

**VISDOM: SMART GUIDE ROBOT
FOR VISUALLY IMPAIRED PEOPLE**

By
LEE ZHEN TING

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

FEBRUARY 2025

COPYRIGHT STATEMENT

© 2025 Lee Zhen Ting. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to extend my sincere gratitude to my supervisor, Mr. Teoh Shen Khang, and my moderator, Mr. Wong Chee Siang, for providing me with the valuable opportunity to be involved in the field study of the Internet of Things. Their continuous support and insightful guidance have been instrumental in helping me complete this project. During challenging moments, their advice and encouragement consistently helped me find effective solutions. I am truly thankful for their dedication and assistance throughout this journey.

ABSTRACT

This report details the development of an autonomous guide robot system for visually impaired individuals using the ROS 2 framework and the Wheeltec R550 Robot platform. Key features include the SMAC Hybrid A* algorithm for global path planning, the Regulated Pure Pursuit Controller for local trajectory execution and dynamic obstacle avoidance, and gmapping for SLAM functionality. The system architecture integrates ROS 2 on a Raspberry Pi, with TCP/IP connectivity enabling remote operation. An Android mobile application, developed using Java and the java.net.Socket library, provides an intuitive and accessible user interface for seamless interaction with the robot. The app incorporates **TalkBack accessibility support**, enabling visually impaired users to navigate the interface using screen reader feedback. Once a destination is selected, the robot guides the user using a combination of **voice prompts** (e.g., "Heading to kitchen", "Be careful to your left") and physical feedback through a mounted guiding stick. The project has demonstrated successful results in mapping and navigation within static indoor environments. Core functionalities such as path planning, autonomous movement, voice feedback, and app-to-robot communication have been thoroughly tested and optimized. The integration of the mobile interface and real-time voice prompting significantly enhances user accessibility and confidence. This system represents a significant advancement in assistive technology, offering improved mobility and independence for visually impaired individuals. The project showcases the practical application of autonomous navigation technologies in real-world scenarios, highlighting the potential of robotics and AI in improving quality of life for people with visual impairments.

Area of Study : Robotic, Internet of Things, Mobile App Development

Keywords: ROS 2, Autonomous Navigation, Assistive Robotics, Obstacle Avoidance, Path Planning, SLAM, Internet of Things(IOT)

TABLE OF CONTENTS

TITLE PAGE	I
COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	3
1.3 Project Scope and Direction	4
1.4 Contributions	4
1.5 Report Organization	5
CHAPTER 2 LITERATURE REVIEW	7
2.1 Existing Robot Framework in Robot Development	7
2.1.1 Robot Operating System (ROS)	7
2.1.2 Robot Operating System 2 (ROS 2)	8
2.1.3 Real-Time Control Framework (RTCF)	9
2.2 Existing Techniques in Autonomous Mobile Robot Development	11
2.3 Existing ROS2 App Projects	13
2.3.1 ROS2 Mobile Android	13
2.3.2 ROS2 App For Controlling A Robot via Bluetooth	14
2.3.3 Simplest Android App To Communicate With A Ros 2 Node	16
2.4 Previous Study On Indoor Guiding Robot Design	19
2.4.1 Lysa Robot Guide Dog	19

2.4.2 Design of a Portable Indoor Guide Robot for Blind People	20
2.4.3 The Construction of Occupancy Grid Map with Semantic Information for the Indoor Blind Guiding Robot	22
2.4.4 Autonomous Indoor Navigation for Visually Impaired: A ROS-Based Intelligent Guide Robot with Deep Learning and SLAM Integration	23
2.4.5 Exploring Levels of Control for a Navigation Assistant for Blind Travelers	25
2.5 Summary	27
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH	29
3.1 System Design Diagram	29
3.1.1 System Architecture Design	29
3.2 Use Case Diagram	31
3.3 Activity Diagram	33
CHAPTER 4 SYSTEM DESIGN	35
4.1 System Block Diagram	35
4.2 System Components Specifications	39
4.3 System Components Interaction Operations	41
CHAPTER 5 SYSTEM IMPLEMENTATION	42
5.1 Hardware Setup	42
5.2 Software Setup	44
5.3 Setting and Configuration	47
5.4 System Operation	51
5.5 Implementation Issues and Challenges	58
5.6 Concluding Remark	59
CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION	60

6.1 System Testing and Performance Metrics	60
6.2 Testing Setup and Result	62
6.3 Project Challenges	64
6.4 Objectives Evaluation	66
6.5 Concluding Remark	68
CHAPTER 7 CONCLUSION AND RECOMMENDATION	69
7.1 Conclusion	69
7.2 Recommendation	70
REFERENCES	71
POSTER	74

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.3.1	Establish Ssh Connection (Left)	13
Figure 2.3.2	Initializing Joystick And Grid map Publisher And Subscriber (Right)	13.
Figure 2.3.3	Robot Control Mockup	14
Figure 2.3.4	Block Diagram Of The Android App	16
Figure 2.3.5	Showcase On How The Communication Works	17
Figure 2.4.1	Lysa Robot Guide Dog	19
Figure 2.4.2	The Design Of The Robot	20
Figure 2.4.3	Autonomous Navigation Framework In ROS	21
Figure 2.4.4	System Hardware	22
Figure 2.4.5	Technical Framework of the system	23
Figure 2.4.6	Components breakdown of Glide	25
Figure 3.1.1	System Architecture Diagram	29
Figure 3.2	Mobile App Use Case Diagram	31
Figure 3.3	Mobile App Activity Diagram	33
Figure 4.1.1	System Block Diagram	35
Figure 4.1.2	System Flow Diagram	36
Figure 5.1.1	Power On the Robot	42
Figure 5.1.2	Modified with Metal Bracket	43
Figure 5.1.3	Finalized Robot Guiding Setup	43
Figure 5.2.1	Burning ROS2 Ubuntu image into Raspberry pi	44
Figure 5.2.2	Pc Software Update	45
Figure 5.2.3	Download Android Studio Linux	46
Figure 5.3.1	New Andoird Project Creation	47
Figure 5.3.2	Setting Project Configuration	48
Figure 5.3.3	Socket Library Import	49
Figure 5.3.4	Start_server Function	49
Figure 5.3.5	Setting up Robot Ip Address	50

Figure 5.3.5	Open a new TCP socket	50
Figure 5.4.1	Connect To The Robot Wifi	51
Figure 5.4.2	Connection Between PC And Robot	51
Figure 5.4.3	Launch The Slam_Gmapping Node	52
Figure 5.4.4	Rviz Mapping Process	52
Figure 5.4.5	Launch Navigation Stack On Robot	53
Figure 5.4.6	Rviz Map After Navigation Stack Is Turn On From Robot	53
Figure 5.4.7	Location Coordinates	54
Figure 5.4.8	blind_navigation_server.py	54
Figure 5.4.9	Execute The Python Script File	54
Figure 5.4.10	BlindNavigationApp Main Menu	55
Figure 5.4.11	Destination Menu	56
Figure 5.4.12	Guiding Operation	57

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Summary Of Existing Robot Framework in Robot Development	10
Table 2.2	Summary Of Existing Techniques in Autonomous Mobile Robot Development	12
Table 2.3	Summary Of Existing ROS2 App Design	18
Table 2.5	Summary of previous study on Indoor Guiding Robot Design.	27
Table 3.2.1	Choose Destination Use Case Description	32
Table 3.2.2	Return to Previous Page Use Case Description	32
Table 3.2.3	Choose Destination Use Case Description	32
Table 4.2.1	Specifications of laptop	38
Table 4.2.2	Specification of Mobile Phone	38
Table 4.2.3	Specifications of microcontroller	38
Table 4.2.4	Specifications of Wheeltec R550 ROS2 Robot	39
Table 6.1.1	Testing Metrics	59
Table 6.1.2	Performance Metrics	60
Table 6.2.1	Test case 1	61
Table 6.2.2	Test case 2	61
Table 6.2.3	Test case 3	62
Table 6.2.4	Test case 4	62

LIST OF ABBREVIATIONS

<i>ROS</i>	Robot Operating System
<i>ROS 2</i>	Robot Operating System 2
<i>RTCF</i>	Real-Time Control Framework
<i>AMR</i>	Autonomous Mobile Robots
<i>LiDAR</i>	Light Detection and Ranging
<i>RPLiDAR</i>	RP Light Detection and Ranging
<i>SDK</i>	Software Development Kit
<i>DDS</i>	Data Distribution Service
<i>TCP</i>	Transmission Control Protocol
<i>IP</i>	Internet Protocol
<i>UDP</i>	User Datagram Protocol
<i>OROCOS</i>	Open Robot Control Software
<i>IMU</i>	Inertial measurement unit
<i>DWA</i>	Dynamic Window Approach
<i>AI</i>	Artificial Intelligence
<i>ODM</i>	Object Detection Module
<i>PPS</i>	Path Planning System
<i>ADR</i>	Autonomous Delivery Robot
<i>SSD</i>	Single Shot Detector
<i>CNN</i>	Convolutional Neural Networks
<i>SLAM</i>	Simultaneous Localization and Mapping
<i>MPU</i>	Microprocessor Unit
<i>AMCL</i>	Adaptive Monte Carlo Localization
<i>SOGM</i>	Semantic Occupancy Grid Map
<i>PID</i>	Proportional-Integral-Derivative
<i>SMAC</i>	Search Motion Planning and Control

CHAPTER 1 INTRODUCTION

1.1 Problem Statement and Motivation

This paper focuses on addressing significant challenges related to providing effective and interactive guidance for visually impaired individuals. Designing and implementing indoor guiding robots for visually impaired individuals presents several significant challenges that must be overcome to ensure effective navigation, seamless user interaction, and practical usability.

Navigation Challenges: Visually impaired individuals frequently encounter difficulties navigating complex indoor environments like airports, shopping malls, and office buildings. These spaces can be confusing due to their large size and intricate layouts, which traditional mobility aids, such as white canes or guide dogs, may struggle to handle effectively. The challenge is to develop a robot capable of guiding users safely and efficiently through these complicated environments.

User Interaction: For a guiding robot to be effective, clear and intuitive communication with the user is essential. The robot should be able to initiate and manage interactions, offering guidance and concluding engagements in a way that is both intuitive and socially acceptable for visually impaired individuals. This demands the integration of technologies like speech recognition and audio feedback to facilitate smooth user-robot interactions. Overcoming this challenge is vital for the robot's overall success.

Design and Usability: Involving visually impaired users in the design process is crucial to ensure the robot's features align with their actual needs and preferences. A participatory design approach allows for creating a robot that is not only technically sound but also user-friendly and effective in real-world scenarios. The challenge here is balancing technical functionality with user-centered design principles.

Technical Development: The robot needs to be robust and adaptable, with the ability to navigate dynamic indoor environments and handle changing conditions. This includes effectively detecting and avoiding obstacles while responding to environmental changes. Ensuring that the robot can adapt to different indoor settings presents a significant technical challenge, requiring advanced sensor integration and sophisticated algorithms. By tackling these challenges, the goal is to create an indoor guiding robot that delivers reliable, intuitive, and user-focused assistance for visually impaired individuals.

The motivation for this project stems from the desire to create a robust, useful, and impactful robot that significantly improves the quality of life for the target audience. The robot is intended to assist in critical tasks such as path planning, obstacle avoidance, and environmental acknowledgment, thereby enhancing the independence and safety of visually impaired individuals.

In this rapidly evolving technological landscape, numerous advancements are reshaping our human experience. Among these technologies, robotics stands out as a transformative force comparable to Edison's development of the lightbulb. Robotics is a multidisciplinary field that intersects various engineering domains and information technology. The contributions of robots extend across diverse sectors, ranging from e-commerce and service industries to healthcare, aerospace, education, and the military.

These robots possess advanced assisting functions, utilizing sophisticated path planning technology, advanced computer vision systems, and effective human-robot interaction to aid in safe and efficient navigation.

Drawing inspiration from the evolution of technology, our goal is to make this advanced robotic solution accessible and beneficial to a wider population. Just as computers were once a luxury due to high prices and technical complexities, autonomous mobile robots currently face similar barriers. By focusing on a robust system design that ensures minimal bugs and smooth operation, I aim to overcome these hurdles. Technological innovation combined with user-centric control will play a crucial role in making these robots practical and reliable for everyday use.

The comprehensive scope of this project integrates cutting-edge technologies with a deep understanding of user needs, aiming to provide substantial benefits in practical scenarios. As the business landscape becomes more competitive, I anticipate a push to popularize such robots among the public. This project aims to follow that trajectory, making sophisticated indoor guidance robots an essential and accessible tool for visually impaired individuals.

1.2 Objectives

Main Objective: Develop an autonomous mobile robot system with mobile app that are capable of assisting visually impaired individuals by providing indoor walking guidance. The robot will guide users along the safest path to their destination and interact with the robot.

Sub-objectives:

1. Implement the robot's ROS 2 system to utilize the core features.
2. Integrate all functions and features with the installed sensors on the robot.
3. Implement core algorithms such as path planning, object avoidance, and SLAM.
4. Perform modifications on the robot to suit its purpose of leading individuals.
5. Develop a fully functional, complete system for the guidance robot.
6. Develop a visually impaired ROS2 mobile app to ease the interaction and function control of the robot.

The primary goal of this project is to develop an autonomous mobile robot capable of assisting visually impaired individuals by providing reliable indoor walking guidance. To achieve this, the project will focus on several key sub-objectives. Firstly, the robot's core features will be **implemented using the Robot Operating System 2(ROS2)**, which serves as the foundation for the system's functionality. This will include setting up the basic architecture and ensuring all essential components are operational. Next, **all functions and features will be integrated with the sensors installed on the robot.** This integration is critical for enabling the robot to perceive its environment and interact effectively with users, ensuring smooth and reliable operation. To support autonomous navigation, essential algorithms such as **path planning, object avoidance, and simultaneous localization and mapping (SLAM) will be integrated.** These algorithms will allow the robot to navigate safely through complex indoor environments, avoiding obstacles and finding the safest routes to the destination. Additionally, **modifications will be made to the robot** to ensure it is optimized for its guiding purpose. These adjustments will tailor the robot's physical and software configurations to meet the specific needs of visually impaired users. The final outcome will be a fully functional guidance robot system with a **ROS2 based app** that rely on socket-based communication protocol using the TCP/IP for robot interaction designed specifically for visually impaired users.

1.3 Project Scope and Direction

This project focuses on developing and delivering a fully operational autonomous mobile robot designed to serve as an indoor guide for visually impaired individuals. The core emphasis is on creating a robust, impactful solution that significantly enhances the quality of life for its users. Key components of the project include advanced path planning, obstacle avoidance, and a robust communication system design that ensures reliability, with minimal bugs and smooth operation. The study will primarily focus on the technical and practical aspects of integrating these features into a cohesive system.

However, the project **will not cover topics** related to **Circuit board Design, Robot design, outdoor navigation, advanced AI learning for unstructured environments, human robot interaction, machine learning object detection**. These areas are outside the project's current scope and will be considered limitations.

The discussion will include theories related to robotics, sensor integration, and path planning algorithms. Metadata will be analyzed to evaluate system performance, user interaction, and feedback. The proposed methodology is designed to be applicable in real-life settings, ensuring that the robot can operate effectively in indoor environments and provide substantial benefits to visually impaired users.

1.4 Contributions

The proposed project is an innovative autonomous mobile robot designed to serve as an indoor guide for visually impaired individuals, significantly enhancing their quality of life. The contributions of this project are outlined as follows:

Firstly, the robot will be complemented by a visually impaired-friendly mobile app, which simplifies access to the robot when users approach it. This app will ensure that the interaction with the robot is intuitive and convenient, making the overall experience seamless and enjoyable for visually impaired users. This focus on accessibility and ease of use is crucial in fostering a positive user experience, which is vital for the robot's acceptance and integration into daily life.

Secondly, the robot will feature advanced path planning algorithms that account for the presence of a human companion. When navigating and taking turns, the robot will consider the human's position and distance, ensuring that sharp turns are avoided to prevent the user from hitting obstacles. This careful consideration of the human companion during navigation highlights the robot's capability to provide safe and effective guidance.

Thirdly, the robot will be equipped with features such as guiding a user from point A to point B and allowing them to choose their destinations. This functionality is central to the robot's purpose, offering precise and reliable guidance to visually impaired individuals as they navigate indoor environments. By enabling users to select their destinations easily and guiding them along the safest routes, the robot ensures both independence and safety.

By integrating these key features, the project aims to provide substantial benefits in practical scenarios for visually impaired individuals. The robot's ability to guide users along the safest path to their destination, represents a significant advancement in assistive technology. These contributions collectively aim to improve the independence and mobility of visually impaired individuals, fostering public acceptance and widespread adoption of this technology. Ultimately, the project seeks to positively influence public perception and encourage the integration of advanced robotic solutions into daily life, potentially transforming industry marketing strategies and driving innovation in assistive technology.

1.5 Report Organization

This report is organized into seven chapters, each focusing on a key aspect of the project, from its conceptual foundation to the final evaluation and future recommendations. Chapter 1 introduces the project by outlining the problem statement, background and motivation, objectives, scope, and contributions. It also includes this section on how the report is structured.

Chapter 2 presents the literature review, which explores existing robotic frameworks such as ROS and ROS 2, real-time control frameworks, and technologies used in autonomous mobile robot development. It also reviews several ROS 2 mobile app projects and guiding robot studies, offering insights into their strengths, limitations, and applicability to this project.

Chapter 3 explains the system methodology and design approach, including visual tools like the system architecture diagram, use case diagram, and activity diagram to illustrate how the system works from both technical and user perspectives.

Chapter 4 discusses the preliminary work, covering the system block diagram, hardware specifications, and how each component interacts. It establishes the groundwork for full implementation.

Chapter 5 focuses on the system implementation, detailing the process of setting up the hardware and software components, configuring the environment, and explaining how the robot operates in real conditions. This chapter also highlights the challenges encountered and how they were addressed.

Chapter 6 evaluates the system through testing, analyzes performance metrics, and discusses the challenges and the extent to which the original objectives were met.

Finally, Chapter 7 concludes the report by summarizing the outcomes and proposing recommendations for future improvements and research directions.

CHAPTER 2 LITERATURE REVIEW

2.1 Existing Robot Framework in Robot Development

2.1.1 Robot Operating System (ROS)

The Robot Operating System (ROS) has revolutionized robotics research and development by facilitating code reuse and providing a distributed framework [1]. It offers packages for controlling various robotic systems and seamlessly integrates with embedded systems and physical components [2]. ROS organizes processes (nodes) into stacks and packages, enabling flexible execution at runtime and fostering easy sharing and distribution. Its open-source nature provides a comprehensive suite of tools for controlling diverse robotic systems, sensors, and teleoperation devices [1].

For autonomous mobile robots (AMR), ROS enables intelligent path planning and obstacle avoidance through the integration of packages like "Gmapping," "Map server," "AMCL," "Move base," "Joy," "Rviz," and "Darknet_ros" [1], [2], [3], [4]. These packages work in concert to allow the robot to plan the shortest path and navigate around static or dynamic obstacles autonomously.

Weakness:

ROS 1 struggles with data delivery over unreliable networks and lacks built-in security features [3]. It encounters difficulties in maintaining consistent data delivery over lossy communication links, such as WiFi or satellite connections, and presents a vulnerability with a single point of failure [3].

2.1.2 Robot Operating System 2 (ROS 2)

Building on this foundation, ROS 2 addresses reliability concerns through the Data Distribution Service (DDS), enhancing security and real-time support [3]. It utilizes User Datagram Protocol (UDP) to improve performance in wireless communications, making it suitable for a wide range of robotics applications [3]. ROS 2's redesign tackles challenges while leveraging community-driven capabilities, accommodating a diverse range of robotics applications from education and research to product development and deployment.

ROS 2's software components are categorized into Middleware for inter-component communication, Algorithms for common robotics tasks like perception and SLAM, and Developer Tools for simulation, visualization, and debugging [3]. It offers APIs that empower developers to enforce application-specific constraints in real-time systems, crucial for safety and performance in applications ranging from humanoids to self-driving cars.

Weakness:

ROS 2, despite its improvements, requires complex coding for real-time support, and the connections between nodes are often rigidly coded, contradicting fundamental ROS principles. The integration of ROS with real-time control in various lesser-known frameworks suffers from limited modularity and incomplete integration with prevalent ROS workflows.

2.1.3 Real-Time Control Framework (RTCF)

Complementing these systems, the Real-Time Control Framework (RTCF) introduces a novel approach for creating modular, high-performance controller architectures within the ROS ecosystem [5]. By leveraging the Open Robot Control Software (OROCOS) and implementing sequential execution of components in a single real-time thread, RTCF ensures low-overhead, real-time safe execution. This approach contrasts with the concurrent execution model of ROS and OROCOS, offering better deterministic behavior and minimal overhead for control applications.

RTCF aims to support control frequencies up to several kilohertz, with desired overhead and jitter within the range of 10 μ s. It prioritizes ease of use, aligning with existing ROS user concepts to facilitate a smooth learning curve for developers transitioning to real-time control applications.

Weakness:

RTCF, while innovative, may face scalability and hardware adaptability issues. Its approach of sequential execution, while beneficial for deterministic behavior, may present challenges in scenarios requiring more complex, parallel processing of multiple robot subsystems.

Despite these challenges, ongoing development continues to enhance these frameworks, pushing the boundaries of autonomous mobile robot capabilities. The robotics community's collaborative efforts are driving improvements in reliability, security, and real-time performance, paving the way for more sophisticated and adaptable robotic systems across various applications and environments.

Table 2.1 Summary Of Existing Robot Framework in Robot Development

FRAME WORK	FEATURES	IMPLEMENTATION WITH AMR	WEAKNESS
ROS [1], [2], [3], [4]	Integrate with robot technological systems <ul style="list-style-type: none"> • locomotion system (control theory) • Kinematics • Perceptions systems • localization system [1] 	Integrated with various ROS packages. [1], [2], [3], [4] <ul style="list-style-type: none"> • "Gmapping" • "Map server" • "AMCL" • "Move base" • "Joy" • "Rviz" • "Darknet_ros" 	<ul style="list-style-type: none"> • Security Concern [3] • Reliability
RTCF [5]	<ul style="list-style-type: none"> • Efficient • low overhead • ease of use 	Functional inherited from ROS	<ul style="list-style-type: none"> • Scalability, Adaptability • Hard to deal with complex scenarios
ROS 2 [3]	<ul style="list-style-type: none"> • Real-time capabilities • Multi-robot support • Communication middleware 	<ul style="list-style-type: none"> • Packages Inherited from ROS • Robot Applications • Graphical Tool (simulation) 	<ul style="list-style-type: none"> • Many still use ROS1 • Inefficient result with real-time control

Summarized approach to the current paper:

This project will used similar approach with Ros 2 framework on developing the project Operating System, Algorithms and so on. Furthermore, the ROS 2 integrated packages like Gmappping will be used for system testing purpose.

2.2 Existing Techniques in Autonomous Mobile Robot Development

In autonomous mobile robot development, perception and navigation techniques play crucial roles. Perception relies on various sensors, with LiDAR being essential for obstacle avoidance and SLAM, offering precise positioning despite sensitivity to vibrations [2], [5], [6]. Depth cameras provide RGB and depth data for navigation and real-time operations [2], [7]. Additional sensors like ultrasonic and infrared proximity detectors enhance obstacle detection in various conditions [4], [6].

Autonomous navigation integrates data from multiple sources, including IMU for real-time motion information . Key components such as AMCL and Move_Base nodes facilitate navigation . Path planning employs both global and local strategies, utilizing algorithms like A* search and Dijkstra method [4], [7], [8], [9]. AI and machine learning contribute to dynamic environment analysis and path optimization .

Object avoidance is achieved through the "move base" package and local planner algorithms . SLAM, particularly the "gmapping" method based on RBPF, enables simultaneous mapping and localization . This approach, often integrated with LIDAR, creates 2D maps for effective navigation [10].

Advanced systems like the RFID Reading System and Object Detection Module enhance path planning and obstacle detection [9]. However, challenges remain in achieving smooth robot movement and optimizing system complexity for better performance in autonomous navigation .

Despite these advancements, the field continues to evolve, addressing issues such as improving ODM accuracy and enhancing overall system efficiency in complex environments. The integration of these technologies promises more robust and adaptable autonomous mobile robots for various applications.

Table 2.2 Summary Of Existing Techniques in Autonomous Mobile Robot Development

Navigation Categories	IMPLEMENTATION WITH AMR	WEAKNESS
Path Planning [4], [7], [8], [9]	A* search algorithm Dijkstra algorithm TEB local path planner machine learning	<ul style="list-style-type: none"> • Memory-intensive for large spaces. • Uniform exploration, potentially inefficient. • Limited in highly dynamic environments. • Dependent on extensive, representative data.
Object Avoidance [4], [7],[9]	"move_base" (ROS package) TEB local planner algorithm (ROS package) RFID Radio Frequency Identification + R-CNN	<ul style="list-style-type: none"> • Configuration complexity • Limited in highly dynamic environments. • Sensitivity to interference
SLAM [4],[10],[11]	Gmapping Ros package Slam_gmapping Gmapping + Real-Time Appearance-Based Mapping(RTAB)	<ul style="list-style-type: none"> • Limited in dynamic environments. • Complexity and resource-intensive processing. • RTAB limitations in highly dynamic environments.

Summarized approach to the current paper:

The project focuses on three key areas of autonomous mobile robot development:

Path planning: Used a extended classis A* algorithm which is SMAC Hybrid-A* (Search Motion Planning and Control) for path planning

Object avoidance: The Regulated Pure Pursuit Controller is used for static, integrates with local costmap for obstacle avoidance.

SLAM: I employ the gmapping algorithm through the slam_gmapping ROS node, creating 2D maps and localizing the robot simultaneously .

2.3 Existing ROS2 App Projects

2.3.1 ROS2 Mobile Android

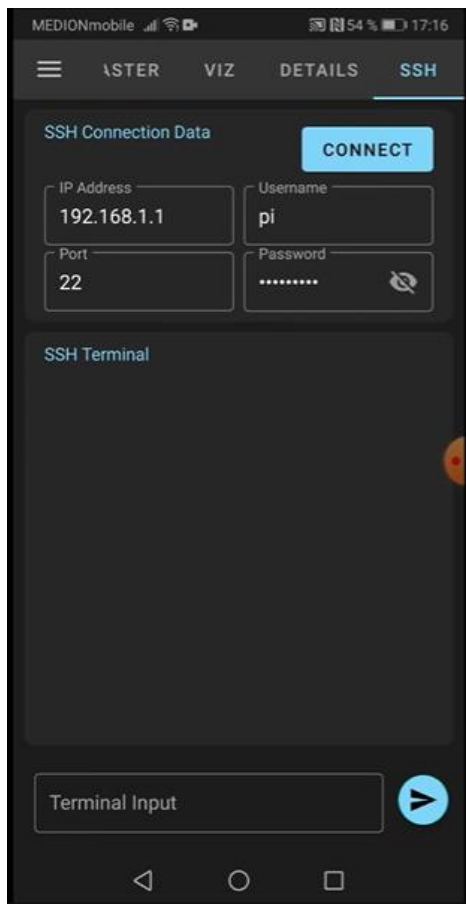


Figure 2.3.1 Establish Ssh Connection (Left)

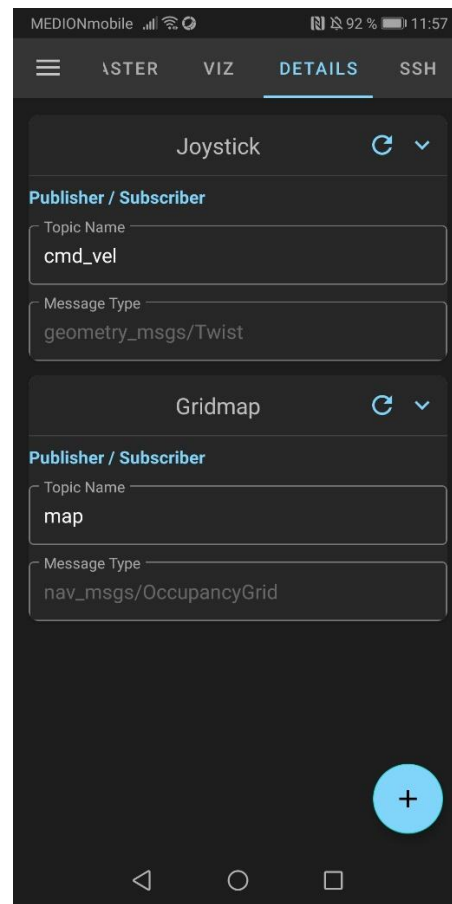


Figure 2.3.2 Initializing Joystick And Grid Map Publisher And Subscriber (Right)

The ROS2-Mobile-Android project represents a significant advancement in the integration of mobile technology with robotics has been proposed [12]. By leveraging the ROS2 framework and the Android platform, this app enables dynamic control and visualization of robots through user-friendly mobile interfaces. It incorporates essential components such as camera views, joystick controls, and GPS integration, offering a comprehensive solution for real-time robot management. Developed with ROS2 Java libraries and built using Android Studio, the app demonstrates the potential for enhancing robot interaction through mobile devices, making robotics more accessible and manageable for a broader range of users.

This work underscores the growing trend of integrating cross-platform solutions in robotics, emphasizing real-time communication and usability. By providing an intuitive interface that

interacts seamlessly with complex ROS2 systems, ROS2-Mobile-Android bridges the gap between advanced robotics and everyday mobile technology. It showcases the ongoing effort to make robotic systems more interactive and user-friendly, catering to both specialists and non-specialists alike.

Weakness:

A key weakness is the absence of several essential widgets, such as path visualization, `rqt_plot` for real-time data plotting, debugging tools, touch-goal interaction, and advanced camera features. These omissions are primarily due to constraints within the `ros2_java` library, which limits the app's ability to fully replicate the capabilities available in desktop-based ROS2 interfaces.

2.3.2 ROS2 App For Controlling A Robot via Bluetooth

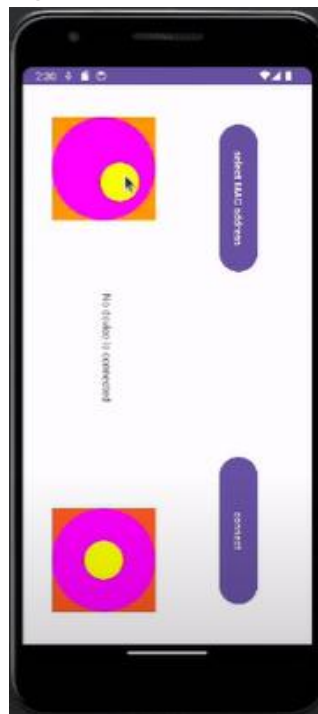


Figure 2.3.3 Robot Control Mockup

The integration of ROS2 with Android via Bluetooth connection has been proposed in this project [13] as a approach to enhance robot control and interaction. Leveraging libraries such as Android Studio and `ros2_java`, this project facilitates seamless communication between

Android devices and the ROS2 ecosystem, enabling efficient interaction with robotic systems. Bluetooth technology is employed as the connection method, offering a reliable and effective means for real-time control. In this setup, the Android application communicates with the robot by sending control commands, such as movement directions, which are processed by ROS2 nodes on the robot. The system then provides feedback to the application, including sensor data and status updates, enhancing the user's ability to monitor and manage robotic operations. Key components include the Android smartphone as the controller, the robot equipped with ROS2 for command execution, the Bluetooth module for wireless communication, and the user interface for command input and data visualization. This project demonstrates the convergence of Android ROS2 libraries and Bluetooth connectivity, highlighting the essential elements for effective control through mobile devices.

Weakness:

While smartphone integration with ROS2 offers functional control, the system is limited by its basic features, lacking advanced options or customizability. Bluetooth communication weakens with distance, causing potential delays and disconnections, which undermines reliability. Additionally, the user interface is unappealing and not user-friendly, reducing engagement and ease of use. These limitations highlight the need for enhancements to improve the system's overall effectiveness.

2.3.3 Simplest Android App To Communicate With A Ros 2 Node

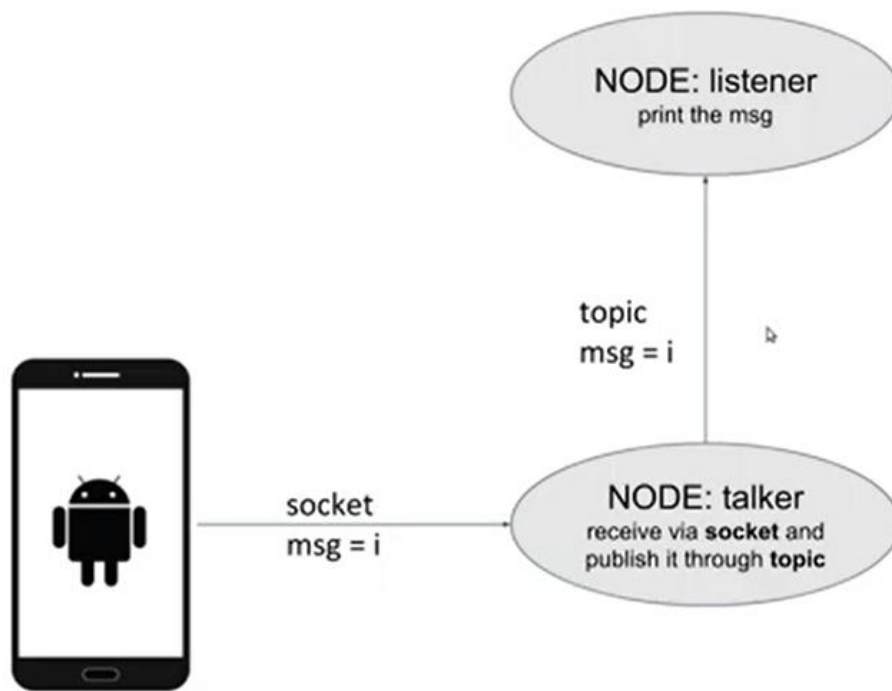


Figure 2.3.4 Block Diagram Of The Android App

This project [14] proposes a method for integrating Android applications with ROS2 systems that demonstrates an effective means of communication between Android devices and ROS2 nodes. Utilizing Android Studio and the `ros2_java` library, the approach involves sending messages from an Android app via a socket connection. These messages are received by a ROS2 node (the "talker") and published to a topic. A corresponding "listener" node subscribes to this topic, enabling the system to process and display the received messages. The setup leverages `rclcpp`, the ROS client library for C++, to write simple publishers and subscribers, facilitating seamless interaction between smartphones and robotic systems. By providing a clear configuration process, the author enhances robot-smartphone communication, showcasing a practical and scalable solution for integrating mobile devices with ROS2. This highlights the potential of such integrations to streamline robotic control through familiar mobile platforms.

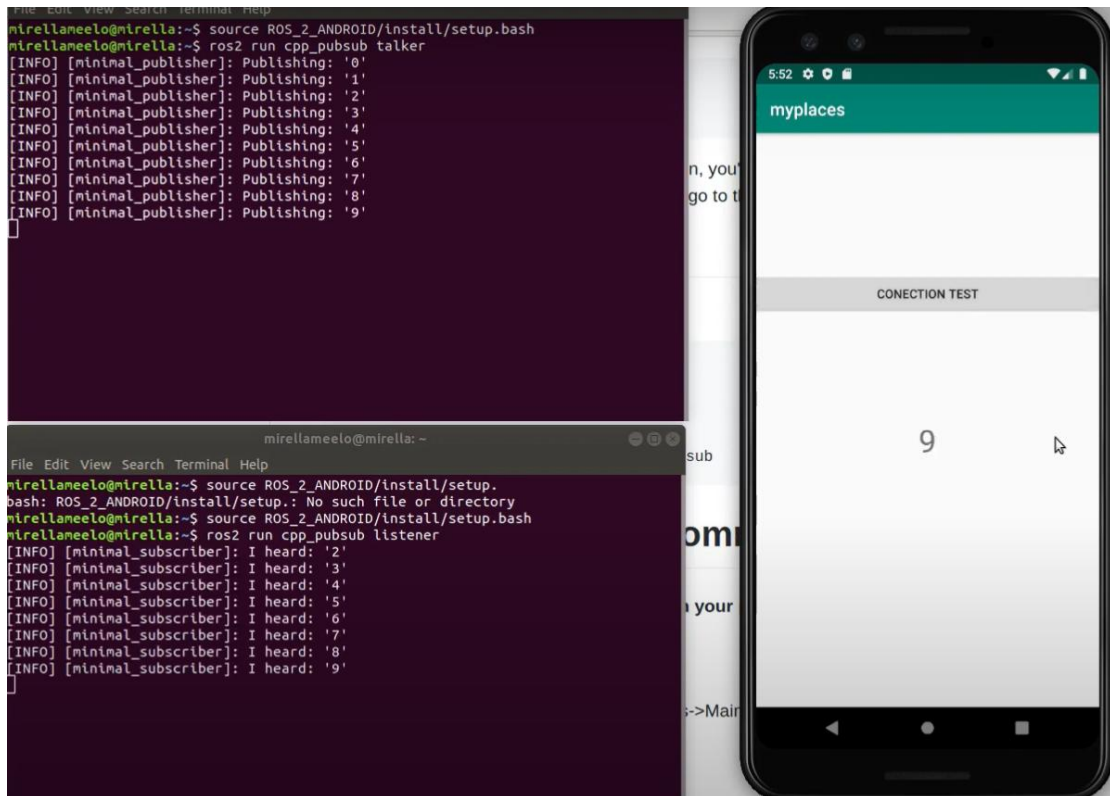


Figure 2.3.5 Showcase On How The Communication Works

Weakness:

A notable weakness of the repository is its limited scope, as it primarily focuses on demonstrating basic publisher and listener functionalities and a simple variable increment test. While effective in showing node communication, it lacks practical control methods for the robot, such as a joystick controller widget or other interactive features. This narrow focus on communication without offering real control mechanisms diminishes the application's utility for more complex robotic tasks, underscoring the need for more comprehensive demonstrations that incorporate actual robot control methods.

Table 2.3 Summary Of Existing ROS2 App Design

Project	Strength	Weakness
ROS2 Mobile Android [12]	<ul style="list-style-type: none"> • Comprehensive solution for real-time robot management • Incorporates camera views, joystick controls, and GPS integration 	<ul style="list-style-type: none"> • Lacks essential widgets (e.g., path visualization, rqt_plot) • Limited by the ros2_java library
ROS2 App for Bluetooth Control [13]	<ul style="list-style-type: none"> • Enables efficient interaction with robotic systems • Uses Bluetooth for wireless communication • Provides real-time control and feedback 	<ul style="list-style-type: none"> • Basic features • Limited by Bluetooth range and potential disconnections • Unappealing interface
Simplest Android App for ROS2 [14]	<ul style="list-style-type: none"> • Demonstrates effective communication between Android and ROS2 nodes • Provides clear configuration process • Scalable solution for integrating mobile devices with ROS2 	<ul style="list-style-type: none"> • Limited scope, focusing only on basic publisher and listener functionalities • Lacks practical control methods for robots • Minimal user interface

Summarized approach to the current paper:

This project will utilize Java programming language and socket TCP/IP login methods for connectivity, integrating ROS 2 with the java.net.Socket client library. Android Studio will be employed for designing the graphical user interface (GUI).

2.4 Previous Study On Indoor Guiding Robot Design

2.4.1 Lysa Robot Guide Dog

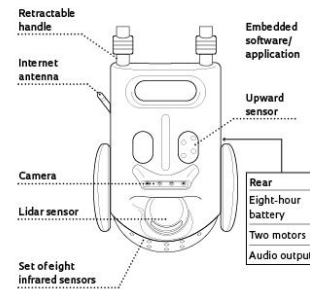
HOW LYSA WORKS

The robot uses a camera, infrared sensors, and laser beams to guide blind people

1. The user downloads the Lysa app on their cell phone. They then indicate where they want to go using voice commands or by tapping the options menu

2. The lidar sensor's laser beams and eight infrared sensors map the environment. These features also aid navigation

3. A 2D map is generated. The robot plans and follows a route to the destination while avoiding people and objects not previously mapped



4. The handle is equipped with a vibrating motor that alerts the user to obstacles ahead if they choose to turn off the audio

5. An upward-pointing infrared sensor checks for overhead obstacles and warns the user of their presence

6. The camera identifies objects during movement, such as chairs, pots, or stairs, and the user is given an audio alert

SOURCE: VIXSYSTEM

Figure 2.4.1 Lysa Robot Guide Dog

Lysa[15], a robotic guide dog developed by startup Vixsystem, is a 4-kilogram, 40-centimeter-tall device resembling a small suitcase with a retractable handle and wheels, designed to assist visually impaired individuals in navigating indoor spaces such as malls, stores, and airports. Equipped with bespoke software, a mobile app, artificial intelligence, multiple sensors, a camera, and a Lidar system, Lysa maps locations, traces routes, and guides users to their destinations. It provides precise directions, identifying and avoiding obstacles both in front of and above the user, and alerts the user through audible warnings and vibrations, offering a smarter alternative to traditional canes.

For the technological aspects, they are employing a multi-faceted approach to navigation and obstacle avoidance. At its core, Lysa utilizes a sophisticated sensor fusion system, integrating GPS, Lidar, cameras, and a sensor screen to create a comprehensive environmental awareness. This is complemented by advanced artificial intelligence algorithms for environment mapping through point triangulation, object recognition, and adaptive path planning. One of Lysa's key advantages is its ability to detect aerial obstacles, a feature absent in traditional guide dogs and many existing assistive devices.

Bachelor of Computer Science (Honours)

Faculty of Information and Communication Technology (Kampar Campus), UTAR

Weakness:

Implementing AI, integrating various sensors, and enabling real-time navigation and object avoidance demand substantial computational power. This significantly increases the production costs, making the product less affordable and accessible to the average consumer. Indoor-only navigation capability and high price point of R\$15,000(MYR 14,100) per unit. The startup is addressing challenges for outdoor use, such as integrating GPS and navigating areas without sidewalks.

2.4.2 Design of a Portable Indoor Guide Robot for Blind People

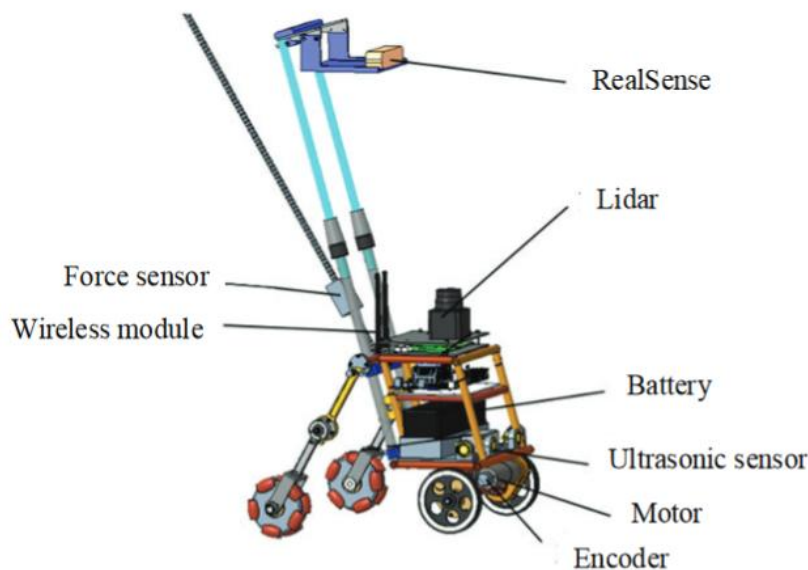


Figure 2.4.2 The Design Of The Robot

Lai et al. [16] present a novel guide robot designed to assist blind individuals in indoor navigation. This wheeled mobile robot addresses limitations of traditional assistive devices, offering a more affordable, portable, and technologically advanced solution. The robot features a foldable design, weighs approximately 6 kg, and employs a four-wheel configuration for stability. At its core, the robot utilizes a sophisticated sensor fusion system, integrating Lidar, a depth camera, RGB camera, ultrasonic sensors, and a force sensor. This multi-sensor approach enables comprehensive environmental awareness and user interaction. The robot's computation is handled by an embedded computer (ECM-SKLU) running Ubuntu and ROS, providing a flexible framework for navigation and control. A key innovation is the traction

system, which combines an elastic rope with a force sensor, offering user-friendly control and self-adaptive length expansion.

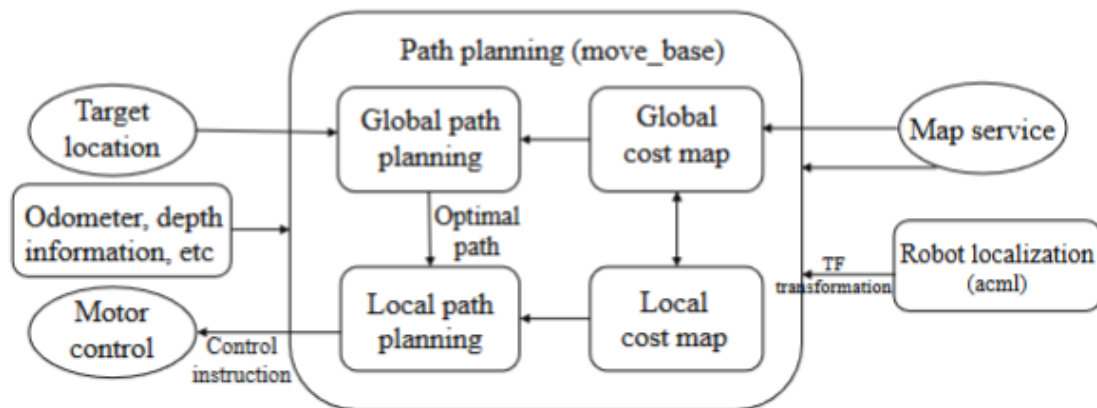


Figure 2.4.3 Autonomous Navigation Framework In ROS

The robot's navigation techniques include map building, localization using ROS AMCL, and both global and local path planning for efficient route calculation and obstacle avoidance. The multi-layer perception system (bottom, middle, top) addresses user concerns about overhead obstacles. User interaction is facilitated through force-based feedback, allowing real-time speed adjustments based on the user's traction force. Experimental results demonstrated the robot's effectiveness in office and factory environments, maintaining stable traction force and speed during guided walks at various speeds (0.5 m/s and 1 m/s).

Weakness:

Limited user interaction, while the force sensor and elastic rope provide some user input, the system lacks more advanced interaction methods like voice commands or tactile feedback, which could enhance user experience and control.

2.4.3 The Construction of Occupancy Grid Map with Semantic Information for the Indoor Blind Guiding Robot

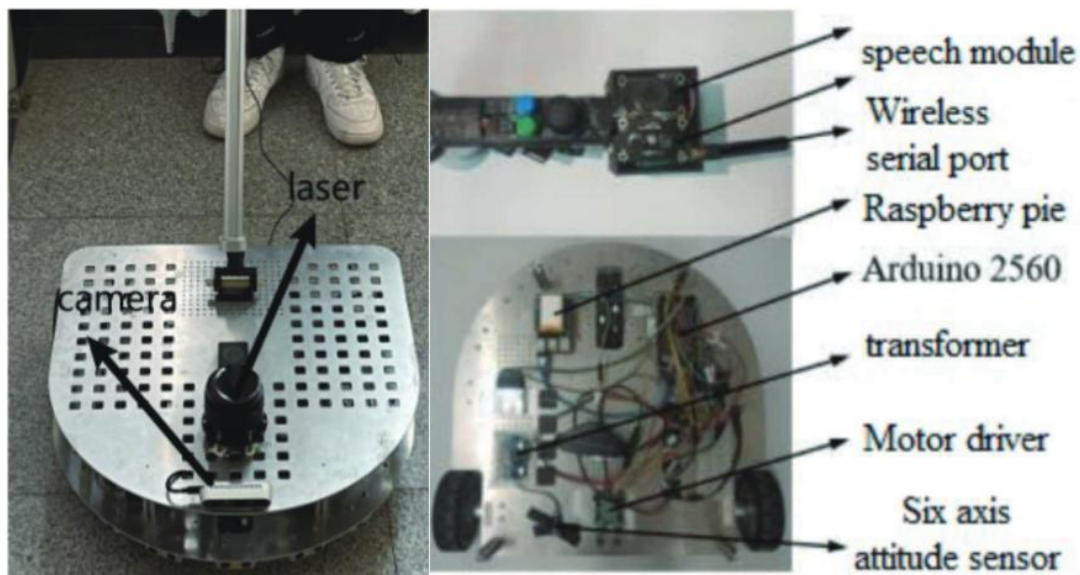


Figure 2.4.4 System Hardware

Wu et al. [17] present an innovative approach to constructing occupancy grid maps with semantic information, addressing limitations in traditional mapping techniques for autonomous navigation. Their method, SOGM (Semantic Occupancy Grid Map), integrates semantic segmentation with occupancy mapping to enhance environmental understanding for robotic systems. For real-time pose estimation and map construction, the authors employ a 2D laser SLAM approach, specifically using an improved and fine-tuned version of the open-source Google Cartographer algorithm. Cartographer, known for its graph optimization capabilities and loop detection, enables the construction of globally consistent maps. The SLAM process incorporates both IMU and lidar data, with local map updates and global loop closure detection to correct cumulative errors. For obstacle detection, the system utilizes the SSD (Single Shot Detection) algorithm combined with the MobileNet lightweight neural network, achieving fast and accurate target detection. This visual data is then combined with depth information from a depth camera to perform visual ranging on obstacles. The authors employ a sophisticated sensor fusion system, along with a ZED2 stereo camera. The SOGM framework further leverages deep learning techniques, specifically a modified U-Net architecture based on ERFNet, to perform semantic segmentation on camera images. This segmentation information is projected onto the 3D point cloud data from the lidar, creating a semantically enriched

representation of the environment. The occupancy grid map is constructed using a probabilistic approach, where each cell's occupancy probability is updated based on both lidar measurements and semantic labels. This integration allows for more nuanced mapping, distinguishing between static obstacles, dynamic objects, and traversable areas. The system operates in real-time with update rates of 10 Hz and can identify obstacles as small as 10 cm.

Weakness:

The occupancy grid with semantics might become computationally expensive for large or complex environments. Reliance on a third-party SDK (XunFei) could introduce compatibility issues or limit customization options.

2.4.4 Autonomous Indoor Navigation for Visually Impaired: A ROS-Based Intelligent Guide Robot with Deep Learning and SLAM Integration

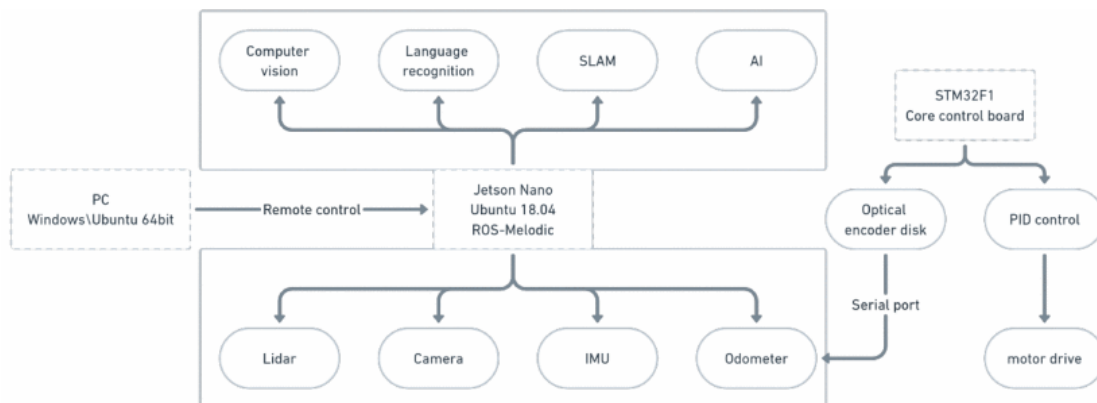


Figure 2.4.5 Technical Framework of the system

Zhang et al. [18] introduce an innovative approach to indoor navigation for visually impaired individuals by integrating advanced robotics and AI technologies. The robot architecture leverages the Robot Operating System (ROS) and deep learning techniques, utilizing a Jetson Nano for main control and STM32F1 series for motor control, operating on ROS-Melodic. Key technologies include Motor Disc Odometry, which provides precise movement tracking through a kinematic model, and SLAM, which processes data from odometers, IMU, and lidar to generate environmental maps and track motion trajectories. The robot's autonomous navigation relies on a 2D navigation package, while object detection is handled by the tiny-

YOLO v7 algorithm within the ROS framework. Comprehensive testing of the system demonstrated high navigation accuracy, robust obstacle avoidance, and effective object detection and tracking, with over 90% recognition rates in non-extreme lighting conditions. Voice interaction was also tested, showing quick response times in both quiet and noisy environments. Zhang et al.'s work significantly advances assistive technology for visually impaired individuals by offering a comprehensive solution that combines precise navigation, efficient obstacle avoidance, and user-friendly voice interaction. They highlight future research directions, including refining SLAM for better mapping and optimizing voice recognition in noisy settings. This study provides valuable insights into the field of assistive robotics, addressing critical challenges in indoor navigation for the visually impaired and enhancing their mobility and independence.

Weakness:

While the voice interaction system performs well in noise-free environments, its response time increases significantly in noisy conditions (from 1.5 seconds to 2-3 seconds). This could be problematic in busy or loud indoor spaces.

2.4.5 Exploring Levels of Control for a Navigation Assistant for Blind Travelers

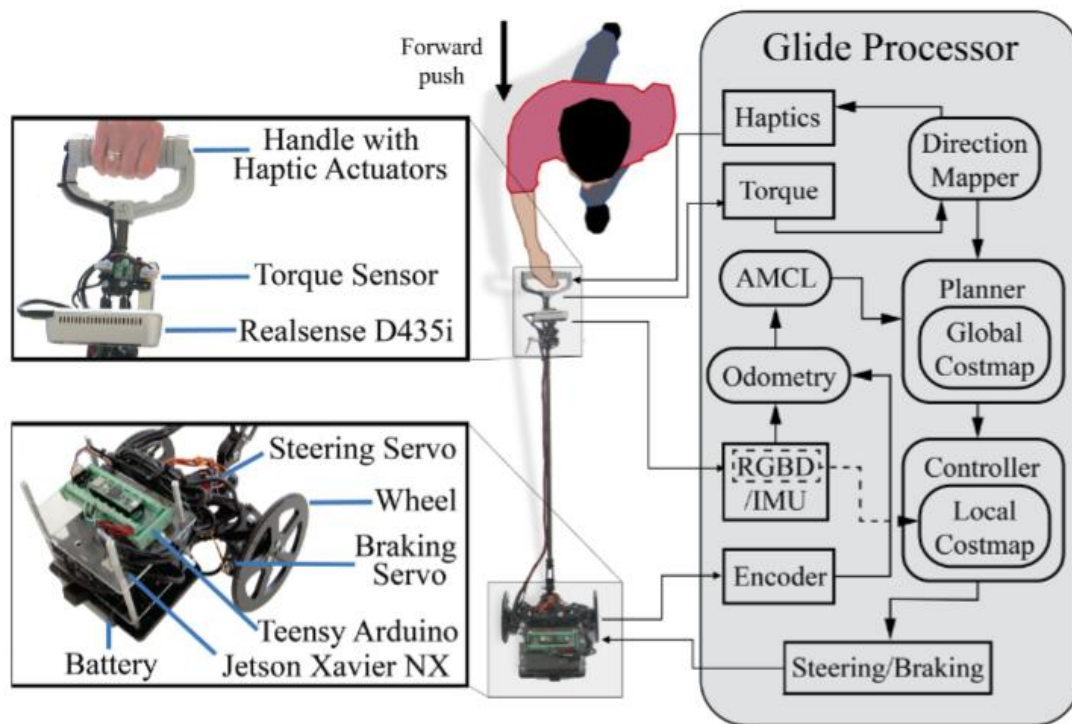


Figure 2.4.6 Components Breakdown Of Glide

Ranganeni et al. [19] present Glide, an innovative robotic navigation assistant designed for visually impaired individuals. Glide represents a novel approach to assistive technology, emphasizing user-centered design and shared control. The system consists of a passive robot that the user pushes, featuring a pole connected to a small platform with steerable, brakeable, and unpowered wheels. This design choice aims to reduce physical workload while maintaining user control over movement pace, addressing limitations of motorized assistants. A Realsense D435i camera detects obstacles, while an IMU and wheel encoders compute odometry. The system uses servos for braking and steering, and a Jetson Xavier NX provides onboard processing. User input is facilitated through a handle equipped with a torque sensor for detecting directional input via twisting. Notably, Glide features a linear array of six vibrotactile actuators in the handle, providing haptic feedback to the user through intuitive spatial patterns. This haptic system aims to reduce cognitive load, function in noisy environments, and avoid overburdening the user's auditory senses.

Glide's navigation system is built on ROS2 (Robot Operating System), specifically utilizing the Nav2 package. It employs adaptive Monte Carlo localization (AMCL) for positioning and a Regulated Pure Pursuit planner for local navigation. The system relies on a pre-rendered floor

plan as a global map, with global plans generated offline for the study. A key aspect of Glide's design is its incorporation of human-in-the-loop control, necessitating a navigation system that can adapt to user input and modify global plans accordingly.

Weakness:

Speed Limitation: The controller struggled with latency issues when users walked too quickly, leading to difficulties in avoiding obstacles. Users often had to stop and back up, and many expressed a desire to walk faster.

2.5 Summary

Table 2.5 Summary Of Previous Study On Indoor Guiding Robot Design

	<i>Mobil e App</i>	<i>Real-time object avoidance</i>	<i>Wheeled robot</i>	<i>Voice feedback interaction</i>	<i>Ai implementatio n</i>	<i>Cost</i>	<i>Haptic feedba ck</i>
<i>Lysa Robot Guide Dog[15]</i>	Yes	Yes	Yes	Yes	Yes	High	Yes
<i>Lai et al[16].</i>	No	Yes	Yes	No	No	Low	Yes
<i>Wu et al[17].</i>	No	Yes	Yes	Yes	Yes	Medium	No
<i>Zhang et al[18].</i>	No	Yes	Yes	Yes	Yes	Medium	No
<i>Ranganeni et al.[19]</i>	No	Yes	Yes	No	No	Medium	Yes

This project aims to propose an autonomous mobile robot that could act as an assistant or movable toolbox to the human. In order to design such a project, I will list out some proposed solutions.

In this project, ROS 2 framework will be adopted to build a comprehensive system that integrates core components such as the operating system, algorithms, and connectivity features. Proposed approach focuses on three key areas of autonomous mobile robot development: path planning, object avoidance, and SLAM.

For path planning, the SMAC Hybrid-A* algorithm serves as the global path planner, generating kinematically feasible paths optimized for non-holonomic robots using Reeds-Shepp curves. This is paired with the Regulated Pure Pursuit Controller for local trajectory execution and real-time motion adjustments. The dual-layer approach ensures efficient global route planning while enabling dynamic responsiveness to environmental changes. Object avoidance is implemented through the Regulated Pure Pursuit Controller's cost-aware velocity scaling, which continuously adapts the robot's speed and path following based on real-time costmap data.

The Android app will utilize socket TCP/IP login methods, leveraging java.net.Socket client library for seamless communication with the robot. Android Studio will be the primary tool for GUI development, ensuring a user-friendly interface.

The system is designed to be cost-effective while incorporating advanced features, including object avoidance, voice feedback when sensing obstacle, all of which contribute to enhancing the user experience and interaction with the robot. This integrated approach aims to create an efficient and adaptive autonomous navigation system for mobile robots, combining the power of ROS 2 with sophisticated algorithms for navigation and mapping.

CHAPTER 3 SYSTEM METHODOLOGY/APPROACH

3.1 System Design Diagram

3.1.1 System Architecture Design

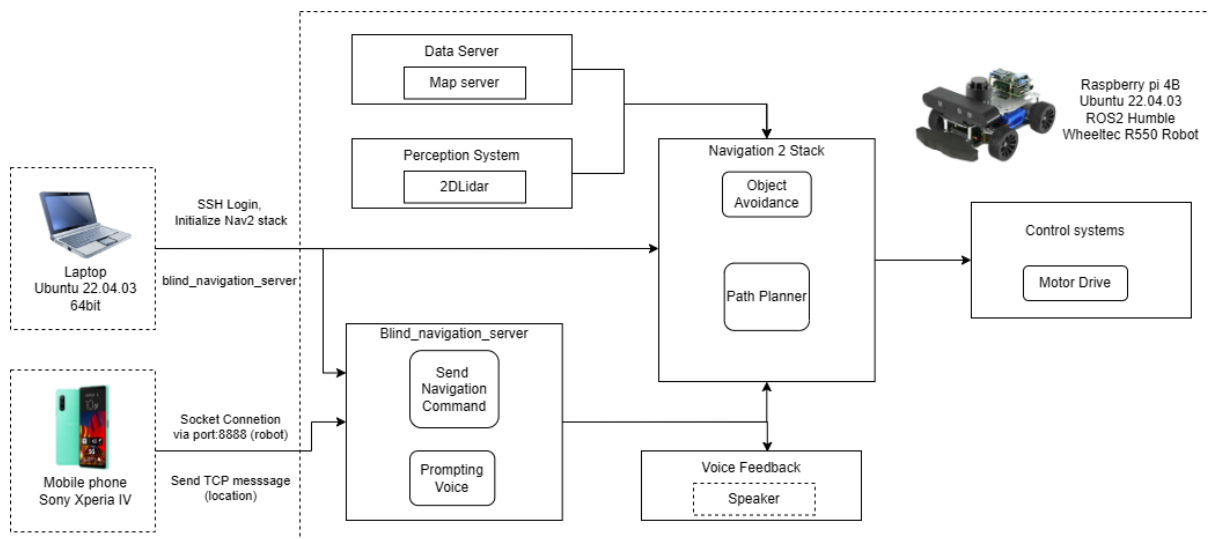


Figure 3.1.1 System Architecture Diagram

The proposed guide robot system presents an innovative and practical solution to assist visually impaired individuals in navigating indoor environments safely and autonomously. This system architecture is carefully designed to integrate a range of hardware and software components, coordinated through the ROS 2 framework (Humble distribution), running on a Raspberry Pi 4B-based Wheeltec R550 robot. The architecture is centered around three primary components: a laptop that serves as the initialization interface, a mobile phone for user interaction and control, and the autonomous robot itself, which performs the core navigation tasks.

The operational workflow begins with a laptop running Ubuntu 22.04.03, which connects to the robot via SSH on robot hotspot network. This interface is responsible for initializing the **Navigation 2 (Nav2) stack** and activating the required ROS 2 nodes. It also has the capability to launch the robot's map server. The map server utilizes a pre-saved map generated by **SLAM**

using the **gmapping** algorithm, enabling the robot to localize itself accurately in a previously mapped environment. Once initialized, the robot can operate independently, guided by real-time data from its onboard perception system.

The perception system of the robot is powered by a **high-precision 2D LiDAR sensor**, which continuously scans the surrounding environment to generate a 2D representation of nearby objects and obstacles. This LiDAR data feeds directly into the Nav2 stack, where it supports both localization and obstacle detection functionalities. Within the Nav2 stack, two critical modules operate in parallel: the Path Planner and the Object Avoidance system. The path planning module uses the SMAC Hybrid-A* algorithm. This algorithm generates smooth and feasible global paths by utilizing Reeds-Shepp curves to accommodate the kinematic constraints of the robot. At the same time, the Object Avoidance module is managed by the Regulated Pure Pursuit Controller, which adapts the robot's trajectory in real time. This controller makes intelligent decisions based on dynamic costmaps, allowing the robot to adjust its speed and heading to avoid unexpected obstacles while still following the global path.

To enhance user accessibility, a mobile phone application running on a Sony Xperia IV is integrated into the system as the primary user interface. The phone communicates with the robot over a local Wi-Fi network using a TCP/IP socket connection on port 8888. Through this app, users can input navigation commands, such as selecting a destination from a list of predefined locations. These commands are transmitted in real time to a custom Python script running on the robot, known as the `blind_navigation_server`. This script is responsible for receiving TCP messages, processing the incoming navigation commands, and interfacing directly with the Nav2 stack to initiate autonomous navigation based on the user's input.

In addition to its core navigation features, the system incorporates an audio feedback mechanism to further support users with visual impairments. A speaker is connected to the robot and controlled by the `blind_navigation_server`. This speaker delivers real-time voice prompts to inform the user about the status of the navigation process. For example, it can announce when the robot has received a destination command, is beginning to move, encounters an obstacle, or has successfully arrived at the destination. These audio cues are critical in providing reassurance and situational awareness to the user throughout the navigation journey..

3.2 Use Case Diagram

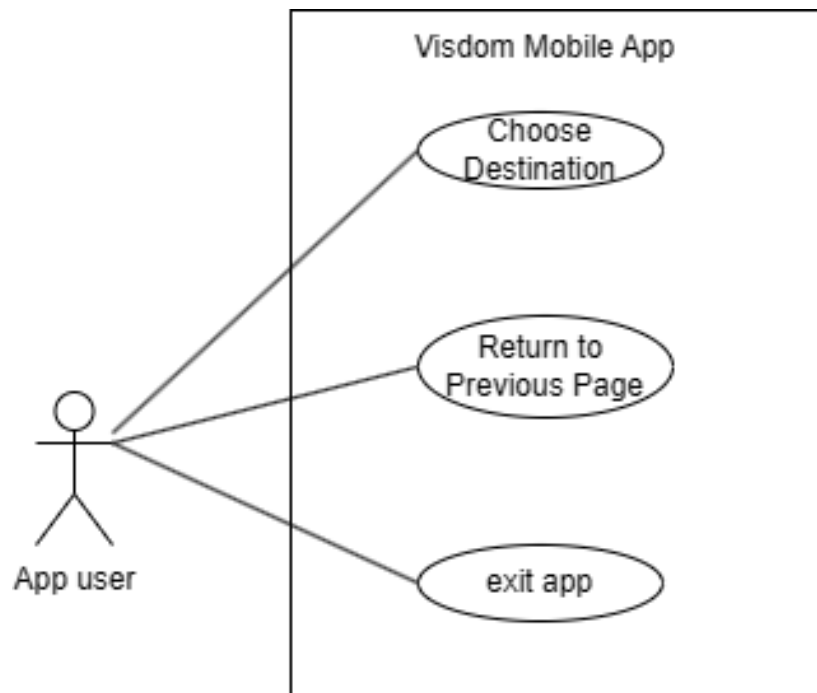


Figure 3.2 Mobile App Use Case Diagram

The use case diagram for the Visdom Mobile App highlights the essential interactions between the app user—typically a visually impaired individual—and the mobile interface designed to control the guide robot system. The diagram captures three primary functions available to the user within the app. First, the user can choose a destination, which serves as the starting point for initiating autonomous navigation with the robot. If needed, the user can return to the previous page, offering flexibility to revise or cancel their choice before proceeding. Finally, the user has the option to exit the app, ensuring a clear and straightforward way to terminate the session once navigation is no longer needed. This minimal yet functional design prioritizes accessibility, ease of use, and user autonomy, making it well-suited for the needs of visually impaired individuals relying on robotic assistance.

Use Case Description

Table 3.2.1 Choose Destination Use Case Description

Name	Choose Destination	
Actor	User	
Description	Use cased to enable the user to choose the destination.	
Main Path	1	User presses on start navigating button
	2	System displays list of destination and return button on next page
	3	User selects one of the destination button.
Alternative Path	-	-

Table 3.2.2 Return to Previous Page Use Case Description

Name	Return to Previous Page	
Actor	User	
Description	Use cased to enable the user to Return to main menu.	
Main Path	1	User presses on start navigating button
	2	System displays list of destination and return button on next page
	3	User selects the return button
Alternative Path	-	-

Table 3.2.3 Choose Destination Use Case Description

Name	Exit App	
Actor	User	
Description	Use cased to enable the user to quit the app.	
Main Path	1	User presses on quit button on the main menu.
Alternative Path	-	-

3.3 Activity Diagram

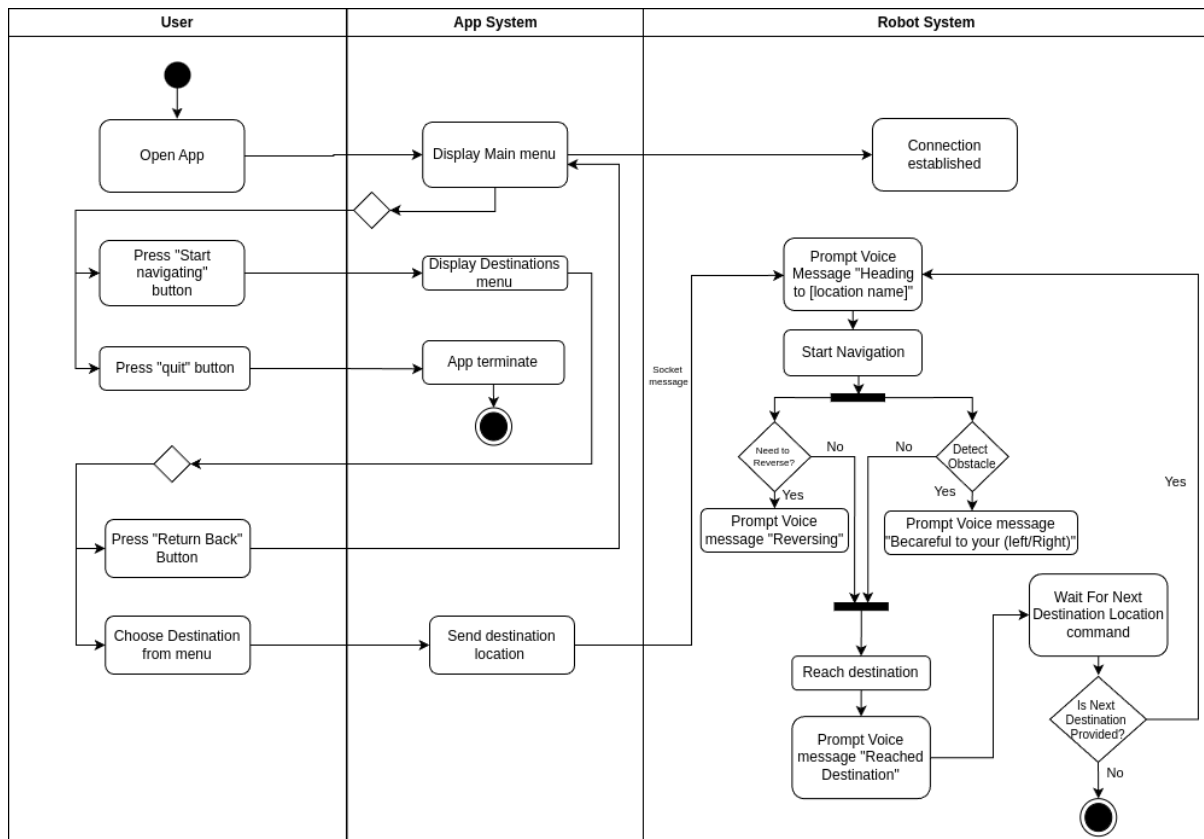


Figure 3.3 Mobile App Activity Diagram

In the proposed system, I have designed an activity diagram that illustrates the interaction between the user, the Visdom mobile app, and the guide robot system. The process begins when the user opens the mobile application. The app then displays the main menu, providing options such as starting navigation or exiting the app. If the user chooses to start navigating, the application transitions to the destination menu, where the user can browse available locations.

Once a destination is selected from the menu, the app sends this location to the robot using a socket message. Meanwhile, on the robot side, once a connection is established, the robot prepares to receive the destination input. Upon receiving it, the robot announces via voice prompt that it is heading to the selected location, then begins autonomous navigation. When the robot try to reverse it will prompt a reversing voice message, if it detects obstacles nearby left or right it will also prompt the voice message to alert the user. When the robot reaches the destination, it informs the user with another voice message indicating arrival.

After arriving, the system enters a standby state where it waits for the next destination command. A decision point checks whether another destination is provided. If yes, the robot continues the navigation cycle; if not, the system ends its operation flow.

The app also includes functions for returning to the previous menu and exiting the application at any point, ensuring flexibility and user control. This diagram effectively communicates the sequential flow and coordination between components, as well as potential paths the user can take, such as quitting or returning during navigation setup. It clearly shows the structured and responsive design of the system to ensure a smooth and user-friendly experience for visually impaired users.

CHAPTER 4 SYSTEM DESIGN

4.1 System Block Diagram

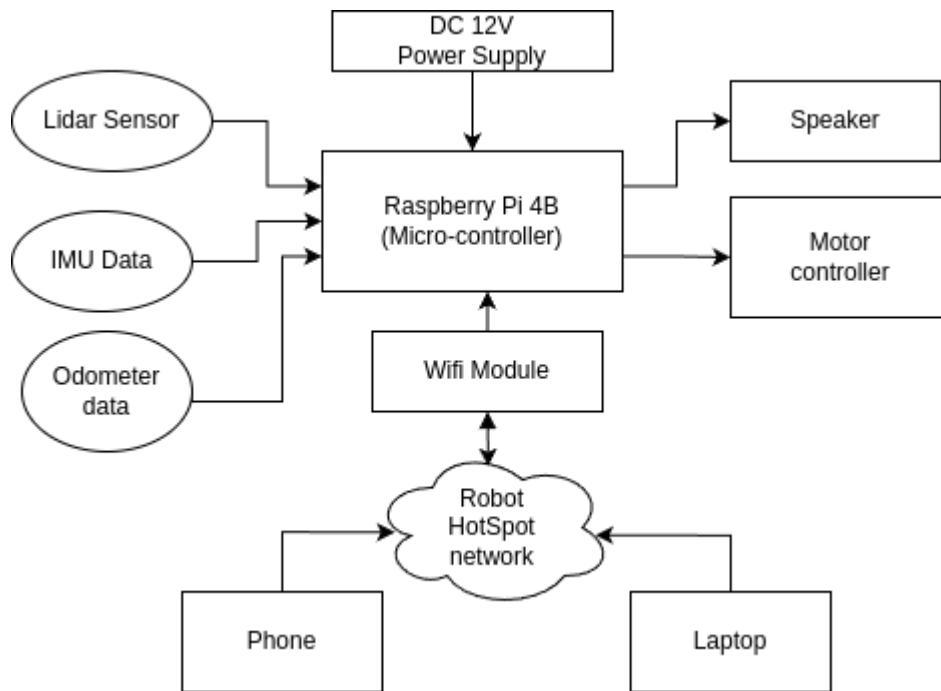


Figure 4.1.1 System Block Diagram

The system block diagram presented above illustrates the complete hardware architecture of the autonomous guide robot developed for assisting visually impaired individuals in indoor navigation. This modular design integrates several key components that work together seamlessly to enable real-time perception, decision-making, motion control, and user interaction.

At the core of the system is the **Raspberry Pi 4B**, which serves as the primary microcontroller and computational unit. This compact but powerful board is responsible for executing all core functionalities, including sensor data processing, navigation control, and wireless communication. It runs the ROS 2 (Robot Operating System 2) framework, which enables modular and scalable system design, making it easier to implement complex features such as mapping, localization, path planning, and user interaction.

Power is supplied through a **DC 12V power supply**, which is distributed appropriately to all the components in the system. The Raspberry Pi, as well as the motor controller, LiDAR, and other sensors, draw power from this centralized source. This ensures stable operation and supports the mobility of the robot without relying on external infrastructure.

The perception capabilities of the robot are achieved through a combination of three key sensors. First, a **LiDAR sensor** (Light Detection and Ranging) provides 2D environmental scanning, which is essential for tasks such as obstacle detection, localization, and building an internal map of the environment. This data forms the basis of the robot's ability to move autonomously and safely in dynamic spaces. Second, **IMU data** (Inertial Measurement Unit) is used to detect the robot's orientation and acceleration, which helps in estimating the motion and maintaining balance. Finally, **odometer data** from the wheels is used to track how far the robot has moved. Together, these sensors contribute to a sensor fusion process, where data is combined to achieve accurate real-time positioning and reliable navigation performance.

For motion control, the Raspberry Pi communicates with a **motor controller** that drives the robot's motors. The controller interprets velocity commands issued by the navigation stack and converts them into motor actuation signals, allowing the robot to follow planned trajectories accurately. The motor controller ensures smooth acceleration, deceleration, and directional changes, which are essential for guiding users safely through various environments.

To facilitate user interaction, especially for visually impaired individuals, a **speaker** is integrated into the system. This speaker provides **voice feedback**, which plays a critical role in informing the user about the robot's status. For example, the robot can announce when it has started moving, when it encounters an obstacle, and when it reaches the destination. These auditory cues enhance the user experience by providing real-time guidance and situational awareness without requiring visual input.

Wireless communication is established through a **WiFi module**, enabling the Raspberry Pi to act as a wireless access point. The robot creates a **dedicated hotspot network**, which both the user's **mobile phone** and a **laptop** can connect to. The mobile phone runs a custom-built application designed for the user to interact with the robot. Through this app, users can select destinations, send commands, and receive status updates. Communication is handled using TCP socket messages, ensuring real-time, low-latency transmission of commands and responses. The laptop, on the other hand, is primarily used for administrative purposes such as

launching the robot system via SSH, monitoring the robot's internal state, and performing debugging or maintenance operations when necessary.

Overall, this system block diagram highlights a thoughtfully constructed robotic platform where sensing, computation, control, and communication are tightly integrated. The design emphasizes ease of use, reliability, and accessibility, making it highly suitable for real-world assistive applications. By combining modern hardware with open-source software and wireless connectivity, the guide robot is positioned as a powerful tool to enhance mobility and independence for users with visual impairments.

System Flow Diagram

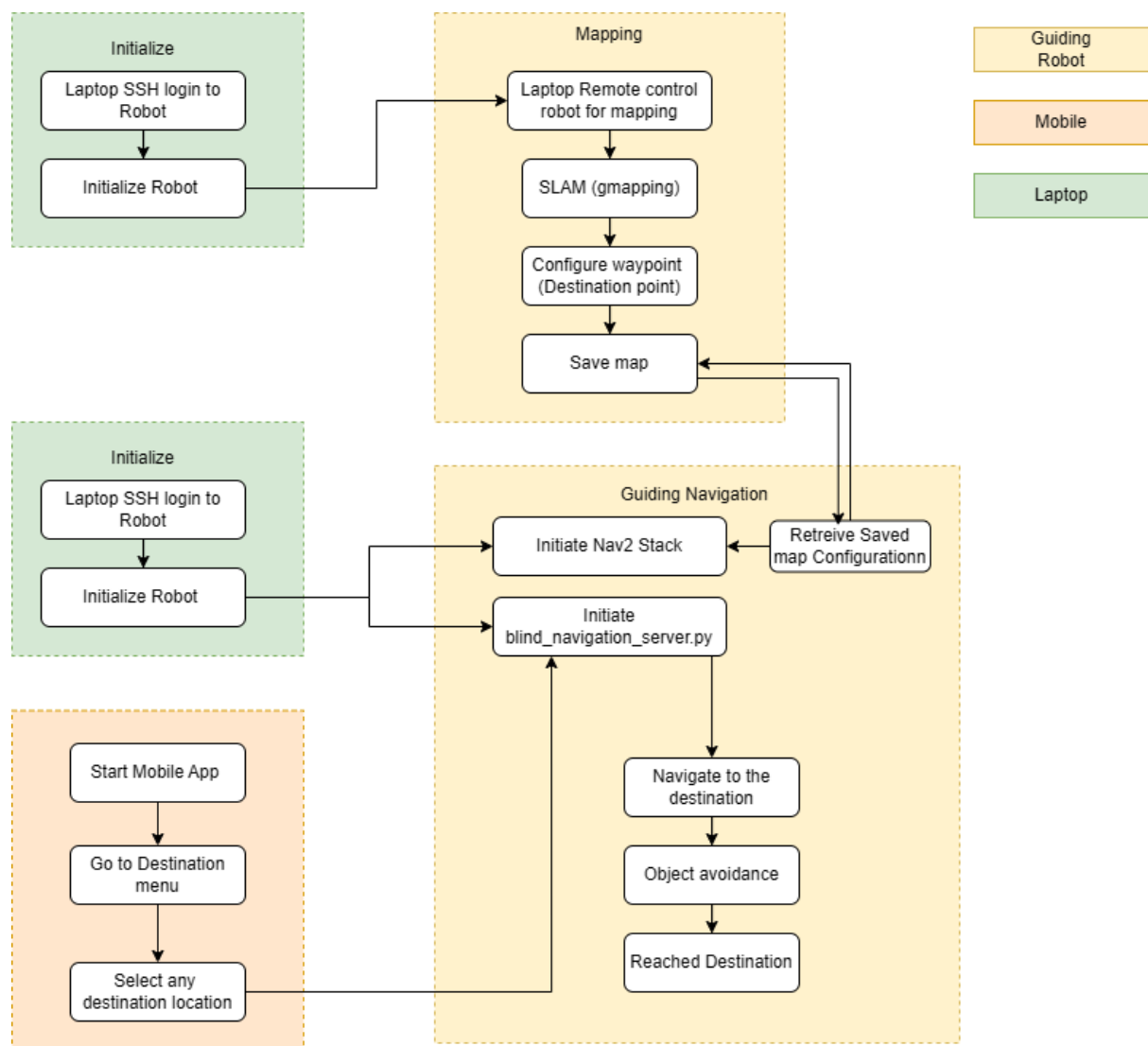


Figure 4.1.2 System Flow Diagram

The Visdom guide robot system is organized into two main phases: Mapping and Guiding Navigation. Both phases require the robot to be remotely accessed and initialized through a laptop via SSH connection over the robot's hotspot network. This setup ensures centralized control and data handling across mapping, path planning, and navigation tasks.

In the Mapping phase, the user begins by connecting the laptop to the robot's hotspot and initiating an SSH session. Through this connection, the robot is initialized and remotely controlled for mapping purposes. Using the ROS 2 framework and the gmapping SLAM algorithm, the robot collects real-time LiDAR scan data as it is manually navigated through the environment. This data is processed onboard to construct a 2D occupancy grid map. Once mapping is complete, the operator configures specific destination waypoints within the map. These waypoints, along with the map, are then saved as static data files which will later be retrieved by the navigation system.

In the Guiding Navigation phase, the same process begins with the laptop re-establishing an SSH connection to the robot's hotspot network. The robot is initialized once more, and the previously saved map configuration is loaded into the Nav2 stack. At this stage, the `blind_navigation_server.py` script created in robot is also launched. This Python server listens for destination commands sent via TCP socket from the mobile application. On the user side, the Android mobile app is started, and the user selects a destination using a TalkBack-accessible interface. The selected destination label is transmitted to the robot through a TCP socket, where the script parses the message and matches it to its predefined waypoint coordinates.

The data flow continues as the robot uses the SMAC Hybrid-A* algorithm to compute a global path from its current position to the selected goal, based on the static map. The Regulated Pure Pursuit controller then handles real-time local path execution, using LiDAR data for dynamic obstacle detection. Throughout navigation, the robot provides the user with voice prompts (e.g., "Heading to kitchen", "Be careful on your left") via its onboard speaker. Once the robot reaches the destination, it sends a final voice prompt to inform the user. This systematic data flow enables the robot to function as an intelligent, interactive guide, helping visually impaired users navigate safely in indoor environments.

4.2 System Components Specifications

Table 4.2.1 Specifications of laptop

Description	Specifications
Model	Illegear Raven-SE
Processor	Intel(R) Core(TM) i5-9300H
Operating System	Windows 11 (Ubuntu 22.04.03 in VMware Prostation)
Graphic	NVIDIA GeForce GTX 1050
Memory	16GB DDR4 RAM
Storage	1TB SATA HDD

Table 4.2.2 Specification of Mobile Phone

Description	Specifications
Model	Sony Xperia 10 IV XQ-CC72
Chipset	Qualcomm SM6375 Snapdragon 695 5G (6 nm)
Operating System	Android 14
CPU	Octa-core (2x2.2 GHz Kryo 660 Gold & 6x1.7 GHz Kryo 660 Silver)
GPU	Adreno 619
Memory	6GB
Storage	128GB

Table 4.2.3 Specifications of microcontroller

Description	Specifications
Model	Raspberry Pi 4 Model B
Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Connectivity	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports

	2 × USB 2.0 ports
GPIO:	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Memory	4GB
SD card support	Micro SD card slot for loading operating system and data storage
Input power	5V DC via USB-C connector (minimum 3A1) 5V DC via GPIO header (minimum 3A1)

Table 4.2.4 Specifications of Wheeltec R550 ROS2 Robot

Description	Specifications
Model	Ackermann steering
Wheel	65mm rubber wheel
Servo	S20F Digital Servo
Car weight	1.8kg
Load Capacity	3kg
Maximum Speed	1.2m/s
Endurance (no load 0.45m's)	7h
Motor	MG513 Metal Gearbox Motor
Encoder	500 line AB phase Incremental Photoelectric Encoder
STM32 Main Control	STM32F407VET6
Laser	Wheeltec N10P
Camera	Astra Series RGB Depth Camera
Microphone	M260C microphone array
Speaker	Wheeltec speaker

4.3 System Components Interaction Operations

In this blind navigation system, the seamless interaction between hardware components is critical for ensuring real-time control, environmental awareness, and enhanced safety for visually impaired users. The system is built around four primary hardware components: an **Android mobile phone (Sony Xperia 10 IV)**, a **robot control laptop (Illegear Raven-SE)**, an **onboard embedded controller (Raspberry Pi 4 Model B)**, and the **Wheeltec R550 ROS2 robot platform**. The **Sony Xperia 10 IV smartphone**, equipped with Android 14 and a Snapdragon 695 5G chipset, serves as the user interface. Through a custom-built Android application, the user or caretaker can issue location commands such as “kitchen” or “toilet.” These commands are transmitted using **TCP socket communication over robot hotspot** to the robot’s Raspberry Pi controller. The app will also feature text talkback functionality whereas when user press and hold the button it will prompt a voice to describe the message. It also works seamlessly with google TalkBack accessibility function. At the heart of the robot is the **Raspberry Pi 4 Model B**, featuring a **quad-core Cortex-A72 processor** and **4GB RAM**, which acts as the onboard processing unit. It handles navigation logic, speech output, and sensor data processing. Communication between the mobile app and the Raspberry Pi is established via a **TCP socket interface**, enabling fast and efficient command transmission over the Robot Hotspot network.

The **Wheeltec R550 ROS2 robot**, equipped with **Ackermann steering**, **65mm rubber wheels**, and an **MG513 gearbox motor**, executes navigation commands using the **ROS 2 Nav2 stack**. Navigation is powered by components like the **RegulatedPurePursuitController** and **SmacPlannerHybrid**, well-suited for Ackermann kinematics. The robot's localization is managed by **AMCL**, using odometry and LIDAR data from the **Wheeltec N10P laser scanner**. For obstacle detection and interaction, the system continuously monitors the `/scan` topic to evaluate nearby surroundings. The **LIDAR sensor** scans the environment, and laser data is processed to check left and right sectors for potential hazards. If any object is detected within a **0.5-meter threshold**, the system provides auditory cues like “Be careful to your left” or “Be careful to your right” through the **Wheeltec onboard speaker**, powered by a speech engine (spd-say) running on the Raspberry Pi.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1 Hardware Setup



Figure 5.1.1 Power On the Robot

System Integration and Setup

The integration of hardware components and the setup of various subsystems form a critical phase in the development of guide robot. This section details the process of assembling and configuring the robot's core elements.

Hardware Integration. The Robot serves as the foundation of the system. As shown in Figure 5.1.1, the robot's chassis houses several key components:

Lidar Sensor: Mounted at the top, this sensor is crucial for environmental mapping and obstacle detection.

Raspberry Pi: Visible in the center, this acts as the robot's brain, running ROS 2 and managing various subsystems.

Motor Controllers: These regulate the robot's wheel movements for precise navigation.

Power Management System: Ensures stable power supply to all components.

Bachelor of Computer Science (Honours)

Faculty of Information and Communication Technology (Kampar Campus), UTAR

Audio System: Integrated for providing voice feedback to users.

The integration process involved careful mounting of each component to ensure optimal performance and minimal interference between systems.



Figure 5.1.2 Modified with Metal Bracket



Figure 5.1.3 Finalized Robot Guiding Setup

Mechanical Mounting of the Guiding Stick

To enable physical guidance and tactile feedback for the user, a mechanical guiding stick is securely mounted onto the robot. This integration begins with the installation of a **metal bracket** in Figure 5.1.2 onto the robot chassis, which serves as a stable anchor point for the attachment. The bracket is carefully positioned to ensure it does not obstruct any sensors or components, while maintaining structural balance. A **stick holder** is then clamped firmly onto the metal bracket, creating a solid interface capable of withstanding dynamic movements during navigation. The guiding stick, typically a telescopic or lightweight rod, is subsequently inserted into the holder and adjusted for ergonomic alignment can be seen in Figure 5.1.3. The entire assembly is tested for **rigidity and stability**, ensuring that it provides a secure, smooth, and comfortable guiding experience for the user. This mechanical setup enhances the user's sense of connection with the robot, promoting confidence and ease while being guided through indoor environments.

5.2 Software Setup

Raspberry Pi software set up

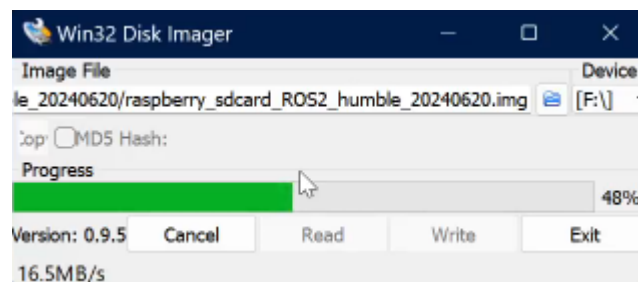
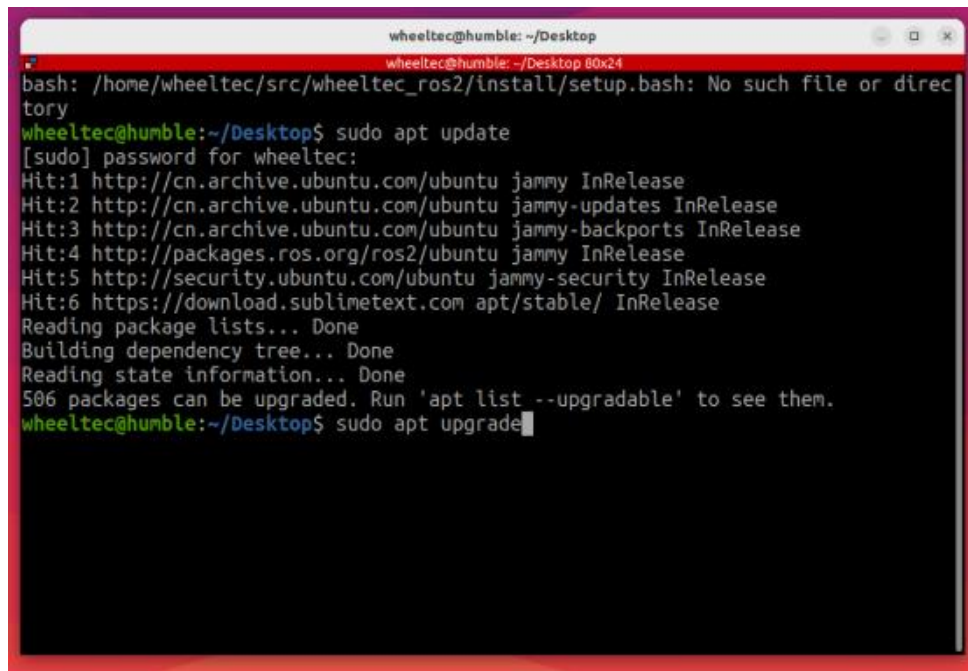


Figure 5.2.1 Burning ROS2 Ubuntu image into Raspberry pi

The Raspberry Pi, serving as the robot's onboard computer, required a specific operating system with ROS2 capabilities. I have used the Wheeltec ROS2 Humble image, which is preconfigured for compatibility with the Robot.

I have obtained the Wheeltec ROS2 Humble image, specifically designed for the robot model. Using Win32 Disk Imager, wrote the image onto a MicroSD card. This process is illustrated in Figure 5.2.1. After the writing process completed, I inserted the MicroSD card into the Raspberry Pi's SD card slot on the robot.

PC Set Up

A terminal window titled 'wheeltec@humble: ~/Desktop' with a red border. The window shows a bash prompt where a file path is entered, resulting in an error. Then, the user runs 'sudo apt update', providing a password. The terminal displays the output of the update command, including hit counts for various repositories and a list of upgradable packages. Finally, the user runs 'sudo apt upgrade' at the end of the visible output.

```
wheeltec@humble: ~/Desktop
bash: /home/wheeltec/src/wheeltec_ros2/install/setup.bash: No such file or directory
wheeltec@humble:~/Desktop$ sudo apt update
[sudo] password for wheeltec:
Hit:1 http://cn.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://cn.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://cn.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://packages.ros.org/ros2/ubuntu jammy InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:6 https://download.sublimetext.com apt/stable/ InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
506 packages can be upgraded. Run 'apt list --upgradable' to see them.
wheeltec@humble:~/Desktop$ sudo apt upgrade
```

Figure 5.2.2 Pc Software Update

In the PC, I performed system updates to ensure all packages were current. This step is crucial for compatibility and security. The update process is demonstrated in Figure 5.2.2. I verified the ROS2 Humble installation and configured the necessary environment variables. Additional development tools and libraries specific to the project requirements were installed .

Android Studio Setup

11. LIMITATION OF LIABILITY

11.1 YOU EXPRESSLY UNDERSTAND AND AGREE THAT GOOGLE, ITS SUBSIDIARIES AND AFFILIATES, AND ITS LICENSORS SHALL NOT BE LIABLE TO YOU UNDER ANY THEORY OF LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES THAT MAY BE INCURRED BY YOU, INCLUDING ANY LOSS OF DATA, WHETHER OR NOT GOOGLE OR ITS REPRESENTATIVES HAVE BEEN ADVISED OF OR SHOULD HAVE BEEN AWARE OF THE POSSIBILITY OF ANY SUCH LOSSES ARISING.

12. Indemnification

12.1 To the maximum extent permitted by law, you agree to defend, indemnify and hold harmless Google, its affiliates and their respective directors, officers, employees and agents from and against any and all claims, actions, suits or proceedings, as well as any and all losses, liabilities, damages, costs and expenses (including reasonable attorneys fees) arising out of or accruing from (a) your use of the SDK, (b) any application you develop on the SDK that infringes any copyright, trademark, trade secret, trade dress, patent or other intellectual property right of any person or defames any person or violates their rights of publicity or privacy, and (c) any non-compliance by you with the License Agreement.

13. Changes to the License Agreement

13.1 Google may make changes to the License Agreement as it distributes new versions of the SDK. When these changes are made, Google will make a new version of the License Agreement available on the website where the SDK is made available.

14. General Legal Terms

14.1 The License Agreement constitutes the whole legal agreement between you and Google and governs your use of the SDK (excluding any services which Google may provide to you under a separate written agreement), and completely replaces any prior agreements between you and Google in relation to the SDK. 14.2 You agree that if Google does not exercise or enforce any legal right or remedy which is contained in the License Agreement (or which Google has the benefit of under any applicable law), this will not be taken to be a formal waiver of Google's rights and that those rights or remedies will still be available to Google. 14.3 If any court of law, having the jurisdiction to decide on this matter, rules that any provision of the License Agreement is invalid, then that provision will be removed from the License Agreement without affecting the rest of the License Agreement. The remaining provisions of the License Agreement will continue to be valid and enforceable. 14.4 You acknowledge and agree that each member of the group of companies of which Google is the parent shall be third party beneficiaries to the License Agreement and that such other companies shall be entitled to directly enforce, and rely upon, any provision of the License Agreement that confers a benefit on (or rights in favor of) them. Other than this, no other person or company shall be third party beneficiaries to the License Agreement. 14.5 EXPORT RESTRICTIONS. THE SDK IS SUBJECT TO UNITED STATES EXPORT LAWS AND REGULATIONS. YOU MUST COMPLY WITH ALL DOMESTIC AND INTERNATIONAL EXPORT LAWS AND REGULATIONS THAT APPLY TO THE SDK. THESE LAWS INCLUDE RESTRICTIONS ON DESTINATIONS, END USERS AND END USE. 14.6 The rights granted in the License Agreement may not be assigned or transferred by either you or Google without the prior written approval of the other party. 14.7 The License Agreement, and your relationship with Google under the License Agreement, shall be governed by the laws of the State of California without regard to its conflict of laws provisions. You and Google agree to submit to the exclusive jurisdiction of the courts located within the county of Santa Clara, California to resolve any legal matter arising from the License Agreement. Notwithstanding this, you agree that Google shall still be allowed to apply for injunctive remedies (or an equivalent type of urgent legal relief) in any jurisdiction. July 27, 2021

☒ I have read and agree with the above terms and conditions

Download Android Studio Meerkat Feature Drop | 2024.3.2 for Linux

android-studio-2024.3.2.14-linux.tar.gz

Figure 5.2.3 Download Android Studio Linux

As part of the system development process, installing the appropriate software development tools is essential for configuring, compiling, and deploying the Android mobile application used in the guide robot system. One of the key components in the development workflow is **Android Studio**, the official Integrated Development Environment (IDE) for Android application development provided by Google. The download for the Linux-compatible version of Android Studio (android-studio-2024.3.2.14-linux.tar.gz) is made available. This version was selected to ensure compatibility with the Linux-based development environment used on the Raspberry Pi and other supporting systems within the project. This step is a critical part of the software environment setup, enabling the development and deployment of the **Visdom Mobile App**, which facilitates communication between the user and the guide robot through socket-based message transmission.

5.3 Setting and Configuration

Setting Android App Project

1. Add new project and then empty activity and click next

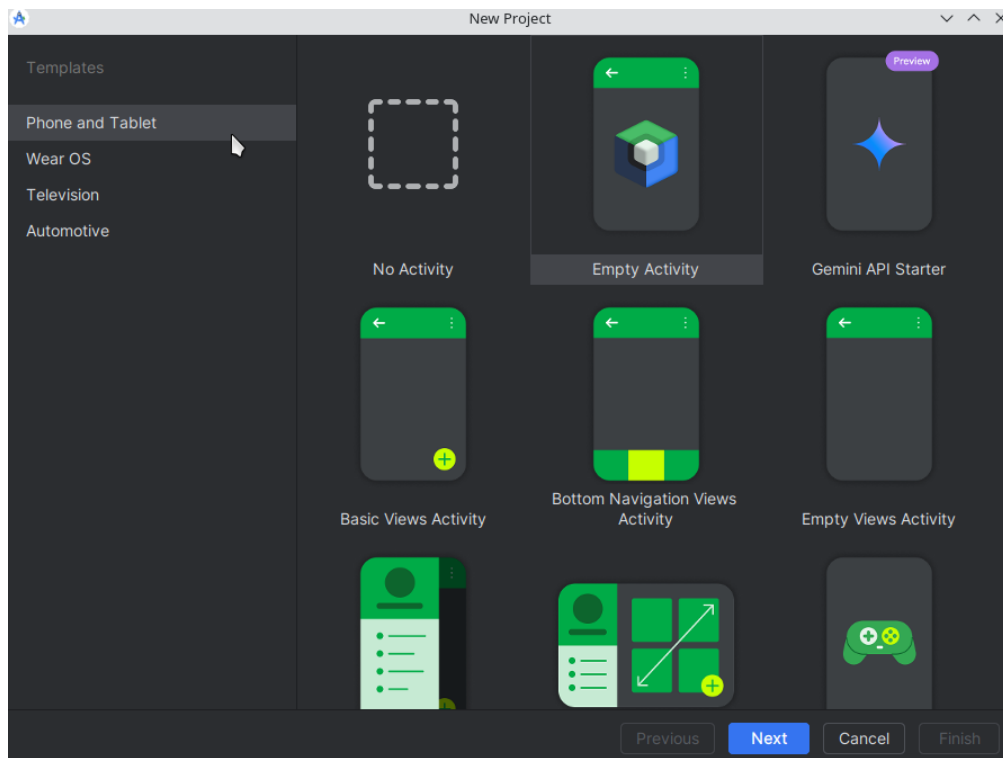


Figure 5.3.1 New Andoird Project Creation

2. In the Project Settings, Set these configuration (Build configuration language will be change to Java after project creation)

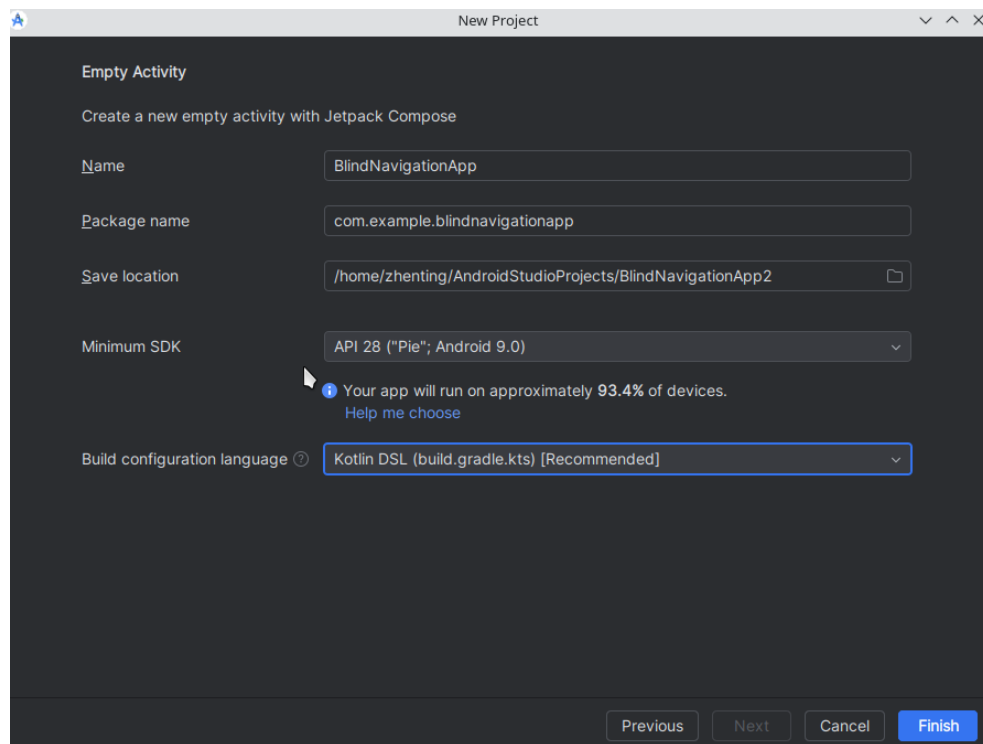


Figure 5.3.2 Setting Project Configuration

3. Click **Finish** and wait for Gradle to build.

Setting Up Communication (TCP socket communication protocol) Configuration between Robot and App

1. Import Socket Library inside the blind_navigation_server.py (python script file inside the robot)

```
1  #!/usr/bin/env python3
2  import rclpy
3  from rclpy.node import Node
4  from rclpy.executors import MultiThreadedExecutor
5  from nav2_simple_commander.robot_navigator import BasicNavigator, TaskResult
6  from geometry_msgs.msg import PoseStamped, Twist, Quaternion
7  from sensor_msgs.msg import LaserScan
8  import socket
9  import threading
10 import math
11 import time
12 import subprocess
13 import shlex
14 import queue
15
```

Figure 5.3.3 Socket Library Import

3. open a TCP server on **port 8888** and listens for incoming connections from the Mobile App

```
def start_server(self):
    HOST = ''
    PORT = 8888
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT))
        s.listen()
        self.get_logger().info(f"Listening for incoming connections on port {PORT}...")
        while True:
            conn, addr = s.accept()
            with conn:
                data = conn.recv(1024).decode('utf-8').strip()
                if data:
                    self.get_logger().info(f"Received navigation command: {data}")
                    self.handle_navigation_command(data)
```

Figure 5.3.4 Start_server Function

4. Setting up Robot Hotspot IP Add (192.168.0.100) and port: 8888 in the mobile app

```
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Locale;

> </> public class NavigationActivity extends AppCompatActivity {

    6 usages
    private TextToSpeech tts;
    3 usages
    private Button buttonReturn;
    1 usage
    private final String ROBOT_IP = "192.168.0.100"; // Replace with your robot's IP
    1 usage
    private final int ROBOT_PORT = 8888; // Port your robot server listens to
```

Figure 5.3.5 Setting up Robot Ip Address

5. By using java.net.socket library Opens a new TCP socket to the robot.

```
public class NavigationActivity extends AppCompatActivity {
    private boolean speakButtonLabel(View v, MotionEvent event) {
        return false;
    }

    1 usage
    private void sendCommand(String command) {
        new Thread(() -> {
            try {
                Socket socket = new Socket(ROBOT_IP, ROBOT_PORT);
                OutputStream output = socket.getOutputStream();
                output.write(command.getBytes());
                output.flush();
                output.close();
                socket.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }).start();
    }
}
```

Figure 5.3.6 Open a new TCP socket

5.4 System Operation

Initialize Robot with PC (Admin Part)



Figure 5.4.1 Connect To The Robot Wifi

The process began with establishing a connection to the robot's Wi-Fi network, as shown in Figure 5.4.1. This step is crucial for enabling remote communication and control.

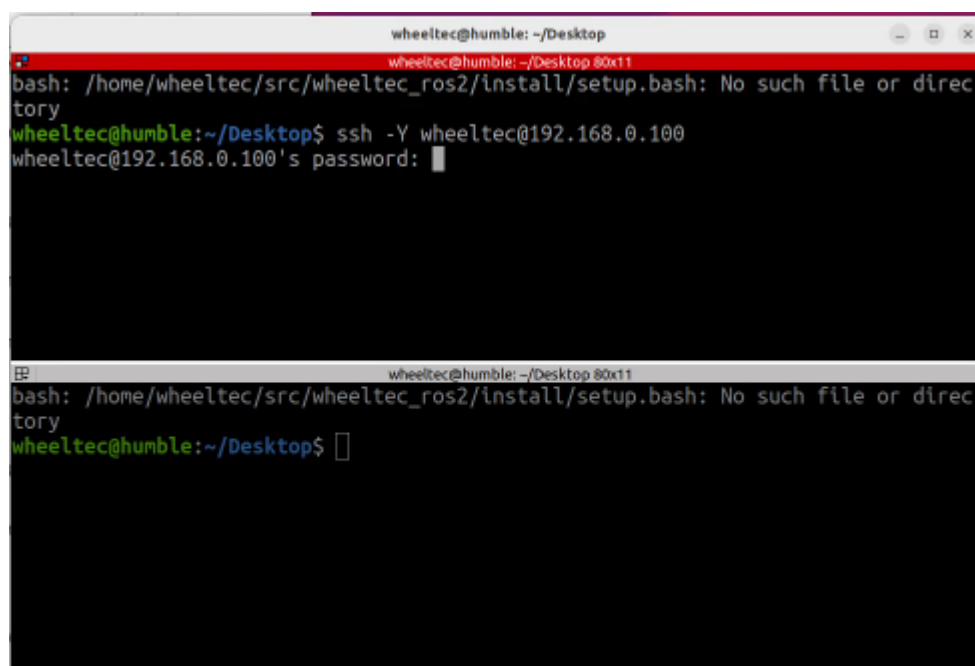
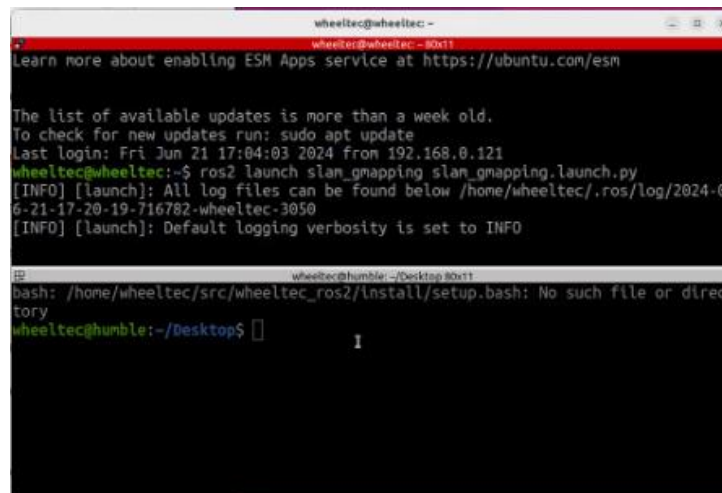


Figure 5.4.2 SSH Connection Between PC And Robot

Once connected, I established an SSH connection between the PC and the robot shown in Figure 5.4.2. This secure connection allows for direct command execution on the robot's system. To enhance visibility and workflow, I utilized a split terminal view.



```
wheeltec@wheeltec: ~  
Learn more about enabling ESM Apps service at https://ubuntu.com/esm  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
Last login: Fri Jun 21 17:04:03 2024 from 192.168.0.121  
wheeltec@wheeltec:~$ ros2 launch slam_gmapping slam_gmapping.launch.py  
[INFO] [launch]: All log files can be found below /home/wheeltec/.ros/log/2024-06-21-17-20-19-716782-wheeltec-3050  
[INFO] [launch]: Default logging verbosity is set to INFO  
  
wheeltec@humble: ~/Desktop 80x11  
bash: /home/wheeltec/src/wheeltec_ros2/install/setup.bash: No such file or directory  
wheeltec@humble:~/Desktop$
```

Figure 5.4.3 Launch The Slam_Gmapping Node

With the connection established, I initiated the SLAM (Simultaneous Localization and Mapping) functionality by executing the command "ros2 launch slam_gmapping slam_gmapping.launch.py" on the robot shown in Figure 5.4.3. This command initializes the mapping functions essential for the robot's navigation capabilities.

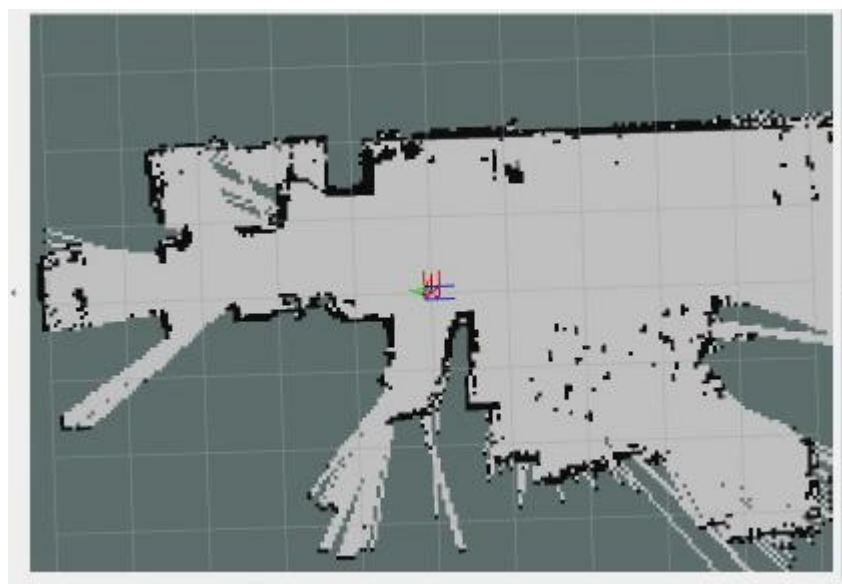


Figure 5.4.4 Rviz Mapping Process

To visualize the mapping process, I launched Rviz from the PC shown in Figure 5.4.4. This tool provides a real-time visual representation of the robot's perception of its environment. In

this case, I conducted a mapping exercise of a living room to test the system's capabilities in a realistic setting.

```
wheeltec@wheeltec:~$ ros2 launch wheeltec_nav2 wheeltec_n
launch.py
[INFO] [launch]: All log files can be found below /home/w
tec/.ros/log/2024-06-21-18-00-25-084622-wheeltec-4893
[INFO] [launch]: Default logging verbosity is set to INFO
```

Figure 5.4.5 Launch Navigation Stack On Robot

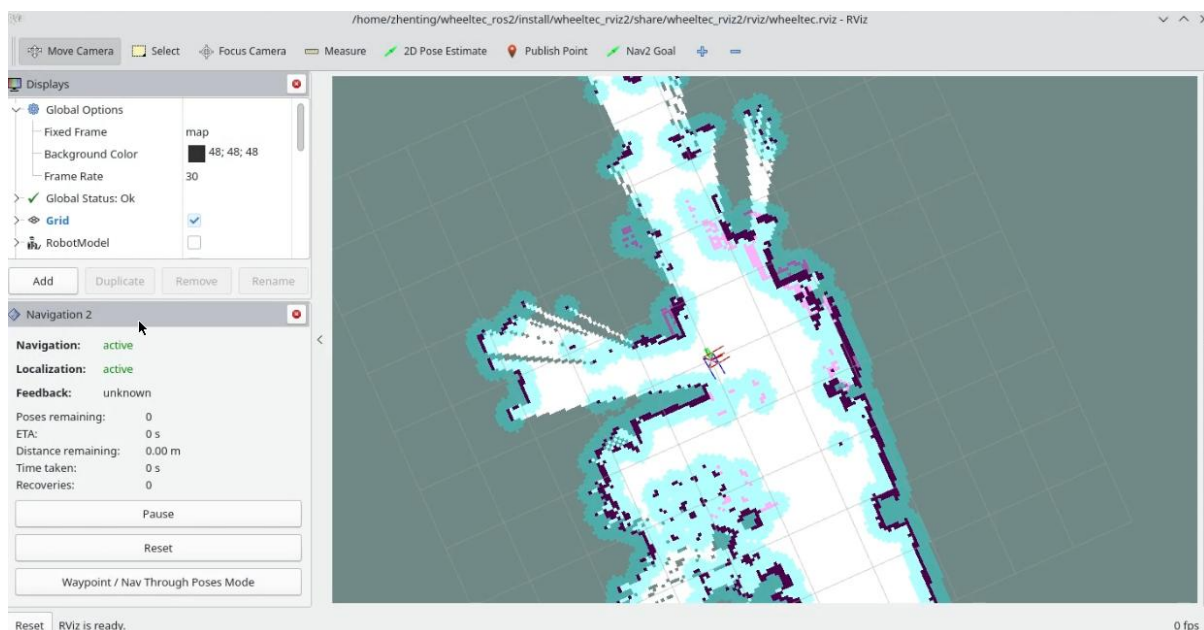


Figure 5.4.6 Rviz Map After Navigation Stack Is Turn On From Robot

Following the mapping process, I have launched the navigation stack on the robot shown in Figure 5.4.5. This step activates the robot's ability to plan and execute movements within the mapped environment. The resulting Rviz map in Figure 5.4.6 visualization demonstrates how the robot perceives the mapped area, including point cloud data.

An important feature of the Rviz interface is the ability to publish points on the map. By hovering the cursor over a location, the x and y coordinates are displayed in the bottom left corner. These coordinates are crucial for defining destination points in the navigation system.

```
# Define locations (x, y, yaw_in_radians)
LOCATIONS = {
    "Starting Point": (-0.0254, -0.0399, 0.128),
    "kitchen": (0.203, 2.98, 0.09),
    "toilet": (0.8, -3.4, 0.00),
    "elevator": (5.0, 5.0, 3.14)
}
```

Figure 5.4.7 Location Coordinates

```
def navigate_to(self, location_name):
    x, y, yaw = LOCATIONS[location_name]
    goal_pose = create_pose_stamped(self.navigator, x, y, yaw)

    self.enqueue_speech(f"Heading to {location_name}")
    self.get_logger().info(f"Navigating to goal: {x} {y}...")

    self.navigator.goToPose(goal_pose)

    while not self.navigator.isTaskComplete():
        feedback = self.navigator.getFeedback()
        time.sleep(0.1)

    result = self.navigator.getResult()
    if result == TaskResult.SUCCEEDED:
        self.enqueue_speech("You have reached your destination")
        self.get_logger().info(f"Goal {location_name} succeeded!")
        return True
    else:
        self.get_logger().error(f"Goal {location_name} failed or was canceled.")
        return False

def navigation_logic_loop(self):
    while rclpy.ok():
        location_name = self.goal_queue.get()
        success = self.navigate_to(location_name)
        if success:
            time.sleep(1.0)
            self.enqueue_speech("Waiting for the next location")

def main():
    rclpy.init()
    server = BlindNavigationServer()
    executor = MultiThreadedExecutor()
    executor.add_node(server)
    try:
        executor.spin()
    except KeyboardInterrupt:
        pass
    finally:
        server.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Figure 5.4.8 blind_navigation_server.py

```
wheeltec@wheeltec:~/blind_navigation_server$ python3 bli
nd_navigation_server.py
```

Figure 5.4.9 Execute The Python Script File

To utilize these coordinates, I have incorporated them into the blind_navigation_server.py script shown in Figure 5.4.7. Destination positions were set from pc by changing the parameter for the robot. Finally, I executed this Python script shown in Figure 5.4.9 to run the robot socket server and wait for the command sent by the mobile app.

Mobile App Operation (User Operate)

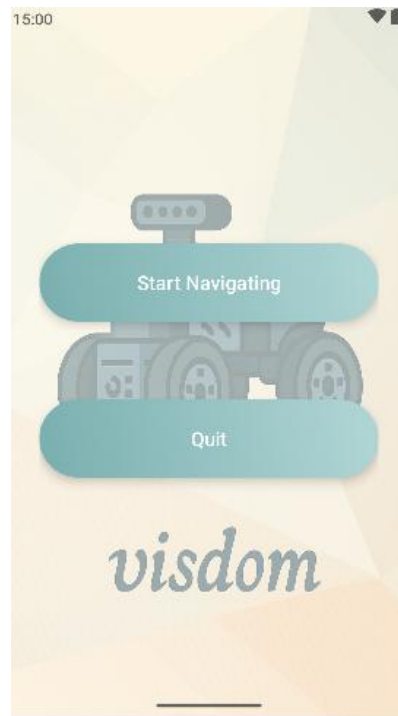


Figure 5.4.10 BlindNavigationApp Main Menu

The BlindNavigationApp serves as the primary user interface for controlling the guide robot. As shown in Figure 5.4.10, the main menu consists of two accessible buttons—**Start Navigating** and **Quit**—laid out in a minimalist and visually calm interface to suit users with visual impairments. Before interacting with the app, users are advised to enable the **TalkBack accessibility feature** on their Android devices. This function allows the device to provide voice feedback when buttons are tapped or held, greatly enhancing usability for blind or visually impaired individuals.

Once TalkBack is enabled, the user can explore the interface by touch. Pressing the **Start Navigating** button transitions the app to the next screen where destination selection is available, allowing the user to initiate a navigation session. Pressing the **Quit** button immediately terminates the application. Additionally, by **pressing and holding** either button, TalkBack reads out the button's label, informing the user of their current selection before they commit to an action. This screen marks the initial interaction point in the overall robot-guidance system, bridging the user's intent with robot execution through accessible mobile interaction.

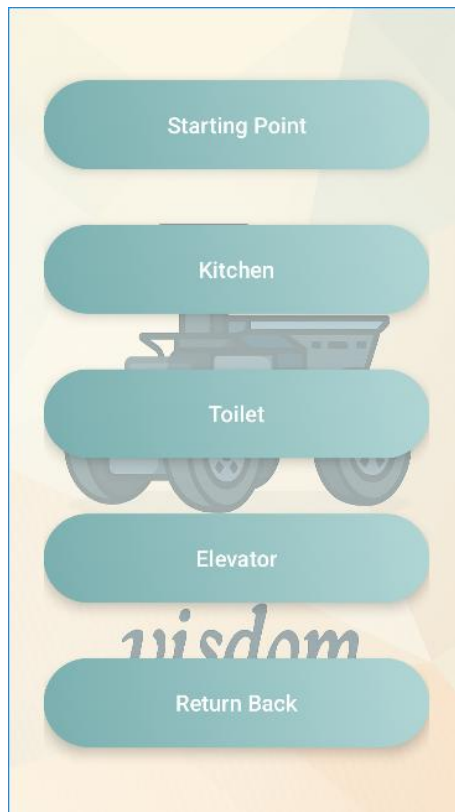


Figure 5.4.11 Destination Menu

Once the user selects "Start Navigating" from the main menu, they are directed to the **Destination Menu**, as shown in Figure 5.4.11. This interface presents a set of predefined locations—including **Starting Point**, **Kitchen**, **Toilet**, and **Elevator**—that the user can choose from. These locations are mapped to specific coordinates within the robot's internal map. The **Return Back** button allows the user to navigate back to the main menu screen if they wish to cancel or change their action.

When the user selects a destination, such as **Toilet**, the mobile app sends a message containing the selected location's name (e.g., "toilet") over a TCP socket to the robot. This message is transmitted using the communication protocol previously established between the app and the Python-based socket server running on the robot. Upon receiving the message, the robot parses the string and matches it to a predefined list of coordinate values representing the destination. Once matched, the robot activates the navigation stack and begins autonomous movement toward the selected location.

Guiding Operation



Figure 5.4.12 Guiding Operation

The image above demonstrates the real-time guiding process. The user, wearing a mask to simulate a visually impaired person, uses TalkBack accessibility and the mobile app's text-to-speech feature to interact with the interface. Once the `blind_navigation_server.py` script initializes, the robot prompts **"Ready for navigation"**, signaling it is prepared to receive commands. When the user selects **"Toilet"**, the robot responds with **"Heading to toilet"** and begins autonomous movement. During navigation, the robot provides contextual feedback such as **"Be careful to your right/left"** when detecting nearby obstacles, and **"Reversing"** if it needs to back up to reach another destination. This combination of spoken cues and autonomous mobility ensures safe and intuitive guidance for the user.

5.5 Implementation Issues and Challenges

Throughout the development and testing of the guide robot system, several implementation challenges were encountered that affected both the stability and user experience of the overall setup.

One of the recurring issues stemmed from the **ROS 2 Navigation Stack (Nav2)**. During operation, the system occasionally experienced **transform timeout errors**, such as invalid "map" frame references or the **global costmap failing to receive necessary data**. These errors disrupted the navigation pipeline and, in some cases, caused the robot to halt unexpectedly or fail to plan a path, requiring manual intervention and restarts. Another significant challenge was related to the **robot-hosted Wi-Fi hotspot**, which formed the backbone of communication between the mobile device and the robot. The hotspot's **slow data transmission speed** introduced delays during system initialization, particularly when launching Nav2 from a remote PC. This limited bandwidth not only slowed down command execution but also posed a **risk of lost or delayed messages**, making it an unreliable foundation for IoT-style communication.

The **voice prompting system**, which plays a key role in guiding visually impaired users, also faced timing inconsistencies. In some instances, the robot would issue voice messages either **too early or too late** relative to the actual robot movement. This lack of synchronization could lead to **user confusion**, as the audio feedback did not always accurately reflect real-time robot behavior.

Additionally, in scenarios where a software bug occurred or the system needed to be restarted, the **initialization time was considerably long**. This includes the time required to establish SSH connections, launch ROS 2 nodes, and stabilize the navigation stack. Such delays proved to be **impractical in real-world deployment**, especially in cases where quick recovery is crucial. These challenges highlight key areas for future improvement in terms of system robustness, real-time responsiveness, and network reliability, especially for applications intended for assistive technology users who rely on consistency and trust in their guidance system.

5.6 Concluding Remark

The guide robot system was successfully built and tested, but the process showed how important it is to plan carefully when working with many different parts—like hardware, software, and how they all talk to each other. This system includes things like a Raspberry Pi, LiDAR sensor, motors, wireless connections, a mobile app, and ROS 2 software. Each part has its own role, and even a small mistake—like a loose wire or a wrong setting—can cause the whole system to stop working properly.

As shown in this chapter, every part matters. Even simple items like the robot's tires or the guiding stick can affect how the robot moves. The software, like the navigation and voice system, also depends a lot on the network and the way everything is connected. If one part doesn't respond in time, the whole system can feel slow or confusing. This means building a working robot is not just about putting parts together, it's about making sure everything works together smoothly and in real time.

One of the most important lessons from this project is that **real-time performance is critical**, especially for a robot that helps **visually impaired people**. The robot needs to act and speak quickly and at the right moment. If there is a delay in voice prompts or the robot hesitates, the user may get confused or feel unsafe. So, the system must always be fast, careful, and clear in its actions. For example, the robot should say “heading to toilet” just as it starts moving, not too early or too late.

In summary, this project showed that building a guide robot needs full understanding of **how every part works together**. It's not just about good code or strong hardware—it's about getting both to work together in real time. The experience gained in this project gives a strong base for future improvements, like making the system faster, more stable, and safer. Since this robot is made for people who depend on it for moving safely, it must always be **trustworthy, clear, and responsive** in everything it does.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

6.1 System Testing and Performance Metrics

Table 6.1.1 Testing Metrics

Test Condition	Description	Test Level	
		Lenient	Harsh
Sharp Turns	Number or degree of sharp turns in the path	Straight or wide-angle paths	Multiple tight-angle turns and L-shaped corners
Obstacle Count	Number of obstacles placed along the route	0–2 Fixed objects	5 or more fixed indoor obstacles
Path Narrowness	Width of available path relative to robot size	Wide corridor (2x robot width)	Just enough width to pass or people passing by
Path Distance	Total travel distance from start to goal	2–3 meters	8–10 meters across multiple rooms
Consecutive Destinations	Multiple goals in one run	One destination only	Multiple destinations set continuously (e.g., kitchen → elevator)

Table 6.1.2 Performance Metrics

Metrics	Description	Performance		
		Bad	Acceptable	Good
Navigation Accuracy	How closely the robot follows the planned path	Deviates > 100 cm	Within 50–100cm	Stays within < 50 cm
Obstacle Avoidance	Success rate of avoiding detected obstacles	< 60% success	60–85% success	> 85% success
Voice Prompt Timing	Whether voice prompts match robot actions in real time	> 2s delay or mismatch	1–2s delay, mostly aligned	Prompts within 1s and consistent
App Communication Reliability	Consistency and success of sending/receiving app commands	Frequent message loss	Occasional delays or retries	Smooth, fast, with no lost messages
Navigation Recovery Speed	Time taken to replan and become operational after error while navigating	> 15s to recover	>10s to recover	Recovers in < 10s
Guided Human Bump into obstacles	Whether the robot leads the user into walls or obstacles	Frequent or unsafe collisions	Rare, minor bumps that don't cause risk	No bumps or near-misses detected

6.2 Testing Setup and Result

Table 6.2.1 Test Case 1

Testing Conditions		
Lenient		Harsh
<ul style="list-style-type: none">▪ Sharp Turn▪ Obstacles Count▪ Path Narrowness▪ Path Distance		<ul style="list-style-type: none">▪ Consecutive Destinations
Performance		
Bad	Acceptable	Good
	<ul style="list-style-type: none">▪ Obstacle Avoidance▪ Voice Prompt Timing	<ul style="list-style-type: none">▪ App Communication Reliability▪ Navigation Recovery Speed▪ Navigation Accuracy ▪ Guided Human Bump into obstacles

Table 6.2.2 Test Case 2

Testing Conditions		
Lenient		Harsh
<ul style="list-style-type: none">▪ Sharp Turn▪ Obstacles Count▪ Path Narrowness		<ul style="list-style-type: none">▪ Consecutive Destinations▪ Path Distance
Performance		
Bad	Acceptable	Good
	<ul style="list-style-type: none">▪ Obstacle Avoidance▪ Voice Prompt Timing▪ Navigation Accuracy	<ul style="list-style-type: none">▪ App Communication Reliability▪ Navigation Recovery Speed▪ Guided Human Bump into obstacles

Table 6.2.3 Test Case 3

Testing Conditions		
Lenient		Harsh
<ul style="list-style-type: none">▪ Path Narrowness		<ul style="list-style-type: none">▪ Consecutive Destinations▪ Obstacles Count▪ Sharp Turn▪ Path Distance
Performance		
Bad	Acceptable	Good
<ul style="list-style-type: none">▪ Obstacle Avoidance▪ Guided Human Bump into obstacles	<ul style="list-style-type: none">▪ Navigation Accuracy▪ Navigation Recovery Speed▪ Voice Prompt Timing	<ul style="list-style-type: none">▪ App Communication Reliability

Table 6.2.4 Test Case 4

Testing Conditions		
Lenient		Harsh
		<ul style="list-style-type: none">▪ Consecutive Destinations▪ Sharp Turn▪ Obstacles Count▪ Path Narrowness▪ Path Distance
Performance		
Bad	Acceptable	Good
<ul style="list-style-type: none">▪ Obstacle Avoidance▪ Voice Prompt Timing▪ Navigation Accuracy▪ Navigation Recovery Speed▪ Guided Human Bump into obstacles		<ul style="list-style-type: none">▪ App Communication Reliability

6.3 Project Challenges

During the development of this guide robot system, several key challenges were encountered that limited the overall performance, robustness, and real-world applicability of the prototype.

The **primary challenge** lies in the hardware limitations of the **Raspberry Pi 4B**, which serves as the core microcontroller. While the Raspberry Pi is compact and cost-effective, it lacks the **computational power and memory bandwidth** needed to handle heavy tasks such as real-time navigation, fast behavior recovery, Ai obstacle detection, and smooth simultaneous sensor communication. This limitation became especially apparent in more demanding test conditions, where even moderate loads would cause delays, missed messages, or system lag. For future iterations, a higher-performance embedded computer, such as the NVIDIA Jetson series, may be more appropriate.

Another major challenge involved **communication between sensors and the microcontroller**. There were frequent issues where sensor messages failed to synchronize or were not received properly by the running ROS 2 nodes. Timeouts, dropped topics, or latency in sensor feedback often disrupted navigation or caused delayed responses in voice prompts and obstacle avoidance. These issues are likely due to **bandwidth overloads or internal I/O bottlenecks**, and they significantly reduce system reliability. Consistent, stable communication across all nodes is critical in a robot where timing and synchronization can directly impact user safety.

Furthermore, all testing was conducted in **controlled indoor environments** with static obstacles. There were no pedestrians, moving obstacles, or unpredictable events, which made the current setup **insufficient for actual deployment**. While the robot shows that it can follow commands and navigate indoor maps, it still falls short of what is needed in dynamic, everyday environments such as shopping malls, train stations, or streets. The system must eventually be capable of reacting to live changes in the environment.

Another area for improvement is **environmental awareness and feedback**. As of now, the robot does not provide contextual awareness to the user beyond its basic voice prompts. In real-world use, it should be able to notify the user of hazards like an upcoming low-hanging branch, puddle, uneven ground, or crowded areas. For a visually impaired person to fully trust and

follow a guide robot, it must serve not only as a path planner but also as a **real-time environment reporter**.

Lastly, a challenge related to the **mechanical platform** was identified. The robot is based on a **4-wheel Ackermann steering vehicle**, which inherently limits its maneuverability—especially when reversing. Unlike differential drive systems, Ackermann platforms require more space to turn and have less flexibility in narrow or complex spaces. This affected the robot's ability to handle sharp turns and respond quickly to dynamic changes in the environment.

In summary, while the current project demonstrates that a guiding robot for visually impaired individuals is a promising and achievable concept, several practical and technical challenges must be addressed. These include upgrading hardware performance, improving sensor communication reliability, enabling dynamic environment handling, and enhancing the user's situational awareness through richer environmental feedback.

6.4 Objectives Evaluation

The main objective of this project was to **develop an autonomous mobile robot system equipped with a mobile application**, designed specifically to assist **visually impaired individuals** by providing reliable indoor walking guidance. The system aimed to not only navigate autonomously using sensor data and intelligent planning but also to interact with users through a simple, accessible Android interface. As of the current development stage, the core system has been successfully built, tested, and demonstrated with most sub-objectives achieved to a working degree.

The **autonomous mobile robot system**, based on the Wheeltec R550 platform and powered by a **Raspberry Pi 4B**, has been successfully assembled and configured. The robot integrates essential components including a **2D LiDAR sensor** for environmental perception, a **motor controller**, and a **speaker** for voice feedback. While the Raspberry Pi proved capable for light operations, it showed some limitations in handling more demanding processes such as simultaneous LiDAR scanning, voice feedback, and path planning—especially under tight or obstacle-rich conditions.

The system was implemented using the **ROS 2 framework**, which enabled the use of robust and modular packages. Key features such as **path planning**, **object avoidance**, and **localization** were realized through the integration of **SMAC Hybrid-A*** for global planning and the **Regulated Pure Pursuit Controller** for local movement control. These algorithms, along with the **Nav2 stack**, allowed the robot to perform autonomous navigation within a mapped indoor environment. The map used for navigation was generated using the **SLAM gmapping algorithm**, and the saved map was loaded by the Map Server to guide the robot across pre-defined paths.

All **hardware components were successfully integrated**, including proper communication with the LiDAR, odometer, and speaker. Careful configuration of ROS 2 parameters and topic synchronization ensured that each component operated under a shared system timeline. The voice feedback system, while functional, still needs refinement in timing accuracy to match robot actions in real time, especially in scenarios involving sudden stops or direction changes.

The **Android mobile application** was also developed and completed using Android Studio, offering a clean and intuitive interface. It supports **TalkBack accessibility features** and uses **TCP socket communication** to send commands to the robot over a Wi-Fi hotspot. The robot, running a custom Python-based `blind_navigation_server.py`, listens on a fixed port and interprets location commands from the app, translating them into navigation goals. This interaction between the robot and the mobile device has been successfully tested and performs reliably.

The project also included **mechanical modifications**, such as attaching a **guiding stick holder** via a custom metal bracket to simulate real-world usage for visually impaired users. This physical guide allows the user to feel movement cues while walking alongside the robot. However, while the system shows that a **guiding robot for visually impaired individuals is possible and promising**, it is **not yet perfect or ready for real deployment**. Testing was done in controlled indoor environments without dynamic obstacles or pedestrians. The system still lacks **environmental awareness features**, such as recognizing overhead hazards or floor-level dangers (e.g., puddles or steps). Additionally, the use of an **Ackermann-steered 4-wheel robot** introduces challenges in reversing and turning in tight spaces—further limiting flexibility in complex layouts.

In summary, the **main objective** of building an autonomous robot and mobile app system has been achieved, and autonomous navigation is functioning well under basic indoor conditions. The **communication between the robot and the mobile app is stable and effective**. Progress has been made in assisting visually impaired users, with a strong foundation laid for further development. However, **real-time responsiveness, environment understanding, and user safety awareness still require improvement** to bring the system closer to real-life application.

6.5 Concluding Remark

This chapter provided a thorough evaluation of the guide robot system through various testing scenarios, performance analysis, and reflection on development challenges. From the results gathered in both controlled and practical tests, it is clear that the system has made strong progress toward becoming a functional assistive tool for visually impaired individuals. The testing process showed that the robot is capable of performing **autonomous navigation**, **obstacle avoidance**, and **command reception** from the Android mobile app. Metrics such as navigation accuracy, voice prompt timing, and communication reliability were measured to assess the system's stability and usability. While several aspects of the system performed well under "easy" test conditions, challenges became more visible under more complex or "real-life" setups, especially in terms of **system responsiveness**, **recovery time**, and **environment awareness**.

Through the analysis in Section 6.3, key challenges were identified, including hardware limitations from using Raspberry Pi, communication delays between ROS 2 nodes and sensors, and the lack of adaptation to dynamic environments. These limitations highlighted the need for stronger hardware and better sensor synchronization to ensure real-time safety and reliability—especially important for a system designed to guide visually impaired users.

Section 6.4 compared the original objectives with actual progress. While major goals such as building an autonomous robot and creating a mobile app interface were achieved, some objectives—such as creating a fully dependable and safe guiding experience—still require further refinement. The robot performs well in basic indoor navigation but would need significant upgrades to handle real-world deployment scenarios.

In conclusion, the system has established a **solid foundation** for a vision-based assistive guidance robot. It proves that such a concept is not only feasible but also implementable using accessible technologies. With future improvements in environment perception, communication robustness, and movement precision, the project has the potential to become a truly helpful solution for the visually impaired community. The evaluation phase has been instrumental in revealing the system's strengths, identifying gaps, and setting a clear path for future development.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

7.1 Conclusion

This project aimed to develop an autonomous mobile robot system paired with a mobile application to assist visually impaired individuals with safe indoor navigation. The goal was to provide real-time guidance using technologies like path planning, obstacle avoidance, and voice feedback.

The system was built using the **ROS 2 framework** on a **Raspberry Pi 4B-based platform**, integrating essential components such as LiDAR, SLAM mapping, SMAC Hybrid-A* global planning, and Regulated Pure Pursuit for local control. These allowed the robot to autonomously navigate mapped environments with reasonable accuracy. In parallel, a custom **Android app** was developed with **TalkBack accessibility** and TCP socket communication, enabling users to select destinations and interact with the robot in real time.

Most core objectives were achieved. The robot could successfully navigate to destinations and respond to commands sent via the mobile app. This demonstrated the feasibility of using low-cost hardware and open-source tools to build an assistive robot. However, some limitations remain, including **hardware performance bottlenecks**, **sensor communication issues**, and a lack of adaptation to **dynamic environments** such as moving obstacles or real-world hazards.

Despite these challenges, the system lays a strong foundation for future development. With improvements in processing power, environment awareness, and user feedback, this solution could become a practical tool for visually impaired individuals. The project proves that assistive robotics is not only achievable but holds real potential to enhance independence and mobility for those who need it most.

7.2 Recommendation

Building upon the current prototype, several enhancements are recommended to improve the robot's functionality, safety, and user experience for visually impaired individuals.

1. Integration of Depth Cameras for Enhanced Distance Estimation

Incorporating depth-sensing cameras, such as the Intel RealSense series, can significantly improve the robot's ability to perceive its environment. These cameras provide real-time 3D data, enabling the robot to estimate distances to obstacles more accurately and offer timely warnings to users.

2. Transition to Omnidirectional or Spherical Robot Platforms

The current 4-wheel Ackermann steering system has limitations in manoeuvrability, especially in tight indoor spaces. Transitioning to an omnidirectional platform, like a meconium-wheeled robot, allows movement in any direction without changing orientation, enhancing navigation in confined areas.

Alternatively, adopting a spherical robot design offers advantages such as uniform movement in all directions and better stability. Spherical robots can navigate complex terrains and have a compact form factor, making them suitable for indoor environments.

3. Enhanced Environmental Awareness

To provide users with comprehensive situational awareness, the robot should be equipped with additional sensors and algorithms capable of detecting various environmental features, such as overhanging obstacles, changes in floor texture, or potential hazards like water puddles. This information can be relayed to users through auditory cues, enhancing safety and trust in the system.

4. Upgrading Processing Capabilities

The Raspberry Pi 4B, while cost-effective, may not suffice for processing intensive tasks like real-time depth mapping and advanced navigation algorithms. Upgrading to more powerful processors, such as the NVIDIA Jetson series, can provide the necessary computational resources for enhanced performance.

REFERENCES

- [1] G. Priyandoko, C. K. Wei, and M. S. H. Achmad, “HUMAN FOLLOWING ON ROS FRAMEWORK A MOBILE ROBOT,” *SINERGI*, vol. 22, no. 2, p. 77, Jun. 2018, doi: 10.22441/sinergi.2018.2.002.
- [2] A. Gaikwad, S. Ohol, M. Kulkarni, A. Agnihotri, and S. Purohit, “ROS based Compact Mobile Robot for Area Mapping, Autonomous Navigation and Path Planning,” Dec. 2021.
- [3] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, “Robot Operating System 2: Design, Architecture, and Uses In The Wild,” *Sci. Robot.*, vol. 7, no. 66, p. eabm6074, May 2022, doi: 10.1126/scirobotics.abm6074.
- [4] H. Zhang, K. Watanabe, K. Motegi, and Y. Shiraishi, “ROS Based Framework for Autonomous Driving of AGVs,” 2019.
- [5] M. Fennel, S. Geyer, and U. D. Hanebeck, “RTCF: A framework for seamless and modular real-time control with ROS,” *Softw. Impacts*, vol. 9, p. 100109, Aug. 2021, doi: 10.1016/j.simpa.2021.100109.
- [6] M. Tkáčik, A. Březina, and S. Jadlovska, “Design of a Prototype for a Modular Mobile Robotic Platform,” *IFAC-Pap.*, vol. 52, no. 27, pp. 192–197, 2019, doi: 10.1016/j.ifacol.2019.12.755.
- [7] P. Guo, H. Shi, S. Wang, L. Tang, and Z. Wang, “An ROS Architecture for Autonomous Mobile Robots with UCAR Platforms in Smart Restaurants,” *Machines*, vol. 10, no. 10, p. 844, Sep. 2022, doi: 10.3390/machines10100844.
- [8] G. Fracapane, H.-H. Hvolby, F. Sgarbossa, and J. O. Strandhagen, “Autonomous Mobile Robots in Hospital Logistics,” in *Advances in Production Management Systems. The Path to Digital Transformation and Innovation of Production Management Systems*, vol. 591, B. Lalic, V. Majstorovic, U. Marjanovic, G. Von Cieminski, and D. Romero, Eds., in IFIP Advances in Information and Communication Technology, vol. 591. , Cham: Springer International Publishing, 2020, pp. 672–679. doi: 10.1007/978-3-030-57993-7_76.

- [9] A. A. Neloy *et al.*, “Alpha-N-V2: Shortest Path Finder Automated Delivery Robot with Obstacle Detection and Avoiding System,” *Vietnam J. Comput. Sci.*, vol. 07, no. 04, pp. 373–389, Nov. 2020, doi: 10.1142/S2196888820500219.
- [10] R. Algabri and M.-T. Choi, “Deep-Learning-Based Indoor Human Following of Mobile Robot Using Color Feature,” *Sensors*, vol. 20, no. 9, p. 2699, May 2020, doi: 10.3390/s20092699.
- [11] S. I. A. P. Diddeniya, W. K. I. L. Wanniarachchi, P. R. S. De Silva, and N. C. Ganegoda, “Efficient Office Assistant Robot System: Autonomous Navigation and Controlling Based on ROS,” *Int. J. Multidiscip. Stud.*, vol. 6, no. 1, p. 64, Jun. 2019, doi: 10.4038/ijms.v6i1.93.
- [12] Y. Chiba, *YasuChiba/ROS2-Mobile-Android*. (Sep. 02, 2024). Java. Accessed: Sep. 03, 2024. [Online]. Available: <https://github.com/YasuChiba/ROS2-Mobile-Android>
- [13] robot mania, *How to Control a Robot Using Smartphone and ROS2*, (Sep. 10, 2023). Accessed: Sep. 06, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=JN_2_bvqtN8
- [14] M. Melo, *mirrellameelo/ROS_2_ANDROID*. (Aug. 30, 2024). C++. Accessed: Sep. 03, 2024. [Online]. Available: https://github.com/mirrellameelo/ROS_2_ANDROID
- [15] L. Wander, “Lysa Robot Guide Dog,” MIT SOLVE. Accessed: Sep. 04, 2024. [Online]. Available: <https://solve.mit.edu/challenges/horizon-prize-2022/solutions/64393>
- [16] J. Lai, H. Lei, S. Bao, L. Du, J. Yuan, and S. Ma, “Design of a Portable Indoor Guide Robot for Blind People,” in *2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, Jul. 2021, pp. 592–597. doi: 10.1109/ICARM52023.2021.9536077.
- [17] Y. Wu, L. Hao, F. Wang, and L. Zu, “The Construction of Occupancy Grid Map with Semantic Information for the Indoor Blind Guiding Robot,” in *2023 IEEE 13th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, Jul. 2023, pp. 225–229. doi: 10.1109/CYBER59472.2023.10256535.
- [18] L. Zhang *et al.*, “Autonomous Indoor Navigation for Visually Impaired: A ROS-Based Intelligent Guide Robot with Deep Learning and SLAM Integration,” in *2024 5th*

- International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, Mar. 2024, pp. 36–39. doi: 10.1109/AINIT61980.2024.10581797.
- [19] V. Ranganeni *et al.*, “Exploring Levels of Control for a Navigation Assistant for Blind Travelers,” in *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, Mar. 2023, pp. 4–12. doi: 10.1145/3568162.3578630.

POSTER



Visdom: Smart Guide Robot for Visually Impaired People

author : Lee Zhen Ting

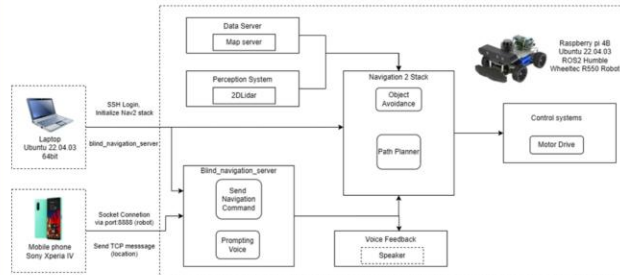
Supervisor: Dr.Teoh Shen Khang



Introduction

With the increasing need for assistive technologies, guide robots offer a promising solution for enhancing the mobility and independence of visually impaired individuals. This project introduces Visdom, an innovative guide robot that combines advanced navigation techniques with user-friendly interfaces.

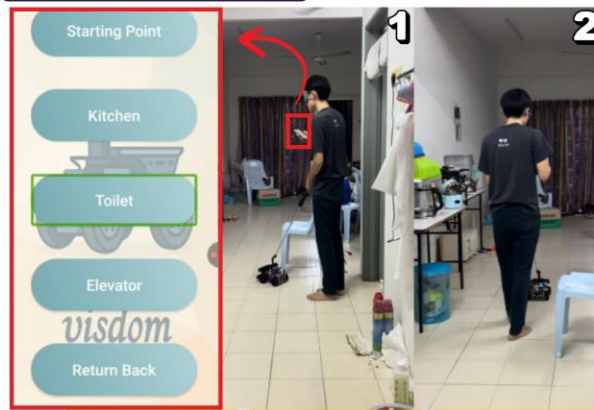
System Design



Project Scope/Objective

- Develop a robust and intelligent guide robot system for visually impaired users
- Implement precise navigation and obstacle avoidance capabilities
- Create an intuitive mobile app interface for seamless user interaction

Results



Proposed Method

- ROS 2-based Wheeltec robot with SLAM for mapping
- SMAC Hybrid-A* and Pure Pursuit for path planning
- Android app with TalkBack for destination selection
- TCP socket for app-to-robot communication
- Voice prompts and guiding stick for user feedback

Conclusion

Visdom shows strong potential as an assistive guide robot using ROS 2 and a custom mobile app. This phase achieved reliable indoor navigation and user interaction. While effective in controlled settings, improvements are needed in obstacle handling and feedback. Future work aims to enhance system responsiveness and real-world usability.