Design Methodology Document - AMRC

210019E - Abithan A.
210292G - Kirushanth G
210498T - Priyankan V.
210624E - Sundarbavan T.

Submitted as a requirement for an assignment in the course module EN2160
at the Electronic and Telecommunication Department, University of Moratuwa
Colombo, Sri Lanka

10 July 2024

Contents

Chapter 1

Review Progress and Plan Next Steps

During the research phase, we identified two companies with notable contributions to the field of autonomous mobile robotics. SEER Robotics emerged as a comprehensive solution provider, developing not just an AMR controller but also a suite of essential components for the assembly and integration of Autonomous Mobile Robots. On the other hand, JHCTECH only creates AMR controller, focusing their expertise on the development of this singular but critical component in the ecosystem of mobile robotic automation.

1.1 SEER Robotics

Based on the websites and videos available about SEER Robotics and its products, it's evident that SEER has made significant advancements in the development of Autonomous Mobile Robot (AMR) controllers and the integration of these controllers with various mobile robots for industrial applications. Here's an analysis of the type of work done by SEER that could relate to the AMR product we intend to develop during your course:

1.1.1 Development of Advanced AMR Controllers

SEER has developed a series of autonomous mobile robot controllers, notably the SRC series, including SRC-3000FS, SRC-2000, and SRC-800. These controllers are designed to be the "brain" of mobile robots, handling path navigation and point-to-point travel. The SRC series highlights SEER's focus on creating versatile and highly integrated controllers capable of supporting multiple navigation modes, including laser SLAM navigation, QR code navigation, and laser reflector navigation.

1.1.2 Support for Various Navigation Modes

The primary navigation mode supported by SEER's controllers is laser SLAM navigation, which allows for fast deployment without the need for additional infrastructure such as magnetic strips or reflectors. This approach to navigation is particularly relevant for projects aiming to develop AMRs that can quickly adapt to new environments.

1.1.3 Integration with a Vehicle Body Library

SEER has built a rich and powerful vehicle body library, enabling the integration of its controllers with a wide variety of mobile robots. This library supports motion models like steering wheel, mecanum wheel, differential wheel, and others, demonstrating SEER's commitment to versatility in AMR design.

1.1.4 Rapid Implementation Tools

The RoboShop Pro tool developed by SEER Robotics underscores SEER's emphasis on easing the AMR development process. This software tool aids in the quick deployment and customization of mobile robots, catering to both integrators looking to build their brand and those seeking customized mobile robots.

1.1.5 Safety and Certification

The mention of the SRC-3000FS as the first safety controller to pass SGS TÜV functional safety certification highlights SEER's attention to safety standards. This aspect is crucial for AMRs operating in environments where human interaction is frequent.

1.1.6 Advanced Features for Complex Environments

SEER's AMRs are equipped with advanced features like pallet identification, shelf leg recognition (leg identification), and following movements, enhancing their capability to operate in complex industrial scenarios. These features are enabled through the use of 3D cameras and LiDAR technology, allowing for precise interaction with the environment.

1.1.7 Components and Integration

SEER Robotics' AMR controllers are a highly integrated device that include:

1. Wi-Fi: For wireless communication within industrial settings, offering connectivity to other devices and systems.
2. Switch: To manage network traffic, which is essential for multi-robot coordination and data transmission.
3. Gyroscope: For maintaining orientation and balance, providing critical data for navigation and motion control.
4. Industrial Computer: Suggesting that the controller has substantial processing power to handle complex tasks and algorithms.
5. IO Board: Indicating a range of input/output capabilities for interfacing with various peripherals and sensors.

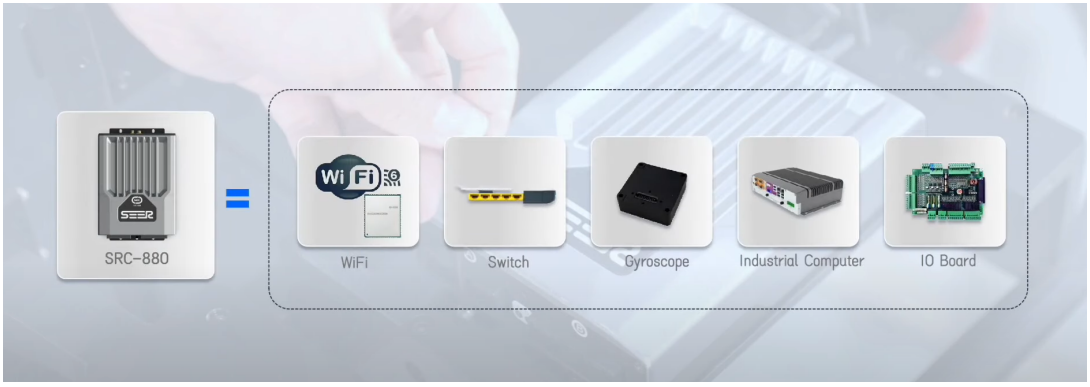


Figure 1.1: SRC 800

1.1.8 Sensor Compatibility

Different laser brands are used for integration. It emphasizes SEER Robotics' commitment to versatility and interoperability. By pre-integrating widely recognized sensor brands, SEER is ensuring that their AMR controllers can be used in various configurations and applications, catering to specific industry needs and allowing for easy customization. Sensor brands used are SICK, LIVOX and Baumer. In addition to this an obstacle avoidance 3D camera can be also connected to the AMR controller.

1.2 JHCTECH - KGEC-6200 Edge Controller

Based on the websites and data sheets,JHCTECH's KGEC-6200 is a high-performance edge controller designed for use in various automated systems, including Autonomous Mobile Robots (AMRs).

1.2.1 High-Performance Processor

The range of Celeron processors with varying numbers of cores, threads, and memory support indicate that the KGEC-6200 can be customized for different levels of computational needs. The processors can handle multi-threaded workloads which is essential for complex tasks in robotics, like SLAM, path planning, and real-time decision-making.

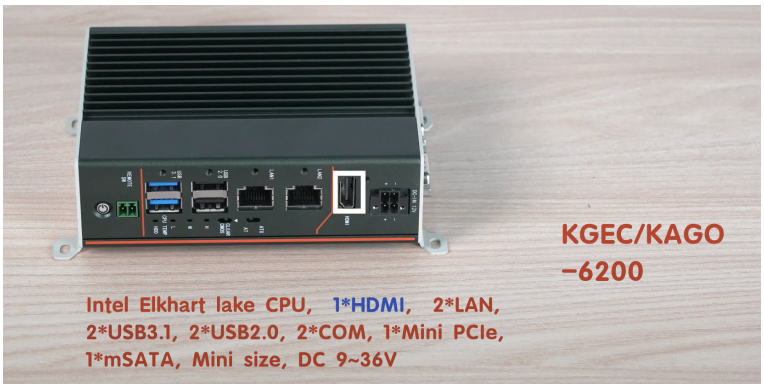


Figure 1.2: KGEC-6200 Edge Controller

1.2.2 Graphics Capabilities

Support for Intel Gen10 UHD Graphics and the ability to support 4K display output indicates the KGEC-6200 can handle high-resolution video processing tasks, which could be beneficial for machine vision applications in robotics.

1.2.3 Real-time Ethernet Protocol and Operating System

With support for real-time Ethernet protocols and compatibility with real-time operating systems like RTX and INtime, the KGEC-6200 is suited for time-critical tasks in AMR operations, ensuring timely and synchronized actions which are crucial in autonomous navigation and coordination.

1.2.4 Support for Mainstream Operating Systems

The ability to run on both Windows and Linux and support for CODESYS, a development environment for programming controller applications, suggests the KGEC-6200 is versatile and can be integrated into various existing robotics and automation ecosystems.

1.2.5 Mini Size, Ultra-Light Weight

The compact and lightweight design of the KGEC-6200 makes it ideal for mobile applications where space and weight are at a premium, such as on-board systems for AMRs.

1.2.6 Fanless Passive Heat Dissipation Design

The fanless design for heat dissipation suggests the controller can reliably operate in a range of environments without the need for active cooling, which is advantageous for industrial or outdoor applications where dust and debris could impede a fan.

1.2.7 Flexible Installation Method and Anti-off Design

These features show that the KGEC-6200 is built for the robust demands of industrial environments, ensuring secure installation and reliable connection even in mobile or vibrating environments, which is typical for AMRs.

1.2.8 Modular Combination

The modular design of the KGEC-6200 allows for customization and easy integration with other system components, which is essential for building AMRs that may require integration with a variety of sensors and actuators.

1.2.9 Modular Design of Wide Voltage Power Supply

The power supply's wide voltage range and protection features make it suitable for mobile applications where power supply stability and flexibility are important.

1.2.10 Target Market

The focus on AGV and AMR markets and applications in various domains such as logistics, delivery, and industrial control aligns with your project's intent to develop an AMR. It shows that JHCTECH is positioning the KGEC-6200 as a versatile controller capable of driving innovation in these areas.

1.3 Next Steps

Based on above study following things have to be reaserched and implented in our project.

1. Focusing on Advanced Navigation: Incorporating laser SLAM or other versatile navigation techniques that don't rely heavily on environmental modifications.
2. Safety and Integration: Designing our AMR controller with safety as a priority, possibly aiming for relevant certifications, and ensuring that our design can easily integrate with existing industrial ecosystems.
3. Rapid Development and Testing Tools: Utilizing or developing software tools similar to RoboShop Pro to streamline the design, implementation, and testing phases of our AMR controller project.
4. Ensuring Compatibility: Design our AMR controller to be compatible with industry-standard sensors and components, which can make it more versatile and appealing for different applications.
5. Focusing on Connectivity: Implement robust and flexible communication options, such as dual-band Wi-Fi, to ensure reliable operation in diverse environments.
6. Prioritizing Modularity: Like SEER's controllers, aim for a modular design that allows for easy integration and upgrades of components.

1.4 User Requirements

The AMR controller, engineered with a focus on seamless compatibility, serves as the central intelligence hub for a diverse array of Autonomous Mobile Robots (AMRs) equipped with LiDAR sensors. It embodies a versatile architecture meticulously crafted to effortlessly integrate with an extensive spectrum of AMR models, irrespective of their manufacturers or underlying software frameworks. At its core, this controller leverages advanced LiDAR integration capabilities, harnessing the power of LiDAR sensors to furnish the AMR fleet with unparalleled environmental awareness and navigational precision. By interfacing seamlessly with LiDAR sensors, the controller facilitates real-time data acquisition, enabling the AMRs to dynamically adapt to their surroundings and navigate complex environments with unparalleled agility and efficiency. Below are the essential requirements for an AMR controller.

1. Universal Compatibility: The AMR controller is designed to be compatible with any AMR that utilizes LiDAR sensors for navigation.
2. Plug-and-Play Functionality: The controller is intended to offer a plug-and-play experience, meaning it can be directly connected to the AMR without the need for extensive coding or customization.
3. Hardware and Software Independence: The controller is designed to be hardware and software independent, meaning it can operate with any AMR platform and is not tied to a specific manufacturer or software ecosystem.
4. Automated Operation: Once connected to the AMR robot, the controller facilitates automated operation, controlling the robot's movements and navigation based on predefined parameters and algorithms.

Chapter 2

Stakeholder Map

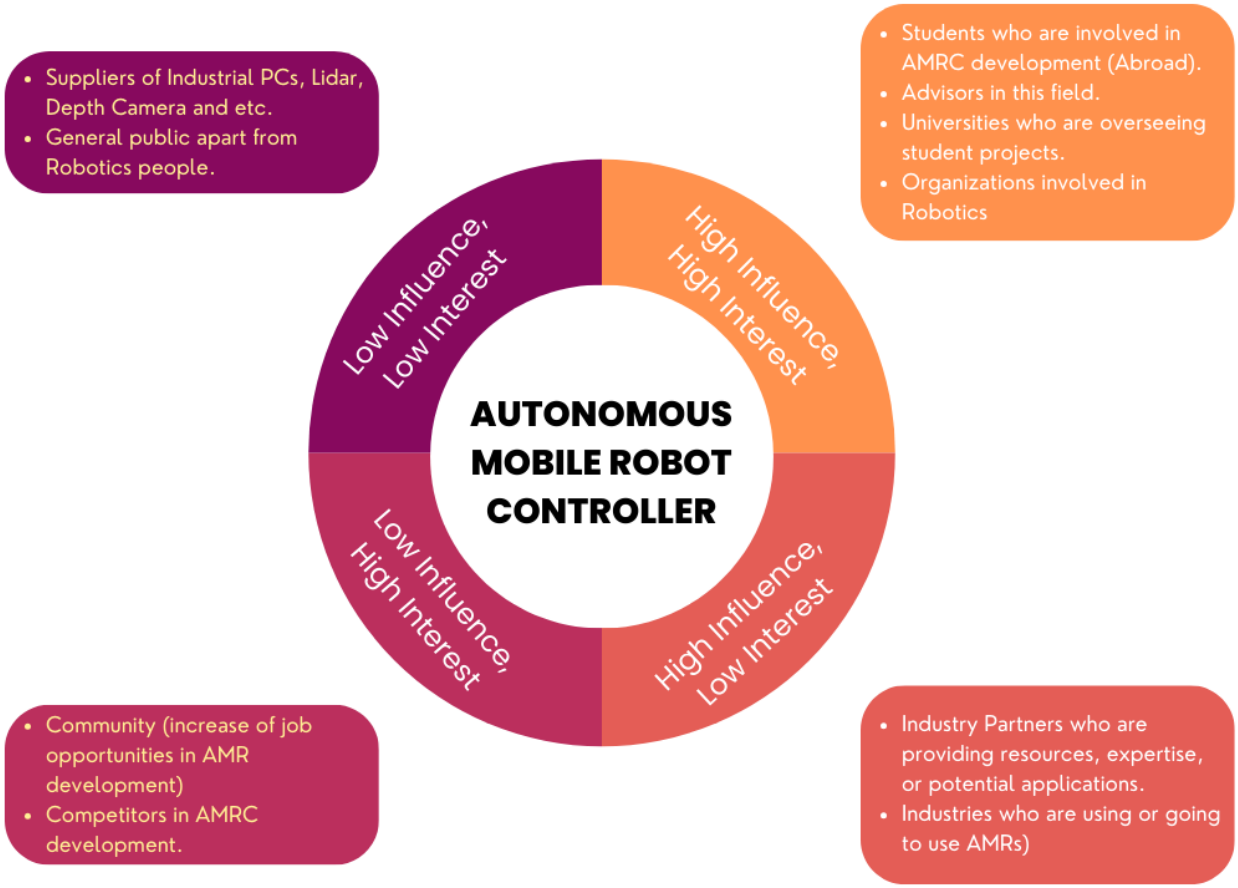


Figure 2.1: Stakeholder Map

Chapter 3

Observe Users

Autonomous Mobile Robot Controller users encompass a diverse range of individuals across various industries, all united by their involvement in deploying, operating, and managing autonomous mobile robots (AMRs). Below, we outline the key user groups and their roles in utilizing the AMRC system:

3.1 End-Users in Warehousing and Logistics

Warehouse Operators: Use AMRs for inventory management, picking, and placing tasks.
Logistics Managers: Oversee the use of AMRs for sorting, loading, and unloading goods.

Example:
Amazon Robotics: Uses AMRs in their warehouses for picking and sorting packages.
Walmart: Employs robots for inventory management and cleaning floors in stores.
FedEx: Uses AMRs for sorting and transporting packages within their distribution centers.
DHL: Employs various robotic systems, including AMRs, for order fulfillment and warehouse operations.

3.2 Manufacturing Industry

Production Line Workers: Interact with AMRs for material handling and assembly line feeding.
Plant Managers: Monitor and coordinate AMR activities to ensure efficient production workflows.

Example:
Tesla: Integrates AMRs for material handling and parts delivery within their factories.
BMW: Utilizes AMRs for transporting materials between different stages of the manufacturing process.

3.3 Healthcare Facilities

Hospital Staff: Use AMRs for transporting medical supplies, lab specimens, or even for sanitization tasks.
Pharmacy Technicians: Leverage AMRs for medication delivery within large hospital environments.

Example:
Aethon: Provides TUG autonomous robots for hospitals to deliver medication, transport laboratory specimens, and manage waste.
Diligent Robotics: Offers Moxi, a hospital robot assistant that helps clinical staff with non-patient-facing tasks.

3.4 Hospitality Industry

Hotel Staff: Use AMRs for room service delivery, guest escorting, and luggage handling.
Facilities Managers: Implement AMRs for cleaning and maintenance tasks.

Example
Savioke: Creates service robots like Relay, which deliver items to hotel guests' rooms.
YOTEL: Uses robots for luggage storage and other guest services.

3.5 Agricultural Sector

Farmers and Agronomists: Utilize AMRs for planting, harvesting, monitoring crops, and distributing materials.

Example:
Naïo Technologies: Provides agricultural robots for weeding, hoeing, and planting.
John Deere: Integrates autonomous tractors and harvesting systems into their product lineup.

3.6 Residential and Personal Services

Home Users: In some cases, AMRs might be used for personal assistance, cleaning, or delivery in residential spaces.

Example:
iRobot: Well-known for their Roomba vacuuming robots, used in residential and commercial settings.
Brain Corp: Develops BrainOS, which powers various types of cleaning robots used in retail and commercial spaces.
Neato Robotics: Designs robotic vacuum cleaners for home use.

3.7 Software Developers and System Integrators

Automation Engineers: Develop custom solutions and integrate AMRs with existing industrial systems.
Robotics Programmers: Write and maintain the software that controls the AMRs and enables them to perform tasks autonomously.

Chapter 4

Need List

The formulation of need list is a pivotal phase in the development of the Autonomous Mobile Robot (AMR) Controller Project. It ensures that the final design aligns with both user expectations and business objectives. This document outlines a comprehensive and categorized list of needs, structured in a manner that connects the design project's requirements with those of its users, enabling the prioritization of these needs based on the outcomes they facilitate or hinder.

Need List:

1. As an end-user (operator) in a warehouse setting, I need an intuitive interface to input destination goals for the AMR so that I can efficiently direct it to desired locations without extensive training.
2. As a safety manager, I need the AMR to adhere to safety protocols and avoid obstacles autonomously so that it can ensure the safety of human workers and other assets in the operational environment.
3. As a logistics coordinator, I need the AMR to optimize its pathfinding in real-time so that it can deliver items more quickly, enhancing overall workflow efficiency.
4. As a facility manager, I need the AMR to operate autonomously across different environments within the facility so that it can be deployed flexibly without the need for constant reconfiguration.
5. As an IT administrator, I need the AMR's software to integrate seamlessly with our existing warehouse management system so that it can synchronize tasks and share data without compatibility issues.
6. As a maintenance technician, I need the AMR to provide diagnostic feedback and alert me to potential issues so that I can proactively address maintenance needs and minimize downtime.
7. As a business stakeholder, I need the AMR to demonstrate a clear return on investment through increased productivity and reduced operational costs so that the business can achieve its financial objectives.
8. As a product developer, I need the AMR controller to be modular and scalable so that it can support future upgrades and integration with new technologies, ensuring the product remains competitive.

Chapter 5

Stimulate Ideas

Stimulating ideas is an essential phase in the design methodology of our Automatic Mobile Robot Controller (AMRC) project. This phase aims to foster a creative environment to break away from established patterns of thinking and generate innovative solutions. Here are some key principles and techniques we applied to stimulate ideas effectively:

5.1 Key Principles

Challenge Assumptions

We started by listing common assumptions about AMR controllers and then systematically challenged each one. This helped us break free from conventional thinking and explore new possibilities.

Encourage Quantity over Quality Initially

Our initial goal was to generate as many ideas as possible without judgment. This approach helped create a diverse pool of ideas from which we could later refine and select the best ones.

Embrace Unconventional Ideas

We actively encouraged wacky and unconventional ideas, looking for inspiration in unusual places and asking, "How else could it be done?" This helped us think outside the box and consider novel solutions.

Record and Discuss Ideas Clearly

All ideas were meticulously recorded and clearly presented. This facilitated further discussion and ensured that no potential solution was overlooked during the evaluation phase.

Inspire Further Creativity through Evaluation

During the evaluation activities, we used feedback to inspire further creativity and refine our ideas. This iterative process ensured continuous improvement and innovation.

5.2 Idea Generation Techniques

We conducted multiple brainstorming sessions with our team and another team (3D Lidar team) This diversity of perspectives helped generate a wide range of ideas.

Mind maps were created to visually organize ideas and explore relationships between different concepts. This helped identify new connections and potential innovations.

We looked at analogies and metaphors from unrelated fields to inspire new ways of thinking about our AMRC. For example, we considered how a smartphone adapts to various apps and contexts, drawing parallels to how our controller could be modular and versatile.

We examined existing AMR controllers and deconstructed their features and functionalities. This reverse engineering process helped us understand current limitations and identify opportunities for improvement.

5.3 Generated Ideas

Here are some of the key ideas generated during this phase:

Modular Design

Develop a modular AMRC that can be easily added to any motor or sensor. This would allow for flexibility in configuration and customization for different applications.

Plug-and-Play Architecture

Create a plug-and-play architecture where components like sensors, cameras, and motors can be connected and configured without extensive programming knowledge.

Adaptive Learning Algorithms

Implement adaptive learning algorithms that allow the controller to learn and improve its navigation and mapping capabilities over time.

Interchangeable Navigation Modes

Design the controller to support multiple interchangeable navigation modes (e.g., laser SLAM, QR code navigation) that can be selected based on the environment and application.

Cloud Integration for Data Sharing

Integrate cloud connectivity to allow data sharing and collaboration across multiple robots and systems, enabling real-time updates and improvements.

User-Friendly Interface

Develop a user-friendly interface with intuitive controls and visualization tools to simplify the setup and operation of the AMRC.

Enhanced Safety Features

Incorporate advanced safety features, such as real-time obstacle detection and avoidance, to ensure safe operation in dynamic environments.

5.4 Conclusion

By applying these principles and techniques, we generated a diverse range of ideas for our Automatic Mobile Robot Controller project. These ideas were grouped, prioritized, and evaluated to inspire further creativity and guide the development of a robust, innovative, and versatile AMRC solution. This approach ensures that we are not only solving existing problems but also paving the way for future advancements in autonomous mobile robotics.

Chapter 6

Overall Block Diagrams

6.1 Block Diagram 1

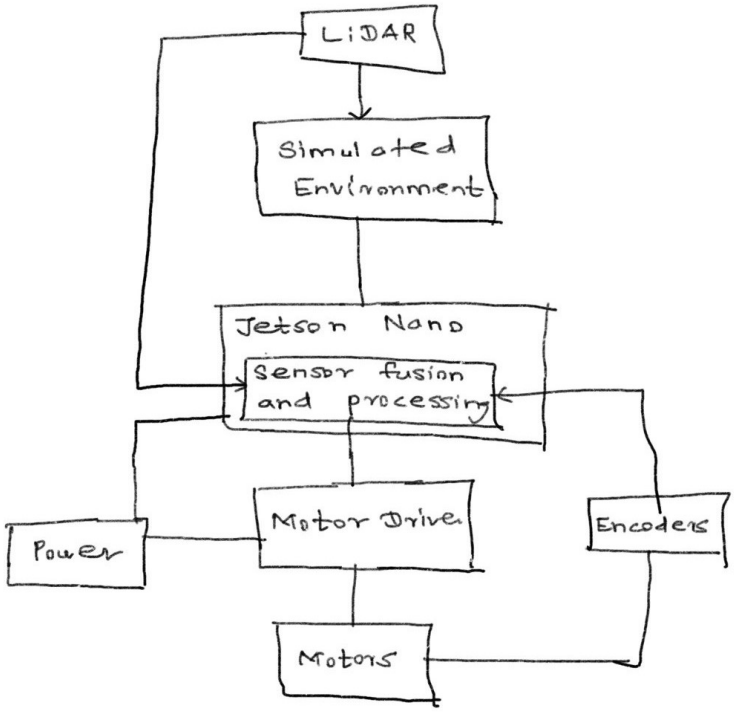


Figure 6.1: Block Diagram

LiDAR technology is used in conjunction with a simulated environment for data generation, crucial for the robot's understanding of its surroundings. NVIDIA Jetson Nano acts as the computational core, receiving sensor data, processing it, and determining the robot's navigation and interaction strategies. Sensor Fusion and Processing is a software module within the Jetson Nano that integrates sensor data for accurate localization and mapping. Motor Driver translates commands from the Jetson Nano into actuation signals, controlling the operation of the motors. Encoders provide feedback to the Jetson Nano regarding the actual movement of the motors, which is essential for closed-loop control.

The chosen design includes LiDAR technology, which is critical for accurate real-time mapping and obstacle detection, enabling the AMR to navigate complex environments safely. Furthermore, encoders attached to the motors offer critical feedback for closed-loop control systems, ensuring that the actual motor movements align with the commands issued by the Jetson Nano. The modular nature of the selected design allows for future upgrades and the integration of additional sensors or actuators without significant reconfiguration.

Specifications (to be decided)

1. Battery type, capacity, and voltage.
2. Motor specifications including voltage, current, torque, and speed.
3. Details on the motor driver's power handling capabilities, efficiency, and thermal performance.
4. LiDAR's range, resolution, and scanning frequency.

6.2 Block Diagram 2

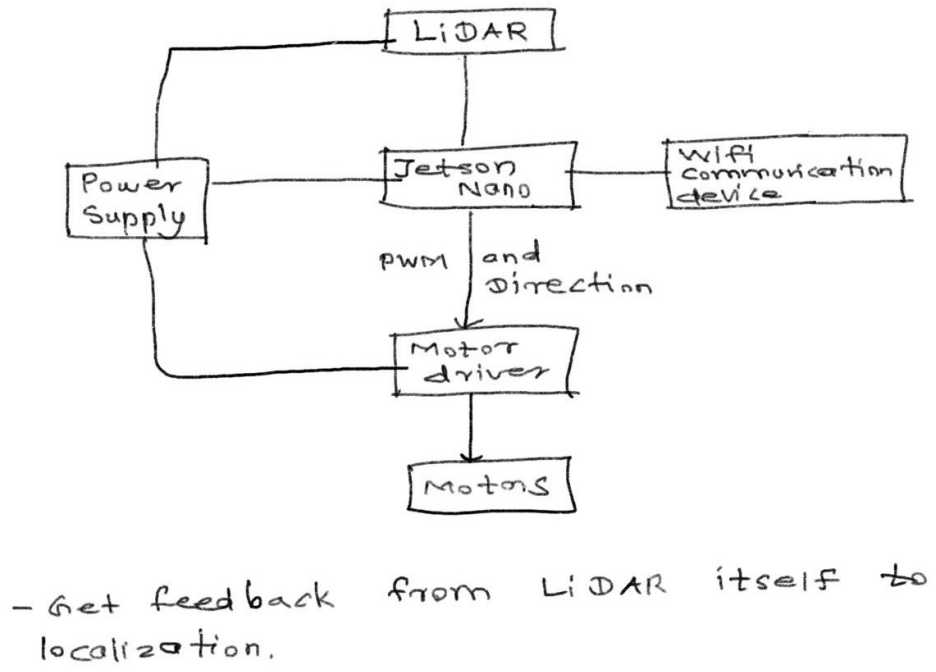


Figure 6.2: Block Diagram - Alt 1

The structure diagram delineates the functional layout of the AMR system. The motor driver receives PWM and direction signals from the Jetson Nano to control the two 12V DC motors, which propel the robot. The Jetson Nano serves as the central processing unit, interfacing with LiDAR for environmental mapping and a Wi-Fi communication device for remote connectivity. The Jetson Nano sends commands to the motor driver for movement while also communicating with external devices. The LiDAR sensor, connected to the Jetson Nano, scans the environment to facilitate navigation and obstacle avoidance.

In the alternative design of the Autonomous Mobile Robot (AMR),the sensor suite can be streamlined by exclusively using LiDAR for environmental mapping and robot localization. This approach simplifies the hardware configuration by eliminating the need for encoders and focuses computational resources on processing the rich spatial data provided by LiDAR. Opting for a LiDAR-centric configuration reduces the overall system complexity and potential points of failure. Although this poses a challenge in terms of ensuring the same level of precision

typically afforded by multiple sensor inputs, it opens up opportunities for system integration and efficiency.

6.3 Block Diagram 3

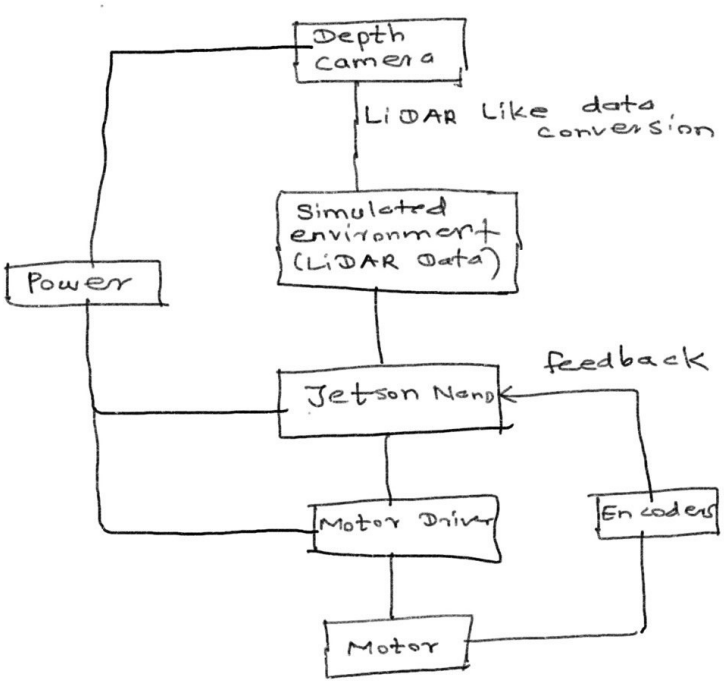


Figure 6.3: Block Diagram - Alt 2

Central to this system is the innovative use of a depth camera, which provides detailed three-dimensional information about the robot's surroundings. This data is then algorithmically transformed into a format akin to that produced by LiDAR sensors, facilitating compatibility with SLAM algorithms traditionally designed for LiDAR inputs. The depth camera data undergoes a conversion process that aligns it with the expectations of the SLAM algorithm on the Jetson Nano, ensuring that the robot can create an accurate map of its environment and effectively determine its location within that map.

Actuation is managed by the motor drivers, which receive commands from the Jetson Nano to control the motors, thereby enabling movement and steering. Encoders attached to the motors provide feedback to the Jetson Nano, creating a closed-loop system that ensures precise control of the robot's trajectory and speed. This feedback mechanism is vital for correcting any discrepancies between commanded and actual movements, which is essential for maintaining accurate localization and consistent performance in dynamic environments.

6.4 Ovrerall Block Diagram Complete Comparison

	Block Diagram 1	**Block Diagram 2**	**Block Diagram 3**
Functionality	7	9	6
Ease of use	7	8	5
Manufacturing feasibility	7	7	6
Cost Effectiveness	9	4	6
Performance	7	8	8
Sustainability	6	8	6
Power Efficiency	9	6	7
Total	**52**	**50**	**44**

6.5 Used Block Diagram

This section provides a detailed breakdown of the hardware and software configuration for the Automatic Mobile Robot (AMR) project. The setup was designed to accommodate the unavailability of certain components such as the Jetson Nano, proper motor encoders, and LiDAR, while still achieving the project's goals for effective autonomous navigation and operation.

6.5.1 System Components and Interactions

The AMR system comprises several key components, each playing a crucial role in the robot's functionality. Here is an overview of the system architecture:

Laptop (Simulation Environment)

The laptop acts as the main control unit, running a simulation environment where the robot's movements are virtually represented. This setup allows for the testing and development of control algorithms without physical trials.

UART Communication

Communication between the laptop and the Arduino Uno is facilitated via UART. This ensures that commands and data are relayed effectively, maintaining synchronization between the simulated environment and the physical robot.

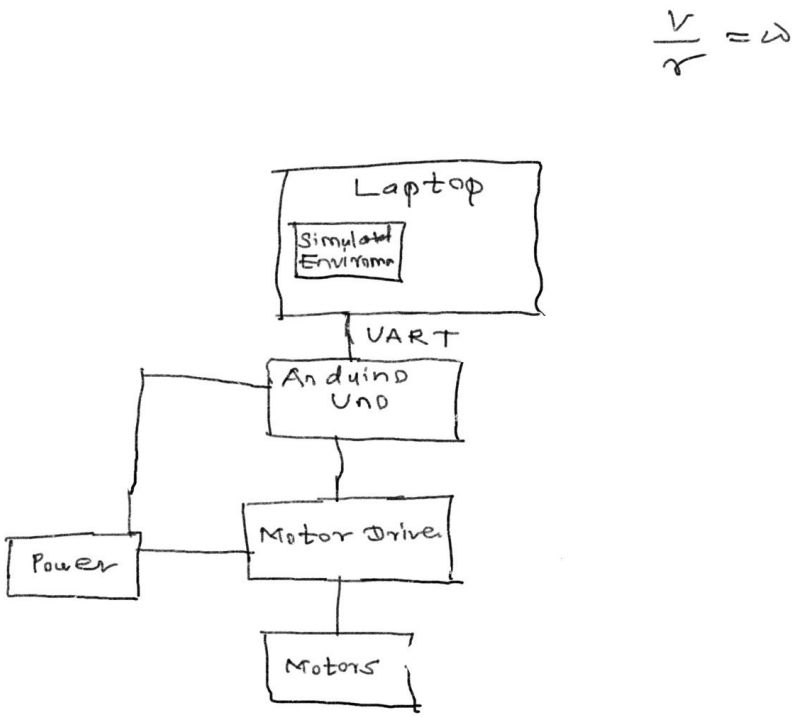


Figure 6.4: Used Block Diagram

Arduino Uno

The Arduino Uno serves as an intermediary controller, programmed to interpret PWM (Pulse Width Modulation) signals and adjust the motor operations accordingly. It receives commands from the laptop and controls the motor drivers.

Motor Driver

The motor driver receives input signals from the Arduino Uno and controls the motors' direction and speed based on these signals. This component is crucial for the precise movement and steering of the robot.

Motors

Motors are the driving force of the robot, executing movements as commanded by the motor driver. They are essential for the robot's mobility and are directly powered by the motor driver.

Power Supply

The power supply provides the necessary electrical power to the motor driver and motors. It ensures that the AMR has sufficient energy to operate effectively and continuously.

Real-Time Synchronization

The hardware and software setup is designed to mirror the movements of the virtual robot in the simulation environment with the real robot. This real-time synchronization is achieved without the need for advanced sensors or high-powered processing units, fitting within the resource constraints and maintaining the project's goals.

Chapter 7

Electronics Design - Motor Driver

7.1 Introduction

This section details the electronic subsystems design of the Autonomous Mobile Robot (AMR) with a focus on the motor driver, essential for motor control and robot locomotion.

7.2 Motor Driver Block diagram 1

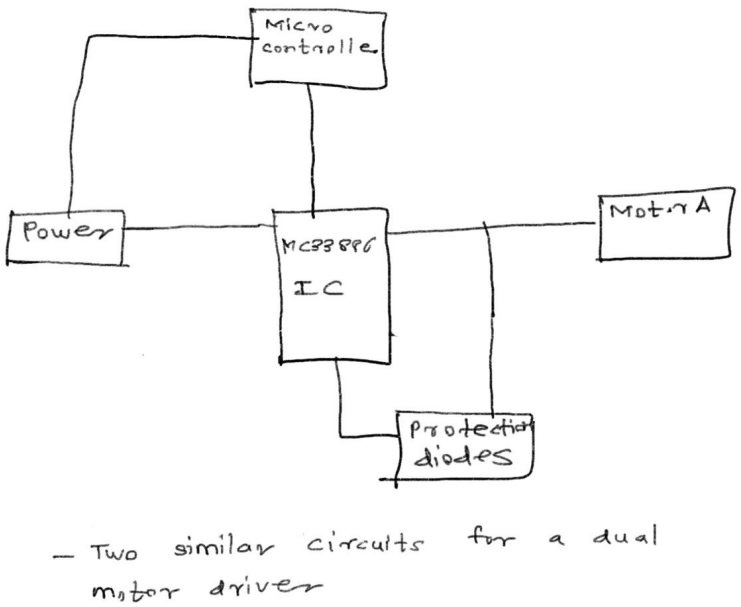


Figure 7.1: Enter Caption

The microcontroller outputs signals to the MC33886 IC, dictating the direction and speed of each motor. For speed control, the microcontroller outputs a PWM signal, which is converted by the MC33886 into a proportional voltage applied to the motors. For direction control, binary signals determine the polarity of the voltage applied across the motor, thus controlling the direction of rotation. Two identical circuits are used, one for each motor (Motor A and Motor B), to allow for independent control which is necessary for steering the AMR. This dual motor driver design is a cornerstone of our AMR's ability to navigate and adapt to the complexities of real-world environments. By leveraging the capabilities of the MC33886 IC, we achieve a balance

between functionality and reliability, making our AMR capable of executing precise movements and maneuvers dictated by the higher-level navigation and SLAM algorithms.

7.3 Motor Driver Block diagram 2

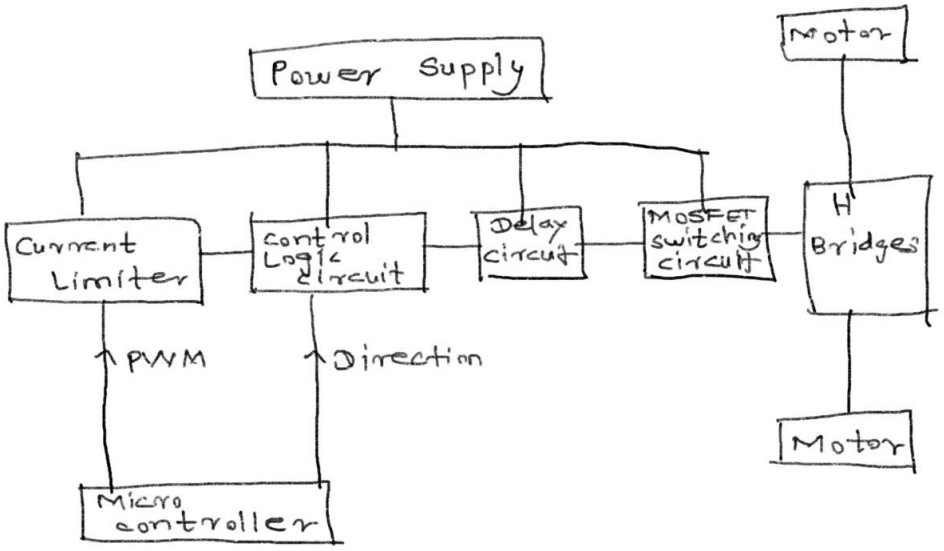


Figure 7.2: Alternative 1

Current Limiters, which are crucial for protecting the motors from over-current conditions that could potentially cause damage used here. Control Logic Circuit receives directional commands from the main processing unit and translates these into control signals for the H-bridges, dictating the direction of the motor rotation. PWM signal is generated by the micro-controller. Delay Circuit ensures that there is no crossover conduction during the switching period between the high-side and low-side MOSFETs within the H-bridges, which could otherwise lead to short circuits. MOSFET Switching Circuit uses MOSFETs to switch the motor current on and off rapidly, allowing for effective speed control through the PWM signal. H-Bridges is a configuration of four switches (typically transistors or MOSFETs) that allow for the control of motor rotation in both directions. H-Bridges enable the motor to go forward, backward, brake, and let the motor coast.

7.4 Motor Driver Block diagram 3

The microcontroller sends control signals to manage motor operation, including speed and direction. Optocoupler provides electrical isolation between the controller and the motor driver circuit. This protects the controller from voltage spikes and allows for safe signal transmission. MOSFET Driver is a dedicated driver circuit that accepts low-power control signals from the optocoupler and translates them into high-current gate signals required to switch the MOSFETs. MOSFETs are used as the switching elements in the half H-bridge configuration. They control the current flow to the motor, allowing for speed control through PWM and directional control.

Design Considerations: The half H-bridge requires two MOSFETs for each motor terminal. One MOSFET connects the terminal to the power supply, while the other connects it to ground. For

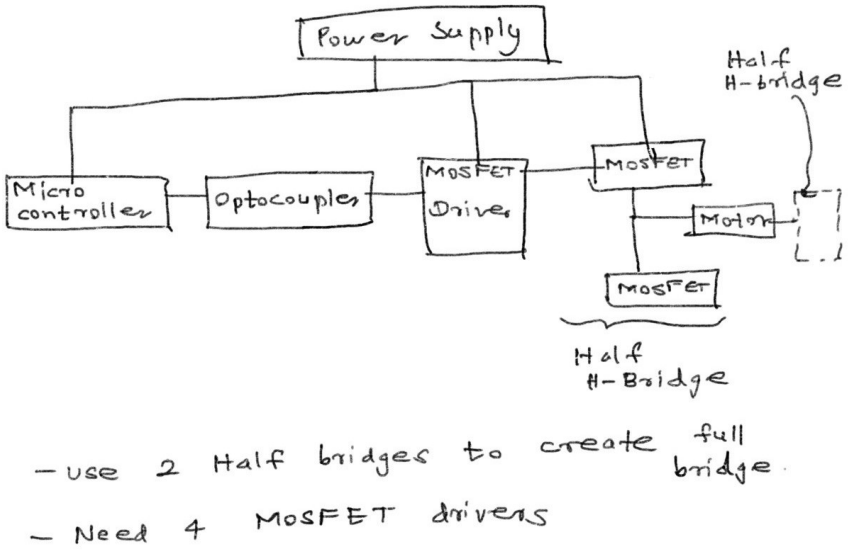


Figure 7.3: Alternative 2

bidirectional control of two motors (as typically used in differential drive robots),we will need two half H-bridges, thus a total of 8 MOSFETs.

7.5 Electronic Design Complete Comparison

	Electronic Design 1	**Electronic Design 2**	**Electronic Design 3**
Less Complexity of Circuit	8	6	5
Integration with Control System	8	8	7
Scalability	7	8	8
Protection Features	8	7	6
Precision of Control	8	7	7
Low Heat Dissipation	5	5	4
Space Efficiency	7	5	4
Low Cost	7	6	5
Power Efficiency	8	8	6
Total	**66**	**60**	**52**

Chapter 8

Mechanical Design - Robot Frame

8.1 Introduction

The mechanical design of an Autonomous Mobile Robot (AMR) plays a pivotal role in ensuring the robustness, functionality, and efficiency of the overall system. Through a series of rigorous analyses and iterative design cycles, our team meticulously crafted and evaluated several sketches to identify the optimal configuration.It's imperative to note that while we've chosen a preliminary design for the AMR frame, this iteration is not yet finalized.Key components such as motor specifications and battery sizes remain to be determined, and their integration will significantly influence the final design. This interim design represents a foundational step in our iterative design process, providing a starting point for further refinement and optimization as we progress towards a comprehensive solution.

8.2 Hand Drawn Sketch 1

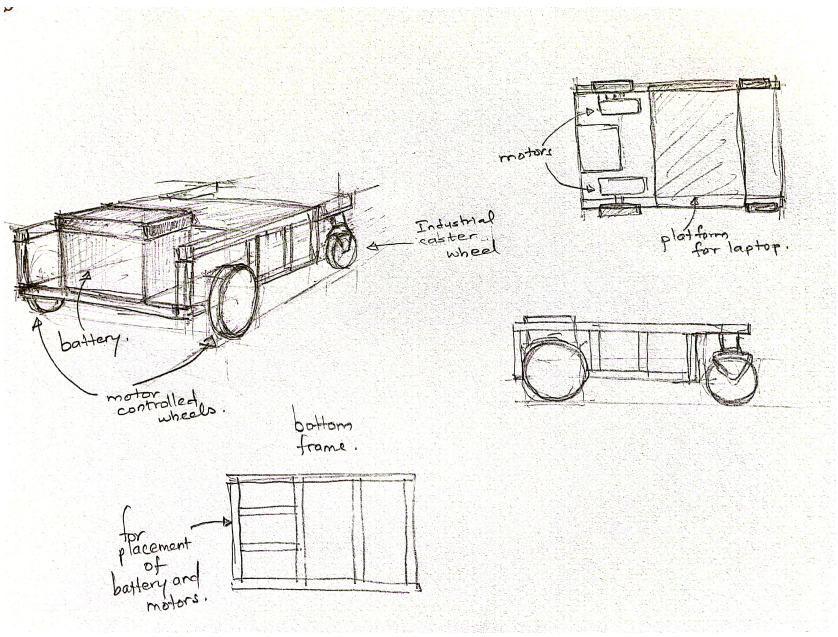


Figure 8.1: Assembly drawing of the system

Specifications for Motion Components(to be decided)

1. Industrial Caster Wheels:
   (a) Diameter: Typically ranging from 100mm to 150mm, depending on load capacity and maneuverability requirements.
   (b) Load Capacity: Each caster wheel should support a minimum load capacity(to be decided) to ensure stability and smooth operation.

(c) Mounting: Bolt or screw mounting mechanism compatible with the frame structure, ensuring secure attachment and ease of maintenance.

2. Motor Driven Wheels:

   (a) Motor Type: DC geared motors selected for their torque, speed, and efficiency characteristics suitable for AMR applications.

   (b) Integration: Direct integration with the motor shaft or through a coupling mechanism for efficient power transmission.

   (c) Wheel Diameter: Matching the diameter of the caster wheels and considering RPM of motors to ensure uniformity in motion and clearance.

   (d) Encoders (Optional): Provision for encoders to enable precise control of wheel movement and feedback for navigation algorithms, enhancing the AMR's accuracy and autonomy.(if not available in motor itself)

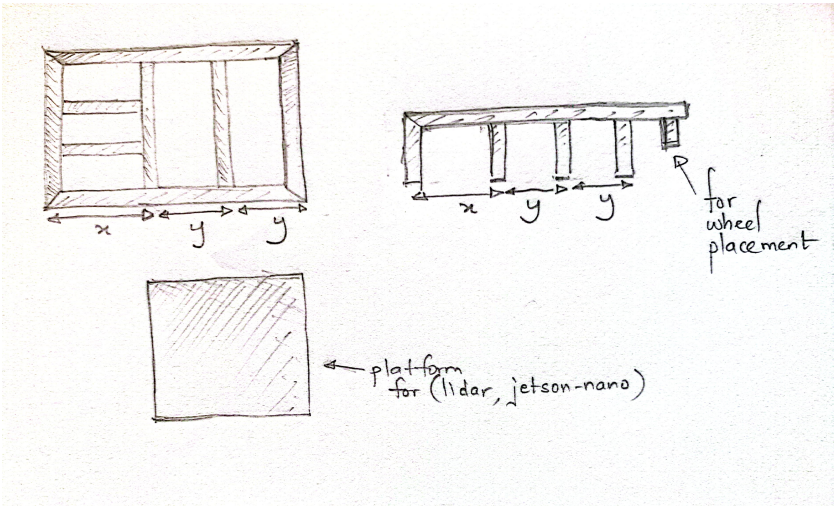


Figure 8.2: Drawing of pieces

8.3 Hand Drawn Sketch 2

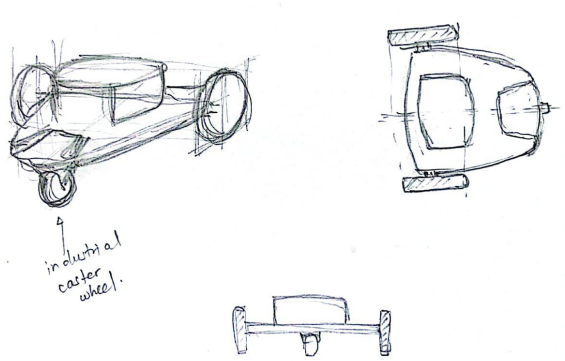


Figure 8.3: Alternative 1

For the three-wheeled design featuring a single caster wheel at the front and two motor-driven wheels at the rear, stability considerations played a pivotal role in our decision-making process. While this configuration offers simplicity and agility in maneuverability, it lacks the inherent stability provided by a four-wheeled setup. The presence of only one caster wheel at the front can result in potential instability, especially when navigating uneven terrain or encountering sudden changes in direction. This design may exhibit a tendency to tip forward or sideways under certain operating conditions, compromising both safety and operational efficiency. Consequently, to prioritize stability and ensure robust performance across diverse environments, we opted for the four-wheeled configuration, leveraging the added balance and control afforded by dual caster wheels at the front and motor-driven wheels at the rear.

8.4 Hand Drawn Sketch 3

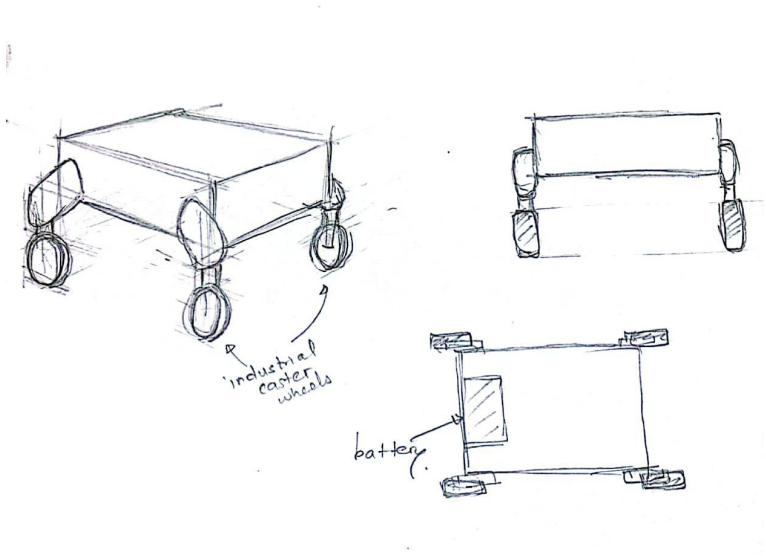


Figure 8.4: Alternative 2

0In the alternative design featuring a four-wheeled configuration with interconnected wheels using joints or limbs to achieve higher ground clearance, several considerations influenced our decision-making process. While this setup offers the advantage of increased ground clearance, crucial for navigating uneven terrain and obstacles effectively, it presents challenges in motor integration. The interconnected nature of the wheels complicates the direct attachment of motors, necessitating the implementation of intermediary mechanisms such as belts or chains for power transmission. This introduces added complexity to the design, requiring careful engineering and maintenance to ensure reliable operation. Additionally, the use of belts or chains may introduce potential points of failure and increased susceptibility to wear over time, posing reliability concerns in prolonged operational scenarios. Given these challenges and the desire for a streamlined, robust design, we ultimately opted for a simpler configuration with direct motor-driven wheels, prioritizing ease of integration, maintenance, and long-term reliability for optimal performance of the AMR in varied environments.

8.5 Mechanical Design Complete Comparison

	Mechanical Design 1	**Mechanical Design 2**	**Mechanical Design 3**
Balancing & Stability	8	5	9
Payload Capacity	8	5	8
Weight & Size	8	9	6
Cost Effectiveness	8	9	4
Integration of Components	9	7	5
Accessibility for Maintenance	7	8	5
Scalability	8	6	6
Total	**56**	**49**	**43**

Chapter 9

Software Design

9.1 Introduction

This section encapsulates the details of the software architecture and the specific components that have been developed and integrated to design the Autonomous Mobile Robot (AMR) with the capabilities required for autonomous navigation and environmental interaction. At the core of this architecture is the utilization of ROS2, which provides a robust framework for robot software development, and SLAM (Simultaneous Localization and Mapping), which is essential for the robot to understand and navigate through its operational environment.

9.2 Software Flowchart

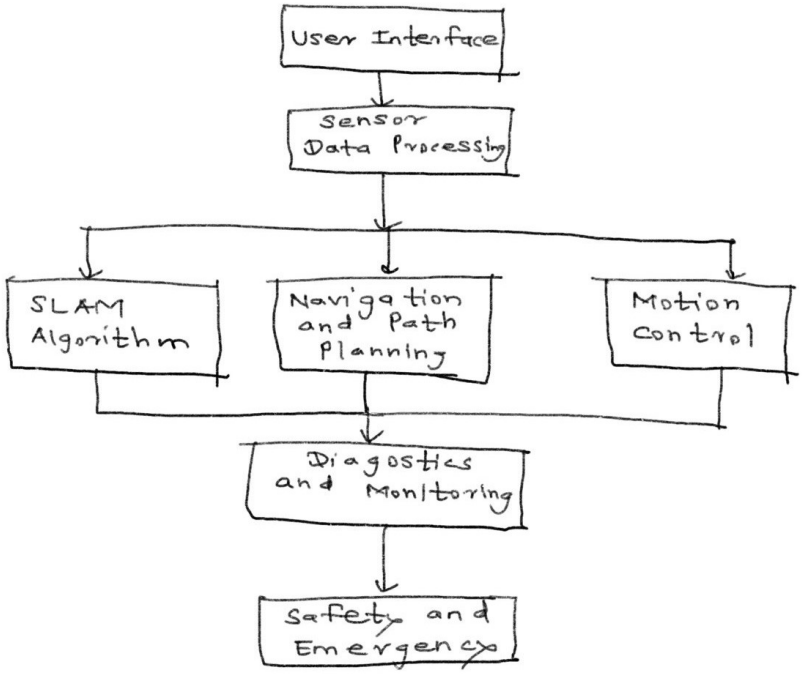


Figure 9.1: Software Flow chart

The software architecture flowchart illustrates the logical sequence and interaction between various components of the AMR's software system. It maps out the process flow from user inputs to robot movement and monitoring, detailing the high-level functionalities and decision-making pathways that govern the AMR's operations.

1. User Interface serves as the initial point of interaction where users can input commands, set navigation goals, and receive feedback from the robot system.

2. Sensor Data Processing acts as the data aggregation center, receiving LiDAR and visual data, and processing this information to produce a perception of the environment.
3. SLAM Algorithm is responsible for creating a map of the environment and determining the robot's position within that map. It receives processed sensor data and uses it to update the map and the robot's location in real-time.
4. Navigation and Path Planning component utilizes the map created by the SLAM algorithm to plan a path from the robot's current location to the desired destination. This module continually updates the path as new environmental data is received.
5. Motion Control takes the planned path and translates it into motor commands that control the robot's speed and direction. This module ensures that the robot follows the path efficiently and adjusts to any changes or obstacles.
6. Diagnostics and Monitoring constantly assesses the robot's system health, including battery levels, sensor status, and motor function. This module is vital for maintaining operational integrity and preventing system failures.

9.3 Software Block Diagram

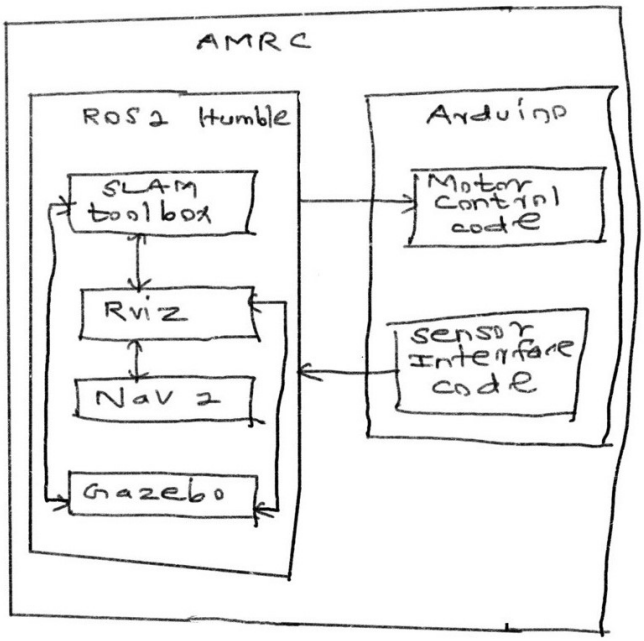


Figure 9.2: Software Block Diagram

9.3.1 Description of Each Block

ROS2 Humble

1. SLAM toolbox: This block includes algorithms for Simultaneous Localization and Mapping (SLAM), which help the robot build a map of the environment while keeping track of its current position.

2. Nav2 Stack: This includes path planning, obstacle avoidance, and other navigation-related functionalities.
3. Rviz: A 3D visualization tool for ROS2 that helps in visualizing the robot's sensor data, state, and the environment.
4. Gazebo: A simulation tool used for testing robot models and algorithms in a virtual environment.
5. Teleop Keyboard: Allows manual control of the robot using a keyboard for testing and teleoperation purposes.

Arduino

1. Motor Control Code: This includes PWM signals, I2C communications, and other control algorithms to drive the robot's motors.
2. Sensor Interface Code: Manages the interface with various sensors (e.g., distance sensors, IMUs) using analog and digital signals.

9.3.2 Workflow

The ROS2 Humble framework handles SLAM, navigation, and overall coordination between the different components. The SLAM algorithm (SLAM toolbox) processes sensor data to create a map and localize the robot within it.The navigation stack (Nav2) uses the map and localization data to plan paths and avoid obstacles, sending commands to the Arduino for motor control.

The Arduino receives motor commands from ROS2 and translates them into appropriate signals to control the motors. It also reads data from sensors and sends it back to ROS2 for processing by the SLAM and navigation algorithms.

Gazebo provides a simulated environment for testing the robot's algorithms without needing a physical robot. Rviz allows for real-time visualization of the robot's state, the map, sensor data, and navigation paths.

The teleop keyboard node in ROS2 enables manual control of the robot for testing purposes, sending direct movement commands to the Arduino. This software block diagram encapsulates the core components and their interactions in the AMRC project, facilitating a modular and flexible design approach.

9.3.3 Modeling Environment and Simulation using Gazebo

Gazebo installation: Instructions for installation on Ubuntu are available on the Gazebo website.
Gazebo installation guide

Capture Real Environment Measurements

Use tape measures to gather precise measurements of the real environment. These measurements should include dimensions, distances between objects, and any relevant features like walls, furniture, or obstacles.

Model the Environment in Gazebo

Based on the measurements obtained, create a 3D model of the environment in Gazebo Building Editor. We can use Gazebo's built-in tools or import existing 3D models to represent the walls, floors, objects, and obstacles within the environment.
Gazebo Building Editor guide

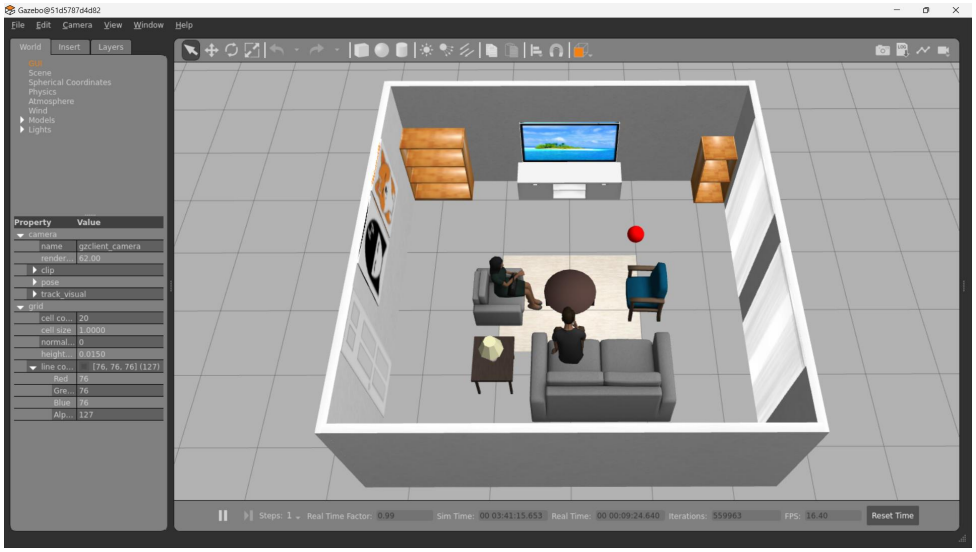


Figure 9.3: Gazebo Modeling

Integrate Simulated LiDAR Sensor

Add a simulated LiDAR sensor to our robot model within Gazebo. Configure the LiDAR sensor's parameters such as range, field of view, and scan rate to match those of the real LiDAR sensor that will be used on the robot. Position the LiDAR sensor appropriately on the robot model to mimic its real-world placement.
Sensors guide

Generate Point Cloud from Simulated LiDAR

Simulate the operation of the LiDAR sensor within Gazebo. Emit virtual laser beams from the LiDAR sensor and calculate the points of intersection with the surfaces of objects in the environment. As the LiDAR sensor scans the environment, Collect the intersection points to form a point cloud representation of the environment from the robot's perspective.

Validate the Point Cloud

Visualize the generated point cloud within Gazebo to ensure that it accurately represents the features and geometry of the environment. Compare the simulated point cloud with the actual measurements of the real environment to verify its accuracy. Adjust the virtual environment or LiDAR sensor parameters as needed to improve alignment between the simulated and real data.

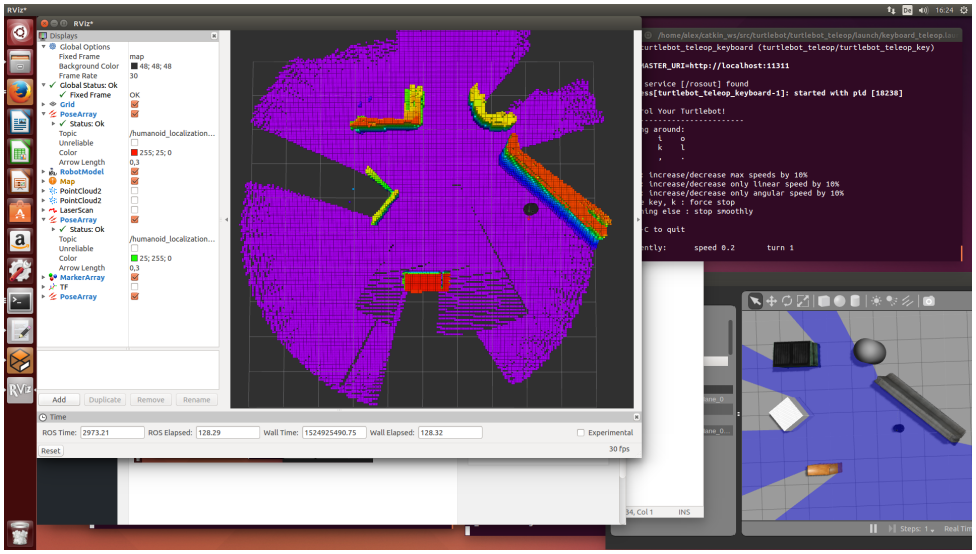


Figure 9.4: Simulated LiDAR Sensor - source

9.4 Alternatives

9.4.1 SLAM Algorithm: Hector SLAM

Hector SLAM was selected for its unique attributes, which make it particularly well-suited for our AMR without the need for odometry data. Known for its efficiency on platforms with limited computational resources, Hector SLAM excels in providing real-time mapping capabilities. It employs a scan matching-based approach, making it an excellent choice for environments with stable and distinct geometrical features. Hector SLAM's ability to operate without additional sensory input, relying solely on the data from a LiDAR sensor, simplifies the system's sensor suite and reduces potential points of failure.

9.4.2 Simulation and Data Generation: MATLAB

For the simulation of LiDAR data and testing of the SLAM algorithm, MATLAB was considered due to its extensive suite of toolboxes and its powerful graphical capabilities. MATLAB's simulation environment enables the generation of synthetic LiDAR sensor data by replicating the sensor's physical characteristics and the environmental interactions of its laser beams. This simulated data is then utilized as input for the Hector SLAM algorithm, providing a controlled setting for optimizing the SLAM parameters and algorithms before deployment in the physical robot.

The rigorous simulation also allows for the evaluation of Hector SLAM's performance in various simulated scenarios, providing insights into the algorithm's robustness and accuracy. By analyzing how the algorithm processes and maps the data, engineers can refine the SLAM implementation, ensuring that the robot can reliably construct and update its internal representation of the operational environment.

The integration of Hector SLAM within the ROS2 framework on the NVIDIA Jetson Nano provides a seamless workflow from data acquisition to actionable navigation commands. The Jetson Nano serves as the central processing unit, interpreting the processed SLAM data to determine the robot's location and to make informed navigation decisions.

9.4.3 Modeling the Environment : Blender

While Gazebo offers robust tools for environment modeling and simulation, Blender presents an alternative approach for creating detailed environments with intricate features. With Blender's powerful 3D modeling capabilities, we can craft realistic scenes comprising various elements such as terrain, buildings, vegetation, and objects. Blender's extensive toolset allows for precise manipulation of geometry, textures, and lighting, enabling the creation of immersive environments that closely resemble real-world settings.

The modeling process in Blender is similar to that in Gazebo. In Blender, We can import pre-existing 3D models, granting me access to a wide array of ready-made assets that I can incorporate into the scenes effortlessly. Additionally, Blender supports various file formats, allowing for seamless integration with other software tools and workflows.

Chapter 10

Final schematic and PCB design

We have designed a PCB layout of motor driver for the motors given by the professor, based on the selected conceptual design.

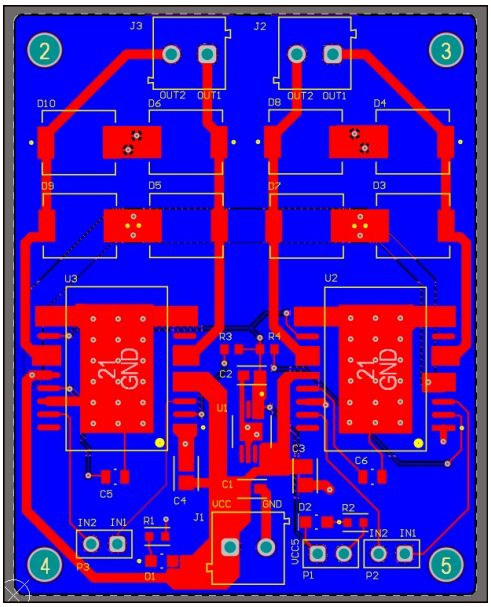


Figure 10.1: PCB

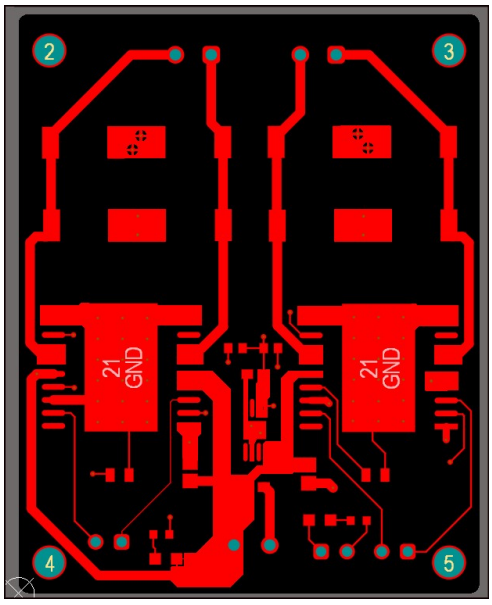


Figure 10.2: Top layer

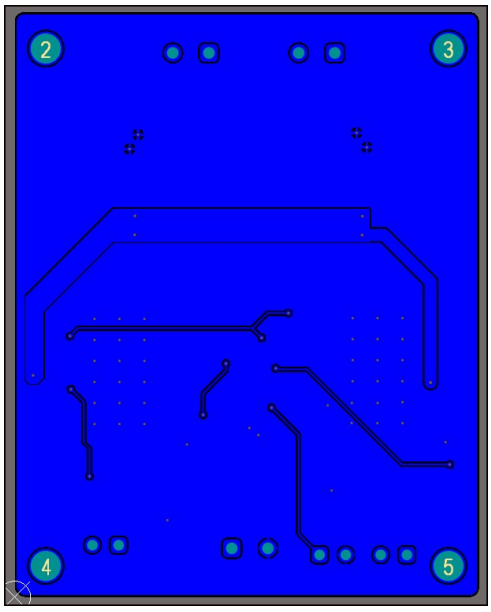


Figure 10.3: Bottom layer

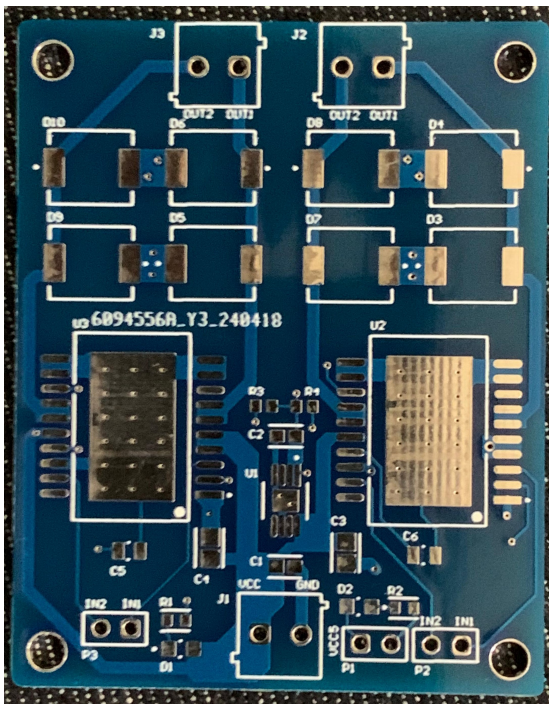


Figure 10.4: Bare PCB Front View

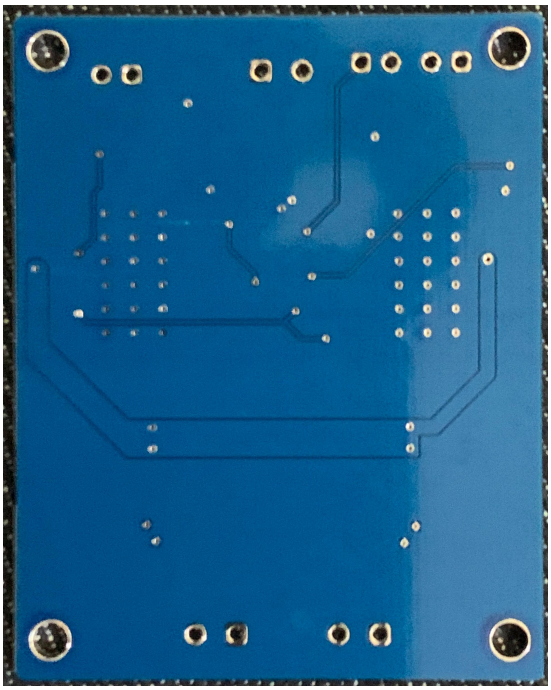


Figure 10.5: Bare PCB Rear View

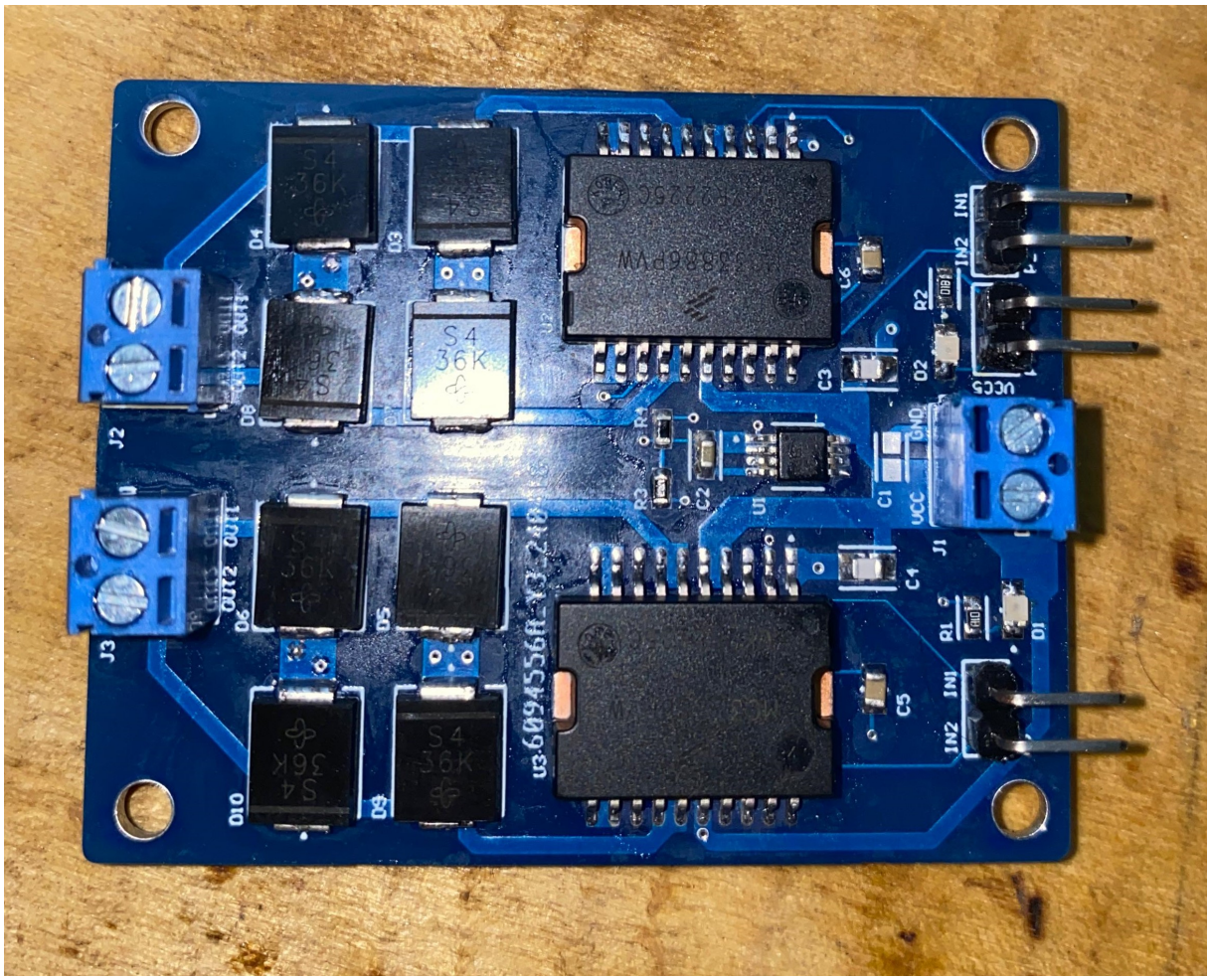


Figure 10.6: Soldered PCB

Chapter 11

Solidworks Design

11.1 Enclosure for PCB

An enclosure has been meticulously designed to accommodate both the motor driver and microcontroller, ensuring optimal functionality and longevity of the components. The design incorporates strategically placed air ventilations to facilitate efficient cooling, thus maintaining the operating temperatures within recommended limits.

11.1.1 Enclosure Top Part

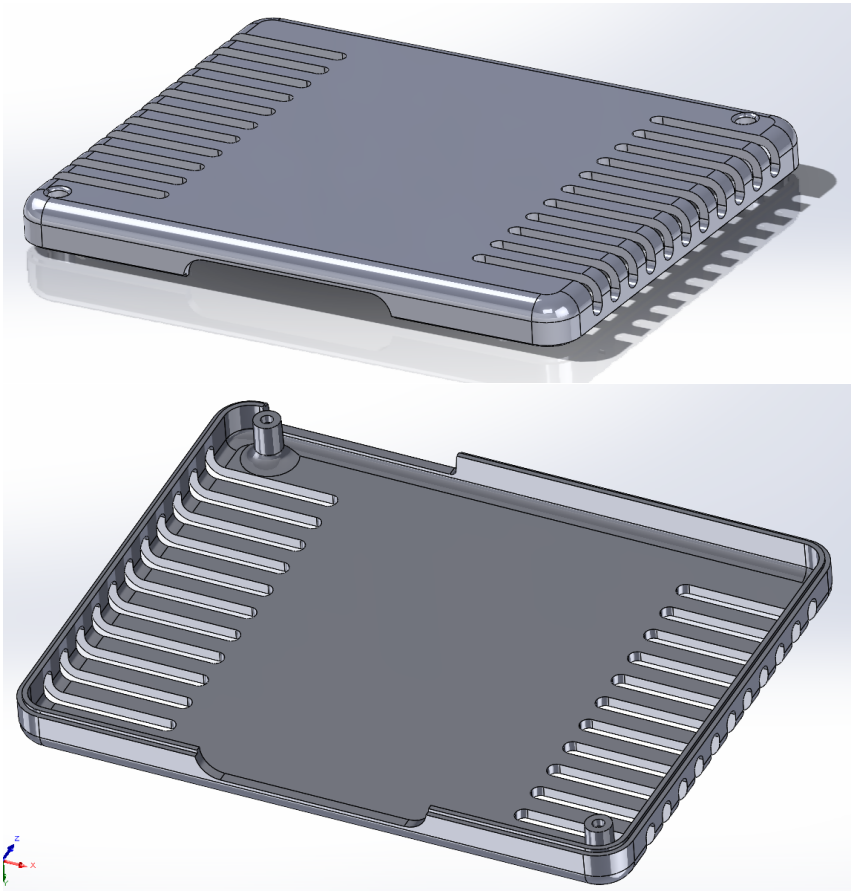


Figure 11.1: Enclosure Top Part

11.1.2 Enclosure Bottom Part

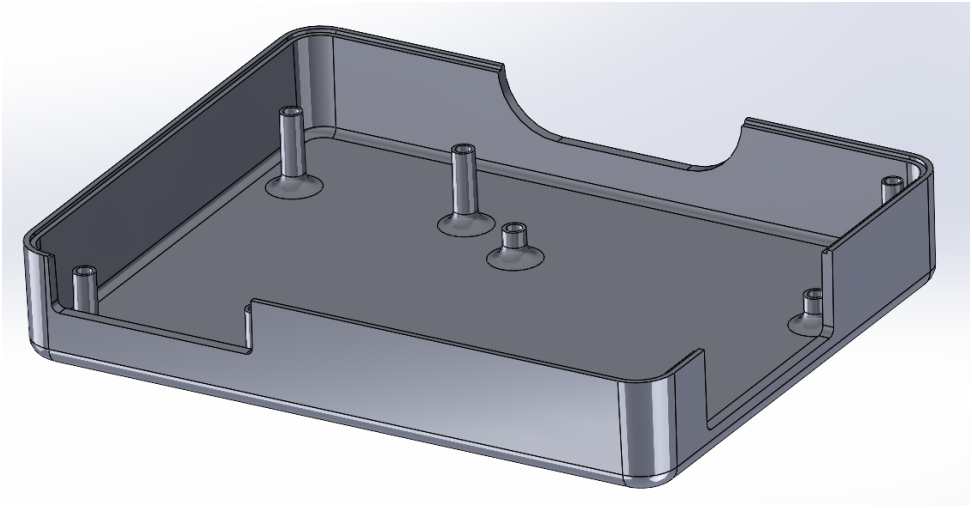


Figure 11.2: Enclosure Bottom Part

11.1.3 Enclosure Assembly

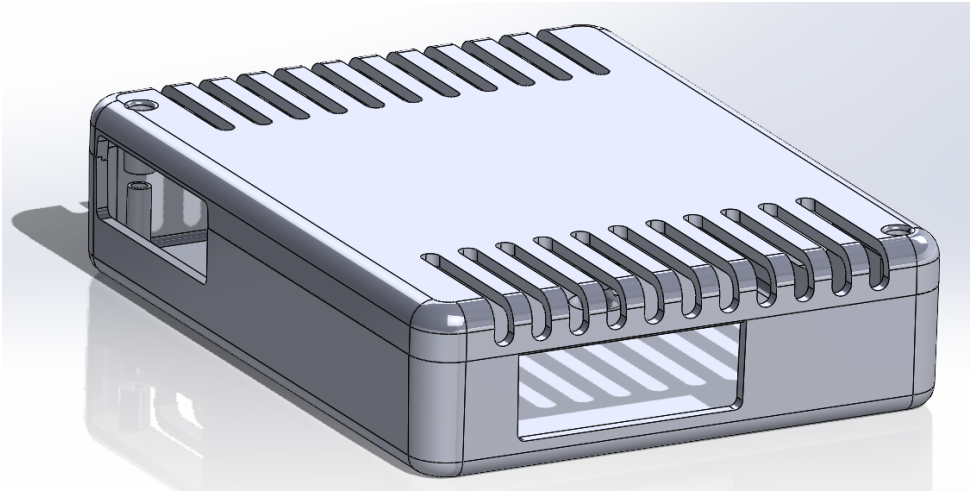


Figure 11.3: Enclosure Assembly

Chapter 12

Final Solidworks Design

12.1 Robot Testing Platform

To test the controller, we employed a design methodology that involved creating a platform using welded components in SolidWorks. Our approach prioritized factors such as stability, weight distribution, and compactness. Through detailed modeling and simulation in SolidWorks, we ensured that the platform met these criteria effectively. This methodology allowed us to refine the design iteratively, ensuring optimal performance for testing the controller.

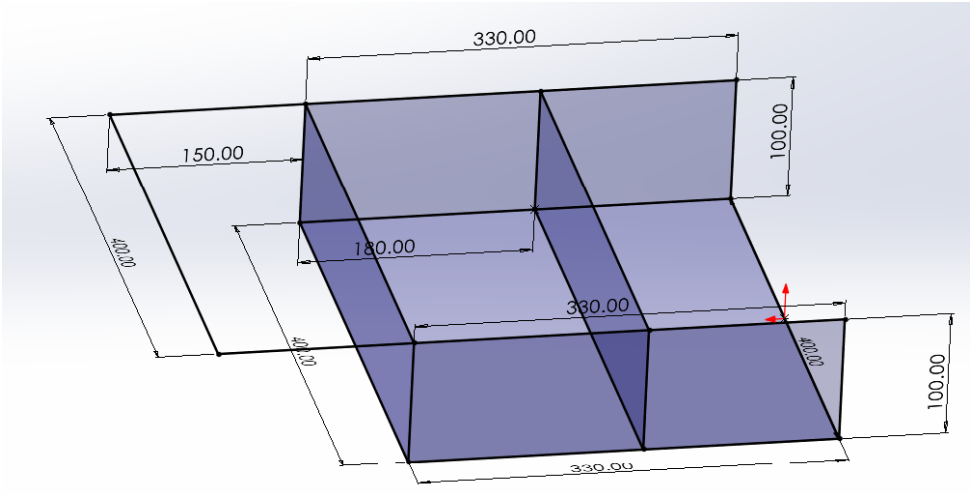


Figure 12.1: Frame Sketch

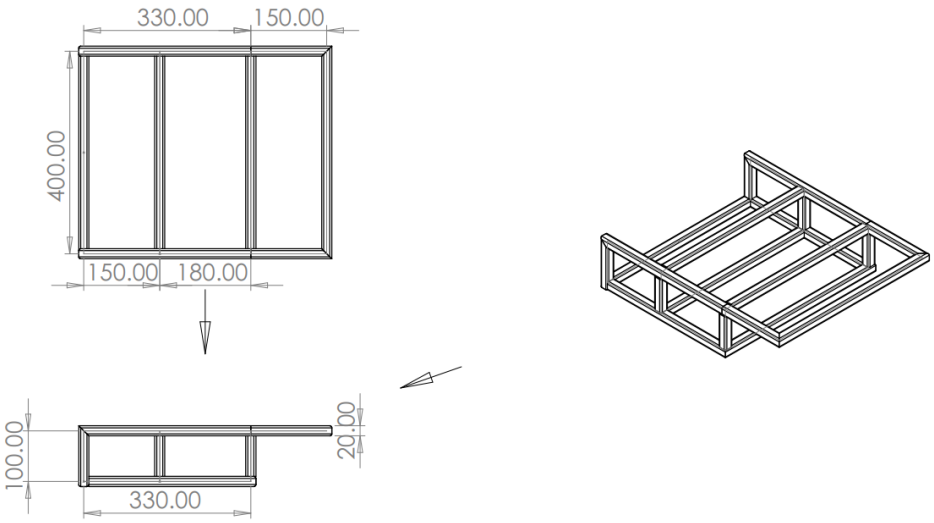


Figure 12.2: Frame Drawing

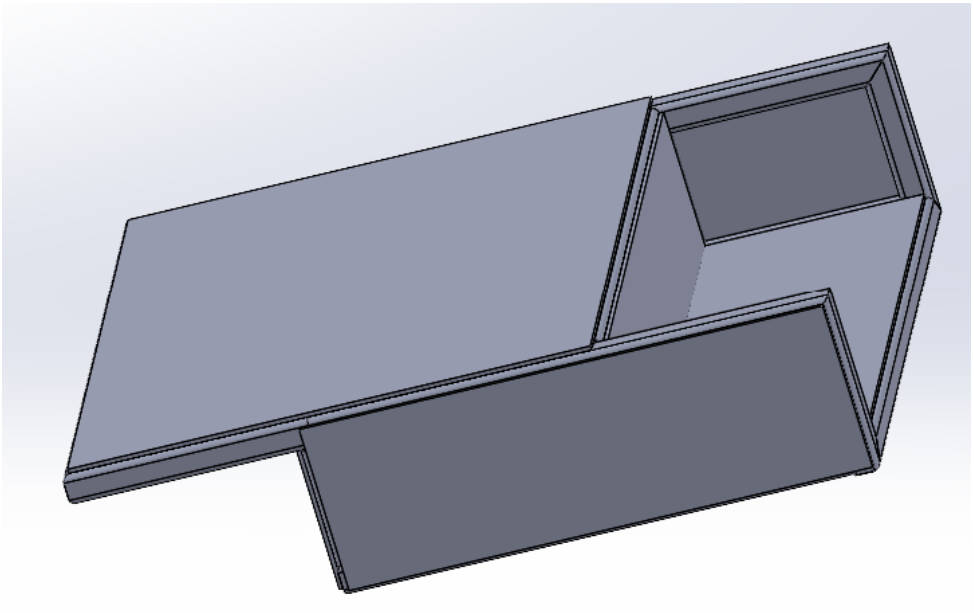


Figure 12.3: Frame View

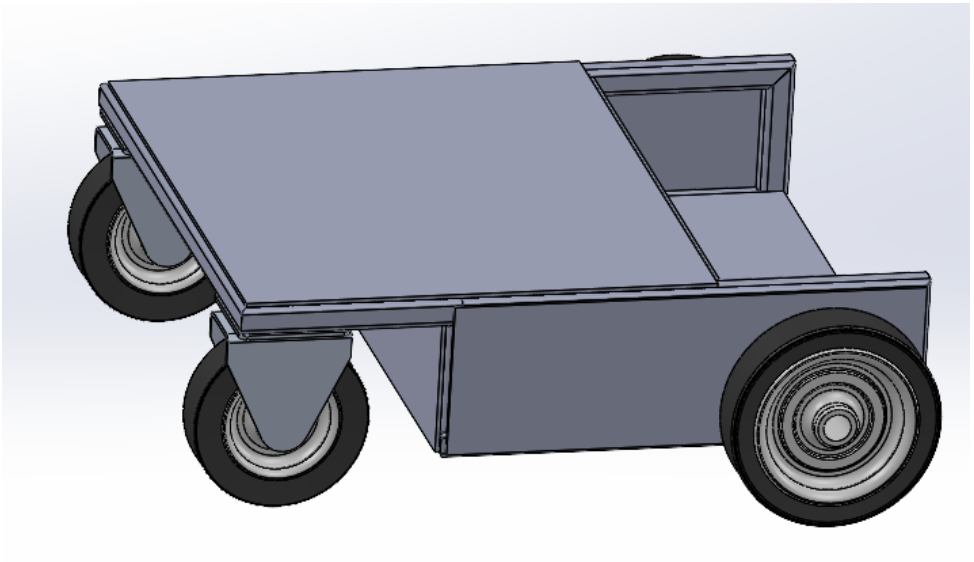


Figure 12.4: Frame Assembly

12.2 Finalised Enclosure for PCB

We revised our initial enclosure design for several reasons, including optimizing component placement for better thermal management, enhancing airflow through improved ventilation, and ensuring compatibility with updated hardware specifications. Adjustments were made to improve structural integrity and ease of mass production and assembly, resulting in a more efficient and reliable enclosure that meets all functional requirements.

12.2.1 Enclosure Drawings

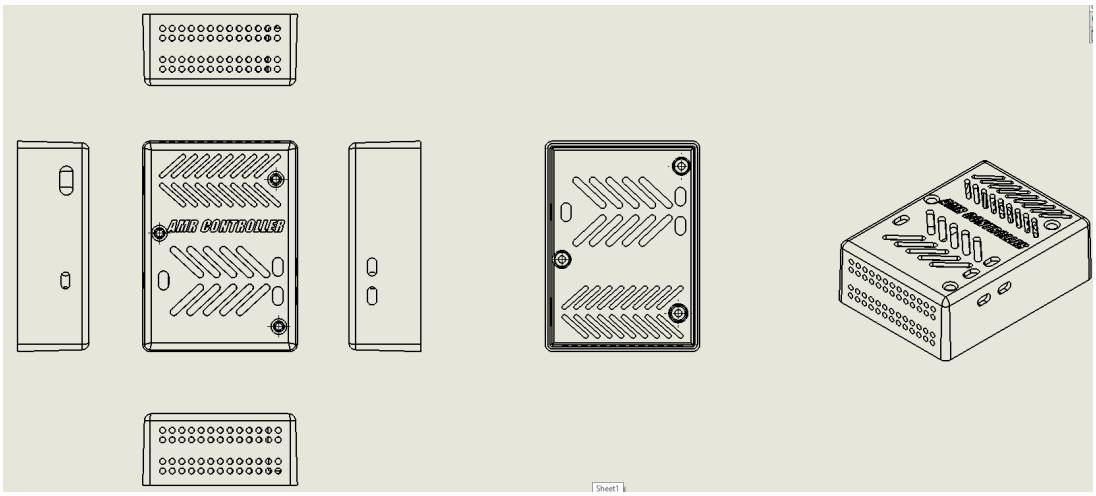


Figure 12.5: Top Enclosure Drawing

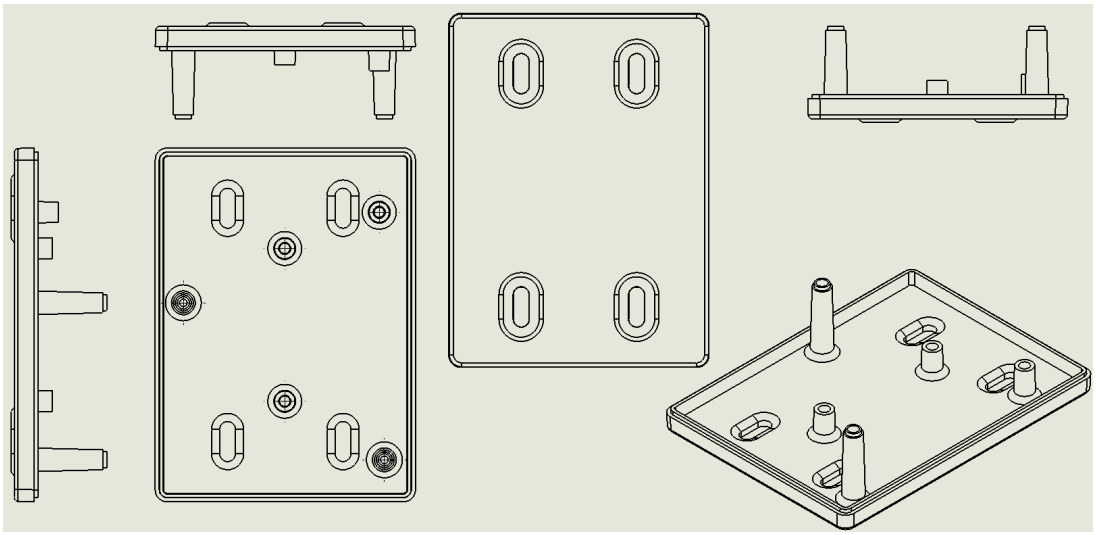
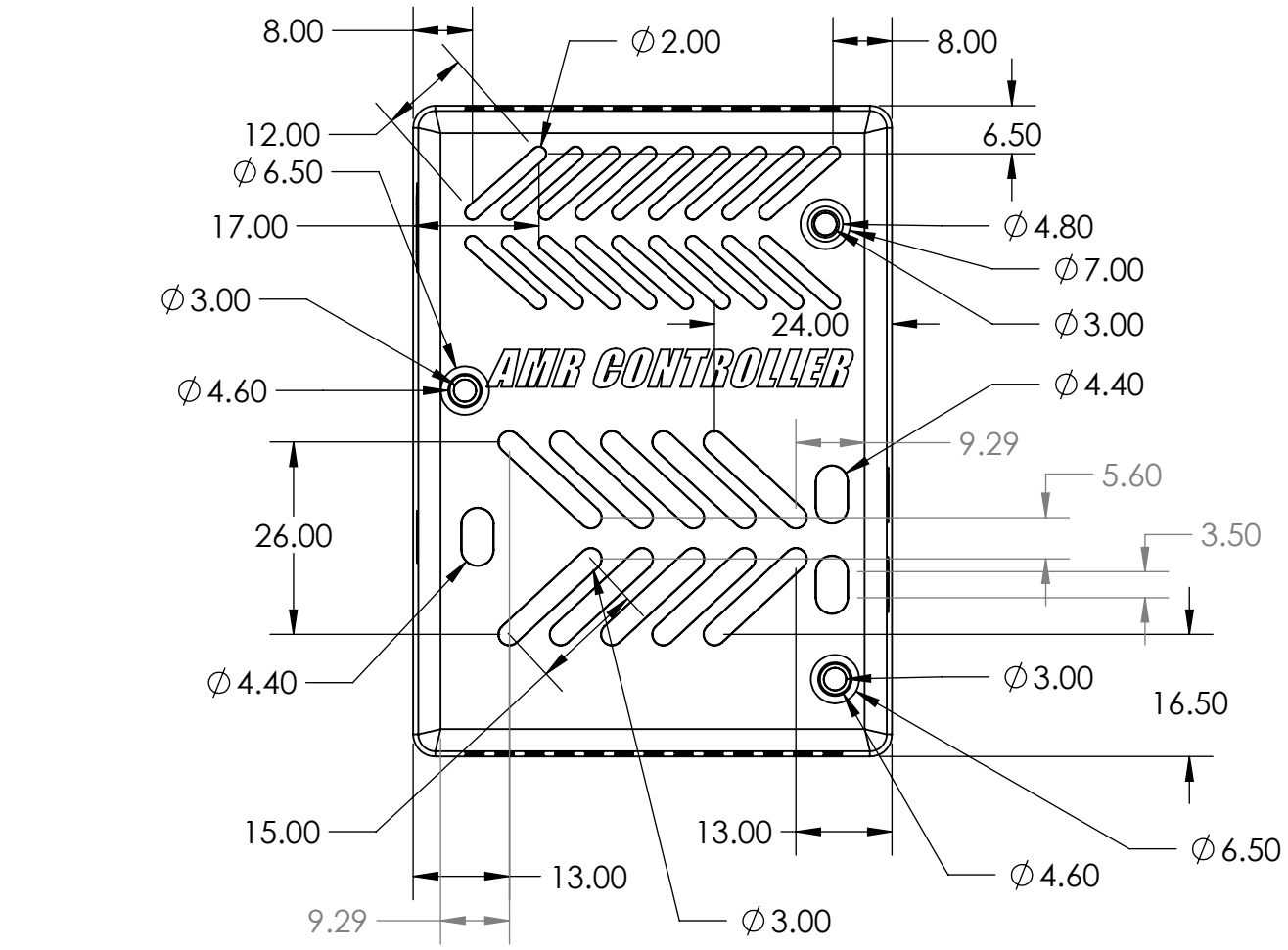
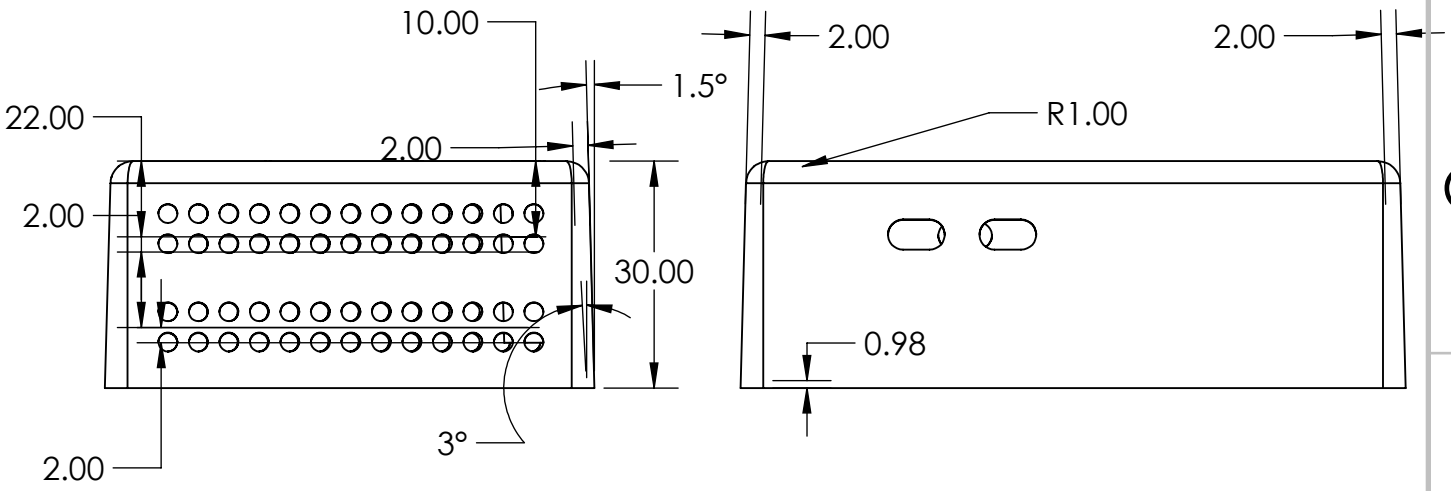
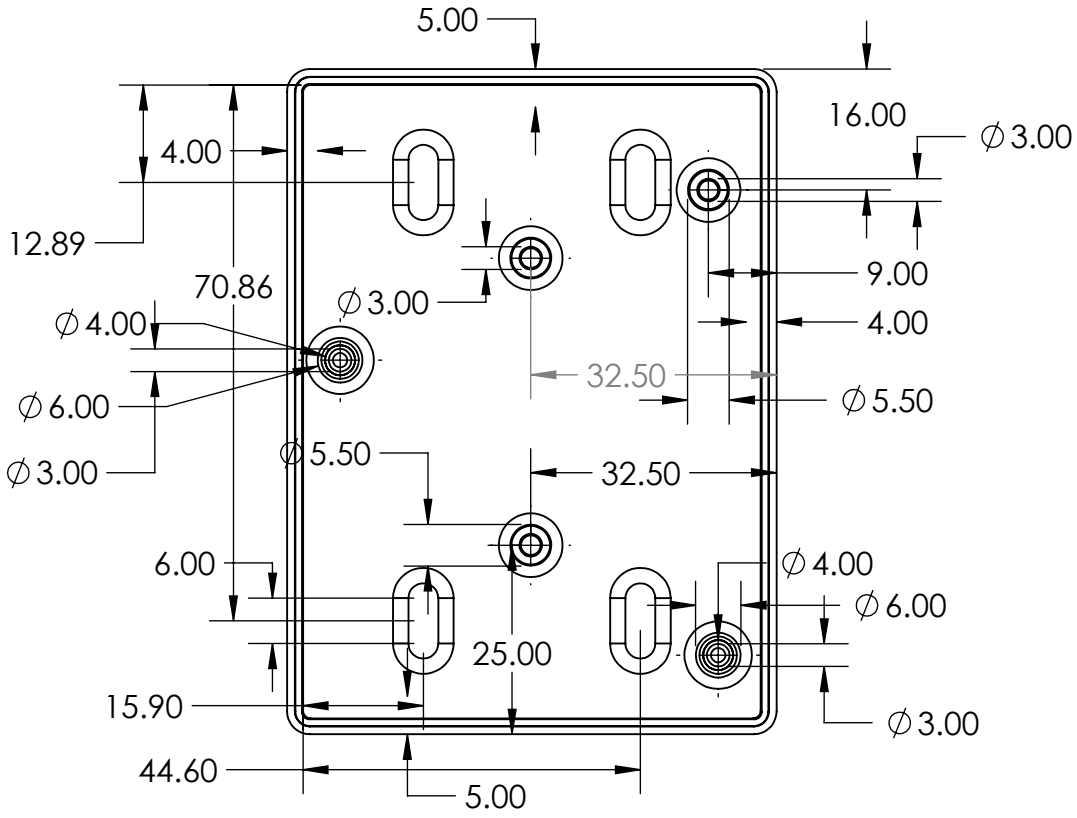


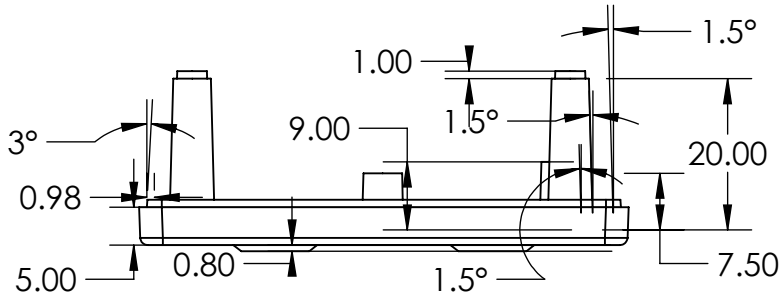
Figure 12.6: Bottom Enclosure Drawing

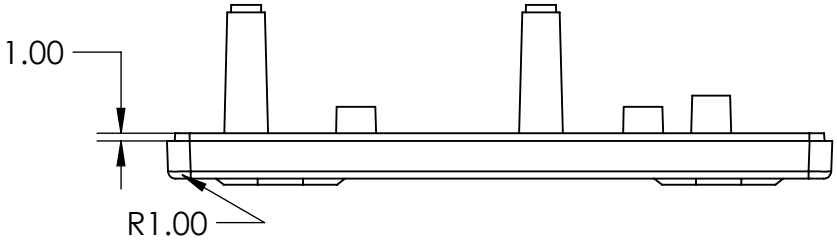




UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH: 06/07/2024		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE				TITLE:
DRAWN	Abithan A.		04/05/2024				Top Part
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO.	A4
				ABS		01	
				WEIGHT:		SCALE: 1:1	SHEET 1 OF 1







UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	REVISION
		06/07/2024					
	NAME	SIGNATURE	DATE			TITLE:	
DRAWN	Abithan A.		04/05/2024			Bottom Part	
CHK'D							
APPV'D							
MFG							
Q.A				MATERIAL:		DWG NO.	A4
				ABS		02	
				WEIGHT:		SCALE: 1:1	SHEET 1 OF 1

12.2.2 Enclosure 3D Model

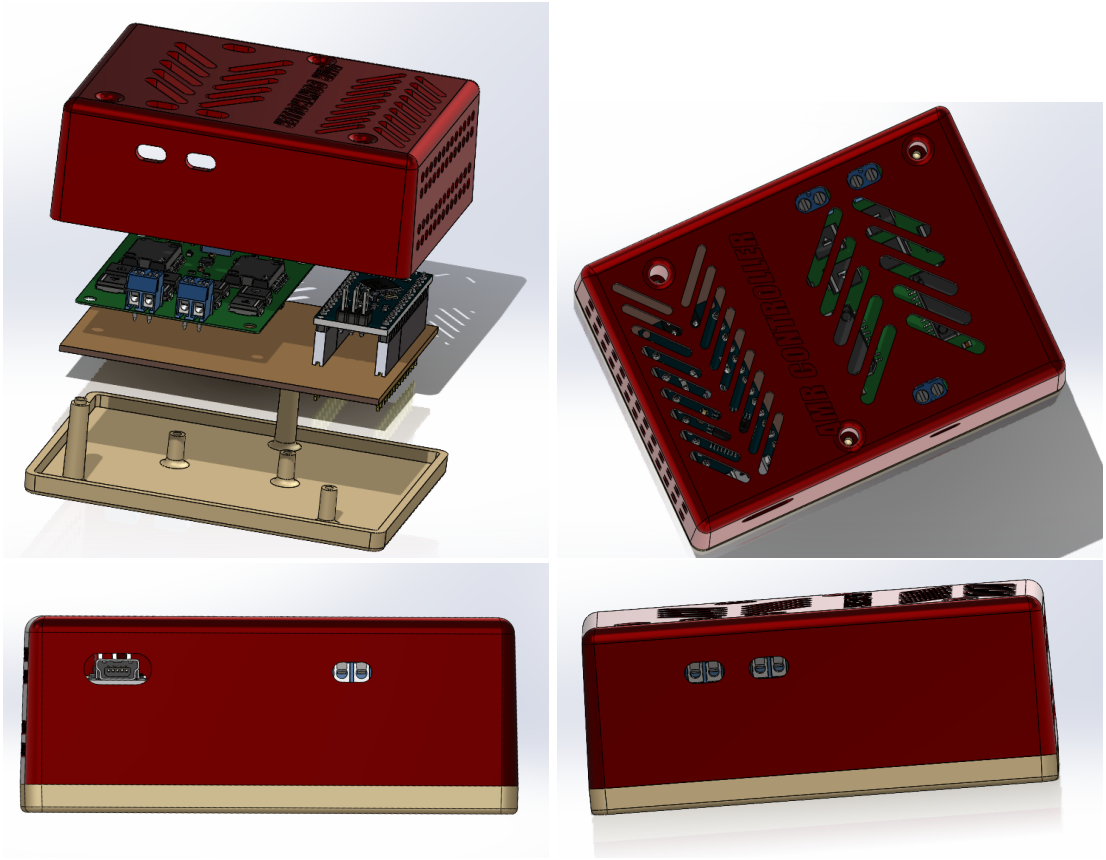


Figure 12.7: Motor Driver Enclosure Design

12.3 Mold Designs

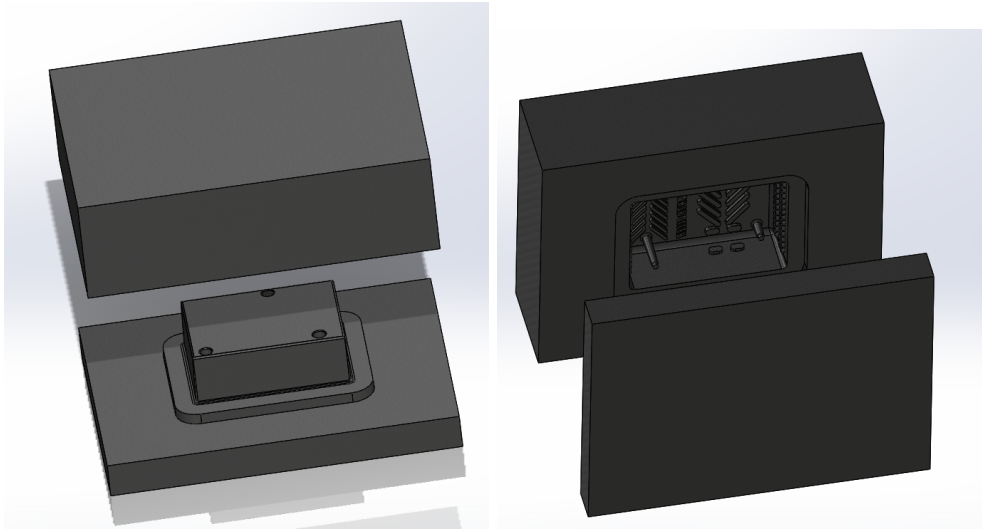


Figure 12.8: Top Enclosure Mold Design

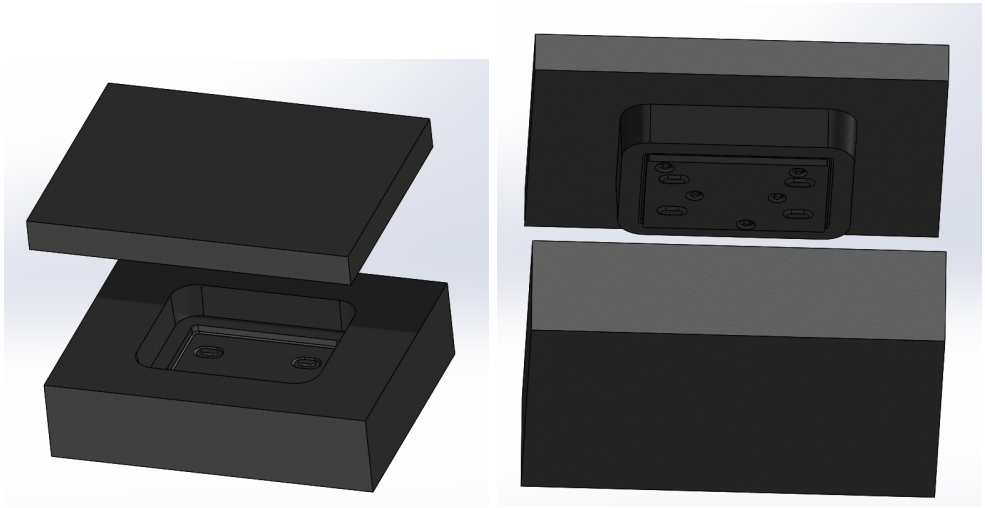


Figure 12.9: Bottom Enclosure Mold Design

12.4 Printed Enclosure

Figure 12.10: Printed Enclosure Interior view

Figure 12.11: Printed Enclosure Exterior view

Chapter 13

Data sheets and References

1. SEER Robotics : https://www.seer-group.com/
2. KGEC-6200 Edge Controller : https://www.jhc-technology.com/amr_controller_kgec-6200_launch
3. L298 IC Datasheet : https://www.st.com/resource/en/datasheet/l298.pdf
4. Jetson Nano : https://developer.nvidia.com/embedded/downloads
5. ROS2 Humble distributuion : https://docs.ros.org/en/humble/index.html
6. Gazebo : https://gazebosim.org/home
7. Simulating LiDAR in Gazebo : https://grauonline.de/wordpress/?page_id=2769
8. Arduino IDE for Linux: https://docs.arduino.cc/software/ide-v1/tutorials/Linux/
9. ROS2 Creating Workspace: https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html
10. Project Repository: https://github.com/Bavan2002/AMR
11. ROS Arduino Bridge code: https://github.com/Bavan2002/ros_arduino_bridge.git