An-Najah National University

Department of Computer Engineering

Hardware Graduation Project
Smart Vacuum Cleaner

Student Names:

Samah Qaradeh & Nour Kawni

Supervisor: Dr. Ashraf Armoush

A report submitted in partial fulfilment of the requirements of
An-Najah National University for
Bachelor degree in Computer Engineering

June 18, 2025

Abstract

Household chores like cleaning can become time-consuming and exhausting. Traditional vacuuming methods require manual effort and can be inefficient, often missing spots or requiring frequent intervention. Smart robotic vacuum cleaners provide an intelligent, automated solution, allowing users to save time and maintain a clean living environment effortlessly. This project aims to develop a smart vacuum cleaning robot capable of cleaning closed areas while autonomously navigating its environment. The robot utilizes LiDAR (Light Detection and Ranging) sensors to create high-precision maps of its surroundings, enabling accurate localization and efficient path planning. By integrating advanced mapping and navigation algorithms, along with obstacle detection and avoidance mechanisms, the robot optimizes coverage and ensures thorough cleaning of the entire room with minimal user intervention.

Acknowledgements

We would like to express our sincere gratitude to all the individuals who supported and contributed to the successful completion of our project. First and Foremost we would like to thank our supervisor, Dr. Ashraf Armoush, for his invaluable guidance, expertise, and continuous support throughout the duration of this project. His insights and feedback played a crucial role in shaping the direction and scope of our project. Finally, we would like to thank our families, friends, and loved ones for their unwavering support and encouragement throughout this project.Their belief in our abilites and their patience during our long working hours were invaluable in our journey towards success.

Disclaimer

This report was written by students at the Computer Engineering Department, Faculty of Engineering, An-Najah National University. It has not been altered or corrected, other than editorial corrections, as a result of assessment and it may contain language as well as content errors. The views expressed in it together with any outcomes and recommendations are solely those of the students. An-Najah National University accepts no responsibility or liability for the consequences of this report being used for a purpose other than the purpose for which it was commissioned.

Contents

List of Figures

List of Abbreviations

LIDAR	Light Detection and Ranging
SLAM	Simultaneous Localization and Mapping
ROS2	Robotic Operating System
URDF	Unified Robot Description Format
Rviz	ROS Visualization
AMCL	Adaptive Monte-Carlo Localization

Chapter 1

Introduction

1.1 Background

The rapid advancement of robotics and automation has significantly influenced modern household management, particularly through the rise of smart home technologies. Robotic vacuum cleaners, such as the Roomba, have become increasingly popular for automating daily cleaning tasks. Despite their widespread adoption, many current models still rely on random or semi-structured navigation methods, which often result in incomplete coverage, inefficient operation, and the need for frequent human intervention. As users seek more intelligent and autonomous solutions, technologies such as LiDAR (Light Detection and Ranging) and SLAM (Simultaneous Localization and Mapping) have emerged as critical enablers for enhancing robotic perception, navigation, and environmental mapping. These innovations open the door to developing more capable and efficient robotic cleaning systems.

1.2 Problem statement

Daily cleaning is essential to maintain a healthy living environment, but it is time consuming and physically demanding. Traditional vacuum cleaners require human involvement and manual operation. There is an increasing demand for smart home solutions that reduce human overwork. This project aims to address this issue by developing an autonomous cleaning robot that can navigate within closed indoor spaces and clean all reachable spots with less human intervention.

1.3 Aims and objectives

Aims: The aim of this project is to develop a smart vacuum cleaner robot that can autonomously navigate the indoor environment with minimal human intervention. The project combines mapping, localization, obstacle detection, and full-coverage room cleaning.
Objectives: The main objectives of this project are to design a robot equipped with cleaning components, utilize the LIDAR sensor for mapping and obstacle detection, implement SLAM for real-time mapping and localization, and develop and test navigation algorithms for efficient path planning and complete area coverage.

1.4 Organization of the report

This report is organized as follows:

1. **Literature Review:** In this chapter, we lay the groundwork by delving into the fundamentals of autonomous navigation in cleaning robots. we provide an overview of previous existing solutions.

2. **Methodology:** This chapter provides an in-depth look into the methodology behind our innovative approach. We describe the design of our robot, hardware components used and algorithms used to implement the robot.

3. **Results:** This chapter is dedicated to presenting and analyzing the outcomes of our work. We share the results that the robot produced from mapping the room, localization to path planning.

4. **Discussion and Analysis:** This chapter encapsulates the entire project, discusses the projects outcomes, strengths, and limitations. This is succinct summary of our achievements and the broader implications of our work

5. **Conclusions and Future Work:** Finally in the last chapter, we will talk about recommendations and possible additions to the project for better accuracy and performance of the robot.

Chapter 2

Literature Review

In the last few decades there was improvement and developing in the autonomous cleaning robots, with models available in the market like Roomba leading early innovations. At the beginning those robots relied heavily on randomized motion or simple infrared detection, which required so many user intervention [1].
Recently there has been advancement in the development of navigation robotics because more advanced robots has been using sensors like LiDAR and algorithms such as SLAM. LiDAR provides accurate distance measurements by emitting laser pulses and calculating the time taken for them to reflect back from objects, enabling robots to build a 2D or 3D map of their environment. LiDAR with SLAM can enable robots to construct a map and localize itself within it [2, 3].
Various SLAM implementations, such as GMapping, Hector SLAM, and Cartographer—have been integrated into robot operating systems like ROS (Robot Operating System), providing open-source frameworks for real-time mapping and localization. These tools allow robots to update their position even in dynamic or partially known environments, which is critical for household settings where furniture and obstacles may frequently change [4].
Path planning algorithms helps robots reach a specific point using shortest distance, this in return helps ensuring complete area coverage. Coverage Path Planning (CPP) algorithms such as Boustrophedon, Spiral, and Grid-based methods are widely studied in the context of cleaning and agricultural robotics. These methods aim to reduce redundancy and maximize coverage efficiency while accounting for obstacles and environment constraints [4].
This projects uses these research directions to help build a vacuum cleaning robot capable of intelligent navigation, mapping and full coverage path planning using ROS2.

Chapter 3

Methodology

3.1 Hardware Components

3.1.1 RPLIDAR A1

For the laser scanner, the RPLIDAR A1 sensor was used with the CP2120 chip, as shown in Figure 3.1. It is a 2D, 360 degree laser scanner capable of scanning within a 12-meter range, allowing the robot to interact with the environment. This laser scanner uses a single laser beam or rotating mirror to map a flat, two-dimensional view of its surroundings by measuring the time it takes for a laser pulse to return after hitting an object [5]. This LIDAR was chosen because of its low cost, availability, and suitability for indoor autonomous navigation.

Figure 3.1: RPLIDAR A1 laser scanner used in this project.

3.1.2 Raspberry Pi 5

Raspberry Pi in Figure 3.2 is a series of small single-board computers (SBCs) used in embedded systems and robotics. In this project, the Raspberry Pi 5 was used with the Ubuntu 24.04.2 operating system to run ROS2 (Robotic Operating System). Raspberry Pi receives data from both the LIDAR sensor and the Arduino Nano, processes the information, and computes the next move of the robot.

3.1.3 Arduino Nano

Arduino Nano in Figure 3.3 is a microcontroller board that is used to control the speed and direction of the DC motors that are responsible for robot movement. Interfaces directly with the motor driver to provide pulse width modulation (PWM) signals for speed control and digital outputs for direction control.
Additionally, the Arduino Nano communicates with the Raspberry Pi. via serial communication, allowing it to act as a motor controller while the Raspberry Pi makes a decision.

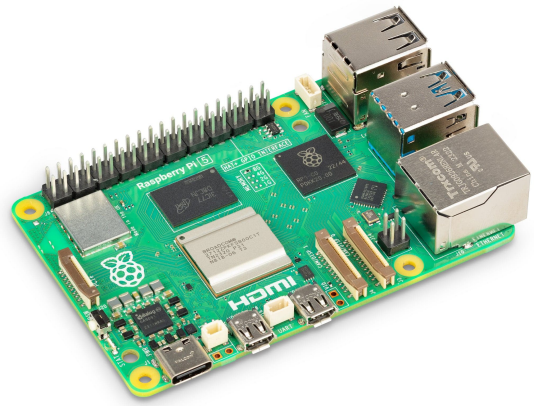

Figure 3.2: Raspberry Pi 5

Figure 3.3: Arduino Nano

3.1.4 DC Motors and Motor Driver

In this project, two DC motors were used as shown in Figure 3.4 to drive the robot. The motors are responsible for the robot's linear and rotational movements. To control the motors, the L298N motor driver in Figure 3.5 was used, a dual H-bridge driver module that allows us to control the speed and direction of each motor independently. It acts as an interface between the Arduino Nano and the motors, receiving low-power control signals from the Arduino and outputting higher-power signals to the motors.

Figure 3.4: JGA25-370 DC Motor

Figure 3.5: L298N Motor Driver

3.1.5 Power Supply

The robot was powered using two separate sources. a 12V 5A battery shown in Figure 3.7 was used to power the motor driver (L298N) which is responsible for controlling the robot's movement. This battery was used with buck converter shown in Figure 3.8, the main purpose of the buck converter is to reduce the input voltage to a lower output voltage, ensuring that the voltage provided to the motor driver does not exceed 12V. For the Raspberry Pi 5 a 5V 3A power bank was used as shown in Figure 3.6.

Figure 3.6: Power Bank

Figure 3.7: Battery

Figure 3.8: Buck Converter

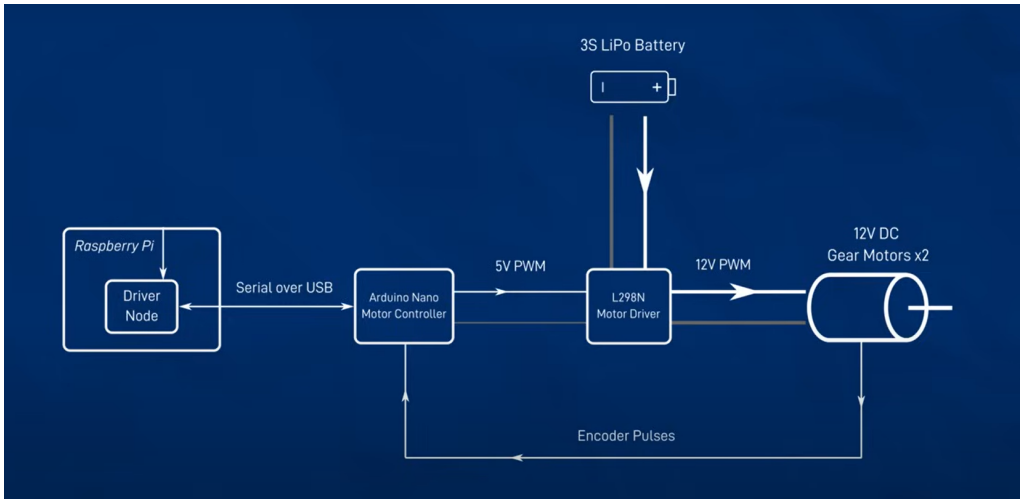

Figure 3.9: Full system architecture showing the interaction between the Raspberry Pi 5, Arduino Nano (motor controller), L298N motor driver, and the DC motors. Communication and power flows are clearly illustrated.

3.2 Robot Chassis Design Specifications

Vacuum cleaners generally have a circular shape, this shape makes it easy for the robot to move and maneuver around furniture and obstacles. In this project, the robot chassis design has two 40 cm circles, made of an aluminum composite panel; this material was chosen because it is light weight and makes our robot faster. The chassis was designed using AutoCAD software to ensure precise placement of all components, and then manufactured using a CNC machine to accurately cut the aluminum panel according to the final design. The first layer shown in

Figure 3.10 is the main layer that will hold the entire robot, it has two cutout rectangles for the two wheels connected to the DC motors, and two cutout circles on either side for the caster wheels; that helps balance the robot, Finally, a central rectangular cutout allocated for installing the vacuum motor. Additional holes for screws or wiring to secure and organize components. The second layer, as shown in Figure 3.11, is dedicated to holding the LIDAR sensor at the center of the robot. The screw hole pattern around the central cutout follows the exact spacing specified in the datasheet of our specific LIDAR model. Finally a third layer between the first and second layer for extra space where we put the RaspberryPi and the power bank as shown in 3.12.

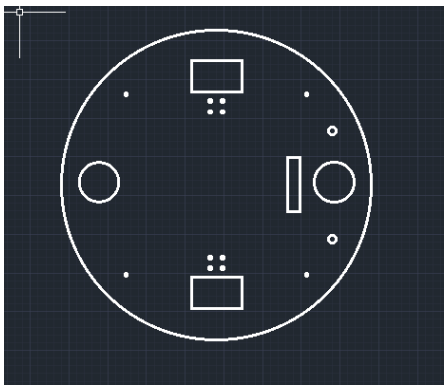

Figure 3.10: AutoCAD Design for the first layer

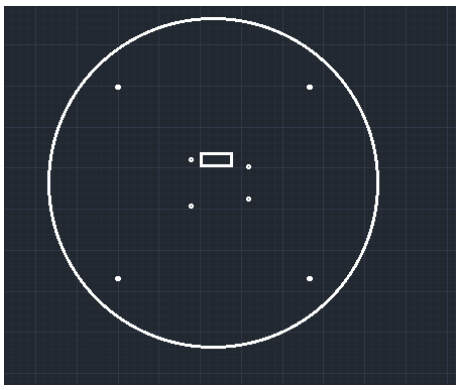

Figure 3.11: AutoCAD Design for the second layer

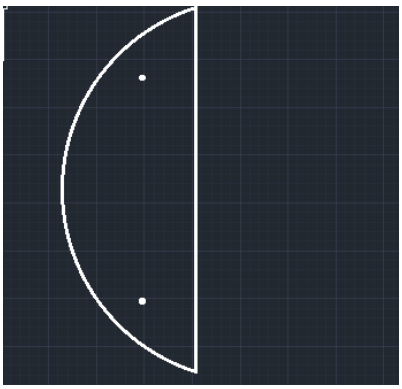

Figure 3.12: AutoCAD Design for the third layer

Figure 3.13: Robot Chassis

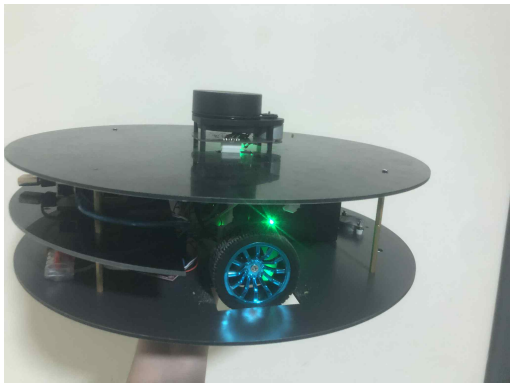

Figure 3.14: Robot Chassis

3.3 Implementation

3.3.1 Publisher Subscriber Architecture

In ROS2 running on raspberry pi, handles communication through publisher-subscriber architecture using topics. A topic which is the core communication system that ROS2 uses to transport data between systems. You can think of topics as channels of communication. A node can publish to that topic: this means that the node will publish data to this topic. A node subscribe to a topic: this means read data from this topic. This architecture allows for loose coupling between components: publishers and subscribers do not need to know about each other's existence, only the topic name and message type.

3.3.2 Unified Robot Description Format

Unified Robot Description Format (URDF) is a structured XML-based format used in ROS 2 to define a robot's physical and kinematic properties in ROS 2. The main purpose of URDF is to simulate the robot, especially when it is impossible to work with a real robot, when you want to create a physical robot that does not exist or if it is too risky to do preliminary tests on the physical robot. In this project, we used URDF to provide a virtual description of the real robot when publishing data in ROS.his virtual model enables visualization tools such as RViz (ROS Visualization) to accurately display the robot based on real joint states. The URDF File defines various aspects of robot, including Links– The rigid bodies that make up the robot, Joints- The connections between links that allow movement, Shapes and meshes– The visual and collision geometry of the robot, Sensors– Components that provide data about the environment, Actuator– Motors and other mechanisms that enable movement, Physical properties– Parameters like mass, inertia, and friction for simulation. The URDF file for the vacuum cleaner robot includes five links:

- **base_link**: The main chassis of the robot, represented by a green cylinder with a radius of 0.2 meters and a height of 0.088 meters.
- **laser**: The LiDAR sensor, represented by a white cylinder placed on top of the chassis, with a radius of 0.1 meters and a height of 0.05 meters.
- **left_wheel** and **right_wheel**: Both are represented by grey cylinders with a radius of 0.0325 meters and a length of 0.026 meters.

- **base_footprint**: A reference link placed at the bottom of the chassis, used for mapping and navigation.

The URDF file also includes four joints:

- **base_lidar_joint**: A fixed joint that connects the LiDAR sensor to the chassis.
- **base_left_wheel_joint** and **base_right_wheel_joint**: Continuous joints that connect the left and right wheels to the chassis.
- **base_footprint_joint**: A fixed joint that connects the `base_footprint` to the chassis at the bottom center.

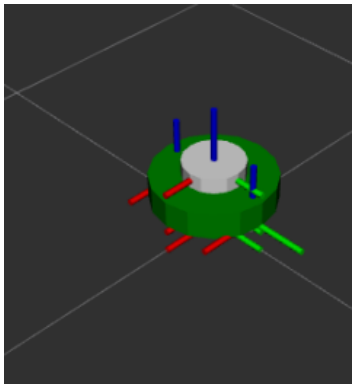

Figure 3.15: The URDF of the robot

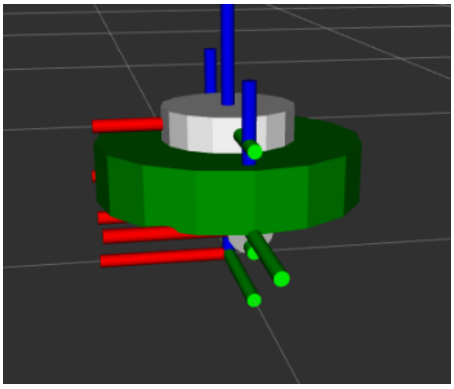

Figure 3.16: Side view of the URDF

TF (Transform Library) is very important in our project. it keeps track of multiple coordinate frames and maintains the relationship between them in a tree structure.

3.3.3 Arduino and velocity calculations

For our robot, it is very important to know the speed of the robot, and it is very important that we can give it any speed we want. For this we used a DC motor that has a built in encoder, from the encoder we can know the specific speed for our robot, we read these values from the Arduino, the DC motor and encoder is connected to Arduino, the encoders has 2 interrupts to measure speed and direction, we only used 1 interrupt because it is enough. In our case we do not have a specific documentation for our encoders, we want to know every revolution how many interrupt we get from the encoder, we calculate it by ourselves, we try many times, and take avareges to calculate it, We noticed that every motor revolution we got 400 interrupts (encoder counts or pulses), so for 1 dc motor if we have 400 counts in a minute we really have a speed for 1 rev per minutes, and so on, using this we can calculate the actual speed, and give the dc a motor a specific speed to go with it. 400 pulses is 1 revolution which is equal to 0.20 meter (because we have a wheel that has a diameter 6.5 cm). So finally from Arduino we got the speed in the unit rev/min for both wheels (later when we connect it with raspberry pi we will convert it to m/s) . We can give 2 wheels specific speed, and read the speeds for 2 wheels. Also we have a high frequency, we can read the velocity for 2 wheels every 20 milli seconds.

3.3.4 Hardware schematic:

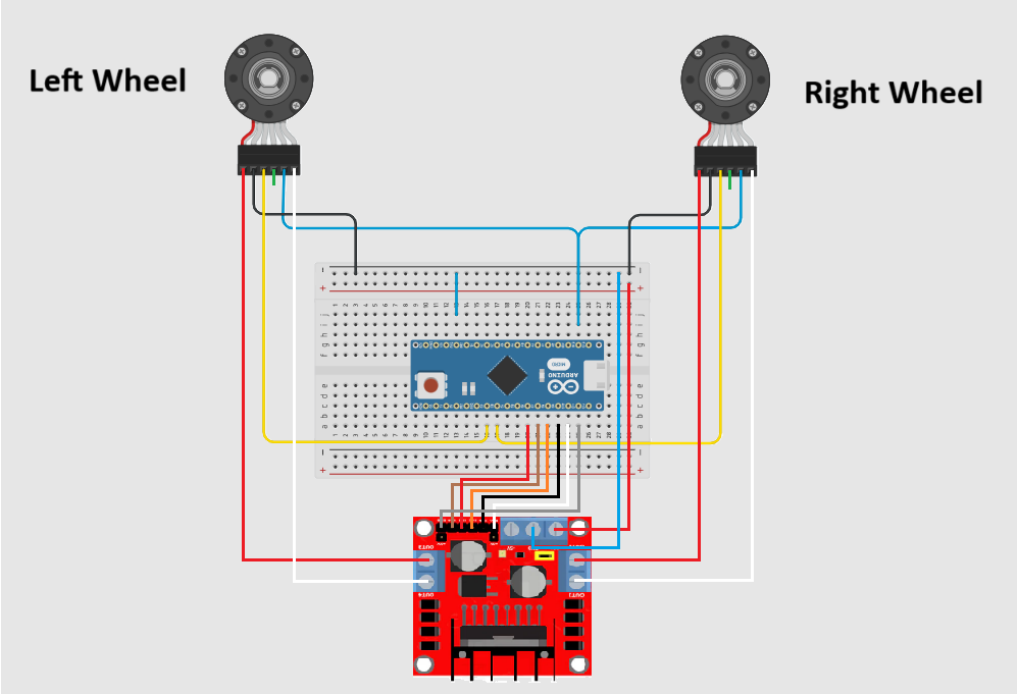

Figure 3.17: Hardware connection

As we said before we just use 1 interrupt pin from each encoder(yellow wire), we connect both dc motors with our driver , we used PWM, to manage the speed for both wheels, not just as enable, because we have different speed, sometimes we want our robot to move fast, sometimes too low (in path planning), for our driver it needs a voltage of 12 volts, and 2 amperes, so we connected it with a battery that can give this power, for Arduino we used nano and connected it with a raspberry pi, using serial cable, our raspberry will read and send the specific speeds for 2 wheels.

3.3.5 Odometry Implementation

In this topic we read the velocity from 2 wheels that are connected with Arduino. Our raspberry pi, reads both velocity for wheels from the Arduino, it reads 2 values, the velocity for right dc motor, and left dc motor, but actually in our ROS program we need the linear and angular velocity for the robot, so we used the velocity for both wheels to find linear and angular velocity using these equations:

$$V_I\ (m/sec) = \frac{V_R + V_L}{2.0} \tag{3.1}$$

$$\omega\ (rad/sec) = \frac{V_R - V_L}{\text{WheelSeparation}} \tag{3.2}$$

But for sure before of that we convert rev/min to the unit m/sec . After calculate angular and linear velocities we publish these values under topic called “odom”. This topic is very important for mapping, localization and path planning . Also this topic calculate accumulative position for the robot, and rotation, but for robot position we do not care a lot about it, because later we will have something called localization.

3.3.6 Robot Movement

When the robot is building the map. It will use a random movement algorithm to move in the room while still avoiding obstacles. In the following diagram 3.18 you can see the robot movement algorithm that we implemented.

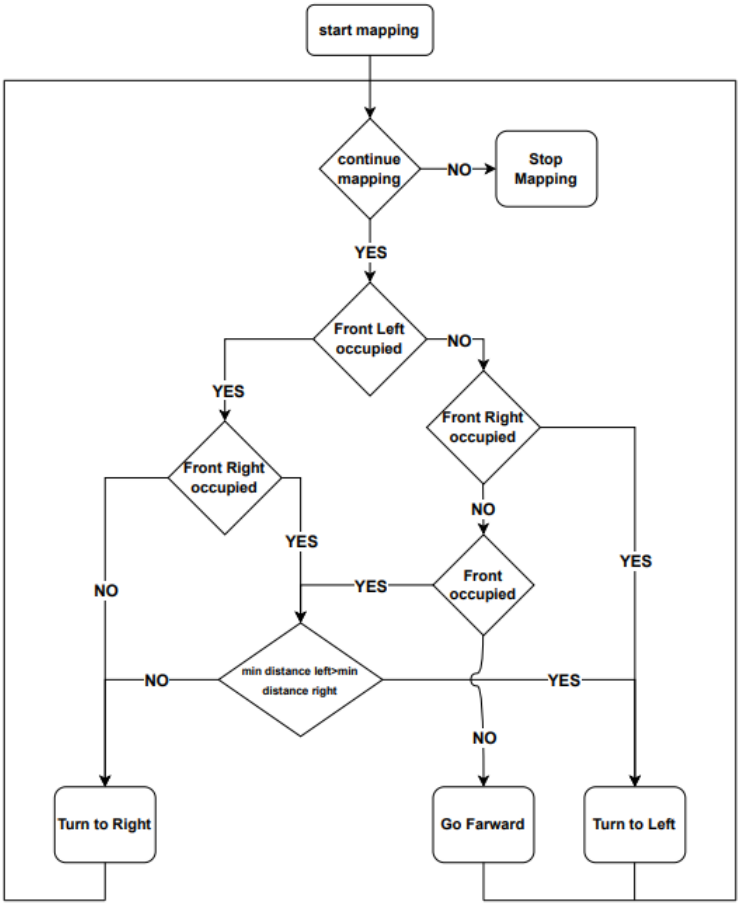

Figure 3.18: Robot Movement Algorithm

3.3.7 Mapping

Mapping is a fundamental part of robot navigation. Before a robot move efficiently in an environment, it needs to understand the space around it. A map is a representation of the environment in which the robot operates. The robot relies on a map to localize itself within the environment and plan trajectories to navigate safely from one point to another [6]. In ROS2, a map is typically an occupancy grid map, where each cell contains values that indicate whether an area is free, occupied by obstacle, or unknown. To build a map there is a technique that allows the robot to create a map while simultaneously determining its own location, this technique is called SLAM. In this robot the cartographer SLAM solution was used due to its simplicity, reliability, and ease of understanding. Cartographer is a real-time Simultaneous Localization and Mapping (SLAM) system that works in both 2D and 3D across various platforms and sensor setups. Developed by Google, Cartographer has even been used in

Google Street View to map building interiors [7]. To build a map in the robot, the robot should be in a closed-door environment and move it according to the robot movement algorithm we explained earlier, the LiDAR should start to detect obstacles and measuring distances and publish them to topic scan, this is essential for the cartographer to start building the map. Once the map is build it should be saved in map server. The map server provides the created map to other navigation applications like the localization or the path planner without the need to run cartographer every time. it works by reading the saved map and publishing it to the map topic every time the map server is launched.

3.3.8 Localization

Localization is determining the robot's position and orientation within that map. In this project, AMCL (Adaptive Monte-Carlo Localization) was used. It is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track a robot's pose against a known map [8].

3.3.9 Path Planning

We already have a pre-built map, so we stored and use it in path planning, here we give a robot a goal to go to it, and the robot really go to it while avoiding all obstacles in it is path, which already exist in the pre-built map. Also we make something to avoid a new dynamic obstacle that exist now, but not in the pre-built map. The robot compute the shortest path to the goal using Dijkstra algorithm. This shortest path take in consideration to avoid all obstacles, also to have a minimum around 30 cm inflation around the obstacle as a recovery plan. To make sure that our robot will not collide with any obstacle. For new dynamic obstacle we used algorithm called DWA (dynamic window), to make a local map, we have both local and global path, global path for pre-built map, local path for few meters around the robot, to make sure to avoid any new obstacles. Our path is dynamic it can change if any new obstacle occurs, or if our robot discovers new shortest and safe path. Also as we said before we used Dijkstra algorithm , we can use A* instead of it, but we prefer Dijkstra because it guarantees to give us a path for any reachable point. The path planner publishes linear and angular velocities to the topic `/cmd_vel`. We wrote a node to subscribe to this topic, convert these two values into the right and left wheel velocities, and send them to the Arduino via serial communication.

$$V_R = \frac{(2 \cdot v) + (\omega \cdot WheelSeparation)}{2 \cdot WheelRadius} \tag{3.3}$$

$$V_L = \frac{(2 \cdot v) - (\omega \cdot WheelSeparation)}{2 \cdot WheelRadius} \tag{3.4}$$

For sure we convert it to units rev/min before sending it to Arduino.

3.3.10 Full Coverage

As we did the path planning from point to point. it is now easy to do full coverage. we followed a simple algorithm that takes the map and generate a goal points that the robot has to follow in order to clean the whole map. this algorithm is zigzag movement. we chose it because of its simplicity and it guarantees the clean of the entire room. The algorithm shown in 3.19 moves back and forth in straight lines, covering the area like a lawnmower.

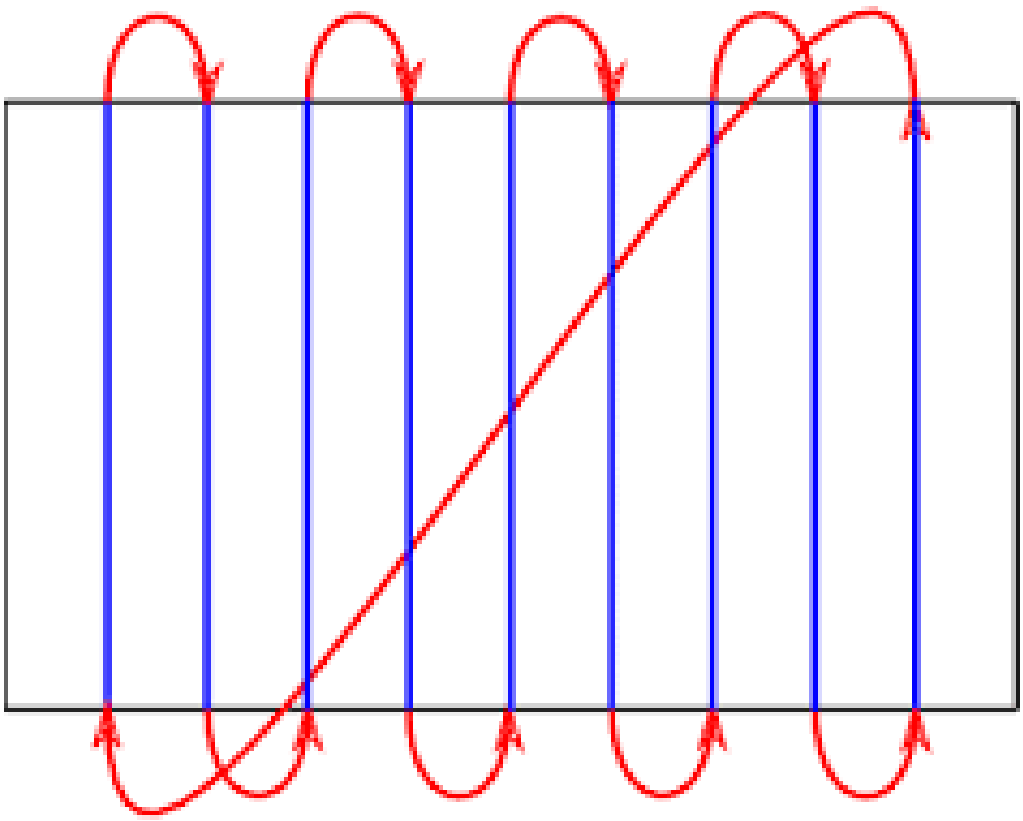

Figure 3.19: Zigzag movement

.

Chapter 4

Results

4.1 Mapping

Figure 4.1 shows the final 2D map generated using Cartographer SLAM algorithm. The robot moved in the room using the robot movement algorithm mentioned earlier in the same time laser scans from RPLIDAR AI sensor resulted in a high quality map.

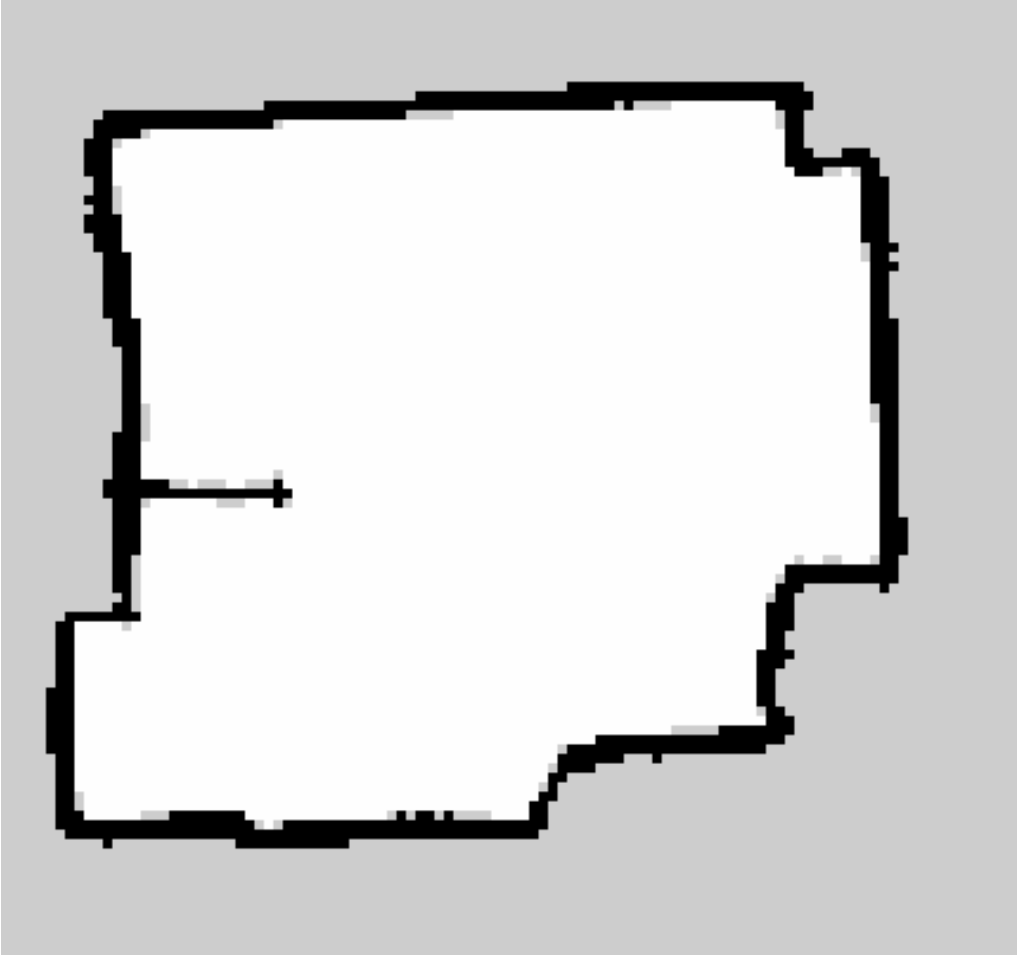

Figure 4.1: Map of the indoor environment generated by the robot using Cartographer SLAM

4.2 Localization

Figure 4.2 shows the robot localize itself within the room using AMCL. The robot is correctly localized when the reading of the laser scan matches the boundaries of the room.

Figure 4.2: Robot Localize itself within the room

4.3 Path Planning

Figure 4.3 shows the robot planning a path to the goal using Dijkistra. it follows that path unitl it reaches the goal. of course the robot must localize itself within the map before moving.

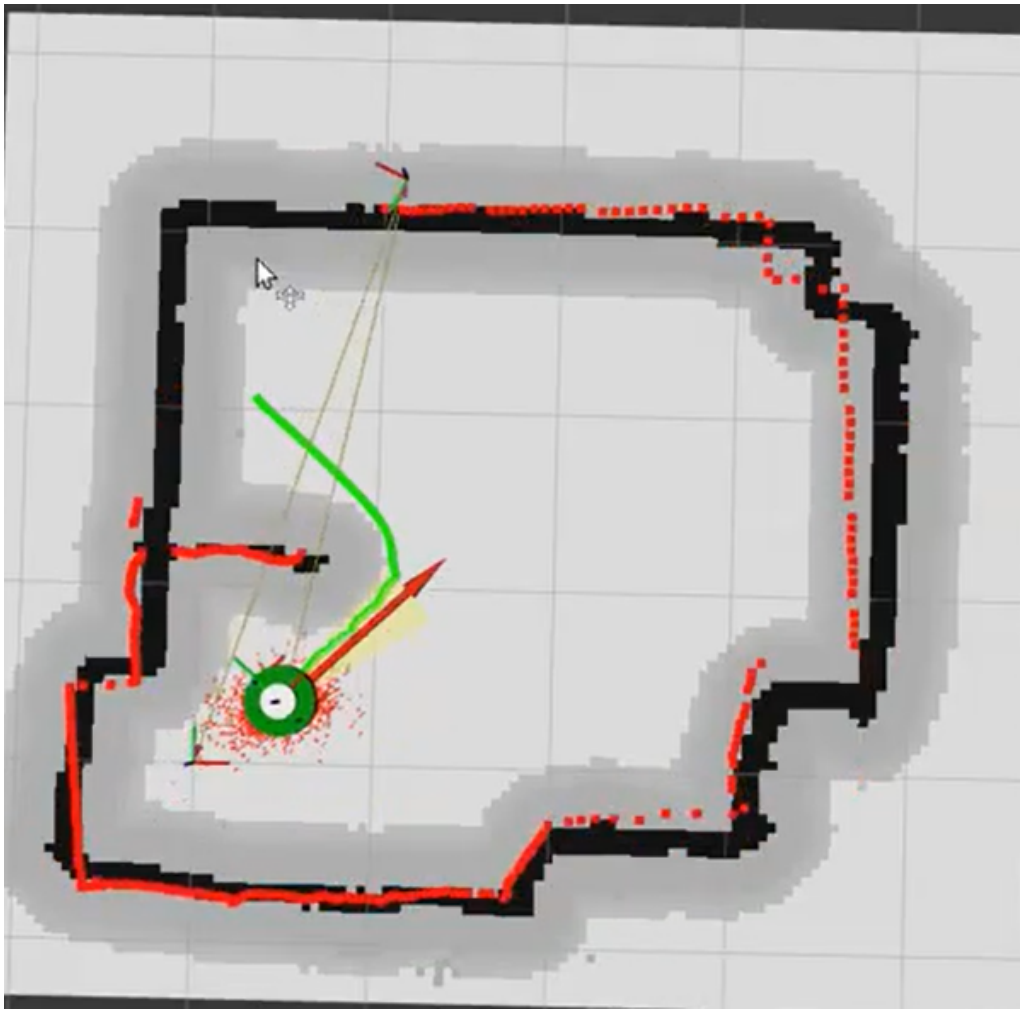

Figure 4.3: Path Planning

4.4 Dynamic Obstacles

Figure 4.4 shows the robot detects and obstacle that is not in the map. therefore it plan a path to reach the goal while still avoiding this dynamic obstacles.

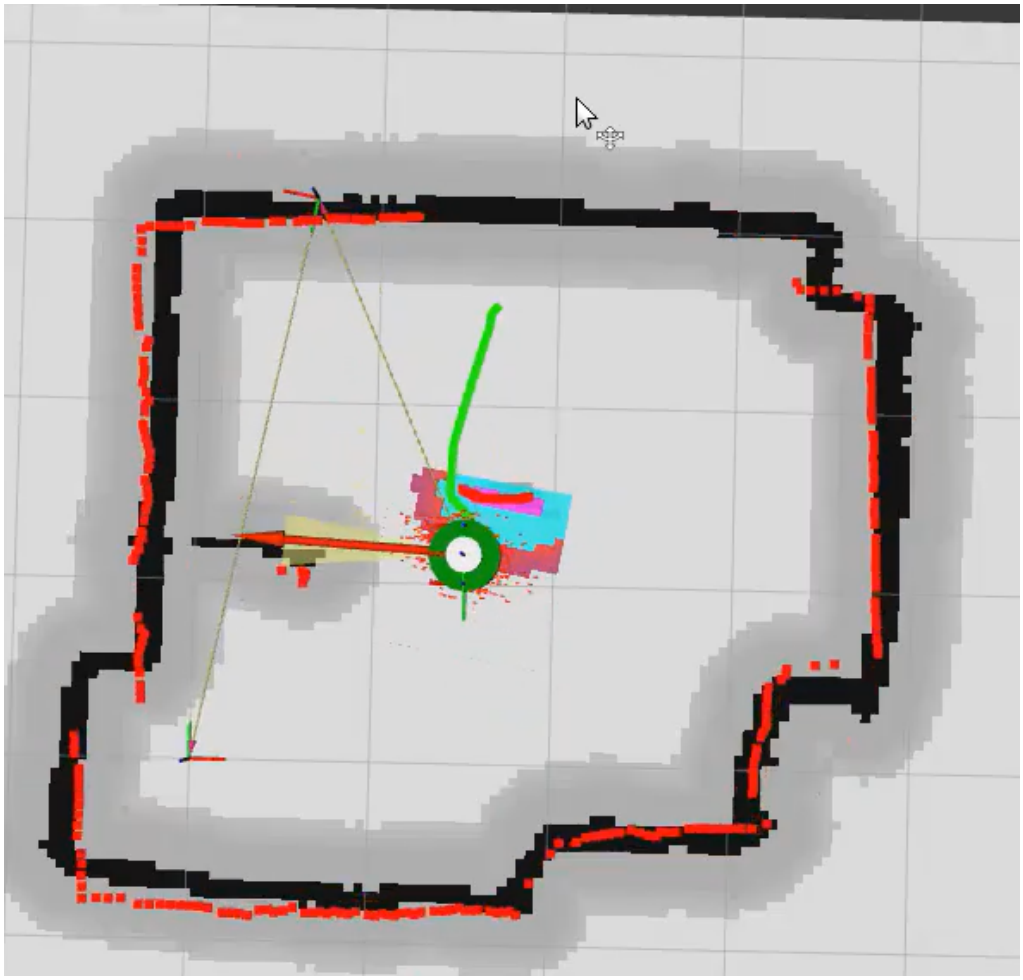

Figure 4.4: Path Planning

4.5 Full Coverage

Figure 4.5 shows the path that the robot will take to clean the entire room. The blue dot represents the base station that the robot will return to once it finishes cleaning.

Figure 4.5: Full Coverage path planning

Chapter 5

Constraints

1. Raspberry Pi 5 needed 5V 5A to operate fully but the power banks available in the market were only 5V 3A that required us to keep the power bank battery above 90 percent in order for the system to operate as needed.
2. When working with DC motors they most definitely won't be identical. This will cause one of the motors to rotate faster than the other one even if they are set at the same speed. This is highly noticeable when moving the robot in a straight line, where the robot will lean to the direction of the faster motor. This becomes even more observable when operating the motors at lower speeds. To accommodate this issue, we have used the encoders' circuit that is built-in, where the encoders readings are used to know the velocity that each motor is operating at.
3. Processing power of the Raspberry Pi was very limited therefore we connected it to our laptop to do simulations and computing.
4. To connect the Raspberry Pi 5 wireless we had to buy a dummpy-hdmi because it did not have build in vnc because Raspberry Pi 5 is a new version.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In conclusion, We needed to build a better autonomous cleaning robot therefore we used smart sensors and algorithms. we built the map of the room using cartographer algorithm then we used adaptive monte-carlo algorithm for the robot to localize itself within the indoor environment, we used Dijkstra to guarantee that our robot reach the goal point using shortest path and finally we used an algorithm to help the robot move throw the entire room and clean it to ensure that no spot is left uncleaned.

6.2 Future work

1. Use ultrasonic sensors to detect obstacles that LiDAR can not see because it is on top of our robot.
2. Use IR sensors to detect stairs so the robot does not fall.
3. Remote Monitoring and Control.
4. Integration with Smart Home Systems.

References

[1] S. Thrun, W. Burgard, and D. Fox, "Probabilistic robotics," 2005.

[2] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," in *IEEE Transactions on Robotics*, vol. 23, no. 1. IEEE, 2007, pp. 34–46.

[3] G. Dissanayake, P. Newman, H. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[4] H. Choset and P. Pignon, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.

[5] Wikipedia contributors, "Lidar — wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/Lidar, 2024, accessed: 2025-05-23.

[6] The Construct, "ROS2 Navigation Course," n.d., accessed: 2025-06-01. [Online]. Available: https://www.theconstruct.ai/robotigniteacademy_learnros/ros-courses-library/ros2-navigation/

[7] ROS-Industrial Consortium, "Ros2 cartographer tutorial," n.d., accessed: 2025-06-01. [Online]. Available: https://ros2-industrial-workshop.readthedocs.io/en/latest/_source/navigation/ROS2-Cartographer.html

[8] Robotics Knowledgebase, "Adaptive monte carlo localization (amcl)," n.d., accessed: 2025-06-01. [Online]. Available: https://roboticsknowledgebase.com/wiki/state-estimation/adaptive-monte-carlo-localization/