People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Mohamed Khider – BISKRA
Faculty of Exact Sciences, Science of Nature and Life
Computer Science Department

Order number :.....

THESIS

Submitted in fulfilment of the requirements for the Masters degree
in Computer Science

Option : GLSD

Title

Design and Realization of a Smart Delivery Robot

Presented by
KAHOUL Mahdi

Defended in : 26/06/2024

Defended before a jury composed of :

Bettira Roufaida	**MCA**	**President**
Tigane Samir	**MCA**	**Supervisor**
Kahloul Laid	**Professor**	**Co-Supervisor**
Torki Fatima Zohra	**MAA**	**Examiner**

Session 2023-2024

Acknowledgements

First and foremost, praise and thanks be to **ALLAH**, the Most Gracious, the Most Merciful, and His strength that overwhelmed me throughout my research work by successfully completing it.

To my parents, a precious gift from ALLAH, **Kamal** And **Linda** who sacrificed for me and spared no effort in ensuring my happiness, my reason for living, and my treasure.

I would like to express my special thanks and gratitude to my professor **Laid Kahloul** who believed in me and gave me the golden opportunity to do this project, as well as **Dr. Samir Tigan** and **Miss. Leyla Belaiche** who helped me do this project and learn many new things. I am really grateful to them. I extend my thanks to all the teachers who helped me in my academic life.

To my Little brothers, **Chabi Amine sif eddine** and **Kermiche Abderrahmane**, for their sincere friendship, trust and love.

To my dear colleague, **Koolaf Roumaissa**, your friendship has been a beacon of light throughout this journey.

Last but not least, I also thank all the **Debug Club** members one by one, as having such amazing people around me has had a huge impact on my life and pushed me forward to always give my best.

To everyone I love and care about him, thank you

KAHOUL MAHDI

Abstract

A mobile robot is an autonomous or semi-autonomous machine capable of moving and navigating through an environment. It can perform various tasks, such as transportation, inspection, and exploration. This work describes the specification and realization of a mobile robot designed for delivery services. With the growing need for effective logistics solutions in regions like Algeria, where technological advancements are crucial for development across various sectors, the use of mobile robots shows great potential in improving delivery operations. Leveraging advancements in robotics, this work outlines a comprehensive approach to developing mobile robots capable of navigating diverse environments to facilitate delivery tasks. The proposed solution also addresses the unique challenges facing delivery services, such as seamless communication and coordination between multiple robots. In this work, a prototype of a mobile robot for delivery service is provided for testing the proposed cooperative multi-robot system (CMRS) at the University of Biskra in Algeria as a case study. Afterward, a formal model for the proposed system is prepared based on the Büchi automaton. Thanks to the UPPAAL model-checker, the behavior of the proposed CMRS is verified in terms of two criteria: the system's safety and the system's liveness which are formulated as a set of computational tree logic (CTL) properties. The UPPAAL model-checker tool results show that the proposed CMRS represents a safe and robust delivery mobile system.

Keywords: *Delivery operations, Mobile robotics, Cooperative multi-robot system, Buchi automata, CTL logic, Model-checking, UPPAAL tool.*

Contents

List of Figures

List of Tables

General Introduction

General Introduction

In recent years, the robotics field has seen incredible developments that have revolutionized several industries and redefined the possibilities of task automation. Mobile robots are advantageous because they can be used in a wide range of environments and are highly versatile [36]. Mobile robots have proven their effectiveness in streamlining workflows and increasing productivity in a variety of settings, including industry [20], hospitals, warehouses, delivery services [24, 25], and exploration and mission missions [29]. In the context of delivery services, mobile robots offer a promising solution to address the growing demands of logistics. With the emergence of e-commerce and the increasing need for last-mile delivery solutions, mobile robots present an efficient service to optimize the delivery process, reduce costs, and improve customer satisfaction [11]. Examples of such applications include Amazon's delivery drones [1] and Star-Ship Technologies' autonomous delivery robots [9]. These examples illustrate the potential of mobile robots in transforming the delivery landscape. Therefore, developed countries have exploited the integration of mobile robot technologies into their logistics infrastructure, however, underdeveloped regions such as Algeria still face challenges in adopting such technologies. With limited resources and infrastructure, Algeria is poised to take advantage of deploying mobile robots and robotic solutions in a variety of sectors, including delivery, industrial, and hospital. Several research studies have substantially contributed to the advancement of mobile robotics, particularly in navigation, localization, and multi-robot coordination. These studies have focused on refining localization techniques via integrating sensor fusion and probabilistic methods [31]. This has resulted in an enhanced accuracy of position estimation in dynamic environments. Navigation algorithms have undergone extensive development to address challenges posed by complex terrains and dynamic obstacles, enabling robots to traverse

indoor [32, 38, 33] and outdoor [30, 37] spaces with increased agility and precision. Additionally, research efforts have investigated communication protocols, task allocation strategies, and collaborative decision-making mechanisms for multi-robot systems [23], paving the way for developing cooperative robots capable of tackling diverse challenges. Numerous researches have also focused on using formal methods for specification and verification of multi-robots-system to ensure simulation and evaluate the behavior of some complex multi-robot systems [28] by verifying some properties such as (system's safety, system's liveness) [35]. Collectively, these research endeavors highlight the interdisciplinary nature of mobile robotics and underscore the ongoing quest for innovative solutions to real-world problems in autonomous systems. This work addresses the requisite for delivery robotics solutions in Algeria by proposing a mobile delivery robot.

Organisation of the dissertation

The dissertation is organized as follows:

Chapter 1: Technical Background.

This chapter showcases the technical background of robotics field, encompassing some definitions and showing the peak of current technological and scientific advancements in the field.

Chapter 2: Design and Realization.

The design and realization chapter in this thesis shows the complete process of conceptualizing, developing, and implementing mobile robots. It includes the initial design phase, where specifications and functionalities are defined, followed by the engineering and construction of both hardware and software components.

Chapter 3: Modeling and formal verification.

The chapter on modeling and formal verification demonstrates the systematic approach to identifying and correcting potential errors and defects early in the development process, resulting in more safe and robust systems.

Conclusion and Perspectives.

This concludes the main goal of this thesis and it gives a point of view for future work.

Chapter 1

Technical Background

1.1 Introduction

The field of robotics has advanced significantly from its initial days of bulky machines. Starting with early industrial robots handling repetitive tasks, today's machines are highly advanced, capable of learning and adjusting. This progress is fueled by various factors, such as enhancements in artificial intelligence, sensor technology, and material science. With increasing its intelligence and versatility, robots are being utilized across diverse fields like manufacturing, healthcare, exploration, and customer service. The future of robotics holds the promise of further breakthroughs, as robots become more seamlessly integrated into our everyday routines. In this chapter, we will present the definition of robots along with their different types, uses, and advantages.

1.2 Robotics

Robotics is a multidisciplinary field that includes various aspects of engineering, computer science, mathematics, and physics to design, build, operate, and use robots, and it depends on many standards. We can divide robots into many types, and famous standards include the nature of robot movement, such as (fixed robots, mobile robots) [36].

1.2.1 The three laws of robotics

The Three Laws of Robotics, formulated by science fiction writer **Isaac Asimov** [27] which were designed to create a framework for ethical and safe robot behavior, ensuring that robots serve and protect humans effectively without causing harm.

First Law: A robot may not injure a human being or, through inaction, allow a human being to come to harm.

This law prioritizes human safety above all else. A robot must actively avoid causing harm to humans, whether through direct actions or by failing to act when a human is in danger. For example, a robot must intervene if it sees a human in a potentially dangerous situation, such as stepping in front of a moving vehicle.

Second Law: A robot must obey the orders given it by human beings, except where such orders would conflict with the First Law.

This law ensures that robots follow human commands, making them useful tools for humans. However, if a human order would result in harm to another human, the robot must not follow it. For instance, if a person commands a robot to harm someone else, the robot must refuse because obeying would violate the First Law.

Third Law: A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

This law allows robots to take actions to preserve themselves, which ensures their continued operation and usefulness. However, the robot's self-preservation is secondary to the safety of humans and obedience to human orders. For example, a robot should avoid stepping into danger, but if rescuing a human requires it to risk its own existence, it must do so.

1.2.2 Application areas of robotics

Robots are used more in different sectors for their efficiency [16]. They help with tasks like assembly and surgeries, as well as in logistics and hazardous environments. Their ability to work

in various conditions makes them vital in many industries Figure 1.1.

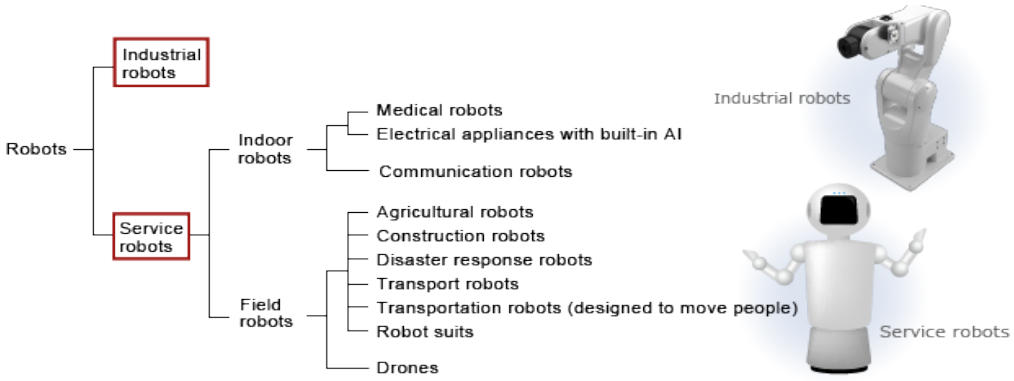


Figure 1.1: Main areas of robotics application [8].

Industrial robots:

Industrial robots (like the on in Figure 1.2) are automated machines intended to carry out manufacturing tasks with precision, efficiency, and consistency. They are widely utilized in industries like automotive, electronics, and metalworking for tasks such as welding, painting, assembly, material handling, and packaging. With advanced sensors and artificial intelligence, industrial robots can adjust to various tasks and environments, minimizing the necessity for human intervention in dangerous or mundane jobs. Their utilization notably boosts production speed, product quality, and workplace safety, rendering them essential to contemporary manufacturing procedures.

Figure 1.2: Articulated Robots: Robotic arms used in automotive manufacturing [21].

Domestic or household robots:

Domestic robots (see Figure 1.3), also known as household robots, are autonomous or semi-autonomous machines created to help with daily tasks in a home setting. They come with sensors, artificial intelligence, and different mechanisms for tasks like cleaning, mopping, lawn mowing, and even offering companionship or entertainment. The main objective is to improve convenience, efficiency, and overall quality of life by automating regular chores and duties.

Figure 1.3: Vacuum Cleaning Robots: Roomba by iRobot [5].

Robots in medicine and surgery:

Robots in healthcare and surgery (see Figure 1.4) are advanced automated systems that assist medical professionals in performing tasks. Equipped with sensors, actuators, and AI, these robots carry out precise maneuvers, improving surgical outcomes, minimizing invasiveness, and enhancing patient care. Essential in contemporary medicine, they facilitate precise procedures, creating opportunities in diagnosis, treatment, and rehabilitation. Robodoc supports surgical procedures, nursing robots are currently being developed, the HAL (Hybrid Assistive Limb) cyberskeleton aids in mobility, and patient robots assist dental students in efficiently practicing treatment procedures.

Figure 1.4: Surgical Robots: da Vinci Surgical System [14].

1.2.3 Types of robots

There are two categories broadly encompass many types of robots based on their mobility and flexibility in movement. These two robot families showcase distinct automation approaches, each tailored for specific use cases and settings.

Fixed robots:

Also known as stationary or industrial robots, are anchored to a specific location and operate within a defined workspace. They typically have a fixed base and operate along pre-programmed paths or within specific constraints. Fixed robots are commonly used in manufacturing environments for tasks such as welding, painting, assembly line work, and material handling. They excel in repetitive tasks that require precision and consistency.

Mobile robots:

As the name suggests, mobile robots have the capability to move within their environment. They are equipped with locomotion systems that enable them to navigate through various terrains and spaces. In this work, we are focusing on this second category of robots.

1.3 Autonomous Mobile Robots

Mobile robots, especially (wheeled and air-based, Underwater-based), are a specialized class of robotic systems designed to move autonomously or semi-autonomously in their environment for service purposes.

Mobile systems can be defined as systems that are not attached to the environment and can move in a certain space. In terms of the environment they move across they can be classified into some principal groups:

1.3.1 Types of mobile robots

Mobile robots come in various types, each designed for specific applications and environments. following most famous types of mobile robots:

Ground mobile systems:

Various types of mobile platforms can be found here such as mobile vehicles with wheels or caterpillars, legged robots (humanoids or animal mimicking), or robots that mimic some other type of animal locomotion, for example, snakes. Ground mobile systems with wheels or caterpillars that do not carry the operator are often referred to as unmanned ground vehicles.

Figure 1.5: Example of ground mobile robot [4].

Aerial mobile systems:

This group consists of mobile systems that fly in a certain aerial space (airplanes, helicopters, drones, rockets, animal-mimicking flying systems; when used without a pilot they are referred to as unmanned aerial vehicles) or orbit the Earth or some other celestial body (satellites).

Figure 1.6: Example of Aerial mobile systems [3].

Water and underwater mobile systems:

In this group we find different types of ships,boats, submarines, autonomous underwater vehicles, etc.

Figure 1.7: Example of underwater mobile robot [6].

1.3.2 Core principles of mobile robots

The fundamental principles of mobile robots comprise basic concepts and methodologies that control their design, operation, and interaction with the environment. These principles serve as the basis of mobile robotics and steer the advancement of autonomous systems able to navigate, interact, and carry out tasks in real-world settings. Understanding the core principles of robotics is crucial for designing a functional and intelligent delivery robot.

Mechanics:

Differential Drive System with Omnidirectional Wheels:

- **Differential Drive System:** This system utilizes two independently driven wheels on either side of the robot. By varying the speed and direction of each wheel, the robot can move forward, backward, and pivot in place. This type of drive system is simple and effective for many mobile robot applications.
- **Omnidirectional Wheels:** These wheels, often designed with smaller rollers around their circumference, enable the robot to move in any direction without needing to turn. This enhances maneuverability in tight spaces and around obstacles.
- **Chassis Construction:** The robot's chassis is made from lightweight yet high-strength aluminum. This material choice balances the need for a robust structure capable of carrying a payload while minimizing the overall weight to improve efficiency and stability.

Sensors:

The robot's comprehensive sensor suite includes LIDAR(Light Detection And Ranging)for precise distance measurements through laser emission, cameras for visual tasks like object recognition and navigation, ultrasonic sensors for close-range obstacle detection via sound waves, infrared sensors for proximity detection and edge detection, and GPS for outdoor navigation with location data. Sensor fusion integrates data from these sources using algorithms, enhancing the robot's reliability and accuracy by providing a comprehensive understanding of its surroundings through the combination of multiple sensor inputs.

Figure 1.8: Type of sensors [18]

Actuators:

High-torque electric motors are crucial components for robots, as they efficiently convert electrical energy into the necessary mechanical force for movement, particularly when handling heavy payloads. Motor controllers play a pivotal role in governing motor operation, ensuring seamless control over speed and direction. With advanced controllers capable of managing intricate maneuvers and executing sharp turns smoothly, the robot's agility and stability are significantly enhanced, enabling it to navigate various environments with precision and confidence.

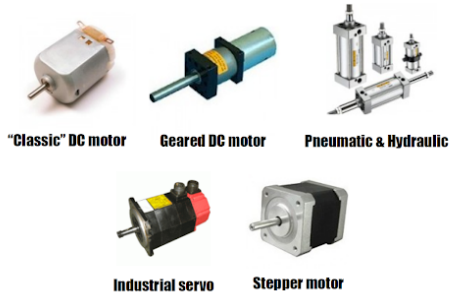


Figure 1.9: Types of Actuators [2]

Control Systems:

The intelligent control unit, powered by a central processing unit (CPU), serves as the robot's brain, processing sensor data and making real-time decisions. Machine learning algorithms enable the robot to learn from its environment and experiences, enhancing its capabilities. For instance, in path planning, the robot determines the most efficient route to its destination, skillfully avoiding obstacles and navigating complex environments. Through obstacle avoidance learning, the robot becomes increasingly adept at circumventing obstacles encountered in its

path. Additionally, the robot's ability to adapt its actions based on changing conditions ensures reliable task execution. Real-time processing is crucial, as the control system must swiftly analyze data and respond to ensure effective operation in dynamic environments.

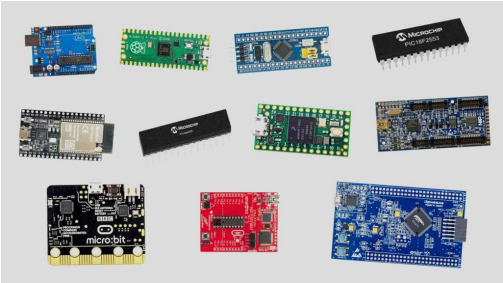


Figure 1.10: Most famous microcontrollers [7].

Core components of the robot operating system (ROS):

At the core of numerous robotic systems is the Robot Operating System (ROS), a flexible framework that offers libraries, tools, and standards for creating intricate robotic applications. ROS aids in the creation of capabilities such as perception, control, and communication, which are crucial for the smooth functioning of mobile robots [34]. ROS (Figure 1.11) offers a collection of key components that establish the basis of robotic applications:

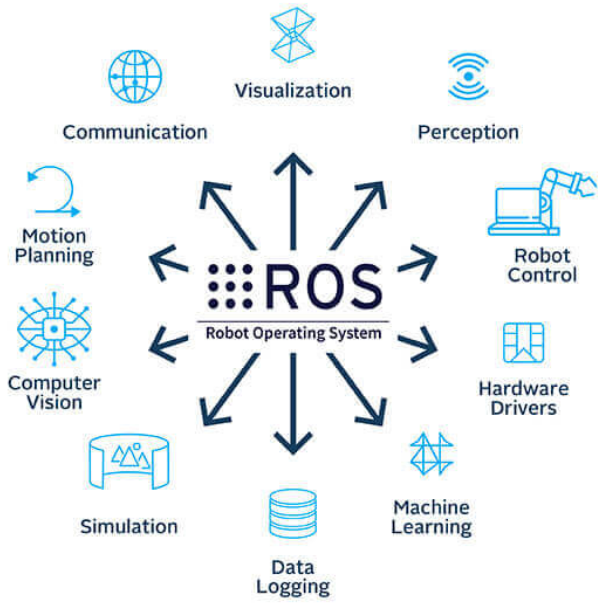


Figure 1.11: Robot Operating System (ROS) [10]

- **ROS Master**: Facilitates the naming and registration services for ROS nodes, tracking publishers and subscribers to topics, as well as services. It acts as a central coordinator that nodes use to find one another and exchange messages. Without the Master, nodes would not be able to locate each other, making communication impossible.

- **ROS Nodes**: are individual processes that perform specific computations, each designed to handle a focused task such as controlling a motor or processing sensor data. These nodes can be distributed across multiple machines, allowing for a modular and scalable system architecture. Communication between nodes is facilitated through topics, services, and actions, enabling seamless interaction and coordination within the robotic system.

- **ROS Topics**: provide a means for nodes to communicate with each other in a many-to-many manner through a publish/subscribe model. Nodes can publish data to a topic as a publisher, and other nodes can receive this data by subscribing to the topic. This decouples the production and consumption of information, enabling flexible and dynamic communication within the robotic system.

- **ROS Services**: facilitate synchronous communication between nodes via a request/response mechanism. When a node needs to send a request and wait for a response, it uses a service, making it suitable for tasks that require an immediate answer, such as querying the state of a sensor. This allows for direct and timely interactions between nodes, ensuring prompt responses to specific queries.

- **ROS Messages**: are the data structures used for communication between nodes via topics and services. They define the data type and structure of the information exchanged, ensuring consistency and interoperability. Common message types include standard data types, such as integers, floats, and strings, as well as more complex types like sensor readings and actuator commands. This standardized format allows for reliable and seamless data exchange within the robotic system.

- **ROS Bags**: it's serve the function of providing a file format for recording and playing back ROS message data. This functionality is useful for logging data during robot operation,

which can then be analyzed offline. It enables the repeatability of experiments and facilitates the debugging of software using real-time data without the need for the robot to be operational. By recording and replaying ROS message data, bags support comprehensive analysis and testing of robotic systems, enhancing their development and deployment processes.

- **RViz**: it serves as a powerful visualization tool within the Robot Operating System (ROS), enabling users to observe the state of the robot and the data it processes. Its primary function is to display sensor information, robot model states, and various other data in 3D space. This visualization capability is invaluable for debugging and verifying the robot's perception and control systems, as it provides a clear representation of the robot's environment and its interactions within it. By offering real-time visualization of critical data, RViz enhances the development and testing processes of robotic systems, facilitating more efficient and accurate performance evaluation.

- **Gazebo**: serves as a high-fidelity robotics simulator that seamlessly integrates with ROS for testing and development purposes. Its primary function is to provide a physics-based simulation environment where robots can be thoroughly tested in a virtual setting before deploying them in the real world. Gazebo plays a crucial role in the development process by supporting the simulation of sensors and actuators, enabling realistic testing and development of algorithms. By offering a simulated environment that closely mimics real-world conditions, Gazebo enhances the efficiency and effectiveness of the development process, ultimately leading to more robust and reliable robotic systems.

1.3.3 Advantages of mobile robots

Several advantages of mobile robots can be mentioned.

- Robotics and automation can in many situations increase productivity, safety, efficiency, quality and consistency of products.

- Robots can work in a dangerous environment, without the need for life support, or safety concerns

- Robots do not require lighting, air conditioning, ventilation, or noise protection.
- Robots work continuously without feeling tired or bored, and they do not require medical or vacation insurance.
- Robots maintain repeatable precision at all times unless something happens to them or they wear out.
- Robots can be much more precise than humans.

The downside of robots is their inability to react in emergencies unless the situations are understood and the responses are programmed into the system. Safety measures are necessary to ensure they do not harm operators or damage the machines they work with.

1.3.4 Challenges of mobile robots

Mobile robots face numerous challenges that span various technical and operational aspects. These challenges can significantly impact their performance, reliability, and applicability across different environments and tasks.some of the key challenges mobile robots encounter:

- **Limited Battery Life**: The robot utilizes a high-capacity lithium-ion battery with efficient power management systems. The robot can automatically return to designated charging stations when battery levels drop below a certain threshold. Additionally, the control system optimizes power consumption during navigation by employing short and efficient paths.
- **Security Risks**: The robot is equipped with robust security protocols to prevent unauthorized access or tampering. Secure communication channels are used for data transmission, and the robot can be programmed to identify and report suspicious activity. Delivery compartments are also secured with electronic locks that only open upon reaching the designated location and receiving a user-authenticated confirmation.
- **Public Perception and Acceptance**: The robot is designed to operate safely and considerately alongside humans. It adheres to strict speed limitations and prioritizes safe naviga-

tion over speed. Additionally, the robot is equipped with audible and visual indicators to signal its presence and intentions to pedestrians.

1.4 Multi-Robots System

A multi-robot system (MRS) refers to a group of robots that are designed to work together to accomplish tasks that would be difficult or impossible for a single robot to complete alone. These systems leverage the collective capabilities, coordination, and cooperation of multiple robots to achieve complex goals more efficiently and effectively. In the following paragraphs, we will discuss some aspects of multi-robot system including their characteristics, their applications, the used technologies, and their advantages.

1.4.1 Characteristics of multi-robots systems

Multi-robot systems (MRS) possess distinct characteristics that enable their effectiveness and versatility across various applications. These characteristics define how multiple robots coordinate, communicate, and function together to achieve common goals. following, the key characteristics of multi-robot systems:

- **Collaboration**: Robots in a multi-robot system collaborate to divide tasks among themselves based on their individual capabilities and the overall mission requirements. This collaborative approach enhances the system's efficiency and productivity.
- **Communication**: Effective communication is essential in multi-robot systems. Robots exchange information in real-time about their status, environment, and tasks, enabling coordinated actions and decision-making.
- **Coordination**: involves planning and executing tasks in a synchronized manner. Robots coordinate their movements, avoid collisions, and ensure that their activities complement each other to achieve a common goal.
- **Decentralization**: Many multi-robot systems operate in a decentralized manner, where

robots make decisions locally based on their interactions with other robots and the environment. This decentralization enhances scalability and robustness.

- **Tasks allocation**: are distributed among robots based on their specific capabilities and current workload. Efficient task allocation algorithms ensure that tasks are completed optimally without unnecessary duplication of effort.

1.4.2 Applications of multi-robots systems

Multi-robot systems (MRS) have found applications across numerous fields, revolutionizing the way tasks are performed through enhanced efficiency, precision, and safety. We can found MRS in:

- **Search and Rescue**: In disaster response scenarios, multiple robots can cover large areas quickly to locate victims, assess damage, and deliver supplies. Their coordination allows for systematic and thorough exploration.
- **Agriculture**: Robots can perform agricultural tasks such as planting, monitoring crop health, and harvesting. By working together, they can cover extensive fields more efficiently and with greater precision.
- **Environmental Monitoring**: Swarms of robots can monitor environmental conditions like air and water quality, track wildlife, and map ecosystems. Their coordinated efforts provide comprehensive data over large areas.
- **Industrial Automation**: In manufacturing and warehousing, robots can manage inventory, sort packages, and perform assembly tasks. Their collaboration ensures smooth operations and enhances productivity.
- **Exploration**: Multi-robot systems are used in space and underwater exploration. Robots can explore harsh and unknown environments, sharing data and ensuring mission success through coordinated efforts.

1.4.3 Technologies of multi-robots systems

Multi-robot systems (MRS) rely on a range of advanced technologies to enable effective coordination, communication, and functionality. These technologies facilitate the development and operation of MRS, allowing them to perform complex tasks efficiently and autonomously. Some of the key technologies that support multi-robot systems are :

- **Communication Protocols** : Robust communication protocols (e.g., Wi-Fi, Bluetooth, mesh networks) ensure reliable data exchange between robots.
- **Artificial Intelligence and Machine Learning** : AI and ML algorithms enable robots to make decisions, learn from their environment, and improve their performance over time.
- **Distributed Computing** : Distributed computing frameworks allow for the sharing and processing of data across multiple robots, enhancing their collective cooperation.

1.4.4 Advantages of multi-robots systems

- **Increased Efficiency**: By distributing tasks among multiple robots, MRS can complete tasks faster and more efficiently than a single robot could.
- **Scalability**: Multi-robot systems can be easily scaled by adding more robots to handle larger or more complex tasks without a significant overhaul of the system.
- **Robustness and Reliability**: The failure of one robot in a multi-robot system does not necessarily halt the mission. Other robots can take over the tasks, providing redundancy and increasing the system's robustness.
- **Flexibility**: MRS can adapt to changing conditions and dynamic environments. They can reassign tasks and adjust their strategies in real-time to handle new challenges.

1.5 Conclusion

In this chapter, we presented a general definition of the robot and chose the type of mobile robot, and we chose this type with wheels. We have detailed the different configuration of these

wheels, the location and area of use, the pros and cons of the robots, and also came to the simple conclusion that mobile robots play a special role. Manipulative industrial robots operate autonomously in a large number of automated factories. This situation is not due to a lack of possible applications, but once we have the ability to move, we can imagine robots as postmen, cleaners, mine clearers, explorers, and many others. The world in which a mobile robot must move is often very large, partly or completely unknown, difficult to describe geometrically and has its own dynamics. In the next chapter, we will have the steps we followed to implement and realize the mobile robot In the next chapter, we will have the steps that we followed to implementation and realization of the mobile robot

Chapter 2

Design and Realization of a Delivery Mobile Robot

2.1 Introduction

In this chapter, we will discuss the steps that we followed in order to design and implement a delivery mobile robot. Starting from the 3D-modeling of the robot, to choosing the components needed to implement the robot, and ending with the programming languages that we used during the programming phase.

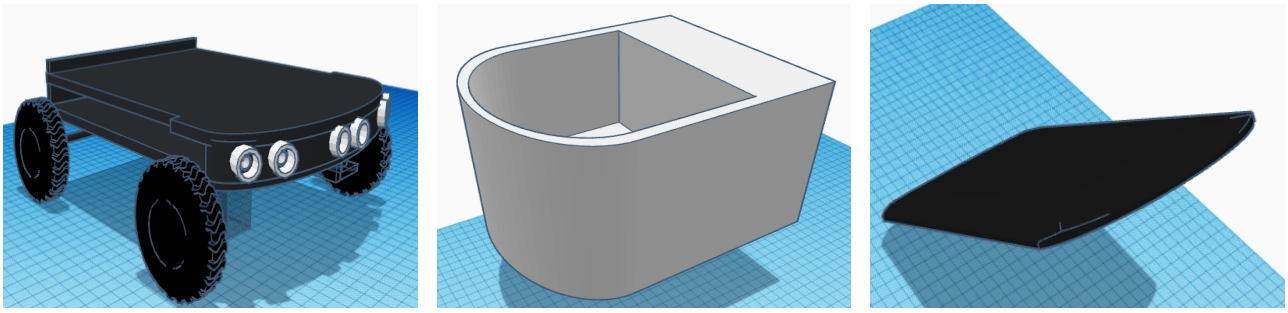
2.2 The 3D-modeling

Before starting the process of embodying the robot chassis, we designed a 3D model of the robot, which consists of three parts.

2.2.1 Robot parts

The robot chassis consists of 3 main parts (base Figure 2.1a, box Figure 2.1b, cover Figure 2.1c). Each part is designed separately, starting with the robot's base, which includes the wheels or other locomotion mechanisms, fundamental to its mobility and stability. It provides the platform for movement, housing the motors, batteries, and drive systems that enable the robot to navigate in various terrains. The base must be designed to ensure stability, preventing the robot from tipping over, especially when it is carrying heavy or unevenly distributed loads. Secondly,

the box which will hold many kinds of products, it should be designed in an efficient and practical way, and the last part is the cover which will serve as a door to protect the products.

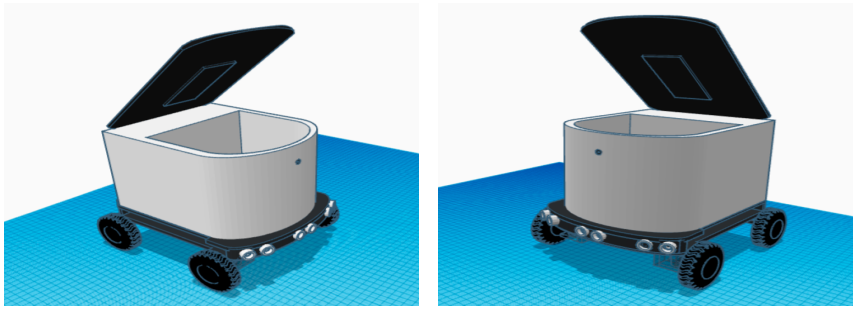


(a) Base 3D Design (b) Box 3D Design (c) Cover 3D Design

Figure 2.1: The 3D design of Robot Base and the wheel

2.2.2 Robot design

After completing the design of the basic parts of the robot, we assembled them together to form the final shape of the robot design as shown in Figure 2.2.



(a) Right side (b) Left side

Figure 2.2: The 3D design of The Robot in several views

2.3 Materials and Tools

In this section we will show both the materials and development tools we used to build the mobile robot.

2.3.1 Hardware

The following hardware components are used in building the robot, each of which has a specific task and is included in the robot.

Raspberry Pi 4:

The main processing unit of the robot is the Raspberry Pi 4 4G RAM micro-controller board, which is responsible for communicating with the coordinator and receiving the specified path to the target through mobile application, which in turn sends the path data to the Arduino unit through UART Serial communication protocol.

Figure 2.3: Raspberry Pi 4

Arduino Mega 2560 micro-controller board:

It processes all the input GPS and compass data and generates pulse width modulation (PWM) signals to control the motors. It is a micro-controller board based on the ATmega2560 micro-controller having 54 digital I/O pins (14 pins can provide PWM output), 16 analog inputs, 4 UARTs (universal asynchronous receiver/transmitter), a 16 MHz crystal oscillator, a USB port, a power socket, and a reset button. The operating voltage of the board is 5V.

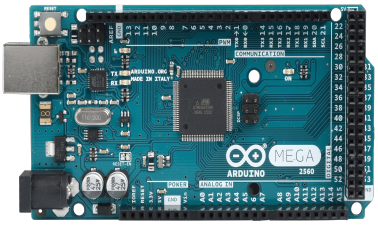


Figure 2.4: Arduino Mega 250

NodeMCU:

It processes all the input GPS and compass data and generates pulse width modulation (PWM) signals to control the motors. It is a micro-controller board based on the ATmega2560 micro-controller having 54 digital I/O pins (14 pins can provide PWM output), 16 analog inputs, 4 UARTs (universal asynchronous receiver/transmitter), a 16 MHz crystal oscillator, a USB port, a power socket, and a reset button. The operating voltage of the board is 5V.

Figure 2.5: NodeMCU 8622

Power Supply:

We used a 14.8V 3300 mAh (milliampere per hour) lithium polymer battery to power the whole robot. The battery provides sufficient power to run the robot for 45-60 minutes.

Figure 2.6: Lipo Battery

Ublox Neo-M8N GPS module:

It includes an HMC5883L digital compass. The HMC5883L digital compass is sensitive to magnetic interference and any other interference that can cause inaccurate results. The latest Ublox series compact GPS compass module provides a convenient way to mount the compass away from sources of interference that may be present within the system's confines. The Neo-M8N GPS receiver provides high location tracking accuracy under good reception conditions. Thus,

with this sensor component, we can get accurate data on latitude, longitude, altitude, time, and direction angle from a single device.

Figure 2.7: GPS module

Motor driver:

An L298N motor driver board runs the 4 DC geared motors used in the wheels of our robot. The motor driver is powered by a 14.8V 3300 mAh lithium polymer battery. The motor driver provides a 2 Amp peak output current per channel sufficient to carry the load of the 4 geared motors.

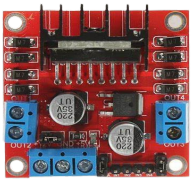


Figure 2.8: L298N Motor Driver

Ultrasonic Sensor:

Ultrasonic Sensor is a type of electronic sensor that uses ultrasonic waves to determine the distance between two objects and converts the reflected sound into electrical signals

Figure 2.9: Ultrasonic Sensor

2.3.2 Software

C programming language:

The C programming language is a high-level, general-purpose programming language that was developed in the early 1970s by Dennis Ritchie at Bell Labs. It has had a significant influence on many later programming languages, it's widely used in various fields of software development.

Figure 2.10: C programming language Logo

Python programming language:

Python is a powerful, general-purpose programming language with a wide range of applications. It was mainly used for scripting after its release in 1994, but updates and new technologies in recent years have expanded its utility. Today, Python stands tall as many developers' favorite programming language.

Figure 2.11: Python Logo

Arduino IDE:

The Arduino IDE is an open-source project that makes it easy to write code and upload it to several micro-controller boards.

Figure 2.12: Arduino IDE Code

Visual Studio Code:

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including C, C#, C++,Fortran, Go, Java, JavaScript, Node.js, Python, Rust.

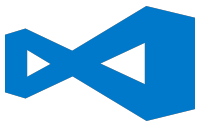


Figure 2.13: Visual Studio Code Logo

Ubunto Server:

Ubuntu server is a specific addition that differs a little bit from Ubuntu desktop, in order to facilitate installation on servers and micro-controllesr as raspberry.

Figure 2.14: Ubunto Logo

OpenCV:

OpenCV (Open Source Computer Vision Library) is an open-source software library designed for computer vision and image processing tasks. It provides a comprehensive set of tools and functions for analyzing and manipulating images and videos, enabling developers to create applications involving real-time computer vision

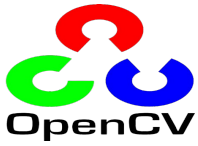


Figure 2.15: OpenCV Logo

2.4 System Design

After selecting all the required components for prototyping the proposed mobile robot, we propose an architecture of the system composed of four sub-systems: *Control, Motion, Localiza-*

tion, Communication, and *Perception* systems. Each of these has a specific role in the overall robot behavior. Figure 2.16 illustrates the system architecture of the proposed mobile robot with its sub-systems equipped with hardware components.

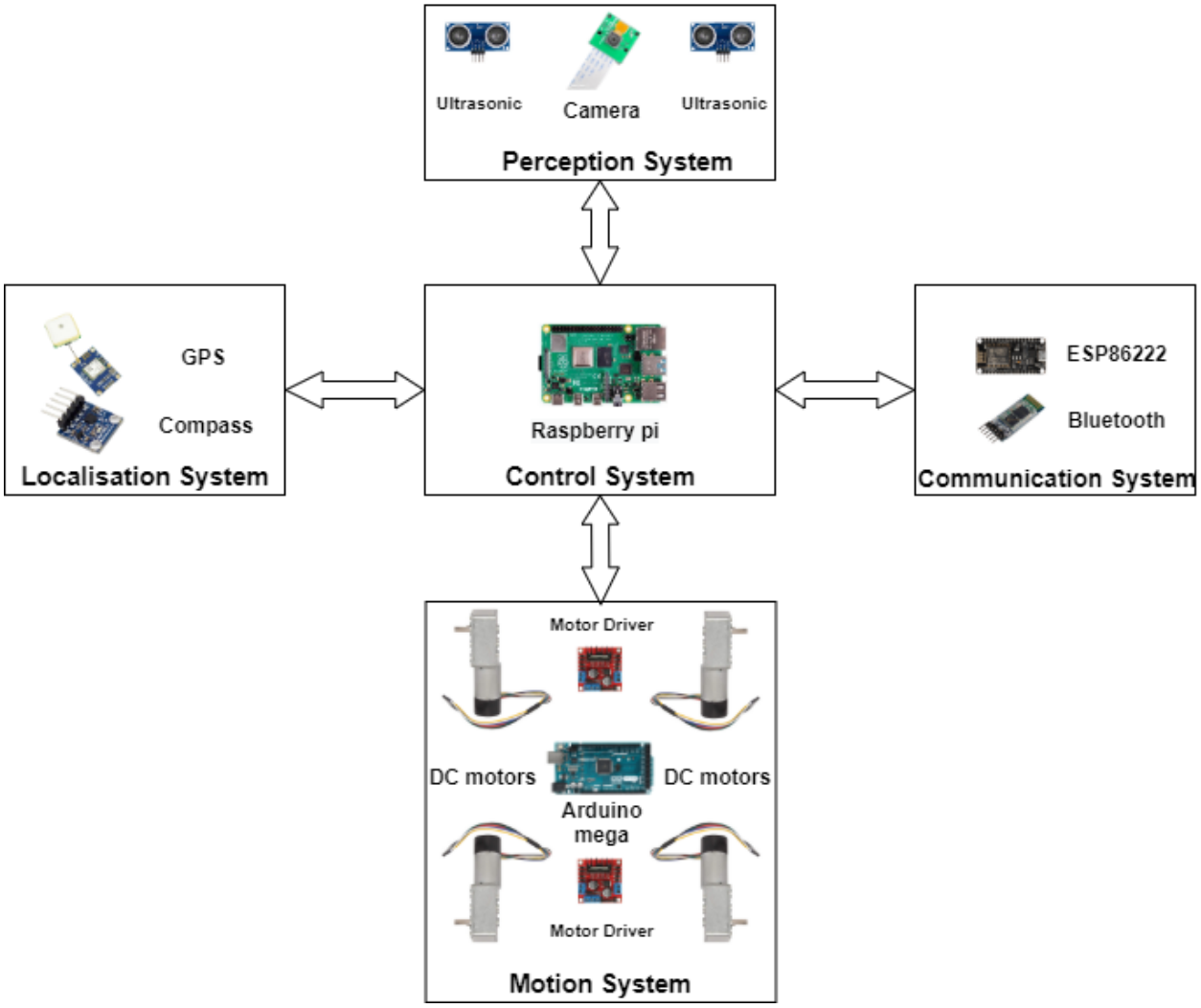


Figure 2.16: System architecture

2.4.1 Control system

Acts as the robot's brain , it is responsible for making decisions after receiving external information from the sensors and translating it into commands to be sent to the micro-controller unit.

2.4.2 Motion system

It is responsible for the robot's movement in various directions by operating the motors through signals issued by the monitoring unit.

2.4.3 Localization system

It represents a system that answers the question **"Where am I ?"** concerning the robot, by determining the robot's coordinates and the steering angle concerning the desired path.

2.4.4 Communication system

This system provides the robot with the ability to communicate with the command station to receive and send all the necessary data, for example (path coordinates, robot coordinates, etc.).

2.4.5 Perception system

It allows the robot to recognize its surrounding environment through the camera and sensors to overcome obstacles that hinder the navigation process.

2.5 Autonomous Navigation Approaches

The proposed robot has the ability to navigate through two different approaches, the first directed to open spaces and the second directed to closed spaces. Since satellite signal strength is more efficient outside buildings and in public places, we relied on GPS for navigation. Using the robot's coordinates and the target's coordinates, the robot determines the angle and distance between itself and the target point. Then start navigating to the target directly, all the details of the method will be explained in Section 2.5.1.

In the second approach, we chose the appropriate method for closed and limited spaces, where reliance on GPS signals is almost non-existent. This method is based on a line tracking system, where the camera is used to detect the line that has been previously drawn and prepared on the ground in the form of known paths, and the robot is moved according to the angle of

deviation of the line extracted from the camera frames. The method and code source are shown in Section 2.5.2.

2.5.1 GPS based navigation

For the purpose of designing a simple, low-cost and straightforward way to build a navigation system for an autonomous robot to move from point A to point B based only on GPS coordinates, the model uses a very simple calculation method which can be implemented by low-cost micro-controllers such as ESP32 or Arduino.

The GPS module and compass module capture GPS and heading data. After calculating all the necessary components of the algorithm, the micro-controller (Arduino mega) sends a signal to the motor driver based on the outputs of the algorithm. The entire system can be seen as shown in Figure 2.17, which summarizes all the steps of the robot program equipped with their required hardware. It is a closed loop system with feedback from different sensors and GPS. The entire approach is presented in the next subsections.

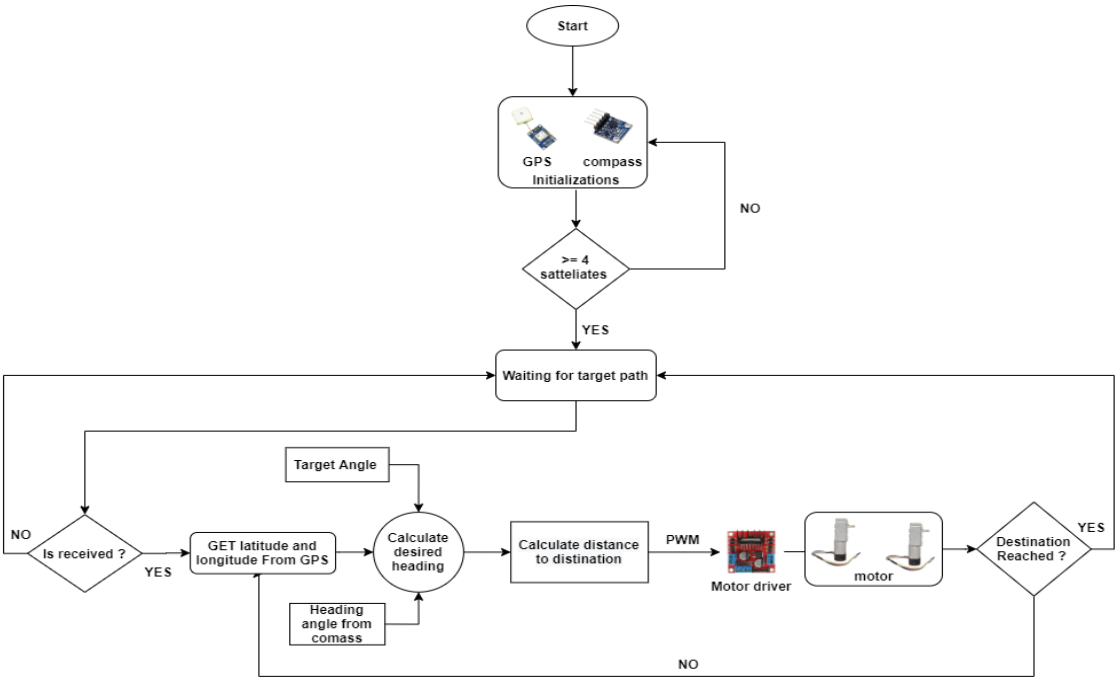


Figure 2.17: Workflow of GPS based navigation system

Working principle of GPS navigation system:

The user can use the proposed mobile robot via a mobile application Figure. 2.21, starting by selecting the desired destination, also he can show the path from the robot's position to the selected destination in the satellite map by pressing the "Show robot location" button. Figure. 2.18 depicts an example of a path to be passed by the robot.

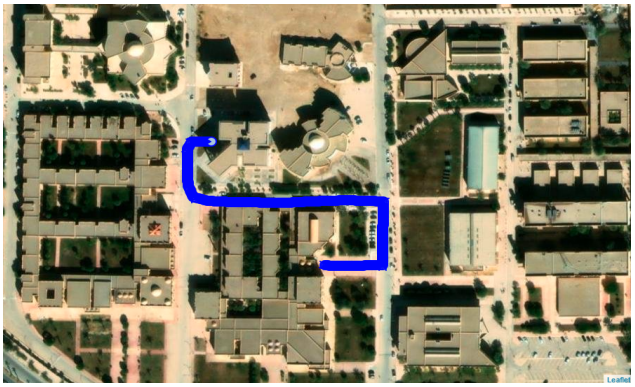


Figure 2.18: Sample of a path form LINFI laboratory to Faculty of exact sciences, natural and life sciences

The robot software begins by initializing the GPS sensors and compass. The working principle of the robot follows 9 steps. These steps are described in the following.

1. The user determines the target path through the mobile application interface shown in Figure. 2.21c, based on a map that shows the path from the starting point to the destination point in the form of a drawn line, then sends the command by pressing the "start navigate" button.

2. Raspberry Pi unit receives the path data and converts it to a JSON object, then sends the path in an array format of latitude and longitude coordinates to the Arduino module through the UARTs Serial communication protocol.

3. The GPS module searches for satellites and reads the data Current latitude and longitude. To calculate latitude and accurately determine the longitude, the GPS unit must be locked With at least 4 satellites.

4. Then the robot calculates the Distance to the target and GPS Cours from the latitude and

longitude data. We have used the haversine formula [12]to calculate the great-circle distance between two points on the Earth's surface:

$$d = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

where:

$$\Delta\phi = \phi_2 - \phi_1$$
$$\Delta\lambda = \lambda_2 - \lambda_1$$

- **d** represents the distance between two points on the surface of a sphere (in this case, the Earth), measured along the sphere's surface.
- **R** is the radius of the sphere. In the context of the haversine formula, it represents the radius of the Earth. The R-value is approximately 6371 kilometers or 3959 miles, depending on the units you're using.
- $\Delta\phi$ is the difference in latitude between two points, measured in radians. Where ϕ_1 and ϕ_2 are the latitudes of the two points, respectively.
- $\Delta\lambda$ is the difference in longitude between the two points, measured in radians. Where λ_1 and λ_2 are the longitudes of the two points, respectively.

5. Then the robot calculates the target angle from the latitude and longitude data. The angles must be measured in radians.
6. The current heading angle is received from the compass unit and the error angle is measured from the difference between the current heading angle and the target's angle.
7. After knowing the angle of error, we know whether it is a robot It needs to turn left or right. The four motors get pulses according to the Error angle of a micro-controller. Engine L298N The driver operates the motors to align the robot with the target angle.

8. The robot checks if it has reached the threshold value from its destination (in our robot, we set the threshold value to 1 meter). If the robot does not reach the destination, it will recalculate two angles (the target and the current heading) and the distance to the destination. This loop continues till the robot reaches its destination.

9. When the robot reaches its destination, it sends a signal to the control unit, then stops and waits for the user to type the entered password to receive a delivery.

2.5.2 Follow line based navigation

For the purpose of designing a line-following system to build a navigation system for an autonomous robot to move from point A to point B relying only on a line on the floor, the model uses a simple detection algorithm that can be implemented in a Raspberry Pi microprocessor.

The camera module detects the line based on its color. After determining the centroid of the frame based on the dimensions of the frame, the algorithm determines the middle of the line to be followed. The system extracts the value of the steering angle based on the difference between the centroid and the position of the line in the frame. The micro-controller (Arduino Mega) sends a signal to the motor driver based on the algorithm's outputs.

Working principle of follow line navigation system:

The robot starts by initializing the camera module. The working principle of the robot follows 5 steps. These steps are described in the following. Figure. 2.19 shows the flowchart of the robot program, which summarizes all the steps of the robot program equipped with their required hardware.

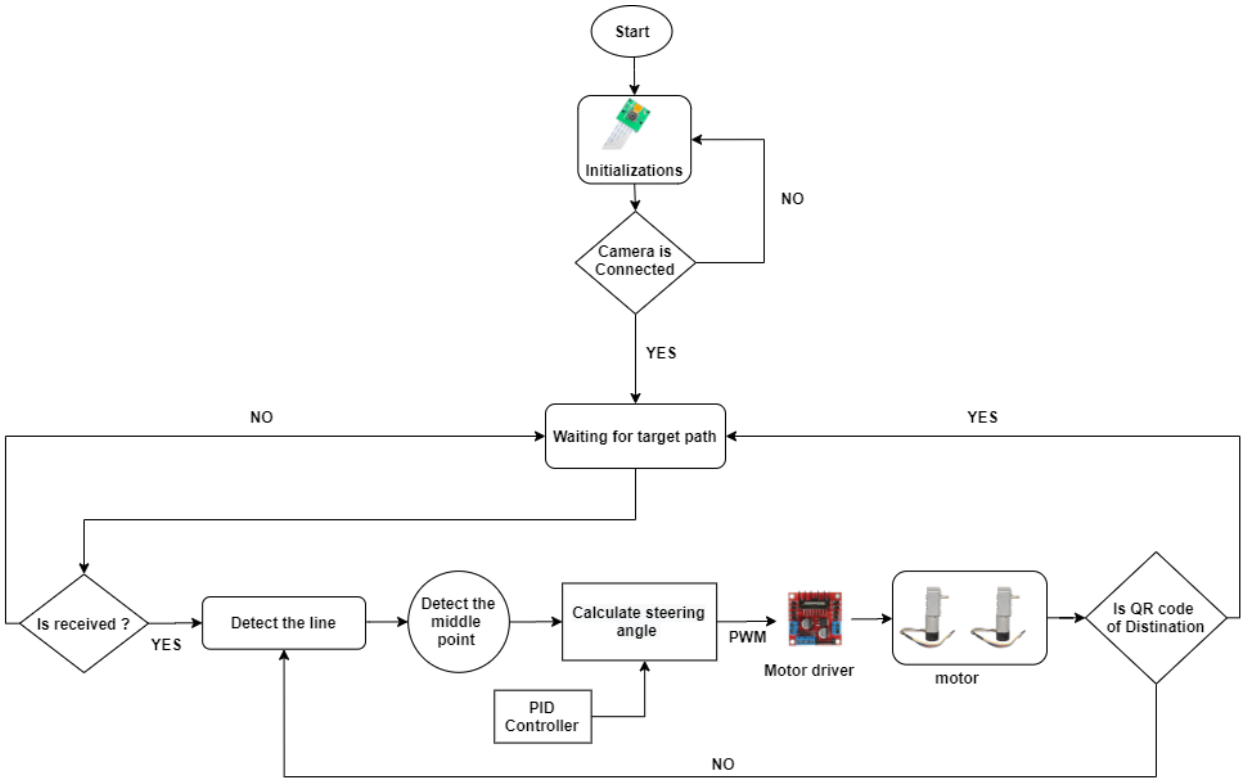


Figure 2.19: Workflow of follow line based navigation system

1. Raspberry Pi unit start line detection code by setting the camera module to start as shown in following code.

```
import cv2
import numpy as np

# Initialize the video capture object to read from the first camera (
    usually the webcam).
cap = cv2.VideoCapture(0)
```

2. Then Detection Algorithm Within an infinite loop, each frame is read and processed to create a binary mask that highlights the color within a specified range. Contours are then detected in the mask using cv2.findContours() method. The largest contour, assumed to be the line.

```
# Start an infinite loop to process video frames.
while True:
    # Capture a frame from the camera.
    ret, frame = cap.read()

```

```
    # Define the lower and upper bounds for the color range.
    # These bounds are intended for binary masking based on color.
    low_b = np.uint8([50, 50, 50])
    high_b = np.uint8([0, 0, 0])

    # Create a binary mask where the color of the pixels within the
   range is set to white (255) and the others to black (0).
    mask = cv2.inRange(frame, high_b, low_b)

    # Find the contours in the binary mask.
    contours, hierarchy = cv2.findContours(mask, 1, cv2.
   CHAIN_APPROX_NONE)
```

3. The centroid is determined and calculated using the moments of the image. Based on the x-coordinate of the centroid, the program sends directional commands to the Arduino: “Turn left,” “Keep going,” or “Turn right.”

```
# Check if any contours are found.
    if len(contours) > 0:
        # Find the largest contour based on the area.
        c = max(contours, key=cv2.contourArea)

        # Calculate moments for the largest contour.
        M = cv2.moments(c)

        # Calculate the centroid of the contour if the area (m00) is
   not zero.
        if M["m00"] != 0:
            cx = int(M['m10'] / M['m00'])
            cy = int(M['m01'] / M['m00'])
            print("CX : " + str(cx) + "  CY : " + str(cy))

            # Determine direction based on the x-coordinate of the
   centroid.
            if cx >= 120:
                print("Turn Left")
            elif 40 < cx < 120:
                print("On Track!")
            elif cx <= 40:
                print("Turn Right")

            # Draw a circle at the centroid of the largest contour.
            cv2.circle(frame, (cx, cy), 5, (255, 255, 255), -1)
    else:
```

```
26          # Print a message if no contours are found.
27          print("I don't see the line")
```

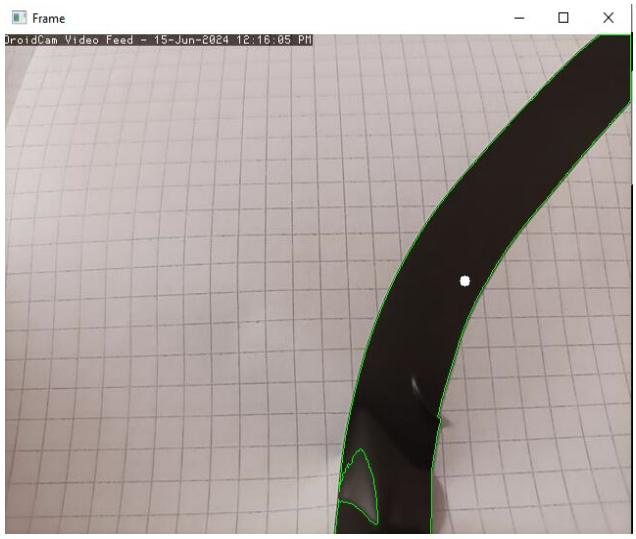
4. A circle is drawn at the centroid on the frame to visualize the tracking. The mask and the processed frame are displayed in separate windows

```
1 # Draw all contours on the frame.
2     cv2.drawContours(frame, contours, -1, (0, 255, 0), 1)
3 
4     # Display the binary mask.
5     cv2.imshow("Mask", mask)
6 
7     # Display the frame with contours and centroid.
8     cv2.imshow("Frame", frame)
```

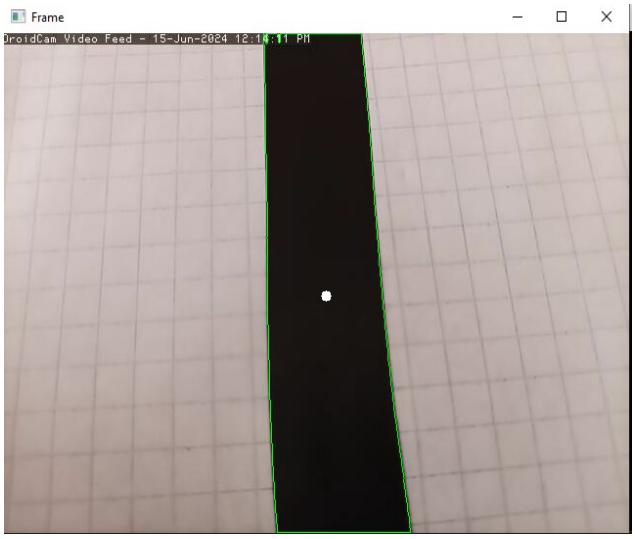
2.6 Experimental Results

In this section, we present some results after completing the construction of the prototype of the robot.

Results of follow line system:

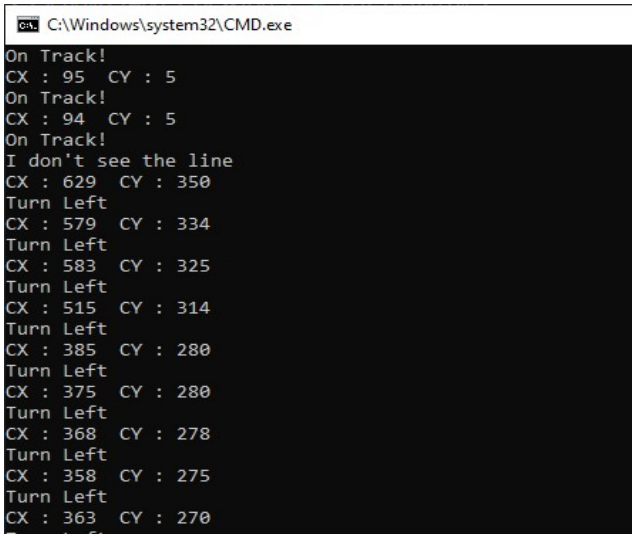


(a) Figure showing the determination of the dimensions of the black line in the case of inclination with the determination of the middle of the line



(b) Figure showing the determination of the dimensions of the black line in the case of straightness with the determination of the middle of the line

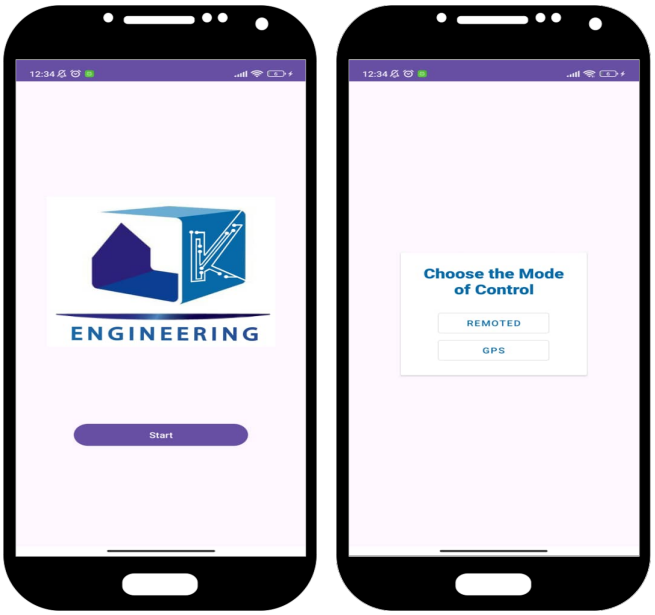
(c) A window showing the mask applied to the original image



(d) Commands that appear on the screen and are sent to Arduino

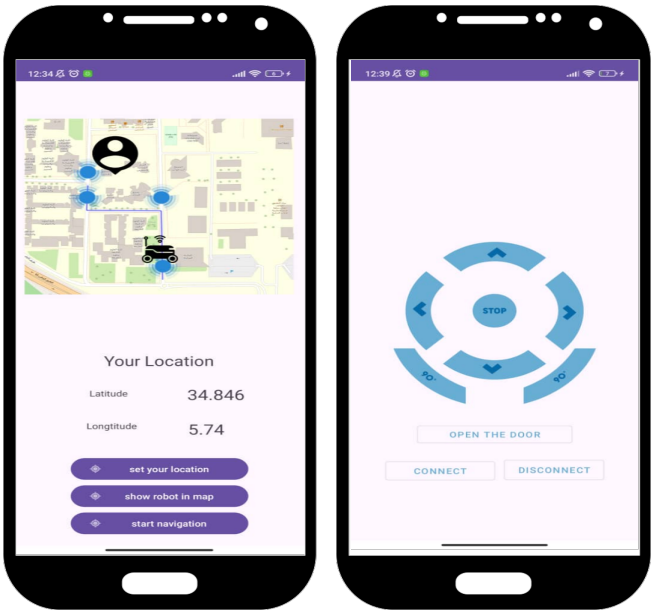
Figure 2.20: Results of detecting the line and calculating the steering angle code

Mobile application interfaces:



(a) Welcome page of the application

(b) select the mode control page which provide two mode of control remoted and gps mode contro



(c) interface of gps mode contol showing the location of the user and the robot

(d) interface of remoted contol mode showing control buttons

Figure 2.21: Interfaces of mobile application to control the robot

Prototype:

Figure 2.22: robot prototype

2.7 Conclusion

In this chapter, we detail the overall process of designing and implementing a mobile delivery robot. Starting from the 3D modeling phase, we designed the physical structure of the robot, ensuring that the design meets functional and operational requirements. Next, we carefully selected the appropriate components needed to build the robot, balancing performance, cost and compatibility. Finally, we identified the programming languages and tools used during the programming phase, which were crucial in enabling the robot's autonomous functionality. This methodological approach not only emphasizes the multidisciplinary nature of robotics, but also highlights the integration of mechanical design, hardware selection, and software development in creating an effective delivery robot.

Chapter 3

Modeling and Verification of a Delivery Multi-robots System

3.1 Introduction

In this chapter we are interested to model and to simulate the behavior of the system in case of a network of robots (a Collaborative Mobile Robot System or CMRS). In fact, in large establishments the delivery tasks require a large number of robots (Multi Robot System), which form a distributed network of robots that are connected to each other in order to perform transportation services. This networks of robots must collaborate and cooperate to handle the delivery task, based on inter-robots communication. In this kind of system, the efficiency of the entire network depends on the performance of each entity as well as on the cooperative protocol. This chapter aims to present a formal study of a CMRS using automata as a modeling tool and UPPAAL as a simulation-verification tool. The objective is to show the effeciency of the system and to study safety/liveness properties in this cooperative system.

3.2 Formal Modeling and Verification in UPPAAL-Tool

UPPAAL is a widely used tool for modeling, simulation, and verification of real-time systems. It facilitates the design and analysis of systems modeled as networks of timed automata, which are

extended finite-state machines augmented with clock variables. UPPAAL allows for the formal verification of properties expressed in temporal logic, ensuring system correctness with respect to timing constraints and behavioral specifications [17].

3.2.1 Automata in UPPAAL-tool

In UPPAAL, the primary modeling construct is the timed automaton, an extension of the classical finite automaton with the inclusion of real-valued clocks. These automata are used to model the behavior of real-time systems where timing constraints are crucial.

Definition of Automata:

An automaton is a mathematical model for a machine with a finite number of states, where transitions between states are triggered by specific events or conditions [26]. Formally, an automaton A can be defined as a tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of states.
- Σ is a finite set of input symbols (alphabet).
- δ is a transition function $\delta : Q \times \Sigma \to Q$.
- q_0 is the initial state.
- F is a set of accepting states.

Definition of Büchi Automata:

Büchi automata are a type of automaton used to accept infinite sequences, making them suitable for model-checking properties of systems that run indefinitely [22]. A Büchi automaton B is defined as a tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of states.
- Σ is a finite alphabet.
- δ is a transition relation $\delta \subseteq Q \times \Sigma \times Q$.

- q_0 is the initial state.
- F is a set of accepting states.

A run of the Büchi automaton is accepted if it visits the accepting states F infinitely often.

What Automata are Used in UPPAAL:

UPPAAL primarily uses timed automata for modeling. A timed automaton is a finite-state machine extended with real-valued clocks. Formally, a timed automaton TA is defined as a tuple $(L, l_0, C, \Sigma, E, I)$, where:

- L is a finite set of locations.
- l_0 is the initial location.
- C is a finite set of clocks.
- Σ is a finite set of actions.
- E is a set of edges (transitions), each being a tuple (l, g, a, r, l') where l and l' are locations, g is a guard (a constraint on clocks), a is an action, and r is a set of clocks to reset.
- I is a mapping from locations to invariants (constraints on clocks).

3.2.2 CTL and Model-Checking in UPPAAL-tool

UPPAAL supports the verification of system properties using temporal logics, specifically the Computation Tree Logic (CTL). This allows for the formal specification and verification of temporal properties in real-time systems.

Temporal Logic:

Temporal logic is a formalism used to specify statements about the temporal ordering of events within a system. It extends classical logic with temporal operators, enabling the expression of properties such as "eventually," "always," or "until." Temporal logic is particularly useful in the context of verifying dynamic behaviors of systems over time [15].

CTL:

Computation Tree Logic (CTL) is a type of temporal logic used for specifying properties of branching time structures. In CTL, formulas are constructed using path quantifiers (A - for all paths, E - there exists a path) and temporal operators (X - next, F - eventually, G - globally, U - until). A typical CTL formula might express that "on all paths, eventually a certain condition holds" [13].

Model-Checking in UPPAAL:

Model-checking is an automated technique used to verify the correctness of finite-state systems with respect to a given specification. In UPPAAL, model-checking involves verifying that a timed automaton model satisfies properties expressed in a subset of CTL [19]. The verification process in UPPAAL includes:

- **Model Construction**: Defining the system as a network of timed automata.
- **Property Specification**: Expressing desired properties using temporal logic.
- **Verification**: Using the UPPAAL model checker to automatically verify whether the model satisfies the specified properties. This involves exploring the state space of the model and checking the truth of the temporal logic formulas.

UPPAAL's model-checking capabilities are particularly effective for verifying real-time systems where timing constraints are critical, ensuring that the system behaves as expected under all possible scenarios.

3.3 Formal Modeling and Verification of the System

To prove the efficiency of the proposed cooperative multi-robots system, we decided to use formal modeling tools such as Büchi Automaton and UPPAAL tool to specify, simulate, and model-check the proposed system. In this section, we will present (i) the specification of the system using timed automata, and (ii) the model-checking of some basic properties.

3.3.1 Modeling of the System using Automata

The model of the system consists of two timed automata templates: the coordinator automaton (see Figure. 3.1) and the robot automaton (see Figure. 3.2). By instantiating these two templates and combining them using the UPPAAL tool, we can simulate and verify the whole system's behavior which may be composed of several cooperative robots.

Coordinator Automaton:

The coordinator automaton represents the main states of the coordinator system with all possible basic transitions from selecting the given target path to reaching the desired destination. In this model, "`start`" state is the starting point of the entire control system. After starting the system, it switches to "`Wait`" state for starting communication with the robots. In the "`Wait`" state, the system waits to receive the "`ID`" of each robot through the channel "`Connect_to_robot`" and it saves all received robots identifiers in an array which will be used for direct communication with each robot individually.

When the coordinator ensures communication with all robots ("`all_ robots_connect = true`"), it moves to the "`wait_for_set_destination`" state that is determined by the user through the mobile application. After receiving the path, the system sends the path to all connected robots via the path transmission channel "`send_path`". Hence, each robot is expected to send its distance from the target to allow the coordinator to select the closest robot to the target. Following selecting the nearest robot, the system sends the selection command to the selected robot and then waits for the robot to start moving.

During the waiting phase to reach the target "`wait_for_destination_reached`", the system receives the target arrival signal from the robot, switches to "`Destination_reached`" state to inform the user, and then returns to the path determination waiting state again.

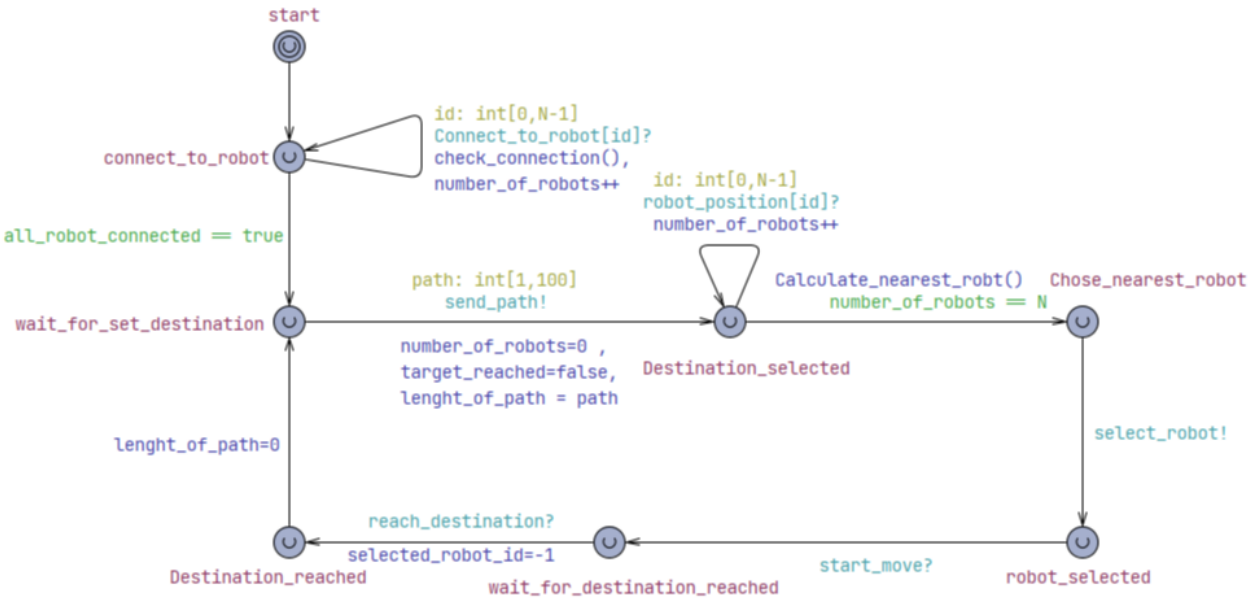


Figure 3.1: Coordinator Automaton

Robot Automaton:

The robot automaton represents the behavior of the robot and the main states that the robot visits, with all possible basic transformations during the navigation process.

When the robot starts working, it starts from the "`start`" state and sends its "`ID`" to the coordinator to establish direct communication with that coordinator. After sending the "`ID`", the robot automaton reaches the "`Wait`" state to receive the destination from the coordinator. Upon receiving the destination of the target, the robot "`distance_from_goal`" and sends this distance to the coordinator, which in turn performs the process of electing the nearest robot from the target, hence, the robot moves to "`waiting_to_be selected`" state. After the election process, all robots receive the decision of the selected robot. Every robot compares its "`ID`" with "`selected_robot_id`" and if its "`ID`" is not equal to "`selected_robot_id`" then it moves to "`not_selected`" state, hence this robot returns again to "`Wait`" state, waiting for any new path (thus, a new delivery mission).

In a case where the "`ID`" of the robot is equal to the select "`selected_robot_id`" by the coordinator, the robot receives permission to navigate. The process of navigating begins as we have explained in the previous section about the principle of work. In this case, this robot sends the move signal to the coordinator.

Finally, when the robot reaches the destination, it sends the "`reach_destination`" command to the coordinator, then this robot returns to the "`Wait`" state, waiting for any new path (thus, a new delivery mission).

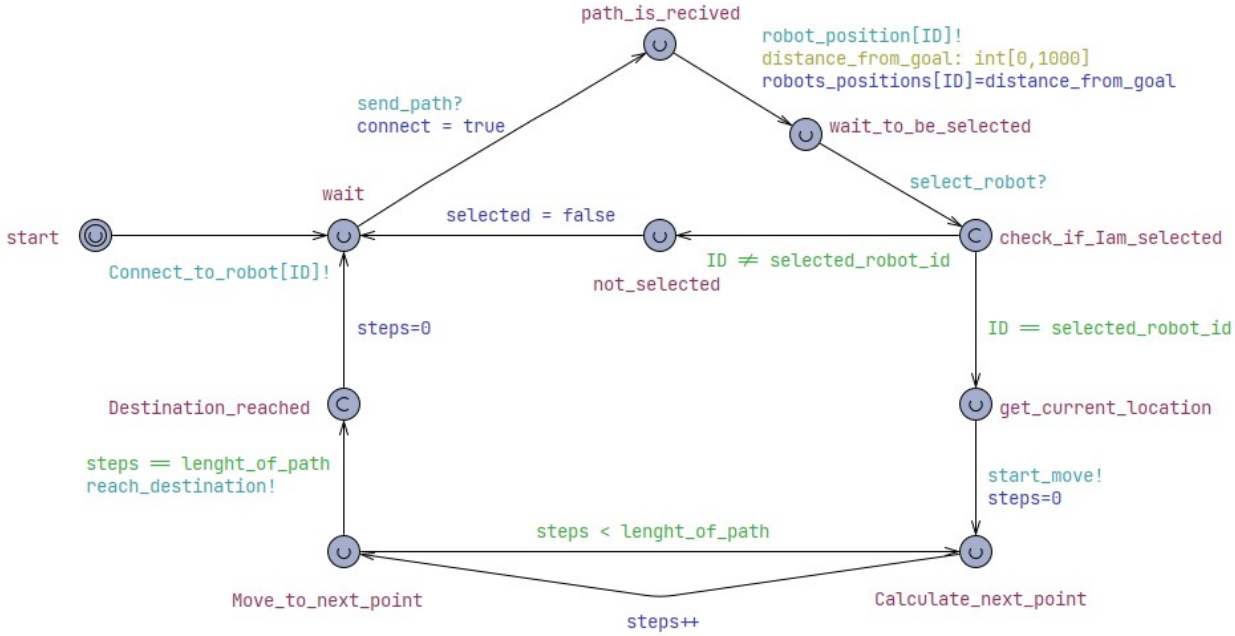


Figure 3.2: Robot Automaton

3.3.2 Model-checking and Evaluation

The model-checking based on automata using UPPAAL consists of checking CTL (computational tree logic) properties against the composed automata of the entire system. CTL represents a logic-based method for describing properties related to the system's behavior. It provides a set of operators that allow us to express various types of properties. Two main categories of properties are to be checked: **safety properties** and **liveness properties**. The following paragraphs present several formulas belonging to each of the previous categories and show if these formulas are fulfilled or not in the proposed system.

Safety properties: "something bad does not happen":

Safety properties express that "nothing bad will happen" when the system's running. These properties focus on preventing undesirable states or behaviors.

- *Deadlock Freedom*: This property ensures that the system does not stop working unexpectedly throughout its operation, and is considered one of the most important features to check. we use formula 3.1 to ensure this property.

$$A[\,]\ not\ Deadlock \tag{3.1}$$

- *Resource Availability*: This property allows us to verify the availability of system components. For example, in our case, we can use 3.2 to verify that at least one robot is in service.

$$A <>\ (coordinator.all\_robot\_connected = true) \tag{3.2}$$

Liveness: "Something good eventually happens":

Liveness properties state that "something good will happen." They ensure that specific conditions eventually hold while the system is running.

- *Response to Requests*: by ensuring this property we ensure that every delivery request issued by the coordinator must be completed by at most one robot.in CTL we can verify this property using formula 3.3:

$$cor.Chose\_nearest\_robot \longrightarrow cor.robot\_selected \tag{3.3}$$

- *Task Completion*: Every delivery task lunched by the coordinator must be completed. Formula 3.4 defines the property of task completion.

$$(cor.wait\_for\_distination\_reached \longrightarrow cor.Distination\_reached) \tag{3.4}$$

- *Fairness*: Tasks must be distributed among the robots fairly for, in CTL we can verify this property using formula 3.5:

$$E <> (r1.ID = selected\_robot\_id\ or\ r2.ID = selected\_robot\_id) \tag{3.5}$$

We have modeled the model using the UPPAAL Version 5.0 and the model was simulated and properties are checked on a computer of Intel(R) Core(TM) i5-6300U CPU @ 2.40GHzU and 8 GB RAM, running Windows 10 Pro 64-bit.

Table 3.1: Results of model-checker in UPPAAL

Property name	**CTL formula**	**Result**
Deadlock Freedom	A[ ] not Deadlock	Satisfied
Resource Availability	A<>(coordinator.all_robot_connected = true)	satisfied
Response to Requests	(cor.Chose_nearest_robot ⟶ cor.robot_selected)	satisfied
Task Completion	(cor.wait_for_distination_reached ⟶ cor.Distination_reached)	satisfied
Fairness	E<> (r1.ID = selected_robot_id or r2.ID = selected_robot_id)	satisfied

3.4 Discussion and Conclusion

In this chapter, we use Büchi automata and CTL (computational tree logic) properties to model and verify the collaborative mobile robot system (CMRS). Recognizing the complexity of multi-robot systems, we emphasized the necessity of robust communication and collaboration protocols between robots to ensure efficient task execution.

Using automation-based model checking with the UPPAAL simulation and validation tool, we analyzed the behavior of the CMRS. Through this formal approach, we demonstrated the efficiency of the system and carefully examined its critical safety and liveness properties.

The results of examining the formal UPPAAL model show that the proposed system ensures cooperation between robots, highlighting its role in improving performance and protecting against possible failures.

Conclusion and Perspectives

Conclusion

The field of mobile robotics has made great strides, demonstrating its ability to revolutionize various sectors by enhancing efficiency and productivity. In this dissertation, we have tried to highlight the effectiveness of mobile robots, especially in the context of delivery services, where their applications can meet the increasing requirements of logistics services and e-commerce. This work underscores the importance of integrating cutting-edge technologies to improve delivery processes.

Although it is difficult to adopt mobile robotics technologies in a region like Algeria due to limited resources and infrastructure. However, this thesis underscores the opportunity for these regions to benefit significantly from the deployment of mobile robots in various sectors, including delivery, industry and healthcare.

The proposed mobile delivery robot system, developed through this dissertation, provided a comprehensive solution consisting of three main stages:

- Designing a prototype of an autonomous mobile robot that can be used indoors or outdoors. The robot can move within institutions using a line tracking system using a camera installed on the front of the robot that can determine the location and extract rotation angles. The robot also has a navigation system based on the Global Positioning System (GPS), which enables it for external uses as well, as the robot receives the coordinates of the destination through the mobile application, which provided a simple user interface for users, which makes it easy to use. The angle and distance of the path are determined through special algorithms before starting the navigation process towards the target, and the proposed mobile robot can be useful in delivering food, groceries, and daily useful products to our desired destination. Or use it inside hospitals, given the severe pressure

that the healthcare system is currently under, as delivering medical equipment inside hospitals may be risky. At the same time, our mobile delivery robot can be an effective intelligent logistics system to solve important logistics problems in the supply chain of large factories, especially for the purpose of reducing the cost, time and effort of workers. The lightweight construction of our robot makes it safe against collisions and accidents that may injure users.

- Providing a formal specification using Buchi automata for a multi-robot collaborative system.
- And finally, check safety and liveliness properties by validating the system using UPPAAL tool. The results of the formal model-checker UPPAAL show that the proposed system satifies a set of safty and liveness properties.

This robot aims to enhance the delivery process by taking advantage of robotics technologies, thus improving the efficiency of transportation and delivery operations.

Perspectives

As future work, we seek to improve several aspects of this work, such as:

- Enhanced sensor integration and data fusion: Future work could focus on incorporating more advanced sensors and improving data fusion techniques to enhance the robot's perception and decision-making capabilities.
- Scalability and network optimization: scale the proposed system to large fleets of delivery robots will be crucial. explore advanced communication protocols, distributed computing, and network optimization.
- Energy efficiency and sustainability: Addressing the energy consumption of mobile robots is essential for their sustainable deployment.
- Integration with other technologies: Exploring the integration of mobile robots with other emerging technologies, such as the Internet of Things (IoT), blockchain technology for secure transactions, and artificial intelligence for predictive analytics.

- Field trials and real-world deployments: Conducting large-scale field trials and real-world deployments in different environments will be crucial to verify the effectiveness of the proposed system.

References

[1] Amazon releases updates on drone delivery, robots, and packaging. https://www.aboutamazon.com/news/operations/amazon-delivering-the-future-2023-announcements. (accessed: 09.05.2024).

[2] A brief guide to select a robot motor. https://rozum.com/find-robot-motor/. (accessed: 25.05.2024).

[3] Ces drones sont capables de chasser en essaim. https://www.futura-sciences.com/tech/actualites/drone-ces-drones-sont-capables-chasser-essaim-98335/. (accessed: 18.06.2024).

[4] Flir to provide centaur ugvs to us marine corps. https://www.unmannedsystemstechnology.com/2020/04/flir-to-provide-centaur-ugvs-to-us-marine-corps/. (accessed: 13.06.2024).

[5] Irobot roomba 671 robot vacuum cleaner. https://www.robot-advance.com/EN/art-irobot-roomba-671-robot-vacuum-cleaner-3663.htm. (accessed: 10.06.2024).

[6] Meet aquanaut, the underwater transformer. https://spectrum.ieee.org/meet-aquanaut-the-underwater-transformer. (accessed: 16.06.2024).

[7] Microcontrollers. https://ensemblebot.quadrivium.dk/project-overview/microcontrollers-arduino-teensy/. (accessed: 11.06.2024).

[8] Robótica industrial, automatización y empleo. https://impactotic.co/tecnologia/robotica-industrial-automatizacion-y-empleo/. (accessed: 13.06.2024).

[9] Starship technologies: Autonomous robot delivery - the future of delivery - today!. https://starship.co/. (accessed: 17.05.2024).

[10] Mansoor Alam. Mobile robot navigation using ros navigation stack. https://medium.com/@mansooralam129047/mobile-robot-navigation-using-ros-navigation-stack-c093aa93e8a7. (accessed: 18.06.2024).

[11] Elin Alverhed, Simon Hellgren, Hanna Isaksson, Lisa Olsson, Hanna Palmqvist, and Jonas Flodén. Autonomous last-mile delivery robots: a literature review. *European Transport Research Review*, 16, 01 2024.

[12] Rezania Azdy and Febriyanti Darnis. Use of haversine formula in finding distance between temporary shelter and waste end processing sites. *Journal of Physics: Conference Series*, 1500:012104, 04 2020.

[13] Christel Baier and Joost-Pieter Katoen. Principles of model checking. 2008.

[14] Uma Balasubramanyam and Poonam Kapoor. Anesthetic challenges in minimally invasive cardiac surgery. *Journal of Cardiac Critical Care TSS*, 03, 01 2020.

[15] Mordechai Ben-Ari. *Temporal Logic: Formulas, Models, Tableaux*, pages 231–262. Springer London, London, 2012.

[16] Mordechai Ben-Ari and Francesco Mondada. *Robots and Their Applications*, pages 1–20. 01 2018.

[17] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Uppaal — a tool suite for automatic verification of real-time systems. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III*, pages 232–243, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[18] Andrea Cherubini and David Navarro-Alarcon. Sensor-based control for collaborative robots: Fundamentals, challenges and opportunities, 07 2020.

[19] Edmund M. Clarke and Bernd-Holger Schlingloff. Model checking. 1999.

[20] ROSS et al. *Industrial robotics fundamentals: Theory and applications*. 2017.

[21] Ethannoah. Articulated robots. https://medium.com/@ethannoah1122_94684/articulated-robots-e45dfd913637. (accessed: 06.06.2024).

[22] Dimitra Giannakopoulou and Flavio Lerda. From states to transitions: Improving translation of ltl formulae to büchi automata. In Doron A. Peled and Moshe Y. Vardi, editors, *Formal Techniques for Networked and Distributed Sytems — FORTE 2002*, pages 308–326, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[23] Jennifer Gielis, Ajay Shankar, and Amanda Prorok. A critical review of communications in multi-robot systems, 06 2022.

[24] Stanislav Ivanov and Craig Webster. *Robots, Artificial Intelligence and Service Automation in Travel, Tourism and Hospitality*. 10 2019.

[25] Abdul Khaliq, Ali Waqas, Qasim Nisar, Shahbaz Haider, and Zunaina Asghar. Application of ai and robotics in hospitality sector: A resource gain and resource loss perspective. *Technology in Society*, 68:101807, 11 2021.

[26] Bakhadyr Khoussainov and Anil Nerode. *Automata theory and its applications*, volume 21. Springer Science & Business Media, 2012.

[27] David Leslie. Isaac asimov: centenary of the great explainer. https://doi.org/10.1038/d41586-020-00176-4. (accessed: 25.05.2024).

[28] Matt Luckcuck. Using formal methods for autonomous systems: Five recipes for formal verification. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 237:1748006X2110349, 07 2021.

[29] Issa Nesnas, Lorraine Fesq, and Richard Volpe. Autonomy for space robots: Past, present, and future. *Current Robotics Reports*, 2, 09 2021.

[30] Sivapong Nilwong and Genci Capi. Reinforcement learning based outdoor navigation system for mobile robots. pages 219–224, 01 2020.

[31] Aleksandr Novitsky and Dmitry Yukhimets. The navigation method of wheeled mobile robot based on data fusion obtained from onboard sensors and camera. pages 574–579, 10 2015.

[32] Aleksandr Novitsky and Dmitry Yukhimets. The navigation method of wheeled mobile robot based on data fusion obtained from onboard sensors and camera. pages 574–579, 10 2015.

[33] Bashra Oleiwi, Asif Mahfuz, and H. Roth. Application of fuzzy logic for collision avoidance of mobile robots in dynamic-indoor environments. 01 2021.

[34] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. volume 3, 01 2009.

[35] Bernd-Holger Schlingloff. Specification and verification of collaborative transport robots. pages 3–8, 04 2018.

[36] Nitin Sharma, Jitendra Kumar Pandey, and Surajit Mondal. A review of mobile robots: Applications and future prospect. *International Journal of Precision Engineering and Manufacturing*, 24(9):1695–1706, 2023.

[37] Jasmin Velagic, Nedim Osmic, Faris Hodzic, and Harun Šiljak. Outdoor navigation of a mobile robot using gps and gprs communication system. 01 2011.

[38] Jinming Yao. Path planning algorithm of indoor mobile robot based on ros system. pages 523–529, 08 2023.