الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
المدرسة الوطنية العليا للتكنولوجيات المتقدمة
Ecole Nationale Supérieure des Technologies Avancées

Department of Electrical Engineering and Industrial Computing

Final Year Project to Obtain the Diploma of Engineering

- Field -
Automatic

- Specialty -
Automation and Industrial Computing

- Subject -

Design and construction of a mobile base for testing different control techniques

Realized by
AOUFI Ali
BELGACEM Mohamed Lyes

Presented publicly, the 23/06/2025

Members of The Jury:

First Last Name	University	Grade	Role
Karima REBAI	ENSTA	MCA	President
Islem BOUCHACHI	ENSTA	MCA	Supervisor
Samir KHELOUAT	ENSTA	MCB	Supervisor
Khadidja ZELLAT	ENSTA	MCA	Examiner
Zahira OUSAADI	ENSTA	MAA	Examiner

Academic year 2024–2025

Dedication

*First of all, I would like to thank **God** for granting me the strength and guidance throughout my educational journey, which has culminated in this project.*

*I also dedicate this work to **my father, my mother, my grandmother, my brother, and my sisters**, my unwavering pillars of support. Their love, sacrifices, and constant encouragement have been my driving force throughout this journey, inspiring me to persevere until the end.*

*I would also like to express my heartfelt thanks to **my partner in this work, Lyes**, for his dedication, enthusiasm, and strong teamwork throughout the entire project. This achievement would not have been possible without our shared commitment.*

Finally, to all my friends and family members who care about me thank you for your continuous support and encouragement.

I love you all. You are appreciated.

Ali

*Above all, I thank **God** for granting me the perseverance and strength to complete this final year project.*

*I warmly dedicate this work to **my father, my mother, my brother, and my sister** the foundation of my determination. Their encouragement and sacrifices have been essential to my personal and academic journey.*

*I extend my sincere appreciation to **my project partner, Ali**, whose collaboration, support, and shared effort have been instrumental to our success. It was a true pleasure working side by side.*

To all my friends and loved ones thank you for believing in me and be beside me to the end.

You mean the world to me.

Lyes

Acknowledgments

We would like to express our deepest gratitude to our supervisors, **Mr. Islem BOUCHACHI** and **Mr. Samir KHELOUAT**, for their exceptional mentorship, unwavering support, and insightful guidance throughout every stage of this project. Their dedication, availability, and constant encouragement have been fundamental to our academic and personal development. Through their constructive feedback and high standards, we have learned valuable lessons that will accompany us well beyond the completion of this work. It has been a privilege to benefit from their experience and commitment.

We are also grateful to all the jury members for their careful evaluation and constructive feedback, which have contributed to the enrichment and improvement of this work.

We would also like to extend our heartfelt thanks to our families and friends for their steadfast support, patience, and encouragement throughout this journey. Their belief in us has been a constant source of motivation and strength. Finally, we wish to express our sincere appreciation to all those who, in one way or another, have contributed to the realization of this project. Whether through technical assistance, moral support, or simple acts of kindness, your contributions have been truly invaluable to us.

الملخص

يعرض هذا العمل تصميم وتطوير قاعدة روبوتية متنقلة متعددة الاستخدامات، تهدف إلى اختبار وتقييم تقنيات التحكم المختلفة. تم تصميم هذا النظام ليكون مرنًا وقابلاً للتعديل، مما يسمح بتطبيق استراتيجيات التحكم مثل DIP وخوارزميات الملاحة وتقنية CMS في بيئات واقعية. يشمل العمل التصميم الميكانيكي، والدمج الإلكتروني، وتطوير خوارزميات تحكم مدمجة لضمان الوظيفة والموثوقية والمرونة. تهدف هذه المنصة إلى دعم البحث والتعليم من خلال توفير أداة عملية للتجريب وتحليل أداء أنظمة التحكم المتقدمة في الروبوتات المتنقلة.

الكلمات المفتاحية: روبوت متنقل، تقنيات التحكم، الملاحة، التحكم الانزلاقي (CMS)، DIP، نظام 2SOR، جازيبو، سوليدووركس.

Summary

This thesis presents the design and development of a mobile robotic base conceived as a versatile platform for testing and validating various control techniques. The system is designed to be modular and adaptable, allowing the implementation of control strategies such as PID, navigation algorithms, and Sliding mode controll (SMC) in real-world environments. The work includes mechanical design, electronic integration, and the development of embedded control algorithms to ensure functionality, reliability, and flexibility. This platform is intended to support both research and educational efforts by providing a practical tool for experimentation and performance analysis of advanced control systems in mobile robotics.

Key Words : mobile robot, control techniques, navigation, SMC, PID, ROS2, Gazebo, SolidWorks.

Résumé

Cette thèse présente la conception et le développement d'une base robotique mobile conçue comme une plateforme polyvalente pour tester et valider différentes techniques de commande. Le système est conçu pour être modulaire et adaptable, permettant la mise en œuvre de stratégies de contrôle telles que la commande PID, les algorithmes de navigation et la commande (SMC) dans des environnements réels. Le travail comprend la conception mécanique, l'intégration électronique ainsi que le développement d'algorithmes de contrôle embarqués afin de garantir la fonctionnalité, la fiabilité et la flexibilité du système. Cette plateforme a pour objectif de soutenir à la fois la recherche et la formation en offrant un outil pratique pour l'expérimentation et l'analyse des performances des systèmes de commande avancés en robotique mobile.

Mots-clés : robot mobile, techniques de commande, navigation, SMC, PID, ROS2, Gazebo, SolidWorks.

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

A* A-star (pathfinding algorithm)

AI Artificial Intelligence

AMCL Adaptive Monte Carlo Localization

AMR Autonomous Mobile Robot

BLDC Brushless Direct Current (motor)

BMS Battery Management System

CAD Computer-Aided Design

CPU Central Processing Unit

CSV Comma-Separated Values (file format)

DDWMR Differential Drive Wheeled Mobile Robot

DOF Degree of Freedom – Total number of independent variables required to describe the robot's position and orientation in space.

DOM Degree of Mobility – Number of controllable independent motions achieved through actuators.

DOS Degree of Steerability – Number of independent passive steering configurations available to the robot.

DRL Deep Reinforcement Learning

EKF Extended Kalman Filter

FEA Finite Element Analysis

GPIO General-Purpose Input/Output

GPS Global Positioning System

I2C Inter-Integrated Circuit (communication protocol)

IMU Inertial Measurement Unit

LiDAR Light Detection and Ranging

MPC Model Predictive Control

Nav2 Navigation 2 (ROS2 navigation stack)

PID Proportional-Integral-Derivative (controller)

PWM Pulse Width Modulation

RL Reinforcement Learning

ROS2 Robot Operating System 2

RViz ROS Visualization Tool

SBC Single-Board Computer

SLAM Simultaneous Localization and Mapping

SMC Sliding Mode Control

TF Transform Frame (ROS-specific coordinate frame transformations)

UAV Unmanned Aerial Vehicle (drone)

UART Universal Asynchronous Receiver-Transmitter

UGV Unmanned Ground Vehicle

URDF Unified Robot Description Format

USB Universal Serial Bus

XML Extensible Markup Language

General Introduction

Mobile robotics represents one of the most rapidly evolving and interdisciplinary fields within contemporary engineering, propelled by significant advancements in sensor technology, embedded computing, control theory, artificial intelligence, and mechatronics. Autonomous mobile robots today play essential roles across diverse sectors, including industrial logistics, healthcare, agricultural automation, exploration, and environmental monitoring. The demand for robotic systems capable of navigating complex, dynamic environments, adapting intelligently to uncertainties, and reliably performing precise tasks is continually increasing.

Recent technological breakthroughs have dramatically expanded the capabilities of mobile robotic systems. Advanced sensors, particularly Light Detection and Ranging (LiDAR), Inertial Measurement Units (IMUs), and high-resolution cameras, combined with powerful embedded computing platforms like Raspberry Pi, facilitate robust environmental perception and real-time decision-making. Moreover, sophisticated algorithms such as Simultaneous Localization and Mapping (SLAM), Model Predictive Control (MPC), Sliding Mode Control (SMC), and Deep Reinforcement Learning (DRL) have significantly improved the autonomy, precision, and adaptability of robotic platforms.

In this context, the main objective of our engineering project is the design and construction **of a versatile and modular mobile robotic platform**. This platform serves as a comprehensive experimental testbed specifically tailored to evaluate, compare, and validate various classical and advanced control strategies. Using affordable and widely accessible hardware components, the system incorporates repurposed hoverboard brushless DC (BLDC) motors, a Raspberry Pi 4 for computational tasks, ZS-X11H motor drivers for precise actuator control, and an integrated sensor suite comprising LiDAR, IMU, and wheel encoders. Our software architecture utilizes the Robot Operating System 2 (ROS 2) framework to ensure modularity, scalability, and ease of integration.

The methodological approach undertaken in this project involves several interconnected phases:

1. **Mechanical and Electrical Design:** Focused on structural robustness, mechanical stability, and modularity, validated through Computer-Aided Design (CAD) modeling, structural simulations, and practical prototyping.

2. **Software and Control Architecture Integration:** Developed around ROS 2, facilitating real-time sensor integration, actuator control, communication, and data visualization, enabling both manual teleoperation and autonomous operation.

3. **Implementation of Control Algorithms:** Included classical control methods (PID) and robust techniques (Sliding Mode Control), emphasizing trajectory tracking, stability, and performance under realistic conditions.

4. **Simulation and Validation:** Employed Gazebo and RViz to simulate realistic operational environments, enabling thorough validation of the mechanical design, control algorithms, and sensor integration before real-world deployment.

5. **Real-World Testing and Analysis:** Conducted systematic practical experiments to validate system performance, analyze control effectiveness, and refine the overall integration based on empirical feedback.

The significance of this project lies not only in the realization of a functional robotic platform but also in its contribution to research and educational initiatives. By providing an adaptable, robust, and scalable testbed, it facilitates practical exploration of advanced control strategies and supports ongoing research in robotics, intelligent systems, and automation.

The document is structured clearly to guide the reader through the project's comprehensive scope:

- **Chapter 1:** Presents foundational concepts and classifications of mobile robots, kinematic and dynamic modeling principles, localization methods, and core challenges.
- **Chapter 2:** Provides detailed insights into implemented control techniques (PID, SMC), theoretical backgrounds, algorithmic descriptions, and comparative performance evaluations.
- **Chapter 3:** Details the complete hardware and software control architecture, emphasizing sensor integration, actuator management, and user interaction within ROS 2.
- **Chapter 4:** Describes the mechanical and electrical design of the robotic platform, material selection, CAD modeling, structural analysis, and integration of electronics.
- **Chapter 5:** Offers comprehensive experimental results, including simulations and real-world validations, assessing system performance across various scenarios and trajectories.
- **Chapter 6:** Concludes with a synthesis of the project's contributions, encountered challenges, and recommendations for future enhancements and research directions.

Annex: To further facilitate replication, understanding, and enhancement of our project, Annex A provides detailed documentation of our ROS 2 workspace structure, clearly outlining all packages, nodes, and launch files developed. This annex serves as a practical reference for future researchers and engineers aiming to build upon this work, ensuring transparency and ease of adaptation for diverse experimental and educational contexts.

Ultimately, this project delivers a versatile, economically viable robotic platform that embodies the intersection of theoretical control concepts and practical engineering implementation, establishing a valuable resource for ongoing and future research in mobile robotics.

Chapter 1

Overview on Mobile Robots

1.1 Introduction

Mobile robots are intelligent machines capable of autonomously moving within their environment [1, 2]. They are distinguished from other types of robot by their ability to move independently and the intelligence required to react and make decisions based on their perception of the world. A mobile robot must have an input data source, a way to interpret this information, and the ability to act (including self-movement) in response to a changing environment. The need to detect and adapt to an unknown environment requires a powerful cognitive system [2]. An autonomous mobile robot can make intelligent decisions by perceiving its surroundings and nearby objects. These robots have quickly transitioned from research laboratories to automated industries, taking on a variety of roles in our daily lives [3].

1.2 Historical Context and Applications of Mobile Robots

The evolution of mobile robotics has been marked by significant milestones since the mid-20th century. A notable early development was *Shakey the Robot*, introduced in the 1960s by the Stanford Research Institute. Shakey was among the first robots capable of perceiving its environment and making autonomous decisions, laying the groundwork for future advancements in robotic autonomy[1].

Over the decades, mobile robots have found applications across various sectors:

- **Industrial Automation**: Autonomous mobile robots (AMRs) have revolutionized manufacturing and logistics by efficiently transporting goods within facilities, thereby enhancing productivity and reducing human labor.
- **Healthcare Services**: In medical settings, mobile robots assist in tasks such as delivering supplies and medications, contributing to improved operational efficiency and patient care.
- **Agriculture**: Robotic platforms are employed for precision farming activities, including crop monitoring and harvesting, leading to increased agricultural productivity.
- **Exploration and Surveillance**: Mobile robots are utilized in hazardous environments for exploration and monitoring purposes, minimizing human exposure to dangerous conditions.
- **Education and Research**: Academic institutions leverage mobile robots as platforms for teaching and experimenting with control algorithms, fostering innovation in robotic technologies.

The diverse applications of mobile robots underscore their integral role in advancing technology and addressing complex tasks across multiple domains.

1.3 Comparison of Holonomic and Non-Holonomic Robots

Mobile robots can be classified based on their kinematic constraints into two main categories:

- **Holonomic robots** are systems in which the number of controllable degrees of freedom (DOF) is equal to the total number of degrees of freedom required to describe their motion. In other words, they can move freely in all directions permitted by their configuration space. These robots can achieve any arbitrary velocity in the plane (e.g., move sideways and rotate simultaneously), and are not subject to non-integrable motion constraints. An example is an omnidirectional robot equipped with Mecanum or Swedish wheels.

- **Non-holonomic robots**, on the other hand, have motion constraints that cannot be expressed as integrable constraints (i.e., they are non-integrable differential constraints). These robots cannot move instantaneously in certain directions due to their mechanical design. For example, a differential drive or car-like robot cannot move sideways, and must instead follow arcs or curves to reorient.

The table below summarizes key differences between holonomic and non-holonomic mobile robots using three key measures: Degree of Freedom (DOF), Degree of Mobility (DOM), and Degree of Steerability (DOS). It also includes the holonomic property of each type.

Robot Type	**DOF**	**DOM**	**DOS**	**Holonomic**
Omnidirectional	3 (x, y, θ)	3	0	Yes
Differential Drive	3 (x, y, θ)	2	0	No
Car-like (Ackermann)	3 (x, y, θ)	2	1	No

Table 1.1: Comparison of mobile robot types with respect to DOF, DOM, DOS, and holonomy.

1.4 Classification of Mobile Robots

Mobile robots come in various forms depending on their locomotion mechanisms, application domains, and environmental interactions. Understanding their classification is essential for selecting appropriate designs and control strategies for specific tasks. This section provides an overview of the main categories of mobile robots, focusing on locomotion types (e.g., wheeled, legged, hybrid) and application-based distinctions (e.g., service, industrial, exploration).

1.4.1 Locomotion-Based Classification

a. Wheeled Robots

These are the most common type due to their simple structure, energy efficiency, high speed, and low production cost [4]. Early developments in mobile robotics used wheeled platforms for simple tasks such as line following, one of the first examples being Professor Walter's turtle robot in 1948. Wheeled robots are popular for testing control architectures and navigation systems due to their mechanical simplicity [5].
They can be further categorized based on their drive system:

- Differential drive robots use actuators to drive each wheel independently, often with a caster wheel for balance [2, 6].
- Omnidirectional robots can move in any direction using Swedish, Mecanum, or spherical wheels [2, 6].

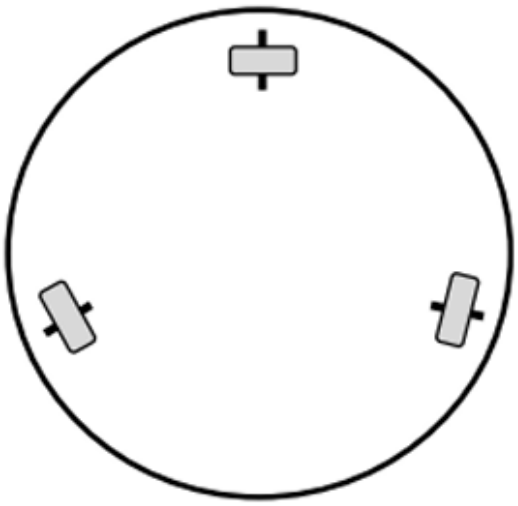


Figure 1.1: Omnidirection Mobile Robot.

- Car-type Mobile Robot mimics car-like motion. Wheels vary in degrees of freedom: standard (1 DOF), caster (2 DOF), and Swedish (3 DOF). No single drive system perfectly balances maneuverability, controllability, and stability[2, 6].

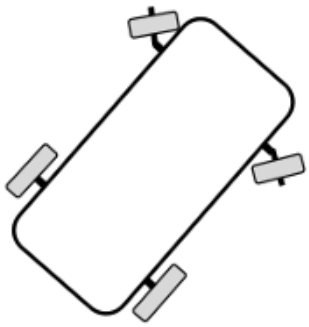


Figure 1.2: Car-type Mobile Robot.

b. Legged Robots

Developed alongside wheeled robots, the study of legged locomotion began over a century ago. Examples include the Walking Truck by Ralph Mosher (1968). Various leg configurations have been explored, with hexapods being popular for their stability during walking[5]. Humanoids resemble the human body and can maintain static balance, while bipeds are capable of walking, running, stair climbing, and jumping, requiring dynamic stability. There are also monopods ("hoppers") that use jumping to stay balanced[2].

Hybrid Systems

Hybrid mobile robots combine two or more types of locomotion mechanisms—like wheels, legs, or tracks—to take advantage of the strengths of each method while minimizing their limitations.

1.4.2 Application-Based Classification

a. Service Robots

Since the 1970s, robotics has evolved to provide useful services to humans. These include personal assistants, automated wheelchairs, and guide robots in museums or hospitals. Applications also cover telepresence and assistive technologies[5, 4].

b. Industrial Logistics Robots

Used in industrial automation and goods distribution, including unmanned ground vehicles (UGVs) for outdoor navigation. Car-like four-wheeled robots are increasingly important in logistics and transport[2, 5].

c. Exploration and Surveillance Robots

Used for planetary exploration, operations in hazardous environments, and military/security surveillance[5]. Emergency rescue and reconnaissance fall under this category, along with the development of autonomous networked robots (ANR) for monitoring.

d. Autonomous Vehicles and Drones

Aerial and underwater robots represent growing fields with challenges in control and wireless communication. Unmanned aerial vehicles (UAVs or drones) perform preprogrammed missions with or without human input. Autonomous ground vehicles (AIVs) are a major research focus. Modern autonomous vehicles integrate locomotion and control systems[4, 2].

1.5 wheelded differentiel mobile robot

A mobile robot **with differential drive** is a type of wheeled robot that uses two independently driven wheels to move. These wheels are typically placed in parallel, one on each side of the robot. By varying the speed and direction of each wheel, the robot can move forward, backward, or rotate in place. This configuration provides simple control and excellent maneuverability, which is why it is commonly used in research, education, exploration, surveillance, and logistics applications [7, 8].

To facilitate the modeling and analysis of the system, we make the following assumptions.

- The robot's body is rigid and does not deform during motion.
- The mass of the mobile platform remains constant.
- The robot operates in a two-dimensional plane (XY-plane).
- The friction between the wheels and the ground is neglected.
- The wheels roll without slipping.

1.5.1 Instantaneous Center of Rotation (ICR) and Stability Considerations

The **Instantaneous Center of Rotation (ICR)** is the point about which a differential drive robot is instantaneously rotating. When the two wheels move at different speeds, the robot follows a circular trajectory, and the ICR is located at the center of this circle. This concept is well illustrated in Figure 1.3.

Let:

- v_r: linear velocity of the right wheel
- v_l: linear velocity of the left wheel
- L: distance between the two wheels (wheelbase)

The horizontal position of the ICR relative to the midpoint between the two wheels (robot center) is given by:

$$\mathrm{ICR}_x = \frac{L}{2} \cdot \frac{v_r + v_l}{v_r - v_l} \tag{1.1}$$

Key observations:

- If $v_r = v_l$, then $\mathrm{ICR}_x \to \infty$: the robot moves in a straight line without rotation.
- If $v_r = -v_l$, then $\mathrm{ICR}_x = 0$: the robot rotates about its own center (on the spot rotation).
- Otherwise, the robot turns around an ICR located at a finite distance along the axis perpendicular to its wheels.

Understanding the position of the ICR is essential for designing control algorithms and planning the robot's motion accurately, particularly when executing curved trajectories or tight turns.

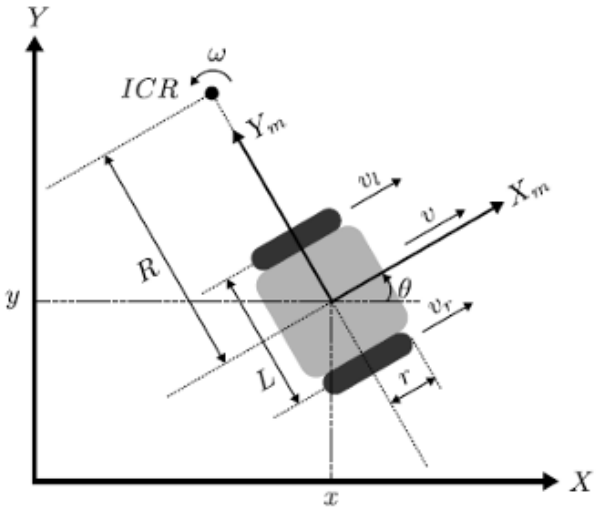


Figure 1.3: Basic structure and rotation behavior of a differential-drive mobile robot.

1.5.2 Center of Gravity (CG) and Stability Analysis:

In addition to the trajectory described by the ICR, the **Center of Gravity (CG)** plays a crucial role in the robot's dynamic stability, particularly during turning or under high acceleration conditions.

For the mobile platform designed in this work:

- Chassis dimensions: 40 cm (length) × 30 cm (width)
- Estimated total mass: 30 kg
- Wheel radius: 8.25 cm

Using the *Mass Properties* feature in SolidWorks, the center of gravity (CG) of the robot was estimated from the complete CAD assembly. The CG is approximately located at:

$$(x_{CG}, y_{CG}) \approx (20 \text{ cm},\ 15 \text{ cm})$$

where the origin of the coordinate system is defined at the rear-center of the robot chassis. This central positioning of the CG provides several key advantages:

- Balanced load distribution between the two drive wheels, ensuring optimal traction.
- Mitigation of pitching effects during acceleration and braking.
- Enhanced lateral stability, reducing the risk of rollover during sharp maneuvers.

Maintaining the CG close to the geometric center and low to the ground enhances the overall dynamic stability of the robot, allowing it to execute aggressive maneuvers safely without the risk of tipping or loss of traction.

Finally, the combination of ICR positioning and CG control is fundamental in ensuring accurate and stable movement of differential drive mobile robots, especially when high performance control techniques (such as MPC or SMC) are implemented.

1.6 Advanced Kinematic and Dynamic Modeling of Differential Drive Robots

Modeling the movement of mobile robots is essential for developing effective control strategies. The kinematic and dynamic models define how the robot moves and responds to inputs and external forces.

1.6.1 Kinematic Model

The mobile platform studied in this project follows a Differential Drive Wheeled Mobile Robot (DDWMR) configuration. In such systems, two wheels are independently driven and aligned along a shared axis, while a passive caster wheel provides balance [9]. The movement of the robot is controlled by adjusting the relative speeds of the left and right wheels, allowing it to go straight, turn, or rotate in place [9].

Consider a differential drive robot moving in a 2D plane, with:

- (x, y): position of the robot
- θ: orientation (heading) of the robot
- v_r, v_l: linear velocities of the right and left wheels

- L: distance between the two wheels

The linear and angular velocities of the robot are:

$$v = \frac{v_r + v_l}{2}$$
$$\omega = \frac{v_r - v_l}{L}$$

Thus, the kinematic model is:

$$\begin{cases} \dot{x} = v\cos\theta = \frac{v_r+v_l}{2}\cos\theta \\ \dot{y} = v\sin\theta = \frac{v_r+v_l}{2}\sin\theta \\ \dot{\theta} = \omega = \frac{v_r-v_l}{L} \end{cases} \tag{1.2}$$

This is a non-holonomic system with the constraint:

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0$$

which forbids lateral (sideways) motion.

1.6.2 Dynamic Model

For dynamics, consider:

- m: mass of the robot
- I_z: moment of inertia about the vertical axis
- F_r, F_l: forces generated by the right and left wheels

The equations of motion (Newton–Euler) are:

$$m\dot{v} = F_r + F_l$$
$$I_z\dot{\omega} = L(F_r - F_l)/2$$

In terms of control inputs (motor torques τ_r, τ_l):
Assuming the wheels have radius r, then:

$$F_r = \frac{\tau_r}{r}, \quad F_l = \frac{\tau_l}{r}$$

Thus:

$$m\dot{v} = \frac{\tau_r + \tau_l}{r}$$
$$I_z\dot{\omega} = \frac{L}{2r}(\tau_r - \tau_l)$$

1.6.3 Summary State-Space Representation

Define state vector:

$$\mathbf{x} = [x, y, \theta, v, \omega]^T$$

and input vector:

$$\mathbf{u} = [\tau_r, \tau_l]^T$$

Then, the state-space form becomes:

$$\dot{\mathbf{x}} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ \frac{1}{mr}(\tau_r + \tau_l) \\ \frac{L}{2I_z r}(\tau_r - \tau_l) \end{bmatrix}$$

This full dynamic model captures both translational and rotational behavior considering actuator torques, making it suitable for more advanced control design such as LQR, SMC, or MPC.

1.7 Localization in Mobile Robots

Regardless of the type of robot, knowing its position and orientation over time is essential to determine where it is and where it is going. Robot localisation requires three coordinates: two Cartesian coordinates to define the position and one angular coordinate to define the orientation. More formally, localisation refers to determining the transformation that maps the robot's coordinate frame to a reference frame fixed in the environment.

A localisation system comprises both the sensors and the data processing algorithms that enable a robot or vehicle to autonomously estimate its motion or position in space [10]. This system typically provides two types of information:

- **Relative localisation**: Based on *dead reckoning*, it uses proprioceptive sensors to estimate movement by relying solely on the robot's internal measurements.
- **Absolute localisation**: It relies on exteroceptive sensors to estimate the robot's position with respect to a fixed reference frame in the environment.

Sensors are fundamental components of any localisation system and can be categorized into two main types:

1.7.1 Exteroceptive sensors

Exteroceptive sensors are essential for enabling robots to perceive and interact with their environment. They collect external information that allows the robot to recognize objects, build a model of its surroundings, and detect interactions such as position, distance, and applied forces. These sensors play a crucial role in tasks such as mapping, obstacle avoidance, object detection, and environmental understanding. Below are some of the most commonly used exteroceptive sensors in mobile robotics:

- **Light Detection and Ranging (LiDAR)**: Measure the distance to nearby objects by analyzing the time it takes for a laser beam to bounce back, enabling detailed 2D or 3D mapping.

- **Cameras (RGB, Stereo, RGB-D)**: Provide visual information. Stereo and depth cameras (e.g., RGB-D sensors) also estimate depth and can be used for object detection, tracking, and visual SLAM.
- **Ultrasonic Sensors**: Emit ultrasonic waves and measure the time of flight to determine the distance to obstacles. They are cost-effective and widely used for short-range detection.
- **Infrared Sensors**: Use infrared light to detect nearby objects or surfaces. Often used in line-following robots or simple obstacle detection.
- **Radar**: Useful in detecting objects in adverse conditions (e.g., fog, rain). Often used in autonomous vehicles for robust obstacle detection.

Each sensor type has its own pros and cons. The right sensor depends on what the robot needs to do and where it operates. For instance, while GPS is fantastic for outdoor navigation, it's not useful indoors. On the other hand, ultrasonic sensors are excellent for avoiding close-up obstacles but aren't suitable for mapping larger areas. By choosing the right mix of sensors, robots can better navigate and interact with their environments autonomously and efficiently. Figure 1.4 illustrate below we can see the most common sensors and their characteristic

1.7.2 Proprioceptive Sensors

Proprioceptive sensors function like the robot's internal senses, providing continuous feedback about its own state. In mobile robots, these sensors play a critical role in estimating motion-related parameters such as speed, acceleration, and orientation. For example :

- **Odometry :** Odometry is the process of estimating a mobile robot's position and orientation using internal sensors, particularly wheel encoders. It is a relative localization technique that assumes that the robot's displacement can be inferred from the motion of its wheels.

  For a differential drive robot, the position update equations are derived as follows:

  - R: Radius of the wheels
  - L: Distance between the two wheels
  - $\Delta\theta_L$ and $\Delta\theta_R$: Angular displacements of the left and right wheels
  - $\Delta s_L = R \cdot \Delta\theta_L$: Distance traveled by the left wheel
  - $\Delta s_R = R \cdot \Delta\theta_R$: Distance traveled by the right wheel
  - $\Delta s = \frac{\Delta s_R + \Delta s_L}{2}$: Linear displacement of the robot
  - $\Delta\theta = \frac{\Delta s_R - \Delta s_L}{L}$: Change in orientation

  If the robot's pose is (x, y, θ), the new pose (x', y', θ') is computed as:

$$x' = x + \Delta s \cdot \cos\left(\theta + \frac{\Delta\theta}{2}\right)$$

$$y' = y + \Delta s \cdot \sin\left(\theta + \frac{\Delta\theta}{2}\right)$$

$$\theta' = \theta + \Delta\theta$$

  These equations are used iteratively to track the robot's trajectory over time.
  Figure 1.4 illustrates below we can see the most common sensors and their characteristic.

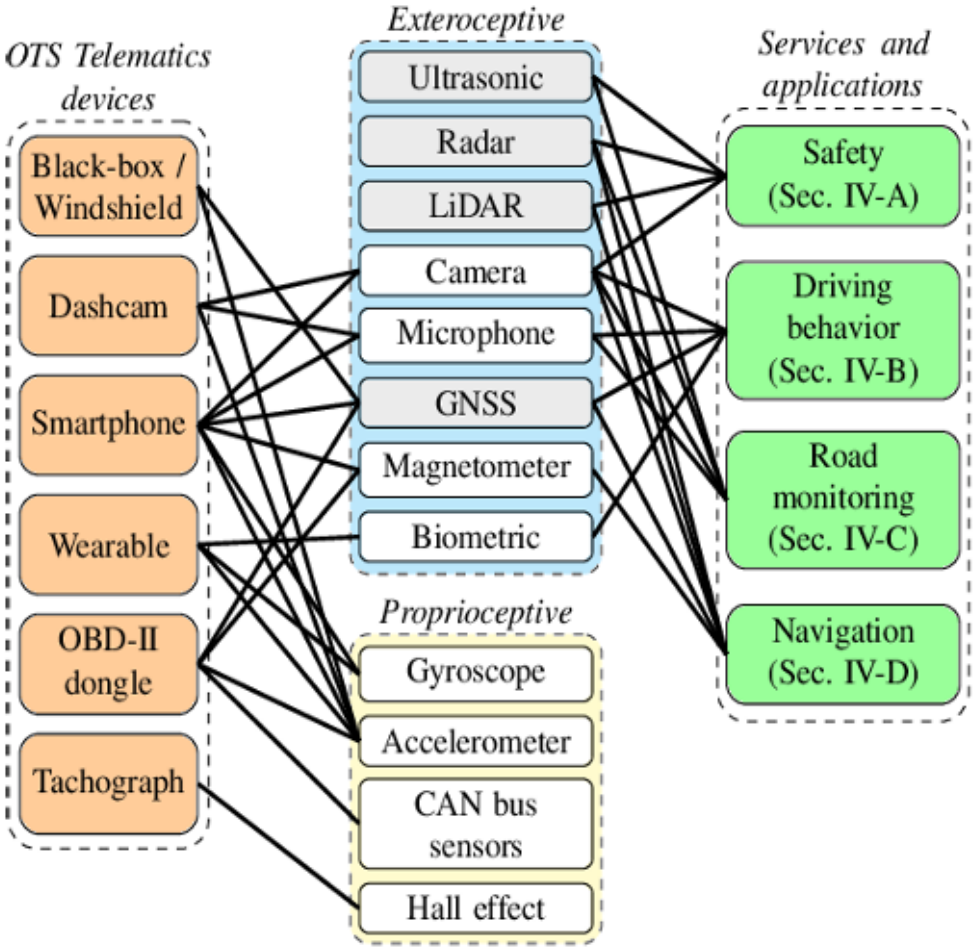


Figure 1.4: Exteroceptive and proprioceptive sensors.

1.8 Overview of Control Techniques in Mobile Robotics

Control strategies in mobile robotics have evolved significantly, encompassing both classical and modern approaches to address the complexities of autonomous navigation and task execution.

1.8.1 Classical Control Methods

Classical control techniques, such as Proportional-Integral-Derivative (PID) controllers, have long been fundamental due to their simplicity and efficiency in linear systems. These methods are particularly effective for basic trajectory tracking and stability control in structured environments. However, their performance can degrade in the presence of system nonlinearities and external disturbances [11].

1.8.2 Intelligent Control Approaches

To overcome the limitations of classical methods, intelligent control strategies have been introduced:

- **Fuzzy Logic Controllers (FLCs)**: FLCs manage uncertainties and imprecise inputs by emulating human reasoning. They have been successfully applied in tasks such as navigation and obstacle avoidance [12].

- **Neural Networks (NNs)**: NNs are capable of learning from data and modeling complex nonlinear systems, which makes them suitable for decision-making in dynamic environments [13].
- **Neuro-Fuzzy Systems**: These combine neural networks and fuzzy logic, leveraging the strengths of both methods for adaptive control in uncertain and changing environments [13].

1.8.3 Advanced Control Techniques

More recent control methods aim to improve adaptability and performance:

- **Model Predictive Control (MPC)**: MPC forecasts future system behavior and optimizes control inputs accordingly, making it suitable for constrained and multivariable systems [12].
- **Reinforcement Learning (RL)**: RL enables robots to learn optimal behaviors through interaction with their environment, guided by reward-based feedback [14].
- **Deep Reinforcement Learning (DRL)**: DRL integrates deep learning with RL, allowing robots to process high-dimensional inputs and learn complex control policies [15].

These techniques collectively enhance the autonomy and intelligence of mobile robots, enabling robust performance in dynamic and uncertain environments.

1.9 Challenges in Mobile Robotics

Although mobile robotics has made significant advancements in recent years, several technical and practical challenges remain. These challenges stem from the complexity of real-world environments, hardware limitations, and the need for real-time processing.

1.9.1 Perception and Environment Understanding

Accurate perception is fundamental to mobile robots for tasks such as mapping, localization, and obstacle avoidance. Sensors like LiDAR, cameras, and IMUs can be affected by noise, occlusions, and changes in environmental conditions (e.g., lighting or weather). Effective sensor fusion remains a complex task, particularly in unstructured or dynamic settings.

1.9.2 Localization and Navigation

Reliable localization is a prerequisite for autonomous navigation. In GPS-denied environments or indoors, techniques like visual odometry or SLAM (Simultaneous Localization and Mapping) are used. However, these approaches can suffer from drift, high computational demands, and difficulties in loop closure detection.

1.9.3 Control under Uncertainty

Mobile robots often encounter uncertain and changing environments, such as uneven terrain or slippery surfaces. Designing robust control algorithms that maintain stability under such conditions is challenging, particularly for non-holonomic systems.

1.9.4 Energy Efficiency and Power Management

Battery-powered mobile robots face energy limitations that constrain their sensing, computation, and actuation capabilities. Efficient power management is crucial for long-term autonomy. Integration of alternative energy sources such as fuel cells introduces further challenges in energy-aware planning and control.

1.9.5 Human-Robot Interaction and Safety

Ensuring safe and intuitive interaction with humans is vital, especially in shared or public environments. Robots must interpret human actions and intentions, follow social norms for navigation, and ensure fail-safe operations to prevent accidents.

1.9.6 Scalability and Multi-Robot Coordination

In multi-robot systems, coordination and communication are major issues. Distributed decision-making, task allocation, and collision avoidance require robust and scalable algorithms that can adapt to dynamic group behaviors and partial knowledge of the environment.

These challenges highlight the interdisciplinary nature of mobile robotics and the ongoing need for innovation in perception, control, and systems integration.

Conclusion

In this chapter, the fundamental concepts of mobile robotics have been examined, with particular attention to robot classification, kinematics, and the control strategies that facilitate autonomous movement. The significance of sensors and perception in navigation has been highlighted, along with the principal challenges encountered by mobile robots in real-world environments.

The mobile base designed and constructed as part of this engineering project provides a flexible platform for testing a variety of control techniques under practical conditions. A thorough understanding of mobile robot principles and applications informs the design decisions behind this experimental platform and underscores its value in evaluating different control algorithms.

Chapter 2

Control Techniques and Navigation Algorithm Implemented

2.1 Introduction

This chapter presents the theoretical foundations of the control strategies and navigation algorithms considered for mobile robotics. It begins with an overview of the kinematic constraints and challenges specific to differential drive robots. Subsequently, classical and advanced control approaches such as Proportional-Integral-Derivative (PID) and Sliding Mode Control (SMC) are detailed in terms of their mathematical principles and control objectives. The chapter also introduces high level navigation concepts including path planning and localization strategies, with emphasis on their role in autonomous mobile robot operation. These foundations provide the necessary background for selecting and comparing suitable control laws in later stages of the project.

2.2 System Model Used for Control

In order to design and implement control algorithms on the mobile robotic platform, it is essential to establish a mathematical model that captures the dynamics of the system. Control strategies such as PID and Sliding Mode Control (SMC) require an analytical representation of the robot's motion, typically in the form of state-space equations.

2.2.1 Nonlinear Kinematic Model

The kinematic model of a differential-drive mobile robot (DDMR) describes the relation between the control inputs (linear and angular velocities) and the robot's pose (x, y, θ). The standard nonlinear kinematic equations are given by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.1}$$

where:

- x, y represent the position of the robot in the inertial frame,
- θ is the robot's orientation,
- v is the linear velocity, and ω is the angular velocity.

This model is nonlinear due to the trigonometric terms, which complicates the design of linear controllers. Thus, for simplification and practical implementation, a linear approximation around a specific operating point is considered.

2.2.2 Linearized Kinematic Model

We perform a first-order Taylor expansion (Jacobian linearization) of the nonlinear model around the operating point $\theta = 0$, which corresponds to a robot facing along the x-axis. Assuming small deviations around this posture, the trigonometric terms become:

$$\cos(\theta) \approx 1, \quad \sin(\theta) \approx 0$$

Substituting into the nonlinear model yields the linearized equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \approx \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.2}$$

Controllability by the Lie Bracket Method (Crochet de Lie) For the linearized kinematic model of the differential drive robot at $\theta = 0$, the system equations are:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

This corresponds to the control vector fields:

$$g_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad g_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

To check controllability via the Lie bracket (crochet de Lie) method, we compute the Lie bracket:

$$[g_1, g_2] = \frac{\partial g_2}{\partial X} g_1 - \frac{\partial g_1}{\partial X} g_2$$

However, both g_1 and g_2 are constant vectors (do not depend on the state X), so all derivatives are zero:

$$[g_1, g_2] = 0$$

Thus, the Lie algebra generated by $\{g_1, g_2\}$ only spans a two-dimensional subspace:

$$\text{span} \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

which is insufficient to control the system in the y direction.

Interpretation: For a differential drive robot linearized at $\theta = 0$, only the forward motion (x direction) and heading (θ) are directly controllable through v and ω. Lateral motion (y direction) is not directly actuated, and the Lie bracket does not generate motion in this direction. This means the linearized system is **not controllable** in the whole state space. This reflects the nonholonomic constraint of the robot: sideways motion is only achievable via nonlinear combinations of forward and rotational commands, a property that is lost after linearization. In contrast, the original nonlinear model is controllable via higher-order Lie brackets.

2.2.3 Justification of the Linear Model

Although the full nonlinear dynamic model of differential-drive mobile robots includes inertial, Coriolis, and constraint forces [16], such models are often complex to identify and require precise knowledge of physical parameters. In their work, Saad et al. [16] propose a Lagrangian-based dynamic model, projected into a reduced-order space that accounts for nonholonomic constraints, and apply an adaptive sliding mode controller to cope with parameter uncertainties and external disturbances. However, in the context of our platform, which operates at low to moderate speeds and primarily uses velocity control via high-level inputs, such complexity is not necessary.

Therefore, we adopt a simplified kinematic model, linearized around a nominal operating point $(x_0, y_0, \theta_0 = 0)$ to ease the controller design. The linearized system described in Equation 2.2 provides sufficient accuracy for implementing classical controllers such as PID and SMC, which will be detailed in the subsequent sections.

This choice balances model fidelity and implementation efficiency, making it well suited for embedded real-time applications and experimental validation on the constructed mobile platform.

2.3 Classical Control

2.3.1 PID Controller

Proportional-Integral-Derivative (PID) control remains one of the most widely used techniques in mobile robotics due to its simplicity, ease of implementation, and effectiveness in regulating system output in the presence of disturbances and noise. In this section, we present the design and implementation of a PID controller based on the linearized kinematic model of the mobile base introduced in the previous section.

Control Objective

The main goal of the PID controller in our system is to regulate the robot's linear and angular velocities (v and ω) so that it can follow a desired trajectory defined by reference signals v_d and ω_d. The controller works at the velocity level, ensuring that:

$$v(t) \to v_d(t), \quad \omega(t) \to \omega_d(t)$$

PID Control Law

The general form of the PID control law for each control variable is:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{d}{dt} e(t) \tag{2.3}$$

where:

- $e(t) = r(t) - y(t)$ is the control error,
- K_P, K_I, and K_D are the proportional, integral, and derivative gains respectively.

In our case, we apply this control structure separately to the linear velocity v and angular velocity ω:

$$u_v(t) = K_{P_v} e_v(t) + K_{I_v} \int e_v(t) dt + K_{D_v} \frac{de_v(t)}{dt} \tag{2.4}$$

$$u_\omega(t) = K_{P_\omega} e_\omega(t) + K_{I_\omega} \int e_\omega(t) dt + K_{D_\omega} \frac{de_\omega(t)}{dt} \tag{2.5}$$

where:

- $e_v(t) = v_d(t) - v(t)$ is the linear velocity error,
- $e_\omega(t) = \omega_d(t) - \omega(t)$ is the angular velocity error.

The tuning of the PID gains was carried out manually through iterative testing on the physical robot and through simulations.

A schematic of the implementation is shown in Figure 2.1.

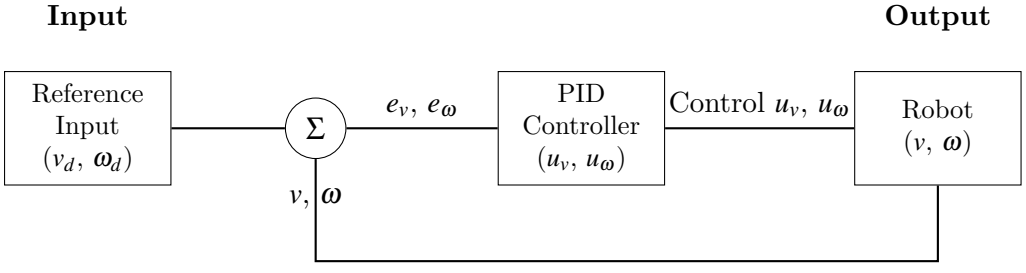


Figure 2.1: Block diagram of a PID controller for a differential robot.

2.4 Sliding Mode Control (SMC)

Sliding Mode Control (SMC) is a robust nonlinear control strategy that excels in managing systems affected by model uncertainties and external disturbances. In the context of mobile robotics, SMC is particularly valued for its strong convergence properties and its ability to maintain stability despite variations in system dynamics. This section introduces the theoretical principles of SMC and discusses its relevance for velocity regulation in differential-drive mobile robots.

Control Objective

Similar to the PID controller, the SMC aims to regulate the robot's linear and angular velocities (v and ω) to follow reference signals v_d and ω_d:

$$v(t) \to v_d(t), \quad \omega(t) \to \omega_d(t)$$

Sliding Surface Definition

The control design begins by defining the tracking errors:

$$e_v = v_d - v, \quad e_\omega = \omega_d - \omega$$

We then construct sliding surfaces s_v and s_ω as:

$$s_v = e_v + \lambda_v \int e_v \, dt, \quad s_\omega = e_\omega + \lambda_\omega \int e_\omega \, dt \tag{2.6}$$

where λ_v and λ_ω are positive gains that determine the convergence rate of the sliding surface.

SMC Control Law

The control inputs u_v and u_ω for the linear and angular velocities are computed using the standard SMC formulation:

$$u_v = K_v e_v + \lambda_v \int e_v \, dt + \eta_v \cdot \text{sat}\left(\frac{s_v}{\phi_v}\right) \tag{2.7}$$

$$u_\omega = K_\omega e_\omega + \lambda_\omega \int e_\omega \, dt + \eta_\omega \cdot \text{sat}\left(\frac{s_\omega}{\phi_\omega}\right) \tag{2.8}$$

Here:

- K_v, K_ω are equivalent control gains,
- η_v, η_ω are switching gains for robustness,
- ϕ_v, ϕ_ω are boundary layer thickness parameters to reduce chattering,
- $\text{sat}(\cdot)$ is the saturation function: $\text{sat}(x) = \max(-1, \min(1, x))$.

A schematic of the implementation is shown in Figure 2.2.

2.5 Autonomous Navigation with A∗Algorithm

Beyond trajectory tracking using feedback controllers, autonomous navigation functionalities were incorporated into the mobile robotic platform using the ROS2 Navigation Stack (Nav2). This framework enables the robot to autonomously navigate to a specified goal location by computing and following collision-free paths in a known environment.

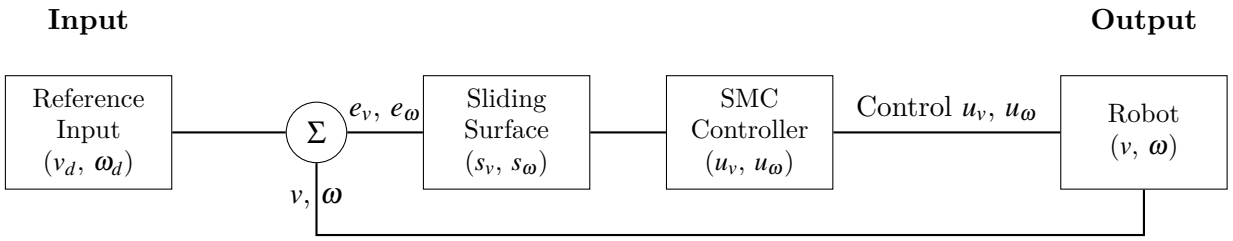


Figure 2.2: Block diagram of an SMC controller for a differential robot.

Navigation Architecture

The navigation stack consists of several coordinated modules:

- **Global Planner (A∗):** Determines an optimal path from the robot's current location to the goal by applying the A∗search algorithm over a 2D costmap.
- **Local Planner (DWB Controller):** Follows the global path while dynamically avoiding obstacles through real-time velocity sampling.
- **Map Server:** Serves the static occupancy grid map of the environment.
- **Localization:** Provided by the AMCL (Adaptive Monte Carlo Localization) algorithm using LiDAR and odometry.
- **Recovery Behaviors:** Executes strategies to recover from failures such as collisions, local minima, or oscillations.

A∗Global Path Planning

The (A*) algorithm is a graph-based search technique that finds the most cost-effective path between a start and goal point by minimizing a total cost function:

$$f(n) = g(n) + h(n)$$

where:

- $g(n)$ is the actual cost from the start node to the current node n,
- $h(n)$ is the heuristic estimate of the cost from n to the goal.

The A* algorithm ensures both completeness and optimality when applied with an admissible heuristic. Its efficiency and reliability make it a standard choice for path planning in structured, known environments.

Figure 2.3 illustrates the conceptual structure and decision process of the A* algorithm.

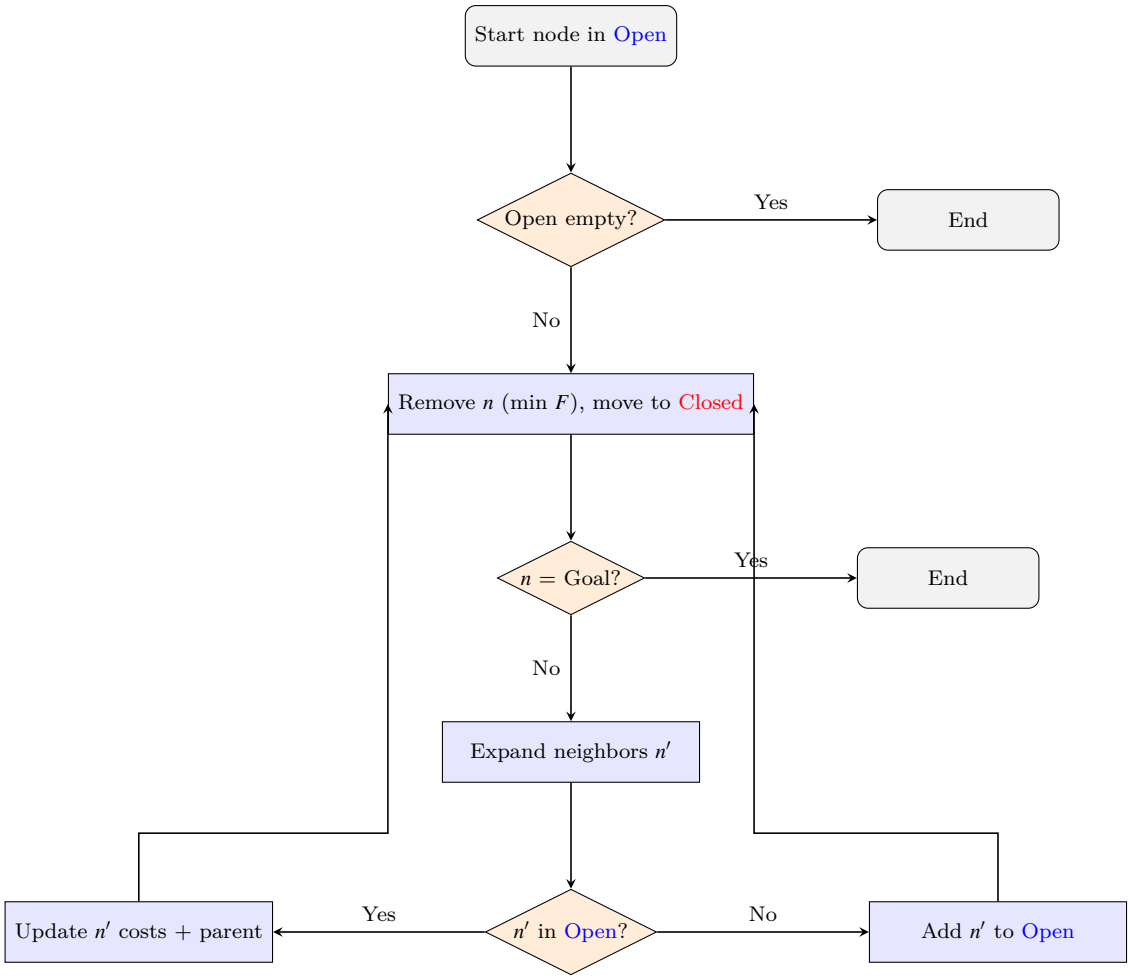


Figure 2.3: A* algorithm flowchart for global path planning.

A practical example of the resulting path generated by A* on a grid-based map is shown in Figure 2.4.

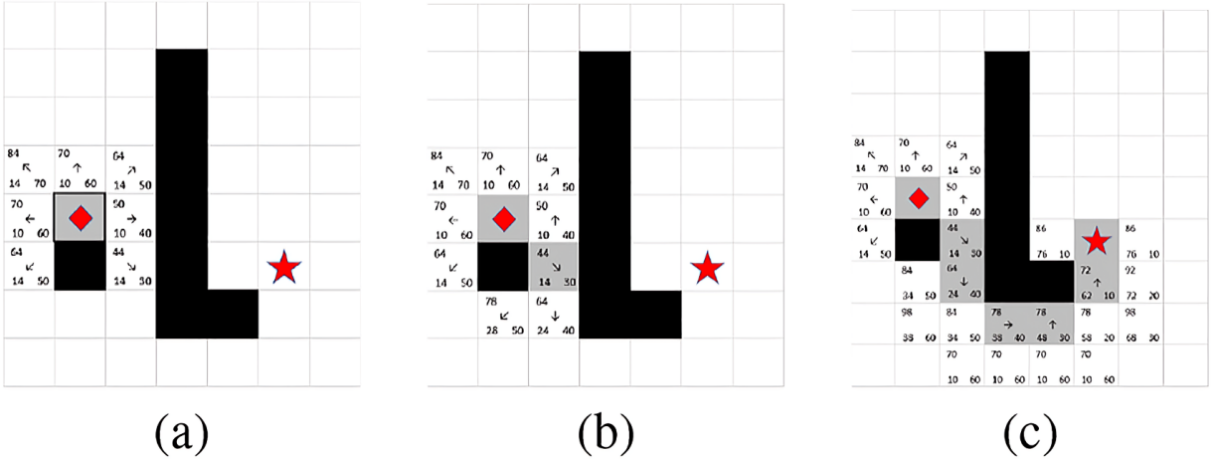


Figure 2.4: Illustration of the pathfinding process using A∗in a grid environment.

2.6 Comparison of Techniques

To evaluate the effectiveness of the implemented control strategies, a qualitative comparison is presented in Table 2.1. The table assesses each method based on common criteria: ease of implementation, robustness to disturbances, energy efficiency, and trajectory tracking accuracy.

Table 2.1: Comparison of Implemented Control Techniques

Control Technique	**Ease of Implementation**	**Robustness**	**Energy Efficiency**	**Trajectory Accuracy**
PID Control	High	Low–Moderate	Moderate	Moderate
Sliding Mode Control (SMC)	Moderate	High	High	High
Navigation (A* with Nav2)	High	High (with recovery)	Optimized via planner	Very High (global+local planning)

Conclusion

This chapter presented the control and navigation strategies applied to a differential-drive mobile robot. A linearized kinematic model served as the foundation for implementing two feedback controllers: Proportional-Integral-Derivative (PID) and Sliding Mode Control (SMC). Both techniques were developed and integrated as ROS 2 nodes to regulate the robot's linear and angular velocities.

The PID controller offered simplicity and ease of tuning, making it suitable for basic velocity tracking tasks. In contrast, the SMC approach demonstrated superior robustness and tracking accuracy, particularly in the presence of system disturbances and uncertainties.

In addition to local control, high-level autonomous navigation was implemented using the ROS 2 Navigation Stack (Nav2) with A* global path planning. This enabled the robot to navigate toward goal positions while avoiding obstacles within a known environment.

The modular design of the ROS 2 framework facilitated the integration of control and navigation components, providing a scalable foundation for future enhancements such as multi-robot coordination or AI-based decision-making.

Chapter 3

Control Architecture

3.1 Introduction

The control architecture of a mobile robotic platform encompasses the full suite of hardware and software components required to enable perception of the environment, informed decision making, and effective actuation. It serves as the backbone of the robotic system, facilitating the coordination between sensing devices, computational logic, control algorithms, and actuator commands. A robust and modular architecture ensures that the system is not only functionally reliable but also adaptable to evolving requirements, scalable for future upgrades, and tolerant to partial faults or disturbances. These qualities are particularly essential for mobile robots operating in dynamic or uncertain environments.

This chapter presents the architectural design implemented for the mobile base developed in this project. Details the integration of key hardware components, including sensors, actuators, and embedded computing units, with the software layers that govern communication, control, and data processing. The organization of the system is explained in terms of functional modules, highlighting how these elements interact within the ROS 2 framework to support both manual teleoperation and high level autonomous navigation. Through this architecture, the platform achieves real time responsiveness, operational flexibility, and a clear separation of concerns, making it suitable for experimental testing and further research.

3.2 General Architecture

The description above outlines a modular and layered control architecture for the mobile robotic platform. This structure divides the system into four main functional blocks: sensing, computation, actuation, and user interaction. Such an approach enhances the scalability and flexibility of the system, allowing for easy integration or replacement of components such as sensors, controllers, or processors without needing to redesign the entire architecture.

The actuation is handled by two brushless DC (BLDC) motors embedded in hoverboard wheels, each equipped with internal Hall-effect sensors that provide real-time rotational speed feedback. Rather than using the original hoverboard motherboard, the system employs a dual BLDC driver (ZS-X11H) for independent motor control. This driver simplifies integration with external microcontrollers (e.g., Arduino) or single-board computers (e.g., Raspberry Pi) and provides pulse output feedback for odometry.

The computational unit, a Raspberry Pi, generates PWM and direction signals for the ZS-X11H motor driver and processes pulse feedback signals from the motors. It also executes high-level control logic, sensor fusion, and system-level communication within the ROS 2 framework, acting as the central processing unit of the robot.

To improve environmental perception, the robot is equipped with an RPLIDAR A1 2D LiDAR sensor. This sensor provides high resolution distance measurements across a 360 degree field of view, enabling the robot to perform crucial tasks such as obstacle detection, SLAM (Simultaneous Localization and Mapping), and autonomous navigation using the ROS 2 navigation stack.

A schematic overview of this full architecture encompassing sensing, computation, actuation, and control layers is shown in Figure 3.1, providing a visual summary of how all modules interact to enable both manual and autonomous operation.

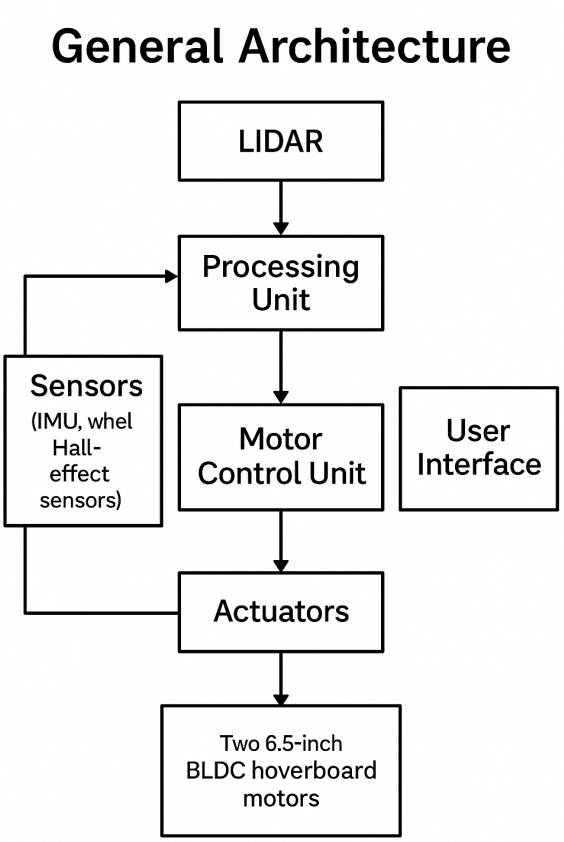


Figure 3.1: General architecture of the mobile base, including the LIDAR sensor.

The architecture comprises the following main components:

- **LIDAR:** An RPLIDAR A1 is used to provide 2D perception of the surrounding environment. It is essential for tasks such as obstacle avoidance, SLAM (Simultaneous Localization and Mapping), and autonomous navigation.
- **Sensors:** The system integrates an Inertial Measurement Unit (IMU) for orientation estimation and Hall-effect sensors embedded in the BLDC motors for wheel odometry. Pulse outputs from the motor drivers provide real-time speed feedback.
- **Processing Unit:** A Raspberry Pi 4 serves as the main processing unit, executing control algorithms, acquiring sensor data, and issuing motor commands. It interfaces with external components via WiFi, GPIO, and USB.
- **Motor Control Unit (MCU):** A dual BLDC motor driver (ZS-X11H) independently controls each motor using PWM and direction signals from the Raspberry Pi. It also outputs pulse feedback signals for odometry computation.
- **Actuators:** The platform uses two 16.5-cm brushless DC hoverboard motors arranged in a differential drive configuration. These motors offer smooth motion and sufficient torque for indoor navigation.
- **Communication Interfaces:** UART is used for serial communication with the Arduino (for odometry feedback), I2C connects the IMU, and USB/Bluetooth/Wi-Fi enable teleoperation, remote debugging, and visualization.

- **User Interface:** Teleoperation is supported via keyboard or joystick. Real-time data visualization and navigation monitoring are handled through tools such as `RViz` in the ROS2 environment.

This control architecture supports real-time feedback, closed-loop motion control, and high-level navigation capabilities, all while utilizing affordable and widely available components.

3.3 Software Frameworks

3.3.1 ROS2

ROS 2 is a widely adopted open-source framework supported by a large and active community. To ensure consistency and maintainability across contributions, ROS 2 enforces a set of development standards and conventions. While these practices are beneficial for large-scale development, they can be challenging for beginners who are just getting started. For example:

- Any modification to the code requires rebuilding the workspace before execution.
- Many files are auto-generated, which can be overwhelming and difficult to understand at first.
- The `colcon build` command must be executed from the root of the workspace; otherwise, the build process and other functionalities may fail.

It is important to note that ROS2 Jazzy is officially supported only on Ubuntu 24.04 LTS. Although it is technically possible to run it on Windows using virtualization tools such as VirtualBox, this approach is not considered professional practice and may lead to performance limitations, compatibility issues, and added setup complexity.

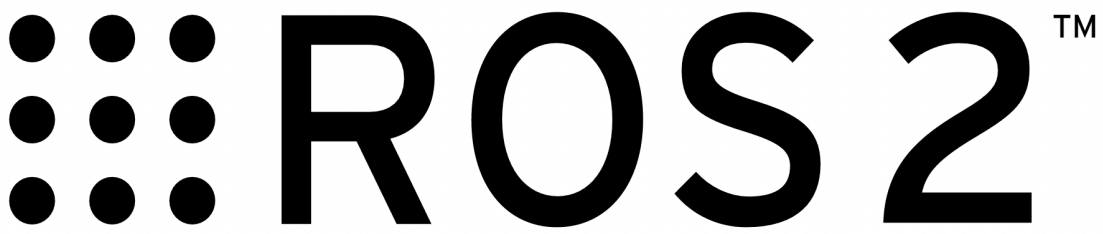


Figure 3.2: ROS2 Logo.

Workspace and Package Structure

To begin any ROS2 project, a workspace must first be created. The typical structure includes:

- **Workspace Root** : A top-level directory (e.g., `pfe`) that contains all packages.
- **src** : A subdirectory inside the workspace where packages are stored.

The following steps create a basic workspace:

```
mkdir -p  /pfe/src
cd  /ros2_ws
colcon build
source install/setup.bash
```

Packages are created inside the `src` directory using either Python or C++ templates:

- **Python package:**

```
ros2 pkg create --build-type ament_python my_python_pkg
```

- **C++ package:**

```
ros2 pkg create --build-type ament_cmake my_cpp_pkg
```

Python vs. C++ Nodes

In ROS2, nodes can be implemented using either Python or C++, each suited for different purposes:

Table 3.1: Comparison between Python and C++ Nodes in ROS 2

Aspect	**Python Nodes**	**C++ Nodes**
Ease of Development	Easier to write and faster to prototype	Requires more setup but offers finer control
Best Suited For	• Sensor integration scripts • High-level control • Debugging and testing	• Real-time or latency-sensitive tasks • Hardware drivers • Image processing and heavy computations
Performance	Suitable for moderate performance tasks	Better suited for high-performance requirements
Syntax	Simpler, interpreted	More complex, compiled
Build System	Uses `ament_python`	Uses `ament_cmake`

Both types of nodes follow the ROS2 lifecycle and communication models (publisher/subscriber, service/client), but their syntax and build systems differ.

Node Communication and Build

Once nodes are written, the workspace must be built:

```
colcon build
source install/setup.bash
```

Then, nodes can be executed using:

```
ros2 run <package_name> <executable_name>
```

ROS2 also allows integration with launch files to start multiple nodes at once.

```
ros2 launch <package_name> <launch_file_name.launch.py>
```

3.4 User Interfaces

3.4.1 Joystick or Keyboard Control

To facilitate manual operation and testing, the robot supports teleoperation using either a joystick or keyboard. The teleoperation node listens for key presses (e.g., `W`, `A`, `S`, `D`) or joystick movements and converts them into velocity commands published on the commands topic.

These commands are consumed by the control node or motor driver node, allowing the operator to:

- Control forward/backward motion and rotation.
- Stop the robot immediately in case of emergency.
- Validate sensor and motor functionality before launching autonomous navigation.

Teleoperation is especially useful during the calibration and debugging phases of the system.

3.4.2 Integration of Control Commands in ROS 2

The robot supports multiple control strategies PID, Sliding Mode Control (SMC), and autonomous navigation using the Nav2 stack. Each method is implemented as a dedicated ROS 2 node and integrated into the overall system through standardized communication interfaces.

PID Control Node

The PID controller is implemented in a Python node `pid_circular_node.py`, located in the `motor_driver` package. This node subscribes to the odometry topic (`/odom`) and publishes velocity commands to the `/demo/cmd_vel` topic.

To run the PID node:

```
ros2 run motor_driver pid_circular_node
```

Sliding Mode Control (SMC) Node

The SMC controller is implemented in the `smc_infinity_node.py`, also in the `motor_driver` package. It performs more robust control, particularly in scenarios with external disturbances or nonlinearities. The node structure is similar to the PID node, using odometry feedback and publishing velocity commands.

To run the SMC node:

```
ros2 run motor_driver smc_infinity_node
```

Navigation Stack (Nav2)

For autonomous operation, the robot uses the ROS 2 Nav2 navigation stack, which includes localization, mapping, and path planning using the A* algorithm. The stack is launched through a launch file that sets up required nodes such as AMCL, planner, controller, and map server.

Each control node can be selectively activated depending on the test scenario, providing a modular and flexible architecture that allows quick switching between manual control (teleoperation), closed-loop control (PID/SMC), and full autonomy (Nav2).

3.4.3 Visualization Tools

Real-time monitoring and visualization of the robot's environment and internal state are performed using RViz, a 3D visualization tool provided with ROS2. It allows developers to:

- Visualize the robot model as shown in Figure 3.3
- Observe live sensor data (e.g., LaserScan, IMU).
- Track the robot's trajectory and estimated pose.
- Debug navigation, mapping, and localization nodes.

RViz is launched on the Raspberry Pi or remotely on a laptop using ROS2 networking. It plays a vital role in validating SLAM and control algorithms during real-world deployment.

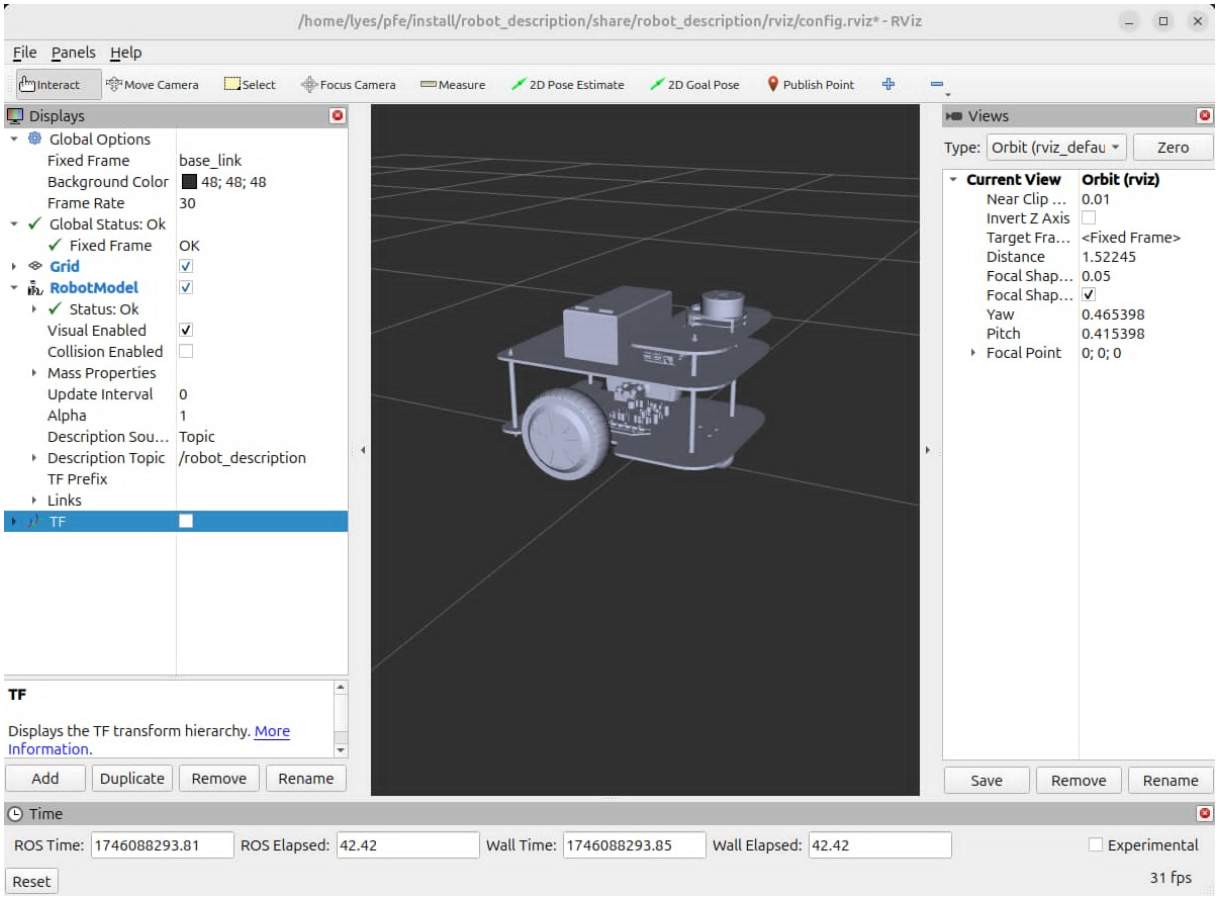


Figure 3.3: Launching the mobile robot's URDF model in ROS

Conclusion

This chapter presented a comprehensive overview of the hardware and software architecture implemented on the mobile robotic platform. The system integrates key components such as the Raspberry Pi 4, RPLIDAR A1, ZS-X11H motor drivers, and various sensors within a modular ROS 2 framework. This architecture supports real time data acquisition, control signal generation, and multi layered decision making.

Multiple control strategies were implemented, including PID and Sliding Mode Control (SMC), alongside full autonomy using the ROS 2 Navigation Stack. Manual operation through joystick or keyboard was also supported for testing and debugging purposes. Visualization and system validation were achieved through tools such as RViz.

The resulting architecture offers a scalable, flexible, and reliable foundation for experimentation, validation of control strategies, and future developments in research and education in mobile robotics.

Chapter 4

Design of the Mobile Base

4.1 Introduction

This chapter outlines the complete design process of the mobile robotic platform, developed as a testbed for control strategy experimentation. It begins with defining functional requirements such as mobility, modularity, and sensor compatibility, followed by the mechanical design considering materials, geometry, and stability. The integration of electrical and electronic components is then detailed, with a focus on the cost effective reuse of a hoverboard motherboard for motor control. Design choices are validated through simulations and CAD modeling, emphasizing modularity to support future enhancements.

4.2 Functional Requirements and Design Specifications

4.2.1 Functional Requirements

The mobile base is designed to serve as a flexible and robust platform for testing a wide range of control techniques. The functional requirements are as follows:

- **Mobility:** The platform must be able to move forward, backward, and rotate in place (differential drive).
- **Control Interface:** It should support manual control via joystick or keyboard, as well as autonomous control through embedded algorithms.
- **Modularity:** The system should allow easy integration of various sensors (e.g., IMU, encoders) and controllers (e.g., microcontroller, Raspberry Pi).
- **Testing Flexibility:** The mobile base should allow implementation and testing of multiple control strategies such as PID, SMC.
- **Stability:** It must maintain stability during movement and when carrying a small payload.
- **Communication:** It should support wired and/or wireless communication (USB, UART, Bluetooth/Wi-Fi).
- **Safety:** Basic safety mechanisms like emergency stop and current protection can be included.

4.2.2 Design Specifications

Based on the functional requirements, the design specifications are summarized in Table 4.1.

Table 4.1: Design Specifications of the Mobile Base

Specification	**Value / Description**
Wheel radius	8.25 cm
Base dimensions	Rectangular shape: 40 cm (length) × 30 cm (width)
Maximum speed of the Motors	Approximately 4–5.5 m/s (adjustable via control)
Power supply	36V Li-ion battery
Motor type	Brushless DC motors (controlled via hoverboard motherboard)
Payload capacity	300 kg
Material	Alucobond and Aluminum (density: 2700 kg/m^3)
Communication interfaces	UART, USB, Bluetooth/Wi-Fi

4.3 Mechanical Design

In this section, we detail the mechanical aspects of the mobile platform, focusing on the chassis design, wheel configuration, and material selection. These elements were carefully engineered to ensure mechanical robustness, functional adaptability, and ease of integration with electronic and control systems.

4.3.1 Chassis Design

The mobile base is built on a rectangular chassis with dimensions of 40 cm × 30 cm. The design includes mounting slots for two rear wheels, a caster wheel at the front for balance, and compartments for housing the battery and control board.

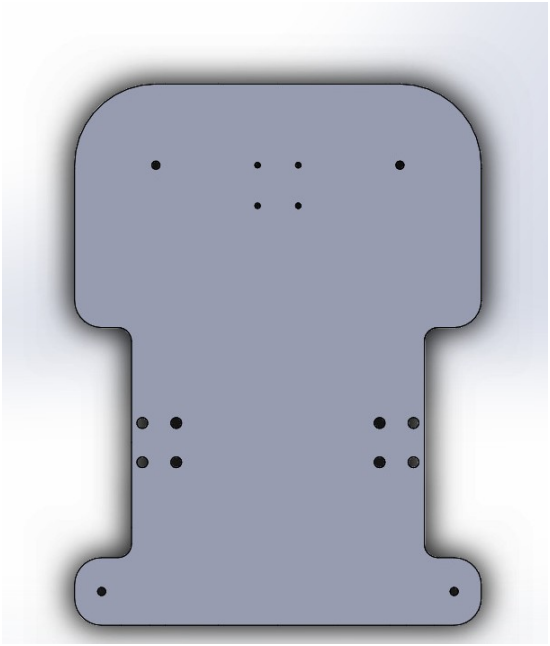


Figure 4.1: 3D CAD Model of the Mobile Base Chassis

4.3.2 Wheel Configuration

The robot uses a differential drive system with two independently controlled rear wheels and one passive caster wheel at the front for stability. Each wheel has a radius of 8.25 cm and is directly coupled to a brushless DC motor.

Figure 4.2: 3D CAD rear wheels.

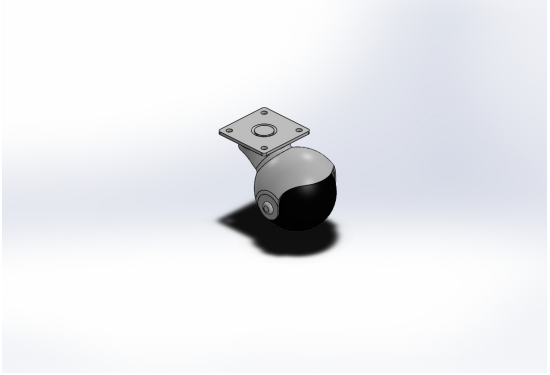


Figure 4.3: 3D CAD Roda Caster .

4.3.3 Material Selection

The chassis and supporting components were fabricated using aluminum for the base and Alucobond for the other two stages, selected for their superior strength-to-weight ratio, durability, and structural stability. Aluminum, with a density of approximately 2700 kg/m^3, provides a robust foundation for the robot's payload and movement requirements, while Alucobond offers enhanced mechanical properties for the subsequent stages, ensuring optimal performance under operational stresses. This material choice improves rigidity and longevity compared to polymer-based alternatives, making it well-suited for demanding robotic applications.

4.4 Structural Analysis of the Chassis

A static structural analysis was performed to verify the mechanical integrity of the chassis under expected operational loads. The study considered the weight of mounted wheels and potential shock forces due to movement or uneven terrain.

The chassis, made of 6mm aluminum sheet, was analyzed under a total load of 3000 N, corresponding to the combined weight of the platform, motors, and control electronics. Fixed constraints were applied at the mounting points of the wheels and caster.

The simulation was carried out using SolidWorks Simulation with a mesh refinement strategy appropriate for thin plate structures.

The structural simulation yielded a maximum displacement of approximately 1.66 mm at the chassis edges, which is acceptable for the application. Strain levels remained under 3.8×10^{-3}, with concentrations near load application points. The peak Von Mises stress reached only 2.12×10^4 Pa, significantly below the aluminum 6061-T6 yield limit of 275 MPa. These results confirm that the chassis design is structurally robust, with a high safety factor and sufficient mechanical rigidity to withstand static and dynamic loads during robot operation.

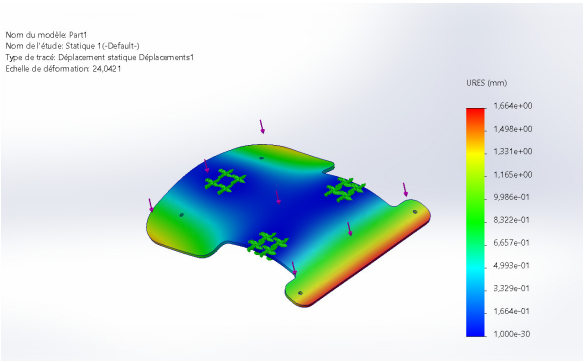


Figure 4.4: Maximum displacement of the chassis.

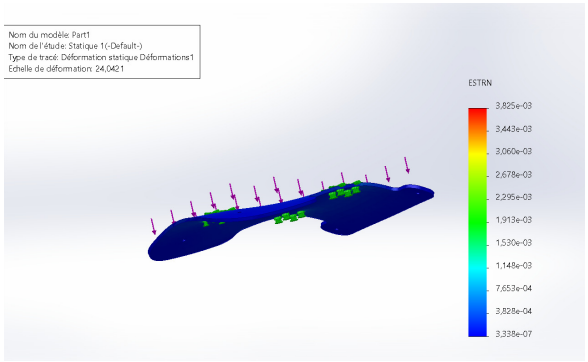


Figure 4.5: Maximum deformation of the chassis.

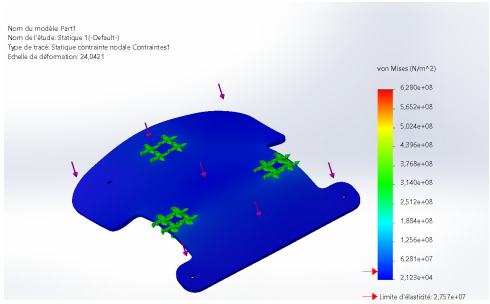


Figure 4.6: Von Mises stress distribution under load.

4.5 Electrical Design

In this section, we examine the electrical architecture of the mobile platform, covering the power system, motor and driver selection, as well as the integration of sensors and feedback mechanisms. The electrical components were chosen and configured to ensure reliable energy delivery, efficient motor control, and accurate feedback acquisition, all of which are essential for precise navigation and control implementation.

4.5.1 Power System

The robot is powered by a **36V lithium-ion (Li-ion) battery pack**, selected for its high energy density (∼250 Wh/kg), lightweight properties, and ability to sustain high current demands. This voltage rating is compatible with the **hoverboard motors** used for locomotion, which operate efficiently at 36V and provide high torque for wheel-driven mobility.

Motor Specifications:

The hoverboard motors are **brushless DC (BLDC) hub motors**, chosen for their:

- **High Torque Output**: Enables smooth acceleration and load-bearing capability.
- **Integrated Design**: Combines motor and wheel into a compact unit, simplifying mechanical assembly.
- **Regenerative Braking**: Recovers kinetic energy during deceleration, improving power efficiency.
- **Low Maintenance**: Brushless design reduces wear compared to brushed motors.

Table 4.2: Hoverboard Motor Specifications

Parameter	**Value**
Rated Voltage	36V
Maximum Power	350W
No-load Speed	300 RPM
Rated Torque	2.5 Nm

Figure 4.7 illustrates the complete wiring configuration of the mobile robotic base. The Raspberry Pi 4 serves as the central processing unit, controlling both hoverboard motors through two ZS-X11H motor drivers via PWM and DIR signals. An Arduino Uno is connected to the motor drivers to read the encoder pulse outputs (S pins) and motor direction signals, which are processed to calculate RPMs and sent to the Raspberry Pi via USB serial communication. The IMU (GY-87) is interfaced to the Arduino through I^2C lines. A 36V Li-ion battery pack provides direct power to the motor drivers and motors, while logic-level components such as the Raspberry Pi and Arduino are powered via regulated supply lines. The entire wiring is routed through a central breadboard for organized signal distribution.

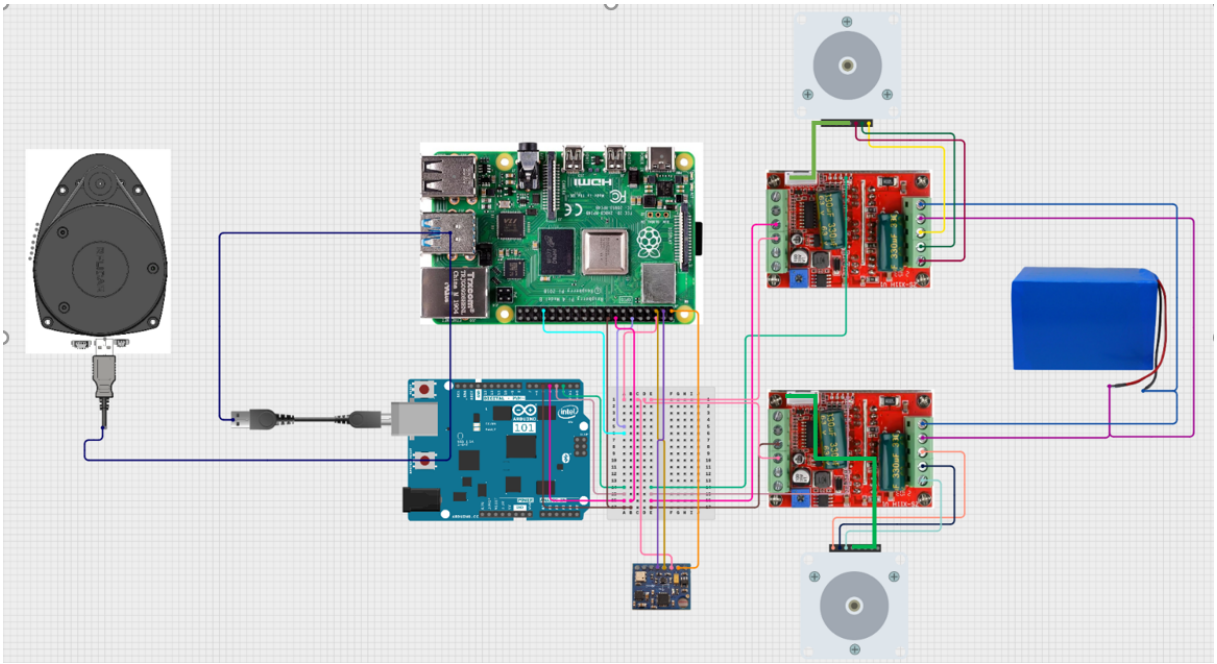


Figure 4.7: Wiring diagram of the electronic components in the mobile robotic base.

4.5.2 Electronic Board

BLDC Motor Driver Integration

To control the brushless DC (BLDC) hub motors of the mobile platform, a commercially available 3-phase Hall Brushless Motor Controller rated for 6–60V and 400W was employed. This compact driver module is well-suited for hoverboard-type motors and supports Hall sensor-based commutation, offering smooth torque output and direction control.

The driver accepts control signals in the form of standard PWM pulses (typically 1–2 ms, as in RC servos), making it compatible with a wide range of microcontrollers and single-board computers. It includes dedicated terminals for Hall sensor feedback, direction selection, braking, and mode configuration. Its built-in features allow for:

- **Forward/Reverse direction control**
- **PWM speed control (1–2 ms signal)**
- **Brake control input**
- **Support for Hall-effect sensors** to enable closed-loop commutation

Its voltage and current ratings (up to 60V and 16A per motor) are ideal for typical hoverboard motors used in differential drive platforms.

Integration Approach

Each BLDC motor is connected to a separate driver module, allowing for full differential control of the mobile base. The Hall-effect sensors embedded in each motor are connected to the corresponding feedback terminals of the driver, enabling the driver to compute rotor position and apply the correct commutation sequence.

The Raspberry Pi 4 generates PWM signals corresponding to desired motor speeds using GPIO pins. Direction control and braking are managed via additional GPIO digital outputs. The control architecture is designed such that the Raspberry Pi calculates motor velocity commands

based on joystick input, sensor feedback, or high-level navigation goals, and then generates the required PWM signals.

This integration strategy provides a clean separation between high-level control (on the Raspberry Pi) and low-level actuation (on the driver), allowing advanced algorithms like PID or MPC to be applied without modifying the motor driver hardware. The driver effectively acts as an analog actuator interface, translating PWM signals into efficient motor control actions.

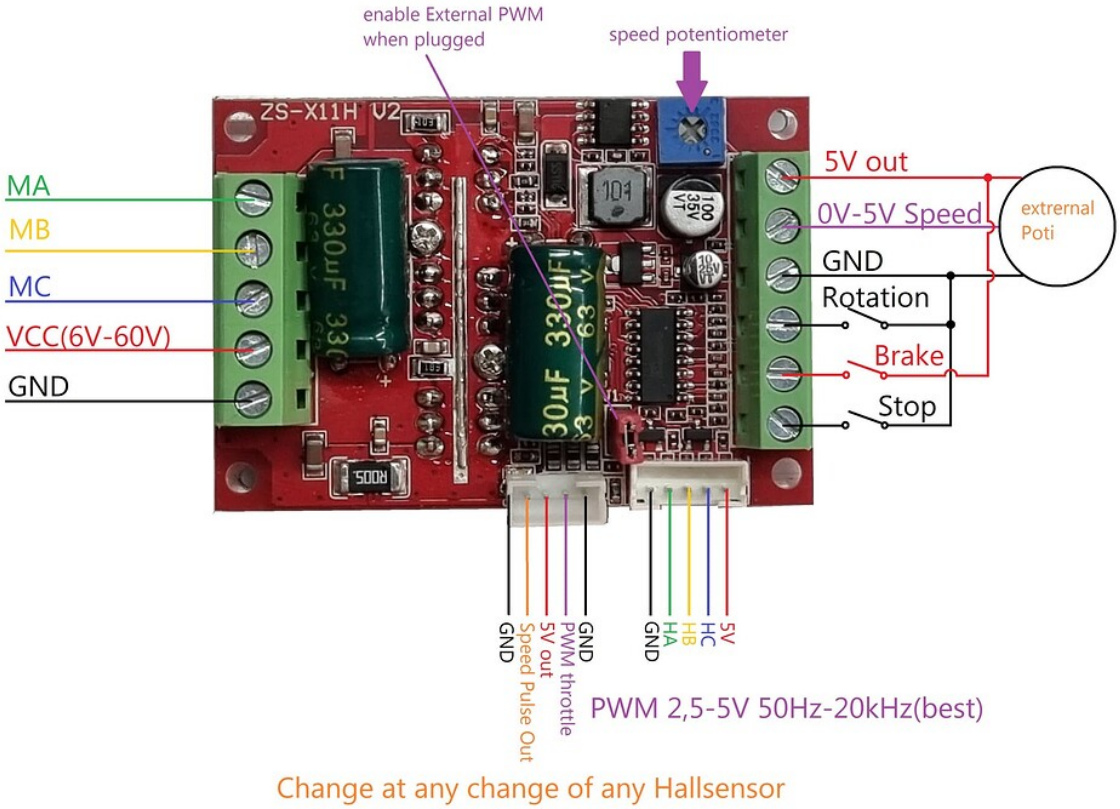


Figure 4.8: Integration of the 3-Phase Hall BLDC Driver with Raspberry Pi and Motors

4.6 Safety Considerations and Constraints

Safety was a central concern in the design and integration of the mobile platform. This section outlines the measures implemented to minimize electrical hazards, overheating, mechanical instability, and control anomalies, particularly with the integration of external BLDC motor drivers.

4.6.1 Electrical Safety

The system incorporates several safety mechanisms to ensure secure power distribution and motor control:

- External BLDC motor drivers include protection features such as overcurrent shutdown and thermal cut-off.
- Voltage regulation is achieved using buck converters to step down 36V to 5V and 3.3V for low-voltage control electronics.

- All electrical wiring is insulated, organized, and routed to minimize short circuits and electrical noise.
- An inline fuse is added at the power input to protect against overcurrent conditions (recommended).

4.6.2 Thermal Management

Thanks to the open-frame architecture of the platform, effective passive cooling is naturally achieved:

- Continuous airflow is enabled by the exposed and ventilated design of the chassis.
- The external motor drivers are placed with sufficient spacing to avoid thermal buildup.
- Heat sinks may be applied to driver MOSFETs or voltage regulators to enhance heat dissipation during high-load operation.

4.6.3 Mechanical Stability

Mechanical safety is ensured by structural reinforcement and careful component mounting:

- The aluminum chassis is designed to evenly distribute weight and minimize stress concentration points.
- All electronic modules and batteries are securely fixed using brackets and vibration-isolating mounts to avoid dislocation during motion.

4.6.4 Control and Operational Safety

Several software-level protections were implemented to avoid unintended behavior:

- Speed limits and acceleration constraints are enforced via control algorithms on the Raspberry Pi.
- A manual emergency stop function is mapped to the keyboard to allow quick interruption of motion.
- A software watchdog continuously monitors communication between the control unit and motor drivers to prevent uncontrolled behavior in case of signal loss.

4.6.5 Power Source Management

The power system is managed with safety and reliability in mind:

- The robot operates from a regulated 36V power supply, compatible with all major subsystems.
- All battery terminals and connectors are properly insulated and mechanically secured.
- Optional integration of a Battery Management System (BMS) can provide additional protection and status monitoring.

4.7 CAD Models and Simulations

Computer-Aided Design (CAD) and simulation tools were used to validate the mechanical design and evaluate the system before physical prototyping.

4.7.1 CAD Modeling

The mechanical structure of the mobile base was modeled using [**SOLIDWORKS**] . The model includes the chassis, wheel mounts, motor brackets, and enclosures for electronics.

4.7.2 Assembly and Exploded View

An exploded view was created to visualize component placement and assembly procedures.

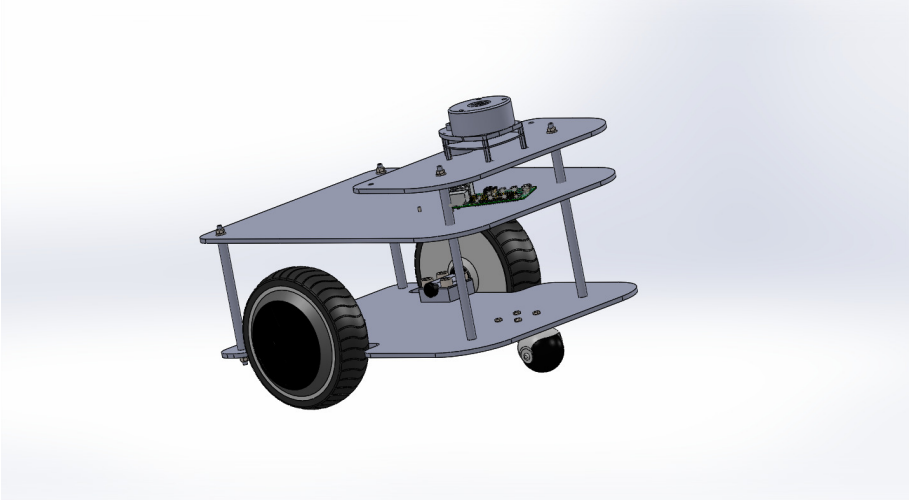


Figure 4.9: Exploded view of the mechanical assembly

4.7.3 Mass and Balance Analysis

The center of mass was estimated to ensure stability and prevent tipping during operation. The symmetry and low profile of the base contribute to its mechanical robustness.

4.7.4 Motion Simulation

The movement and turning behavior of the platform were simulated in [**Gazebo**] . Simulations validated:

- Torque requirements under various payloads.
- Stability and traction during motion.
- Rotational response for different wheel configurations.

4.7.5 Structural Analysis

A basic structural Finite Element Analysis (FEA) was conducted to identify stress points and verify that the chassis can handle expected forces during operation.

Table 4.3: Estimated Weight of Mobile Robot Components

Component	**Description / Assumptions**	**Estimated Mass (kg)**
Aluminum Chassis	40 × 30 cm, 6 mm thick sheet	1.94
BLDC Motors (2×)	6.5-inch hoverboard motors	6.000
Battery	36V Li-ion pack (approx. 4.4–6.6 Ah)	1.500
Motor Drivers	2 external BLDC drivers (PWM-controlled)	0.200
Raspberry Pi 4	Pi + case	0.100
Sensors	IMU + connectors	0.050
Power Electronics	Buck converters, relay, wiring, fuses	0.150
Caster Wheel	Front balancing wheel (metal or rubber)	0.300
Mounting Hardware	Standoffs, brackets, screws	1.250
Total Estimated Weight		**11.49**

Conclusion

This chapter detailed the multidisciplinary design of a mobile robotic platform structured for flexibility, robustness, and experimental control development. Starting from clearly defined functional requirements, the system was engineered with a differential drive configuration using two 6.5-inch hoverboard BLDC motors, providing high torque and stability. The mechanical design, based on aluminum and Alico materials, was validated using CAD modeling and static structural simulations to ensure mechanical integrity under operational loads.

The electrical system was built around a 36V Li-ion battery pack powering independent motor drivers (ZS-X11H), with control and feedback managed by a Raspberry Pi 4 and an Arduino Uno. Sensor integration included IMU and LIDAR systems to support autonomous navigation and mapping in ROS 2. A comprehensive wiring diagram and modular component layout ensured safe operation, ease of debugging, and future expandability.

Through this layered architecture comprising sensing, computation, actuation, and communication the mobile base is well-suited for testing a variety of control strategies, from manual teleoperation to fully autonomous navigation. The system is designed for rapid prototyping, educational experimentation, and applied robotics research.

Chapter 5

Experimental Results and Simulation

5.1 Introduction

Before real-world tests were performed, simulations were conducted to validate the mechanical model, control algorithms, and sensor integration. These simulations ensured the stability and performance of the system under various operating conditions. The simulation environment combined ROS2 **Jazzy** with **Gazebo** to provide a realistic testbed that mimics the behavior and environment of the actual robot.

5.2 Simulation Study

For a better understanding of our simulation project, the directory structure of our `PFE` workspace is illustrated in Annex for further reference and reproduction.

5.2.1 Controller Evaluation: PID vs. SMC on Trajectories

This section presents a detailed evaluation of two controllers PID and Sliding Mode Control (SMC) applied to two reference trajectories: a circular path and an infinity-shaped path. The simulation environment was configured with the path centered at coordinates (1,1) and a radius of 1 meter. Each controller was tested with specific gain parameters, and performance was analyzed based on tracking error, overshoot, convergence, and velocity behavior.

PID Controller Evaluation

Trajectory: Circular Path The circular trajectory is defined parametrically as:

$$\begin{aligned} x_d(t) &= x_c + R \cdot \cos(\omega t) \\ y_d(t) &= y_c + R \cdot \sin(\omega t) \end{aligned} \tag{5.1}$$

where:

$$\omega = \frac{v_{\max}}{R} \quad \text{and} \quad (x_c, y_c) = (1,1),\ R = 1$$

PID Gains Used The PID controller parameters used for stable tracking are:

Table 5.1: PID Controller Parameters for Circular Trajectory

Parameter	Value
K_p^{lin}	1.2
K_i^{lin}	0.03
K_d^{lin}	0.3
K_p^{ang}	0.6
K_i^{ang}	0.07
K_d^{ang}	0.05
K_{cte}	0.6

Simulation Results

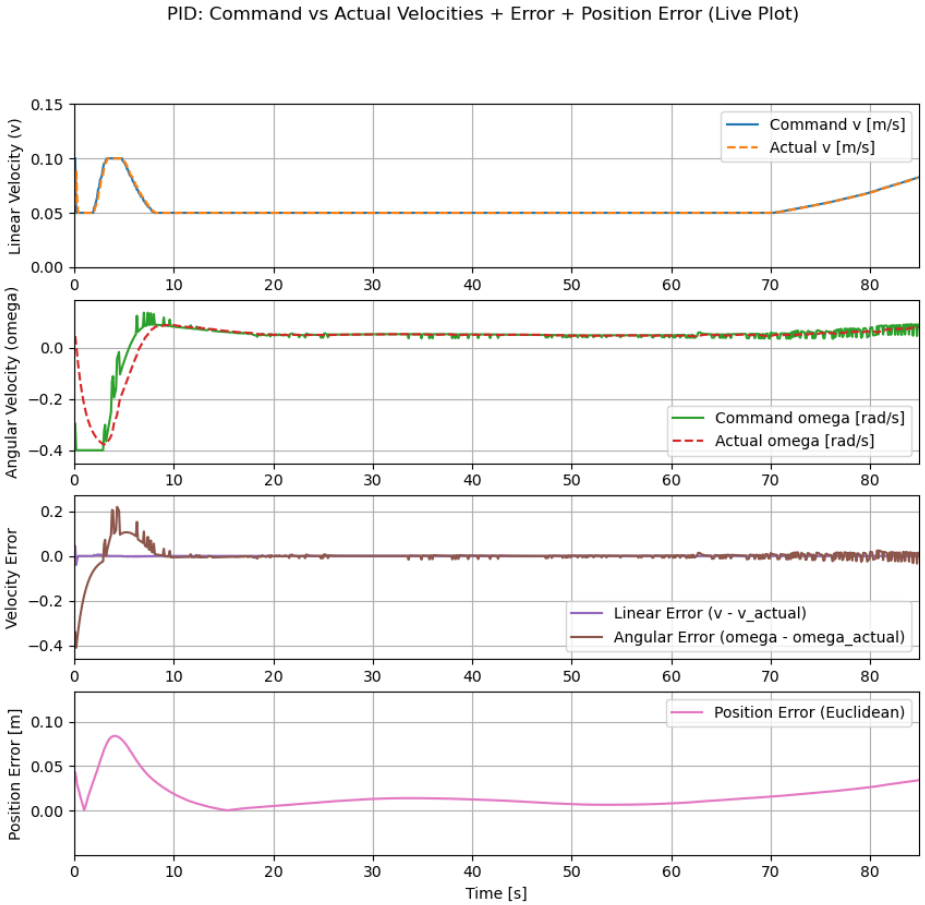


Figure 5.1: PID controller for the circular trajectory tracking.

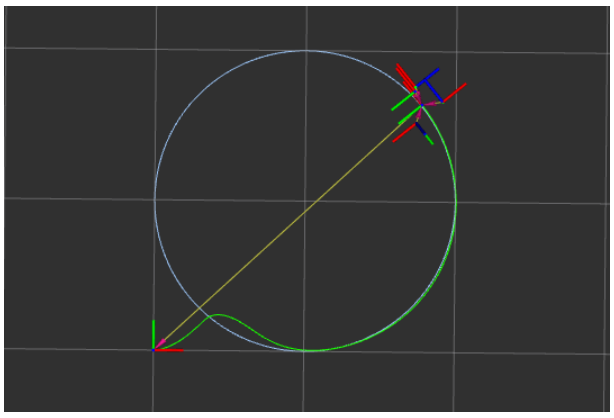


Figure 5.2: Visualization of the circular path tracked by the robot in simulation (PID controller).

Performance Interpretation. Figure 5.1 shows accurate tracking of both linear and angular velocities, with smooth responses and minimal steady-state error. The plot also illustrates how the position and velocity errors converge rapidly after initial transients.

Figure 5.2 confirms that the robot closely follows the planned circular trajectory, with the mean position error remaining below 5 cm throughout the experiment. The results demonstrate

the effectiveness of the tuned PID controller for reliable and repeatable path following.

Trajectory: Infinity Path The reference infinity-shaped trajectory is generated parametrically as:

$$\begin{aligned} x_d(t) &= x_c + a \cdot \sin(\omega t) \\ y_d(t) &= y_c + b \cdot \sin(2\omega t) \end{aligned} \tag{5.2}$$

where:

$$a = 1, \quad b = 0.5, \quad (x_c, y_c) = (1,1), \quad \omega = \frac{2\pi}{T}, \quad T = 40\text{s}$$

PID Gains Used The PID parameters were selected for robust trajectory tracking based on experimental tuning:

Table 5.2: PID Parameters for Infinity Trajectory

Parameter	Value
K_p^{lin}	1.2
K_i^{lin}	0.0
K_d^{lin}	0.1
K_p^{ang}	1.5
K_i^{ang}	0.0
K_d^{ang}	0.2

Simulation Results

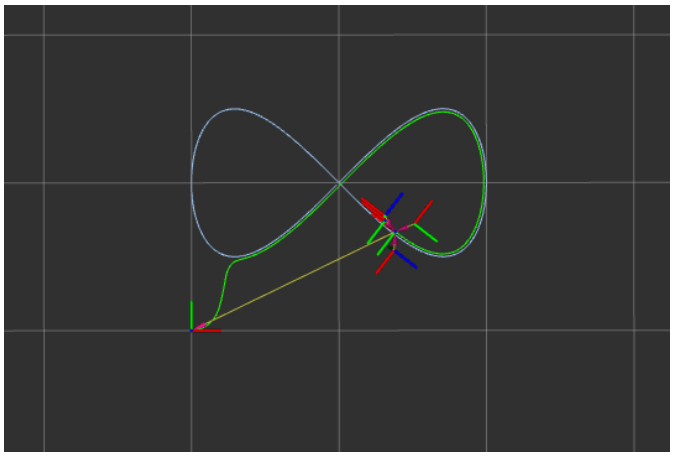


Figure 5.3: Visualization of the infinity-shaped trajectory tracked by the robot in simulation (PID control).

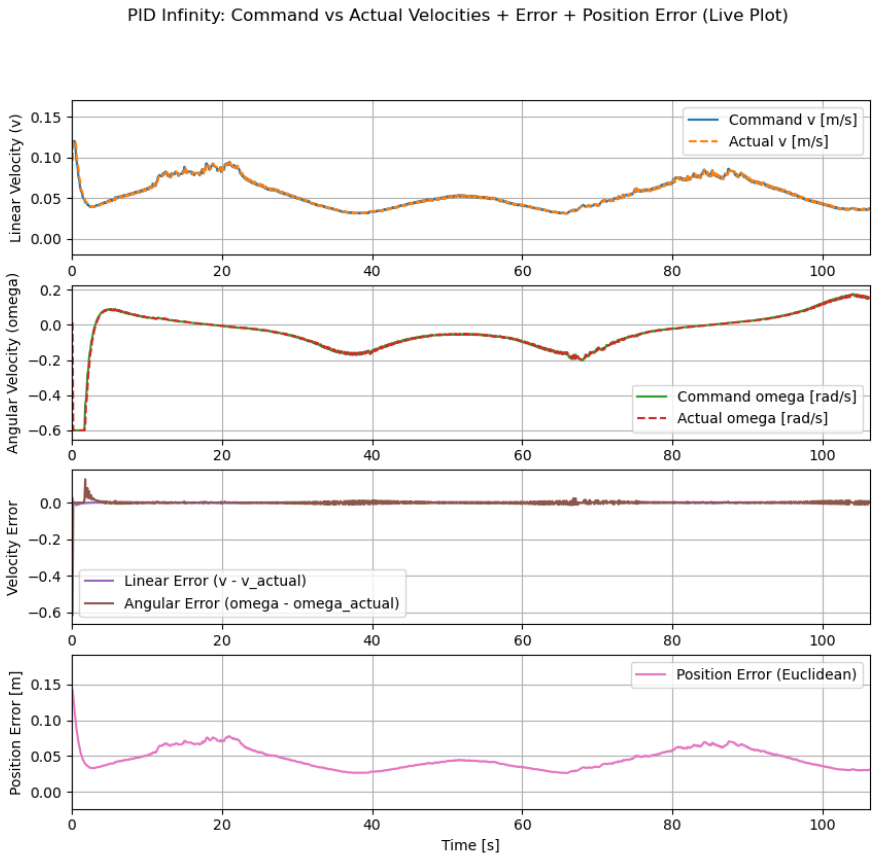


Figure 5.4: PID controller for the infinity trajectory tracking.

Performance Interpretation. Figure 5.4 demonstrates that the PID controller ensures accurate tracking of both linear and angular velocities, with position and velocity errors converging rapidly after initial transients.

Figure 5.3 shows that the robot closely follows the reference infinity trajectory, with a typical mean error below 10 cm, validating the effectiveness of the controller for complex path following.

SMC Controller Evaluation

Trajectory: Circular Path The reference trajectory is defined as:

$$\begin{aligned} x_d(t) &= x_c + R \cdot \cos(\omega t) \\ y_d(t) &= y_c + R \cdot \sin(\omega t) \end{aligned} \tag{5.3}$$

where $(x_c, y_c) = (1, 1)$ is the circle center, $R = 1\,\text{m}$ is the radius, and $\omega = \frac{2\pi}{T}$ with $T = 40\,\text{s}$ is the period of the trajectory.

SMC Gains Used The Sliding Mode Controller (SMC) was configured with the following parameters:

Table 5.3: SMC Controller Parameters for Circular Trajectory

Parameter	Value
λ_θ	1.0
k_θ	2.0
k_p	10.0

Simulation Results

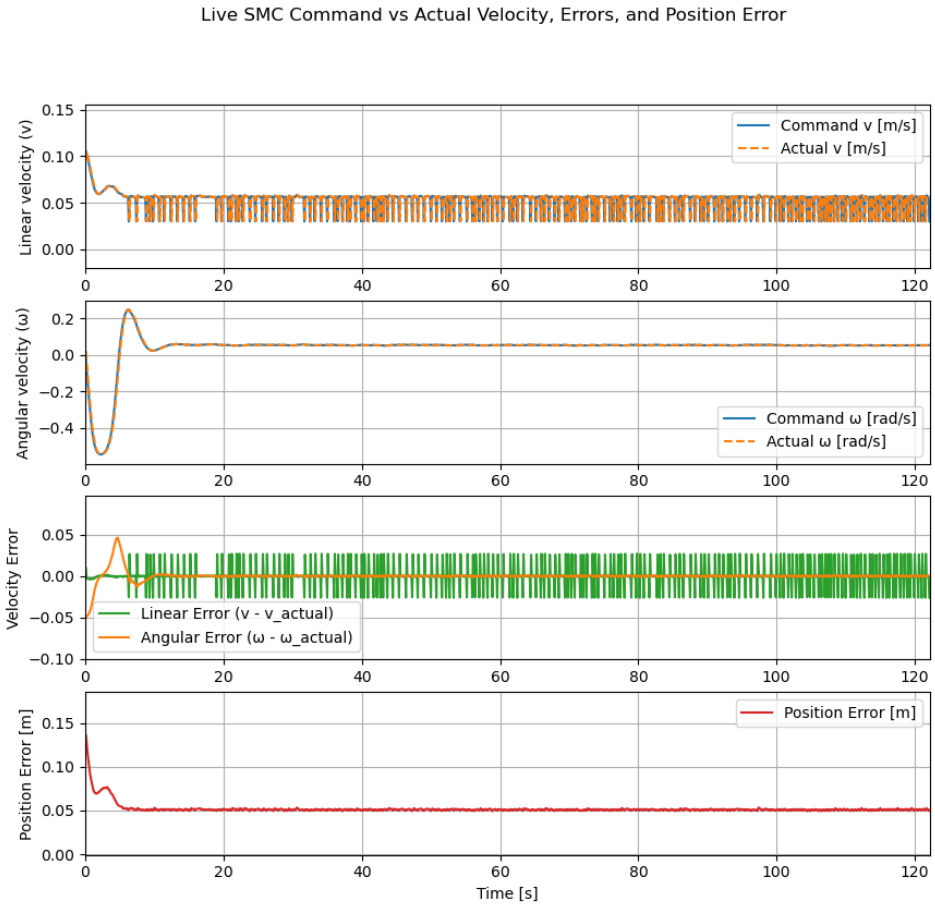


Figure 5.5: SMC controller for the circular trajectory tracking.

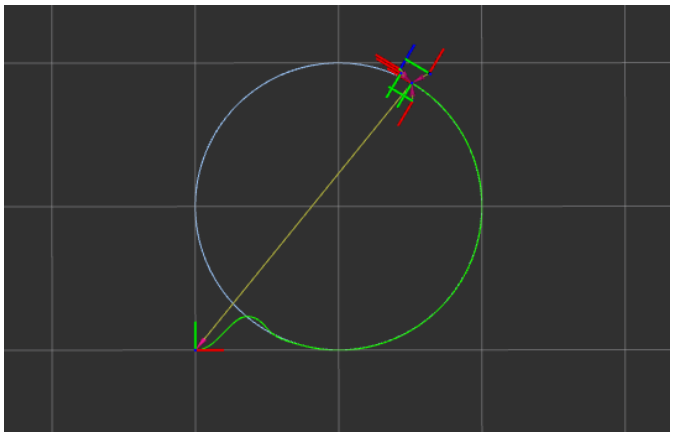


Figure 5.6: Visualization of the circular path tracked by the robot in simulation (SMC controller).

Performance Interpretation. Figure 5.5 demonstrates that the SMC controller provides robust tracking of both linear and angular velocities, with rapid convergence and low steady-state error. The position error plot shows that, after initial transients, the robot's path error

quickly decreases and remains within a narrow bounded range.

Figure 5.6 shows that the robot successfully follows the desired circular trajectory under the SMC control law, with slight deviations mainly during the transient phase. These results highlight the robustness of the SMC controller for circular path tracking, even in the presence of model uncertainties or external disturbances.

Trajectory: Infinity Path The reference trajectory for the infinity shape is defined as:

$$\begin{aligned} x_d(t) &= x_c + a \cdot \sin(\omega t) \\ y_d(t) &= y_c + b \cdot \sin(2\omega t) \end{aligned} \tag{5.4}$$

with:

$$(x_c, y_c) = (1, 1), \quad a = 1.0, \quad b = 0.5, \quad \omega = \frac{2\pi}{T}$$

where T is the period for a full loop.

SMC Gains Used The Sliding-Mode Controller was configured with the following parameters:

Table 5.4: SMC Controller Parameters for Infinity Trajectory

Parameter	Value
λ_theta	2.0
k_s	4.0
k_p	6.0

Simulation Results

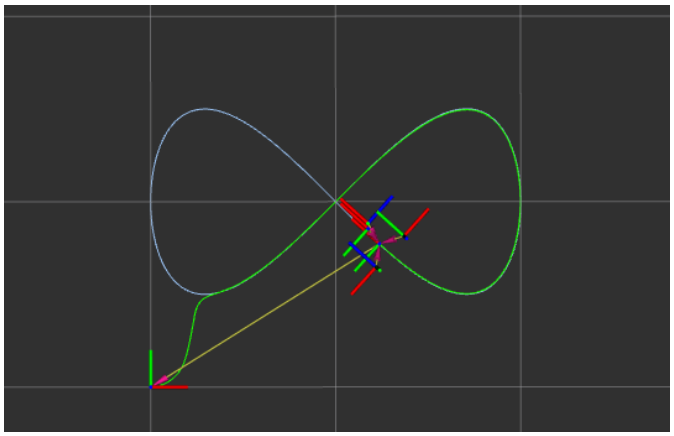


Figure 5.7: Visualization of the infinity path tracked by the robot in simulation (SMC controller).

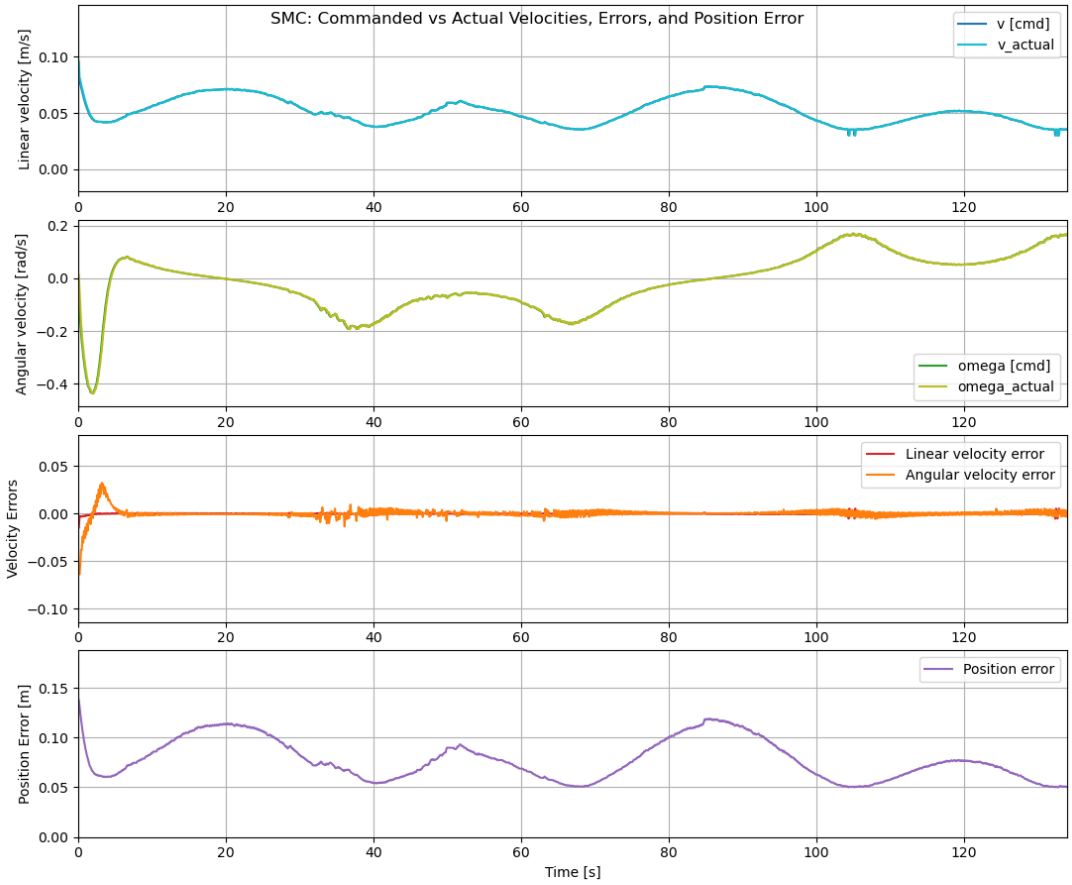


Figure 5.8: SMC controller for infinity trajectory tracking.

Performance Interpretation. Figure 5.8 shows that the SMC controller ensures robust tracking for both linear and angular velocities, with errors rapidly converging towards zero after initial transients. The position error remains bounded throughout the task, confirming the controller's effectiveness in handling the challenging trajectory.

Figure 5.7 visually confirms that the robot accurately follows the desired infinity path, with only minor deviations. This highlights the robustness of the SMC controller for complex trajectory tracking in the presence of modeling errors and external disturbances.

Comparative Performance Table: SMC vs PID Controllers (Simulation Results)

Table 5.5: Empirical Performance Comparison: SMC vs PID Controllers

Controller	Trajectory	Overshoot	Steady-State Error	Tracking Smoothness
SMC	Circular	Moderate	$\sim$ 2–3 cm	Moderate
SMC	Infinity	Noticeable	$\sim$ 4–6 cm	Oscillatory
PID	Circular	Very Low	$<$ 2 cm	Excellent
PID	Infinity	Low	$<$ 3 cm	Good

Interpretation: The experimental results indicate that the PID controller achieves better overall trajectory tracking than the SMC controller in these experiments:

- **PID Controller:** Demonstrates low overshoot, fast settling time, and minimal steady-state error. The tracking curves are notably smooth and closely follow the reference

trajectory, both for circular and infinity paths. Position error remains low and stable after the initial transient.

- **SMC Controller:** Shows some oscillation, especially on the infinity path, and exhibits a slightly higher steady-state error. While SMC provides robustness in theory, practical tuning challenges or system noise can result in less smooth tracking compared to PID.
- **Comparison:** For your system and tuning, the PID controller is clearly preferable, providing higher tracking accuracy and smoothness. The SMC controller might require further tuning of its gains to reach comparable performance, particularly on complex trajectories.

Conclusion: The PID controller provides superior tracking precision and smoothness compared to SMC. However, SMC retains its value in scenarios with high disturbance or model uncertainty, where its inherent robustness can become advantageous.

5.2.2 Navigation Stack Integration and Mapping Validation

To validate the robot's autonomous navigation capabilities, the SLAM Toolbox was integrated into the simulation environment to enable real-time mapping and localization. The robot was deployed in various simulated worlds created using **Gazebo**, including narrow corridors and structured indoor scenes.

Mapping Phase: The robot was manually teleoperated using a wireless joystick via the `/teleop_manete` package. As the robot moved, the SLAM Toolbox processed the laser scan data from the simulated LiDAR to incrementally build an occupancy grid map. This allowed real-time mapping without prior knowledge of the environment.

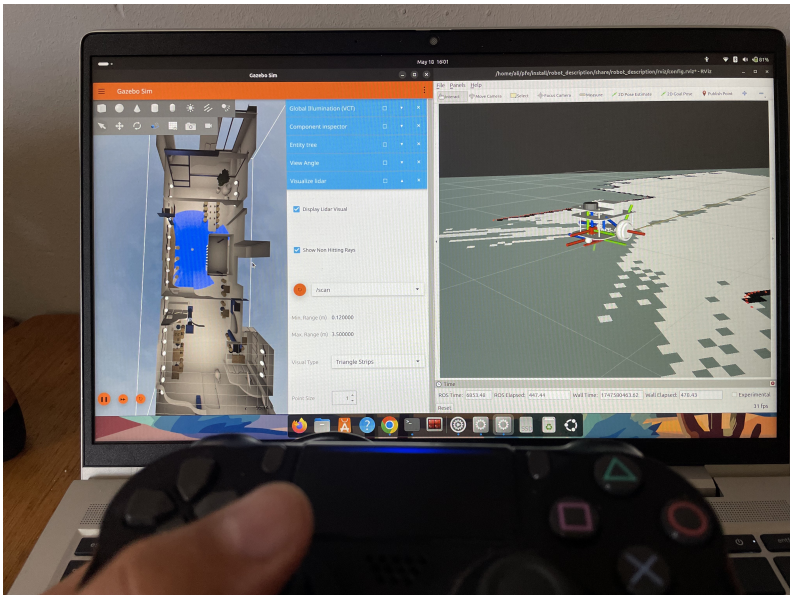


Figure 5.9: SLAM-based mapping using joystick in a Gazebo environment.

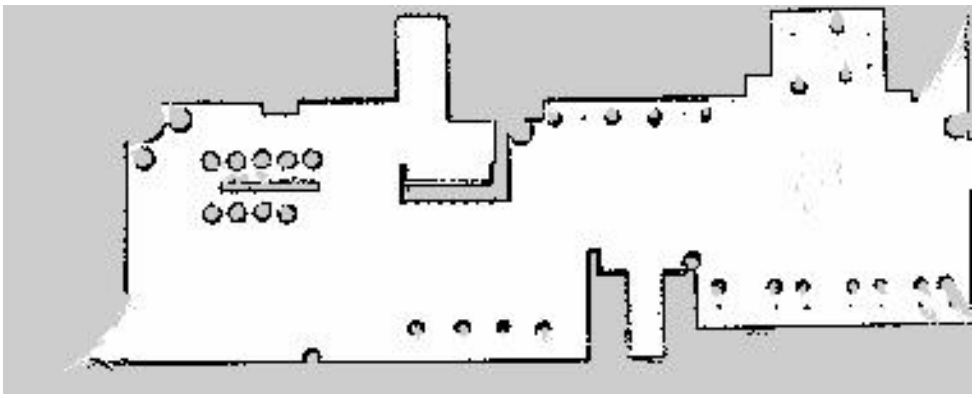


Figure 5.10: map created in real-time using SLAM Toolbox.

World Used in Simulation: The environment used for testing was a Gazebo world designed to mimic indoor corridors. It included walls, obstacles, and free space suitable for both mapping and navigation.

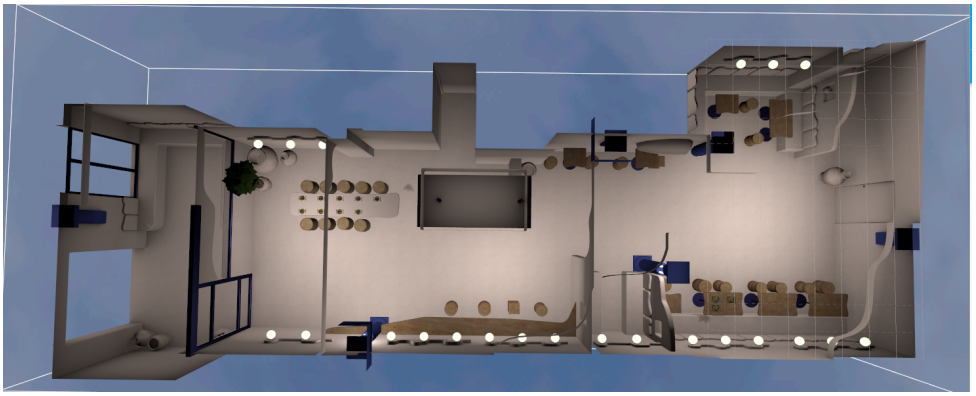


Figure 5.11: Simulated environment in Gazebo used for SLAM-based navigation tests.

Mapping Behavior: The robot started from position (0,0) and gradually explored the environment under manual control. The SLAM node published the `/map` topic and continuously updated the map in RViz. Once exploration was complete, the map was saved using the `map_saver` utility provided by Nav2.

This mapping process was essential to test and visualize the accuracy of the simulated sensor setup and robot kinematics prior to autonomous navigation experiments.

Nav2 Stack: After generating the map from our simulated environment, we launched the Nav2 stack and initiated the navigation process using RViz. This involves first setting the robot's initial pose (start position) and then selecting a goal pose that defines the destination. The global planner (typically A*) computes a path on the previously generated map, and the local planner ensures the robot follows it while avoiding obstacles. An example of this process is illustrated in Figure 5.12.

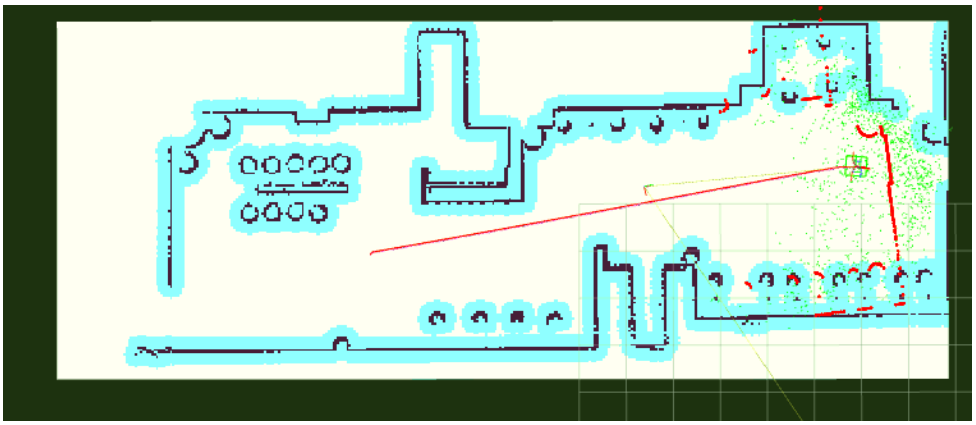
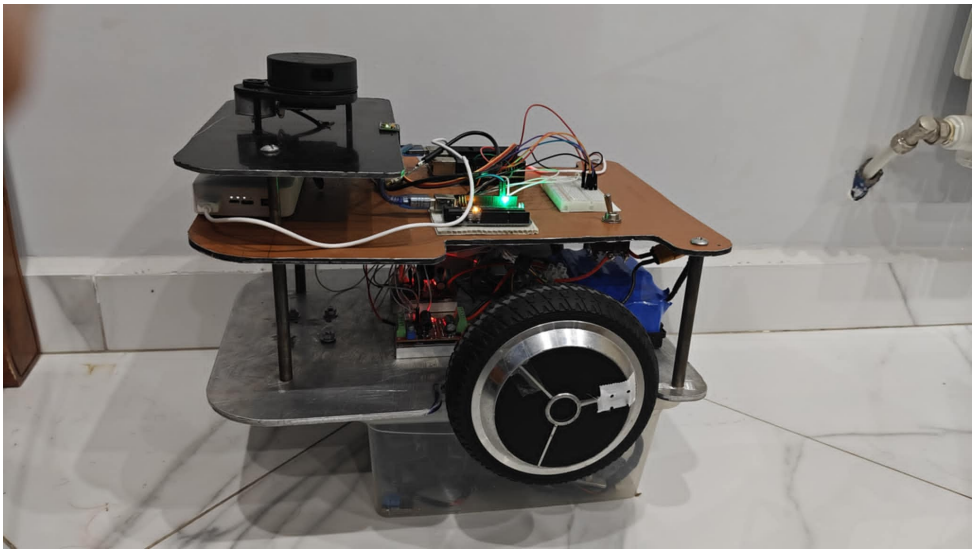


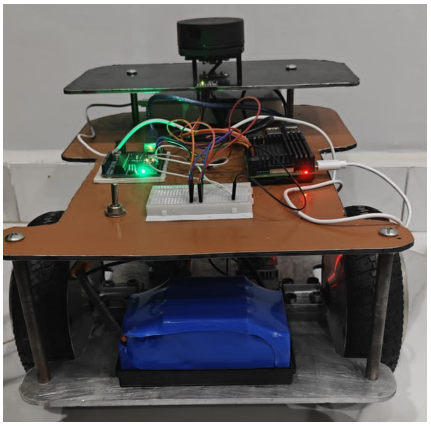
Figure 5.12: Navigation test in RViz.

5.3 Real-World Validation and Odometry Challenges

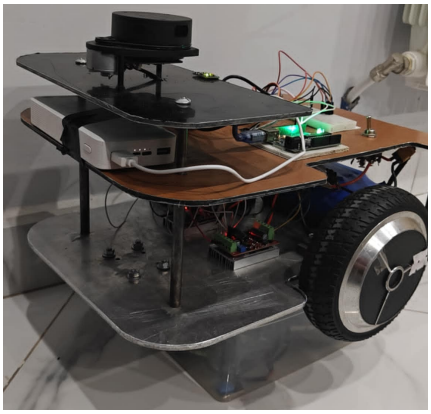
In this validation, we used the robot designed in Figure 5.13, which was specifically developed and built as part of this project.



Rear view



Back view



Top view

Figure 5.13: Different views of the realized robotic platform used for experimental validation.

The experimental evaluation was structured in three main phases: initial manual teleoperation, systematic controller testing and autonomous navigation.

5.3.1 Odometry Error and IMU-Based Solution

During experimental validation, significant odometry drift was observed, leading to accumulated errors in encoder-based odometry. These errors caused positional inaccuracies that grew over time, especially during prolonged or complex trajectories. This drift became particularly evident during autonomous navigation tasks, impacting the robot's ability to accurately follow the desired path.

To overcome this challenge, a simplified strategy was implemented: using orientation data from the Inertial Measurement Unit (IMU) to provide the robot's rotation (heading) value for odometry. By directly extracting real-time orientation measurements from the IMU, heading drift was significantly reduced during trajectory tracking. This approach allowed the robot to rely on the IMU for heading estimation instead of solely on wheel encoder-based odometry.

This modification was particularly useful in reducing the effects of odometry drift, and it enabled more accurate and reliable path following in real-world tests. The IMU's real-time orientation data was incorporated into the odometry calculation, without fusion from encoder-derived angles, allowing for a direct evaluation of the IMU's impact on navigation accuracy.

5.3.2 Manual Teleoperation

Initially, the robot was operated manually using a wireless joystick interface. This allowed us to verify the correct response of the actuators, validate the wiring and software stack, and ensure that the robot could safely execute basic motion commands (forward, reverse, turns, and stops). Manual control was essential for performing initial trajectory tests, such as following circular and infinity-shaped paths, while closely observing the response of the odometry and IMU feedback in real-world conditions.

5.3.3 Controller Validation: PID and SMC

To rigorously evaluate the tracking performance of our control strategies, both the PID and Sliding Mode Control (SMC) algorithms were tested on the real robot for two representative trajectories: a circular path and an infinity-shaped path. For each controller, the robot was commanded to follow the respective reference trajectory, and its actual path was recorded and compared to the desired reference. The controller evaluation focused on analyzing tracking accuracy, robustness to disturbances, and the convergence behavior of position and velocity errors.

Performance was visualized by plotting both the commanded and actual paths, alongside error plots. This dual-trajectory, dual-controller approach provided a comprehensive validation of the control methods under real-world conditions.

PID Controller Validation

The PID controller was tested on a circular trajectory, and the results were visualized in the following figures.

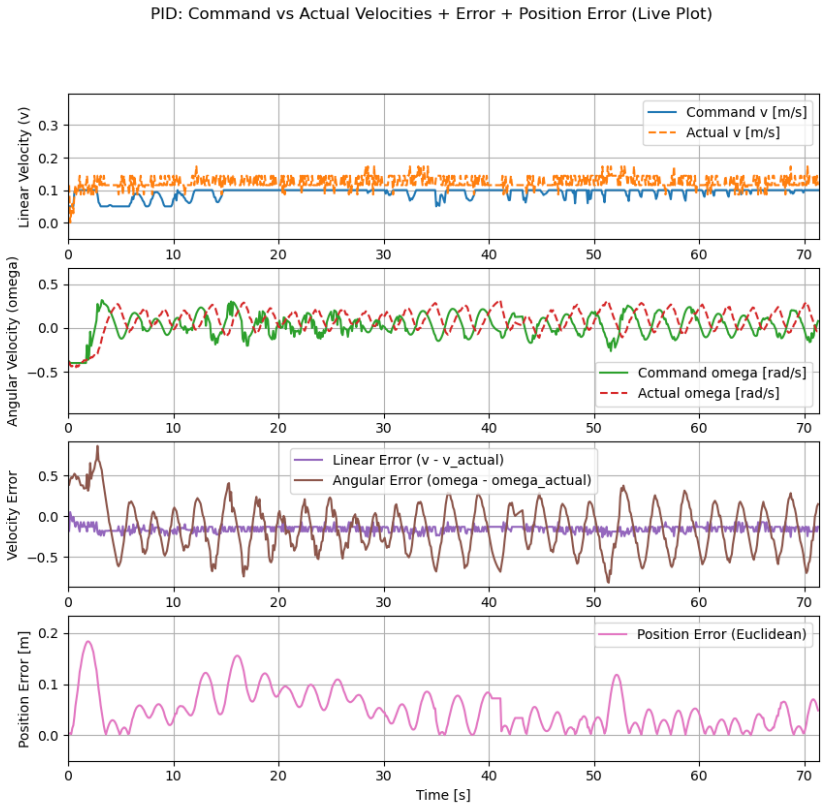


Figure 5.14: Commanded vs Actual Velocities + Errors for PID Controller (Live Plot).

Commanded vs Actual Velocity Response: The first figure displays the live plot of commanded velocities (in blue) and actual velocities (in orange) as the robot follows the circular trajectory. This plot also includes velocity errors and position tracking errors over time, providing insight into the controller's ability to match the commanded motion and maintain trajectory accuracy.

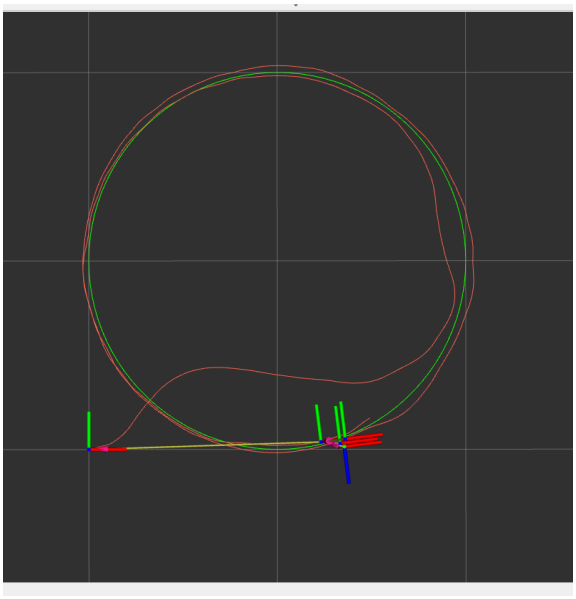


Figure 5.15: PID Circular Path Following: Commanded vs Actual Path.

Path Following Accuracy: The second image shows the robot's actual path compared to the commanded circular path . The robot closely follows the desired path, with minimal deviations that can be attributed to dynamic perturbations or small control imperfections.

This evaluation confirms the PID controller's effectiveness in controlling the robot's motion and maintaining accurate trajectory tracking, though there is still some room for improvement in reducing errors during perturbations or high-speed maneuvers.

SMC Controller Validation

The Sliding Mode Controller (SMC) was tested on a circular trajectory, and the results were visualized in the following figures.

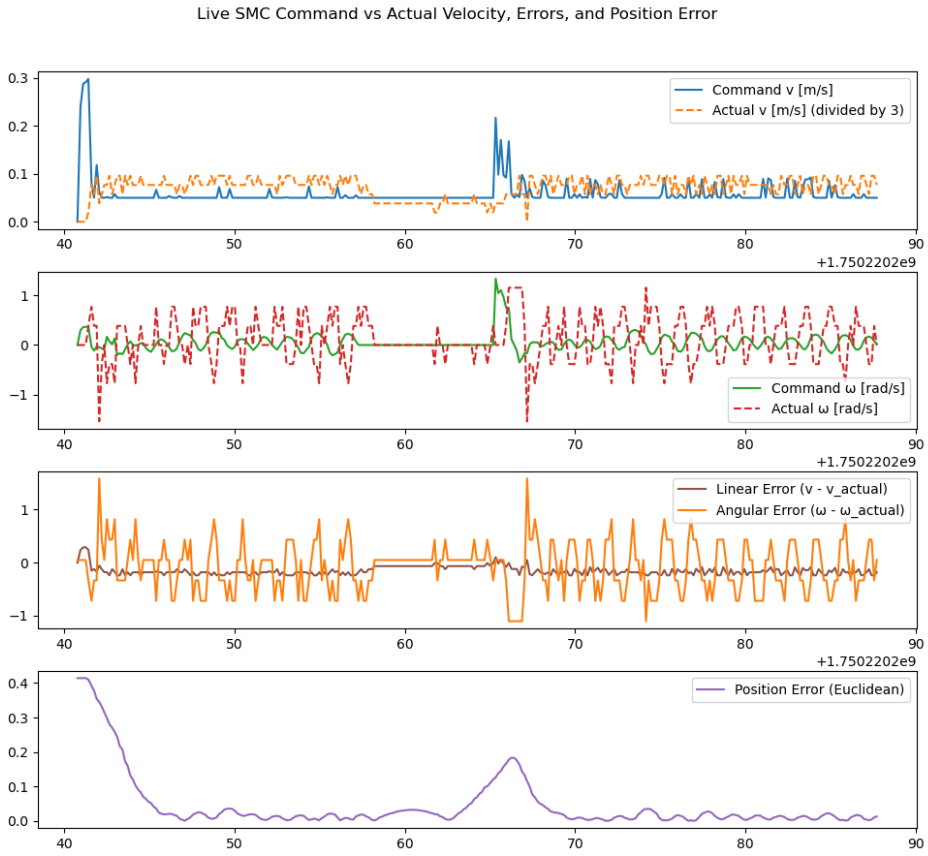


Figure 5.16: Commanded vs Actual Velocities + Errors for SMC Controller (Live Plot).

Commanded vs Actual Velocity Response: The first figure displays the live plot of commanded velocities (in blue) and actual velocities (in orange) as the robot follows the circular trajectory. This plot also includes velocity errors and position tracking errors over time, providing insight into the controller's ability to match the commanded motion and maintain trajectory accuracy. The linear and angular velocity errors are evident, illustrating the dynamic performance of the SMC controller.

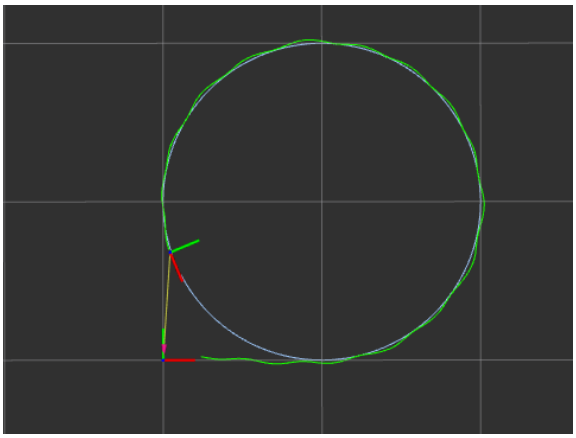


Figure 5.17: SMC Circular Path Following: Commanded vs Actual Path.

Path Following Accuracy: The second image shows the robot's actual path compared to the commanded circular path. The robot closely follows the desired path, with minor deviations that can be attributed to dynamic perturbations or small control imperfections. The position error, shown in the lower part of the live plot, demonstrates how the robot maintains relatively low deviation during its trajectory execution.

This evaluation confirms the SMC controller's effectiveness in controlling the robot's motion and maintaining accurate trajectory tracking. However, there is still some room for improvement in reducing errors during perturbations or high-speed maneuvers.

5.3.4 Autonomous Navigation

After successful manual validation, the robot was deployed for autonomous navigation experiments using the ROS 2 navigation stack.

In this phase, the robot was initially driven manually around the environment to generate a 2D occupancy grid map 5.18. This process was achieved using the onboard LiDAR sensor, coupled with the SLAM Toolbox, which incrementally processed the sensor data to create an accurate and detailed map of the surroundings. This map serves as the foundational element for autonomous navigation, enabling the robot to understand its environment and localize itself within it.

Once the map was completed, navigation goals were defined in RViz 5.19, 5.20. These goals represented target locations within the mapped environment that the robot was tasked to reach autonomously. The robot, using the ROS 2 navigation stack, autonomously planned a path to navigate from its starting position to the defined goal. This process employed the A* global path planning algorithm, which was implemented in the Nav2 stack. A* is a well-known and widely used algorithm that generates the most efficient path from the start to the goal by considering the grid map's obstacles.

During the experiments, the robot's planned path and its executed trajectory were visualized in RViz, providing real-time feedback and allowing for easy monitoring of the robot's movement. The real-time visualization ensured that both the planned and actual paths could be compared, helping to identify any deviations or errors in path execution. This validation process proved essential for evaluating the robot's performance in dynamic and real-world environments.

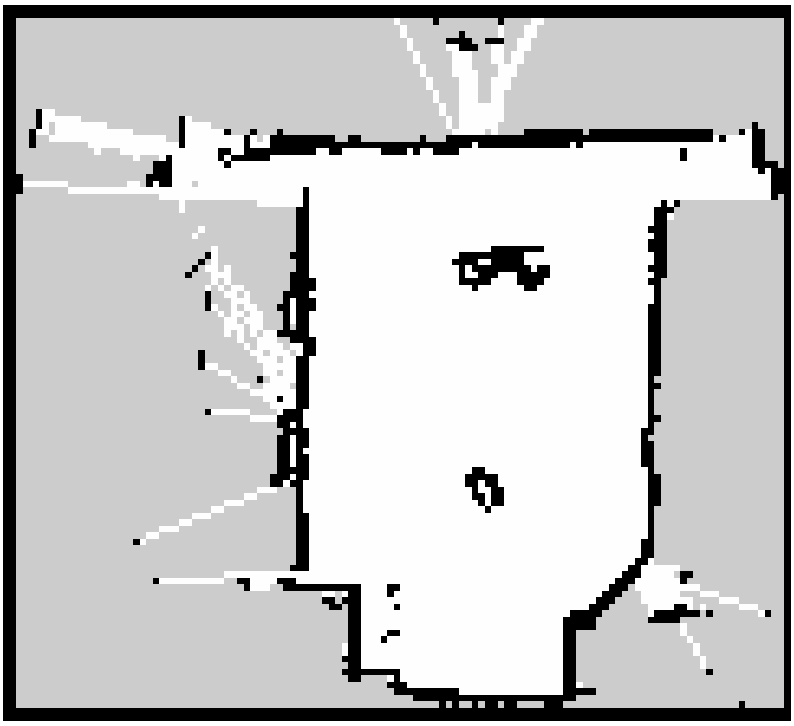


Figure 5.18: Occupancy grid map created by manually driving the robot via joystick teleoperation in the dorm room.

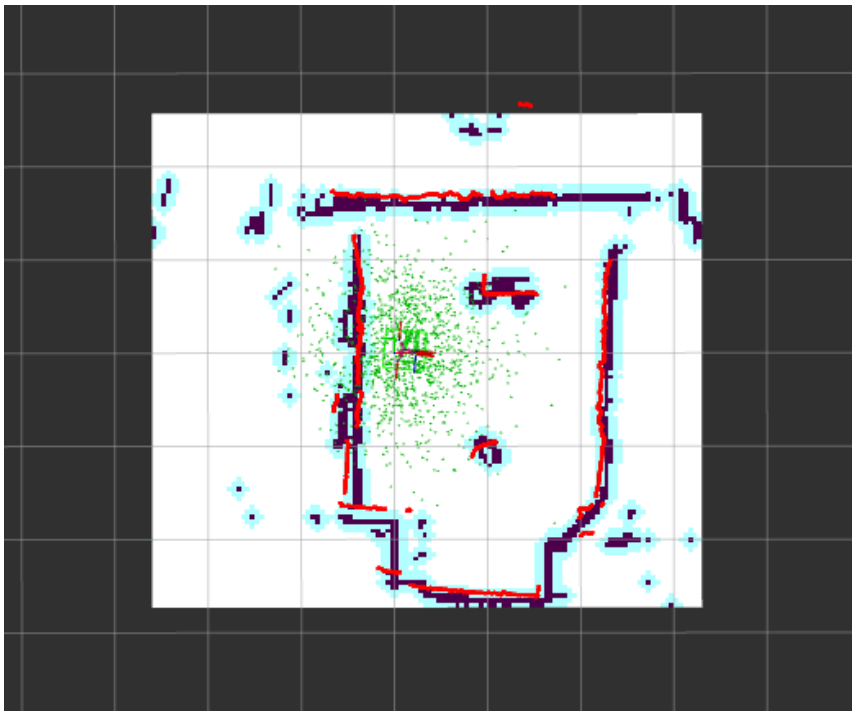


Figure 5.19: Autonomous navigation in a real indoor environment.

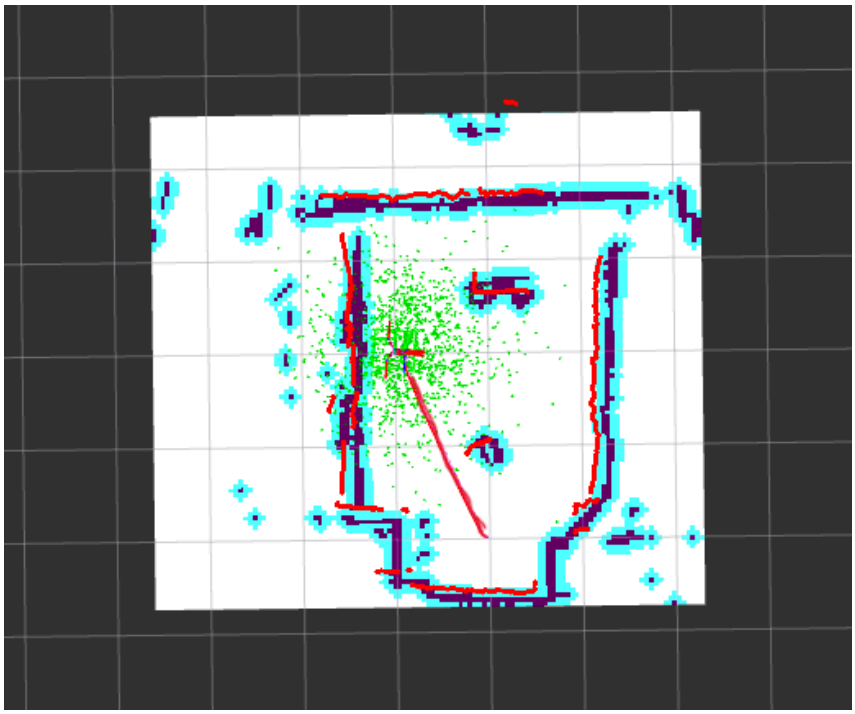


Figure 5.20: Path created by the robot to reach the chosen goal.

These experiments demonstrate the robot's ability to autonomously explore and navigate in a real-world environment, relying on the generated map and the standard ROS 2 Nav2 planning and navigation algorithms.

5.3.5 Experimental Procedure and Observations

Multiple experimental runs were conducted. The robot followed predetermined trajectories including circular and infinity-shaped paths both under manual teleoperation and autonomous navigation modes.

- **Encoder-Only Odometry:** Demonstrated noticeable drift after repeated trajectory cycles, resulting in significant deviations from intended paths.
- **IMU-Only Orientation:** Produced improved heading stability and reduced drift, allowing the robot to follow predefined trajectories more reliably. However, some residual position error remained due to the lack of linear velocity correction.
- **Navigation Stack Integration:** The robot successfully executed planned navigation tasks using the ROS 2 stack, but odometry drift remained a limiting factor for long-duration runs without sensor fusion.

These real-world experiments confirmed both the advantages and limitations of using IMU-only orientation for mobile robot navigation. In particular, the integration of IMU data with odometry proved advantageous for reducing heading drift and improving trajectory tracking. However, the results also highlighted the importance of full sensor fusion combining both IMU and wheel encoder data to achieve precise and robust localization in real-world environments.

Conclusion

This chapter presented a comprehensive evaluation of the mobile robot's performance through both simulation and real-world validation. The simulation results, supported by Gazebo and ROS 2, confirmed the effectiveness of PID and SMC controllers in tracking predefined trajectories such as circular and infinity paths. The real-world validation, conducted using the custom-built robotic platform, revealed the practical limitations of encoder-only odometry and the benefits of directly using IMU-based orientation estimation. While the IMU provided improved heading stability, residual positional errors suggested that future enhancements should include sensor fusion strategies to achieve high-precision localization. Overall, the consistency between simulation and physical results validates the control strategies and simulation framework established throughout this study.

Chapter 6

Conclusion and Future Work

6.1 Introduction

This work focused on the design and development of a mobile robot platform capable of implementing and evaluating various control strategies. This chapter provides a reflective synthesis of the work completed and highlights the main outcomes. From the initial stages of conceptual design to practical implementation and testing, the project offered valuable insights into both theoretical and applied aspects of mobile robot control.

The objective was not only to construct a functional system, but also to explore its performance under different control laws and to analyze its behavior in both simulated and real-world scenarios. The development process involved multiple layers of learning, adaptation, and problem-solving, encompassing hardware integration, sensor interfacing, and the deployment of control algorithms within the ROS2 framework.

The following sections present a concise summary of the core contributions, outline the technical and practical challenges encountered during the project, and propose directions for future research and platform enhancement. These reflections serve to conclude the current study while laying the groundwork for continued advancement in mobile robotics.

6.2 Summary of Work

This project began with the identification of core requirements for a mobile robotic platform intended to test control strategies. A differential drive base was chosen for its simplicity and broad applicability. We designed the mechanical structure using aluminum and verified its robustness through structural simulations.

The electronic architecture was carefully integrated around hoverboard motors and a Raspberry Pi 4, supported by BLDC motor drivers and various sensors. ROS2 was employed as the software backbone, allowing modular development of control nodes in both Python and C++.

We successfully implemented and tested several control strategies including manual control, PID and Sliding Mode Control (SMC) in both simulation and real world settings. The use of Gazebo provided a controlled environment for tuning and evaluation before deployment. Data logging and performance analysis were used to compare the effectiveness of each method.

6.3 Main Contributions

The primary contributions of this project are summarized as follows:

- **Design and fabrication** of a compact and modular mobile robot using repurposed hoverboard hardware and off-the-shelf electronics.
- **Integration of ROS2 framework** for sensor interfacing, actuator control, and real-time visualization.
- **Implementation of classical and advanced control techniques**, including PID and SMC, with an emphasis on trajectory tracking and robustness.
- **Validation through simulation and experiments**, with metrics such as tracking error, control effort, and energy efficiency used to evaluate performance.
- **Establishment of a reproducible and extensible platform**, which can serve as a testbed for future work in control theory, SLAM, or AI-based robotics.

6.4 Challenges Encountered

While the project successfully reached its primary objectives, it was not without significant technical and logistical challenges that had to be addressed throughout the development process:

- **Hardware availability and procurement delays:** One of the most significant challenges encountered was the limited local availability of key components such as BLDC motors, motor drivers, and LiDAR sensors. As a result, essential components including the RPLIDAR A1 and external BLDC motor drivers had to be imported from abroad, leading to extended delivery times and subsequent delays in the integration and testing phases. Due to the lack of appropriate commercial BLDC motors, hoverboard motors were repurposed for the project. These motors provided a practical and cost-effective solution, offering sufficient performance to meet the requirements of the control strategies implemented.

- **Motor driver compatibility:** Our initial attempt to reuse the original hoverboard motherboard revealed serious limitations related to undocumented communication protocols, restricted control interfaces, and poor responsiveness to external PWM commands. These issues constrained our ability to implement custom control algorithms and forced us to switch to dedicated external drivers that offered full compatibility with standard ROS-based control interfaces.

- **Sensor noise and latency:** Working with cost-effective sensors, particularly low-end IMUs and wheel encoders, introduced challenges such as signal noise, drift, and latency. These issues significantly affected odometry and localization accuracy, especially during long-duration experiments. As a result, resolving odometry reliability became a time-consuming task, requiring extended effort to diagnose and address the root causes. To mitigate these effects, filtering strategies were implemented and sensor data streams were calibrated, ultimately enhancing the system's performance in both simulation and real-world operation.

- **ROS 2 Jazzy debugging and system configuration:** A major technical obstacle stemmed from the use of ROS 2 Jazzy, the most recent ROS 2 distribution at the time of development. Although it introduced performance improvements and improved support for modern middleware, it also lacked backward compatibility with many widely used packages. The unavailability of stable versions of certain libraries (e.g., `slam_toolbox`, `robot_localization`, `teleop_twist_joy`) created dependency conflicts and forced us to adapt or build from source. Additionally, maintaining node synchronization, managing TF trees, and ensuring real-time performance under Jazzy's stricter security and middleware constraints added considerable complexity to our software development workflow.

- **Physical environment constraints:** One major limitation was the lack of sufficient indoor space to perform real-world tests of the mobile base. This restriction made it difficult to conduct consistent experiments and validate control algorithms under realistic conditions.

- **Time-intensive software development:** Another significant challenge was the substantial amount of time required to develop, debug, and structure the software workspace for the robot. As shown in the annexed workspace tree, every program ranging from basic hardware drivers to advanced controllers and navigation modules required meticulous attention, with the shortest scripts exceeding 100 lines and some reaching over 600 lines of custom code. This extensive software development demanded patience and rigorous iterative testing to achieve a stable final version.

- **Computational resource limitations:** Running simultaneous resource-intensive applications such as Gazebo, RViz, and screen recording software for demonstration and analysis placed a heavy load on the available CPU and GPU. This often resulted in simulation slowdowns or failures to run at real-time speed, thereby limiting our ability to conduct complex experiments and record results effectively. Managing system resources and scheduling simulations during periods of minimal background activity were necessary compromises to mitigate these computational bottlenecks.

Despite these difficulties, each challenge became a source of technical growth. Overcoming them not only strengthened our understanding of robotics development but also led to a robust modular platform that supports further research and experimentation in control strategies and autonomous navigation.

6.5 Future Improvements

While the current platform provides a solid foundation for testing control algorithms, several enhancements can be considered in future work.

6.5.1 Hardware Upgrades

- **Enhanced Power System:** Integrating a smart battery management system (BMS) could improve energy monitoring and prolong battery life.
- **Sensor Suite Expansion:** Adding a depth camera or GPS module would enrich localization and mapping capabilities.
- **Motor Feedback Precision:** Using high-resolution encoders would improve motion control accuracy, especially for fine maneuvers.
- **Chassis Optimization:** Increasing the overall size of the chassis could enhance load distribution and mechanical stability. Since the hoverboard motors are designed to support substantial loads, a larger and more spacious chassis would be better suited to fully exploit their capabilities and ensure structural balance during movement.

6.5.2 Software Extensions

- **SLAM and Navigation:** The current system successfully integrates the ROS 2 Nav2 stack along with SLAM Toolbox, enabling autonomous mapping and navigation. Future improvements may focus on optimizing path planning parameters, enhancing map merging capabilities, or testing in larger, unstructured environments.
- **Sensor Fusion Algorithms:** An Extended Kalman Filter (EKF) has already been implemented to fuse IMU and odometry data. Further enhancements could include tuning the EKF parameters, integrating additional sensor sources such as GPS or vision-based odometry, and improving robustness under dynamic conditions.
- **Remote Monitoring Dashboard:** A valuable future extension would be the development of a web-based dashboard for remote operation, visualization of live telemetry, and logging of system performance metrics.

6.5.3 New Control Techniques

- **Model Predictive Control (MPC):** MPC can offer better trajectory planning and constraint handling, especially in dynamic environments.

- **Reinforcement Learning (RL):** Implementing RL algorithms for path planning or obstacle avoidance could enable self-learning behaviors.
- **Hybrid Controllers:** Combining classical methods (e.g., PID) with intelligent algorithms (e.g., fuzzy logic or neural networks) could improve adaptability in uncertain environments.

Conclusion

This final year project has led to the successful design, construction, and implementation of a versatile mobile robotic platform specifically tailored for the evaluation and validation of diverse control strategies. The approach adopted throughout the project was both systematic and multi-disciplinary beginning with a thorough analysis of functional requirements, progressing through mechanical modeling and material selection, and culminating in the integration of advanced electronic components and a modular software architecture based on the ROS 2 framework.

One of the primary outcomes of this work is the realization of a functional and adaptable differential drive mobile base. The robot integrates hoverboard BLDC motors, external motor drivers, IMU and LiDAR sensors, and a single-board computer (Raspberry Pi), enabling closed-loop control, sensor feedback, and autonomous navigation. The platform has been validated both in simulation and through real-world experimentation, confirming its suitability for implementing classical controllers such as PID and more advanced techniques like Sliding Mode Control (SMC).

Throughout the development process, multiple engineering challenges were encountered, including limited availability of key components, integration of cost-effective sensors, sensor noise mitigation, and the physical limitations of the test environment. Addressing these challenges provided an opportunity to acquire deeper insights into system level integration, real-time control, and the configuration of robotics middleware. Notably, the application of SLAM Toolbox and the ROS 2 Nav2 stack demonstrated the system's capacity for autonomous localization and navigation in structured environments.

Moreover, the modular and scalable design of the robotic base ensures that it can be extended to include other control strategies, sensors (e.g., camera, GPS), and higher-level autonomy features such as path planning, obstacle avoidance, and multi-robot coordination. As such, it offers a reliable testbed for future experimentation in the domains of mobile robotics, embedded systems, and intelligent control.

In summary, this project not only fulfilled its initial technical objectives but also contributed to the development of valuable engineering skills and practical know-how in robotics. It lays a strong foundation for future academic research or industrial applications in autonomous systems. The platform developed is expected to continue serving as a pedagogical and experimental tool for validating control algorithms and advancing robotic technologies.

Appendix A

Annex A:PFE workspace

```
src
 mtor_driver
      motor_driver
          __init__.py
          motor_driver_node.py
          odometry_node.py
          pid_circular_node.py
          SMC_circular.py
          smc_infinity_node.py
      package.xml
      resource
      setup.cfg
      setup.py
      test
   robot_bringup
      config
          ekf.yaml
          gazebo_bridge.yaml
          nav2_params.yaml
      launch
          robot_gazebo.launch.py
      package.xml
      resource
      robot_bringup
          __init__.py
      setup.cfg
      setup.py
      test
      world
         ionic.sdf
   robot_description
      launch
      meshes
          base_link.STL
          castor_link.STL
          lidar_Link.STL
          wheel_left_link.STL
          wheel_right_link.STL
      package.xml
      resource
      robot_description
          __init__.py
      rviz
          config.rviz
          nav2_default_view.rviz
      setup.cfg
      setup.py
      test
      urdf
          ali.sdf
          lyes.urdf
   teleop_manete
       launch
           manete2.launch.py
           manete.launch.py
       package.xml
       resource
       setup.cfg
       setup.py
       teleop_manete
           __init__.py
           joy_cmd_vel.py
           manete_control.py
       test
```

Figure A.1: Directory structure of the ROS 2 workspace

Bibliography

[1] R. Raj and A. Kos, "A comprehensive study of mobile robot: History, developments, applications, and future research perspectives," *Applied Sciences*, vol. 12, no. 14, p. 6951, 2022.

[2] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, 2019.

[3] M. A. Niloy, A. Shama, R. K. Chakrabortty, M. J. Ryan, F. R. Badal, Z. Tasneem, M. H. Ahamed, S. I. Moyeen, S. K. Das, M. F. Ali, *et al.*, "Critical design and control issues of indoor autonomous mobile robots: A review," *IEEE Access*, vol. 9, pp. 35338–35370, 2021.

[4] U. Jahn, D. Heß, M. Stampa, A. Sutorma, C. Röhrig, P. Schulz, and C. Wolff, "A taxonomy for mobile robots: Types, applications, capabilities, implementations, requirements, and challenges," *Robotics*, vol. 9, no. 4, p. 109, 2020.

[5] A. Gimenez and R. Barber, "Mobile robots history,"

[6] K. Shabalina, A. Sagitov, and E. Magid, "Comparative analysis of mobile robot wheels design," in *2018 11th International Conference on Developments in esystems Engineering (dese)*, pp. 175–179, IEEE, 2018.

[7] M. Ni *et al.*, "Modelling and control for a class of mobile robots," *IFAC Proceedings Volumes*, vol. 40, no. 18, pp. 115–120, 2007.

[8] B. A. Francis and M. Maggiore, *Flocking and rendezvous in distributed robotics*. Springer, 2016.

[9] A. Stefek, M. Kelemen, and P. Grznár, "Energy-efficient motion planning of differential wheeled mobile robots for industrial and service applications," *Applied Sciences*, vol. 10, no. 18, p. 6381, 2020.

[10] P. K. Panigrahi and S. K. Bisoy, "Localization strategies for autonomous mobile robots: A review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 6019–6039, 2022.

[11] O. Abdelaziz *et al.*, "A survey of control techniques for mobile robots," *arXiv preprint arXiv:2310.06006*, 2023.

[12] A. Meghdari and A. Ghahari, "Trajectory tracking control of a mobile robot using fuzzy logic controller with optimal parameters," *Robotica*, 2022.

[13] A. Al Mamun, M. Rashed, and I. Sarker, "An intelligent control model for mobile robot navigation using neural networks and fuzzy logic," *Journal of Intelligent & Robotic Systems*, 2024.

[14] R. Luo *et al.*, "Reinforcement learning-based mobile robot path planning and control," *Electronics*, vol. 11, no. 11, p. 1754, 2022.

[15] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[16] M. Saad, R. Fareh, and B. Belobo Mevo, "Adaptive sliding mode control of wheeled mobile robot with nonlinear model and uncertainties," in *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*, pp. 1–6, IEEE, 2018.