



Palestine Polytechnic University
College of Information Technology and Computer Systems Engineering
Department of Computer Systems Engineering

Street Painting Robot

Islam Alama
Lama Halayka

Supervised by:
Dr. Amal Dweik

*To fulfill the requirements for a bachelor's degree in the field of Computer
Systems Engineering*

2024-2025

Acknowledgment

In the name of the Almighty, the most compassionate, the most merciful, who bestowed upon us strength and knowledge, enabling the successful completion of this project. Embarking on this endeavor provided us with valuable experience , we extend our gratitude to all those who contributed. Our heartfelt appreciation goes to our graduation project supervisor Dr. Amal Al-Dweik , and for the Eng.Mohammad Al Mohtaseb , for their unwavering guidance, encouragement, and support throughout the semester. We also extend our thanks to the dedicated educators in the College of Information Technology and Computer Engineering, who selflessly shared their wisdom, shaping us into diligent engineers. A special acknowledgment is due to our families, whose generous encouragement and steadfast support have been constant throughout our lives. To our friends, we express sincere gratitude for your unwavering support during this significant chapter of our journey. In conclusion, we extend our thanks to everyone who played a role in supporting and encouraging us, ultimately contributing to the successful completion of our graduation project.

Abstract

The manual road marking process is often time-consuming and prone to errors, leading to delays in road construction projects. To address these challenges, this project introduces a street painting robot that significantly enhances the efficiency and accuracy of street marking. This positively impacts traffic flow, economic activity, and road safety.

In order to efficiently achieve its goals, the system receives instructions about the painting process from a mobile application and integrates advanced technologies, including obstacle detection and a valve-controlled paint system for precise and regulated paint flow. Equipped with an ultrasonic sensor, the system continuously monitors paint levels and provides real-time alerts for low levels or operational problems. The robot autonomously navigates the streets and applies paint with precision, using a map-based localization system to determine its position and plan its movements. Adjust its path dynamically in response to obstacles, ensuring seamless operation and accurate line marking.

After implementing and testing the system, it is successfully achieves its goals, the system minimizes human labor, reduces project timelines, and delivers superior accuracy and reliability in road construction projects, marking a significant step toward smarter, automated infrastructure development.

Keywords: Mobile robot, street marking, obstacle detection, navigation, painting, localization.

المخلص

تمثل عملية رسم العلامات على الطرق يدويًا تحديًا كبيرًا نظرًا لما تتطلبه من وقت طويل وما قد ينتج عنها من أخطاء، مما يتسبب في تأخير مشاريع الطرق ويؤثر على السلامة المرورية. لمواجهة هذه المشكلات، يقدم هذا المشروع روبوتًا لرسم الخطوط يعمل على تحسين كفاءة ودقة عملية التخطيط للطريق، مما يساهم في تعزيز انسيابية المرور ودعم النشاط الاقتصادي وتحسين مستويات الأمان على الطرق.

من أجل تحقيق أهدافه بكفاءة، يتلقى النظام تعليمات حول عملية الطلاء من تطبيق محمول ويدمج تقنيات متقدمة، بما في ذلك اكتشاف العوائق ونظام طلاء يتم التحكم فيه بواسطة صمام لتدفق الطلاء بدقة وتنظيم. مزودًا بمستشعر بالموجات فوق الصوتية، يراقب النظام مستويات الطلاء باستمرار ويوفر تنبيهات في الوقت الفعلي للمستويات المنخفضة أو المشكلات التشغيلية. يتنقل الروبوت بشكل مستقل في الشوارع ويضع الطلاء بدقة، باستخدام نظام تحديد المواقع القائم على الخريطة لتحديد موقعه والتخطيط لحركته. اضبط مساره ديناميكيًا استجابة للعقبات، مما يضمن التشغيل السلس وعلامات الخطوط الدقيقة.

بعد تنفيذ النظام واختباره، حقق أهدافه بنجاح، حيث يقلل النظام من العمالة البشرية، ويقلل من الجداول الزمنية للمشروع، ويوفر دقة وموثوقية فائقة في مشاريع بناء الطرق، مما يمثل خطوة مهمة نحو تطوير البنية التحتية الذكية والآلية.

الكلمات المفتاحية: الروبوت المتنقل، وضع علامات الطرق، اكتشاف العوائق، الملاحية، الطلاء، تحديد الموقع.

Contents

1	Introduction	1
1.1	Preface	1
1.2	Problem statement	1
1.3	Aims and objectives	1
1.4	Requirements	2
1.4.1	Functional Requirements	2
1.4.2	Non-functional Requirements	2
1.5	System Description	3
1.6	Limitations and constraints	4
1.7	Schedule	4
1.8	Report outline	4
2	Background	5
2.1	Preface	5
2.2	Theoretical background	5
2.2.1	Simultaneous Localization and Mapping	5
2.2.2	Localization	5
2.2.3	Obstacles Avoidance	6
2.2.4	Navigation	6
2.2.5	IoT protocols	6
2.2.6	Start Point Calculation for Line Painting	7
2.2.7	Laser Data Filtering	7
2.3	Literature review	7
2.3.1	Automatic Wood Plates Painting Machine	7
2.3.2	Automatic wall painting machine	8
2.3.3	Vertical Wall Printer	8
2.3.4	Powder Coating Machine	8
3	System Design	11
3.1	Preface	11
3.2	System components and Design options	11
3.2.1	hardware component and design options	11
3.2.1.1	Mobile Robot	11
3.2.1.2	Processing Unit	12
3.2.1.3	Obstacle Avoidance Sensor	13

3.2.1.4	Paint Level Sensor	14
3.2.1.5	Flow Control Valve	14
3.2.1.6	Relay	15
3.2.1.7	Motor	16
3.2.2	Software Components Options	16
3.2.2.1	Robot Operating System	16
3.2.2.2	Development Tool	17
3.2.2.3	Mobile Application Language	18
3.2.2.4	Python Programming Language	18
3.2.2.5	Visual Studio Code	18
3.3	Conceptual system design	19
3.4	Sequence diagrams	21
3.5	Schematic diagram	22
3.6	Pseudo-Code	22
3.7	Summary	23
4	System Implementation and Testing	24
4.1	Overview	24
4.2	Implementation Issues	24
4.2.1	Hardware Implementation	24
4.2.1.1	MP 400 Mobile Robot	24
4.2.1.2	Arduino	25
4.2.1.3	Ultrasonic Sensor	25
4.2.1.4	Servo Motor	26
4.2.1.5	Valve and Relay	27
4.2.1.6	Laser Scanner	27
4.2.2	Software implementation	29
4.2.2.1	ROS-Related Implementation	29
4.2.2.1.1	Installing Ubuntu Mate 20.04 Operating System	29
4.2.2.1.2	Installing ROS Noetic	29
4.2.2.1.3	MP 400 Installation	29
4.2.2.1.4	Mapping Using SLAM	29
4.2.2.1.5	Navigation	30
4.2.2.2	Robot Painting System Testing	31
4.2.2.2.1	Line Start Point Calculation	31
4.2.2.2.2	Data Processing for Line Start and End Points	31
4.2.2.2.3	Hardware Control Based on Position	32
4.2.2.2.4	Laser filter data	32
4.2.3	Mobile Application Implementation	33
4.2.3.1	MQTT Protocol	33
4.2.3.2	User Interface Design	33
4.3	Challenges	34
4.4	System Validation and Testing	34
4.4.1	System Testing	34

4.4.1.1	Hardware Component Testing	34
4.4.1.1.1	Ultrasonic Sensor Testing	34
4.4.1.1.2	Laser Scanner Testing	34
4.4.1.1.3	Valve Testing	34
4.4.1.1.4	Relay Testing	35
4.4.1.2	System Testing	35
4.4.1.2.1	Mobile Application Connectivity Testing	35
4.4.1.2.2	Line Drawing Accuracy Testing	35
4.4.2	Obstacle in Line Drawing Path	36
4.4.3	System Validation	36
4.5	Summary	36
5	Discussion of Results	37
5.1	Preface	37
5.2	Discussion of Results	37
5.2.1	Streamlined Operation	37
5.2.2	Handling of Obstacles	37
5.2.3	Impact of Line Parameters	37
5.2.4	Integration with the Mobile Application	38
5.2.5	System Resilience	38
5.3	Summary	38
6	Conclusion and future work	39
6.1	Preface	39
6.2	Conclusion	39
6.3	Future work	40

List of Figures

1.1 System Description	3
2.1 Autonomous robot navigation pipeline	6
3.1 Mp-400	12
3.2 Positions of the laser scanner in robot MP-400	13
3.3 Ultrasonic Sensor	14
3.4 System block diagram	20
3.5 System conceptual diagram	20
3.6 Sequence diagram of a street painting robot system	21
3.7 Schematic diagram for DC Motor(MP400 robot) with Ardunoi	22
4.1 Robot processor accessed via VNC	24
4.2 Arduino connected to MP 400 via USB	25
4.3 Ultrasonic sensor postion	25
4.4 Low-paint notification sent to mobile app	26
4.5 Roller position controlled by servo motor	26
4.6 Placement of the valve	27
4.7 Laser scanner mounted on the front of the robot	28
4.8 RViz visualization showing detected obstacles (red) and navigation path (blue).	28
4.9 Generated 2D map	30
4.10 Laser data filtering code implementation.	32
4.11 Mobile app user interface showing data exchange with the robot	33
4.12 Testing the Ultrasonic Sensor	34
4.13 MQTT Data Published in the Mobile Application	35
4.14 Compare the target coordinates	35
4.15 Comparison file for validating robot position against saved data.	36
4.16 notification the user obstacle in Line Drawing Path	36

List of Tables

1.1 Project schedule in the summer and the first semester	4
2.1 Comparison of Related Literatures.	9
3.1 Comparison between mobile robot options	11
3.2 Comparison between Processing Unit options	12
3.3 Comparison between Obstacle Avoidance Sensor options	13
3.4 Comparison between Paint Level Sensor options	14
3.5 Comparison between Flow Control Valve options	15
3.6 Comparison between Relay options	15
3.7 Comparison between Motor options	16
3.8 Differences between MRPT and ROS Noetic	17
3.9 Comparison Development Tool options	17
3.10 Comparison of Mobile Applications Language	18

List of Acronyms

MRPT	Mobile Robot Programming Toolkit
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
LiDAR	Light Detection and Ranging
IoT	Internet of Things
MQTTP	Message Queuing Telemetry Transport Protocol
HTTPS	Hypertext Transfer Protocol Secure
PID	Proportional-Integral-Derivative
SCADA	Supervisory Control and Data Acquisition
HMI	Human-Machine Interface
AI	Artificial Intelligence
PLC	Programmable Logic Controller
USD	United States Dollar
HDMI	High-Definition Multimedia Interface
USB	Universal Serial Bus
IR	Infrared
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
HTML	Hypertext Markup Language
JDK	Java Development Kit
JVM	Java Virtual Machine

Chapter 1

Introduction

1.1 Preface

Large road projects often experience delays in opening roads to cars, which can significantly impact traffic flow and economic activity. These delays are often attributed to the reliance on manual street layout planning, a process that is not only time-consuming but also prone to errors. Therefore, there is a pressing need to automate the street planning process for open roads. Implementing a robotic street planning system represents a crucial step toward improving efficiency and reducing delays in road projects. By automating the planning process, these systems can expedite the creation of road layouts with greater precision and accuracy. This automation minimizes human error and ensures that roads are ready for use in a more timely manner.

1.2 Problem statement

Manual processes for street layout planning are time consuming, prone to errors, and cause significant delays in road construction projects. These delays negatively impact traffic flow, economic activity, and road safety. This project proposes an innovative solution: an automated street painting robot to streamline the process, reduce project timelines, minimize human errors, and enhance overall safety. Addressing this critical issue will benefit commuters, businesses, and local communities by improving traffic flow, supporting economic activity, and ensuring precise and accurate road layouts.

1.3 Aims and objectives

In this project, we propose a system that aims to provide the following features:

1. Develop an automated street painting robot system. In order to achieve this aim, the following objectives should be accomplished:
 - (a) Design and integrate a user interface that enables the robot to receive precise instructions from users via Wi-Fi. Instructions include street dimensions (length and width), line position (edge or center), line type (solid or dashed), and number of lines to paint.
 - (b) Utilize distance sensors to accurately determine the robot's position on the street relative to its dimensions and the desired line position. Calculate the distance to the designated painting location and navigate the robot accordingly.

- (c) Integrate map-based localization to analyze received location data, ensuring the robot can navigate to and pinpoint the designated drawing area accurately.
- 2. Implement efficient painting functionality. In order to achieve this aim, the following objectives should be accomplished:
 - (a) Direct the robot to the specified painting location based on distance sensor calculations.
 - (b) Initiate the painting process using a controlled paint flow valve to ensure precise and clear line markings.
 - (c) Continuously monitor the paint level in the tank. Alert the user if the paint level is low, if there are obstacles ahead, or upon completion of the painting task.

1.4 Requirements

This section will present the list of the functional and non-functional requirements of the system:

1.4.1 Functional Requirements

The functional requirements of an automated mapping system are crucial to ensuring that the system can efficiently map and locate drawing areas. The following is a list of functional requirements:

1. The system should be able to find the location of the mapping .
2. The system should be able to check for obstacles during navigation.
3. The system should be able to provide feedback to users when there is no paint in the pot.
4. The system should be able to navigate the targeted area autonomously without constant user interaction, requiring only initial data input (e.g., starting location, color of the line).
5. The system should be able to control the flow of paint and draw lines clearly.

1.4.2 Non-functional Requirements

the following is a list of the non-functional requirements :

1. Reliability: The system must be reliable and stable with minimal downtime or system failures. To ensure consistent and efficient operation, the painting robot will proactively check for obstacles and paint levels before starting work, preventing potential issues that could lead to incorrect painting or malfunctions. Additionally, error notifications will be promptly sent to the app during operation, enabling rapid resolution and minimizing downtime.
2. Response time: The system must have real-time response capabilities, which are crucial for generating alerts, starting work and mapping promptly.
3. Availability: The system must be highly available to ensure it can operate at any time it is needed. This is achieved by Modular software design, which promotes high availability by enabling efficient troubleshooting and maintenance without performance impact. Robust error handling and testing ensure continuous operation.

4. Accuracy: The system must ensure highly accurate and precise line painting. This is achieved by the integration of algorithms that control the paint, robot motion, speed, and painting location. These algorithms continuously monitor the robot's position and painting parameters, guaranteeing consistent and precise line painting and enhancing the system's accuracy.

1.5 System Description

Our system primary objective is to design a mechanism for drawing lines on roads through two stages:

1. The first stage is system configuration and data analysis:

Data reception and initial analysis: The nearby user sends work area dimensions to the robot using a mobile phone app. It receives painting details such as location (end or center), color, length in meters, and number of lines.

Location analysis and positioning: The system utilizes map-based localization to analyze the received location data and accurately identify the designated drawing area. In addition, it evaluates the number and length of lines in relation to the overall street dimensions to ensure accurate planning and feasibility.

Paint and obstruction inspection: Before starting the planning process, the system checks that there is enough paint to start using a sensor and verifies that there are no obstacles in front of the robot. If there is an obstacle or less in painting, the robot will send a notification to the user for manual intervention.

2. The second stage is starting the paint process:

Starting the painting process: The painting process is controlled by a valve and relay system that regulates the paint flow. The relay activates the valve, allowing it to open or close as needed.

Send the report after completion: Once the painting is completed, the robot returns to the predetermined location and sends a report explaining what it has done, including any potential errors or problems identified during the process.

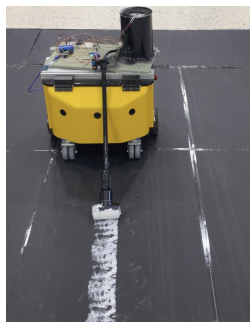


Figure 1.1: System Description

1.6 Limitations and constraints

here are some of the system limitations and constraints:

- **Environmental Conditions:** The system may not function optimally in extreme weather conditions, such as heavy rain or extreme temperatures. These conditions could affect the accuracy and reliability of the sensors.
- **Wi-Fi connection:** The system relies on a stable Wi-Fi connection for full functionality. Operation may be hindered or interrupted in areas with weak .
- **Terrain Limitations:** The robot's current design is optimized for operation on newly constructed or well-paved roads. Its functionality on uneven or unpaved terrain may be limited.

1.7 Schedule

The tasks of the system implementation and operation are distributed along the summer and the first semester summarized in Table 1.1.

Table 1.1: Project schedule in the summer and the first semester

	The summer semester			The first semester			
Week	1 - 2	3- 7	8 - 10	1 - 5	6 - 9	10 - 14	15
Selection of project Idea							
Collecting the Data and system analysis							
System Design							
System Implementation							
System testing							
system operation							
Documentation							

1.8 Report outline

This report is organized as follows: Chapter 1 provides a brief introduction to the system including the problem statement, the system requirements and system description. Chapter 2 discusses the most related keys theoretical basis and a discussion of the literature review. Chapter 3 outlines the project's design, encompassing both hardware and software aspects. It discusses design choices, the conceptual background of the software, and presents a schematic diagram. Chapter 4 explains the system implementation, testing process, and challenges faced during implementation. Chapter 5 explains the results and discussion. Finally, Chapter 6 concludes with a summary of the work and recommendations for future improvements.

Chapter 2

Background

2.1 Preface

This chapter introduces the theoretical concepts essential to our project. Following this, we'll dive into a literature review, comparing our project with what's been done before. This comparison helps us highlight the unique aspects and innovations our project brings to the table. In essence, this chapter provides the background needed to understand our project's roots and its place among previous efforts in the field.

2.2 Theoretical background

This section delves into the core theoretical principles that underpin the development of our automated street painting robot system. We will explore the fundamental concepts, algorithms, and equations that govern the system's functionality in achieving precise and efficient road marking.

2.2.1 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a fundamental concept in robotics that involves constructing a map of the environment while simultaneously estimating the robot's position within that map. SLAM combines localization and mapping to enable a robot to autonomously explore and navigate in unknown environments. This algorithm utilizes sensor data, such as Light Detection and Ranging (LiDAR) to incrementally build the map and refine the robot's position estimate [1].

2.2.2 Localization

Localization is the process of determining the precise position of a robot within its environment. It involves estimating the robot's coordinates (e.g., x , y , and z), localization is crucial for the robot to understand its position relative to the surrounding objects and to accurately navigate and interact with the environment depth sensor to obtain information about the surrounding environment. There are common localization methods, including simultaneous localization and mapping [2].

2.2.3 Obstacles Avoidance

Obstacle avoidance is essential for the autonomous operation of the robot. The laser scanner plays a crucial role in this process by emitting laser beams and measuring the time it takes for the reflected light to return after hitting an object. This information is used to calculate the distance between the scanner and the obstacle, allowing the robot to navigate around it [3].

1. Emission of laser beams: The SICK S300 consists of a rotating laser that continuously emits beams in a 170-degree arc.
2. Echo reception: When a laser beam strikes an object, it is reflected back to the scanner, and the sensor captures this reflected light.
3. Timing: The system measures the time it takes for the reflected laser light to return.
4. Distance Calculation: Based on the speed of light, the distance to an object is calculated using the Equation 2.1 :

$$Distance = Time * Speed\ of\ light / 2 \quad (2.1)$$

Based on this information, the code is written so that it takes the appropriate action to overcome obstacles, such as ordering the robot to stop or turn.

2.2.4 Navigation

Navigation refers to the process of guiding a robot from one location to another in a given environment. Figure 2.1 provides a representation of the process that involves determining the robot's path, avoiding obstacles, and reaching the desired destination. Various algorithms and techniques are used for navigation, including path planning, object detection, and recognition [4][5].

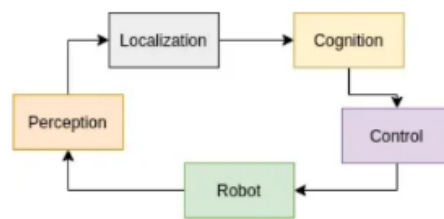


Figure 2.1: Autonomous robot navigation pipeline

2.2.5 IoT protocols

IoT protocols are sets of rules and standards that govern the way IoT devices communicate with each other and with other systems over the internet [6]. There are many IoT protocols and standards available, and different projects and use cases might require different kinds of devices and protocols. Some of the most important IoT protocols and standards include MQTT and HTTP.

2.2.6 Start Point Calculation for Line Painting

The process of determining the starting point of the first line is essential for accurate and consistent operation. The robot calculates this point based on its initial position and user-provided offsets for movement along the x and y axes. These offsets, communicated via a mobile application, define how far the robot should move from its current location to position itself at the starting point of the first line. This ensures that the first line begins precisely at the intended location, setting the foundation for subsequent lines to follow the defined pattern and maintain consistent spacing and alignment.

2.2.7 Laser Data Filtering

The laser filtering algorithm ensures safe and efficient operation by detecting obstacles in the robot's path during line drawing. The robot continuously scans its surroundings using a laser sensor, focusing on a defined angle range and a specified distance directly in front of its movement path. It filters out irrelevant data and identifies obstacles based on their proximity to the robot's trajectory. If an obstacle is detected within a critical range along the path of a line, the system eval-

uates its impact on the operation. When the obstacle poses a significant hindrance to completing the line, the robot skips the current line, logs the skipped line information, and notifies the user via the mobile application. This process maintains operational continuity while ensuring safety and minimizing delays.

2.3 Literature review

Many projects have shown interest in enhancing efficiency, precision, and effort-saving in automated street painting. This review discusses various design proposals aimed at improving accuracy and efficiency in street layout painting processes, following are discussions of such works.

2.3.1 Automatic Wood Plates Painting Machine

The project in [12] aims to design an automatic machine for painting wooden plates using a spray painting technique. While this machine enhances the traditional manual painting process by ensuring higher quality, efficiency, and safety, it has several limitations. It requires frequent manual adjustments and cannot be monitored remotely, which limits its operational flexibility. The machine can handle wooden plates up to dimensions of 220*120 cm and has a production rate of 30-40 square meters per hour. Although it is equipped with SCADA technology and an HMI touch screen for diagnosing and resolving errors, the lack of remote control and limited self-checking capabilities mean that manual intervention is often necessary. This can reduce overall efficiency and productivity.

2.3.2 Automatic wall painting machine

The system in [13] focuses on designing and developing an automatic wall painting machine to improve efficiency, safety, and the quality of wall painting tasks. While the machine is designed to save up to 85% on labor costs and increase productivity by 2-5 times, it has significant drawbacks. It requires frequent manual adjustments and does not support remote monitoring or control, which can be a major inconvenience in large-scale projects. The machine features a spray gun mounted on a mobile platform that moves both vertically and horizontally, controlled by an Arduino Mega 2560 microcontroller with stepper motors for precise movements. However, the lack of remote control capabilities and limited automation in obstacle handling make it less versatile compared to the Automated Street Painting Robot. The robot not only provides real-time updates and remote monitoring but also includes integrated AI for optimal path planning and autonomous rerouting around obstacles, ensuring continuous and efficient operation with minimal manual intervention.

2.3.3 Vertical Wall Printer

The system in [14] is an innovative solution designed to handle various wall painting tasks with efficiency and precision. This machine is particularly suitable for large-scale wall painting projects, offering the capability to paint different textures and heights of walls. It is equipped with extendable arms to reach various wall heights and uses advanced obstacle detection sensors, including IR and ultrasonic sensors, to ensure smooth operation. The system provides alerts for low paint, obstacles, task completion, and maintenance needs, with real-time updates sent to a central system. Controlled via an advanced HMI system integrated with a central control system, the Vertical Wall Printer ensures precise start/stop and monitoring operations, making it a reliable choice for extensive wall painting projects.

2.3.4 Powder Coating Machine

The Powder Coating Machine in [15] is designed to provide an efficient and uniform coating for metal pieces, especially those with complex shapes. This machine uses a powder coating technique, which involves preheating metal pieces and immersing them in a powder basin to achieve a smooth and even finish. The system includes sensors to detect obstacles in the painting path, ensuring continuous operation with minimal interruptions. It offers alerts for operational status, errors, and maintenance needs through a centralized control panel and HMI. The start/stop system is controlled via PLC and HMI, allowing for precise operational control. While the machine is fixed and uses a conveyor belt to move pieces through different stages of the painting process, it excels in providing high-quality, uniform coatings, making it ideal for industrial applications. Same as the previous project, the difference between this system and ours is the Control System, the mobile access, and some components.

Our system is designed to receive painting instructions from a mobile application to initiate the

painting process. It incorporates an alert mechanism that notifies users via the application about low paint levels and detects obstacles. Furthermore, the robot ensures safe and uninterrupted operation through continuous obstacle monitoring. Table 2.1 lists more differences between our project and the previously mentioned projects.

Table 2.1: Comparison of Related Literatures.

Feature	Automatic Wood Plates Painting Machine	Automatic Wall Painting Machine	Vertical Wall Printer	Powder Coating Machine	Our Project
System Alerts	Displays errors and status via HMI	Basic alerts for low paint and obstacles via HMI	Provides alerts for low paint, obstacles, task completion, and maintenance. Real-time updates to a central system	Alerts for operational status, errors, and maintenance needs through a centralized control panel and HMI	Alerts for low paint level, obstacles, task completion. Real-time updates sent to mobile devices and a mobile app
Start/Stop System	Controlled via HMI	Controlled via manual and automatic modes using a basic control panel	Controlled via an advanced HMI system, integrated with a central control system for start/stop and monitoring	Controlled via PLC and HMI for precise start/stop and operational control	Automatically controlled via mobile app, receives parameters for painting
Mobility	Fixed	Fixed	Fixed with extendable arms for reaching various wall heights	Fixed system with a conveyor belt for moving pieces through different stages of the painting process	Mobile with wheels for navigating streets
Versatility in Painting	Limited to wooden plates	Limited to wall painting	Capable of painting various wall textures and heights	Versatile in applying powder coating to different shapes and sizes, especially suited for parts with complex geometries	Capable of drawing multiple lines

Feature	Automatic Wood Plates Painting Machine	Automatic Wall Painting Machine	Vertical Wall Printer	Powder Coating Machine	Our Project
Obstacle Detection and Handling	Fault diagnostic system with sensors and HMI to identify and display errors	Uses IR sensors to detect obstacles, alerts user, can stop to avoid collisions. Manual intervention required for rerouting	Uses advanced obstacle detection sensors, including IR and ultrasonic, with automated rerouting and manual override options	Equipped with sensors to detect obstacles in the painting path, ensuring smooth operation and minimal interruptions	Uses distance sensors and ultrasonic sensors with navigation algorithms for autonomous obstacle avoidance
Mobile Robot Usage	Not used	Not used	Not used	Not used	Yes
Autonomous Movement	Not applicable	Requires manual adjustments for non-flat surfaces	Semi-autonomous with operator assistance	Not applicable	Fully autonomous for navigation and painting
Precision	Moderate	Moderate	High	Moderate	High

Chapter 3

System Design

3.1 Preface

This chapter provides an overview of the essential hardware and software components intended for our project. It explores various alternatives for each component, presents a conceptual description of the system, and introduces a general block diagram. Additionally, the chapter delves into system algorithms and methodologies through the use of flowcharts. Schematic diagrams depict the interactions and interfaces between components.

3.2 System components and Design options

By comparing the available components and evaluating different hardware and software choices, the aim is to identify the most suitable components that align with the project requirements and objectives.

3.2.1 hardware component and design options

3.2.1.1 Mobile Robot

We need a robot to design the project and to link the components together and communicate with each other, which will be given instructions by the user through the mobile application. Table 3.1 presents the list of options for such a robot.

Table 3.1: Comparison between mobile robot options

Requirements	MP-400 [18]	Clearpath Jackal [17]	Turtlebot 2 [16]
Cost	1,400 USD	15,000 USD	1,200 USD
Speed	1.5 m/s	2.0 m/s	0.65 m/s
Reliability	<ul style="list-style-type: none">- Exceptional odometry- Extended battery life- Stable power supply for complex sensor setups- Versatile customization for industrial applications	<ul style="list-style-type: none">- High accuracy on rough terrain- Long-lasting battery for outdoor research- Robust power supply- Customizable structure for research and development	<ul style="list-style-type: none">- Adequate for indoor navigation- Moderate battery life- Basic power supply- Customizable primarily for educational purposes
Operating System	PlatformPilot, ROS, ROS 2	PlatformPilot, ROS, ROS 2	ROS
Battery Capacity	2,200-3,000 mAh	270,000 mAh	2,200-3,000 mAh
Environment	Indoor/Outdoor	Indoor/Outdoor	Primarily Indoor

For our project, the MP-400 was selected as the mobile robot platform due to its optimal combination of speed (1.5 m/s) and robust construction. The robot's differential drive system and large wheels enable precise maneuverability across various outdoor terrains, as shown in Figure 3.1. Its compatibility with ROS, ROS 2, and PlatformPilot offers flexibility in software development and integration. The design is effective for localization, navigation, and collision avoidance, making it suitable for our system[18].





Figure 3.1: Mp-400

3.2.1.2 Processing Unit

The processing unit is a critical component that drives the system's functionality. It receives and processes sensor data, executes algorithms, and controls system components. When comparing and evaluating three choices—Raspberry Pi 4, Arduino, and MP-400 Processor—we have studied the possible options, compared them, and chosen the most suitable for our project. These are presented in Table 3.2.

Table 3.2: Comparison between Processing Unit options

Processing Unit			
Requirement	Raspberry Pi 4b [45]	Arduino [44]	MP-400 Processor [46]
Processor	Quad-core ARM Cortex A72, 1.5 GHz	Dual-core Tensilica LX6, 240 MHz	Intel Core i5
RAM	2GB, 4GB, or 8GB LPDDR4 RAM	520 KB SRAM	8GB RAM
Storage	MicroSD card slot	Up to 16 MB Flash	200+ GB SSD
Processing Power	Low	Moderate	High
Connectivity	Ethernet, Wi-Fi, Bluetooth	UART, I2C, SPI, GPIO	Ethernet, Wi-Fi, Bluetooth
I/O Ports	GPIO, I2C, I2S, SPI, UART, PWM	GPIO, I2C, I2S, SPI, UART, PWM	HDMI, USB, Ethernet
Images			Nothing




Using the Arduino and MP-400 processor together in our project provides a complementary mix of features, enhancing the robot's capabilities for precise street line drawing :

- **MP-400 Processor:** The primary processing unit is the embedded computer within the MP-400 mobile robot, which can be accessed via HDMI, USB, and Ethernet sockets. This on-board computer manages the core processing tasks of the system, including navigation, obstacle detection, and control algorithms [46].
- **Arduino:** Used to integrate real-time sensor data processing and manage the robot's operational tasks efficiently, including monitoring critical inputs and executing control commands.

3.2.1.3 Obstacle Avoidance Sensor

Obstacle avoidance is a critical aspect of mobile robotics, requiring sensors that can accurately detect and measure distances to objects in the environment. We compared three of the most efficient options: laser scanner sensors, infrared (IR) sensors, and ultrasonic sensors. The comparison is presented in Table 3.3.

Table 3.3: Comparison between Obstacle Avoidance Sensor options

Obstacle Avoidance Sensor			
Feature	Sick S300 Expert laser scanner Sensor [29]	Generic IR sensor [31]	HC-SR04 Ultrasonic sensor [30]
Technology	Laser-based	Infrared light	Ultrasonic sound waves
Depth Sensing Range	Up to 4 meters	2 cm – 30 cm	Maximum range 1-4 meters
Field of View	170 degrees	Narrow	Narrow
Environmental Impact	Not affected	Effective in various lighting conditions	May be affected by temperature and humidity
Accuracy	High	Low	Medium
Images			

For our system, a laser scanner Sick S300 Expert was selected as the primary obstacle avoidance sensor due to its superior accuracy, wide 170-degree field of view, and robustness to varying light conditions. These attributes make it highly suitable for outdoor operations. As shown in Figure 3.2, the MP-400 robot features a laser scanner strategically positioned at the front, enhancing the robot's ability to detect and avoid obstacles effectively.

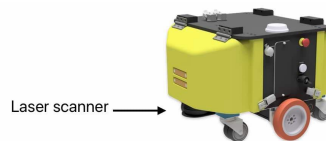


Figure 3.2: Positions of the laser scanner in robot MP-400

3.2.1.4 Paint Level Sensor

A Paint Level sensor is a crucial component for measuring the level of the painting in our system. There are two available options for the Sensor to choose from shown in Table 3.4 .

Table 3.4: Comparison between Paint Level Sensor options

Paint Level Sensor		
Requirement	IR Sensor [32] [31]	HC-SR04 Ultrasonic Sensor [30]
Technology	Infrared light	Ultrasonic sound waves
Method of Measuring Paint Level	Reflects IR light off the paint surface to measure distance.	Sends ultrasonic waves and measures the return time to calculate the distance.
Range	2 cm – 30 cm	2 cm to 4 meters
Accuracy	± 3	± 1

An ultrasonic sensor was chosen as a device that measures the distance of an object by using sound waves. It emits high-frequency sound waves and then listens for their echo to determine the distance of an object. Ultrasonic sensors are commonly used in robotics, automation, and automotive industries is shown in figure 3.3 [30].

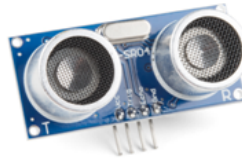




Figure 3.3: Ultrasonic Sensor

The ultrasonic sensor is used to measure the distance between the sensor and the paint level in the tank. The program calculates the distance between the paint level and the sensor and then determines the amount of painting in the tanks .

3.2.1.5 Flow Control Valve

A valve is essential for managing the flow of paint to ensure even application. It allows for precise adjustment of the paint flow rate, ensuring consistent and controlled street markings, Two options for the flow control valve are considered in Table 3.5.

Table 3.5: Comparison between Flow Control Valve options

Flow Control Valve		
Feature	Solenoid Valve	Proportional Valve
Operation Method	Electromechanical, on/off control	Electromechanical, variable flow control
Control Type	Digital (on/off)	Analog (variable)
Response Time	Fast	Moderate to Fast
Suitability for Our System	Suitable for systems requiring quick start/stop control	Suitable for systems requiring precise flow adjustments
Cost	18 USD to 30 USD	Up to 40 USD
Images		



The Solenoid Valve has been selected as the flow control valve due to its fast response time and compatibility with digital on/off control. This makes it ideal for our application where the paint flow needs to be controlled quickly and reliably.

3.2.1.6 Relay

A switch that is electrically controlled is known as a "relay." A set of working contact terminals and a set of input terminals for one or more control signals make up this device. The switch may have any number of contacts in various contact configurations, such as make contacts, break contacts, or combinations of both. Relays are used when multiple circuits need to be controlled by a single signal or when a circuit needs its own, low-power signal. In order to refresh the signal coming in from one circuit by transmitting it on another circuit, relays were first used in long-distance telegraph circuits. Early computers and telephone exchanges made extensive use of relays to carry out logical operations[48][47].

For our system, two relay options have been considered, as shown in Table 3.6.

Table 3.6: Comparison between Relay options

Relay [51][52]		
Feature	Channel Relay Module	Electromechanical Relay
Control Compatibility	Directly controlled by microprocessor	May require a driver circuit or transistor
Switching Speed	Fast	Moderate
Cost	2 USD	8 USD
Images		




The Channel Relay Module has been selected for the system due to its ease of implementation with the microprocessor.

3.2.1.7 Motor

A servo motor is a precise rotary actuator used to control the position, speed, and torque of mechanical components. In this system, the servo motor is utilized to lift and lower the paint stick during the painting process. This mechanism ensures that the paint stick is lowered to apply paint when needed and raised when the robot moves without painting, optimizing efficiency and accuracy.

To determine the most suitable option, three types of actuators have been compared, as shown in Table 3.7.

Table 3.7: Comparison between Motor options

Motor			
Feature	Standard Servo Motor	Stepper Motor	DC Motor
Control Precision	High (Specific Angle Control)	Moderate (Step-Based Control)	Low (Requires Feedback Loop)
Torque	Moderate	High	High
Response Speed	Fast	Moderate	Very Fast
Cost	5 USD	12 USD	8 USD
Images			

The Servo Motor was chosen for its ability to provide precise angular control, essential for accurately lifting and lowering the paint stick. Compared to DC motors and stepper motors, the servo motor offers simpler integration with microcontrollers, lower power consumption, and does not require additional feedback systems or complex control circuits. Its lightweight design, efficiency, and cost-effectiveness make it the ideal choice for this application, delivering reliable and precise performance while minimizing system complexity.

3.2.2 Software Components Options

There are several other options for robot operating system frameworks that can be considered for the project, including:

3.2.2.1 Robot Operating System

ROS is a flexible framework for developing robot software. It provides a wide range of libraries, tools, and drivers that can be used for building inspection tasks. ROS supports various programming languages and offers a rich ecosystem of pre-built packages that can be leveraged for perception, mapping, navigation, and other functionalities[43]. There are several other options for robot operating system frameworks that can be considered for the project, including:

1. ROS Noetic

a framework and toolset designed for the development of robotic software. It supports component-based architecture and programming in various languages [41].

2. MRPT

is a collection of C++ libraries and algorithms for mobile robotics applications. It offers localization, mapping, path planning, and other essential functionalities[42]. The differences between MRPT and ROS Noetic are shown in Table 3.7.

Table 3.8: Differences between MRPT and ROS Noetic

Robot Operating System		
Characteristic	ROS Noetic [41]	MRPT (Mobile Robot Programming Toolkit) [42]
Inter-platform operability	Multi-language support	Does not support multi-language platform
Language	Python and C++	C++
Supported Systems	Ubuntu 20.04 LTS, Debian	Cross-platform (Windows, Linux, macOS)
Tools	Tons of tools	Inbuilt tools and external packages available
Support high-end sensors and actuators	Yes	Limited
High-end capabilities	Yes	Yes
Modularity and Active community	Yes	Yes

There are two programming languages available for development within the ROS noetic framework: C++ and Python. C++ is a powerful and efficient programming language widely used in robotics, Python was chosen because it offers a wide range of libraries and tools for development and prototyping

3.2.2.2 Development Tool

The mobile app or a web app each option has different technologies and languages to ensure a simple user experience, Table 3.8 shows the differences between them.

Table 3.9: Comparison Development Tool options

Development Tool [39][40]		
Characteristic	Web app	Mobile app
Access	Requires opening a browser and navigating to the website, less convenient outdoors.	Immediate access after installation, optimized for quick access and navigation.
Offline Access	Requires a proper internet connection	Can be accessed even offline.
Loading Speed	Generally faster browsing experience.	Faster performance once installed.
Language	Frontend: JavaScript, HTML, CSS. Backend: PHP, Django.	Frontend: Flutter. Backend: Python, Java.
Database	MySQL, MongoDB.	MySQL

After careful comparison, a mobile application was chosen ,It offers optimized access and usability, making it suitable for outdoor use. The mobile app provides faster performance once installed. These advantages make the mobile app the ideal choice for our project.

3.2.2.3 Mobile Application Language

This section evaluates various mobile development languages. The comparison is summarized in Table 3.9.

Table 3.10: Comparison of Mobile Applications Language

Mobile Application Language	Flutter [35]	Kotlin [36]	Java [37]	React Native [38]
Supported Platforms	Android Jelly Bean v16, 4.1.x and iOS 8+	Android and iOS 8+	Android apps	Android 4.0.3+ versions and iOS 8+
Language Stack	Dart	JS and Native	Java (works on JVM)	JS and React.js
Performance	Removed JS bridging, enhanced app speed	Interoperable with Java and JVM	Fewer bugs	Higher performance, close to native apps
Pricing	Open-source platform	Free of cost	Paid updates for JDK	Open-source

We chose Flutter framework because it has high performance and is easy for beginners to programming and the User interface is easy to use as shown in Table 3.8 and the flutter have a feature called hot Reload while your application is running, you can make changes to the code and apply them to the running application. No recompilation is needed, and when possible, the state of your application is kept intact [35].

3.2.2.4 Python Programming Language

Python is an open-source computer programming language and a high-level dynamically typed one that is among the most popular general-purpose programming languages. It is more quickly than other programming languages built in data structures. Python is combined with dynamic typing and dynamic binding which makes it has an easy structure that enhances readability and reduces the cost of code maintenance and debugging. Python programs is easy, while languages can pick up on Python very quickly. Also, beginners of use a python language find the clean syntax. and the indentation structure is easy to learn. Furthermore[34].

3.2.2.5 Visual Studio Code

An integrated development environment. widely used, free, and open-source code editor with extensive features and support for various programming languages.

3.3 Conceptual system design

The proposed system is designed to automate the process of applying street markings efficiently and accurately. The core components of the system include the MP-400 mobile robot, the MP400 processor, an Arduino microcontroller.

- **Central Processing and Control:** The M400 processor serves as the central processing unit, receiving painting instructions and GPS coordinates from a mobile application. This allows the system to precisely control the robot's actions and ensure accurate street markings, as illustrated in the general block diagram in Figure 3.4.
- **Map-Based Localization and Path Planning:**
Map-Based Navigation: The system uses a pre-defined map for localization and navigation. The instructions sent from the mobile application are processed by the Arduino microcontroller, which works in conjunction with the robot's mapping system to determine the exact location for street markings.
Path Planning: The processor and Arduino collaborate to adjust and plan the robot's path dynamically, ensuring it follows the designated route with precision for accurate paint application.
- **Obstacle Detection and Avoidance:** A laser scanner continuously monitors the robot's surroundings to detect obstacles and measure distances to nearby objects. If an obstacle is detected within a predefined proximity, the robot's path is automatically adjusted to avoid it.
- **Paint Flow Control:** The system incorporates a solenoid valve and relay to regulate the flow of paint. This allows for precise control over paint application, ensuring consistent and clear street markings.
- **Paint Level Monitoring:** An ultrasonic sensor monitors the paint tank's content, sending notifications to the mobile application in case of low paint levels or unexpected obstructions. This ensures timely alerts and efficient operation of the system.

This comprehensive setup, is detailed in the system's conceptual diagram, as shown in Figure 3.5.

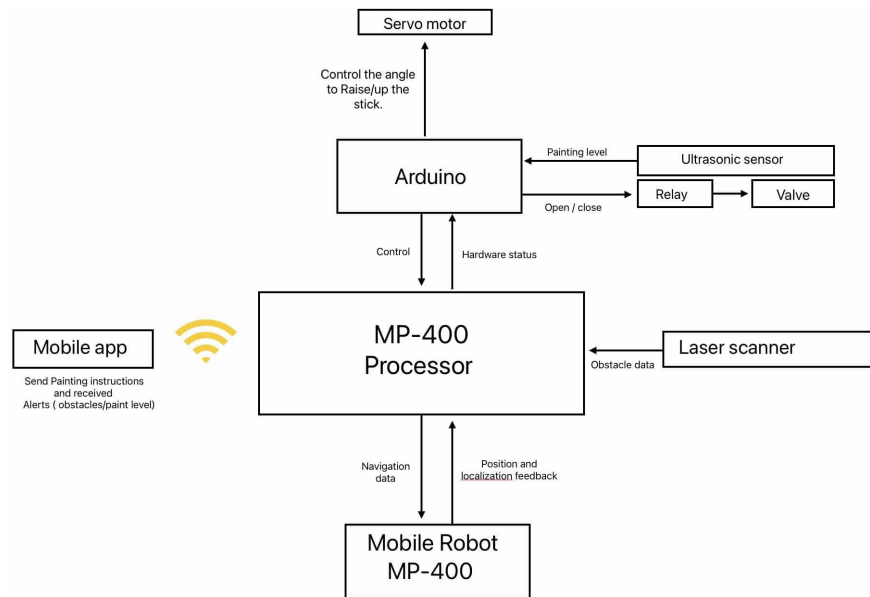


Figure 3.4: System block diagram

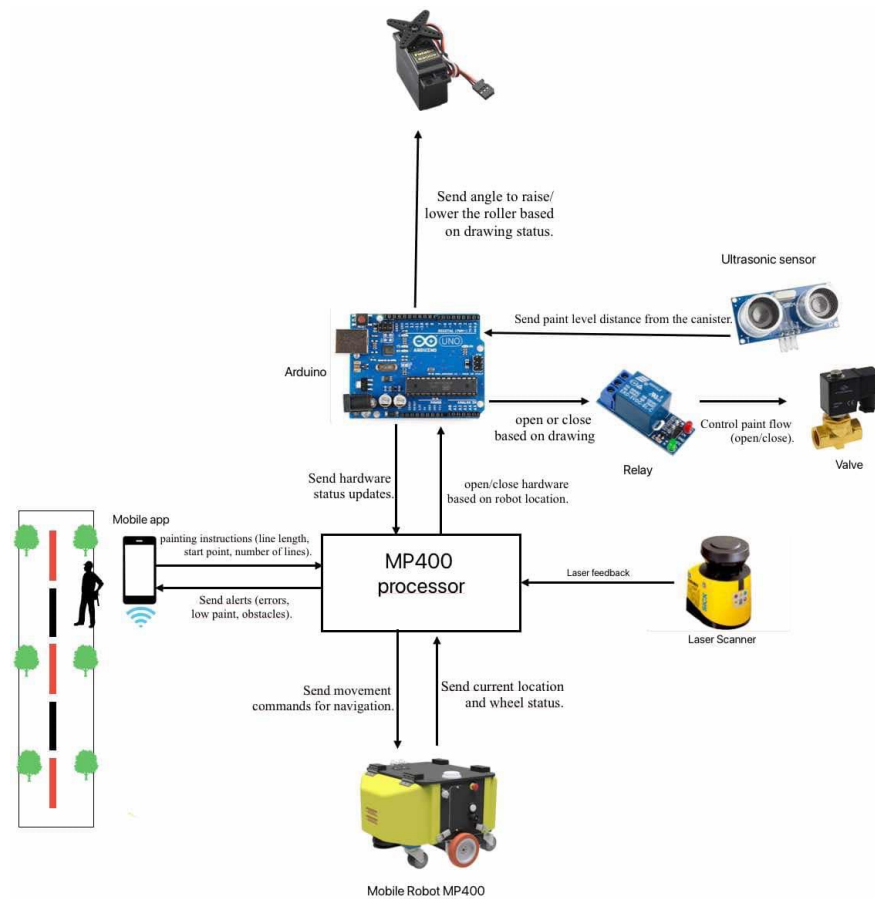


Figure 3.5: System conceptual diagram

3.4 Sequence diagrams

Figure 3.6 shows the sequence diagram of a street painting robot system.

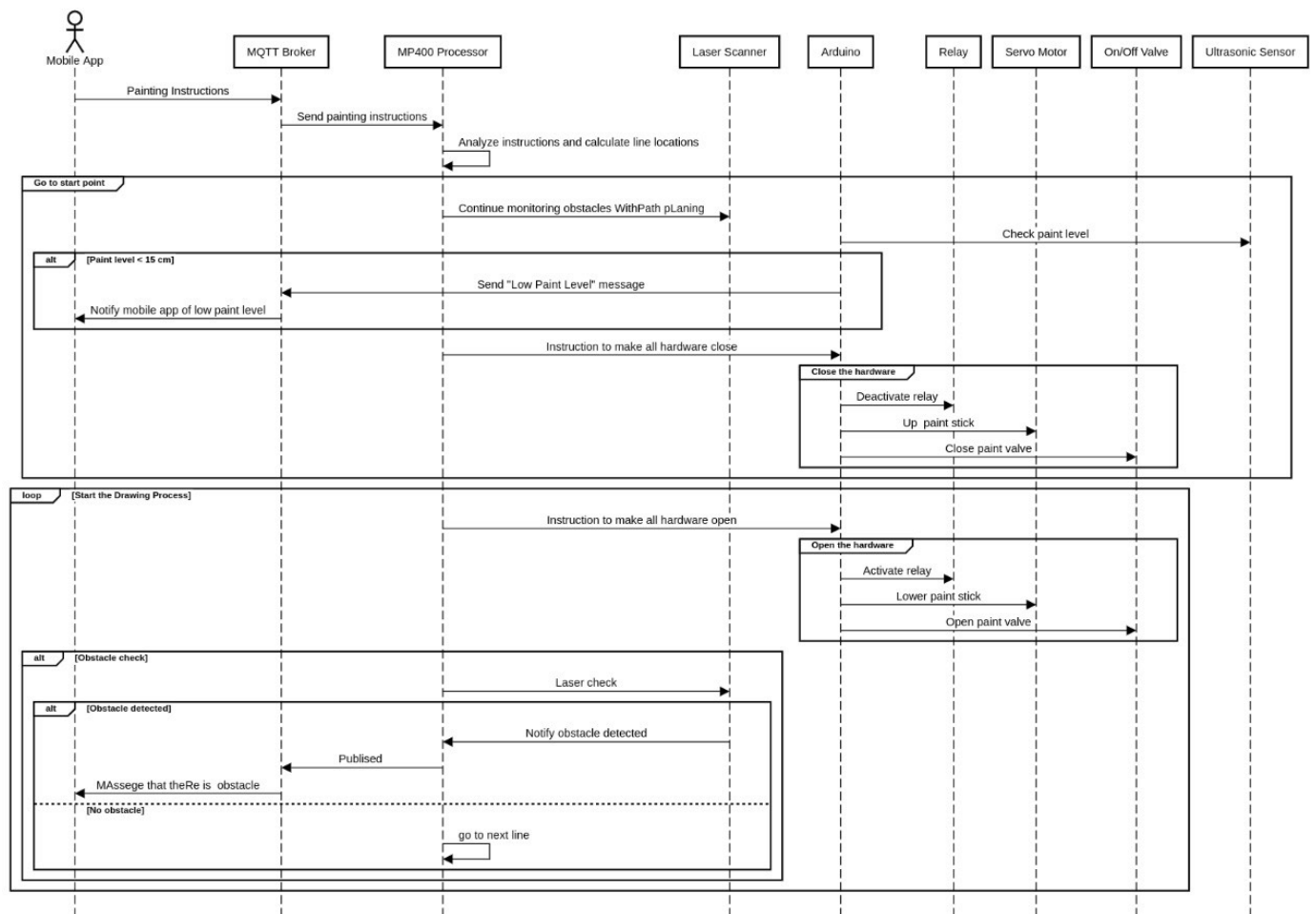


Figure 3.6: Sequence diagram of a street painting robot system

3.5 Schematic diagram

Figure 3.7 Schematic Diagram of Ultrasonic Sensor, Valve, Relay, and Servo Motor with Arduino

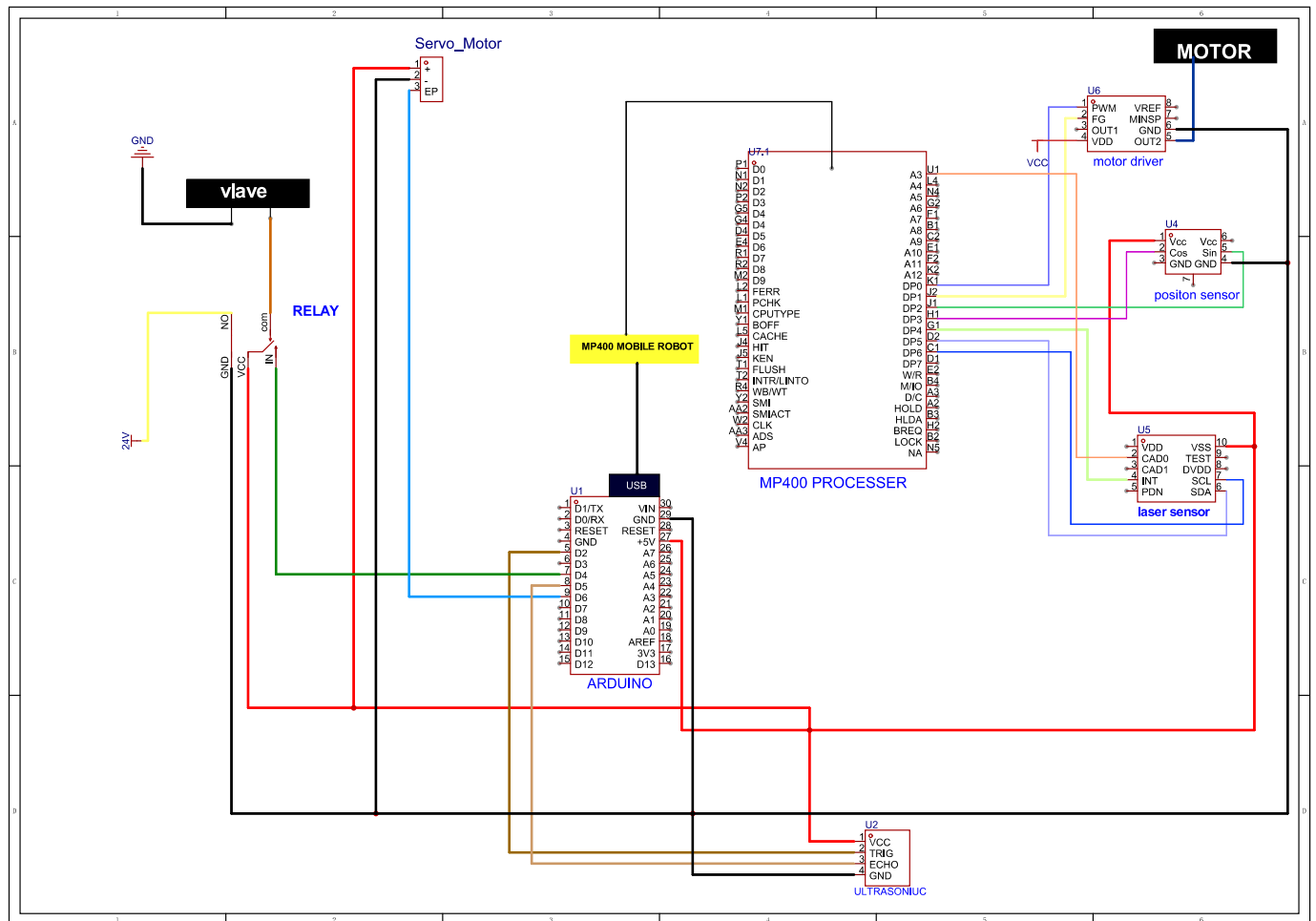


Figure 3.7: Schematic diagram for DC Motor(MP400 robot) with Arduino

The MP400 mobile robot is an autonomous system designed for navigation and task execution, with onboard processors managing movement, obstacle avoidance, and peripheral integration.

Arduino, connected to the MP400 via USB, controls specific components. The servo motor is connected to pin 6, the ultrasonic sensor to pins 2 (TRIG) and 3 (ECHO), and the relay to pin 4, powered by a 24V supply with its COM terminal connected to the valve, as shown in Figure 3.7.

MP400 processor manages navigation, motor control, and communication with the Arduino, sending commands for peripheral operations and processing feedback for synchronized tasks like paint flow and obstacle detection.

3.6 Pseudo-Code

This pseudo-code outlines the procedure for a mobile robot that is designed to navigate autonomously, avoid obstacles, and manage painting operations. The robot receives operational instructions from a mobile application, processes sensor data to check for obstacles, and continues the process until

the drawing task is completed.

Algorithm 1: Mobile Robot Painting and Obstacle Avoidance (Using AMCL Pose)

```

Begin
Set SAFE_DISTANCE = 0.5 meters
Set PAINT_LEVEL_THRESHOLD = 15 cm
WAIT UNTIL instructions are received FROM the mobile application
RECEIVE line_details AND painting_parameters
Initialize hardware state: CLOSE valve, RESET servo (Raise paint stick)
while task IS NOT completed do
  READ robot_position (x_current, y_current) FROM AMCL Pose
  READ paint_level
  if paint_level < PAINT_LEVEL_THRESHOLD then
    PUBLISH "Paint level is low" TO mobile application
    HALT painting operation UNTIL refilled
    CONTINUE TO next iteration
  READ distance_to_object FROM Laser Scanner
  if distance_to_object ≤ SAFE_DISTANCE then
    PUBLISH "Obstacle detected" TO mobile application
    CHANGE path TO avoid obstacle
    CONTINUE TO next iteration
  if robot_position IS AT target_location AND paint_level ≥
    PAINT_LEVEL_THRESHOLD AND distance_to_object > SAFE_DISTANCE then
    ACTIVATE relay
    LOWER paint stick USING servo motor
    OPEN valve TO start paint flow
    MOVE robot ALONG calculated path (start → end)
    CLOSE valve TO stop paint flow
    RAISE paint stick USING servo motor
    DEACTIVATE relay
  SEND "Task Complete" notify TO mobile application
End

```

3.7 Summary

In this chapter, we have discussed the system hardware and software components with their alternatives. The conceptual description of the system and the general flow of the system with all necessary diagrams are presented, too.

Chapter 4

System Implementation and Testing

4.1 Overview

This chapter provides an overview of the software and hardware implementation, testing and validation, issues and challenges related to the implementation.

4.2 Implementation Issues

4.2.1 Hardware Implementation

This section describes the hardware components used in the project and their respective functionalities.

4.2.1.1 MP 400 Mobile Robot

The MP 400 is the primary processing unit responsible for managing the overall operation of the robot. **It handles:**

- Navigation and mapping tasks using ROS (Robot Operating System).
- Obstacle detection through the laser scanner and path planning.

Access to the processor for monitoring and debugging is achieved using a VNC application installed on a laptop, which allows remote operation, as shown in Figure 4.1.

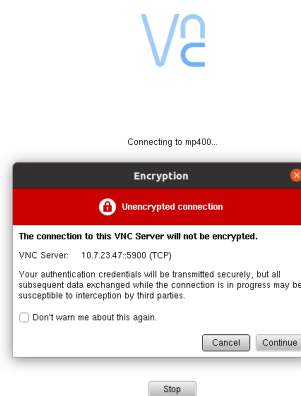


Figure 4.1: Robot processor accessed via VNC

4.2.1.2 Arduino

The Arduino Uno serves as a secondary processor and is used for direct hardware control. It is connected to the MP 400 processor via a USB cable as shown in figure 4.2, The Arduino performs the following functions:

- Opening and closing the valve.
- Controlling the servo motor angle to lift the paint roller based on the robot position.
- Controlling the relay state (ON/OFF).

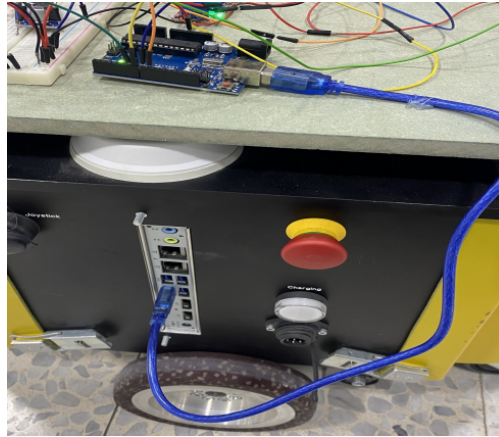


Figure 4.2: Arduino connected to MP 400 via USB

4.2.1.3 Ultrasonic Sensor

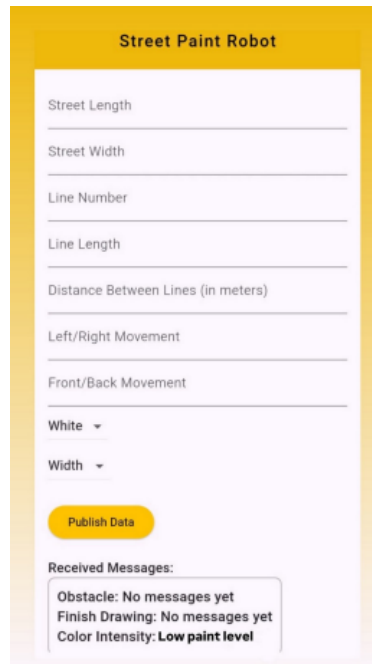
An Ultrasonic Sensor is mounted on top of the paint canister to monitor the paint level as shown in figure 4.3. The sensor continuously measures the distance from the top of the canister to the paint surface.

- If the distance drops below 15 cm, it indicates that the paint level is low.
- A warning message is sent to the mobile application via the MQTT protocol to alert the user as shown in figure 4.4.

This system ensures uninterrupted operation by alerting the user before the paint runs out.



Figure 4.3: Ultrasonic sensor position



The screenshot shows a mobile application interface titled "Street Paint Robot". It features a series of input fields for configuring the robot's painting task: "Street Length", "Street Width", "Line Number", "Line Length", "Distance Between Lines (in meters)", "Left/Right Movement", "Front/Back Movement", a color selection dropdown currently set to "White", and a width selection dropdown. A yellow "Publish Data" button is located below these fields. At the bottom, a "Received Messages:" section displays three status messages: "Obstacle: No messages yet", "Finish Drawing: No messages yet", and "Color Intensity: Low paint level".

Figure 4.4: Low-paint notification sent to mobile app

4.2.1.4 Servo Motor

The Servo Motor plays a critical role in controlling the position of the paint roller, The roller is placed at the back of the robot to avoid interference with the laser scanner at the front:

- It lifts the roller with Angle 0° :when the robot is not in the painting zone.
- It lowers the roller with Angle 120° :when the robot begins drawing lines as shown in figure 4.5 .

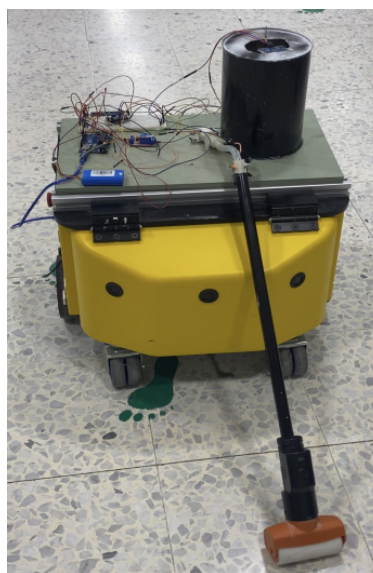


Figure 4.5: Roller position controlled by servo motor

4.2.1.5 Valve and Relay

The Valve and Relay System is critical for controlling the flow of paint from the canister to the roller during the painting process. **Valve Functionality:**

- The valve opens to allow paint flow only when the robot begins drawing the lines.
- The valve closes in all other positions.

Valve Placement:

The valve is strategically placed at the lower end of the canister to allow smooth paint flow . This ensures consistent and uninterrupted paint flow without requiring additional pressure mechanisms as shown in figure 4.6.

Relay Control:

The relay acts as an electronic switch that controls the ON/OFF states of the valve:

- ON: The relay activates the valve to start the paint flow when the robot enters the painting zone.
- OFF: The relay deactivates the valve, stopping the paint flow at the end of each line.

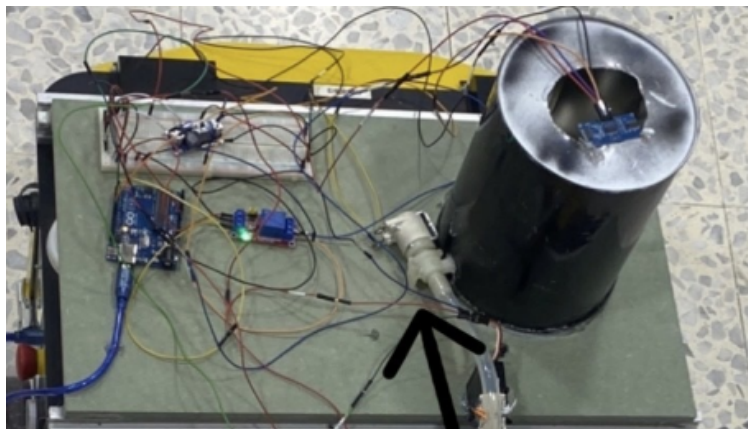


Figure 4.6: Placement of the valve

4.2.1.6 Laser Scanner

The Laser Scanner is mounted at the front of the robot to detect obstacles and provide environmental data for navigation as shown in figure 4.7. Its main functionalities are:

1. Obstacle Detection: Continuously scans the area in front of the robot and Detects obstacles within 0.5 meters as shown in figure 4.8 .
2. Mapping and Navigation: used to Generates a 2D map of the environment using SLAM algorithms, Provides essential data for path planning.

The laser scanner ensures safe and autonomous movement during line drawing operations.



Figure 4.7: Laser scanner mounted on the front of the robot

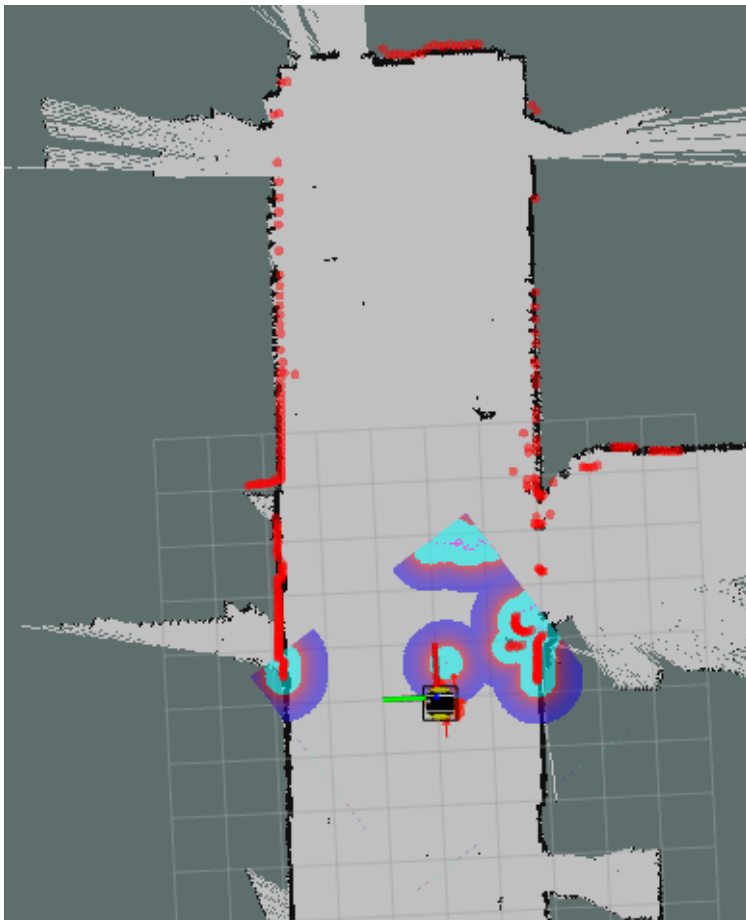


Figure 4.8: RViz visualization showing detected obstacles (red) and navigation path (blue).

4.2.2 Software implementation

This section covers the initial setup of the software environment, This step is critical to ensure the system supports the development and operation of the robotic application.

4.2.2.1 ROS-Related Implementation

This section covers all installations and configurations related to ROS Noetic and its usage for the robot operations.

4.2.2.1.1 Installing Ubuntu Mate 20.04 Operating System

We used Ubuntu Mate 20.04 to ensure stability and compatibility with ROS Noetic.

4.2.2.1.2 Installing ROS Noetic

ROS Noetic serves as the core framework for the system, providing essential tools for robot control, sensor integration, and navigation. It includes critical packages such as ros environment and catkin to enable operation with the MP 400 robot and its sensors.

1. Installation Commands:

To install the latest ROS Noetic on Ubuntu Mate 20.04, use the following commands:

```
$wget
https://raw.githubusercontent.com/qboticslabs/ros_install_noetic/master/
ros_install_noetic.sh && chmod +x ./ros_install_noetic.sh &&
./ros_install_noetic.sh
$sudo apt install ros-noetic-desktop-full
```

2. Configure the ROS environment:

```
$echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
$source ~/.bashrc
```

4.2.2.1.3 MP 400 Installation

Installing essential MP 400 packages, not included with ROS Noetic, is crucial. These packages include mp400 apps, launch files, mp400 viz, and mp400 bringup. They enhance the robot's capabilities for various tasks. Use the following commands to install the packages:

```
$git clone https://github.com/neobotix/neo_mp_400
$git clone https://github.com/neobotix/neo_simulation.git
```

4.2.2.1.4 Mapping Using SLAM

We chose the GMapping algorithm, a laser scanner-based SLAM, to build a 2D map. The steps are:

1. Launch the basic robot drivers and hardware interface:

```
$roslaunch neo_mp_400 navigation_basic_amcl.launch
```

2. Start the GMapping algorithm for SLAM:

```
$roslaunch neo_mp_400 gmapping_basic.launch
```

3. Launch RViz to visualize the map and robot position:

```
$roslaunch neo_mp_400 rviz_navigation.launch
```

4. Control robot movement using teleoperation:

```
$roslaunch teleop_twist_keyboard teleop_twist_keyboard.py
```

5. Save the generated map as shown in figure 4.9:

```
roslaunch map_server map_saver -f /path/to/save/the/map
```

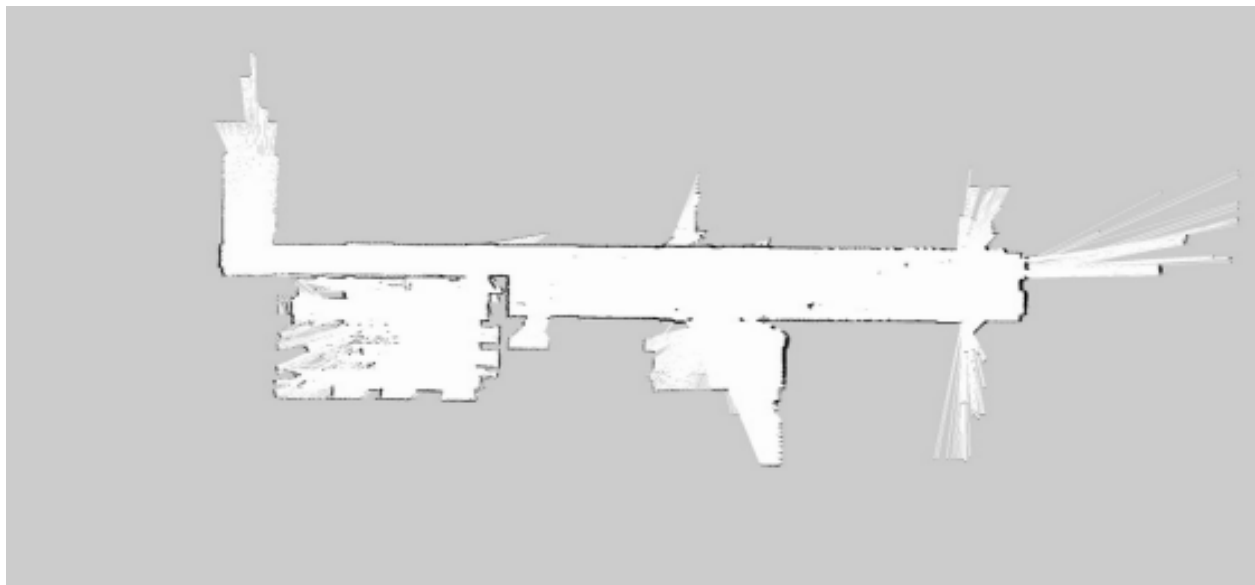


Figure 4.9: Generated 2D map

4.2.2.1.5 Navigation

The navigation module enables the robot to autonomously plan and execute movements using the generated map. The system dynamically adjusts the robot path to reach predefined goals while avoiding obstacles using the command:

```
$roslaunch neo_mp_400 navigation_basic_amcl.launch
```

The robot navigation area and movement can be visualized using the RViz tool as shown in in figure 4.8 , where:

- Blue lines represent the planned navigation path.
- Red zones indicate obstacles detected by the laser scanner.

4.2.2.2 Robot Painting System Testing

This section provides a detailed explanation of the software implementation related to the system. Key code snippets are included for clarity, covering robot movement, position detection, and line handling.

4.2.2.2.1 Line Start Point Calculation

To determine the robot's starting point for drawing lines, the system calculates the position based on the robot's current pose and user-specified parameters like front-back and left-right movements.

```
def calculate_start_point():
    global x_current, y_current
    start_x = x_current + LEFT_RIGHT_MOVEMENT
    start_y = y_current + FRONT_BACK_MOVEMENT
    rospy.loginfo(f"Calculated Start Point: ({start_x:.2f}, {start_y:.2f})")
    return start_x, start_y
```

4.2.2.2.2 Data Processing for Line Start and End Points

The robot accepts user input (such as line length, number of lines, and spacing) and calculates the starting and end points for each line sequentially.

```
def calculate_lines(start_x, start_y, line_length, line_spacing, num_lines, direction):
    lines = []
    current_x, current_y = start_x, start_y

    for i in range(1, num_lines + 1):
        if direction == 'Length': # Horizontal movement
            if i == 1:
                start = (current_x, current_y)
                end = (current_x + line_length, current_y) # Move along X-axis
            elif i % 2 == 0:
                start = (lines[-1][1][0], lines[-1][1][1] + line_spacing) # Move up in Y-axis
                end = (start[0] - line_length, start[1]) # Move back along X-axis
            else:
                start = (lines[-1][1][0], lines[-1][1][1] + line_spacing) # Move up in Y-axis
                end = (start[0] + line_length, start[1]) # Move forward along X-axis
        elif direction == 'Width': # Vertical movement
            if i == 1:
                start = (current_x, current_y)
                end = (current_x, current_y + line_length) # Move along Y-axis
            elif i % 2 == 0:
                start = (lines[-1][1][0] + line_spacing, lines[-1][1][1]) # Move right in X-axis
                end = (start[0], start[1] - line_length) # Move back along Y-axis
            else:
                start = (lines[-1][1][0] + line_spacing, lines[-1][1][1]) # Move right in X-axis
                end = (start[0], start[1] + line_length) # Move forward along Y-axis
        else:
            rospy.logwarn("Invalid direction. Stopping calculation.")
            break
        lines.append((start, end))

    return lines
```

4.2.2.2.3 Hardware Control Based on Position

The hardware control (valve, relay, servo motor) is managed based on the robot position and goals. The valve opens when the robot reaches a line start and closes when the line is complete.

```
def update_hardware_state(open_valve):
    global hardware_open, servo_position

    if open_valve and not hardware_open:
        hardware_run_pub.publish("start")
        rospy.loginfo("Valve and servo: OPEN")
        hardware_open = True
        servo_position = "OPEN"
    elif not open_valve and hardware_open:
        hardware_stop_pub.publish("stop")
        rospy.loginfo("Valve and servo: CLOSED")
        hardware_open = False
        servo_position = "CLOSED"
```

4.2.2.2.4 Laser filter data

The robot processes laser scan data to detect obstacles within a 270-degree field of view. To determine if an obstacle is present within a specific distance and directly in front of the robot, it is necessary to apply a filtering mechanism to the laser data. This filtering ensures that only the relevant data within the robot's forward direction is analyzed, improving the efficiency of obstacle detection and navigation.

```
def scan_callback(scan_msg):
    global robot_state, previous_state, last_no_obstacle_time, last_obstacle_time

    # Extract angle and range data
    angle_min = scan_msg.angle_min
    angle_max = scan_msg.angle_max
    angle_increment = scan_msg.angle_increment
    ranges = scan_msg.ranges

    # Validate ranges
    if not ranges or all(math.isinf(r) for r in ranges):
        return

    # Define front angles (clamped to valid ranges)
    front_angle_min = max(angle_min, -0.09072665) # -6.14 degrees in radians
    front_angle_max = min(angle_max, 0.09072665) # +6.14 degrees in radians

    index_min = max(0, int((front_angle_min - angle_min) / angle_increment))
    index_max = min(len(ranges), int((front_angle_max - angle_min) / angle_increment))

    # Filter valid front range values
    front_ranges = [r for r in ranges[index_min:index_max] if not math.isinf(r) and not math.isnan(r)]
    if not front_ranges:
        return
```

Figure 4.10: Laser data filtering code implementation.

4.2.3 Mobile Application Implementation

The mobile application serves as a bridge between the robot and the user, enabling real-time communication, monitoring, and control. Integration is achieved using the MQTT protocol, which ensures efficient and lightweight data exchange.

4.2.3.1 MQTT Protocol

The MQTT protocol is used to facilitate communication between the robot and the mobile application. It follows a publish/subscribe model where:

- The robot publishes real-time status updates (e.g., paint level, obstacles) to designated topics.
- The mobile application subscribes to these topics to receive updates instantly.
- The user can also send control commands (e.g., start/stop painting) to the robot by publishing messages from the app.

A Python script handles the MQTT communication by subscribing to and publishing messages to appropriate topics from the ROS nodes.

4.2.3.2 User Interface Design

A simple and user-friendly mobile application was developed to allow real-time interaction with the robot. The user interface is designed for clarity and ease of use, as shown in Figure 4.11:

- Displays the data of the drawing process being sent to the robot.
- Indicates the button used to publish data to the MQTT broker.
- Displays messages received from the robot, such as: Low paint level alerts, Obstacle detection notifications and Completion of the line drawing process.
- Provides a button to stop the robot operation.

Figure 4.11: Mobile app user interface showing data exchange with the robot

4.3 Challenges

- An issue occurred due to incorrect installation of the robot's packages, which was not immediately identified. The problem was resolved by reinstalling all the necessary packages.
- The robot displayed angular movement during operation. This was addressed by creating a new map and making adjustments to the angle settings.
- Transferring data between the robot processor and the mobile application posed a challenge. This issue was resolved by implementing the MQTT protocol, which facilitated seamless communication.
- The limited testing space for the robot's movement presented difficulties, as the area was insufficient to simulate real-world scenarios effectively.
- Testing line drawing in small and confined areas, particularly within the university environment, posed challenges for precision and accuracy.

4.4 System Validation and Testing

This section focuses on testing individual hardware components and verifying their functionality within the integrated system to ensure proper operation.

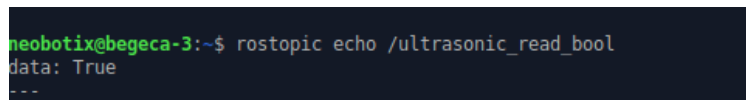
4.4.1 System Testing

4.4.1.1 Hardware Component Testing

To ensure each hardware component functions correctly, the following tests were performed:

4.4.1.1.1 Ultrasonic Sensor Testing

If the paint level is below 15 cm, a warning message is sent to the mobile application as shown in figure 4.12.



```
neobotix@begeca-3:~$ rostopic echo /ultrasonic_read_bool
data: True
---
```

Figure 4.12: Testing the Ultrasonic Sensor

4.4.1.1.2 Laser Scanner Testing

Verify the /scan topic:
rostopic list

4.4.1.1.3 Valve Testing

- Test the valve functionality by manually activating it and observing the paint flow:
rostopic echo /valve state

4.4.1.1.4 Relay Testing

Verify relay behavior by observing its ON/OFF state using the command :
rostopic echo /relay control

4.4.1.2 System Testing

This section explains how to comprehensively test the system's overall functionality, ensuring autonomous operation, mobile app communication, correct line drawing, and obstacle handle.

4.4.1.2.1 Mobile Application Connectivity Testing

Execute the MQTT script to send data using the mobile application. Confirm that the data is successfully published and saved to a file. The published data and its corresponding results are displayed in Figure 4.13.

```
Data saved to mqtt_data.py
Message received on topic ppu/pro-->draw/line: street length:1,street width:1,line number:4,line length:3,line color:White,distance between lines:1,left right movement:1,front back movement:1,line direction:Length
Parsed data: {'street_length': '1', 'street_width': '1', 'line_number': '4', 'line_length': '3', 'line_color': 'White', 'distance between lines': '1', 'left_right_movement': '1', 'front_back_movement': '1', 'line_direction': 'Length'}
Data saved to mqtt_data.py
```

Figure 4.13: MQTT Data Published in the Mobile Application

4.4.1.2.2 Line Drawing Accuracy Testing

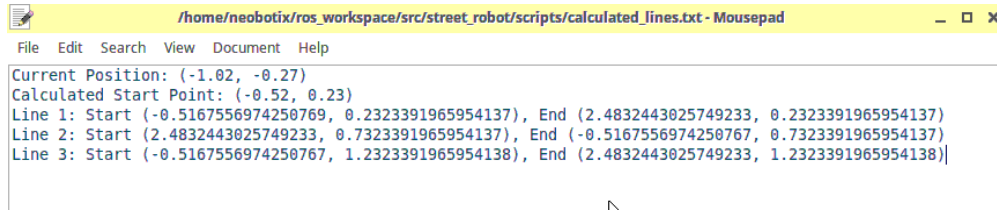
To verify that the robot draws lines accurately and in the correct position:

1. Compare the intended movement (target positions) with the actual movement by saving the target data to a file and comparing it with the real robot movement as shown in figure 4.15.
2. Steps to verify the drawing:
 - Use the robot navigation goal and check its actual movement using pose data.
 - Compare the target coordinates (start and end points of the line) with the actual coordinates the robot reached as shown in Figure 4.14.
 - Calculate the error in position equation 4.1:

$$Error(\%) = \frac{Target\ Position - Actual\ Position}{Target\ Position} \times 100 \quad (4.1)$$

```
[INFO] [1736934189.122162]: Matched: Calculated: (3.062087332110173, 1.1101173716061457), Actual: (3.0342683972824915, 0.8361831381950104)
[INFO] [1736934189.123616]: Matched start point for Line 2
[INFO] [1736934189.258101]: Goal reached.
[INFO] [1736934189.352184]: --- Successfully reached Line 2 Start ---
[INFO] [1736934189.353422]: Reached Start of Line 2. Opening hardware.
[INFO] [1736934189.554885]: --- Moving to Line 2 End: Target = (x: 0.06208733211017306, y: 1.1101173716061457) ---
[INFO] [1736934189.556410]: Sending goal to: x = 0.06, y = 1.11, orientation = 0
[INFO] [1736934118.438294]: Matched: Calculated: (0.06208733211017306, 1.1101173716061457), Actual: (0.34798841365050054, 1.0770959821864221)
```

Figure 4.14: Compare the target coordinates



```

/home/neobotix/ros_workspace/src/street_robot/scripts/calculated_lines.txt - Mousepad
File Edit Search View Document Help
Current Position: (-1.02, -0.27)
Calculated Start Point: (-0.52, 0.23)
Line 1: Start (-0.5167556974250769, 0.2323391965954137), End (2.4832443025749233, 0.2323391965954137)
Line 2: Start (2.4832443025749233, 0.7323391965954137), End (-0.5167556974250767, 0.7323391965954137)
Line 3: Start (-0.5167556974250767, 1.2323391965954138), End (2.4832443025749233, 1.2323391965954138)

```

Figure 4.15: Comparison file for validating robot position against saved data.

4.4.2 Obstacle in Line Drawing Path

- If an obstacle is detected in the line drawing path, the robot immediately skips the line to avoid delays since the entire painting process is time-sensitive and takes only a few minutes.
- A notification is sent to the mobile application, specifying that the line has been skipped along with the line number, ensuring the user is informed promptly.
- his process ensures efficient operation without compromising the overall painting workflow, as shown in Figure 4.16.

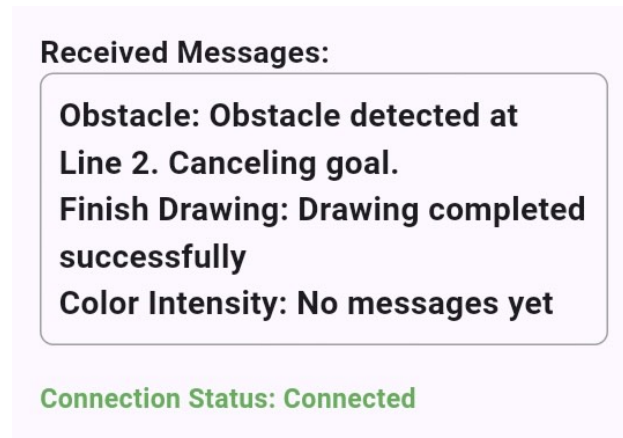


Figure 4.16: notification the user obstacle in Line Drawing Path

4.4.3 System Validation

After connecting and testing each component, the robot is fully assembled, and the Python code is run to verify system functionality. The robot correctly goes to the starting point, draws lines, identifies its location on the map, and interacts with the mobile app in real time.

4.5 Summary

In this chapter, we reviewed the hardware and software implementation. Each component was fully explained, followed by a discussion of the validation and testing procedures, including unit and integration testing, to ensure the system's functionality. Lastly, we addressed the issues and challenges faced during the implementation phase.

Chapter 5

Discussion of Results

5.1 Preface

This chapter examines the performance of the Street Painting Robot, emphasizing its operational efficiency, obstacle-handling capabilities, and the role of user-configurable parameters in achieving project objectives. Key results are discussed to showcase its functionality and reliability.

5.2 Discussion of Results

The Street Painting Robot showcases the integration of robotics and IoT for automating street painting. The following key aspects highlight the system's functionality and effectiveness:

5.2.1 Streamlined Operation

The robot automates the line-drawing process with minimal human intervention, following pre-defined paths and executing tasks with precision. This ensures consistent results and simplifies operations for the user.

5.2.2 Handling of Obstacles

During operation, the robot detects obstacles in its path and takes immediate action by skipping obstructed lines. This approach maintains the overall workflow, as delays are minimized. Notifications detailing skipped lines and their respective numbers are sent to the mobile application, keeping the user informed.

5.2.3 Impact of Line Parameters

Testing revealed that the performance of the system is influenced by the length of lines and spacing between them:

- **Longer Lines:** Drawing longer lines (e.g., 5 meters) results in smoother execution compared to shorter lines (e.g., 2 meters). This is attributed to fewer adjustments during movement.
- **Wider Spacing:** Increased spacing between lines reduces overlap, improving the clarity and uniformity of the painted results.

5.2.4 Integration with the Mobile Application

The mobile application plays a critical role in managing the system. It allows the user to configure painting parameters, monitor progress, and receive status updates, ensuring full control over the operation.

5.2.5 System Resilience

The robot effectively manages challenges such as low paint levels and obstacles without significant interruptions. Automated responses and real-time notifications ensure smooth and continuous operation, enhancing the overall user experience.

5.3 Summary

The Street Painting Robot demonstrates efficient and reliable performance in automating road marking. It effectively integrates robotics and IoT to streamline operations, handle obstacles, and adapt to user-defined parameters, ensuring precision and continuity in its tasks.

Chapter 6

Conclusion and future work

6.1 Preface

The chapter introduces a summary of the project and future work.

6.2 Conclusion

The Street Painting Robot automates the process of road marking through a carefully integrated system of hardware and software. The MP-400 mobile robot, equipped with advanced sensors, an Arduino microcontroller, and ultrasonic sensors, works in tandem with the Robot Operating System (ROS) to deliver precise and autonomous street painting capabilities.

The process begins with a mobile application, where users configure painting parameters such as street dimensions, line lengths, colors, and spacing. These details are transmitted to the robot via MQTT communication, enabling real-time updates and centralized control. During operation, the robot receives instructions, calculates line positions, and begins its task. If the paint level is low or an obstacle is detected, the system sends notifications to the mobile app, ensuring the operator is always informed.

The robot navigates using map-based localization, calculating paths for each line and dynamically adjusting based on real-time feedback from sensors. It ensures accuracy by managing paint flow with servo motors and valves, while ultrasonic sensors monitor the paint level in the canister. This approach guarantees precise and continuous operation without human intervention.

As a conclusion , the Street Painting Robot demonstrates the effective integration of robotics and IOT to automate street painting. By streamlining the line-drawing process, reducing errors, and enabling real-time communication through the mobile app, the system offers a reliable, efficient, and scalable solution for Road infrastructure projects .

6.3 Future work

1. **Advanced Shape and Curve Painting:**

Enhancing the robot's capabilities to include the painting of complex patterns such as curves and geometric shapes (e.g., rectangles and circles), thereby broadening its functional scope.

2. **Simultaneous Multi-Line Painting:**

Developing the ability for the robot to paint multiple lines concurrently, significantly improving operational efficiency and reducing the overall time required for large-scale street marking projects.

3. **AI-Powered Image Analysis for Autonomous Drawing:**

Incorporating artificial intelligence to enable the robot to analyze input images and autonomously replicate them as street markings. This feature would allow the robot to translate intricate designs or symbols directly onto surfaces with precision.

References

- [1] MathWorks, "What Is SLAM (Simultaneous Localization and Mapping) – MATLAB Simulink," [Online]. Available: <https://www.mathworks.com>. Accessed: Jul. 7, 2024.

- [2] IEEE Xplore, "Localization and Navigation for Indoor Mobile Robot Based on ROS," IEEE Transactions on Robotics, vol. 40, no. 2, pp. 256–268, Aug. 2024. [Online]. Available: <https://ieeexplore.ieee.org>.

- [3] SICK, "S300 Safety Laser Scanners," SICK Technical Reports, Sep. 2024. [Online]. Available: <https://www.sick.com>. Accessed: Oct. 5, 2024.

- [4] ResearchGate, "Robot navigation with obstacle avoidance in unknown environment," International Journal of Advanced Robotics, vol. 38, no. 3, pp. 189–200, Nov. 2024. [Online]. Available: <https://www.researchgate.net>.

- [5] Control.com, "An Introduction to Simultaneous Localization and Mapping (SLAM) for Robots," Automation Journal, Oct. 2024. [Online]. Available: <https://control.com>.

- [6] Nabto, "A complete guide to IoT protocols standards in 2023," IoT Journal, vol. 21, no. 4, pp. 345–359, Dec. 2024. [Online]. Available: <https://www.nabto.com>.

- [7] Coursera, "What is the internet of things (IoT)? with examples," IoT Online Learning, Jul. 2024. [Online]. Available: <https://www.coursera.org/articles/internet-of-things>. Accessed: Aug. 12, 2024.

- [8] P. Gupta, "Sensor Device," TechTarget Journal, vol. 10, no. 5, pp. 22–30, Oct. 2024.

References

[Online]. Available: <https://www.techtarget.com/whatis/definition/sensor>.

[9] SparkFun, "Ultrasonic Distance Sensor HC-SR04," SparkFun Electronics, Nov. 2024. [Online]. Available: <https://www.sparkfun.com/products/15569>.

[10] MaxBotix, "How HC-SR04 Ultrasonic Sensor Works Interface It With Arduino," Robotics World, vol. 12, no. 8, pp. 112–120, Dec. 2024. [Online]. Available: <https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work>.

[11] Omega Engineering, "What is a PID Controller?" Omega Technical Reports, vol. 15, no. 3, pp. 45–53, Jul. 2024. [Online]. Available: <https://www.omega.com>.

[12] PPU Documentation, "Automatic Wood Plates Painting Machine documentation," PPU Research Papers, Aug. 2024. [Online]. Available: <https://scholar.ppu.edu>.

[13] PPU Documentation, "Automatic Wall Painting Machine documentation," PPU Research Papers, Sep. 2024. [Online]. Available: <https://scholar.ppu.edu>.

[14] PPU Documentation, "Vertical Wall Printer," Robotics Innovation Reports, Nov. 2024. [Online]. Available: <https://scholar.ppu.edu>.

[15] L. and Y. Yousef, "Powder Coating Machine," Robotics Engineering Studies, Dec. 2024. [Online]. Available: <https://scholar.ppu.edu>.

[16] Clearpath Robotics, "TurtleBot 2 - Open source personal research robot," Robotics Research Catalog, vol. 20, no. 5, pp. 115–123, Jul. 2024. [Online]. Available: <https://clearpathrobotics.com>.

[17] Clearpath Robotics, "Jackal UGV - Small Weatherproof Robot - Clearpath," Robotics Research Catalog, Aug. 2024. [Online]. Available: <https://clearpathrobotics.com>.

[18] Neobotix, "Mobile Robot MP-500," Robotics Journal, vol. 34, no. 6, pp. 78–85, Oct. 2024. [Online]. Available: <https://www.neobotix-robots.com/products/mobile-robots/mobile-robot-mp-500>.

References

[19] TurtleBot, "A 'Getting started' guide for developers interested in robotics," Learn TurtleBot and ROS, Jul. 2024. [Online]. Available: <https://learn.turtlebot.com>.

[20] ROS Components, "Online store for robotic products supported by ROS," Robotics Components Reviews, vol. 18, no. 7, pp. 90–97, Dec. 2024. [Online]. Available: <https://www.roscomponents.com>.

[21] Raspberry Pi, "Raspberry Pi 4 Model B," Raspberry Pi Technical Datasheets, vol. 12, no. 3, pp. 23–29, Jul. 2024. [Online]. Available: <https://www.raspberrypi.com>.

[22] TurtleBot, "A 'Getting started' guide for developers interested in robotics," Learn TurtleBot and ROS, Robotics Guides, Aug. 2024. [Online]. Available: <https://learn.turtlebot.com>.

[23] ResearchGate, "Features of Kinect Sensor V2," ResearchGate Technical Reports, Sep. 2024. [Online]. Available: <https://www.researchgate.net>.

[24] Neobotix, "Laser Scanner (Page 36) in MP500 Manual," Robotics Technical Documentation, vol. 21, no. 5, pp. 102–110, Oct. 2024. [Online]. Available: <https://www.neobotix-robots.com>.

[25] Neobotix, "Mechanical Properties and Sensor Position for MP500," Robotics Systems Manuals, vol. 22, no. 6, pp. 45–52, Nov. 2024. [Online]. Available: <https://www.neobotix-docs.de>.

[26] IJSTE, "Indoor SLAM using Kinect Sensor," International Journal of Science Technology and Engineering, vol. 30, no. 4, pp. 123–130, Dec. 2024. [Online]. Available: <https://www.ijste.org>.

[29] Generation Robots, "LIDAR URG-04LX Sensor," Robotics Sensor Reviews, vol. 19, no. 7, pp. 87–94, Sep. 2024. [Online]. Available: <https://www.generationrobots.com>.

References

[30] Murata, "Basic knowledge about ultrasonic sensors: Features of each type of ultrasonic sensor," Murata Technical Articles, vol. 15, no. 3, pp. 54–61, Jul. 2024. [Online]. Available: <https://www.murata.com>.

[31] Tech Zero, "IR Sensor: What is the IR Sensor or Infrared Obstacle Sensor Module," Tech Zero Guides, vol. 14, no. 5, pp. 98–105, Aug. 2024. [Online]. Available: <https://techzeero.com/sensors-modules/ir-sensor/>.

[32] Tech Zero, "IR Sensor," Tech Zero Guides, vol. 14, no. 6, pp. 120–125, Sep. 2024. [Online]. Available: <https://techzeero.com/sensors-modules/ir-sensor/>.

[34] "Welcome to Python.org," Python.org Technical Resources, vol. 20, no. 7, pp. 145–152, Jul. 2024. [Online]. Available: <https://www.python.org/>.

[35] Medium, "What is Flutter? A Complete Guide to Google's Framework," Medium Development Articles, vol. 18, no. 4, pp. 56–62, Aug. 2024. [Online]. Available: <https://medium.com/@growsolu-is-flutter-a-complete-guide-to-googles-framework>.

[36] Android Developers, "Kotlin," Android Developer Documentation, vol. 22, no. 8, pp. 220–225, Sep. 2024. [Online]. Available: <https://developer.android.com>.

[37] Baeldung, "An Overview of the JVM Languages," Baeldung Technical Articles, vol. 15, no. 9, pp. 34–40, Oct. 2024. [Online]. Available: <https://www.baeldung.com>.

[38] Wikipedia, "React (JavaScript library)," Wikipedia Technology Resources, vol. 13, no. 7, pp. 200–206, Sep. 2024. [Online]. Available: <https://en.wikipedia.org>.

[39] CodeInstitute, "What is the Difference Between Web App and Mobile App?" CodeInstitute Tutorials, vol. 25, no. 3, pp. 78–85, Jul. 2024. [Online]. Available: <https://codeinstitute.net>.

[40] TechTarget, "Mobile website vs. app: What's the difference?" TechTarget Insights, vol. 27, no. 5, pp. 101–110, Aug. 2024. [Online]. Available: <https://www.techtarget.com>.

References

- [41] "ROS Noetic Ninjemys," wiki.ros.org, [Online]. Available: <https://wiki.ros.org/noetic>.
- [42] "MRPT," docs.mrpt.org, [Online]. Available: <https://docs.mrpt.org/reference/latest/index.html>.
- [43] "ROS," ros.org, [Online]. Available: <https://ros.org>.
- [44] R. Teja, "Getting Started with ESP32: Introduction to ESP32," Electronics Hub, vol. 18, no. 3, pp. 67–72, Aug. 2024. [Online]. Available: <https://www.electronicshub.org>.
- [45] Raspberry Pi, "Raspberry Pi 4 Model B Specifications," Raspberry Pi, vol. 12, no. 7, pp. 132–137, Jul. 2024. [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>.
- [46] Neobotix GmbH, "Mobile Robot MP-500," Neobotix Technical Resources, vol. 20, no. 5, pp. 90–95, Sep. 2024. [Online]. Available: <https://www.neobotix-robots.com/products/mobile-robots/mobile-robot-mp-400>.
- [47] Kunkune, "Relay Modules: Understanding the Basics," Kunkune Electronics, vol. 15, no. 4, pp. 88–92, Jul. 2024. [Online]. Available: <https://kunkune.co.uk/blog/understanding-the-basics-what-is-a-relay-module>.
- [48] JYCircuitBoard, "The Working of Relay Module Circuits: From Schematic to Functionality," JYCircuitBoard Technical Blog, vol. 14, no. 6, pp. 45–51, Aug. 2024. [Online]. Available: <https://www.jycircuitboard.com>.
- [49] DigiKey, "CIT Relay and Switch," DigiKey Product Insights, vol. 16, no. 9, pp. 101–107, Oct. 2024. [Online]. Available: <https://www.digikey.com/en/products/detail/cit-relay-and-switch>.
- [50] Omega Engineering, "Technical Principles of Valves," Omega Engineering Articles, vol. 19, no. 8, pp. 140–145, Sep. 2024. [Online]. Available: <https://www.omega.com/en-us/resources/valves-technical-principles>.

References

[51] Components101, "5V Single Channel Relay Module Pinout, Features, Applications, Working Datasheet," Components101, vol. 22, no. 3, pp. 70–75, Aug. 2024. [Online]. Available: <https://components101.com>.

[52] Pickering Test, "Electromechanical Relay (EMR) Characteristics," Pickering Test Knowledge Base, vol. 18, no. 4, pp. 102–109, Sep. 2024. [Online]. Available: <https://www.pickeringtest.com/en-in/kb/hardware-topics/relay-characteristics/emr-relays>.

[53] IQS Directory, "Solenoid Valve Overview," IQS Technical Guides, vol. 17, no. 5, pp. 66–72, Jul. 2024. [Online]. Available: <https://www.iqsdirectory.com/articles/solenoid-valve.html>.

[54] Norgren, "Why Use a Proportional Valve?" Norgren Blog, vol. 21, no. 6, pp. 120–125, Sep. 2024. [Online]. Available: <https://www.norgren.com/en/support/blog/why-use-a-proportional-valve>.