



الجمهورية الجزائرية الديمقراطية الشعبية

Republique Algerienne Democratique Et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة الشهيد الشيخ العربي التبسي - تبسة -

Université Elchahide Cheikh Larbi Tébessi – Tébessa –

Faculté des Sciences et de la Technologie

Département de Génie électrique

## MEMOIRE

Présenté pour l'obtention du diplôme de Master Académique

En : Génie électrique

Spécialité : électronique d'instrumentation

Par : MAIFI YOUCEF

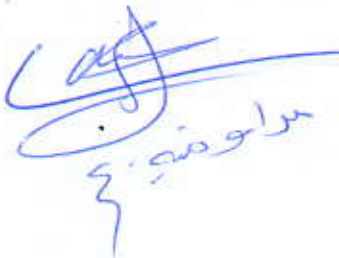
Sujet

**Contrôle visuel d'une base mobile pour le suivi de trajectoire et l'évitement d'obstacles.**

Présenté et soutenu publiquement, le 04/07/2023, devant le jury composé de :

MEROUMIA ABDALLAH  
DJABRI RIADH  
NEZZARI HASSAN  
MERABTI HALIM

Président  
Rapporteur  
Examineur  
Examineur

  
Maifi Youcef

Promotion : 2022/2023



الجمهورية الجزائرية الديمقراطية الشعبية

**People's Democratic Republic of Algeria**

وزارة التعليم العالي والبحث العلمي

**Ministry of Higher Education and Scientific Research**

العربي التبسي - تبسة - جامعة الشهيد الشيخ

**Elchahide Cheikh Larbi Tébessi University – Tébessa –**

**Faculty of Science and Technology**

**Department of Electrical Engineering**

## **MEMORY**

Presented for obtaining the **Academic Master 's degree**

**In: Electrical engineering**

**Specialty: instrumentation electronics**

**By : MAIFI YOUCEF**

### **Subject**

**Visual control of a mobile base for  
trajectory tracking and obstacle  
avoidance .**

Presented and presented publicly, on **04/07/2023** , before the jury composed of:

**MEROUMIA ABDALLAH  
DJABRI RIADH  
NEZZARI HASSAN  
MERABTI HALIM**

**President  
Rapporteur  
Examiner  
Examiner**

Promotion: **2022/2023**

# Abstract

Mobile robots have become increasingly common in recent years due to their ability to perform tasks in a variety of environments. These robots can be used in various applications such as transportation, surveillance, exploration, and many others.

In this project, our goal is to design and build a mobile robot with two wheels capable of navigating indoor environments. The robot is designed to be compact, lightweight, and agile, with the ability to move in any direction. The mechanical structure of the robot consists of two wheels, each driven by a separate motor, and a free-spinning wheel for stability. Additionally, the robot is equipped with sensors to enable it to navigate and interact with its environment.

The robot's software was developed using a combination of programming languages, including Python and C++. ROS (Robot Operating System) was also utilized, incorporating software modules for control, path tracking, obstacle detection and avoidance, as well as image processing. The project presented several challenges, including selecting suitable components and integrating various hardware and software modules. However, through the design, construction, and testing of the robot, we gained valuable experience in the field of robotics and learned important lessons that can be applied to future projects.

Overall, the mobile robot we built showcases the significant potential of robots in various aspects of life and highlights the importance of interdisciplinary collaboration in the development of complex systems.

**Keywords:** mobile robots, tracking, obstacle detection and avoidance, image processing.

# الملخص

أصبحت الروبوتات المتنقلة شائعة بشكل متزايد في السنوات الأخيرة بسبب قدرتها على أداء المهام في مجموعة متنوعة من البيئات. يمكن استخدام هذه الروبوتات في تطبيقات مختلفة مثل النقل والمراقبة والاستكشاف وغيرها الكثير.

في هذا المشروع، هدفنا هو تصميم وبناء روبوت متنقل يحتوي على عجلتين وقادر على التنقل في البيئات الداخلية. تم تصميم الروبوت ليكون صغير الحجم وخفيف الوزن ومتحرك، مع القدرة على الحركة في أي اتجاه. يتكون الهيكل الميكانيكي للروبوت من عجلتين، يتحرك كل منهما بواسطة محرك منفصل، وعجلة دوران حرة للثبات. كما تم تجهيز الروبوت بمستشعرات لتمكينه من التنقل والتفاعل مع بيئته.

تم تطوير برمجيات الروبوت باستخدام مزيج من لغتي البرمجة Python و C++. إضافة إلى نظام ROS، تشمل وحدات البرمجيات التحكم وتتبع المسار وكشف العوائق وتجنبها إضافة إلى معالجة الصور. قدم المشروع عددًا من التحديات، بما في ذلك اختيار المكونات المناسبة وتكامل وحدات الأجهزة والبرمجيات المختلفة. ومع ذلك، من خلال عملية التصميم والبناء واختبار الروبوت، تمكنا من اكتساب خبرة قيمة في مجال الروبوتات وتعلم دروس مهمة يمكن تطبيقها في المشاريع المستقبلية.

بشكل عام، يظهر الروبوت المتنقل الذي قمنا ببنائه الإمكانيات الكبيرة للروبوتات في تطبيقات مختلفة من نواحي الحياة ويسلط الضوء على أهمية التعاون المشترك بين التخصصات المختلفة في تطوير الأنظمة المعقدة.

**كلمات مفتاحية:** الروبوتات المتنقلة، تتبع المسار، كشف العوائق وتجنبها، معالجة الصور.

# Thanks

I would like to express my deep gratitude to **Allah** for granting me the breath of life, strength, health, and the intelligence required to complete this work.

Thank you, Professor **Nezzari Hassan**, for your guidance, knowledge, and support throughout this project. Your dedication to teaching has been invaluable, and I'm grateful for the opportunity to learn from you.

Thank you also to Professor **Merabti Halim** for guiding me remotely and providing me with the knowledge to complete the project.

Thanks are extended to all those who have taught and guided me throughout my academic journey until I reached this level.

Last but not least, I want to thank myself for the determination and hard work I've put into achieving my goals.

# Dedication

This thesis is devoted to:

My parents, whose unwavering support, assistance, trust, and motivation have been constants  
throughout my life,

My beloved sibling, who has provided unwavering encouragement and moral support,

My dear sisters, for their unswerving support and motivation,

Without all of you, I would have never reached this point,

I extend my heartfelt gratitude for everything you've done for me.

# Contents:

Abstract .....	iii
Thanks .....	v
Dedication .....	vi
List of Figures .....	vii
General Introduction .....	1
Chapter I: General information on mobile robotics definition and modeling	
I.1 Introduction: .....	3
I.2 Model of the robots: .....	3
I.2.1 Unicycle mobile robots: .....	3
I.2.2 Tricycle mobile robots: .....	4
I.2.3 Omnidirectional mobile robots: .....	4
I.3 Navigation and localization: .....	5
I.3.1 Navigation: .....	5
I.3.2 Localization: .....	5
I.3.3 Path planning: .....	5
I.4 Modeling of the unicycle mobile robot: .....	6
Chapter II: Predictive control and ROS	
II.1 The predictive control: .....	9
II.1.1 Introduction: .....	9
II.1.2 Basic principle of predictive control: .....	9
II.1.3 Nonlinear predictive control: .....	9
II.2 ROS (Robot Operating System): .....	11
II.2.1 Description: .....	11
II.2.2 Visualization tools: .....	12

## Chapter III: Simulation and experimental outcomes

III.1 Introduction: .....	14
III.2 Nonlinear predictive control: .....	15
III.2.1 Controller synthesis: .....	15
III.2.2 Implementation results: .....	15
III.3 ROS control: .....	18
III.3.1 Simulation with Gazebo: .....	18
III.3.2 Visualization with RViz: .....	19
General conclusion .....	20
References .....	21
Appendix A: Description of the mobile robot and accessories .....	22
Appendix B: The localization system .....	25
Appendix C: Particle Swarm Optimization .....	30
Appendix D: Description of the mobile robot .....	33



# List of Figures:

Fig. C.1: Unicycle mobile robot geometry .....	3
Fig. I.2: Tricycle mobile robot geometry .....	4
Fig. I.3: Omnidirectional mobile robot geometry .....	4
Fig. I.4: Illustration of the parameters .....	6
Fig. II.1: ROS Humble .....	11
Fig. III.1: Simulation of trajectory tracking NMPC with avoidance of fixed obstacles .....	16
Fig. III.2: Simulation of NMPC for control of wheel commands and angular velocity (robot).....	16
III.3: NMPC Trajectory Tracking Experimentation of the Robot .....	17
Fig. III.4: NMPC Experimentation of Wheel Motor Commands and Angular Velocity (robot) .....	17
Fig. III.5: The structure of the robot .....	18
Fig. III.6: 3D simulation of the robot in a house .....	18
Fig. III.7: 3D visualization of the robot and the sensor scans .....	19
Fig. III.8: 3D visualization of the mapping and navigation .....	19
Fig. A.1: Arduino UNO .....	23
Fig. A.2: Arduino RF-Nano .....	23
Fig. A.3: The NRF24L01 transceiver module .....	24
Fig. A.4: The L298N motor drive .....	24
Fig. A.5: The Pololu N20 gearmotor .....	24
Fig. A.6: Lithium batteries .....	24
Fig. B.1: Vision System .....	25
Fig. B.2: Calculate the center of gravity .....	26
Fig. B.3: Image acquired from the camera .....	28
Fig. B.4: Image after applying the Gaussian filter .....	28
Fig. B.5: The image in the HSV color space .....	28
Fig. B.6: Extraction of the blue color .....	28

Fig. B.7: Final result .....	29
Fig. C.1: The PSO illustration .....	30
Fig D.1: Raspberry pi 4 .....	33
Fig D.2: Arduino Nano .....	33
Fig D.3: RPLIDAR .....	33
Fig D.4: Camera .....	33
Fig D.5: Wheel connected with motor .....	34
Fig D.6: The L298N module .....	34
Fig D.7: LiPo Battery .....	34
Fig D.8: The internal components of the robot .....	35
Fig D.9: The robot's external appearance .....	35
Fig D.10: The bottom part of the robot .....	36
Fig D.11: The power circuits .....	36

# General Introduction

Mobile robotics is a rapidly evolving field that focuses on the development of robots capable of autonomous movement. These robots, equipped with a mobile base, combine various disciplines such as control engineering, mechanics, computer science, and electronics. They are designed to navigate and perform tasks in diverse environments.

Among the different types of mobile robots, wheeled robots are the most widely studied and utilized. Their simplicity and versatility make them suitable for a wide range of applications, including autonomous systems and robotics research.

The concept of mobile robotics encompasses two primary operating modes: teleoperation and autonomy. Teleoperated robots are controlled remotely, with commands transmitted through a control interface such as a keyboard or joystick. On the other hand, autonomous robots possess the ability to perceive their surroundings, make decisions, and navigate their environment without direct human intervention.

A key challenge in mobile robotics is ensuring the robot's ability to follow predefined trajectories while avoiding both fixed and mobile obstacles. This includes scenarios where multiple robots operate in the same environment, requiring coordination and collision avoidance mechanisms. Achieving autonomy in mobile robots has therefore become a major research focus, driven by the need for advanced navigation capabilities and the ability to adapt to dynamic environments.

In this project, we specifically concentrate on autonomous mobile robots. The main objective is to develop and implement control algorithms that enable trajectory tracking and obstacle avoidance. The study includes scenarios involving both fixed and mobile obstacles, with a particular emphasis on interactions between multiple robots within the same workspace. Two robots (“Bimo\_1” and “Bimo\_2”) were employed for experimentation.

The remainder of this thesis is structured as follows:

Chapter 1 provides a general overview of mobile robotics, including definitions and modeling approaches.

Chapter 2 introduces nonlinear predictive control and ROS control as viable control strategies for mobile robots.

Chapter 3 presents the results obtained from simulations and experiments conducted during the project.

Finally, the conclusion summarizes the findings and offers insights for future research directions.

# **Chapter I**

## **General information on mobile robotics definition and modeling**

## I.1 Introduction:

Mobile robotics is a field of robotics that involves the design, construction, and operation of robots that are capable of movement in a variety of environments. These robots can be used for a wide range of applications, from exploring space and underwater environments to assisting with manufacturing and logistics operations in warehouses.

Mobile robots are typically equipped with a variety of sensors, such as cameras, lidar, and ultrasound sensors, to help them navigate their environment and detect obstacles. They also often have onboard processors and algorithms that allow them to make decisions and perform tasks autonomously, without the need for human intervention.

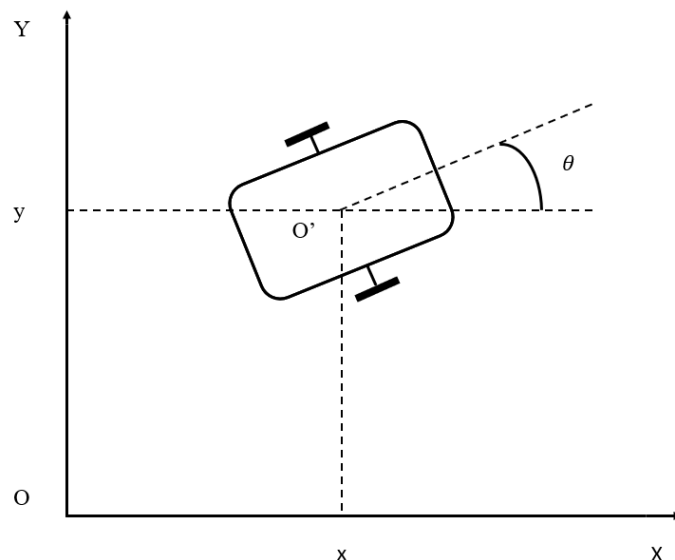
Some common examples of mobile robots include unmanned aerial vehicles (UAVs), autonomous cars, and delivery robots. Mobile robots are also used in a variety of other industries, such as agriculture, healthcare, and security. With advances in technology and artificial intelligence, the field of mobile robotics is constantly evolving, and is expected to play an increasingly important role in many aspects of modern society.

## I.2 Model of the robots:

In this section, we will present the model considered for each agent of the network of robots and the basics on consensus theory that are used in the control law's design:

### I.2.1 Unicycle mobile robots:

These are robots powered by two independently driven fixed wheels. A caster wheel is located on the axis connecting the two driven wheels to provide stability (Fig. I.1).



*Fig. I.1: Unicycle mobile robot geometry*

### I.2.2 Tricycle mobile robots:

This robot is powered by a single centrally mounted and always steerable wheel. It is driven and controlled by two fixed non-motorized wheels located at the rear on the same axis (Fig. I.2).

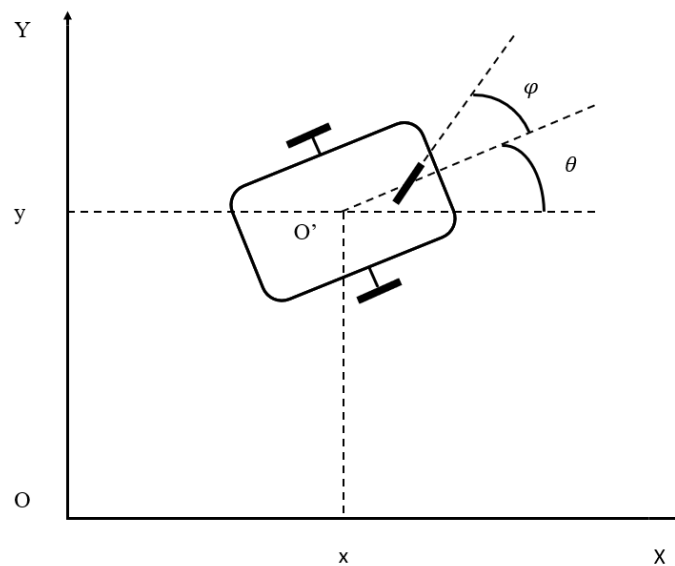


Fig. I.2: Tricycle mobile robot geometry

### I.2.3 Omnidirectional mobile robots:

The omnidirectional robot is equipped with a set of three steerable off-centered wheels or with three wheels placed at the vertices of an equilateral triangle (Fig. I.3).

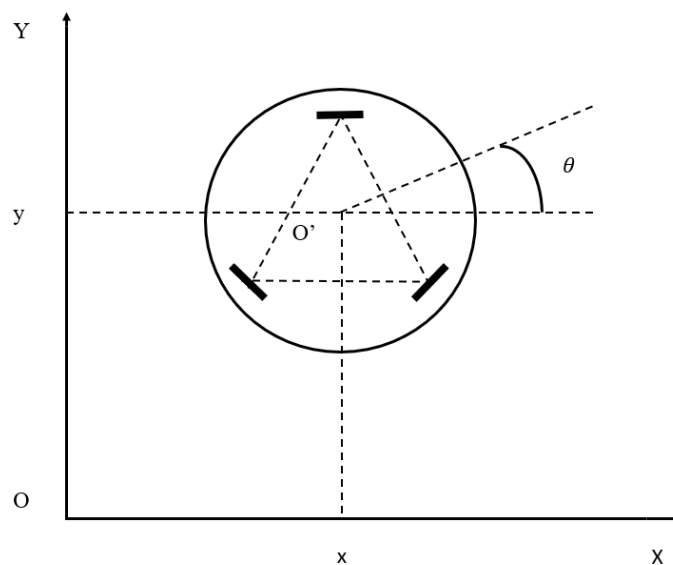


Fig. I.3: Omnidirectional mobile robot geometry

## **I.3 Navigation and localization:**

### **I.3.1 Navigation:**

The navigation problem consists of determining a directed path towards a goal. The representation of the environment based on sensory information is essential for this purpose. To perform autonomous navigation tasks, a mobile robot must possess a number of functionalities. It needs to localize itself within its environment, determine its position relative to fixed and moving obstacles present in the environment, and be able to avoid them.

### **I.3.2 Localization:**

Localization is necessary for the navigation of a mobile robot. The robot must have the ability to localize itself within its environment by estimating its position and orientation relative to a fixed reference point.

Localization approaches can be divided into two main classes:

- **Relative Localization:** The use of sensors allows determining the position of the robot at any given moment, given its initial position. The most common method is odometry, which involves using optical encoders mounted on the motor axes to measure wheel rotations.
- **Absolute Localization:** It is based on a global understanding of the environment. This can be achieved through technologies such as GPS (Global Positioning System), cameras, or rangefinders.

### **I.3.3 Path planning:**

Path planning involves determining how a robot will move and maneuver in a workspace. This problem involves calculating a collision-free path between a starting position and a destination position. An obstacle can be either static with respect to a fixed reference frame or dynamic (mobile) with respect to a fixed reference frame [1].

Path planning can be divided into two classes:

- **Local Path Planning:** It is performed while the robot is in motion, using data from local sensors. In this case, the robot has the ability to generate its path in response to changes in the environment.
- **Global Path Planning:** It is applicable when the environment (obstacles, etc.) is static and known. The trajectory is pre-determined, and the planning algorithm produces a complete path from the starting point to the destination point.

## I.4 Modeling of the unicycle mobile robot:

Among the various types of mobile robots, the unicycle robot has been chosen for practical tests or simulations. This robot is simple and widely used. In the following, we briefly present the kinematic model of this robot.

Let  $R = (O, \vec{x}, \vec{y})$  be an arbitrary fixed frame, and  $R' = (O', \vec{x}', \vec{y}')$  be a mobile frame attached to the robot, where  $O'$  represents the center of gravity of the mobile robot (Fig. I.4), typically the center of the axis of the drive wheels. This defines a position vector of the robot  $q$ :

$$q = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \quad (1.1)$$

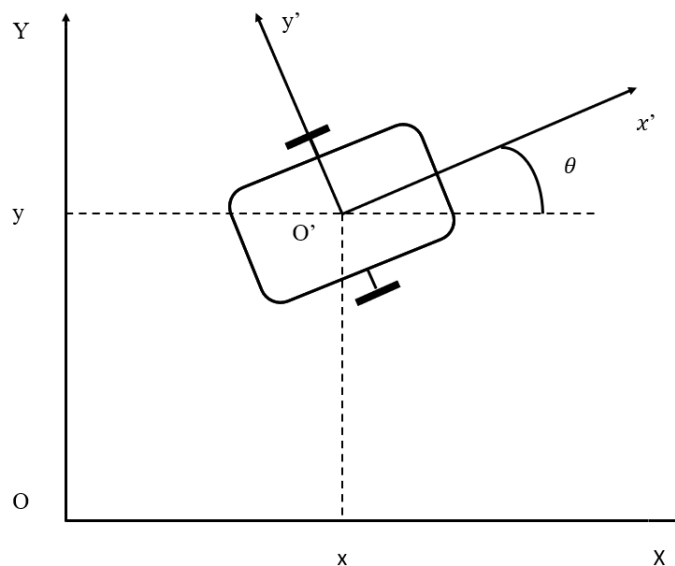


Fig. I.4: Illustration of the parameters

Generally, for the kinematic control of mobile robots, a velocity control model is commonly used.

In this case, there are no complex geometric or inertial parameters to identify. The calculation of the control is therefore simpler. The following assumptions are generally accepted:

- The mobile robot behaves like a rigid vehicle moving in a horizontal plane.
- The wheels are assumed to be rigid and roll without slipping on the ground.
- The contact of the wheel with the ground is reduced to a point, not a surface.

The instantaneous center of rotation (ICR) of the robot is a point with zero velocity, where it can rotate instantaneously. It is located on the axis of rotation of the wheels.



Let's consider:

- $\omega$ : The rotational velocity of the robot around the ICR (Instantaneous Center of Rotation).
- $v_r$  and  $v_l$ : Respectively, the velocities of the right and left wheels.

It is often important to relate the robot's pose to the control of its wheels. Generally, it is preferred to express this control using two other velocities:  $\theta$  (angular velocity) and  $v$  (longitudinal velocity). The relationships between these velocities are as follows:

$$v = \frac{(v_r + v_l)}{2} \quad (1.2)$$

$$\omega = \theta \quad (1.3)$$

Relating the derivative of the pose to the control  $V = (v \ \omega)^T$  is straightforward. A simple geometric consideration yield:

$$\dot{x} = v \cos \theta \quad (1.4)$$

$$\dot{y} = v \sin \theta \quad (1.5)$$

$$\dot{\theta} = \omega \quad (1.6)$$

This is the kinematic model of the mobile robot.

# **Chapter II**

## **Predictive control and ROS**

## II.1 The predictive control:

### II.1.1 Introduction:

The predictive control refers to a control strategy or technique that utilizes predictive models to anticipate future system behavior and make control decisions accordingly. It involves predicting the future states or outputs of a system based on current and past information, and then using this prediction to optimize control actions and achieve desired performance. The predictive control is commonly used in various fields such as process control, robotics, and autonomous systems.

### II.1.2 Basic principle of predictive control:

The basic principle of predictive control refers to the fundamental concept underlying the predictive control strategy. It involves using a predictive model of the system dynamics to predict future behavior and optimize control actions. The principle can be summarized as follows:

- **Prediction:** A mathematical model of the system is used to predict the future states or outputs based on the current and past information. This model takes into account the system dynamics, constraints, and disturbances.
- **Optimization:** An optimization algorithm is employed to determine the optimal control actions that will minimize a specified objective function. The objective function typically includes criteria such as tracking desired setpoints, minimizing control effort, and satisfying constraints.
- **Receding Horizon:** The control actions are calculated over a finite time horizon, but only the first control action is implemented. As time progresses, the horizon is shifted, and the process is repeated, allowing for adaptive and dynamic control.

By continuously updating the prediction and optimizing the control actions, predictive control aims to achieve optimal performance while accounting for system constraints and disturbances. This principle forms the basis for the design and implementation of predictive control algorithms in various applications.

### II.1.3 Nonlinear predictive control:

#### II.1.3.1 The general problem of NMPC:

Consider a nonlinear system described by the following discrete state model:

$$x(k + 1) = f(x(k), u(k)) \quad (2.1)$$

With :

- $x(k) \in R^n$  : System status.
- $u(k) \in R^m$  : Inputs.

The function  $f$  is assumed to be continuous.

Constraints can be imposed on the states and/or the control inputs. These constraints are expressed in the following form:

$$u(k) \in \mathcal{U} \quad (2.2)$$

$$x(k) \in \mathcal{X} \quad (2.3)$$

Where:

- $\mathcal{U}$  : is a convex compact set.
- $\mathcal{X}$  : is convex and closed.
- With:  $f(0,0) = 0$  .

The optimization problem associated with nonlinear predictive control is then given by:

$$\min_u J_N(x, k, \mathcal{U}) \quad (2.4)$$

The most common form of the cost function is:

$$J_N(x, k, \mathcal{U}) = F(x(k + N)) + \sum_{i=k}^{k+N-1} L(x(i), u(i)) \quad (2.5)$$

With :

- $N$  : Prediction horizon (which is equal to the input horizon).
- $F = x(k + N)$  : Cost on the final state  $x(k + N)$ .

Similar to the linear case, the solution provides a sequence of control inputs, and only the first element of the sequence is applied to the system.

### II.1.3.2 Solution methods:

As mentioned above, the optimization problem of nonlinear predictive control is generally non-convex. The classical method for solving this type of problem is the Sequential Quadratic Programming (SQP) algorithm, which is an extension of the active set method in quadratic programming. Some authors have proposed various methods for efficiently solving this problem. These include the multiple shooting method, the approach based on nonlinear least squares, and the use of fast metaheuristic algorithms such as Particle Swarm Optimization (PSO).

In this work, we employ the PSO algorithm for online optimization problem solving. The details of this algorithm are described in Appendix C.

## II.2 ROS (Robot Operating System):

In this chapter we present secondly, the development of a mobile robot with obstacle avoidance capabilities using ROS (Robot Operating System). The robot is built with a Raspberry Pi as the main control unit and integrates a LiDAR sensor and a camera for perception and obstacle detection, the description of the mobile robot is in the Appendix D. ROS enables seamless integration and control of the robot's hardware and software components, facilitating autonomous navigation while avoiding obstacles.

### II.2.1 Description:

ROS stands for Robot Operating System. It is an open-source framework and middleware widely used in the field of robotics for developing and controlling robots. ROS provides a collection of software libraries and tools that help developers create robot applications.



*Fig. II.1: ROS Humble*

Here are some key features and concepts related to ROS:

Nodes: ROS applications are composed of multiple nodes, which are independent processes that communicate with each other by passing messages.

Messages: Nodes communicate with each other by exchanging messages. Messages are data structures used to represent information such as sensor readings, motor commands, and other robot-related data.

Topics: Nodes can publish messages to topics or subscribe to receive messages. Topics act as message buses, allowing multiple nodes to communicate with each other indirectly.

Services: Services provide a request-response mechanism in ROS. A node can offer a service, and other nodes can send requests to that service and receive a response.

Actions: Actions extend the functionality of services by enabling long-running tasks with feedback. They are useful for executing complex actions that require ongoing communication between nodes.

Packages: ROS code is organized into packages, which are self-contained units that contain libraries, executables, configuration files, and other resources.

Launch files: ROS provides launch files that allow you to start multiple nodes with predefined parameters and configurations in a single command.

Visualization: ROS provides various visualization tools, such as RViz and Gazebo for visualizing robot models, sensor data, and other information in a graphical interface.

Community: ROS has a large and active community of developers who contribute to its development and provide support through forums, mailing lists, and other channels.

ROS supports multiple programming languages, including C++, Python, and more. It is widely used in research, academia, and industry for developing robotic systems and has a vast ecosystem of packages and libraries that can be leveraged to accelerate development.

## **II.2.2 Visualization tools:**

ROS provides several visualization tools that can be used to visualize various aspects of robotic systems and data. Here are some popular ROS visualization tools:

- RViz:

RViz (ROS Visualization) is a 3D visualization tool in ROS. It allows you to visualize robot models, sensor data, and other information in a 3D scene. RViz supports the visualization of robot models, sensor data (such as laser scans and point clouds), TF (transform) frames, and interactive markers. RViz is highly customizable and can be configured to display specific robot components or data according to your needs.

- Gazebo:

Gazebo is a powerful robotics simulator that integrates well with ROS. It provides a 3D graphical environment where you can simulate robots, sensors, and their interactions. Gazebo allows you to visualize and interact with simulated robots, and it provides tools for visualizing sensor data, environment models, and robot behavior. Gazebo's visualization capabilities can be used alongside other ROS tools for a comprehensive understanding of your robotic system.

These are just a few examples of ROS visualization tools available in the ROS ecosystem. ROS has a rich collection of visualization packages and tools that can be utilized based on the specific visualization needs. The selection of tools depends on the type of data that visualized, the level of interactivity required, and the specific aspects of the robotic system that analyze and understand.

# **Chapter III**

## **Simulation and experimental outcomes**

### III.1 Introduction:

In this chapter, we provide the simulation findings and experimental results for trajectory tracking with avoidance of stationary and mobile obstacles.

The robots (“Bimo\_1” and “Bimo\_2”) employed in the study are detailed in Appendix A and D.

The model for both robots aligns with the description provided in Chapter 1, which we reiterate below:

$$\begin{aligned} \dot{x} &= v \cos \theta \\ \dot{y} &= v \sin \theta \\ \dot{\theta} &= \omega \end{aligned} \tag{3.1}$$

With:

$$v = \frac{(v_r + v_l)}{2} \tag{3.2}$$

The two control signals are the speeds of the left wheels  $v_l$  and the right wheels  $v_r$ . These commands must satisfy the following constraints:

$$v_r \in [-0.5; +0.5](m/s) \tag{3.3}$$

$$v_l \in [-0.5; +0.5](m/s) \tag{3.4}$$

In this project, we consider two objectives:

- The objective of this project is to achieve trajectory tracking for the Bimo\_1 of a mobile robot.
- Additionally, the goal is to ensure trajectory tracking for the Bimo\_2 of a mobile robot operating in the environment, while effectively avoiding any fixe or moving obstacles present, and send a live topic of the robot environment to the user interface.

The robot’s localization is achieved using a vertically positioned camera placed above the robot, which captures top-down images of the robot along with an ArUco marker placed on it (Bimo\_1). The specifics of the localization system are provided in Appendix B.

We apply the control algorithm nonlinear predictive control with constraints.



## III.2 Nonlinear predictive control:

### III.2.1 Controller synthesis:

The prediction model used is the robot's kinematic model.

The cost function to minimize is in the following form:

$$J = \sum_{k=0}^{N_p} ((x(t+k) - x_r(t+k))^2 + (y(t+k) - y_r(t+k))^2) \quad (3.5)$$

#### The controller parameters:

The selection of controller parameters must ensure both algorithm convergence and computational efficiency in command calculations:

- Prediction horizon:  $N_p = 5$
- Control horizon:  $N_u = 1$

Regarding the Particle Swarm Optimization (PSO) algorithm:

- Number of particles: **15**
- Number of iterations: **30**

### III.2.2 Implementation results:

The autonomous mobile robot must exhibit efficient obstacle avoidance capabilities. The approach for calculating new commands involves introducing additional constraints on the distance between the mobile robot and the obstacle, ensuring that the robot does not approach the obstacle within a close range ( $d \leq 0.02 \text{ m}$ ).

#### III.2.2.1 Trajectory tracking with fixed obstacle avoidance:

##### Simulation:

The minimum allowable distance ( $d$ ) between the robot and the obstacle:

$$d = 0.02 \quad (3.6)$$

Figures (Fig. III.1) and (Fig. III.2) depict the trajectory tracking of a lemniscate path by the mobile robot with the avoidance of many fixed obstacles. The figures illustrate the commands for the left and right wheel speeds, as well as the angular velocity. The trajectory tracking is performed flawlessly without any errors, and the commands exhibit smooth transitions.

When the robot approaches an obstacle, a sudden change in the speed of one of the wheels is observed, depending on the orientation of the robot.

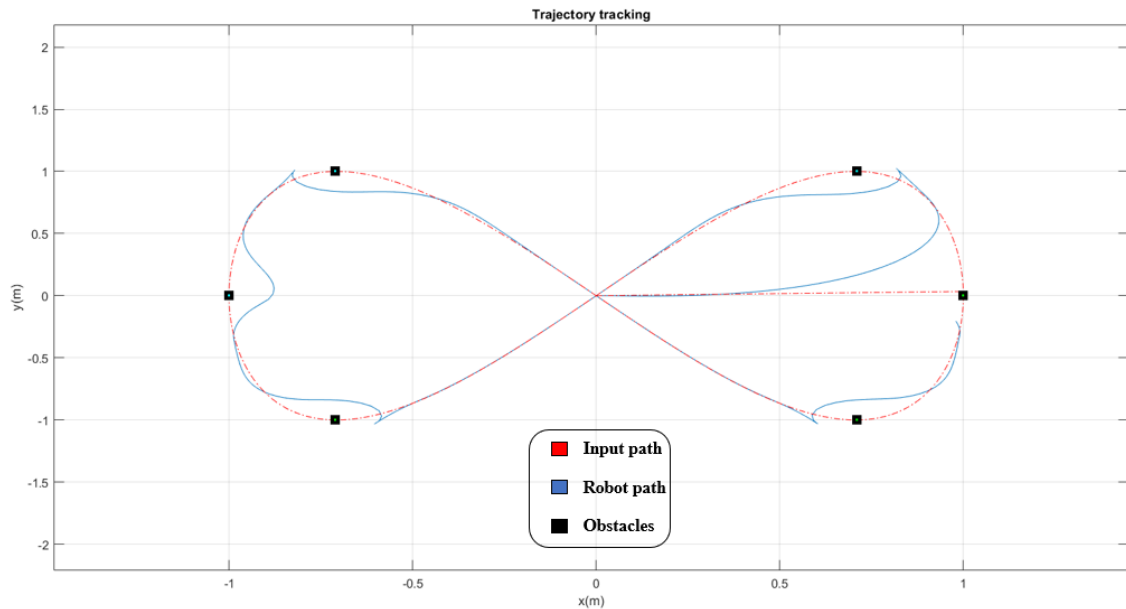


Fig. III.1: Simulation of trajectory tracking NMPC with avoidance of fixed obstacles

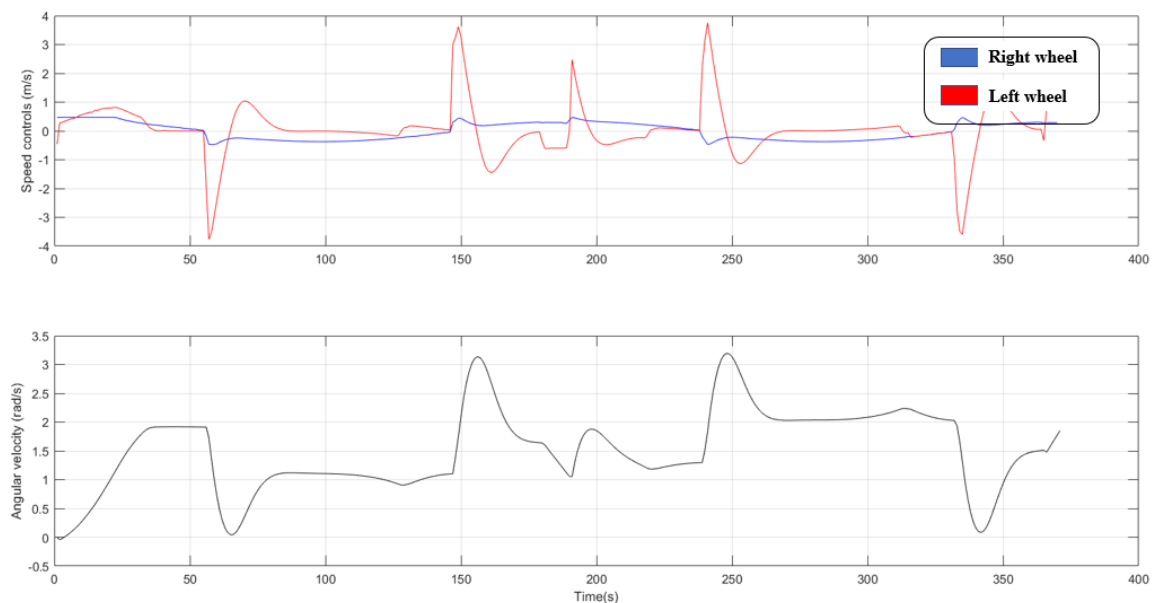


Fig. III.2: Simulation of NMPC for control of wheel commands and angular velocity (robot)

### Experimentation:

We keep the same parameters used in the simulation. The sampling period is set to 100ms, which meets the requirements for command calculation and the Shannon theorem.

The two figures (Fig. III.3) and (Fig. III.4) represent, respectively, the trajectory tracking of the mobile robot following a lemniscate path, the velocities of the left and right drive wheels, as well as the angular velocity.

The trajectory tracking was almost perfect, except for small errors observed during turns, which are considered acceptable. The commands are not smooth.

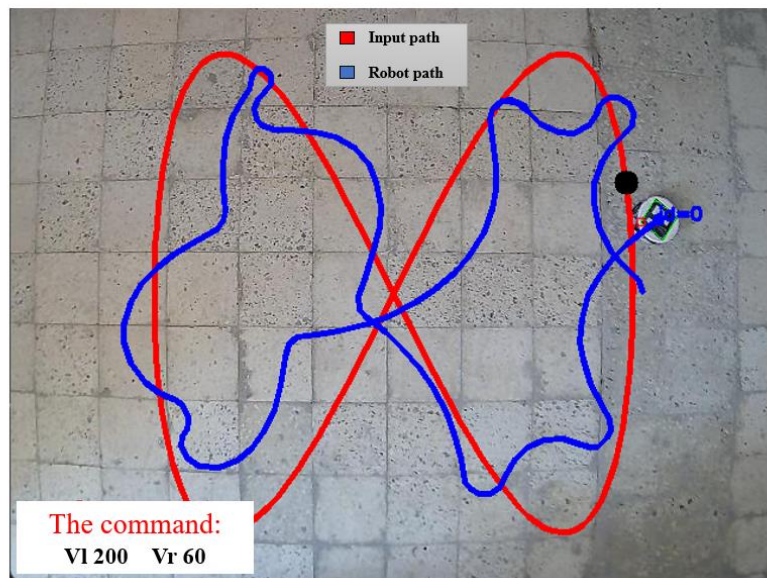


Fig. III.3: NMPC Trajectory Tracking Experimentation of the Robot

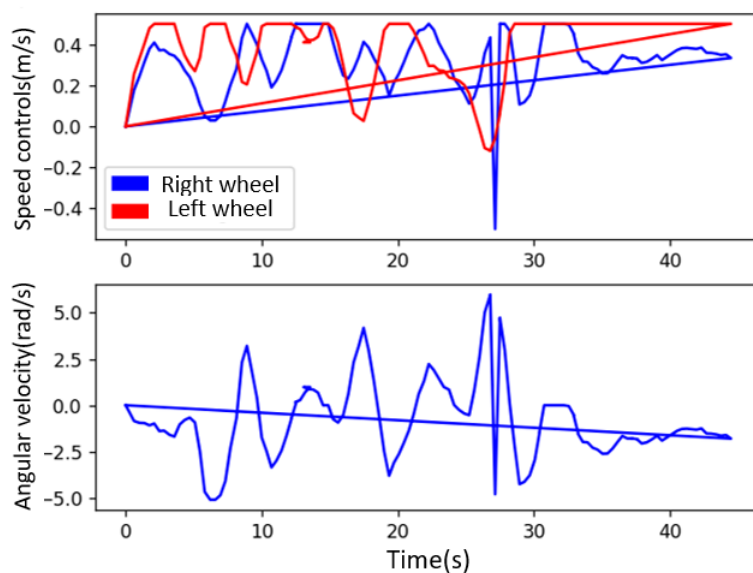


Fig. III.4: NMPC Experimentation of Wheel Motor Commands and Angular Velocity (robot)

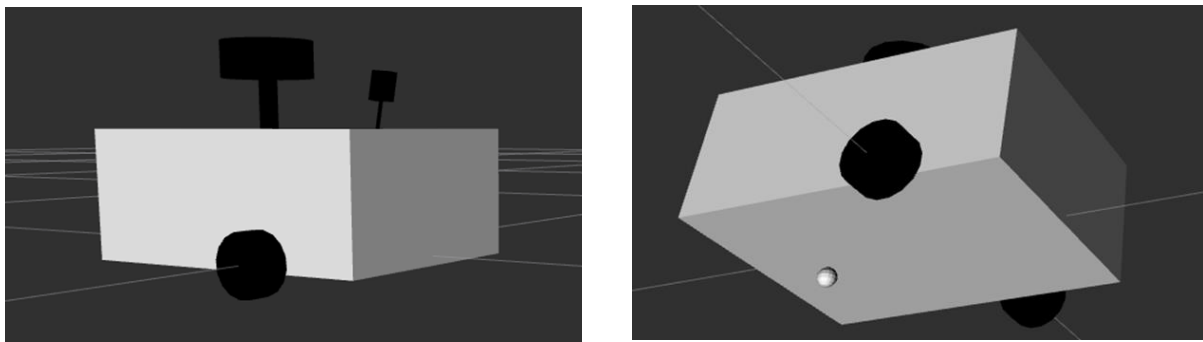
### III.3 ROS control:

Because of the unavailability of essential electronic components required to finish the project in the Algerian market, we conducted a simulation of the robot within the ROS integrated environment and achieved the results illustrated below.

The assembly and arrangement of all the robot's components are detailed in Appendix D.

#### III.3.1 Simulation with Gazebo:

To create a realistic simulation environment, we designed a model of a house for the robot's presence. Additionally, we modeled the robot and its accessories, including the LiDAR sensor and camera, as depicted in the following image:



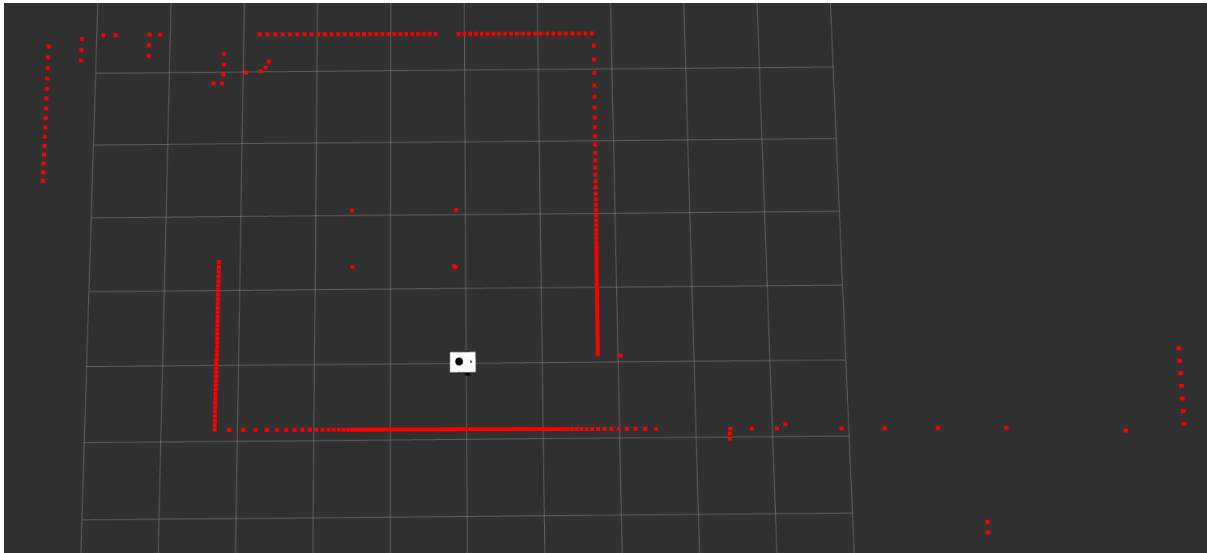
*Fig. III.5: The structure of the robot*



*Fig. III.6: 3D simulation of the robot in a house*

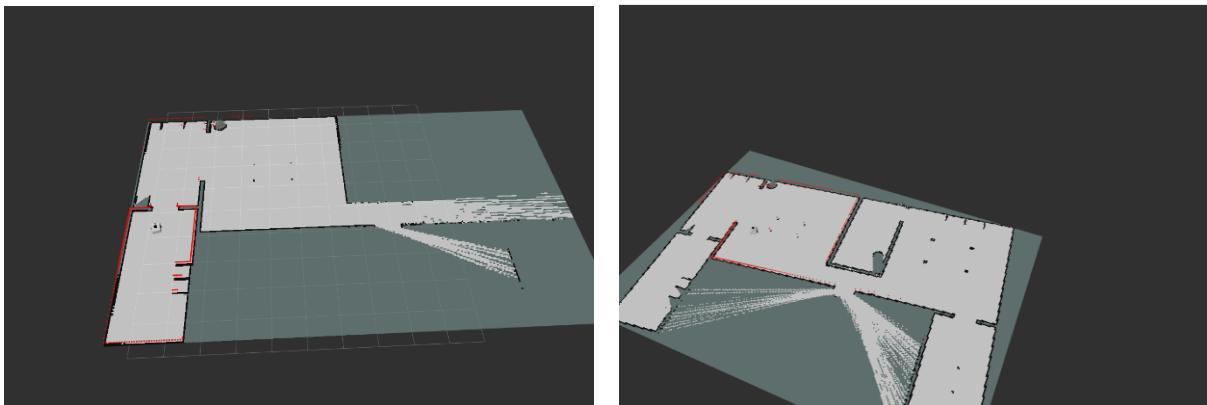
### III.3.2 Visualization with RViz:

To make a 3D visualization we use RViz, is tool commonly used with the Robot Operating System (ROS) for simulating and debugging robotic applications. It allows us to visualize various data related to a robot's sensors, movement, and the environment in a 3D environment.



*Fig. III.7: 3D visualization of the robot and the sensor scans*

Alternatively, we use ROS tools for mapping and navigation with the SLAM toolbox and Nav2.



*Fig. III.8: 3D visualization of the mapping and navigation*

# General conclusion

In this end-of-study project, our main focus has been on achieving effective trajectory tracking while avoiding both fixed and moving obstacles. This is a critical challenge in the field of mobile robotics, as it involves navigating environments with a mix of stationary and moving objects, such as humans or other robots. The primary goal is to prevent any potentially dangerous collisions.

Nonlinear Model Predictive Control (NMPC) emerged as a suitable choice due to its predictive capabilities and adaptability to multivariable systems. However, a significant challenge was the computational time required, which we addressed by incorporating the Particle Swarm Optimization (PSO) algorithm. This optimization technique effectively reduced the computation time, particularly for shorter prediction horizons, and yielded highly satisfactory results.

The utilization of the Robot Operating System (ROS) in the development of mobile robots represents a significant advancement in the field of robotics. ROS provides a versatile and comprehensive framework that facilitates the creation of highly capable and adaptable robotic systems.

Mobile robots equipped with ROS have the potential to revolutionize various industries, from manufacturing and logistics to healthcare and exploration. They can enhance efficiency, safety, and precision in a multitude of applications, contributing to advancements in automation, surveillance, inspection, and more.

As a potential avenue for future exploration, we propose combining Bimo\_1 with Bimo\_2. This integration would allow us to leverage the respective advantages of both approaches and potentially enhance overall control performance.

# References

- [1] Tzafestas, Spyros G. Introduction to mobile robot control. Elsevier, 2013.
- [2] Bayle, Bernard. "Robotique mobile." Ecole Nationale Supérieure de Physique de Strasbourg, Université Louis Pasteur, France (2008) : 2007-2008.
- [3] Zidani, Ghania. Commande Robuste d'un Robot Mobile à Roues. Diss. Université de Batna 2, 2017.
- [4] Lefebvre, Olivier. Navigation autonome sans collision pour robots mobiles nonholonomes. Diss. Institut National Polytechnique de Toulouse-INPT, 2006.
- [5] E.F. Camacho, C. Bordons, "Model predictive control", Ed. Springer-Verlag, London, 2004.
- [6] H. Merabti, commande prédictive par la théorie des intervalles flous et métaheuristiques, Thèse Doctorat en Sciences Faculté des sciences de l'ingénieur, Université des frères Mentouri Constantine, 2015.
- [7] Garrido-Jurado, Sergio, et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion." Pattern Recognition 47.6 (2014): 2280-2292.
- [8] El Dor, Abbas. Perfectionnement des algorithmes d'optimisation par essaim particulaire : applications en segmentation d'images et en électronique. Diss. Université Paris-Est, 2012.

# Appendix A

## Description of the mobile robot and accessories

### A.1 Description of the system:

The robot consists of a fiberglass chassis with a circular shape, equipped with a caster wheel to ensure the balance of the robot, and two independent drive wheels powered by DC motors (Fig. A.5). The motors are powered by two lithium batteries (Fig. A.6) through an L298N motor driver (Fig. A.4). These two wheels allow the robot to move forward, backward, and turn left or right.

A suspended camera is employed to provide localization and obstacle detection for the robot. The acquired images are analyzed and processed using Python and the OpenCV library. The processing tasks, as well as the computation of control signals, are performed on a computer system equipped with an Intel® Core i5 processor operating at 3.4 GHz and 8GB of RAM. The control signals are transmitted to the robot using the NRF24L01 radio transceiver module.

### A.2 The accessories:

#### A.2.1 Arduino:

Two types of Arduino boards were used in the project. An Arduino UNO board (Fig. A.1) was employed to receive commands from the computer through a serial port and transmit them via the NRF24L01 module. These commands were then received by another Arduino RF-Nano board (Fig. A.2) mounted on the robot. The Arduino RF-Nano board decoded the commands and forwarded them to the L298N motor driver for execution.

The Arduino UNO board, with its versatile capabilities and built-in USB interface, served as the main communication interface between the computer and the robot. It received the commands from the computer through the serial port, allowing for easy and reliable data transfer.

#### A.2.2 The NRF24L01 transceiver module:

The NRF24L01 is a wireless transceiver module that provides both transmission and reception capabilities. It operates in the 2.4 GHz frequency range and uses the Nordic Semiconductor's Enhanced Shock Burst™ protocol for efficient data transfer.

The NRF24L01 module (Fig. A.3), a wireless transceiver, was connected to the Arduino UNO board. It enabled wireless communication between the computer and the robot, providing a convenient way to send commands without the need for physical connections.

On the robot side, the Arduino RF-Nano board received the commands transmitted by the Arduino UNO board via the NRF24L01 module. The Arduino RF-Nano board acted as the



control unit for the robot, decoding the received commands and sending appropriate signals to the L298N motor driver.

### A.2.3 The L298N motor drive:

The L298N (Fig. A.4) is a dual H-bridge motor controller that can be easily controlled using TTL (Transistor-Transistor Logic) signals from Arduino RF-Nano. It is designed to control the speed and direction of two DC motors simultaneously.

The L298N module has several input pins that can be connected to Arduino RF-Nano digital output pins to control the motor operation. These input pins include two enable pins for enabling or disabling the motor outputs, and four control pins for setting the motor direction and speed.

By appropriately setting the input signals, the L298N module can control the rotation direction of each motor (forward or reverse) as well as adjust the motor speed using Pulse Width Modulation (PWM) signals. This allows for precise control over the movement of the motors.

### A.2.4 The motors:

The Pololu N20 gearmotor (Fig. A.5) is a miniature high-power DC motor with a 12V operating voltage. It features long-life carbon brushes and a metal gearbox with a gear ratio of 51.45:1. The motor has a compact size with dimensions of 10 × 12 mm, and the output shaft of the gearbox is in a D-shape, measuring 9 mm in length and 3 mm in diameter.

Here are the key specifications of the Pololu N20 gearmotor:

- No-load speed: 625 RPM at 100 mA.
- Stall torque: 1.1 kg-cm.
- Stall current: 0.8 A.



Fig. A.1: Arduino UNO

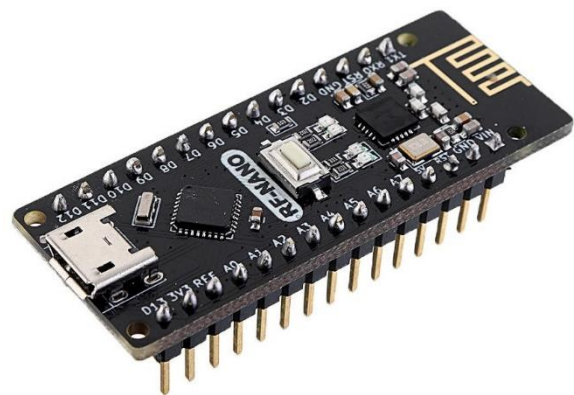
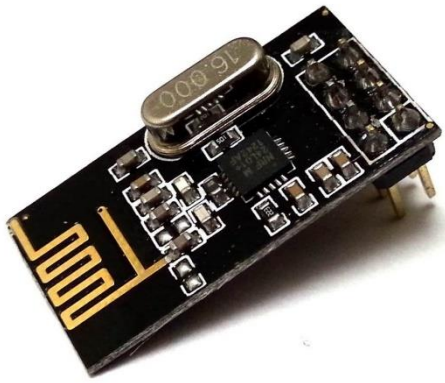
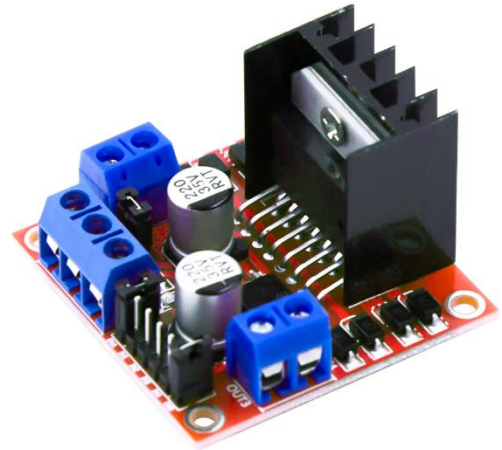


Fig. A.2: Arduino RF-Nano



*Fig. A.3: The NRF24L01 transceiver module*



*Fig. A.4: The L298N motor drive*



*Fig. A.5: The Pololu N20 gearmotor*



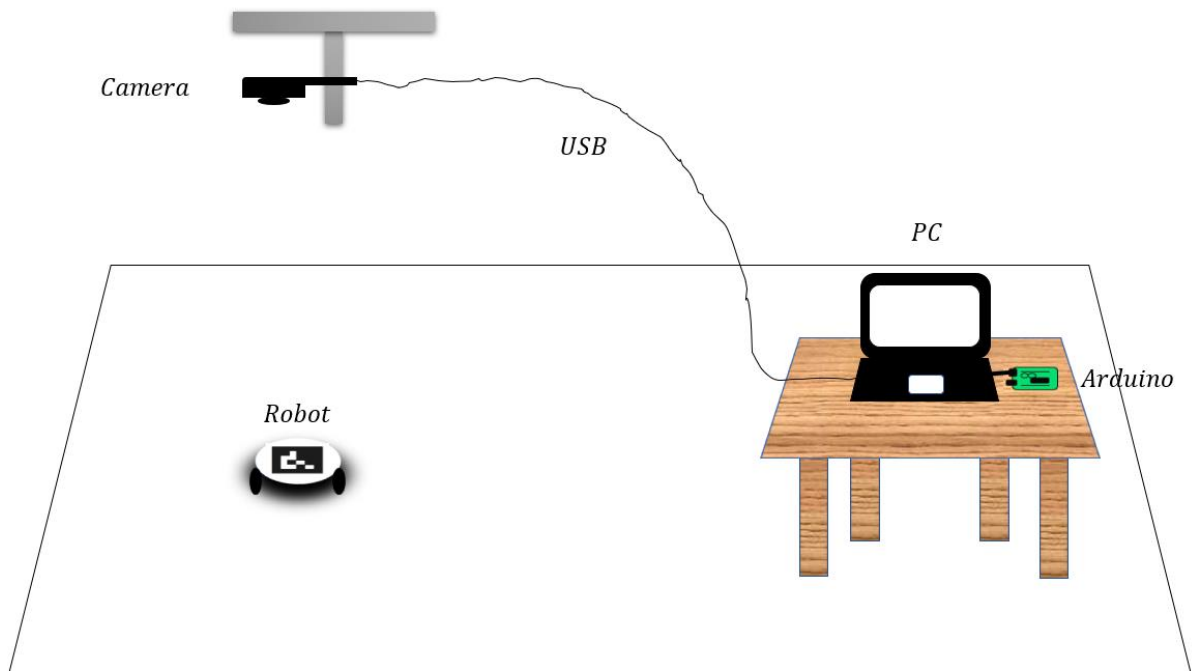
*Fig. A.6: Lithium batteries*

# Appendix B

## The localization system

### B.1 The robot localization:

The identification of the robot's position and orientation is achieved through the use of a vertically positioned camera placed above the robot. This camera captures top-view images of the robot and a marker called ArUco is placed on the robot for identification (Fig. B.1).



*Fig. B.1: Vision System*

#### B.1.1 ArUco Marker:

The ArUco module is based on the ArUco library, an open-source library for square fiducial marker detection developed by Rafael Muñoz and Sergio Garrido [7].

An ArUco marker is a type of fiducial marker used in computer vision and augmented reality applications. It is a square-shaped marker that consists of a black border and an inner pattern of binary bits. Each ArUco marker has a unique binary pattern, which allows for its identification and tracking.

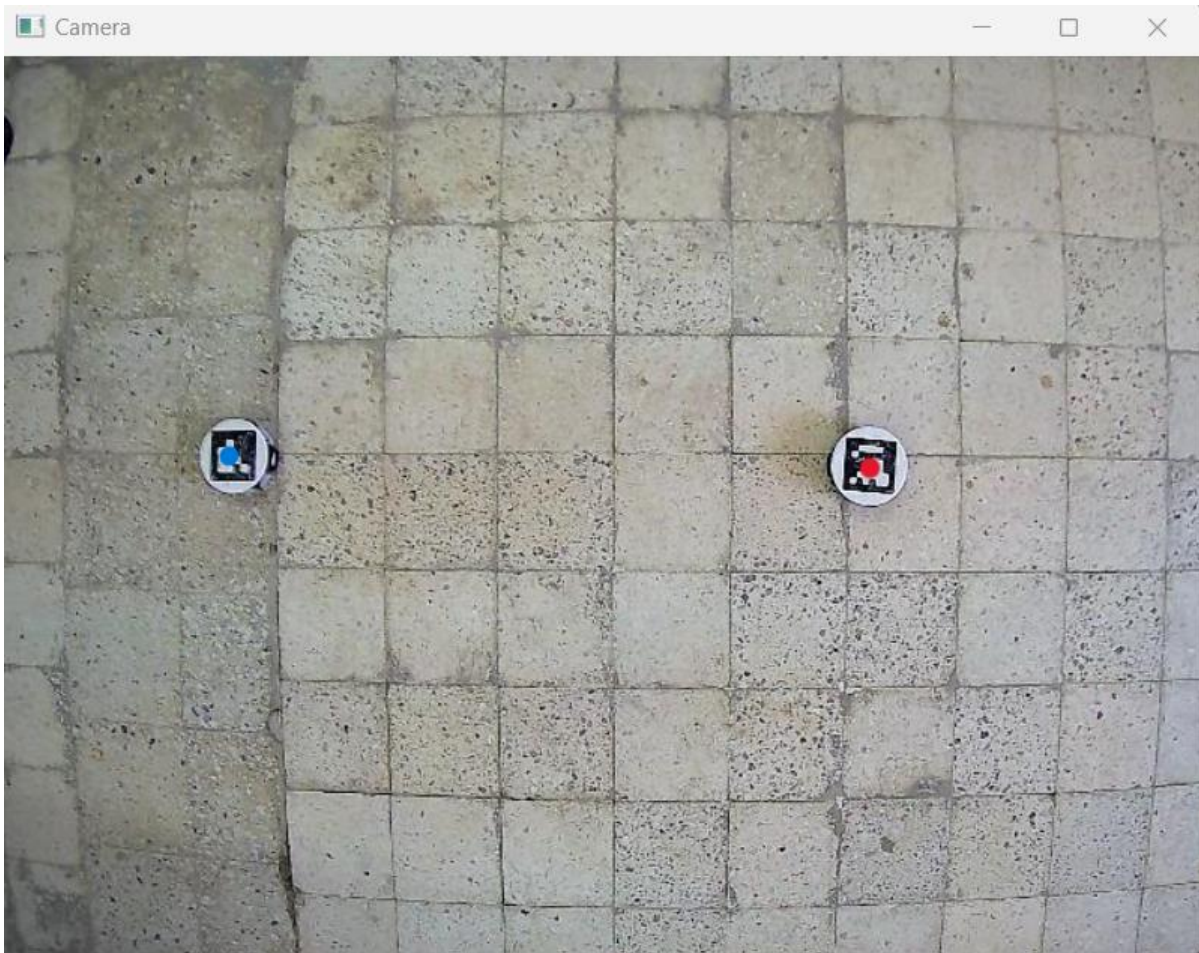
The ArUco markers are designed to be easily detectable and recognizable in images or video frames. They are typically printed on paper or displayed on a screen. The markers can be of different sizes, with larger markers having more bits in their binary pattern, allowing for higher precision in detection and tracking.

The detection of ArUco markers involves image processing techniques such as edge detection, contour analysis, and pattern recognition. Once a marker is detected, its position

and orientation can be determined relative to the camera or image frame. This information can then be used for various purposes, such as object tracking, camera calibration, pose estimation, and augmented reality overlay.

ArUco markers are widely used in robotics, augmented reality applications, camera calibration, and computer vision research. They provide a simple and effective way to track objects and determine their position and orientation in a visual environment.

Moments are used to calculate the center of gravity, and these coordinates represent the pixel position of the robot (x, y).



*Fig. B.2: Calculate the center of gravity*

After detecting the centroid (Fig. B.2), the next step is to calculate the real-world position of the robot using the relationship between the actual distance and the pixel distance, which is given by the following equation:

$$D_{real} = D_{pixel} \times \frac{H}{F} \quad (\text{B.1})$$

With:

- $F$ : The focal length of the camera.
- $H$ : The height of the camera relative to the ground.

The camera is fixed during the experiment ( $H = \text{constant}$ ), and the focal length depends on the camera, so it is constant. This allows us to rewrite the previous equation as follows:

$$D_{real} = D_{pixel} \times factor \quad (B.2)$$

The actual distance between one corner of the marker and the next corner is known ( $D_{real} = 9.1\text{cm}$ ), and the distance in pixels ( $D_{pixel}$ ) is calculable (using the Pythagorean theorem).

$$D_{pixel} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (B.3)$$

With:

- $x_1$  and  $y_1$ : Coordinates of the first corner.
- $x_2$  and  $y_2$ : Coordinates of the second corner.

Using these results, we can calculate the value of the constant *factor*, which provides us with a real estimation of the robot's displacement.

The  $\text{atan2}$  function with two arguments ( $\sin(\theta)$ ,  $\cos(\theta)$ ) is used to calculate the orientation of the robot.

Either:  $x = \cos(\theta)$  and  $y = \sin(\theta)$

The  $\text{atan2}$  function is defined as follows:

$$\text{atan2} = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{indefinite} & \text{if } x = 0 \text{ and } y = 0 \end{cases} \quad (B.4)$$



## B.2 Obstacle detection:

To detect the fixed obstacle (Fig. B.4), we apply several image processing functions based on color.

Firstly, we start by using a Gaussian filter to reduce noise (Fig. B.5).



*Fig. B.3: Image acquired from the camera*

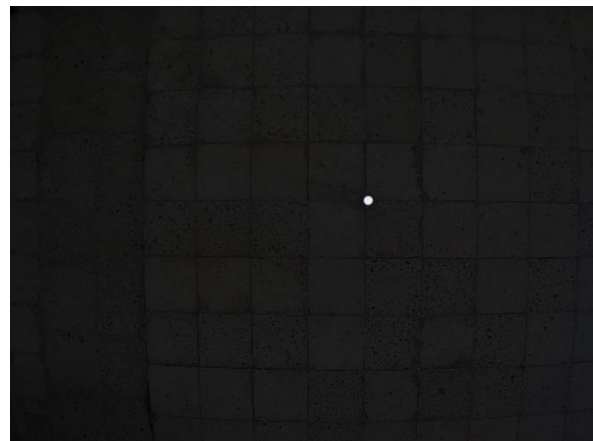


*Fig. B.4: Image after applying the  
Gaussian filter*

Next, instead of using the usual RGB color space, we will use the HSV color space, which has the desirable property that allows us to identify a specific color. The next step is to extract our specific color from the resulting image using a mask.



*Fig. B.5: The image in the HSV color space*



*Fig. B.6: Extraction of the blue color*

Once these operations are completed, we use moments to calculate the center of gravity.



*Fig. B.7: Final result*

# Appendix C

## Particle Swarm Optimization

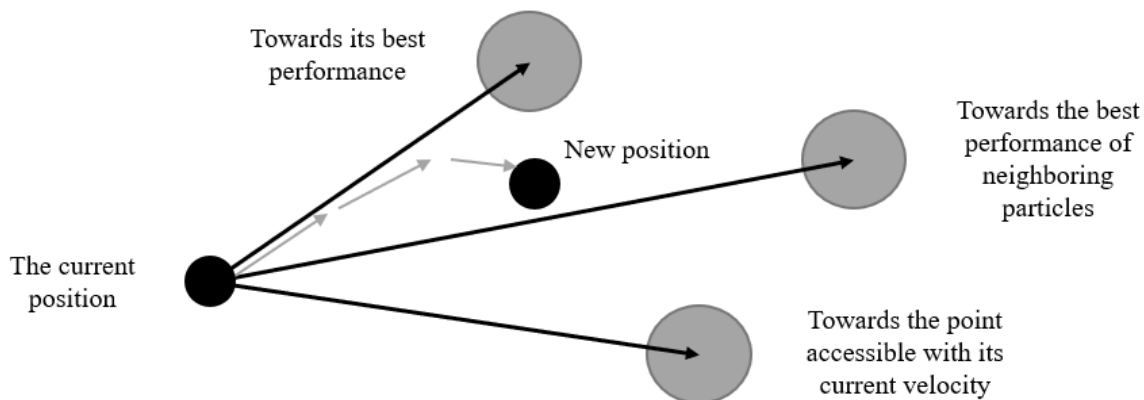
The solution to the optimization problem has a significant influence on the performance of our mobile robotics system. The task at hand is to choose powerful algorithms that can find high-quality solutions with minimal computational time. Recently, the majority of algorithms used to solve these optimization problems are population-based metaheuristics.

Within this class of algorithms, we find metaheuristics that manipulate a population of solutions. We can distinguish genetic algorithms (GA), ant colony optimization (ACO), gravitational search algorithm (GSA), and the one that particularly interests us and has been used in our work, which is Particle Swarm Optimization (PSO), was introduced in 1995.

A particle swarm corresponds to a population of simple agents (particles or birds). Each agent is considered a solution to the problem, where it has a position, a velocity, and a memory that allows it to remember its best performance and the best performance achieved by the entire swarm of particles [8].

In the search for the global optimum, the movement of a particle is influenced by:

- The inertia component refers to the current velocity of the particle.
- The cognitive component refers to the particle's memory of the best solution it has encountered so far.
- The social component refers to the best site collectively reached by the swarm, which is typically determined by the best solution achieved by the particle's neighboring particles.



*Fig. C.1: The PSO illustration*



For a mathematical formalization of the movement of each particle  $i$ , the following equations are introduced:

Velocity update:

$$v_i(t + 1) = \omega v_i(t) + c_1 r_1 (p_{best} - x_i(t)) + c_2 r_2 (g_{best} - x_i(t)) \quad (C.1)$$

where:

- $v_i(t + 1)$ : is the updated velocity of particle  $i$  at time  $t + 1$ .
- $v_i(t)$ : is the velocity of particle  $i$  at time  $t$ .
- $\omega$ : is the inertia weight controlling the influence of the previous velocity.
- $c_1$  and  $c_2$ : are the cognitive and social acceleration coefficients, respectively.
- $r_1$  and  $r_2$ : are random numbers between 0 and 1.
- $p_{best}$ : is the personal best position of particle  $i$ .
- $g_{best}$ : is the global best position.
- $x_i(t)$ : is the current position of particle  $i$ .

Position update:

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \quad (C.2)$$

Where:

- $x_i(t + 1)$ : is the updated position of particle  $i$  at time  $t + 1$ .

These equations describe how the velocity and position of each particle are updated in each iteration of the PSO algorithm, taking into account the personal best position, global best position, and random values for exploration and exploitation.

Here is the pseudo-code for the Particle Swarm Optimization (PSO) algorithm:

```
Initialize the population of particles with random positions and velocities
Initialize the personal best positions  $p_{best}$  of each particle to their current positions
Initialize the global best position  $g_{best}$  to the best position among all particles
while (stopping criterion is not met) do
  for each particle do
    Update the particle's velocity using the PSO equations
    Update the particle's position based on the new velocity
    Evaluate the particle's performance using the objective function
    Update the particle's personal best position if necessary

    if the particle's performance is better than  $g_{best}$  then
      Update the global best position to the particle's position
    end if
  end for
end while
Return the global best position as the optimal solution
```

This pseudo-code outlines the main steps of the PSO algorithm, including initialization, velocity and position updates, performance evaluation, and the update of personal and global best positions. The algorithm continues until a stopping criterion is met, such as reaching a maximum number of iterations or achieving sufficient convergence. Finally, the global best position is returned as the optimal solution to the optimization problem.

# Appendix D

## Description of the mobile robot

### D.1 Description of the system:

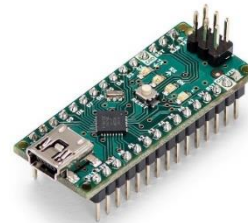
The robot is composed of several parts, which have been assembled together to give us its final form, including:

#### D1.1 Control System:

The mobile robot necessitates a central processing unit to handle data from its array of sensors and to facilitate decision-making. In this regard, our choice has fallen upon the Raspberry Pi 4 (Figure D.1), which serves as the primary computing brain. Additionally, we have established a connection to an Arduino NANO board (Figure D.2), functioning as a secondary processing unit. This Arduino board provides valuable feedback concerning wheel rotation speed and direction, further enhancing the robot's control capabilities.



*Fig D.1: Raspberry pi 4*



*Fig D.2: Arduino Nano*

#### D.1.2 Sensors:

We have enhanced the robot's capabilities by incorporating a range of sensors that enable it to gather information about its surrounding environment. These sensors consist of the following:



*Fig D.3: RPLIDAR*



*Fig D.4: Camera*

### D.1.3 Mobility System:

The mobility system refers to the mechanical components and mechanisms that enable a robot to move within its environment. It encompasses various elements that determine how the robot can navigate and traverse different terrains. Key components of a typical mobility system include:

**Wheels:** Wheels are a common choice for robotic mobility. They can come in different configurations, such as two-wheel differential drive (Fig D.5), allowing the robot to move forward, backward, turn, and change direction.

**Motor Control:** The L298N module (Fig D.6) can control two DC motors independently. It allows you to control the direction of rotation (forward or backward) and the speed of each motor.



*Fig D.5: Wheel connected with motor*



*Fig D.6: The L298N module*

### D.1.4 Power Source:

The mobile robot is battery-powered, and the choice of battery type and capacity depends on the robot's intended use and runtime requirements. In our case we chose a 11.1v LiPo battery (Fig D.7), with 6500mAh capacity:



*Fig D.7: LiPo Battery*

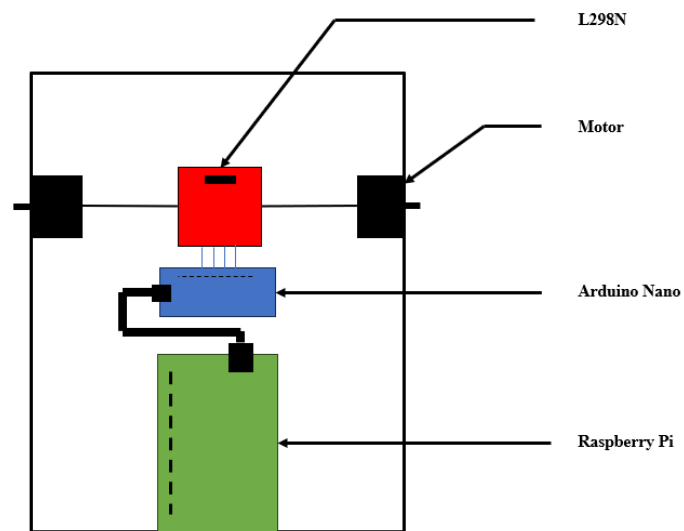
## D.2 Robot structure:

### D.2.1 Chassis:

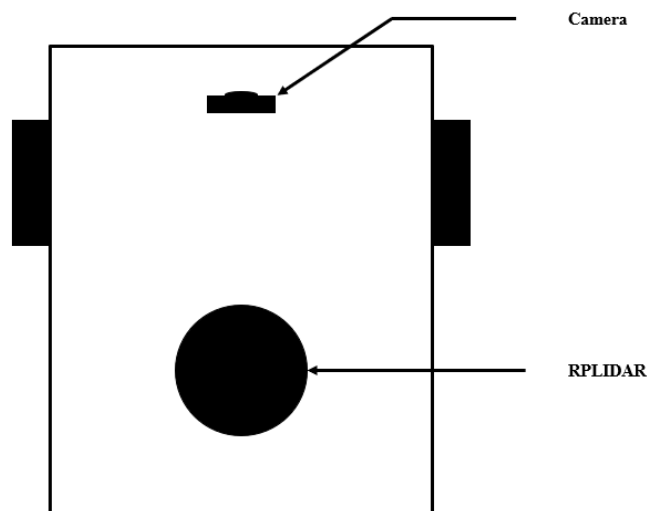
The robot has a chassis that provides it with a sturdy structure and allows it to support various components. The chassis can be made from fiberglass, metal, or plastic, depending on the design requirements.

### D.2.2 Assemble the robot parts:

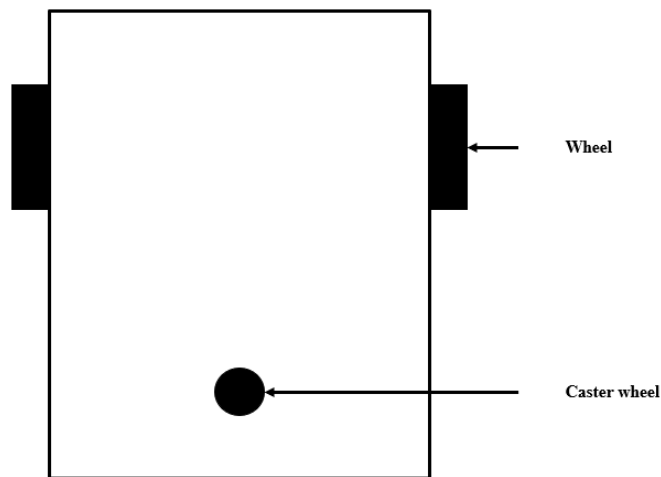
The corresponding images show how we connected the robot parts:



*Fig D.8: The internal components of the robot*



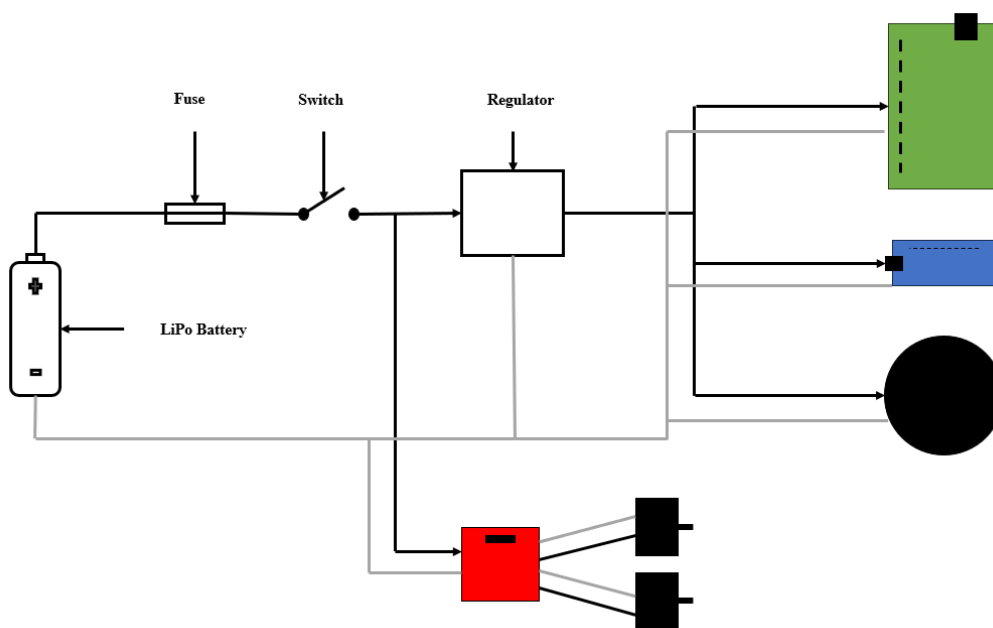
*Fig D.9: The robot's external appearance*



*Fig D.10: The bottom part of the robot*

### **D.2.2 The power circuit:**

For the power circuit, a 11.1v rechargeable Li-Po battery is employed, connected in series with a fuse to safeguard the circuit components against short circuit. A switch controls the robot's power on/off functionality, while a voltage regulator steps down the voltage from 11.1v to 5v, ensuring a consistent power supply to all the components from a single power source.



*Fig D.11: The power circuits*