AUTONOMOUS MOBILE VACUUM CLEANING ROBOT

A DESIGN AND FABRICATION PROJECT REPORT

ABSTRACT

The Autonomous Mobile Vacuum Cleaning Robot represents a pioneering project that employs cutting-edge technology to redefine the realm of automated cleaning. By leveraging the capabilities of Robot Operating System 2 (ROS2) and LiDAR technology, this advanced robot seamlessly integrates a sophisticated robot chassis with LiDAR sensors. These sensors empower the robot to autonomously generate precise maps of its surroundings, strategically plan cleaning routes, and navigate through spaces with the agility to avoid obstacles. The inclusion of a user-friendly interface enhances its adaptability, positioning it as a versatile solution suitable for both residential and commercial environments.

The project's primary objectives revolve around maximizing the potential of ROS2 for robust communication, utilizing LiDAR technology for accurate mapping and obstacle avoidance, and designing an intuitive interface that facilitates seamless control and monitoring. The methodology adopted for this project involves the integration of ROS2 middleware, LiDAR sensors, and a meticulously crafted robot chassis, forming a comprehensive autonomous cleaning system. This integration not only contributes to the practical applications of autonomous cleaning but also serves as an invaluable educational tool.

Thus, the Autonomous Mobile Vacuum Cleaning Robot stands as a groundbreaking endeavour poised to revolutionize indoor cleaning processes. By harmonizing state-of-the-art technologies, this project not only addresses the immediate demand for autonomous cleaning solutions but also opens up new possibilities in the fields of robotics and automation. Its potential impact extends beyond functionality, offering a gateway for exploration and innovation in the dynamic landscape of robotics and intelligent systems.

INDEX

CHAPTER NO	TITLE	PAGE NO
	ACKNOWLEDGEMENT	
	ABSTRACT	
	LIST OF FIGURES	
1	**INTRODUCTION**	**1**
2	**LITERATURE SURVEY**	**3**
3	**PROBLEM DEFINITION**	**5**
4	**OBJECTIVES**	**6**
5	**EXISTING METHODOLOGY**	**8**
6	**PROPOSED METHODOLOGY**	**10**
7	**HARDWARE REQUIREMENTS**	**13**
	7.1 ARDUINO UNO R3	13
	7.2 RASPBERRY PI 4B	15
	7.3 LiDAR SENSOR	16
	7.4 N20 MICRO GEAR MOTOR WITH ENCODER	17
	7.5 L293D MOTOR DRIVER	19
	7.6 VACUUM MECHANISM	20
	7.7 SPINNING BRUSHES	22
	7.8 DC-DC CONVERTER	23
	7.9 RELAY MODULE	24
	7.10 Li-ION BATTERY	25
8	**SOFTWARE REQUIREMENTS**	**26**
	8.1 ARDUINO IDE	26
	8.2 ROS2	27

8.2.1 SLAM 29

8.2.2 AMCL 31

8.2.3 Robot State Publisher Package 32

8.2.4 ROS2_Control Package 34

8.2.5 RPLiDAR Package 36

8.2.6 Navigation2 Package 38

8.2.7 Cartographer 40

9 MODELING 42

9.1 CHASSIS DESIGN 42

9.2 BLOCK DIAGRAM 43

9.3 CIRCUIT DIAGRAM 44

9.4 WORKFLOW 44

9.5 APPLICATIONS 47

10 FABRICATION 50

10.1 3D PRINTED PARTS 50

10.2 ASSEMBLY 51

10.3 IMPLEMENTATION 52

11 SOFTWARE IMPLEMENTATION 57

11.1 ARDUINO PROGRAMMING 57

11.2 ROS2 PROGRAMMING 62

12 RESULT & DISCUSSION 63

13 CONCLUSION 64

14 FUTUREWORK 65

15 REFERENCES 66

16 BIBLIOGRAPHY 67

LIST OF FIGURES

Fig 1.1	-	Vacuum Cleaning Robot
Fig 5.1	-	Manual Sweeping
Fig 5.2	-	Vacuum Cleaning
Fig 5.3	-	Mop and Bucket
Fig 5.4	-	Steam Cleaning
Fig 6.1	-	Proposed System
Fig 7.1	-	Arduino Uno R3
Fig 7.2	-	Raspberry Pi 4B
Fig 7.3	-	LiDAR Sensor
Fig 7.4	-	N20 Micro Metal Gear Motor with Encoder
Fig 7.5	-	L293D Motor Driver
Fig 7.6	-	Vacuum Mechanism
Fig 7.7	-	Spinning Brushes
Fig 7.8	-	DC-DC converter
Fig 7.9	-	Relay Module
Fig 7.10	-	Li-ion Battery
Fig 8.1	-	Arduino IDE
Fig 8.2	-	ROS Humble
Fig 8.2.1	-	SLAM
Fig 8.2.2	-	AMCL
Fig 8.2.3	-	Robot State Publisher Node
Fig 8.2.4	-	ros2_control package
Fig 8.2.5	-	RPLiDAR Visualization
Fig 8.2.6	-	Navigation2 Package
Fig 8.2.7	-	Cartographer
Fig 9.1.1	-	Top Chassis
Fig 9.1.2	-	Bottom Chassis

Fig 9.2	-	Block Diagram
Fig 9.3	-	Circuit Diagram
Fig 9.4	-	AMVCR
Fig 10.1.1	-	Cleaning Motor Clamp
Fig 10.1.2	-	Cleaning Brushes
Fig 10.1.3	-	N20 Motor Clamp
Fig 10.2.1	-	Assembly - Isometric View
Fig 10.2.2	-	Assembly - Front View
Fig 10.3.1	-	Arduino UNO R3
Fig 10.3.2	-	Raspberry Pi 4B
Fig 10.3.3	-	RP LiDAR
Fig 10.3.4	-	N20 Gear Motor
Fig 10.3.5	-	L293D Motor Driver
Fig 10.3.6	-	Cleaning Brushes
Fig 10.3.7	-	DC-DC Converter
Fig 10.3.8	-	Relay Module
Fig 10.3.9	-	Li-Ion Battery
Fig 10.3.10	-	Standoff
Fig 10.3.11	-	Wheels
Fig 10.3.12	-	Ball Castor Wheel
Fig 10.3.13	-	Acrylic Sheet
Fig 10.3.14	-	Breadboard
Fig 10.3.15	-	Jumper Wires
Fig 10.3.16	-	Mini High-Speed Motor
Fig 13.1	-	AMVCR
Fig 16.1	-	Laser Cutting
Fig 16.2	-	3D Printing

CHAPTER 1

INTRODUCTION

In the realm of robotics and automation, the Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR emerges as a groundbreaking project, poised to revolutionize the way indoor environments are maintained. This innovative undertaking harnesses the power of advanced technologies, specifically Robot Operating System 2 (ROS2) and Light Detection and Ranging (LiDAR), to create a self-navigating vacuum cleaner capable of autonomously mapping and cleaning indoor spaces.

The core concept behind this project is to seamlessly integrate a robot chassis with LiDAR sensors, enabling the robot to perceive its surroundings, determine its location, plan efficient cleaning paths, and navigate autonomously while avoiding obstacles. The utilization of ROS2, a robust middleware designed for robotic applications, enhances communication and coordination between the various components of the robot, contributing to its overall efficiency.

Fig 1.1 Vacuum Cleaning Robot

One of the key features of this project lies in its potential widespread applications. The autonomous cleaning robot is designed for deployment in both residential and commercial settings, offering a practical solution to the ever-growing demand for efficient and autonomous cleaning processes. The fusion of ROS2 and LiDAR technologies not only addresses the functional requirements of autonomous cleaning but also provides a platform for exploration and learning in the fields of robotics and automation.

The significance of this project extends beyond its immediate application; it serves as a practical and educational opportunity for enthusiasts, students, and professionals seeking to delve into the intricate world of robotics. By combining cutting-edge technologies, the project offers insights into sensor integration, autonomous navigation, and system control, making it a valuable resource for those looking to expand their knowledge and expertise in these domains.

The subsequent sections of the project report will delve into the methodology employed, detailing the integration of ROS2 middleware and LiDAR sensors, the design and functionality of the robot chassis, and the development of a user-friendly interface for control and monitoring. As the project unfolds, it will become evident that this Autonomous Mobile Vacuum Cleaning Robot not only presents a technological marvel but also opens doors to practical applications and educational opportunities in the dynamic and evolving field of robotics.

CHAPTER 2

LITERATURE SURVEY

ROS2 and Robotics

Abstract

This literature review explores the pivotal role of ROS2 in contemporary robotics. ROS2, with its real-time capabilities and enhanced communication protocols, has revolutionized robot software development.

Conclusion

In conclusion, ROS2's contributions to real-time capabilities and communication within robotic systems make it a key enabler for our project. The findings suggest that ROS2 offers a robust and flexible framework for developing autonomous robotic systems.

RP LiDAR Technology in Robotic Navigation

Abstract

This literature review provides a comprehensive overview of the applications of LiDAR technology in robotic navigation. It examines the principles behind 2D and 3D LiDAR sensors and their pivotal role in mapping, obstacle detection, and localization

Conclusion

In summary, the review highlights the indispensable role of LiDAR technology in robotic navigation. The findings emphasize the importance of LiDAR in our project's success, as it enables 360-degree environmental perception and precise mapping, which are fundamental for autonomous navigation in various environments.

Advanced Path Planning and Navigation for Robotic Cleaners

Abstract

This literature review focuses on path planning and navigation for cleaning robots, examining relevant algorithms and strategies for efficient cleaning while avoiding obstacles, particularly applicable to our project.

Conclusion

In conclusion, advanced path planning and navigation strategies are crucial for autonomous cleaning robotics. These methods enhance the efficiency and effectiveness of our robot in real-world applications.

User Interaction and Human-Robot Interface in Cleaning Robotics

Abstract

This literature review investigates the significance of user interaction and interface design, with a focus on user-friendly control interfaces for autonomous robots, particularly cleaning robots. It assesses how these interfaces enhance the user experience and allow remote control.

Conclusion

In summary, the review underscores the pivotal role of user interaction and interface design in cleaning robotics. The findings emphasize the importance of user-friendly interfaces for controlling and monitoring our autonomous robot, ensuring a positive user experience and efficient operation.

CHAPTER 3

PROBLEM DEFINITION

Manual sweeping and mopping, while traditional, pose challenges such as being labour-intensive and time-consuming, particularly in larger areas. Thoroughly cleaning hard-to-reach spaces and corners can also be problematic with this method. Vacuum cleaning, although effective, introduces issues like noise pollution and dependency on electricity, limiting its application in quiet environments or during power outages. The use of a broom and dustpan may disperse dust into the air, potentially causing respiratory issues, and this method is not suited for wet cleaning. Mop and bucket techniques face challenges such as the potential spreading of dirt in the water and creating slip hazards on wet floors. Steam cleaning, despite its effectiveness in removing dirt and sanitizing surfaces, can be time-consuming and often comes with a high initial cost, making it less accessible for some users. Identifying these problems highlights the necessity for technological innovations, like the Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR, to overcome these limitations and revolutionize floor cleaning processes.

The Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR project was born out of a response to various real-time challenges in people's lives. These include time constraints, inefficiencies in manual cleaning, obstacle navigation difficulties, and the need for user convenience. The robot leverages LiDAR technology to create accurate indoor maps, offering a more efficient and autonomous cleaning solution while providing a user-friendly interface for control. Safety features are also integrated to ensure safe and reliable operation, ultimately offering a practical and educational solution to these real-world problems.

CHAPTER 4

OBJECTIVES

The primary objective of this project is to design and fabricate an "Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR." The project centres around the development of a cutting-edge, self-navigating vacuum cleaner with ROS2 serving as its core software platform. This involves integrating advanced features such as mapping, localization, efficient path planning, and obstacle avoidance, all powered by LiDAR technology. The primary focus is to create a highly autonomous system capable of efficiently and intelligently cleaning indoor spaces.

To achieve this, the project aims to implement a user-friendly interface for control and monitoring, ensuring accessibility for users in both residential and commercial settings. The interface will allow users to interact with and supervise the robot's cleaning operations seamlessly. The project team will leverage the capabilities of ROS2, a robust and flexible middleware designed for robotic applications, to enhance communication and coordination among various components, contributing to the overall efficiency of the system.

Continuous testing and optimization of the robot's performance are integral aspects of the project. This iterative approach ensures that the robot evolves to meet the highest standards of functionality and reliability. Comprehensive documentation will be generated throughout the development process, providing valuable insights and knowledge for future reference and knowledge sharing within the robotics community.

Scalability, safety, and reliability are paramount considerations in the project. The design and implementation of the robot will be geared towards ensuring that it can adapt to different environments and cleaning scenarios. Safety features will be integrated to guarantee secure operation, and reliability will be a key factor in the robot's day-to-day functionality.

Furthermore, the project will explore real-world applications of the Autonomous Mobile Vacuum Cleaning Robot in various indoor settings. This practical exploration aims to assess the robot's practicality, usability, and effectiveness in diverse environments. By doing so, the project seeks to validate the versatility and applicability of the robot in addressing the cleaning needs of different users and spaces. Overall, the multifaceted objectives of the project encompass technological innovation, usability, safety, and real-world applicability, positioning it as a comprehensive and impactful venture in the field of robotics.

CHAPTER 5

EXISTING METHODOLOGY

There are various existing methodologies for cleaning floors and collecting dust, ranging from traditional manual methods to advanced automated technologies. Some of the common methods include:

Manual Sweeping

This traditional approach involves using brooms, dustpans, and mop and bucket combinations. It's a labour-intensive process and is typically used in homes and small spaces.

Fig 5.1 Manual Sweeping

Vacuum Cleaning

Vacuum cleaners are widely used for collecting dust and debris from floors. They use suction to remove dirt and often come with various attachments to clean different surfaces and hard-to-reach areas.

Fig 5.2 Vacuum Cleaning

Broom and Dustpan

Similar to manual sweeping, a broom and dustpan are used to sweep and collect dust and debris. This method is often used in conjunction with other cleaning methods.

Mop and Bucket

A mop and bucket are used to wet-clean floors by mopping them with water and cleaning solution. This method is effective for removing stains and sticky residue.

Fig 5.3 Mop and Bucket

Steam Cleaning

Steam cleaning uses hot water vapor to clean and sanitize floors. It's effective in removing dirt and killing bacteria, making it suitable for various floor types.

Fig 5.4 Steam Cleaning

CHAPTER 6

PROPOSED METHODOLOGY

The below is the proposed methodology which outlines the step-by-step process for the development of Autonomous Mobile Vacuum Cleaning Robot,

Hardware Setup

- Ensure proper integration and compatibility of hardware components.
- Acquire and assemble the necessary hardware components, including a robot chassis, wheels, LiDAR sensors, motors, actuators, and a vacuum system.

ROS2 Implementation

- Develop and configure the Robot Operating System 2 (ROS2) as the core software platform.
- Create ROS2 nodes and packages for sensor integration, control, and communication.

Sensor Integration

- Mount LiDAR sensors on the robot chassis to provide 360-degree environmental perception.
- Implement sensor drivers and data processing to collect and interpret LiDAR data.

Mapping and Localization

- Utilize LiDAR data to generate accurate maps of the robot's surroundings.

- Implement SLAM (Simultaneous Localization and Mapping) algorithms within ROS2 to allow the robot to localize itself within the generated maps.

Path Planning and Obstacle Avoidance

- Develop path planning algorithms that consider both global and local planners to efficiently cover cleaning areas.
- Implement obstacle detection and avoidance strategies using LiDAR data to ensure the robot avoids collisions.

User Interface Development

- Create a user-friendly control interface, such as a mobile app or web-based dashboard, to allow users to start, stop, schedule cleaning sessions, and monitor the robot's status and progress.

Testing and Optimization

- Conduct comprehensive testing in both simulated and real-world environments to assess the robot's performance.
- Continuously optimize algorithms and sensor integration to improve cleaning efficiency and accuracy.

Documentation and Educational Resources

- Prepare detailed documentation, assembly instructions, and code explanations to enable others to replicate and understand the project.
- Develop tutorials and educational resources to promote knowledge sharing in robotics and automation.

Scalability and Adaptability

- Design the robot with scalability in mind, allowing for easy adaptation to different indoor environments.

- Consider potential extensions and additional features for future iterations of the robot.

Safety and Reliability Measures

- Implement safety features, including collision avoidance algorithms and emergency stop mechanisms, to ensure safe operation in various environments.
- Prioritize the reliability of the robot to minimize operational failures during cleaning tasks.

Real-World Applications Evaluation

- Explore and assess the practicality and usability of the autonomous vacuum cleaner in real-world scenarios, such as homes, offices, and different indoor spaces.

This methodology is intended to create a functional, adaptable, and user-friendly solution for autonomous cleaning.

Fig 6.1 Proposed System

CHAPTER 7
HARDWARE REQUIREMENTS

7.1 ARDUINO UNO R3

The Arduino Uno R3 is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. The board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). The Uno R3 also compatible with other software and hardware because of its Atmega328 microcontroller board.

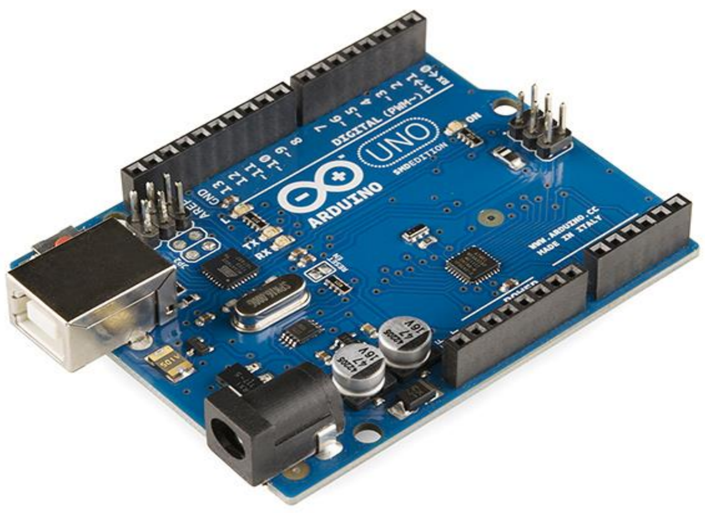


Fig 7.1 Arduino Uno R3

The various parts of the board are,

Microcontroller: The ATmega328P is the brain of the board, responsible for executing the code that is uploaded to the board. It has 32KB of flash memory for storing the code and 2KB of SRAM for temporary data storage.

Digital Input/output Pins: The board has 14 digital input/output pins (labelled 0-13) which can be used to read digital signals or send digital signals to other devices.

Analog Input Pins: The board has 6 analog input pins (labelled A0-A5) which can be used to read analog signals from sensors or other devices.

Power Jack: The power jack is used to connect an external power supply to the board. It can accept voltage between 7-12V.

USB Connector: The USB connector is used to connect the board to a computer for programming and power supply.

ICSP Header: The ICSP (In-Circuit Serial Programming) header is used for programming the microcontroller directly.

Reset Button: The reset button can be used to restart the microcontroller and run the code from the beginning.

LEDs: The board has 3 LEDs, one for indicating power status, one for indicating when data is being transmitted via the serial port and another one as L led.

Crystal Oscillator: The 16 MHz crystal oscillator is used to provide a stable clock signal for the microcontroller.

Voltage Regulator: The voltage regulator is used to convert the input voltage to a stable 5V to power the board.

Power Pins: The board has several power pins that can be used to supply power to other devices.

GND Pins: The board has several GND pins that can be used as a reference voltage for other devices.

7.2 RASPBERRY PI 4B

The Raspberry Pi 4 Model B is a versatile single-board computer developed by the Raspberry Pi Foundation. It is equipped with a powerful quad-core ARM Cortex-A72 processor, available in various RAM configurations (such as 2GB, 4GB, or 8GB). This microcontroller offers enhanced performance compared to its predecessors.

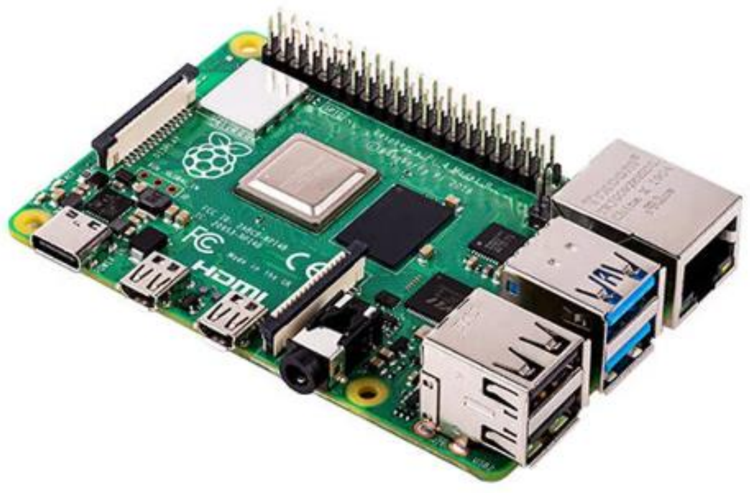


Fig 7.2 Raspberry Pi 4B

Processor: The heart of the Raspberry Pi 4 is the Broadcom BCM2711 system-on-chip (SoC), featuring a quad-core ARM Cortex-A72 processor clocked at 1.5 GHz. It provides the computational power needed for various tasks.

Memory: RAM options for the Raspberry Pi 4 range from 2GB to 8GB LPDDR4-3200 SDRAM, enabling efficient multitasking and performance for different applications.

Ethernet: A Gigabit Ethernet port allows wired network connections.

Wi-Fi and Bluetooth: Dual-band 802.11ac Wi-Fi and Bluetooth 5.0 offer wireless connectivity.

USB Ports: It has multiple USB ports, including two USB 3.0 and two USB 2.0 ports, for connecting peripherals like keyboards, mice, storage devices, and more.

HDMI Ports: Two micro HDMI ports support dual 4K video output.

GPIO (General Purpose Input/Output): The GPIO header contains pins that can be programmed for digital input or output. These pins allow interfacing with external devices and components, making it suitable for various hardware projects.

MicroSD Card Slot: It uses a microSD card for storage, allowing users to load operating systems and store data.

Camera and Display Interfaces: It features a camera serial interface (CSI) for connecting a Raspberry Pi Camera Module and a display serial interface (DSI) for compatible displays.

Power Connector: The USB-C power connector supplies power to the board. It requires a 5V USB-C power supply with a sufficient current rating.

Audio/Video Output: The 3.5mm audio/video jack supports composite video output and audio output.

7.3 LiDAR SENSOR

Here we are using RPLIDAR A1M8. LIDAR, which stands for Light Detection and Ranging, is a remote sensing technology that measures distances by emitting laser light and analysing the reflected pulses. It plays a significant role in various applications, including robotics, surveying, autonomous vehicles, and geographical mapping.

The core component of a LIDAR system is the LIDAR sensor. This sensor emits laser pulses and measures the time it takes for these pulses to bounce off objects and return. By calculating the time taken and knowing the speed of light, the sensor determines the distance to objects in its path. LIDAR sensors typically rotate or scan to capture a full 360-degree view, creating a detailed 3D map of the surroundings.

Fig 7.3 LiDAR Sensor

In robotics, LIDAR is crucial for navigation and obstacle detection. It provides accurate distance measurements and creates a precise map of the robot's environment, allowing it to navigate safely. Autonomous vehicles often rely on LIDAR to sense nearby objects, pedestrians, and other vehicles, ensuring safe and efficient navigation.

The technology has evolved, leading to the development of smaller, more affordable LIDAR sensors with improved accuracy and range. With advancements, LIDAR has become increasingly instrumental in a wide array of fields, contributing to enhanced spatial mapping, object detection, and real-time decision-making in various industries.

7.4 N20 MICRO GEAR MOTOR WITH ENCODER

Here, we are using N20 Micro Metal Gear Motor with Encoder. It is a compact yet powerful motor designed for precise control and accurate feedback in robotics and automation applications. It is renowned for its small size, high torque, and integrated encoder that provides valuable data for controlling the motor's speed and position.

The motor's gearhead is composed of metal gears, making it durable and capable of handling moderate loads efficiently. The inclusion of an encoder is particularly beneficial as it enables the motor to provide feedback on the motor shaft's position and rotational speed, ensuring precise control and facilitating tasks that demand accurate movements and rotations.

Its small form factor makes it suitable for various robotic projects, such as small-scale vehicles, miniature robotic arms, and motion control systems. The encoder delivers feedback in the form of pulses, allowing for fine-tuning of movements and enabling the motor to maintain consistent speeds and positions.

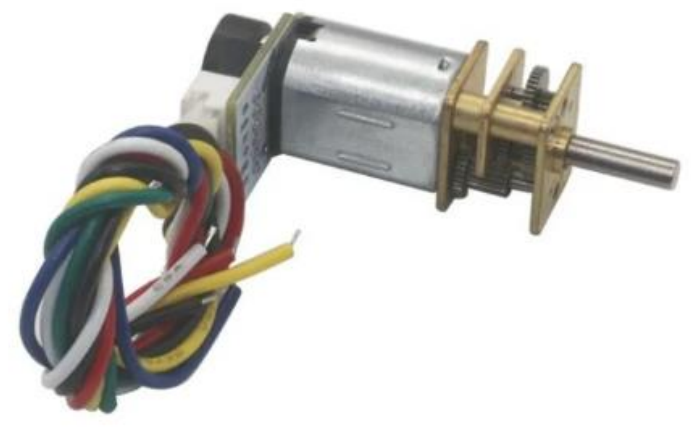


Fig 7.4 N20 Micro Metal Gear Motor with Encoder

By integrating the N20 Micro Metal Gear Motor with an encoder into a system, developers and engineers can achieve better control, increased accuracy, and smoother operation in their robotics projects, enhancing the overall performance and efficiency of their creations.

7.5 L293D MOTOR DRIVER

Here, we are using L293D Motor Driver. The L293D Motor Driver stands as a widely utilized integrated circuit (IC) in various electronic applications for motor control. This H-bridge driver allows bidirectional manipulation of DC motors, stepper motors, and diverse motor types. Its versatility lies in its capability to facilitate both clockwise and counterclockwise rotations, offering control for numerous motor functionalities.

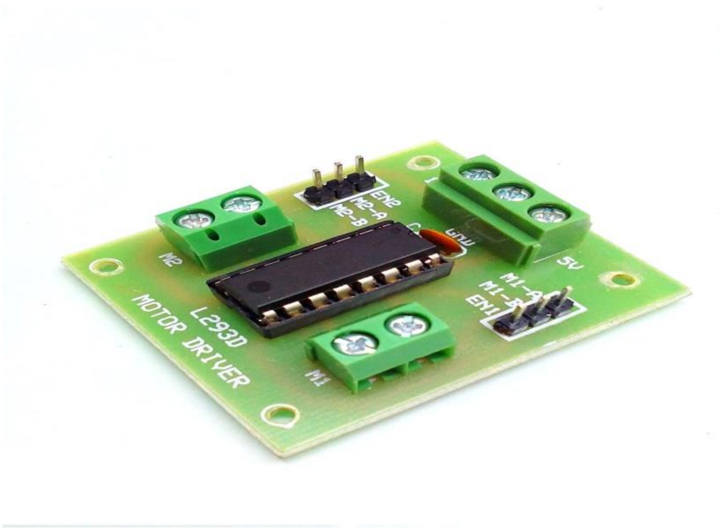


Fig 7.5 L293D Motor Driver

With dual input pins allocated for each motor, the L293D allows precise management: one pin for determining the motor's direction and the other for Pulse Width Modulation (PWM) control, enabling regulated speed adjustments. The IC features four output pins that directly link to the motor terminals, ensuring seamless connectivity.

Operating within a broad voltage range of 4.5V to 36V, the L293D accommodates diverse motor requirements. It can handle up to 600mA of current per channel, enabling efficient power delivery to the motors. Additionally, this IC incorporates protective mechanisms like thermal shutdown and crossover current protection, enhancing its reliability and durability in operation.

Its widespread use spans robotics and a spectrum of electronic projects where motor control is pivotal. The L293D interfaces seamlessly with microcontrollers and various control circuits, providing a versatile solution for motor manipulation in a myriad of applications, ranging from hobbyist projects to sophisticated robotic systems.

7.6 VACUUM MECHANISM

The vacuum mechanism in the "Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR" project is a crucial component responsible for efficiently collecting dust and debris from the floor. The vacuum system is designed to work seamlessly with the overall autonomous cleaning functionality of the robot. Here's an overview of the vacuum mechanism in this project:

Suction System

The vacuum mechanism incorporates a powerful suction system capable of effectively pulling in dirt, dust, and debris from the floor surface. The strength of the suction system is optimized to ensure efficient cleaning without causing damage to the flooring.

Adjustable Power Settings

To enhance adaptability to different types of flooring and cleaning requirements, the vacuum system may feature adjustable power settings. Users could potentially customize the suction power based on the cleaning task, whether it's a routine cleaning or tackling more stubborn dirt.

Filtration System

An efficient filtration system is integrated into the vacuum mechanism to trap fine particles and allergens, ensuring that the expelled air is clean. This is particularly important for maintaining air quality during the cleaning process.

Dustbin or Collection Container

The vacuum system includes a dustbin or collection container where the collected dust and debris are stored. This container is designed for easy removal and cleaning, promoting user convenience and regular maintenance.

Autonomous Operation Integration

The vacuum mechanism is seamlessly integrated into the autonomous operation of the robot. It collaborates with the LiDAR-based navigation and obstacle avoidance systems to ensure that the cleaning paths are optimized, and the robot covers the entire designated area.

User-Friendly Control

The vacuum mechanism is designed to be controlled through the user-friendly interface mentioned in the project objectives. Users can monitor and control the vacuum system, adjusting settings and initiating or pausing cleaning operations as needed.

Safety Features

Safety features may be incorporated into the vacuum mechanism to prevent any potential hazards during operation. These features could include sensors to detect obstacles and automatically adjust the robot's behavior to avoid collisions.

Battery Management

The vacuum mechanism operates in conjunction with the robot's power management system. It ensures that the vacuum system functions efficiently while optimizing the usage of the robot's battery power to achieve an appropriate balance between cleaning performance and operational time.

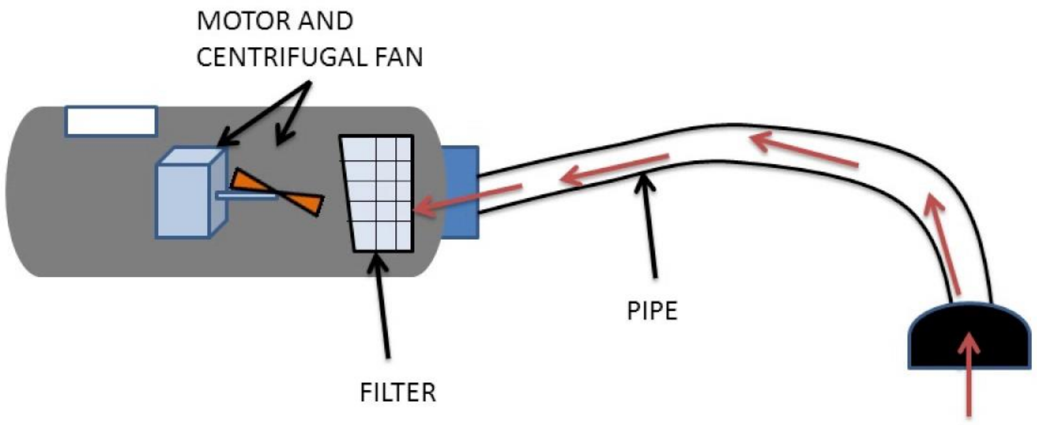


Fig 7.6 Vacuum Mechanism

The vacuum mechanism plays a pivotal role in the overall effectiveness of the cleaning robot, contributing to its ability to autonomously and thoroughly clean indoor environments. The integration of advanced features ensures that the vacuum system aligns with the project's objectives of creating an efficient, user-friendly, and autonomous cleaning solution.

7.7 SPINNING BRUSHES

Spinning brushes are commonly used in vacuum cleaning robots to complement the main suction mechanism and enhance the cleaning efficiency, especially in reaching edges and corners. These brushes are strategically positioned to sweep debris towards the suction inlet, ensuring a more thorough cleaning process.

Positioning

Spinning brushes are often positioned at the periphery of the vacuum robot, usually near the front or sides. This placement allows them to reach and agitate dust and debris along walls, edges, and corners where the main suction inlet might have limitations.

Fig 7.7 Spinning Brushes

Rotational Movement

The spinning brushes feature a rotational movement that sweeps debris inward toward the centre of the robot or the main suction opening. This movement helps gather particles that might otherwise be missed by the suction alone.

7.8 DC-DC CONVERTOR:

The DC-DC 12V to 3.3V, 5V, 12V power module is a versatile component designed to efficiently convert a 12V input voltage into multiple output voltages, including 3.3V, 5V, and 12V.

Fig 7.8 DC-DC Converter

This compact module incorporates voltage regulators for each output, ensuring stable and accurate voltage levels. With considerations for efficiency, current ratings, and thermal management, it caters to a variety of electronic applications, providing a reliable power supply solution. The module often includes protection features such as overcurrent and overvoltage protection, making it robust and safeguarding connected devices. Its multi-output design simplifies power supply management, and compatibility with different loads enhances its versatility for integration into diverse electronic systems.

7.9 RELAY MODULE

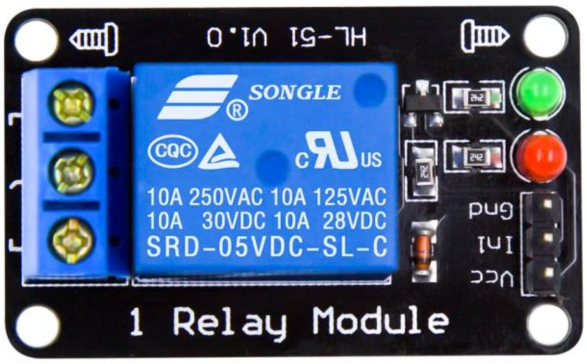


Fig 7.9 Relay Module

A relay module is an electronic device equipped with one or more relays that enable the controlled switching of electrical circuits. Operating at low input control voltages, typically compatible with microcontrollers, these modules act as intermediaries between low-voltage control signals and higher-voltage applications.

They come in various channel configurations, allowing independent control of multiple devices. With features like load ratings, protective measures, LED indicators, and compatibility with microcontrollers, relay modules offer versatility for applications ranging from home automation to industrial control systems.

7.10 Li-ION BATTERY

Lithium-ion batteries are commonly used in Arduino projects because of their high energy density and long lifespan. They are well suited for portable and mobile applications, making them a great choice for projects that require a small, lightweight power source. Li-ion batteries are also relatively low maintenance, which is ideal for use in applications that are not frequently monitored.

Fig 7.10 Li-ion Battery

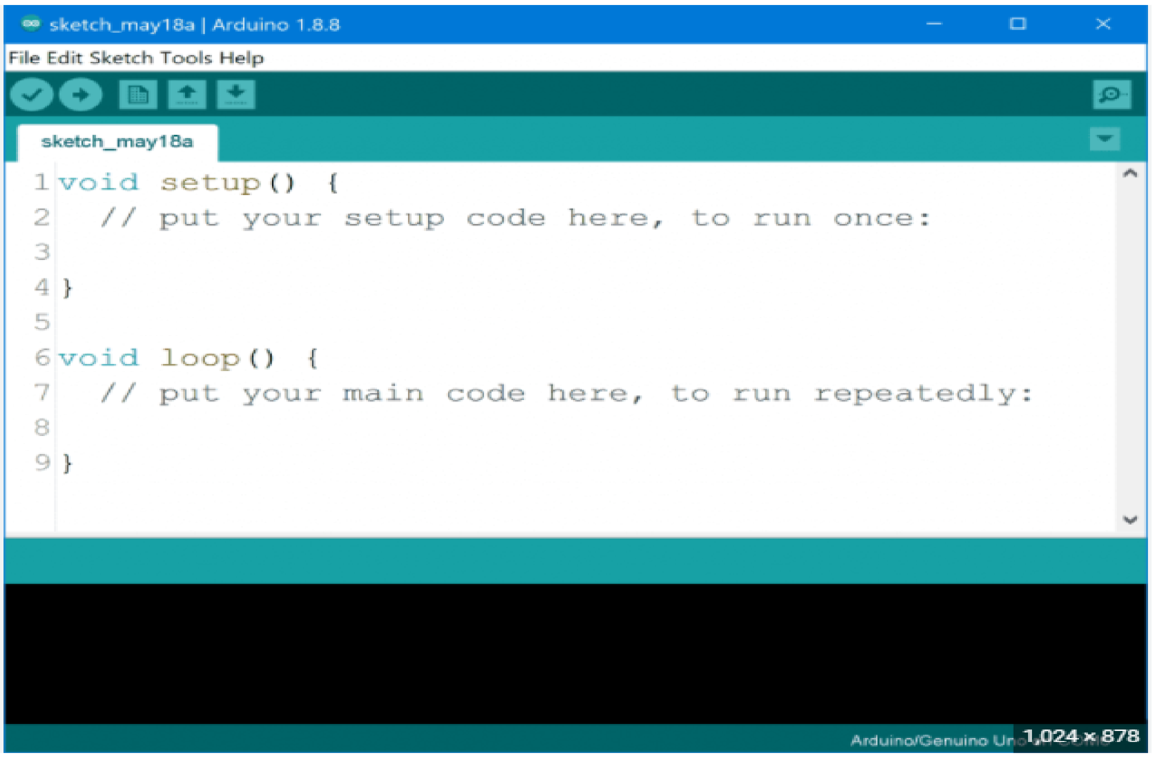
The batteries are usually connected to an Arduino board via a voltage regulator, which ensures the voltage level is appropriate for the board. Li-ion batteries are also commonly used in combination with solar panels, to increase the battery life of the Arduino projects by recharging it during the day. However, it is important to note that Li-ion batteries should be handled carefully, as overcharging or overheating can lead to decreased performance or even damage to the battery.

CHAPTER 8

SOFTWARE REQUIREMENTS

8.1 ARDUINO IDE

The Arduino Integrated Development Environment (IDE) is a user-friendly software platform used for programming Arduino microcontrollers. Developed to simplify the process of writing code for Arduino boards, the IDE offers an accessible environment for both beginners and experienced developers to create, upload, and debug code for various electronic projects.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig 8.1 Arduino IDE

Key features of the Arduino IDE include:

Code Editor: The IDE provides a text editor where users write and edit their code. It offers syntax highlighting, auto-indentation, and code suggestions to assist programmers in writing clean and error-free code.

Board Manager: Arduino supports a wide range of boards, and the IDE includes a board manager that allows users to select the specific Arduino board they are working with. This feature ensures compatibility between the code and the hardware.

Library Manager: Arduino IDE incorporates a library manager that grants access to a vast collection of pre-written code libraries. These libraries offer ready-to-use functions and code snippets, simplifying complex tasks and enabling users to easily integrate sensors, displays, communication protocols, and more into their projects.

Serial Monitor: It includes a serial monitor tool that facilitates communication between the Arduino board and a computer. This feature allows real-time monitoring and debugging of data being sent and received by the microcontroller.

Upload and Compilation: The IDE compiles the written code into machine-readable instructions for the Arduino board. It allows users to upload their compiled code onto the board via a USB connection, making the programming process seamless.

Community and Support: Arduino boasts a vibrant and active community of users, providing forums, tutorials, and extensive documentation. This community support is invaluable for troubleshooting, sharing ideas, and learning from others' projects.

8.2 ROS2 (ROBOT OPERATING SYSTEM 2)

ROS (Robot Operating System) is an open-source, flexible framework designed to aid in the development of robot software. Despite its name, ROS is not an operating system in the traditional sense but rather an ecosystem comprising tools, libraries, and conventions aimed at simplifying the creation of robust and complex robotic systems. It was initially developed by Willow Garage and is now maintained by the Open Robotics organization.

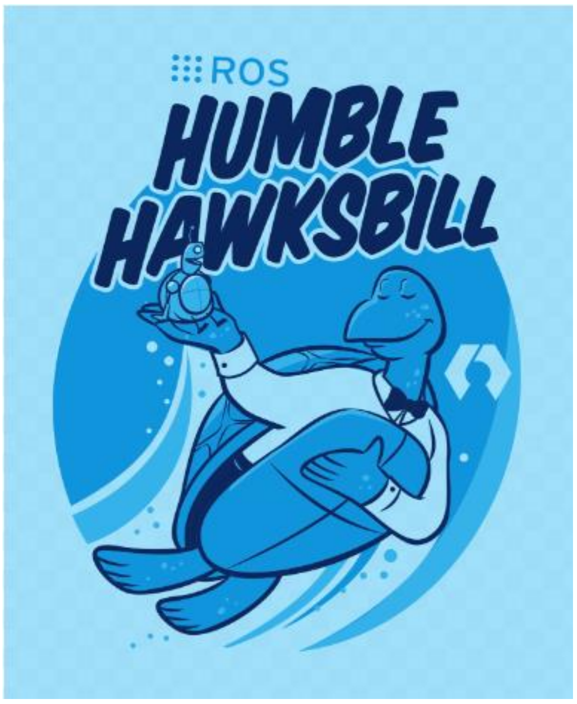


Fig 8.2 ROS Humble

Key components and features of ROS include:

Nodes and Communication: ROS operates using a distributed computing architecture, with individual software modules called nodes. Nodes can communicate with each other through a publisher-subscriber architecture, exchanging messages across a network. This modular design facilitates scalability and flexibility in robot system development.

Packages and Libraries: ROS offers a vast array of packages and libraries that cover various functionalities crucial for robotics, such as perception, navigation, manipulation, and more. These pre-existing components significantly expedite development and encourage code reuse.

Toolset: ROS provides a suite of powerful command-line tools for various tasks, including package management, visualization, debugging, simulation, and data recording and playback. Tools like rviz for visualization and rosbag for recording and replaying data streamline the development and testing processes.

Community and Collaboration: ROS boasts an active and expansive community of researchers, developers, and enthusiasts. This community actively contributes to the ecosystem by sharing code, tutorials, best practices, and participating in forums, fostering collaborative advancements in robotics.

Compatibility and Adaptability: ROS supports various programming languages such as C++, Python, and more, allowing developers to leverage their preferred language. Additionally, it can be deployed on multiple operating systems, including Linux variants like Ubuntu.

8.2.1 SLAM (Simultaneous Localization And Mapping)

The ROS SLAM Toolbox amalgamates libraries, tools, and algorithms crucial for Simultaneous Localization and Mapping (SLAM). Vital for autonomous systems, SLAM enables real-time navigation and environmental mapping. This toolbox within ROS empowers robots by integrating algorithms, facilitating sensor data interpretation, and aiding in constructing dynamic maps of their surroundings.

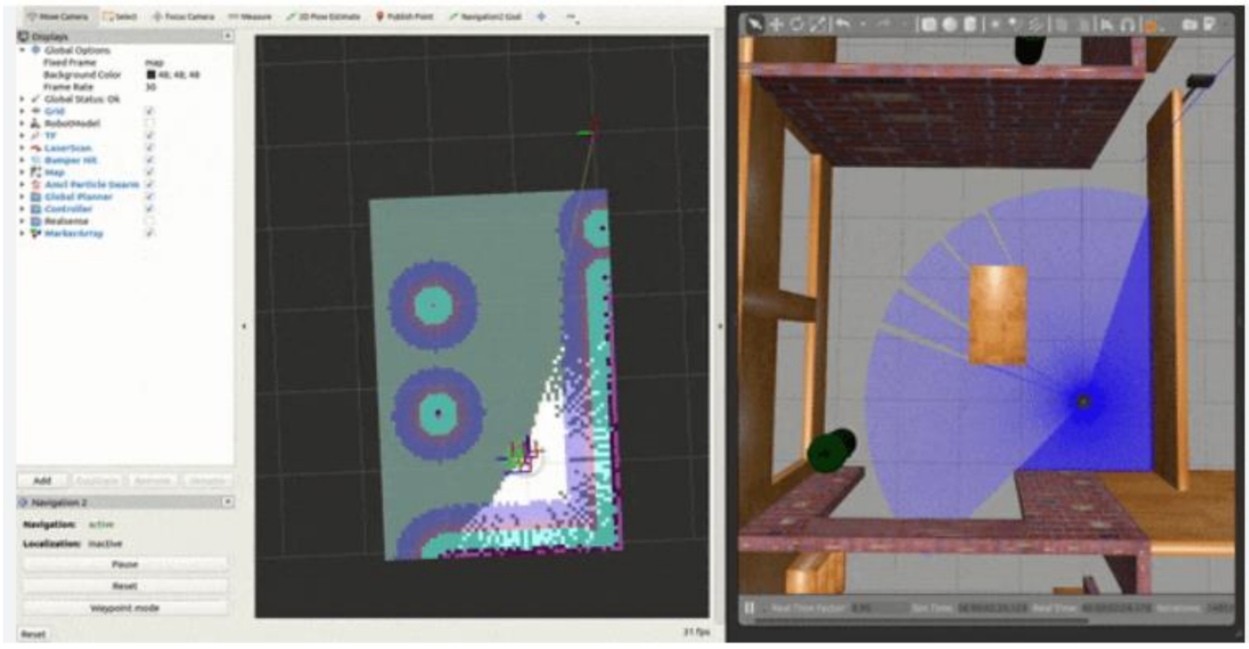


Fig 8.2.1 SLAM

It's an essential framework for robotic systems, underpinning their ability to navigate unknown terrains while concurrently creating and updating maps, contributing significantly to the advancement of autonomous robotics.

Within ROS, the SLAM Toolbox encompasses various components and functionalities:

Algorithmic Support: It integrates multiple SLAM algorithms such as Graph-based SLAM (GraphSLAM), FastSLAM, EKF-based SLAM (Extended Kalman Filter), and more. These algorithms utilize sensor data, often from cameras, LiDAR, or other range sensors, to estimate the robot's pose (localization) while simultaneously creating a map of the environment.

ROS Nodes and Message Passing: The SLAM Toolbox typically operates through ROS nodes, enabling modularity and easy integration with other ROS components. Nodes communicate via ROS messages, facilitating seamless data exchange among different parts of the robot system.

Visualization Tools: Visualization plays a crucial role in understanding the SLAM process. ROS often includes visualization tools like RViz (ROS Visualization) that allow users to visualize robot movements, sensor data, maps, and the localization process in a graphical interface.

Parameter Tuning and Optimization: The toolbox often offers parameters that users can adjust to optimize SLAM performance based on specific environmental conditions or sensor characteristics.

Community Contributions and Support: Just like the broader ROS ecosystem, the SLAM Toolbox benefits from a vibrant community. This community contributes code, updates, improvements, and provides support through forums, documentation, and tutorials.

8.2.2 AMCL (Adaptive Monte Carlo Localization)

AMCL, part of ROS, enables mobile robots' precise positioning within an environment via probabilistic localization, employing Monte Carlo or Particle Filter Localization. It estimates the robot's pose by maintaining a set of particles representing potential positions, adapting to changes and sensor data. This adaptive approach facilitates accurate localization, crucial for autonomous navigation in varying environments. AMCL's implementation in ROS contributes significantly to mobile robotics, ensuring effective and reliable positioning for autonomous systems in diverse real-world scenarios.

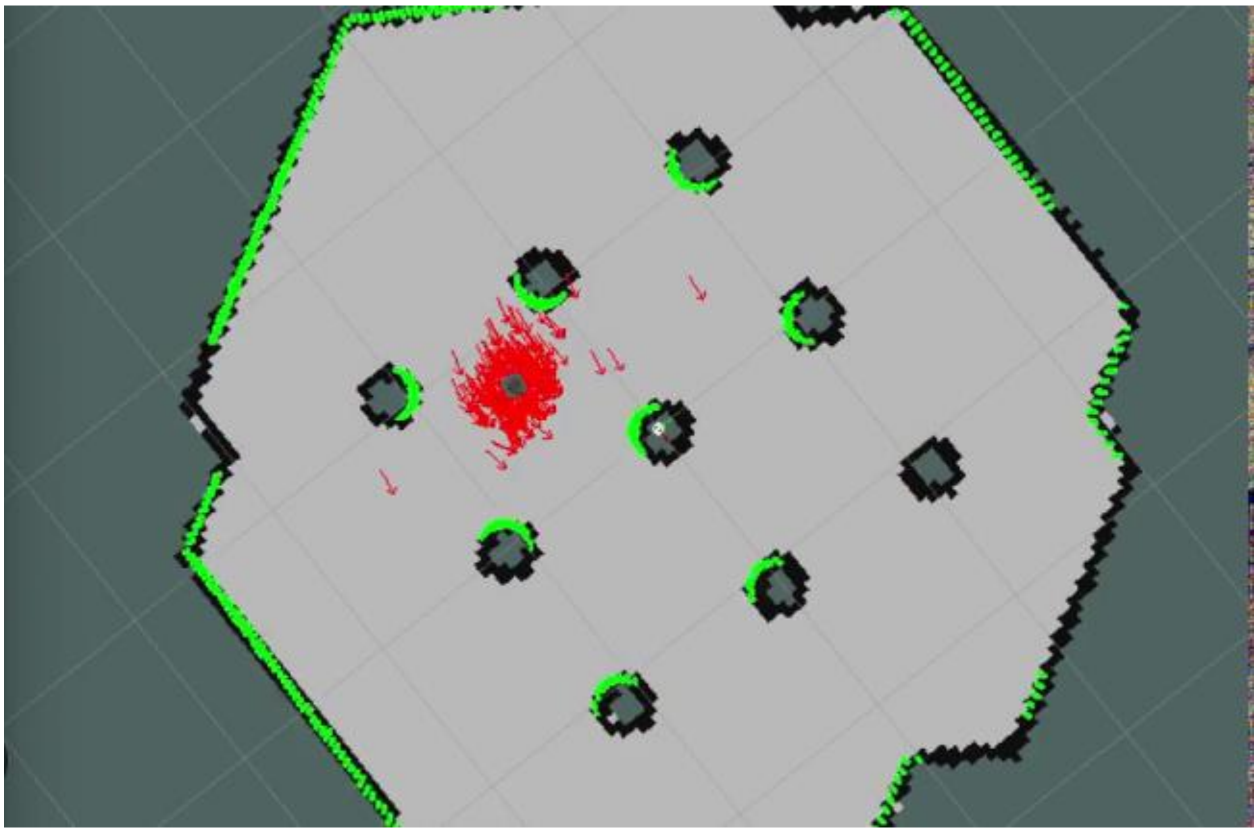


Fig 8.2.2 AMCL

Key features of AMCL include:

Particle Filter Localization: AMCL uses a particle filter-based approach to estimate the robot's pose (position and orientation) within a known map. It maintains a set of particles representing potential robot poses, with each particle assigned a weight based on how well it aligns with sensor measurements and the map.

Adaptive and Robust: AMCL adapts dynamically to changes in the environment and sensor data, adjusting the distribution of particles to improve localization accuracy. It's robust in handling uncertainties, such as sensor noise or dynamic environments.

Map Representation: It requires a map of the environment where the robot operates. This map can be provided in various formats like occupancy grids, allowing the robot to localize itself within the given map.

Integration with ROS: AMCL is seamlessly integrated into the ROS ecosystem, enabling communication with other ROS nodes responsible for sensor data acquisition, motion planning, and navigation.

Parameter Tuning: The package provides configurable parameters, allowing users to fine-tune the localization algorithm based on specific robot hardware, sensor characteristics, or environmental conditions.

Real-time Localization: By continuously updating the robot's pose estimation based on incoming sensor data, AMCL enables real-time localization, crucial for **accurate robot navigation.**

8.2.3 Robot State Publisher Package

The Robot State Publisher package in ROS (Robot Operating System) is a vital tool used in robotic systems to publish the state of a robot's components and its transformations in a standardized manner. It plays a crucial role in enabling robot visualization, simulation, and control by providing a consistent representation of the robot's configuration.

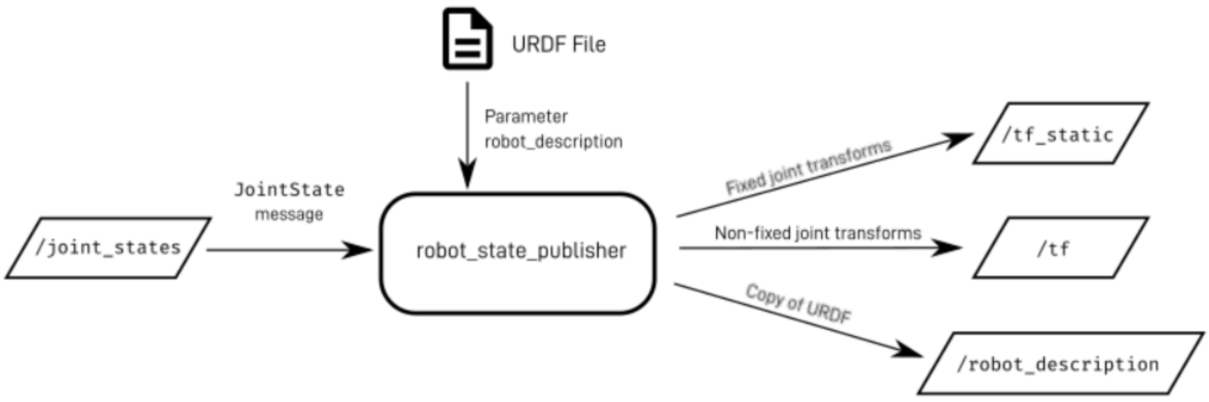


Fig 8.2.3 Robot State Publisher Node

Key functionalities and components of the Robot State Publisher package include:

tf2 Library Integration: The package utilizes the tf2 library, which manages coordinate frames and transformations in ROS. It helps in defining and broadcasting the relationship between different parts of the robot, such as joints, links, and coordinate frames, enabling a comprehensive representation of the robot's state.

URDF (Unified Robot Description Format): Robot State Publisher predominantly operates by using URDF, a standardized XML format in ROS that describes a robot's mechanical structure, including links, joints, sensors, and their spatial relationships. The package reads the URDF file and publishes the robot's state based on this description.

Joint State Publisher: Within the Robot State Publisher package, there is often a Joint State Publisher node that interfaces with joint state messages. It collects joint state data, such as positions, velocities, and efforts, and publishes this information, aiding in robot motion visualization and control.

Visualization and Simulation: The published state information is crucial for visualization tools like RViz (ROS Visualization) or Gazebo (a ROS-compatible robotics simulator). RViz uses this data to render the robot model accurately, facilitating visualization and debugging. Gazebo can use the robot model to simulate robot behaviors accurately.

Interactive Markers: Robot State Publisher can incorporate interactive markers, which allow users to manipulate and interact with robot components within visualization tools, aiding in intuitive robot control and simulation setup.

ROS Integration and Parameterization: Being a part of ROS, the Robot State Publisher integrates seamlessly into ROS-based robotic systems. It offers various parameters that can be tuned to customize its behavior, such as the rate of publishing, frame names, or specific joint configurations.

Support for ROS Ecosystem: The package aligns with the broader ROS ecosystem, ensuring compatibility and ease of integration with other ROS nodes, tools, and functionalities.

8.2.4 ROS2_Control Package

The ros2_control package in ROS 2 is a framework that aids in the development of robot controllers and hardware interfaces, providing a standardized and modular approach to control robots. It focuses on abstracting the hardware control layer, making it easier to develop control algorithms while ensuring compatibility with various robot hardware.

Key components and functionalities of the ros2_control package include:

Hardware Abstraction Layer (HAL): The package defines an abstraction layer that separates the hardware-specific details from the control algorithms. This HAL allows users to write controller code without being concerned about the underlying hardware interface specifics, promoting code reusability and portability across different robot platforms.

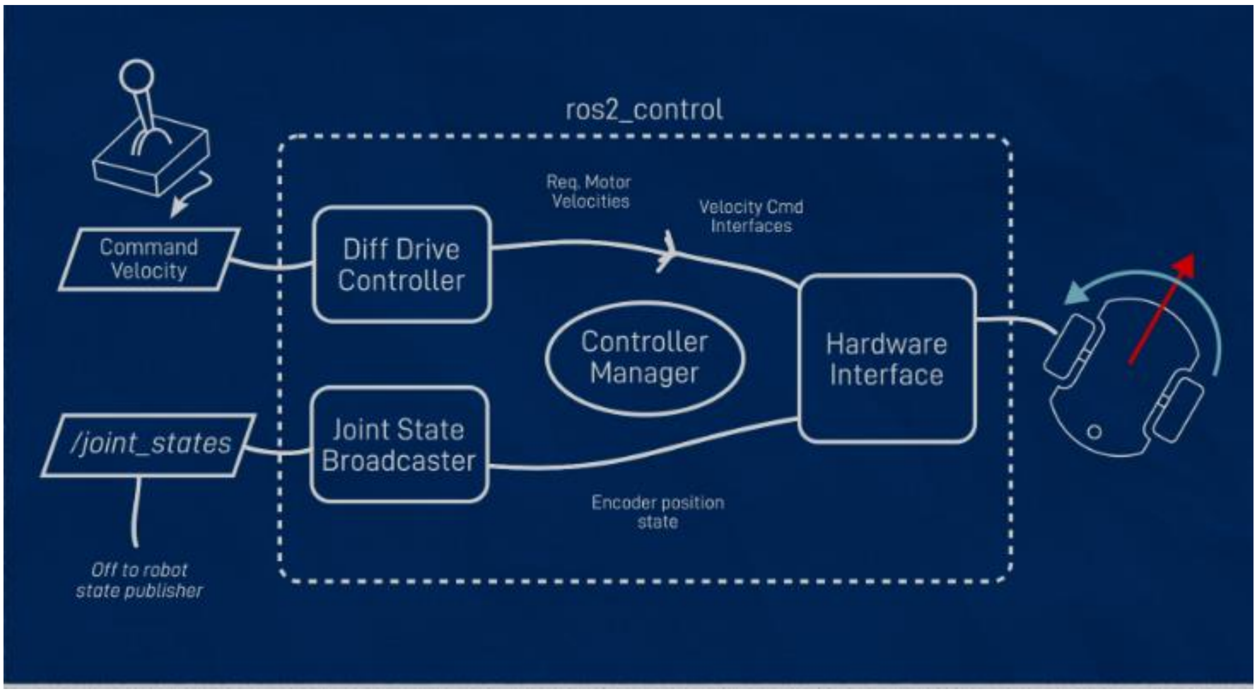


Fig 8.2.4 ros2_control package

Hardware Interfaces: ros2_control provides a set of standard hardware interfaces that define how controllers interact with different hardware components such as joints, sensors, and actuators. Examples include joint position controllers, velocity controllers, and efforts controllers. These interfaces abstract the low-level hardware control, allowing developers to focus on higher-level control strategies.

Controller Manager: The Controller Manager is responsible for loading, managing, and switching between different controllers at runtime. It handles the lifecycle of controllers and their interactions with the hardware interfaces, facilitating seamless transitions between control modes.

Real-time Safe Execution: ros2_control aims to provide real-time capabilities, enabling control loops to execute in predictable and deterministic ways. This is crucial for robots requiring precise and consistent control, especially in scenarios like robotics arms or systems with strict timing requirements.

System Composition: The framework supports the composition of complex robot systems by allowing the combination of various controllers and hardware interfaces. This flexibility enables the control of multi-joint robots or robots with diverse sensors and actuators.

Configuration and Parameterization: Users can configure and parameterize controllers and hardware interfaces through ROS 2 parameters, enabling easy adjustments and tuning of control algorithms.

Integration with ROS 2: ros2_control aligns with the ROS 2 ecosystem, ensuring compatibility with other ROS 2 packages, tools, and middleware. It adheres to ROS 2's design principles, leveraging ROS 2 features for seamless integration.

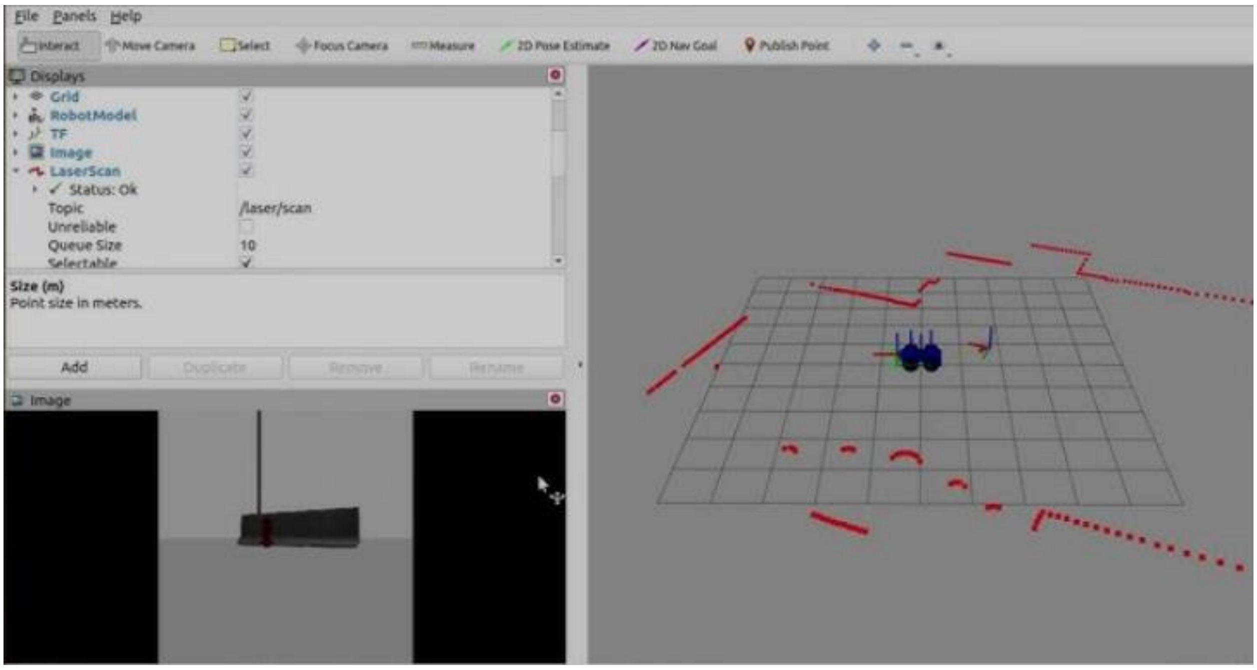
8.2.5 RPLiDAR Package

The RPLIDAR package in ROS is a driver that facilitates communication with RPLIDAR laser scanners, developed by Slamtec. RPLIDAR is a cost-effective and versatile 2D laser range scanner used in various robotic applications for environment perception, mapping, and navigation. The ROS RPLIDAR package provides the necessary interface to integrate RPLIDAR sensors into ROS-based robotic systems.

Key components and functionalities of the RPLIDAR package include:

Driver Implementation: The package consists of a ROS driver that communicates with the RPLIDAR sensor using USB, UART, or other interfaces. It establishes a connection with the RPLIDAR device, reads data from the sensor, and publishes the acquired laser scan data as ROS messages.

ROS Nodes and Topics: The RPLIDAR package typically includes ROS nodes responsible for handling sensor data. These nodes publish laser scan data on ROS topics, allowing other ROS nodes to subscribe to this data for further processing, mapping, localization, or obstacle avoidance.



Grid ✓
RobotModel ✓
TF ✓
Image ✓
LaserScan ✓
✓ Status: Ok
Topic /laser/scan
Unreliable
Queue Size 10
Selectable ✓

Size (m)
Point size in meters.

Fig 8.2.5 RPLiDAR Visualization

Laser Scan Data Processing: The driver processes raw data from the RPLIDAR sensor and converts it into ROS-compatible laser scan messages. These messages contain information about the distance and angle measurements obtained by the laser scanner, crucial for mapping the robot's surroundings.

Visualization Tools: ROS RPLIDAR integration often includes visualization tools such as RViz (ROS Visualization) or other custom visualization nodes. These tools enable users to visualize the laser scan data, creating maps or observing the environment in real-time, aiding in debugging and sensor assessment.

Parameter Configuration: Users can configure various parameters of the RPLIDAR sensor, such as rotation speed, scanning frequency, or data acquisition settings, through ROS parameters, allowing customization based on specific robot requirements.

Compatibility and Integration: The package ensures compatibility with the ROS ecosystem, making it easy to integrate RPLIDAR sensors into ROS-based robotic platforms and applications. It adheres to ROS conventions, facilitating seamless communication and interoperability with other ROS nodes and packages.

Community Support and Documentation: The ROS RPLIDAR package benefits from community contributions, support, and extensive documentation, providing users with resources, tutorials, and troubleshooting assistance.

8.2.6 Navigation2 Package

Navigation2 is a ROS (Robot Operating System) package that builds on the capabilities of the original Navigation Stack in ROS. It is designed to provide a comprehensive and modular navigation system for mobile robots.

ROS 2 Compatibility: Navigation2 is specifically designed for ROS 2, the latest version of the Robot Operating System. ROS 2 introduces improvements over ROS 1, including enhanced real-time capabilities and better support for various platforms.

Modular Architecture: Navigation2 follows a modular architecture, allowing users to choose and configure different components based on the specific requirements of their robot and environment. This modular approach makes it flexible and adaptable to various scenarios.

Map Server: The package includes a map server component that handles the management and distribution of maps. Maps are crucial for robot navigation as they provide a representation of the environment.

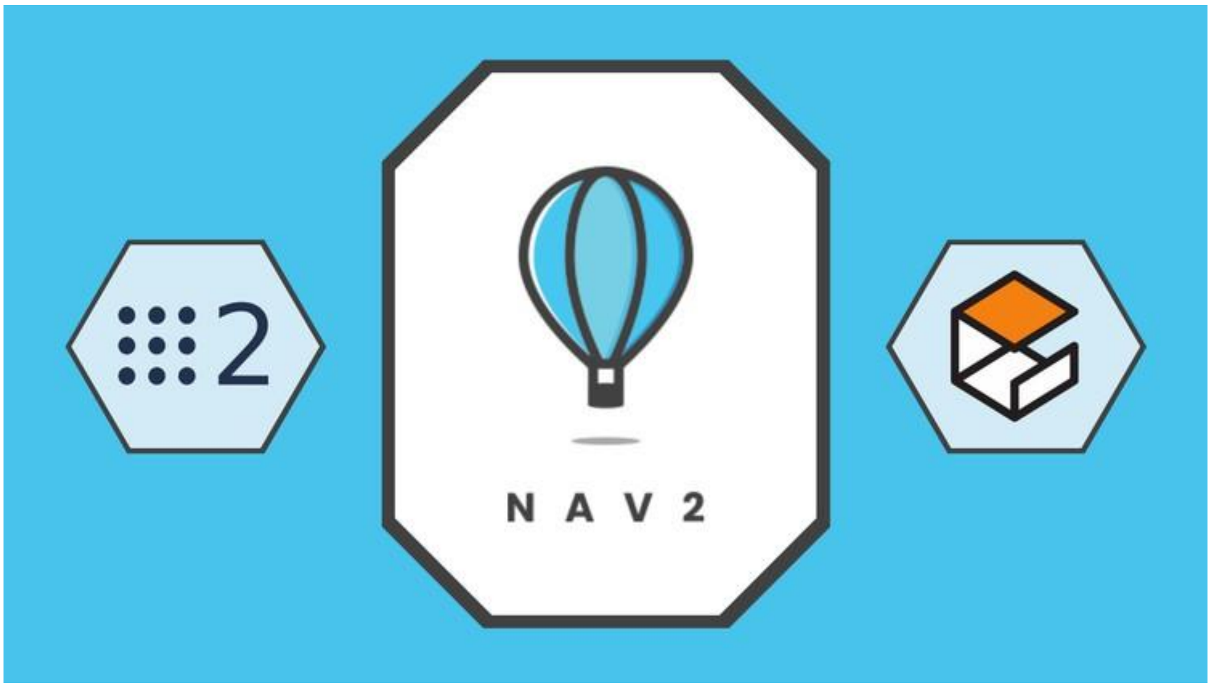


Fig 8.2.6 Navigation2 Package

Localization: Navigation2 incorporates localization techniques to determine the robot's position within the environment. Common methods include AMCL (Adaptive Monte Carlo Localization) and other localization algorithms.

Path Planning: Path planning is a critical component that helps the robot navigate from its current location to a specified goal while avoiding obstacles. Navigation2 supports various path planning algorithms to optimize robot trajectories.

Obstacle Avoidance: The package includes obstacle avoidance mechanisms to ensure that the robot can navigate around or through obstacles in its path. This often involves the use of sensor data, such as laser scans or depth information.

Global and Local Planners: Navigation2 typically employs both global and local planners. The global planner determines a high-level path from the start to the goal, while the local planner adjusts the robot's trajectory to navigate through the environment at a finer level.

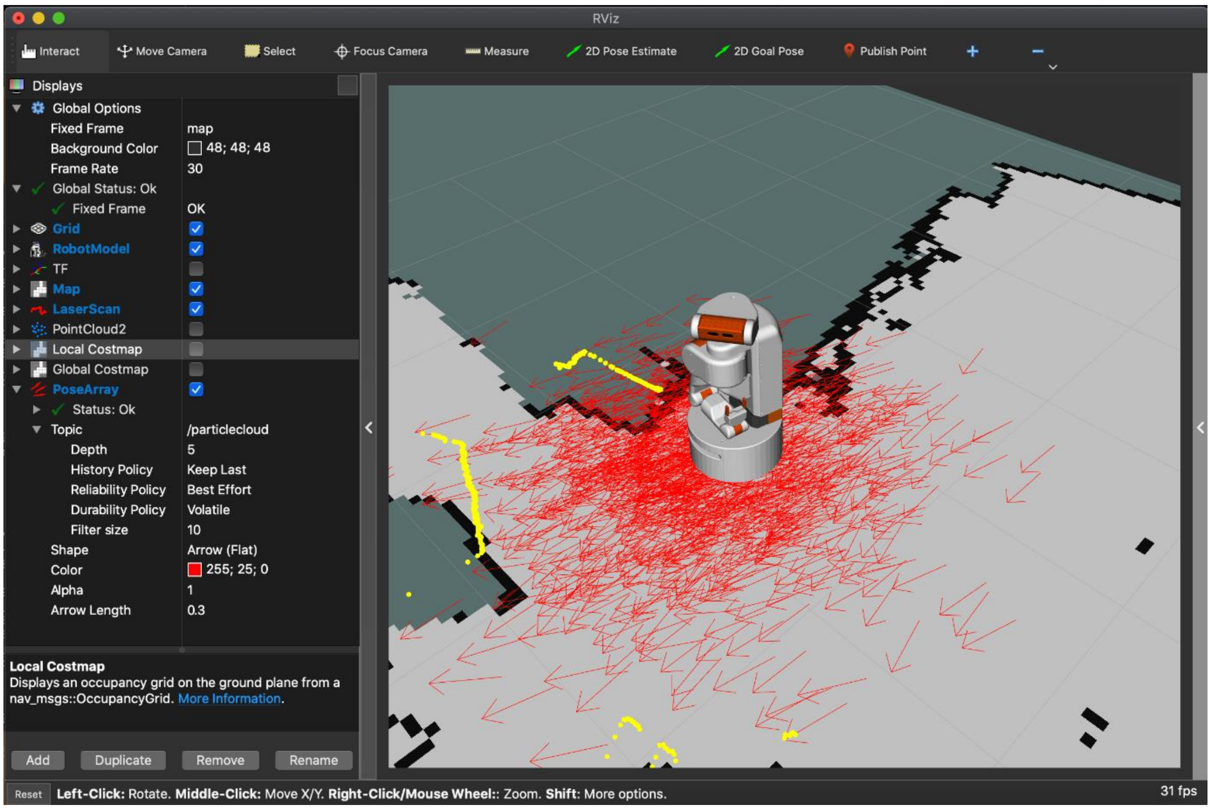
Costmap: Costmaps represent the traversability of different areas in the environment. Navigation2 utilizes costmaps to help the robot make informed decisions about its path by assigning costs to different regions based on obstacles and terrain characteristics.

Integration with ROS Tools: Navigation2 integrates with various ROS tools, allowing users to visualize and interact with navigation-related data using tools such as Rviz (ROS visualization) and other debugging tools.

Community Support: Being part of the ROS ecosystem, Navigation2 benefits from a supportive community that contributes to its development, provides documentation, and shares knowledge and experiences.

8.2.7 Cartographer

Cartographer, initially developed by Google, is a powerful simultaneous localization and mapping (SLAM) package that has been adapted for ROS 2, the latest version of the Robot Operating System. This integration allows roboticists to leverage the capabilities of Cartographer within the context of ROS 2, taking advantage of the system's enhanced real-time capabilities and robust middleware. Cartographer excels in real-time SLAM by processing sensor data from lidar and IMU sensors to construct detailed 2D or 3D maps of the robot's environment. Lidar scans contribute to accurate pose estimation, while IMU data aids in scenarios where lidar information alone may be insufficient, such as during temporary signal loss. The ROS 2 implementation of Cartographer typically consists of ROS 2 nodes responsible for processing sensor data, mapping, and communicating with other components in a robot's software architecture. With support for both 2D and 3D mapping, Cartographer in ROS 2 becomes a valuable tool for robots navigating diverse environments.



Global Options	
Fixed Frame	map
Background Color	48; 48; 48
Frame Rate	30
Global Status: Ok	
Fixed Frame	OK
Grid	✓
RobotModel	✓
TF	
Map	✓
LaserScan	✓
PointCloud2	
Local Costmap	
Global Costmap	
PoseArray	✓
Status: Ok	
Topic	/particlecloud
Depth	5
History Policy	Keep Last
Reliability Policy	Best Effort
Durability Policy	Volatile
Filter size	10
Shape	Arrow (Flat)
Color	🟥 255; 25; 0
Alpha	1
Arrow Length	0.3

Local Costmap
Displays an occupancy grid on the ground plane from a nav_msgs::OccupancyGrid. More Information.

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift:** More options.

Fig 8.2.7 Cartographer

Users can configure and fine-tune Cartographer parameters to adapt the SLAM algorithm to specific sensor characteristics, robot dynamics, and environmental conditions. Additionally, Cartographer seamlessly integrates with the ROS Navigation Stack, allowing generated maps to be utilized for path planning and navigation. As part of the ROS ecosystem, Cartographer benefits from extensive documentation and community support, providing users with valuable resources such as guides, tutorials, and forums for addressing installation, configuration, and troubleshooting challenges. Given the dynamic nature of software development, staying updated with the latest documentation and community discussions ensures users can harness the full potential of Cartographer in ROS 2 for their robotic applications.

CHAPTER 9

MODELING

9.1 CHASSIS DESIGN

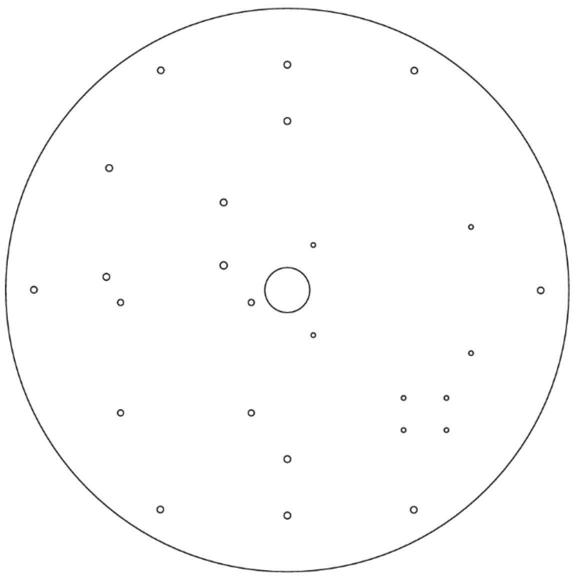


Fig 9.1.1 Top Chassis

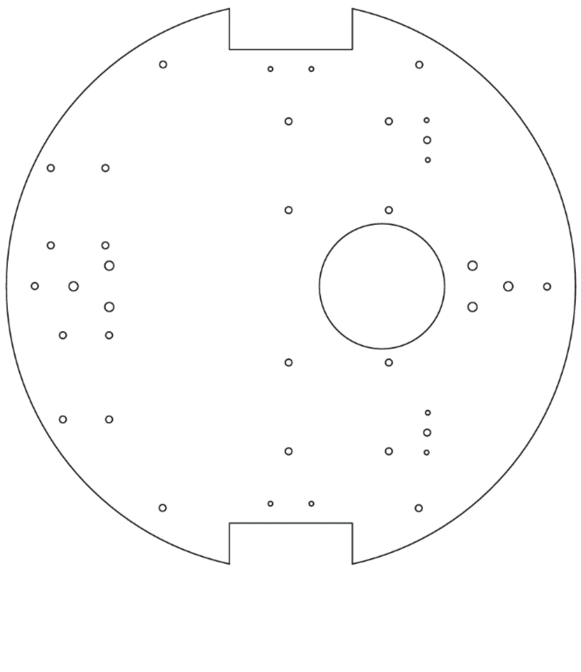
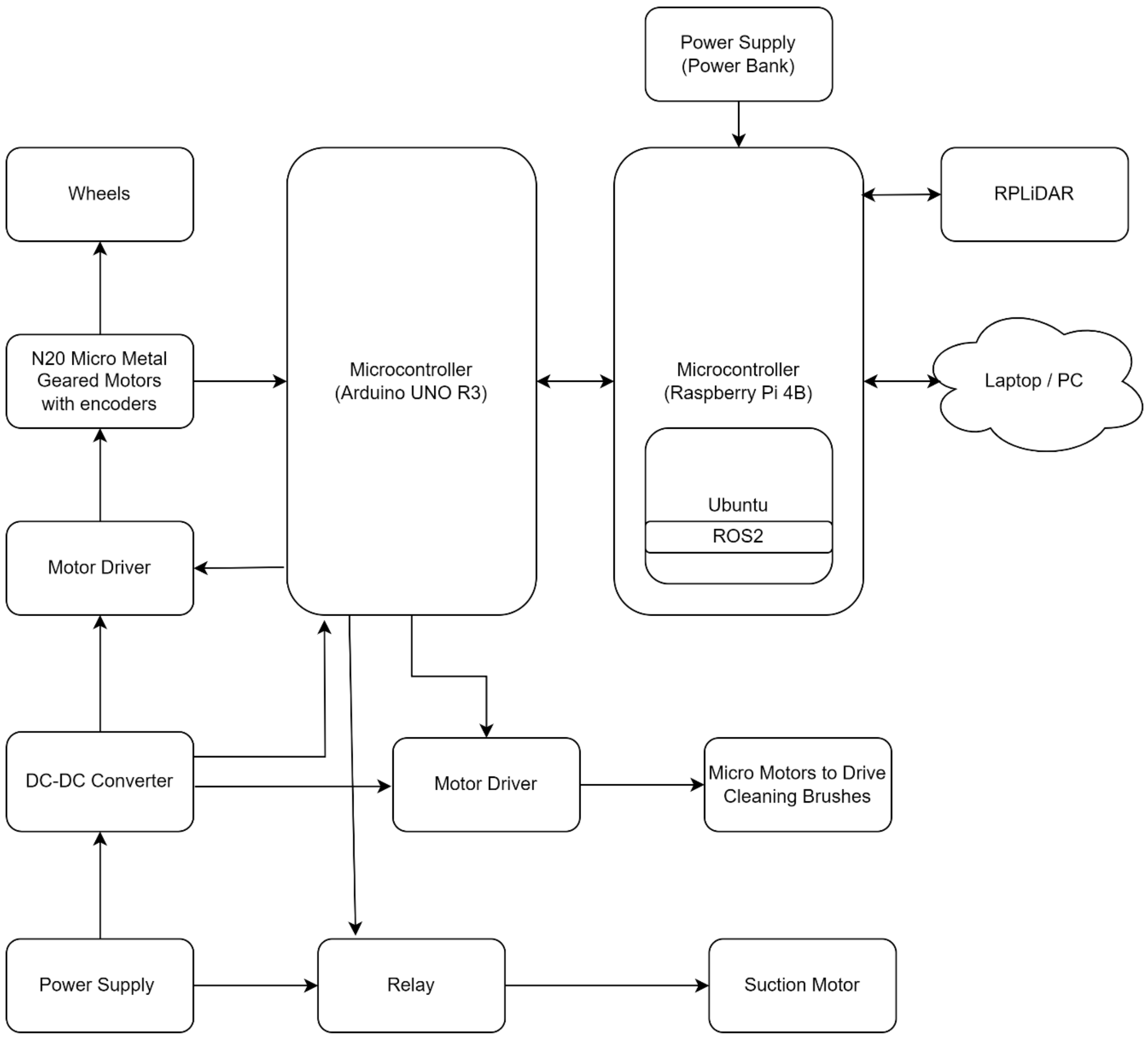


Fig 9.1.2 Bottom Chassis

The above chassis parts were fabricated using Laser cutting machine.

9.2 BLOCK DIAGRAM



Power Supply (Power Bank)
Wheels
RPLiDAR
N20 Micro Metal Geared Motors with encoders
Microcontroller (Arduino UNO R3)
Microcontroller (Raspberry Pi 4B)
Laptop / PC
Ubuntu
ROS2
Motor Driver
DC-DC Converter
Motor Driver
Micro Motors to Drive Cleaning Brushes
Power Supply
Relay
Suction Motor

Fig 9.2 Block Diagram

9.3 CIRCUIT DIAGRAM

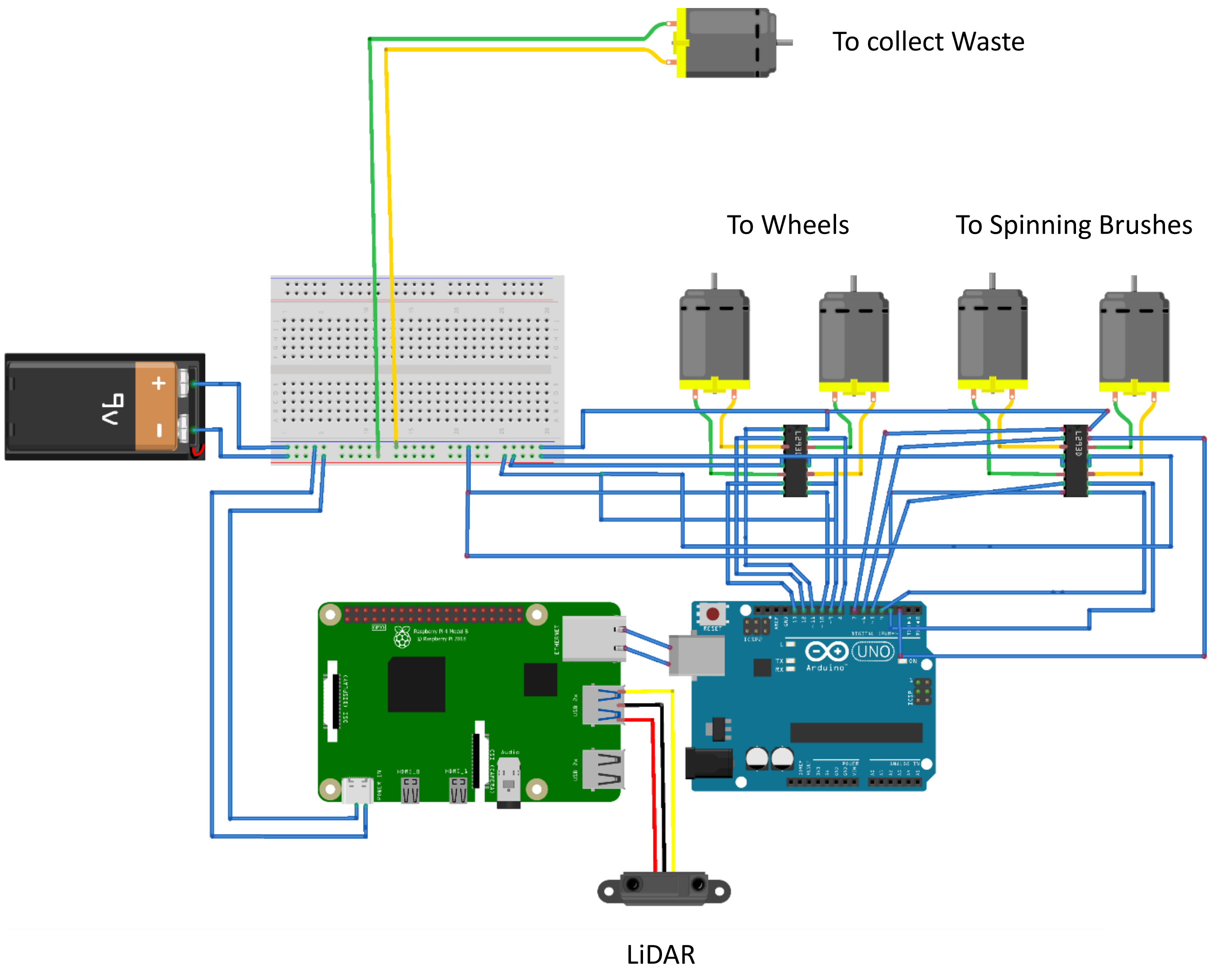


Fig 9.3 Circuit Diagram

9.4 WORKFLOW

The working mechanism of the Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR involves a combination of hardware components, software algorithms, and sensor technologies to achieve autonomous cleaning of indoor environments.

Initialization and Setup

The robot begins by initializing its components, including the robot chassis, LiDAR sensors, actuators, and the vacuum system. This involves ensuring proper power supply, establishing communication links, and preparing the system for operation.

Sensor Integration

LiDAR sensors, mounted on the robot chassis, continuously scan the surroundings in a 360-degree field of view. These sensors provide detailed information about the environment, detecting obstacles, walls, and other features.

ROS2 Initialization

The Robot Operating System 2 (ROS2) is initiated, acting as the core software platform. ROS2 nodes are activated, enabling communication and coordination among different components of the robot.

Mapping and Localization

Using data from the LiDAR sensors, the robot generates accurate maps of its surroundings. Simultaneous Localization and Mapping (SLAM) algorithms, implemented within ROS2, allow the robot to simultaneously localize itself within these maps as it navigates the environment.

Path Planning

Path planning algorithms are developed to determine optimal cleaning paths for the robot. These algorithms consider the generated maps and aim to cover the entire cleaning area efficiently, avoiding unnecessary overlap.

Obstacle Avoidance

The robot utilizes LiDAR data for obstacle detection and avoidance. When encountering obstacles in its path, the robot's path planning algorithms dynamically adjust to navigate around them, ensuring the robot can clean effectively while avoiding collisions.

User-Friendly Interface

A user-friendly interface, such as a mobile app or web-based dashboard, is provided for users to control and monitor the robot. Users can initiate or stop cleaning sessions, schedule tasks, and receive real-time updates on the robot's status and progress through this interface.

Autonomous Cleaning Operation

With all systems initialized and configured, the robot autonomously navigates the indoor environment based on the generated maps. It follows the planned cleaning paths, adapting to real-time changes in its surroundings using LiDAR data.

Continuous Optimization

The robot continuously optimizes its algorithms and sensor integration during operation. This iterative process enhances cleaning efficiency, adaptability to different environments, and overall system performance.

Safety Measures

Safety features, including collision avoidance algorithms and emergency stop mechanisms, are implemented to ensure safe operation. These features enhance the reliability of the robot, minimizing risks during cleaning tasks.

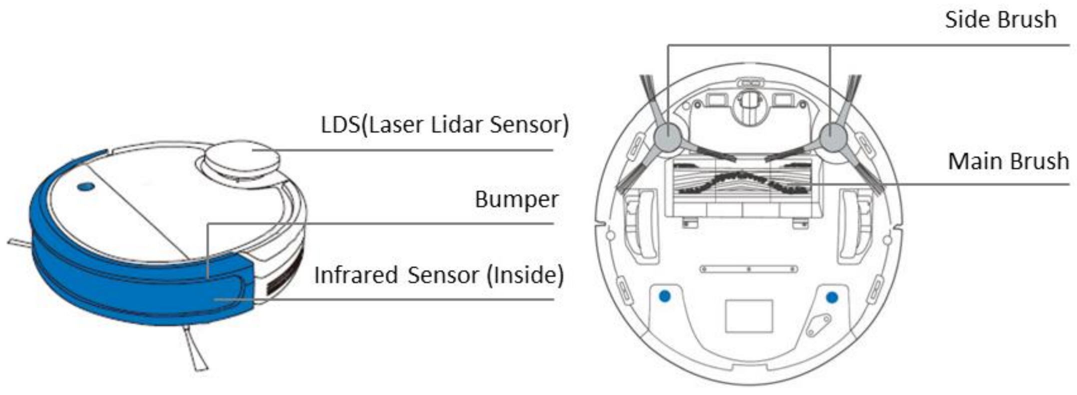


Fig 9.4 AMVCR (Autonomous Mobile Vacuum Cleaning Robot)

Throughout the cleaning process, the robot's status is monitored in real-time through the user interface. This allows users to intervene if necessary and provides insights into the robot's performance.

9.5 APPLICATIONS

The Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR has versatile applications across various settings, providing efficient and autonomous cleaning solutions.

Residential Cleaning

The robot is well-suited for cleaning tasks in homes, apartments, and residential spaces. Its ability to autonomously navigate and clean indoor environments makes it a valuable tool for homeowners seeking a convenient and efficient cleaning solution.

Office Spaces

In office environments, the robot can autonomously clean conference rooms, hallways, and workspaces, contributing to a cleaner and healthier work environment. Its autonomous operation allows it to perform cleaning tasks during non-working hours without human intervention.

Commercial Buildings

The robot can be deployed in commercial buildings, including shopping malls, hotels, and restaurants, to autonomously clean large indoor spaces. Its efficiency and adaptability make it suitable for maintaining cleanliness in high-traffic areas.

Educational Institutions

Autonomous cleaning robots can be employed in schools, colleges, and universities to clean classrooms, corridors, and other common areas.

Healthcare Facilities

In healthcare settings such as hospitals and clinics, maintaining cleanliness is crucial. The robot can navigate through different sections of healthcare facilities, including patient rooms and corridors, contributing to hygiene maintenance.

Laboratories and Clean Rooms

The robot's autonomous cleaning capabilities make it suitable for use in controlled environments like laboratories and clean rooms. It can help minimize contamination risks by providing consistent and thorough cleaning without manual intervention.

Industrial Spaces

In industrial settings, the robot can be utilized for cleaning manufacturing floors, warehouses, and assembly lines. Its autonomous operation allows for efficient cleaning without disrupting ongoing industrial processes.

Retail Environments

Retail spaces, including supermarkets and shopping centers, can benefit from the robot's ability to autonomously clean aisles and common areas. This enhances the overall cleanliness and presentation of the retail environment.

Smart Homes and IoT Integration

The robot's compatibility with ROS2 allows for integration with smart home systems and IoT devices. Users can potentially schedule cleaning sessions, monitor the robot's status remotely, and integrate it into broader smart home automation setups.

Educational and Research Purposes

The project serves as an educational tool for robotics and automation enthusiasts, students, and researchers. It provides hands-on experience in developing and implementing autonomous systems, making it a valuable resource for learning and experimentation.

The applications of the Autonomous Mobile Vacuum Cleaning Robot extend to any indoor environment where autonomous cleaning is desirable. Its adaptability, efficiency, and user-friendly features make it a versatile solution for enhancing cleanliness and reducing the manual effort required for routine cleaning tasks.

CHAPTER 10

FABRICATION

10.1 3D PRINTED PARTS

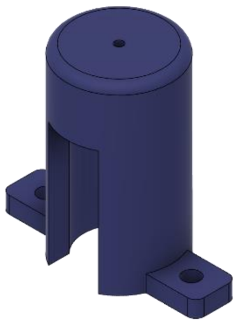


Fig 10.1.1 Cleaning Motor Clamp

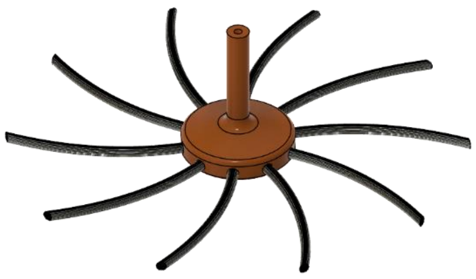


Fig 10.1.2 Cleaning Brushes

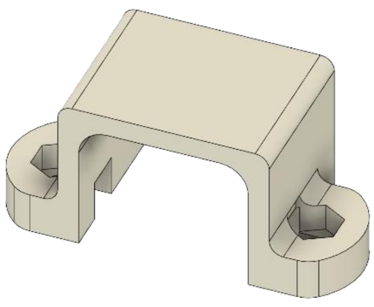


Fig 10.1.3 N20 Motor Clamp

10.2 ASSEMBLY

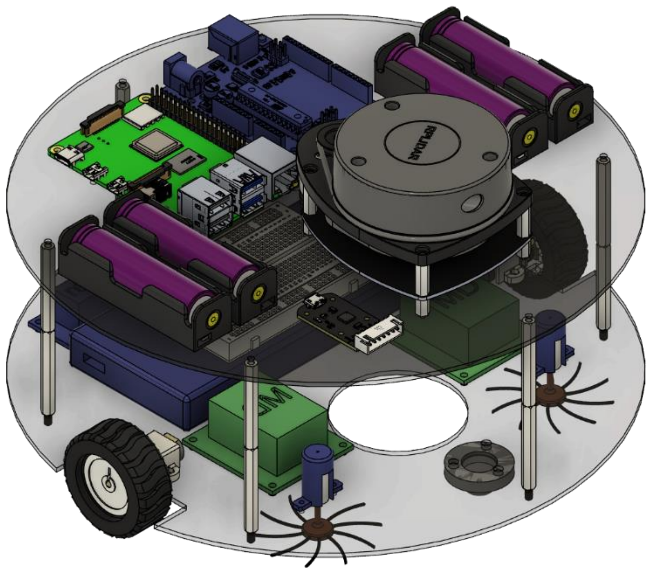


Fig 10.2.1 Assembly – Isometric View

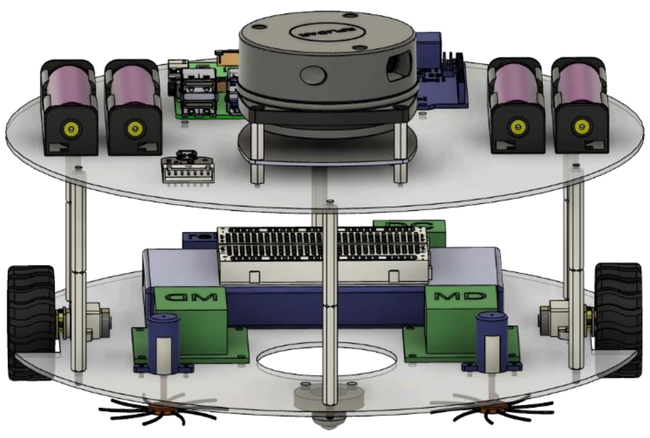


Fig 10.2.2 Assembly – Front View

10.3 IMPLEMENTATION

S. No	Component Name	Nos	Specifications
1	Raspberry Pi 4B	1	2GB
2	SD Card (C10)	1	32GB
3	Motor for Wheels	2	150RPM
4	Motor Driver (L293D For Wheels)	1	2 Motors
5	Wheels	2	44mm Dia, 3mm Shaft Dia
6	Arduino UNO	1	UNO R3
7	RPLiDAR A1	1	6 Meters Range
8	Batteries Li-Ion with Charger	4	2200MAh
9	Power Bank	1	20000MAh
10	Chassis	2	Acrylic Sheet 2mm thickness
11	Standoff	12	40mm, M3 Screws
12	Castor Wheel	2	With 2 Extra Nuts
13	Jumper Wires	As Req	Male To Male, Female to Male, Female to Female
14	DC To DC Converter	1	3.3V 5V 12V
15	HDMI(Fem) To Micro HDMI(Male)	1	Converter
16	Screws & Nuts	As Req	M3, M2, M1
17	Motor for Vacuum, Cleaners with Propellers	1	High RPM
18	Cleaning Brushes	2	3D Printed Part
19	Drivers for Cleaning Motors	1	2 Motors
20	Relay	1	5V 30V@10A

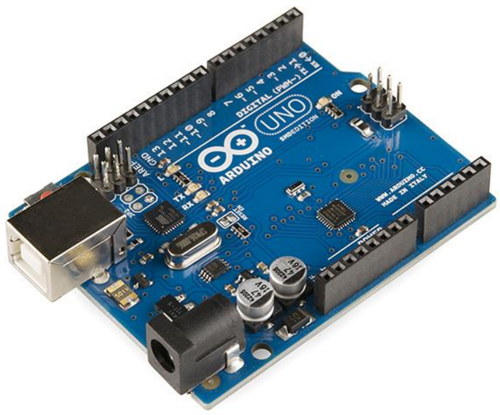


Fig 10.3.1 Arduino UNO R3

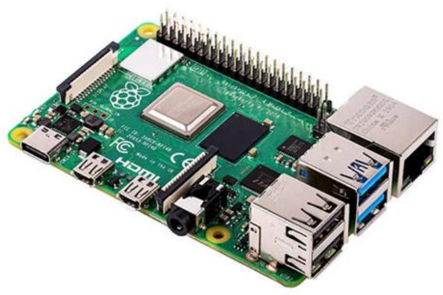


Fig 10.3.2 Raspberry Pi 4B

Fig 10.3.3 RP LiDAR

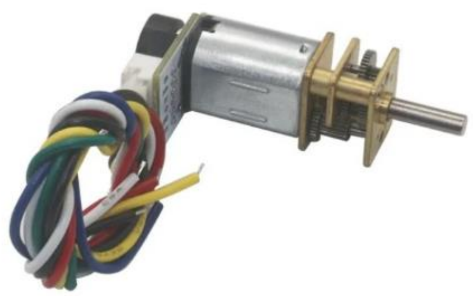
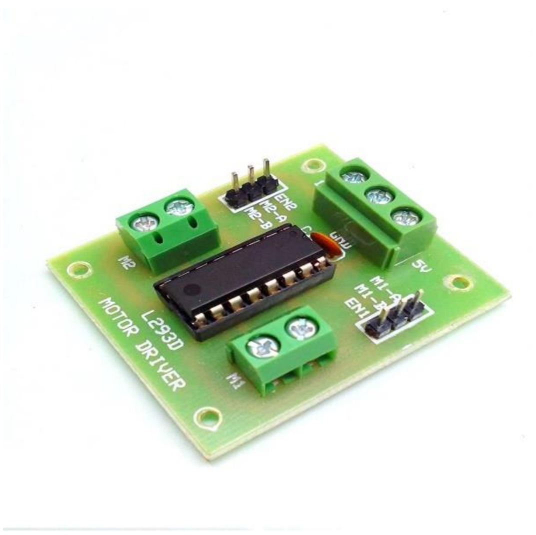


Fig 10.3.4 N20 Gear Motor



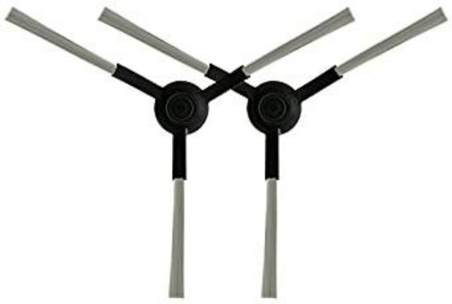
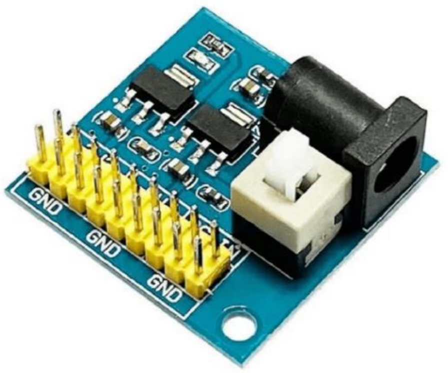


Fig 10.3.5 L293D Motor Driver

Fig 10.3.6 Cleaning Brushes



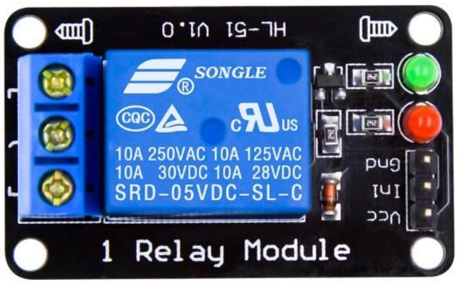


Fig 10.3.7 DC-DC Converter

Fig 10.3.8 Relay Module

Fig 10.3.9 Li-Ion Battery

Fig 10.3.10 Standoff

Fig 10.3.11 Wheels

Fig 10.3.12 Ball Castor Wheel

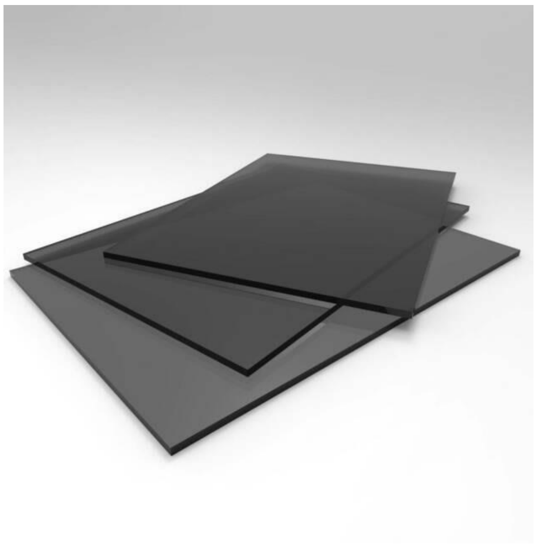


Fig 10.3.13 Acrylic Sheet

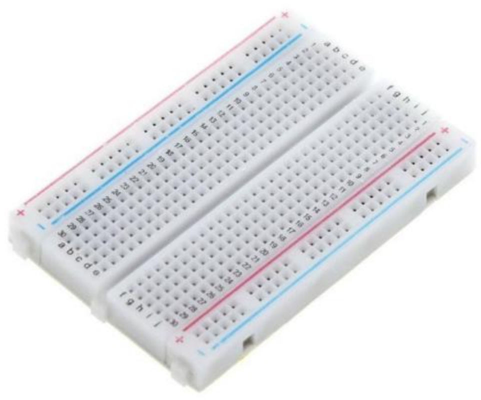


Fig 10.3.14 Breadboard

Fig 10.3.15 Jumper Wires

Fig 10.3.16 Mini High-Speed Motor

CHAPTER 11

SOFTWARE IMPLEMENTATION

11.1 ARDUINO PROGRAMMING

ENCODER PULSES BY DEFAULT INTERRUPT PINS

```
const int encoderPinA = 2; // Replace with your chosen pins
const int encoderPinB = 3;
 // Volatile as it's modified within an ISR
volatile int encoderCount_M1 = 0;
volatile int encoderDirection_M1 = 0;
int previousStateA_M1 = 0;
int previousStateB_M1 = 0;
void setup() {
pinMode(encoderPinA, INPUT);
pinMode(encoderPinB, INPUT);
attachInterrupt(digitalPinToInterrupt(encoderPinA), updateEncoder, CHANGE);
attachInterrupt(digitalPinToInterrupt(encoderPinB), updateEncoder, CHANGE);
Serial.begin(9600);
}
void loop() {
  // Your main code here
}
void updateEncoder() {
```

```
// Read the state of the encoder pins
int currentStateA_M1 = digitalRead(encoderPinA);
int currentStateB_M1 = digitalRead(encoderPinB);
if (currentStateA_M1 != previousStateA_M1)
{
      if (currentStateA_M1 == currentStateB_M1)
      {
      encoderCount_M1++;
      encoderDirection_M1 = 1;  // Clockwise rotation
      }
      else
      {
      encoderCount_M1--;
      encoderDirection_M1 = -1;  // Counter-clockwise rotation
      }
}
else if (currentStateB_M1 != previousStateB_M1){
      if (currentStateA_M1 == currentStateB_M1)
      {
      encoderCount_M1--;
      encoderDirection_M1 = -1;  // Counter-clockwise rotation
      }
```

```
        else
        {
        encoderCount_M1++;
        encoderDirection_M1 = 1;  // Clockwise rotation
        }
  }
  // Update previous state variables
  previousStateA_M1 = currentStateA_M1;
  previousStateB_M1 = currentStateB_M1;
Serial.println(encoderCount_M1);
}
```

ENCODER PULSES BY CHANGE INTERRUPTS

```
volatile int M1_C1 = 6;
volatile int M1_C2 = 7;
volatile int encoderCount_M1 = 0;
volatile int encoderDirection_M1 = 0;
int previousStateA_M1 = 0;
int previousStateB_M1 = 0;
void setup() {
Serial.begin(9600);
pinMode(M1_C1,INPUT_PULLUP);
pinMode(M1_C2,INPUT_PULLUP);
```

```
  // Enable PCIE2 Bit3 = 1 (Port D)
  PCICR |= B00000100;
  // Select PCINT23 Bit7 = 1 (Pin D7)
  PCMSK2 |= B11000000;
}
void loop() {
  // No code in Loop
}
ISR (PCINT2_vect)
{
  // Serial.print("Channel 1 = ");
  // Serial.println(digitalRead(M1_C1));
  // Serial.print("Channel 2 = ");
  // Serial.println(digitalRead(M1_C2));

  int currentStateA_M1 = digitalRead(M1_C1);
  int currentStateB_M1 = digitalRead(M1_C2);
  if (currentStateA_M1 != previousStateA_M1)
  {
        if (currentStateA_M1 == currentStateB_M1)
        {
        encoderCount_M1++;
        encoderDirection_M1 = 1;  // Clockwise rotation
        }
        else
        {
        encoderCount_M1--;
        encoderDirection_M1 = -1;  // Counter-clockwise rotation
        }
}
else if (currentStateB_M1 != previousStateB_M1){
        if (currentStateA_M1 == currentStateB_M1)
        {
        encoderCount_M1--;
        encoderDirection_M1 = -1;  // Counter-clockwise rotation
        }
        else
        {
        encoderCount_M1++;
        encoderDirection_M1 = 1;  // Clockwise rotation
        }
}
// Update previous state variables
previousStateA_M1 = currentStateA_M1;
```

```
  previousStateB_M1 = currentStateB_M1;
Serial.println(encoderCount_M1);
}
```

11.2 ROS2 PROGRAMMING

NAVIGATION PARAMETERS FILE

```
controller_server:
ros__parameters:
controller_frequency: 20.0
min_x_velocity_threshold: 0.001
min_y_velocity_threshold: 0.5
min_theta_velocity_threshold: 0.001
failure_tolerance: 0.3
progress_checker_plugins: ["progress_checker"]
goal_checker_plugins: ["general_goal_checker"] # "precise_goal_checker"
controller_plugins: ["FollowPath"]
```

CHAPTER 12

RESULT & DISCUSSION

Result

The deployment and testing of the Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR yielded promising results. The robot demonstrated effective autonomous navigation, obstacle avoidance, and cleaning capabilities in various indoor environments. LiDAR sensors successfully mapped and localized the robot, allowing it to traverse the space while avoiding obstacles. The path planning algorithms efficiently covered cleaning areas, contributing to thorough and systematic cleaning. Real-time monitoring through the user-friendly interface provided users with control and insights into the robot's progress. The robot's performance was evaluated in both simulated and real-world scenarios, validating its reliability and adaptability.

Discussion

The successful implementation of ROS2 and LiDAR technology in this project significantly enhanced the robot's autonomy and efficiency. The combination of accurate mapping, simultaneous localization, and obstacle avoidance ensured smooth operation and effective cleaning. The user-friendly interface facilitated seamless control and monitoring, making the robot accessible to users with varying technical backgrounds. The project's modular design allowed for easy integration of additional features and sensors, opening avenues for further enhancements.

CHAPTER 13

CONCLUSION

In conclusion, the Autonomous Mobile Vacuum Cleaning Robot with ROS2 and LiDAR presents a robust and practical solution for autonomous indoor cleaning. The integration of cutting-edge technologies, including ROS2 middleware and LiDAR sensors, resulted in a versatile robot capable of navigating and cleaning various indoor environments autonomously.

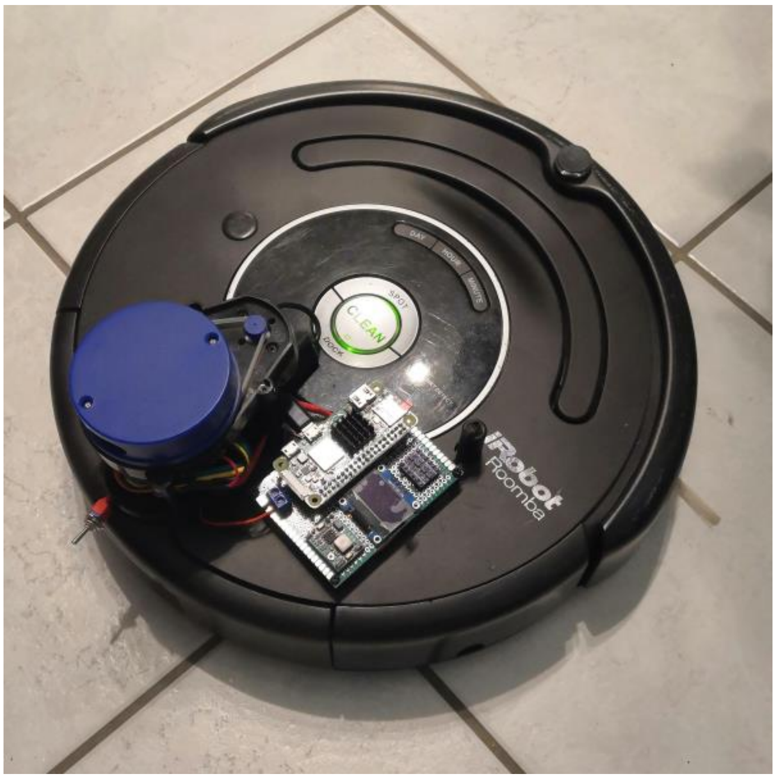


Fig 13.1 AMVCR

The project successfully addressed the objectives of efficient cleaning, user-friendly control, and adaptability to different spaces. The comprehensive testing and positive results validate the feasibility and effectiveness of the robot in real-world applications.

CHAPTER 14

FUTUREWORK

Advanced Mapping Techniques: Explore and implement advanced mapping techniques to enhance the accuracy and resolution of generated maps, allowing for more precise navigation and cleaning.

Machine Learning Integration: Investigate the integration of machine learning algorithms for improved decision-making, allowing the robot to adapt its cleaning strategies based on learned patterns and user preferences.

Localization Enhancements: Research and implement enhancements to localization techniques, potentially incorporating additional sensor modalities to improve accuracy in dynamic environments.

Human-Robot Interaction: Integrate features for improved human-robot interaction, including natural language processing for voice commands and enhanced communication interfaces.

External Sensor Integration: Explore the integration of external sensors, such as cameras and environmental sensors, to expand the robot's perception capabilities for a more comprehensive understanding of its surroundings.

Real-Time Adaptive Planning: Develop algorithms for real-time adaptive planning that can dynamically adjust cleaning paths based on changes in the environment or user-defined preferences.

These future directions aim to further enhance the capabilities, intelligence, and applicability of the Autonomous Mobile Vacuum Cleaning Robot, making it even more versatile and adaptable to evolving user needs and technological advancements.

CHAPTER 15

REFERENCES

[1] Nogendra Kumar Sahu, Nitesh Kumar Sharma, M. R. Khan, Deepesh Kumar Gautam,“ Comparative Study on Floor Cleaner”, Journal of Pure Applied and Industrial Physics, ISSN 2229-7596, Vol.8(12),233-236, December 2018.

[2] Neel Shaileshbhai Desai, “A Survey on Automatic Vacuum Cleaner for Commercial Places”, International Journal of Advanced Research in Electrical,Electronics and Instrumentation Engineering, ISSN (Print) : 2320 – 3765, Vol. 6, Issue 2, February 2017.

[3] Manya Jain, Pankaj Singh Rawat, Jyoti Morbale,” Automatic Floor Cleaner”, International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056 Volume: 04 Issue: 04 Apr -2017

[4] Manreet Kaur, Mohali Preeti Abrol,” Design and Development of Floor Cleaner Robot (Automatic and Manual)”, International Journal of Computer Applications (0975 – 8887) Volume 97– No.19, July 2014.

[5] PriyaShukla and Simmy S.L. “Design and Inspection of cleaning robot”, Issued volume 3,Issued 6 sept 2014.

[6] S Monika, K Aruna Manjusha, S V S Prasad, B.Naresh,” Design and Implementation of Smart Floor Cleaning Robot using Android App”, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-4S2 March, 2019

[7] Satyinder Singh, Deviyagupta , Shikha Sharma, Mansi Sharma , Neetika Sharma, “Robotic Vacuum Cleaner”, Volume 7, Issued 2017.

[8] C.R.Balamurugan, P.Kirubha, S.ArunKanna, E.R.Hariprasath, C.Anupriya, “Bluetooth Based Automatic Floor Cleaning System”, volume 11, issued 2018.

CHAPTER 16

BIBLIOGRAPHY

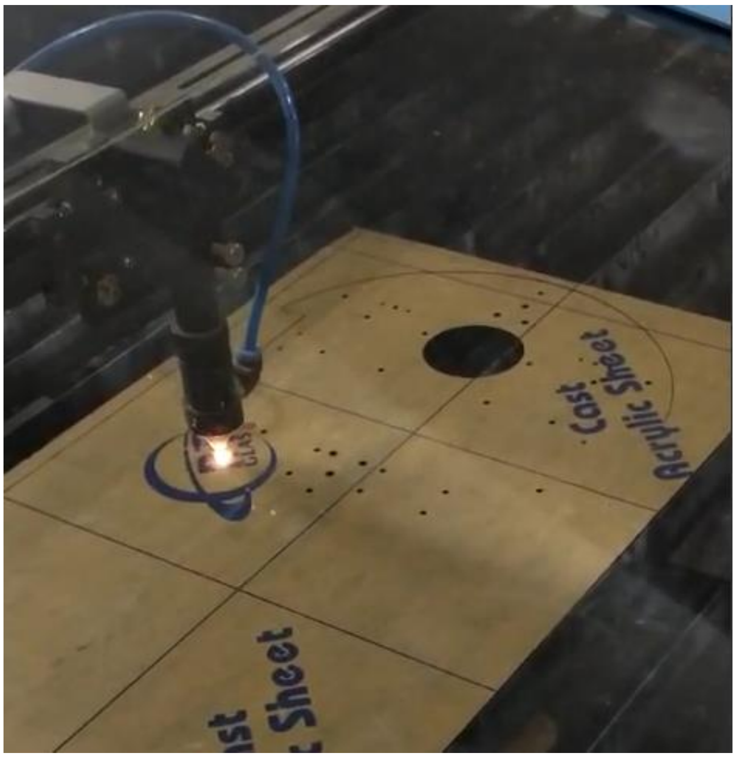


Fig 16.1 Laser Cutting

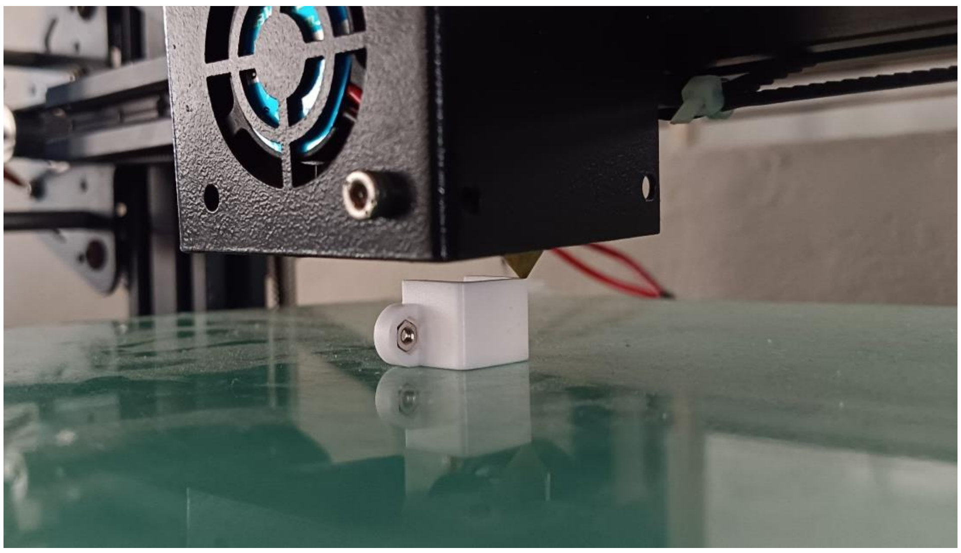


Fig 16.2 3D Printing