AUTOMED - AUTONOMOUS MEDICAL DELIVERY ROBOT

The Project Phase II report submitted in fulfillment
of the requirements for the award of the Degree of
Bachelor of Technology
in
Computer Science and Engineering
under the
APJ Abdul Kalam Technological University

by

SREEJITH A
(MGP21UCS137)

SAINTGITS COLLEGE OF ENGINEERING(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KOTTAYAM, KERALA (INDIA)

May 2025

SAINTGITS COLLEGE OF ENGINEERING(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KOTTAYAM, KERALA (INDIA)

BONAFIDE CERTIFICATE

This is to certify that the project entitled "**AUTOMED - Autonomous Medical Delivery Robot**" submitted by **Sreejith A (MGP21UCS137)** for the award of the **Bachelor of Technology** in **Computer Science and Engineering** is a bonafide record of the Project Phase II carried out by them under my guidance and supervision at "**Saintgits College of Engineering (Autonomous)**".

Er. Gayathri J L
Assistant Professor
(Project Advisor)

Dr. Arun Madhu
Head of Department

SEAL

SAINTGITS COLLEGE OF ENGINEERING(AUTONOMOUS)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KOTTAYAM, KERALA (INDIA)

DECLARATION

I, **Sreejith A (MGP21UCS137)** hereby declare that this project report entitled "**AUTOMED - Autonomous Medical Delivery Robot**" is the record of the original work done by me under the guidance of **Er.Gayathri J L**, Assistant Professor, Department of Computer Science and Engineering, Saintgits College of Engineering. To the best of my knowledge, this work has not formed the basis for the award of any degree/diploma/fellowship or a similar award to any candidate in any University.

Place: **Signature of the Student**
Date:

COUNTERSIGNED

Er.Gayathri J L
Assistant Professor
(Project Advisor)

Dr.Anju Pratap
Er. Safad Ismail
(Project Coordinator(s))

Contents

Acknowledgements

I express my gratitude to our principal, **Dr. Sudha T**, Principal, Saintgits College of Engineering (Autonomous), for providing me with an excellent ambiance that laid a strong foundation for my work.

I extend my heartfelt thanks to **Dr. Arun Madhu**, Head of the Department of Computer Science and Engineering at Saintgits College of Engineering (Autonomous), who has been a constant support in every step of my project and a source of strength in completing this phase.

I express my sincere thanks to our Project Coordinators **Dr.Anju Pratap**, Professor and **Er. Safad Ismail**, Assistant Professor in the Computer Science and Engineering Department, for providing me with all the necessary facilities, valuable and timely suggestions, and constant supervision that contributed to the successful completion of this phase of my project.

I also thank **Er. Gayathri J L**, Assistant Professor in the Computer Science and Engineering Department, for her supervision, encouragement, help, and support.

I am highly indebted to all the faculty members of the department for their valuable guidance and prompt assistance. I extend my heartfelt thanks to my parents, friends, and well-wishers for their unwavering support and timely help.

Last but not least, I thank Almighty God for helping me successfully complete this phase of my final year project.

SREEJITH A

List of Figures

Abreviations

ROS	–	Robot Operating System
SLAM	–	Simultaneous Localization and Mapping
LiDAR	–	Light Detection and Ranging
Wi-Fi	–	Wireless Fidelity
IMU	–	Inertial Measurement Unit
RFID	–	Radio Frequency Identification
IoT	–	Internet of Things
EKF	–	Extended Kalman Filter
ANN	–	Artificial Neural Network
SARN	–	Socially Aware Robot Navigation
HIS	–	Hospital Information System
EHR	–	Electric Health Record
DWA	–	Dynamic Window Approach
RMDS	–	Robotic Medicine Delivery System

Abstract

Robotics is rapidly advancing, with autonomous navigation emerging as a pivotal area of innovation. This paper introduces an autonomous robotic solution designed to optimize payload delivery within indoor hospital environments, addressing challenges posed by dynamic obstacles like humans and static structures. By leveraging SLAM with LiDAR sensor, the robot performs real-time mapping and localization, enabling precise navigation and destination marking within the hospital environment. RFID technology is integrated not only for destination identification but also as a security measure to ensure that only authorized medical staff can access the payload. A camera module is employed for precise orientation upon reaching the destination, ensuring accurate docking and payload delivery alignment. Ultrasonic sensors, LIDAR, and motorized configurations contribute to smooth operation through effective obstacle detection and avoidance. Additionally, the system records destinations and monitors human activity to facilitate seamless navigation within the hospital. Traditional delivery tasks, often managed by human staff, can suffer inefficiencies due to fatigue, human errors, and breaks, resulting in increased costs and reduced productivity. This robotic solution overcomes these limitations by automating mapping, navigation, and delivery processes. Users can interact with the system via a mobile app, allowing for real-time path tracking, manual control, or autonomous navigation to specified destinations. Upon arrival, the robot completes payload delivery after user verification using a sliding mechanism. By integrating advanced technologies, this solution enhances efficiency, reliability, and security, representing a transformative approach to improving productivity in healthcare.

Chapter 1

Introduction

Hospitals and healthcare facilities face numerous challenges that impact efficiency, patient care, and overall operational effectiveness. Labor shortages have become a growing concern, straining medical staff who are already overburdened with high workloads. Additionally, managing logistics within a hospital—such as transporting medical supplies, delivering lab samples, and ensuring timely medication distribution—remains a complex and resource-intensive task. These inefficiencies contribute to delays in treatment, increased operational costs, and the risk of human errors, all of which can affect patient outcomes and hospital performance. Furthermore, maintaining strict hygiene standards, especially in the wake of global health crises, requires solutions that minimize human contact while ensuring reliable service delivery.

To address these challenges, AUTOMED, an autonomous hospital robot, offers a revolutionary approach to healthcare logistics and patient support. This advanced robotic system is designed to navigate busy hospital environments, optimize resource allocation, and automate routine tasks with precision. By leveraging cutting-edge technologies such as LiDAR-based navigation, ultrasonic sensors, and real-time mapping, AUTOMED significantly enhances efficiency while maintaining safety in dynamic hospital settings. Unlike traditional manual logistics systems, AUTOMED operates consistently without fatigue, reducing the chances of delays and errors in critical healthcare workflows.

Our vision for AUTOMED extends beyond simple automation; we envision a fully integrated, intelligent system that seamlessly collaborates with healthcare professionals. AUTOMED not only transports medical supplies while ensuring security but also interacts with hospital staff through mobile interfaces and real-time monitoring systems. By incorporating SLAM techniques, AUTOMED can dynamically adjust to changes in hospital conditions, efficiently navigating through evolving environments while avoiding obstacles and optimizing routes.

The benefits of AUTOMED are substantial. By taking over time-consuming and repetitive tasks, it frees up healthcare staff to focus on more critical patient care responsibilities. This not only improves efficiency but also enhances job satisfaction by reducing the burden of

non-medical tasks. Moreover, AUTOMED contributes to a safer hospital environment by minimizing unnecessary human contact, thereby lowering the risk of infection transmission. Its ability to operate 24/7 ensures that hospital logistics run smoothly without disruption, leading to improved patient outcomes and optimized resource utilization. In the long term, integrating AI-driven analytics and expanding its roles into patient monitoring and assistance can further revolutionize hospital operations, making healthcare more responsive, efficient, and scalable for future demands.

1.1 Project Objective

The project objective is to engineer AUTOMED, an autonomous medical delivery robot optimized for navigating and facilitating item delivery in a medical setting. This entails integrating SLAM, dynamic detection features, designing a flexible payload handling mechanism, and developing an intuitive mobile application interface for effortless control. Through real-world deployment and iterative refinement, the project seeks to validate AUTOMED's effectiveness in enhancing operational efficiency and productivity across diverse industrial contexts.

1.2 Project Scope

The AUTOMED project aims to develop a versatile, autonomous navigation robot tailored to enhance operational efficiency across various medical environments, including clinics, hospitals, and pharmacies. This initiative focuses on creating a robotic solution that minimizes human intervention, optimizes delivery processes, and adapts to diverse operational needs. By significantly reducing the need for human labor in repetitive tasks, AUTOMED allows staff in hospitals to focus on more complex duties, by handling resource transport works. Additionally, AUTOMED's adaptability enables it to navigate crowded human and obstacle populated areas like hospital hallways, patient rooms, and doctor offices, making it a valuable asset in the medical industry. By focusing on these core aspects, the AUTOMED project delivers a comprehensive robotic solution that enhances operational efficiency, reduces reliance on human labor, and meets the specific needs of different operational environments.

1.3 Project Overview

AUTOMED represents an endeavor aimed at improving navigation and item delivery tasks across healthcare environments. This innovative robotic system is meticulously designed to operate autonomously, with high efficiency and flexibility. AUTOMED's key features include its localization and mapping, advanced object detection capabilities, a versatile payload handling mechanism, and an intuitive mobile application interface for seamless control. AUTOMED can help improve operational standards, enhancing productivity, and streamlining processes in the healthcare sector.

Chapter 2

Literature Review

Autonomous navigation is a core component of RMDS, enabling robots to move efficiently and safely in dynamic hospital environments. The ability of robots to navigate without human intervention is important for tasks involving medicine delivery, patient monitoring, and logistics. Several studies have explored the use of SLAM techniques for autonomous navigation in indoor environments. SLAM is a widely used approach for allowing robots to map their surroundings while simultaneously localizing themselves within the map. Singh et al. (2022) proposed an adaptive SLAM approach using an ANN with an adaptive squashing function, which showed an improvement compared to traditional methods in complex environments. Their work demonstrated that the proposed method was capable of navigating social environments with dynamic obstacles while adhering to social norms [1]. Liu et al. (2024) delivered a comprehensive review of sensing technologies for indoor autonomous mobile robots, highlighting the importance of LiDAR, vision-based systems, and multi-sensor fusion systems in SLAM applications [2]. Ngoc et al. (2023) compared different SLAM methods, including GMapping, Cartographer SLAM, and SLAM Toolbox, within a ROS2 framework. Their study showed that SLAM Toolbox offered the highest accuracy x and reliability for autonomous navigation in dynamic environments [3].

Path planning is another crucial aspect of autonomous navigation. Alatise and Hancke (2020) reviewed various sensor fusion techniques and navigation algorithms for mobile robots, emphasizing on the importance of obstacle detection and avoidance in dynamic environments. They highlighted the use of LiDAR, ultrasonic sensors, and vision-based systems for real-time navigation and mapping [4]. A. Steenbeek (2020) introduced a deep learning-based approach for indoor localization and navigation using a monocular vision system. The CNN-based method used improved object detection and environmental awareness for mobile robots [5].

Sensor fusion is a core technology for enhancing the accuracy and reliability of autonomous navigation. By integrating data coming from multiple sensors, robots are able to better perceive their environment and make informed decisions. Alatise and Hancke (2020) explored the integration of multiple sensors, including LiDAR, ultrasonic sensors, and cameras, for localization and navigation. They highlight the importance of combining competitive, complementary,

and cooperative sensor fusion techniques which can enhance the robustness of robotic systems [4]. Liu et al. (2024) analyzed the advantages and limitations of single-sensor approaches and introduced multi-sensor fusion techniques, for application in indoor autonomous robots [2].

IoT has modernized healthcare by enabling smooth communication between devices and systems. Robotic systems based on the IoT can supply real-time data monitoring, remote control, and enhanced decision-making capabilities. S. Selvaraj et al. (2024) developed an IoT-embedded robotic system that was connecting to a centralized cloud-based controller. This allowed healthcare personnel to monitor patients and allow robot control remotely, improving the performance and safety of medicine delivery [6]. Liu et al. (2024) emphasized the role of IoT-enabled sensor networks in improving real-time data processing and decision-making for autonomous robots [2].

As robots increasingly interact with humans in social environments, it is of importance to ensure that their behavior is in line with social norms and expectations. Socially aware robotics focuses on creating robots that can traverse human-populated environments safely. Singh et al. (2022) handled the difficulties of SARN by adding social conventions and human motion prediction into their SLAM algorithm. Their approach ensured that the robot could navigate crowded environments without causing discomfort or disruption to humans [1]. Ngoc et al. (2023) explored the use of YOLO-based human detection models in SLAM frameworks, enhanced robots' ability to detect and respond to human presence in indoor environments [3].

Despite significant growth in RMDSs, several challenges like dynamic environments, where hospitals are highly chaotic, with constantly changing obstacles such as moving patients, staff, and equipment remain. Alatise and Hancke (2020) gave importance to the need for robust navigation algorithms that can adapt to these changes in real-time [4]. Another challenge is product cost and scalability, as high cost of advanced sensors and IoT infrastructure tend to be a barrier to the widespread adoption of robotic systems in healthcare. S. Selvaraj et al. (2024) believed that future research should give more importance on developing cost-effective solutions that can be easily applied to different healthcare settings [6]. Regulatory compliance is another critical challenge, as ensuring that robotic systems comply with healthcare regulations and safety standards is essential. Singh et al. (2022) gave importance for conducting rigorous testing and validation to ensure the safety and reliability of robotic systems in healthcare [1].

Chapter 3

Requirement Analysis

3.1 Feasibility Study

AUTOMED is an autonomous indoor delivery robot designed specifically for healthcare environments such as hospitals and clinics. It is used for transporting medical supplies, equipment, and documents efficiently within healthcare facilities. AUTOMED integrates key algorithms such as SLAM for real-time mapping, A* for efficient path planning, and sensor fusion via an EKF for precise navigation and obstacle avoidance. The project demonstrates the feasibility of using cost-effective yet high-performance components like the Raspberry Pi 4B, 2D LiDAR, IMU and relays to ensure reliable performance in dynamic, real-world hospital environments while maintaining affordability and security.

3.2 Software Requirement

Operating Environment:
AUTOMED is developed using the ROS2 framework on a Raspberry Pi 4B running Ubuntu OS, which provides a robust environment for handling complex robotic tasks. The core functionality of the robot, including sensor data processing, motor control, and integration with the mobile management system, is managed by Python-based ROS nodes. This setup allows for modular development, enabling real-time navigation, obstacle avoidance, and seamless communication within hospital networks.

Navigation and Control Software:
AUTOMED's control software utilizes ROS2 nodes written in Python for path planning while a combination of LiDAR, IMU and Ultrasonic sensors facilitate navigation. The A* algorithm is implemented for optimal path planning from the pharmacy or storage unit to patient rooms, doctors' offices, or designated delivery points. Obstacle detection and avoidance are managed in real-time by processing data from LiDAR and ultrasonic sensors, ensuring safe navigation through hospital corridors and around medical staff and equipment.

Sensor Integration:
AUTOMED uses sensory data from the LiDAR and IMU for mapping hospital environments. It also uses LiDAR, IMU, and Ultrasonic sensors to obtain environmental information for smooth navigation while avoiding obstacles. When manual control is implemented, the camera is used for receiving and transmitting visual data to authorized healthcare personnel for monitoring and control. Upon reaching its destination, AUTOMED uses a sliding mechanism implemented using a Rack and Pinion mechanism for precise delivery of medical supplies based on user authentication by RFID tag. The sensor fusion technique, an EKF is used to ensure seamless integration of multiple sensory inputs for efficient operation while minimizing processing loads on the Raspberry Pi unit.

Mobile Application:
The mobile application serves as the primary user interface for controlling AUTOMED within a healthcare facility. Developed using the Flutter framework, the app allows authorized medical staff to input delivery destinations, schedule automated routes, and monitor the robot's status in real-time. The connection between the robot and the app occur over a secure hospital WiFi network to ensure a reliable and secure connection. The application features an intuitive control interface for monitoring medicine deliveries, tracking item status, and receiving real-time notifications regarding obstacles, system status updates, and successful deliveries.

System Integration:
System integration for AUTOMED utilizes ROS2 middle-ware to facilitate seamless communication between navigation, control, and sensor modules. This approach involves establishing efficient communication protocols within the ROS framework to manage data exchange effectively. The integration ensures that various components, including LiDAR, IMU, cameras, and ultrasonic sensors, work in harmony, enabling real-time processing and decision-making. Additionally, the architecture of ROS2 supports scalability and flexibility, allowing for easy updates and enhancements in system functionality within healthcare infrastructures.

3.3 Hardware Requirement

Microcontroller:
AUTOMED utilizes the Raspberry Pi 4B as its central processing unit, leveraging its powerful processing capabilities and support for the ROS2 framework. Arduino Uno processing unit is also used in secondary functionalities including manual control, payload security and delivery. This choice ensures that Rasberry Pi 4B efficiently manages complex tasks, including sensor data processing, motor control,navigation algorithms, and communication with the hospital network.

Sensors:

For navigation and obstacle detection, AUTOMED incorporates a combination of a RPLidar A1M8, ultrasonic sensors, a USB camera module, and a BNO085 IMU. The LiDAR sensor enables precise mapping and real-time obstacle detection, while ultrasonic sensors provide additional distance measurements to avoid collisions with medical staff and hospital equipment. The camera modules assist with object recognition and enhanced environmental awareness. The IMU contributes to accurate orientation and motion tracking, helping the robot maintain stability and navigate hospital corridors more reliably. Additionally, an RFID reader is integrated to control access to the payload drawer, ensuring that only authorized personnel can open and close the storage compartment for secure medical supply handling.

Motors and Motor Control:
AUTOMED's mobility is powered by DC motors controlled through relays, providing precise speed and directional control essential for navigating hospital environments. Additionally, the robot is equipped with a motorized delivery tray mechanism, enabling it to place or retrieve medical supplies securely at designated locations, such as nurse stations, doctor's office or patient rooms. This integrated motor system ensures smooth, adaptable movement and reliable handling of sensitive medical equipment.

Power Supply:
AUTOMED is powered by a 12V rechargeable battery pack, offering portability and extended operational time. To maintain stable power across all components, including the LiDAR, Raspberry Pi, cameras, and sensors, buck converters regulate the voltage, ensuring reliable, consistent performance in real-time medical deliveries and operations.

Chassis and Structural Components:
AUTOMED's chassis is made from lightweight and strong aluminum, using a custom design created in Fusion 360. The structure gives solid support to all parts of the robot while keeping it easy to move and durable for daily use. The outer look of the robot was designed using Photoshop, giving it a clean and professional appearance suitable for healthcare settings. A lot of time and effort went into carefully placing and positioning each electronic component inside the robot to make sure everything fits well and works reliably. Important parts like sensors and processing units are firmly secured so they stay stable during movement and operation.

Payload Handling Mechanism:
AUTOMED is equipped with a motorized tray mechanism for secure transport of medical supplies, documents, and lightweight equipment. Using a Rack and Pinion mechanism controlled by geared motors, the delivery tray smoothly extends and retracts to facilitate safe handoff of medical items to nurses or designated collection points. To ensure security, the payload drawer employs RFID-based authorization, allowing only designated healthcare personnel to access its contents. Authorized users must scan their RFID tags to unlock and retrieve items, ensuring

controlled access and preventing unauthorized handling of critical medical supplies.

Communication Modules:
To facilitate interaction with hospital systems, AUTOMED utilizes WiFi connection enabling seamless connection with the mobile management system anywhere within the hospital environment. This module allows users to send commands, monitor real-time updates, and track the robot's delivery status remotely, ensuring efficient and responsive medical logistics.

3.4 Applications

Hospital and Clinical Environments :
AUTOMED enhances efficiency in hospitals and clinics by autonomously transporting medical supplies, documents, and equipment. It navigates through hallways, nurse stations, and patient rooms with precision, ensuring timely and secure deliveries. This capability reduces the workload on healthcare staff, allowing them to focus on patient care while improving logistics and response times in critical environments.

Pharmacy and Laboratory Assistance :
AUTOMED facilitates the automated delivery of medications and lab samples between hospital pharmacies, laboratories, and wards. This ensures fast and error-free transport of critical items, reducing human intervention and minimizing contamination risks.

Emergency Response Support :
In high-pressure medical scenarios where ED supports rapid transport of emergency medical kits, blood samples, and urgent prescriptions to critical areas, ensuring timely intervention when every second counts.

3.5 General Requirement

Autonomous Navigation:
The navigation system of the robot is built to allow fully autonomous movement from a user-set starting point to a chosen destination. For mapping and localization, the system uses SLAM with data from a LiDAR sensor and an IMU. These inputs are combined using an EKF , which helps the robot understand its exact position and orientation in real-time. For path planning, the robot uses the A* algorithm to find the best route to the destination. This combination of mapping, localization, and smart path planning ensures accurate and efficient movement within medical environments.

Obstacle Detection:
A critical requirement for AUTOMED is its ability to detect and avoid static as well as dynamic

obstacles autonomously. This involves using a combination of ultrasonic and LiDAR sensors to continuously scan the environment for potential hazards. The robot's control system processes this data in real-time to identify obstacles and make immediate decisions to slow down and avoid or stop if necessary. The obstacle detection and control system shows robust capabilities to handle both static and dynamic obstacles, ensuring safe operation in busy environments such as clinics and hospitals.

Payload Handling:
AUTOMED must be capable of securely transporting items from one location to another. This requires a payload tray equipped with a rack and pinion mechanism for smooth extension and retraction, enabling the robot to load and unload items with precision. The tray should be designed to hold a variety of items securely, preventing them from being tampered with during transport. This functionality is essential for applications in healthcare where precise delivery is critical.

User Interface and Interaction:
AUTOMED must provide an intuitive and user-friendly interface for controlling and monitoring its operations. This is typically achieved through a mobile application that connects to the robot via WiFi network. The app should allow users to input destinations, schedule tasks, and monitor the robot's status in real-time. It must include authentication features to secure access and control, ensuring that only authorized personnel can operate the robot. The application should also provide notifications and alerts regarding the robot's activities, such as arrival at a destination or encountering an obstruction, thereby enhancing user engagement and operational efficiency.

Chapter 4

Design

The design of AUTOMED integrates robust hardware components, sophisticated software systems, and efficient database architecture to create a reliable and versatile robotic solution for healthcare environments. The hardware design focuses on creating a durable and adaptable structural framework, ensuring stable and efficient movement, and facilitating secure payload handling. The software design encompasses advanced navigation and object detection capabilities, providing autonomous operation with minimal user intervention. Additionally, a user-friendly mobile application serves as the primary interface for operators, offering real-time monitoring and control. The database architecture supports efficient task management, secure user authentication, and a comprehensive feedback system to continuously improve performance. Together, these elements create a cohesive and effective robotic system capable of enhancing operational efficiency in healthcare environments like hospitals and clinics.

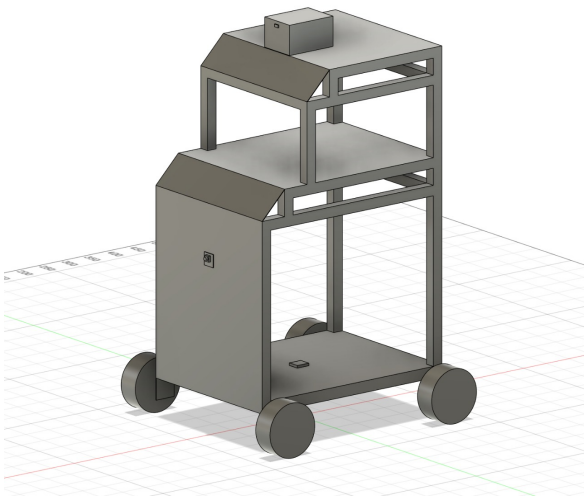

Figure 4.1: Robot Design View(1)

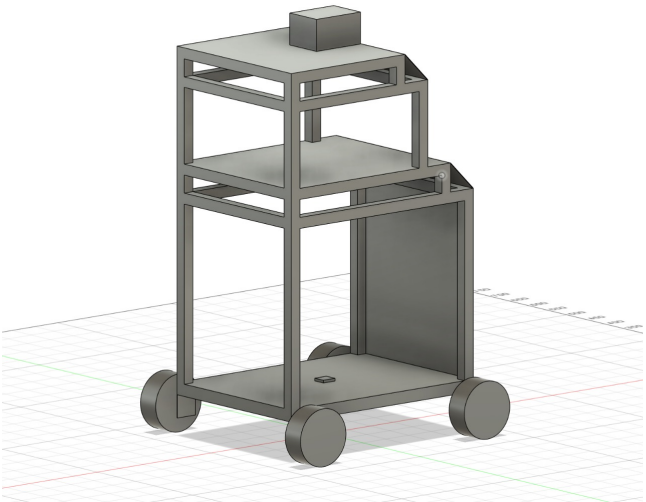

Figure 4.2: Robot Design View(2)

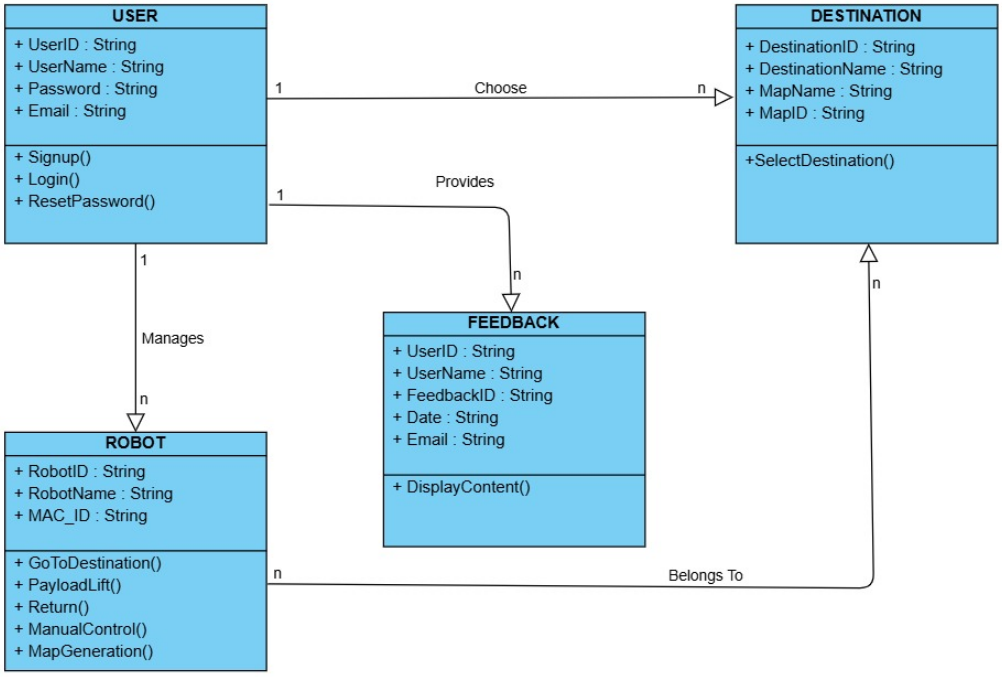

Figure 4.3: Class Diagram

4.1 Hardware Design

Processing Units:

The AUTOMED uses the following two processing units for handling all its robot operations:

- **Raspberry Pi 4B:** The primary processing unit used here is a Raspberry Pi 4B, which serves as the central controller for the robot, managing data processing, sensor integration, and decision-making. Running the ROS, the Raspberry Pi handles critical algorithms like SLAM for mapping and localization, EKF for sensor fusion and path planning to navigate the environment. It acts as the brain of the robot, coordinating between sensors, actuators, and communication modules, and executing commands based on the robot's perception of its surroundings.

- **Arduino Uno:** The Arduino serves as a secondary processing unit, receiving commands from the Raspberry Pi for motor control and to manage additional tasks such as payload loading and unloading, as well as RFID-based security identification and verification for secure payload delivery. Its integration helps offload specific functions from the Raspberry Pi, reducing computational strain and improving the overall efficiency and responsiveness of the robotic system.

Sensors:

The robot's sensor suite includes several components for environmental perception and internal feedback:

- **RPLIDAR A1M8:** Provides precise distance measurements and mapping data, helping the robot detect obstacles and build a 2D map of its surroundings.

- **BNO085 IMU:** Tracks the robot's orientation and movement, providing acceleration and rotational data to maintain stability and adjust its position while also providing an inbuilt sensor fusion capability.
- **USB Camera:** Captures visual data for object detection and recognition, enhancing the robot's ability to identify obstacles and interact with its environment.
- **Ultrasonic Sensors:** Offer close-range obstacle detection, acting as a secondary safeguard for collision avoidance.
- **RFID:** Enhances payload security by ensuring authorized access and tracking of transported items, preventing unauthorized handling or misplacement.

Actuators:
The actuators enable the robot to perform physical actions based on commands from the processing unit:

- **Robot Gear Motors:** Drive the wheels, enabling the robot to move forward, reverse, and turn as directed with higher torque than normal motors. The same gear motors also drive the sliding mechanism implemented using rack and pinion mechanism.

Power System:
The power system consists of a battery, voltage regulators, and relays, all working together to provide stable and controlled power to the robot. The battery supplies the required energy, while voltage regulators like buck converters ensure that each component receives the correct voltage, protecting sensitive electronics from power fluctuations. Relays are used to control the motors, allowing basic switching operations for movement. This setup ensures reliable performance and consistent operation of the robot during tasks.

Chassis:
AUTOMED features a durable aluminum chassis, providing a stable foundation and robust support for all components. The outer wall coverings are made of acrylic, ensuring a lightweight yet protective enclosure. Secure metal mounts securely hold electronic parts, ensuring precise positioning and ease of assembly. Sensors, the Raspberry Pi, and other hardware are securely affixed using screws and adhesive where needed, offering stability and resilience during navigation in hospital settings.

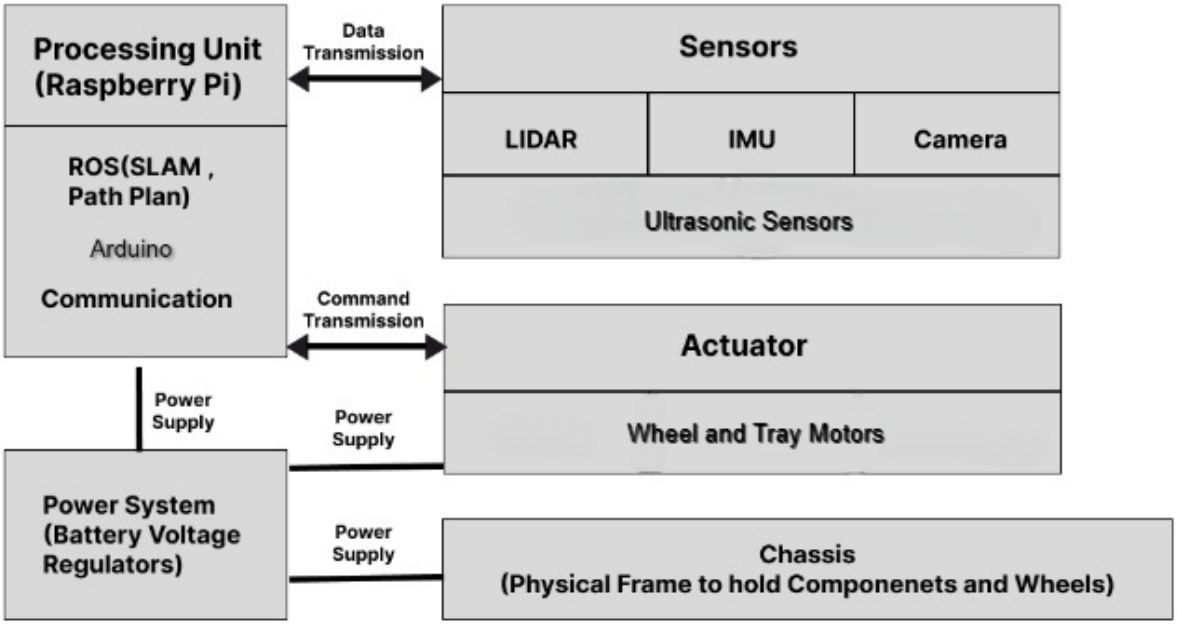

Figure 4.4: Hardware Design

4.2 Software Design

4.2.1 Mobile App Architecture

AUTOMED's mobile application, developed using Flutter, provides a user-friendly interface for medical staff to control and monitor the robot. In this mobile app architecture, each component serves a distinct role in managing user access, authentication, and navigation within the app. The components have been structured to create a seamless user experience, supporting new user registration, account login, password recovery, and user-friendly control. The mobile application provides the user with tools to control the robot and monitor its performance. It acts as an interface for enabling autonomous item deliveries as well as manual control over the robot with the help of visual data from the camera modules. Its simple and easy-to-use design gives a comfortable experience without any special knowledge requirements.

Profile and User Authentication pages:
The mobile application leverages the Firebase framework for robust user authentication and account management. Firebase supports seamless user registration, secure sign-in, password recovery, and real-time profile updates. The signup page enables new users to create an account, while the login page authenticates user credentials before granting access to the app. In case of forgotten credentials, the reset password page allows users to securely update their password after identity verification. Once registered, users can complete their personal information on the profile page, which includes details such as name, email, phone number, and address. The profile page also allows users to update their information at any time, ensuring

flexibility and accuracy in account management.

Home page:
The home page serves as the central hub of the mobile application and is the first screen users see upon logging in. It features a default map interface and a left sidebar listing the destinations that the robot can autonomously navigate to. Users can select from multiple map options and switch to a manual control page via a dedicated button. The home page also includes functionality for scanning and selecting a robot to connect with. Once a robot is selected, the page displays real-time notifications and status updates related to the robot's operations. As the control center for autonomous navigation, the home page allows users to input the desired destination, guiding the robot accordingly. Notifications are automatically generated in response to key events, such as the successful delivery of a payload, providing users with clear feedback throughout the process.

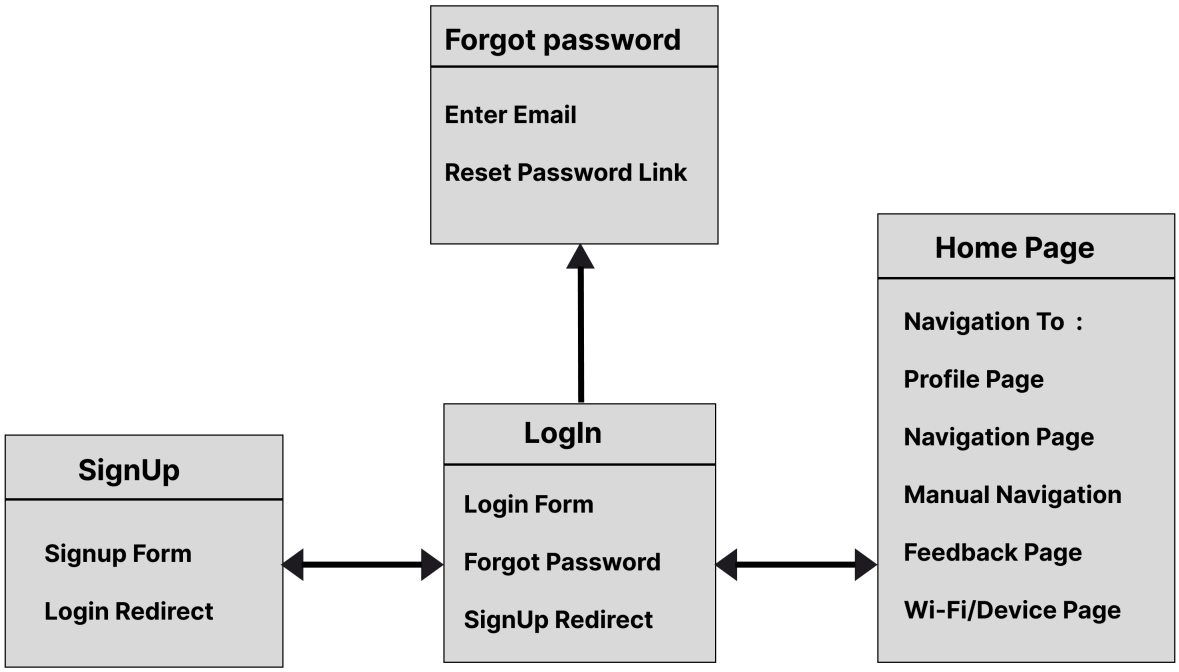

Figure 4.5: Mobile App Architecture

Manual navigation page:
The manual navigation page enables users to control the robot directly through intuitive on-screen buttons that send movement commands. It also features a live video feed from the robot's camera modules, providing real-time visual feedback during navigation. This visual data is particularly valuable for detecting and addressing issues that may occur during operation, such as identifying obstacles that hinder the robot's path, especially in situations where manual intervention is required.

Feedback page:
The feedback page is a feature which the users can use to inform the developers of any issues or difficulties they are facing. There is a facility for rating the problem to be submitted which allows efficient management of feedback from users. This page provides an opportunity to improve the mobile application as well as robot by reviewing the user feed backs and resolving them.

4.2.2 Robot Software Backend

In an indoor medical autonomous navigation robot, the core processing and decision-making happens through the programs loaded within the robot processing unit. This back end encompasses all the essential software components and algorithms that interpret sensor data, build environmental maps, plan routes, and ensure safe navigation. For an indoor robot equipped with a camera modules, IMU, RPLidar, and ultrasonic sensors, the back end brings together various technologies that work in tandem to provide real-time perception, mapping, and path planning.

Key back-end components include SLAM for mapping and localization, the A* algorithm for path finding, ROS as the framework coordinating data flow, and sensor fusion with EKF is used to merge data from multiple sensors. Together, these components allow the robot to understand its environment and make intelligent decisions about where and how to move, making the back end critical to achieving autonomy in complex indoor spaces. This combination of technologies ensures that the robot can navigate safely, avoid obstacles, and reach its goals without human intervention.

SLAM:
In an indoor navigation robot, SLAM is essential for creating and updating a map of the environment in real-time while determining the robot's position within it. The RPLidar provides accurate distance measurements to map walls, obstacles, and room layouts, while the IMU describes the appropriate orientation of the robot which contribute visual data for more detailed mapping and object recognition. Ultrasonic sensors add close-range data, helping the robot detect nearby obstacles that might be missed by other sensors for dynamic obstacle avoidance. Together, these inputs allow SLAM to generate a precise map, enabling the robot to navigate effectively in indoor spaces.

A* Algorithm:
The A* algorithm would guide the robot's navigation by finding the shortest, most efficient path to a destination within the mapped environment. By leveraging the SLAM-generated map, A* can plan a route that avoids obstacles detected by the RPLidar and Ultrasonic sen-

sors. This algorithm enables real-time adjustments, helping the robot to navigate around unexpected obstacles or adjust its path when certain areas become inaccessible. A* ensures that the robot follows an optimal path, balancing speed and safety in indoor navigation.

ROS:

ROS serves as the foundational framework that connects all the robot's components, allowing them to communicate and operate together seamlessly. ROS organizes data from RPLidar, and IMU into nodes, enabling easy management of SLAM, A*, and sensor fusion processes. It also provides tools for real-time visualization and debugging, making it easier to monitor the robot's perception and path-planning decisions. ROS's modular structure is ideal for managing the robot's complex software, ensuring reliable integration of each component and easy scalability.

Sensor Fusion:

Sensor fusion in AUTOMED combines data from the RPLidar, IMU, and ultrasonic sensors to support SLAM-based indoor mapping and reliable navigation. By integrating inputs from these sensors, the system compensates for individual sensor limitations—such as IMU drift or LiDAR occlusion—creating a more accurate and consistent environmental model. To further enhance navigation precision, an EKF is employed to correct slight deviations in the robot's path. The EKF uses a probabilistic approach, maintaining a set of weighted particles to estimate the robot's position and orientation in real time. As sensor data is continuously fused and processed through the EKF, the robot can adjust its trajectory dynamically, ensuring smooth and accurate movement even in cluttered or low-visibility indoor environments.

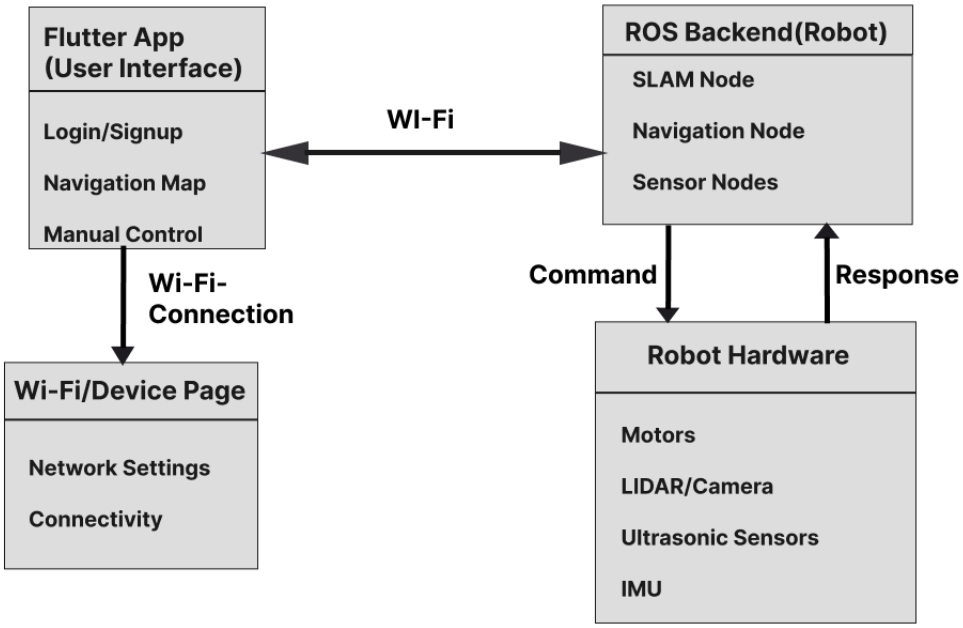

Figure 4.6: Mobile Application-ROS Integration Design

4.3 Project Architecture

The AUTOMED robot uses the sensory data for accurate and smooth motion of the robot. It uses the RPLidar and IMU to obtain environmental information which is processed using sensor fusion and fed to SLAM for mapping and localizing the robot within the environment. This is performed throughout the robot mapping phase until the entire area is covered. Once a map is generated, the sensors LiDAR, IMU and Ultrasonic sensors are used for navigating the mapped environment utilizing the A* path planning algorithm for efficient path traversal. When manual control is implemented visual data is transferred to the mobile app for the user control. The sensor usage which is performed selectively as described allows efficient use of energy and processing power of the Raspberry Pi.

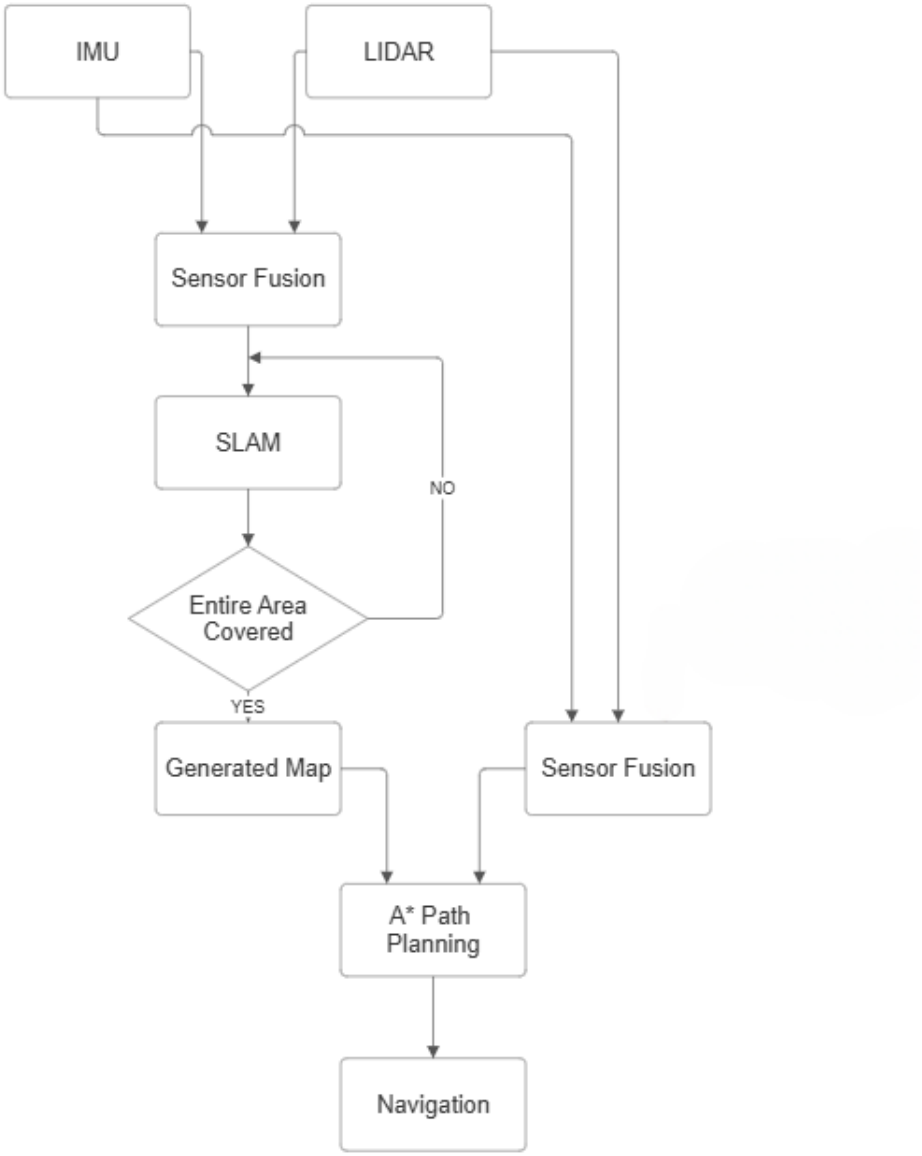

Figure 4.7: Mapping and Navigation Work Flow

4.3.1 Database Structure

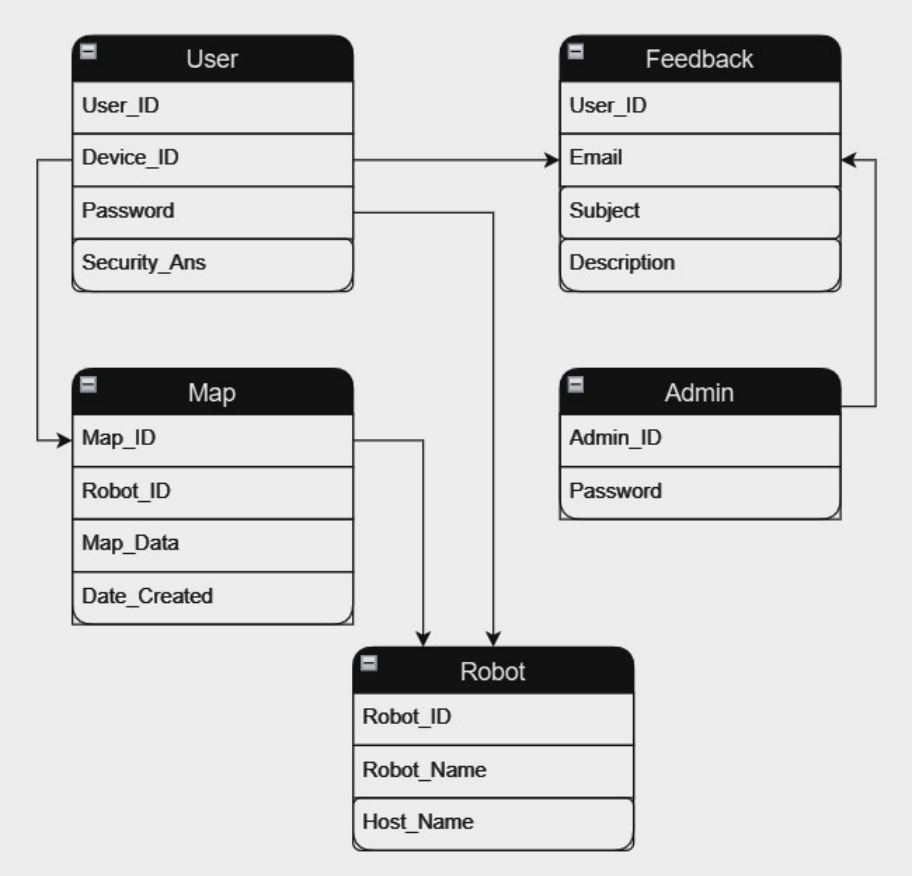

Figure 4.8: Mobile Application Database Schema

User Authentication:
AUTOMED's mobile application includes a user authentication system using firebase to ensure secure access. User credentials are securely stored, and encryption is applied during transmission to prevent unauthorized access. This authentication system ensures that only verified users can control and interact with AUTOMED, adding a layer of security to its operations and safeguarding the robot from unauthorized use.

User Feedback System:
The AUTOMED mobile application incorporates a feedback feature, allowing users to share their experiences, suggest improvements, and report issues encountered during operation. This feedback is stored in a structured format within the database, enabling developers to review and analyze user insights. By continuously gathering and acting on this feedback, AUTOMED's functionality and user satisfaction can be enhanced over time, supporting ongoing improvements to its performance and usability.

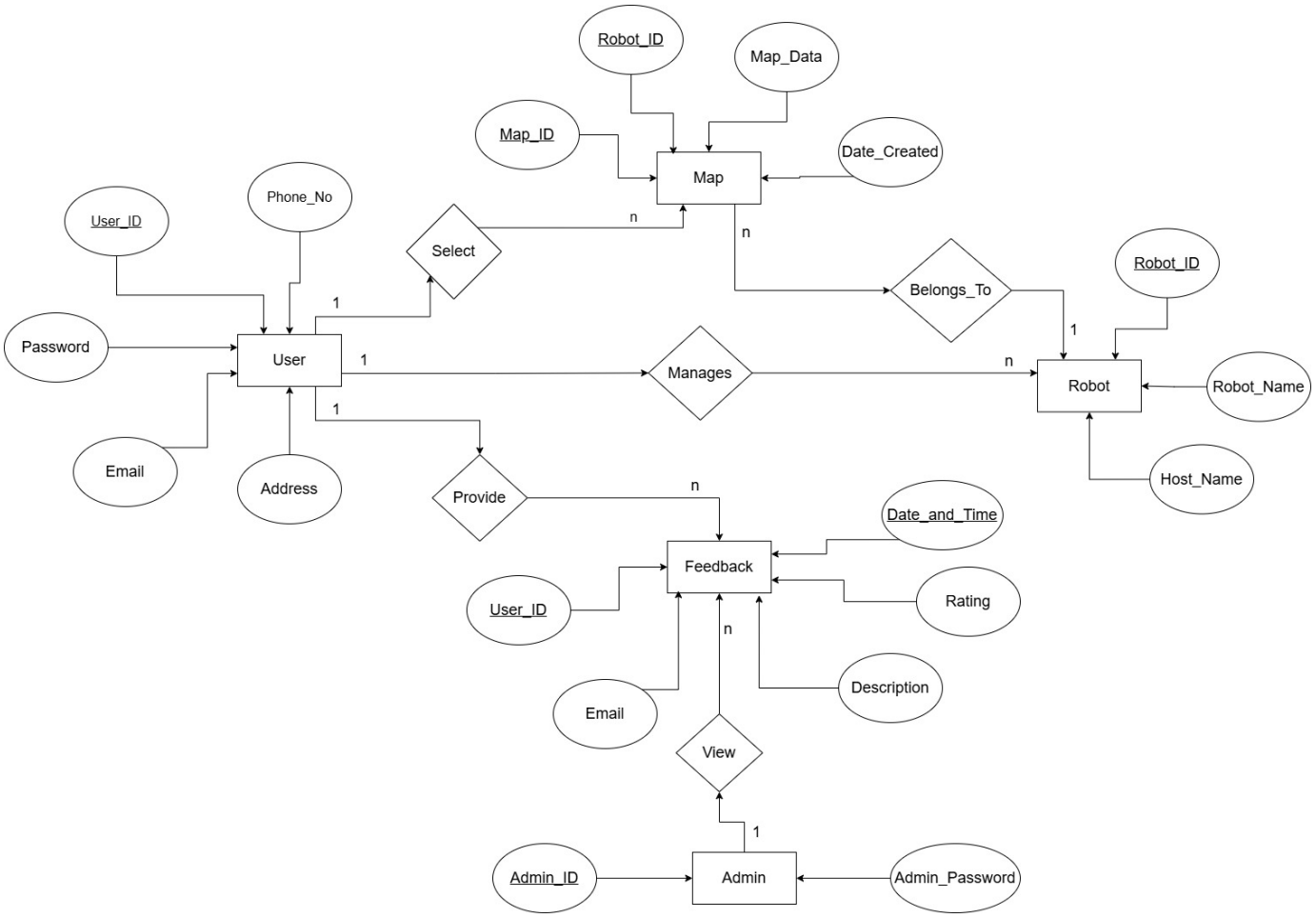

Figure 4.9: ER Diagram

Chapter 5

Methodology

The design and implementation of AUTOMED, a versatile autonomous navigation and delivery robot, progresses through carefully structured phases. Core technologies such as SLAM, A*, and a Flutter-based mobile application were integrated to enable real-time navigation, object detection, path planning, and seamless user interaction. Each phase of development contributed to achieving a robust, autonomous system capable of operating in dynamic indoor environments.

Figure 5.1: Front View of Robot

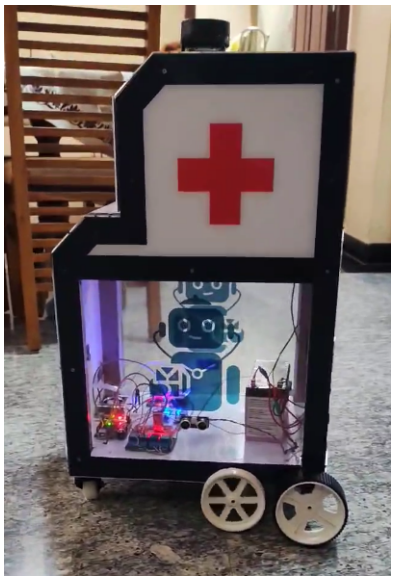

Figure 5.2: Side View of Robot

5.1 Components Selection

The project began with defining the functional requirements of AUTOMED for its intended environments, which included clinics, healthcare facilities, and hospitals. Key hardware and software components were identified based on these requirements:

- **Hardware:** Components such as Raspberry Pi, LiDAR, camera modules, ultrasonic sensors, and IMU were chosen for their reliability in real-time perception and navigation.

- **Software:** ROS2 acts as the real-time middle-ware that coordinates communication, data exchange, and control among the sensors, processing algorithms, and actuators in an autonomous robot. SLAM, EKF and A* were designated as the primary algorithms for object detection, mapping, sensor fusion and path planning, respectively. The Flutter framework was chosen for developing a mobile app to facilitate user interaction with AUTOMED.

5.2 Environment Mapping and Localization using SLAM

SLAM was implemented to allow AUTOMED to build and update a map of its surroundings while determining its own position. SLAM was crucial for AUTOMED's autonomy in dynamically changing environments.

- **Implementation and Testing:** Data from LiDAR and IMU were fed into the SLAM algorithm, generating a real-time map of the environment. The robot's location was updated continuously to maintain precise navigation. SLAM's performance was rigorously tested using ROS tools like RViz and Gazebo, simulating complex layouts with frequent updates. By optimizing the robot's performance using sensor fusion and filtering techniques, the SLAM system achieved higher accuracy in localization and mapping, even in cluttered and crowded spaces.
- **Integration with ROS2:** SLAM enabled AUTOMED to generate accurate, high-resolution maps, supporting safe and effective navigation in real-time. The continuous updates to the map ensured that even dynamic obstacles were accounted for, improving AUTOMED's adaptability in complex settings.

5.3 Path planning and Navigation using A* Algorithm

The A* algorithm was employed as the primary path-finding solution, leveraging the SLAM-generated map for efficient route planning. A* ensured that AUTOMED could navigate to a specified destination while avoiding obstacles detected by SLAM.

- **Implementation and Testing:** The A* algorithm was employed as the primary path planning solution, using the SLAM-generated map for efficient, global route planning. A* enabled AUTOMED to navigate to specified destinations by calculating optimal paths across the known environment, avoiding obstacles identified by SLAM.
- **Integration with ROS2:** Through iterative testing, the A* algorithm was integrated into AUTOMED's navigation system, with parameters fine-tuned to balance computation time and path efficiency. This optimization ensured that A* could effectively use the SLAM-generated map for global route planning.

5.4 Development of the User Interface with Flutter

The Flutter framework is used to develop a mobile application that serves as the primary user interface for controlling AUTOMED and monitoring its activities. The application is designed to provide an intuitive and interactive user experience.

- **Application Features:** The key features provided include user authentication, autonomous navigation, manual navigation, and real-time robot tracking. Users can connect to their selected robot via WiFi and view details such as battery status upon connection. For navigation, users can either input destinations for autonomous navigation or manually control AUTOMED using visual data from the camera modules. Notifications are integrated to alert users of obstacles, successful deliveries, and system status updates during operation. The home page containing the map and destinations as well as the manual navigation page used to control the robot manually can be seen in Figure 5.3 and Figure 5.4 respectively.
- **Integration and Testing:** The application is connected to the robot via a WiFi network, with real-time data exchange facilitated through ROS2. The mobile application depends on the communication with the ROS2 for location and visual data. Additionally, a feedback system is integrated to allow users to report issues, contributing to continuous improvement.

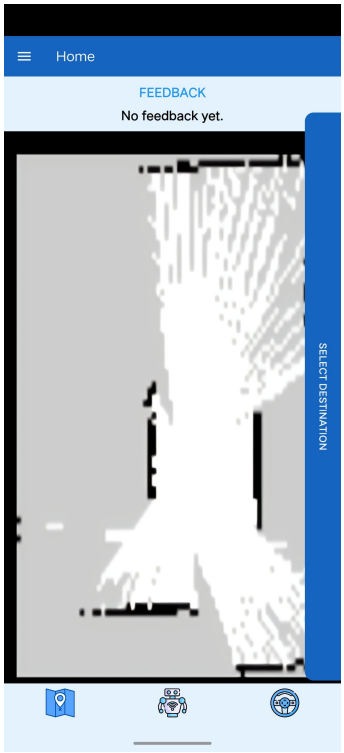

Figure 5.3: Home Page

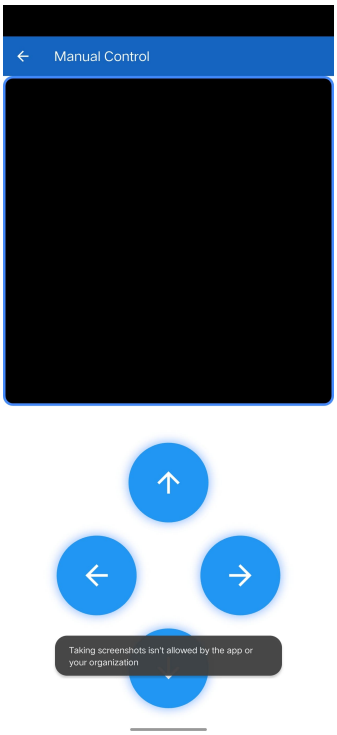

Figure 5.4: Manual Nav Page

5.5 Full System Integration

The design and implementation of AUTOMED, a versatile autonomous navigation and delivery robot, were solely focused on the healthcare sector, aiming to ensure efficient, secure, and reliable medical supply transportation within hospitals and clinical facilities. The development followed a structured approach, progressing through defined phases involving the integration of core robotics technologies such as SLAM, the A* algorithm for optimal path planning, and a Flutter-based mobile application for intuitive human-robot interaction.

The project began with establishing functional requirements tailored to medical environments such as reliable delivery in cluttered corridors, sterile area compliance, and authorized access control. These needs informed the selection of key hardware components including the Raspberry Pi 4B, RPLidar, ultrasonic sensors, IMU, camera modules, wheel encoders, and RFID modules. Central to the robot's architecture is the Raspberry Pi, which runs ROS2, serving as the real-time middle-ware that manages inter-process communication, sensor data exchange, control coordination, and integration with high-level planning and perception modules.

AUTOMED's autonomous navigation is grounded in SLAM, allowing the robot to construct and localize itself within 2D occupancy grid maps of hospital corridors and rooms. Sensor data from the RPLidar and IMU are fused using the robot_localization package, enhancing pose estimation accuracy and stability across varying indoor conditions. To improve localization further and handle minor path deviations caused by drift or uneven terrain, an EKF was applied. EKF utilizes a probabilistic approach to refine the robot's real-time state estimates through weighted particle distributions, allowing for precise corrections during navigation.

Figure 5.5: Map Scan

The A* path planning algorithm was used to compute optimal routes from the robot's current location to user-defined destinations on SLAM-generated maps. Integrated within the move_base ROS2 framework, A* ensures computational efficiency and adaptability, enabling AUTOMED to avoid obstacles such as hospital beds, carts, and personnel. For dynamic obstacle avoidance, real-time sensor data from the RPLidar and perimeter-mounted ultrasonic sensors are processed within costmap_2d, and local planning is managed using the DWA planner to calculate collision-free velocity commands.

Security and payload management are handled by an Arduino microcontroller, which interfaces with an RFID reader and a rack-and-pinion drawer mechanism. The RFID system ensures that only authorized personnel can initiate or receive deliveries by scanning identification tags at both pickup and drop-off points. Upon successful authentication, the Arduino activates the payload drawer system to securely load or unload medical supplies. The Arduino communicates with the Raspberry Pi via serial communication, ensuring synchronized task execution.

Control and interaction with AUTOMED are provided via a Flutter-based mobile application, which connects to the robot using a Wi-Fi WebSocket API. During initial setup, the application connects over Bluetooth, transitioning to Wi-Fi for operational use. The app displays a static map generated through SLAM, allowing users to select destination points for autonomous navigation. These goals are transmitted to the ROS2 stack via ROSBridge, enabling real-time command execution. The app supports manual control for scenarios requiring human intervention—such as navigating tight spaces or repositioning the robot—by offering directional input controls and a live camera feed.

The mobile application is secured through Firebase Authentication, which manages user accounts, login credentials, and session validation. Firebase is also used to deliver real-time notifications such as successful deliveries, obstacle detections, and system errors, keeping users informed of the robot's operational status. Furthermore, the app features a feedback module, allowing healthcare staff to report issues, request features, or suggest improvements directly to developers. Feedback is logged in Firebase Firestore, providing the development team with valuable insights for iterative updates.

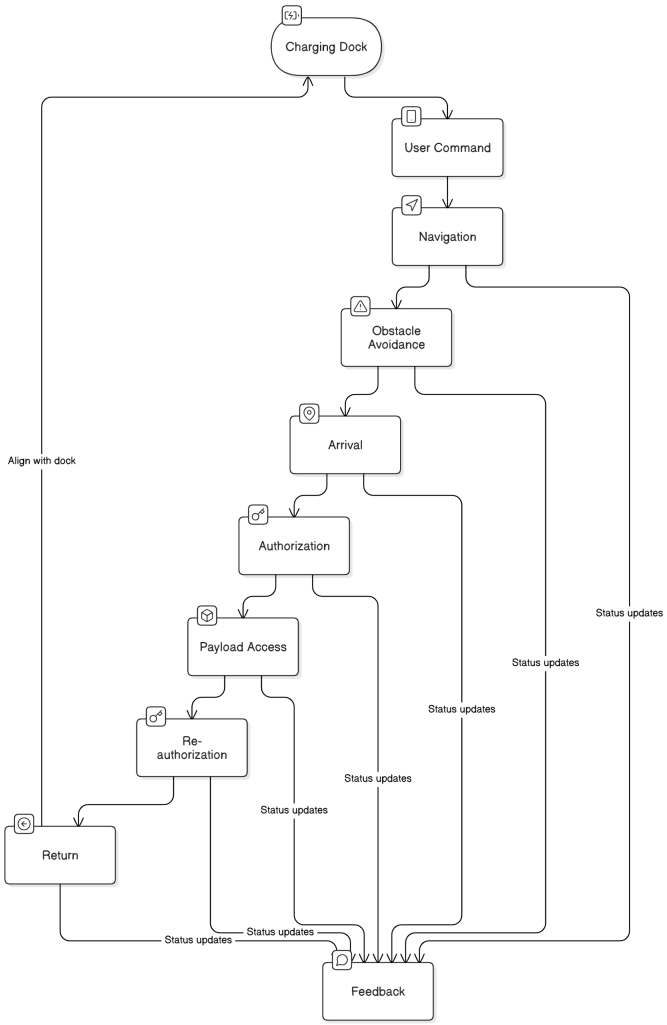

Figure 5.6: Robot Operation Flow

Final system integration combined all software and hardware components into a fully functional solution, tested in real-world hospital simulations using ROS tools like RViz and Gazebo. These tests validated the effectiveness of SLAM-based mapping, A*-driven path planning, sensor fusion with EPF, and secure payload handling. The application layer enabled seamless control and monitoring, giving healthcare staff autonomy and flexibility in using the system.

Chapter 6

Conclusion and Future Scope

AUTOMED represents a significant advancement in the integration of AMRs within the healthcare sector. Purpose-built to address the complex and dynamic challenges of hospital environments, AUTOMED seamlessly combines advanced object detection, obstacle avoidance, and precise navigation technologies. Leveraging robust solutions such as SLAM for real-time mapping and localization, alongside the A* pathfinding algorithm, the system navigates hospital corridors, patient rooms, and sterile areas with accuracy and without interrupting critical medical workflows.

A key strength of AUTOMED lies in its intuitive Flutter-based mobile application, which offers secure and streamlined interaction for healthcare professionals. Features such as user authentication, profile management, robot selection, manual control, and real-time notifications make the app an essential interface for managing robotic operations. Additionally, the integrated payload handling system, supported by a reliable lifting mechanism, enables the autonomous transport of medical supplies and equipment, further enhancing operational efficiency within clinical settings.

In the broader context of healthcare robotics, AUTOMED aligns with the emerging role of AMRs in transforming hospital logistics, improving patient care, and optimizing clinical workflows. With advancements in artificial intelligence—including deep learning models like YOLO and U-Net for object detection and semantic segmentation—AUTOMED is well-equipped to recognize patients, medical tools, and critical areas, while adhering to stringent healthcare protocols. By addressing key challenges such as workforce limitations, logistical inefficiencies, and the need for contactless service, AUTOMED showcases the potential of intelligent robotic systems in shaping the future of healthcare delivery.

Future Scope:
Although AUTOMED is fully functional and has successfully demonstrated its effectiveness in real-world hospital environments, its potential for future development remains considerable. Integration with HIS and EHR would enable real-time synchronization of delivery schedules,

patient requirements, and inventory management. This would allow AUTOMED to operate with greater contextual awareness, adapting dynamically to changing hospital demands and enhancing interdepartmental coordination.

Looking ahead, the incorporation of advanced AI capabilities such as voice and facial recognition for secure, hands-free operation, and predictive route optimization based on real-time hospital activity could significantly elevate AUTOMED's autonomy and efficiency. The deployment of collaborative multi-robot systems could also enable coordinated tasks across multiple units, improving scalability in larger healthcare facilities. With potential applications in geriatric care, emergency logistics, and telemedicine support, AUTOMED is poised to evolve into a critical component of future intelligent healthcare ecosystems—addressing accessibility challenges, supporting overburdened medical staff, and enhancing patient-centered care.

Chapter 7

Appendix

Robot Backend(ROS)

LIDAR Odometry Node:

```
class LidarOdometryNode : public rclcpp::Node
{
  public:
    LidarOdometryNode() : Node("lidar_odometry_node")
    {
      // Initialize parameters
      parameter_initilization();

      // Get parameters
      this->get_parameter("max_correspondence_distance", max_correspondence_distance);
      this->get_parameter("transformation_epsilon", transformation_epsilon);
      this->get_parameter("maximum_iterations", maximum_iterations);
      this->get_parameter("scan_topic_name", scan_topic_name);
      this->get_parameter("odom_topic_name", odom_topic_name);
      this->declare_parameter("publish_tf", false);
      publish_tf_ = this->get_parameter("publish_tf").as_bool();

      // Setup lidar odometry
      lidar_odometry_ptr = std::make_shared<LidarOdometry>(max_correspondence_distance, transformation_epsilon, maximum_iterations);

      // Create publisher and subscriber
      odom_publisher = this->create_publisher<nav_msgs::msg::Odometry>(odom_topic_name, 100);
      scan_subscriber = this->create_subscription<sensor_msgs::msg::LaserScan>(
        scan_topic_name, 1000, std::bind(&LidarOdometryNode::scan_callback, this, std::placeholders::_1)
      );

      // Initialize TF broadcaster
      tf_broadcaster_ = std::make_shared<tf2_ros::TransformBroadcaster>(this);
    }

  private:
    rclcpp::Publisher<nav_msgs::msg::Odometry>::SharedPtr odom_publisher;
    rclcpp::Subscription<sensor_msgs::msg::LaserScan>::SharedPtr scan_subscriber;
    std::shared_ptr<LidarOdometry> lidar_odometry_ptr;
    std::shared_ptr<tf2_ros::TransformBroadcaster> tf_broadcaster_;
    bool publish_tf_ = false;

    // Initialize parameters
    void parameter_initilization() {
        this->declare_parameter<double>("max_correspondence_distance", 1.0);
        this->declare_parameter<double>("transformation_epsilon", 0.005);
        this->declare_parameter<double>("maximum_iterations", 30);
        this->declare_parameter<std::string>("scan_topic_name", "scan");
        this->declare_parameter<std::string>("odom_topic_name", "scan_odom");
    }
```

Figure 7.1: LIDAR Odometry Node(a)

```
    // Callback for scan data
    void scan_callback(const sensor_msgs::msg::LaserScan::SharedPtr scan_msg) {
        auto point_cloud_msg = laser2cloudmsg(scan_msg);
        auto pcl_point_cloud = cloudmsg2cloud(point_cloud_msg);

        auto scan_data = std::make_shared<ScanData>();
        scan_data->timestamp = scan_msg->header.stamp.sec + scan_msg->header.stamp.nanosec / 1e9;
        scan_data->point_cloud = pcl_point_cloud;

        lidar_odometry_ptr->process_scan_data(scan_data);
        publish_odometry();
    }

    // Publish odometry data
    void publish_odometry() {
        auto state = lidar_odometry_ptr->get_state();
        nav_msgs::msg::Odometry odom_msg;

        odom_msg.header.frame_id = "odom";
        odom_msg.child_frame_id = "base_link";
        odom_msg.header.stamp = this->get_clock()->now();

        odom_msg.pose.pose = Eigen::toMsg(state->pose);
        odom_msg.twist.twist = Eigen::toMsg(state->velocity);

        odom_publisher->publish(odom_msg);

        // Publish TF if needed
        if (publish_tf_ && tf_broadcaster_) {
           geometry_msgs::msg::TransformStamped odom_tf;
           odom_tf.header.stamp = this->get_clock()->now();
           odom_tf.header.frame_id = "odom";
           odom_tf.child_frame_id = "base_link";
           odom_tf.transform.translation.x = state->pose.translation().x();
           odom_tf.transform.translation.y = state->pose.translation().y();
           Eigen::Quaterniond quat(state->pose.rotation());
           odom_tf.transform.rotation = tf2::toMsg(quat);

           tf_broadcaster_->sendTransform(odom_tf);
        }
    }
};

int main(int argc, char ** argv)
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<LidarOdometryNode>());
  rclcpp::shutdown();
  return 0;
}
```

Figure 7.2: LIDAR Odometry Node(b)

IMU Node:

```
class IMUNode(Node):
    def __init__(self):
        super().__init__("bno085_imu_node")
        i2c = busio.I2C(board.SCL, board.SDA)
        self.imu = BNO08X_I2C(i2c)
        self.imu.enable_feature(BNO_REPORT_ROTATION_VECTOR)
        self.imu_pub = self.create_publisher(Imu, "imu/data", 10)
        self.timer = self.create_timer(0.1, self.publish_imu_data)

    def publish_imu_data(self):
        try:
            quat = self.imu.quaternion  # Get quaternion data
            if quat is None:
                self.get_logger().warn("Quaternion data not available yet.")
                return
            msg = Imu()
            msg.header.stamp = self.get_clock().now().to_msg()
            msg.header.frame_id = "imu_link"

            # Set IMU orientation (quaternion data)
            msg.orientation.w = quat[0]
            msg.orientation.x = quat[1]
            msg.orientation.y = quat[2]
            msg.orientation.z = quat[3]

            # Set covariance values
            msg.orientation_covariance = [
                0.0025, 0.0, 0.0,
                0.0, 0.0025, 0.0,
                0.0, 0.0, 0.0025
            ]
            msg.angular_velocity_covariance = [0.02] * 9
            msg.linear_acceleration_covariance = [0.04] * 9

            # Set zero values for angular_velocity and linear_acceleration (if not provided)
            msg.angular_velocity.x = msg.angular_velocity.y = msg.angular_velocity.z = 0.0
            msg.linear_acceleration.x = msg.linear_acceleration.y = msg.linear_acceleration.z = 0.0

            # Publish IMU data
            self.imu_pub.publish(msg)
            self.get_logger().info(f"Published IMU: {quat}")

        except Exception as e:
            self.get_logger().error(f"IMU Read Error: {e}")

def main(args=None):
    rclpy.init(args=args)
    node = IMUNode()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()
```

Figure 7.3: IMU node

Extended Kalman Filter:

```
ekf_filter_node:
  ros__parameters:
    frequency: 15.0
    two_d_mode: true

    # Frames
    map_frame: map
    odom_frame: odom
    base_link_frame: base_link
    world_frame: odom

    # IMU Configuration

    #imu0: /imu/data
    #imu0_config: [false, false, false,  # x, y, z (Position)
    #              false, false, true,  # roll, pitch, yaw (Orientation)
    #              false, false, false,    # Vx, Vy, Vz (Velocity)
    #              false, false, false,   # Roll, Pitch, Yaw Velocity
    #              false, false, false]  # Acceleration

    #imu0_differential: false
    #imu0_remove_gravitational_acceleration: true
    #imu0_queue_size: 10
    #imu0_frame: imu_link
    #imu0_relative: true

    # LiDAR Odometry Configuration
    odom0: /scan_odom
    odom0_config: [true, true, false,   # x, y, z (Position)
                   false, false, true,  # roll, pitch, yaw (Orientation)
                   false, false, false,   # Vx, Vy, Vz (Velocity)
                   false, false, false, # Roll, Pitch, Yaw Velocity
                    false, false, false] # Acceleration

    odom0_differential: false
    odom0_queue_size: 10
    odom0_frame: odom
    odom0_relative: false

    print_diagnostics: true
    publish_tf: true
    publish_acceleration: false
    sensor_timeout: 0.1
```

Figure 7.4: Extended Kalman Filter

Mapping Parameters:

```
slam_toolbox:
  ros__parameters:

    solver_plugin: solver_plugins::CeresSolver
    ceres_linear_solver: SPARSE_NORMAL_CHOLESKY
    ceres_preconditioner: SCHUR_JACOBI
    ceres_trust_strategy: LEVENBERG_MARQUARDT
    ceres_dogleg_type: TRADITIONAL_DOGLEG
    ceres_loss_function: None

    odom_frame: odom
    map_frame: map
    base_frame: base_link
    scan_topic: /scan
    use_map_saver: true
    mode: mapping  # options: mapping, localization

    debug_logging: false
    throttle_scans: 1
    transform_publish_period: 0.02  # If 0, never publishes odometry
    map_update_interval: 5.0
    resolution: 0.05
    min_laser_range: 0.2  # for rastering images
    max_laser_range: 6.0  # for rastering images
    minimum_time_interval: 0.5
    transform_timeout: 1.0
    tf_buffer_duration: 30.0
    stack_size_to_use: 40000000  # Larger stack size for serialization of large maps
    enable_interactive_mode: true
```

Figure 7.5: Mapping Parameters(a)

```
    use_scan_matching: true
    use_scan_barycenter: true
    minimum_travel_distance: 0.5
    minimum_travel_heading: 0.5
    scan_buffer_size: 10
    scan_buffer_maximum_scan_distance: 6.0
    link_match_minimum_response_fine: 0.1
    link_scan_maximum_distance: 1.5
    loop_search_maximum_distance: 3.0
    do_loop_closing: true
    loop_match_minimum_chain_size: 10
    loop_match_maximum_variance_coarse: 3.0
    loop_match_minimum_response_coarse: 0.35
    loop_match_minimum_response_fine: 0.45
    use_tf_scan_transforms: true
    scan_queue_size: 1000

    correlation_search_space_dimension: 0.5
    correlation_search_space_resolution: 0.01
    correlation_search_space_smear_deviation: 0.1

    loop_search_space_dimension: 8.0
    loop_search_space_resolution: 0.05
    loop_search_space_smear_deviation: 0.03

    distance_variance_penalty: 0.5
    angle_variance_penalty: 1.0

    fine_search_angle_offset: 0.00349
    coarse_search_angle_offset: 0.349
    coarse_angle_resolution: 0.0349
    minimum_angle_penalty: 0.9
    minimum_distance_penalty: 0.5
    use_response_expansion: true
```

Figure 7.6: Mapping Parameters(b)

Mobile Application(Flutter Code)

Home Page:

```
import '../screens/login.dart';
import '../screens/profile_page.dart';
import '../screens/feedback.dart';
import '../screens/control_screen.dart';
import '../screens/wifi_setup_screen.dart';

final FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  void enableBackgroundMode() async {
    final androidConfig = FlutterBackgroundAndroidConfig(
      notificationTitle: "Robot Feedback",
      notificationText: "Listening for robot feedback...",
      notificationImportance: AndroidNotificationImportance.normal,
      notificationIcon: AndroidResource(name: 'ic_launcher', defType: 'mipmap'),
    );
    final success = await FlutterBackground.initialize(androidConfig: androidConfig);
    if (success) {
      await FlutterBackground.enableBackgroundExecution();
    }
  }

  late WebSocketChannel _channel;
  String latestStatusMessage = 'No feedback yet.';

  @override
  void initState() {
    super.initState();
    enableBackgroundMode();

    _channel = WebSocketChannel.connect(
      Uri.parse('ws://192.168.1.56:8080'),
    );

    _channel.stream.listen((message) {
      try {
        final decoded = jsonDecode(message.toString());
        if (decoded is Map && decoded.containsKey('command')) return;
      } catch (_) {}

      setState(() {
        latestStatusMessage = message.toString();
      });
```

Figure 7.7: Home Page(a)

```
      showFeedbackNotification(message.toString());
    });
  }

  @override
  void dispose() {
    _channel.sink.close();
    super.dispose();
  }

  void showFeedbackNotification(String message) async {
    const androidDetails = AndroidNotificationDetails(
      'feedback_channel_id',
      'Robot Feedback Notifications',
      channelDescription: 'Notifies when feedback is received from robot',
      importance: Importance.max,
      priority: Priority.high,
    );

    const notificationDetails = NotificationDetails(android: androidDetails);

    await flutterLocalNotificationsPlugin.show(
      0,
      'Robot Feedback',
      message,
      notificationDetails,
    );
  }

  void _onBottomNavTap(int index) {
    if (index == 2) {
      _channel.sink.add(jsonEncode({'command': 'manual'}));
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => const ControlScreen(
            mjpegUrl: 'http://192.168.1.56:5000/video_feed',
            websocketUrl: 'ws://192.168.1.56:8080',
          ),
        ),
      );
    } else if (index == 1) {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => const WifiSetupScreen()),
      );
    } else {
      setState(() {
        _selectedIndex = index;
      });
    }
  }
```

Figure 7.8: Home Page(b)

```
  Future<void> _logout() async {
    await FirebaseAuth.instance.signOut();
    if (!mounted) return;
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (context) => const LoginPage()),
    );
  }

  @override
  Widget build(BuildContext context) {
    final user = FirebaseAuth.instance.currentUser;

    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.blue[800],
        title: const Text('Home', style: TextStyle(color: Colors.white)),
      ),
      body: Column(
        children: [
          Text(
            latestStatusMessage,
            textAlign: TextAlign.center,
            style: const TextStyle(color: Colors.black, fontSize: 20, fontWeight: FontWeight.bold),
          ),
        ],
      ),
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _selectedIndex,
        onTap: _onBottomNavTap,
        items: const [
          BottomNavigationBarItem(
            icon: Icon(Icons.map),
            label: '',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.wifi),
            label: '',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.control_point),
            label: '',
          ),
        ],
      ),
    );
  }
}
```

Figure 7.9: Home Page 3

Manual Navigation Page:

```
class ControlScreen extends StatefulWidget {
  final String mjpegUrl;
  final String websocketUrl;

  const ControlScreen({super.key, required this.mjpegUrl, required this.websocketUrl});

  @override
  _ControlScreenState createState() => _ControlScreenState();
}

class _ControlScreenState extends State<ControlScreen> {
  late WebViewController _webViewController;
  String? _pressedButton;
  Timer? _commandTimer;
  bool _webViewError = false;
  bool _isWebSocketConnected = false;
  late WebSocketChannel _webSocketChannel;

  @override
  void initState() {
    super.initState();
    // Restart camera stream
    http.get(Uri.parse("http://192.168.255.41:5000/start")).then((response) {
      debugPrint("Camera start response: ${response.statusCode}");
    }).catchError((error) {
      debugPrint("Error starting camera: $error");
    });
    _initializeWebView();
    _connectWebSocket();
  }

  /// Establish WebSocket Connection
  void _connectWebSocket() {
    try {
      _webSocketChannel = WebSocketChannel.connect(Uri.parse(widget.websocketUrl));
      setState(() => _isWebSocketConnected = true);
    } catch (e) {
      debugPrint("WebSocket Connection Failed: $e");
      setState(() => _isWebSocketConnected = false);
    }
  }
```

Figure 7.10: Manual Navigation Page(a)

```
/// Initialize WebView for Camera Stream
void _initializeWebView() {
  _webViewController = WebViewController()
    ..setJavaScriptMode(JavaScriptMode.unrestricted)
    ..setBackgroundColor(Colors.black)
    ..setNavigationDelegate(NavigationDelegate(
      onProgress: (int progress) => debugPrint("WebView loading: $progress%"),
      onPageStarted: (String url) => setState(() => _webViewError = false),
      onPageFinished: (String url) => debugPrint("WebView finished: $url"),
      onWebResourceError: (WebResourceError error) {
        debugPrint("WebView Error: ${error.description}");
        setState(() => _webViewError = true);
      },
    ))
    ..loadRequest(Uri.parse(widget.mjpegUrl));
}

/// Send WebSocket Command
void _sendWebSocketCommand(String command) {
  if (_isWebSocketConnected) {
    try {
      _webSocketChannel.sink.add(jsonEncode({"command": command}));
      debugPrint("Sent WebSocket command: $command");
    } catch (e) {
      debugPrint("WebSocket Error: ${e.toString()}");
      setState(() => _isWebSocketConnected = false);
    }
  } else {
    debugPrint("WebSocket Not Connected");
  }
}

/// Start Sending WebSocket Command Repeatedly
void _startSendingCommand(String command) {
  if (!_isWebSocketConnected) return;

  setState(() => _pressedButton = command);
  _sendWebSocketCommand(command);
  _commandTimer = Timer.periodic(const Duration(milliseconds: 200), (timer) {
    _sendWebSocketCommand(command);
  });
}

/// Stop Sending Command (Stop Movement)
void _stopSendingCommand() {
  _commandTimer?.cancel();
  if (_isWebSocketConnected) {
    _sendWebSocketCommand('K'); // Stop Command
  }
  setState(() => _pressedButton = null);
}
```

Figure 7.11: Manual Navigation Page(b)

```
@override
Widget build(BuildContext context) {
  return ScreenUtilInit(
    designSize: const Size(393, 852),
    builder: (context, child) {
      return Scaffold(
        appBar: AppBar(
          backgroundColor: Colors.blue[800],
          title: const Text('Manual Control', style: TextStyle(color: Colors.white)),),
        backgroundColor: Colors.white,
        body: Column(
          children: [
            AspectRatio(
              aspectRatio: 1,
              child: Container(
                decoration: BoxDecoration(
                  color: Colors.black,
                  border: Border.all(color: Colors.blueAccent, width: 3.w),
                  borderRadius: BorderRadius.circular(12.r),),
                child: _webViewError
                    ? Center(
                        child: ElevatedButton(
                          onPressed: () {
                            setState(() => _webViewError = false);
                            _initializeWebView();
                          },
                          style: ElevatedButton.styleFrom(
                            backgroundColor: Colors.blue[800],
                            padding: EdgeInsets.symmetric(horizontal: 20.w, vertical: 10.h),
                          ),
                          child: Text("Retry", style: TextStyle(fontSize: 16.sp, color: Colors.white)),
                        ),)
                    : ClipRRect(
                        borderRadius: BorderRadius.circular(10.r),
                        child: WebViewWidget(controller: _webViewController),
                      ),),),
            const SizedBox(height: 10),
            Expanded(
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  _buildControlButton(command: 'P', icon: Icons.arrow_upward),
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      _buildControlButton(command: 'R', icon: Icons.arrow_back),
                      SizedBox(width: 60.w),
                      _buildControlButton(command: 'S', icon: Icons.arrow_forward),
                    ],),
                  _buildControlButton(command: 'Q', icon: Icons.arrow_downward),
                ],),),],),);},);}
```

Figure 7.12: Manual Navigation Page(c)

```
  /// Build Control Buttons
  Widget _buildControlButton({required String command, required IconData icon}) {
    bool isPressed = _pressedButton == command;
    return GestureDetector(
      onTapDown: (_) => _startSendingCommand(command),
      onTapUp: (_) => _stopSendingCommand(),
      onTapCancel: _stopSendingCommand,
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 200),
        margin: EdgeInsets.all(8.r),
        decoration: BoxDecoration(
          color: _isWebSocketConnected
              ? (isPressed ? Colors.green : Colors.blue)
              : Colors.grey,
          shape: BoxShape.circle,
          boxShadow: [
            BoxShadow(
              color: isPressed && _isWebSocketConnected
                  ? Colors.greenAccent.withOpacity(0.5)
                  : Colors.blueAccent.withOpacity(0.5),
              blurRadius: 10,
              spreadRadius: 2,
            ),],),
        padding: EdgeInsets.all(28.r),
        child: Icon(icon, size: 40.sp, color: Colors.white),
      ),);}


  @override
  void dispose() {
    _commandTimer?.cancel();
    _webSocketChannel.sink.close();
    // Shutdown camera stream
    http.get(Uri.parse("http://192.168.255.41:5000/shutdown")).then((response) {
      debugPrint("Camera shutdown response: ${response.statusCode}");
    }).catchError((error) {
      debugPrint("Error shutting down camera: $error");
    });
    super.dispose();
  }
}
```

Figure 7.13: Manual Navigation Page(d)

References

[1] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39830–39846, 2020.

[2] A. Steenbeek, "Cnn based dense monocular visual slam for indoor mapping and autonomous exploration," Master's thesis, University of Twente, 2020.

[3] K. J. Singh, D. S. Kapoor, K. Thakur, A. Sharma, A. Nayyar, S. Mahajan, and M. A. Shah, "Map making in social indoor environment through robot navigation using active slam," *IEEE Access*, vol. 10, pp. 134455–134465, 2022.

[4] S. Selvaraj, "An iot intelligent approach for safety and efficiency of robotic medicine delivery in hospitals," *Journal of Electrical Systems*, vol. 20, pp. 820–827, 04 2024.

[5] H. T. Ngoc, N. N. Vinh, N. T. Nguyen, and L.-D. Quach, "Efficient evaluation of slam methods and integration of human detection with yolo based on multiple optimization in ros2.," *International Journal of Advanced Computer Science & Applications*, vol. 14, no. 11, 2023.

[6] Y. Liu, S. Wang, Y. Xie, T. Xiong, and M. Wu, "A review of sensing technologies for indoor autonomous mobile robots," *Sensors*, vol. 24, no. 4, p. 1222, 2024.

[7] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li, "A comparative analysis of lidar slam-based indoor navigation for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6907–6921, 2021.

[8] D. Meiller, *Modern App Development with Dart and Flutter 2: A Comprehensive Introduction to Flutter*. Walter de Gruyter GmbH & Co KG, 2021.

[9] M. Kusuma, C. Machbub, *et al.*, "Humanoid robot path planning and rerouting using a-star search algorithm," in *2019 IEEE international conference on signals and systems (ICSigSys)*, pp. 110–115, IEEE, 2019.

[10] F. Foroughi, Z. Chen, and J. Wang, "A cnn-based system for mobile robot navigation in indoor environments via visual localization with a small dataset," *World Electric Vehicle Journal*, vol. 12, no. 3, p. 134, 2021.

automed

ORIGINALITY REPORT

11%	8%	6%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to APJ Abdul Kalam Technological University, Thiruvananthapuram Student Paper	2%
2	Mammen Jacob Ambrayil, Prejith Prasanna Kumar, Sandhra Merin Sabu, Thomas P. George, G. Gokulnath. "BLOCK-FQ: A blockchain based food quality assurance", AIP Publishing, 2022 Publication	1%
3	mdu.diva-portal.org Internet Source	1%
4	Ammu Kuriakose, Dinnu Sebastian, Esther Mahima Mathew, Hannu Mathew, G Er.Gokulnath. "ALIKAH- A Clickbait and Fake News Detection System using Natural Language Processing", 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019 Publication	<1%
5	journals.orclever.com Internet Source	<1%
6	thesai.org Internet Source	<1%
7	catalog.lib.kyushu-u.ac.jp Internet Source	<1%
8	www.hindawi.com Internet Source	<1%
	Submitted to Ohio University	