical scale. Designing scalable test cases for hardware, software, and the network portion is required for the robot's success. Python is known for rapidly prototyping test cases and supports packages to test three previous areas. The final step(s) is to pass the tests while continuously experimenting with the robot in a controlled environment with all components working together.

We believe this extensive explanation was necessary for users to understand the robot's constraints, capabilities, and purpose of being a proof-of-concept project. With further development, refinement, and expanded services, robots can become a successful product to launch that benefits many people by exceeding our previously defined design specifications. Consequently, improving the overall satisfactory experience clients have with our product.

Chapter 5: Comparison of ChatGPT with other Similar Platforms

5.1 Riley's AI Comparison

Through researching and understanding the working principles of RFID there were a few paragraphs of a source that were confusing. In one of these cases I put the paragraph into ChatGPT by OpenAI in the GPT-4o version and asked if it could explain it (Appendix E.1). The source I was looking at was focused on the difference between inductive and backscatter coupling [14]. The paragraph I was confused on was under the heading inductive coupling and explained how the reader communicates with the tag. ChatGPT's response had two sections. The first was a breakdown section that I would say somewhat successfully touched on and explained some of the key words and concepts from the paragraph. It correctly gave definitions to amplitude modulation, frequency modulation, and phase modulation. It also correctly determined that the RFID tags mentioned were passive tags that do not generate their own signal. However, in one place in the breakdown section and in the second section on why it's important, ChatGPT states that this method of communication is called backscatter communication. This is false. After that I no longer trusted the earlier parts of the explanation on what I was confused on and sought out other sources.

To compare ChatGPT to other similar AI platforms I put the exact same prompt into DeepAI's AI Chat (Appendix E.2). The result was a near identical format with numbering and bullet points. DeepAI also gave definitions in the same manner to amplitude modulation, frequency modulation, and phase modulation and also had a summary paragraph at the end of its response. DeepAI however, did not incorrectly identify the communication method as backscatter coupling. It did not mention what this method was called. Overall, I would say DeepAI's response was better and it did not give very obvious false information.

In trying to understand and clearly write about how capacitive transducers work in ultrasonic sensors I asked ChatGPT by OpenAI in the GPT-4o version and DeepAI to explain the concept to me (Appendix E.3 and E.4). Both AI chats gave me information that matches other sources and were both very similar to one another. The ChatGPT response included a formula for finding capacitance, but I think the DeepAI response had a clearer explanation of how a physical force converts to an electrical signal.

I am a visual learner so I wanted a picture of a capacitive transducer in action so I asked ChatGPT to draw me a picture. In short, this was really unhelpful. The drawing was confusing and the labels were not real words and were faded. DeepAI can not create images but described a simple diagram that I could visualize or draw myself. This was actually very clear and starting with the basic structure, where the dielectric material is found, and where arrows would go to indicate movement. Although this is helpful I think the pictures in my other sources with human drawn or created images were best at helping me visualize and understand this topic.

Another way I used ChatGPT by OpenAI in the version GPT-4o was when brainstorming points to write about for ethical constraints and impacts of the project. I described the goals and objectives of this project and asked what ethical constraints should be looked at. It gave me a list of key ethical considerations. The first point was "Worker Displacement & Job Loss". This was a point I was already planning on touching upon in the Social Constraints section but this influenced me to combine these two constraints sections because the key points overlap. The second and fifth points respectively were "Safety and Liability" and "Environmental Impact" and although these do overlap with ethical responsibility I think they should be left for the Health and Safety Constraints and Environmental Constraints sections of the report. The third point ChatGPT made was "Data Privacy & Security" and this was not something I thought about and I decided to include this in the Ethical and Social Constraints section of this report [E.5], [16].

5.2 Melyia's AI Comparison

My responsibility for research was to understand and compare the different SLAM protocols and determine which was optimal for an indoor environment. In the beginning of my research process I had resorted to ChatGPT to give me insight on which SLAM protocols were more ideal in certain situations. Our initial idea was to use only TOF sensors for SLAM however when I asked ChatGPT to "Give a recommendation for the

list of ToF-based SLAM libraries” (Appendix E.6) It outputted 4 different types of SLAM Cartographer, RTAB Map, Open VSLAM, and ORB-SLAM. I originally had researched cartographer SLAM due to us only using the TOF sensors however I did not find many sources that had only used one type of sensor most of them emphasized sensor fusion. ChatGPT-4o was able to offer some sources received from the IEEE site and helped make a decision on adding more sensors to the system. For example, “Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment”[32] was a great source in giving a comparison between VSLAM, LiDAR and other types that would work in the context of an indoor environment. This helped me point myself in the direction of trying to achieve greater accuracy as the robot is mapping out its environment. I compared this with my own search on TOF based SLAM and they listed out TOF sensors and the use of a monocular camera. The data matched with GPT-4o when I made an additional prompt to see its compatibility with the Raspberry Pi.

Another prompt, I asked was to “Make a comparison between Raspberry Pi and other microcontrollers. Why is it best for this project” (Appendix E.7). My goal was to gauge the ideal single-board computer that would be able to handle SLAM and be compatible with the peripherals we would be using for this project. It gave me a full detailed table of the possible single-board computers and microcontrollers that ranged from the Raspberry Pi, Nvidia Jetson, ESP32 and Arduino Mega. I had no prior knowledge on the components of each microcontroller so the ease of having the significant hardware specifications allowed me to narrow down the most optimal one. It compiled the full table CPU, RAM, power efficiency and much more as compared to reading each of their respective data sheets. This also included what microcontroller would be the most optimal, which would be the Raspberry Pi 5 since it would have a better handle on the computational load of VSLAM. It talked about how the real-time processing and wireless connectivity would be ideal since I mentioned that we would be integrating a mobile app. It also included microcontrollers to rule out this compared to my prior knowledge of other microcontrollers. The Nvidia Jetson Nano was mentioned on the list since it is more complex to use and it lacks the built-in WiFi needed for our application. In my research the cost of the Jetson Nano will be over budget if used. The Arduino Mega and the ESP 32 were not recommended since they are not made for SLAM and do not hold the same capabilities as a single board computer. This is factual throughout all microcontrollers and we need the real time capabilities for autonomous navigation.

Another prompt I created was to verify our use of TOF sensors with SLAM (Appendix E.8) in accordance with the original idea we had. Our initial thought was to use just the TOF sensors on our robot after LiDAR was not ideal for our project due to our price constraints. However, we had to fact check and it mentioned the drawbacks of only utilizing TOF sensors for navigating. This listed out what a TOF sensor is and why it is comparable to LiDAR for SLAM which kind of seem not relevant for the initial answer however, I created another prompt explaining why TOF sensors are not used and what approach should be used with VSLAM. I used another source “Real-Time Interference Artifacts Suppression in Array of ToF Sensors”[32] that talks of the various errors of TOF sensors that are caused by sensor parameters and properties or environment configuration. Which align with the majority of sensors and that is why redundancy is

important for increasing accuracy of sensor data. It did recommend that it was optional to use a camera and an IMU to help with tracking orientation and movement. This did influence my decision after watching a video “Combining Events, Images, and IMU for Visual SLAM in HDR and High-Speed Scenarios” where they tested different scenarios that an IMU with an event and standard camera would act in a dynamic range of movement. They proved that by combining the event camera and IMU they were able to gather significantly more information in order for the robot to autonomously navigate in a changing environment (lights on and off) [58].

5.3 Brandon’s AI Comparisons

The two models I decided to compare were GPT4.5 and O3-Mini-High. The contents of designing and figuring out how to integrate a network to the project was more challenging than I thought. Initially I thought the whole approach would be closely related to the networks class I am taking. While this is a somewhat true statement, it is also untrue due to one thing I overlooked: I am taking a theoretical framework class. So, everything I thought of initially ended up not being valid in terms of building the network off from scratch. Another issue was deciding how I was going to help recommend software to co-develop the app on flutter, and make it work as a cross-platform application. There were many questions that I had, and little ideas on where to approach. Some ideas appeared at random in the middle of writing, and I decided to really focus on them. One great example is the AGILE Scrum and TDD frameworks. Although not traditional standards for constraints, the methodologies are considered commonly practiced systems that restrict freedoms in the industry, and primarily focus on specific goals that targets how to finish work within teams. Not only finish the work, but finish work in a dynamically changing environment hence why I decided to include Scrum and TDD as part of the Design constraints sections. However, I got ideas or things I did not consider from ChatGPT to look into. Some ideas of which I did not think could happen when programming a Network interface, like a battery depleting its battery due to high network traffic. ChatGPT for both models helped me consider ideas that I did not think about. But importantly, it also helped me refine areas in my paper that I could talk about more.

The model 4.5 was better at giving factual recommendations of direct resources. I used it mainly to help me figure out what specific topics I was missing that can be added. For example, the unittest library from python, pytest, asyncio, and even asked it a question about multithreading as there’s a supported library too for developing parallel distributed threads over a network and it said this all works and gave me examples with detailed explanations. I cross checked all their recommended sources as well when I asked for it to give me some, and this especially happened with information that seemed more niche than generally distributed. Going back to the example with a phone's battery life, this is actually one example of this situation happening. GPT gave me an insight which I did not think about, with the worst thing being it did not give me sources when I asked for them after making the claim. This made me research deeper for any leads in this claim, and how poorly designed network interfaces cause worse performance on both hardware/software ends. Saving the suspense there was no direct information detailing

how or what can happen that leads to Pi's or other hardware components to burnout, or shorten their lifespan by poor design. There were greater articles focused on connecting Pi to a network vs. explaining why or what causes the worst case scenario. After thinking about this, I realized it made sense why that was happening. Common sense told me who would willingly expose the Raspberry Pi about their poorly developed interface, and tell someone on a published article on the internet what led to this happening. Regardless, if that person learns from their mistake, and corrects it, their reputation or reliability on not making the same mistake is lessened. This is the idea of maintaining a reputation, and if something happened we likely would briefly speak about it. Not going into deep with documentation detailing how/why this all happened. This forced me to find outside sources, cross compare them with the Pi using its official documentation, and eventually after a great amount of effort came to the same conclusion that Pi had made.

Whenever I used O3-Mini-High it was to help go about a creative avenue to figure out what to write for the Network part of my research. This is because GPT 4.5 as previously mentioned is better at factual reasoning, but not good at creative problem solving. This is where the later models come into play, and have the overall superior ability in terms of performance. I would ask a question such as "why do I not need to implement the Sliding-Window concept into the Network interface?", "how come we really don't use the OSI layer versus the TCP/IP protocol? Is it because OSI uses UDP? Or because the theoretical framework was deemed impractical?", "What is the Polling concept you mentioned earlier, and why is this a problem in development? In addition, when you mentioned we do polling this implies it's a technique we use, but say it can become a problem over the network? How come you make these claims?", and other questions like this. I mainly used O3-Mini-High while cross checking their answers to check if they matched up with the claims they told me. Given that I have access to them because I own a premium membership, I can say in comparison to the free versions, the later models at least 4.5 and O3-Mini/High are very accurate in their responses. However, this is to be taken as the exception. ChatGPT improves the more the user interacts with them. I normally play around with GPT, do my research on best practices or ideas on how to type prompts into it, and use it in a way that helps me learn while boosting my overall productivity. I even on occasion ask it to give quick performance feedback on areas of improvement, ask it to be analytical/direct, or very explanatory like how the Nobel prize winner Richard Feynman explained material. Anyways, the point being that I am very used to ChatGPT, the model has learned a lot of ways how I think of information, and I figured out naturally before starting this project where I discovered the false-rate/specific usages that certain models excel over others. In addition, I also knew in general despite how good these models are, depending on how we write our prompts it makes more mistakes the more confusing a prompt becomes. Like everyone else, I had my fair share of experiences with this too, and even this occurred on my project when asking it questions about AGILE/TDD being considered a design constraint. Initially it said yes, but more than once it said no. However, after doing research it is considered a Design Constraint due to the nature of how Scrum (the framework I chose), to work on a project while restricting usages of different resources down to a few items for the planned Sprint is the definition of a Design Constraint. Despite it creatively helping me solve multiple problems, and it being correct around 90% of the time from personal experience, it is at

times incorrect. Which then implies if a later model is wrong sometimes, and the article from OpenAI themselves mentioned false-rates per model, this means there is a chance that GPT4.5 is wrong too despite its intended purpose. Overall, this was how I engaged with AI to help me throughout the research process for the Autonomous Payload System.

5.4 Ibrahim's AI Comparison

While conducting my research, a large chunk of it consisted of researching different motor types, their working principles, and their control methods. While this stuff was generally within my understanding with some research, I fell behind on the mechanical understanding of them. Namely, when determining specifications for the motor (i.e. torque), I failed to understand the necessary calculations and factors that came into play.

This is where I first took to using AI. I asked two different LLMs about motor requirements and torque calculations and got two very different results. The two LLMs I used for this comparison were OpenAI's GPT-4o and DeepSeek. With no previous context, I gave the AI some basic requirements for the system and asked it to calculate the torque necessary if two motors were used [E.8]. When asked this, GPT-4o gave a shorter, seemingly more ideal response, not taking into account any real world factors such as friction and giving an answer of 10 Nm. However, DeepSeek gave a much more in depth response, providing more detailed calculations and accounting for external forces such as friction, giving an answer of 0.3481 Nm. They also both started off with different assumptions, other than the given information. GPT-4o assumed an acceleration of 1 m/s^2 whereas DeepSeek assumed an acceleration of 0.5 m/s^2 . They also assumed different wheel radii, with GPT-4o assuming 10 cm and DeepSeek assuming 5 cm.

I then told GPT-4o to assume a radius of 5 cm and acceleration of 0.5 m/s^2 to match the initial conditions of DeepSeek. Upon prompting GPT-4o to change the parameters to match, it gave a torque value much closer to that given by the DeepSeek model, 0.25 Nm. The discrepancy between the two values can still be attributed to the fact that DeepSeek took into account friction and other natural forces, making it more realistic for a real world application.

Another instance I used AI for was again for the motors. I needed to fully understand how to drive the motors and control them properly, being able to change speed and direction through the MCU. I again prompted both models to explain to me how to drive a brushed DC motor using an MCU and a motor driver IC [E.9]. Both models had a lot of overlap in this prompt. They both followed the same general process: first explaining the components needed, explaining how it works, and then explaining the hardware connections. DeepSeek went more in depth again on the explanations of how it works, explaining all the pins on a motor driver IC and how to control it using the MCU. GPT-4o also explained this but it was a more surface level explanation.

Both models also listed a couple of examples of MCUs and motor driver ICs, listing the same ones. When it came down to actually explaining, they both assumed the use of an Arduino. They also both provided me with code to control a motor and motor driver

using Arduino. GPT-4o provided 2 sets of code: basic on/off code and speed control using PWM. DeepSeek provided me with testing and debug guidelines to ensure everything works and warned me about overheating. Both models provided me with different tips and notes at the end of the output, with GPT-4o seemingly focusing more on protecting the circuit itself and DeepSeek focusing on both protecting the circuit and the user.

Overall, after experimenting with both models, it would seem that DeepSeek takes a more analytical approach to answering prompts than GPT-4o does. It will often take into account real world factors and assume more realistic values, whereas GPT-4o will assume ideal values and neglect real world factors to do any calculations and analysis.

Chapter 6: Hardware Design

This chapter focuses on the hardware design aspect of the robot. It provides a detailed overview of the different hardware components and their integration to provide the desired functionality.

6.1 Subsystem Block Diagram

The subsystem block diagrams provide a more detailed breakdown of the key functional components of the robot. While the overall project block diagram seen at the beginning of this paper in figure 2 illustrates the high-level interactions between the major hardware components, the subsystem block diagrams focus on specific parts of the system. Namely, the motor controlling process, sensor integration and communication, and voltage regulation and power distribution. Each subsystem block diagram highlights the internal connections and data flow within different parts of the robot's hardware. By examining the subsystems individually in this way, we are able to gain a deeper understanding of how each component functions and how they each contribute to the overall functionality of the robot.

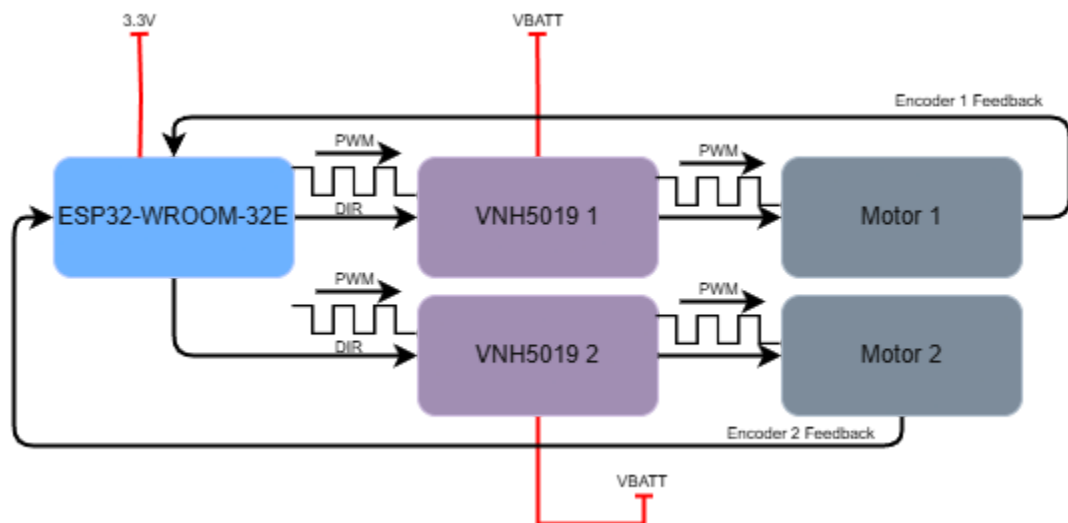


Figure 5: Motor Control Subsystem Block Diagram

Beginning with the motor control subsystem, the concept is fairly straightforward. There are two voltage inputs, one for the MCU to control the speed and direction through PWM and logic pin. The other for the motor controller ICs to actually drive the motors with the necessary power.

The MCU sends a PWM signal to each of the motor controller ICs that are responsible for controlling the speed. The ICs then generate an appropriate signal using the battery voltage to drive the motors. The direction of the motor is determined by a pin on the IC whether it is driven HIGH or LOW by the MCU.

The encoders built-in on the motors will provide feedback to the MCU to control the speed and ensure consistent movement.

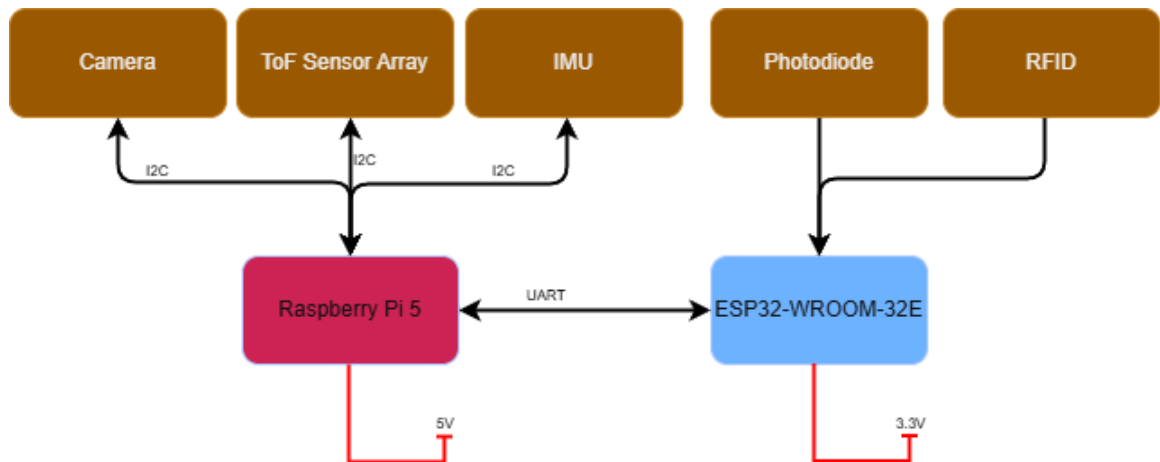


Figure 6: Sensor Subsystem Block Diagram

For the sensor subsystem, the sensors either communicate directly with the MCU or the Raspberry Pi 5. The sensors that are integrated with the Raspberry Pi communicate via I2C. The sensors go through a multiplexer on a custom PCB HAT that attaches to the GPIO pins of the Raspberry Pi. These sensors are used by SLAM for object avoidance and rerouting purposes.

The photodiode and the RFID are not used by SLAM and do not communicate via I2C. They are only needed for payload detection and identification. Any necessary information will be communicated between the MCU and the Raspberry Pi via UART.

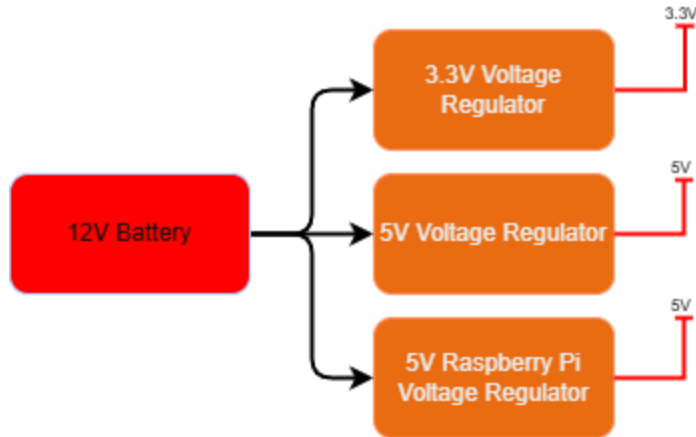


Figure 7: Power Delivery Subsystem Diagram

The power delivery subsystem starts with the 12V SLA battery. The battery connects directly to the 3 different regulators that then output their respective voltages. Outside of this subsystem, the 3.3V and 5V outputs are distributed to all of the sensors and the MCU. The 5V from the Raspberry Pi voltage regulator is directly sent to the Raspberry Pi to power the SBC.

6.2 Schematic Diagram

This section breaks down the hardware design of the robot and presents the complete schematic. The schematic is divided into several key sections that allow for the engineer to easily understand it. Each section corresponds to different functions of the system, including the microcontroller, power management, and communications. Additionally, there are several PCBs that are separate from the main PCB for different reasons. Overall, the schematic has been designed with ease of use, functionality, and modularity in mind.

6.2.1 Power Delivery

The entire system will be powered by the 12V SLA battery. Multiple voltage regulators will be used to provide the necessary power to different components in the system. The MCU will be powered by one of these regulators, at 3.3V. This means that all of the logic of the MCU will also operate at 3.3V.

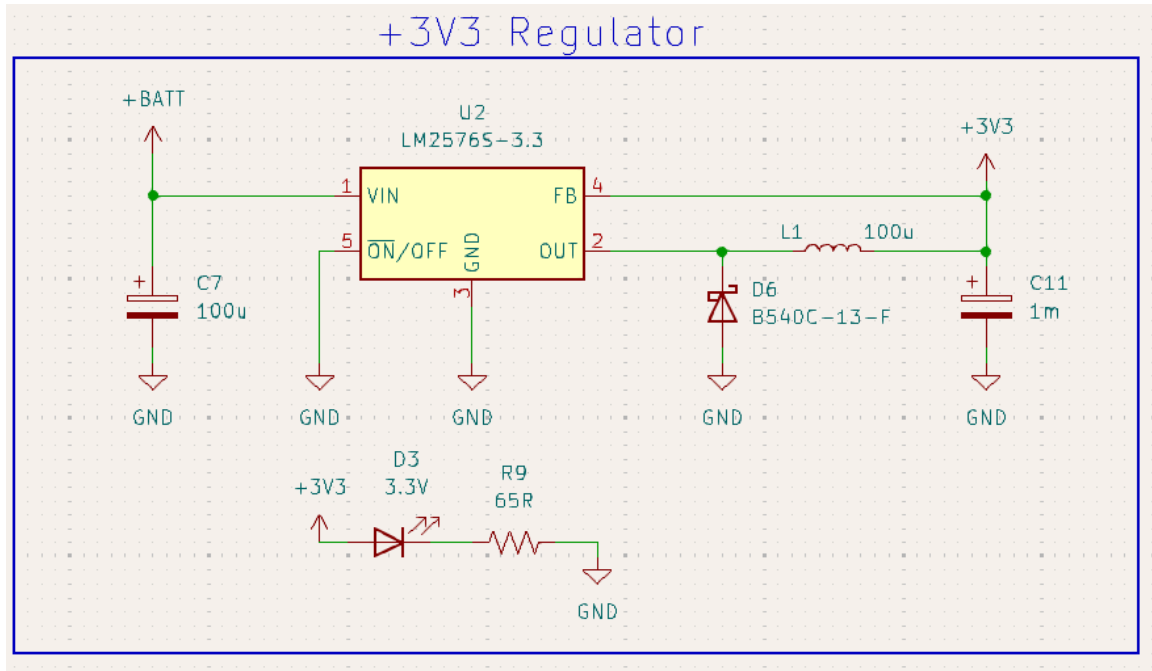


Figure 8: 3.3V Regulator Circuit

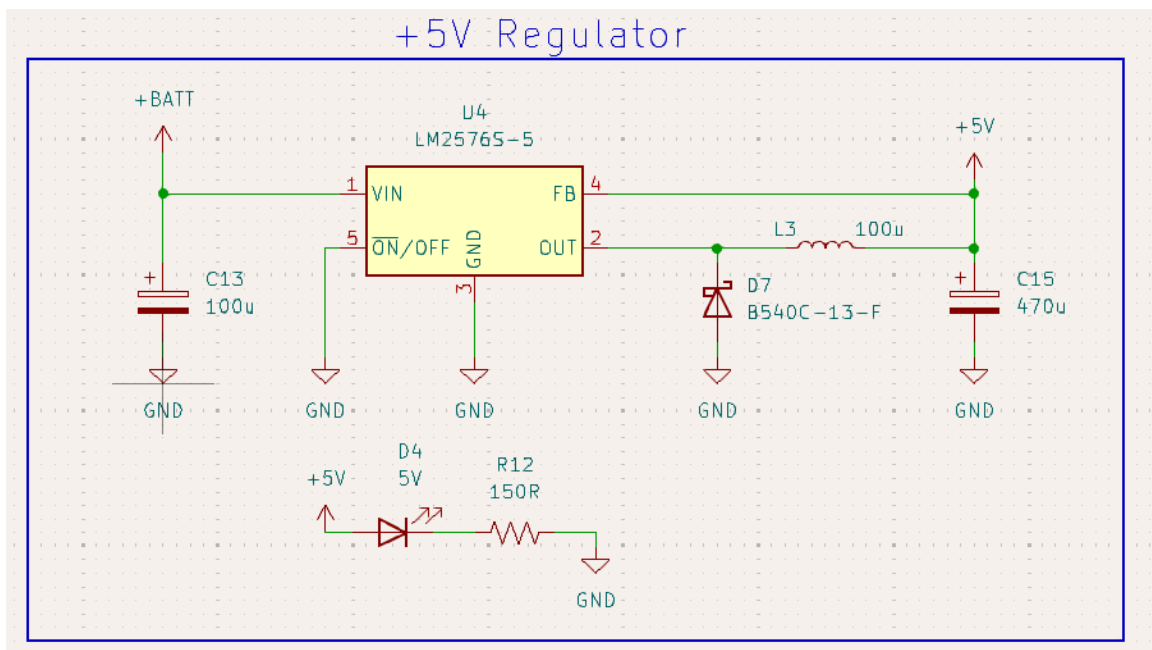


Figure 9: 5V Regulator Circuit

For both the 5V and the 3.3V voltage regulation circuits, a LM2576 regulator was used. The variable regulation of the component makes for easy planning and efficient use. The output voltage is determined by the component values and the footprints will be identical. Both regulator circuits also include a built-in status LED to show visual operation of the output voltages. The inclusion of such status LEDs simplifies the troubleshooting process. The series resistor values for the status LEDs were calculated using Ohm's law.

$$\text{Equation 7: } R = \frac{V_{cc} - V_f}{I_f}$$

With R being the resistor value, V_{cc} as the supply voltage, V_f as the forward voltage of the LED, and I_f as the forward current of the LED.

On the other hand, the Raspberry Pi 5 has much higher power demands and needs its own, high power voltage regulator.

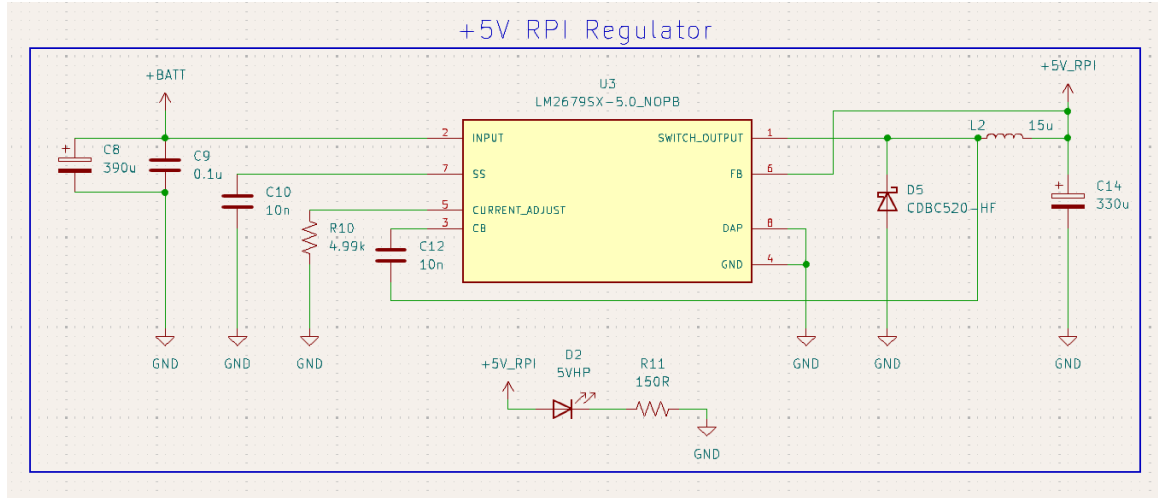


Figure 10: 5V Raspberry Pi 5 Regulator Circuit

Since the power demands of the Raspberry Pi 5 are potentially so much higher than those of the rest of the system, excluding the motors, it requires a different regulator IC. Using the LM2679, the circuit seen in Figure 10 was designed. Similar to the previous regulators, a status LED with the appropriate series resistance value was included to indicate operation. This circuit is capable of providing up to 5A of current and the Raspberry Pi 5 requires at most 5A depending on usage. No other device will be powered directly from this regulator to avoid overcurrenting the regulator and damaging the circuit and the Raspberry Pi.

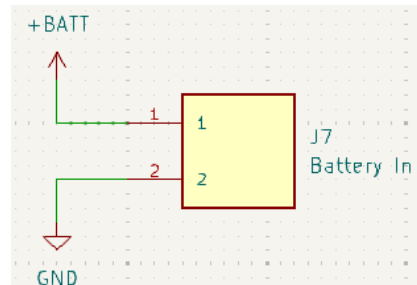


Figure 11: Battery Input Terminal

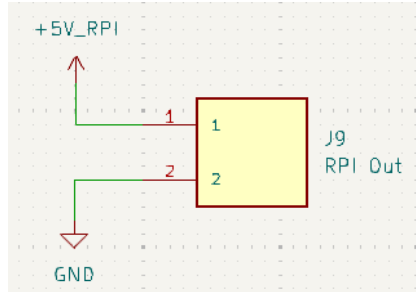


Figure 12: Raspberry Pi Output Terminal

The battery will connect directly to a 2 pin screw terminal capable of handling the current and voltage requirements of the system. This terminal then provides the input voltage to each of the regulators, allowing them to output the necessary voltages.

6.2.2 Motor Controls

The motor control schematic is a crucial part of the robot's hardware design, responsible for driving the DC motors that enable precise movement of the robot. Since the robot has two motors and they need to be individually controlled, two motor driver circuits were designed.

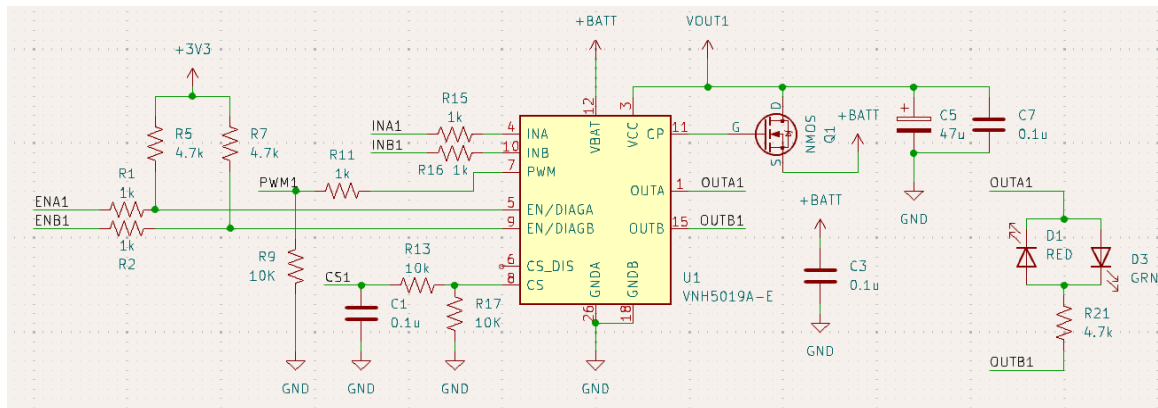


Figure 13: Motor Driver Circuit

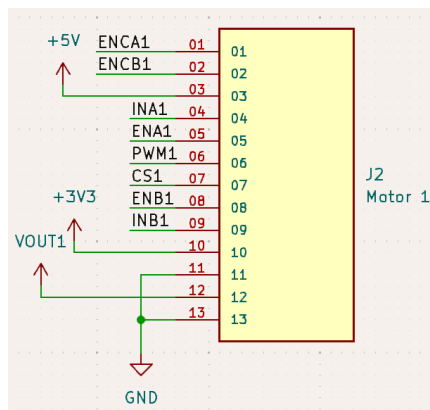


Figure 14: Motor Driver Header

Each motor has a driver circuit identical to the one seen in figure 13. The circuit takes a voltage input from the battery and interfaces with the MCU to control speed and direction of each motor. Each motor also has its own header pins to enable precise control, seen in figure 14. These pins do not have to handle high power so they are standard berg connectors. The MCU will connect to this header using an appropriate cable to provide the required inputs and outputs. The PWM will come from the MCU to control the speed of each motor, labeled PWM1 in figure 14. The direction of the motor is controlled by the INA and INB pins of the driver. These pins allow the motors to change direction as well as break, depending on the input combination, either logic HIGH or logic LOW, a total of 4 possible inputs per motor.

Similar to the previous schematics, there are 2 indicator LEDs for each motor driver circuit. These LEDs will light up depending on the direction the current is flowing. In other words, if the motor spins clockwise, one LED turns on; if it spins counterclockwise, the other LED turns on. This is extremely useful as it can be used to test that the motor driver outputs are working as expected before connecting the motor to the circuit. These LEDs can also be used to detect problems and troubleshoot as needed. Since the motor speed is controlled with PWM, the LEDs will also change brightness depending on the speed of the motor.

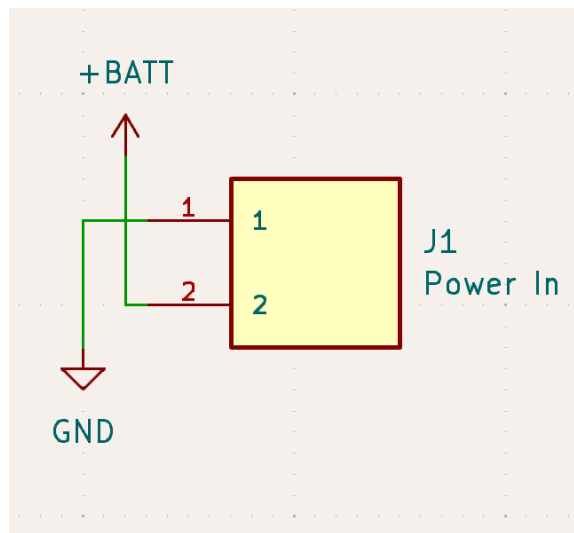


Figure 15: Motor Terminal

Due to the high power demands of the motors, standard connectors cannot be used for the power delivery from the battery. A 2 position screw terminal block capable of handling the high power was used to connect directly to the battery, providing that power to the motor controller ICs.

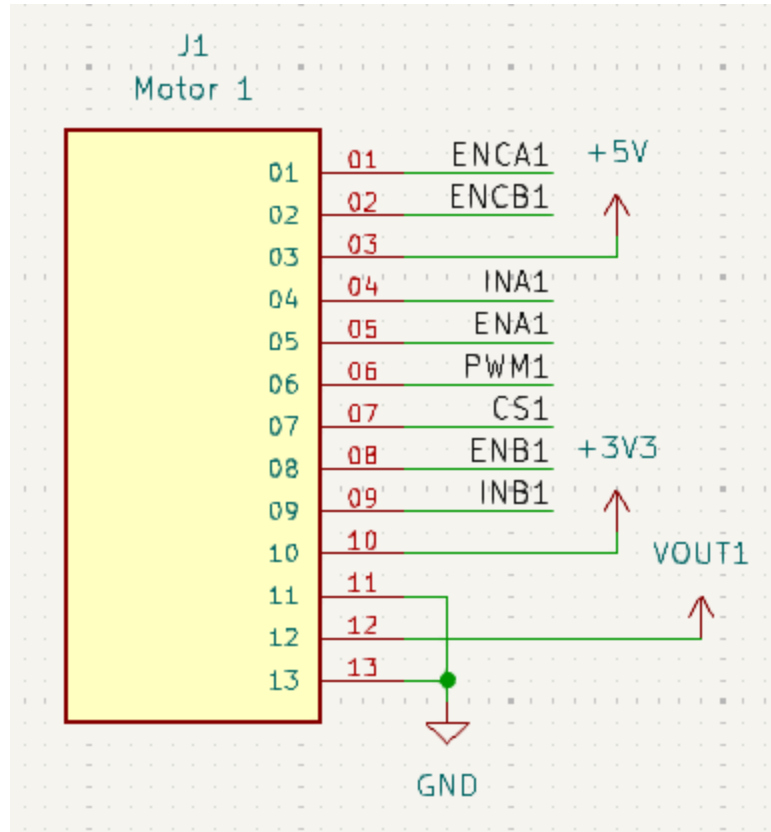


Figure 16: Motor Header on Main PCB

Due to the high power demands of the motors, the motor driver board will be a daughter board that connects to the main PCB using the connector seen in figure 14. This is done to reduce the risk of failure and damage to other components. Figure 16 shows the matching header connector on the main PCB.

6.2.3 Raspberry Pi 5 HAT

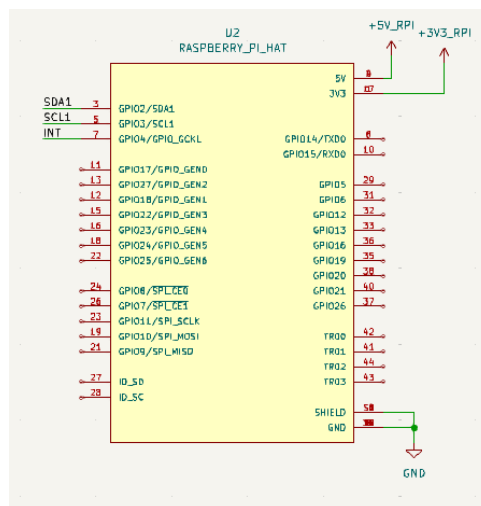


Figure 17: Raspberry Pi 5 HAT

Another separate PCB was designed for the devices that are installed directly on the Raspberry Pi 5. The PCB is designed to be a HAT (Hardware Attached on Top), plugging directly into the Raspberry Pi's 40 I/O pins and allowing the components to interface with it. The devices installed on this board are used for SLAM, providing the necessary data through the use of various sensors. All of the required sensors communicate using I2C. With a total number of 4 devices, 3 of which sharing the same permanent hardware address, the Raspberry Pi does not have enough buses to communicate with all of the devices, and it removes the possibility of adding any additional I2C devices in the future.

In order to overcome this and enable proper I2C communication with all of the devices on the Raspberry Pi, a multiplexer was used.

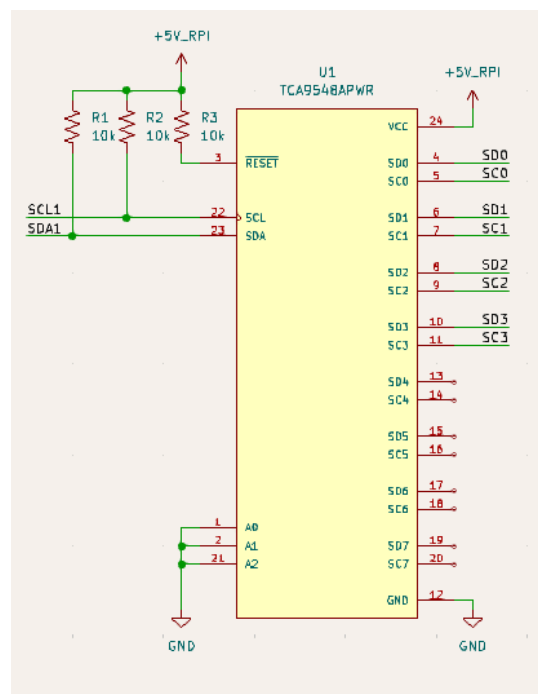


Figure 18: TCA9548APWR 8:1 Multiplexer

Seen in figure 18 is the TCA9548APWR 8:1 bus switch, which can control up to 8 other I2C buses while using only 1 bus on the main controller, the Raspberry Pi 5 in this case. This IC is included on the HAT that fits onto the Raspberry Pi 5. The inclusion of the multiplexer resolves any I2C target address conflicts.

The devices installed on the HAT include 3 ToF sensors and 1 IMU sensor. These 4 sensors connect to the SCL and SDA lines of the multiplexer and allow the Raspberry Pi to communicate with them properly.

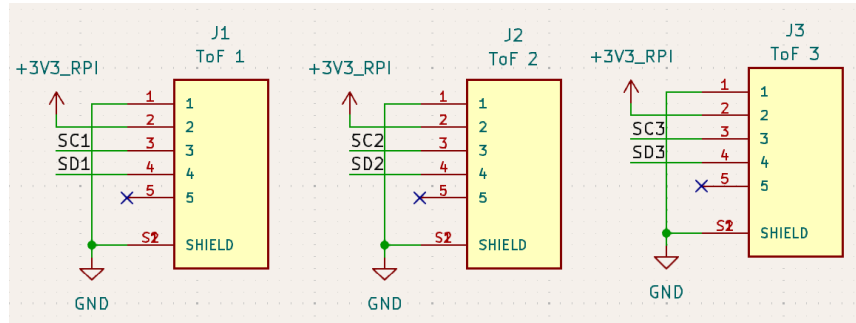


Figure 19: ToF Sensor Plugs

Each of the 3 ToF sensors must be mounted on the front of the robot to be used for object detection. Thus, they cannot simply sit on the PCB directly. Each sensor will connect to a cable that will plug into the HAT and break out the pins. Since each ToF sensor has an identical address, they will occupy one bus each on the multiplexer.

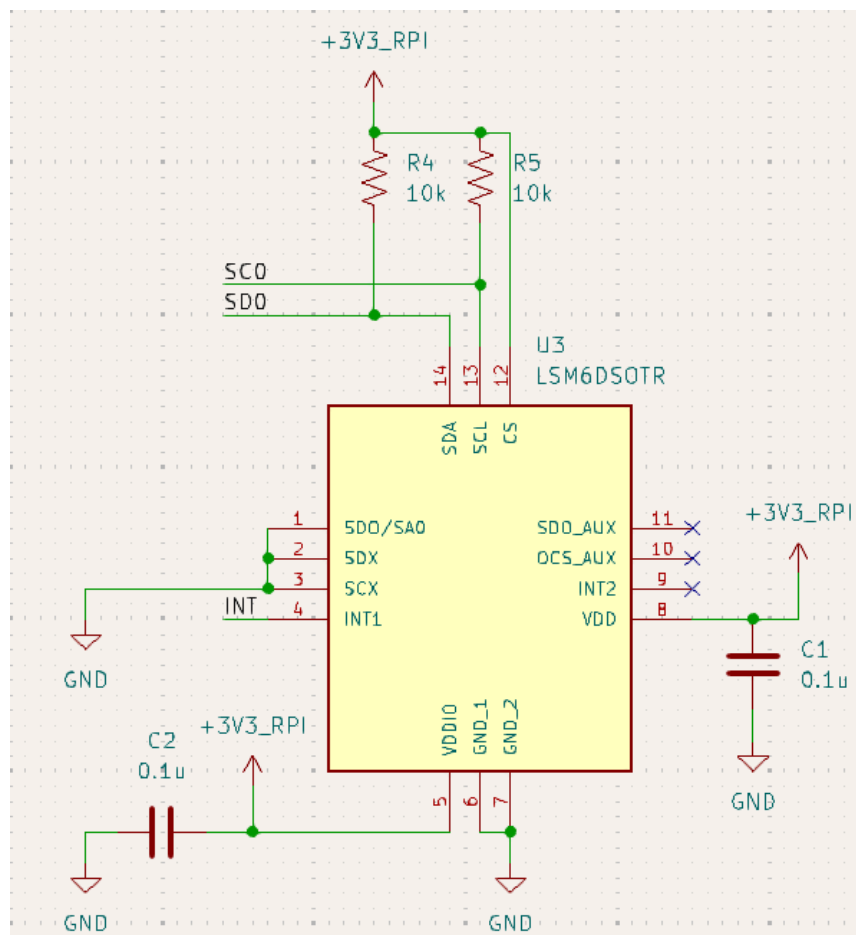


Figure 20: IMU Sensor

Similarly, the IMU sensor will communicate via I2C and connect to one of the I2C buses from the multiplexer.

6.2.4 RFID Sensor

The final part of the schematic is another daughter board, responsible for breaking out the RFID sensor module. The RFID module comes with 0.5 mil pitch header pins installed, as opposed to the standard 1 mil pitch. The PCB for this module consists of 18 pins to solder the module directly to, which then breaks out to a 10 pin 1 mil pitch header. This header will be connected by wires to a matching header on the main PCB to then interface with the MCU. The RFID module cannot be directly mounted on the main PCB because the sensor must be oriented and placed in a proper location to be able to read the RFID tags on the payloads.

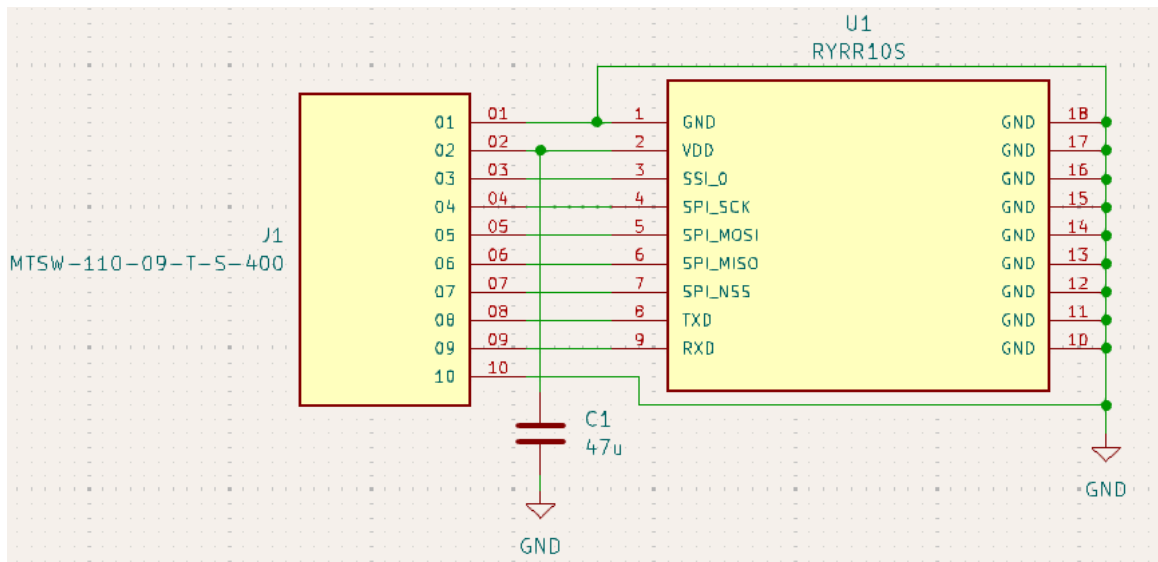


Figure 21: RFID Module Breakout Board

The RFID module supports communication via both UART and SPI. However, due to design constraints and simplification of the PCB, the implementation we opted for was for SPI communication. To achieve this, the SSI_0 pin from the RFID module, which determines the communication protocol, is permanently pulled high to 3.3V, placing the module in SPI mode by default.

Previously, a slide switch was considered to toggle between UART and SPI by manually altering the SSI_0 logic level between high at 3.3V or low to ground/0V. However, as mentioned, due to the limited number of UART channels, supporting both communication protocols became impractical. As a result, the schematic was simplified and made more compact with the fixed functionality of the design.

All of the relevant signals – SPI1_NSS (chip select), SPI1_CLK (clock), SPI1_MISO (Master In Slave Out), and SPI1_MOSI (Master Out Slave In) – were routed directly from the MCU to the RFID connector, allowing the team to just plug in the wires in accordingly and use the module. A 47uF decoupling capacitor was also placed near the power input to provide stability during RFID operation. The module is powered with the 3.3V regulated supply line, ensuring safe and stable operation.

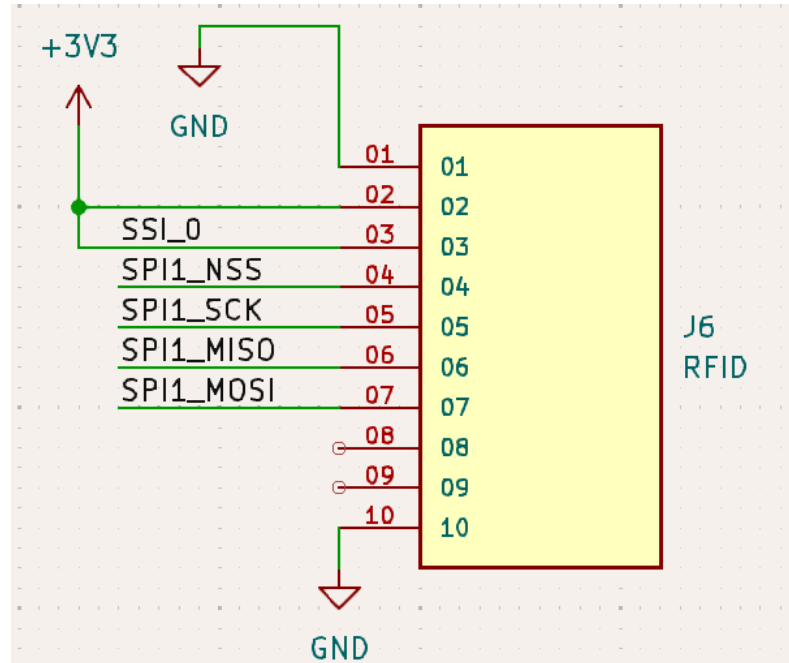


Figure 22: RFID Header and Configuration on Main PCB

6.2.5 Main PCB

The main PCB houses the MCU along with the supporting components and any additional necessities. The MCU was designed to have any capabilities as needed.

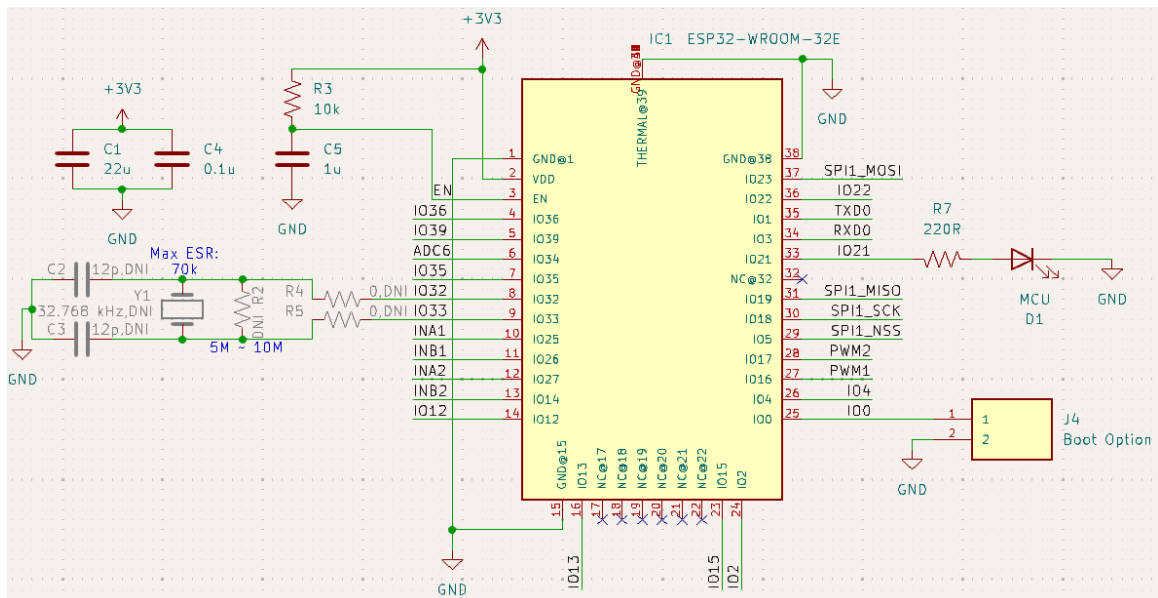


Figure 23: Microcontroller Circuit Schematic

The base schematic for the MCU was built in accordance with the ESP32's datasheet. The design starts with a simple RC circuit, a 10 kΩ pull-up resistor on the enable (EN) pin and a 1 μF decoupling capacitor, ensuring the MCU starts up reliably with a stable

voltage supply. Rather than a traditional external high-speed oscillator, the ESP32 contains an internal crystal oscillator and does not require an external clock for normal operation. However, the design includes a 32.768 kHz low-speed crystal oscillator footprint that is not populated in the case that real-time clock (RTC) support is needed in the future. The corresponding capacitors and resistors are also marked as DNI (do not install), leaving the footprints empty on the PCB when manufactured for future use.

A set of decoupling capacitors are placed near the power supply rail to ensure noise suppression and smooth voltage delivery to the MCU and other peripherals. These capacitors filter out voltage ripples and transient noise that may affect the performance of the MCU.

An onboard LED is connected through a resistor to one of the GPIO pins of the MCU, which will be programmed to blink during operation. Once manufactured, this LED will serve as an indicator for monitoring code execution and proper functionality.

A boot option header is connected between IO0 and ground, allowing the module to be manually placed into flashing mode. Pulling IO0 low during reset or powering up enables programming via UART, which is crucial to allow the team to use the MCU.

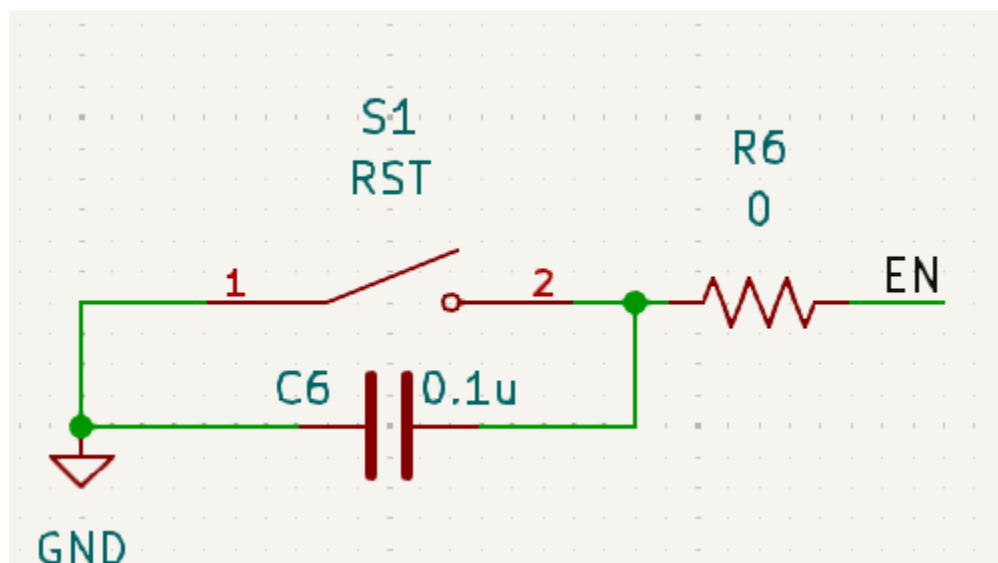


Figure 24: Reset Button Configuration

The schematic also includes a manual reset button connected directly to the EN pin of the EPS32. Pressing the button momentarily pulls the EN pin to ground, causing the MCU to power down and immediately reboot upon release of the button. This is useful during firmware updates, troubleshooting, and uploading new code.

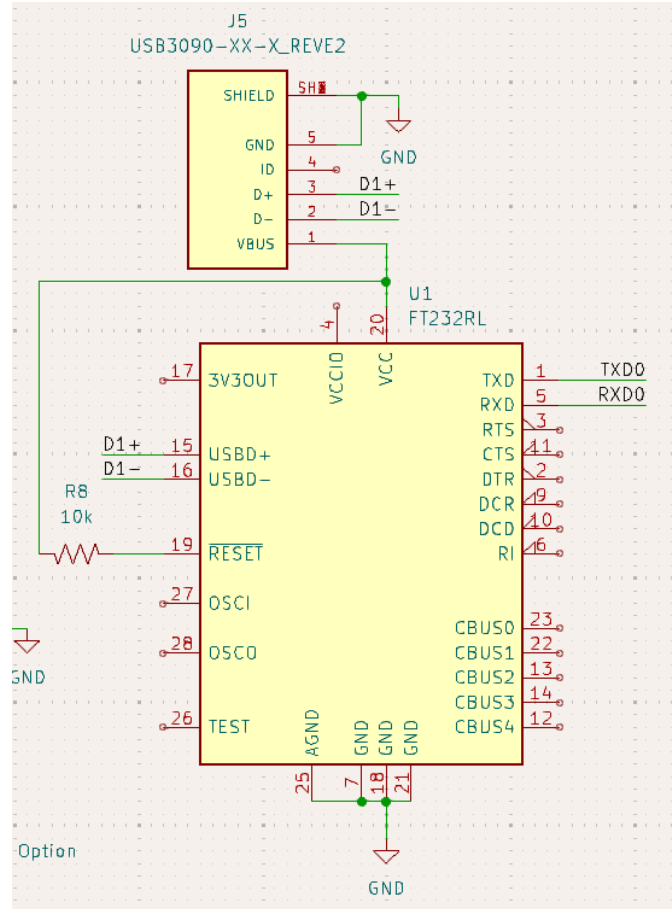


Figure 25: USB-to-UART Interface

In order to program the MCU and establish communication with a host device like a computer, a USB-to-UART interface was implemented using the FT232RNL chip. This IC converts USB signals to Transistor-Transistor Logic (TTL) level UART, allowing for direct communication between a computer and the MCU without the need for any external boards.

The transmitter and receiver UART pins, TXD and RXD, are routed from the ESP32's own UART pins, TXD0 and RXD0, forming a serial connection between the IC and the MCU. This allows for both programming the ESP32 as well as monitoring debug logs via a serial terminal on a computer.

A micro USB connector is used as the physical interface for the user on the board. The USB data lines, D+ and D-, are routed between the micro USB and the FT232RL. The VBUS line from the micro USB is connected to the VCC pin of the IC, providing power to the chip when a USB cable is connected.

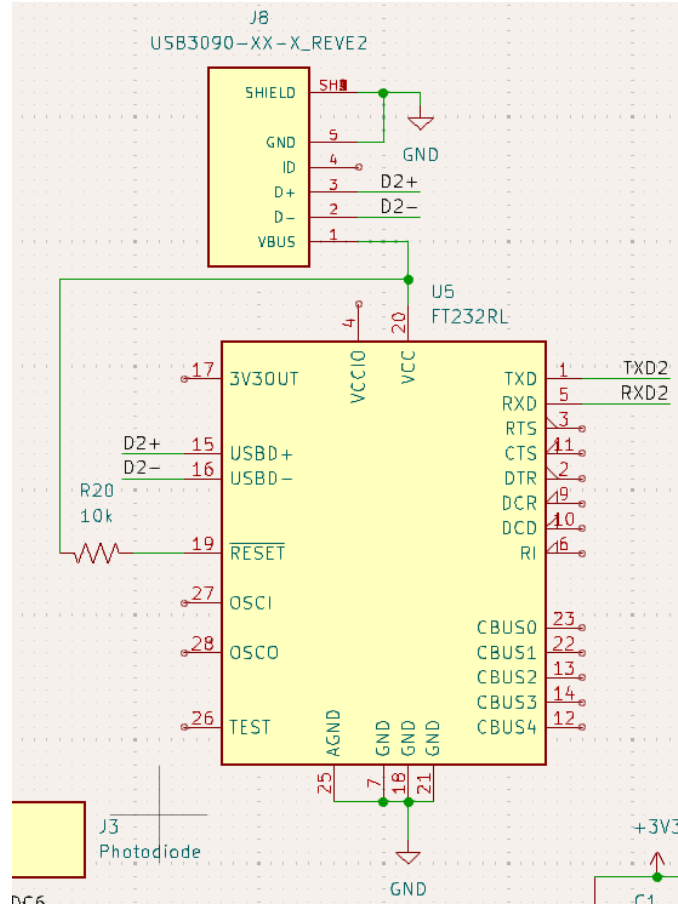


Figure 26: USB-to-UART Interface – Raspberry Pi 5 Connection

Similarly, an identical circuit was designed in order to have a second micro USB slot that would be plugged into one of the existing USB slots on the Raspberry Pi 5. This will allow for communication between the two devices and for them to share any necessary data with one another.

6.3 Architecture

The architecture of the system defines the structure and interaction between its hardware and software components. The Raspberry Pi will handle most of the high-level processing while the MCU will serve as the central processing unit (CPU) of the robot, handling data, processing inputs, managing communication, and controlling the motors.

The Raspberry Pi will handle and control all of the I2C sensors. This is due for two reasons: SLAM running on the Raspberry Pi needs to interface directly with those sensors, and there are not enough I2C buses to handle the sensors. Since some of the sensors share addresses, separate I2C buses are needed. However, the Raspberry Pi may run into the same problem. Thus, a multiplexer was integrated onto the Raspberry Pi so it communicates with that instead of the devices directly. The multiplexer allows the

Raspberry Pi to control up to 8 I2C buses worth of I2C devices. This gives us plenty of space to integrate the sensors with identical addresses.

6.4 Structural Illustration

Since the robot is designed to be autonomous and carry payloads, it must have a flat, rigid surface to carry the payloads. The robot will be shaped as a box with sides measuring 60 cm x 60 cm. The chassis will be constructed out of sheets of plywood, cut to size. It will be screwed together to ensure sturdiness. The robot must also not shake or vibrate with movement, so internal support beams will be placed across the chassis. This will prevent any vibrations from causing issues with the sensor data.

In order to maintain balance, the robot will have 3 wheels in total: 2 motorized wheels and 1 caster wheel. The motorized wheels will be along the sides towards the back of the chassis. The caster wheel will be smaller and directly under the chassis towards the front, creating three points of contact with the ground. The caster wheel will be smaller to keep the robot level and parallel with the ground.

Atop the chassis there will be four extruded corners as a guide on where to place the payload to ensure the RFID tag can be properly read by the sensor. Towards the front of the chassis there will be 3 angled panels with a ToF sensor mounted perpendicular to each panel facing forward, making a total of 3 ToF sensors. On the middle panel will be the Raspberry Pi camera module that is used for the visual part of SLAM. The RFID module will mount along the back of this pillar, facing backwards and ready to read the RFID tags from the payloads.

The hollow inside of the chassis will house all of the remaining electronics, including the Raspberry Pi 5 and the PCBs. The chassis will have a removable top to provide adequate access to the electronics for any maintenance and troubleshooting necessary. The bulk of the weight will come from the battery and will be placed appropriately to balance the robot. The battery terminals will be easily accessible to charge and replace the battery as needed.

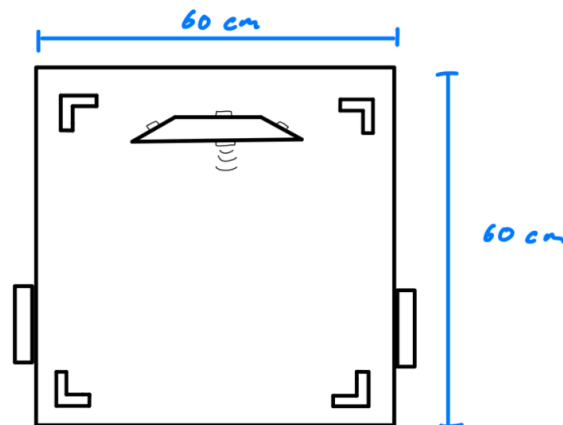


Figure 27: Top Down View of Robot Diagram

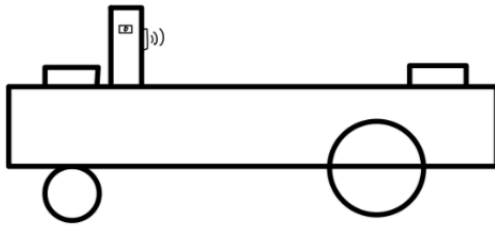


Figure 28: Side View of Robot Diagram

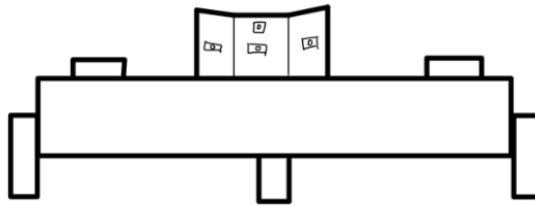


Figure 29: Front View of Robot Diagram

Chapter 7: Software Design

7.1 Activity Flow Diagram

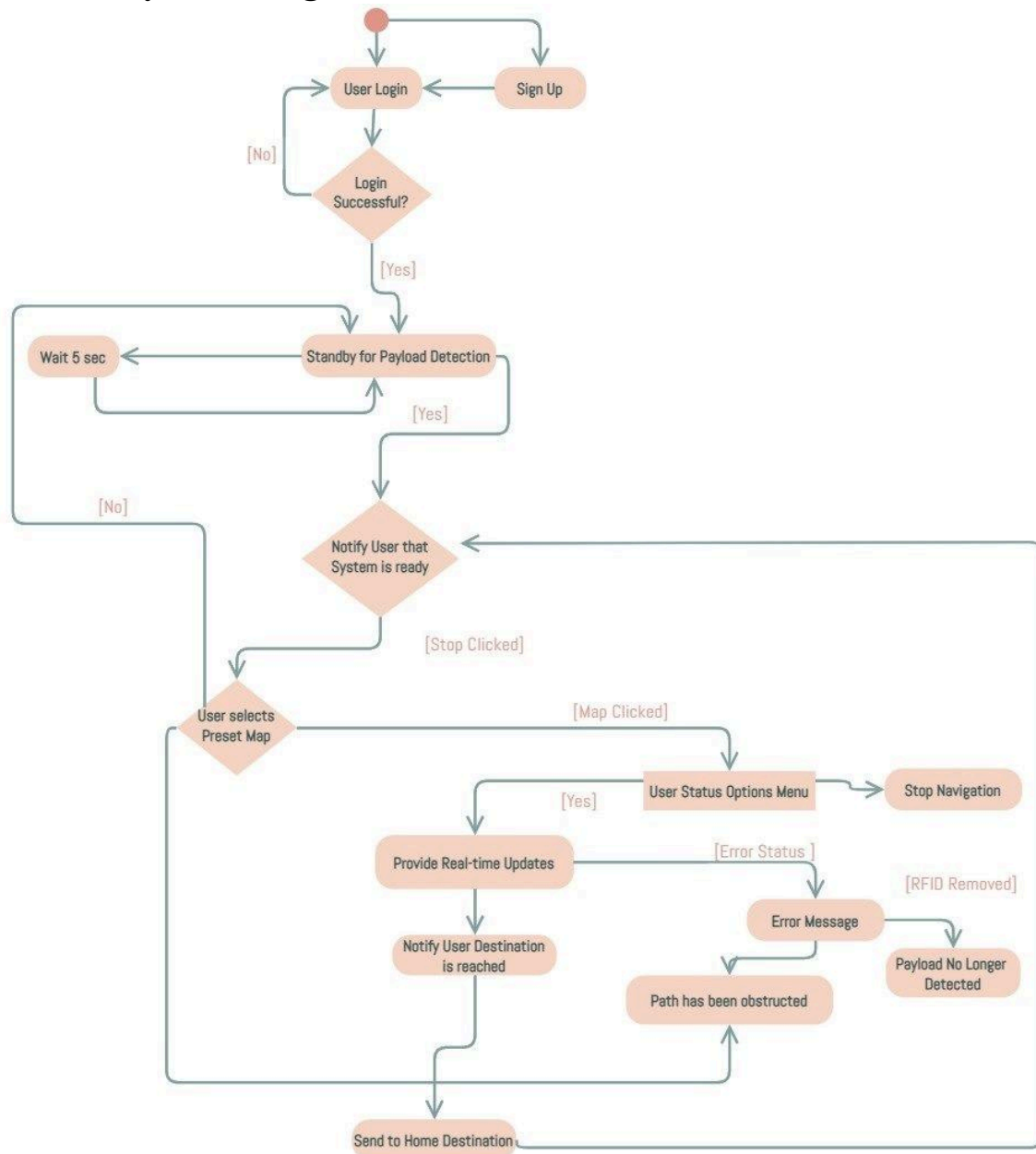


Figure 30: Activity Flow Diagram

7.2 Use Case

Figure 31 is a use case diagram and it shows the relationships between each functionality of the robot, the communication protocols being used, each sensor being used in the system, the user, and the mobile app. The robot is initiated by a payload being placed on the top of the robot by the user. It then will identify the package by first using the payload

detector (photoresistor) to indicate that there is a package, and then use the Payload Identification (High Frequency RFID reader module) to read the payload's RFID tag. This then leads into the next task of generating the robot's route using ORB SLAM. Both of these tasks include a UART connection to relay information between the Raspberry Pi and the microcontroller. After the route is generated with SLAM the robot moves into the movement state. This state will use the IMU and camera to properly determine its exact movement. This state also includes updating the robot's status on the mobile app which includes a wifi connection to the mobile app. While in the movement state, the robot will also be continuously checking for objects which will be accompanied by the camera and the object detector (optical time of flight). While continuously checking for objects the robot might detect an object within 2m. In this case it will move into the noMovement state. This will include generating a new route with ORB SLAM. The object detector will use I2C to communicate information to the MCU and the Payload Identification will use SPI to communicate information to the MCU.

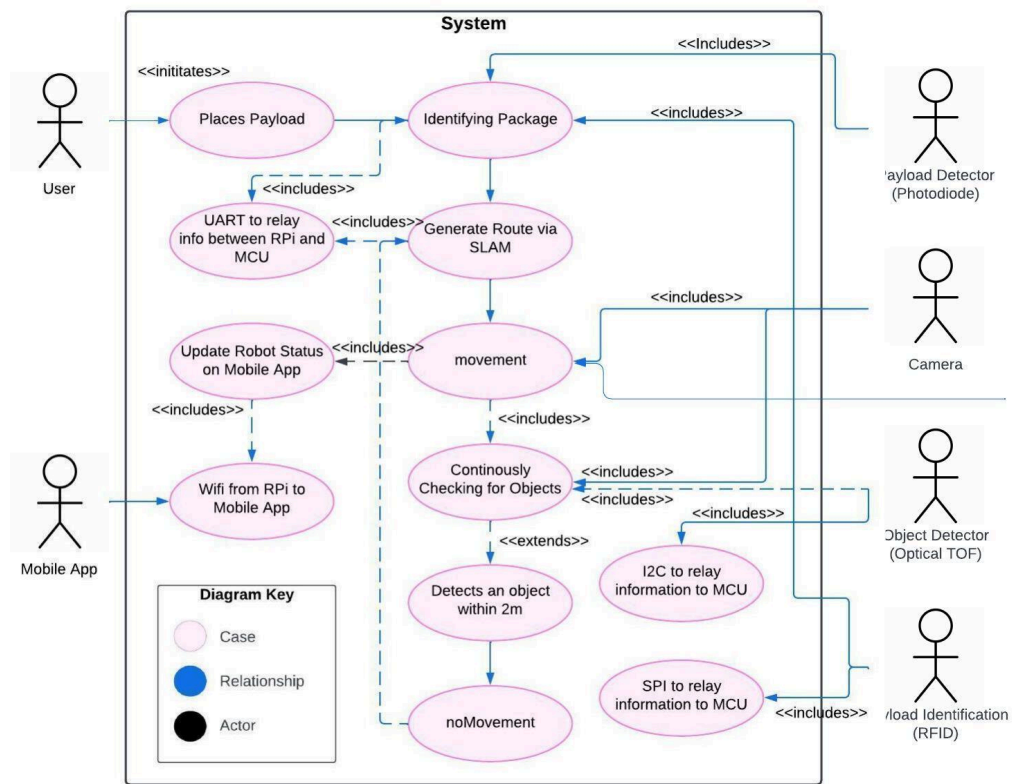


Figure 31: Use Case Diagram

7.3 System State Diagram

This system state diagram shown in Figure 32 shows the flow of each state and the case statements used to determine changing states. One specification of the robot's design is that it must be able to detect an obstacle and avoid running into it. This will be achieved with a distance sensor that reports how far away an object is in front of it. When the

sensor reports an object within 2 meters then the robot will stop movement and find a new route to its destination.

The states and key variables are outlined and described below.

idle: The robot starts in the idle state and is not moving during this state.

readMode: The state when the RFID module will be switched to its reading mode.

normalMovement: This state is when the robot is following the path created by SLAM.

payloadDetected: This will equal 1 when there is a payload detected and 0 when there is no payload.

tagFound: This will equal 1 when the RFID tag successfully reads an RFID tag and 0 when there is no tag.

routingDone: This will equal 1 when SLAM is finished rerouting or 0 while it is still rerouting.

finishedPath: This will equal 1 when the robot has finished following the SLAM path and 0 when it is still completing the path.

movement: The robot will be in the state of normal movement. This means it is following the route from SLAM.

noMovement: The robot is stationary. It is waiting for a new route from SLAM.

obstacleDistance: The approximate distance that an obstacle may be from the distance sensor.

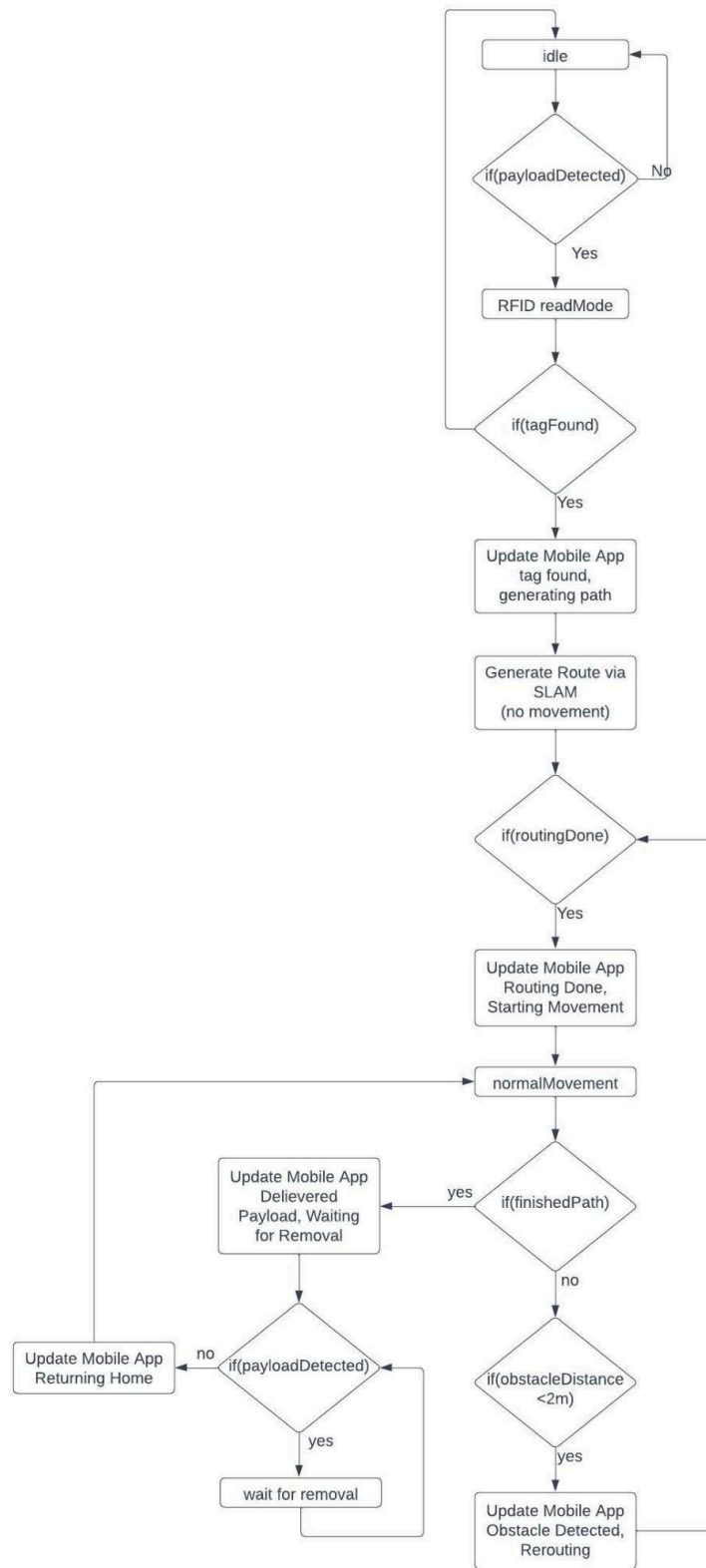


Figure 32: System State Diagram

7.4 Connection & Communication to an Arduino or Raspberry Pi

A network is required for the mobile app to connect to the Raspberry Pi and wirelessly transmit data between the two. There are two types of networks, local and remote and their differences will impact the approach towards developing the Network program for our robot. As previously mentioned in CH3 using WiFi and TCP over UDP, this device is classified under the Internet of Things (IoT) category because we are connecting a phone to the robot that is sending data over a wireless local network (LAN). This requires us to develop a reliable implementation of the Network interface. Moreover, the lack of a traditional RTOS in Pi will impact the latency of packets being transported over a network as RTOS applications do not favor time-delays between communicating devices, the speed which each hardware component interacts with each other, and packet corruption the information sent to the mobile app are some of the areas we are addressing in this projects IoT portion. This provides greater scope as to why we decided to use the national standard for network development TCP/IP, also known as, Internet Stack for Pi as that is what is programmed into Pi's OS as part of the Linux distribution system. This makes our choice of using TCP over UDP more practical as it is compatible with the Pi's OS, and reduces the development issues in the future, while focusing more on implementing the reliable design standard from the 1970's.

Moving on, this network application will have to have each hardware component interact with each other wirelessly through our local area network, ensuring that the information sent back to the user is accurate via the Mobile App, and then communicate reliably to Pi and other components of our robot. This is important because improper communication can lead to degradation of hardware performance (implied in CH3's WiFi Table), or the potential of physically frying the hardware components such as the Pi on the robot. So we had to find a suitable alternative while ensuring that we reduced the amount of software applications the team had to learn. In addition, improve our reliable tests for the code, and ensure the packets were being properly distributed between the Hardware/Software layers. We decided to compare two application options for our project: the AWS Gateway from Amazon and IoT.

Briefly explained, AWS IoT integrates and takes the data being sent from the Hardware components of Pi back to the IoT service, and processes the data to make updates using a MQTT protocol. Considering this and the other tools this service provided to ensure the Network was properly designed it became clearer that this service should be used for this project. One example being FreeRTOS to help replace Pi's lack of RTOS support, improving the speed between hardware and software over the network.

As mentioned, Raspberry Pi's use the later Internet Service Layer: Internet Stack Protocol and not OSI (theoretical framework). The primary difference being OSI has 7-layers and Internet-Stack has only 5-layers. The Internet Stack is primarily made and focused on TCP oriented applications for most use-cases. TCP previously mentioned is considered a more reliable service, however after discussing with a Networks professor there are discussions about replacing TCP with UDP. The primary reason is taking UDP's pros and downsize the cons. The pros for UDP are: being naturally faster, efficient, and better for

overall system-level performance in terms of hardware or other RTOS oriented software/hardware. Not only this but it is great in situations where reliability is not very important, and packet loss is generally considered acceptable. This contrasts our position in incorporating TCP over UDP because UDP is claimed to be better than TCP for prior reasons. In addition, it is better for other components like Video-Streaming (cameras), gaming, and VoIP. For example, let's take the gaming application into consideration, increased packet-loss and decreased reliability is deemed acceptable in this context. Why? It is because UDP can increase the overall throughput and reduce latency (useful for gaming) because of the decreased reliability and increased packet loss. This makes sense because the less packets being recovered the greater speed it will retain, and lead to greater throughput produced in the system. The trade-off becomes speed vs. accuracy. The general rule of thumb is the more packets are lost, the greater inaccuracies develop, and in other contexts such as mission critical operations this is unacceptable.

Hence why most protocols established the standard of integrating TCP because of its reliability. Now, we are not saying UDP is never used. The prior reasons demonstrate UDP has a special usage deemed important. UDP in many ways can be considered better than TCP, but their primary issue is being unreliable, from losing packets when trying to use this protocol. Yet, with the development of HTTP3 to primarily incorporate UDP, the new claim is now having an improved connection with less packet-loss which gives relative reliability like TCP, but with the speed improvement maintained as for UDP's notable strength, along with other positives about UDP over TCP.

Until then the main reliable standard remains to be TCP with its 3-Way Handshake which ensures reduced packet-loss between sender and receiver. While Networkers play the role of recovering lost packets and sending them back to the network properly, TCP has a greater chance of reducing the number of packets to be recovered, and improving the overall accuracy of the information being sent, while reducing the produced throughput for a system. Additionally, TCP has advanced techniques operating at the OS level, having protocols involving Sliding Window with SACK TCP, which is commonly used in everyday applications because it increases the throughput in TCP-based applications (hardware and software). Additionally, we have other things like Firewalls, low-level implementation of Network Security with TLS/SSL, and other methods to enhance the reliability/scalability of a Network.

The interaction of this, especially in Firewalls, are how we ensure that malicious attackers cannot penetrate through a network, and take over that device through techniques like DDoS or accessing a remote connection to a network. Preventing (unless granted authorized permission) special circumstances where the user conducts a remote connection to a network and enters a scenario of potentially legal/company policy violations. In short, accessing a network remotely in the literal definition has the chance of getting someone fired, or in the worst case be sent to jail/prison if done wrong (the process or unauthorized permission). This demonstrates the legality and company related issues working on a Network interface has in store, and is because remote access has the potential of taking over someone else's computer at the hardware level, which is why it totes the line of ethics/legal repercussions.

Moving on, this project requires establishing which network implementation standard will be used when developing the network code. One option is implementing MQTT along with the TCP protocol to interact between client and server. The reason for this is because TCP is considered a reliable standard for packet-transmission. Although it was documented UDP is better for real-time oriented tasks due to acceptance of packet loss, one downfall of UDP is there is a higher chance of sending unreliable data to the user with the App-Robot connected interface. The Network program is supposed to connect both entities with each other, along with other devices as well to send a request for the Payload to move from one location to the next.

Additionally, as highlighted in our stretch goals the Payload system will eventually integrate a map functionality in it, and this requires greater accuracy of packets while not affording much packet-loss. So the fundamental concepts that we learned although being essential to understand how packets move through a network, how the Sliding-Window technique works, what the SACK TCP does, and RDT3.0 FSM operates as well; all of this is already implemented by Linux at the lowest level, and reinventing the wheel is only necessary if the available features/standards established in the industry does not work or Amazons services does to solve particular issues the application has. Moreover, the availability of FreeRTOS provided by this software package has made using UDP rendered useless as that will increase the speed of which communications between Pi and Hardware devices interact. One great example (pre-RTOS) was using IoT to send Pi data over to a Ultrasonic Sensor that only took a delay of 5 seconds each over MQTT as defined by the sleep timer implemented. This can be pushed to operate faster over a TCP connected server which is operating on a Client-to-Server protocol between the two.

This makes the core idea in developing a process involving packets being sent between two communication lines between the Sender and Receiver unnecessary (not unimportant); some of the methods are Go-Back-N repeat method, Selective Repeat (SR), and SACK TCP.

Furthermore, different routing algorithms are pre-implemented into these devices like within Linux in the Network Stack of the Internet Stack Layered protocol (5-Layer process). We can implement these algorithms to meet our requirements, and have the option of including a self-implemented routing algorithm not previously made or integrated either from Linux or in AWS IoT to handle a certain scenario. So, if we take a step-back we see many things are already abstracted for us in the Network Layer.

We believe this is how it should be as we can only access the Application layer when developing solutions, and the remaining 4 layers are outside our control. The only important things Network Engineers think about is improving the recovery process of packet-loss, corruption, and out-of-order sequence while constantly improving how much less we can lose packets by ensuring more are recovered quickly to handle special cases. This is why instead of custom-creating security protocols for TLS/SSL we instead import the Python library as it is already coded in the backend to be highly-secure. As mentioned in CH3's table, WiFi uses TLS/SSL for maintaining security over the network. It is important to mention UDP has its own version of this, but does not perform similar actions.

We can also import sockets to perform socket programming in Python, and we decide whether we want to use TCP or UDP to meet the standard we are going for. In this case, using MQTT requires the use of TCP, the sockets for developing websockets will be imported and reference TCP libraries to effectively create a software-to-hardware paradigm .

For these reasons, these features will be implemented because they are necessary, and if doing so will introduce situations where not developing other network components at the lower-level inside the Application Layer. Thus, the SR method will not be used for managing the communication lines, Go-Back-N repeat method, or even the SACK TCP unless necessary or need to implement additional features to improve overall performance over a network. A great Network interface is simple but reliable. Instead, using fewer libraries/different software applications that come with the tools chosen. This again includes using FreeRTOS to replace Pi's RTS with a Real Time Operating System to help improve low-level performance between hardware modules overall. The standard implemented MQTT with TCP while letting things like the 3-way Handshake, SACK protocols to manage packets sent between a Client and Servers, and any dedicated TCP-based routing algorithm are already developed at levels beyond my knowledge be taken care of at the OS level and with the IoT software application.

As mentioned previously the two kinds of network connections are local and remote. Briefly explained, local is the connection you have when connecting with WiFi in an establishment like the following: Homes, Schools, Companies, Restaurants, and more.

However, remote internet is a connection over the World-Wide-Web (WWW) and bypasses many proxies to access a remote server somewhere, and in turn gets past through greater security measures to connect with such a router. Typically this form of connection requires Port 3389 (for Windows machines) to bypass all security requirements, and directly allow someone to be directly controlling the machine the router is connected to. In essence, the connected entity takes over the machine physically, and as it can be imagined is considered a security risk if the proper protocols/permissions were not granted. This enters into the legal domain or disciplinary actions of whomever you gained access to.

In short, accessing this is called RDP (Remote Desktop Protocol) and it does by proxy what the name implies. However, we would rather not engage in legality issues when designing our robot, so we are avoiding remote connection and instead opting for wireless connection.

There are some potential harms that a Computer Network can cause to the physical hardware very briefly. The Raspberry Pi 5 has the ability to act like a router which is considered a Network hardware device. Now it is important to mention in CH3 the WiFi section that it consumes more power than Bluetooth. This means designing poor network code can increase the amount of power used by the Pi. This is where we decided the best approach is the divide and conquer strategy really focused on managing the traffic distributed from App to the robot's hardware components without increasing greater power consumption. This is why Network traffic increases loads of traffic traveling into

one node, and the overload can cause hardware failures to happen, which leads to our conclusion that poorly designed Network code causes the hardware to degrade in overall performance.

7.5 ORB SLAM Algorithm

In our project, ORB-SLAM is going to play the central role in localization and path planning of the robot so that it can autonomously navigate inside the building accurately. ORB-SLAM has a feature-based Visual SLAM algorithm that detects and tracks visual landmarks in the environment using Oriented FAST and Rotated BRIEF (ORB) feature descriptors.

The ORB features are computationally inexpensive and still capable with the Raspberry Pi Camera V2 which takes a monocular video stream that continuously estimates the robot's position and orientation using ORB-SLAM. The camera will be placed strategically to get the best field of view and consistently detect detectable features including walls, floor tiles, and shelving in the warehouse or lab. To improve the accuracy and stability of the process, the system will use additional sensor data of a Time of Flight (ToF) Sensor and Inertial Measurement Unit (IMU). The ToF sensor will help to detect obstacles at short range. This additional information would help provide additional depth measurements to verify and refine the position estimates provided by the visual ORB-SLAM system. The IMU, generally a combination of gyroscope and accelerometer, will provide the inertial data that helps preserve position in low-feature or occluded areas to limit drift during times when visual data is lacking. The system utilizes an enhanced localization framework that combines data from the camera, ToF, and IMU which is more robust and is capable of working in a variety of indoor conditions.

To integrate ORB-SLAM with the Raspberry Pi, we will use Robot Operating System 2 (ROS2). This will help to synchronize the streams of sensor data and the SLAM algorithm with each other. ROS2 nodes will process the camera feed and synchronize it with IMU data while publishing the robot's pose to other subsystems, such as the route planner and mobile app interface. When the robot detects a payload using the built-in RFID, it will initiate ORB-SLAM-based path planning to find the best way to the delivery site. Throughout this time, ORB-SLAM will consistently revise the robot's position. This guarantees that it will be efficient to move on the basis of this location and allow the system to re-plan paths in case of obstacles along the way. ORB-SLAM serves as the backbone of autonomous navigation, facilitating real-time updates to the user-facing mobile app, thus enhancing the overall efficacy, robustness and smartness of the delivery robot systems.

7.5.1 Yolov5 and OpenCV

Alongside ORB-SLAM, we will be using softwares that requires us to process the output that is provided by the Raspberry Pi V2 camera and properly interpret them. OpenCV will be the main image processor that we will use to handle the frames recorded from the video feed and interpret them for the robot's navigation. Yolov5 is an object detection

model that is built on pytorch that allows an input of images to act as a way to navigate around objects or people. These processes use visual data alongside deep learning to be able to identify, locate, and classify objects in the field of view of a camera. The system breaks data into a grid and predicts bounding boxes and a class probability for each grid cell at the same time, providing very fast and accurate detections. It is mostly used as an assistant to the robot's perception enabling it to see dynamic and static obstacles, like people, boxes, or other robots, right from the video feed. From this visual information, the robot can deduce how to navigate and avoid collision with its environment. Yolov5 will require using pytorch and a Anaconda as a way of running the installation then getting acquainted with learning models with the capabilities of TensorFlow [A.98].

As opposed to some other object detection tools, Yolov5 takes in the input of camera images without the need of using LiDAR sensors that are a more expensive alternative. This will be an ideal complement to ORB-SLAM once the map has been made and processed. It is necessary to have a visual aid when unforeseen obstacles appear in the robots path while navigating. It also has different packages that can be lightweight enough for the Raspberry Pi 5 and holds up in terms of detection accuracy. In terms of ROS compatibility, Yolov5 is capable of being run with ROS2-jazzy so we can work with the sensor data that is gathered from the TOF and IMU for accurate deductions. In other tools such as the dynamic window approach, Yolov5 already has the capabilities to use various methods based on its learning model.

7.6 Mobile App

7.6.1 Application Overview

For our mobile application, It will include the current status of when the payload would be detected via RFID, the current status of the robot's destination on the map, error messages regarding the status of the robot and the capabilities to abort the navigation process. For the main authentication of the application, we wanted the user to have to create an account to simulate that each user would have their separate record of each trip taken and the payload that they traveled with. For this we concluded we would have a simple database for the user and also have the capabilities to sign in with their google account. With this we can have some mode of security with the app and make the experience different for each user.

The main function would solely rely on the communication between the Raspberry Pi and ESP32-WROOM-32E to give real-time updates to the user during the navigation process. After the user successfully signs in, we would then prompt the user to begin their process with the payload that would contain the RFID tag within it. When the user places the payload on the platform of the robot, the RFID scanner would read the data provided from the tag in the payload. This would typically mean if the RFID senses any compatible tag near the platform then the system would notify the user on the app that the navigation will be ready to begin.

The status tracker itself will allow the user to visually monitor the robot's progress beginning from when the payload has been identified, then the user will begin the process with the map that has been presented. When the robot is 1m from the destination, the user will receive a final notification indicating that the payload has been successfully delivered and this would mean the user can begin another process or send the robot back to its original point. Our aim was to make the application as simple to navigate and use and then further plan for future features

For implementing the application, we are using Flutter, to create the app due to it being open source and a very popular mode for mobile development. Flutter allows for rapid development for mobile applications which can run on both Android and iOS from a single codebase, significantly reducing development time and resource overhead. Flutter's extensive widget library and flexible UI design tools will make it possible to create a clean and user-friendly interface, including login/register screens, status update cards, and notification pop-ups for delivery events. The authentication system can be implemented using Firebase Authentication or a lightweight backend database to store user information, depending on the final system architecture.

Furthermore, the other reason why we needed a language/software application to run on Android/iOS is because our team has different OS's for our phones ie Androids and IOS. That hurdle will prevent us from being able to test the application, and newest improvements on all of our ends. So, the end result was us pondering about what language we can use to co-develop an App that supports these different environments, while not costing me money to do so since Apple charges for apps being on their app-store, and has everything we need packaged in one (all-in-one solution).

First, adding more languages defeats the all-in-one solution mentality. Reducing the amount of languages, and software environments to master would benefit us in terms of increased productivity with our work. Decreasing complexity enough where there will be less bottlenecks than progress in our application. For example, initially I thought of using Java and Swift to co-develop the same app in different environments. This would have been a waste of time because we would start from scratch twice, and there is no cross-platform support on Android or iOS. So I asked one of my fellow teammates, and they recommended Flutter. After further discussion, and research, DART (Flutter) became the ideal choice for app development when creating our app. Then the research showed Flutter having a large list of libraries addressing different parts of our project.

For example, it includes a packaged library which supports the Lidar system, Matrix's, HashMaps, SLAM support, Network WebSocket support, and more. From all of this we decided that Flutter will be mainly used for the mobile app. However, due to it primarily being a Front-End application specializing in mobile app development, it naturally did not interact or establish communications between the Pi and our App.

Flutter as a whole is known as a reliable language to establish connections over a network for the Mobile app. However, it lacks the stronger capabilities of managing packets being sent between servers on the backend. Using Python as the second language for

implementing the Network interface was better at ensuring the application itself knows when a network protocol will happen on the client-side of the service. We can think of the interaction as the standard client-server interaction over a network. This is one form of a technique used to manage requests coming in and out, moreover, We will be implementing this style of connection to communicate between two hosts using one server, and a middleman to bridge the gap so that the application can work. The middle-man is Python with all its extensive libraries enabling the process to execute faster, but again this is a topic for the Python section to be explained. Furthermore, the idea of using Flutter is that it is building a mobile app on the front-end to integrate with other components of the hardware that we need interaction with. We need to register this information over the internet, send it back to the mobile application, and then process it such that the user get's accurate feedback where their Payload system is currently located with accurate updates. So, we make sure the process between Front-End and Back-End is maintained, ensuring the data stored inside the packets is being sent.

7.6.2 User Interface

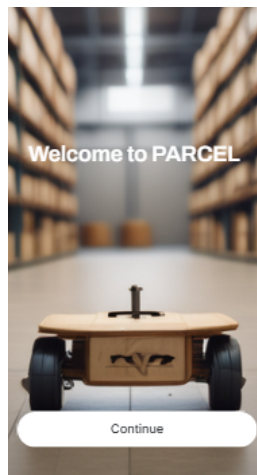


Figure 33: Launch Screen

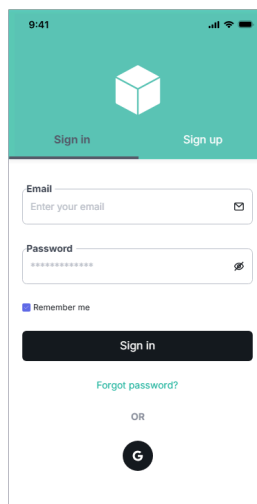


Figure 34: Sign In Screen

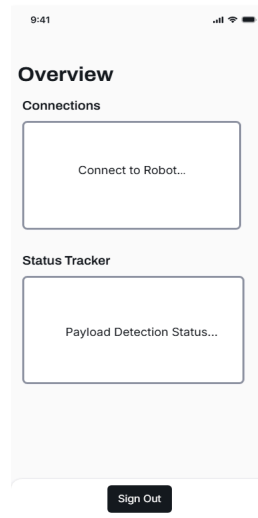


Figure 35: Dashboard Screen

Chapter 8: System Fabrication and Prototyping Construction

The robot consists of a total of four PCBs, each one serving a role within the overall architecture. These PCBs were designed to ensure safety, modularity, and ease of integration.

8.1 Mainboard

The mainboard acts as the central hub for the entire system. It houses the ESP32 MCU and includes connectors for interfacing with all the peripheral boards and sensors. This board also manages the power distribution across the robot, with a 2 position screw terminal to connect to the 12V battery.

The power distribution section consists of 3 different regulators, as well as their supporting components:

- A 3.3V switching voltage regulator for the majority of the components, including the MCU.
- A 5V switching voltage regulator used solely for the rotary encoder for the DC motors.
- A high-power 5V switching voltage regulator dedicated solely to supplying the appropriate power to the Raspberry Pi.

The board makes use of various power planes to ensure a higher efficiency when delivering power to and from the regulators, as well as providing appropriate ground return for all components. It also consists of 4 layers, with one of them entirely being a ground plane, further enhancing the availability of ground for the components.

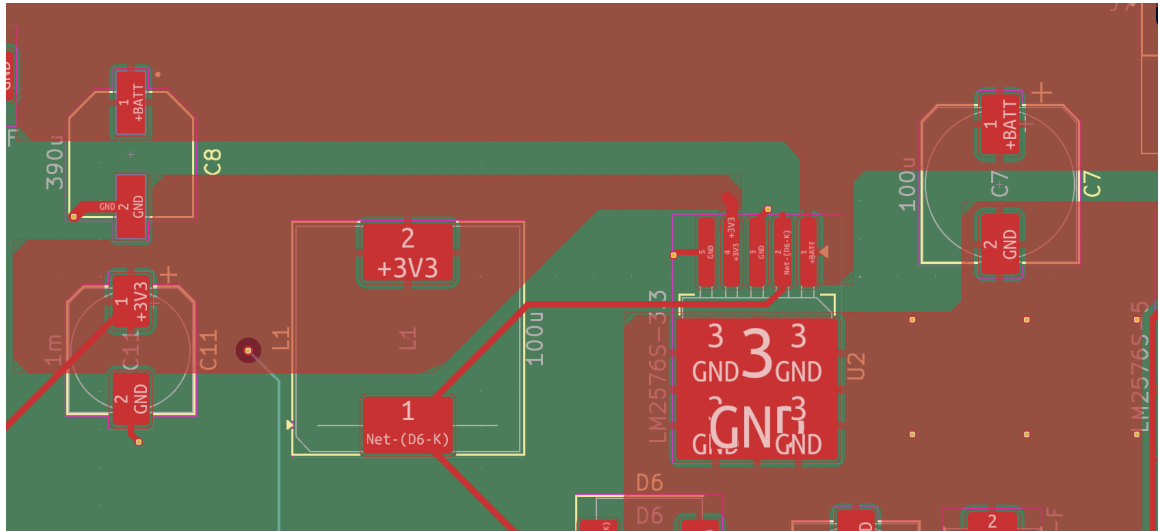


Figure 37: 3.3V Voltage Regulator Layout

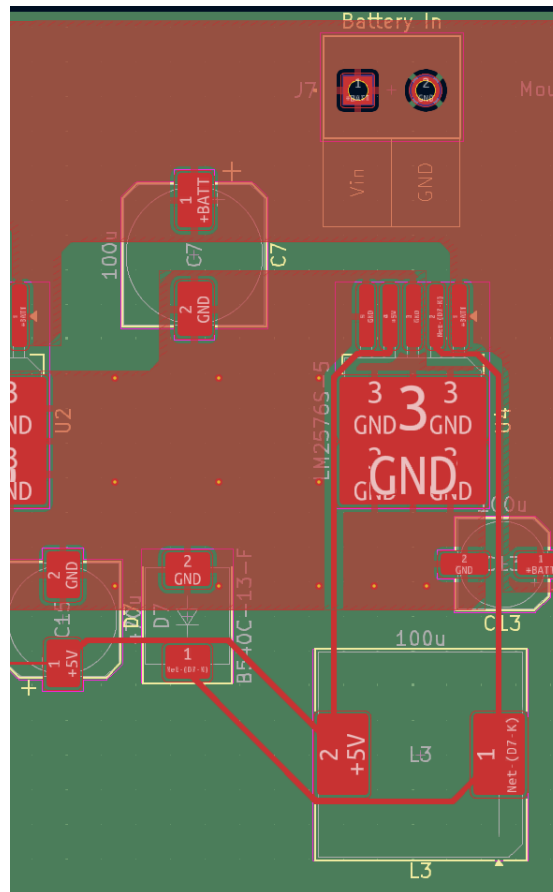


Figure 38: 5V Voltage Regulator Layout

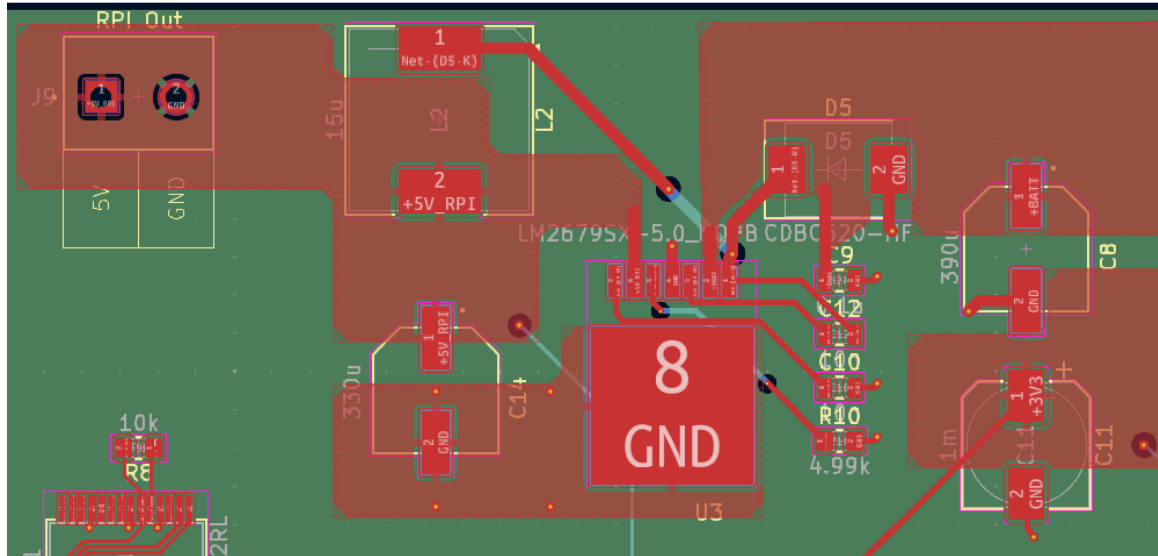


Figure 39: 5V High Power Voltage Regulator Layout

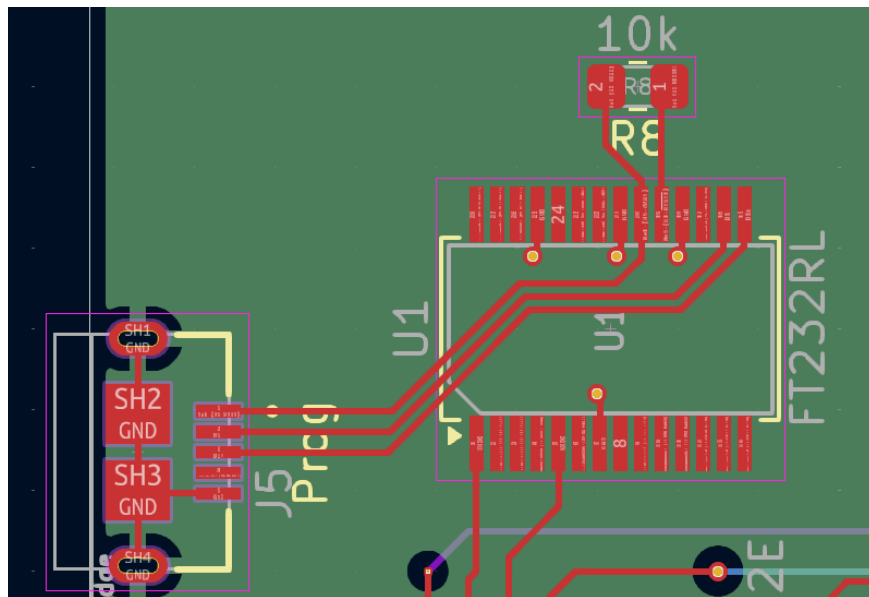


Figure 40: USB-to-UART Connection to Micro-USB Programming Port

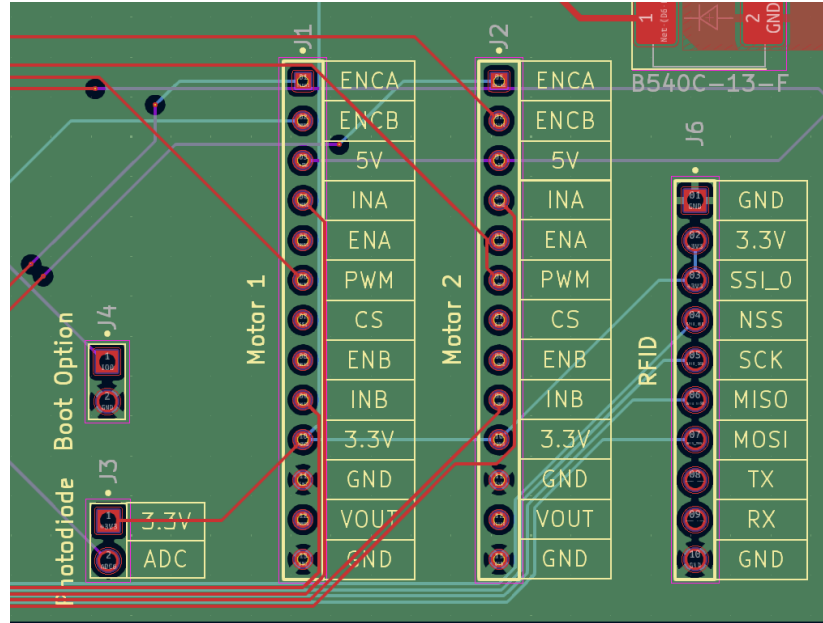


Figure 41: Peripheral and Sensor Connection Headers

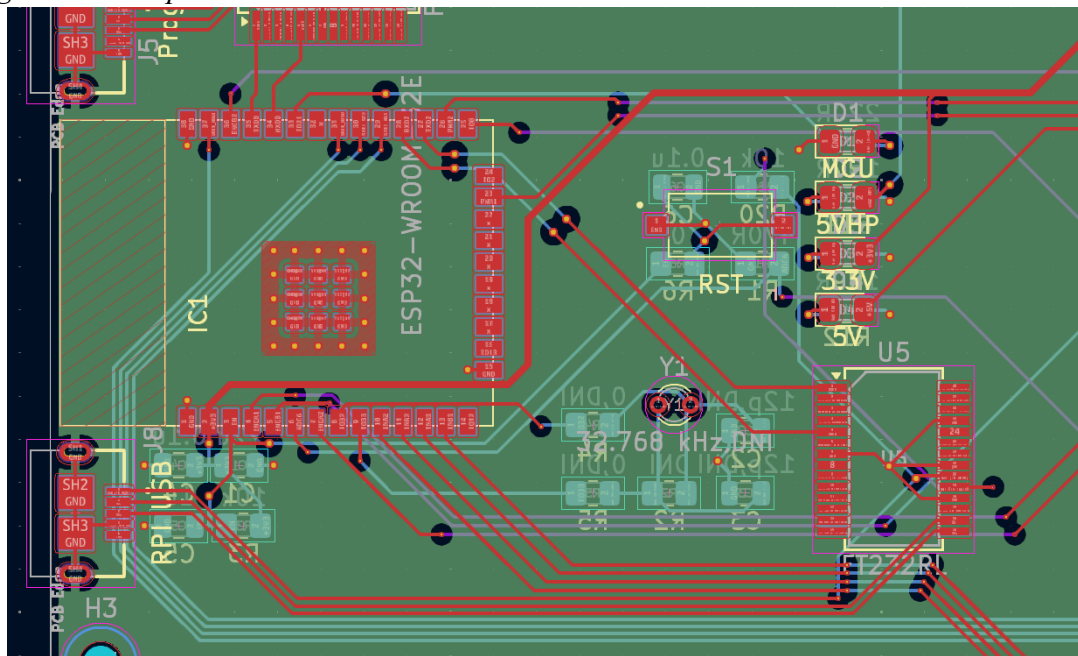


Figure 42: Microcontroller and Status LEDs Layout

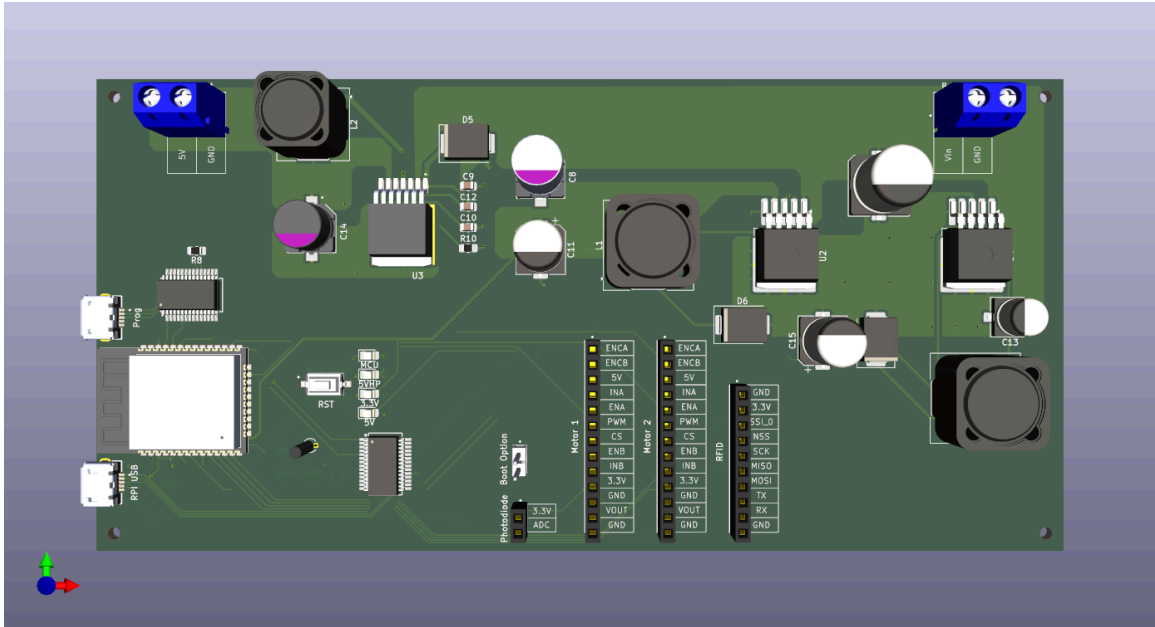


Figure 43: 3D Render of Mainboard - Front

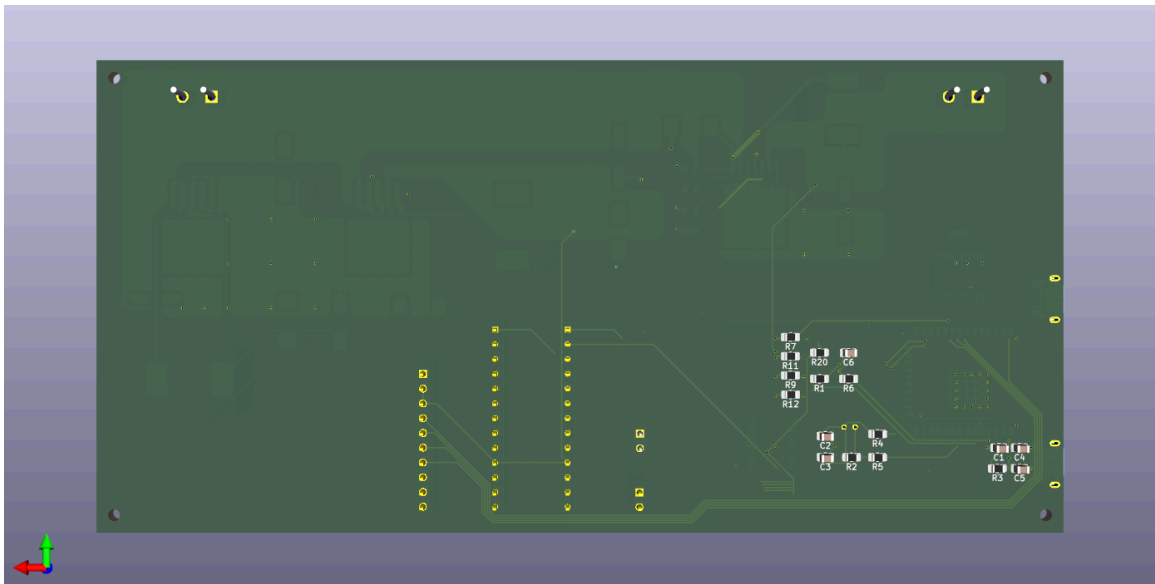


Figure 44: 3D Render of Mainboard - Back

8.2 Motor Controller

A single motor controller board was designed to drive both the motors. The PCB consists of the same circuit duplicated, with only the connections differing. The motor controller PCB interfaces with the mainboard via two dedicated 13 position header connectors, routing the appropriate signals to and from the PCB. Above the VN5019 motor driver IC are the 6 position headers that connect to each of the motors. This board also has status LEDs, two for each motor driver. One or the other illuminates based on the current flow and thus based on the motor direction. Similar to the mainboard, power planes are

used here to ensure high currents can be handled properly. Additionally, mounting holes are placed in the corners to allow the PCB to be mounted directly to the chassis where needed.

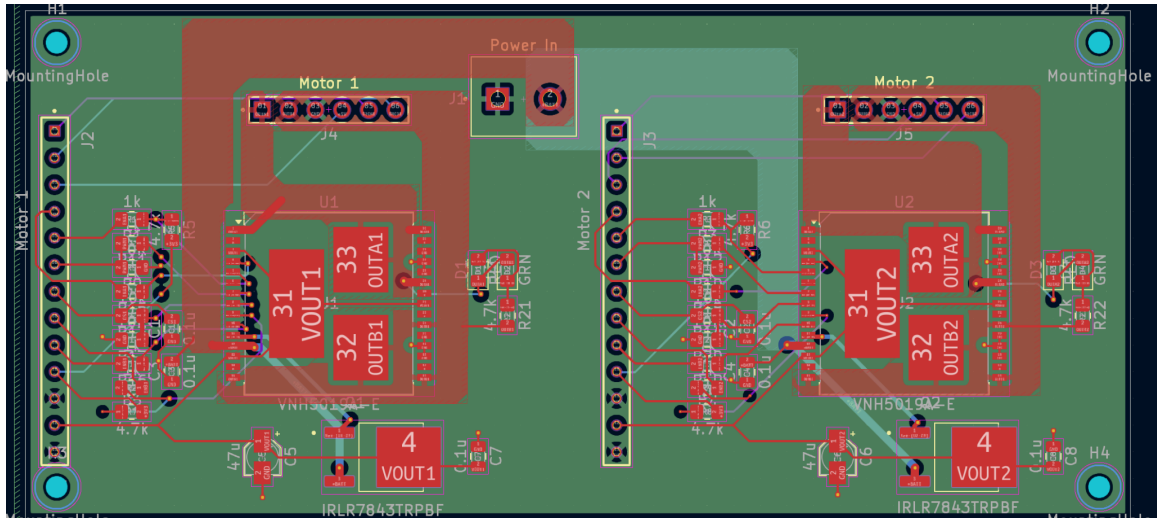


Figure 45: Motor Controller PCB Layout

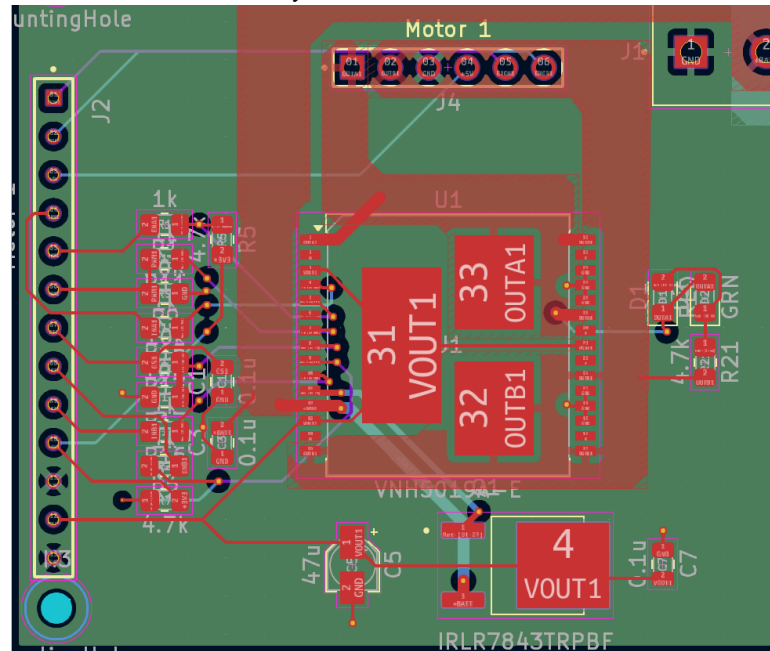


Figure 46: Motor Controller Components Layout - Motor 1

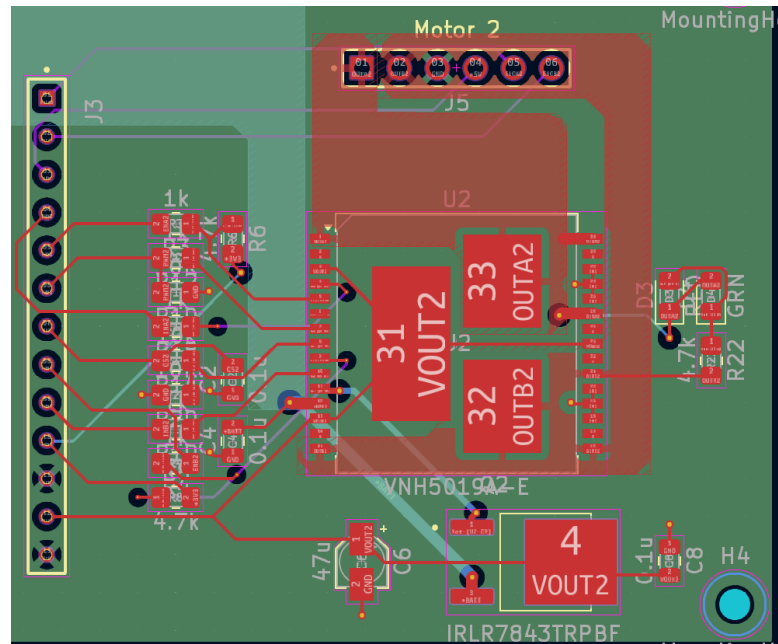


Figure 47: Motor Controller Components Layout - Motor 2

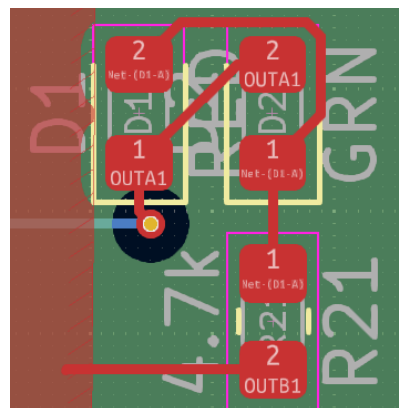


Figure 48: Dual LED Direction Indicator

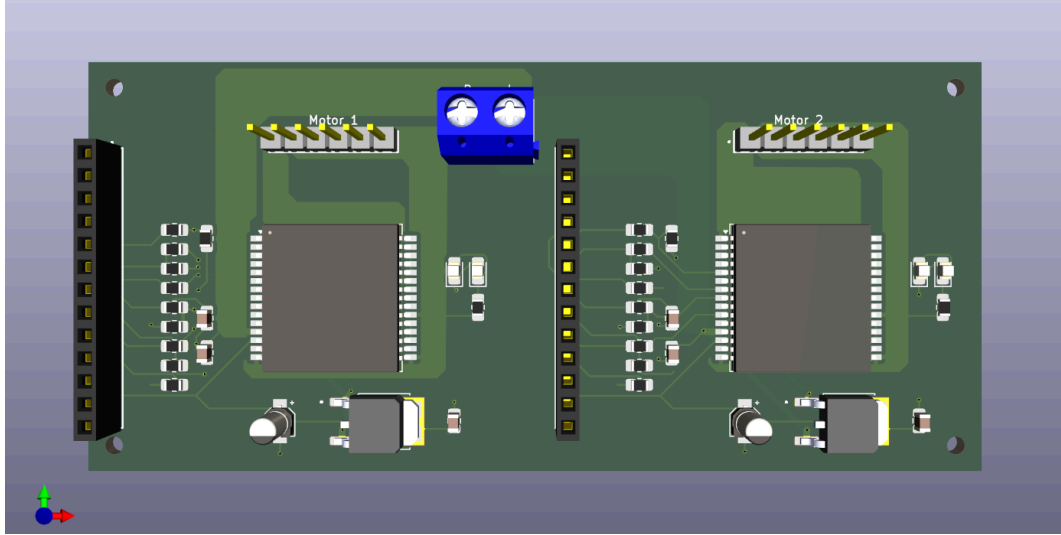


Figure 49: 3D Render of Motor Controller Board - Front

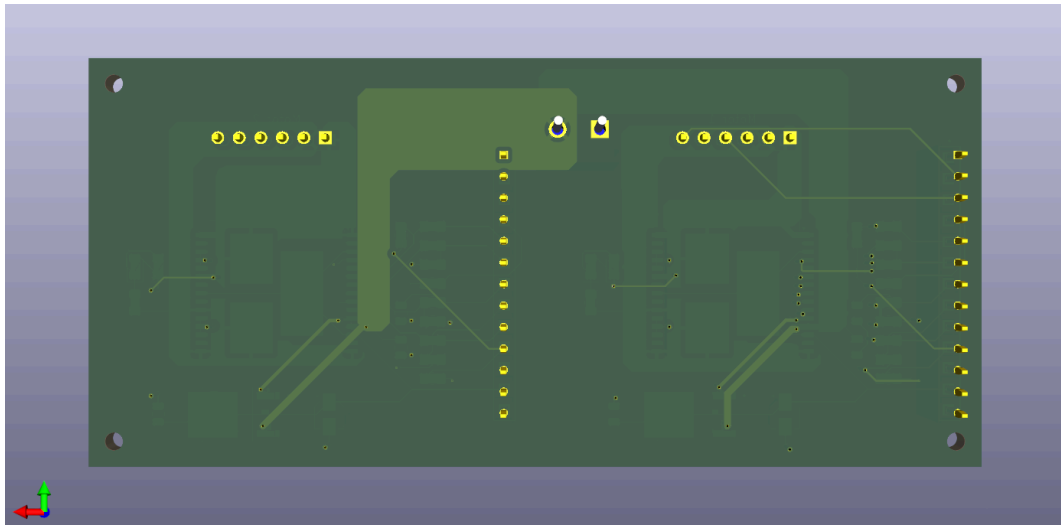


Figure 50: 3D Render of Motor Controller Board - Back

8.3 Raspberry Pi 5 HAT

This custom HAT interfaces with the Raspberry Pi 5, mounting directly onto the SBC and breaking out the necessary pins, integrating all of the SLAM-related sensors. Due to the limited availability of I2C buses on the Raspberry Pi and shared default addresses on the ToF sensors, an I2C multiplexer was incorporated to allow independent control and communication with each sensor via a single I2C bus from the Raspberry Pi.

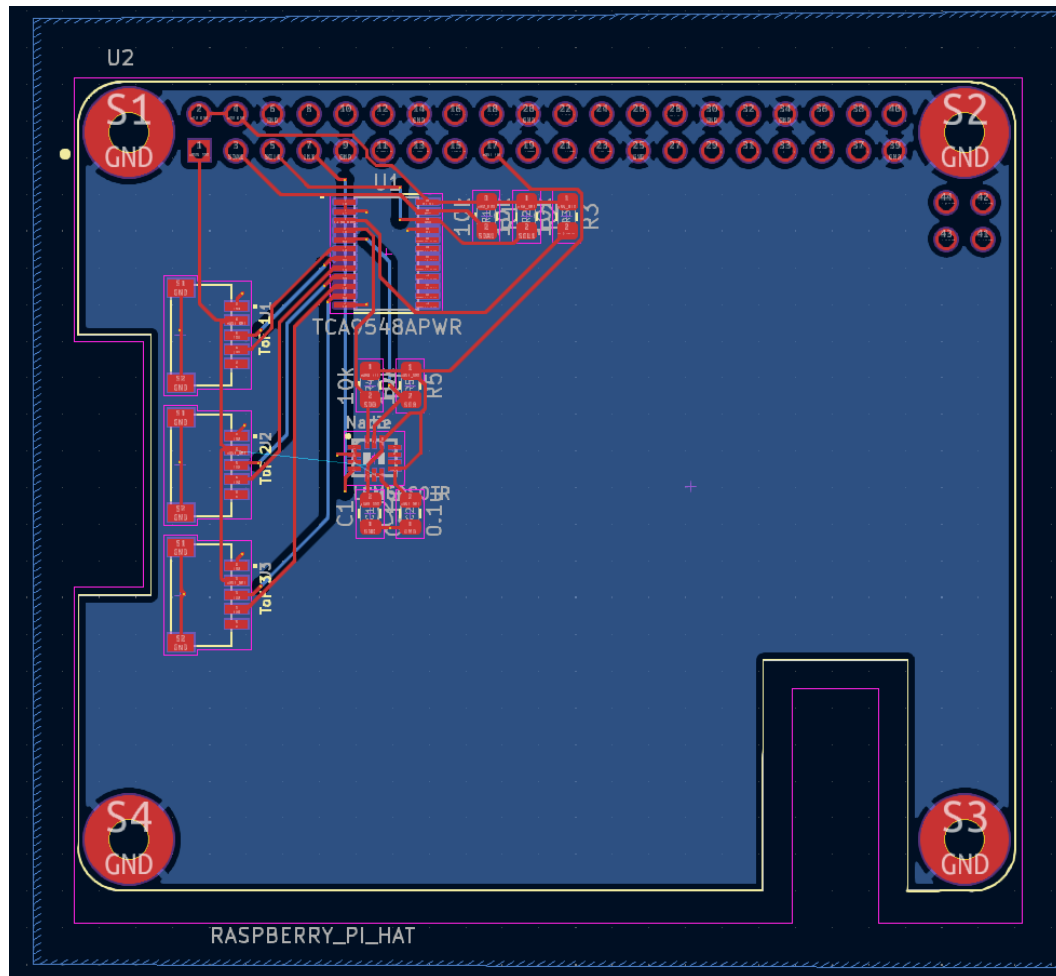


Figure 51: Raspberry Pi HAT PCB Layout

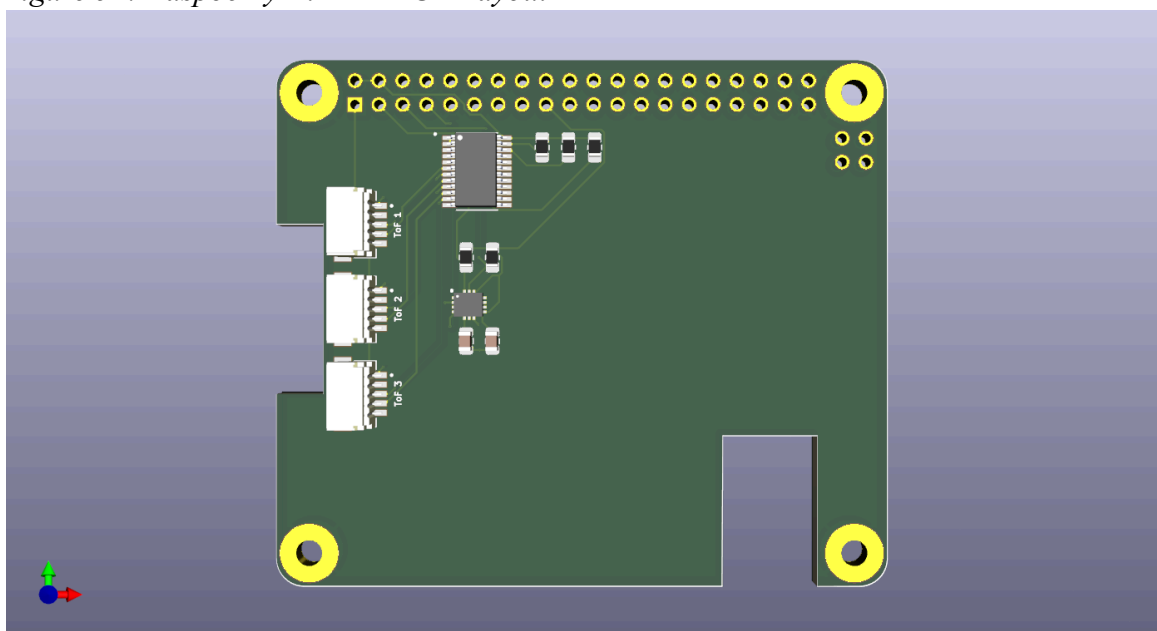


Figure 52: 3D Render of Raspberry Pi HAT - Front

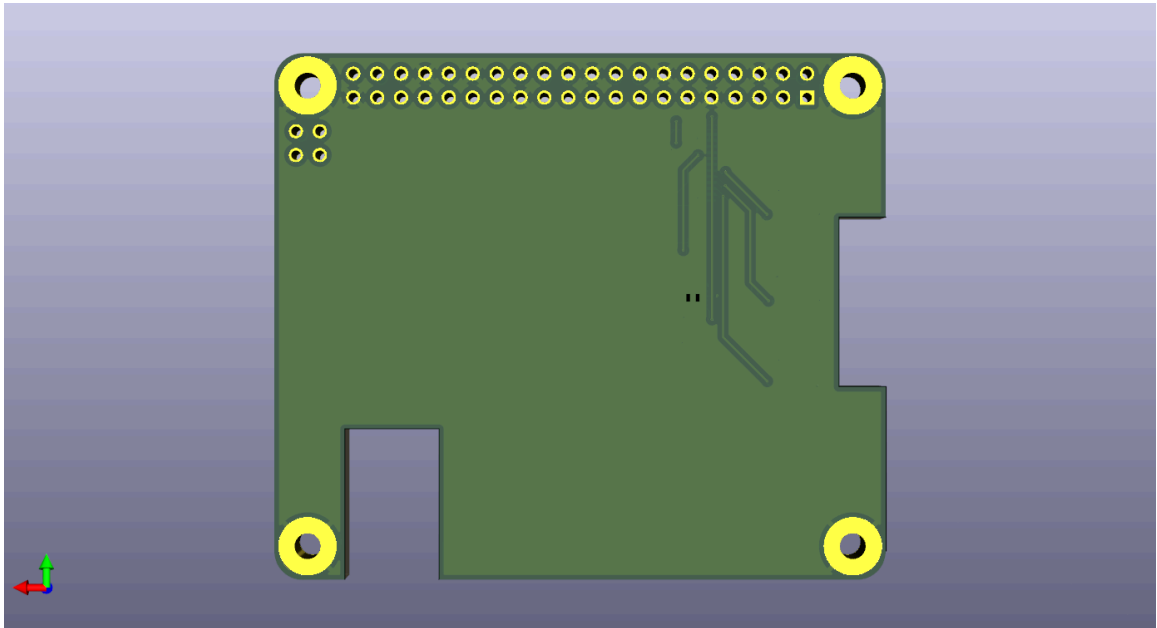


Figure 53: 3D Render of Raspberry Pi HAT - Back

8.4 RFID Reader Breakout Board

The RFID module used in the design has a nonstandard pin pitch of 50 mils, which is incompatible with the standard 100 mil headers and makes functionality difficult. To address this, a custom breakout board was created. This PCB adapts the 50 mil pitch to a 100 mil pitch, allowing the engineers to connect the RFID module to the mainboard via a standard 100 mil pitch 10 position header. This also gives the added benefit of allowing the module to be repositioned as needed to scan and identify the payload.

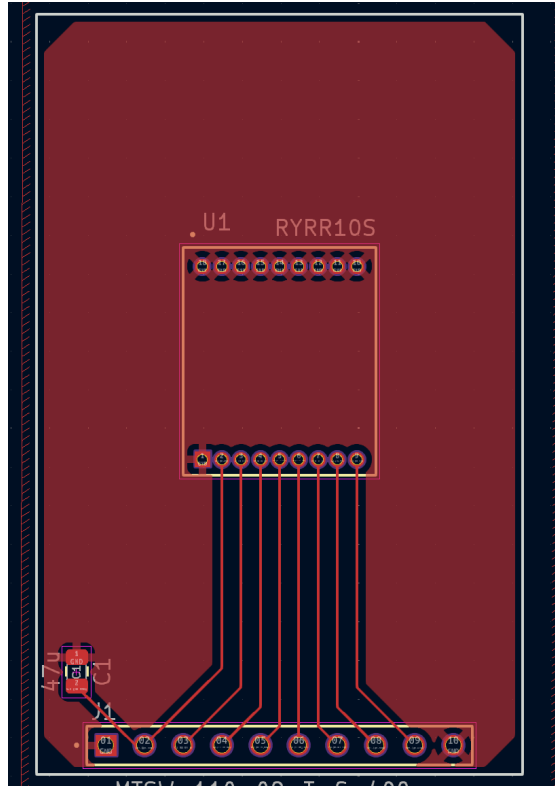


Figure 54: RFID Reader Breakout PCB Layout

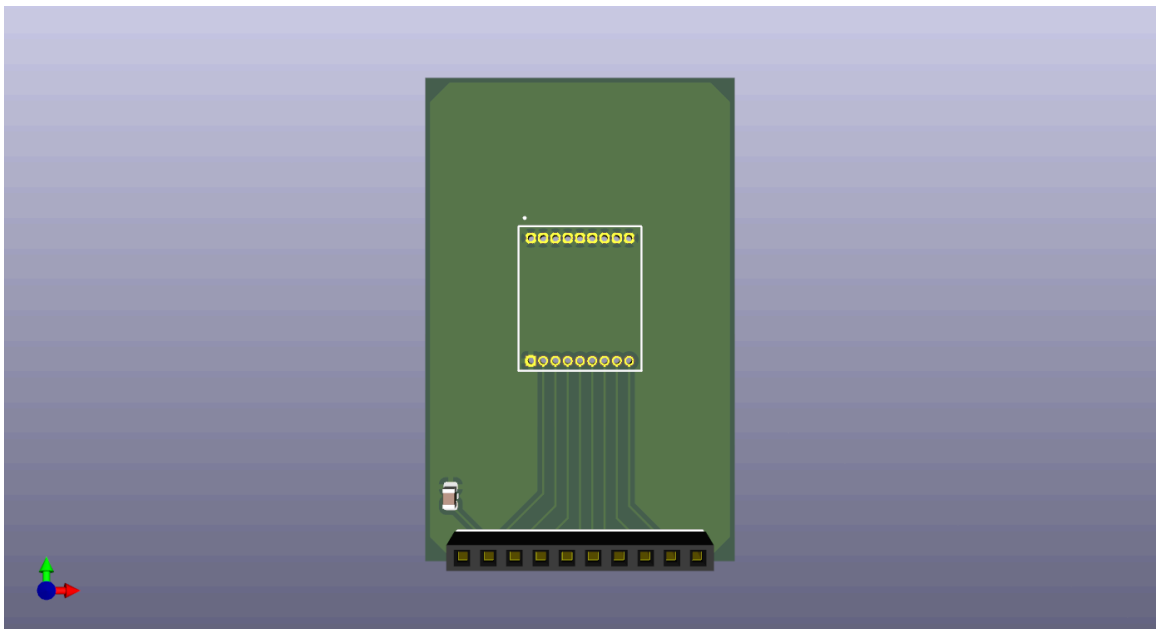


Figure 55: 3D Render of RFID Reader Breakout Board - Front

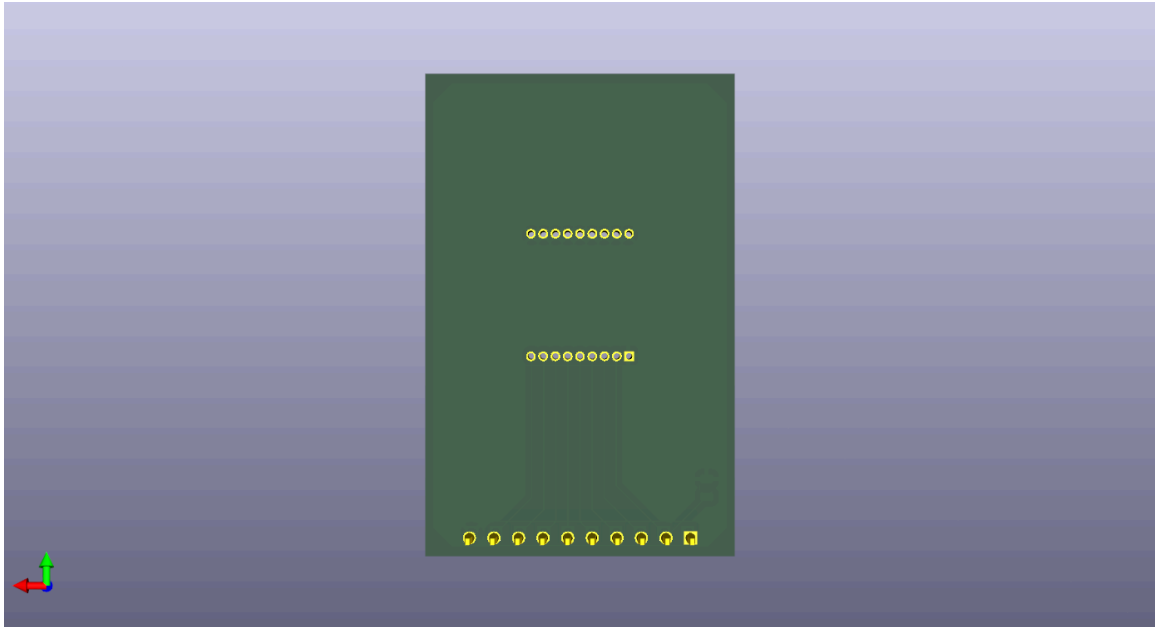


Figure 56: 3D Render of RFID Reader Breakout Board - Back

Chapter 9: System Testing and Evaluation

9.1 Hardware Testing

9.1.1 ESP32-WROOM-32E Microcontroller

This project uses the ESP32-WROOM-32E development board for prototyping. To ensure that they are working correctly, the boards were first plugged into power through their USB ports into a computer. The RGB LED powering on verifies that the device is receiving power. After this, GPIO pin voltages could be verified using a digital multimeter. The reset button functionality could be verified by pushing it. More information on testing if the coding environment and uploading code works correctly can be found later on however these steps ensure that the hardware component itself works properly.

9.1.2 Raspberry Pi 5

Setting up and testing the Raspberry Pi 5 starts with setting up the micro-SD card to have an OS on it. We chose to use Ubuntu for the OS and installed it with the Raspberry Pi Imager from a laptop. Then we put it in the Raspberry Pi 5. To test to make sure everything on the Raspberry Pi 5 we used a monitor, mouse, keyboard, power supply, and HDMI cable that are known to work. Then we plugged everything in and waited for the Raspberry Pi desktop to show up. If any of these did not work it would indicate there was a problem with the Raspberry Pi or one of the ports. Next, we checked the wifi connection with a known wifi and performed a “sudo apt update” and “sudo apt upgrade”

which must fetch the latest information about available packages from the internet. This was successful so it confirms that internet connection works correctly.

9.1.3 Lidar Sensor

This project utilizes a 360 2D laser sensor for 12 meter range distance measurement as part of SLAM. Initial testing was conducted to ensure reliable operation and communication with the Raspberry Pi via UART. After downloading the necessary packages, the sensor's address was successfully detected. The output values were displayed on the terminal and verified by using a physical ruler and moving an object in front of the sensor.

9.1.4 Photodiode

A photodiode is used to detect when a payload is present on top of the robot by detecting the amount of light coming through. The photodiode was first tested on a breadboard and the output voltage was measured with a multimeter under various lighting conditions to ensure that the component was responsive to changes in light intensity.

Once initial functionality was confirmed, the photodiode was connected to one of the analog input pins on the ESP32. A program was written to read and print analog values in real time over serial and the same test as before. By placing a payload above the photodiode, we were able to record the values that would be used to determine whether a payload is present or not.

9.1.5 Raspberry Pi V2 Camera

In regards to testing the camera, this solely depends on the Raspberry Pi's ability to function. In order to test out the camera, we had to configure the CSI-2 interface and begin to configure it so that it displays within the raspberry pi OS. Through configuration we need to have ROS2 Jazzy to be installed correctly with the right version of Ubuntu. We would have to test how the Raspberry Pi can handle these programs and what would have to adjust to operate SLAM via our Raspberry Pi camera. We then started off by downloading the OpenCV package which will allow us to preprocess images and video received from the camera and have it displayed as an output for VSLAM. By running a sample test script we're able to see how the processor will take in imaging information and also be able to stream real-time video.

Another test that we have to use is how the camera will operate under certain conditions. The goal of this project is to accurately implement VSLAM so that the map can be generated properly for path planning. This would test the constraints on how much the image can make out details in various scenarios. For object detection, we would have to test a sample object to ensure that the camera will display an indicator whether something is picked up. This will involve ORB-SLAM and how the camera will be able to extract key points and features. This will determine how the map would be generated and then used within our navigation software.

9.1.6 Motors

The robot uses four brushed DC gear motors, each controlled by a VNH5019 motor driver IC. However, initial testing did not include the motor driver ICs. Initial testing was conducted on a single motor and involved the use of lab equipment. A function generator was used to simulate a 12V PWM signal that connected to the leads of the motor. By adjusting the duty cycle of the PWM signal on the function generator, we were able to successfully control the speed of the motor.

We then moved on to using the ESP32 to generate the PWM signal to control the speed. However, the ESP32 can only generate a signal with a voltage of 3.3V-5V. Thus, on a breadboard, a MOSFET was used to isolate the EPS32 from the motors. The PWM signal from the ESP32 was fed into the gate of the MOSFET, a DC power supply was providing 12V to the drain of the MOSFET, and the drain of the MOSFET connected to the motor. The outcome of this was that we could now control the motor using the EPS32. However, the signal became inverted and this needed to be taken into account when adjusting the duty cycle in the code. As this is just for prototyping and testing, it was not deemed necessary to further build a circuit to fix the inversion.

9.2 Software Testing

9.2.1 ESP32-WROOM-32E

To ensure that the coding environment has been properly set up and been connected to the ESP32-WROOM-32E development board we used a simple code that will have an LED turn on and off. We started off by creating a GitHub repository and cloning it onto a local device. To run code on the ESP32 we choose using the PlatformIO extension on VScode and we choose the Arduino framework. The ESP32 was connected to an LED and resistor on a breadboard and the simple Blink code was uploaded. The blink code correctly flashed the LED on and off. From this point, we started working on different peripherals to run using the ESP32 and began running some example codes to ensure these parts can run in the environment.

9.2.2 RFID Module and Tags SPI Communications

To test if the SPI communication between the ESP32 and the RFID module is working properly the RFID module was connected to a breadboard and the pins were properly connected to the ESP32. There are two channels of SPI communication on the ESP32. There is High Speed SPI (HSPI) and Very High Speed SPI (VSPI). VSPI was chosen to test the communication but this choice between the two for testing was arbitrary. VSPI uses the pins found in Table 31 on the ESP32.

Table 31: ESP32 VSPI Pins [C.8]

ESP32 Pins for HSPI

MOSI	MISO	SCLK	CS
GPIO 23	GPIO 19	GPIO 18	GPIO 5

Next, we had to test the reading and writing functionality of the code. After some research, a few online sources with explanations and examples and an open-source GitHub repository created by Miguel Balboa were helpful in coding the RFID module to read and write [A.95][A.96]. This code, found in Appendix D.1, asks if the user wants to perform reading or writing and to enter an “r” or “w” respectively. Once uploaded to the ESP32 and the pins connected via a breadboard the RFID module does read when the PICC tags are put close and writes when put into write mode.

9.2.3 ESP32 to Raspberry Pi UART Communications

The ESP32 microcontroller and Raspberry Pi 5 will communicate with a UART communication protocol. This will be tested by connecting the ESP32 development board by the micro-USB port into one of the Raspberry Pi’s USB ports. The serial UART connection will happen through this connection.

The procedure for testing if this connection is active is to

- 1) Upload the C++ code to the ESP32 through your computer.
- 2) Connect with the micro-USB on the ESP32 to a USB port on the Raspberry Pi.
- 3) Run the Python code on the Raspberry Pi
- 4) Watch the terminal on the Raspberry Pi and the on board LED on the ESP32 for proper behavior.

The code on the Raspberry Pi is written in Python and it simply creates a serial connection with the correct port. In an infinite loop it writes a ‘0’ or a ‘1’ to the serial connection which will then be read by the ESP32. These writes are separated by a read line, a print line, and a time delay. The read line will take what the ESP32 is communicating across the line and store it in a variable. The print line prints what was just read to the serial monitor and the time delay is there is enough time to see that light turned on or off on the ESP32 [D.2].

The code on the ESP32 establishes a serial connection, sets the on board LED pin to output, and on first start-up sends a message across the serial connection. In a loop it will continuously check that the serial is available and when it is it will read from it. If the data is a 1, the on board LED pin is set to HIGH to turn it on. If the data is a 0, the on board LED pin is set to LOW [D.3].

The code written for this test will have the Raspberry Pi sending a signal that the ESP32 will interpret and determine when to turn the on board LED on and off. The ESP32 will send serial messages that the Raspberry Pi will print out to the terminal.

9.2.4 I2C Communications

The two components that will use I2C as their communication protocol is mainly the IMU. The ESP32 will be the controller device and the sensors will be peripheral devices.

The MPU6050 IMU sensor that we are using in this project will be using mode 1. Mode 1 I²C / MIPI I3CSM slave interface between the IMU and the ESP32. I will not need to connect to other external sensors or the camera module. For calibration...

9.2.5 TCP Network Communications

We will be addressing common issues that can occur in networks. Some problems include network congestion, QoS/bandwidth throttling, latency variability (random spikes), restricted bandwidth, packet loss, out-of-order packets, fragmentation, and interrupted connections. These are not the only possible issues, but they are important enough to include in our test portion. We will address them by simulating these conditions in real time. For example, we can simulate periodic network congestion using tools like Wondershaper or NetLimiter to vary the bandwidth available to our robot. In addition, we can address fragmentation problems by issuing specific fragmentation commands and sending them into the network. Afterwards, we will determine how the correction algorithms handle this load and then progressively increase the load for each situation. Given the academic context in which the project is being developed, it is reasonable to assume this will be enough to determine whether the network portion meets our standards. This is even more imperative when dealing with packets.

Packets will likely be the main issue we will fix because they suffer from what we call “Confirmation Chain Degradation.” (CCD) Confirmation chain degradation occurs when packets experience one of four problems:

1. The sender loses packets.
2. The sender transmits packets out of order.
3. The receiver sends ACKs out of order.
4. ACKs become corrupted after being sent back to the sender.

These problems can occur in combination, but typically only one of the four affects packets at a time. For this reason, we decided that the best approach is to burden our server with progressively lost packets, increase both packet and ACK loss/corruption, and observe how it recovers using common network standards.

Another common issue is connection disconnects, which typically occur when TCP times out. This can happen with the built-in FSM RDT protocol, Go-Back-N, and Selective Repeat sliding-window protocols when they experience a long delay that triggers the timeout. Because this affects both the sender and the receiver, we will progressively inject RST packets into the server. From this unit testing we can determine whether the server can handle disconnects well and decide whether we need to improve it.

These are a few of the things we can do to test the network portion of the project. They are not the only possible tests, but they exemplify how we would approach the network portion to ensure the reliability of our system.

9.2.6 SLAM Toolbox

ORB-SLAM requires that we utilize our visual sensors such as the Raspberry Pi V2 in our case and the 3 TOF sensors that will work as sensor input. Once the system is set up, the Raspberry Pi V2 camera should be calibrated using a chessboard pattern to obtain accurate parameters (focal length, principal point, distortion coefficients) so ORB-SLAM will be able to recognize these within the video stream. In order for SLAM to function we need all sensors to be recognized and be calibrated to start generating the indoor map.

It is important to monitor the system for SLAM initialization, map generation accuracy, and camera pose tracking stability. Additionally, ToF sensor data should be logged to ensure the robot correctly detects surrounding objects and possible collision opportunities. We need to validate the ESP32's functionality by confirming it can consistently transmit sensor readings to the Raspberry Pi, with the lowest latency. RViz or Pangolin's built-in viewer as a visual debugging tool can also demonstrate the trajectory and point cloud map generated by ORB-SLAM. For the main system test we will be using Gazebo and RViz within ROS2 Jazzy Jalisco. We would then create a urdf with the necessary joint links to simulate our differential drive robot. We created a virtual environment with a sample map to simulate an indoor environment and that will allow us to understand how SLAM will generate the map for path planning. Then we can simulate an endpoint for the robot to navigate and then run our process.

This will require multiple runs to ensure that the view of the camera within RViz is accurately gathering features and keypoints so that the calibration of ORB-SLAM is working. This will then need to be translated to the Raspberry Pi to ensure that the tests that were initially run are working.

9.2.7 Mobile App

Throughout the development of the mobile app, we constantly tested whether the UI was responsive to user input as far as authentication for login and logout. We had initialized a firebase authentication to ensure the user, email and password were in the database for the backend. We created and tested our API using postman to understand and test what our API calls are outputting.

Our app will mainly function around the connection between flutter and the raspberry pi to receive notifications from SLAM. For this we will test a package called flutter-pi that will allow us to add our Raspberry Pi 5 device and connect to it in real-time. We would use a software called Flask that will allow us to ping from the pi to the app. Within the code, we would code in the test cases whether the connection status has connected or not. We would need to assess latency times and handle error cases within the HTTP server. As for the RFID connection we would need to program the dependencies for the RYRR10S reader module into the Raspberry Pi to recognize the tag that will be placed on it. When

the Raspberry Pi publishes that the payload has been detected, the app should actively listen for the message and then display a toast notification reflecting the change in robot state to “Payload Identified.”

Further testing will involve how the app will receive the notification that the delivery has been made and if it will receive an error code for when the robot. We must update the UI to reflect the status, such as if it's “In Progress” or “Completed.” Additionally, edge cases should be tested including handling invalid or unknown RFID tags and verifying automatic reconnection or error prompts in the app.

Chapter 10: Senior Design II Project Updates and Challenges

10.1 ORB SLAM Challenges and Switching to 2D SLAM with LiDAR

When choosing ORB-SLAM for our main SLAM algorithm, it was hard to account for how the ROS distribution in hardware made it ultimately harder to build and run altogether. With ORB-SLAM it was not able to build properly therefore it couldn't be solely run on the Raspberry Pi since it required a more powerful computer to run it. Hence, why we made the decision to use a 2D lidar sensor since we were able to borrow it and use it alongside the SLAM toolbox which is the staple for lidar SLAM. This allowed us to run SLAM solely on the Raspberry Pi without the strain of relying on the camera and its calibration. Navigation 2 also was easier to implement in SLAM since their packages are directly compatible with one another for easy tuning.

10.2 RFID Module and IMU Part Switch

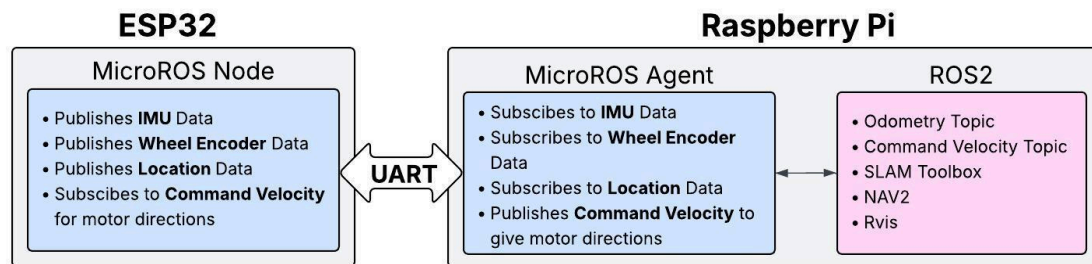
Originally we chose the LSM6DSO because the features of the chip suited our project but we then switched to use the IIM-20670 because Dr. Weeks says we needed pins on the bottom and we accidentally over-filtered on DigiKey. Ultimately we had a hard time programming this IMU and did not want to spend more time trying to get this one working. We switched to using the IMU that we were using on a development kit, the MPU-6050, and broke it out on a separate PCB so we could place it somewhere else.

The original RFID module that we were using had very poor documentation and no example code or libraries online for UART communication. The RFID module we ended up using was the PN532 because it had an easy to use UART library.

10.3 Further Explanation of UART Communication between Raspberry Pi and ESP32 using MicroROS Framework

In order to communicate effectively and efficiently from the sensor data to NAV2 on ROS2 we used the MicroROS framework to communicate over UART. MicroROS is designed to natively communicate from a microcontroller to the ROS2 with the same node architecture that ROS2 uses.

The Raspberry Pi starts the MicroROS Agent and waits for the ESP32 to create the MicroROS Node and initiate a connection. With the same node the ESP32 has multiple publishers and publishes the IMU data, Wheel Encoder data, RFID Location information, and it also subscribes to the Command Velocity topic to know how to move the motors. The MicroROS Agent then will communicate this information to ROS2 automatically because the ESP32 nodes are subscribed to the ROS2 topics that are being used in RViz and NAV2.



10.4 Odometry Challenges and Sensor Calibration and PID Controller

One of the biggest challenges of this project was with calculating odometry, or the process of estimating a robot's position and orientation over time by using motion sensors. Having correct odometry is crucial for the SLAM algorithm to determine where the robot is in relation to everything else. In the project odometry was calculated with Motor Encoders and the IMU. The strategies to improve the accuracy of the odometry are outlined in each of the sections below.

10.4.1 Calibrating the IMU with a more precise IMU library

The final IMU chosen for this project was the MPU6050 and originally we were using the Adafruit_MPU6050 library. This library is easy to use and has a few calibration options. The most accurate way to calibrate an IMU is to add values to the offset hardware registers and these values will be used before internal filtering. The Adafruit_MPU6050 library does not have direct access to these registers and the only way to calibrate would be to add manual biases to the values after filtering and converting from raw data to SI units. In order to calibrate it by adding offset values to the offset hardware registers we switched to a more robust library that had a PID tuning algorithm for calibration and functions built in to set the offset values. Below are tables that show the values before and after calibration.

Table: IMU Measurements in SI Units Before Calibration

Accelerometer in m/s ²			Gyroscope rad/s		
Ax	Ay	Az	Gx	Gy	Gz
7.10	-2.50	6.16	-3.40	1.14	0.20

7.64	-2.55	6.21	-3.19	1.21	0.05
7.76	-2.48	6.15	-3.28	1.07	0.24
7.72	-2.57	6.15	-3.28	1.07	0.24
7.71	-2.48	6.14	-3.27	1.23	0.31

Table: IMU Measurements in SI Units After Calibration

Accelerometer in m/s ²			Gyroscope rad/s		
Ax	Ay	Az	Gx	Gy	Gz
-0.01	0.16	9.71	0.00	-0.00	0.00
-0.01	0.16	9.68	-0.00	-0.00	-0.00
-0.07	0.17	9.62	0.00	0.00	-0.00
-0.09	0.19	9.65	0.00	0.00	-0.00
0.01	0.20	9.67	0.00	0.00	0.00

As you can see after proper calibration the numbers are much more accurate to what is happening. The Gyroscope in all 3 directions should be 0 because it is not moving or rotating at all during calibration. The acceleration measuring the Z-direction should be 9.81 m/s² because that is the speed of gravity. This is not completely accurate but it is much closer after calibration.

10.4.2 Using an IMU Filter Node

Even after calibration we noticed the IMU data values were still fluctuating a lot even when the IMU was not moving. This variation is just noise and it's from the quality of the IMU we chose. To minimize the noise when not moving we used a ROS2 IMU Filter Node that would take the raw IMU data (still in SI units) and then filter it to reduce noise before going into NAV2 and SLAM toolbox.

10.4.4 Implementing a PID Controller to reduce drift

Originally we had a major amount of drift on our robot. The left wheel would consistently lag behind the right wheel and our robot would end up turning in a circle after going for more than 5 seconds. At first we thought this could be a mechanical problem with the shape of our chassis and if the wheels were on straight. After rebuilding the chassis and fixing mechanical issues the drift still persisted. In order to reduce drift and allow for accurate odometry we needed to implement a PID controller. This would

allow the motors to actually move at the speed we set them to and then be able to accurately calculate the position of each increment of movement.

During the process of incrementing the PID controller we also realized that we had to increase the frequency that the nodes on the Raspberry Pi and ESP32 were publishing and receiving subscription messages. Before it was publishing data and calculating odometry information at 1 Hz. In an ideal condition PID controllers should be running at clock speed but because we did not want to overload the ESP32 processor we only increased the speed to 10 Hz. This was fast enough for the PID controller to actually update the PWM to change the motors to be the accurate velocity we were trying to achieve.

PID controllers have three gain constants: proportional, integral, and derivative. And there is conflicting information on sources on the internet whether you should adjust k_p , k_i then k_d last or adjust k_p , k_d , and then k_i last. We tuned the k_p until it was oscillating, then k_d to reduce oscillations, and lastly k_i to smooth variations.

We also recommend starting when the wheels are off the ground not under any load. This will allow you to see if your frequency is enough and adjusting your gain constants do what is expected of that. When running underload there's going to be more factors of the robots weight, uneven ground, etc that will influence the charts. After you know that the PID controller is working correctly then put it under load at the speed that you expect to operate at most frequently.

Also, make sure to find a software to in real time view the PWM signals and current velocity calculations. I was using the PlatformIO extension on VSCode to code and it was really easy to use the Teleplot extension with this.

10.4.5 Raspberry Pi 5 and ESP32 Failures and Instability

As previously mentioned, the Raspberry Pi 5 and the ESP32 communicated with one another via UART. The first hardware issue we encountered with these two components was the sudden failure in communication between them. Despite working perfectly fine just a couple of hours prior, the ESP32 was failing to initialize UART communication with the Raspberry Pi 5. After taking measurements and running tests, we were able to deduce that the Raspberry Pi 5 UART channel was no longer functioning. We then swapped it for another Pi and continued testing and ensured it could communicate with the ESP32. However, just a few hours later, when trying to upload new code to the ESP32, we found that we were unable to at this point and again ran some tests and found that the ESP32 needed to be replaced.

Once the ESP32 was replaced and we began to test the robot again, we found that the second Raspberry Pi was no longer turning on. We discussed and ordered another one but the same thing happened again with the third Pi. With Dr. Weeks, we took a look at the regulator powering the Pi and the voltage at the Pi and observed that it was at 5.2V, just under the maximum input voltage for the Pi of 5.25V. We adjusted the voltage output of the regulator to 5V and we no longer encountered that issue.

We also ran into a number of issues with the ESP32 chips in that they failed to communicate with either the RFID module or the IMU board, but functioned fully otherwise. We continued to test new chips until we installed one that could communicate fine with all of the devices.

On the day of our live demo, after making a plan to record everything all at once with the app integrated, the ESP32 suddenly stopped communicating with the Pi again. The UART channel on the ESP32 worked just fine when running tests, but on the Pi it was unable to transmit proper signals: an indication that the UART channel was once again fried. We discussed it with both Dr. Chan and Dr. Weeks and hypothesized that this could be due to poor ground return in the circuit with the Pi and resulting in a ground loop that leads to differences in potential. To counter this, we used a short piece of solder wick to ground the Pi to the main circuit from a second point.

10.5 Network Challenges on ESP32, Raspberry Pi 5, and Mobile App

Another major challenge was getting the network stack working as a reliable bridge between the app and the hardware. Initially, we used a router and tried to configure it as an access point (AP), but setup and debugging consumed too much time. We switched to using phone hotspots as the AP so we could more easily test the app's interaction with the robot.

We then ran into issues with the MQTT broker: the app and ESP32 were having trouble connecting to our cloud broker (EMQX). On top of that, which hotspot we were connected to mattered. Changing hotspots affected the app's ability to receive telemetry from the ESP32 and the livestream/livemap images from the Raspberry Pi 5. Hosting the web servers also required extra configuration so they would work correctly over the AP and not be blocked by the network setup.

To address this, I configured the ESP32 as an AP web server that both published telemetry to the broker and stored the same data locally as JSON. The app then pulled and decoded this data from the ESP32's web server using HTTP GET requests. I used a similar approach on the Raspberry Pi 5, but without involving the broker: we set up a standard HTTP server on the Pi, connected it to my hotspot, and sent JPEG images of the livemap and livestream over the network for the app to decode and display.

The final recurring issue was changing IP addresses. Every time we swapped an ESP32 or Raspberry Pi board, the device would receive a new IP address, which meant we had to update the app's configuration to reach the correct web servers. This was annoying but ultimately a minor issue that did not break core functionality. To build confidence before full integration, we repeatedly mocked data on both the ESP32 and Raspberry Pi and tested until the app consistently handled telemetry and image data as expected.

Chapter 11: Administrative Content

11.1 Budget

Below is our budget estimation, The cost of the materials will primarily be covered by the group and we aim to put in at most 500\$ with possibility that this value may change.

Table 32: Budget Estimation

Part Description	Quantity	Estimated Cost	Total Cost
Battery	1	\$30	\$30
Sensor	6	\$5	\$30
MCU	20	\$0.50	\$10
Wheel and Motor	2	\$80	\$160
PCB + Shipping	5	\$4	\$35
Wood	1	N/A	\$50
Estimated Cost	\$315		
Max Budget	\$500		

11.2 Bill of Materials

Table 33: Bill of Materials

Bill of Materials					
Item	Quantity	Manufacturer	Part Number	Price	Total Price
Microcontroller	10	Espressif Systems	ESP32-WR OOM-32E	\$4.66	\$46.62
Raspberry Pi 5	1	Raspberry Pi	SC1110	\$50.00	\$50.00
ToF	6	TinyCircuits	AST1024	\$9.95	\$59.70
RFID	2	Reyax	RYRR10S	\$6.00	\$12.00
Photodiode	5	Vishay	BPW82	\$1.13	\$5.65
Camera	1	Raspberry Pi	SC0023	\$29.95	\$29.95

IMU	3	ST Micro	LSM6DSO	\$4.00	\$12.00
Motor	2	Pololu	2828	\$51.95	\$103.90
Battery	1	WEIZE		\$44.99	\$44.99
Charger	1	Beikalone		\$9.98	\$9.98
Adj VR	6	TI	LM2576	\$3.72	\$22.32
High Power VR	3	TI	LM2679	\$6.05	\$18.15
32.768 kHz Oscillator	10	Citizen Finedevice Co Ltd	CFS-20632 768DZBB	\$0.16700	\$1.67
Materials for Demo					
Microcontroller Board Dev	1	Initeq		\$36.99	\$36.99
Total Price:	\$432.74				

11.3 Project Milestones

Table 34: Senior Design 1 Milestones

Senior Design 1 - Spring 2025		
Week	Tasks	Assignment Due Dates
1-2	Form Group and Discuss Project Ideas	N/A
3	Refine Project Idea Attend Bootcamp Form Review Committee Write Divide and Conquer Document (10 pages)	Divide and Conquer Document 1/24
4	D&C Meeting Revise D&C Document	
5	Revise D&C Document Start Individually Researching Components (Chapter 3)	Revised D&C 2/7
6	Individually Researching Components (Chapter 3)	

7	Individually Researching Components (Chapter 3) Standards and Design Constraints (Chapter 4) Comparison of ChatGPT (Chapter 5)	
8	Standards and Design Constraints (Chapter 4) Comparison of ChatGPT (Chapter 5)	
9	Hardware Design Chapter 6 Software Design Chapter 7	Quizzes 3/7
10	Hardware Design Chapter 6 Software Design Chapter 7	
11	Review each others sections and final adjustments on Midterm Report (60 pages)	Midterm Report 3/24
12	Midterm Report Revisions Chapter 8 PCB Layout Chapter 9 System Testing and Evaluation	
13	Submit Midterm Report Revision Chapter 8 PCB Layout Chapter 9 System Testing and Evaluation	Midterm Report Revision 4/4
14	Chapter 8 PCB Layout Chapter 9 System Testing and Evaluation	
15	Review each others sections and final adjustments on Final Report Make Mini Demo Video	
16	Submit Final Report Submit Mini Demo Video	Final Report 4/22 Mini Demo 4/22

Table 35: Senior Design 2 Milestones

Senior Design 2 - Fall 2025		
Week	Tasks	Assignment Due Dates
	PCB Design and Ordering	9/18/25
1-2	PCB Testing and Construct Body of Robot	9/24/25
3-7	Develop Software	10/13/25
8	Full Assembly Test	

9-12	Research Solutions, Make Changes	11/7/25
13	Full Assembly Test 2	11/21/25
14	Practice Demo and Final Presentation	11/23/25
15	Live Demo	11/26/25

11.4 Work Distributions

The table below outlines the distribution of work amongst teammates and details the assigned tasks to each of the group members. The major tasks are listed with a group member with primary responsibilities and a group member with secondary responsibilities. This is crucial to ensure that all work is reviewed by another member. This also helps to ensure that if an emergency happens to a group member the rest of the team has a plan to continue work. All tasks rely in some way on other member's tasks so collaboration is present among the group members at every level of this project to ensure these systems are designed in a manner that they fit other system's requirements. This ensures that all members are aware of and involved in the design and implementation of systems, regardless of if they are directly responsible for it or not.

Table 36: Work Distributions

Task	Primary	Secondary
Research and Part Selection		
RFID	Riley	Ibrahim
Distance Sensor	Riley	Ibrahim
Payload Detector	Riley	Ibrahim
Visual Sensor (Cameras)	Melyia	Brandon
IMU	Ibrahim	Melyia
Microcontrollers	Ibrahim	Riley
Motors	Ibrahim	Riley
Single Board Computers	Melyia	Ibrahim
Power Supply	Ibrahim	Brandon
Wifi/Bluetooth Connection	Brandon	Melyia
Network Communication	Brandon	Ibrahim

SLAM Protocols	Melyia	Brandon
Image Processor	Melyia	Brandon
Mobile App	Brandon	Melyia
Hardware Design		
Subsystem Block Diagram	Ibrahim	Brandon
Schematic Diagrams	Ibrahim	Brandon
Software Design		
Activity Flow Diagram	Melyia	Riley
Use Case Diagram	Riley	Melyia
System State Diagram	Riley	Melyia
Connection and Communication	Brandon	Riley
SLAM Algorithm	Melyia	Riley
Mobile App	Brandon	Melyia
System Fabrication		
PCB Layout	Ibrahim	Brandon
Prototyping Construction	Ibrahim	Riley
Hardware Testing		
Microcontroller	Riley	Ibrahim
Raspberry Pi 5	Melyia	Brandon
RFID	Riley	Ibrahim
Photodiode	Riley	Ibrahim
Camera	Melyia	Brandon
Motors	Ibrahim	Riley
Software Testing		
Embedded System	Melyia	Ibrahim

UART Communication	Brandon	Melyia
I2C Communication	Brandon	Riley
TCP Network Communication	Brandon	Melyia
Mobile App	Melyia	Brandon
System Testing and Integration		
Device Communication	Melyia	Brandon
Sensor Communication	Riley	Melyia
Navigation Testing	Melyia	Brandon
Constructing Robot	Ibrahim	Riley
Motors	Ibrahim	Brandon

Chapter 12: Conclusion

In conclusion, the Autonomous Delivery Robot designed in this report, developed as a senior design project in the Electrical and Computer Engineering department at the University of Central Florida, represents a step forward in the effort to modernize and optimize the process of transporting payloads across large warehouses. As outlined in the executive summary, the primary objective of the Autonomous Delivery Robot is to optimize payload transportation within warehouse environments to reduce human labor and streamline warehouse operations. To do this the robot system designed to solve this problem will navigate with minimal user intervention and that could operate independently. The design of this robot demonstrates the capabilities and benefits of combining embedded systems, sensors, wireless communication, and mobile apps to solve real-world logistics problems. The solution devised by these engineering students involves key features such as obstacle avoidance, RFID-based payload identification, automatic path planning via the SLAM API, and a user-friendly mobile application for controlling and monitoring.

The RFID based payload detection will be extremely easy for the user to be able to identify payloads and determine the payloads' locations. Payloads are identified using RFID tags so the robot can then determine the payload's final destination automatically and accurately. After identifying the payload the robot plans the most efficient route to deliver the payload to its designated drop-off point. The navigation system combines real-time sensor data with pre-mapped coordinates to adapt to any obstacles found along the way to deliver the payload. The SLAM algorithm will determine the best path to the final destination read by the RFID module. The SLAM algorithm was chosen to be implemented as it will be able to have an understanding of the robot's location along the

path. It will be our main focus for navigating within an indoor environment using localization and pre-mapping techniques. SLAM algorithms have to calculate the current position and orientation by using an environment map in order to function well. Along the way, the robot will have real-time obstacle avoidance with the help of the optical time of flight sensor. This sensor was determined the best and fastest way to determine if there is an obstacle in the robot's path because of its fast speeds and longer ranges at an inexpensive price point. The mobile application has been designed with its main goal to be to give the user detailed insight into the state of the robot. It will keep the user informed throughout the robot's operations with status updates. This project features a custom made PCB design that includes the layout and connections of all major hardware components. Namely, the ESP32 microcontroller, Raspberry Pi, RFID module, IMU, camera, and optical time of flight sensor. The main hardware systems are the motor controlling process, sensor integration and communication, and voltage regulation and power distribution. It also includes basic circuit protection features such as voltage regulators and decoupling capacitors.

This document serves as a comprehensive record of the Autonomous Delivery Robot design process, covering fundamental technologies and concepts, part selection, implementation, design constraints and testing limitations, and administrative and financial aspects. The design phase allowed the team to deeply explore the theoretical and technical considerations behind each part of the system. The major design constraints were time and the team's financial constraints. This report has multiple detailed schematics, block diagrams, and flowcharts that were created to visualize the relationship and dependencies between components and guide the implementation in the next phase of the project. All the components, communication protocols, and software algorithms and programs were carefully selected based on technical specifications, cost, compatibility, and availability. This semester the team completed the process of sourcing and ordering all major components required for the build and we now have all major components in our possession.

The meticulous planning of this project, coupled with the innovative solutions devised by the University of Central Florida's engineering students, demonstrate that we are on track for the successful deployment and operation of Autonomous Delivery Robot at the end of Senior Design 2 in the Fall 2025 semester. In Senior Design 2 the full assembly of the robot was accomplished using the components ordered this semester and integrated 2D SLAM algorithm with all the sensors. We implemented and calibrated the navigation and the sensor system. Following this was the testing of the complete system and iterative system improvements and changes. Along the way we refined the mobile app and ensured that it properly updates during each phase of delivery.

In conclusion, this autonomous delivery robot project has laid a strong foundation for future implementation through detailed planning, thoughtful design, and strategic component selection. As the team moves forward with the project, the insights gained from the research and design phases of this project will undoubtedly contribute to successful completion of future prototyping, testing, and executing this project next

semester. The completed design will lay a strong foundation for future development and real-world implementation of an intelligent, autonomous delivery system.

Appendices

Appendix A – References

- [1] America, “Starship Technologies Autonomous Delivery Robots- Robot Guide,” *Bots & Drones America*, 2022. <https://botsanddrones.co/commercial-robots/f/starship-technologies-autonomous-delivery-robots---robot-guide?blogcategory=%40Starship+Technologies> (accessed Jan. 19, 2025).
- [2] C. Bai, P. Dallasega, G. Orzes, and J. Sarkis, “Industry 4.0 technologies assessment: A sustainability perspective,” *International Journal of Production Economics*, vol. 229, no. 229, p. 107776, Nov. 2020, doi: <https://doi.org/10.1016/j.ijpe.2020.107776>.
- [3] “Starship Technologies - Starship Robot,” *Wevolver*. <https://www.wevolver.com/specs/starship-technologies-starship-robot>
- [4] “Our Robots - Starship Technologies: Autonomous robot delivery - The future of delivery - today!,” *Starship Technologies: Autonomous robot delivery - The future of delivery - today!*, Nov. 15, 2024. <https://www.starship.xyz/our-robots/>
- [5] “Index of /seniordesign/fa2022sp2023,” *Ucf.edu*, 2022. <https://www.ece.ucf.edu/seniordesign/fa2022sp2023/> (accessed Jan. 24, 2025).
- [6] Half As Interesting, “How to Read Barcodes,” *YouTube*, Nov. 02, 2017. <https://www.youtube.com/watch?v=XPuTZMp-HE8> (accessed Feb. 06, 2025).
- [7] Taptic Digital, “How Does a Barcode Scanner Work?,” *YouTube*, Mar. 02, 2024. <https://www.youtube.com/watch?v=aMCjw20DftY> (accessed Feb. 06, 2025).
- [8] Scanova, “How Do QR Codes Work: Explained in Minutes,” *YouTube*. Feb. 21, 2022. Accessed: Feb. 06, 2025. [Online]. Available: <https://www.youtube.com/watch?v=pamazHwk0hg>
- [9] P. McGlynn, “How RFID Technology Works: Revolutionizing the Supply Chain,” *BlueFletch*, Oct. 03, 2023. <https://bluefletch.com/rfid-technology-how-it-works-and-has-the-power-to-transform-the-supply-chain/>
- [10] ALL ABOUT ELECTRONICS, “What is RFID? How RFID works? RFID Explained in Detail,” *www.youtube.com*, Jun. 25, 2017. <https://www.youtube.com/watch?v=Ukfpq71BoMo>
- [11] “RFID and the Differences in Passive, Semi-Passive, and Active Tags - Computype,” *Computype*, Jan. 11, 2022. https://computype.com/blog/rfid-and-the-difference-in-passive-semi-passive-and-active-tags/?srsltid=AfmBOorcX6uRaLTkHuovC0CTEj4RV06d2Knw1mrhwQNll-4hSpB_QYeK (accessed Feb. 06, 2025).

- [12] “Passive RFID versus Active RFID - What is the difference?,” *RFID4U*. <https://rfid4u.com/rfid-passive-active/> (accessed Feb. 07, 2025).
- [12] “How to Select a Correct Tag – Frequency,” *RFID4U*. <https://rfid4u.com/rfid-frequency/>
- [13] kandan84, “How Wireless Energy Transfer Works,” *YouTube*. Feb. 12, 2012. Available: <https://www.youtube.com/watch?v=-Wf7aadxBkE>
- [14] “Inductive and Backscatter Coupling - How Passive RFID works,” *RFID4U*. <https://rfid4u.com/inductive-and-backscatter-coupling/> (accessed Apr. 07, 2021).
- [15] ALL ABOUT ELECTRONICS, “NFC Explained: What Is NFC? How NFC Works? Applications of NFC,” *YouTube*, Mar. 11, 2018. <https://www.youtube.com/watch?v=eWPtt2hLnJk> (accessed Feb. 06, 2025).
- [16] OpenAI. (2025). ChatGPT (GPT-4o) [Large language model]. <https://chat.openai.com/chat>
- [17] element14presents, “How Do Ultrasonic Distance Sensors Work? - the Learning Circuit,” *www.youtube.com*. Jan. 27, 2021. Accessed: Feb. 18, 2025. [Online]. Available: <https://www.youtube.com/watch?v=2ojWO1QNprw>
- [18] “LIDAR, Optical Distance & Time of Flight Sensors - ams-osram - Ams,” *ams-osram*, 2022. <https://ams-osram.com/innovation/technology/depth-and-3d-sensing/lidar-optical-distance-and-time-of-flight-sensors> (accessed Feb. 18, 2025).
- [19] SiLC Technologies Inc., “FMCW vs. ToF LiDAR,” *YouTube*, May 18, 2020. <https://www.youtube.com/watch?v=viW5n1xuTq8> (accessed Feb. 18, 2025).
- [20] Texas Instruments, “Understanding Optical Time-of-Flight (ToF) Technology,” *YouTube*, Sep. 19, 2018. <https://www.youtube.com/watch?v=TpjnooXhOmY> (accessed Feb. 18, 2025).
- [21] Kate, “Time of Flight Sensor vs. LiDAR: What Are the Differences?,” *Precision Measurement Technologies*, Mar. 06, 2024. <https://pmt-fl.com/time-of-flight-sensor-vs-lidar-what-are-the-differences/> (accessed Feb. 18, 2025).
- [22] P. Kumar, “What Is a ToF sensor? What Are the Key Components of a ToF camera? – E-con Systems,” *e-con Systems*, Oct. 17, 2023. <https://www.e-consystems.com/blog/camera/technology/what-is-a-time-of-flight-sensor-what-are-the-key-components-of-a-time-of-flight-camera/> (accessed Feb. 19, 2025).

- [23] A. Shaik, "Electronic Devices and Circuits - What is electronics - Electronics definition," *Physics-and-radio-electronics.com*.
<https://www.physics-and-radio-electronics.com/electronic-devices-and-circuits.html>
 (accessed Feb. 27, 2025).
- [24] Butler Technologies, "Force Sensing Resistors & Their Applications," *butlertechnologies.com*, Jul. 06, 2022.
<https://butlertechnologies.com/blog/force-sensing-applications> (accessed Feb. 27, 2025).
- [25] "Design for Manufacturing -Guidelines." Available:
<https://www.unm.edu/~bgreen/ME101/dfm.pdf>
- [26] Zheng, "Embracing the Versatility of ICODE SLIX RFID Labels Compliant with ISO15693 and NFC Type 5 Specifications," *RFID Card*, Jun. 12, 2023.
<https://www.rfidcard.com/the-difference-between-iso15693-and-iso14443/> (accessed Mar. 11, 2025).
- [27] Application Note A Basic Guide to I2C, Texas Instruments, Accessed: March 13, 2025 [Online]. Available:
<https://www.ti.com/lit/an/sbaa565/sbaa565.pdf?ts=1741852202490>
- [28] R. S. D. Souza, "I2C Protocol - Prodigy Technovations," *Prodigy Technovations*, May 18, 2023. <https://www.prodigytechno.com/i2c-protocol> (accessed Mar. 13, 2025).
- [29] E. Pena and M. G. Legaspi, "UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices," *Analog.com*, 2024.
<https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>
- [30] "ISO 14443 NFC Standard - GoToTags," *GoToTags*, Jan. 22, 2024.
<https://gototags.com/help/nfc/standards/iso-14443> (accessed Mar. 14, 2025).
- [31] M. Filipenko and I. Afanasyev, "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment," *IEEE Xplore*, 2018.
<https://ieeexplore.ieee.org/abstract/document/8710464/> (accessed Apr. 10, 2021).
- [32] J. Volak, J. Bajzik, S. Janisova, D. Koniar, and L. Hargas, "Real-Time Interference Artifacts Suppression in Array of ToF Sensors," *Sensors*, vol. 20, no. 13, p. 3701, Jul. 2020, doi: <https://doi.org/10.3390/s20133701>.
- [33] "24mm Brushless DC Planetary Gear Motor, 12V, 23RPM," RobotShop USA.
<https://www.robotshop.com/products/e-s-motor-24mm-brushless-dc-planetary-gear-motor-12v-23rpm?qd=7d075d6ffcc24e24152ea900c098d2d5> (accessed Mar. 19, 2025).

- [34] “150:1 Metal Gearmotor 37Dx73L mm 12V with 64 CPR Encoder (Helical Pinion),” Pololu.com. <https://www.pololu.com/product/2828> (accessed Mar. 19, 2025).
- [35] K. Araujo, “Lithium Battery Technologies - Li-Ion and LiFePO₄ Battery Pack Systems,” *Epectec.com*, 2020. <https://www.epectec.com/batteries/lithium-battery-technologies.html>
- [36] Lighting Global, Lithium-ion Battery Overview, Issue 10, May 2012. [Online]. Available: https://www.lightingglobal.org/wp-content/uploads/2012/05/67_Issue10_Lithium-ionBattery_TechNote_final.pdf. [Accessed: Mar. 19, 2025].
- [37] C. Resende, “Best Power Tool Battery Types: NiCd VS NiMH VS li-ion VS li-polymer,” *Best Power Tools For Sale, Expert Reviews and Guides*, Nov. 03, 2017. <https://www.powertoollab.com/power-tool-battery-types/> (accessed Mar. 19, 2025).
- [38] “Rechargeable Batteries explained in detail (NiMH, NiZn, NiCd, RAM),” *michaelbluejay.com*. <https://michaelbluejay.com/batteries/rechargeable.html>
- [39] Y. Mekonnen, H. Aburbu, and A. Sarwat, “Life cycle prediction of Sealed Lead Acid batteries based on a Weibull model,” *Journal of Energy Storage*, vol. 18, pp. 467–475, Aug. 2018, doi: <https://doi.org/10.1016/j.est.2018.06.005>.
- [40] I. Cowie, “All About Batteries, Part 3: Lead-Acid Batteries,” *EE Times*, Jan. 13, 2014. https://www.eetimes.com/all-about-batteries-part-3-lead-acid-batteries/?_ga (accessed Mar. 19, 2025).
- [41] “Primary and Rechargeable Battery Chemistries with Energy Density,” *www.epectec.com*. <https://www.epectec.com/batteries/chemistry/>
- [42] Y. E. Durmus *et al.*, “Side by Side Battery Technologies with Lithium-Ion Based Batteries,” *Advanced Energy Materials*, vol. 10, no. 24, p. 2000089, May 2020, doi: <https://doi.org/10.1002/aenm.202000089>.
- [43] K. Katona, H. A. Neamah, and P. Korondi, “Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot,” *Sensors*, vol. 24, no. 11, p. 3573, Jan. 2024, doi: <https://doi.org/10.3390/s24113573>.
- [44] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, “Path Planning for Autonomous Mobile Robots: A Review,” *Sensors*, vol. 21, no. 23, p. 7898, Nov. 2021, doi: <https://doi.org/10.3390/s21237898>.
- [45] MathWorks, “What Is SLAM (Simultaneous Localization and Mapping) – MATLAB & Simulink,” *www.mathworks.com*. <https://www.mathworks.com/discovery/slam.html>

- [46] Z.-X. Li, G. Cui, C. Li, and Z. Zhang, “Comparative Study of Slam Algorithms for Mobile Robots in Complex Environment,” Oct. 2021, doi: <https://doi.org/10.1109/crc52766.2021.9620122>.
- [47] A. Dewan, A. Kumar, H. Singh, V. S. Solanki, and P. Kaur, “Advancement in SLAM Techniques and Their Diverse Applications,” Dec. 2023, doi: <https://doi.org/10.1109/smart59791.2023.10428583>.
- [48] Basheer Al-Tawil, T. Hempel, A. Abdelrahman, and Ayoub Al-Hamadi, “A review of visual SLAM for robotics: evolution, properties, and future applications,” *Frontiers in robotics and AI*, vol. 11, Apr. 2024, doi: <https://doi.org/10.3389/frobt.2024.1347985>.
- [49] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, “Visual SLAM: What Are the Current Trends and What to Expect?,” *Sensors*, vol. 22, no. 23, p. 9297, Nov. 2022, doi: <https://doi.org/10.3390/s22239297>.
- [50] “Understanding how V-SLAM (Visual SLAM) works,” Kudan global, Aug. 20, 2020. <https://www.kudan.io/blog/visual-slam-the-basics/>
- [51] Y. Cai, Y. Ou, and T. Qin, “Improving SLAM Techniques with Integrated Multi-Sensor Fusion for 3D Reconstruction,” *Sensors*, vol. 24, no. 7, pp. 2033–2033, Mar. 2024, doi: <https://doi.org/10.3390/s24072033>.
- [52] Y. Liu and Y. Tan, “A Review of Visual SLAM Systems Based on Multi-Sensor Fusion,” 2024 9th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), pp. 304–310, Nov. 2024, doi: <https://doi.org/10.1109/iciibms62405.2024.10792819>.
- [53] P. Mazurek and T. Hachaj, “SLAM-OR: Simultaneous Localization, Mapping and Object Recognition Using Video Sensors Data in Open Environments from the Sparse Points Cloud,” *Sensors*, vol. 21, no. 14, p. 4734, Jul. 2021, doi: <https://doi.org/10.3390/s21144734>.
- [54] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015, doi: <https://doi.org/10.1109/tro.2015.2463671>.
- [55] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” *IEEE Xplore*, Nov. 01, 2011. <https://ieeexplore.ieee.org/document/6126544>
- [56] “Mapping & Navigation using RTAB-Map — AgileX LIMO Documentation,” Trossenrobotics.com, 2022. https://docs.trossenrobotics.com/agilex_limo_docs/demos/slam_nav/rtabmap.html (accessed Mar. 20, 2025).

- [57] I. Al-Darraj et al., “A Technical Framework for Selection of Autonomous UAV Navigation Technologies and Sensors,” *Computers, Materials & Continua*, vol. 68, no. 2, pp. 2771–2790, 2021, doi: <https://doi.org/10.32604/cmc.2021.017236>.
- [58] UZH Robotics and Perception Group, “Ultimate SLAM? Combining Events, Images, and IMU for Visual SLAM in HDR and High-Speed Scenarios,” YouTube, Mar. 22, 2018. <https://www.youtube.com/watch?v=0hDGFFJQfmA> (accessed Mar. 21, 2025).
- [59] N. Mohamad Yatim and N. Buniyamin, “PARTICLE FILTER IN SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM) USING DIFFERENTIAL DRIVE MOBILE ROBOT,” *Jurnal Teknologi*, vol. 77, no. 20, Dec. 2015, doi: <https://doi.org/10.11113/jt.v77.6557>.
- [60] “What Is A Monocular Camera ?,” Kentfaith.co.uk, Aug. 17, 2023. https://www.kentfaith.co.uk/blog/article_what-is-a-monocular-camera_72
- [61] A. Schirn, “ISO 10218-1:2025—Robots And Robotic Devices Safety - ANSI Blog,” The ANSI Blog, Oct. 25, 2023. <https://blog.ansi.org/iso-10218-1-2025-robots-and-robotic-devices-safety/>
- [62] “ISO 10218-1:2025(en) Robotics — Safety requirements — Part 1: Industrial robots,” Iso.org, 2025. <https://www.iso.org/obp/ui/en/#iso:std:iso:10218:-1:ed-3:v1:en>
- [63] “ISO 10218-2:2025(en) Robotics — Safety requirements — Part 2: Industrial robot applications and robot cells,” Iso.org, 2025. <https://www.iso.org/obp/ui/en/#iso:std:iso:10218:-2:ed-2:v1:en> (accessed Mar. 21, 2025).
- [64] M. Labbé and F. Michaud, “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, Oct. 2018, doi: <https://doi.org/10.1002/rob.21831>.
- [65] A. Macario Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel, “A Comprehensive Survey of Visual SLAM Algorithms,” *Robotics*, vol. 11, no. 1, p. 24, Feb. 2022, doi: <https://doi.org/10.3390/robotics11010024>.
- [66] X. Zhang, H. Zhang, C. Qian, B. Li, and Y. Cao, “A LiDAR-intensity SLAM and loop closure detection method using an intensity cylindrical-projection shape context descriptor,” *International Journal of Applied Earth Observation and Geoinformation*, vol. 122, p. 103419, Jul. 2023, doi: <https://doi.org/10.1016/j.jag.2023.103419>.
- [67] “1. Bare Metal User Guide.” *Intel*, www.intel.com/content/www/us/en/docs/programmable/683211/current/bare-metal-user-guide.html. Accessed 24 Mar. 2025.
- [68] *Build Hat - Raspberry Pi Documentation*, www.raspberrypi.com/documentation/accessories/build-hat.html. Accessed 24 Mar. 2025.

[69] *Buy an RP2350 – Raspberry Pi*, www.raspberrypi.com/products/rp2350/. Accessed 24 Mar. 2025.

[70] Deland-Han. “Ports That Are Used by RDS - Windows Server.” *Ports That Are Used by RDS - Windows Server | Microsoft Learn*, learn.microsoft.com/en-us/troubleshoot/windows-server/remote/ports-used-by-rds. Accessed 24 Mar. 2025.

[71] “Introduction - About Scapy.” *Introduction - Scapy 2.6.1 Documentation*, scapy.readthedocs.io/en/latest/introduction.html#quick-demo. Accessed 24 Mar. 2025.

[72] lynnlynn 11133 bronze badges, et al. “Why Can’t One Use a Raspberry Pi (or Any Other Computer) as a Microcontroller?” *Electrical Engineering Stack Exchange*, 1 June 1966, electronics.stackexchange.com/questions/582172/why-cant-one-use-a-raspberry-pi-or-any-other-computer-as-a-microcontroller.

[73] “Network Load Balancing: Round Robin vs Least Connections.” *GeeksforGeeks*, GeeksforGeeks, 5 July 2024, www.geeksforgeeks.org/network-load-balancing-round-robin-vs-least-connections/.

[74] “Networking and Data.” *Flutter*, docs.flutter.dev/get-started/fundamentals/networking. Accessed 24 Mar. 2025.

[75] “PSUTIL Documentation.” *Psutil Documentation - Psutil 7.0.0 Documentation*, psutil.readthedocs.io/en/latest/. Accessed 24 Mar. 2025.

[76] *Raspberry Pi Documentation - Microcontrollers*, www.raspberrypi.com/documentation/microcontrollers/pico-series.html. Accessed 24 Mar. 2025.

[77] “Raspberry Pi Real-Time OS (RTOS): Which to Choose.” *All3DP*, 25 Feb. 2025, all3dp.com/2/rtos-raspberry-pi-real-time-os/.

[78] *Remote Access - Raspberry Pi Documentation*, www.raspberrypi.com/documentation/computers/remote-access.html. Accessed 24 Mar. 2025.

[79] “Scapy.” *Wikipedia*, Wikimedia Foundation, 5 Nov. 2024, en.wikipedia.org/wiki/Scapy.

[80] “Selective Acknowledgments (SACK) in TCP.” *GeeksforGeeks*, GeeksforGeeks, 8 Aug. 2022, www.geeksforgeeks.org/selective-acknowledgments-sack-in-tcp/.

[81] Shopify. “Shopify/Toxiproxy: :Alarm_clock: A TCP Proxy to Simulate Network and System Conditions for Chaos and Resiliency Testing.” *GitHub*, github.com/Shopify/toxiproxy. Accessed 24 Mar. 2025.

- [82] *Silicon - Raspberry Pi Documentation*, www.raspberrypi.com/documentation/microcontrollers/silicon.html. Accessed 24 Mar. 2025.
- [83] Silveira, Otávio Simões. “A Beginner’s Guide to Unit Tests in Python (2023).” *Dataquest*, 6 Mar. 2023, www.dataquest.io/blog/unit-tests-python/.
- [84] Team, Digiteum. “Benefits of Raspberry Pi and AWS IOT Connection.” *Digiteum*, Digiteum, 24 May 2024, www.digiteum.com/connect-raspberry-pi-aws-iot-platform/.
- [85] “What Is Test Driven Development (TDD)?” *GeeksforGeeks*, GeeksforGeeks, 19 June 2024, www.geeksforgeeks.org/test-driven-development-tdd/.
- [86] “The Practical Test Pyramid.” *Martinfowler.Com*, martinfowler.com/articles/practical-test-pyramid.html. Accessed 24 Mar. 2025.
- [87] Wavelength Electronics, “PHOTODIODE BASICS – Wavelength Electronics,” *Teamwavelength.com*, Jan. 13, 2020. <https://www.teamwavelength.com/photodiode-basics/>
- [88] “OpenCV: Introduction,” *Opencv.org*, 2025. <https://docs.opencv.org/4.11.0/d1/dfb/intro.html> (accessed Apr. 03, 2025).
- [89] Wikipedia Contributors, “ISO/IEC 7810,” *Wikipedia*, Jan. 31, 2025.
- [90] “VisionWorks 1.5.3,” *NVIDIA Developer*, 2025. <https://developer.nvidia.com/embedded/visionworks-1-5-3> (accessed Apr. 03, 2025).
- [91] F. Alvi, “Computer Vision and Image Processing: Understanding the Distinction and Interconnection,” *OpenCV*, Dec. 13, 2023. <https://opencv.org/blog/computer-vision-and-image-processing/>
- [92] “Flutter on Mobile,” *flutter.dev*. <https://flutter.dev/multi-platform/mobile>
- [93] D. Zou and M. Y. Darus, “A Comparative Analysis of Cross-Platform Mobile Development Frameworks,” vol. 05, pp. 84–90, Aug. 2024, doi: <https://doi.org/10.1109/isci62787.2024.10667693>.
- [94] M. Grusin, “Serial Peripheral Interface (SPI) - learn.sparkfun.com,” *Sparkfun.com*, 2019. <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>
- [95] R. Teja, “How to Write Data to RFID Card using RC522 RFID and Arduino?,” *ElectronicsHub USA*, Mar. 03, 2021. <https://www.electronicshub.org/write-data-to-rfid-card-using-rc522-rfid/>
- [96] A. Balboa, “miguelbalboa/rfid,” *GitHub*, May 31, 2020. <https://github.com/miguelbalboa/rfid>

[97] S. Pokuciński and D. Mrozek, “Object Detection with YOLOv5 in Indoor Equirectangular Panoramas,” *Procedia Computer Science*, vol. 225, pp. 2420–2428, Dec. 2023, doi: <https://doi.org/10.1016/j.procs.2023.10.233>.

[98] Ultralytics, “Quickstart,” Ultralytics.com, 2023.
<https://docs.ultralytics.com/quickstart/#conda-docker-image> (accessed Apr. 22, 2025).

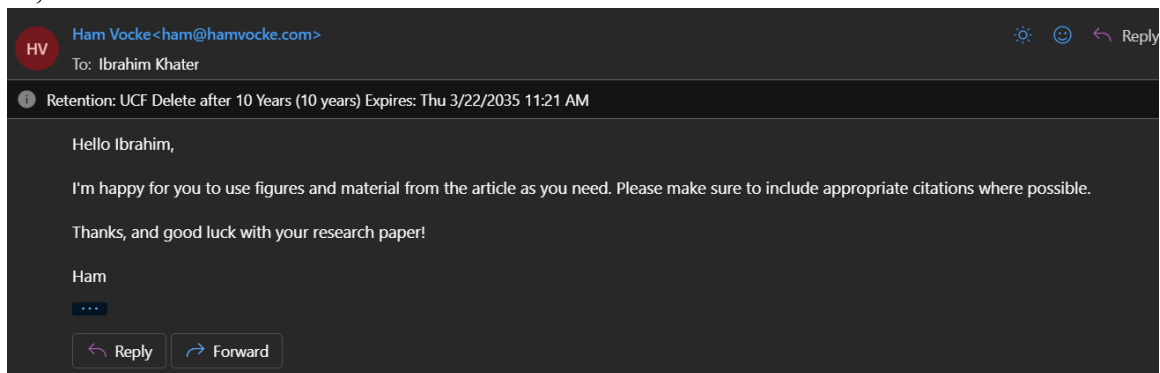
[99] P. Ogren and N. E. Leonard, “A convergent dynamic window approach to obstacle avoidance,” *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, Apr. 2005, doi: <https://doi.org/10.1109/tro.2004.838008>.

[100] M. Kobayashi, H. Zushi, T. Nakamura, and N. Motoi, “Local Path Planning: Dynamic Window Approach With Q-Learning Considering Congestion Environments for Mobile Robot,” *IEEE Access*, vol. 11, pp. 96733–96742, 2023, doi: <https://doi.org/10.1109/access.2023.3311023>.

[101] “asr_navfn - ROS Wiki,” Ros.org, 2024. https://wiki.ros.org/asr_navfn (accessed Apr. 22, 2025).

Appendix B - Copyright Permission

[1] Permission to use the images was granted by author Ham Vocke via email on March 24, 2025.



Appendix C - Data Sheet

[1] TOF Distance Sensor Wiring, Tiny Circuits, Accessed: Feb 27, 2025 [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/423/AST1024_Web.pdf.

[2] Gravity: VL53L0X ToF Laser Range Finder, DFRobot, Accessed: Feb 27, 2025 [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/827/SEN0245_Web.pdf.

- [3] Pmod ToF Reference Manual, Digilent, Accessed: Feb 27, 2025 [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/206/Pmod_ToF_RM_Web.pdf
- [4] NXP founded by Phillips, “MFRC522 Contactless Reader IC”, 112132 datasheet, May 2007. Accessed: Mar 11, 2025 [Online]. Available: <https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/60/CN0090%20DATASHEET.pdf>
- [5] Reyax Technology Corporation, “RYRR10S Multiprotocol Fully Integrated 13.56MHz RFID & NFC Antenna Module”, 56312E33 datasheet, July 2022. Accessed Mar 11, 2025 [Online]. Available: https://reyax.com/upload/products_download/download_file/RYRR10S_EN.pdf
- [6] Reyax Technology Corporation, “RYRR20I_DE Multiprotocol Fully Integrated 13.56MHz +3.3V UART Interface RFID Antenna Module”, May 2023. Accessed Mar 11, 2025 [Online]. Available: https://reyax.com/upload/products_download/download_file/RYRR20I_DE.pdf
- [7] Excel Technology, “Pepper C1 Module User Manual”, Sep. 2021. Accessed Mar 11, 2025 [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/5341/Pepper_C1_Module.pdf
- [8] Espressif Systems, “ESP32-WROOM-32,” v3.4, Feb. 2023 https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
- [9] STMicroelectronics, “Medium-density performance line Arm®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. Interfaces,” DS5319, Sep. 2023, <https://www.st.com/resource/en/datasheet/stm32f103cb.pdf>
- [10] Microchip Technology, “ATmega48A/PA/88A/PA/168A/PA/328/P Data Sheet,” DS40002061B, 2020, <https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-P-A-328-P-DS-DS40002061B.pdf>
- [11] Microchip Technology, “PIC18(L)F2X/4XK22 Data Sheet,” DS40001412H, 2021, [https://ww1.microchip.com/downloads/en/DeviceDoc/PIC18\(L\)F2X-4XK22-Data-Sheet-40001412H.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/PIC18(L)F2X-4XK22-Data-Sheet-40001412H.pdf)
- [12] RobotZone, “23 rpm HD Premium Planetary Gear Motor Spec Sheet,” https://www.servocity.com/content/downloads/23_rpm_hd_precision_planetary_gearmotor_-_specifications.pdf

- [13] Toshiba, “TB9051FTG,” Mar. 2019, https://toshiba.semicon-storage.com/info/TB9051FTG_datasheet_en_20190206.pdf?did=30232&prodName=TB9051FTG
- [14] STMicroelectronics, “Automotive fully integrated H-bridge motor driver,” DocID1570, June 2017, <https://www.st.com/content/ccc/resource/technical/document/datasheet/1c/26/60/a8/3e/70/4c/e1/CD00234623.pdf/files/CD00234623.pdf/jcr:content/translations/en.CD00234623.pdf>
- [15] Texas Instruments, “DRV8871 3.6-A Brushed DC Motor Driver With Internal Current Sense (PWM Control),” SLVSCY9B, Aug. 2015, <https://www.ti.com/general/docs/suppproductinfo.tsp?distId=10&gotoUrl=https%3A%2F%2Fwww.ti.com%2Flit%2Fgpn%2Fdrv8871>
- [16] NXP, “MC33926 5.0 A throttle control H-bridge,” Feb. 2020, <https://www.nxp.com/docs/en/data-sheet/MC33926.pdf>
- [17]Raspberry Pi ltd, “Raspberry Pi 4 Computer Model B,” 2021. Available: <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf>
- [18]RaspberryPi5,2023.Available:
<https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf>
- [19]M. Gpu, “DATA SHEET NVIDIA Jetson Nano System-on-Module Maxwell GPU + ARM Cortex-A57 + 4GB LPDDR4 + 16GB eMMC.” Accessed: Mar. 20, 2025. [Online]. Available:
https://openzeka.com/en/wp-content/uploads/2022/08/JetsonNano_DataSheet_DS09366001v1.1.pdf?srsId=AfmBOorifd2SpQMpmeGtQsGHaR85nKlhOtCZ8nHbLjnW_NwBY-iXB1Zf
- [20]“Arduino®MEGA2560Rev3.”Available:
<https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf>
- [21]Raspberry Pi, “Raspberry Pi Documentation - Camera,” [www.raspberrypi.com](https://www.raspberrypi.com/documentation/accessories/camera.html).
<https://www.raspberrypi.com/documentation/accessories/camera.html>
- [22]“Arducam Datasheet 1080P Day and Night Vision USB Camera Module B0490.” Accessed: Mar. 21, 2025. [Online]. Available:
https://www.arducam.com/downloads/datasheet/B0490_1080P_IMX462_USB_Camera_Datasheet.pdf
- [23] “Arducam Datasheet 12MP Wide Angle USB Camera Module B0433.” Accessed: Mar. 21, 2025. [Online]. Available:
https://www.uctronics.com/download/Amazon/B0433_Ardcam_12MP_USB_Camera

[Module_Datasheet.pdf?srltid=AfmBOor6lPabu0S69Z8xULGacIg21lWzubFleCtFyyKaf24OAn_fYu3e](#)

[24] “MPU-6000 and MPU-6050 Product Specification Revision 3.3 MPU-6000/MPU-6050 Product Specification.” Available: <https://www.haoyuelectronics.com/Attachment/GY-521/mpu6050.pdf>

[25] “MPU-9250 Product Specification Revision 1.1 MPU-9250 Product Specification.” Available: <https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>

[26] UMW, “AMS1117,” Available: <https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/5011/AMS1117.pdf>

[27] UMW, “UMW LM2596/LM2596HV,” Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2577/UMW%20LM2596_HV.pdf

[28] Texas Instruments, “TPS56x200 4.5-V to 17-V Input, 2-A, 3-A Synchronous Step-down Voltage Regulator in 6-Pin SOT-23,” SLVSCB0E, May 2023, Available: <https://www.ti.com/lit/ds/slvscb0e/slvscb0e.pdf?ts=1742825277552>

[29] Monolithic Power Systems Inc., “MP2315S,” Dec. 2015, Available: https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP2315S/document_id/981

[30] Texas Instruments, “LM317 3-Terminal Adjustable Regulator ,” SLVS044Y, Sep. 1997, Available: <https://www.ti.com/lit/gpn/lm317>

[31] Texas Instruments, “TPS56624x 3-V to 16-V Input Voltage, 6-A Synchronous Buck Converter in a SOT-563 Package ,” SLUSEN0A, Feb. 2022, Available: <https://www.ti.com/lit/ds/symlink/tps566242.pdf>

[32] Texas Instruments, “LM2576xx Series SIMPLE SWITCHER® Power Converter 3-A Step-Down Voltage Regulator,” SNVS107G, June 1999, Available: <https://www.ti.com/lit/ds/symlink/lm2576hv.pdf>

[33] Diodes Incorporated, “AP1501A,” DS31191, Feb. 2020, Available: https://www.diodes.com/assets/Datasheets/products_inactive_data/AP1501A.pdf

[34] Monolithic Power Systems Inc., “MP1584,” Aug. 2011, Available: https://www.monolithicpower.com/en/documentview/productdocument/index/version/2/document_type/Datasheet/lang/en/sku/MP1584EN-LF-Z/document_id/204/

[35] Texas Instruments, “LMR33630 SIMPLE SWITCHER® 3.8-V to 36-V, 3-A Synchronous Step-down Voltage Converter,” SNVSAN3F, Aug. 2017, Available: <https://www.ti.com/lit/gpn/lmr33630>

[36] TDK InvenSense, “MPU-6000 and MPU-6050 Product Specification,” PS-MPU-6000A-00, Aug. 2013, Available: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>

[37] TDK InvenSense, “High Precision 6-Axis MEMS MotionTracking™ Device,” , DS-000347, July 2023, Available: https://invensense.tdk.com/wp-content/uploads/2020/04/ds-000347_icm-42688-p-datasheet.pdf

[38] STMicroelectronics, “LSM6DSO,” DS12140, June 2024, Available: <https://www.st.com/resource/en/datasheet/lsm6dso.pdf>

Appendix D - Software Code

D.1 RFID Read/Write and Print to Serial Monitor

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 5
#define RST_PIN 21

MFRC522 mfrc522(SS_PIN, RST_PIN);

byte block = 8;
byte readData[18];
char mode = ' ';
String userInput = "";
bool waitingForInput = false;

// Default key for MIFARE cards (0xFF 0xFF 0xFF 0xFF 0xFF 0xFF)
MFRC522::MIFARE_Key key;

void cleanup() {
    mfrc522.PICC_HaltA();
    mfrc522.PCD_StopCrypto1();
    mode = ' ';
    userInput = "";
    waitingForInput = false;
    Serial.println("\nDone. Type 'r' or 'w' again:");
    delay(1000);
}

void setup() {
    Serial.begin(9600);
    while (!Serial);
```

```

delay(2000); // delay before set up

SPI.begin();
mfrc522.PCD_Init();

for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

Serial.println("RFID Ready");
Serial.println("Type 'r' to read or 'w' to write:");
}

void writeToTag(String message) {
    byte dataBlock[16];

    // Fill array with message chars and null terminator
    for (int i = 0; i < 16; i++) {
        if (i < message.length()) {
            dataBlock[i] = message[i];
        } else if (i == message.length()) {
            dataBlock[i] = '\0'; // null terminator
        } else {
            dataBlock[i] = 0;      // fill rest with zeros
        }
    }

    MFRC522::StatusCode status = mfrc522.MIFARE_Write(block, dataBlock,
16);
    if (status != MFRC522::STATUS_OK) {
        Serial.print("Write failed: ");
        Serial.println(mfrc522.GetStatusCodeName(status));
    } else {
        Serial.println("Write successful!");
    }
}

void readFromTag() {
    MFRC522::StatusCode status;
    byte size = 18;

    status = mfrc522.MIFARE_Read(block, readData, &size);
    if (status != MFRC522::STATUS_OK) {
        Serial.print("Read failed: ");
        Serial.println(mfrc522.GetStatusCodeName(status));
    } else {
        Serial.print("Data read: ");
        Serial.println((char*)readData); // Stops at null terminator
    }
}

void loop() {
    if (Serial.available()) {
        String input = Serial.readStringUntil('\n');
        input.trim();

        if (mode == 'w' && waitingForInput) {
            userInput = input;

```

```

        waitingForInput = false;
        Serial.print("You entered: ");
        Serial.println(userInput);
        Serial.println("Now scan the tag...");
    } else if (input.equalsIgnoreCase("w")) {
        mode = 'w';
        Serial.println("Write mode selected. Enter message to write (max 15
characters):");
        waitingForInput = true;
    } else if (input.equalsIgnoreCase("r")) {
        mode = 'r';
        Serial.println("Read mode selected. Scan your tag...");
    } else {
        Serial.println("Invalid input. Type 'r' or 'w':");
    }
}

if ((mode == 'r') || (mode == 'w' && !waitingForInput &&
userInput.length() > 0)) {
    if (!mfrc522.PICC_IsNewCardPresent()) return;
    if (!mfrc522.PICC_ReadCardSerial()) return;

    MFRC522::StatusCode status;
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
block, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        Serial.print("Authentication failed: ");
        Serial.println(mfrc522.GetStatusCodeName(status));
        cleanup();
        return;
    }

    if (mode == 'w') {
        writeToTag(userInput);
    } else if (mode == 'r') {
        readFromTag();
    }

    cleanup();
}
}

```

D.2 Raspberry Pi Test Code for Communicating with ESP32

```

#!/usr/bin/env python3

import serial
import time

if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    ser.reset_input_buffer()
    while True:
        ser.write(b'1 ')

```

```

        line = ser.readline().decode('utf-8').rstrip()
        print(line)
        time.sleep(1)
        ser.write(b'0 ')
        line = ser.readline().decode('utf-8').rstrip()
        print(line)
        time.sleep(1)

```

D.3 ESP32 Test Code for Communicating with Raspberry Pi

```

#define outputPin 2

void setup() {
    Serial.begin(9600);
    pinMode (outputPin, OUTPUT);
    digitalWrite(outputPin, LOW);

    if (Serial.available() > 0)
    {
        Serial.println("ESP32: hello!");
    }
}

void loop() {
    if (Serial.available() > 0) {

        String data = Serial.readStringUntil(' ');

        if(data == "1")
        {
            digitalWrite(outputPin, HIGH);
        }
        if (data == "0")
        {
            digitalWrite(outputPin, LOW);
        }

    }
}

```

D.4 TOF Sensor Test Code [C.1]

```

/*****
**
* Time of Flight Distance Sensor Wiring Example
* This example shows how to use continuous mode to take, and print
* range measurements with the VL53L0X Wiring. It is based on
* vl53l0x_ContinuousRanging_Example.c from the VL53L0X API.
*
* The range readings are in units of mm.
*
* Hardware by: TinyCircuits
* Modified by: Laveréna Wienclaw for TinyCircuits

```

```

* Last Modified: 12/18/19

*****
*/

#include <Wire.h>          // For I2C communication
#include "VL53L0X.h"      // For interfacing with the Time-of-Flight
Distance sensor
#include <Wireling.h>     // For interfacing with Wirelings

VL53L0X distanceSensor; // Name of sensor
const int tofPort = 0;   // Port # of sensor (Found on Wireling Adapter
Board)

const int averageCount = 1;
int average[averageCount];
int averagePos = 0;

#if defined(ARDUINO_ARCH_AVR)
#define SerialMonitorInterface Serial
#elif defined(ARDUINO_ARCH_SAMD)
#define SerialMonitorInterface SerialUSB
#endif

void setup() {
    delay(200);                // Sensor Startup time
    SerialMonitorInterface.begin(115200); // Set Baud Rate
    Wire.begin();              // Begin I2C communication

    // Enable power & select port
    Wireling.begin();
    // Select the port of the distance sensor (this number corresponds
    // with port #'s on the Wireling adapter board)
    Wireling.selectPort(tofPort);

    // Initialize the distance sensor and set a timeout
    distanceSensor.init();
    distanceSensor.setTimeout(500);

    // Start continuous back-to-back mode (take readings as
    // fast as possible). To use continuous timed mode
    // instead, provide a desired inter-measurement period in
    // ms (e.g. sensor.startContinuous(100))
    distanceSensor.setMeasurementTimingBudget(200000);
    distanceSensor.startContinuous();
}

void loop() {
    // Calculate the average position of the distance sensor
    unsigned long averageRead = 0;
    average[averagePos] = distanceSensor.readRangeContinuousMillimeters();
    averagePos++;
    if (averagePos >= averageCount) averagePos = 0;
    for (int i = 0; i < averageCount; i++) {
        averageRead += (unsigned long)average[i];
    }
}

```

```
averageRead /= (unsigned long)averageCount;

// Print the average position to the TinyScreen and the Serial Monitor
SerialMonitorInterface.print (averageRead);
SerialMonitorInterface.println("mm");
}
```

Appendix E – ChatGPT Prompts and Outcomes

E.1 Explain Paragraph on Inductive Coupling ChatGPT Prompt and Outcome

Prompt: “Explain this: The interrogator communicates with the tag by modulating a carrier wave by varying the amplitude, phase, or frequency of the carrier. This modulation can be directly detected as current changes in the coil of the tag. The tag communicates with the interrogator by varying how much it loads its antenna. This in turn affects the voltage across the interrogator’s antenna. By switching the load on and off rapidly, the tag can create sideband frequencies, which are then coupled into the interrogator antenna.”

Outcome:

“This passage describes how communication happens between an RFID tag and an RFID reader (interrogator) using electromagnetic waves.

Breaking it Down:

1. Interrogator to Tag Communication (Forward Link)
 - The interrogator (reader) sends a signal to the tag using a carrier wave (a high-frequency electromagnetic wave).
 - To encode information, the interrogator modulates (changes) the carrier wave by adjusting one of its properties:
 - Amplitude Modulation (AM) – changing the strength of the wave.
 - Frequency Modulation (FM) – changing the wave’s frequency.
 - Phase Modulation (PM) – shifting the phase of the wave.
 - The tag receives this modulated signal, and the changes appear as current variations in its coil (for passive tags that use inductive coupling).
2. Tag to Interrogator Communication (Backscatter Link)
 - The tag does not generate its own signal; instead, it modifies the way it interacts with the reader’s signal.
 - It does this by changing how much it loads its antenna, meaning it alternates between absorbing more or less energy from the interrogator’s signal.
 - This change in load affects the voltage on the reader’s antenna, creating a detectable pattern.
 - If the tag switches its load on and off rapidly, it produces sideband frequencies (additional frequency components) that can be detected by the reader.

Why is This Important?

This method, known as backscatter communication, allows passive RFID tags (which have no battery) to communicate by reflecting and modulating the incoming signal. Since the tag doesn't need to generate its own radio signal, it consumes very little power and can operate using the energy received from the reader. Would you like a more in-depth explanation with diagrams or examples?"

E.2 Explain Paragraph on Inductive Coupling DeepAI Prompt and Outcome

Prompt: "Explain this: The interrogator communicates with the tag by modulating a carrier wave by varying the amplitude, phase, or frequency of the carrier. This modulation can be directly detected as current changes in the coil of the tag. The tag communicates with the interrogator by varying how much it loads its antenna. This in turn affects the voltage across the interrogator's antenna. By switching the load on and off rapidly, the tag can create sideband frequencies, which are then coupled into the interrogator antenna."

Outcome:

"This explanation describes the basic principles of how communication occurs between an interrogator (also known as a reader) and a passive RFID tag.

1. Interrogator and Carrier Wave:

- The interrogator emits a continuous carrier wave, which is a high-frequency signal that can be modulated to convey information. This modulation can happen in three primary ways:
 - Amplitude Modulation (AM): The strength (amplitude) of the carrier wave is varied.
 - Phase Modulation (PM): The phase (the position of the wave) of the carrier signal is altered.
 - Frequency Modulation (FM): The frequency (the rate at which the wave oscillates) of the carrier wave is changed.

2. Detection by the Tag:

- The RFID tag, which typically has a coil and other electronic components, detects these modulated changes in the carrier wave. As the interrogator's signal interacts with the tag's coil, the amplitude, phase, or frequency variations induce an alternating current in the tag's coil. This induced current powers the tag and allows it to operate.

3. Tag Communication Back to Interrogator:

- The tag communicates back to the interrogator by varying the load on its antenna. This is essentially changing how much power the tag draws from the interrogator's field.
- When the tag changes its load, it alters the amount of energy that is reflected back to the interrogator, which in turn affects the voltage observed at the interrogator's antenna.

4. Creating Sideband Frequencies:

- By quickly switching its load on and off, the tag produces changes in the reflected signal that can be detected as sideband frequencies (additional frequencies outside the main carrier signal).
- These sideband signals contain the information the tag wishes to transmit back to the interrogator. The interrogator picks up these variations in the voltage and, through signal processing, decodes the information transmitted by the tag.

Overall, this process allows for two-way communication between the interrogator and the tag without requiring the tag to have its own power source, as it relies on the energy from the interrogator's carrier wave."

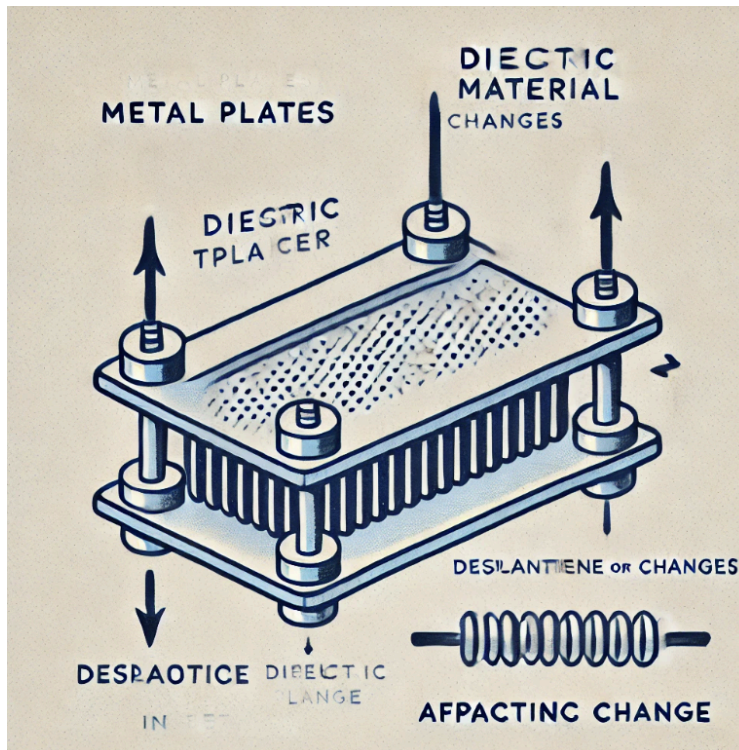
E.3 Explain Capacitive Transducers ChatGPT Prompt and Outcome

Prompt: "Explain how capacitive transducer's work in a few sentences."

Outcome: "A capacitive transducer works by converting physical changes, such as displacement, pressure, or humidity, into variations in capacitance. It consists of two conductive plates separated by a dielectric material, where the capacitance is given by $C = \epsilon A/d$ (ϵ = permittivity, A = plate area, d = separation distance). When an external force changes the plate spacing, dielectric constant, or overlapping area, the capacitance changes, which can then be measured and converted into an electrical signal."

Prompt: "draw a picture"

Outcome:



E.4 Explain Capacitive Transducers DeepAI Prompt and Outcome

Prompt: "Explain how capacitive transducer's work in a few sentences."

Outcome: "Capacitive transducers work by measuring changes in capacitance due to variations in physical parameters such as displacement, pressure, or temperature. They consist of two conductive plates separated by a dielectric material; when an external force alters the distance between the plates or the properties of the dielectric, the capacitance changes. This change in capacitance can be converted into an electrical signal, allowing the transducer to measure the physical quantity accurately. The output can then be processed for various applications in sensing and instrumentation."

Prompt: "draw a picture"

Outcome:

"I'm unable to draw pictures directly, but I can describe how you can visualize a capacitive transducer.

1. Basic Structure: Imagine two parallel plates, labeled as "Plate A" and "Plate B." These plates are separated by a small distance (let's say d). They are typically rectangular or circular in shape.
2. Dielectric Material: In between the two plates, you can represent a dielectric material (which could be air or another insulator). You can label it "Dielectric."

3. Capacitance Representation: Next to the plates, you can write the formula for capacitance:

$$C = \frac{\epsilon A}{d}$$

where C is capacitance, ϵ is the permittivity of the dielectric material, A is the area of one of the plates, and d is the distance between the plates.

4. Illustrate Changes: You can show arrows indicating that if the plates move closer together or further apart (indicating changes in external forces), the distance d changes, which affects the capacitance C .

5. Electrical Connection: On one side of the plates, you can illustrate wires leading to an electrical circuit, representing that the change in capacitance leads to a change in the electrical signal that can be measured.

This simple diagram effectively demonstrates how capacitive transducers operate. If you have drawing tools or software, you can create such an illustration based on this description!”

E.5 Ideas for Ethical Constraints

Prompt: “Here is some information about my senior design project: This project’s focus is to build a robot to aid in transporting payloads within a warehouse or other industrial environment. The robot is able to safely transport payloads from one location to another and return to its home location while avoiding obstacles. Design a mobile application for the user to view the robot’s status and indicate target location. The robot is able to detect if a payload is on top of it and identify the payload correctly. The robot is able to create a new path to the target location if the original path is blocked. To achieve the above goals, our team has certain objectives for the robot in order to perform as expected. Our robot will include a photoresistor to detect the presence of a payload on top of the robot, ensuring that the robot can confirm when an item is loaded or removed. A short range HF RFID sensor will be used to identify specific payloads, allowing the system to differentiate and track payloads effectively. To navigate safely, the robot will use an array of distance sensors to detect objects in its path and reroute its path accordingly. The robot’s navigation system will be powered by an API/mapping toolbox running on an onboard Raspberry Pi. Additionally, a mobile application will allow the user to select target delivery locations and offer an intuitive interface for interaction and monitoring. The robot with the features and functionalities described in the following section will achieve our main goals of transporting payloads. Given what this project is about, what are some ethical constraints or ethical impacts of a project like this.”

Outcome:

“There are several ethical considerations and potential impacts for a warehouse payload-transporting robot like yours. Here are a few key ones:

1. **Worker Displacement & Job Loss**
 - Automating warehouse transportation could reduce the need for human workers, potentially leading to job displacement.
 - Ethical responsibility: Companies should consider retraining or upskilling workers to operate, maintain, or oversee the robots rather than replacing them entirely.
2. **Safety & Liability**
 - If the robot malfunctions or misidentifies a payload, it could cause delays or financial losses.
 - More critically, if the robot collides with a human or another machine, it could cause injuries or damage.
 - Ethical responsibility: Ensure rigorous testing, implement fail-safes, and establish clear liability policies.
3. **Data Privacy & Security**
 - If the mobile app collects user data, there should be safeguards to protect sensitive information.
 - RFID tracking raises concerns about unauthorized scanning or tracking of payloads.
 - Ethical responsibility: Secure communication between the robot, mobile app, and cloud storage, and limit data collection to only what is necessary.
4. **Fairness & Accessibility**
 - Small businesses or lower-income warehouses might struggle to afford such automation, leading to an uneven playing field in the industry.
 - Ethical responsibility: Consider cost-effective solutions or models that make automation more accessible.
5. **Environmental Impact**
 - If not designed efficiently, the robot could contribute to excessive electronic waste or energy consumption.
 - Ethical responsibility: Use sustainable materials, optimize power consumption, and explore recycling options for components.
6. **Reliability & Bias in Decision-Making**
 - If the navigation system has flaws, it might prioritize certain paths over others, causing inefficiencies.
 - Ethical responsibility: Ensure that dynamic path planning does not create unintended biases (e.g., favoring one route type over another for no justified reason).

Would you like me to expand on any of these or help integrate them into your technical document?”

E.6 Which raspberry pi model would be recommended?

Prompt: “Which raspberry pi model would be recommended?”

Outcome:

If you're looking for a Raspberry Pi model to handle SLAM and wireless communication efficiently, here are your best options ranked by performance:

1. Raspberry Pi 5 (Best Choice - If Available)

Pros:

- CPU: Quad-core Cortex-A76 @ 2.4 GHz (Much faster than Pi 4B)
- RAM : Up to 8GB LPDDR4X
- PCIe 2.0 Interface : Can connect a fast external SSD for better data processing
- Improved GPU & I/O Speeds
- Better heat management

Cons:

- Still not powerful enough for complex real-time SLAM with high-end LiDAR
- May require active cooling if running heavy processing

Best For: Lightweight SLAM (e.g., ORB-SLAM2, Cartographer) + Mobile App Connectivity

2. Raspberry Pi 4B (Good for Light SLAM, but Limited)

Pros:

- Decent performance for 2D SLAM (low-resolution LiDAR) or visual SLAM with low-resolution cameras
- Supports WiFi/Bluetooth for mobile app communication
- Runs ROS (Robot Operating System)

Cons:

- CPU struggles with real-time SLAM on large maps
- No dedicated AI acceleration
- Limited PCIe connectivity

Best For: Pre-mapping environments and offloading SLAM processing elsewhere (e.g., cloud, Jetson Nano)

3. Raspberry Pi with Coral USB Accelerator (Better AI Support)

If you want better SLAM performance, you can pair a Raspberry Pi 4B or Pi 5 with a Coral USB Accelerator (Google's Edge TPU).

Boosts ML tasks like feature extraction in ORB-SLAM

Helps with image-based localization

4. Alternative: NVIDIA Jetson Nano / Xavier NX (If You Need Serious SLAM)

If you need real-time SLAM with LiDAR & AI , Raspberry Pi models won't cut it. Instead, consider:

- Jetson Nano : Affordable, 128-core GPU, optimized for SLAM & AI vision
- Jetson Xavier NX : More expensive but handles high-end SLAM with AI-based path planning

Final Recommendation

- Lightweight SLAM & Mobile App? → Raspberry Pi 5 (Best option if available)
- Basic SLAM (2D or monocular)? → Raspberry Pi 4B (May struggle with real-time)
- Want better AI support? → Raspberry Pi 4B/5 + Coral USB Accelerator
- Need real-time SLAM with complex sensors? → Jetson Nano / Xavier NX

Would you like recommendations on SLAM libraries optimized for Raspberry Pi?

E.7 Make a comparison between Raspberry Pi 5 and other microcontrollers why is it best for this project proposal

Prompt: “Make a comparison between Raspberry Pi 5 and other microcontrollers why is it best for this project proposal”

Outcome:

Based on your project proposal for an **autonomous delivery robot**, the **Raspberry Pi 5** is a strong candidate compared to other microcontrollers and SBCs (Single Board Computers). Here's a **detailed comparison** and justification for why the Raspberry Pi 5 is the best choice.

1. Comparison of Raspberry Pi 5 vs. Other Microcontrollers/SBCs

Feature	Raspberry Pi 5	Raspberry Pi 4B	NVIDIA Jetson Nano	Arduino Mega 2560	ESP32
CPU	Quad-core Cortex-A76 @ 2.4GHz	Quad-core Cortex-A72 @ 1.5GHz	Quad-core ARM Cortex-A57 @ 1.43GHz	8-bit AVR	Dual-core Xtensa LX6 @ 240MHz
RAM	Up to 8GB LPDDR4X	Up to 8GB LPDDR4	4GB LPDDR4	8KB SRAM	520KB SRAM
GPU	VideoCore VII (Much Faster)	VideoCore VI	128-core Maxwell GPU	None	None
Power Efficiency	Good	Moderate	High power usage	Extremely low	Very low

I/O Ports	USB 3.0, PCIe, GPIO	USB 3.0, GPIO	USB 3.0, GPIO	Limited GPIO	Limited GPIO
Wireless Connectivity	WiFi 6, Bluetooth 5.0	WiFi 5, Bluetooth 5.0	No (requires adapter)	WiFi No WiFi/Bluetooth	WiFi, Bluetooth
Software Support	Linux, ROS2, OpenCV, AI models	Linux, ROS2, OpenCV	Linux, AI-based CUDA acceleration	Limited C-based programming	RTOS, C, Python
Cost	~\$60-\$80	~\$40-\$70	~\$100-\$150	~\$40	~\$10

2. Why Raspberry Pi 5 is the Best Choice for Your Project

Your project needs **SLAM-based navigation, real-time processing, and wireless connectivity** for mobile app integration. Here's why **Raspberry Pi 5 is the best option**:

A. SLAM and Route Planning Capabilities

- **Quad-Core 2.4GHz CPU:** Handles SLAM (Simultaneous Localization and Mapping) better than Pi 4.
- **8GB RAM Option:** Prevents memory bottlenecks when running real-time **LiDAR-based SLAM**.
- **PCIe Expansion:** Allows connecting **AI accelerators (like Coral TPU)** for improved object detection.
- **Better GPU:** Helps in **image processing for vision-based SLAM** using OpenCV.

Why not Raspberry Pi 4B?

- Pi 4B can handle lightweight SLAM but struggles with **real-time LiDAR processing**.
- Slower CPU means **higher latency** in route recalculations.

Why not Jetson Nano?

- Jetson Nano is great for AI-based navigation but lacks **built-in WiFi and PCIe expansion**, increasing complexity.
- Higher power consumption makes it **less battery-efficient**.

B. Wireless Communication for Mobile App

- **Built-in WiFi 6 & Bluetooth 5.0:** Ensures **fast and stable** connection to your mobile app.

- **Supports WebSockets & MQTT:** Ideal for **real-time monitoring** of the robot's location/status.
- **Can act as a Web Server:** You can host a REST API for app control.

Why not Arduino Mega or ESP32?

- These are **microcontrollers**, not full-fledged computers.
- Cannot handle SLAM, LiDAR, or complex wireless networking.

C. Power Efficiency for Battery-Powered Operation

- **Lower power draw (~5W-7W average)** compared to Jetson Nano (10W+).
- **More efficient battery use**, extending operational time without frequent recharges.

Why not Jetson Nano?

- Jetson Nano's **higher power draw requires bigger batteries**, adding weight to the robot.

D. Expandability & Future-Proofing

- **Supports ROS2 (Robot Operating System):** Ensures easy integration with SLAM, LiDAR, and other sensors.
- **PCIe Port:** Allows adding **NVMe storage** for faster processing or an **AI accelerator** (e.g., Coral TPU).
- **USB 3.0 Ports:** Faster **sensor data transfer** (e.g., LiDAR, depth cameras).

Why not Raspberry Pi 4?

- Pi 4 lacks **PCIe expansion**, limiting future upgrades.
- Pi 5 has **faster processing and GPU performance** for handling real-time tasks better.

3. Conclusion: Best Choice for Your Autonomous Delivery Robot

Raspberry Pi 5 is the best microcontroller/SBC for your project because it:

1. **Efficiently runs SLAM** algorithms for indoor navigation.
2. **Has built-in WiFi 6 & Bluetooth 5.0** for seamless app communication.
3. **Supports advanced sensor integration** (LiDAR, RFID, Ultrasonic).
4. **Consumes low power**, making it ideal for battery-powered robots.
5. **Future-proof with PCIe expansion**, allowing AI/ML acceleration if needed.

Would you like help with **choosing specific SLAM software and mobile app frameworks** for Raspberry Pi 5?

E.8 Motor Torque Calculation - DeepSeek vs ChatGPT-4o

Prompt: “There is a system that can weigh up to 20kg. It has 2 motorized wheels and one caster wheel. What motor torque is needed for each motor? ”

Outcome:

ChatGPT-4o:

To determine the motor torque required for each motorized wheel, we need to consider several factors:

Given Data:

- **Total weight of the system:** 2020 kg
- **Number of driven wheels:** 22 (motorized wheels)
- **Number of free wheels:** 11 (caster wheel)
- **Acceleration desired:** aa (assumed, can be adjusted)
- **Wheel radius:** rr (assumed, must be provided)

Step 1: Compute the Required Force

The force needed to accelerate the system is given by Newton’s second law:

$F = m \cdot a$
where:

- $m = 20 \text{ kg}$ is the system's mass,
- a is the desired acceleration.

Assuming a moderate acceleration for a mobile robot, let's use $a = 1 \text{ m/s}^2$:

$$F = 20 \times 1 = 20 \text{ N}$$

Since two wheels provide the driving force, each wheel must contribute half:

$$F_{\text{each wheel}} = \frac{F}{2} = \frac{20}{2} = 10 \text{ N}$$

Step 2: Compute the Required Torque

Torque (τ) is given by:

$$\tau = F_{\text{each wheel}} \times r$$

where:

- r is the radius of the wheel.

For example, if the wheel radius is $r = 0.1 \text{ m}$ (10 cm):

$$\tau = 10 \times 0.1 = 1 \text{ Nm}$$

Step 3: Consider Friction and Inclines (Optional)

If friction or inclines are significant, the torque requirement increases. You may want to consider:

- **Rolling resistance:** Usually a small fraction of weight.
- **Incline resistance:** If climbing slopes, additional torque is needed.

Would you like to refine this based on real-world conditions like wheel size, surface friction, or desired speed?

Prompt 2: “assume 0.5 m/s^2 acceleration and a 5cm radius”

Outcome:

Let's calculate the required torque using your updated values:

Given Data:

- **Mass of system:** $m = 20 \text{ kg}$
- **Acceleration:** $a = 0.5 \text{ m/s}^2$
- **Number of driven wheels:** 2
- **Wheel radius:** $r = 5 \text{ cm} = 0.05 \text{ m}$

Step 1: Compute Required Force

Using Newton's second law:

$$F = m \cdot a \quad F = 20 \times 0.5 = 10 \text{ N}$$

Each of the two motorized wheels shares the load:

$$F_{\text{each wheel}} = \frac{10}{2} = 5 \text{ N}$$

Step 2: Compute Required Torque

Torque is given by:

$$\tau = F_{\text{each wheel}} \times r \quad \tau = 5 \times 0.05 = 0.25 \text{ Nm}$$

Final Answer:

Each motor needs to provide **at least 0.25 Nm** of torque to achieve the desired acceleration, not accounting for losses due to friction, inclines, or inefficiencies. You may want to add a **safety margin** (e.g., 1.5x to 2x the torque) to account for these factors.

Let me know if you need further refinements! 🚀

DeepSeek:

To determine the required motor torque for each motor in your system, we need to consider several factors, including the total weight of the system, the wheel diameter, the desired acceleration, and the friction between the wheels and the ground.

Key Assumptions:

1. **Total Weight (W):** 20 kg (including the system and any payload).
2. **Number of Motorized Wheels:** 2.
3. **Caster Wheel:** 1 (assumed to provide stability but not contribute to propulsion).
4. **Wheel Diameter (D):** Let's assume a typical wheel diameter of 10 cm (0.1 m) for calculation purposes.
5. **Desired Acceleration (a):** Let's assume a moderate acceleration of 0.5 m/s².
6. **Coefficient of Rolling Friction (μ):** Typically around 0.01 to 0.03 for smooth surfaces. We'll use 0.02 for this calculation.
7. **Gravitational Acceleration (g):** 9.81 m/s².

Step-by-Step Calculation:

1. **Calculate the Total Force Required to Accelerate the System:**

$$F_{\text{acceleration}} = m \times a = 20 \text{ kg} \times 0.5 \text{ m/s}^2 = 10 \text{ N}$$

2. **Calculate the Force Required to Overcome Rolling Friction:**

$$F_{\text{friction}} = \mu \times m \times g = 0.02 \times 20 \text{ kg} \times 9.81 \text{ m/s}^2 = 3.924 \text{ N}$$

3. **Total Force Required:**

$$F_{\text{total}} = F_{\text{acceleration}} + F_{\text{friction}} = 10 \text{ N} + 3.924 \text{ N} = 13.924 \text{ N}$$

4. **Torque Required per Motor:**

Since there are two motors, each motor will share the load equally. The torque required per motor can be calculated using the formula:

$$\tau = F_{\text{total}} \times D/2 \times 2$$

Where:

- $D/2$ is the radius of the wheel.

- 1/2 is the factor to account for two motors sharing the load.

Plugging in the values:

$$\tau = 13.924 \text{ N} \times 0.1 \text{ m} \times 12 = 0.3481 \text{ Nm}$$

Conclusion:

Each motor should have a torque of at least **0.35 Nm** to ensure the system can accelerate at 0.5 m/s^2 while overcoming rolling friction.

Additional Considerations:

- **Safety Margin:** It's often a good idea to add a safety margin (e.g., 20-30%) to account for uncertainties like uneven surfaces, additional friction, or higher acceleration requirements.
- **Motor Efficiency:** Ensure the motors can sustain the required torque over the desired operating time without overheating.
- **Wheel Slippage:** Ensure the wheels have sufficient traction to prevent slippage, especially if the system operates on inclined surfaces or rough terrain.

If you have specific requirements for acceleration, wheel diameter, or surface conditions, you can adjust the calculations accordingly.

E.9 DC Motor Control MCU

Prompt: “how can i drive a brushed dc motor using a microcontroller and a motor driver IC”

Outcome:

ChatGPT-4o:

Driving a **brushed DC motor** using a **microcontroller** and a **motor driver IC** requires several key components and steps. Here's a breakdown of how to do it:

1. Components Needed

- **Microcontroller (MCU)** (e.g., Arduino, ESP32, STM32, etc.)
- **Motor Driver IC** (e.g., L298N, L293D, DRV8871, TB6612FNG)
- **Brushed DC Motor**
- **Power Supply** (Matching the voltage and current requirements of the motor)
- **Wires & Breadboard** (For connections)

2. How It Works

A **microcontroller** alone cannot supply enough current to drive a motor, so a **motor driver IC** is used as an interface between the microcontroller and the motor. The

microcontroller sends **PWM (Pulse Width Modulation) signals** and logic levels to control speed and direction.

3. Connection Steps

(A) Basic Connections

1. Power Supply:

- Connect **VCC** of the motor driver to the motor's power source.
- Connect **GND** of the motor driver to the **GND** of both the MCU and power supply.

2. Motor Connections:

- Connect the motor's terminals to the **OUT1 and OUT2** pins of the driver IC.

3. Microcontroller to Motor Driver:

- **IN1 and IN2** (Motor Direction Control) → Connect to digital output pins of MCU.
- **PWM Input (if available, e.g., ENA on L298N)** → Connect to a PWM pin of the MCU for speed control.

4. Enable Pin (if applicable):

- Some drivers require an **enable pin** (e.g., L298N's ENA). Set HIGH (or use PWM for speed control).
-

4. Writing Code (Example for Arduino)

(A) Basic ON/OFF Control

```
#define IN1 9
```

```
#define IN2 10
```

```
void setup() {  
  pinMode(IN1, OUTPUT);  
  pinMode(IN2, OUTPUT);  
}
```

```

void loop() {

    digitalWrite(IN1, HIGH);

    digitalWrite(IN2, LOW); // Motor moves forward

    delay(2000); // Run for 2 sec


    digitalWrite(IN1, LOW);

    digitalWrite(IN2, HIGH); // Motor moves in reverse

    delay(2000); // Run for 2 sec

}

```

(B) Speed Control using PWM

```

#define IN1 9

#define IN2 10

#define PWM_PIN 11 // PWM control (ENA on L298N)


void setup() {

    pinMode(IN1, OUTPUT);

    pinMode(IN2, OUTPUT);

    pinMode(PWM_PIN, OUTPUT);

}


void loop() {

    analogWrite(PWM_PIN, 128); // 50% speed (0-255 range)

    digitalWrite(IN1, HIGH);

    digitalWrite(IN2, LOW); // Forward direction

```

```
delay(3000);

analogWrite(PWM_PIN, 200); // Increase speed
delay(3000);

digitalWrite(IN1, LOW);
digitalWrite(IN2, HIGH); // Reverse direction
delay(3000);

analogWrite(PWM_PIN, 0); // Stop motor
delay(2000);
}
```

5. Notes

- **Motor voltage** must match the **motor driver's rated voltage**.
- **Use a flyback diode** (if not built-in) across the motor terminals to protect against voltage spikes.
- If using **high current motors**, consider **MOSFET-based** H-bridge circuits.

Let me know if you need help choosing a motor driver! 🚀

DeepSeek:

Driving a brushed DC motor using a microcontroller and a motor driver IC involves several steps. Below is a general guide to help you set up the system:

Components Needed:

1. ****Microcontroller**** (e.g., Arduino, PIC, STM32, etc.)
2. ****Motor Driver IC**** (e.g., L298N, L293D, TB6612FNG, etc.)

3. **Brushed DC Motor**
4. **Power Supply** (appropriate for the motor and driver IC)
5. **Decoupling Capacitors** (if required by the motor driver IC)
6. **Diodes** (for flyback protection, if not integrated in the motor driver IC)
7. **Resistors** (for current sensing or pull-up/pull-down, if needed)
8. **Connecting Wires**
9. **Breadboard or PCB** (for prototyping)

Step-by-Step Guide:

1. Understand the Motor Driver IC:

- **Input Pins:** These are connected to the microcontroller to control the motor.
- **Output Pins:** These are connected to the motor terminals.
- **Power Pins:** These include VCC (logic voltage), VM (motor voltage), and GND.
- **Enable Pins:** Some drivers have enable pins to turn the motor on/off or control speed via PWM.

2. Connect the Motor Driver IC to the Microcontroller:

- **Control Pins:** Connect the input pins of the motor driver IC to the GPIO pins of the microcontroller.
- **Enable Pins:** If your motor driver has enable pins, connect them to PWM-capable pins on the microcontroller if you want speed control.
- **Ground Connection:** Ensure that the GND of the microcontroller and the motor driver IC are connected.

3. Connect the Motor to the Motor Driver IC:

- **Motor Terminals:** Connect the two terminals of the brushed DC motor to the output pins of the motor driver IC.
- **Power Supply:** Connect the motor power supply (VM) to the motor driver IC. Ensure the voltage matches the motor's rated voltage.
- **Flyback Diodes:** If your motor driver IC does not have built-in flyback diodes, connect diodes across the motor terminals to protect the circuit from voltage spikes generated by the motor.

4. Power the Circuit:

- **Logic Voltage:** Connect the VCC pin of the motor driver IC to the microcontroller's voltage level (usually 3.3V or 5V).
- **Motor Voltage:** Connect the VM pin to the motor's power supply. This should be separate from the microcontroller's power supply if the motor requires a higher voltage.

5. Write the Microcontroller Code:

- **Direction Control:** Use the GPIO pins to set the direction of the motor by setting the appropriate logic levels on the input pins of the motor driver IC.
- **Speed Control:** If your motor driver supports PWM, use the microcontroller's PWM output to control the speed of the motor by varying the duty cycle.

Example Code (Arduino):

```
```cpp
// Define motor control pins
const int in1 = 9; // Input 1 for motor driver
const int in2 = 8; // Input 2 for motor driver
const int enA = 10; // Enable pin for motor driver (PWM)

void setup() {
 // Set the motor control pins as outputs
 pinMode(in1, OUTPUT);
 pinMode(in2, OUTPUT);
 pinMode(enA, OUTPUT);
}

void loop() {
 // Rotate motor clockwise
 digitalWrite(in1, HIGH);
 digitalWrite(in2, LOW);
 analogWrite(enA, 255); // Full speed
 delay(2000);

 // Stop motor
 digitalWrite(in1, LOW);
 digitalWrite(in2, LOW);
 delay(1000);

 // Rotate motor counter-clockwise
 digitalWrite(in1, LOW);
 digitalWrite(in2, HIGH);
 analogWrite(enA, 128); // Half speed
 delay(2000);

 // Stop motor
 digitalWrite(in1, LOW);
 digitalWrite(in2, LOW);
 delay(1000);
}
```
```

6. ****Test and Debug:****

- ****Test the Motor:**** Upload the code to the microcontroller and observe the motor's behavior.
- ****Adjust PWM:**** Modify the PWM values to adjust the motor speed.
- ****Check for Overheating:**** Ensure that the motor driver IC is not overheating. If it does, consider adding a heat sink or using a more powerful driver.

Additional Tips:

- **Current Sensing:** Some motor drivers have current sensing pins that allow you to monitor the motor's current draw.
- **Protection Circuits:** Consider adding fuses or current-limiting resistors to protect the circuit from overcurrent conditions.
- **Heat Management:** Ensure proper heat dissipation for the motor driver IC, especially if driving high-power motors.

By following these steps, you should be able to successfully drive a brushed DC motor using a microcontroller and a motor driver IC.