

الجمهورية الجزائرية الديمقراطية الشعبية  
Democratic and Popular Republic of Algeria  
وزارة التعليم العالي و البحث العلمي  
Ministry of Higher Education and Scientific Research  
جامعة سعد دحلب البليدة  
SAAD DAHLAB University of BLIDA  
كلية التكنولوجيا  
Faculty of Technology  
قسم الإلكترونيك  
Department of Electronics



CDTA

## Master's Thesis

Field: Electronics  
Specialisation: Embedded systems

Presented by:

**Aymen Abdelkader TAIBI**

&

**Mohamed Elhadi BAHAR**

# Development of a hardware and software architecture for an intelligent wheelchair

Proposed by:

**Ms. Yasmina AMER EL KHEDOUD**

**Mrs. Sara BOURAINE**

**Mrs. Djamila NACEUR**

Academic year: **2022-2023**

## Appreciation

---

We would like to thank first and foremost **ALLAH**, the Almighty who gave us the strength, patience and will to complete this work.

Alhamdulillah.

We are deeply thankful to **Mrs. NACEUR**, as she has been one of the most influential figures in our lives. Her guidance, wisdom, and work ethics have been instrumental in shaping our personal and professional growth. We will forever cherish the lessons and values she imparted to us.

We are immensely grateful to **Mrs. BOURAINE** for her invaluable help and guidance throughout our internship. Her expertise and support have been instrumental in this academic success.

We would really like to thank our promoter **Ms. AMER EL KHEDOUD**, for his competent advice, his availability, his support, and his patience. We are very grateful for everything you have done.

We deeply thank **the members of the jury** for their expertise and dedicated efforts in evaluating this study.

Not to forget **Mr. TIAR**, thank you for your sacrifices and support during the challenging times of this master's thesis.

Our thanks also go to all those who have contributed directly or indirectly to the realization of this modest work.

I would like to express my sincere gratitude and appreciation to:

**My beloved parents**, who have always supported me with their unconditional love, encouragement and prayers. They have been my source of inspiration and motivation in pursuing my dreams and goals.

I am deeply thankful to **my teachers**, who have imparted their knowledge, wisdom and experience to me.

I am especially grateful to **Ms. Naceur**, who has been my supervisor and mentor for this thesis. She has provided me with valuable feedback, constructive criticism and insightful suggestions that have improved the quality and clarity of my work.

Last but not least, I would like to acknowledge the contribution of my close people, especially **G.A** who have been there for me in times of joy and sorrow. They have cheered me up, comforted me and helped me in various ways.

I am truly blessed to have them in my life. This thesis is dedicated to all of them.

**BAHAR**

---

**ملخص:** يركز هذا المشروع على تطوير هندسة معمارية مادية وبرمجية لكروسي متحرك ذكي استنادًا إلى متحكم STM32 core M4. يهدف المشروع إلى تمكين التحكم اليدوي في الكروسي المتحرك باستخدام عصا تحكم لاسلكية عبر تقنية البلوتوث في وضع التحكم اليدوي، مع دمج نظام التشغيل الروبوتي (ROS) للملاحة الذاتية في وضع التحكم الذاتي. يسمح آلية اختيار الوضع المزدوج للمستخدمين بالتبديل بسهولة بين وضعي التحكم اليدوي والتحكم الذاتي. توفر واجهة البرنامج تجربة سهلة الاستخدام للتحكم اليدوي. تم دمج ROS لتمكين الكروسي المتحرك من الملاحة الذاتية في محيطه.

**كلمات المفاتيح:** نظام التشغيل الروبوتي, الملاحة الذاتية, STM32, بلوتوث.

---

**Résumé:** Ce projet se concentre sur le développement d'une architecture matérielle et logicielle pour une chaise roulante intelligente basée sur le microcontrôleur STM32 core M4. Le projet vise à permettre le contrôle manuel de la chaise roulante à l'aide d'un joystick sans fil via Bluetooth en mode manuel, tout en intégrant le système d'exploitation robotique (ROS) pour la navigation autonome en mode autonome. Un mécanisme de sélection à double mode permet aux utilisateurs de passer facilement d'un mode à l'autre. L'interface logicielle offre une expérience conviviale pour le contrôle manuel. L'intégration de ROS permet à la chaise roulante de naviguer de manière autonome dans son environnement.

**Mots clés:** ROS, Navigation, STM32, Bluetooth.

---

**Abstract:** This project focuses on the development of a hardware and software architecture for an intelligent wheelchair based on the STM32 core M4 microcontroller. The project aims to enable manual control of the wheelchair using a wireless joystick via Bluetooth in the manual mode, while incorporating the Robot Operating System (ROS) for autonomous navigation in the autonomous mode. A dual-mode selection mechanism allows users to seamlessly switch between manual and autonomous modes.

The software interface provides a user-friendly experience for manual control. Integration of ROS empowers the wheelchair to autonomously navigate its surroundings.

**Keywords:** ROS, Navigation, STM32, Bluetooth.

---

## List of acronyms and abbreviations

<i>APT</i>	<b>Advanced Packaging Tool</b>
<i>ARM</i>	<b>Advanced RISC Machines</b>
<i>AWS</i>	<b>Amazon Web Services</b>
<i>ADCs</i>	<b>Analog-to-Digital Converters</b>
<i>CDTA</i>	<b>Centre for the Development of Advanced Technologies</b>
<i>CAN</i>	<b>Controller Area Network</b>
<i>DACs</i>	<b>Digital-to-Analog Converters</b>
<i>EEPROM</i>	<b>Electrically Erasable Programmable Read-Only Memory</b>
<i>GPS</i>	<b>Global Positioning System</b>
<i>GUIs</i>	<b>Graphical User Interfaces</b>
<i>HMI</i>	<b>Human Machine-Interface</b>
<i>IWDG</i>	<b>Independent Watchdog</b>
<i>I/O</i>	<b>Input/Output</b>
<i>ICR</i>	<b>Instant Centre of Rotation</b>
<i>I2C</i>	<b>Inter-Integrated Circuit</b>
<i>KB</i>	<b>Kilobyte</b>
<i>LIDAR</i>	<b>Light Detection and Ranging</b>
<i>LXQt</i>	<b>Lightweight Qt Desktop Environment</b>

<i>LXDE</i>	<b>Lightweight X11 Desktop Environment</b>
<i>LED</i>	<b>Light-Emitting Diode</b>
<i>LTS</i>	<b>Long-Term Support</b>
<i>MAC</i>	<b>Media Access Control</b>
<i>MHz</i>	<b>Mega Hertz</b>
<i>NASA</i>	<b>National Aeronautics and Space Administration</b>
<i>PC</i>	<b>Personal Computer</b>
<i>PID</i>	<b>Proportional-Integral-Derivative</b>
<i>PWM</i>	<b>Pulse Width Modulation</b>
<i>QEI</i>	<b>Quadrature Encoder Interface</b>
<i>RAM</i>	<b>Random-Access Memory</b>
<i>RTC</i>	<b>Real-Time Clock</b>
<i>RISC</i>	<b>Reduced Instruction Set Computer</b>
<i>RPM</i>	<b>Revolution Per Minute</b>
<i>ROS</i>	<b>Robot Operating System</b>
<i>SPI</i>	<b>Serial Peripheral Interface</b>
<i>SRAM</i>	<b>Static Random-Access Memory</b>
<i>USB</i>	<b>Universal Serial Bus</b>

# Table of content

General Introduction .....	13
----------------------------	----

## Chapter 01: Generalities

Introduction .....	15
1.1 Traditional Wheelchairs .....	16
1.1.1 Overview .....	16
1.1.2 Types of Traditional Wheelchairs .....	17
1.1.3 Advantages and Disadvantages .....	18
1.1.4 Challenges Faced by Users .....	19
1.2 Mobile robots .....	19
1.2.1 Evolution of robotics .....	19
1.2.2 Definition .....	20
1.2.3 Mobile robots architectures .....	20
1.2.4 Advantages .....	21
1.2.5 Popular Open - Source mobile robots .....	22
1.3 Intelligent wheelchair .....	24
1.3.1 Evolution .....	24
1.3.2 Features and Capabilities .....	25
1.3.3 Benefits .....	25
1.3.4 Human-Machine Interface (HMI) .....	26
Conclusion .....	27

## Chapter 02: System Design

Introduction .....	28
2.1 Hardware .....	29
2.1.1 Microcontrollers .....	29
2.1.2 Bluetooth module (HC-06) .....	31
2.1.3 Bluetooth module (HC-05) .....	31
2.1.4 Motor driver .....	32
2.1.5 Brushless DC motor .....	32
2.1.6 DC-DC converter (TMR 9-4819WI) .....	33
2.2 Software .....	34
2.2.1 ROS (Robot Operating System) .....	34
2.2.2 UBUNTU (LINUX) .....	36
2.3 Deployment Scenario (Circuit) .....	37
2.3.1 Block diagram .....	37
2.3.2 Full electric schematic .....	41
Conclusion .....	41

## Chapter 03: From concept to reality

Introduction .....	42
3.1 The robotic system used .....	43
3.2 The kinematic model .....	43
3.3 Step-by-Step Workflow .....	44
3.4 Supply voltage .....	48
3.5 PID results .....	49



3.6 Implementation on the wheelchair	49
3.7 Wheelchair in motion	52
Conclusion	54
General Conclusion	55
Annexes	56
Bibliography	69

## List of figures

Figure 1.1. Lightweight folding manual wheelchair	17
Figure 1.2. GTM Endeavour rigid frame Wheelchair	18
Figure 1.3. TurtleBot 4	23
Figure 1.4. ROSbot	23
Figure 1.5. Husky	24
Figure 2.1. The AT91SAM3X8E pinout	30
Figure 2.2. STM32 NUCLEO-F446RE pinout	30
Figure 2.3. HC-06 Bluetooth module	31
Figure 2.4. HC-05 Bluetooth module	31
Figure 2.5. The JYQD-V7.3E3 motor driver	32
Figure 2.6. Brushless motor's encoder	33
Figure 2.7. TMR 9-4819WI Converter	33
Figure 2.8. ROS applications	35
Figure 2.9. Communicating ROS nodes on Ubuntu	36
Figure 2.10. Full block diagram	37
Figure 2.11. Control block process	38
Figure 2.12. Wireless joystick	40
Figure 2.13. Full design schematic	41
Figure 3.1. Wheelchair dimensions	43
Figure 3.2. Function executed during interruption	45
Figure 3.3. Step response ZIEGLER - NICHOLS method	46
Figure 3.4. Step-by-Step Workflow	48
Figure 3.5. Supply conversion	48

Figure 3.6. Creating the connection between ros nodes	50
Figure 3.7. Example of a base scan topic data using the command “rostopic echo ...	50
Figure 3.8. Example on how to publish a message via ROS publisher	51
Figure 3.9. Example on how to read a message via ROS subscriber	51
Figure 3.10. Platform testing	52
Figure 3.11. Start of the test	52
Figure 3.12. Wheelchair moving forward	53
Figure 3.13. Wheelchair turning left	53

## **Annexes**

Figure A-4.1. The JYQD-V7.3E3 wiring diagram	64
Figure A-6.1. Outline Dimensions	66

## List of tables

Table 2.1. Microcontrollers feature comparison	29
Table 2.2. Motor driver - Motor connections (Power)	39
Table 2.3. Motor driver - Motor connections (HALL)	39
Table 2.4. Left motor driver - Microcontroller	39
Table 2.5. Right motor driver - Microcontroller	40
Table 3.1. PID parameters	47

## Annexes

Table A-2.1. HC-06 Bluetooth module pinout	62
Table A-3.1. HC-05 Bluetooth module pinout	63
Table A-4.1. The JYQD-V7.3E3 control pinout	63
Table A-4.2. The JYQD-V7.3E3 power pinout	64
Table A-4.3. The JYQD-V7.3E3 hall sensor pinout	64
Table A-5.1. Brushless motor pinout	65
Table A-6.1. TMR 9-4819WI pinout	65

## General Introduction

In today's society, accessibility and independence for individuals with mobility impairments are crucial aspects that directly impact their quality of life. Traditional wheelchairs have played a significant role in providing mobility assistance, but advancements in technology have paved the way for innovative solutions. However, many existing wheelchair designs still lack the capability to seamlessly integrate hardware and software components to create a truly intelligent and user-friendly experience.

The problem lies in the limited options available for individuals who require mobility assistance. Traditional manual wheelchairs require physical effort and may not provide the desired level of freedom and autonomy. Additionally, although available, they often lack the robustness and reliability needed for everyday use. This creates a significant challenge for individuals with mobility impairments, restricting their ability to navigate their surroundings effectively.

To address this problem, our project aims to design and develop an intelligent wheelchair that can be operated both manually and autonomously. By combining cutting-edge hardware and software technologies, our project aims to bridge the gap between conventional wheelchairs and advanced assistive devices, offering a more intuitive and personalised mobility solution.

Our proposed solution involves the integration of a sophisticated hardware system and communication modules. This hardware infrastructure will work in tandem with a robust software framework. By integrating these components seamlessly, we intend to create a wheelchair that can adapt to the user's specific needs and preferences, allowing for effortless and intuitive control.

The development of this intelligent wheelchair will have far-reaching benefits for individuals with mobility impairments. It will provide them with a greater sense of independence, enabling them to navigate their environment more efficiently and comfortably.

In conclusion, the development of a hardware and software architecture for an intelligent wheelchair presents an exciting opportunity to revolutionise mobility assistance for

individuals with mobility impairments. Through the seamless integration of cutting-edge technologies, our project aims to provide an innovative solution that enhances accessibility, independence, and overall quality of life for wheelchair users.

The first chapter defines the generalities of robots generally and wheelchair robots particularly. We will provide an overview of traditional manual wheelchairs, discussing their types and inherent limitations. We then will explore the evolution of robotics, focusing on mobile robots and their various architectures. The advantages and features of mobile robots were highlighted to showcase their versatility and applicability in different industries. Finally, we will delve into the concept of intelligent wheelchairs, discussing their evolution, features, advantages, and benefits for users with mobility impairments.

In the second chapter, we will delve into the world of hardware and software components that constitute the foundation of our intelligent wheelchair project. The hardware components encompass the mechanical and electrical elements that contribute to the physical structure and locomotion capabilities of the wheelchair. Additionally, the software components encompass the algorithms, operating systems, and user interfaces that drive the intelligent features and allow seamless interactions between the user and the wheelchair.

In the third chapter, we will discuss the details of the implementation process, by turning our focus on the practical aspects of the project, embarking on the schematics and the connections that made this project feasible.

## Introduction

In today's rapidly advancing technological landscape, the field of assistive technologies has garnered significant attention, particularly in the realm of mobility assistance for individuals with limited physical abilities. The development of intelligent wheelchairs represents a promising avenue for addressing the limitations of traditional manual wheelchairs and providing individuals with enhanced mobility, independence, and accessibility.

This chapter serves as a foundational introduction to the intelligent wheelchair project, aiming to establish a context and provide general information related to the project's objectives and significance.

To begin, we will provide an overview of traditional manual wheelchairs. We will discuss their types. However, it is important to highlight the limitations inherent in these traditional wheelchairs. Factors such as physical exertion required for operation and a lack of adaptability to different user needs pose significant challenges and restrict the users' full potential.

In addition, we will examine the advancements in assistive technologies that have revolutionised the field of mobility assistance. The emergence of intelligent wheelchairs has introduced a new paradigm, integrating cutting-edge hardware and software technologies to create more intuitive and user-friendly mobility solutions. We will explore existing intelligent wheelchair solutions and highlight their key features, emphasising how they address the limitations of traditional wheelchairs and enhance the mobility experience for individuals with mobility impairments.

Finally, we will discuss some about the intelligent wheelchair, its evolution, its features and its advantages allowing us to have the full understanding on how an intelligent wheelchair can make people's lives easier.

This chapter aims to establish a solid groundwork for the forthcoming exploration of the intelligent wheelchair project by delving into its overarching concepts. By comprehensively grasping the hardware and software architecture, evaluation and user testing, as well as other pivotal aspects, we can pave the way for the development of an intelligent wheelchair. This transformative aim seeks to empower individuals with limited mobility, significantly augmenting their quality of life. With a clear understanding of the problem, objectives, and profound significance, we embark on a journey towards creating an intelligent wheelchair solution.

## **1.1 Traditional Wheelchairs**

### **1.1.1 Overview**

Traditional manual wheelchairs have long been the primary means of mobility for individuals with limited physical abilities. These wheelchairs consist of essential components such as the frame, wheels, seating system, and propulsion mechanisms. The frame provides structural support and stability, while the wheels allow for movement and manoeuvrability. The seating system includes features like a seat cushion, backrest, and armrests to provide comfort and support.

In terms of operation, traditional manual wheelchairs rely on the push-rim propulsion technique. The user propels the wheelchair forward by grasping the push rims attached to the rear wheels and pushing them in a circular motion. This action transfers the user's upper body strength into forward movement. [1]

However, traditional manual wheelchairs come with inherent limitations. One of the significant challenges is the physical effort required for propulsion. Users often experience fatigue and strain from prolonged self-propulsion, especially in situations that require covering long distances or navigating inclines. This physical demand can restrict the user's independence and limit their ability to explore their surroundings freely. [1]

### **1.1.2 Types of Traditional Wheelchairs**



Several types of traditional wheelchairs are available to cater to different user requirements. Folding wheelchairs are popular due to their compactness and ease of transportation. These wheelchairs can be folded or disassembled for storage or travel purposes [2], offering convenience and portability as shown in **Figure 1.1**.

Lightweight wheelchairs are designed to reduce the overall weight of the wheelchair, enhancing manoeuvrability and ease of handling. They often incorporate materials such as aluminium or titanium to achieve a lighter frame without compromising structural integrity. [2]

Rigid-frame wheelchairs provide increased stability and durability compared to folding wheelchairs. They feature a solid frame construction, reducing flex and allowing for efficient energy transfer during propulsion. Rigid-frame wheelchairs are known for their responsiveness and manoeuvrability [2], The **Figure 1.2** shows this type of wheelchairs.



**Figure 1.1.** Lightweight folding manual wheelchair. [3]



**Figure 1.2.** GTM Endeavour rigid frame Wheelchair. [4]

### **1.1.3 Advantages and Disadvantages**

Traditional wheelchairs offer certain advantages that have made them widely used in the mobility assistance sector. Firstly, they are relatively affordable compared to more advanced assistive devices. This affordability makes them accessible to a broader range of individuals with limited financial resources. [5]

Secondly, traditional wheelchairs are simple in design and operation, requiring minimal maintenance. This simplicity ensures reliability and ease of use for the user, even in resource-constrained environments. [5]

However, traditional wheelchairs also have their disadvantages. Limited manoeuvrability is a common challenge, especially in tight spaces or crowded areas. The larger turning radius and lack of specialised features for complex environments can restrict the user's ability to navigate freely. [5]

Another drawback is the physical strain placed on the user's upper body when self-propelling the wheelchair. Prolonged exertion can lead to fatigue, muscle strains,

and potential long-term health issues. Users may require assistance or experience limitations in performing certain activities that demand higher physical effort. [5]

#### **1.1.4 Challenges Faced by Users**

Individuals using traditional wheelchairs encounter various challenges in their daily lives. One significant challenge is navigating steep inclines or ramps. The physical effort required to propel the wheelchair uphill can be extremely demanding and may pose safety risks. Likewise, descending slopes can be equally challenging due to the need for controlled braking and stability. [6]

Traversing obstacles such as curbs, uneven terrain, or thresholds can be problematic with traditional wheelchairs. The smaller wheel size and lack of suspension systems make it difficult to absorb shocks and maintain stability. This can lead to discomfort, jarring movements, and potential accidents. [6]

Operating in crowded spaces or environments with tight corners and narrow pathways can also be a challenge. The larger turning radius of traditional wheelchairs makes manoeuvring around obstacles or through congested areas more cumbersome. [6]

## **1.2 Mobile robots**

### **1.2.1 Evolution of robotics**

The roots of robotics can be traced back to ancient times when automata, mechanical devices that imitated human or animal movements, were created. However, significant advancements in robotics began in the 20th century. In 1954, George Devol and Joseph Engelberger developed the first industrial robot, the Unimate, which revolutionised manufacturing processes by automating repetitive tasks in factories. [7]

Since then, robotics has experienced exponential growth. The advent of microprocessors and advanced control systems in the 1970s paved the way for more sophisticated and intelligent robots. The integration of artificial intelligence and machine learning in the 21st century has further propelled the field, enabling robots to

learn, adapt, and interact with humans and their environment in more intuitive ways.[7]

Today, robotics encompasses a wide range of applications. From autonomous drones and self-driving cars to robotic surgical systems and humanoid robots, the field has expanded into diverse areas, each with its unique set of challenges and opportunities.[7]

### 1.2.2 Definition

Mobile robots are autonomous or semi-autonomous robotic systems capable of navigating and operating in dynamic and unstructured environments. These robots are equipped with mechanisms such as wheels, tracks, or legs that allow them to move and traverse various terrains. They incorporate onboard sensors, such as cameras, LiDAR, and inertial sensors, to perceive and understand their surroundings, enabling them to plan and execute actions effectively. Mobile robots are designed to perform a wide range of tasks, including exploration, surveillance, transportation, delivery, and inspection, with the ability to adapt to changing environments and interact with objects and humans in their vicinity. Their mobility and autonomy make them versatile tools in various industries, such as logistics, agriculture, healthcare, and search and rescue operations.

### 1.2.3 Mobile robots architectures

There are several different types of mobile robot architectures, each with its own unique design and features. Some of the most common mobile robot architectures are:

- **Differential Drive Robot:** This type of robot has two wheels that rotate independently, allowing it to move forward, backward, and turn in place.
  - **Examples:** Roomba vacuum cleaning robot; Turtlebot mobile robot platform. [8]
- **Holonomic Robot:** A holonomic robot has wheels that can rotate independently in any direction, allowing for omnidirectional movement. [8]
  - **Examples:** Omniwheel robot; Mecanum wheel robot.

- **Tracked Robot:** This type of robot uses tracks instead of wheels, which provides more stability and the ability to traverse rough terrain. [8]
  - **Examples:** Clearpath Robotics' Husky mobile robot; Boston Dynamics' Spot robot.
- **Legged Robot:** Legged robots have legs instead of wheels, which provides them with the ability to climb stairs and navigate uneven terrain. [8]
  - **Examples:** Boston Dynamics' Atlas robot; ANYmal robot by ANYbotics.
- **Wheeled Robot:** Wheeled robots have wheels, which can be either omnidirectional or unidirectional, providing them with a simple and efficient way to move around. [8]
  - **Examples:** Autonomous delivery robots, such as the ones made by Starship Technologies and Kiwibot; Mars rovers, such as NASA's Curiosity and Perseverance rovers.
- **Swarm Robot:** Swarm robots are designed to work together in a group, where each robot performs a specific task, and collectively they achieve a more complex goal. [8]
  - **Examples:** Kilobots, a low-cost, modular robot designed for large-scale swarm robotics research; Harvard University's TERMES robots, designed for cooperative construction.
- **Humanoid Robot:** A humanoid robot is designed to look like a human, with arms, legs, and a head, and it can perform a wide range of human-like movements. [8]
  - **Examples:** Boston Dynamics' Atlas robot; Hanson Robotics' Sophia robot.

#### 1.2.4 Advantages

- **Versatility:** Mobile robots are designed to operate in various environments and perform a wide range of tasks. They can navigate through complex terrains, move in confined spaces, and adapt to different operating conditions, making them versatile tools for diverse applications.
- **Automation and Efficiency:** Mobile robots can automate repetitive and labour-intensive tasks, leading to increased efficiency and productivity. They can work autonomously, reducing the need for human intervention and freeing up human resources for more complex or critical tasks.

- **Improved Safety:** Mobile robots can be deployed in hazardous or dangerous environments, mitigating risks to human workers. They can perform tasks that are physically demanding, involve exposure to toxic substances, or require working at heights, ensuring the safety and well-being of human operators.
- **Enhanced Accuracy and Precision:** Mobile robots are equipped with sensors and advanced control systems that enable them to perform tasks with high accuracy and precision. They can navigate with precision, manipulate objects with dexterity, and execute actions consistently, minimising errors and improving overall task quality.
- **Continuous Operation:** Mobile robots can operate continuously without the need for breaks or rest, increasing productivity and operational uptime. They can work around the clock, ensuring uninterrupted performance and timely completion of tasks.

### 1.2.5 Popular Open - Source mobile robots

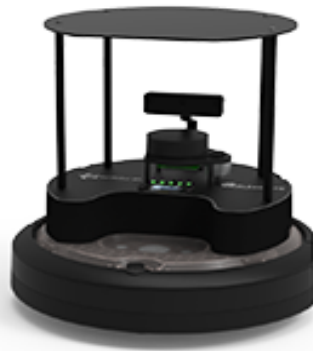
Open source mobile robots are robots that are developed using open source hardware and software components. These robots are designed with the intention of allowing developers, researchers, and enthusiasts to access the hardware design, software code, and resources needed to modify, customise, and build upon the existing robot platform.

Open-source mobile robots provide an accessible and collaborative platform for the robotics community, fostering innovation, knowledge sharing, and collaborative development. They often come with a comprehensive set of documentation, tutorials, and community support, enabling users to understand and utilise the robot's capabilities effectively.

There are many open source mobile robots available, with varying capabilities and features. Some of the most popular ones include:

- **TurtleBot:** TurtleBot is a widely recognized open-source mobile robot platform that was developed as a collaboration between Willow Garage and Clearpath Robotics. It is designed for education, research, and prototyping purposes.

TurtleBot combines a mobile base with a range of sensors, including a 3D camera and a laser scanner, and is compatible with the Robot Operating System (ROS). [9]



**Figure 1.3.** TurtleBot 4. [9]

- **ROSbot:** ROSbot is a mobile robot platform specifically designed for working with the Robot Operating System (ROS). It offers a robust hardware setup, including differential drive wheels, a variety of sensors (e.g., laser scanner, inertial measurement unit), and an onboard computer. ROSbot allows users to leverage the capabilities of ROS for perception, mapping, navigation, and other robotic tasks. [10]



**Figure 1.4.** ROSbot. [10]

- **Husky:** Husky is an open-source robotic development platform offered by Clearpath Robotics. It is a rugged and versatile mobile robot designed for research and outdoor applications. Husky combines a robust chassis, high-torque motors, and a variety of sensors (e.g., LIDAR, GPS) to navigate and interact with its environment. It is compatible with ROS and provides a powerful platform for robotics development. [11]



**Figure 1.5.** Husky. [11]

## **1.3 Intelligent wheelchair**

Intelligent wheelchairs represent a significant advancement in mobility technology, designed to provide enhanced independence and accessibility for individuals with mobility impairments. These technologically advanced devices are equipped with a range of features and capabilities that go beyond traditional wheelchairs, empowering users to navigate their environments more efficiently and comfortably. [12]

### **1.3.1 Evolution**

Over the years, wheelchair technology has undergone remarkable advancements, leading to the development of intelligent wheelchairs. These advancements have been driven by a growing recognition of the importance of mobility and independence for individuals with disabilities. The integration of



advanced electronics, sensors, and computing capabilities has revolutionised the functionality, control, and safety of wheelchairs.

### 1.3.2 Features and Capabilities

Intelligent wheelchairs incorporate a range of features and capabilities that set them apart from traditional wheelchairs. These include:

- **Automated Propulsion:** Intelligent wheelchairs can be powered electrically, eliminating the need for manual propulsion. [12]
- **Obstacle Detection and Avoidance:** Built-in sensors and perception systems enable the wheelchair to detect and navigate around obstacles in its path. [12]
- **Advanced Control Systems:** Intelligent wheelchairs offer sophisticated control mechanisms, such as joystick control, voice commands, or gesture recognition, allowing users to operate the wheelchair intuitively. [12]
- **Customization Options:** The ability to customise seating positions, backrests, and other parameters ensures optimal comfort and support for individual users. [12]

### 1.3.3 Benefits

Intelligent wheelchairs provide numerous benefits to individuals with mobility impairments, including:

- **Enhanced Mobility:** Intelligent wheelchairs offer improved manoeuvrability, allowing users to navigate tight spaces, crowded areas, and various terrains with greater ease. [12]
- **Increased Independence:** By reducing dependence on assistance from others, intelligent wheelchairs empower users to carry out daily activities independently, enhancing their self-confidence and quality of life. [12]

- **Accessibility:** The advanced capabilities of intelligent wheelchairs enable users to access a wider range of environments and facilities that may have previously been challenging or inaccessible. [12]
- **Safety and Stability:** Intelligent wheelchairs incorporate stability features, such as anti-tip mechanisms and adaptive control systems, ensuring a safer and more stable mobility experience. [12]

#### 1.3.4 Human-Machine Interface (HMI)

HMI refers to the system or interface that enables communication and interaction between humans and machines. It encompasses the hardware and software components that allow users to control, monitor, and receive feedback from machines or computer systems. [13]

In the context of intelligent wheelchairs, the HMI is the interface through which individuals with mobility impairments interact with and control the wheelchair. It provides a means for users to convey their commands and preferences to the wheelchair and receive feedback about its status and actions. [13]

The HMI in intelligent wheelchairs can take various forms depending on the design and capabilities of the wheelchair. [13]

- **Joystick Control:** Traditional wheelchair control often involves a joystick that the user can manipulate to steer the wheelchair in different directions. The joystick translates the user's movements into commands that control the wheelchair's motors for propulsion.
- **Voice Commands:** Voice recognition technology allows users to control the wheelchair by speaking specific commands or instructions. The wheelchair's software interprets the voice commands and carries out the corresponding actions, such as moving forward, turning, or stopping.
- **Gesture Recognition:** Some intelligent wheelchairs incorporate gesture recognition technology, where the user can perform specific hand or body movements that are

detected by sensors. These gestures are then translated into wheelchair commands, providing an alternative control method for individuals with limited hand dexterity.

- **Touchscreen Interfaces:** Touchscreens or graphical user interfaces (GUIs) can be employed to provide a visual interface for controlling and monitoring the wheelchair. Users can interact with the touchscreen by tapping or swiping on specific icons or buttons to initiate actions or adjust settings.

## **Conclusion**

This chapter provided a foundational introduction to our project, establishing the context and highlighting its objectives and significance. It discussed the limitations of traditional manual wheelchairs, it also explored the advancements in assistive technologies and how intelligent wheelchairs address these limitations.

Furthermore, the chapter delved into the evolution, features, and advantages of intelligent wheelchairs, showcasing their ability to make people's lives easier. The chapter also touched upon the importance of the human-machine interface (HMI) in intelligent wheelchairs.

In the next chapter, we will delve into a small overview of the hardware and software components that form the foundation of our intelligent wheelchair project, exploring also the schematic and the work steps (circuit, wiring...etc).

## Introduction

This chapter focuses on the hardware and software components essential for the development of the system discussed in this thesis. The hardware section explores microcontrollers, Bluetooth modules, motor drivers, brushless DC motors, encoders, and DC-DC converters. On the other hand, the software section introduces the Robot Operating System (ROS) and the Linux-based Ubuntu operating system.

By detailing the components chosen for our project, we aim to provide a comprehensive understanding of the underlying technologies that power our intelligent wheelchair's performance. Furthermore, we will examine the integration of these components, showcasing how they work in harmony to deliver a safe, efficient, and user-centric mobility solution.

With an overview of the hardware and software components covered in this chapter, we venture into the subsequent sections, which delve into the design and deployment scenario of the system. This encompasses a detailed exploration of the control block, power block, joystick block, and their respective functionalities. These blocks play crucial roles in the operation and control of the system, enabling seamless interaction and control between various components. By comprehending the intricate design and functionality of these blocks, we gain a holistic understanding of the system's architecture and operation.

In conclusion, this chapter serves as a comprehensive examination of the essential hardware and software components that form the foundation of the system presented in this thesis. By delving into the intricacies of microcontrollers, Bluetooth modules, motor drivers, brushless DC motors, encoders, DC-DC converters, ROS, and Ubuntu, we aim to provide a thorough understanding of the underlying technologies and their integration within the system. This knowledge forms the basis for the subsequent exploration of the system's design and deployment, ensuring a comprehensive analysis of the intelligent wheelchair's functionality and performance.

## 2.1 Hardware

### 2.1.1 Microcontrollers

During our research, we extensively explored various families of microcontrollers and their corresponding development boards, thoroughly examining their features, specifications, and capabilities.

After conducting our research, we have narrowed down our options to two boards: the AT91SAM3X8E (Arduino DUE) and the STM32 NUCLEO-F446RE.

These boards have distinct features and capabilities, making the decision a crucial one.

#### ❖ Comparative table

Microcontroller	Processor	Memory	Clock speed	Peripherals	Price
AT91SAM3X8E	ARM Cortex-M3	- Flash memory: 512 KB - SRAM: 96 KB - EEPROM: 4 KB	Up to 84 MHz	USB: 1 CAN: 1 I2C: 2 SPI: 3 UART: 4 USART: 4 ADC: 12 DAC: 2 PWM: 12 GPIO: 54	\$40
STM32 NUCLEO-F446RE	ARM Cortex-M4	- Flash memory: 512 KB - SRAM: 128 KB	Up to 180 MHz	USB: 1 CAN: 2 I2C: 3 SPI: 4 UART: 2 USART: 4 ADC: 3 DAC: 2 PWM: 17 GPIO: 64	\$25

**Table 2.1.** Microcontrollers feature comparison. [14] [15]

N.B. From this comparative table, we think that our choice is clear. The STM32 NUCLEO-F446RE board which is based on the ARM Cortex-M4 processor.

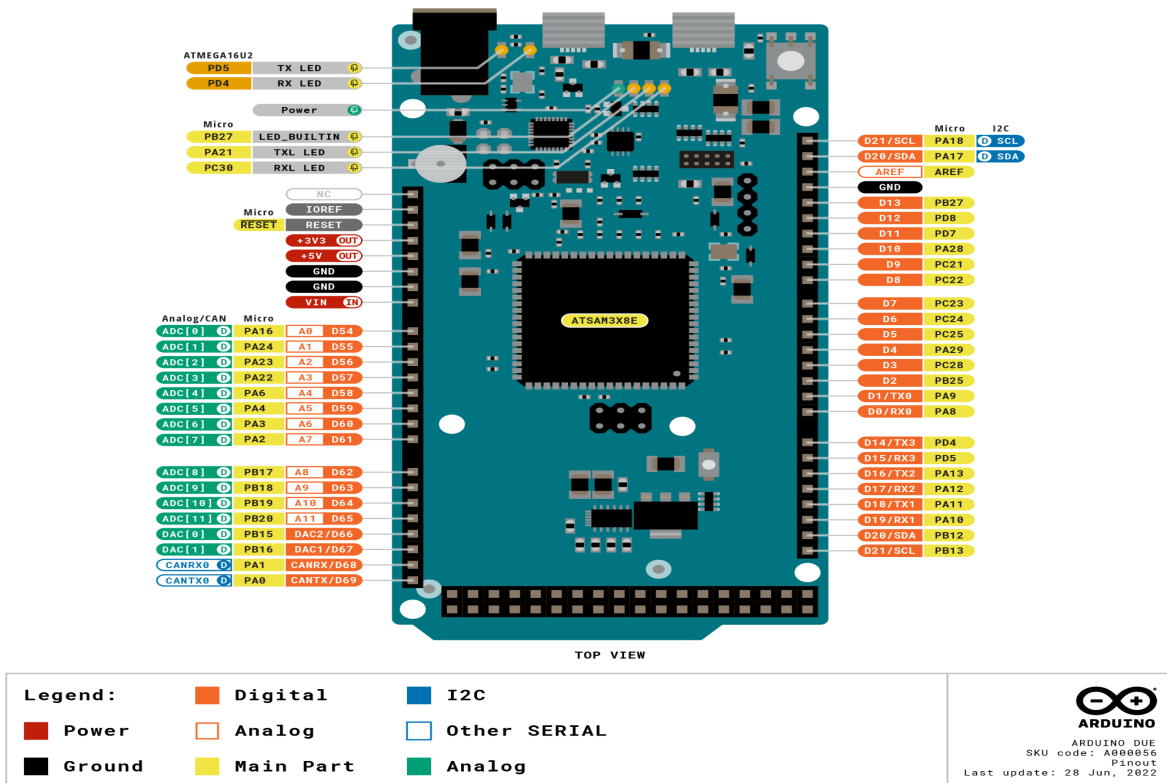


Figure 2.1. The AT91SAM3X8E pinout. [14]

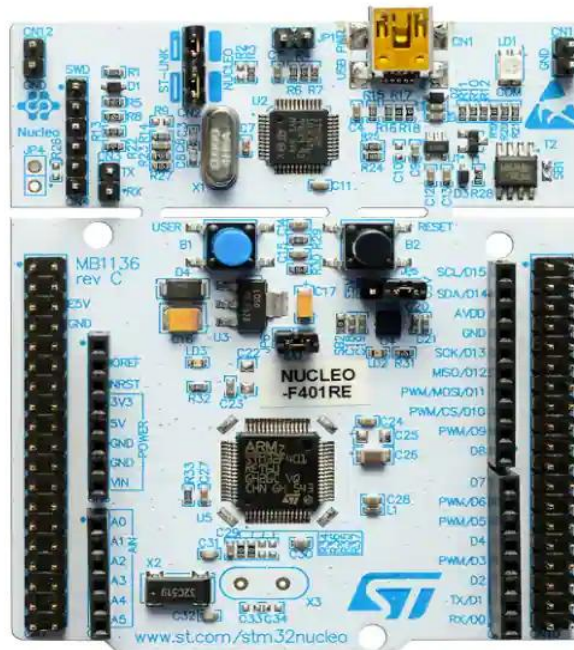


Figure 2.2. STM32 NUCLEO-F446RE pinout. [15]

### 2.1.2 Bluetooth module (HC-06)

The HC-06 module is primarily designed as a Bluetooth slave module, which means it can be paired with a Bluetooth master device (such as a smartphone, computer, or another Bluetooth-enabled microcontroller) to establish a wireless connection. [16]

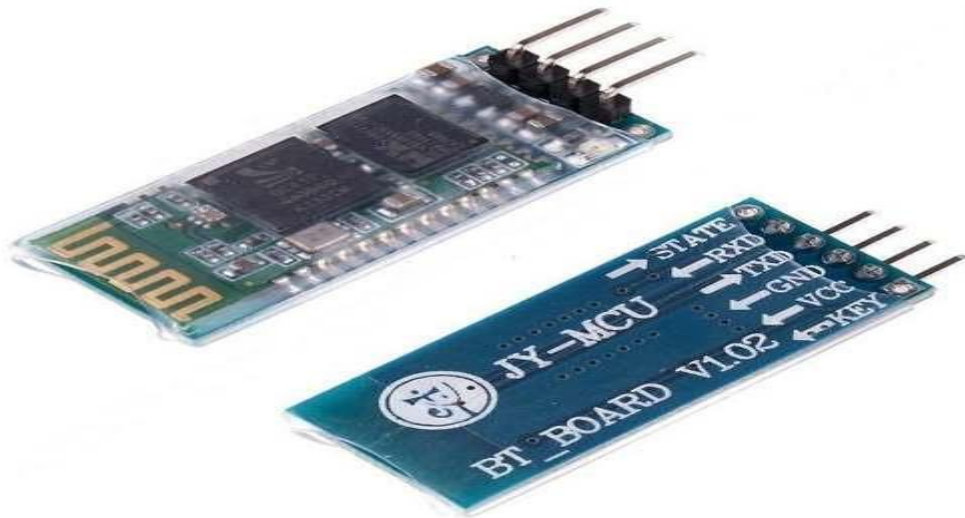


Figure 2.3. HC-06 Bluetooth module. [16]

### 2.1.3 Bluetooth HC-05

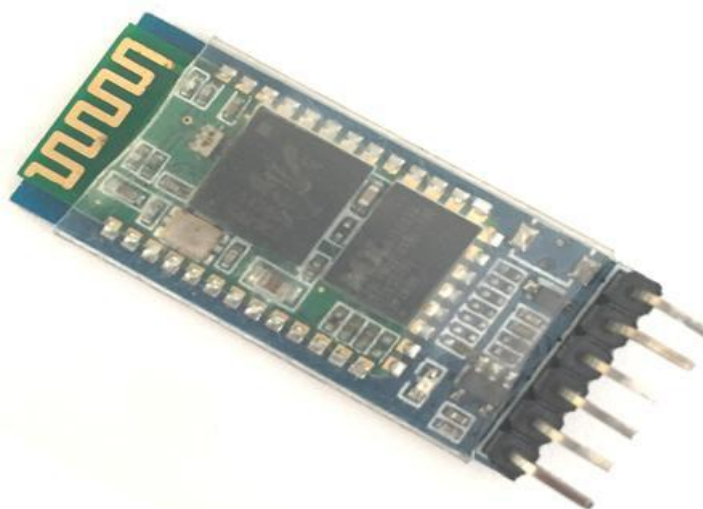
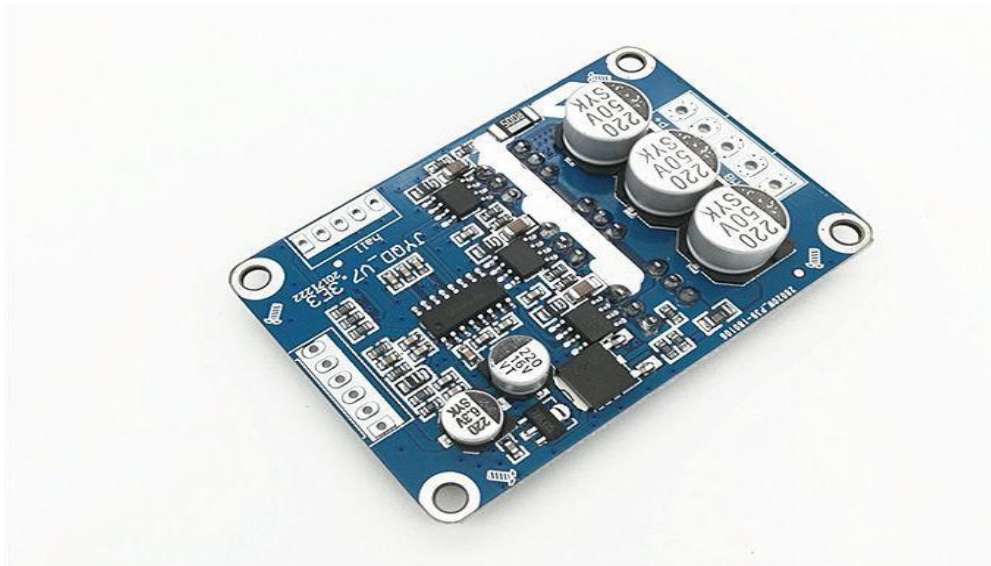


Figure 2.4. HC-05 Bluetooth module. [17]

### 2.1.4 Motor driver

The JYQD-V7.3E3 is a specific variant of a motor driver designed for controlling brushless motors. Brushless motors are commonly used in various applications such as robotics, drones, electric vehicles, and industrial machinery.

We used the **JYQD-V7.3E3** which plays a crucial role in controlling the operation of brushless motors by converting electrical signals from a control system into the appropriate current and voltage levels required by the motor. They provide the necessary power and control signals to drive the motor's three phases efficiently.



**Figure 2.5.** The JYQD-V7.3E3 motor driver. [18]

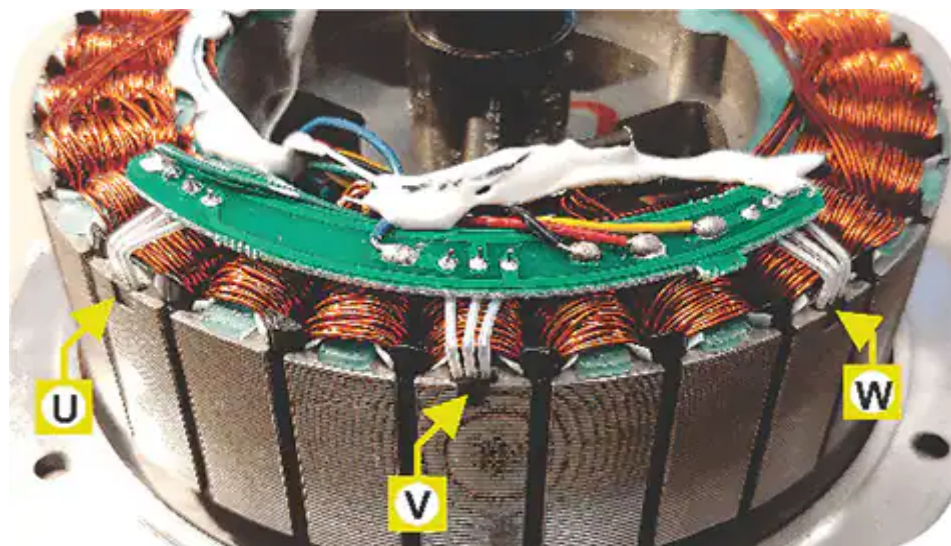
### 2.1.5 Brushless DC motor

A BLDC is a brushless DC motor, which is a type of electric motor that uses permanent magnets instead of brushes and commutators to rotate the rotor. A BLDC has three phases: A, B and C, which are connected to an electronic speed controller (ESC) that controls the current flow through the windings. A BLDC also has Hall effect sensors that detect the position of the rotor and provide feedback to the ESC. [19]



## □ Encoder

An encoder is a device that measures the position and speed of a BLDC motor. It can be optical, magnetic, or capacitive, depending on the principle of operation. An encoder can help to improve the performance and efficiency of a BLDC motor by providing feedback to the controller. [20]



**Figure 2.6.** Brushless motor's encoder. [20]

### 2.1.6 DC-DC converter (TMR 9-4819WI)

The DC-DC converter is a product from TRACO Power that offers high isolation, an efficiency up to 98% and reliability in a compact SIP package. It can convert an input voltage range of 18V to 75V to a regulated output voltage of 9V with a maximum current of 1A. [21]



**Figure 2.7.** TMR 9-4819WI Converter. [21]

## 2.2 Software

### 2.2.1 ROS (Robot Operating System)

It is an open-source framework or middleware that provides a collection of software libraries and tools to help developers build and manage robot applications. Despite its name, ROS is not an operating system in the traditional sense; instead, it is a flexible and distributed framework designed to run on top of a real-time operating system (such as Linux) and provides a set of services that facilitate communication between various components of a robot system. [22]

ROS offers a wide range of functionalities for robot development, including hardware abstraction, low-level device control, message-passing between different nodes, package management, visualisation tools, and more. It follows a modular and decentralised approach, where different software components, known as nodes, can be created to perform specific tasks and communicate with each other using a publish-subscribe messaging model. [22]

One of the key advantages of ROS is its large and active community. The open-source nature of ROS encourages collaboration, knowledge sharing, and the development of reusable software components. This ecosystem has resulted in a rich repository of pre-existing packages that can be leveraged to accelerate robot development and experimentation. [22]

ROS has gained significant popularity in the robotics field and is widely used in both research and industrial applications. It supports a variety of robots, ranging from small educational platforms to complex humanoid robots and autonomous vehicles. [22]

#### The publish-subscribe messaging model

The publish-subscribe messaging model facilitates communication between different nodes in a distributed system. [23]

In the publish-subscribe model, nodes in a ROS system are categorised into two roles: publishers and subscribers. Publishers are responsible for sending messages, while

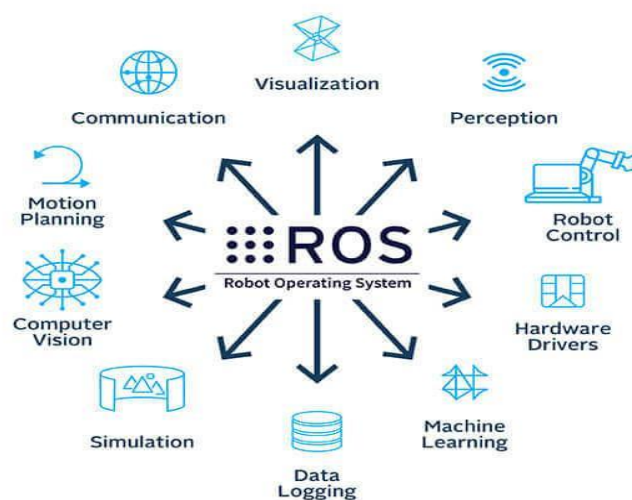
subscribers receive those messages. The key idea is that publishers and subscribers are decoupled from each other and do not have direct knowledge of one another. [23]

**Publishers:** A node in ROS that generates and sends messages on a particular topic is known as a publisher. Publishers have the freedom to choose the topic they want to publish on. They can publish messages on multiple topics simultaneously if needed. When a publisher has a new message, it publishes it on the chosen topic. [23]

**Subscribers:** A node in ROS that is interested in receiving messages on a particular topic is called a subscriber. Subscribers subscribe to specific topics and wait for incoming messages. They can subscribe to multiple topics if required. When a subscriber is registered for a topic, it will receive any messages published on that topic. [23]

**Topics:** Topics act as the communication channels in the publish-subscribe model. They represent a specific stream of messages with a defined message type. Publishers and subscribers connect to topics to publish and receive messages, respectively. Topics are identified by their names, which are strings in ROS. [23]

**Message Passing:** When a publisher sends a message on a topic, the message is distributed to all the subscribers currently subscribed to that topic. Each subscriber receives a copy of the message and can process it independently. This decoupling allows for flexible and scalable communication between nodes in a ROS system. [23]



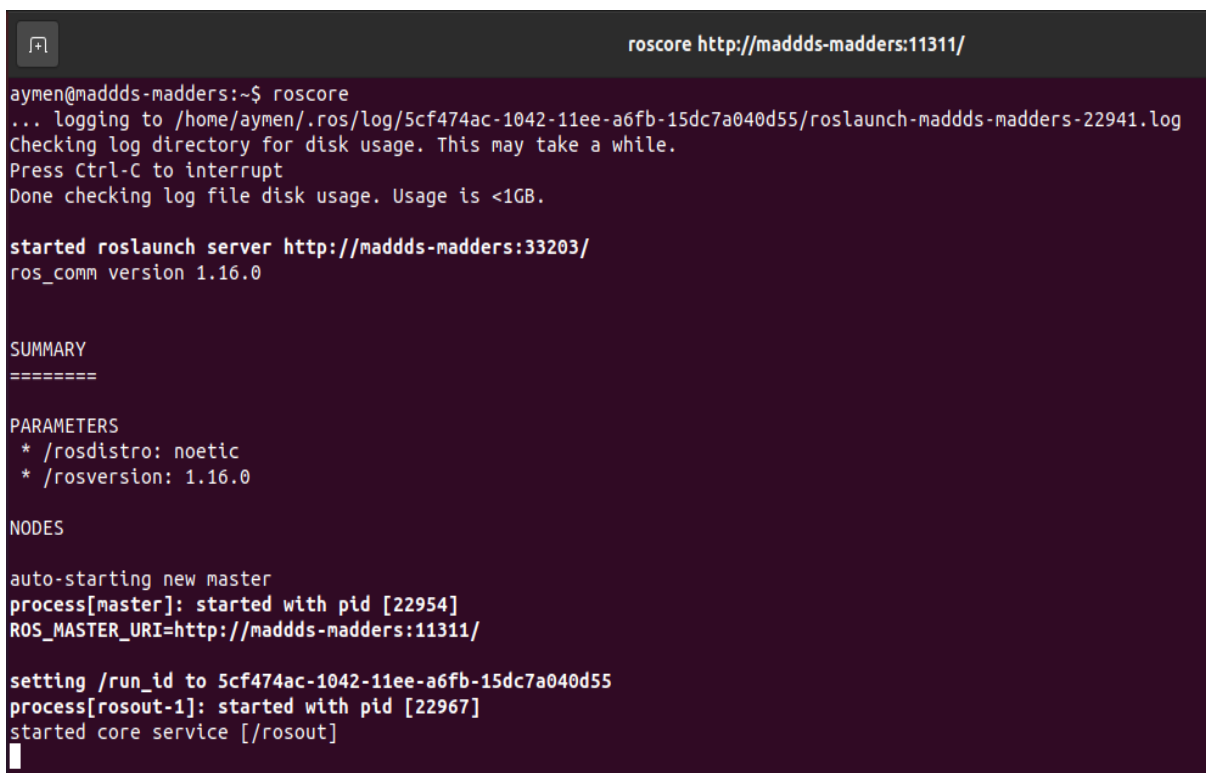
**Figure 2.8.** ROS applications. [24]

## 2.2.2 UBUNTU (LINUX)

Linux is a free and open-source operating system kernel that serves as the foundation for many different operating systems, commonly referred to as Linux distributions or distros. It was originally created by Linus Torvalds in 1991 and has since gained widespread popularity and adoption. [25]

Linux is known for its stability, security, and flexibility. It is designed to be highly customizable and adaptable to a wide range of hardware platforms, from personal computers to servers, mobile devices, embedded systems, and even supercomputers. [25]

Ubuntu is one of the most popular and widely used Linux distributions. It is based on the Debian distribution and is known for its user-friendly approach, stability, and extensive community support. Ubuntu aims to provide a complete, reliable, and free operating system for personal computers, servers, and the cloud. [25]

A terminal window with a dark background and light text. The title bar at the top reads 'roscore http://maddds-madders:11311/'. The terminal content shows a user named 'aymen' at a prompt '~\$' running the 'roscore' command. The output includes logging information, a disk usage check, and the start of a 'roslaunch server' at 'http://maddds-madders:33203/'. It lists parameters for 'roscore' such as '/roscore: noetic' and '/roscore: 1.16.0'. It also shows the start of a 'roscore master' process with PID [22954] and a 'roscore' node with PID [22967].

```
aymen@maddds-madders:~$ roscore
... logging to /home/aymen/.ros/log/5cf474ac-1042-11ee-a6fb-15dc7a040d55/roslaunch-maddds-madders-22941.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://maddds-madders:33203/
ros_comm version 1.16.0

SUMMARY
=====

PARAMETERS
* /roscore: noetic
* /roscore: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [22954]
ROS_MASTER_URI=http://maddds-madders:11311/

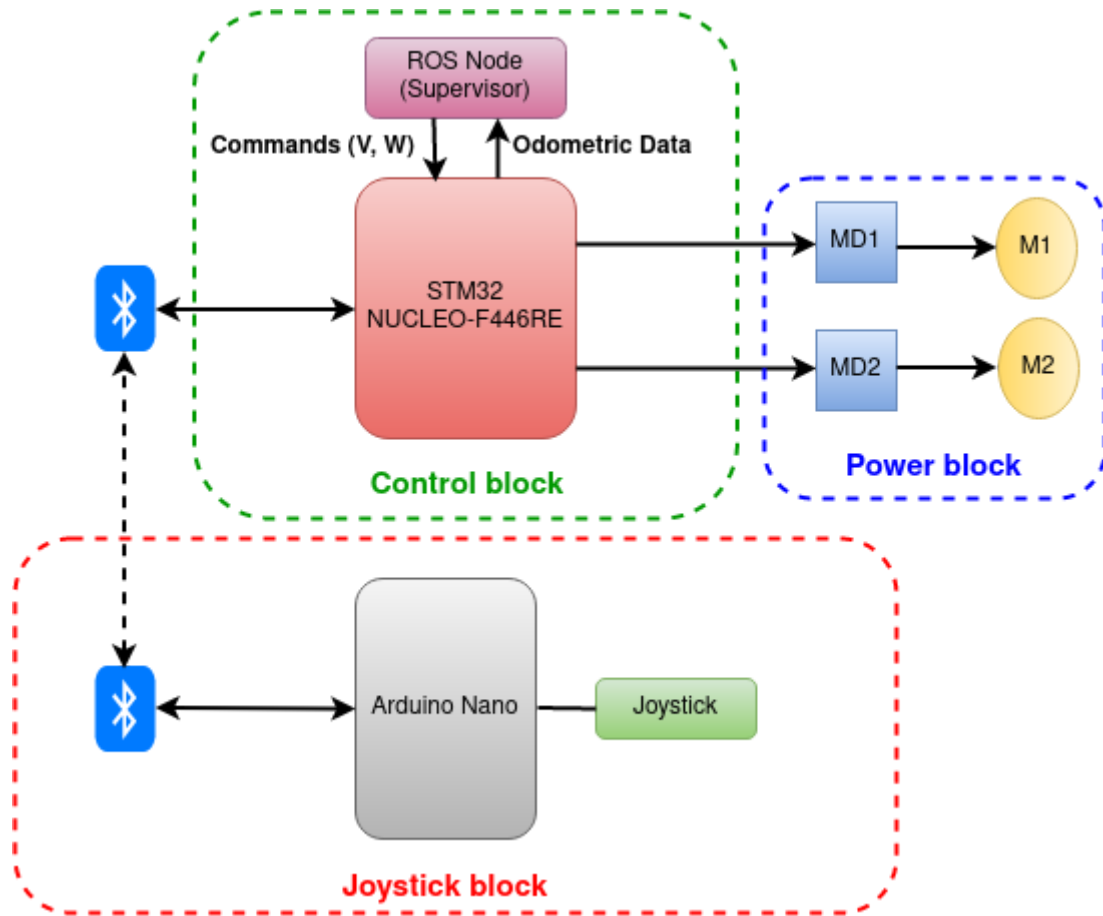
setting /run_id to 5cf474ac-1042-11ee-a6fb-15dc7a040d55
process[roscore-1]: started with pid [22967]
started core service [/roscore]
```

Figure 2.9. Communicating ROS nodes on Ubuntu.

- **N.B.** Please note that for more detailed information, you can refer to the Annexes section.

## 2.3 Deployment Scenario (Circuit)

### 2.3.1 Block diagram



**Figure 2.10.** Full block diagram.

We divided our work into three blocks, each with its own functionality.

The first block is the control block, which receives data, chooses the appropriate mode, calculates the speed, velocity...etc . It then sends the commands to the next block.

The second block is the power block, consisting of motors and motor drivers. It receives the commands from the control block and acts accordingly to execute the desired actions.

Finally, we have the joystick block, which includes a joystick, an LED, and Bluetooth... etc. This block is responsible for sending and receiving data related to the joystick's movements and controlling the overall system.

## ★ Control block

This block is based on an STM32 core M4, our microcontroller is responsible for choosing which mode to use, either the manual or the autonomous one after the user presses one of the two buttons.

The microcontroller receives the joystick's data (coordinates **X** and **Y**), converts it to a right and left wheel speed, those later on can be used to control the power block (for manual mode).

The microcontroller interacts also with the ROS supervisor, by sending the odometric data and receiving the angular, linear velocities and the commands to control the power block (for autonomous mode).

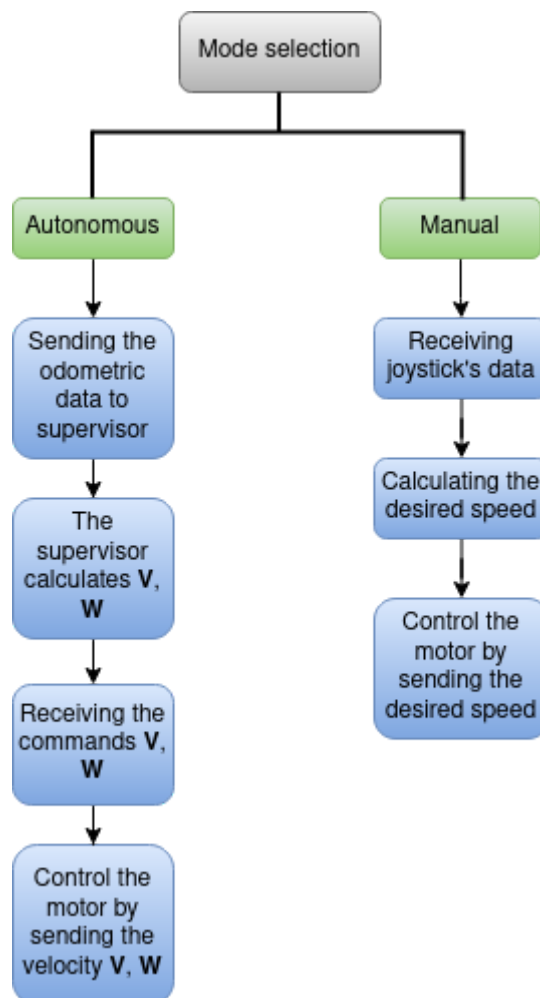


Figure 2.11. Control block process.

★ **Power block**

❖ Motor driver - Motor connections

Motor (Power)	Motor driver
Motor phase A (Yellow)	MA
Motor phase B (Green)	MB
Motor phase C (Blue)	MC

**Table 2.2.** Motor driver - Motor connections (Power) .

Motor (HALL effect)	Motor driver
Hall a (Yellow)	Ha
Hall b (Green)	Hb
Hall c (Blue)	Hc
GND	GND
5V output	5V

**Table 2.3.** Motor driver - Motor connections (HALL).

❖ Motor driver - Microcontroller connections

- Left motor driver

Motor driver	Microcontroller
Z / F	PC7
VR	PB9
EL	/
GND	GND

**Table 2.4.** Left motor driver - Microcontroller.

- Right motor driver

Motor driver	Microcontroller
Z / F	PC6
VR	PA5
EL	/
GND	GND

**Table 2.5.** Right motor driver - Microcontroller.

- **N.B.** Please note that for more detailed information about Pins functionality or wiring, you can refer to **Annex A-4** in the Annexes section.

★ **Joystick block**



**Figure 2.12.** Wireless joystick.

Where:

- ① Joystick.
- ② LED indicating manual or autonomous mode.
- ③ Remote Control.
- ④ Remote Control Holder.
- ⑤ Motors Stop Button.



### 2.3.2 Full electric schematic

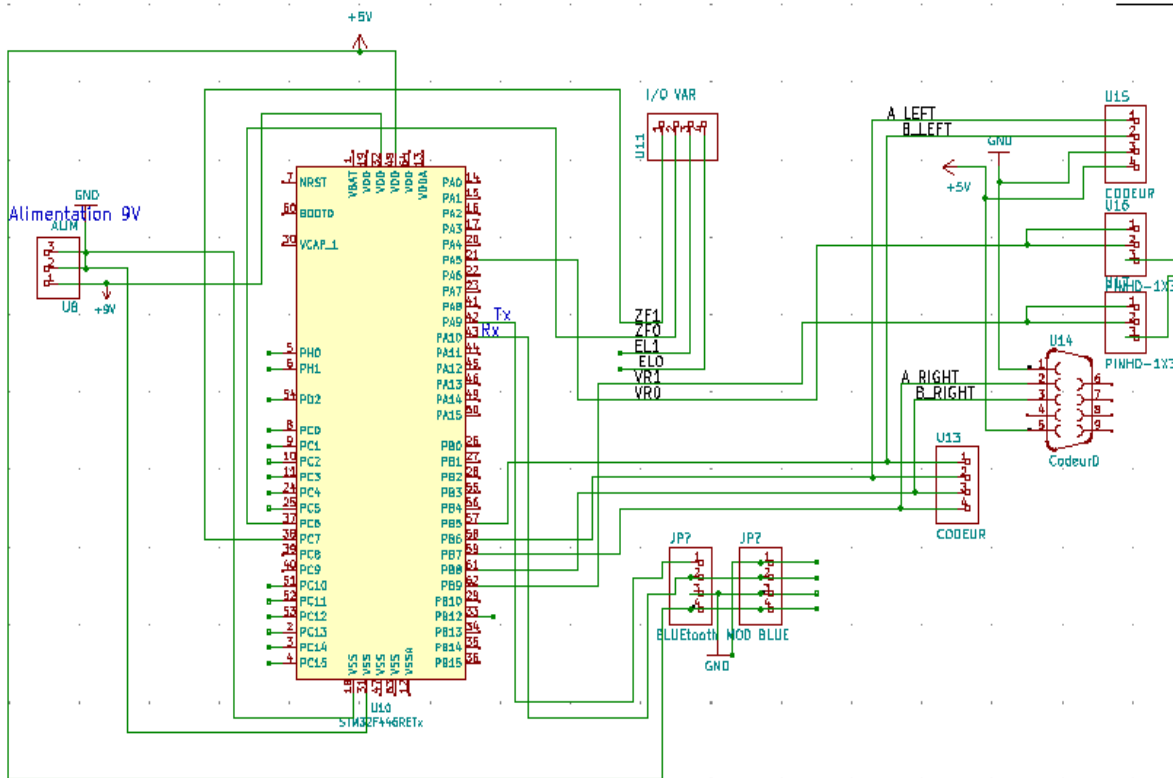


Figure 2.13. Full design schematic.

## CONCLUSION

In conclusion, this chapter has provided a comprehensive exploration of the hardware and software components essential for the development of the system discussed in this thesis. Alongside the detailed analysis of microcontrollers, Bluetooth modules, motor drivers, brushless DC motors, encoders, and DC-DC converters, we have also examined the integration of these components in the design and deployment scenario. This integration encompasses the control block, power block, joystick block, and their respective functionalities.

In the next chapter, we will focus on the detailed step-by-step workflow and present the results of the system. This next chapter will provide an in-depth exploration of the system's implementation process and highlight the outcomes and findings derived from the deployed system.

## Introduction

This chapter focuses on the robotic system used in this project, which aimed to transform a mechanical wheelchair into an intelligent wheelchair. We begin by providing an overview of the wheelchair's characteristics, including dimensions and weight. The kinematic model of the wheelchair, involving the control of two fixed, non-steerable wheels, is presented, along with the calculation of right and left wheel speeds.

The chapter further outlines a step-by-step workflow, covering the transfer of joystick data, calculation of desired speed and direction, measurement of motor speed, implementation of control algorithms, and regulation of motor speed using PWM signals. The power supply system, consisting of a 36V battery, motor drivers, DC-DC converter, and microcontroller, is discussed, highlighting its role in enabling the wheelchair's movement.

Results of the PID (Proportional-Integral-Derivative) control algorithm are presented, demonstrating the determination of suitable parameters for stable control. The chapter concludes with an exploration of the implementation of the system on the wheelchair, including the creation of ROS-based connections, reading base scan data, publishing and subscribing messages, and showcasing the wheelchair's autonomous movement capabilities through testing.

The successful integration of the hardware and software components, as demonstrated in this chapter, lays the groundwork for us to delve into the results obtained from the implementation.

### 3.1 The robotic system used

In this project, we were provided with a mechanical wheelchair by the CDTA (Advanced Technology Development Centre). Our objective was to develop a suitable hardware and software architecture to transform it into an intelligent wheelchair. Here are its characteristics:

- Length: 750mm
- Width: 720mm
- Height: 940mm
- Chassis weight: 20kg
- Total weight of the chair: 35 kg

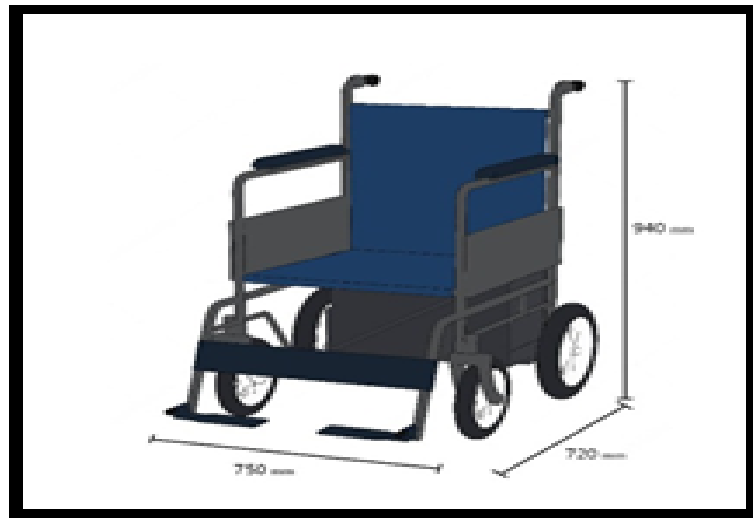


Figure 3.1. Wheelchair dimensions.

### 3.2 The kinematic model

It has two fixed, non-steerable wheels independently controlled, one or more idler wheels are added to the front or rear of the robot to ensure its stability Right and left wheel speeds:

$$V_r = (R + L) * \Omega \text{ [26]}$$

$$V_l = (R - L) * \Omega \text{ [26]}$$

Where

$\Omega$ : the angular speed of the robot with respect to the ICR (Instant centre of rotation).

### 3.3 Step-by-Step Workflow

In this section, we divided the control process into steps to make it easier to understand:

#### ★ Step 01: Joystick Data transfer

The joystick is connected to an Arduino Nano. The Arduino Nano sends the joystick data to our STM32 board via Bluetooth. It consists of two potentiometers that measure the horizontal and vertical movement of the stick, and a push button that detects when the stick is pressed. The joystick is connected to an HC-05 bluetooth module, which sends the data from the joystick to the Arduino Nano via serial communication. The Arduino Nano receives the data and performs the corresponding actions based on the joystick input.

#### ★ Step 02: Calculating the desired speed and direction

The STM32 board receives the joystick data from the HC-05 Bluetooth module and processes it to determine the speed and direction desired. The joystick data consists of two values: the x-axis and the y-axis, which represent the horizontal and vertical movements of the joystick. The STM32 board converts these values into a speed and a direction for each motor. The speed is proportional to the distance of the joystick from the centre, and the direction is determined by the angle of the joystick. The STM32 board will later on send the speed and direction signals to the motor drivers, which adjust the voltage and current of the motors accordingly. (Please refer to Chapter 3, Section 2 "Kinematic Model").

### ★ Step 03: Calculating the motor speed

The next step is to read the motor speed in RPM (revolution per minute) which is a measure of how fast a rotating object spins around its axis. It is calculated by dividing the number of revolutions by the time in minutes. For example, if a wheel makes 300 revolutions in one minute, its RPM is 300.

To do that we have to attach an interruption onto both motors encoder's channel B, the function that will be executed during the interruption is to calculate the pulses per period and then we use this formula to finally read the speed in RPM:

$$\Omega \text{ (tr/mn)} = \frac{60 \cdot N_{measure}}{N_{encoder} \cdot \Delta T \cdot F_{reduction}}$$

With:

$N_{measure}$ : Number of pulses measured during the period  $\Delta T$ .

$N_{encoder}$ : Initial encoder resolution multiplied by 4.

$\Delta T$ : Servo program sampling period.

$F_{reduction}$ : Reduction factor.

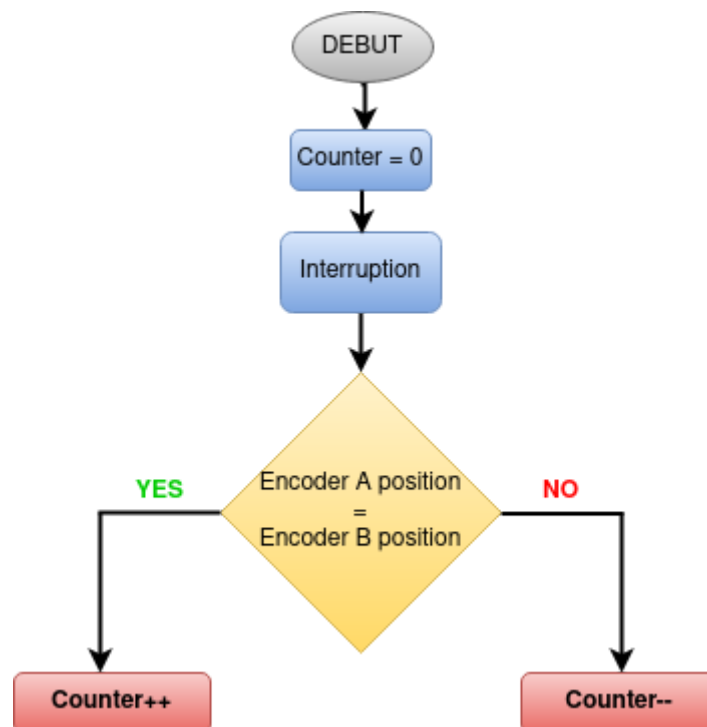


Figure 3.2. Function executed during interruption.

## ★ Step 04: Control

After calculating the motor speed, we perform the PID algorithm which stands for Proportional-Integral-Derivative, a type of feedback controller that adjusts the output of a system based on the error between the desired and actual values.

$$U(t) = K_p * e(t) + K_i * \int e(t) dt + K_d * \frac{de(t)}{dt} \quad [27]$$

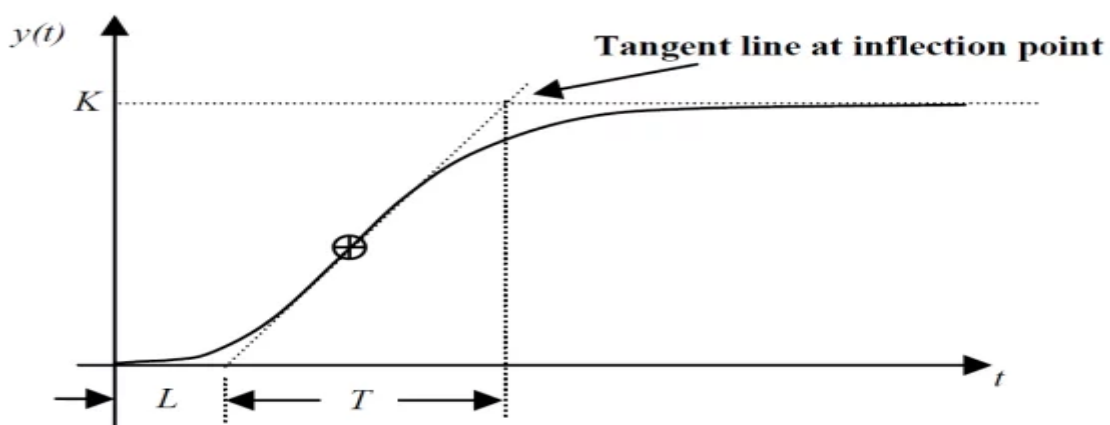
Where:

- $U(t)$ : is the output.
- $e(t)$ : is the error.
- $K_p$ : is the proportional gain.
- $K_i$ : is the integral gain.
- $K_d$ : is the derivative gain.
- $t$ : is the time.

We then used the ZIEGLER-NICHOLS step response method to calculate the PID parameters which uses the characteristics of the system's step response to determine the controller parameters  $K_p$ ,  $K_i$  and  $K_d$ .

The **figure 3.3.** represents a graphical explanation on how to determine the characteristics  $U$  and  $T$  which can be used to calculate our parameters.

**MATLAB** is used for the display of the step response of the system, so we obtained the graph below:



**Figure 3.3.** Step response ZIEGLER - NICHOLS method.

Based on the U and T values, the **table 3.1.** helps us calculate the parameters of our PID algorithm

Algorithm	Kp	Ti = Ki	Td = Td
P	$\frac{T}{L}$		
PI	$\frac{T \cdot 0.9}{L \cdot K}$	3.33 L	
PID	$\frac{T \cdot 1.2}{L \cdot K}$	2 L	0.5 L

**Table 3.1.** PID parameters. [27]

### ★ Step 05: PWM motor speed and limitations

A PWM signal is a method for creating digital pulses to control analog circuits. There are two primary components that define a PWM signal's behaviour:

- Duty cycle: A duty cycle is the fraction of one period when a system or signal is active. We typically express a duty cycle as a ratio or percentage.
- Frequency: Frequency is the number of times a PWM signal switches on and off in one second. We measure frequency in Hertz (Hz).

By changing the duty cycle and frequency of a PWM signal, we can adjust the amount of power delivered to a load, such as a motor or an LED.

After the regulation we set the PWM duty cycle according to the desired speed calculated earlier using the PID algorithm and then generate it to the motor speed pin (VR).

We have to consider the motor speed limitation in order to reduce the risk of overheating and damage.

If duty cycle < 800  $\Rightarrow$  duty cycle = 800 ; If duty cycle > 2000  $\Rightarrow$  duty cycle = 2000

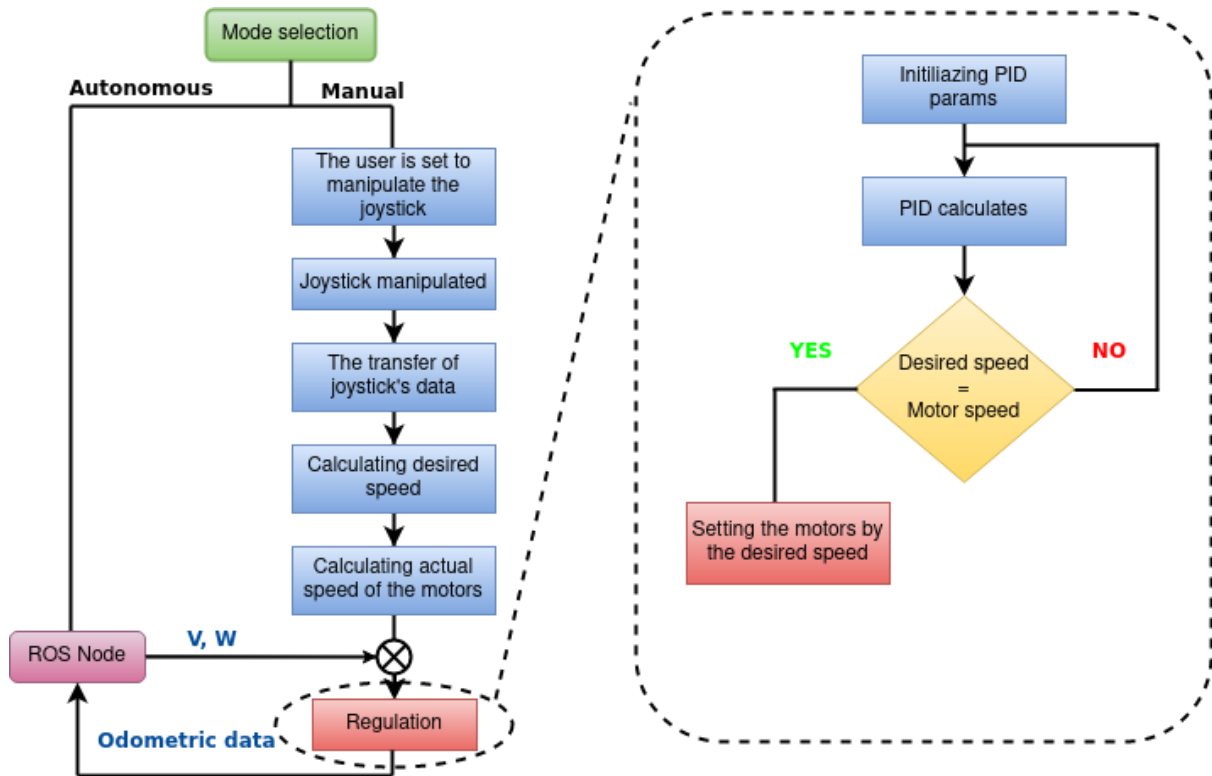


Figure 3.4. Step-by-Step Workflow.

### 3.4 Supply voltage

The power supply is a crucial part of the robotic system that we developed for this project. It consists of four main elements: a 36v battery, two motor drivers, a dc-dc converter and a microcontroller. The 36v battery is the source of energy for the system, which delivers the required voltage and current to the motors that enable the robot to move. The motors are controlled by the motor drivers, which receive signals from the microcontroller to adjust the speed and direction of the robot. The microcontroller is also powered by the 36v battery, but through a dc-dc converter that reduces the voltage to 9v.



Figure 3.5. Supply conversion.



### 3.5 PID results

After smoothing the step response graph, plotting the tangent to the inflection point of the new graph allows us to obtain the values of L and T.

$$L = 0.624s$$

$$T = 0.636s$$

These values allow us to calculate the adjustment parameters of the PID corrector:

$$K_p = \frac{T}{L} = 1.22$$

$$T_i = K_i = 2 * T_u = 1.24$$

$$T_d = K_d = 0.5 * T_u = 0.31$$

After several tests with different values of  $K_p$ ,  $K_i$ ,  $K_d$ , we found that the best results were obtained with the following values:

$$K_p = 2.5 ; K_i = 5 ; K_d = 1.5$$

**N.B.** We note that due to the change in dynamics caused by the presence of a person sitting in the wheelchair, the PID parameters have undergone significant adjustments. The initial parameters were calculated when the wheelchair was empty, and the introduction of the person's weight and posture has resulted in a notable change in these parameters.

### 3.6 Implementation on the wheelchair

Before trying to read a base scan data or creating a publisher...etc. Every ros communication needs the “roscore” command.

“roscore” is the command used to start the ROS Master, which is a crucial component of the ROS system. The ROS Master provides naming and registration services to other nodes in the ROS network. It allows nodes to discover each other, publish and subscribe to topics, and use various ROS services.

```

roscore http://maddds-madders:11311/

aymen@maddds-madders:~/tutorial_workspace$ roscore
... logging to /home/aymen/.ros/log/e8f00372-1d17-11ee-9dbf-d9100d35a55d/rosLaunch-maddds-madders-14827.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://maddds-madders:34849/
ros_comm version 1.16.0

SUMMARY
=====

PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.16.0

NODES
auto-starting new master
process[master]: started with pid [14838]
ROS_MASTER_URI=http://maddds-madders:11311/

setting /run_id to e8f00372-1d17-11ee-9dbf-d9100d35a55d
process[rosout-1]: started with pid [14851]
started core service [/rosout]

```

Figure 3.6. Creating the connection between ros nodes.

❖ Reading base scan data from a bag file

To do that, the command “rostopic echo” reads all the data of the base scan topic, which can contain scan\_time, the minimum and maximum range, all the ranges...etc.

```

roscore http://maddds-madders:11311/

aymen@maddds-madders:~/Downloads$ rostopic echo /scan
header:
  seq: 387
  stamp:
    secs: 1355457746
    nsecs: 671435629
  frame_id: "/base_laser"
angle_min: 0.0
angle_max: 6.2831854820251465
angle_increment: 0.01745329238474369
time_increment: 0.0006650062277913094
scan_time: 0.0
range_min: 0.30000001192092896
range_max: 5.0
ranges: [4.9679999351501465, 5.125999927520752, 4.869999885559082, 5.364999771118164, 5.017000198364258, 5.427000045776367, 5.214000225067139,
5.556000232696533, 5.9039998054504395, 5.544000148773193, 5.715000152587891, 5.85099983215332, 0.007000000216066837, 5.830999851226807, 5.830999851226807, 0.052
99999937415123, 0.007000000216066837, 0.007000000216066837, 0.05299999937415123, 0.05299999937415123, 0.05299999937415123, 3.490000009536743,
3.2920000553131104, 3.1519999504089355, 2.9739999771118164, 2.828000068664551, 2.7290000915527344, 2.5899999141693115, 2.510999917984009, 2.40
89999198913574, 2.3269999027252197, 2.236999988555908, 2.1710000038146973, 2.1010000705718994, 2.0380001068115234, 1.9830000400543213, 1.92799
99732971191, 1.8769999742507935, 1.8309999704360962, 1.7860000133514404, 1.7380000352859497, 1.6950000524520874, 1.6829999685287476, 0.0529999
9937415123, 1.593000054359436, 1.5570000410079956, 1.531000018119812, 1.5010000467300415, 1.475000023841858, 1.4479999542236328, 1.42200005054
47388, 1.3969999551773071, 1.3769999742507935, 1.3569999933242798, 0.05299999937415123, 0.05299999937415123, 0.05299999937415123, 1.2699999809
265137, 1.2610000371932983, 1.246999979019165, 1.2319999933242798, 1.218999981880188, 1.2029999494552612, 1.190000057220459, 1.177999973297119
1, 1.166000085830688, 1.1549999713897705, 1.1449999809265137, 1.1349999904632568, 1.125, 1.1160000562667847, 1.1080000400543213, 1.0980000495
910645, 1.090999960899353, 1.0839999914169312, 1.062999963760376, 0.05299999937415123, 0.05299999937415123, 0.05299999937415123, 0.05299999937
415123, 1.0490000247955322, 1.0429999828338623, 1.0390000343322754, 1.034000039100647, 1.031000018119812, 1.0269999504089355, 1.02400004863739
01, 1.0219999551773071, 1.0180000066757202, 1.0149999856948853, 1.0140000581741333, 1.0140000581741333, 1.0119999647140503, 1.0190000534057617,
1.0080000162124634, 1.0080000162124634, 1.0130000114440918, 1.0149999856948853, 1.0160000324249268, 1.0190000534057617, 1.0199999809265137,
1.0230000019073486, 1.0260000228881836, 1.027999997138977, 1.031999945640564, 1.0360000133514404, 1.0399999618530273, 1.0449999570846558, 1.04
49999570846558, 0.05299999937415123, 1.062000036239624, 1.065999984741211, 1.0720000267028809, 1.0789999961853027, 1.0850000381469727, 1.09200
0076293945, 1.1009999513626099, 1.1089999675750732, 1.1180000305175781, 1.1269999742507935, 1.1369999647140503, 1.1460000276565552, 1.15600000
18119812, 1.16700000553131104, 1.180099994277954, 1.1920000314712524, 1.2050000429153442, 1.218999981880188, 1.2319999933242798, 1.248000025749
2065, 1.26300008114440918, 1.2799999713897705, 1.2970000505447388, 1.315000057220459, 1.3359999656677246, 1.3550000198734863, 1.378000028980835
, 1.4010000278881836, 1.4240000247955322, 1.4500000476837158, 1.4789999723434448, 1.5089999437332153, 1.5410000085830688, 1.5750000476837158,
1.6100000143051147, 1.6460000276565552, 1.6859999895095825, 1.7230000495910645, 1.7669999599456787, 1.809999942779541, 1.8600000143051147, 1.9
129999876022339, 1.9529999494552612, 0.05299999937415123, 0.05299999937415123, 0.05299999937415123, 2.2750000953674316, 2.328000068664551, 0.0

```

Figure 3.7. Example of a base scan topic data using the command “rostopic echo”.

❖ Publishing messages

After running “roscore”, the command “source devel/setup.bash” is used to set up the environment variables required to properly configure and run ROS packages.

By sourcing the **setup.bash** script, we ensure that our terminal session is properly configured to work with the ROS packages in our workspace.

We need then to create a code (node) that publishes the messages which in our case was written in **Python** programming language, to execute the code we use the command “**roscrun name\_of\_package name\_of\_the\_publisher\_node**”

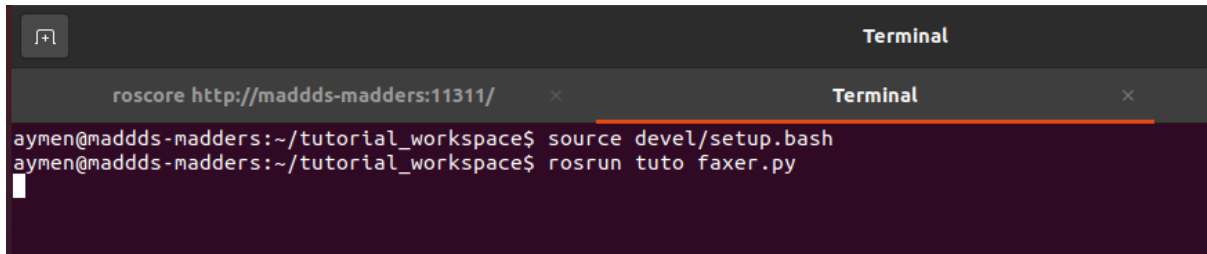


Figure 3.8. Example on how to publish a message via ROS publisher.

### ❖ Subscribing messages

Subscribing (reading) messages have the same steps as publishing, running “roscrun” than sourcing the **setup.bash** script and finally after creating a subscriber (node), we execute the code by running the command:

“**roscrun name\_of\_package name\_of\_the\_subscriber\_node**”

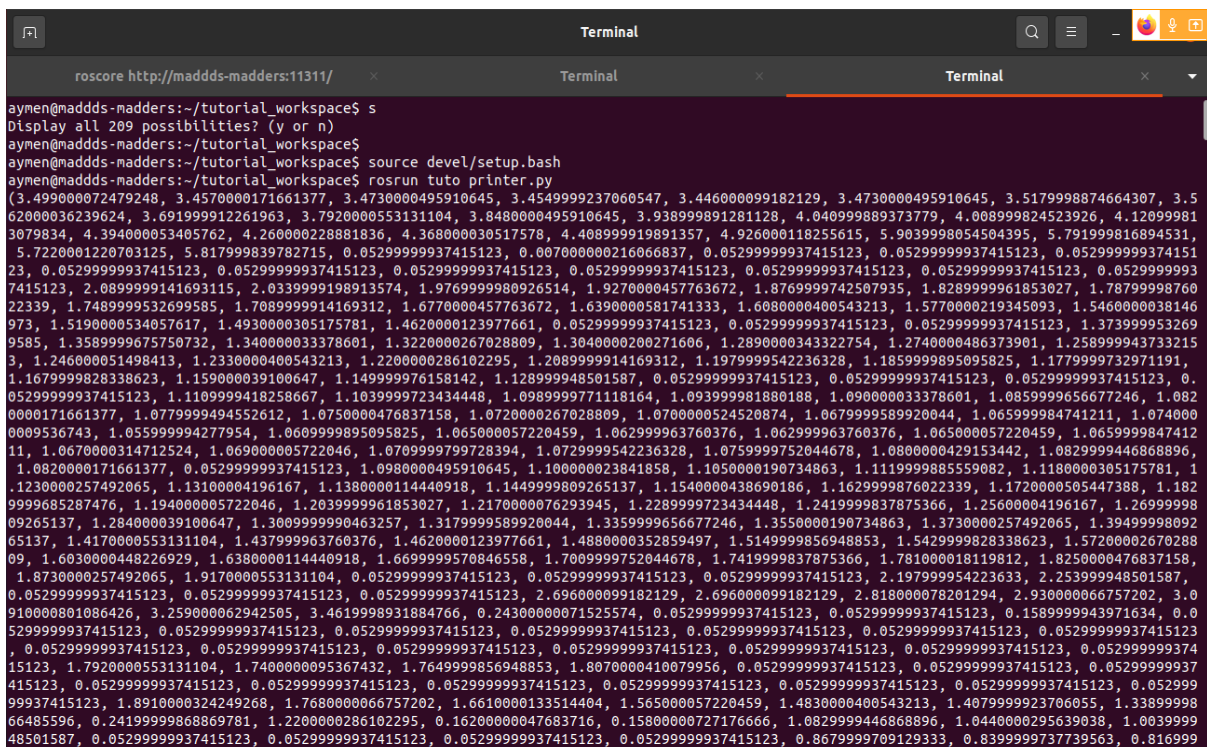
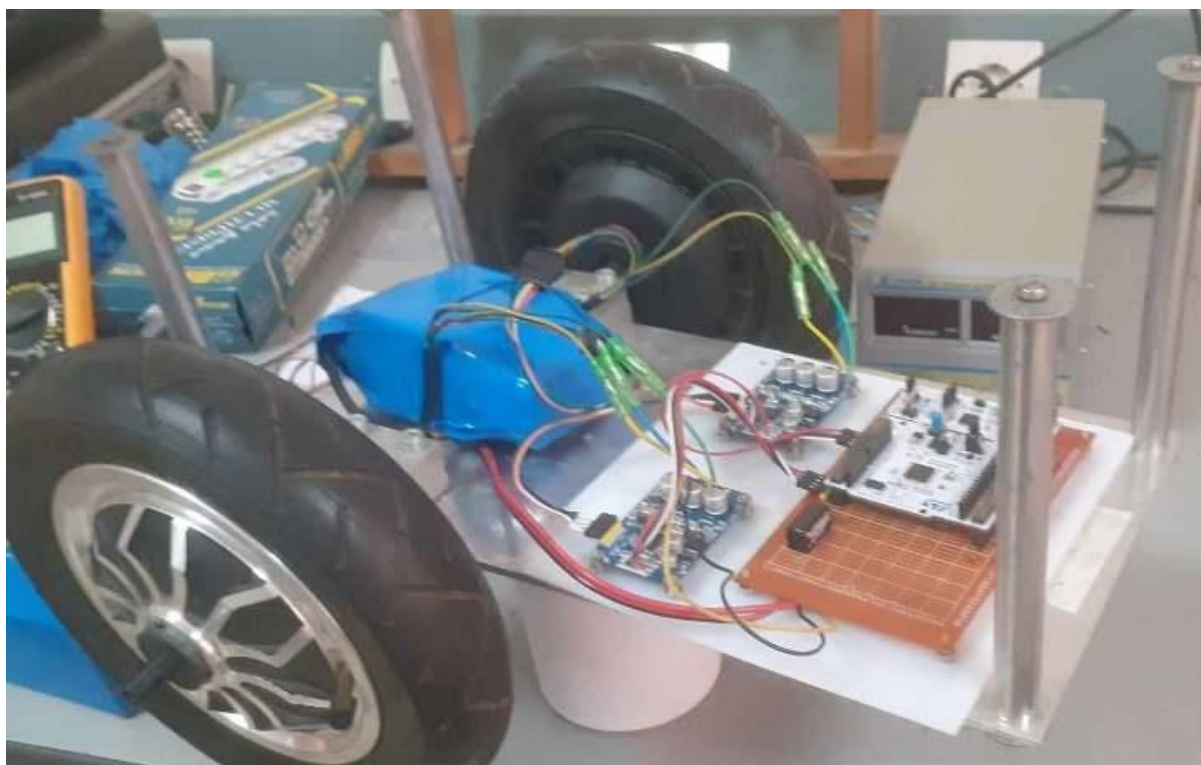


Figure 3.9. Example on how to read a message via ROS subscriber.

### 3.7 Wheelchair in motion



**Figure 3.10.** Platform testing.



**Figure 3.11.** Start of the test.





**Figure 3.12.** Wheelchair moving forward.



**Figure 3.13.** Wheelchair turning left.

## → Discussion

- The testing process involved an evaluation of the developed architecture, we tested the speed, the direction and the obstacle detection of the robot. It gave us valuable insights of our project.
- During the testing phase, we conducted a series of evaluations to assess the functionality and performance of our project. Firstly, we tested the project on a platform (figure 3.11.), where we verified its successful operation. Once confirmed, we proceeded to implement the project on the wheelchair itself. In the initial test, as depicted in the first picture, the wheelchair autonomously started functioning. Subsequently, in the second picture, it demonstrated forward movement, showcasing its ability to navigate in a controlled manner. Finally, in the last picture, the wheelchair executed a left turn, displaying its autonomous manoeuvring capabilities which implies that the speed of the left wheel is higher than the right one.
- These testing results highlight the successful implementation of our hardware and software architecture, enabling the wheelchair to operate autonomously and fulfil its intended functionalities.

## Conclusion

In conclusion, this chapter successfully implemented the hardware and software architecture for the intelligent wheelchair system. Through a step-by-step workflow, including the transfer of joystick data, calculation of desired speed and direction, motor speed measurement, the system demonstrated controlled movement and autonomous manoeuvring capabilities.

Finally, we conducted a series of tests to evaluate the performance of the wheelchair. Through real-world navigation scenarios, we captured images that showcased the successful implementation of the control system. These tests affirmed the effectiveness of our design and demonstrated its ability to manoeuvre the wheelchair reliably and safely.

## General conclusion

In this study, we focused on the development of a hardware and software architecture for an intelligent wheelchair. The main objective was to create a motherboard that enables the wheelchair to be controlled wirelessly using a joystick. While our project primarily focused on the implementation of the joystick control mode, we also proposed future enhancements using the Robot Operating System (ROS) to enable manual control of the wheelchair.

We covered the various parts and technologies used in the creation of the intelligent wheelchair throughout the thesis. We created and put into use a unique motherboard that works perfectly with the current wheelchair system. The designed architecture's viability and functionality were demonstrated through the successful implementation and testing of the wireless joystick control mode.

Despite the fact that time restrictions limited the scope of our amelioration efforts, we provided an outline for a prospective strategy for future work. We can investigate additional control modes and features by integrating ROS into the system, which will improve the intelligent wheelchair's overall usability and adaptability.

The development of an intelligent wheelchair that can increase the mobility and independence of people with mobility disabilities is made possible thanks in large part to this research. The created architecture serves as the framework for additional developments and improvements in the area of assistive technology.

In conclusion, the development of the intelligent wheelchair's motherboard and the successful use of the wireless joystick control mode show the possibility for creating cutting-edge assistive technologies. The wheelchair's manual control capabilities can be added with additional ROS development, giving users more options for navigating their surroundings. This study adds to the increasing body of knowledge in the field of assistive robotics and has the potential to enhance the lives of those who have mobility challenges.

### Annex A-1: Microcontrollers

#### ★ AT91SAM3X8E (Arduino DUE)

The AT91SAM3X8E is a microcontroller from Atmel, now part of Microchip Technology, based on the ARM Cortex-M3 architecture. It is a high-performance microcontroller that offers a range of features, making it suitable for a wide variety of applications. [14]

One of the notable features of the Arduino Due is its 32-bit ARM architecture, which provides enhanced computing power compared to traditional 8-bit Arduino boards. This allows for more complex and demanding applications to be developed. The increased clock speed, larger memory capacity, and numerous I/O options make it suitable for a wide range of projects, including robotics, data acquisition systems, and high-performance applications. [14]

Arduino Due is also compatible with the Arduino software and library ecosystem, making it easier for beginners and experienced developers alike to get started with programming and prototyping. Its versatility, ample resources, and community support have made it a popular choice among electronics enthusiasts and professionals working on advanced projects. [14]

#### □ Features

- **Processor:** The AT91SAM3X8E is based on the ARM Cortex-M3 processor, which is a 32-bit RISC architecture designed for low-power, high-performance embedded applications. The Cortex-M3 processor supports Thumb-2 instruction set, which provides a balance between code density and performance. The processor can run at a clock speed up to 84 MHz, which makes it suitable for real-time applications. [14]



- **Memory:** The AT91SAM3X8E has 512 KB of flash memory for program storage and 96 KB of SRAM for data storage. It also has a 4 KB EEPROM for non-volatile data storage. [14]
- **Communication Interfaces:** The AT91SAM3X8E offers a variety of communication interfaces, including USB, Ethernet, CAN, I2C, and SPI. These interfaces can be used to communicate with external devices and networks.
  - I. **USB:** It offers USB 2.0 Full Speed device interface, which allows it to communicate with other USB devices such as a PC, USB storage devices, and other USB-enabled devices. [14]
  - II. **Ethernet:** It offers a 10/100 Mbps Ethernet MAC (Media Access Control) interface, which enables it to connect to a network and communicate with other devices. [14]
  - III. **CAN:** The AT91SAM3X8E has a CAN controller, which provides a communication protocol widely used in the automotive industry for connecting various components of a vehicle. [14]
  - IV. **I2C:** The AT91SAM3X8E has up to two I2C interfaces, which provide a low-speed synchronous communication protocol for connecting to other devices such as sensors, EEPROMs, and other microcontrollers. [14]
  - V. **SPI:** The AT91SAM3X8E has up to 4 SPI interfaces, which provide a high-speed synchronous communication protocol for connecting to other devices such as flash memory, digital-to-analog converters, and other microcontrollers. [14]
  - VI. **USART:** The AT91SAM3X8E has up to four USART interfaces, which provide a standard serial communication protocol for connecting to other devices such as sensors, actuators, and other microcontrollers. [14]

- **Peripherals:** A wide range of configurable peripherals, including timers, ADCs, DACs, PWMs, and more. These peripherals can be used to interface with external devices and sensors.
  - I. **Timers and Counters:** The AT91SAM3X8E has several timers and counters that can be used for a variety of purposes, including generating periodic interrupts, measuring time intervals, and controlling pulse width modulation (PWM) signals. [14]
  - II. **ADCs:** It contains a 12-bit, 2 ADC that can convert analog signals into digital data, making it suitable for applications that require sensing or monitoring analog signals. [14]
  - III. **DACs:** It contains a 12-bit, 2-channel DAC that can convert digital signals into analog voltage outputs. This peripheral is useful in applications such as audio processing, motor control, and power supply control. [14]
  - IV. **Watchdog Timer:** The AT91SAM3X8E has a built-in watchdog timer that can be used to reset the system in case of a software or hardware failure. [14]
  - V. **PWM Generators:** It has 12 PWM pins that can be used to generate variable-width pulses, making it useful for applications such as motor control, LED dimming, and audio processing. [14]
  - VI. **Quadrature Encoder Interface (QEI):** The AT91SAM3X8E has a QEI module that can interface with quadrature encoders and provide position and velocity information. This is useful in applications such as robotics and motion control. [14]
  - VII. **Real-Time Clock (RTC):** The AT91SAM3X8E has a built-in RTC that can provide accurate time and date information, making it useful in applications such as data logging and event scheduling. [14]

## ★ STM32 NUCLEO-F446RE

The STM32 Nucleo-F446RE is a development board based on the STM32F446RE microcontroller from the STM32 family, which is built on the ARM Cortex-M4 processor architecture. It offers a wide range of features and peripherals, making it suitable for a variety of applications. [15]

The STM32 Nucleo-F446RE board provides an Arduino Uno V3 and ST morpho connector layout, facilitating compatibility with a vast ecosystem of shields and expansion boards. This makes it easier to prototype and develop projects quickly, leveraging existing libraries and resources. [15]

Additionally, the board features built-in ST-Link/V2-1 debugger/programmer, enabling easy debugging and programming of the microcontroller. It also supports mbed, a popular development platform, offering a comprehensive set of software libraries and online tools for rapid prototyping and development. [15]

### □ Features

- **Processor:** It is based on the STM32F446RE microcontroller, which utilises the ARM Cortex-M4 processor core. The ARM Cortex-M4 is a 32-bit RISC processor architecture known for its high performance and efficient power consumption running at a maximum frequency of 180 MHz. It supports the Thumb-2 instruction set, which combines 16-bit and 32-bit instructions to provide a good balance between code density and performance. This instruction set allows the processor to execute complex tasks efficiently while minimising memory usage. [15]
- **Memory:** The board includes 512 KB of flash memory for program storage and 128 KB of RAM for data storage, allowing for the execution of complex tasks and storage of variables. [15]
- **Communication Interfaces:** It supports various communication interfaces, such as USART, SPI, I2C, USB, CAN, and more. These interfaces enable seamless connectivity with other devices and systems.

- I. **USB:** There is 1 USB interface available on the board, enabling connection to a computer or other USB-enabled devices. This interface can be used for data transfer, device enumeration, and communication with USB peripherals. [15]
  - II. **CAN:** The board offers up to 2 CAN interfaces, which are commonly used in automotive and industrial applications for robust and reliable communication between nodes in a network. [15]
  - III. **I2C:** The board supports up to 3 I2C interfaces. I2C is a popular communication protocol used for connecting various devices, including sensors, EEPROMs, and real-time clocks, allowing for data transfer and control signals. [15]
  - IV. **SPI:** It provides up to 4 SPI interfaces, which enable high-speed synchronous communication with peripheral devices such as sensors, displays, and memory chips. [15]
  - V. **USART:** The board features up to Three USART interfaces, which allow for serial communication with external devices, such as sensors, displays, or other microcontrollers. [15]
- **Peripherals:** A wide range of configurable peripherals, including timers, ADCs, DACs, PWMs, and more. These peripherals can be used to interface with external devices and sensors.
    - I. **Timers:** STM32 microcontrollers typically offer a large number of timers, with some models offering up to 14 timers. These timers can be used for a variety of functions, including generating PWM signals, measuring input signals, and triggering events at specific intervals. [15]
    - II. **ADCs:** The board features up to 12 analog inputs for ADC conversion. It includes three separate ADC interfaces. Each interface can have multiple channels, allowing for simultaneous conversion of analog signals from different sources. [15]

- III. **DACs:** The STM32 NUCLEO-F446RE provides two DAC channels, DAC1 and DAC2. These channels allow for the conversion of digital signals to analog voltages, enabling precise control of analog outputs, such as audio signals or voltage references. [15]
- IV. **PWM Generators:** It typically offers 17 PWM generators, which can be used to generate PWM signals for controlling the speed of motors or the brightness of LEDs. [15]
- V. **Watchdog Timer:** The STM32 NUCLEO-F446RE incorporates an independent watchdog timer (IWDG). This timer helps ensure system reliability by monitoring the software execution and resetting the microcontroller if necessary. [15]
- **Arduino-Compatible Form Factor:** The NUCLEO-F446RE follows the Arduino Uno R3 pinout, which makes it compatible with a vast array of Arduino shields and expansion boards. This allows for easy integration of additional hardware modules and sensors, enabling rapid prototyping and expansion of functionality. [15]
- **On-Board Debugger/Programmer:** The STM32 NUCLEO-F446RE integrates an ST-LINK/V2-1 debugger and programmer. This on-board debugger allows for programming and debugging of the microcontroller directly from the development board, eliminating the need for an external programmer. It provides a convenient and seamless development experience. [15]

## **Annex A-2: HC-06 Bluetooth module**

The HC-06 module is based on Bluetooth version 2.0, which is an older version of the Bluetooth protocol. It supports the Serial Port Profile (SPP), allowing for easy serial communication between devices. [16]

It communicates using a serial UART interface. It can be connected to a microcontroller or other devices using the module's TX (transmit) and RX (receive) pins. [16]

The HC-06 module typically operates at a voltage of 3.3V, which is compatible with many microcontrollers and embedded systems. It requires a separate power supply for operation. [16]

#### □ HC-06 PINOUT

PIN	Functionality
VCC	Supply power to the HC-06 module (typically connected to a 3.3V power source).
GND	The Ground pin is connected to the ground reference of the power supply.
TXD	The transmit pin is used for sending data.
RXD	The receive pin is used for receiving data.
STATE	It provides status information about the module, such as connection status or pairing status (Optional pin).
EN/KEY	The Enable pin is used to enable or disable the module by pulling it high or low, respectively (Optional pin).

**Table A-2.1.** HC-06 Bluetooth module pinout. [16]

#### **Annex A-3:** HC-05 Bluetooth module

The HC-05 and the HC-06 use the the same breakout board (even have the same screen print) but have some noticeable differences:

- The HC-06 does not have a button switch. [16]
- The HC-06 only has 4 header pins. [16]
- The HC-06 does not have pins 31-34. [16]

They also have different firmwares. The HC-05 can be a master or slave. The HC-06 is a slave only. This means the HC-05 can initiate a connection to another device and the HC-06 can only accept a connection from another device. [16]

□ **HC-05 Pinout**

PIN	Functionality
VCC	Supply power to the HC-06 module (typically connected to a 3.3V power source).
GND	The Ground pin is connected to the ground reference of the power supply.
TXD	The transmit pin is used for sending data.
RXD	The receive pin is used for receiving data.
STATE	It provides status information about the module, such as connection status or pairing status (Optional pin).
EN/KEY	The Enable pin is used to enable or disable the module by pulling it high or low, respectively (Optional pin).

**Table A-3.1.** HC-05 Bluetooth module pinout. [16]

**Annex A-4:** The JYQD-V7.3E3 pinout

● Control pinout

PIN	Functionality
5V	Driver board internal output voltage
Z / F	Rotating direction control ports. (“5V” or no connect = Forward direction, “0V” or connect to GND = reverse direction.)
VR	Speed control port. Analog voltage linear speed regulation 0.1v -5V
EL	Enable port control. (“5V” or no connect = allow operation, “GND” to forbid operation)
Signal	Speed pulse signal output
GND	Used for Drive board internal control

**Table A-4.1.** The JYQD-V7.3E3 control pinout. [19]

- Power pinout

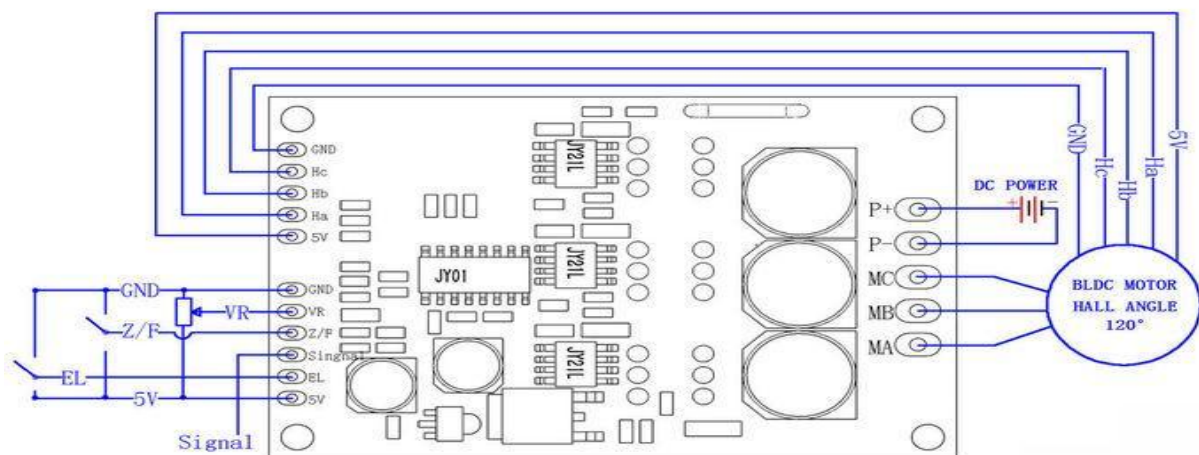
PIN	Functionality
MA	Motor phase A
MB	Motor phase B
MC	Motor phase C
GND	DC -
VCC	DC +

**Table A-4.2.** The JYQD-V7.3E3 power pinout. [19]

- Hall sensor pinout

PIN	Functionality
HA	Hall A
HB	Hall B
HC	Hall C
GND	GND
5V	5V output

**Table A-4.3.** The JYQD-V7.3E3 hall sensor pinout. [19]



**Figure A-4.1.** The JYQD-V7.3E3 wiring diagram. [19]



### Annex A-5: BLDC motor pinout

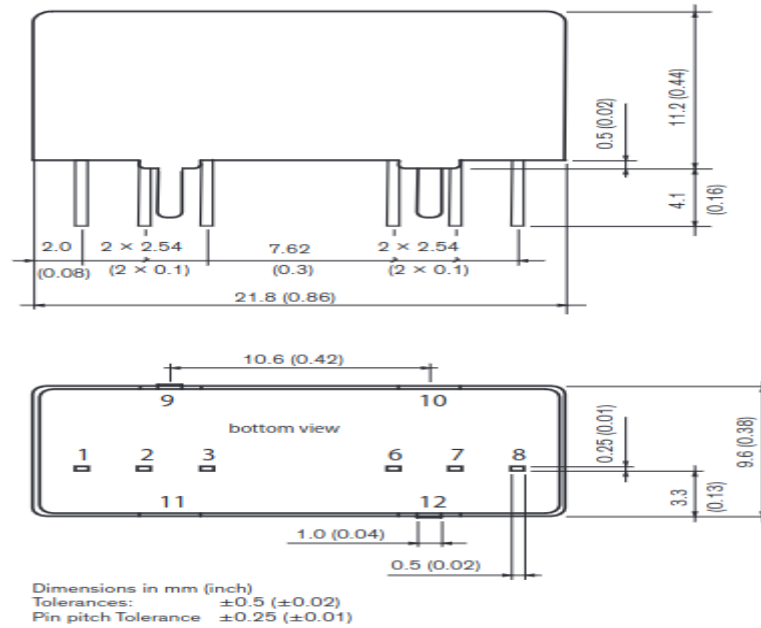
PIN	Functionality
M1	Phase A
M2	Phase B
M3	Phase C
H1	Hall sensor A
H2	Hall sensor B
H3	Hall sensor C
GND	Ground
VCC	Power supply

**Table A-5.1.** Brushless motor pinout

### Annex A-6: TMR 9-4819WI DC-DC converter pinout

PIN	Functionality
1	- Vin (GND)
2	+ Vin (Vcc)
3	Remote
6	+ Vout
7	- Vout
8	NC

**Table A-6.1.** TMR 9-4819WI pinout. [22]



**Figure A-6.1.** Outline Dimensions. [22]

## Annex A-7: ROS (Robot Operating System)

### □ Features and benefits

- **Communication Infrastructure:** ROS provides a robust communication infrastructure that enables efficient data exchange and message-passing between different components of a robot system. It follows a publish-subscribe model, where nodes can publish messages to specific topics, and other nodes can subscribe to those topics to receive the messages. This decentralised approach promotes modularity and simplifies the development of complex robot architectures. [23]
- **Package Management:** ROS has a package management system that allows users to easily share and distribute software components, libraries, and tools. This package system makes it simple to find, install, and update packages, accelerating development by leveraging existing functionality and reducing duplication of effort. [23]
- **Hardware Abstraction:** ROS provides hardware abstraction layers that allow developers to write robot-agnostic code. This abstraction simplifies the process of interfacing with various hardware devices, sensors, and actuators. By abstracting

the hardware details, ROS enables code reusability and portability across different robot platforms. [23]

- **Visualization and Debugging Tools:** ROS offers a range of visualisation and debugging tools that assist developers in understanding and monitoring their robot systems. These tools include graphical interfaces for visualising robot states, logging and visualisation of sensor data, real-time plotting, and introspection tools for inspecting the internal state of running nodes. [23]
- **Robotics Libraries:** ROS provides a collection of software libraries tailored for robotics applications. These libraries cover areas such as robot perception, control, mapping, navigation, and manipulation. They serve as building blocks for developing robot algorithms and can be readily integrated into ROS-based projects. [23]
- **Simulation Capabilities:** ROS supports simulation environments, such as Gazebo and Stage, that allow developers to simulate and test their robot systems in a virtual environment before deploying them on physical robots. Simulation helps reduce risks and costs associated with hardware testing and enables rapid prototyping and algorithm development. [23]
- **Community and Ecosystem:** One of the significant advantages of ROS is its large and active community. The ROS community provides extensive support, documentation, tutorials, and a repository of user-contributed packages. This collaborative ecosystem fosters knowledge sharing, accelerates development, and encourages innovation in the field of robotics. [23]

## **Annex A-8: UBUNTU (LINUX)**

### **□ Key aspects and features**

- **User-Friendly Interface:** Ubuntu offers a user-friendly graphical interface that makes it accessible to users with varying levels of technical expertise. The default desktop environment used by Ubuntu is called GNOME, which provides an intuitive and modern user interface.

- **Software Repository:** Ubuntu utilises the Advanced Packaging Tool (APT) package management system. It provides access to a vast software repository, known as the Ubuntu Software Center or Ubuntu Software, which allows users to easily browse, install, and update software applications.
- **Regular Release Cycle:** Ubuntu follows a predictable release schedule, with new versions being released every six months. These releases are typically supported with security updates and bug fixes for a specific period. The long-term support (LTS) releases are supported for five years, making them well-suited for stability-focused environments.
- **Ubuntu Flavours:** Alongside the standard Ubuntu edition, there are official variants known as "flavours" that feature different desktop environments. These include Kubuntu (KDE Plasma), Xubuntu (Xfce), Lubuntu (LXQt/LXDE), Ubuntu MATE (MATE), and Ubuntu Budgie (Budgie). These flavours cater to different user preferences and system requirements.
- **Community Support:** Ubuntu has a large and active community of users and contributors. The community provides extensive support through forums, documentation, wikis, and other resources. Ubuntu also encourages user participation and contributions to the development and improvement of the distribution.
- **Cloud and Server Capabilities:** Ubuntu has a strong presence in the cloud and server environments. Ubuntu Server Edition is specifically optimised for server deployments, offering a robust and scalable platform. Additionally, Ubuntu is widely used in cloud computing platforms, such as Amazon Web Services (AWS) and Microsoft Azure.
- **Ubuntu Touch:** Ubuntu Touch is a mobile version of Ubuntu designed for smartphones and tablets. It provides a touch-optimised interface and integration with mobile hardware. Although it is not as widely used as Android or iOS, Ubuntu Touch offers an alternative for those interested in a Linux-based mobile operating system.

## Bibliography

- [1] “The Manual Wheelchair What the SCI Consumer Needs to Know”, SCI Model Systems, 2011. [Online]. Available:  
[https://msktc.org/sites/default/files/lib/docs/SCI\\_wheelchairs2\\_final.pdf](https://msktc.org/sites/default/files/lib/docs/SCI_wheelchairs2_final.pdf). [Accessed June 17, 2023].
- [2] “ Manual Mobility - The Basics”, SUNRISE MEDICAL, December, 2013. [Online]. Available:  
<https://www.sunrisemedical.ca/education-in-motion/clinical-corner/december-2013/manual-mobility-the-basics>. [Accessed June 18, 2023].
- [3] “Lightweight folding manual wheelchair”, Tig Heaven. [Online]. Available:  
<https://tigheaven.com/product/lightweight-folding-manual-wheelchair/>. [Accessed June 18, 2023].
- [4] “GTM Endeavour rigid frame Wheelchair”, Momentum Healthcare. [Online]. Available:  
<https://www.momentumhealthcare.ie/product/gtm-endeavour-wheelchair/>. [Accessed June 18, 2023].
- [5] “ Power Wheelchairs vs Manual Wheelchairs – Advantages and Drawbacks of Each”, MED+, November 06, 2018. [Online]. Available:  
<https://www.medplushealth.ca/blog/power-wheelchairs-vs-manual-wheelchairs-advantages-and-drawbacks-of-each/>. [Accessed June 19, 2023].
- [6] “8 Common Barriers for Users of Wheelchairs”, Strong Go. [Online]. Available:  
<https://www.stronggo.com/blog/8-common-barriers-users-wheelchairs>. [Accessed June 19, 2023].
- [7] Elena Garcia, Maria Antonia Jimenez, Pablo Gonzalez-de-Santos, Manuel Armada, “The evolution of robotics research”, IEEE Robotics & Automation Magazine, April, 2007. [Online]. Available:  
[https://www.researchgate.net/publication/3344813\\_The\\_evolution\\_of\\_robotics\\_research](https://www.researchgate.net/publication/3344813_The_evolution_of_robotics_research). [Accessed June 19, 2023].

- [8] Francisco Rubio, Francisco Valero and Carlos Llopis-Albert, “A review of mobile robots: Concepts, methods, theoretical framework, and applications”, International Journal of Advanced Robotic Systems, March - April 2019. [Online]. Available: [https://www.researchgate.net/publication/332469850\\_A\\_review\\_of\\_mobile\\_robots\\_Concepts\\_methods\\_theoretical\\_framework\\_and\\_applications](https://www.researchgate.net/publication/332469850_A_review_of_mobile_robots_Concepts_methods_theoretical_framework_and_applications). [Accessed June 20, 2023].
- [9] Clearpath Robotics. "TurtleBot 4" Clearpath Robotics. [Online]. Available: <https://clearpathrobotics.com/turtlebot-4/>. [Accessed June 21, 2023].
- [10] “ROSbot ( 2R | 2 PRO | 2 )”, HUSARION. [Online]. Available: <https://husarion.com/manuals/rosbot/>. [Accessed June 22, 2023].
- [11] “Husky UNMANNED GROUND VEHICLE”, Clearpath Robotics. [Online]. Available: <https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/>. [Accessed June 23, 2023].
- [12] Jesse Leaman, Hung Manh La, “A Comprehensive Review of Smart Wheelchairs: Past, Present and Future”, IEEE Transactions on Human-Machine Systems, April, 2007. [Online]. Available: [https://www.researchgate.net/publication/316184407\\_A\\_Comprehensive\\_Review\\_of\\_Smart\\_Wheelchairs\\_Past\\_Present\\_and\\_Future](https://www.researchgate.net/publication/316184407_A_Comprehensive_Review_of_Smart_Wheelchairs_Past_Present_and_Future) . [Accessed June 25, 2023].
- [13] Harpreet Kaur Channi, Pardeep Kumar, Himanshi Sehgal, “Human Machine Interface (HMI)”, LAP Germany, October, 2019. [Online]. Available: [https://www.researchgate.net/publication/350890191\\_human\\_machine\\_interface](https://www.researchgate.net/publication/350890191_human_machine_interface). [Accessed June 26, 2023].
- [14] “Due”, Arduino. [Online]. Available: <https://docs.arduino.cc/hardware/duemk>. [Accessed June 26, 2023].
- [15] STMicroelectronics, “Data brief”, STMicroelectronics. [Online]. Available: <https://www.st.com/en/evaluation-tools/nucleo-f446re.html>. [Accessed June 26, 2023].
- [16] Martyn Currey, “HC05 and HC06 zs040 Bluetooth modules. First Look”, Martyn Currey, July 20, 2017. [Online]. Available: [https://www.academia.edu/27364779/HC\\_05\\_and\\_HC\\_06\\_zs\\_040\\_Bluetooth\\_modules\\_First\\_Look](https://www.academia.edu/27364779/HC_05_and_HC_06_zs_040_Bluetooth_modules_First_Look). [Accessed June 27, 2023].

- [17] COMPONENTS101, “HC-05 - Bluetooth Module”, COMPONENTS101, 16 July 2021. [Online]. Available: <https://components101.com/wireless/hc-05-bluetooth-module>. [Accessed June 27, 2023].
- [18] JUYI TECH, “JYQD-V7.3E3 3 Phase BLDC Motor Driver 15A Current PWM Speed Control”, JUYI TECH. [Online]. Available: <https://www.bldcmotor-driver.com/sale-11476358-jyqd-v7-3e3-3-phase-bldc-motor-driver-15-a-current-pwm-speed-control.html>. [Accessed June 29, 2023].
- [19] Md. Rifat Hazari, Effat Jahan, Md. Ettaker Siraj, Md. Tauhedull Islam Khan, Ahmed Mortuza Saleque, “Design of a Brushless DC (BLDC) motor controller”, ResearchGate, April 2014. [Online]. Available: [https://www.researchgate.net/publication/286581600\\_Design\\_of\\_a\\_Brushless\\_DC\\_BLDC\\_motor\\_controller](https://www.researchgate.net/publication/286581600_Design_of_a_Brushless_DC_BLDC_motor_controller). [Accessed June 30, 2023].
- [20] Don Johannek, “Using BLDC Hall Sensors as Position Encoders – Part 1”, DigiKey, February 19, 2019. [Online]. Available: <https://www.digikey.com/en/blog/using-bldc-hall-sensors-as-position-encoders-part-1>. [Accessed June 30, 2023].
- [21] “TMR 9-4819WI”, TRACOPOWER, June 29, 2023. [Online]. Available: [https://www.tracopower.com/sites/default/files/products/datasheets/tmr9wi\\_datasheet.pdf](https://www.tracopower.com/sites/default/files/products/datasheets/tmr9wi_datasheet.pdf). [Accessed July 02, 2023].
- [22] Olivier Stasse, “Robot Operating System”, LAAS-CNRS, 2016. [Online]. Available: <https://homepages.laas.fr/ostasse/Teaching/ROS/rosintro.pdf>. [Accessed July 04, 2023].
- [23] Yuhei Sugata, Takeshi Ohkawa, Kanemitsu Ootsu, Takashi Yokota, “Acceleration of Publish/Subscribe Messaging in ROS-compliant FPGA Component”, June 07 - 09, 2017. [Online]. Available: <https://dl.acm.org/doi/pdf/10.1145/3120895.3120904>. [Accessed July 06, 2023].
- [24] “Programming a robot with ROS”, AutomationWare. [Online]. Available: <https://automationware.it/ros-eng/?lang=en>. [Accessed July 07, 2023].
- [25] Richard Bolden, “Ubuntu”, The SAGE Encyclopedia of Action Research - Chapter: Ubuntu, August 11, 2014. [Online]. Available: [https://www.researchgate.net/publication/259849297\\_Ubuntu](https://www.researchgate.net/publication/259849297_Ubuntu). [Accessed July 09, 2023].

- [26] Peter MITROUCHEV, Radoslav Deliyski, Pencho Venkov, “Modélisation cinématique des robots mobiles par visio-algorithme”, August, 2007. [Online]. Available: [https://www.researchgate.net/publication/27610565\\_Modelisation\\_cinematique\\_des\\_robots\\_mobiles\\_par\\_visio-algorithme](https://www.researchgate.net/publication/27610565_Modelisation_cinematique_des_robots_mobiles_par_visio-algorithme). [Accessed July 12, 2023]
- [27] Liuping Wang, “Basics of PID Control”, Wiley-IEEE Press, March ,2020. [Online]. Available: [https://www.researchgate.net/publication/346653954\\_Basics\\_of\\_PID\\_Control](https://www.researchgate.net/publication/346653954_Basics_of_PID_Control). [Accessed July 14, 2023].