REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre: IA_StartUp14/2023

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : Intelligence Artificielle (IA)

Smart floor cleaner

Par :
Sara BENABDALLAH
Maroua MOUADA

Soutenu le 03/07/2023 devant le jury composé de :

Toufik BENDAHMANE	MAB	Président
Meftah ZOUAI	MCB	Rapporteur
Asma AMMARI	MCB	Rapporteur
Ammar HAMIDA	MAB	Examinateur

Année universitaire 2022-2023

Acknowledgment

Acknowledgements, First and foremost, we would like to express our gratitude to ALLAH, the Almighty, for giving us the strength and courage to complete this work.

Our heartfelt thanks go to ***Dr. Meftah ZOUAI*** *and* ***Dr. Asma AMMARI*** *for their availability, guidance, direction, and encouragement throughout this project. their support has been invaluable.*

Last but not least, we would like to acknowledge our professors who have taught us throughout our academic journey.

Dedication

I dedicate this work :

To my mother for her love, affection, and sacrifices.

To my father for his support, encouragement, and the trust he has bestowed upon me.

To my grandmother, my source of energy, without whom I wouldn't have reached this far.

To my siblings, my pillars of strength.

To my loyal friend.

To every member of my family and everyone who loves me.

Maroua MOUADA

With deep love and gratitude, this dedication is for my incredible mother, father, and siblings.

Your unwavering support, boundless love, and shared moments have shaped me into who I am today.

Thank you for being my guiding lights and the foundation of my existence.

Sara BENABDALLAH

ملخص

في السنوات الأخيرة، أثارت تكنولوجيا الذكاء الاصطناعي و الأشياء المتصلة بالانترنت الكثير من الاهتمام، خاصة في مجال المنازل الذكية التي تشمل العديد من اجهزة و تطبيقات الانترنت الأشياء. من أهم ميزات المنازل الذكية هي منظفات الارضيات الذكية، التي تهدف الى جعل مهمة تنظيف الأرضيات مهمة سهلة و سريعة. الهدف من هذا المشروع هو هو انجاز روبوت منظف ارضيات ذكي. هذا الروبوت يهدف إلى تسهيل الاعمال المنزلية المملة و المتعبة.

تصميم روبوت منظف الأرضيات الذكي باستغلال تقنيات الانترنت الأشياء و الذكاء الاصطناعي، هذا الجهاز سيكون باستطاعته تنظيف الأرضيات بشكل جيد و اتخاذ اللحظة المثلى لبدء عملية التنظيف. هذا المشروع يتضمن العديد من المجالات التقنية منها علم الروبوتات، البرمجة، الذكاء الاصطناعي و الالكترونيات. النتيجة النهائية هي حل مفيد و مبتكر لتحسين الحياة اليومية.

الكلمات المفتاحية: إنترنت الأشياء، مكنسة الأرضيات الذكية، الذكاء الاصطناعي، الروبوتيات.

Résumé

Ces dernières années, la technologie de l'Internet des objets et l'intelligence artificielle ont suscité beaucoup d'intérêt, notamment dans le domaine des maisons intelligentes qui regroupent de nombreuses applications IoT. Parmi les caractéristiques les plus remarquables de la maison intelligente se trouvent les aspirateurs intelligents, qui visent à rendre la vie quotidienne plus confortable en simplifiant la tâche de nettoyage des sols.

L'objectif de ce travail est de créer un aspirateur intelligent pour les sols. Ce robot de nettoyage est conçu pour faciliter et automatiser les corvées ménagères fastidieuses et fatigantes.

La réalisation d'un projet d'aspirateur intelligent pour les sols implique la conception et la construction d'un robot capable de nettoyer les sols de manière autonome. En exploitant la technologie de l'IoT et de l'IA, le robot aspirateur intelligent peut nettoyer les sols parfaitement et décider du moment optimal pour démarrer le processus de nettoyage. Ce projet nécessite une approche multidisciplinaire, mobilisant des connaissances et des compétences dans des domaines tels que la robotique, la programmation, l'électronique et l'IA. Le résultat final est une solution utile et innovante qui peut améliorer la vie quotidienne en prenant soin d'une tâche banale.

Mots clés : Internet des objets, nettoyeur de sol intelligent, intelligence artificielle, robotique.

Abstract

In recent years, Internet of Things technology and Artificial intelligence have generated a lot of interest, particularly in the field of smart homes which includes many IoT applications. Among the most notable features of the smart home are smart floor cleaners, which aim to make daily life more comfortable by simplifying the task of cleaning floors.

The goal of this work is to create a smart floor cleaner. This cleaning robot is intended to facilitate and automate tedious and tiring household chores.

Creating a smart floor cleaner project involves designing and building a robot capable of autonomously cleaning floors. By leveraging IoT and AI technology, the smart floor cleaner robot can clean floors adequately and can decide the optimum moment to start the cleaning process. This project requires a multidisciplinary approach, involving knowledge and skills in fields such as robotics, programming, electronics and AI. The Final result is a functional innovative solution that can improve daily life by taking care of a mundane task.

keywords : Internet of things, smart floor cleaner, Artificial Intelligence, Robotics.

Table of Contents

Acknowledgment I

Dedication II

General Introduction 1

1 Fundamental Concepts of The Internet of Things 3

1.1 Introduction . . . 4

1.2 Definitions . . . 4

1.2.1 Internet of Things . . . 4

1.2.2 M2M . . . 4

1.2.3 Connected Object . . . 5

1.3 Components of an IoT system . . . 5

1.4 Architecture of an IoT system . . . 6

1.5 Communication Protocols . . . 6

1.5.1 MQTT Protocol . . . 6

1.5.2 AMQP Protocol . . . 7

1.5.3 CoAP Protocol . . . 8

1.5.4 HTTP Protocol . . . 8

1.6 Main Characteristics of IoT . . . 9

1.7 Advantages and Challenges of Internet of Things . . . 10

1.7.1 Advantages of Internet of Things . . . 10

1.7.2 Challenges of Internet of Things . . . 11

1.8 Application Domains of Internet of Things . . . 12

1.9 Conclusion 14

2 Generalities about Robotics **15**

2.1 Introduction 16
2.2 Robotics 16
2.3 Functioning of Robots 16
2.4 Mapping and navigation 17
2.4.1 Mapping 17
2.4.2 Navigation 18
2.5 Types of Robots 19
2.5.1 Autonomous Mobile Robots 19
2.5.2 Articulated Robots 19
2.5.3 Cobots 19
2.5.4 Hybrid Robots 20
2.6 Technological Advancements That Impact Autonomous Mobile Robots 20
2.6.1 Hardware 21
2.6.2 Software 21
2.7 Cleaning Robots 22
2.7.1 Types of Cleaning Robots 22
2.7.2 Types of Floor Cleaning Robots 24
2.8 Related Works 25
2.9 Conclusion 26

3 System Conception **27**

3.1 Introduction 28
3.2 General Architecture 28
3.3 Detailed architecture 29
3.3.1 Perception Module 30
3.3.2 Movement Module 30
3.3.3 Processing Module 30
3.3.4 Energy Module 30

3.4 Methodologies . . . 31
3.4.1 Mapping and localization . . . 31
3.4.1.1 Mapping . . . 31
3.4.1.2 Localization . . . 34
3.4.2 Navigation . . . 34
3.4.3 Uncleaned Spots Treatment . . . 36
3.4.3.1 Self-Organizing Maps Algorithm . . . 37
3.4.3.2 A* Algorithm . . . 38
3.4.4 Adaptive Mop Lifting Mechanism . . . 40
3.4.5 Artificial Neural Network Model . . . 41
3.5 UML Diagrams . . . 42
3.5.1 Use Case Diagram . . . 42
3.5.2 Sequence Diagram . . . 43
3.6 Conclusion . . . 46

4 System Implementation **48**
4.1 Inroduction . . . 49
4.2 Development Environments . . . 49
4.2.1 Programming Languages and Frameworks . . . 49
4.2.1.1 Python . . . 49
4.2.1.2 C language . . . 50
4.2.1.3 Java . . . 50
4.2.1.4 Numpy . . . 50
4.2.1.5 Tensorflow . . . 51
4.2.1.6 MiniSom . . . 51
4.2.2 Development tools . . . 51
4.2.2.1 Jupyter . . . 52
4.2.2.2 Arduino IDE . . . 52
4.2.2.3 Android studio . . . 52
4.3 Electronic Schema . . . 53

4.3.1 Hardware Presentation 54
4.3.1.0.1 NodeMCU ESP8266 : 54
4.3.1.0.2 Stepper Motor : 54
4.3.1.0.3 Ultrasonic Sensor : 54
4.3.1.0.4 Servo motor : 55
4.3.1.0.5 PIR sensor : 55
4.3.1.0.6 KY-037 sensor : 56
4.4 Hardware Installation 56
4.5 Source Code Implementation 58
4.5.1 Environment Perception 58
4.5.2 Robot's Movement 60
4.5.3 Mapping 62
4.5.4 Self-Organizing map function 62
4.5.5 A* function 63
4.5.6 Robot's Direction 64
4.5.7 ANN Model Architecture 65
4.6 Results and Discussion 66
4.6.1 Hardware Aspect 66
4.6.1.1 Distance Detection Result 66
4.6.1.2 Obstacles Presentation 67
4.6.2 Software Aspect 67
4.6.2.1 Result of Mapping Procedure 67
4.6.2.2 Result of SOM Algorithm 68
4.6.2.3 ANN Model Result 68
4.6.2.4 Mobile Application 70
4.7 Conclusion 71

General Conclusion **72**

Bibliography **73**

List of Figures

1.1 Components of Internet of Things system [6] . . . 5
1.2 Iot system architecture [13] . . . 6
1.3 MQTT Protocol [49] . . . 7
1.4 AMQP Protocol [3] . . . 7
1.5 CoAP Protocol [37] . . . 8
1.6 HTTP Protocol [16] . . . 9
1.7 Application domains of IoT [39] . . . 14

2.1 Functioning of a robot [29] . . . 17
2.2 Types of maps [10] . . . 18
2.3 Robots Types [12] . . . 20
2.4 Cleaning Robots [21] . . . 22
2.5 Window Cleaning Robots [28] . . . 23
2.6 Lawn Mowing Robots [43] . . . 23
2.7 Floor Cleaning Robots [17] . . . 24
2.8 Types of Floor Cleaning Robots [17] . . . 25

3.1 General architecture . . . 28
3.2 Detailed architecture of the system. . . . 29
3.3 Mapping Mathematical Model . . . 32
3.4 Robot behaviour . . . 35
3.5 Robot's behavior facing obstacles . . . 35
3.6 Robot Leaving Uncleaned Spots . . . 37

3.7 ANN Model Architecture 41
3.8 Use Case Diagram 42
3.9 System Starting and Environment Perception Diagram 43
3.10 Mapping and Localization and Obstacle Avoidance Diagram 44
3.11 No Obstacle Case Diagram 45
3.12 Uncleaned Spots Phase Diagram 46

4.1 Python logo 49
4.2 C language logo 50
4.3 Java logo 50
4.4 Numpy logo 51
4.5 Tensorflow logo 51
4.6 Jupyter logo 52
4.7 Arduino ide logo 52
4.8 Android studio logo 53
4.9 Electronic Schema 53
4.10 NodeMCU ESP2866 54
4.11 Stepper motor 54
4.12 Ultrasonic sensor 55
4.13 Servo motor 55
4.14 PIR sensor 56
4.15 KY-037 sensor 56
4.16 Hardware installation 57
4.17 External Appearance of TheRobot 58
4.18 ANN Model Architecture 66
4.19 Obstacles Detection 66
4.20 Obstacles Presentation 67
4.21 Environment Map 68
4.22 Self-Organizing Maps Result 68
4.23 Training and Validation Accuracy 69

4.24 Training and Validation Loss 70
4.25 Mobile Application 70

List of Algorithms

1 Obstacles presentation algorithm 33
2 Algorithm of occupancy grid map 34
3 navigation algorithm 36
4 Self Organizing Maps 38
5 A* algorithm 39
6 Direction algorithm 40
7 Adaptive mop lifting mechanism algorithm 41

List of Source Codes

4.1 function perception_rotation() 58
4.2 function obstacle_rotation() 59
4.3 function move_forward() 60
4.4 function move_Backward() 60
4.5 function move_left() 61
4.6 function move_Right() 61
4.7 function draw_map() 62
4.8 function SOM() 62
4.9 function A_star() 63
4.10 function direction() 64

General Introduction

Context

In recent years, the term "Internet of Things" (IoT) has gained significant attention, driven by the exponential expansion of interconnected devices like smart home appliances, wearables, and industrial machinery. By integrating these devices into the IoT framework, we can establish intelligent, streamlined, and interconnected systems that enhance various aspects of business operations, smart cities, healthcare, and numerous other domains.

By harnessing the power of sensors, connectivity, and automation, a **smart floor cleaner** forms an integral component of the IoT ecosystem. Through the utilization of its sensors and artificial intelligence algorithms, it maximizes its efficiency by adapting to the surrounding environment. This robotic solution exemplifies the potential of IoT technology in streamlining domestic responsibilities and enhancing the overall quality of everyday life.

Problematic and Motivation

Throughout the course of human history, cleaning has consistently remained a laborious and time-consuming chore. Various methods were employed to clean spaces, yet they all demanded significant effort. As the population grew and work commitments intensified, finding the time to clean rooms became increasingly challenging. Consequently, the existing cleaning system was deemed inefficient in meeting the demands of modern lifestyles.

Advancements in domains such as the Internet of Things, Artificial Intelligence, Robotics, and Electronics serve as a driving force to develop innovative solutions that significantly en-

hance the quality of our lives in numerous ways.

Objectives

The primary objective of this project is to develop and implement an intelligent floor-cleaning robot that incorporates various components, including the ESP8266 microcontroller, ultrasonic sensor, motors, and more. This autonomous mobile robot is designed to navigate unfamiliar environments, detect and avoid obstacles, construct an occupancy grid map of its surroundings, and accurately determine its own position within the map. Additionally by using an Artificial Neural Network model, it is equipped with the capability to autonomously decide the optimal timing for operation, such as when the user is away from home or workplace, or during periods of sleep, in order to minimize chemical exposure and transform floor cleaning into an effortless and automated task. This versatile robot is suitable for deployment in diverse settings, including offices, homes, and industrial environments, and can be easily activated with a simple button.

After the general introduction, the thesis will be divided into four chapters as follows :

Chapter 01 :Fundamental Concepts of Internet of Things, The initial topic covered in this chapter is the Internet of Things, where we present its components, architecture, protocols, advantages and challenges and finally its application domain.

Chapter 02 :Generalities about Robotics, This chapter delves into the realm of Robotics, elucidating its conceptual framework, the navigation techniques, and the categorization of robots, with an emphasis on autonomous mobile robots to which our floor cleaning robot belongs.

Chapter 03 :System Conception, In this chapter, we will describe the system that we will develop and explain the functioning of each part of it.

Chapter 04 :System Implementation, This chapter will present the implementation tools and code details, as well as discuss the expected results to be obtained.

Chapter 1

Fundamental Concepts of The Internet of Things

1.1 Introduction

The Internet of Things (IoT) is an emerging topic that holds significant technical, economic, and social importance. Everyday objects such as consumer products, durable goods, industrial and public components, trucks, cars, sensors, and more are being connected to the internet and equipped with powerful data analysis capabilities. This transformation promises to radically disrupt the way we work, live, and entertain ourselves.

The initial topic covered in this chapter is the Internet of Things, where we present its components, architecture, protocols, advantages and challenges. The remainder of the chapter focuses on the presentation of the application domains of the Internet of Things. To wrap up, we conclude the chapter.

1.2 Definitions

This section defines some important terms in the field of IoT.

1.2.1 Internet of Things

The concept of "Internet of Things" or "IoT" refers to a network of objects that relies essentially on electronic devices and electronic components such as sensors and electronic boards. These objects can be physical or virtual devices, as well as sensors or actuators. Thus, these devices can produce, exchange, and use data with minimal human intervention [44].

1.2.2 M2M

M2M (Machine-to-Machine) technology involves the automated and streamlined transmission of information between two or more distinct devices. Common examples include smart meters for homes, telematics services for vehicles, asset tracking, wearable technologies, and automated supply chain management (SCM). Although M2M technology is generally designed to operate without manual human assistance, real-time interventions can still occur from time to time [7].

1.2.3 Connected Object

Connected objects, also known as connected devices or IoT devices, encompass physical objects that establish internet connectivity, enabling them to communicate and interact with other devices or systems. These objects possess sensors, software, and network connectivity, empowering them to collect and exchange data and perform a diverse array of functions. Connected objects span from commonplace household devices such as thermostats, smart speakers, and appliances to industrial machinery, vehicles, and wearable devices. Through their connectivity and communication abilities, these objects facilitate remote monitoring, control, and automation, resulting in heightened efficiency, convenience, and innovation potential across multiple domains [32].

1.3 Components of an IoT system

In an IoT system, four fundamental components play crucial roles. First, IoT sensors are capable of collecting data from the physical environment. Second, IoT gateways act as bridges between sensor networks and cloud services. Third, cloud functions facilitate advanced analytics and monitoring of IoT devices. Lastly, user interfaces provide a visible and tangible means for users [6]. As shown in Figure 1.1.

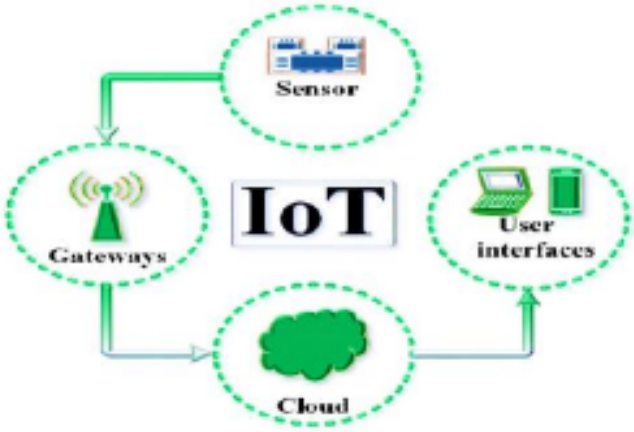

FIGURE 1.1 – Components of Internet of Things system [6]

1.4 Architecture of an IoT system

The architecture of IoT comprises several connected system blocks to ensure the collection, storage, and processing of data generated by sensors in massive data warehouses, as well as the execution of commands sent by the user via an application on the object's actuators.

However, there is no single standard reference IoT architecture model as it encompasses a variety of technologies [13]. A six-layer model is shown in the following Figure 1.2.

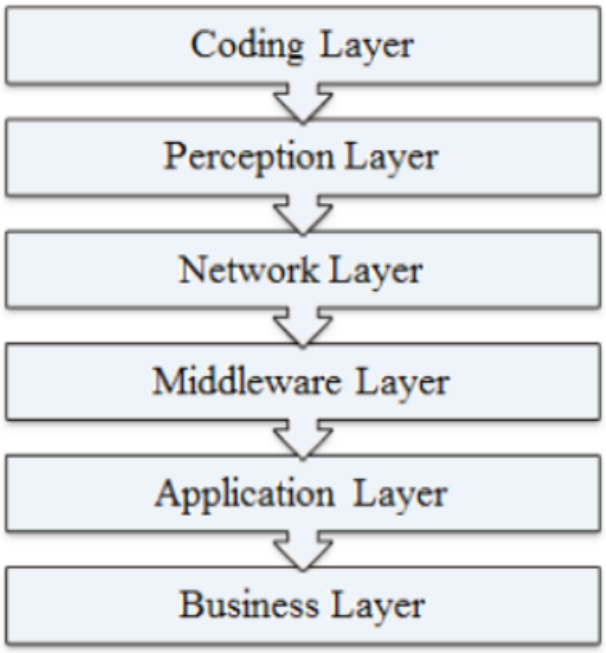

FIGURE 1.2 – Iot system architecture [13]

1.5 Communication Protocols

Several communication protocols can be used to develop IoT systems, including :

1.5.1 MQTT Protocol

MQTT protocol is used in situations where clients have minimal code footprint and are connected to unreliable or limited bandwidth networks. It is mainly used for machine-to-machine

(M2M) connections or for the Internet of Things. MQTT was developed in 1999 to enable monitoring devices in the oil and gas industry to send their data to remote servers. The protocol uses a PUSH/SUBSCRIBE topology and works over TCP/IP. MQTT is based on events, thus minimizing data transmission [46]. Figure 1.3 summarizes the flow of information in MQTT architecture.

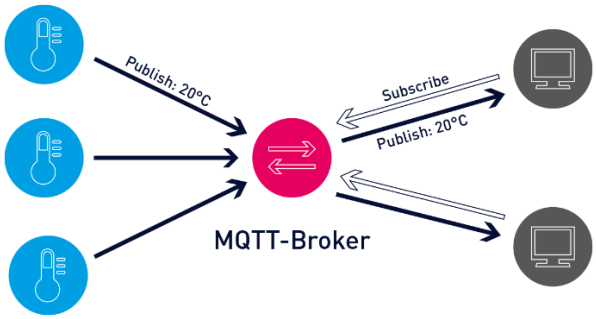

FIGURE 1.3 – MQTT Protocol [49]

1.5.2 AMQP Protocol

AMQP (Advanced Message Queuing Protocol)is a messaging protocol that serves as an alternative to expensive Message Oriented Middleware (MOM) products in the world of IoT. It works similarly to MQTT, but replaces the publish/subscribe concept with producer/consumer. Through its "exchange" mechanism, AMQP allows messages to be routed from a producer to multiple topics using various criteria. This means that multiple consumers can consume the same message through different topics [31]. As shown in Figure 1.4.

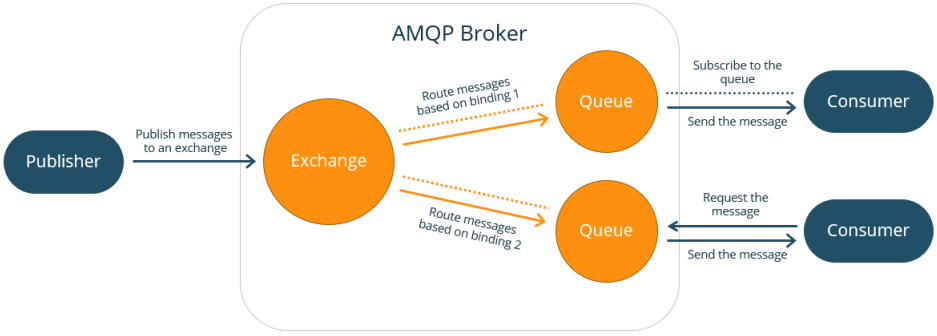

FIGURE 1.4 – AMQP Protocol [3]

1.5.3 CoAP Protocol

CoAP (Constrained Application Protocol) is a Web transfer protocol optimized for constrained networks and devices used in wireless sensor networks to create the Internet of Things. It allows managing the resources of communicating objects and sensors identified by URIs, using a client-server interaction model based on exchanging requests-responses and methods similar to the HTTP protocol [8]. Figure 1.5 illustrates more details.

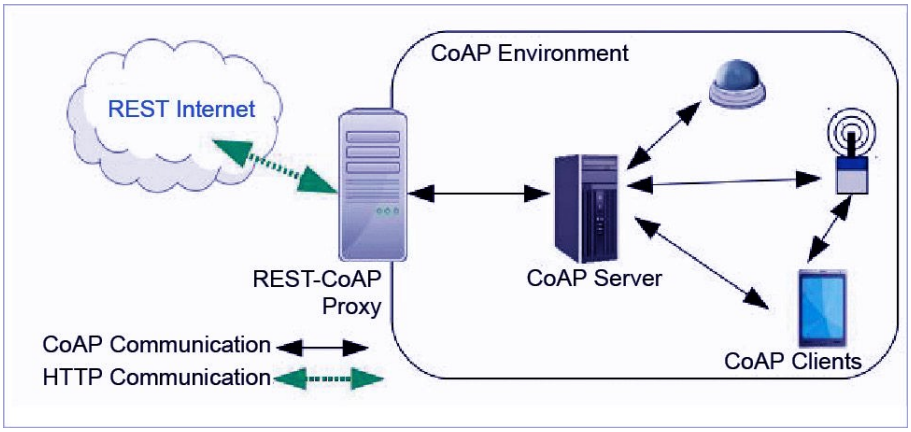

FIGURE 1.5 – CoAP Protocol [37]

1.5.4 HTTP Protocol

The Hypertext Transfer Protocol (HTTP) is an application-level protocol that provides the necessary speed and lightness for distributed, collaborative, hypermedia information systems. It is an object-oriented, stateless, and generic protocol that is suitable for various tasks, including distributed object management systems and name servers, by extending its request methods. HTTP is characterized by its ability to type data representation, which enables systems to be built independently of the transferred data [26]. Figure 1.6 shows HTTP protocol's architecture.

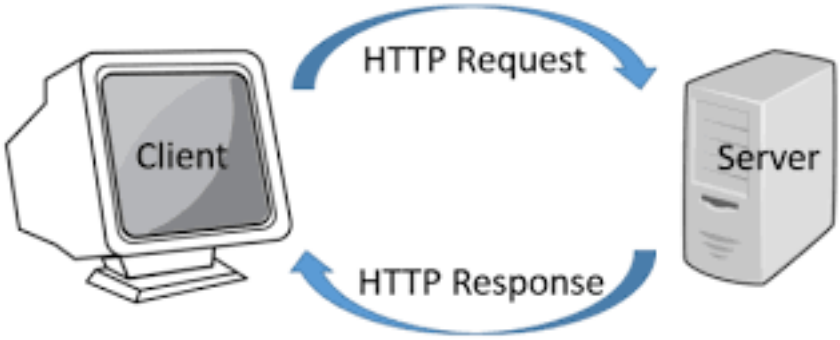

FIGURE 1.6 – HTTP Protocol [16]

1.6 Main Characteristics of IoT

The key characteristics of IoT are as follows [36] :

1. **Interconnectivity**

   refers to the ability of devices and systems to communicate and exchange data with each other. This allows for seamless integration and automation of various processes. Interconnectivity is facilitated through the use of common communication protocols and standards.

2. **Connectivity**

   refers to the compatibility and accessibility of the network.Where accessibility is the ability to access a network while compatibility provides the shared capability to consume and produce data.

3. **Heterogeneity**

   In the IoT, devices are heterogeneous, using different hardware platforms and networks. These devices can communicate with each other or service platforms through a variety of networks.

4. **Dynamic change**

   The status of devices is dynamic, meaning they can change between sleeping and waking up, as well as being connected or disconnected. Additionally, the context of devices, such as number, location and speed, can also vary.

5. **Thing-related services**

   With the help of IoT, we can provide thing-related services while adhering to the limitations of the things themselves. This includes ensuring semantic consistency and confidentiality protection between physical things and their corresponding virtual representations.

6. **Massive scale**

   The number of devices that require management and communication with one another in IoT will be at least ten times greater than the devices currently connected to the Internet.

7. **Security**

   Establishing a comprehensive security system for IoT is crucial to ensure the protection of personal data and the physical well-being of both creators and recipients.

1.7 Advantages and Challenges of Internet of Things

The Internet of Things offers numerous advantages in our day-to-day lives. However, it also presents various challenges. In this section, we will discuss in detail these two controversial aspects.

1.7.1 Advantages of Internet of Things

The Internet of Things provides several advantages for individuals, businesses, and society at large. One of its main benefits is its capacity to improve efficiency, and productivity, and save on costs and efforts. Additionally, it enhances decision-making and provides more personalized experiences. Furthermore, IoT devices can increase safety and security as they can detect and alert individuals or systems to potential hazards or threats. Besides, IoT technology can reduce waste and improve sustainability by optimizing resource usage and minimizing environmental impacts. Ultimately, the IoT has the potential to revolutionize the way we live, work, and interact with each other and our surroundings [23].

1.7.2 Challenges of Internet of Things

While the Internet of Things presents numerous advantages, there are also several challenges that need to be addressed [24].

— **a. Resources limitations**

Resource limitation is one of the critical challenges associated with the Internet of Things. This includes computational limitations due to their low-power processors and slow CPUs, memory limitations due to the lack of sufficient storage space and finally energy limitations due to the use of small and low-power microcontrollers and sensors in IoT devices and the limited availability of power sources.

— **b. Scalability**

As the number of IoT devices connected to the network increases exponentially, and more services are added to IoT application platforms, the developed solutions struggle to keep up with the system requirements. It is therefore crucial to design the right architecture and systems capable of providing the necessary scalability to handle the anticipated growth.

— **c. Security and privacy**

Security is a significant challenge in IoT due to the large number of connected devices that creates a complex and diverse attack surface, making it difficult to secure them all adequately and the limited processing power and memory of IoT devices that makes them more susceptible to security breaches.

— **d. Data manipulation**

Data management will pose a major challenge for the further expansion of IoT, given the addition of numerous new applications and a vast number of devices.

— **f. Price**

It is important to consider the cost of backend servers required to process the data collected by sensors. This cost will depend, of course, on the type and quantity of data being collected and the number of sensors responsible for data collection. Software costs should also be taken into account, both for the servers and for the sensors themselves.

1.8 Application Domains of Internet of Things

IoT has a wide range of application domains across various industries such as health care, agriculture, security and home automation. These domains will be detailed in this section [38] :

1. **Health care :** Hospitals worldwide are increasingly adopting the Internet of Things to enhance productivity and improve patient care. They utilize connected devices such as X-ray and imaging machines, connected monitors, energy meters, and more.
2. **Digital revolution in response to energy imperatives :**In the energy sector, the Internet of Things addresses major challenges such as depletion of natural resources, increasing global energy demands, market price volatility, and labour shortages.
3. **Agriculture :**The rapid growth of the global population, changes in dietary habits, and climate disruptions are three major factors that make modern agriculture a daily challenge, among others. IoT enables farmers to automate their farming practices, thereby increasing production efficiency. Additionally, IoT has allowed farmers to gain a better understanding of different crops and the environmental factors that influence their agricultural productivity.
4. **Road safety :** Connected cars have become particularly popular in recent years and play a crucial role in enhancing road safety. Thanks to the digital revolution, the automotive industry has seen unprecedented opportunities.

   Numerous devices have been developed for connected cars, including :

   - Autonomous emergency call systems.

   - Dashboards synchronized with smartphones.

   - Dedicated applications on digital platforms.
5. **Smart Cities :** Cities, also known as "smart cities" or "connected cities," are undergoing their digital transformation to tackle the challenges of modern society. Urban planning, economy, and sustainable development are major concerns for the cities of tomorrow, which must meet the needs of a growing population with increasingly limited resources.
6. **Smart homes :** The term "Smart Home" refers to a practical home arrangement in which devices and appliances can be remotely controlled from anywhere with an internet connec-

tion, using a mobile device or another networked device. The various devices in the Smart Home are connected via the internet, enabling the user to remotely control various functions such as home security, temperature, lighting, and home entertainment. A variety of robots are employed to assist with these tasks. Here are some examples :

— **a.Security robots :** Security robots, equipped with cameras and sensors, can detect movements and sounds. They can be programmed to patrol the house and alert homeowners in case of potential threats.

— **b.Entertainment robots :** Entertainment robots are used for leisure activities such as playing music, displaying videos, or controlling smart televisions.

— **c.Personal assistant robots :** Personal assistant robots are designed to assist with daily tasks such as setting reminders, making phone calls, and sending messages. These robots are equipped with voice recognition and natural language processing capabilities, allowing users to interact with them through spoken commands or prompts.

— **d.Smart home hubs :** Smart home hubs serve as the central control unit for a smart home. They can be programmed to control various devices and equipment in the house, such as thermostats, lights, and security systems.

— **e.Cleaning robots :** Cleaning robots are specifically designed to perform various cleaning tasks autonomously, using sensors and navigation systems that allow them to avoid obstacles while cleaning. Cleaning robots can be a valuable aid for individuals with limited time or physical capabilities to regularly clean their homes. They can also be used in professional environments such as offices and hotels to maintain clean workspaces. Here are a few examples of cleaning robots :

-Toilet scrubbing robot.

-Laundry-folding cabinet.

-Mobile refrigerator.

-Home organization robot.

-Window-washing robot.

-Self-cleaning oven.

-Smart vacuum cleaner.

-Smart floor cleaner (mop).

The following Figure 1.7 summarizes the most popular application domains of IoT.

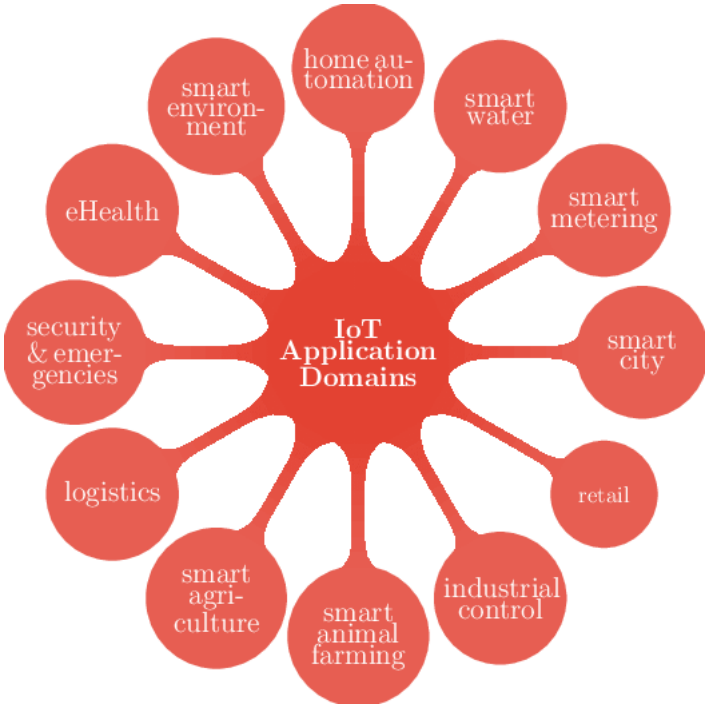

FIGURE 1.7 – Application domains of IoT [39]

1.9 Conclusion

This chapter has delved into the fundamental concepts of the Internet of Things, beginning with an elucidation of the salient terminologies, followed by an explication of the constituent components and architecture of IoT systems. Subsequently, expounded on IoT key characteristics and explored the prevalent protocols in its applications. Then identified and discussed the various advantages and challenges that IoT systems present. Finally, it examined the domains of application of the Internet of Things, with a particular focus on the utilization of robotics in smart homes, particularly cleaning robots, which serves as the principal context of this project that aims to propose a prototype of a smart floor cleaning robot.

The upcoming chapter will delve into the realm of robotics with a particular emphasis on the cleaning robot.

Chapter 2

Generalities about Robotics

2.1 Introduction

Robotics is an intriguing and swiftly developing discipline that integrates engineering, computer science, and artificial intelligence to create, construct, and implement machines that have the ability to independently perform tasks or work alongside humans. These machines, commonly referred to as robots, are designed to replicate human actions, senses, and cognitive abilities, allowing them to engage with the real world and execute intricate tasks with accuracy and effectiveness.

This chapter delves into the realm of Robotics, elucidating its conceptual framework and expounding on the intrinsic nature of robots along with their operational dynamics. Subsequently, an exploration ensues, shedding light on the navigation techniques employed by robots to navigate unfamiliar terrains. The discourse then gravitates towards the categorization of robots, with an emphasis on autonomous mobile robots to which our floor cleaning robot belongs. A comprehensive dissection of cleaning robots, specifically floor-cleaning robots, follows suit. Culminating the chapter, a compendium of related works pertaining to our project is presented, and then we wrap up with a conclusion.

2.2 Robotics

Robotics is an interdisciplinary field that benefits from mechanical engineering, electrical and electronic engineering, computer science, cognitive sciences, biology, and many other disciplines. Robotics is the art, knowledge base, and know-how of designing and using robots in human activities.A robot is a programmable machine that can perform a range of automated tasks, typically controlled by a computer program or an electronic circuit and it can be equipped with intelligence to form an intelligent robot [34].

2.3 Functioning of Robots

Robots are artificial devices specifically designed to perform a variety of tasks with high precision and accuracy. They are typically equipped with a mechanical body, sensors, actuators,

and a control system. Sensors are used for the robot to perceive its environment, while actuators enable the robot to interact with the world through physical movements. The control system utilizes programs and algorithms to guide the robot's actions and decisions [29]. Figure 2.1 summarizes the functioning of robots.

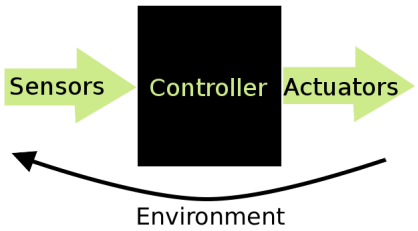

FIGURE 2.1 – Functioning of a robot [29]

2.4 Mapping and navigation

In robotics, mapping and navigation are two key elements. Mapping is the process that allows creating a representation of the robot's environment using sensors and algorithms. Navigation, on the other hand, aims to find a continuous path connecting the robot from its initial configuration to a desired final configuration within that environment. The figure below illustrates the navigation mechanism of mobile robots.

2.4.1 Mapping

In robotics, mapping involves creating a representation of the environment that allows the robot to navigate and interact with its surroundings. This operation requires the use of sensors to collect data about the environment, which are then processed to create a map. Mapping is a crucial element in many robotic applications, including autonomous vehicles, mobile robots, and drones. By creating an accurate map of the environment, a robot can navigate more efficiently and avoid obstacles, enabling it to accomplish tasks more quickly [47]. There are several commonly used map representations in robotic systems, including Point Cloud maps, Geometric maps, and Occupancy Grid maps. Each map type serves a specific purpose in capturing and representing the environment [10]. Figure 2.2 shows the most known map types.

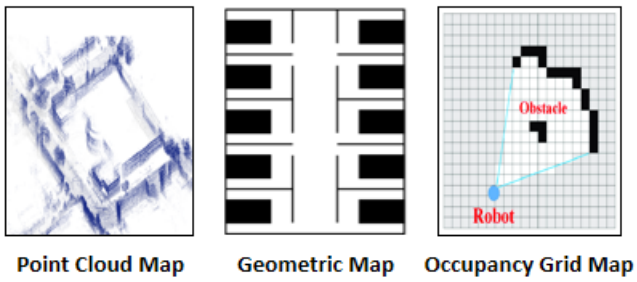

FIGURE 2.2 – Types of maps [10]

2.4.2 Navigation

Navigation in robotics pertains to empowering a robot to strategize and implement its motions with the purpose of accomplishing predefined objectives within its operational surroundings. This process encompasses tasks such as ascertaining the robot's position and generating a path or trajectory that facilitates reaching a target destination or fulfilling a particular task. [1] In robotics, a diverse range of navigation techniques exists, here highlighted some of the most commonly utilized ones :

1. **Systematic Navigation :**Systematic navigation is a navigation approach employed by autonomous mobile robots and intelligent systems to explore and traverse unfamiliar environments in a methodical and effective manner. Within systematic navigation, the robot adheres to a predefined algorithm or strategy to systematically explore the surroundings. This typically involves leveraging sensors and other data inputs to make informed decisions regarding movement and obstacle avoidance [30].

2. **Reactive Navigation :** Reactive navigation entails instantaneous and real-time reactions to the robot's environment. By leveraging sensor feedback, the robot can promptly adapt and modify its movements as necessary. Reactive navigation commonly employs techniques such as obstacle avoidance and reactive behaviors to enable dynamic navigation and collision avoidance [5].

3. **Global Path Planning :**Global path planning focuses on identifying the most efficient path from an initial position to a target location within the environment. This process takes into account various factors such as obstacles, robot capabilities, and objectives to generate a path that minimizes distance, circumvents obstacles, and optimizes other relevant

criteria. Commonly employed algorithms for global path planning include A* (A-star) and Dijkstra's algorithm. These algorithms analyze the environment's characteristics to determine the optimal route for the robot to follow [48].

2.5 Types of Robots

The types of robots can vary based on their application, functionality, and the environment in which they operate. We will mention them in this section.

2.5.1 Autonomous Mobile Robots

Autonomous Mobile Robots (AMRs) move throughout the world and make near real-time decisions as they navigate. Technologies such as sensors and cameras help them gather information about their environment. Embedded processing equipment assists in analyzing the data and making informed decisions, whether it's avoiding an approaching worker, precisely selecting the right package, or choosing a suitable surface to disinfect. These are mobile solutions that require limited human input to perform their tasks [45].

2.5.2 Articulated Robots

Also known as robotic arms, articulated robots are designed to mimic the functions of a human arm. The term "humanoids" is used to identify robots that perform human-centric functions and often take forms similar to humans. They use many similar technological components as AMRs to detect, plan, and act when performing tasks such as providing directions or offering concierge services [27].

2.5.3 Cobots

Cobots, also known as collaborative robots, are designed to work alongside or directly with humans. While most other types of robots perform their tasks independently or in strictly isolated work areas, cobots can share spaces with workers to assist them in accomplishing more.

They are often used to eliminate manual, dangerous, or tedious tasks from daily workflows. In some cases, cobots can operate by sensing and learning from human movements [11].

2.5.4 Hybrid Robots

Different types of robots are often combined to create hybrid solutions capable of handling more complex tasks. For example, an AMR can be combined with a robotic arm to create a robot capable of manipulating packages within a warehouse. As more functionalities are combined into unique solutions, computational capabilities are also consolidated [25].
Figure 2.3 shows a few examples.

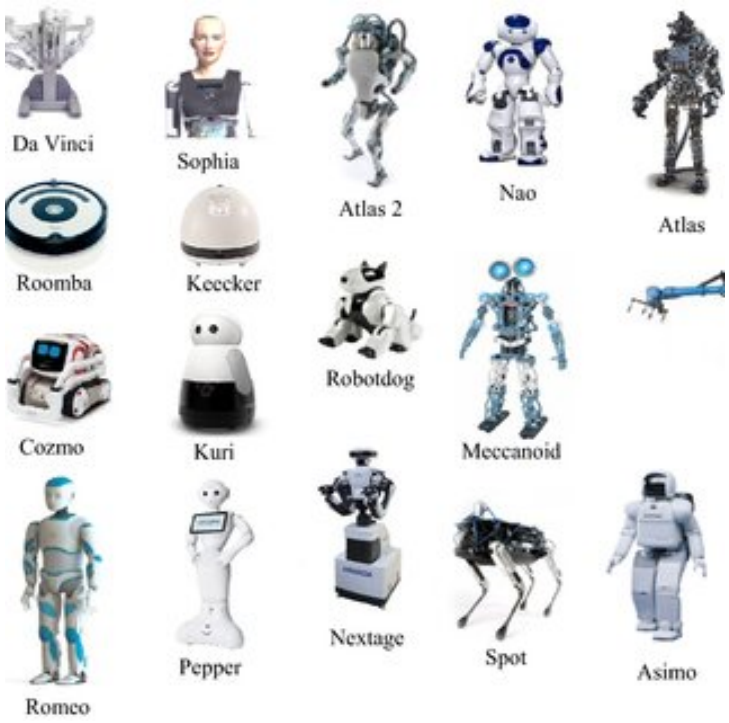

FIGURE 2.3 – Robots Types [12]

2.6 Technological Advancements That Impact Autonomous Mobile Robots

The technological advancements that have had a significant impact on the evolution of Autonomous Mobile Robots (AMRs) include [15] :

2.6.1 Hardware

1. Sensors : AMRs are typically equipped with a wide range of small, cost-effective, and energy-efficient sensing technologies that provide input data for autonomous navigation.
2. Locomotion : The locomotion mechanism of robots refers to how autonomous mobile robots move in their environment. Unlike fixed robot arms, AMRs are designed to move freely and navigate through various types of terrain, including indoor environments, outdoor spaces, and areas with obstacles or uneven surfaces.
3. Batteries : A limited battery capacity and long charging times were weaknesses of AGVs that reduced their performance, usability, and computational power. Higher energy capacity and improvements in charging methods, ranging from conventional plug-in power supplies to wireless energy transmission, have a significant impact on AMR battery management.
4. Manipulation equipment : By combining AMRs with various manipulation equipment into a single unit, new services and material handling operations can be achieved.
5. Processing devices : AI-focused processor architectures such as Intel Nervana, NVIDIA Xavier, and Kneron AI SoC have an impact on the operational level of decision-making in today's AMRs. The computation of complex decisions enables new ways of dynamic routing and scheduling, navigation and classification, as well as an appropriate reaction to obstacles.

2.6.2 Software

1. Simultaneous Localization and Mapping (SLAM) : SLAM, which is a supporting technology for real-time navigation, encompasses the two activities of creating detailed maps of the environment and calculating the position of an AMR on a map.
2. Motion planning : Motion planning is an essential part of vision-based guidance systems and equipment manipulation. By utilizing input data from the environmental representation, the motion planner can calculate the size and dynamics of the robot, as well as a feasible and collision-free path from the initial point to the final position.

3. Artificial Intelligence : Enabled by hardware advancements, AI techniques can be applied to assist AMRs in both navigation and service delivery.

2.7 Cleaning Robots

Current research focuses on creating robots to assist humans in their daily lives. A new generation of robots is being developed to coexist with humans and provide services in homes, workplaces, and public spaces. Automated cleaning robots that belong to the autonomous mobile robots can be used to autonomously clean floors, pools, lawns, and windows, even in the presence of obstacles. The following Figure 2.4 shows some examples of these robots.

FIGURE 2.4 – Cleaning Robots [21]

2.7.1 Types of Cleaning Robots

There are various types of cleaning robots, each developed for specific cleaning tasks. Here are a few examples [21] :

1. **Window Cleaning Robots**

   Designed to automatically clean windows, glass walls, and other flat surfaces, window cleaning robots are specialized robotic devices that require no human intervention. They

often utilize suction or magnetic technology and need to be lightweight, small, compact, and easily transportable, an example is shown in Figure 2.5.

FIGURE 2.5 – Window Cleaning Robots [28]

2. **Lawn Mowing Robots**

   Lawn mowing robots are autonomous devices specifically designed to mow and maintain lawns without the need for human intervention. With integrated sensors, these robots can detect the boundaries of the lawn as well as obstacles, and they can be programmed to work at specific times. They are powered by a battery, and their cutting system uses blades or wires to trim the grass. A lawn mowing robot is shown in Figure 2.6.

FIGURE 2.6 – Lawn Mowing Robots [43]

3. **Floor Cleaning Robots**

   Floor-cleaning robots are autonomous devices designed to clean floors without human intervention. They use sensors to detect obstacles and map the cleaning area, and have various cleaning functions such as vacuuming, brushing, mopping, or sweeping. They are widely used in homes, offices, and public spaces to maintain cleanliness and hygiene. An example is presented in Figure 2.7.

FIGURE 2.7 – Floor Cleaning Robots [17]

2.7.2 Types of Floor Cleaning Robots

There are several types of floor-cleaning robots available today that can make this task much easier and more efficient. In this discussion, we will explore some of the most common types.

1. **Robotic Vacuum Cleaners**

   Robotic vacuum cleaners, also known as robot vacuums, are a popular type of floor-cleaning robot. These robots use a suction technique that combines powerful suction and brushes to remove dirt, dust, and debris from floors and carpets. Most robot vacuums are equipped with a strong motor and a high-efficiency filter that can capture even the smallest particles [18].

2. **Robotic Mop** A robotic mop, also known as a sweeping or mopping robot, is another type of floor-cleaning robot. This device uses a cleaning technique that involves rotating and back-and-forth movements to scrub and wipe the floor. Cleaning pads or a water tank with a cleaning solution are used to make the cleaning process more effective [20].

3. **Scrubbing Robot**

   A scrubbing robot is designed to deep clean floors using a combination of rotating brushes or scrub pads and a water tank. The brushes or pads spin at high speeds, creating a scrubbing motion that can effectively remove dirt and tough stains [51].

4. **Pool Cleaning Robots**

   Pool cleaning robots are specifically designed to clean swimming pools autonomously. They utilize a combination of brushes, filters, and water jets to scrub the walls and floor of the pool, collect debris, and circulate the water to maintain a clean and hygienic swimming environment [9].

Some of the floor cleaning robot types are exhibited in Figure 2.8.

FIGURE 2.8 – Types of Floor Cleaning Robots [17]

2.8 Related Works

In [52], The authors proposed a smart floor-cleaning robot developed using a combination of techniques, hardware, and software. The development process followed Pressman's research and development methodology, which involved analyzing, designing, implementing, and testing the robot. The hardware components included an Omni-wheel system for navigation, a vacuum cleaner for dust cleaning, and a floor polishing motor. The robot's movements and actions were controlled by an Arduino microcontroller, while its software utilized a Bluetooth communication system for control via an Android smartphone. This integration of hardware and software enabled the robot to autonomously navigate, avoid obstacles within a 15 cm range, and efficiently clean different types of dirt, leaving less than 20 percent of dirt on the floor. The creation of this robot contributes to robotics advancements and provides valuable insights for automated floor cleaning systems.

In [1], the proposed system consists of an Arduino Mega, 3 ultrasonic sensors (one at the front and the other two on the left and right), two geared DC motors, a motor driver, an LCD screen, a GSM module, and a vacuum pump. The Arduino Mega controls all the processes to be performed in the system. The ultrasonic sensors are used to calculate the distance between the obstacle and the system. The geared DC motors are driven by the motor driver (L293D) to

control the robot. The GSM module is used to transmit commands via the cellular network. The system can be started and stopped by sending a message from a mobile phone. The vacuum pump is used to suction impurities from the floor. The LCD screen is used to display the system's status.

In this article [19], an automatic vacuum cleaner is designed. It consists of an RC car to which a vacuum cleaner is attached. An ultrasonic sensor is attached to the front of the car, which is used to measure the distance if an obstacle is detected. If an obstacle is detected, the car changes direction according to the code. The vacuum cleaner consists of a CPU fan that runs on battery power. At the front of the vacuum cleaner, a hose is attached to suction dust from the floor. The vacuum cleaner has space to collect the dust. Once it is filled, it needs to be removed and cleaned manually. The vacuum cleaner will be carried on the RC car, and the wheel direction depends on the code uploaded to the Arduino.

2.9 Conclusion

In this chapter, we delved into the fundamental aspects of robots, encompassing their definition, operational principles, and navigation techniques. We also explored the diverse types of robots that exist in the field. Additionally, we provided an overview of the current state-of-the-art in robotics, accompanied by a concise analysis of relevant works related to our project. Through this exploration, we established the hierarchical positioning of our smart floor cleaning robot, as it falls within the category of autonomous mobile robots, further sub-classified as cleaning robots, and ultimately specialized as a floor cleaning robot.

Looking ahead to the upcoming chapter, we will delve into the conception of our project, elucidating the methodologies and techniques employed in the development of our prototype.

Chapter 3

System Conception

3.1 Introduction

This chapter delves into the conception of the proposed Autonomous mobile cleaning floors robot, addressing its design, both at a general and detailed level. The objective is to provide a comprehensive understanding of the operational mechanisms employed by the robot.

Topics covered include the techniques and equations utilized to create an accurate occupancy grid map representing the robot's environment, as well as the navigation strategies Systematic navigation, Self-Organizing Maps and A star, the Artificial Neural Network model integrated into the robot. To enhance comprehension, these algorithms and techniques are illustrated through diagrams that depict the operating principles of the device. Lastly, the chapter is concluded by summarizing the key insights obtained throughout the work.

3.2 General Architecture

This section, presents the general architecture of the proposed system, as shown in Figure 3.1.

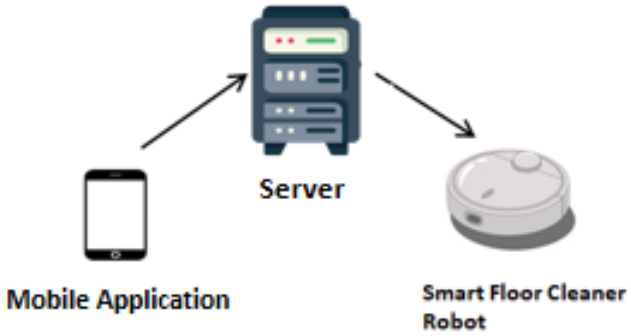

FIGURE 3.1 – General architecture

This architecture aims to enable users to control and monitor the floor cleaner through a mobile application.

1. **Mobile Application :** It serves as a user interface to control the intelligent floor cleaner in two modes, manual starting and automatic starting.
2. **Server :**It is a key element of the architecture that enables the communication between

the mobile application and the intelligent floor cleaner. You can send commands from the mobile application to the robot and receive data from the robot's environment and display it on the mobile application.

3. **Robot :**The smart floor cleaner is equipped with sensors capable of detecting obstacles and perceiving the environment and also a control system to manage the movement and cleaning functions.

3.3 Detailed architecture

The robot consists of five necessary modules, each with its own role. The first is a perception module that provides data about the robot's environment. This information is then sent to the processing module, which controls everything in the system, particularly the motion module, which is responsible for directing and moving the robot. All these modules communicate via the communication module and draw their power from a rechargeable power bank. The following Figure 3.2 illustrates the detailed architecture of the system.

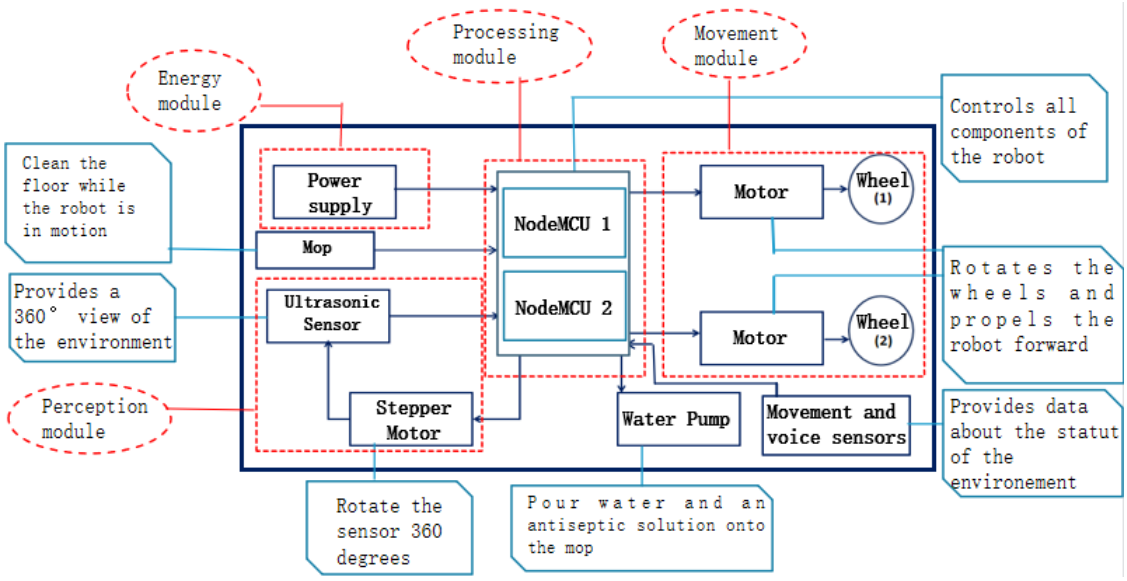

FIGURE 3.2 – Detailed architecture of the system.

3.3.1 Perception Module

This module consists of a stepper motor and an ultrasonic sensor. It is responsible of the environment's perception. The ultrasonic sensor is used to measure the distance between the robot and surrounding objects. The motor enables the sensor to rotate 360 degrees, providing a panoramic view of the environment. These information are used to map the environment in which the robot is located and determine whether it can navigate a given path or not. Additionally, this module involves a PIR (motion and movement sensor) and a KY-037 (sound sensor) ; sensors that provide data used in the decision-making model.

3.3.2 Movement Module

This module consists of two motors and is responsible for managing the movements and displacements of the robot. It works closely with the processing module, which utilizes the information processed by the perception module to guide the robot towards comfortable positions while avoiding obstacles in its path.

3.3.3 Processing Module

This module is considered the brain of the robot as it is responsible for managing all the other modules. It consists of two esp8266 nodes in communication.

3.3.4 Energy Module

In this project, it is necessary to have a reliable power source to ensure the proper functioning of the system. For this purpose, we have chosen to use a power bank.

In addition to the communication protocols which consist of EspNow, HTTP and TCP/IP.

3.4 Methodologies

This section encompasses the methodologies employed to ensure the good functionality and performance of the robot.

3.4.1 Mapping and localization

By utilizing mapping and localization, the robot can create a map of its environment and localize itself on that map. This enables the robot to navigate accurately in unknown environments, avoid obstacles, and plan routes efficiently.

3.4.1.1 Mapping

To provide the robot with a comprehensive environmental perception, an ultrasonic sensor is integrated with a stepper motor capable of 360° rotation. The sensor emits ultrasonic waves and utilizes the time-of-flight principle to measure the duration it takes for the waves to reflect off objects. By applying the following formula :

$$Distance = \frac{\text{Speed of Sound Waves} \times \text{Time of Flight}}{2}$$

This mechanism grants the automaton the ability to perceive its surroundings and make decisions based on the obtained distance measurements.

The distances obtained from the ultrasonic sensor are stored in a JSON (JavaScript Object Notation) file and then transmitted to a computer acting as a server.

By leveraging the distance measurements obtained from the JSon file and the cosine and sine functions a grid comprising binary values is constructed. In this grid, each cell is assigned a value of "1" if it is occupied by an obstacle, and "0" if it is unoccupied. Figure 3.3 below gives more details.

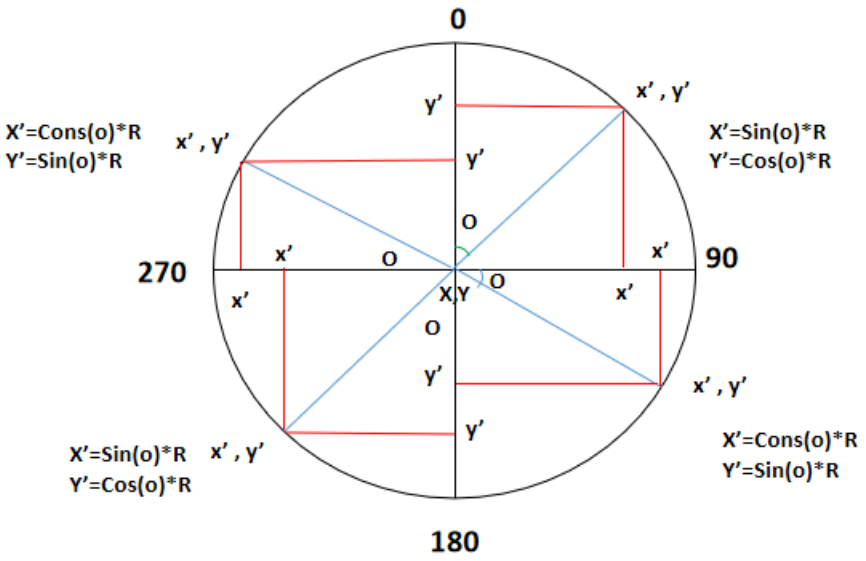

FIGURE 3.3 – Mapping Mathematical Model

The Algorithm presented below provides a summary of how the environment is represented in the grid for one rotation.

Algorithm 1 Obstacles presentation algorithm

```
/* gather data and store it in a json file*/
For degree from 0° to 360° do
    Distance = (SpeedOfSoundWaves × TimeOfFlight) / 2
    JSon file ← Distance;
End for;
/* present obstacles in a grid */
For each row in Grid do
    For each column in Grid do
        If (distance ≤ threshold) then
            If (degree ≤ 90) or (degree > 180 and degree ≤ 270) then
                y=int(diameter*math.cos(angle));
                x=int(diameter*math.sin(angle);
                If (degree ≤ 90) then
                    SetGrid[pos_x − y][x + pos_y] to 1;
                else
                    SetGrid[pos_X + y][pos_Y − x]to1;
            If (degree > 90 and degree ≤ 180) or(degree > 270 and degree ≤ 360) then
                y=int(diameter*math.sin(angle));
                x=int(diameter*math.cos(angle));
                If (degree > 90 and degree ≤ 180)
                    SetGrid[pos_X + y][x + pos_Y] to 1;
                else
                    SetGrid[pos_X − y][pos_Y − x] to 1;
        else
            SetGrid[row][column] to 0 //Unoccupied;
End for;
```

3.4.1.2 Localization

In order to accurately localize and trace the path taken by our robot, the following method is utilized : First, set the motor's speed to a constant value of 10 units. Next, calculate the distance covered in a single rotation by leveraging the diameter of the wheels and every movement done by the robot is sent to the server to follow its positions.

The Algorithm presented below illustrates the systematic process employed to construct the occupancy grid map of the robot's environment.

Algorithm 2 Algorithm of occupancy grid map

```
/* creating the occupancy grid map */
For each row in mapGrid do
    For each column in mapGrid do
        If (mapGrid[row][column]is 0) then
            SetColor of [row][column] to white; // unccupied
        If (mapGrid[row][column]is 1) then
            SetColor of [row][column] to black; // occupied
        If (mapGrid[row][column]is 2) then
            SetColor of [row][column] to red; // robot's position
        If (mapGrid[row][column]is 3) then
            SetColor of [row][column] to green; //discovered
        If (mapGrid[row][column]is 4) then
            SetColor of [row][column] to grey; // undiscovered
    End for;
End for;
```

3.4.2 Navigation

To bestow the robot with autonomous navigation capabilities, the known "grid-based navigation" method is used, a prominent type of systematic navigation elucidated in the preceding chapter. This section delves into the details of applying this technique to optimize the robot's efficiency in performing room cleaning operations.

The following depiction 3.4 showcases the systematic exploration of a room employing the grid-based navigation paradigm.

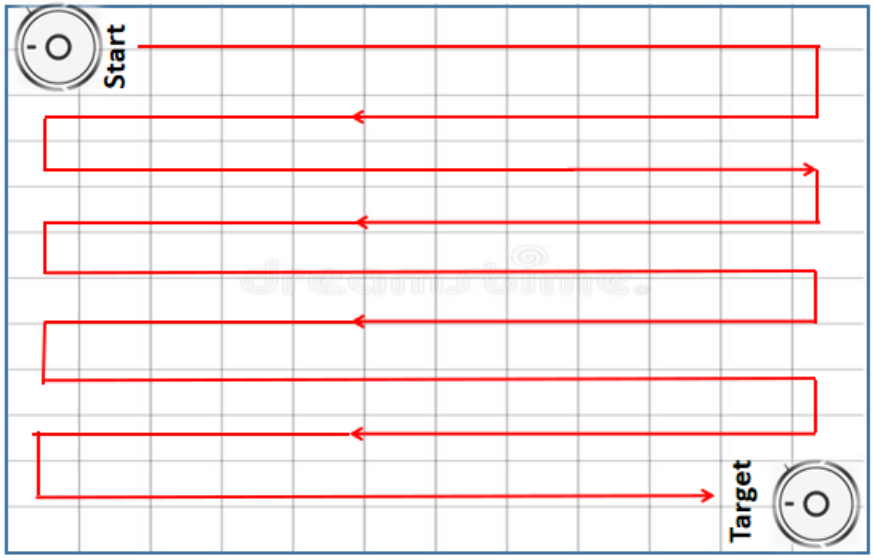

FIGURE 3.4 – Robot behaviour

Now, when the robot encounters an obstacle, it faces two distinct scenarios. The first scenario is when the dimensions of the obstacle are smaller or equal to those of the robot. The second scenario occurs when the dimensions of the obstacle surpass the dimensions of the automaton. The following Schema 3.5 shows the robot's behavior in these two cases :

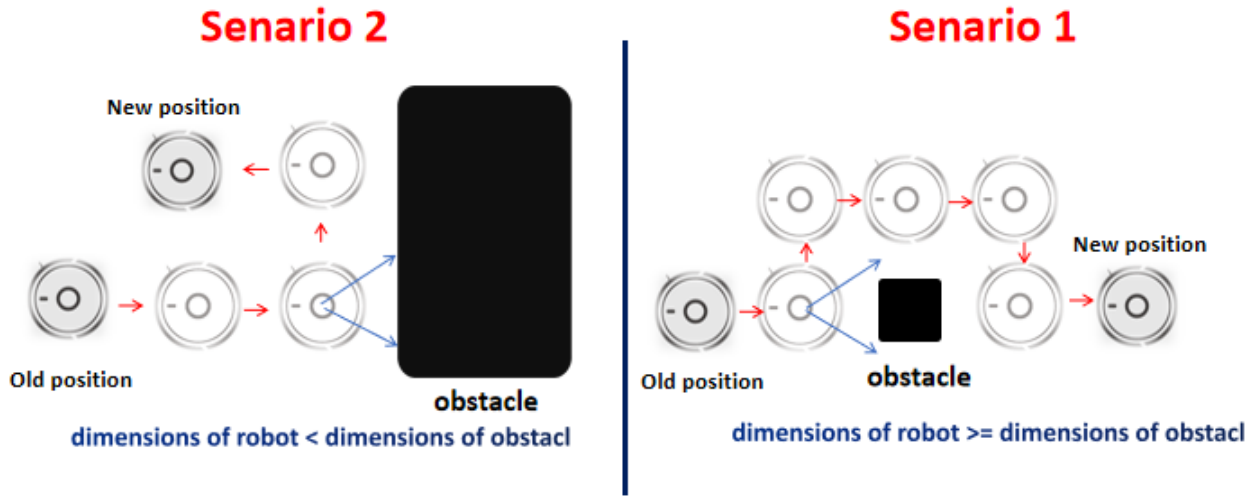

FIGURE 3.5 – Robot's behavior facing obstacles

To elucidate the total behaviour of the proposed device following algorithmic procedure is presented.

Algorithm 3 navigation algorithm

```
While (robotposition ≠ target) do
        /*Check if the adjacent grid cell is reachable and free of obstacles*/
        If (reachable and free) then
            Move the robot to the adjacent grid cell;
            Update the robot's position in the grid-based map;
        else
            If (DimensionOfObstacle ≤ DimensionOfRobor) then
                Stop the robot's forward movement;
                Adjust the robot's direction to move around the obstacle;
            else
                Stop the robot's forward movement;
                Adjust the robot's direction to move along with the obstacle;
                Adjust the robot's direction to move around the obstacle;
            End If;
        End If;
End While;
```

3.4.3 Uncleaned Spots Treatment

Usually, during the robot navigation, the robot leaves behind certain uncleaned spots. The Figure 3.6 illustrates this issue :

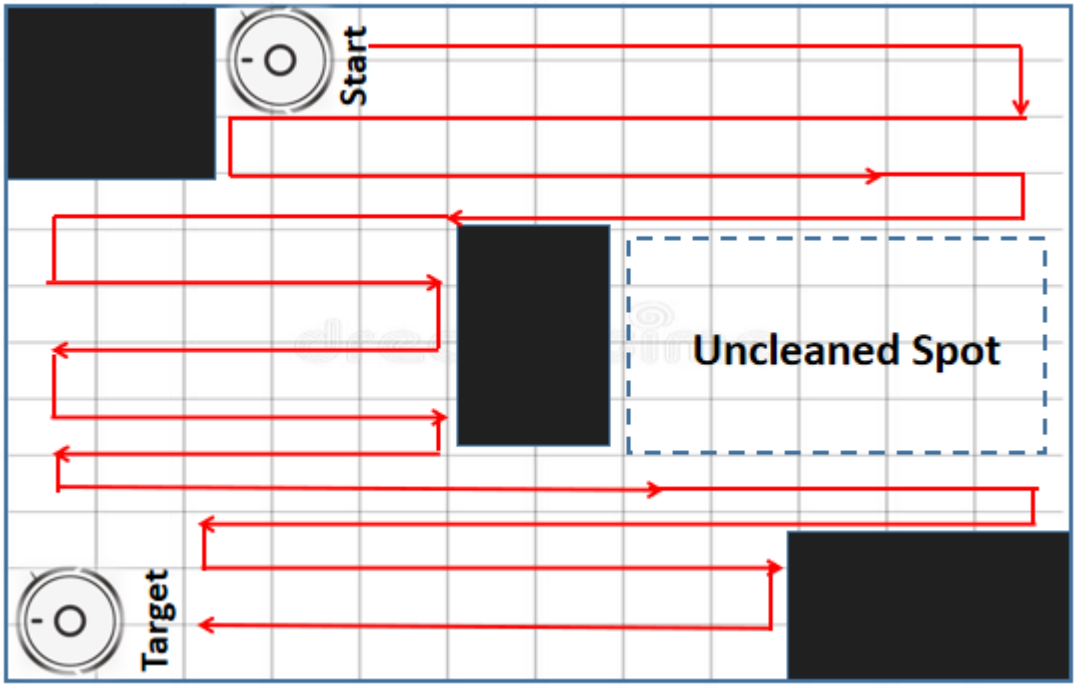

FIGURE 3.6 – Robot Leaving Uncleaned Spots

3.4.3.1 Self-Organizing Maps Algorithm

To address this challenge, the **Self-Organizing Maps (SOM)** algorithm is employed. Which is an unsupervised learning and data visualization algorithm within artificial neural networks. Its purpose is to organize and represent high-dimensional data in a lower-dimensional space, all while maintaining the topological relationships between data points. This is accomplished by training a grid of neurons to recognize various input patterns, effectively grouping similar patterns and creating clusters [33].

The following pseudo Algorithm shows the way we took advantage of the **SOM** algorithm to find relations with the shortest distances between the uncleaned spots :

Algorithm 4 Self Organizing Maps

```
function InitializeSOM(size) :
        Create a SOM grid with size x size nodes;
        Initialize the weight vectors for each node randomly;
        Set the initial learning rate and neighborhood radius;
        Set the maximum number of iterations;
functionTrainingSOM() :
        Repeat until convergence
            For each iteration do
                Select robot's position node from the tour;
                Find the best matching unit (BMU) in the SOM for the se-
                lected node;
                Update the weights of the BMU and its neighboring nodes;
                Adjust the weights using the learning rate and neighborhood function;
                Update the tour;
function OutputSOM() :
        Output the final SOM tour;
function SOM(size,nodes) :
        InitializeSOM(size);
        TrainingSOM();
        OutputSOM();
```

3.4.3.2 A* Algorithm

After employing the Self-Organizing Map (SOM) algorithm to discover the shortest path between the uncleaned spots, the **A*** algorithm is utilized to discover the shortest path from a spot to the next spot in the map.

Algorithm 5 A* algorithm

Function A* (start, goal) :

Initialize the open list;

Initialize the closed list;

$gScore[start] \leftarrow 0$ //*Actaul cost*;

$fScore[start] \leftarrow gScore[start] + heuristic(start, goal)$;

Add the starting node on the open list;

While the open list is not empty **do**

find the node with the least f on the open list;

pop the node off the open list;

generate the node's successors;

set successors parents to node;

For *each successor* **do**

If (*successor is goal*)**then**

Stop search;

else

gScore(successor)=gScore(node) + distance between successor and node;

hScore(successor)=distance(goal, successor);

fScore(successor) = gScore(successor) + hScore(successor);

End for;

push node on the closed list;

End while;

The Algorithm provided below outlines the sequential steps for transitioning from one cell to another based on the A* algorithm results, specifically for a single direction. It ensures consistent treatment in all directions.

Algorithm 6 Direction algorithm

Function $direction(x, y, new_x, new_y, orientation, path)$:

if (orientation is "forward") **then**

if($x = new_x$)**then**

if($y + 1 = new_y$)**then**

orientation ← "$right$";

append"$right$"**and**"$forward$"**to**$path$;

if($y - 1 = new_y$)**then**

orientation ← "$left$";

append "left" **and** "forward" **to** path;

if($y = new_y$)**then**

if (x-1 = new$_x$)**then**

orientation ← "$forward$";

append "forward" **to** path;

if (x+1 = new$_x$)**then**

orientation ← "$backward$";

append "forward" **and** "right" **to** path;

return (orientation,path)

3.4.4 Adaptive Mop Lifting Mechanism

The adaptive mop lifting mechanism incorporated in the smart floor cleaner is a key feature that enhances its cleaning efficiency. When the robot comes to a halt to perceive the environment or when it detects a previously cleaned area based on its previous journey, it lifts up its mops, avoiding redundant cleaning and conserving energy. This mechanism is done by the use of two vector springs that are pushed by a servo motor. The following Algorithm explains this process.

Algorithm 7 Adaptive mop lifting mechanism algorithm

```
While not completely cleaned
        if robot is stopped to perceive Or spot is already cleaned
            lift mops
        else :
            lower mops
        end if
end while
```

3.4.5 Artificial Neural Network Model

To empower autonomous determination of the optimal timing for initiating mopping tasks, intelligent decision-making capabilities are integrated. This is achieved through the incorporation of a PIR sensor for motion detection and a KY-037 sound detection sensor, supported by the implementation of an Artificial Neural Network (ANN) model.

The ANN is trained using a comprehensive dataset that encapsulates various aspects of the user's routine. This dataset includes the weekday, monthday, month, hour, sleep status, vacation status, volume, and location (inside or outside).

The Artificial Neural Network (ANN) model utilizes the given dataset as input. With its intricate architecture, the model effectively analyzes the dataset to determine whether it should initiate the cleaning process. The following Schema 3.7 shows the model's architecture :

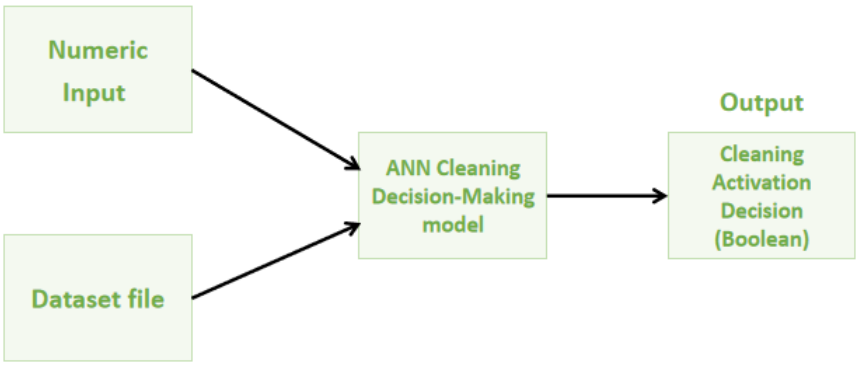

FIGURE 3.7 – ANN Model Architecture

3.5 UML Diagrams

In this section, detailed UML diagrams are presented to provide a comprehensive understanding of the proposed system.

3.5.1 Use Case Diagram

Figure 3.8 presents a use case diagram to outline the key functionalities and interactions of the system.

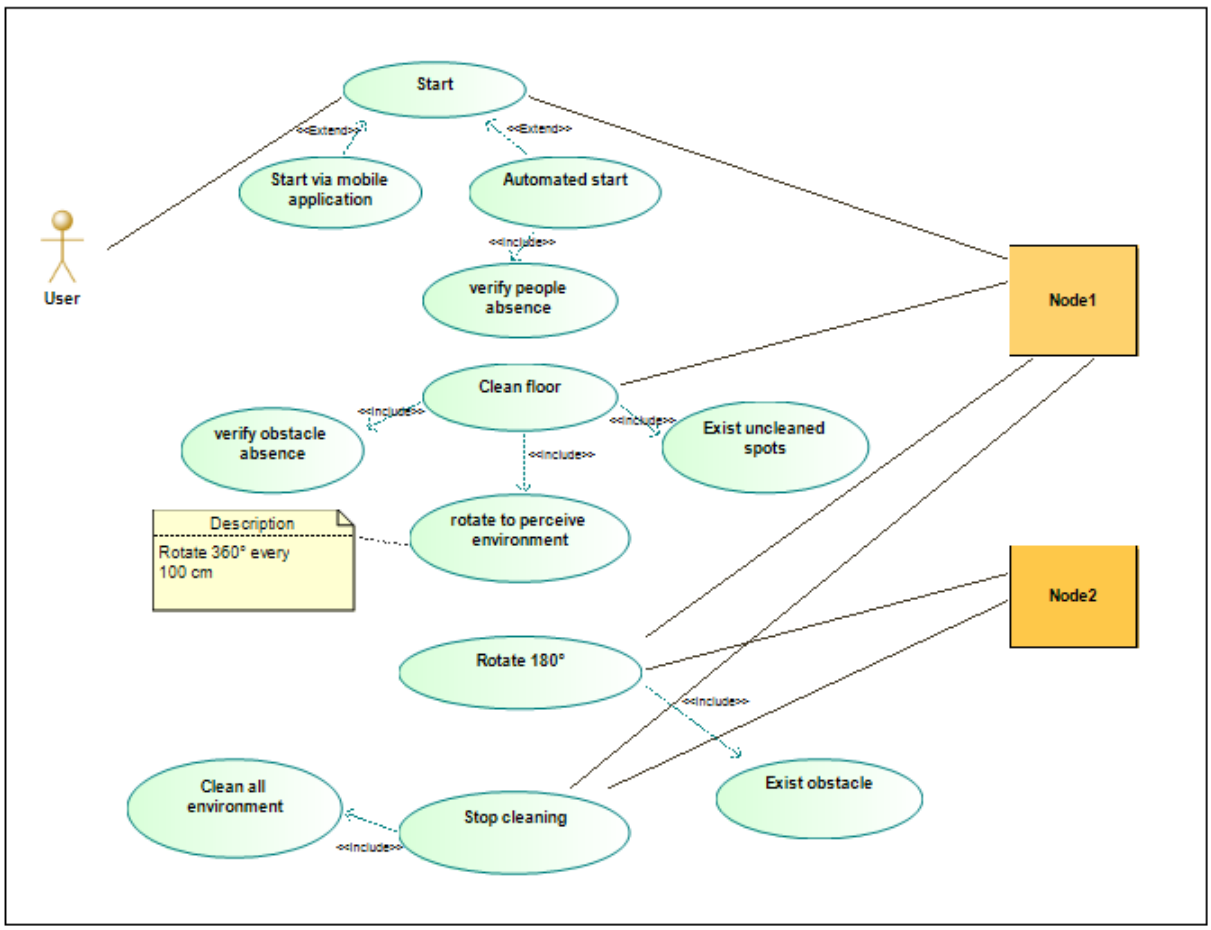

FIGURE 3.8 – Use Case Diagram

3.5.2 Sequence Diagram

In this section sequence diagram is exhibited to provide a clear and concise illustration of the system's execution flow and the collaboration between components that was divided into 4 sub-diagrams.

- Figure 3.9 shows the first sub-diagram that illustrates the starting and the environment perception phase.

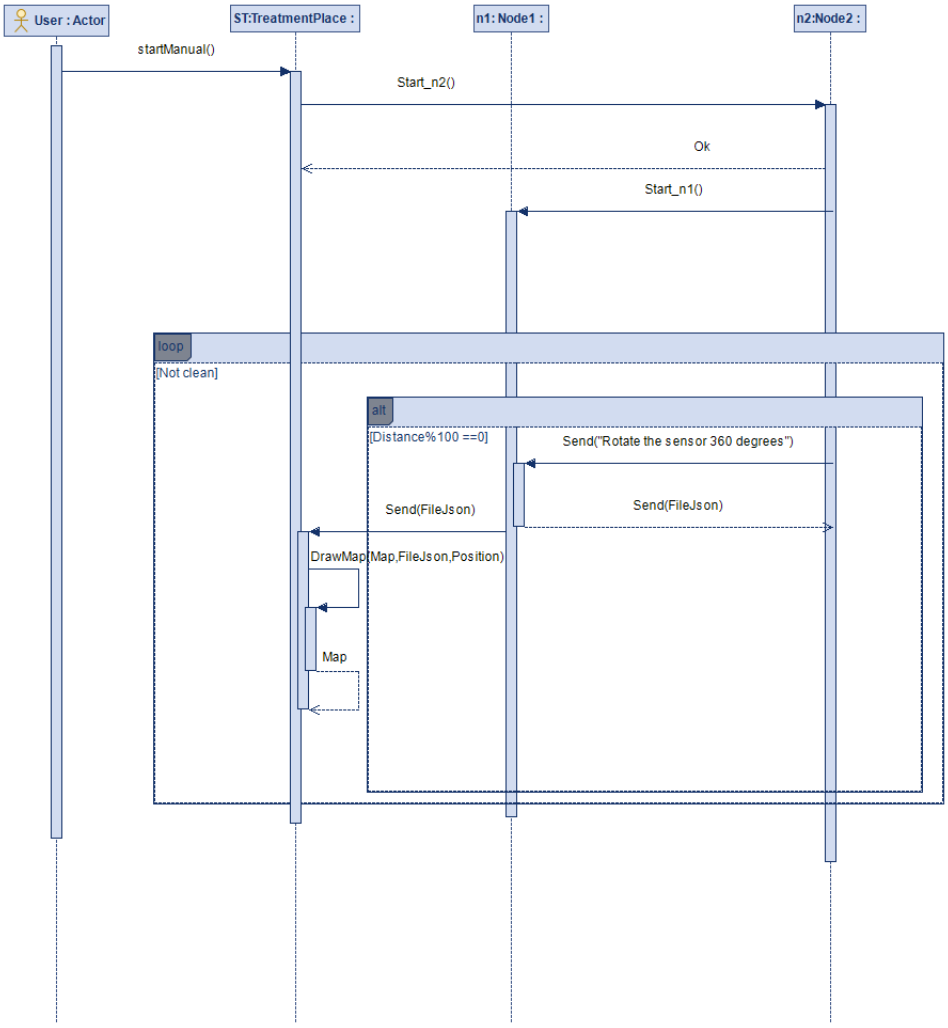

FIGURE 3.9 – System Starting and Environment Perception Diagram

- Figure 3.10 shows the second sub-diagram that demonstrates the mapping and localization in addition to the obstacle avoidance phase.

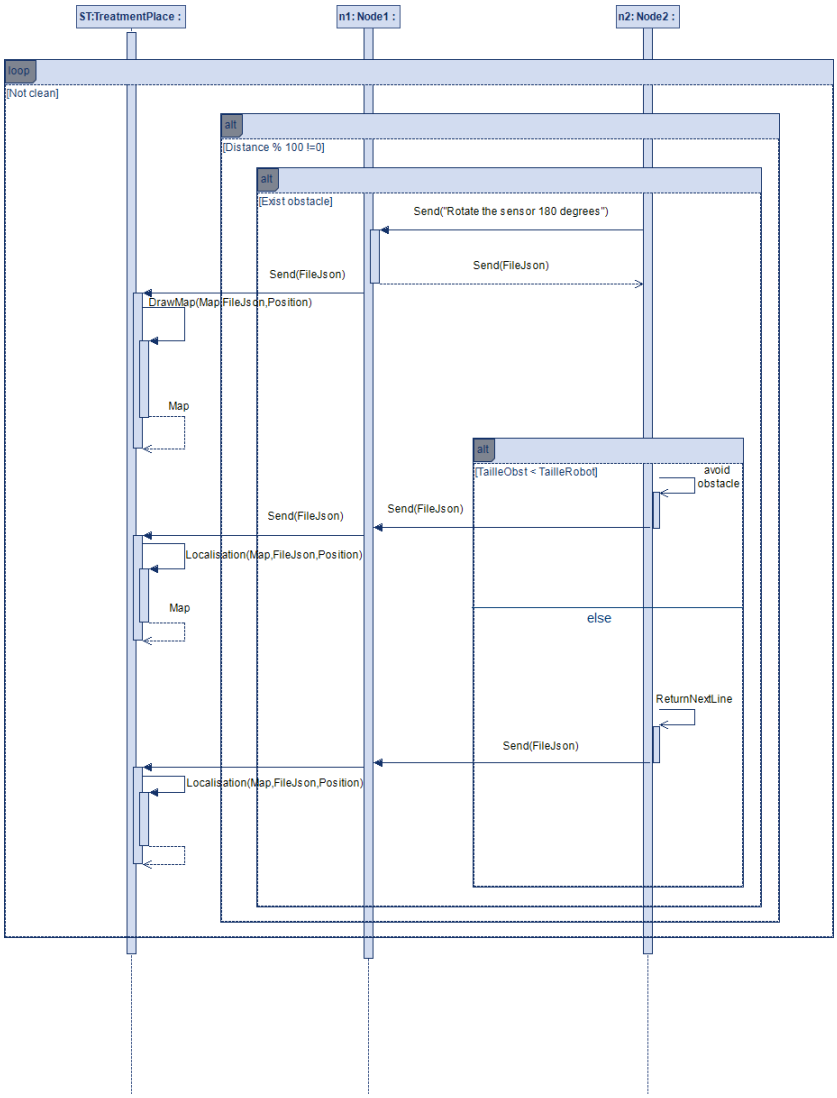

FIGURE 3.10 – Mapping and Localization and Obstacle Avoidance Diagram

- Figure 3.11 displays the third sub-diagram that shows the robot's behaviour when it doesn't face any obstacle.

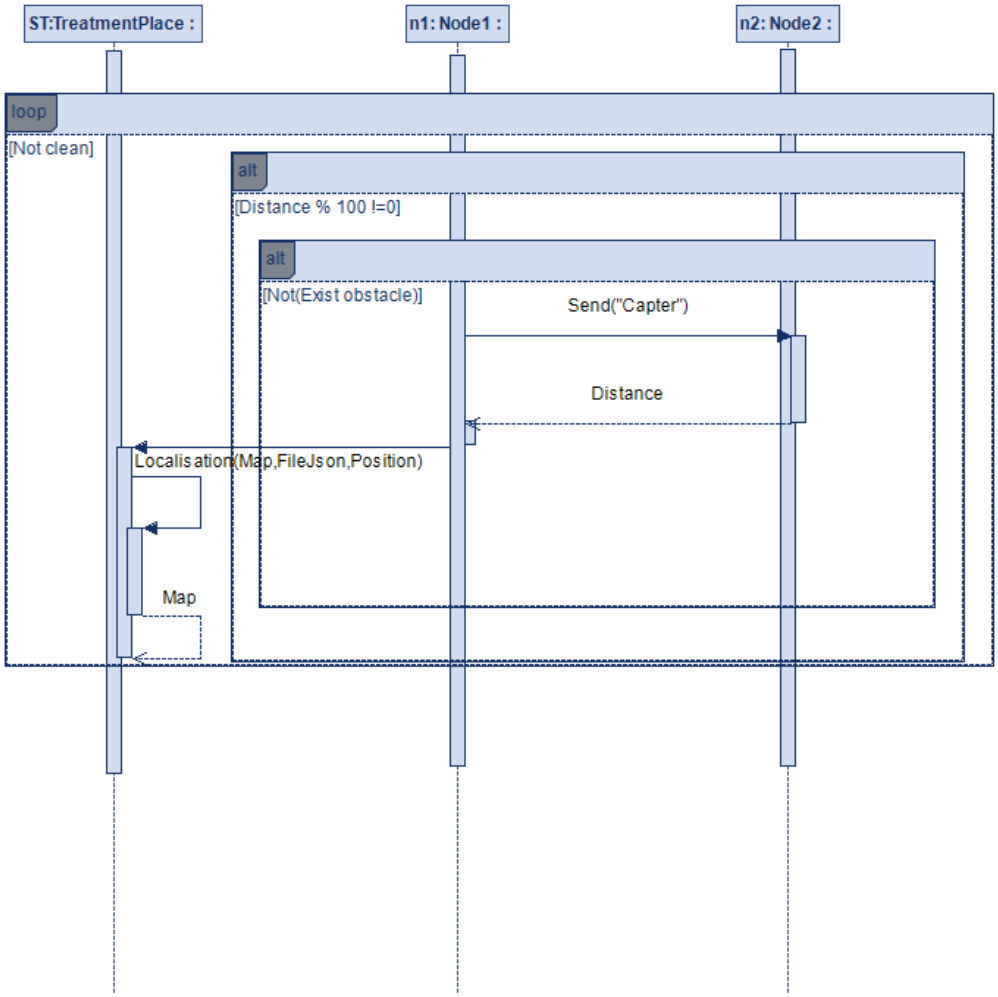

FIGURE 3.11 – No Obstacle Case Diagram

- The Figure 3.12 depicts the fourth sub-diagram that exhibits the robot's behaviour when cleaning the uncleaned spots it leaves behind.

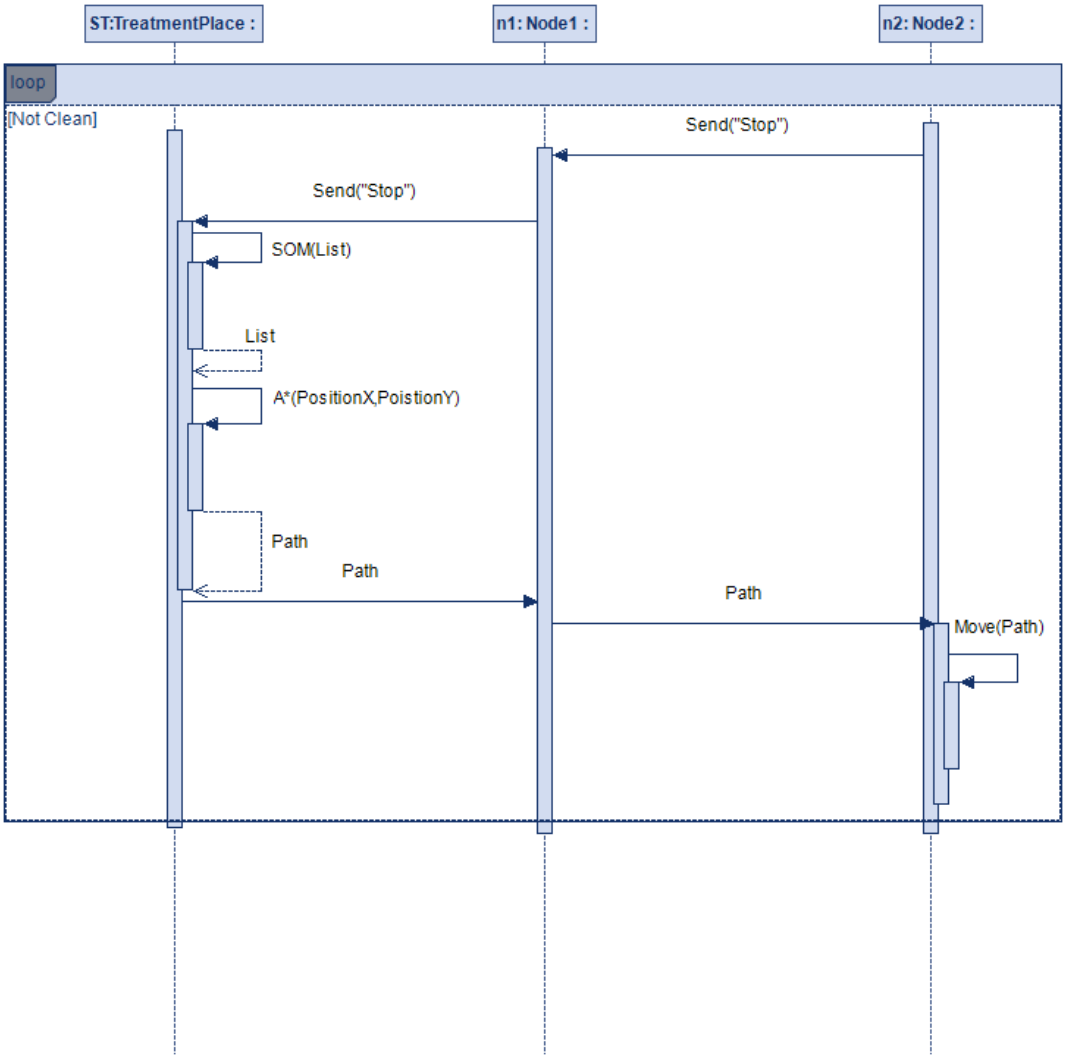

FIGURE 3.12 – Uncleaned Spots Phase Diagram

3.6 Conclusion

In conclusion, this chapter has provided a comprehensive overview of the conception and design of our system. We have explored the general and detailed architecture of our project, delving into the operational mechanisms of our robot. The discussion encompassed a range of topics, including the techniques and equations used to create an accurate map of the robot's environment, the navigation algorithms employed, and the integration of intelligence into our robot. Through the use of diagrams, we have visually illustrated the operating principles of our robot, enhancing understanding. Accordingly, in this chapter, we have gained valuable insights

into the intricacies of the proposed system, setting the stage for further exploration in the subsequent chapter by delving into the practical implementation of the designed system.

Chapter 4

System Implementation

4.1 Inroduction

In the preceding chapter, a comprehensive overview of the system design was offered, delving into the intricacies of each step individually. Building upon that foundation, this chapter elucidates the key programs, tools, programming languages, and libraries employed during the project implementation. Furthermore, showcases the hardware components utilized and offers illustrative code examples. To culminate, it presents compelling results attained through the endeavors.

4.2 Development Environments

This section provides an overview of the programming languages, frameworks, and development tools utilized in the proposed project.

4.2.1 Programming Languages and Frameworks

The programming languages and frameworks used are presented as follows.

4.2.1.1 Python

Python is an object-oriented, high-level, open-source programming language that emphasizes code readability and ease of use. It was created in the late 1980s by Guido van Rossum. Python is widely used in various domains, including web development, standalone software development, and artificial intelligence [40]. Python's logo is presented in Figure 4.1.

FIGURE 4.1 – Python logo

4.2.1.2 C language

C is a high-level, general-purpose programming language created in the early 1970s by Dennis Ritchie at Bell Labs. It is a compiled language and has had a significant influence on the development of many other programming languages. C is still widely used today, particularly in domains such as operating systems, system programming, and embedded systems [41]. Figure 4.2 shows the C language's logo.

FIGURE 4.2 – C language logo

4.2.1.3 Java

Java is a high-level, general-purpose programming language originally developed by James Gosling at Sun Microsystems in the mid-1990s. It is an object-oriented language designed to be platform-independent. Java is widely used for developing a variety of applications, including mobile and web applications, enterprise software, and scientific computing [4]. In the following Figure 4.3 Java's logo is provided.

FIGURE 4.3 – Java logo

4.2.1.4 Numpy

NumPy is a Python library used for data analysis. It provides a multidimensional array object, along with various derived objects such as masked arrays and matrices, and a variety of

functions for fast array operations, including mathematical, logical, and shape manipulation functions. NumPy is widely used in academic, financial, data science, and other scientific computing domains [35]. Numpy's logo is presented in Figure 4.4

FIGURE 4.4 – Numpy logo

4.2.1.5 Tensorflow

TensorFlow is an open-source Machine Learning library created by Google, which enables the development and execution of Machine Learning and Deep Learning applications [2]. Figure 4.5 below shows the TensorFlow logo.

FIGURE 4.5 – Tensorflow logo

4.2.1.6 MiniSom

MiniSom is a Python library package utilized for unsupervised machine learning, specifically for implementing self-organizing maps [22].

4.2.2 Development tools

We have utilized a comprehensive set of development tools and environments throughout the course of our project. Here are the key tools and environments we employed.

4.2.2.1 Jupyter

Jupyter is an open-source web application that allows the creation and sharing of documents that contain real-time code, equations, visualizations, and narrative text. It is compatible with over 100 programming languages, including Python, R, and Julia, and provides users with the ability to write and execute code in an online environment [50]. Figure 4.6 below shows the Jupyter logo.

FIGURE 4.6 – Jupyter logo

4.2.2.2 Arduino IDE

The Arduino IDE (Integrated Development Environment) is a software application used for writing, compiling, and uploading code to Arduino boards. It provides a user-friendly interface for creating and editing code, as well as a library of pre-written code for common functions. The Arduino IDE simplifies the development process for Arduino projects by providing a streamlined environment for writing and managing code [14]. Figure 4.7 displays Arduino's logo.

FIGURE 4.7 – Arduino ide logo

4.2.2.3 Android studio

Android Studio is an integrated development environment (IDE) designed for creating Android applications. It offers a comprehensive set of tools for developing, testing, and debugging

applications. It supports multiple programming languages, including Kotlin and Java. Android Studio is the official development environment for Android applications. It provides a rich and feature-packed environment that enables developers to efficiently build and enhance Android apps [42]. Figure 4.8 presents Android Studio's logo.

FIGURE 4.8 – Android studio logo

4.3 Electronic Schema

This section clarifies how the electronic devices were interconnected during the smart floor cleaner robot preparation as shown in Figure 4.9 :

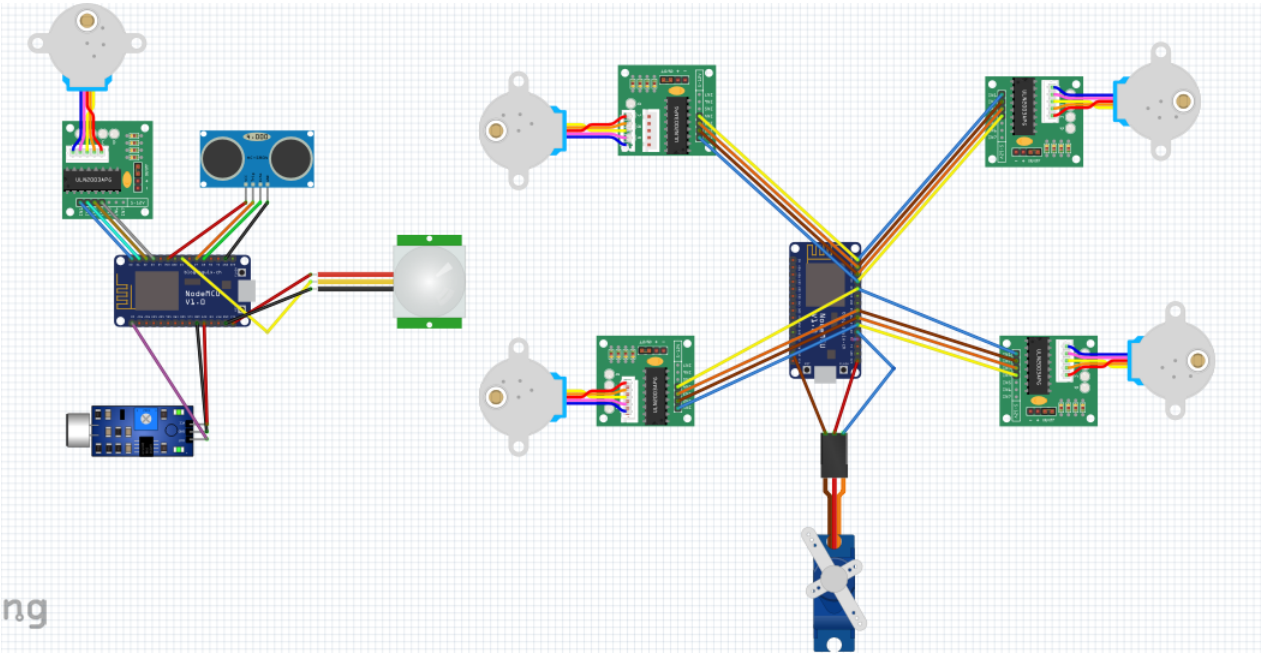

FIGURE 4.9 – Electronic Schema

4.3.1 Hardware Presentation

To successfully carry out the project efficiently, several electronic tools were used, including :

4.3.1.0.1 NodeMCU ESP8266 : a microcontroller board equipped with built-in WiFi connectivity based on the ESP8266. It can be programmed using the LUA language or through the Arduino IDE. Additionally, the NodeMCU features several digital and analog input/output pins, as well as UART, SPI, I2C, and CAN ports. It is a versatile and popular choice for IoT projects, allowing for wireless communication and integration with various sensors and devices. Figure 4.10 presents the microcontroller board.

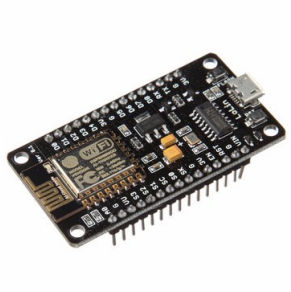

FIGURE 4.10 – NodeMCU ESP2866

4.3.1.0.2 Stepper Motor : the motor presented in the Figure 4.11 is a type of electric motor that operates by performing movements in discrete steps. Unlike conventional DC motors, it is a brushless synchronous motor that divides a complete rotation into a certain number of steps.

FIGURE 4.11 – Stepper motor

4.3.1.0.3 Ultrasonic Sensor : an electronic device that measures the distance to an object by emitting ultrasonic waves (frequencies higher than what the human ear can perceive) and

receiving the waves reflected by the object. This sensor consists of two main components : the transmitter and the receiver. The transmitter sends an ultrasonic pulse through a piezoelectric crystal to the receiver. Figure 4.12 shows the sensor.

FIGURE 4.12 – Ultrasonic sensor

4.3.1.0.4 Servo motor : a type of electric motor that is commonly used in robotics and automation systems. It is designed to provide precise control over angular position, velocity, and acceleration. This motor is shown in Figure 4.13

FIGURE 4.13 – Servo motor

4.3.1.0.5 PIR sensor : Pyroelectric Infra-Red (PIR) sensors find application in various fields, particularly in security systems. These sensors are capable of detecting the presence of humans and animals by sensing the thermal radiation emitted from their bodies. This functionality can be utilized to initiate specific actions, such as activating doors, capturing video footage, and more. This sensor is presented in Figure 4.14

FIGURE 4.14 – PIR sensor

4.3.1.0.6 KY-037 sensor : The KY-037 sensor is a module that detects sound or vibrations in its surroundings. It typically includes a built-in microphone or vibration sensor, along with supporting circuitry for signal processing as shown in Figure 4.15. The sensor is designed to provide a simple interface for measuring sound levels or detecting vibrations in various applications.

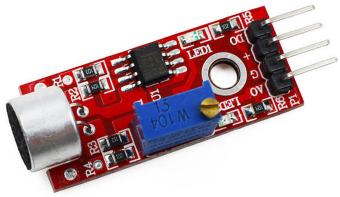

FIGURE 4.15 – KY-037 sensor

4.4 Hardware Installation

This section presents the hardware installation and sensors used in the system and delves into the essential components to achieve the goal of the project"smart floor cleaner robot."

The robot comprises two main units :

- **Unit1 :** incorporates a NodeMcu to which a stepper motor is connected, on top of it is an ultrasonic sensor positioned strategically in the middle of the robot. Also, KY-037 and PIR sensors are connected to the node and positioned in the front of the robot.
- **Unit2 :** incorporates another NodeMcu that connects two stepper motors in the front for the two mops, as well as two stepper motors in the back for the two wheels. Additionally, there is a servo motor that controls the lifting and lowering of the mops.

The robot also contains a water pump that spreads water with the sanitizing solution. Figure 4.16 below shows our smart floor cleaner robot's hardware installation :

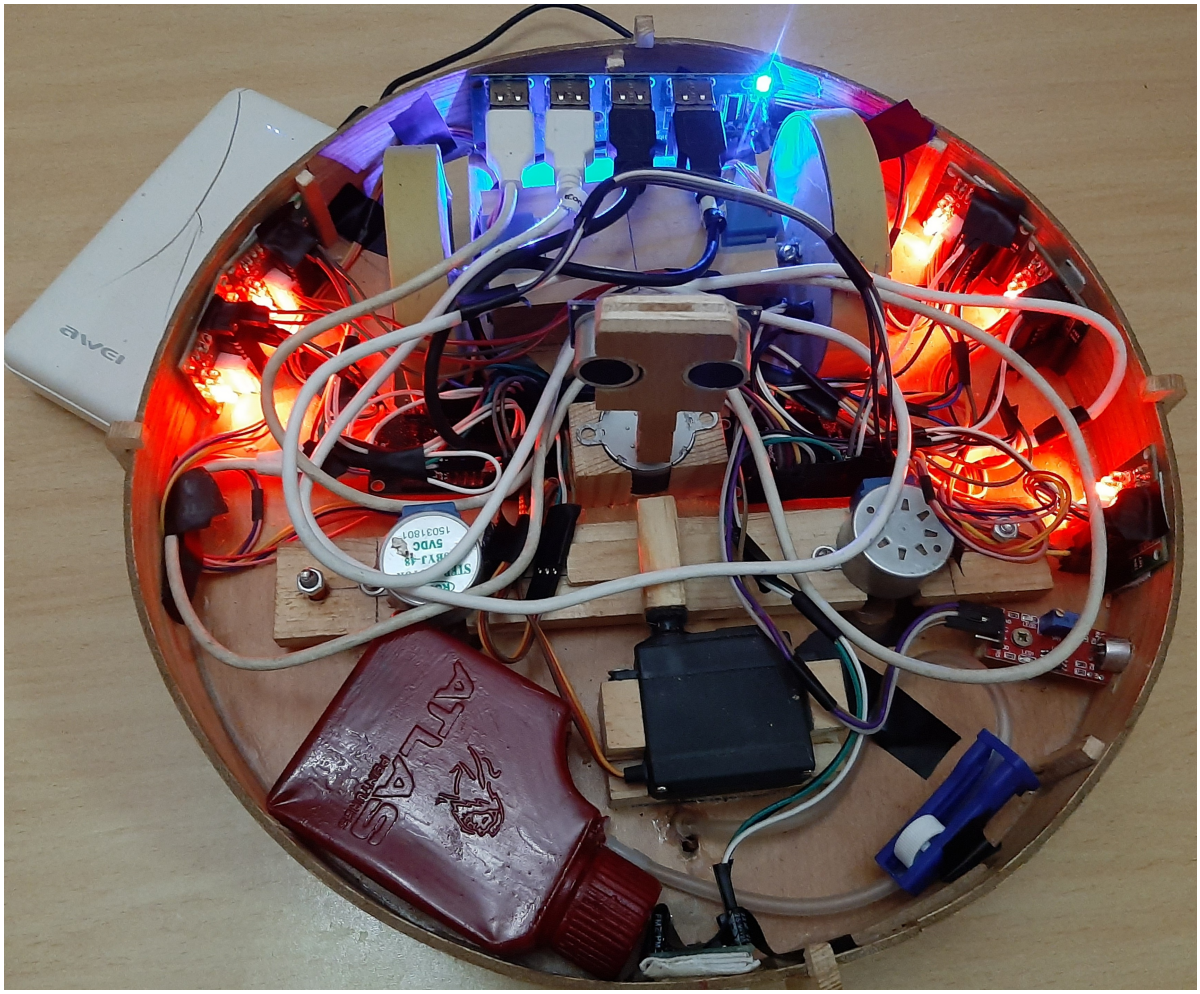

FIGURE 4.16 – Hardware installation

Figure 4.17 illustrates the external appearance of the robot.

FIGURE 4.17 – External Appearance of TheRobot

4.5 Source Code Implementation

This part demonstrates sections on the code used to develop and program the robot.

4.5.1 Environment Perception

Excerpts from the source code that empower the robot to perceive its environment and store the collected data in a JSON file are presented. The first function comes into play every one meter to perceive, while the second function handles scenarios involving obstacles.

1. **360° rotation function :** Environment perceiving function.

```
void Rotate_sensor_360(){
   int i=0;
   DynamicJsonDocument doc(4096);
```

```
    while(i<2048){ //0 TO 360 rotation
       float Distance ;
       Distance= Distance detection(i);
       doc.add(Distance);
       sensor.step(16);
       i=i+16;
    }
    i=0;
    while(i<2048){
        sensor.step(-16);
        i=i+16;
        }

  String json;
  serializeJson(doc, json);
  File file = SPIFFS.open("/data.json", "w");
  file.println(json);
  file.close();
}
```

Source Codes 4.1 – function perception_rotation()

2. **90° rotation function :** Obstacles detection function.

```
void Rotate_sensor_90(){
  DynamicJsonDocument doc(4096);
  sensor.step(512);
  int i=0;
  int k=0;
  while(i<1024){
      int Distance= capter(i);
      doc.add(Distance);
      sensor.step(-16);
```

```
      i=i+16;
  }
  sensor.step(512);
  String json;
  serializeJson(doc, json);
  File file = SPIFFS.open("/data.json", "w");
  file.println(json);
  file.close();
}
```

Source Codes 4.2 – function obstacle_rotation()

4.5.2 Robot's Movement

This part showcases the key functions responsible for enabling the robot's movements within its environment.

1. **Forward moving function :** function to move the robot forward.

```
void move_forward(){
  int i=1;
  while(i<=73){
     motor1.step(1);
     motor2.step(-1);
     i=i+1;
     Serial.println(i);
  }
}
```

Source Codes 4.3 – function move_forward()

2. **Backward moving function :** function to move backwards.

```
void move_Backward(){
  int i=1;
```

```
  while(i<=1264){
     motor1.step(-1);
     motor2.step(1);
     i=i+1;
  }
}
```

Source Codes 4.4 – function move_Backward()

3. **Left moving function :** function to move the robot to the left.

```
void move_left(){
  int i=1;
  while(i<=730){
     motor2.step(-1);
     motor1.step(-1);
     i=i+1;
  }
}
```

Source Codes 4.5 – function move_left()

4. **Right moving function :** function to move the robot to the right.

```
void move_Right(){
  int i=1;
  while(i<=730){
     motor2.step(1);
     motor1.step(1);
     i=i+1;
  }
}
```

Source Codes 4.6 – function move_Right()

4.5.3 Mapping

The most important function of the mapping procedure is provided.

```
# Function of building the robot's map
def draw_map(self):
            self.canvas.delete("all")
            for i in range(self.rows):
                for j in range(self.cols):
                    value = self.matrix[i][j]
                    if value>=0:
                        color = "gray" \\ undiscoverd spot
                        if value == 1:
                            color = "black" \\obstacle
                        elif value == 2:
                            color = "red" \\robot's position
                        elif value == 3:
                            color = "green" \\cleaned spot
                        elif value == 4:
                            color = "white" \\ discovered spot
```

Source Codes 4.7 – function draw_map()

4.5.4 Self-Organizing map function

Here, the self-organizing map function using the minisom library is exhibited.

```
def SOM(matrix,first_visit):
    np.random.seed(10)
    x, y = np.where(matrix == 0)
    obstacles_x, obstacles_y = np.where(matrix == 1)
    N_neuron = matrix.shape[0] * matrix.shape[0] * 5
    som = MiniSom(1, N_neuron, 2, sigma=4, learning_rate=0.1,
    neighborhood_function='gaussian', random_seed=0)
```

```
    points = np.array([x, y]).T
    obstacles = np.array([obstacles_x, obstacles_y]).T
    som.random_weights_init(points)
    last_iteration = None
    distance_traveled = None
    min_error_iteration = None
    min_error = float('inf')
    min_error_visit_order = None
    plt.figure(figsize=(10, 9))
    for i, iterations in enumerate(range(5, 61, 5)):
        som.train(points, iterations, verbose=False, random_order=False
        visit_order = np.argsort([som.winner(p)[1] for p in points])
        first_visit_index = np.where(visit_order == (first_visit - 1))[0][0]
        visit_order = np.concatenate((visit_order[first_visit_index:],
        visit_order[:first_visit_index]))

        error = som.quantization_error(points)
        if error < min_error:
            min_error = error
            min_error_iteration = iterations
            min_error_visit_order = visit_order
 x_y_coord = np.array([x[min_error_visit_order] , y[min_error_visit_order]]).T

    return min_error, min_error_visit_order, x_y_coord
```

Source Codes 4.8 – function SOM()

4.5.5 A* function

The A star function used to find the shortest path from one cell to another is provided.

```
def AStar(matrix, start, end):
    rows = len(matrix)
```

```
    cols = len(matrix[0])
    visited = [[False for _ in range(cols)] for _ in range(rows)]
    dx = [-1, 1, 0, 0]
    dy = [0, 0, -1, 1]
    queue = [(0, start, [start])]
    while queue:
        queue.sort()
        current_cost, (x, y), path = queue.pop(0)
        if (x, y) == end:
            return path

        if visited[x][y]:
            continue
        visited[x][y] = True
        for i in range(4):
            new_x = x + dx[i]
            new_y = y + dy[i]
            if is_valid_move(new_x, new_y, matrix, visited):

                new_cost = current_cost + 1
                priority = new_cost + heuristic((new_x, new_y), end)

                new_path = path + [(new_x, new_y)]
                queue.append((priority, (new_x, new_y), new_path))
    return None
```

Source Codes 4.9 – function A_star()

4.5.6 Robot's Direction

The function that directs the robot between the cells given by the A star function is as follows.

```
def direction(x,y,new_x,new_y,orientation,path):
```

```
if(orientation=="forward"):
    if(x==new_x):
        if(y+1==new_y):
            path.append("right")
            path.append("forward")
            orientation="right"
            return orientation,path
        if(y-1==new_y):
            path.append("left")
            path.append("forward")
            orientation="left"
            return orientation,path
    if(y==new_y):
        if(x-1==new_x):
            path.append("forward")
            orientation="forward"
            return orientation,path
        if(x+1==new_x):
            path.append("right")
            path.append("right")
            path.append("forward")
            orientation="bacward"
            return orientation,path
```

Source Codes 4.10 – function direction()

4.5.7 ANN Model Architecture

The ANN model's architecture used to start the robot's functioning in the most optimum moment is shown in Figure 4.18.

```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 layer_normalization_5 (Laye  (None, 8)                16
 rNormalization)

 dense_35 (Dense)            (None, 512)               4608

 dense_36 (Dense)            (None, 512)               262656

 dense_37 (Dense)            (None, 512)               262656

 dense_38 (Dense)            (None, 512)               262656

 dense_39 (Dense)            (None, 512)               262656

 dense_40 (Dense)            (None, 512)               262656

 dense_41 (Dense)            (None, 1)                 513

=================================================================
Total params: 1,318,417
Trainable params: 1,318,417
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 4.18 – ANN Model Architecture

4.6 Results and Discussion

This section showcases and discusses results obtained from both the hardware and software aspects of the proposed system.

4.6.1 Hardware Aspect

To begin, the results obtained through the hardware aspect of our work are presented.

4.6.1.1 Distance Detection Result

Figure 4.20 below shows the data gathered by the ultrasonic sensor.

```
20:34:06.550 -> Angle: 196.88  Distance: 8.79 cm
20:34:06.618 -> Angle: 199.69  Distance: 6.19 cm
20:34:06.656 -> Angle: 202.50  Distance: 6.46 cm
20:34:06.705 -> Angle: 205.31  Distance: 15.35 cm
20:34:06.743 -> Angle: 208.13  Distance: 4.42 cm
20:34:06.783 -> Angle: 210.94  Distance: 4.71 cm
20:34:06.861 -> Angle: 213.75  Distance: 6.75 cm
```

FIGURE 4.19 – Obstacles Detection

4.6.1.2 Obstacles Presentation

The Figure below 4.20 shows the result of obstacles detection in one rotation.

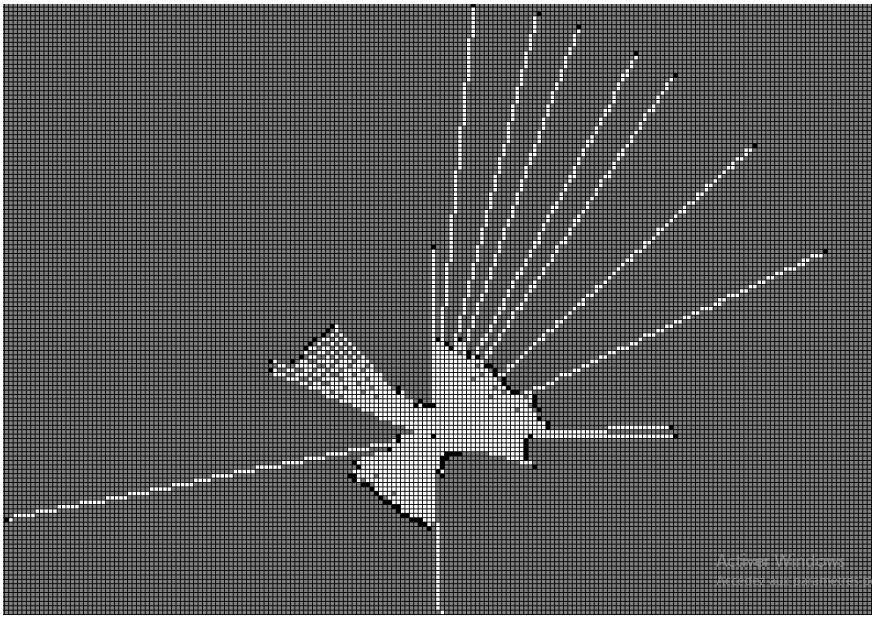

FIGURE 4.20 – Obstacles Presentation

4.6.2 Software Aspect

This section showcases and delves into the results of the mapping process, path-finding intelligent algorithms, and our decision-making model.

4.6.2.1 Result of Mapping Procedure

Figure 4.21 below represents the result of the mapping procedure.

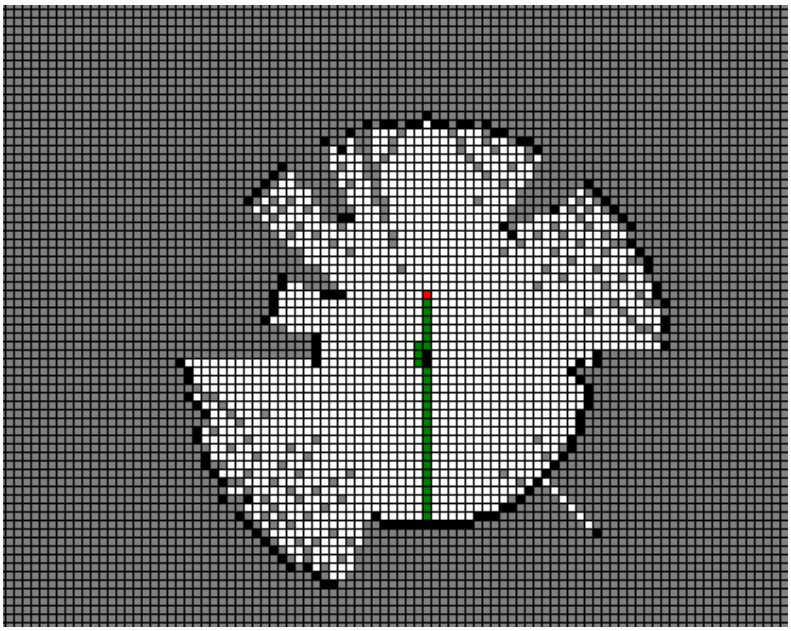

FIGURE 4.21 – Environment Map

4.6.2.2 Result of SOM Algorithm

The following Figure 4.22 presents the results gathered by several iterations of the execution of the Self-Organizing maps Algorithm with the error value of each iteration.

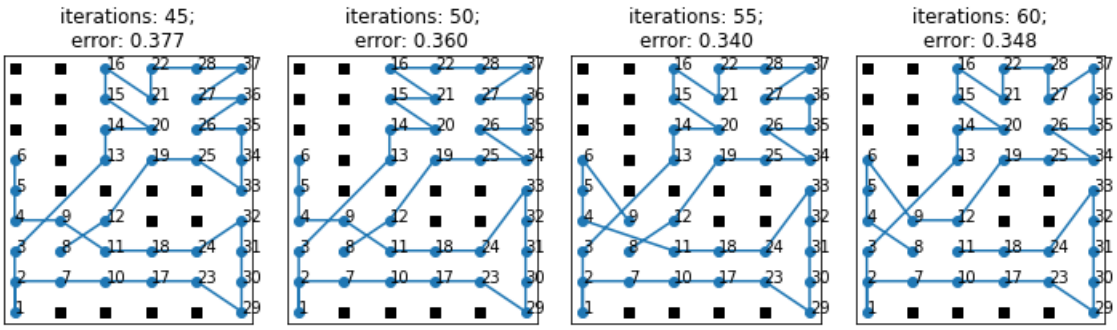

FIGURE 4.22 – Self-Organizing Maps Result

4.6.2.3 ANN Model Result

After conducting experiments using SVM and logistic regression, an ANN model was employed to enhance the inefficient obtained results.

The comparison Table 4.6.2.3 clearly indicates that the ANN model surpassed both SVM

and logistic regression in terms of accuracy, precision, recall, F1 score, and loss. These findings confirm the effectiveness of investigating the ANN model for achieving superior performance in the current classification task.

Based on the obtained results, the proposed model demonstrated better performance. It achieved an accuracy of 0.95, surpassing the LR and SVM models, which achieved a lower accuracy of 0.92. Moreover, in terms of precision, the ANN model showcased a precision of 0.86, slightly outperforming the logistic regression and SVM models with a precision of 0.85. The ANN model also excelled in the Recall function, attaining a value of 0.95, whereas the LR and SVM models achieved a lower recall of 0.92. Similarly, in the F1 Score function, the ANN model achieved a higher score of 0.92 compared to the LR and SVM models, which obtained a score of 0.88. Notably, the proposed ANN model exhibited a significantly lower loss value of 0.03, whereas the LR and SVM models rendered higher loss values of 2.62.

TABLE 4.1 – Comparison Table

	Accuracy	**Precision**	**Recall**	**F1 Score**	**loss**
LR	**0.92**	**0.85**	**0.92**	**0.88**	**2.62**
SVM	**0.92**	**0.85**	**0.92**	**0.88**	**2.62**
Proposed ANN model	**0.95**	**0.86**	**0.95**	**0.92**	**0.05**

- Figure 4.23 below shows the accuracy of the model :

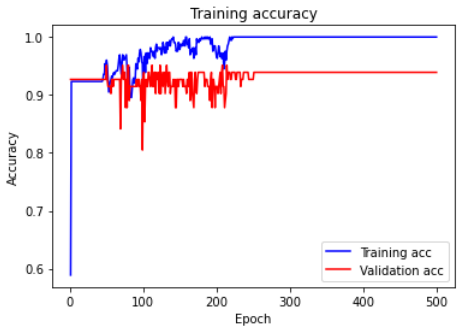

FIGURE 4.23 – Training and Validation Accuracy

- Figure 4.24 below shows the loss of the model :

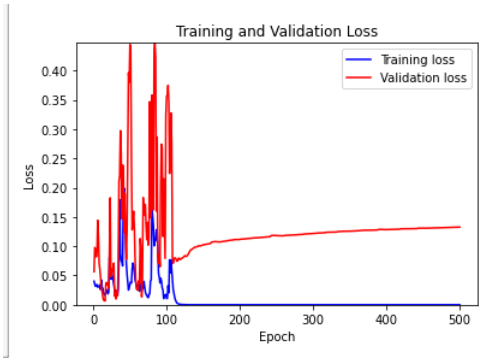

FIGURE 4.24 – Training and Validation Loss

4.6.2.4 Mobile Application

The mobile application is designed to remote control the robot using a server. As shown in Figure 4.25 it can be used to start the device in both modes (manual starting and intelligent starting), it also shows the map of the startup's location.

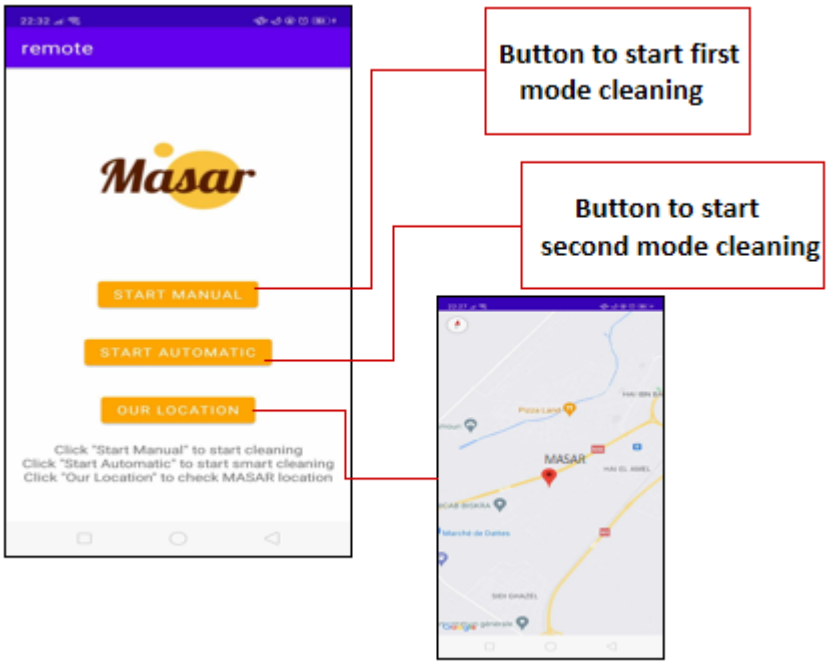

FIGURE 4.25 – Mobile Application

4.7 Conclusion

This chapter has outlined the tools, software, and hardware employed in the development of the proposed system. Furthermore, also it has demonstrated the code snippets generated for each phase of the system's architecture, along with the procedure for interconnecting electronic devices to construct a prototype of a smart floor cleaner, while also individually defining each device. Lastly, it has showcased certain outcomes achieved throughout the progression of the system's development.

General Conclusion

The rapid growth of interconnected devices and the development of technologies such as the Internet of Things, Artificial Intelligence, Robotics, and Electronics have paved the way for innovative solutions that significantly enhance the quality of our lives.

The primary objective of the project discussed in this context is to develop and implement a smart floor-cleaning robot, utilizing components such as NODEMCU microcontrollers, ultrasonic sensor and motors. This robot is designed to navigate unfamiliar environments, generate accurate maps, detect and avoid obstacles and make decisions regarding path selection. Moreover, through the integration of a PIR sensor (motion sensor) and a KY-037 sensor (sound sensor), an artificial intelligence framework based on an ANN model has been developed. This enables the robot to autonomously determine the ideal timing for operation, enhancing its efficiency and adaptability.

In this thesis, four chapters were presented, the first chapter delved in IoT, its components, architecture, protocols, advantages and challenges and finally its application domains. The second chapter, delved into robotics, its conceptual framework, navigation techniques, and the categorization of robots, with an emphasis on autonomous mobile robots, also some related works to our project were discussed. The third chapter described the system that we developed and explained the functioning of each part of it. Finally, the fourth chapter showed the implementation tools and code details, as well as a discussion of the results we obtained.

In the future, we will equip our system with both Lidar to enhance visibility and a more powerful micro-controller like the raspberry pi.

Bibliographie

[1] Faiza . and Wan Rahiman. A comprehensive study for robot navigation techniques. *Cogent Engineering*, 6 :1–25, 07 2019.

[2] Martín Abadi, Michael Isard, and Derek G Murray. A computational model for tensorflow : an introduction. In *Proceedings of the 1st acm sigplan international workshop on machine learning and programming languages*, pages 1–7, 2017.

[3] Sarah A. Al-Qaseemi, Hajer A. Almulhim, Maria F. Almulhim, and Saqib Rasool Chaudhry. Iot architecture challenges and issues : Lack of standardization. In *2016 Future Technologies Conference (FTC)*, pages 731–738, 2016.

[4] Ken Arnold, James Gosling, and David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.

[5] Omur Arslan and Daniel E Koditschek. Sensor-based reactive navigation in unknown convex sphere worlds. *The International Journal of Robotics Research*, 38(2-3) :196–223, 2019.

[6] Mouadh Bali, Abdelkamel Tari, Abdullah Almutawakel, and Okba Kazar. Smart design for resources allocation in iot application service based on multi-agent system and dcsp. *Informatica*, 44 :373–386, 10 2020.

[7] Yassine Banouar. *Gestion autonomique de la QoS au niveau middleware dans l'IoT*. Theses, Université Paul Sabatier - Toulouse III, September 2017.

[8] Malti Bansal and Priya. Performance comparison of mqtt and coap protocols in different

simulation environments. *Inventive Communication and Computational Technologies : Proceedings of ICICCT 2020*, pages 549–560, 2021.

[9] V Ramos Batista and FA Zampirolli. Optimising robotic pool-cleaning with a genetic algorithm. *Journal of Intelligent & Robotic Systems*, 95 :443–458, 2019.

[10] Oussama El Hamzaoui. *Localisation et cartographie simultanées pour un robot mobile équipé d'un laser à balayage : CoreSLAM*. PhD thesis, Paris, ENMP, 2012.

[11] Shirine El Zaatari, Mohamed Marei, Weidong Li, and Zahid Usman. Cobot programming for collaborative industrial tasks : An overview. *Robotics and Autonomous Systems*, 116 :162–180, 2019.

[12] Farbod Fahimi. *Autonomous robots*. Springer, 2009.

[13] M Umar Farooq, Muhammad Waseem, Sadia Mazhar, Anjum Khairi, and Talha Kamal. A review on internet of things (iot). *International journal of computer applications*, 113(1) :1–7, 2015.

[14] Mohamed Fezari and Ali Al Dahoud. Integrated development environment "ide" for arduino. *WSN applications*, pages 1–12, 2018.

[15] Giuseppe Fragapane, René de Koster, Fabio Sgarbossa, and Jan Ola Strandhagen. Planning and control of autonomous mobile robots for intralogistics : Literature review and research agenda. *European Journal of Operational Research*, 294(2) :405–426, 2021.

[16] David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, and Sailu Reddy. *HTTP : the definitive guide*. " O'Reilly Media, Inc.", 2002.

[17] Jens-Steffen Gutmann, Kristen Culp, Mario E Munich, and Paolo Pirjanian. The social impact of a systematic floor cleaner. In *2012 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 50–53. IEEE, 2012.

[18] Bram Hendriks, Bernt Meerbeek, Stella Boess, Steffen Pauws, and Marieke Sonneveld. Robot vacuum cleaner personality and behavior. *International Journal of Social Robotics*, 3 :187–195, 2011.

[19] Yuda Irawan, Muhardi Muhardi, Rian Ordila, and Roni Diandra. Automatic floor cleaning robot using arduino and ultrasonic sensor. *Journal of Robotics and Control (JRC)*, 2021.

[20] Uman Khalid, Muhammad Faizan Baloch, Haseeb Haider, Muhammad Usman Sardar, Muhammad Faisal Khan, Abdul Basit Zia, and Tahseen Amin Khan Qasuria. Smart floor cleaning robot (clear), 2015.

[21] Jaeseok Kim, Anand Kumar Mishra, Raffaele Limosani, Marco Scafuro, Nino Cauli, Jose Santos-Victor, Barbara Mazzolai, and Filippo Cavallo. Control strategies for cleaning robots in domestic applications : A comprehensive review. *International Journal of Advanced Robotic Systems*, 16(4) :1729881419857432, 2019.

[22] G Uday Kiran and D Vasumathi. Predicting parkinson's disease using extreme learning measure and principal component analysis based mini som. *Annals of the Romanian Society for Cell Biology*, pages 16099–16111, 2021.

[23] S Korade, Vinit Kotak, and A Durafe. A review paper on internet of things (iot) and its applications. *Int. Res. J. Eng. Technol*, 6(6) :1623–1630, 2019.

[24] Rob Kranenburg and Alex Bassi. Iot challenges. *Communications in Mobile Computing*, 1, 12 2012.

[25] Shivesh Kumar, Hendrik Wöhrle, José de Gea Fernández, Andreas Müller, and Frank Kirchner. A survey on modularity and distributivity in series-parallel hybrid robots. *Mechatronics*, 68 :102367, 2020.

[26] Tapio Levä, Oleksiy Mazhelis, and Henna Suomi. Comparing the cost-efficiency of coap and http in web of things applications. *Decision Support Systems*, 63 :23–38, 2014.

[27] Zhe Li, Gongfa Li, Ying Sun, Guozhang Jiang, Jianyi Kong, and Honghai Liu. Development of articulated robot trajectory planning. *International Journal of Computing Science and Mathematics*, 8(1) :52–60, 2017.

[28] Zhenjing Li and Lap Tam. A survey on techniques and applications of window-cleaning robots. *IEEE Access*, PP :1–1, 08 2021.

[29] Zhibin Li, Shuai Li, and Xin Luo. An overview of calibration technology of industrial robots. *IEEE/CAA Journal of Automatica Sinica*, 8(1) :23–36, 2021.

[30] Anbalagan Loganathan and Nur Syazreen Ahmad. A systematic review on recent advances in autonomous mobile robot navigation. *Engineering Science and Technology, an International Journal*, 40 :101343, 2023.

[31] Jorge E Luzuriaga, Miguel Perez, Pablo Boronat, Juan Carlos Cano, Carlos Calafate, and Pietro Manzoni. A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 931–936. IEEE, 2015.

[32] Paulo Mendes. Social-driven internet of connected objects. 03 2011.

[33] Dubravko Miljković. Brief review of self-organizing maps. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1061–1066, 2017.

[34] S.B. Niku. *Introduction to Robotics : Analysis, Control, Applications*. Wiley, 2020.

[35] Travis E Oliphant et al. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[36] Keyur Patel, Sunil Patel, P Scholar, and Carlos Salazar. Internet of things-iot : Definition, characteristics, architecture, enabling technologies, application future challenges, 05 2016.

[37] Dr. Yusuf Perwej, Kashiful Haq, Dr. Firoj Parwej, and Mumdouh M. The internet of things (iot) and its application domains. *International Journal of Computer Applications*, 182 :36–49, 04 2019.

[38] Dr. Yusuf Perwej, Kashiful Haq, Dr. Firoj Parwej, and Mumdouh M. The internet of things (iot) and its application domains. *International Journal of Computer Applications*, 182 :36–49, 04 2019.

[39] Riccardo Petrolo, V. Loscri, and Nathalie Mitton. Towards a smart city based on cloud of things. 08 2014.

[40] Why Python. Python. *Python Releases Wind*, 24, 2021.

[41] Dennis M Ritchie, Stephen C Johnson, ME Lesk, BW Kernighan, et al. The c programming language. *Bell Sys. Tech. J*, 57(6) :1991–2019, 1978.

[42] Dima Rodriguez. Tutoriel android sous android studio. 2015.

[43] Haydar Sahin and Levent Guvenc. Household robotics - autonomous devices for vacuuming and lawn mowing. *Control Systems, IEEE*, 27 :20 – 96, 05 2007.

[44] Arbia Riahi Sfar, Zied Chtourou, and Yacine Challal. A systemic and cognitive vision for iot security : A case study of military live simulation and security challenges. In *2017 International Conference on Smart, Monitored and Controlled Cities (SM2C)*, pages 101–105, 2017.

[45] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[46] Dipa Soni and Ashwin Makwana. A survey on mqtt : a protocol of internet of things (iot). In *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*, volume 20, pages 173–177, 2017.

[47] Sebastian Thrun et al. Robotic mapping : A survey. 2002.

[48] Emmanouil G Tsardoulias, A Iliakopoulou, Andreas Kargakos, and Loukas Petrou. A review of global path planning methods for occupancy grid maps regardless of obstacle density. *Journal of Intelligent & Robotic Systems*, 84 :829–858, 2016.

[49] Ivan Vaccari, Maurizio Aiello, and Enrico Cambiaso. Slowite, a novel denial of service attack affecting mqtt. *Sensors*, 20(10) :2932, 2020.

[50] Jiawei Wang, Tzu-yang Kuo, Li Li, and Andreas Zeller. Assessing and restoring reproducibility of jupyter notebooks. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, page 138–149, New York, NY, USA, 2021. Association for Computing Machinery.

[51] Zhi Yan, Simon Schreiberhuber, Georg Halmetschlager, Tom Duckett, Markus Vincze, and Nicola Bellotto. Robot perception of static and dynamic objects with an autonomous floor scrubber. *Intelligent Service Robotics*, 13(3) :403–417, 2020.

[52] S Yatmono, M Khairudin, H S Pramono, and A Asmara. Development of intelligent floor cleaning robot. *Journal of Physics : Conference Series*, 1413(1) :012014, nov 2019.