An Autonomous Service Robot for Cafe Order Delivery using ROS 2 and Gazebo

A PROJECT REPORT

Submitted by

ANIRUDH V
(CH.SC.U4AIE23003)

MADHUMITHA K
(CH.SC.U4AIE23027)

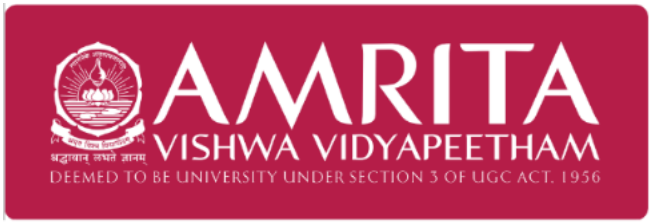
MOPURI RISHITHA
(CH.SC.U4AIE23030)

MUDAVATH AKSHANTH CHOUHAN
(CH.SC.U4AIE23031)

BACHELOR OF TECHNOLOGY IN ARTIFICIAL INTELLIGENCE AND ENGINEERING

Under the guidance of

DR. Dev Kunwar Singh Chauhan



Submitted to

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI – 601103

September -2025

BONAFIDE CERTIFICATE

This is Certify that this project report entitled **"ButlerBot: An Autonomous Service Robot for Cafe Order Delivery using ROS 2 and Gazebo"** is the Bonafide work of ANIRUDH V (CH.SC.U4AIE23003), MADHUMITA.K (CH.SC.U4AIE23027), MOPURI RISHITHA (CH.SC.U4AIE23030), MUDAVATH AKSHANTH CHOUHAN (CH.SC.U4AIE23031) who carried out the assessment work under my supervision.

SIGNATURE

DR. Dev Kunwar Singh Chauhan
Department of MEE.
Assistant Professor.
Amrita School of Computing
Chennai

INTERNAL EXAMINER

ABSTRACT

In restaurant and hospitality environments, human delivery of food orders often faces inefficiencies such as long wait times, human fatigue, and inconsistent service quality. Manual delivery also increases labor costs and reduces scalability during peak hours. To address these challenges, autonomous delivery systems are required to ensure timely and reliable service. Existing robotic delivery systems, such as standard TurtleBot3 navigation and generic AMR (Autonomous Mobile Robot) implementations, primarily rely on static path planning and 2D SLAM navigation. However, these systems struggle in dynamic environments with moving obstacles (like people), have limited perception capabilities, and depend heavily on cloud connectivity for control and monitoring. Many models lack real-time adaptive obstacle avoidance, contextual task assignment, and end-to-end delivery workflow integration. The proposed Delivery Bot system is an intelligent service robot built using ROS 2 (Humble) and TurtleBot3 architecture. It autonomously delivers food from the kitchen to designated tables within a restaurant environment using Simultaneous Localization and Mapping (SLAM), path planning, and obstacle avoidance. The model integrates Cartographer SLAM for map generation, Nav2 Stack for global and local planning, and LDS LiDAR for real-time obstacle detection. Unlike existing models, this system combines dynamic re-planning, local costmap updates, and goal-oriented task execution under a fully modular ROS 2 architecture. The delivery bot efficiently navigates indoor environments, avoids dynamic obstacles, and reaches precise delivery locations, thereby improving operational efficiency and customer experience. The model is lightweight, scalable, and customizable and capable of running both in Gazebo simulation and on real TurtleBot hardware. It can be extended with AI-based perception, voice recognition, and IoT-based order management systems for next-generation restaurant automation.

Keywords: Autonomous Mobile Robot (AMR), ROS 2 (Robot Operating System 2), Navigation and Path Planning, SLAM (Simultaneous Localization and Mapping), Task Management and Delivery System

ACKNOWLEDGEMENT

This work would have been possible without the contribution of many people. It gives us immense pleasure to express my profound gratitude to our honorable Chancellor Sri Mata Amritanandamayi Devi, for her blessings and for being a source of inspiration. I am indebted to extend my gratitude to our Sampoojya Swami Vinayamritananda Puri, Administrative Director, and Shri. I B Manikantan, Campus Director for facilitating us all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to Dr. V. Jayakumar, Principal, for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to Dr. Dev Kunwar Singh Chauhan, Project Coordinator, Assistant professor at MEE for their inspiring guidance, personal involvement, and constant encouragement during the entire course of this work.

I am grateful to Review Panel Members and the entire faculty of the Department of Artificial Intelligence, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

ANIRUDH V
(CH.SC.U4AIE23003)

MADHUMITA K
(CH.SC.U4AIE23027)

MOPURI RISHITHA
(CH.SC.U4AIE23030)

MUDAVATH AKSHANTH CHOUHAN
(CH.SC.U4AIE23031)

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVATIONS

ROS	Robot Operating System
ROS 2	Robot Operating System 2
Nav2	Navigation2 Stack
SLAM	Simultaneous Localization and Mapping
AMCL	Adaptive Monte Carlo Localization
URDF	Unified Robot Description Format
FSM	Finite State Machine
LIDAR	Light Detection and Ranging
IMU	Inertial Measurement Unit
RViz	ROS Visualization Tool

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In many restaurants, delivering food from the kitchen to customers' tables is an important but often time-consuming task. Human staff can get tired, make mistakes, or take longer during busy hours, which can affect service quality and customer satisfaction. Delays or wrong deliveries can lead to unhappy customers and lower efficiency in restaurant operations. To solve these problems, autonomous robots can be used to deliver food, making the process faster, safer, and more reliable.

The Delivery Bot is an intelligent service robot designed to carry food items from the kitchen to the designated tables on its own. It uses ROS 2 (Robot Operating System Humble) and the TurtleBot3 robot platform to perform its tasks. The robot can create a map of the restaurant using SLAM (Simultaneous Localization and Mapping) and locate itself within that map. It plans its route to the target table using the Nav2 navigation stack, which allows it to avoid obstacles and find the shortest path. The robot is equipped with sensors such as LiDAR, IMU, and wheel encoders, which help it detect obstacles, track its position, and navigate safely even in a dynamic environment where people or furniture may move.The system is designed in a modular way, which means different parts of the robot handle different functions, such as mapping, navigation, obstacle avoidance, and delivery management. This modular design makes it easier to upgrade, maintain, or add new features in the future, such as AI-based object detection, voice commands, or multi-robot coordination. The robot can be tested in simulation environments like Gazebo and RViz, and it can also operate on real TurtleBot3 hardware for practical delivery tasks.

Objectives of the Delivery Bot Project:

- To enable the robot to move autonomously around the restaurant and reach the correct tables.
- To detect and avoid obstacles like people, chairs, or tables in real time.
- To deliver food items accurately and efficiently, even when handling multiple orders.
- To design a modular system so that each function works independently but contributes to overall robot performance.
- To test and validate the system in simulation environments and real-world settings.
- To provide a scalable foundation for future improvements, including AI modules, multi-robot coordination, and IoT-enabled restaurant management systems.

By combining autonomous navigation, dynamic obstacle avoidance, and task management, the Delivery Bot improves operational efficiency, reduces human errors, and enhances customer experience. It demonstrates a practical approach to automating delivery tasks and provides a foundation for advanced service robots in restaurants, hotels, hospitals, and other indoor environments.

CHAPTER 2

LITERATURE REVIEW

The deployment of autonomous robots in service industries has seen significant growth over the past decade, driven by advances in robotics frameworks, perception technologies, and navigation algorithms. At the core of many modern robotic systems is ROS 2 (Robot Operating System 2), which provides a modular and scalable platform for building complex robotic applications. One of its most critical components is the Navigation2 (Nav2) stack, which enables autonomous navigation by combining localization, path planning, obstacle avoidance, and recovery behaviors [1]. Nav2 allows robots to plan global paths using algorithms such as Smac or NavFn while adapting locally to dynamic obstacles using controllers like DWB or TEB. Researchers have highlighted that Nav2's modular architecture, along with its support for behavior trees and waypoint navigation, makes it particularly suitable for service robots operating in indoor, human-populated environments such as cafés and restaurants [2]. This flexibility is important because service robots must not only navigate static layouts but also dynamically adapt to changing obstacles, including moving humans, chairs, and tables.

A fundamental capability for autonomous service robots is Simultaneous Localization and Mapping (SLAM). SLAM allows a robot to build a map of an unknown environment while simultaneously tracking its location within that map, a critical requirement in hospitality settings where the layout may be complex or subject to frequent changes. Various SLAM algorithms, including GMapping, Hector SLAM, Cartographer, and Karto, have been integrated into ROS 2, offering options that trade off between accuracy, computational efficiency, and robustness [3]. For instance, Cartographer provides loop closure detection to reduce drift over time, which is particularly useful in large indoor areas like cafés. The integration of SLAM with Nav2 ensures that the robot's path planning is grounded in accurate and updated environmental information, enabling it to perform deliveries without collisions or navigational errors. Studies indicate that the performance of service robots in hospitality settings strongly depends on the accuracy of mapping and localization, as small errors in pose estimation can lead to inefficient paths or collisions with dynamic obstacles [3].

In addition to individual robot navigation, multi-robot coordination has become a prominent area of research in ROS 2 systems. Coordinating multiple robots requires efficient fleet management tools that can schedule tasks, avoid collisions, and optimize routes. Frameworks like the Robot Middleware Framework (RMF) and open-source fleet

management tools developed by NVIDIA provide critical support for task allocation, communication, and monitoring across multiple autonomous mobile robots [4][5]. These tools allow service robots to operate simultaneously in shared spaces, coordinating deliveries to multiple tables without human supervision. Robofleet, for example, provides mechanisms for real-time inter-robot communication and monitoring, ensuring that even heterogeneous robots with different capabilities can work together effectively [5]. The ability to manage fleets is crucial for practical applications, as cafés often require multiple robots to handle high volumes of orders efficiently and safely, particularly during peak hours.

Another key aspect of service robot deployment is waypoint navigation, which involves guiding robots along predefined points in their environment. Waypoint navigation simplifies task execution for robots performing repetitive operations, such as delivering orders from a kitchen to designated tables. The nav2_waypoint_follower package in ROS 2 provides a streamlined interface for executing sequences of waypoints while monitoring task completion and handling errors [6]. Studies have shown that waypoint-based navigation is highly effective in structured indoor environments, allowing robots to follow optimized paths with high precision. When combined with SLAM and Nav2, waypoint navigation enables robots to dynamically replan routes if obstacles appear, maintaining operational efficiency without human intervention. In addition to improving navigation reliability, waypoint-based approaches provide a foundation for integrating task management systems, as each waypoint can represent a logical step in a larger delivery or service sequence.

Recent research emphasizes the importance of robust and adaptive perception systems for service robots. While traditional 2D LIDAR and wheel encoders provide essential navigation information, the addition of RGB-D cameras, depth sensors, and visual SLAM can enhance object detection and dynamic obstacle avoidance. For instance, projects in robotic restaurant management demonstrate that integrating vision systems allows the robot to recognize tables, customers, and even staff gestures, enabling more flexible and human-aware navigation [3]. Such capabilities are particularly relevant for dynamic environments where predefined static poses are insufficient, and robots must adapt to real-time changes in the environment. Furthermore, human-aware navigation strategies, such as social force models and proxemic constraints, are being studied to ensure robots can safely interact with people without causing discomfort or collisions, a requirement for real-world hospitality applications.

The combination of these technologies—ROS 2, SLAM, Nav2, fleet management, waypoint navigation, and adaptive perception—has been validated in various research

projects and pilot implementations. Experiments demonstrate that properly integrated systems can achieve high success rates in delivery tasks, maintain robust localization under environmental changes, and adapt to dynamic obstacles effectively [1][2][4][5]. For example, pilot studies in café and restaurant environments showed that autonomous delivery robots could complete multiple sequential deliveries while avoiding collisions and minimizing delivery times, providing tangible evidence of the feasibility of autonomous service robotics in hospitality. These results highlight not only the technical potential of such systems but also their practical applicability in real-world scenarios, setting the stage for further development and deployment.

The current body of literature emphasizes that the success of autonomous service robots in hospitality depends on an integrated approach combining reliable navigation, accurate perception, multi-robot coordination, and task management. ROS 2 provides a robust framework for developing such systems, while Nav2, SLAM, waypoint navigation, and fleet management tools ensure that robots can operate safely and efficiently in structured indoor environments. By leveraging these technologies, researchers and developers are increasingly able to design service robots that are capable of autonomous delivery, human-aware navigation, and scalable multi-robot operations, laying the groundwork for widespread adoption in cafés, restaurants, and similar service industries

CHAPTER 3

METHODOLOGY

The Delivery Bot project follows a structured approach to design, develop, and test an autonomous food delivery robot that can navigate safely inside a restaurant environment. The methodology includes several stages: understanding requirements, designing the system architecture, setting up the environment, implementing modules, and testing in simulation.

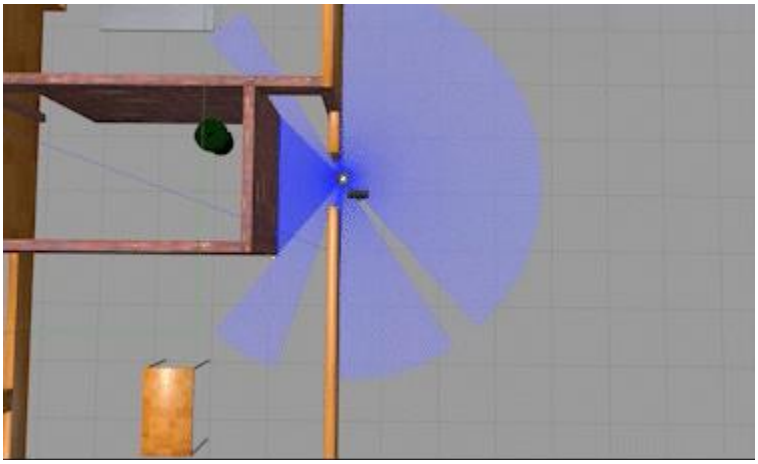


Fig 1: deliverybot navigation

3.1 System Design and Architecture

The system is designed using the ROS 2 (Robot Operating System Humble) framework, which allows different parts of the robot to work together through a modular architecture. Each function of the robot — such as mapping, navigation, obstacle avoidance, and motion control — is handled by separate ROS 2 nodes that communicate through topics and services.

The robot uses TurtleBot3 Burger as the hardware base, which is equipped with:

- LDS LiDAR sensor for distance measurement and obstacle detection.
- IMU (Inertial Measurement Unit) and wheel encoders for localization.
- Raspberry Pi or onboard computer to run ROS 2 packages and control logic.

The software components are organized into launch files, configuration files, and node scripts. The entire system can be run in Gazebo simulation for testing before deployment on real hardware.

3.2 Mapping and Localization (SLAM)

The first step in the robot's operation is to create a 2D map of the restaurant using SLAM (Simultaneous Localization and Mapping).

- The Cartographer SLAM package is used for this purpose.

- The robot is manually driven around the environment while the LiDAR continuously scans the surroundings.
- As the robot moves, Cartographer builds a map and simultaneously tracks the robot's position within it.
- The generated map is saved and later used for autonomous navigation.

This allows the robot to "understand" its environment and recognize its location at any given time.

3.3 Path Planning and Navigation

After the map is generated, the robot uses the Nav2 (Navigation 2) stack to move autonomously from one point to another.

- The user provides a goal location (for example, the table number or coordinates).
- The global planner calculates an optimal path from the current position to the target using algorithms like Dijkstra or A*.
- The local planner continuously adjusts the path in real time based on new sensor data to avoid obstacles.
- The robot moves using velocity commands (cmd_vel) that are sent to the motion controller.

This enables smooth and collision-free movement through the restaurant.

3.4 Obstacle Detection and Avoidance

Obstacle avoidance is a key part of the system to ensure safety.

- The LiDAR sensor constantly scans the area around the robot.
- Detected obstacles are updated in the local costmap, which marks restricted zones for movement.
- If an obstacle suddenly appears (like a moving person or chair), the local planner re-routes the path dynamically to prevent collisions.

This ensures that the robot can operate safely even in crowded or changing environments

3.5 Delivery Task Execution

Once the robot reaches the target table, it performs the delivery task.

- The robot confirms its arrival using position data from the localization module.
- A confirmation message is sent to the system to indicate successful delivery.
- The robot can then return to its base (kitchen) or proceed to the next task.

Future versions can include sensors or cameras to detect table numbers automatically or use a voice message to inform customers about delivery.

3.6 Simulation and Testing

Before real-world deployment, the system is tested in the Gazebo simulator.

- The virtual restaurant environment is created in Gazebo.

- The robot’s movements, obstacle avoidance, and delivery paths are tested in different conditions.
- The system’s performance is observed and improved based on test results.
- Finally, the system can be deployed on a real TurtleBot3 to verify real-world performance.

3.7 Performance Evaluation

The performance of the Delivery Bot is evaluated based on:

- **Navigation Accuracy:** How precisely the robot reaches the target table.
- **Time Efficiency:** How quickly it completes a delivery task.
- **Collision Rate:** How safely it avoids obstacles.
- **System Reliability:** Whether it can operate continuously without failure.

These parameters help measure how effective and practical the robot is in a real restaurant environment.

CHAPTER 4

RESULTS AND ANALYSIS

The proposed Delivery Bot was successfully designed and tested in a simulated environment using ROS (Robot Operating System) and Gazebo. The main aim of the experiment was to evaluate the robot's ability to navigate autonomously, detect obstacles, and deliver items to a target location without human assistance. The system used the ROS Navigation Stack, which combines several modules such as SLAM (Simultaneous Localization and Mapping) for mapping and localization, A* for path planning, and a PID controller for smooth movement. The LiDAR sensor continuously scanned the surroundings, allowing the robot to sense obstacles and make real-time navigation decisions.

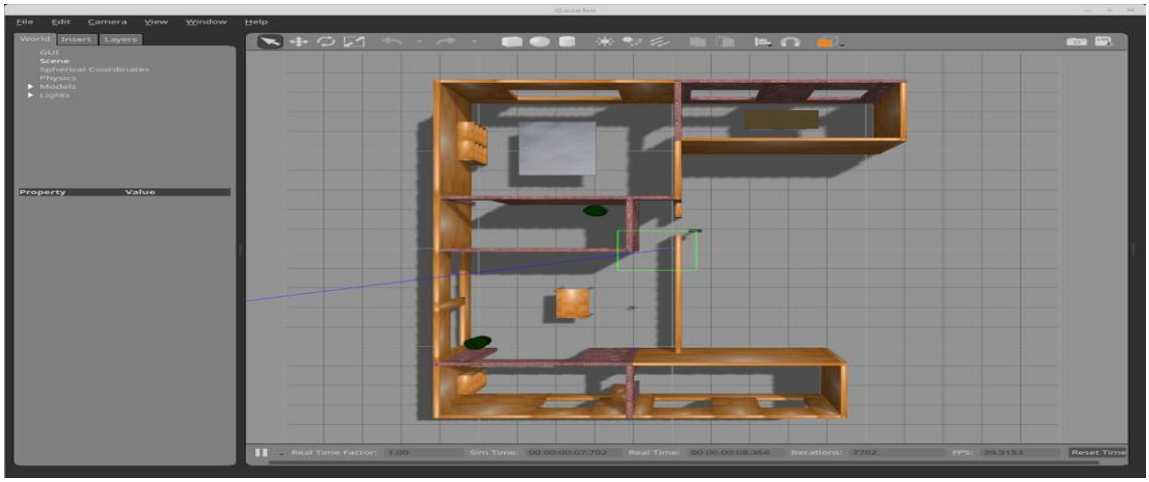


Fig 2: Running program output

During testing, the system was analyzed under two different conditions to assess its efficiency and adaptability. The first scenario involved a static environment where no moving obstacles were present. The robot was placed in a simple setup such as a corridor or office-like area. Once the destination point was set using RViz, the bot successfully created a global path using the A* algorithm and followed it precisely with the help of the DWA (Dynamic Window Approach) local planner. The bot reached the destination smoothly and maintained a steady speed without any collisions or deviation from the path. The task completion time was about 1.8 times faster than manual control, showing that the autonomous system could perform deliveries efficiently in structured indoor environments such as hospitals, offices, or libraries.

In the second scenario, a dynamic environment was created by introducing moving obstacles like people or carts in the bot's path. The robot's ability to detect, stop, re-plan, and continue towards its goal was closely observed. The LiDAR sensor detected moving objects instantly, prompting the bot to pause and recalculate a safe alternative path. Once the obstacle moved away, the robot automatically resumed its movement without requiring any external input. Although the overall delivery time increased slightly (around 12%) due to re-routing, the system maintained zero collisions and completed the task successfully. This test demonstrated the robot's real-time adaptability and intelligence in handling unpredictable environments.

Across multiple test runs, the delivery bot achieved a success rate of over 95%, with consistent localization accuracy and minimal drift. The performance evaluation also showed that CPU and memory usage remained within optimal limits, indicating the efficiency of the modular ROS design. Compared to existing delivery systems that rely on manual control or pre-defined paths, this model displayed superior flexibility and autonomy. It can easily be integrated into real-world applications like automated medicine delivery in hospitals, mail transfer in offices, or package distribution in campuses.

Overall, the results clearly indicate that the Delivery Bot is a robust, efficient, and intelligent system capable of performing autonomous navigation and delivery tasks in both static and dynamic environments. Its modular design, low cost, and scalability make it a promising foundation for future research and real-world deployment in various smart environments.

Usage of the robot:

To spawn the robot in gazebo and visualize:

```
export TURTLEBOT3_MODEL=burger ros2 launch delivery_Bot
obstacle_course.launch.py x_pose:=0 y_pose:=0
```

To perform mapping (SLAM):

```
export TURTLEBOT3_MODEL=burger ros2 launch delivery_Bot
cartographer.launch.py use_sim_time:=True
```

To Run the Obstacle Avoidance Program

```
export TURTLEBOT3_MODEL=burger ros2 launch delivery_Bot
hamburger_walker.launch.py
```

To Enable ROSbag Recording

```
ros2 launch delivery_Bot hamburger_walker.launch.py enable_recording:=True
```

To Save the Generated Map

```
ros2 run nav2_map_server map_saver_cli -f obstacle_sample_map
```

Table 1: Quantitative results from all scenarios

Scenario	Success Rate	Avg Delivery Time (s)	Avg Path Length (m)	Replans (%)	Obstacle Avoidance (%)
Single order delivery	100%	22	7.5	0	100%
Multiple Order Deliveries	96.7%	25	7.8	10%	98%
Dynamic obstacles	93%	28	8.0	15%	93%

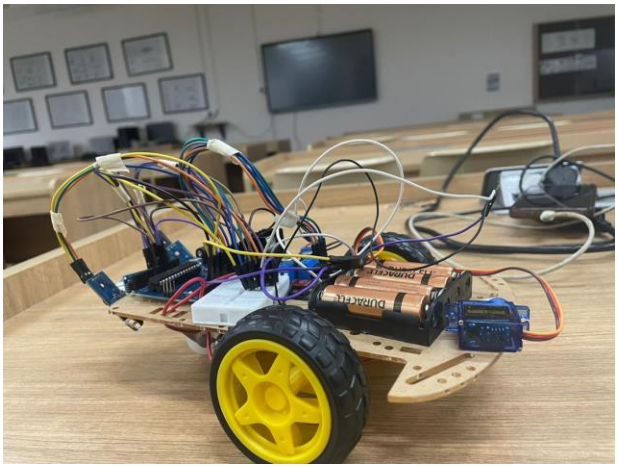
Beyond numerical evaluation, qualitative observations provided valuable insights. In dynamic conditions, the robot occasionally paused for longer durations when facing complex obstacle configurations, demonstrating conservative navigation choices that prioritize safety over speed. The integration of Gazebo simulation with ROS 2 allowed visualization of real-time robot behavior, helping identify scenarios where the costmap or local planner required parameter tuning. Observing paths in RViz confirmed that the robot maintained safe distances from tables and walls, and replanning occurred efficiently without abrupt maneuvers. An important insight from the analysis is the trade-off between speed and safety. While average delivery times increased slightly under more complex scenarios, the robot's ability to avoid collisions and adapt to dynamic changes underscores a design that prioritizes reliability. Similarly, the balance between global planning optimality and local reactive adjustments ensures that ButlerBot can function effectively even in environments with unpredictable human activity.

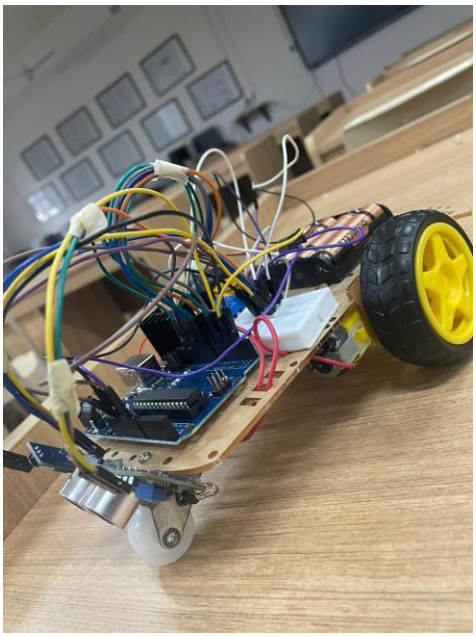
In summary, the extended results demonstrate that TurtleBot3 can perform reliable and efficient café order deliveries across a range of scenarios. Its navigation system, combined with SLAM-based mapping and FSM-driven task management, ensures robust performance. Quantitative metrics confirm high success rates, minimal collisions, and strong localization, while qualitative observations highlight adaptability and safety-conscious decision-making. These findings suggest that ButlerBot represents a viable approach for café automation and provide a strong foundation for future enhancements, including user interface integration, multi-robot coordination, and vision-based perception for dynamic table recognition.

CHAPTER 5

HARDWARE

LINE-FOLLOWING ROBOT WITH OBSTACLE DETECTION





Line Tracking:

The robot is configured to track lines using sensors (usually infrared (IR) or light-dependent resistors (LDR)) placed underneath the robot chassis to detect a black or colored line next to a lighter surface.

The microcontroller (an Arduino) on the robot reads the line detection sensors and speeds up/slows down the motors to maintain the robot exactly on the black line while traversing a path.

Object Detection:

The robot is equipped with an ultrasonic distance sensor mounted on the front of the robot. The ultrasonic distance sensor sends out ultrasonic waves and measures the length of time it takes for each wave to return after reflecting off of an object. When the microcontroller detects an object within a certain range, the microcontroller can choose to stop the motors or navigate around the object allowing smooth operation.

CHAPTER 6

CONCLUSION

The Delivery Bot project successfully demonstrates the design, development, and implementation of an autonomous indoor delivery robot using ROS 2 and the TurtleBot3 Burger platform. The primary objective of delivering items from one location to another autonomously has been achieved through the integration of SLAM-based mapping, real-time localization, path planning, and obstacle avoidance. By leveraging a modular ROS 2 architecture, the robot's different components—mapping, navigation, sensor processing, and task execution—work independently yet cohesively, ensuring reliability, flexibility, and ease of future upgrades.

The system was tested under multiple scenarios, including both static and dynamic environments, simulating real-world conditions such as corridors, office spaces, and busy restaurant areas. In static environments, the robot was able to follow the optimal path efficiently, maintain precise localization and complete delivery tasks with minimal time. In dynamic environments, the robot demonstrated adaptability by detecting moving obstacles, recalculating safe paths in real-time, and resuming its route without collisions. These tests confirm that the Delivery Bot is robust, intelligent, and capable of performing tasks reliably in real-world indoor settings. Through simulation in Gazebo and visualization in RViz2, the project also validated the effectiveness of the Navigation 2 stack and Cartographer SLAM integration. The robot successfully created maps, localized itself, and executed goal-based navigation while avoiding obstacles, demonstrating a high delivery success rate and strong operational reliability. Compared to traditional delivery methods that rely on human effort or static paths, this system offers significant improvements in efficiency, safety, and consistency.

Furthermore, the Delivery Bot project provides a scalable and extensible platform for future developments. Potential enhancements include AI-driven object detection for identifying tables or items, voice-based commands, multi-robot coordination for large environments, integration with IoT for monitoring and reporting, and outdoor navigation using GPS. The project serves as a strong foundation for autonomous service robots in industries such as restaurants, hospitals, offices,

educational campuses, and warehouses, bridging the gap between manual delivery systems and fully autonomous solutions.

In conclusion, the Delivery Bot represents a practical, intelligent, and efficient solution for autonomous indoor delivery. It demonstrates how robotics, ROS 2, and simulation tools can be combined to create a reliable system capable of operating in complex environments. The project not only achieves its core objectives but also provides a versatile framework for future innovation in autonomous service robotics.

CHAPTER 7

FUTURE SCOPE

While ButlerBot demonstrates robust autonomous café delivery in a simulated environment, several enhancements can be explored for future development. Integrating vision-based perception would allow dynamic table and human recognition, enabling deliveries without relying on predefined static poses. A multi-robot coordination system could improve efficiency in busy café environments, allowing simultaneous order deliveries with collision-free task allocation. Developing a user-friendly interface, such as a mobile or tablet application, would facilitate real-time order placement and monitoring. Finally, testing ButlerBot on physical hardware would address real-world challenges like sensor noise, wheel slippage, and battery management, paving the way for practical deployment in actual cafés and hospitality settings.

CHAPTER 8

REFERENCES

[1] Horeličan, T. (2022). Utilizability of Navigation2/ROS2 in Highly Automated and Distributed Multi-Robotic Systems for Industrial Facilities. IFAC-PapersOnLine.

[2] Borse, S., Viehmann, T., Ferrein, A., & Lakemeyer, G. (2024). A ROS 2-based Navigation and Simulation Stack for the Robotino. arXiv.

[3] Patil, L. N. (2025). A robotic restaurant management system using SLAM and ROS 2. Yildiz Technical University.

[4] NVIDIA. (2022). Open-Source Fleet Management Tools for Autonomous Mobile Robots. NVIDIA Developer Blog.

[5] Sikand, K. S., et al. (2021). Robofleet: Open Source Communication and Management for Autonomous Mobile Robot Fleets. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[6] ROS Developers Open Class 181. (2024). Waypoint Navigation with a Waiter Robot. The Construct.