PID Control of Autonomous Line-Following Robot

Achref Ben Mabrouk- A49925

Work carried out under the guidance of :

Prof. José Gonçalves

MSc in Electrical and Computer Engineering

2025-2026

PID Control of Autonomous Line-Following Robot

UC Project Report

Electrical and Computer Engineering

School of Technology and Management

Achref Ben Mabrouk- A49925

2025-2026

Dedication

"I am thankful to all those who said NO to me. It's because of them I'm doing it myself."

— **Albert Einstein** —

This work is dedicated with love and gratitude to the people who have shaped my journey, believed in my potential, and stood by me through every challenge.

To my dear father, Ben Mabrouk Youssef, a man of great integrity and endless dedication. Your tireless efforts and quiet strength have always inspired me to strive for the best. You have never spared any sacrifice to ensure our comfort and success, and for that, I am forever grateful.

To my tender and loving mother, Zidani Jamila, my first source of love, support, and encouragement. You are the heart of our family, the light that guides me in difficult times, and the voice that comforts me when I doubt myself.

To my beloved siblings, Ben Mabrouk Kamel and Ben Mabrouk Mariem, thank you for being my pillars of strength. Your support has been unwavering, and your presence an invaluable source of motivation and stability throughout this journey.

I also extend a heartfelt acknowledgment to Khaled, Chams, Seif, Bilel, and Faten. Each of you holds a very special place in my heart. Your friendship, understanding, and constant presence have brought meaning and comfort to my life.

May God bless you abundantly with peace, happiness, and a life filled with purpose. From the bottom of my heart — thank you

Special thanks

I would like to begin by sincerely thanking the members of the jury for the honor of evaluating and reviewing this project. Your constructive feedback, careful attention, and insightful comments have played an important role in improving the quality of this work. I am truly grateful for your time and consideration.

My deepest appreciation goes to my academic supervisor, Professor José Gonçalves, for his continuous guidance, support, and encouragement throughout this project. His expertise, patience, and dedication have been essential to the development and successful completion of this work. I am especially thankful for his availability, constructive advice, and motivation, which helped me stay focused and overcome challenges.

I would also like to extend my gratitude to all the professors and administrative staff of the Polytechnic Institute of Bragança. Their professionalism, kindness, and commitment to education have made my academic experience both enriching and inspiring. I am particularly grateful for the knowledge and skills I have gained during my studies, which have greatly contributed to my personal and professional growth.

Finally, I would like to express my heartfelt thanks to my family, colleagues, and friends for their constant support, patience, and encouragement throughout this journey. Your belief in me and your understanding during challenging times have been a continuous source of motivation and strength. This achievement would not have been possible without you.

Abstract

This project aims to design and build an autonomous robot capable of precisely following a line in a complex environment. The main objective is to ensure reliable and autonomous tracking of a challenging path, providing both accuracy and motion stability.

The robot is based on an ESP32 platform, equipped with five TCRT5000 infrared sensors arranged in a line to detect the lateral position of the line, and Hall-effect encoders on the wheels to measure the distance traveled and map the robot's trajectory. The kinematic modeling of the differential-drive robot was performed in MATLAB/Simulink to validate expected motions, including straight paths with different initial orientations.

The control system relies on a PID controller that adjusts motor speeds based on the error between the estimated line position (calculated using a weighted average of sensor readings) and the target position (center of the line). The control loop runs every 20 ms, providing fast and stable responses to disturbances.

The code is modularized into distinct parts: sensor reading, encoder handling, PID control, motor driving, and odometry, which improves maintainability and clarity. Tests demonstrate that the robot successfully completes the map with near-perfect line following, although minor inaccuracies in trajectory plotting are attributed to encoder limitations.

In conclusion, the project meets its goals of precise and autonomous line following, paving the way for future improvements such as odometry error correction and advanced navigation algorithms.

Keywords: robotics, line following, PID, ESP32

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robotics has evolved into one of the most dynamic fields of modern engineering, combining mechanics, electronics, and computer science to create intelligent systems capable of interacting with their environment [1]. Autonomous mobile robots, in particular, represent a crucial step toward the realization of intelligent systems capable of executing complex missions without human intervention. These robots are increasingly employed in logistics, manufacturing, healthcare, and research laboratories, where they perform tasks such as transport, inspection, and navigation in structured and semi-structured environments [2].

Among the many types of autonomous systems, the line-following robot has become one of the most representative platforms for studying and experimenting with key concepts in robotics, such as perception, control, and motion planning [3]. Although conceptually simple, designing a robot capable of following a line accurately, while maintaining speed and stability, presents significant technical challenges. It requires precise sensor calibration, efficient signal processing, robust motor control, and an optimized feedback algorithm capable of handling real-time variations.

This project aims to design, model, and implement an autonomous differential-drive line-following robot using an ESP32 microcontroller, infrared sensors, and wheel encoders. The robot is equipped with a PID (Proportional-Integral-Derivative) controller that continuously corrects its trajectory based on sensor data, ensuring

smooth, accurate, and reliable navigation [4]. The system integrates both theoretical modeling—through MATLAB/Simulink simulations—and practical experimentation, bridging the gap between control theory and embedded implementation.

1.1 Problem Description and Motivation

The central problem addressed by this project is achieving precise autonomous navigation using low-cost hardware and a simple control structure. The challenge lies in maintaining the robot's alignment with a predefined path despite variations in light, surface reflectivity, motor response, and mechanical imperfections.
Traditional on/off control strategies for line-following robots often produce oscillations, overshoot, and instability, particularly when the robot navigates curved or complex trajectories. To overcome these limitations, the present work employs a PID control algorithm combined with a weighted-average sensor technique. This approach allows continuous estimation of the line position and smooth adjustment of the robot's orientation, resulting in higher precision and faster convergence to the desired trajectory.
The motivation for this work stems from the desire to create an educational and research-oriented robotic platform that can be used to test advanced control and navigation algorithms. The combination of affordable hardware components, embedded programming, and real-time control makes this system a valuable foundation for future improvements, such as odometry-based correction, adaptive control, or AI-assisted navigation.

1.2 Objectives of the Project

The main objectives of this project are:

- To design and construct a compact, reliable, and efficient autonomous line-following robot;

- To develop and implement a PID-based control algorithm capable of maintaining accurate trajectory tracking;
- To model and simulate the kinematics of a differential-drive robot in MATLAB/Simulink for validation purposes;
- To integrate odometry using magnetic encoders for position estimation and trajectory reconstruction;
- To evaluate and validate the robot's performance experimentally, analyzing its accuracy, stability, and responsiveness.

1.3 Methodology Overview

The proposed methodology combines both simulation and real-world experimentation. The kinematic model of the differential-drive robot was first developed to describe its motion mathematically and simulate various trajectories. Based on this model, a control system was designed and implemented in C++ using the Arduino framework on the ESP32 microcontroller.

A modular architecture was adopted, separating the system into distinct subsystems for sensing, actuation, computation, and power supply. The integration of five TCRT5000 infrared sensors provides the robot with environmental awareness, while encoders attached to the EMG30 motors enable odometric feedback. The PID controller processes these inputs to regulate the wheel speeds through an L298N motor driver, ensuring stable and accurate motion.

1.4 Structure of the Document

This report is structured into four main chapters, each focusing on a specific stage of development:

- **Chapter 1 – Theoretical background:** Provides an overview of mobile robotics, including robot classifications, locomotion mechanisms, and autonomous navigation principles. It introduces the concept of line-following robots and their practical applications.
- **Chapter 2 – Design and Specifications:** Describes the mechanical, electronic, and software design of the autonomous robot. It details the sensors, motors, power supply, and control architecture, presenting the full technical specifications of the system.
- **Chapter 3 – Modeling of the Differential-Drive Robot:** Develops the mathematical and kinematic models that describe the robot's motion. MATLAB/Simulink simulations are used to validate the theoretical model and analyze different motion scenarios.
- **Chapter 4 – Implementation of a Weighted-Average PID Controller and Experimental Validation:** Explains the implementation of the PID control algorithm, the weighted-average line detection method, and the integration of odometry. It presents and discusses the experimental results obtained from the real robot.

Finally, the **Conclusion** summarizes the findings, evaluates the robot's performance, and proposes future directions, such as enhancing odometry accuracy, incorporating sensor fusion, or extending the system to more complex navigation tasks.

1.5 Background and Motivation

Over the past decade, autonomous mobile robots have become increasingly relevant in industrial and research applications, ranging from logistics and manufacturing to service robotics [1]. However, building an autonomous system capable of interpreting its environment and maintaining stable control remains a challenging task.

This project was motivated by the need to explore practical applications of control theory — particularly PID regulation — in real embedded systems [5]. By combining theoretical modeling with real-world implementation, this work bridges the gap between simulation and hardware experimentation.

1.6 Expected Contributions

This report contributes to the understanding and realization of:

- A practical implementation of a PID-controlled differential-drive robot using low-cost components;
- A methodology for integrating odometry-based feedback with line-following control;
- A validated experimental platform that can be extended to more advanced autonomous navigation systems.

Chapter 2

Theoretical Background

Robotics combines mechanical sciences, electronics, and programming. Robotics engineers aim to make robots operate autonomously by providing them with devices that enable perception of their surroundings and the ability to make appropriate and rational decisions without human intervention.

Robots have gradually been integrated into various fields of activity. Industry is undoubtedly the sector that benefits most from this technological advancement. For example, manipulators are the most widely used type of robot and have been integrated into multiple applications, such as the automotive industry. Since then, increasing awareness of the need for mobile robots has led to extensive research in this domain. Today, mobile robots are widely used in industry (product transport, agriculture, public works, and space exploration). There is also a strong trend toward expanding the environments in which robots operate to include office and domestic environments (service robots) [6].

This chapter provides a comprehensive overview of mobile robotics and the current state of autonomous navigation. It begins by defining what a mobile robot is and its fundamental capabilities, including perception, decision-making, and actuation.

It then presents the main types of mobile robots—wheeled, tracked, legged, aerial, and underwater—outlining their advantages, limitations, and areas of application.

Furthermore, it explores key aspects of autonomous navigation, such as the role of

sensors, localization, mapping, path planning, trajectory tracking, and obstacle avoidance. The importance of exteroceptive and proprioceptive sensors in acquiring both environmental and internal state information is emphasized, as they are essential for precise localization and control.

Finally, several practical examples, including line-following robots, Automated Guided Vehicles (AGVs), and service robots, demonstrate how theoretical principles are applied in real-world contexts.

Overall, this chapter establishes the theoretical and technical foundation necessary for the design and implementation of an autonomous line-following robot.

2.1 Definition of a Mobile Robot

A robot is an automated mechanical system capable of performing one or more tasks by executing a program. It is equipped with perception, decision-making, and action capabilities that enable autonomous operation based on environmental perception.

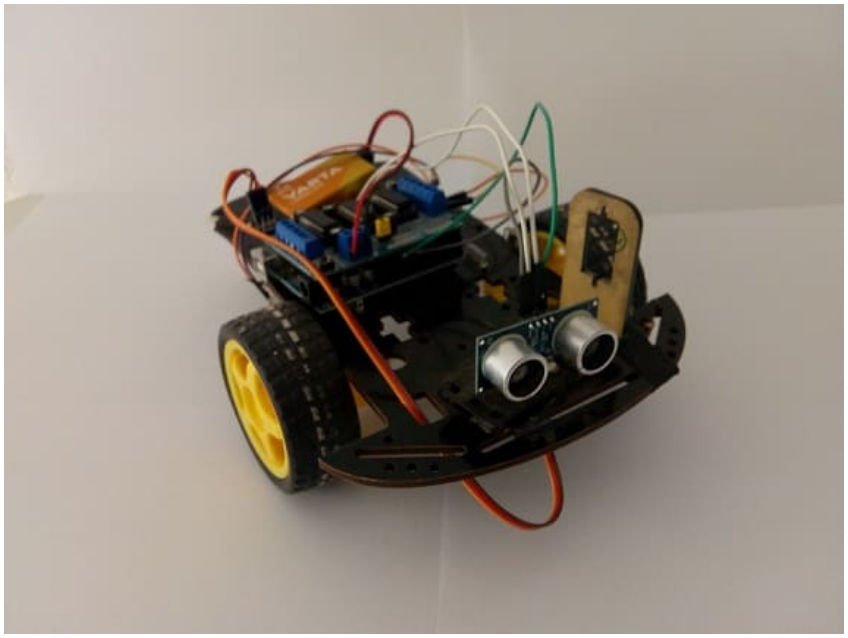


Figure 2.1: Intelligent mobile robot

Source: Robotique.site, 2024

2.2 Classification of Mobile Robots

Mobile robots can be classified according to the environment in which they operate and the locomotion mechanisms they use. The most relevant classification is based on their degree of autonomy. They can be categorized as follows:

2.2.1 Wheeled Mobile Robots

Wheeled mobile robots are currently the most common. Due to the simplicity of their locomotion mechanism, these robots can move in all directions with significant acceleration and speed, depending on wheel configuration and size. Three- and four-wheeled designs are the most prevalent.

Figure 2.2: All-terrain robot (wheel-driven mobile robot)

Source: The Rider Post, 2023

The three main classes of wheeled robots are:

- **Ackermann Steering Robots:** Equipped with four or three wheels (often two drive wheels and one or two steering wheels), providing a balance between stability and maneuverability.

- **Differential Robots:** Equipped with two independently driven wheels and, optionally, one or two passive caster wheels. This configuration enables precise motion control and stable movement on varied terrain.
- **Omnidirectional Robots:** Equipped with special wheels (such as Swedish or Mecanum wheels) that allow motion in any direction without changing orientation.

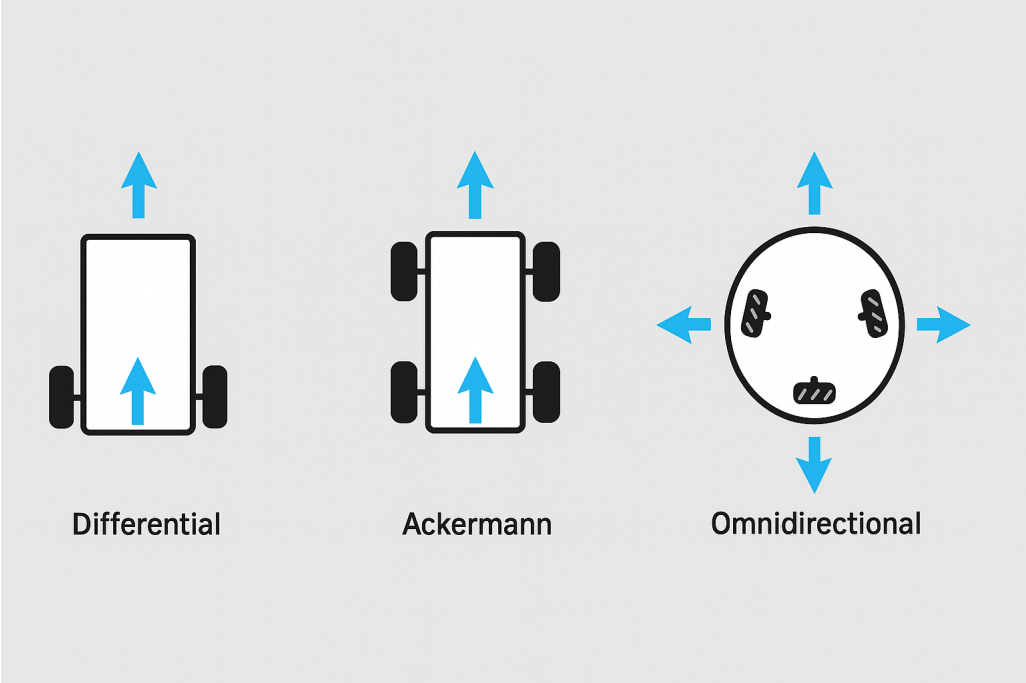


Figure 2.3: Main classes of wheeled robots

2.2.2 Tracked Mobile Robots

Tracked locomotion systems offer strong ground adhesion and a high capacity to overcome obstacles, making them suitable for uneven, loose, or low-quality surfaces such as sand, mud, or rocky terrain. Their large contact area provides stability and reduces ground pressure, minimizing the risk of sinking compared to wheeled systems [7], [8]. However, these benefits come at the cost of lower speed and higher energy consumption due to increased friction and mechanical complexity [9], [10]. Despite these limitations,

tracked robots remain preferred for tasks requiring robustness and obstacle-climbing capability, such as military operations, demining, rescue, and planetary exploration [11].

Figure 2.4: Tracked mobile robot (crawler-type chassis)

Source: SAT UAV, n.d.

2.2.3 Legged Mobile Robots

Legged robots offer an effective solution for operations in difficult or dangerous areas inaccessible to humans. Their articulated legs allow traversal of uneven terrain, obstacle negotiation, and movement in complex environments where wheels or tracks are inefficient. Inspired by biological locomotion, these robots combine stability, adaptability, and energy efficiency, making them valuable for exploration, rescue, and industrial missions in challenging settings.

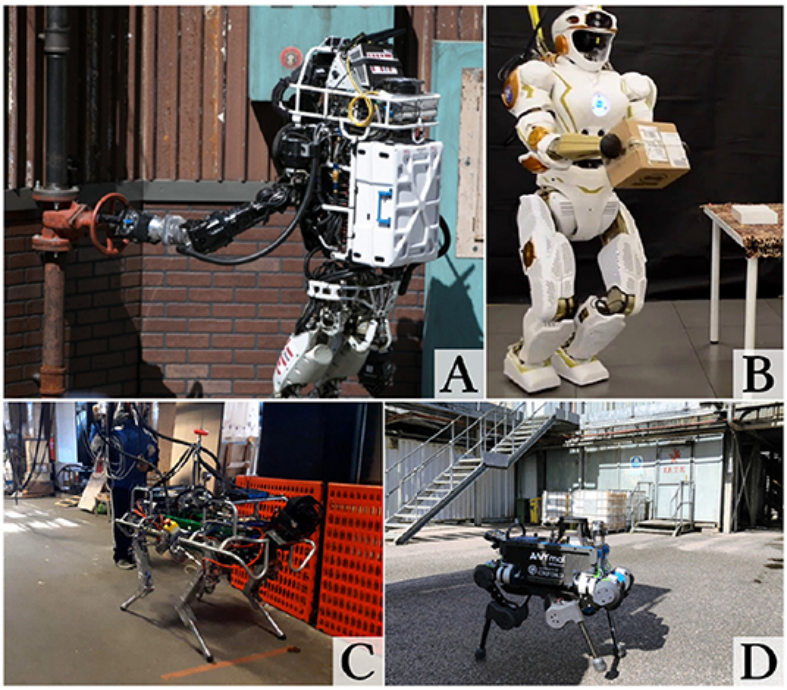


Figure 2.5: Examples of legged mobile robots

Source: Camurri et al. (2020), Frontiers in Robotics and AI

2.2.4 Flying Mobile Robots

An Unmanned Aerial Vehicle (UAV), or drone, is an aircraft operating without an onboard pilot. Initially developed for military purposes, drones now serve in civil and industrial applications, including mapping, agriculture, surveillance, hazardous environment operations, and media production. Their rapid development has been enabled by advances in sensors, power systems, and artificial intelligence, which allow autonomous flight, obstacle avoidance, and coordinated swarm operations.

Figure 2.6: Examples of flying robots (UAVs)

Source: UAV.com (About Us)

2.2.5 Underwater Mobile Robots

Underwater robots are autonomous or remotely operated vehicles designed for marine environments. They can rapidly acquire and securely store diverse data—physical, acoustic, or visual. Many serve as scientific platforms equipped with samplers or sensors such as sonars, cameras, and chemical or biological detectors. These robots are essential for oceanographic exploration, environmental monitoring, scientific research, and industrial inspection of underwater infrastructure and offshore energy systems.

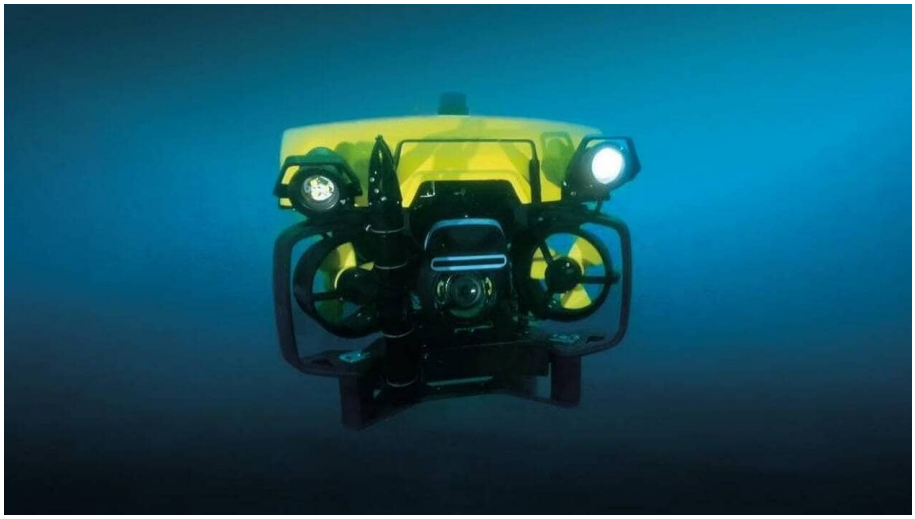


Figure 2.7: Example of an underwater robot

Source: Defence Industry Europe, 2024

2.3 Autonomous Navigation of Mobile Robots

Autonomous robots perform tasks and navigate environments without direct human control. Their autonomy depends on the ability to perceive, analyze, and interpret surroundings using sensory data. This perception relies on integrated sensors, actuators, and control programs that process information and make decisions. These components allow real-time adaptation, anticipation of situations, and effective operation in both structured and unstructured environments.

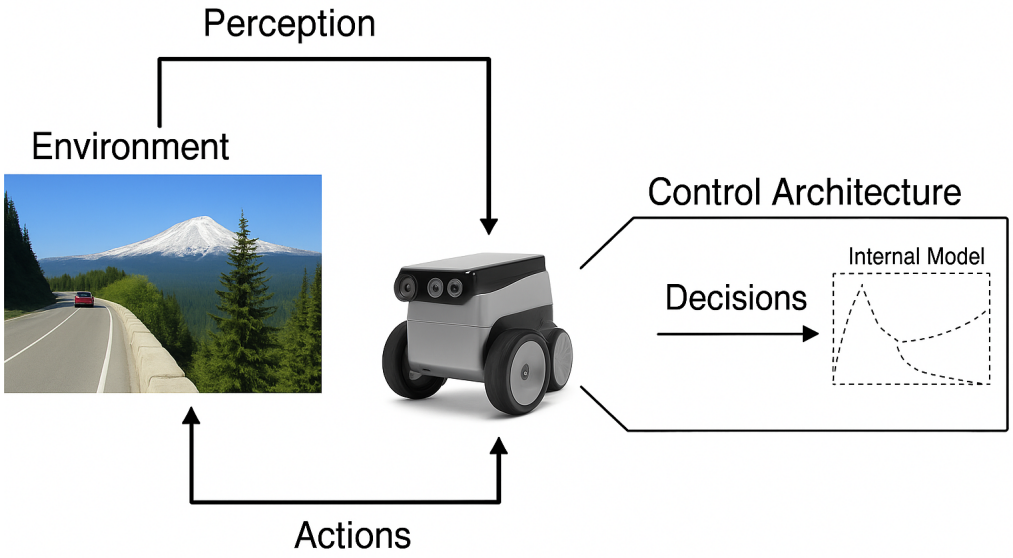


Figure 2.8: Autonomous navigation of a mobile robot

2.3.1 Fundamental Components of Autonomous Action

To perform tasks effectively, autonomous robots depend on three core components: perception, decision-making, and actuation.

- **Perception:** Information about the environment is acquired and interpreted through sensors (visual, acoustic, tactile, etc.), providing the basis for contextual awareness.
- **Decision-making:** Based on perceived data, appropriate actions are selected according to task objectives and environmental constraints. This process often uses artificial intelligence and planning algorithms.
- **Actuation:** The robot interacts with its environment through actuators and control loops to execute selected actions reliably.

Together, these form the autonomy cycle (perception–decision–actuation) [12]. For

example, upon detecting an obstacle, the decision module replans a route, and the actuation module adjusts the wheels to avoid collision.

2.3.2 Sensors in Mobile Robotics

In mobile robotics, sensors are divided into exteroceptive and proprioceptive types:

- **Exteroceptive sensors:** Gather information about the environment. Examples include ultrasonic, LiDAR, and infrared sensors for obstacle detection; cameras for vision; light sensors for illumination; microphones for sound; and tactile sensors for contact detection.
- **Proprioceptive sensors:** Provide information about the robot's internal state, such as encoders for wheel rotation, IMUs for motion, and electrical sensors for current and voltage monitoring. These support estimation of position, orientation, and trajectory.

2.3.3 Localization, Mapping, and Planning

Autonomous navigation involves three key questions:

- **Where am I? (Localization)** – The robot estimates its pose relative to a reference frame using odometry or external cues (e.g., GPS, landmarks). Sensor fusion or SLAM algorithms combine these for accuracy [13].
- **Where am I going? (Mapping / Place Recognition)** – A representation of the environment is constructed as a metric or topological map, or a hybrid of both [14].
- **How do I get there? (Path Planning and Control)** – A path is computed (A*, Dijkstra, or RRT) and tracked through control algorithms, such as PID, to follow planned trajectories [15].

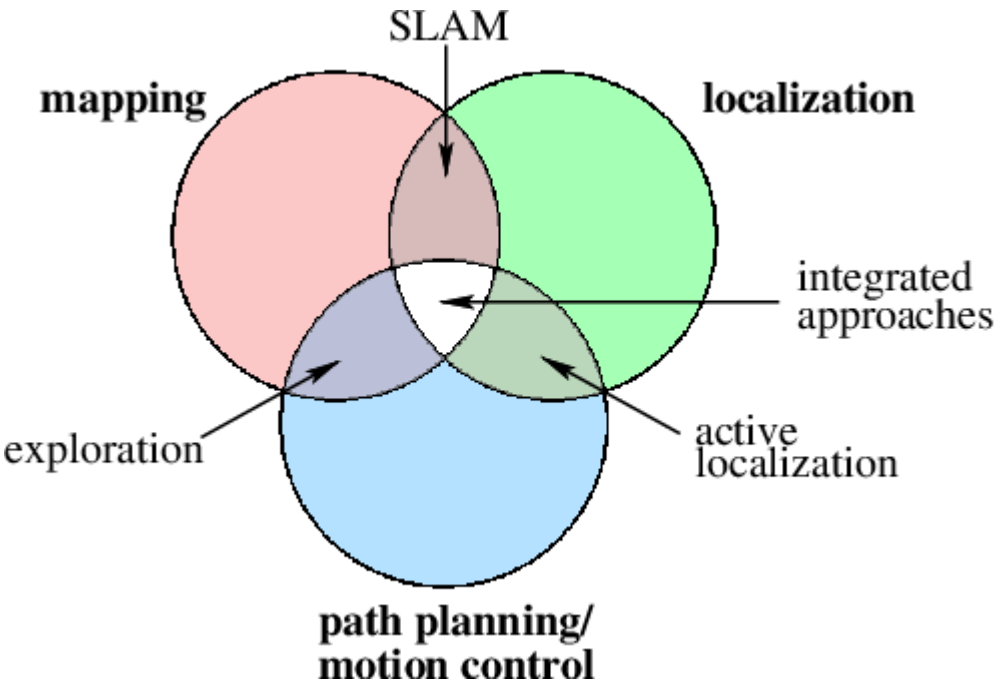


Figure 2.9: Mapping, localization, and path planning tasks

Source: Adapted from Stachniss (2006) [16].

2.3.4 Line-Following Robots and Their Applications

A line-following robot tracks a visible line on the ground using optical or infrared sensors. This method simplifies trajectory tracking by providing a predefined path. Line followers are common in education and research but also used industrially in Automated Guided Vehicles (AGVs) and warehouse automation (e.g., Amazon robots) [17].

Figure 2.10: Amazon's warehousing robots

Source: FrenchWeb, 2020

Modern applications include service robots in hospitals and hospitality sectors, following floor markings to deliver items or assist staff. A typical design employs multiple IR sensors and a PID controller to adjust wheel speeds for accurate path tracking [15].

2.4 Conclusion

This chapter introduced the fundamental concepts of mobile robotics, including robot types, locomotion modes, and principles of autonomous navigation. It also examined sensors, localization, mapping, and path planning as core components of autonomy. The study of line-following robots illustrated the practical implementation of these principles through feedback control and PID regulation. The knowledge presented here provides the foundation for the next chapter, which describes the design and development of an autonomous line-following robot, detailing its mechanical construction, electronic components, and control system.

Chapter 3

DESIGN AND SPECIFICATIONS

This chapter provides a detailed description of the map and the technical design of the autonomous robot, covering the process from functional requirements to hardware and electronic integration. The robot has been specifically engineered to follow a black line with high precision and full autonomy. To meet this objective, a modular architecture was adopted, enabling a clear separation of sensing, control, and actuation subsystems. The perception layer relies on an array of infrared sensors that detect line contrast, while the decision-making and control logic are handled by an ESP32 microcontroller, ensuring fast signal processing and flexible programmability. Actuation is achieved through efficient motor drivers coupled to DC motors, delivering smooth motion and precise trajectory correction. The mechanical structure has been carefully optimized to balance stability, weight distribution, and responsiveness, thus enhancing both speed and accuracy during navigation. Furthermore, the design takes into account dimensional constraints, power efficiency, and robustness, allowing reliable operation in competition-like environments. The following sections provide a comprehensive overview of the technological choices, dimensional considerations, and critical specifications that guided the development of the robot.

3.1 Map Design and navigation Strategy

The operational environment of the robot is defined as a controlled arena consisting of a continuous black trajectory line laid over a high-contrast white surface. This design ensures clear line visibility for the infrared sensors, minimizing perception errors due to ambient lighting variations or surface irregularities. The map configuration has been intentionally structured to incorporate a variety of navigation challenges, each aimed at testing specific aspects of the robot's perception, control, and decision-making capabilities:

- **Straight paths:** These segments allow the robot to maintain a constant velocity while ensuring accurate alignment with the line. They provide an effective framework for assessing the performance of the PID controller, particularly in terms of stability and smooth corrective adjustments.
- **Sharp turns:** These sections require the robot to execute abrupt trajectory changes, testing both the responsiveness of the control algorithm and the effective use of the outermost infrared sensors for detecting significant line deviations.
- **Intersections:** At points where multiple paths converge, the robot must apply decision-making strategies to correctly interpret the intended trajectory, highlighting the importance of both sensing accuracy and logical control routines.

The dimensions, complexity, and geometric diversity of the track are deliberately calibrated to evaluate not only the robot's line-following efficiency but also its robustness in handling sudden transitions, error recovery, and potential obstacle detection. In this way, the map serves as a comprehensive benchmark for validating the overall functionality and adaptability of the autonomous system.

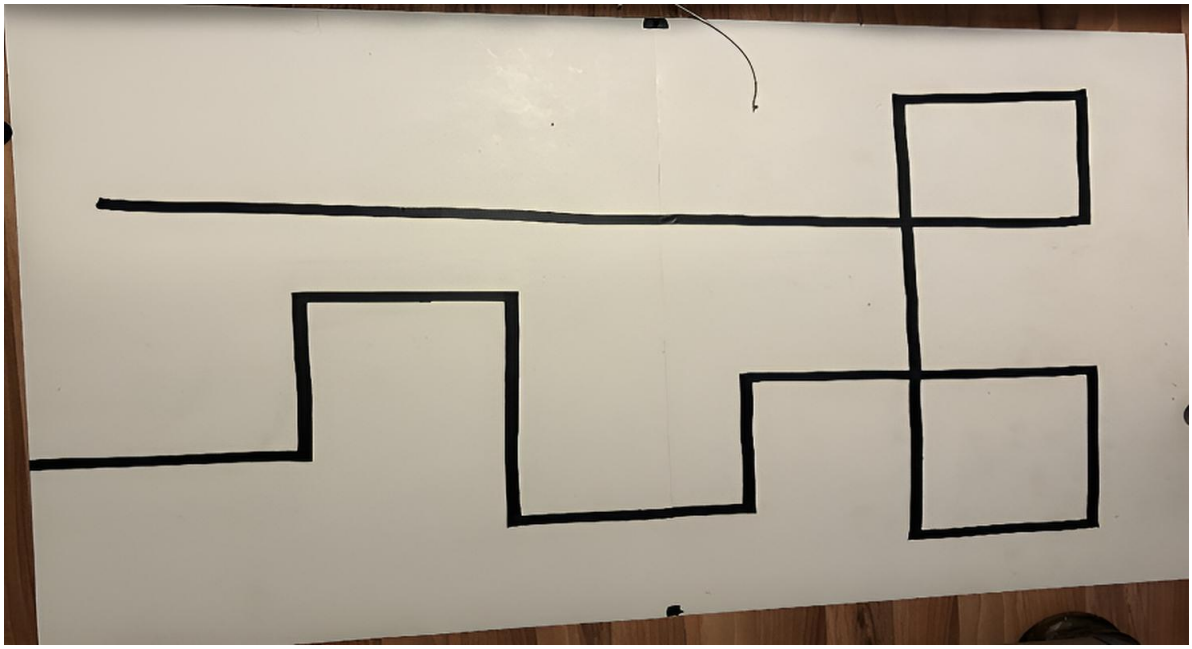


Figure 3.1: The operational environment

3.2 Functional and Operational Requirements

The autonomous robot has been designed to fulfill specific functional requirements that ensure reliable performance in line-following tasks under varying track conditions. These requirements encompass the sensing subsystem, control logic, and motion execution strategies. The following subsections detail the key aspects of these functionalities.

3.2.1 Line Detection and Navigation Strategy

The navigation of the robot is governed by a line-following mechanism that ensures continuous tracking of a predefined path. This system is built upon a set of five TCRT5000 infrared reflective sensors, strategically arranged in a linear array to maximize detection accuracy. The sensors are calibrated to distinguish the contrast between the black trajectory line and the white background, thereby enabling precise real-time feedback for motion control.

3.2.1.1 Sensor Configuration and Spatial Arrangement

The five-sensor array is mounted at the front-bottom section of the robot to provide optimal ground visibility. Their spatial distribution plays a critical role in balancing sensitivity and coverage:

- **Central Sensor (S3)**: Serves as the primary reference point for line alignment and acts as the baseline input for the control system.
- **Adjacent Sensors (S2 on the right, S4 on the left)**: Provide correctional feedback by detecting slight deviations from the line and prompting steering adjustments.
- **Outermost Sensors (S1 on the far right, S5 on the far left)**: Dedicated to handling extreme conditions such as sharp turns, wide deviations, and intersections.
- **Inter-sensor spacing**: Standard spacing of 1 cm between central and adjacent sensors, and 1.2 cm between adjacent and outermost sensors, ensuring a balance between resolution and coverage.

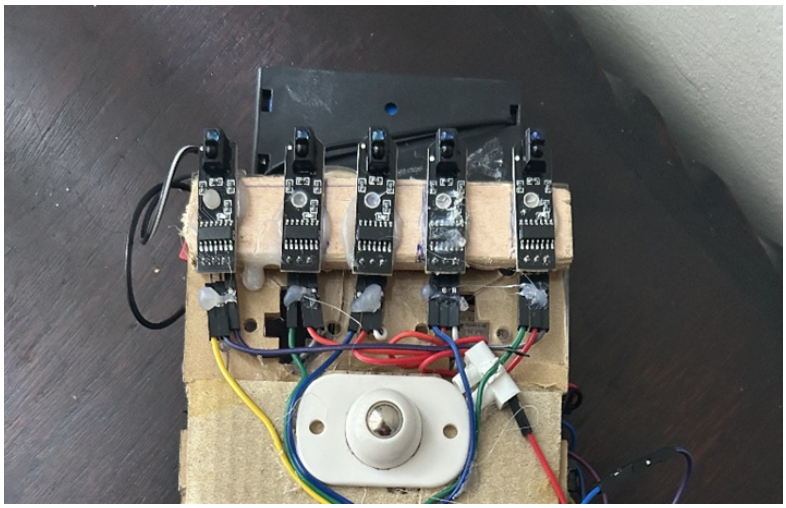


Figure 3.2: Infrared sensors arranged

This configuration allows the system to combine fine-grained correction with robust handling of abrupt trajectory variations.

3.2.1.2 Motion Control Logic

The control logic integrates sensor inputs with a PID-based algorithm to determine wheel speed adjustments. The navigation system recognizes three main motion scenarios:

- **Sharp Turns**: When a significant deviation is detected by the outermost sensors, the robot executes a corrective maneuver by dynamically adjusting wheel speeds (reducing velocity on one side while increasing it on the other).
- **Intersection Handling**: If all sensors simultaneously detect the line, the system interprets this as a potential intersection. The robot is programmed to continue forward until further deviation is registered.
- **Trajectory Maintenance**: During normal operation, the central and adjacent sensors continuously provide error feedback to the PID controller.

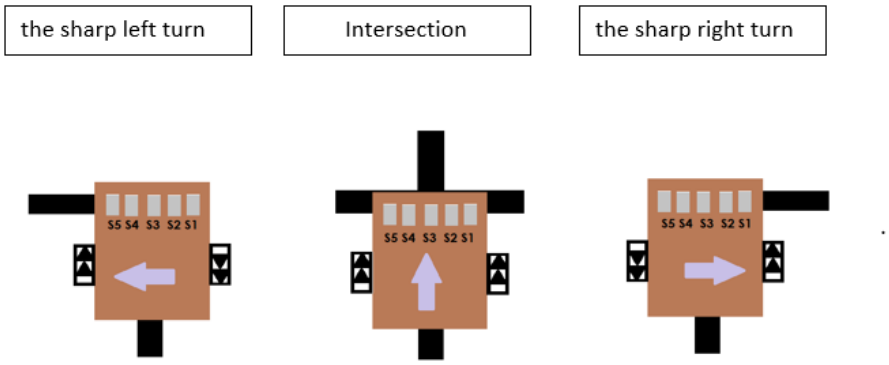


Figure 3.3: Infrared sensors arranged

3.2.2 Technical Specifications

The table below presents the **key technical specifications** of the autonomous robot, highlighting the choices made to balance performance, precision, and efficiency in line-following tasks.

Category	**Details**
Dimensions	25 cm × 25 cm × 10 cm — a compact structure designed to enhance maneuverability and stability during navigation.
Weight	2 kg (including battery) — optimized to ensure low inertia for precise motion control while maintaining structural robustness.
Power Supply	3 ×3.7 V, 2500 mAh LiPo battery.
Microcontroller	ESP32 (dual-core, 240 MHz) — chosen for its computational power.
Sensors	5× TCRT5000 infrared sensors for line detection, EMG30 magnetic encoders for odometry.
Motor Control	L298N module — provides bidirectional control of DC motors, supporting PWM speed regulation and precise motion adjustments.
Motors	2× EMG30 DC motors (12 V, 170 RPM) with gearboxes — delivering high torque for rapid and controlled turns, suitable for dynamic maneuvers.

Table 3.1: Technical specifications of the autonomous robot.

This configuration ensures an optimized integration of lightweight design, responsive actuation, and precise sensing, supporting robust autonomous navigation and efficient line-following performance.

3.3 Mechanical Design

The mechanical design of the autonomous robot focuses on stability, modularity, and precise motion control. It encompasses the chassis, wheels, support mechanisms, and

motor mounts, all optimized for reliable line-following performance.

Chassis: The chassis houses all the robot's components, including the mechanical assemblies, electronic boards, and battery. An almost square frame with rounded corners, made of 3 mm acrylic, was selected to ensure a lightweight yet rigid structure, providing balanced weight distribution and stability during motion.

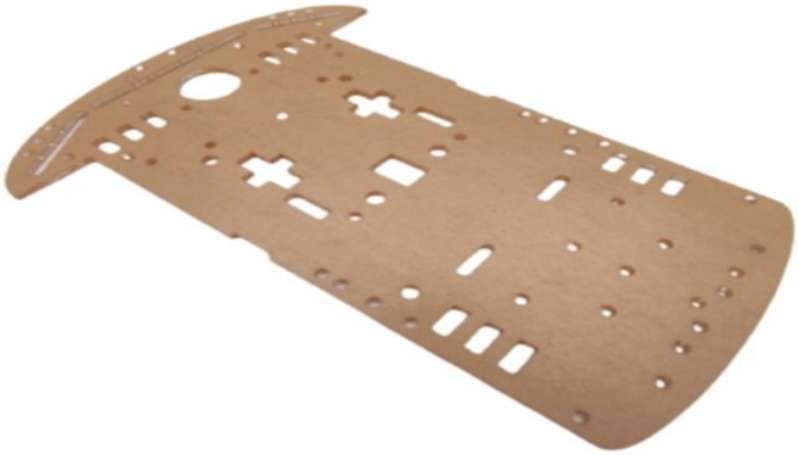


Figure 3.4: Robot's chassis

wheels: The robot is equipped with two identical plastic wheels fitted with rubber tires, each with a diameter of 69 mm. These provide traction and smooth rolling on the competition surface.

Figure 3.5: Robot's wheels

Ball Wheel: To enhance stability and prevent tipping, a ball wheel is placed at the front of the robot. This additional support ensures the mechanical balance of the three contact points of the robot with the ground.

Figure 3.6: Ball Wheel

Motor Mounts: Motors are fixed on the chassis using Hellermann cables and hot glue. This setup allows for firm attachment while maintaining easy adjustability during assembly.

Final System Structure: The final assembly integrates all mechanical components with electronic modules, including sensors, motor drivers, and the ESP32 microcontroller. Hot glue is used for securing lightweight components, ensuring a compact, stable, and modular platform.

Figure 3.7: the final systeme structure

3.4 Electronic Design and Control System

The electronic architecture and control logic form the brain of the robot, integrating sensing, actuation, computation, and power subsystems. The system is designed for efficient line-following, trajectory correction, and real-time decision-making. The robot comprises four primary electronic subsystems:

- Computation Subsystem: ESP32 microcontroller handles signal processing, control algorithms, and sensor-actuator interfacing.
- Actuation Subsystem: L298N motor driver controls the EMG30 DC motors, enabling precise speed and direction adjustments.
- Sensing Subsystem: TCRT5000 infrared sensors and magnetic encoders provide real-time line detection and odometry feedback.
- Power Subsystem: 3-series 18650 Li-ion batteries supply voltage to motors and electronics.

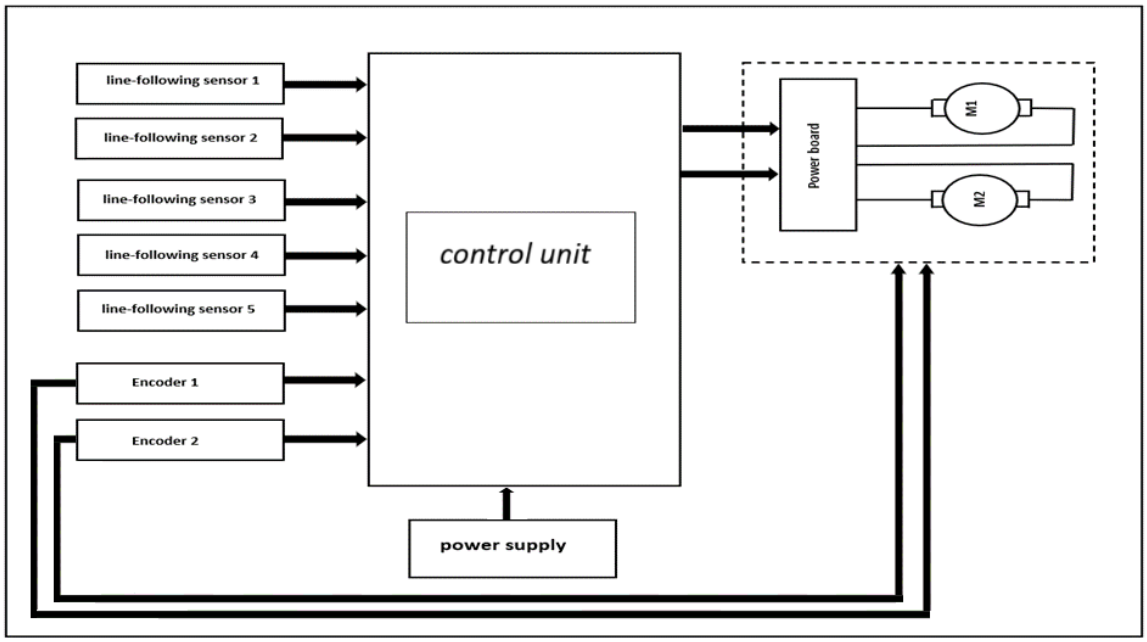


Figure 3.8: schematic diagram

3.4.1 Computation Subsystem

Component: ESP32 DevKit, The ESP32 acts as the central processing unit, executing PID control for line-following and coordinating sensor-actuator interactions. Key features include:

- Dual-core Tensilica Xtensa LX6 processor (240MHz).
- Integrated Wi-Fi and Bluetooth for remote monitoring.
- Multiple GPIOs, ADCs, and PWM channels.
- 512KB RAM, 30 accessible pins, built-in RESET and BOOT buttons.

The ESP32 interfaces with sensors and actuators through dedicated GPIOs, ensuring real-time processing and control.

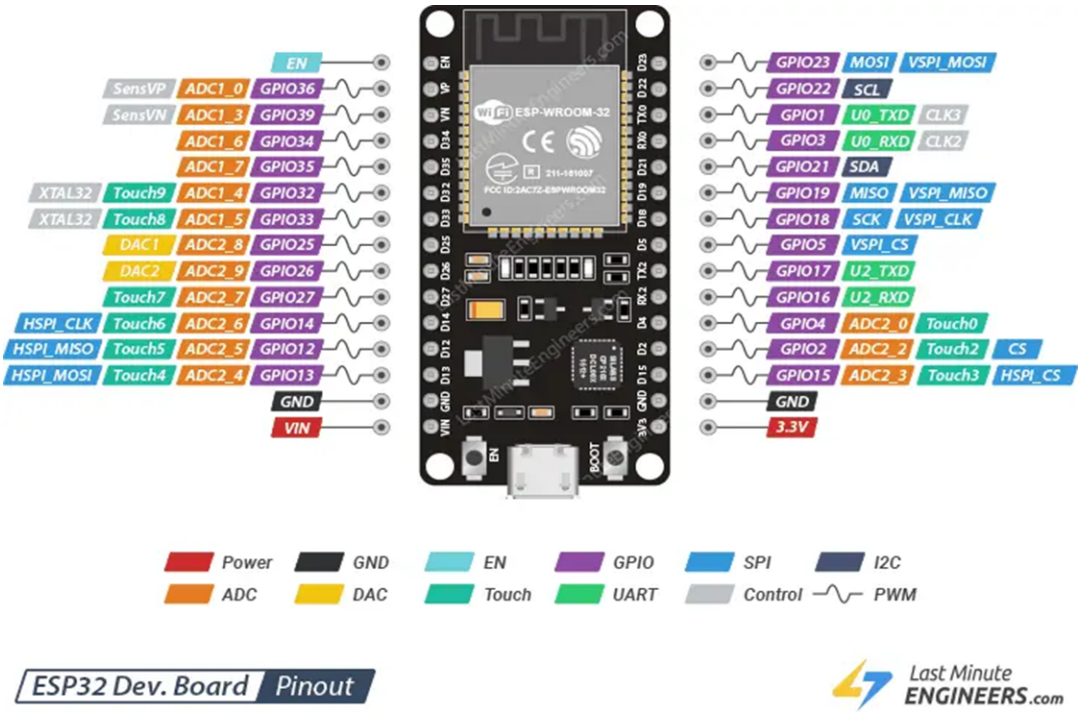


Figure 3.9: ESP32 devkit GPIOS

3.4.2 Actuation Subsystem

Motors: EMG30 DC motors with 30:1 gearbox and integrated magnetic encoders, offering precise speed and torque suitable for PID control.

- Rated voltage: 12V
- Rated torque: 1.5kg · cm
- Speed: 170RPM nominal (216RPM no load)
- Encoder resolution: 360pulses per revolution

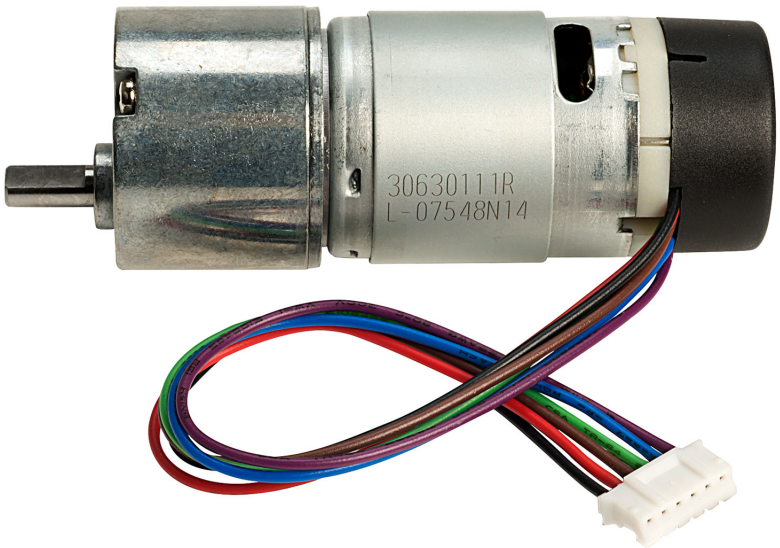


Figure 3.10: EMG30 Motor

Motor Driver: The L298 is the motor driver used to control the EMG30 motor . It works by using a H-Bridge circuit, allowing the motor to run in both directions. The L298 can supply up to 2A of current per channel, making it suitable for medium-sized motors. It also features basic protections against overheating and short circuits. In our robotic system, the L298 controls the motors that enable movement and direction changes, driven by commands from the microcontroller ESP32.

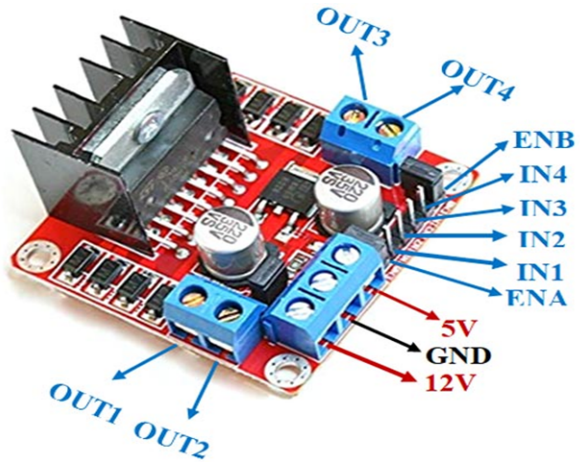


Figure 3.11: L298 driver

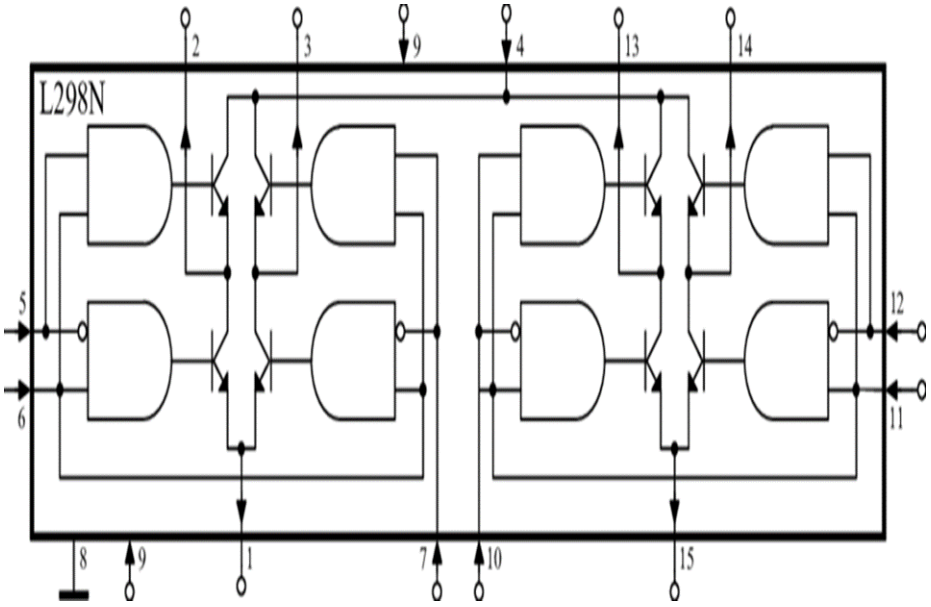


Figure 3.12: L298 dual H-bridge

3.4.3 Sensing Subsystem

The sensing subsystem is critical for enabling precise navigation, line-following, and odometry of the robot. It integrates multiple sensors that provide real-time feedback to the control system, ensuring accurate trajectory correction and robust operation under various conditions. Component: TCRT5000 Infrared Sensors.

3.4.3.1 Infrared Sensor (TCRT5000):

The TCRT5000 is a reflective infrared sensor commonly employed in mobile robotics for line detection, proximity measurement, and surface differentiation. It integrates an infrared emitter (LED) and a phototransistor detector, forming a compact optoelectronic transducer capable of converting reflected infrared radiation into an electrical signal. **Operating Principle:**

Figure 3.13: TCRT5000

- The sensor operates on the principle of reflective photometry:
- The IR LED emits a continuous beam of infrared light towards the target surface.
- The phototransistor detects a portion of the emitted light that is reflected back from the surface.
- The resulting phototransistor current is converted into a voltage output, which is proportional to the reflectivity of the surface.

- Dark surfaces (e.g., black lines) absorb most infrared light, producing a low voltage output, while bright surfaces (e.g., white background) reflect more IR light, producing a high voltage output.

This output can be utilized in either analog form for fine-grained signal processing or through a digital comparator for threshold-based line detection.

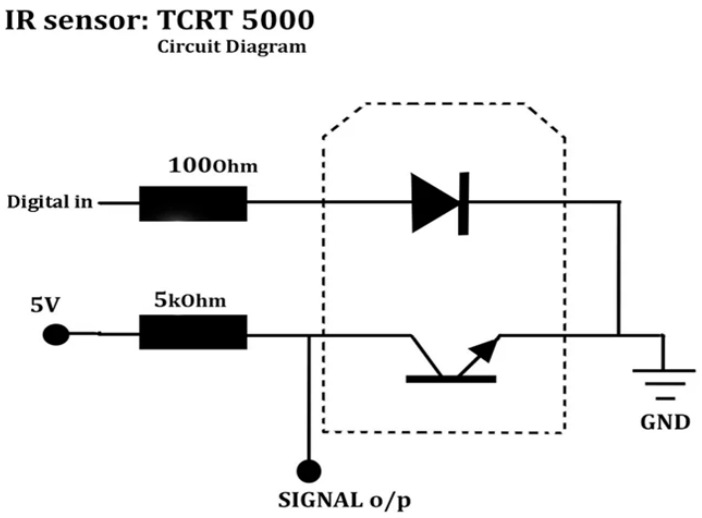


Figure 3.14: circuit diagram of TCRT5000

Technical Characteristics:

Feature	**Specification**
Spectral range of IR emitter	∼950 nm
Phototransistor response time	$<$ 100 us, enabling rapid detection of moving lines
Supply voltage	3.3–5 V DC
Output type	Analog (0–Vcc) or digital (via comparator)
Detection distance	Typically 1–15 mm, depending on surface properties
Power consumption	∼10–15 mA per sensor

Table 3.2: Technical Characteristics of the TCRT5000 Sensor

3.4.3.2 Magnetic Encoders (EMG30 Motors)

The EMG30 motors are equipped with magnetic encoders that provide precise measurements of wheel rotation, enabling real-time calculation of the robot's position and speed.

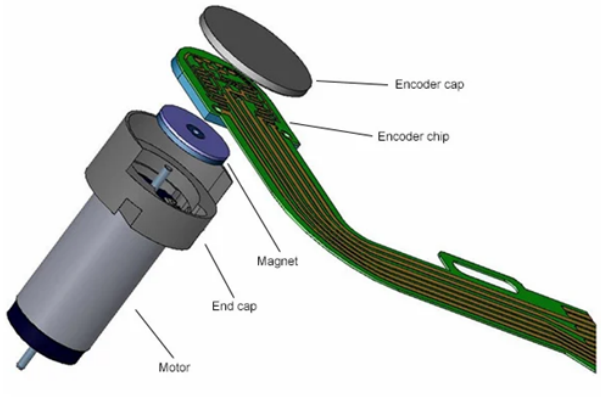


Figure 3.15: Magnetic Encoders (EMG30 Motors)

Operating Principle

- A small magnet attached to the motor shaft generates a magnetic field.
- A Hall-effect sensor detects changes in the field, producing two digital signals (A and B) in square-wave form.
- The signals are in quadrature (90degre out of phase), which allows determination of both rotation direction and distance traveled.

Pulse Counting and Resolution

Parameter	Value / Description
Encoder resolution	360 counts per revolution (CPR)
Wheel circumference	6.5 cm $\times \pi \approx$ 20.4 cm
Linear travel per pulse	$\approx$ 0.0567 cm
Quadrature decoding	1440 ticks per revolution (using rising and falling edges of A and B)
Precision	High accuracy for odometry and trajectory control

Table 3.3: Pulse counting and resolution of EMG30 encoders

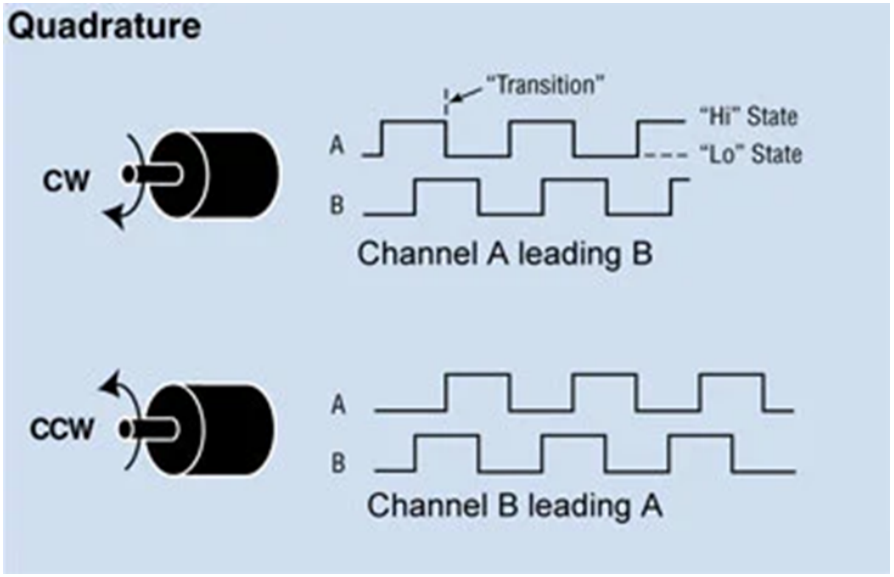


Figure 3.16: Quadrature signals A and B illustrating rotation direction detection
Source: Adapted from Bilal et al. (2018) [18].

3.4.4 Power Subsystem

This section describes in detail the robot's power subsystem, including battery configuration, voltage regulation and power distribution. The system is designed to power both the motors (via the L298) and the ESP32, while ensuring stability and safety. **Battery configuration** Batteries used: 3 18650 lithium cells (3.7V each, 3200

Figure 3.17: 18650 lithium batteries

mAh). Connection : In series: The 3 batteries are connected in series to obtain a total voltage of 11.1V ($3.7V \times 3$). Total capacity: 3200 mAh (capacity remains unchanged in series). **Autonomy calculations** Battery capacity: 3200 mAh at 11.1V. Total consumption : Motors: 500 mA (on average, at 6V). ESP32 + sensors: 135 mA (at 5V). Theoretical autonomy :

$$\text{Autonomie (h)} = \frac{3200\,mAh}{500\,mAh} \approx 6.4 \text{ heures} \tag{3.1}$$

This calculation assumes continuous use at full load. In practice, autonomy will be 3–4 hours .

3.4.5 Circuit Design

The electronic circuit of the robot was modeled and simulated using Proteus 8 Design Suite, which offers an efficient environment for embedded system prototyping and schematic validation. Since the overall system architecture is not overly complex, this software provided an adequate platform to represent and verify the different interconnections. The schematic includes the infrared sensor array, the motor driver, the

DC motors with their integrated encoders, the ESP32 DevKit microcontroller, and the battery module. As illustrated in the diagram, all these components are interconnected to form a compact and functional system, where sensing, computation, actuation, and power supply are integrated into a single architecture. This circuit diagram not only serves as a clear visualization of the system but also as a preliminary verification step, ensuring proper wiring and compatibility before the physical implementation of the hardware.

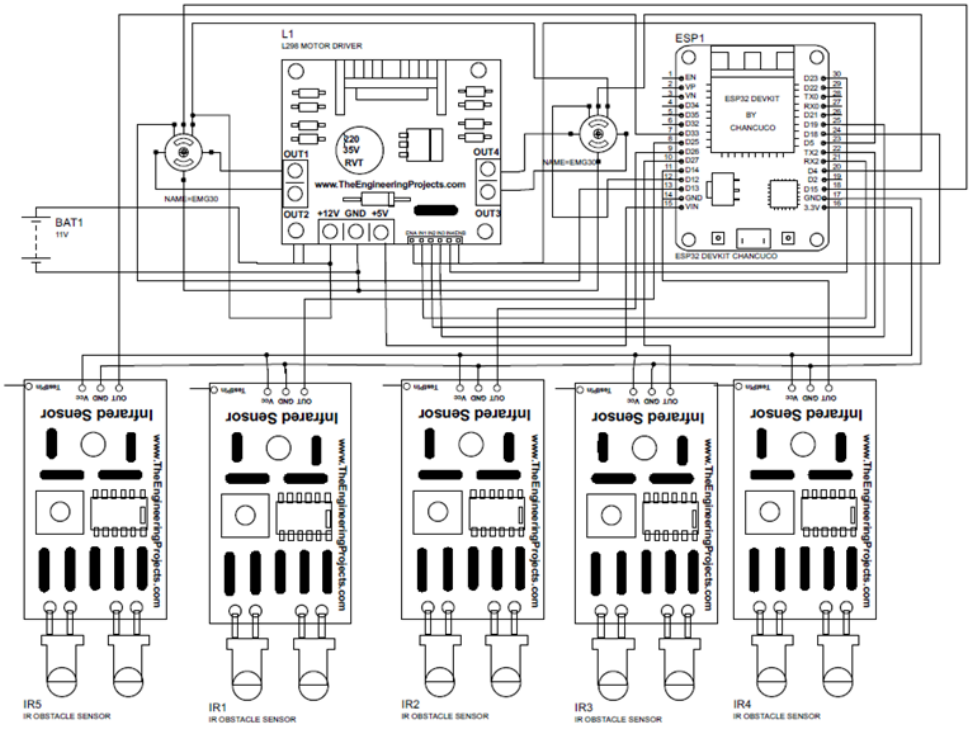


Figure 3.18: System schematic designed in Proteus 8

3.5 Conclusion

In this chapter, the design and specifications of the autonomous line-following robot have been thoroughly detailed. Starting from the functional and operational

requirements, the system was decomposed into sensing, control, and actuation subsystems, each carefully engineered to ensure high precision and robustness in navigation. The track design was structured to test a wide range of navigation challenges, while the electronic and mechanical architectures were optimized for stability, responsiveness, and energy efficiency. The resulting configuration strikes a balance between compactness, accuracy, and adaptability, providing a reliable platform for autonomous operation. This comprehensive design phase forms the foundation for the theoretical modeling and mathematical analysis of the robot. To better understand and predict its motion, the next chapter, Chapter III: Modeling of the Differential-Drive Robot, will establish the kinematic models that describe the robot's behavior. This model will serve as the analytical basis for implementing the control strategy and to validate the performance of the proposed system.

Chapter 4

Modeling of the Differential-Drive Robot

Wheeled mobile robots are among the most widely used robotic systems across various fields. Their popularity is largely due to their low energy consumption, low mechanical complexity, and high mobility, which make them highly suitable for a wide range of applications. Consequently, they play an essential role in research and development, serving as both practical tools and platforms for testing advanced control and navigation strategies.

When high speeds or heavy loads are involved, it becomes necessary to consider the dynamics of the robot that is, its mathematical modeling, which provides a formal description of motion under the influence of forces and torques. In general, mathematical modeling is a fundamental step in robot control, as it enables the design of algorithms for trajectory tracking, stability and planning.

For control purposes, two main types of models are typically employed: the kinematic model, which describes the geometry of motion without considering forces, and the dynamic model, which accounts for forces, torques, and inertia. In our case, we will focus exclusively on the kinematic model, as it provides sufficient accuracy for our objectives while remaining simpler to implement. As mentioned in the previous chapter,

there are several types of robots, including Ackermann steering robots, differential robots, and omnidirectional robots. In this work, we have chosen to use a differential-type robot because of its simple construction and its interesting kinematic characteristics[19]. In this chapter, we present the kinematic modeling of the differential-drive robot and verify the validity of the modeling through simulation.

4.1 Kinematic Modeling

Kinematics is the most fundamental study of the behavior of mechanical systems. In mobile robotics, we must understand the mechanical behavior of the robot both to design mobile robots suited to their tasks and to understand how to create control software for the robot's hardware instance[20].

4.1.1 Differential Drive Robot Kinematics

Differential-drive mobile robots using two drive wheels are the most common type of mobile robots. These robots are powered by two motorized wheels that are mechanically and electrically independent, and they may include a number of passive support wheels to maintain balance . These robots are commonly known as Differential Drive Mobile Robots (DDMR).

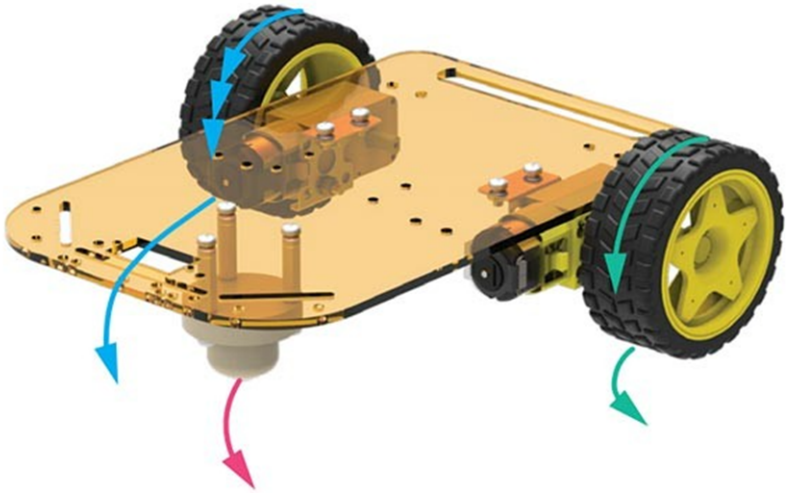


Figure 4.1: Free body of a differential-drive mobile robot

4.1.2 Reference Frames

The position of a differential-drive mobile robot can be described using two coordinate systems: the *inertial coordinate system* and the *robot coordinate system*.

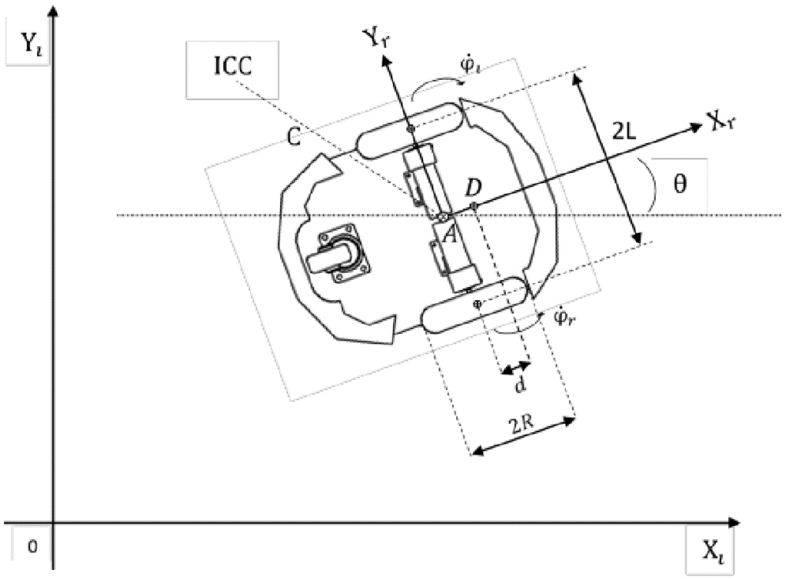


Figure 4.2: Position of the robot

Source: Adapted from Mekkaoui and Zemalache (2024) [21].

The inertial coordinate system is a global frame (laboratory frame) that is fixed in the environment in which the robot moves and serves as a reference frame. The robot coordinate system is a local frame attached to and moving with the robot.
The inertial frame is denoted by $\{X_l, Y_l\}$ and the robot frame by $\{x_r, y_r\}$. The point C, which is the midpoint of the axis between the two wheels, is taken as the origin of the robot frame [20].

Symbol / Element	**Description**
A	Point A is the center of the axis of the two drive wheels.
C	Point C is the instantaneous center of curvature (ICC) as the robot rotates.
D	Point D represents the robot's center of mass (or center of inertia).
d	Distance between points A and C.
2L	The distance between the two drive wheels.
2R	Diameter of the robot's wheels, so R is the radius.
$\dot{\varphi}_r, \dot{\varphi}_l$	Angular velocities of the right and left wheels, respectively.
θ	Orientation angle of the robot with respect to the global reference frame (X_l, Y_l).

Table 4.1: Symbols and Elements of the Differential-Drive Robot

The robot's position is described by point $C\{X, Y\}$ when expressed in the Cartesian coordinates of the inertial frame. The robot frame and the inertial frame are connected using a basic transformation matrix.
The transformation of velocities between the global frame and the local frame is given by:

$$\dot{X}_l = R(\theta)\, \dot{X}_r \tag{4.1}$$

where $R(\theta)$ is the rotation matrix defined as:

$$R(\theta) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

4.1.3 Motion Model of the Robot

For a differential-drive robot, the robot's motion is directly determined by the relative velocities of its two independently driven wheels. Forward motion is achieved when both wheels rotate at the same linear velocity, causing the robot to move along a straight path while maintaining its current orientation. The linear velocity of the robot in this case is equal to the average of the left and right wheel velocities, ensuring smooth translation without rotation.

Turning motions are generated by introducing a velocity difference between the wheels. A left turn occurs when the right wheel rotates faster than the left wheel, causing the robot to pivot around an instantaneous center of curvature (ICC) located to the left of the robot. Conversely, a right turn is produced when the left wheel rotates faster than the right, with the ICC located to the right. The sharper the difference in wheel speeds, the tighter the turning radius, enabling precise trajectory control for navigation in constrained environments.

When the wheels rotate in opposite directions at equal speeds, the robot performs an in-place rotation around its central vertical axis. This allows the robot to change its heading without translating, which is particularly useful for orientation corrections or maneuvering in tight spaces.

Additionally, the differential-drive configuration allows for continuous adjustment of both linear and angular velocities, making it highly versatile for path following, obstacle avoidance, and autonomous navigation tasks. By controlling the wheel speeds independently, complex trajectories, including curves, arcs, and spins, can be executed efficiently with simple kinematic equations.

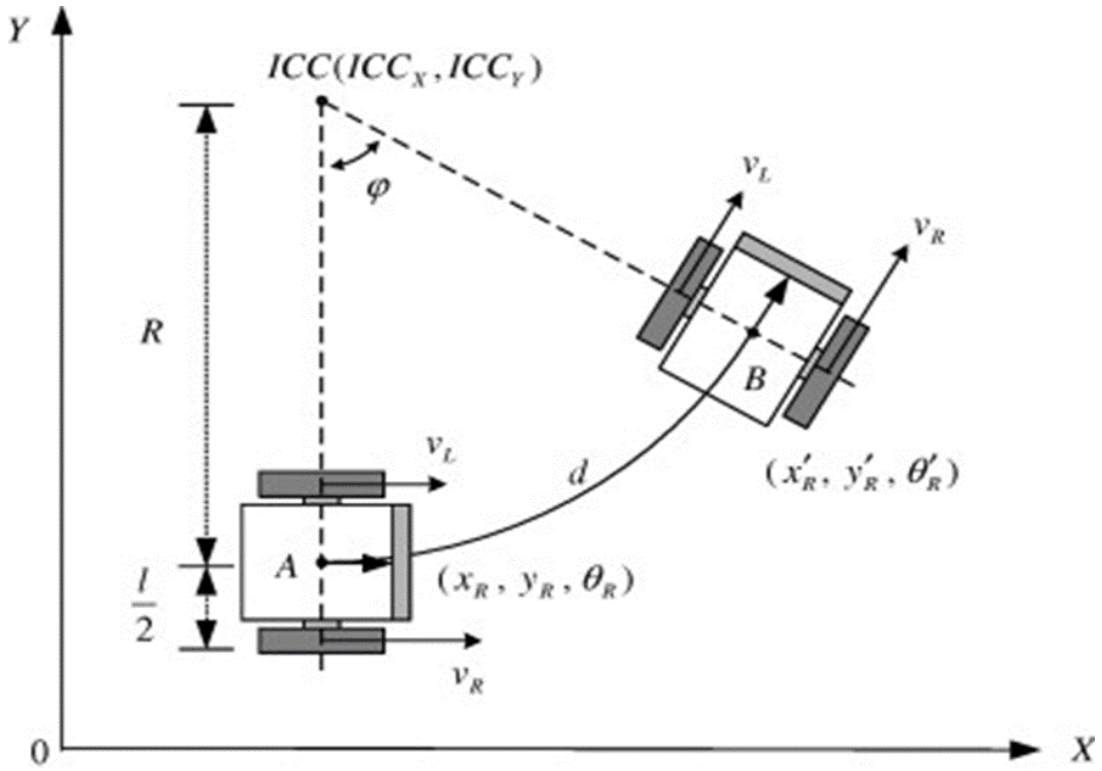


Figure 4.3: The robot motion

Source: Adapted from Han et al. (2008) [22].

4.1.4 Forward and Inverse Kinematics

We consider the forces acting on the right and left wheels, resulting from their respective linear speeds V_R and V_L (Figure 3.3). The overall motion of the robot is determined by its linear velocity V and angular velocity ω.

Single-Wheel Motion: If only the right wheel rotates while the left wheel remains stationary, the robot pivots around the left wheel. The instantaneous linear velocity at the center point C, located midway between the two wheels, is given by:

$$V_{CR} = \frac{V_R}{2} \tag{4.3}$$

Similarly, if only the left wheel rotates while the right wheel is stationary, the linear

velocity at C is:

$$V_{CL} = \frac{V_L}{2} \tag{4.4}$$

By combining these contributions, the overall linear velocity of point C is:

$$V = V_{CR} + V_{CL} = \frac{V_L + V_R}{2} \tag{4.5}$$

The angular velocity can also be calculated from the motion of each wheel. If only the right wheel moves, the robot rotates around the left wheel with:

$$\omega_{CR} = \frac{V_R}{2L} \tag{4.6}$$

For the left wheel alone:

$$\omega_{CL} = \frac{V_L}{2L} \tag{4.7}$$

Combining these, the angular velocity of the robot is:

$$\omega = \omega_{CR} + \omega_{CL} = \frac{V_L + V_R}{2L} \tag{4.8}$$

Both Wheels in Motion: When both wheels rotate simultaneously with different velocities, the robot follows a curved trajectory around a point called the Instantaneous Center of Curvature (ICC), located along the axis shared by the two wheels. The angular velocity around the ICC is the same for both wheels and satisfies:

$$\omega = \frac{V_R}{R + L} = \frac{V_L}{R - L} \tag{4.9}$$

From this, the individual wheel velocities are:

$$V_R = \omega(R + L) \tag{4.10}$$

$$V_L = \omega(R - L) \tag{4.11}$$

The angular velocity can also be expressed as:

$$\omega = \frac{V_R - V_L}{2L} \tag{4.12}$$

and the linear velocity as:

$$V = \frac{V_R + V_L}{2} = \omega R \tag{4.13}$$

The radius of curvature is therefore:

$$R = \frac{V}{\omega} = L\frac{V_R + V_L}{V_R - V_L} \tag{4.14}$$

The coordinates of the ICC are:

$$\text{ICC} = \begin{bmatrix} ICC_x \\ ICC_y \end{bmatrix} = \begin{bmatrix} x_R - R\cos\theta \\ y_R - R\sin\theta \end{bmatrix} \tag{4.15}$$

The forward kinematicsin the local robot frame is:

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2L} & -\frac{1}{2L} \end{bmatrix} \begin{bmatrix} V_R \\ V_L \end{bmatrix} \tag{4.16}$$

The inverse kinematicsis given by:

$$\begin{bmatrix} V_R \\ V_L \end{bmatrix} = \begin{bmatrix} 1 & L \\ 1 & -L \end{bmatrix} \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{4.17}$$

Transforming from the local frame to the global frame:

$$\begin{bmatrix} \dot{x}_G \\ \dot{y}_G \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos\theta & \frac{1}{2}\cos\theta \\ \frac{1}{2}\sin\theta & \frac{1}{2}\sin\theta \\ \frac{1}{2L} & -\frac{1}{2L} \end{bmatrix} \begin{bmatrix} V_R \\ V_L \end{bmatrix} \tag{4.18}$$

Finally, if the wheel velocities are given in terms of angular speed ω_R and ω_L and wheel radius r:

$$V_R = \omega_R r, \quad V_L = \omega_L r \tag{4.19}$$

the kinematic model of the differential-drive robot becomes:

$$\begin{bmatrix} \dot{x}_G \\ \dot{y}_G \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta & \frac{r}{2}\sin\theta \\ \frac{r}{2L} & -\frac{r}{2L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{4.20}$$

4.2 Simulation of the Differential-Drive Robot Model

In this section, it is presented a detailed simulation of our differential drive robot using the Simulink environment in MATLAB. The primary objective of this simulation is to validate the kinematic model of the robot by examining the evolution of its position (x, y) and orientation θ in response to the input velocities of the left and right wheels, V_L and V_R.

The simulation model is constructed using Simulink blocks that implement the differential equations governing the robot's motion, ensuring that the system accurately reflects real-world dynamics. Various scenarios are explored by modifying the initial conditions and wheel velocities, allowing the observation of typical behaviors such as straight-line motion, circular paths, and more complex trajectories.

For each scenario, the system's step response to changes in wheel velocities is analyzed, providing insight into how the robot adjusts its motion over time. Each simulation

output includes:

- Angular orientation $\theta(t)$ over time
- Time response of the robot position $(x(t), y(t))$
- Robot trajectory in the XY plane

Through these simulations, it is demonstrated how different wheel speed inputs directly influence the global motion of the differential drive robot. This analysis confirms the validity of the implemented kinematic model and highlights the effectiveness of the robot's response to dynamic inputs, providing a solid foundation for further control design and experimental validation.

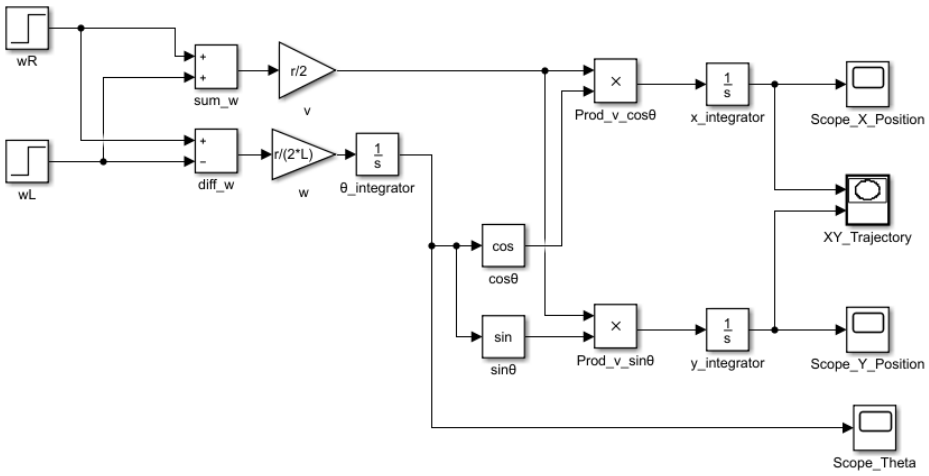


Figure 4.4: Kinematic Model in Simulink

4.2.1 Simulation Scenarios

Three representative cases with different initial orientations and wheel velocity inputs are considered to illustrate straight-line and curved motions of the robot. Each case assumes constant wheel speeds and a specified initial heading.

4.2.1.1 Case 1: Straight motion - zero initial orientation

Both wheels are driven at the same constant velocity $V_R = V_L = 0.2$ m/s and the robot's initial orientation is set to $\theta(0) = 0$. With equal wheel speeds, the robot experiences purely translational motion with zero angular velocity (i.e. $\omega = 0$).

In this scenario, the orientation $\theta(t)$ remains at zero for all t, and the robot moves straight along the positive x-axis. The $x(t)$ plot shows a linear increase (since x-velocity is constant), while $y(t)$ stays zero throughout. This produces a straight-line trajectory along $y = 0$, confirming that the model behaves as expected when $V_R = V_L$ (no turning).

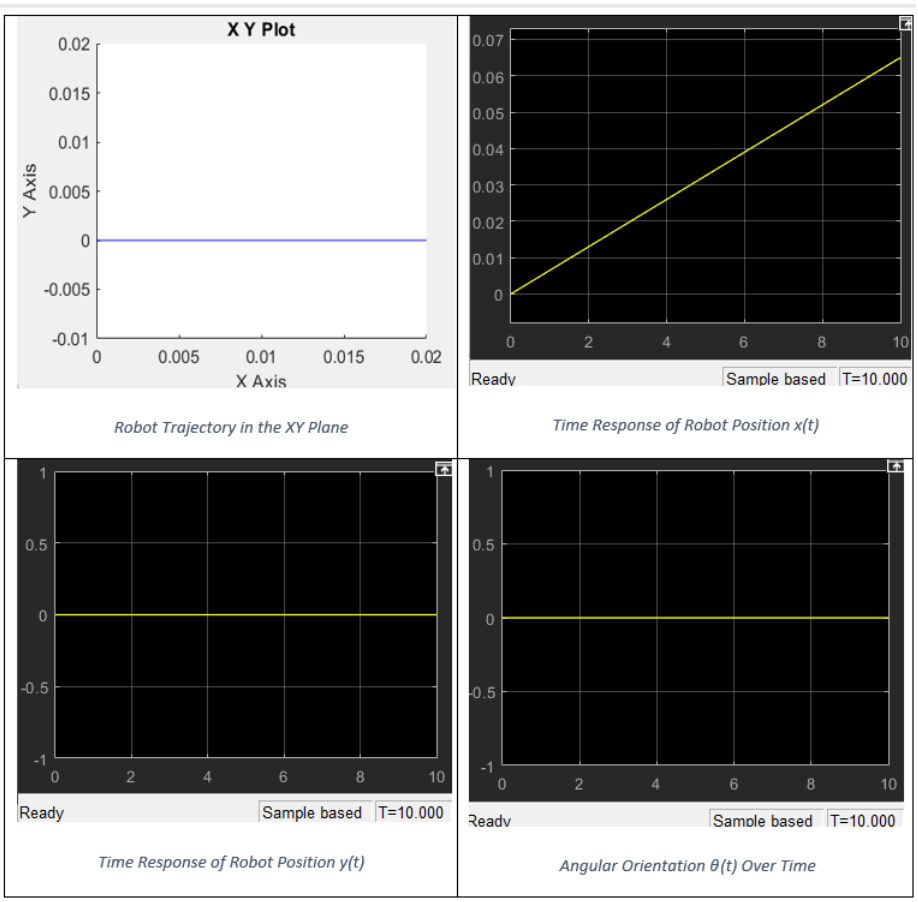


Figure 4.5: Straight motion with zero initial orientation

4.2.1.2 Case 2: Straight motion - non-zero initial orientation

Again we set $V_R = V_L = 0.2$ m/s, but now the initial orientation is $\theta(0) = \pi/3$. The wheel speeds are still equal, so $\omega = 0$ and the robot's heading does not change. However, because θ is non-zero, the robot moves in a straight line at an angle. In world coordinates, the velocity components are $\dot{x} = v\cos\theta$ and $\dot{y} = v\sin\theta$. Thus both $x(t)$ and $y(t)$ increase linearly, and the trajectory is a straight line inclined at $\pi/3$ radians. The orientation plot shows $\theta(t) = \pi/3$ constant, indicating the robot maintains its initial heading. This confirms the kinematic model correctly accounts for an initial heading: equal wheel speeds produce straight-line motion along the direction of that heading.

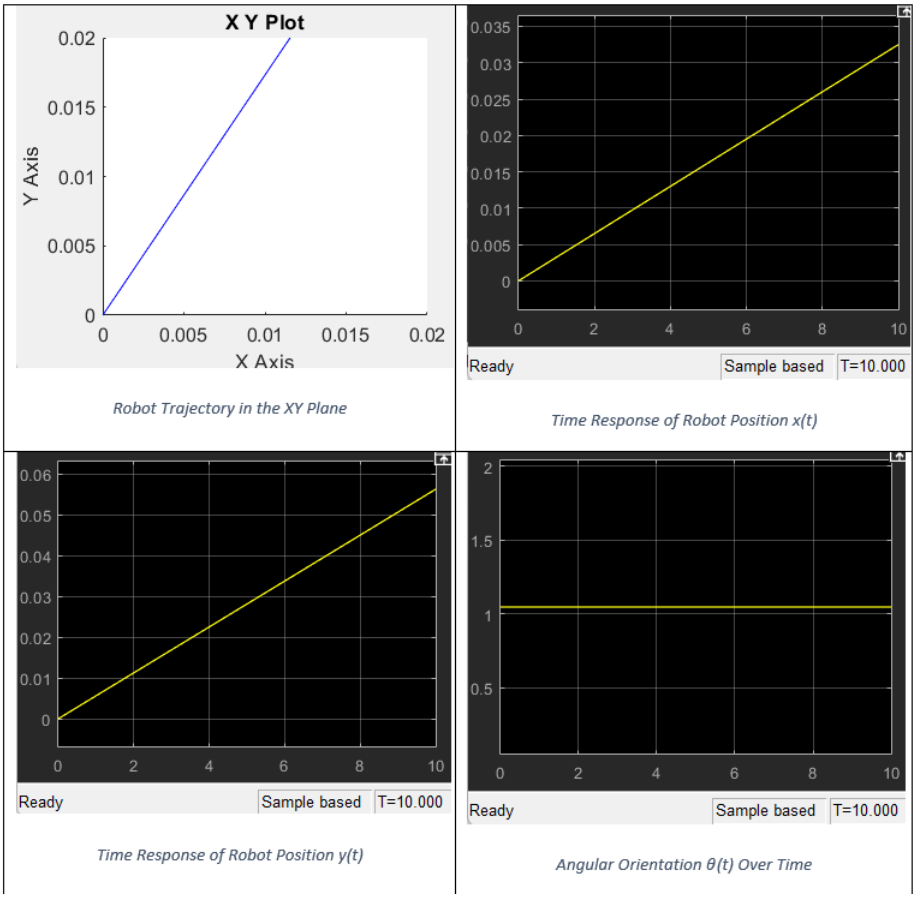


Figure 4.6: Straight motion with non-zero initial orientation

4.2.1.3 Case 3:Curved motion - unequal wheel speeds - non-zero orientation

In this case, the wheels are driven at different speeds: $V_R = 0.9$ m/s, $V_L = 0.2$ m/s, and the initial orientation is $\theta(0) = \pi/3$. The unequal speeds produce a nonzero angular velocity ($\omega \neq 0$), so the robot follows a curved path. According to the kinematic equations, the robot's forward speed and rotation rate are

$$v = \frac{V_R + V_L}{2}, \quad \omega = \frac{V_R - V_L}{2L},$$

which leads to motion combining translation and rotation:. In the simulation, $\theta(t)$ increases steadily over time, indicating continuous turning. The $x(t)$ and $y(t)$ responses are no longer linear; instead they curve, reflecting the circular component of motion. The resulting trajectory in the XY-plane forms a circular path rather than a straight line. The initial orientation $\pi/3$ sets the starting angle of the circle, while the difference in wheel speeds determines its radius. This behavior is exactly what the differential-drive kinematics predict when $V_R \neq V_L$.

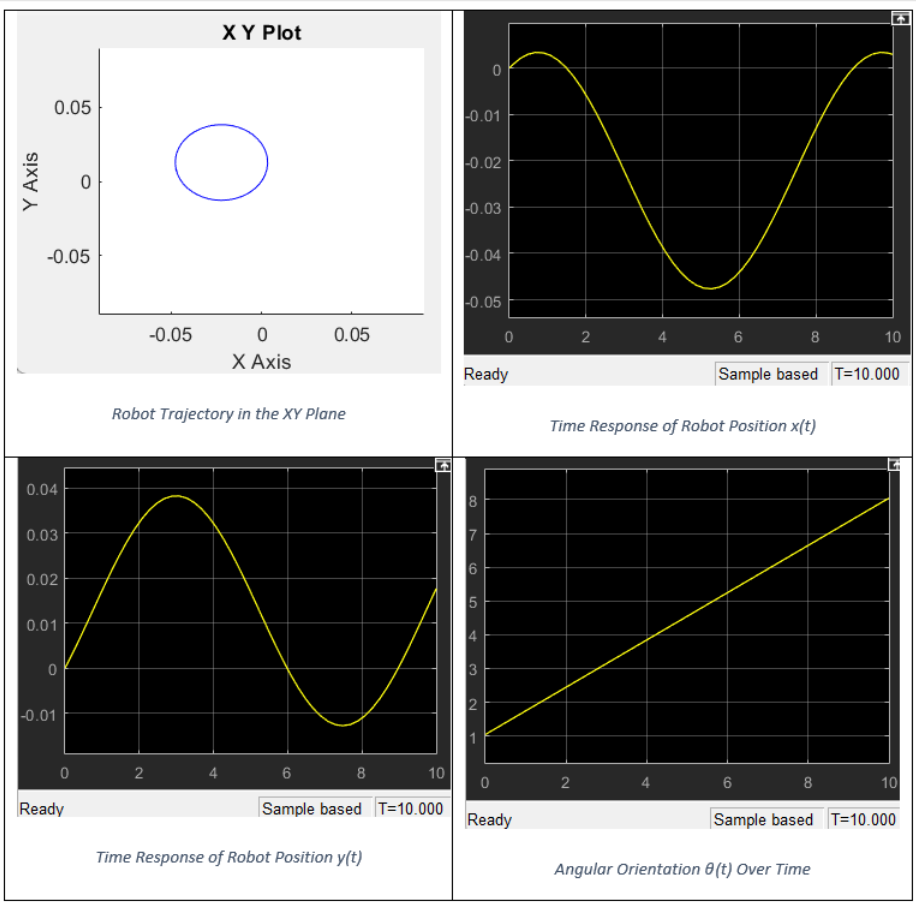


Figure 4.7: Curved motion with unequal wheel speeds and non-zero orientation

4.2.2 Simulation Results and Discussion

The three scenarios demonstrate the expected behavior of the differential-drive model. In Cases 1 and 2 (equal wheel speeds), the robot travels in a straight line: along the x-axis when $\theta = 0$, and along the $\pi/3$ direction when $\theta(0) = \pi/3$. In both cases, the angular velocity is zero and $\theta(t)$ remains constant, as seen in the simulations. In Case 3 (unequal wheel speeds), the robot undergoes combined translation and rotation, producing a curved trajectory. Throughout all simulations, the Simulink model outputs the robot's pose $[x(t), y(t), \theta(t)]$ consistent with theory.

These results confirm that the implemented kinematic model correctly captures the robot's motion under different wheel speed inputs, validating the step-response behavior

and overall model fidelity.

4.3 Conclusion

In this chapter, it is presented a comprehensive kinematic modeling of the differential-drive robot, detailing both its local and global motion representations. We derived the forward and inverse kinematics equations, showing how the wheel velocities determine the robot's linear and angular velocities, as well as its trajectory in the Cartesian plane.

Through simulation in MATLAB/Simulink, it is validated the accuracy of the kinematic model under different conditions. Three representative scenarios were analyzed: straight motion with zero initial orientation, straight motion with non-zero initial orientation, and curved motion with unequal wheel speeds. The results confirmed that the robot behaves exactly as predicted by the differential-drive kinematics. Equal wheel speeds produced straight-line motion along the expected direction, while unequal speeds resulted in circular motion, with the radius of curvature determined by the velocity difference and initial orientation.

Overall, this chapter establishes a solid foundation for controlling the differential-drive robot. The validated kinematic model provides the necessary framework for implementing higher-level control strategies, trajectory tracking, and autonomous navigation. In the next chapter, a line-following robot based on PID control and odometry-based feedback will be presented, developed using the previously validated kinematic model.

Chapter 5

Implementation of a Weighted-Average PID Controller and Experimental Validation

5.1 Introduction

The transition from a kinematic model to a functional autonomous robot necessitates a robust software implementation and a rigorous validation methodology. This chapter details the core of our project: the design, implementation, and performance analysis of a real-time control system for a differential-drive line-following robot.
The system's intelligence is built upon two pillars:

- **Sensor Algorithm:** A sensor algorithm based on a weighted-average method that provides a high-fidelity estimate of the robot's lateral deviation from the line. This approach supersedes traditional binary sensor logic, offering superior precision and smoothness.
- **PID Controller:** A Proportional-Integral-Derivative (PID) controller that translates the positional error into a differential drive command, enabling accurate and stable path tracking.

To quantitatively validate the controller's performance, we implemented a wheel odometry system using incremental encoders. This allows for the estimation of the robot's pose (x, y, θ) in the world frame. The recorded odometry data is subsequently visualized and analyzed using a custom Python script, providing an objective measure of the robot's trajectory and the effectiveness of our control strategy.
This chapter will thoroughly explain the mathematical foundation of the weighted-average algorithm, the PID control law, the odometry model, and their seamless integration in the microcontroller code. Finally, the experimental results will be presented and discussed to conclude on the system's overall performance.

5.2 Sensor Data : The Weighted-Average

The primary challenge in line following is accurately determining the robot's position relative to the line using discrete, noisy sensor inputs. To address this, it is employed a continuous estimation technique that provides a high-fidelity lateral position of the line, enabling smooth and precise control of the robot.

5.2.1 Sensor Configuration and Preprocessing

The robot is equipped with five IR analog sensors arranged linearly across its front. Each sensor returns a value v_i inversely proportional to the reflected IR light intensity, i.e., low value on the black line and high value on the white surface.
A fixed threshold T is empirically determined to distinguish between the line and the background. The sensor activation a_i is computed as:

$$a_i = \max(0, T - v_i) \tag{5.1}$$

where a_i is the activation value for sensor i. This formulation ensures that $a_i = 0$ when the sensor is off the line, and increases proportionally to the strength of line detection (i.e., the darkness of the surface beneath it).

5.2.2 Mathematical Formulation of the Weighted Average

Each sensor is assigned a predetermined weight w_i based on its lateral displacement (in centimeters) from the robot's central axis. For the symmetric sensor array, the calibrated weights are:

$$w = [-4.1, -1.8, 0.0, 1.9, 4.2]$$

The estimated lateral position of the line, x, is calculated as the centroid of the sensor activations:

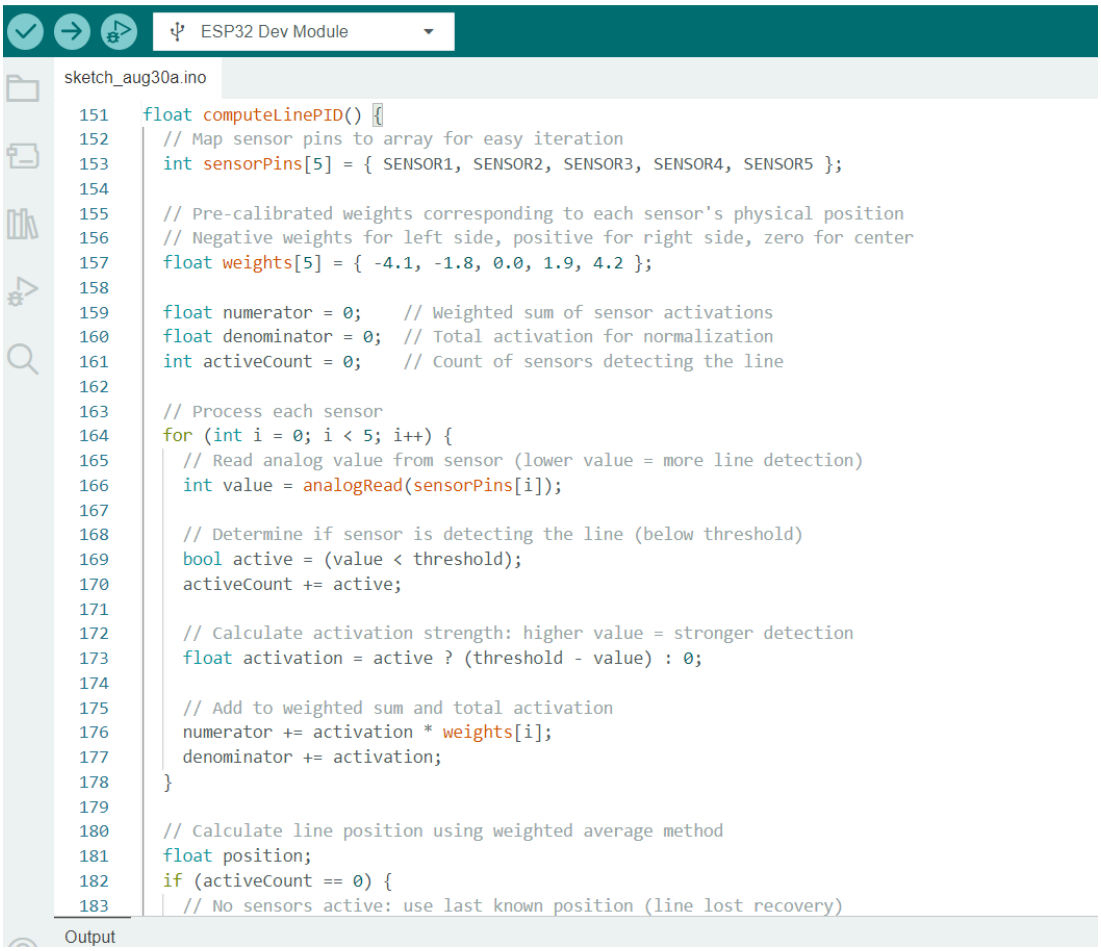
$$x = \begin{cases} \frac{\sum_{i=0}^{4} w_i \, a_i}{\sum_{i=0}^{4} a_i}, & \text{if } \sum_{i=0}^{4} a_i > 0 \\ x_{\text{last}}, & \text{if } \sum_{i=0}^{4} a_i = 0 \\ 0, & \text{if } a_i > 0 \text{ for all } i \end{cases} \tag{5.2}$$

where x_{last} represents the previously estimated position, allowing the algorithm to maintain continuity when the line is temporarily lost by all sensors.
This weighted-average method offers three key advantages:

- **High Resolution:** Provides a continuous position estimate, effectively achieving sub-sensor spacing resolution.
- **Smooth Response:** Small changes in the line's position result in proportional changes in x, eliminating the jerky motion caused by binary thresholding.
- **Inherent Robustness:** Normalization by the sum of activations reduces sensitivity to variations in ambient light or tape reflectivity.

5.2.3 Implementation in Microcontroller Code

The algorithm is efficiently implemented in the computeLinePID() function as shown in the picture




```
151  float computeLinePID() {
152    // Map sensor pins to array for easy iteration
153    int sensorPins[5] = { SENSOR1, SENSOR2, SENSOR3, SENSOR4, SENSOR5 };
154
155    // Pre-calibrated weights corresponding to each sensor's physical position
156    // Negative weights for left side, positive for right side, zero for center
157    float weights[5] = { -4.1, -1.8, 0.0, 1.9, 4.2 };
158
159    float numerator = 0;    // Weighted sum of sensor activations
160    float denominator = 0;  // Total activation for normalization
161    int activeCount = 0;    // Count of sensors detecting the line
162
163    // Process each sensor
164    for (int i = 0; i < 5; i++) {
165      // Read analog value from sensor (lower value = more line detection)
166      int value = analogRead(sensorPins[i]);
167
168      // Determine if sensor is detecting the line (below threshold)
169      bool active = (value < threshold);
170      activeCount += active;
171
172      // Calculate activation strength: higher value = stronger detection
173      float activation = active ? (threshold - value) : 0;
174
175      // Add to weighted sum and total activation
176      numerator += activation * weights[i];
177      denominator += activation;
178    }
179
180    // Calculate line position using weighted average method
181    float position;
182    if (activeCount == 0) {
183      // No sensors active: use last known position (line lost recovery)
```

Figure 5.1: Implementation of the weighted-average algorithm for line position estimation

5.3 PID Control: Theory and Tuning

The PID (Proportional-Integral-Derivative) controller is the fundamental component of our control system, designed to minimize the discrepancy between the desired position (setpoint, in our case the center of the line, $x_{set} = 0$) and the estimated position of the robot $x(t)$. By continuously adjusting the motor velocities based on the measured error, the PID controller ensures that the robot follows the line accurately and smoothly.

5.3.1 PID Control Law

The error signal is defined as the difference between the setpoint and the current position of the robot:

$$e(t) = x_{set} - x(t) = 0 - x(t) \tag{5.3}$$

The control signal $u(t)$ is computed as a weighted sum of three terms, corresponding to the proportional, integral, and derivative components of the controller:

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{5.4}$$

where:

- **Proportional (P), K_p:** The proportional term produces an output that is directly proportional to the current error. This provides an immediate corrective action. A higher K_p increases the system's responsiveness but may lead to overshoot and oscillatory behavior if excessively large. For our line-following robot, K_p allows rapid correction when the robot deviates from the center of the line.

- **Integral (I), K_i:** The integral term accumulates the error over time, addressing any steady-state offsets caused by unbalanced motors, sensor misalignment, or friction differences. By integrating past errors, the controller gradually corrects persistent deviations. However, a large K_i can introduce instability due to *integral windup*, where the integral term grows excessively large before the system can respond.

- **Derivative (D), K_d:** The derivative term predicts the future behavior of the error by evaluating its rate of change. This term provides damping, reduces overshoot, and enhances system stability. In practical terms, K_d helps the robot anticipate sharp turns on the line and avoid oscillations.

In the current project, it was chosen to use the parallel structure for the PID controller

because of its simplicity and the ability to set the proportional, integral and derivative actions independently

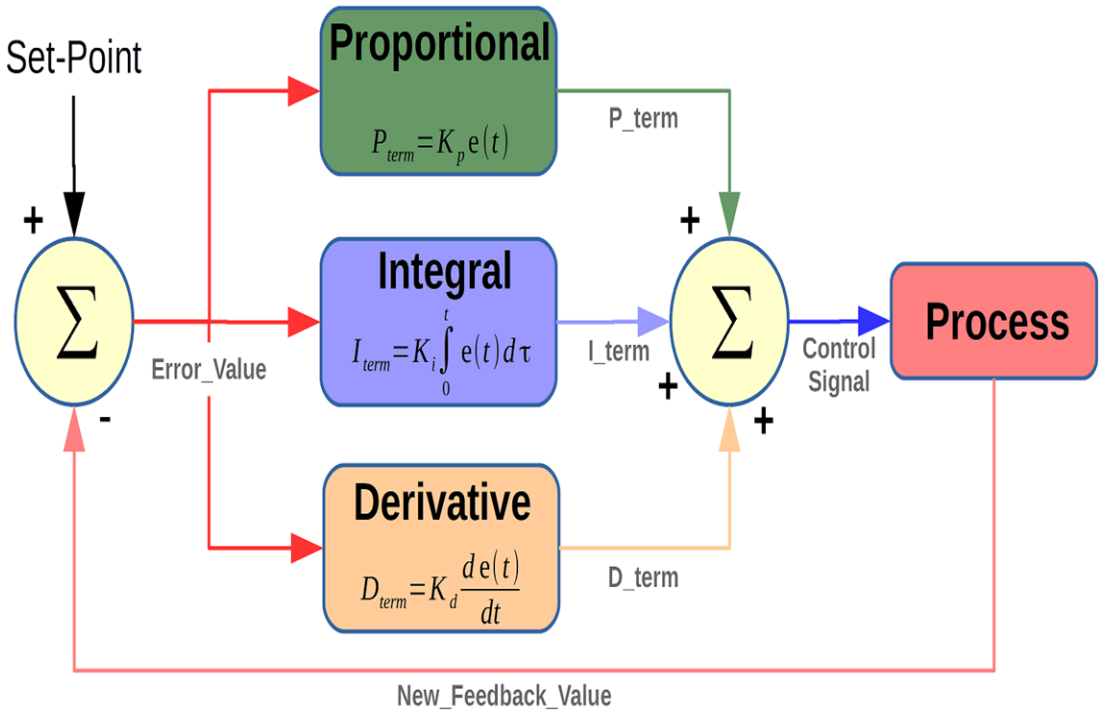


Figure 5.2: Parallel PID controller

Source: HelloPLC, 2020

5.3.2 Motor Control and Turning

The PID control output $u(t)$ is applied to modulate the speeds of the left and right wheels using differential steering:

$$\begin{aligned} v_{\text{left}} &= V_{\text{base}} + u(t) \\ v_{\text{right}} &= V_{\text{base}} - u(t) \end{aligned} \tag{5.5}$$

where V_{base} is the nominal forward speed of the robot. By increasing the speed of one wheel while decreasing the speed of the other, the robot can smoothly correct its trajectory to stay aligned with the line. This differential adjustment is directly proportional to the PID output, ensuring precise and responsive line following.

5.3.3 PID Controller Tuning Strategy

The theoretical PID control law provides a solid framework for trajectory tracking and error correction. However, its practical effectiveness depends entirely on the careful selection of the three gains: proportional (K_p), integral (K_i), and derivative (K_d). The process of finding suitable values, known as *tuning*, is critical for achieving a system that is simultaneously responsive, stable, and robust to disturbances.
In this study, a tuning strategy was adopted, which is widely recognized as an effective and intuitive method in practical robotics applications, where accurate system modeling is often challenging due to nonlinearities, friction, and sensor noise. The tuning strategy followed a structured, incremental process, described in detail below.

5.3.3.1 Initialization

At the beginning of the tuning procedure, the integral and derivative gains were set to zero:

$$K_i = 0, \quad K_d = 0$$

This effectively reduces the control law to a simple **Proportional (P) controller**. The base speed of the robot was set to a moderate value of $V_{base} = 83$ (PWM units). This ensured that the motors had sufficient speed to react quickly, while leaving enough room for speed differential during turns.

Proportional Gain (K_p) Tuning: With $K_i = 0$ and $K_d = 0$, The tuning process began by adjusting only the proportional gain K_p. The tuning procedure was as follows:

- **Low K_p:** Starting with a small value ($K_p = 2$), the robot responded sluggishly to deviations from the line. The corrective action was too weak, leading to overshooting the centerline and, in many cases, complete loss of the line, especially in curves.

- **Increasing K_p:** We gradually increased K_p to 5, then 10, and finally 15. As K_p increased, the robot's corrections became more rapid and assertive, enabling it to stay closer to the line.
- **Oscillation Onset:** Around $K_p \approx 15$, the robot began to oscillate around the line, even on straight sections. This "hunting" behavior is the hallmark of an excessively high K_p in a P-only controller.
- **Optimal Selection:** The ideal K_p lies just below the threshold of oscillation. We selected $K_p = 10$, which provided strong corrective action without destabilizing oscillations.

Derivative Gain (K_d) Tuning: The oscillations observed at higher K_p values can be dampened by the derivative term, which acts as a predictive element counteracting rapid changes in the error signal. We introduced K_d with the following observations:

- **Initialization:** K_d was introduced with a small initial value, approximately one-tenth of K_p (i.e., $K_d = 1$).
- **Effect of K_d:** As K_d was increased ($K_d = 2, 4, 6$), oscillations were progressively dampened. The robot's trajectory became smoother, with more controlled corrections and reduced overshoot.
- **Optimal Value:** An excessively large K_d amplifies sensor noise, particularly in environments with reflective surfaces or electrical interference. After experimentation, we determined that $K_d = 8$ provided the best trade-off: it dampened oscillations effectively while maintaining robustness against noise.

Integral Gain (K_i) Tuning: Even with well-tuned K_p and K_d, a small *steady-state error* may persist due to mechanical asymmetries such as uneven wheel diameters or misaligned sensors. The integral term addresses this by accumulating past error values:

- **Initialization:** We began with a very small integral gain, $K_i = 0.005$, to minimize the risk of instability.
- **Effect of K_i:** The integral term gradually reduced steady-state deviations, ensuring the robot stayed centered over longer distances.
- **Windup Risks:** Excessive K_i ($K_i > 0.05$ in our tests) led to *integral windup*, where accumulated error commanded excessive corrections, resulting in overshoot and eventual instability.
- **Final Value:** We selected $K_i = 0.01$, which was sufficient to compensate for long-term biases without destabilizing the system.

Final Tuned Gains: After systematic testing and incremental refinement, the final tuned parameters were:

$$K_p = 10, \quad K_i = 0.01, \quad K_d = 8$$

This combination achieved the desired balance between responsiveness, stability, and robustness. The proportional term provided strong corrective action, the derivative term ensured smooth damping of oscillations, and the integral term compensated for persistent small biases. Together, these gains enabled the robot to follow the line with high accuracy, as demonstrated in the experimental results.

5.3.3.2 Odometry Model

Odometry provides an estimate of the robot's pose (x, y, θ) in the global frame by integrating the wheel displacements over time. Each drive wheel is equipped with a quadrature magnetic encoder, producing a discrete number of pulses (or *ticks*) proportional to the wheel's rotation.

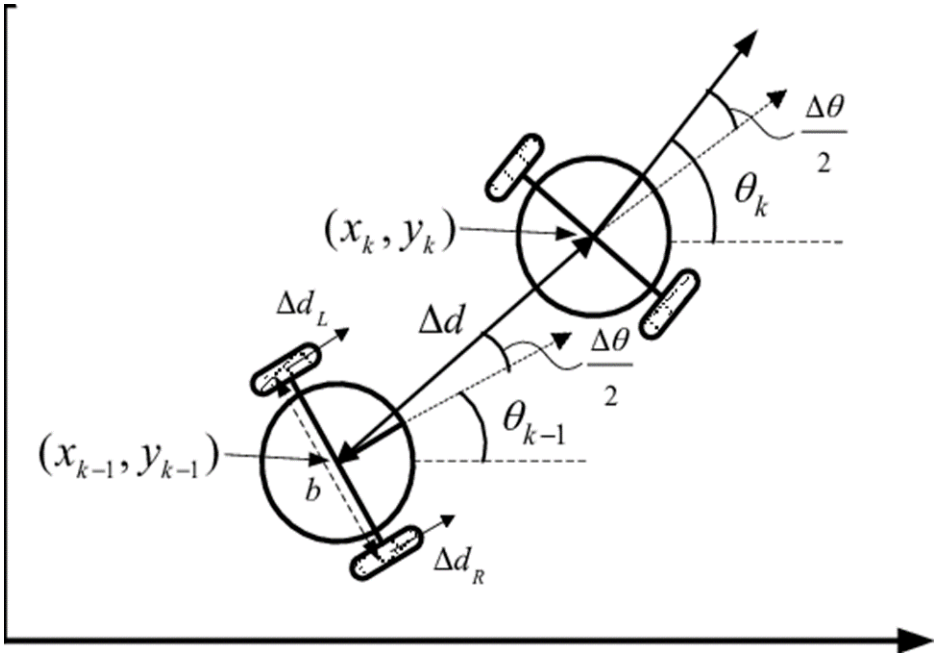


Figure 5.3: Odometry model of a two-wheel differential mobile robot

Source : Adapted from Lee et al. (2011) [23].

Let N_{enc} denote the number of encoder ticks per full revolution of a wheel.
The incremental displacement of each wheel, ΔS_r for the right wheel and ΔS_l for the left wheel, over a sampling interval Δt is given by:

$$\Delta S_r = \frac{\Delta \text{ticks}_r}{N_{enc}} \cdot \pi D, \qquad \Delta S_l = \frac{\Delta \text{ticks}_l}{N_{enc}} \cdot \pi D, \tag{5.6}$$

where D is the wheel diameter, and $\Delta\text{ticks}_r, \Delta\text{ticks}_l$ are the encoder increments measured during Δt.
From these, the linear and angular displacements of the robot are:

$$\Delta S = \frac{\Delta S_r + \Delta S_l}{2}, \qquad \Delta\theta = \frac{\Delta S_r - \Delta S_l}{L}, \tag{5.7}$$

with L being the distance between the two drive wheels (wheelbase).
To update the robot's global pose, the *midpoint integration method* is used. This method assumes that the robot follows a circular arc during Δt, and integrates motion

using the average orientation $\theta_k + \frac{\Delta\theta}{2}$, which significantly reduces integration :

$$\theta_{k+1} = \theta_k + \Delta\theta, \tag{5.8}$$

$$x_{k+1} = x_k + \Delta S \cdot \cos\Big(\theta_k + \tfrac{\Delta\theta}{2}\Big), \tag{5.9}$$

$$y_{k+1} = y_k + \Delta S \cdot \sin\Big(\theta_k + \tfrac{\Delta\theta}{2}\Big). \tag{5.10}$$

When $\Delta\theta \to 0$, these equations naturally reduce to straight-line motion:

$$x_{k+1} \approx x_k + \Delta S \cos(\theta_k), \quad y_{k+1} \approx y_k + \Delta S \sin(\theta_k).$$

5.3.3.3 Implementation and Data Logging

This odometry model was implemented in the function `updateOdometry()`, executed periodically at a fixed sampling rate (here, 10 Hz). The Arduino code below shows the core computations:

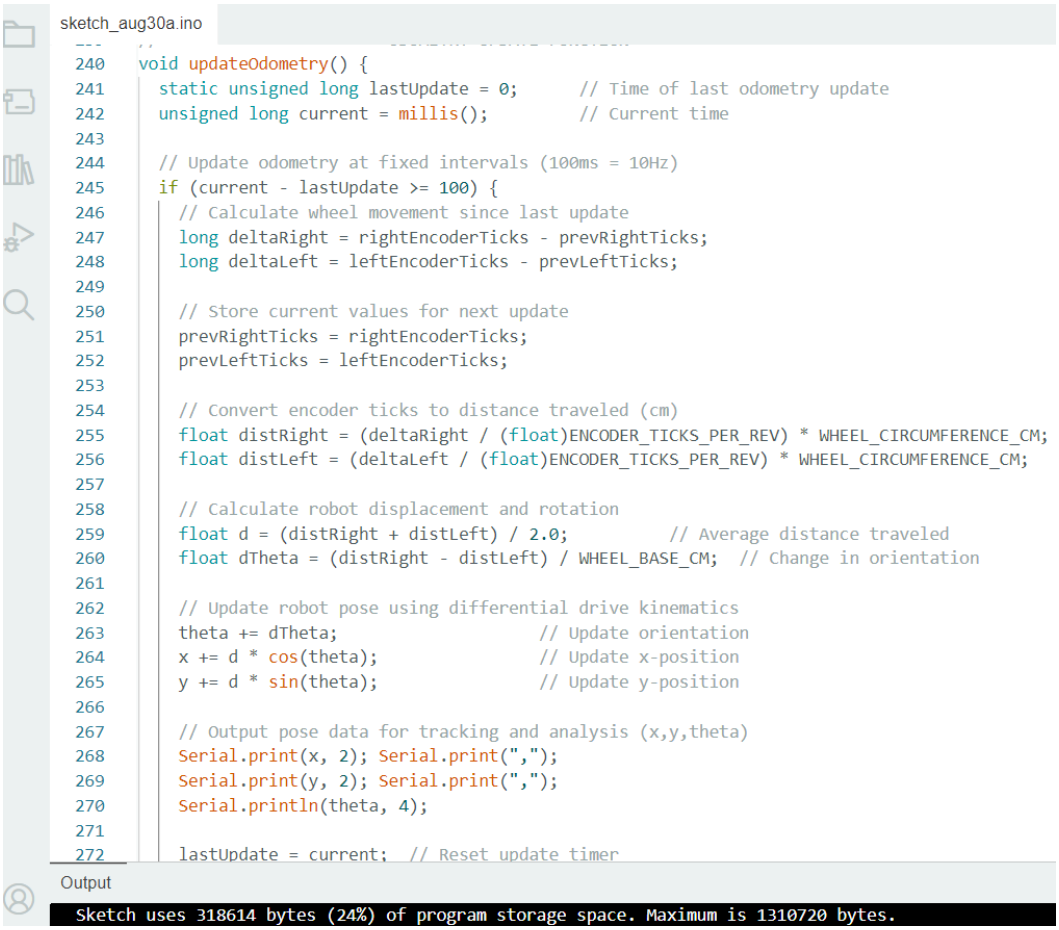


Figure 5.4: Core section of the odometry update function

The encoder resolution is represented in the firmware by the constant `ENCODER_TICKS_PER_REV`, which corresponds to N_{enc} in the mathematical model. The wheel circumference $C = \pi D$ is precomputed for efficiency.

Pose estimates (x, y, θ) are continuously streamed through the serial interface and logged on a host computer at 10 Hz. This logging allows for *post-processing analysis*, including trajectory reconstruction, drift quantification, and performance comparison with ground-truth measurements (e.g., from visual tracking or motion capture systems).

5.4 Experimental Results and Discussion

5.4.1 Experimental Setup and Data Logging

The proposed control architecture, combining the weighted-average PID algorithm and wheel odometry, was deployed on the Arduino microcontroller for real-time execution. During the experiment, the robot computed its pose (x, y, θ) incrementally from encoder ticks, while motor commands were continuously adjusted by the PID correction derived from the infrared sensor array.

To enable offline analysis, the estimated poses were transmitted via serial communication and recorded in a text file on a host computer. This log served as the data source for the subsequent visualization stage. In this way, the embedded control system operated independently in real time, while the evaluation of performance was deferred to post-processing.

5.4.2 Python-Based Trajectory Visualization

The collected odometry data were processed using a dedicated Python script, which transforms raw encoder logs into a visual trajectory. The procedure includes:

- **File Reading:** Parsing each line of the text file into numerical values of x, y, and θ.
- **Trajectory Construction:** Storing successive positions as arrays and connecting them to form the reconstructed path.
- **Orientation Representation:** Drawing arrows every 20th data point to indicate the heading.
- **Final Position Highlighting:** Marking the last recorded pose with a red point.
- **Graphical Output:** Plotting the complete trajectory using Matplotlib, as illustrated in the Figure.

This Python analysis complements the real-time Arduino implementation, enabling objective evaluation of the robot's performance.

5.4.3 Analysis of the Recorded Trajectory

The graph from odometry data shows the reconstructed trajectory. The blue curve represents the odometry-based path, red arrows indicate orientation, and the red dot marks the final position.

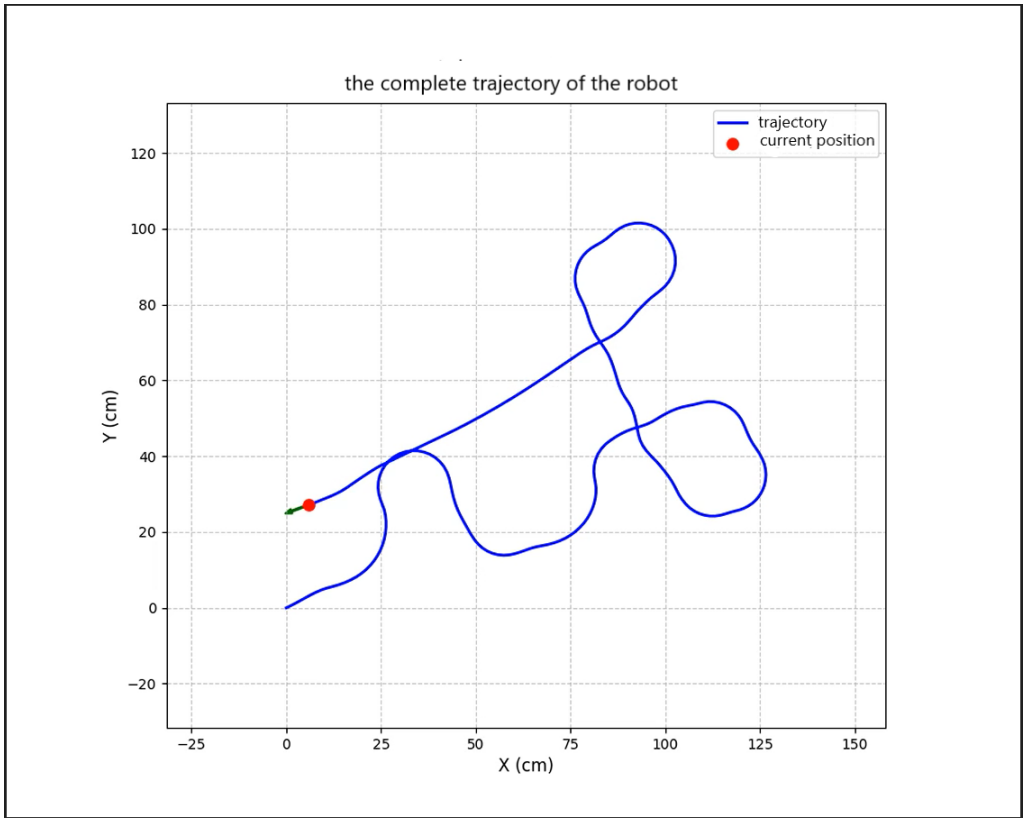


Figure 5.5: Reconstructed trajectory from odometry data

Key observations include:

- **Successful Map Completion:** The robot followed the entire path without losing the line, demonstrating robustness of the weighted-average PID controller.

- **PID-Related Deviations:** Minor lateral deviations occur in curves, reflecting the limitations of fixed-gain PID under varying curvature and sensor noise.
- **Straight-Line Drift:** Slight drift in straight segments is revealed only by odometry logs, highlighting the importance of offline analysis.
- **Oscillatory Behavior:** Some transitions show small oscillations typical of a PID near its stability boundary, but these remained bounded.
- **Cumulative Odometry Error:** Errors from wheel slip, encoder quantization, and imperfect calibration gradually accumulate, producing a small trajectory mismatch.

5.4.4 Sources of Error

The trajectory deviations of the line-following robot mainly originate from two categories of errors: control-related and odometry-related. These factors influence the accuracy, stability, and robustness of the robot's motion.

Control-Related Errors

- **Motor asymmetries and wheel misalignment:** Small differences in motor performance and slight mechanical misalignments of the wheels introduce persistent deviations from the reference trajectory. This leads to one wheel driving slightly faster than the other, producing curved motion instead of straight lines.
- **Sensor noise and sensitivity:** The infrared sensors (TCRT5000) are sensitive to lighting conditions, surface reflectivity, and electrical noise. Fluctuations in sensor readings can disturb the proportional and derivative terms of the control algorithm, causing oscillations .
- **Latency in control response:** Delays in processing sensor data and applying motor commands created a mismatch between the robot's actual state and the corrective action.

Odometry-Related Errors

- **Wheel slip:** During acceleration and sharp turns, the wheels slip against the surface, leading to discrepancies between measured encoder counts and actual displacement. .
- **Numerical integration drift:** The continuous integration of wheel displacements to estimate position accumulates errors over time, producing increasing deviation from the true trajectory.

5.4.5 Strengths of the Approach

Despite the imperfections, the method demonstrates several strengths:

- The weighted-average sensor fusion yields a smooth, continuous position estimate.
- The PID controller ensures reliable path following and robustness to noise.
- The Arduino-Python workflow allows real-time control with detailed offline analysis.

5.4.6 Perspectives for Future Work

Improvements can target:

- **Adaptive Control:** Using adaptive or fuzzy PID gains for varying path curvatures.
- **Error Quantification:** Employing external ground-truth tracking to compute numerical error metrics.
- **Odometry Reset:** Using map features (line intersections) to periodically correct odometry.

5.4.7 Conclusion of Experimental Validation

The experiments validate the effectiveness of the proposed system. The robot successfully completed the test map, confirming the reliability of the weighted-average PID controller. The Python-based trajectory reconstruction revealed small deviations and odometry drift, highlighting both the strengths and limitations of the approach. Integrating Arduino real-time control with Python offline visualization provides a comprehensive evaluation framework and uncovers subtle errors not detectable through visual inspection alone.

5.5 Conclusion

This chapter presented the implementation and validation of a weighted-average PID controller for a differential-drive line-following robot. By combining a continuous sensor algorithm with a systematically tuned PID controller, the system achieved smooth and accurate path tracking beyond what binary logic approaches typically allow. The integration of wheel odometry enabled real-time pose estimation and objective post-experimental analysis. Experimental results confirmed successful path following with only minor deviations due to sensor noise, motor asymmetries, and cumulative odometry errors—highlighting the trade-offs between responsiveness, stability, and robustness in PID-based control. Overall, the controller demonstrated strong performance, validating its suitability for line-following tasks. The Arduino–Python workflow effectively bridged real-time control with offline trajectory analysis, offering a solid methodology for debugging and performance evaluation. While the system meets the project objectives, further improvements such as adaptive control, ground-truth validation, and hybrid odometry correction remain promising directions for achieving greater accuracy and robustness in autonomous navigation.

Bibliography

[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, 2016.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[3] S. Khanna and R. Sharma, "Line following robot using pid controller," in *International Conference on Intelligent Communication and Computational Techniques (ICCT)*, IEEE, 2017, pp. 201–205.

[4] K. Ogata, *Modern Control Engineering*. Prentice Hall, 2010.

[5] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, 1995.

[6] D. A. et MELIK NASSIMA, "Study and design of a target-following robot based on pic16f877," OUM EL BOUAGUI University ,Biskra, 2012.

[7] M. Ceccarelli et al., "Tracked locomotion systems for ground mobile robots: A review," *Machines*, vol. 10, no. 8, p. 648, 2022.

[8] G. Vago, "Tracked and wheeled mobile robots: A comparative study," *International Robotics Journal*, 2013.

[9] Guoxing Intelligent Robotics, *Exploring the advantages of tracked chassis*, `https://www.gxsuprobot.com/Exploring-the-Advantages-of-Guoxing-Intelligent-Robot-Tracked-Chassis-id68318107.html`, Accessed: 2025-10-26, 2021.

[10] Xspire Robotics, *Wheeled chassis or tracked chassis: Which is better?* https://www.xspirebot.com/industry-news/wheeled-chassis-or-tracked-chassis.html, Accessed: 2025-10-26, 2023.

[11] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 3rd. MIT Press, 2022.

[12] V. J. E. Jiménez, G. Macher, D. Watzenig, and E. Brenner, "Safety of the intended functionality validation for automated driving systems by using perception performance insufficiencies injection," *Biomimetics*, vol. 6, no. 3, p. 55, 2021. DOI: 10.3390/biomimetics6030055 [Online]. Available: https://www.mdpi.com/2624-8921/6/3/55

[13] H. I. Christensen, "Slam: Simultaneous localization and mapping — path planning perspective," in *ICRA-02 SLAM Workshop*, PDF en ligne disponible, IEEE, May 2002, p. 11. [Online]. Available: http://hichristensen.com/hic-papers/icra02slam.pdf

[14] S. Thrun, J.-S. Gutmann, D. Fox, W. Burgard, and B. J. Kuipers, "Integrating topological and metric maps for mobile robot navigation: A statistical approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, ser. AAAI-98, Téléchargé en PDF, consulté en 2025, 1998. [Online]. Available: https://cdn.aaai.org/AAAI/1998/AAAI98-140.pdf

[15] A.-S. Documentation, *Line-follower robot – differential robot projects*, Acrome-SMD Docs, Site web, consulté avril 2025, 2025. [Online]. Available: https://docs.acrome.net/smd-applications/robotics/differential-robot-projects/line-follower-robot

[16] C. Stachniss, "Exploration and mapping with mobile robots," PhD Thesis, Ph.D. dissertation, Universität Bonn, Bonn, Germany, 2006.

[17] D. o. E. Narayana Engineering College, "Line-following robot carrying medicine for health care management system," in *Proceedings of the IndusEdu IJREISS*

Workshop, Document PDF en ligne, consulté en 2025, 2025. [Online]. Available: https://www.indusedu.org/pdfs/IJREISS/IJREISS_4108_37301.pdf

[18] M. Bilal, M. O. Khan, A. Mughal, and N. Ali, "Design and control of 6 dof robotic manipulator," B.Sc. Thesis, M.S. thesis, University of Engineering and Technology Lahore, Pakistan, 2018.

[19] B. B. Mevo, "Contribution à la commande adaptative et robuste d'un robot mobile de type avec modèle non linéaire," Ph.D. dissertation, École de Génie, Université du Québec en Abitibi-Témiscamingue (UQAT), Canada, May 2019.

[20] B. D. Hirpo and W. Zhongmin, "Design and control for differential drive mobile robot," *School of Mechanical Engineering, Tianjin University of Technology and Education*, Oct. 2017.

[21] M. Mekkaoui and M. K. Zemalache, "Dynamic sliding mode and backstepping controllers for trajectory tracking of wheeled mobile robot," *Przegląd Elektrotechniczny*, vol. 100, no. 4, pp. 221–225, 2024. DOI: 10.15199/48.2024.04.43

[22] S. Han, B. Choi, and J. Lee, "A precise curved motion planning for a differential driving mobile robot," *Mechatronics*, vol. 18, no. 9, pp. 486–494, 2008. DOI: <ÃăvÃľrifier>

[23] K. Lee, C. Jung, and W. Chung, "Accurate calibration of kinematic parameters for two wheel differential mobile robots," *Journal of Mechanical Science and Technology*, vol. 25, pp. 1603–1611, 2011. DOI: 10.1007/s12206-011-0334-y