SELF DRIVING WHEELCHAIR FOR DISABLED PERSON BASED ON SIMULTANEOUS LOCALIZATION AND MAPPING

NUTT JATURAT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN BIOMEDICAL ENGINEERING
SCHOOL OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2024

KMITL-2024-EN-M-317-225

Thesis Title	Self Driving Wheelchair for Disabled Person Based on Simultaneous Localization and Mapping
Student	Mr. Nutt Jaturat
Student ID.	66016036
Degree	Master of Engineering
Program	Biomedical Engineering
Academic Years	2024
Advisor	Prof.Dr. Chuchart Pintavirooj

ABSTRACT

Wheelchairs are very common medical equipment that you all might see in various places. Wheelchairs come in a variety of types, depending on the type of patient who needs to use them. But if a patient has a disease that affects them, they can't use a manual wheelchair or an automatic wheelchair. So that is why our project will occur. This research focuses on patients who have diseases that affect muscles, such as Amyotrophic Lateral Sclerosis (ALS), Spinal Cord Injury (SCI), Multiple Sclerosis (MS), and Guillain-Barré Syndrome (GBS), which affect patients unable to control a normal type of wheelchair. This research designed a wheelchair that is suitable for those patients. Patients can control the wheelchair to their destination with only 1 click on their laptop, and they can also avoid objects. This wheelchair is suitable for patients who are living in a room or home and can create a map of it for navigation.

Our wheelchair uses the Robot Operation System (ROS) for control and the library Simultaneous Localization and Mapping (SLAM). SLAM is the method that builds the map from an unknown environment by using an RPLidar sensor to detect the distance between the object wall and our wheelchair. And use an ultrasonic sensor at the front of the wheelchair to help detect objects that are so low that the RPLIdar can't detect them and avoid them. For making maps with high accuracy This research needs to calibrate our wheelchair to have the highest accuracy in building maps and navigation as well.

This result of this project show that This research successfully able to building the map and navigation with and with out object during patient sit on the wheelchair. For using patient just click the destination on the map and our wheelchair automatically go to

destination by them self. The velocity of navigation at the first time will high but it will decreasing when it found object or when it reach the destination. Although the smart wheelchair can perform well there some limitation in building the map as This research know RPLidar is the laser it will go through the mirror wall and it effect to the map.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to Prof. Dr. Chuchart Pintavirooj, my project advisor, for his invaluable guidance and support throughout this project. It was his kindness, expertise, and willingness to provide constructive feedback that helped me complete this project successfully.

I would also like to extend my appreciation to the Biomedical Engineering Department of the Faculty of Engineering for providing me with the necessary research equipment and facilities, which were instrumental in completing this project.

Lastly, I would like to express my gratitude to all those who supported and assisted me during this project, including those who may not have been mentioned in this thesis. Their contributions have been invaluable to me.

TABLE OF CONTENTS

LIST OF TABLES

Tables page

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols /Abbreviations	Term
ROS	Robot Operating System
SLAM	Simultaneous localization and mapping
IoT	Internet of Things
PC	Personal computer
ALS	Amyotrophic Lateral Sclerosis
SCI	Spinal Cord Injury
MS	Multiple Sclerosis
GBS	Guillain-Barré Syndrome

CHAPTER 1
INTRODUCTION

This chapter opens with an overview of the statement of the problem. We then describe the project objectives and scope of the study.

1.1 Background of the study

Medical devices are essential to our daily lives because they facilitate patient care. Wheelchairs are also the most essential piece of equipment for these patients. Wheelchairs are one of the most common medical devices that can be found almost everywhere, but especially in hospitals, to assist patients who are unable or have difficulty walking to reach their destination. Wheelchairs come in various types, such as manual wheelchairs and electric wheelchairs; each type has a specific use. For patients who are unable to move certain body parts, the electronic wheelchair is the optimal choice. According to evidence and observation, 1 percent of the world's population required a wheelchair in 2010, with approximately 3.6 million people in the United States and 49 percent using a wheelchair at the Canadian Institute for the Elderly [1]. So now you can see the importance and popularity of using electronic wheelchairs for patients. The use of electronic wheelchairs has been on the rise as the number of elderly people increases. Our project needs to create a wheelchair that is suitable for patients who are unable to walk or are difficult to walk to reach their destination by themselves, and our wheelchair must be easily used.

As we know, wheelchairs come in various types, and each type has a specific type suitable for each patient. At present, there are many wheelchairs that are developed and controlled by using the brain, hand, eyes, et cetera [2-4]. to help patients easily control wheelchairs in other ways. But in our project, we need to focus on patients who have the ALS disease and have the problem that nerve cells can’t send commands to other cells, which will make other cells die and have the problem instead of being able to move their arms, legs, and fingers. It means the wheelchair our team needs must be suitable for ALS patients as well. So we must create a wheelchair that can go to the destination with a single click for patients who might be using it, and we must also avoid obstacles along the way.

1.2 Objective

1. To create a smart wheelchair that improves the quality of life for those with mobility impairments through wireless technology
2. To create a smart wheelchair that can avoid obstacles during navigation.
3. To create an easily controllable automatic wheelchair for patients.

1.3 Scope of study

1. Utilize Raspberry Pi hardware to control the overall system, along with RPlidar for navigation and map creation.
2. The Raspberry Pi will be connected to and controlled by a laboratory laptop during the operation of the system.
3. Able to create a map of the environment of interest and can control the robot in the map.
4. Test and calibrate the correct angular and linear motion with and without a load weight.
5. Wheelchair is capable of autonomous navigation to reach the destination.

CHAPTER 2

REVIEW OF RELATED LITURATURE AND STUDIES

This chapter explains the definition and functionality of a wheelchair. The programming techniques utilized in this project include Linux, Ubuntu, and ROS. In addition. The theory of components and some theory of wheelchairs will be reviewed in detail to better understand.

2.1 What is wheelchairs

The wheelchair is the chair that has a wheel to attach to bring the patient to their destination. The wheelchair has many types, like commonly used assistive products, to enhance mobility and improve quality of life for people who have difficulty walking or are unable to walk [5, 6].

Mobility devices are extremely important for patients who are unable to control their bodies while walking, as well as for patients who suffer from diseases that impair their ability to control their bodies. The mobility device not only allows the patient to travel to their destination independently, but also without the need for someone to care for them [6]. This demonstrates that mobility devices, such as wheelchairs, are extremely important for patients, not only in hospitals but also outside of them.

2.2 Wheelchairs

2.2.1 Type of wheelchairs

Wheelchairs are transportation assistance for individuals who are incapable of walking independently. The purpose of wheelchairs is to enable mobility to the desired location through the use of wheels. It is essential to select the appropriate wheelchair for the patient, as this will significantly impact the patient's safety and comfort during transfers to other places. Wheelchairs are currently integrating a variety of technologies to facilitate and support work. Today, This research can separate types of wheelchairs into two main 1.Maual wheelchair 2. Electronic wheelchair [7, 8]. and there are many types of wheelchairs that can break it down into many things. That is why there are many wheelchairs because we need to make them suitable for every patient who has various types of bodies and various diseases in this world right now.

2.2.1.1 Manual wheelchairs

Manual wheelchair: this is the first and most common wheelchair. We can see manual wheelchairs, for example, perhaps in the hospital or hotel, for helping patients and people. Manual wheelchair: this is not only used to help patients who can't walk but also to help patients who came to the hospital and have very nervous diseases and can't walk. A manual wheelchair is the best wheelchair for use in hospitals because it is lightweight and easy to maintain. The categories of manual wheelchairs include transport, standard, folding, rigid, and dynamic tilt [9].

1. **Transport wheelchair**

A transport wheelchair is designed to be used by an individual patient who doesn't require a wheelchair in their daily life. This type of wheelchair is used to transport a person from one location to another when they are unable to walk [9].

2. **Standard wheelchair**

Standard wheelchairs are more adaptable in nature than transport wheelchairs. Adjustable in height, the arm rests permit the upper extremities to be set appropriately. Adjustable in height, the foot rests guarantee the lower extremities are in the correct position. Additionally, the rear wheel position is a little adjustable. This facilitates a modification in the vertical distance between the seat and the floor, a factor that may prove critical when it comes to environment access, foot propulsion, and transfers. Vertically adjusting the rear wheel position permits a modification in the seat angle, resulting in a stationary tilt of the wheelchair. This feature potentially facilitates upper extremity propulsion by enabling access to the rear wheel. [9].

3. **Folding wheelchair**

Folding a wheelchair is lighter than a wheelchair. The difference between a folding wheelchair and a standard wheelchair is that we can adjust the placement of the rear in both vertical and horizontal [9].

4. **Rigid wheelchair**

Rigid wheelchairs are lighter than folding wheelchairs and don't have a cross brace to help reduce their weight. Rigid wheelchair design makes it easiest for patients to control the wheelchair [9].

2.2.1.2 Electric wheelchair

Electric wheelchairs are made to be comfortable for patients while traveling; patients can control the wheelchair by themselves with a joystick on the arm rest. Electric wheelchairs still have many types, like manual wheelchairs, to make them comfortable for everyone, but electric wheelchairs have the disadvantage of having a heavy weight on them, but they still have the manual push option.

1. Transportable electric wheelchair

Also known as travel wheelchairs, they are designed to be easy to carry to any place. You can take it on a car, train, or other vehicle.

2. Indoor electric wheelchair

The application of this wheelchair is its light weight because it is made of a light frame [9]. Indoor wheelchairs can glide over every surface, whether carpeted or tiled.[10]

From the previous example, the wheelchair is present in the world right now. There are several varieties, as it depends on the patient's object and the conditions in which it must be used.

2.3 Disease that force patient need to using wheelchair.

Disease is a harmful deviation from the normal structure or functional state of an organism. When you get a disease, you will exhibit signs or symptoms of its abnormal state [11]. There are many diseases that force people to use wheelchairs, such as Amyotrophic Lateral Sclerosis (ALS), Spinal Cord Injury (SCI), Multiple Sclerosis (MS), and Guillain-Barré Syndrome (GBS).

1. Amyotrophic Lateral Sclerosis (ALS)

It is a disease caused by the degeneration of the nerves. There is a chance of approximately 20% from the family [12]. Progressive degeneration of upper and lower motor neurons is the defining feature of this incurable disease, which causes respiratory failure and death within an average of two to three years. The cause of ALS disease is still largely unknown[13].

2. Spinal Cord Injury (SCI)

This research suggests that SCI injuries will have profound repercussions on our daily lives. Mobility and automatic function are also impacted by SCI injuries.[14]. So in SCI, if you

get a cervical level 1 to 3 spinal cord injury, the muscles of the torso, arms, and legs will all be paralyzed. Cervical level 4 paralyzes the muscles in your torso, arms, and legs. On cervical level 5, it will force your torso and leg muscles. Totally paralyzed [15].

3. Multiple Sclerosis (MS)

Multiple sclerosis results from autoimmune disease-triggered environmental factors that act on a genetically susceptible host. MS has three stages: the pre-clinical stage, which can be detected by only MRI; the relapsing-remitting stage; and the progressive stage, which evolves from the relapsing stage [16]. MS, a progressive disease with or without relapses [17], can affect one's ability to move freely and easily.[18]

4. Guillain-Barré Syndrome (GBS)

GBS is the syndrome that most severely affects paralytic neuropathy. In 1 year, patients who will have GSB will occur in 100,000 per year [19]. GSB can probably effectively improve the mobility of patients for moving in daily life.

Many diseases that were already talked about in the previous section will have an effect on mobility in daily life. Now we will know the necessary wheelchair that is effective for a patient who has a disease that impacts their mobility right now. Our project is still focusing on ALS patients, or Amyotrophic Lateral Sclerosis. ALS is the disease that will occur in normal daily life, and many people have problems with it. We are looking to build and develop wheelchairs suitable for ALS patients who have weak muscles to use our wheelchairs easily.

2.4 Developing wheelchair

In this section, we will look at wheelchairs that have already been developed for patients in the past, how they are appropriate for patients, and how patients can use those wheelchairs in their daily lives. An example of developing a wheelchair for disabled people is a smart wheelchair based on eye tracking. This wheelchair was developed to be able to be controlled by using eye tracking. Designed for people who have locomotor disabilities. This wheelchair includes four modules. 1.imaging processing module; 2.wheelchiar-controlled module; 3. SMS manager module; and 4. appliance-controlled module. The principle of eye tracking in wheelchairs is to capture images, which are then transmitted to proceed using

OpenCV. The coordinates of eyeball movement are then wirelessly transmitted to wheelchair control modules to control wheelchair movement [4].

The wheelchair still has to use the brain to control it. This research used computer vision (CV) and augment reality (AR) with a brain control wheelchair (BCW) and proposed the CVAR-BCW, controlled by using a translucent head-mounted display (HMD) for an automatic detection environment and showing the detected targets through AR technology. After the user chooses the destination, a wheelchair can automatically navigate to it.[2]

A wheelchair for handicapped people conducting body motion is still interesting. Also, this wheelchair is able to be operated by a hand movement device and a smart phone, making it suitable for people who are unable to walk without other people helping. The user will wear a gesture system in their hand, and when they move their hand, their wheelchair will move in line with their hand movement.[3]

There are many research papers for wheelchairs that have been developed in the past and are still being developed in the future. This means wheelchairs are very necessary for people, and many people need to use wheelchairs easily. Electric wheelchairs now have many ways to control them and can make them go to destinations, but there are still a few wheelchairs that can go to destinations by themselves; they just select the area to which they need to go. So it means that if our wheelchairs are able to go to destinations by themselves and are able to avoid obstacles, this wheelchair can develop in the future to help the lives of patients who have a hard time walking to destinations.

2.5 Linux

Linux is an operating system just like IOS and Mac OS. In fact, a Linux operating system powers Android. An operation system is a piece of software that manages the communication between software and hardware. Why is Linux popular? Linux has evolved into one of the most reliable computer ecosystems on the planet. Combine that reliability with zero cost of entry, and you have the perfect solution for a desktop platform [20].

2.5.1 Python

As we know, Python is a programming language like other programming languages. A programming language is a set of instructions written by a programmer to deliver instructions to the computer to perform or accomplish a task. [20]. Now there are many different

computer languages, such as Java, LISP, PHP, and Perl, and don't forget C. Each programming language has a distinct strong point, like Java, which is easily portable programming. PHP can access databases and splice them into webpages in its specialism for PHP. The strong point of Python is that it's easy to understand, and it's also free software with a large and friendly community of developers around it. [21]

2.5.2 Ububtu

Ubuntu is a popular virtual private server or personal computer operating system that is free and open-source. The British corporation Canonical introduced Ubuntu in 2004. Ubuntu is extremely well-liked due to its simple interface. Ubuntu's interface is simple and easy to use. Ubuntu protects against breaches with AppArmor and other complex security measures. An extensive assortment of applications, many of which are exclusive to the operating system, are available for installation on Ubuntu. Ubuntu enables users to modify privacy settings while implementing a stringent policy regarding the protection of personal information. The default Ubuntu interface operates with a minimal system footprint, requiring no more than 1 GB of RAM. Consequently, the operating system functions on low-end hardware. Ubuntu is an open-source Linux distribution that is provided at no cost to users.[22]

2.6 Robot operation system (ROS)

The importance of this for control robots is to build programs for control robots. But it is very difficult and takes a lot of time to do it. But ROS can solve this problem. ROS, or Robot Operation System, is designed so that users can easily control robots by themselves. The definition of ROS is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms and powerful developer tools, ROS has what you need for your next robotics project. And it's all open source [23]. ROS is primary for the development of robots, but there are other goals of ROS.

- Thin: ROS is designed to be as thin as possible—we won't wrap your main()—to enable the portability of code developed for ROS to other robot software frameworks. An additional consequence of this is that ROS can be readily integrated with various robot software frameworks. ROS has been integrated with Player, OpenRAVE, and Orocos so far.

- Language independence: Implementing the ROS framework in any current programming language is simple. It has been implemented in Python, C++, and Lisp at present, and experimental libraries have been developed in Java and Lua as well.
- Easy testing: ROS has a built-in unit/integration test framework called rostest that makes it easy to bring up and tear down test fixtures.
- Scaling: Major runtime systems and development processes are suitable for ROS.[24]

Conclusion ROS's primary purpose is to make code reuse in robotics research and development easier. A distributed process framework called ROS enables runtime-loose connectivity and the execution of distinct program designs. ROS organizes these operations into stacks and packages, making them easy to divide and distribute. Additionally, ROS enables code aggregation, which speeds up the distribution of labor. All of these features can be integrated with ROS infrastructure tools [24].

2.6.1 Navigate

Navigate Stack is simple and conceptual; it takes information from odometry and sensor streams and outputs a velocity command to send to a mobile base. For navigate stack use, the robot must be running ROS, have a TF transform tree in place, and publish sensor data using the correct ROS Message type. Navigate Stack is designed to be as general as possible, but there are three main requirements that restrict its use [25]:

1. It is meant for both differential-drive and holonomic-wheeled robots only. It assumes that the mobile base is controlled by sending desired velocity commands in the form of: x velocity, y velocity, and theta velocity [25].
2. The mobile base must have a planar laser installed somewhere. This laser is used for map-building and localization [25].
3. Developed on a square robot, the Navigation Stack performs best on nearly square or circular robots. It does work on robots of arbitrary shapes and sizes, but it may have difficulty with large rectangular robots in narrow spaces like doorways [25].

2.6.2 move_base

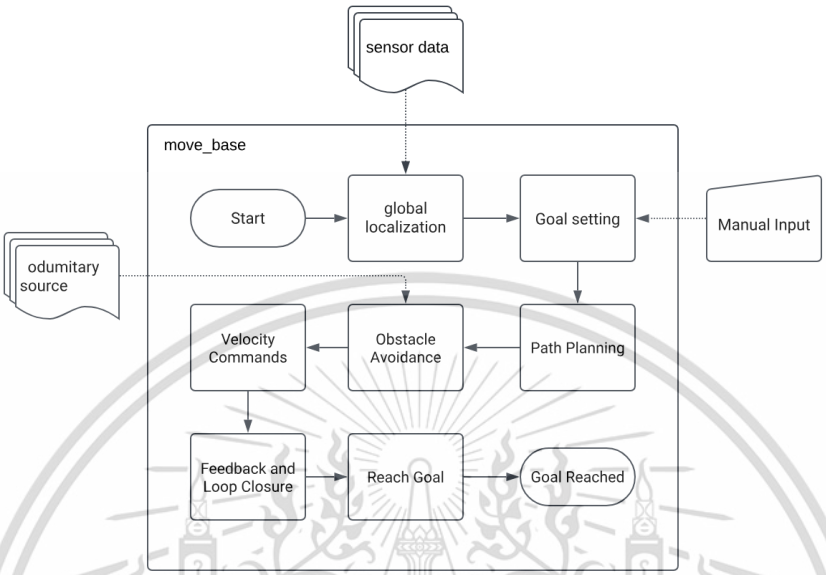


Figure 2.1 Diagram of move_base

The move-base package offers tools for carrying out actions that target the outside world. A node provides access to a function. The navigation core supports any global planner. The Global Planner, as defined in the package nav core, is the move base node that links a global and local planner to navigate a task. To carry out navigation tasks, the node move base also stores two costmaps, one for the global planner and the other for the local planner[26].

In order to configure, run, and communicate with the navigation stack on a robot, the move base node is preparing a ROS interface. The motion is seen at a high level in Figure 2.1 and is interacting with other elements. The blur red nodes are dependent on the robot platform; the gray nodes are a choice but must be offered for all systems, while white nodes must also be provided [26].

2.6.3 Gmapping

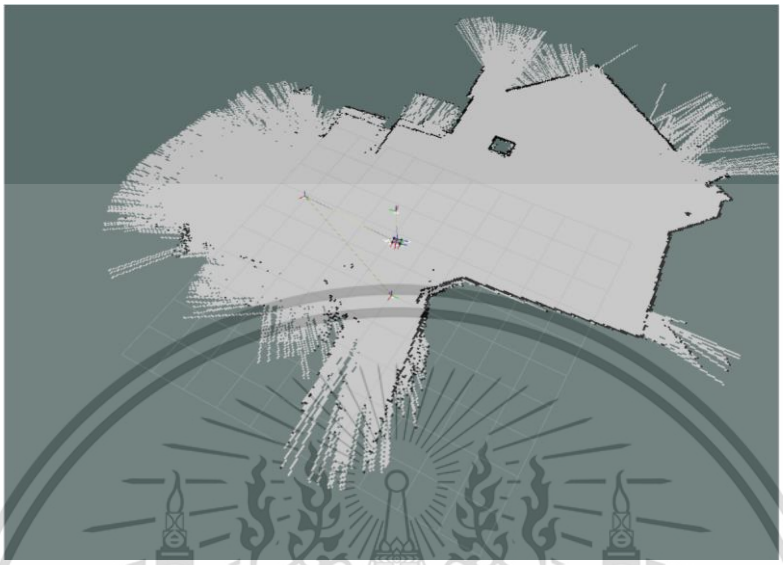


Figure 2.2 Example map from Gmapping

A ROS wrapper for OpenSlam's Gmapping can be found in this package. Simultaneous Localization and Mapping (SLAM) using lasers is offered by the gmapping package as the ROS node slam gmapping. With the use of slam gmapping, laser and posture data gathered by a mobile robot can be used to produce a 2-D occupancy grid map.[27]

2.6.4 Teleop

```
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
anything else : stop

CTRL-C to quit
```

Figure 2.3 Show Teleop application [28]

When you press the letter “I” on your keyboard, Teleop sends the order to the robot's motor, causing it to move forward, as seen in Figure 2.3 . Teleop functions like a

joystick. As with the arrow, you can control your robot by pressing any other button. You can change the linear speed by pressing W to speed things up or X to slow things down. Additionally, you can vary the angular speed by pressing e to speed up and c to slow down. The greatest component to determine whether your motor will operate properly is the teleop [28].

2.6.5 Simultaneous Localization and mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) has mechanisms to make the slam easy to navigate. The robot generates a map through a range of environments, collectively called SLAM, with sensors used in SLAM such as laser sensors, sonar sensors, and vision [29]. SLAM is the library of the ROS that received sensors to detect the environment and to build maps in any place that was possible for building and using maps to navigate the robot to its destination and avoid obstacles.

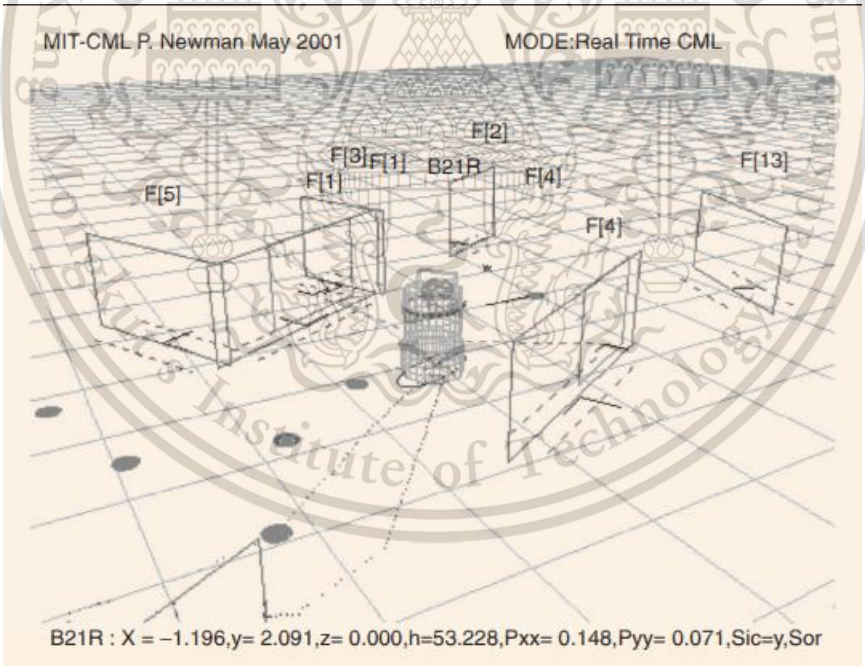


Figure 2.4 A real-time SLAM visualization by Newman [30] .

2.7 Component of Smart Wheelchair

2.7.1 Raspberry Pi

The Raspberry Pi is a low-cost session border controller (SCB). Raspberry Pi, first released in 2012, has several generations, which can be categorized into three distinct models: Raspberry Pi A,B, and Zero. The fundamentals of these Raspberry Pi's are mostly similar; each model has a system-on-ship that consists of an integrated central processing unit (CPU), an on-chip graphics processing unit (GPU), on-board memory, and a power input of 5 V DC. All models also have a port to connect a dedicated camera, as well as an array of general-purpose input/output (GPIO) pins that can be used to communicate with a wide range of electronics, from LEDs and buttons to servos and motors, power relays, and a huge range of sensors. General models also feature an Ethernet connection, which, when used in combination with GPIO ports, gives the Raspberry Pi huge versatility. The Raspberry Pi has all the functions of a standard computer; you can connect a mouse, keyboard, and screen without any configuration and have control over a Linux desktop environment or other operating system [31].

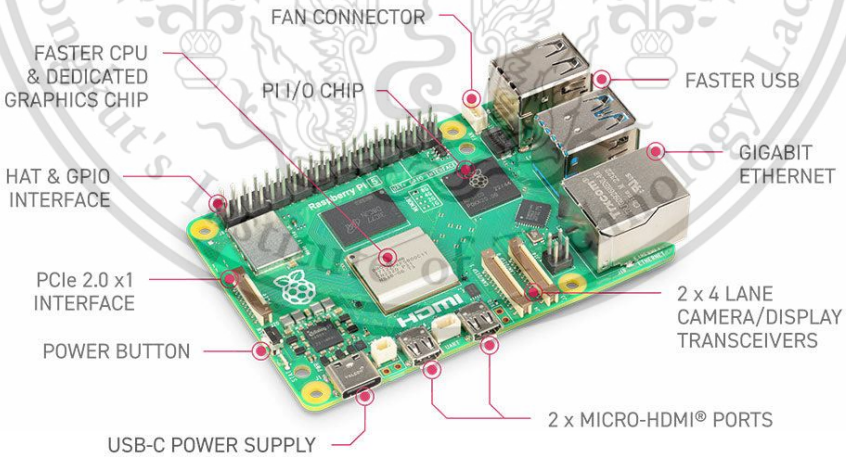


Figure 2.5 Raspberry Pi 4 [32]

The Raspberry Pi has been purposely built as a mini computer at a fraction of the low cost of a PC to encourage everyone to solve problems creatively. That means the Raspberry Pi is a great research and development tool that can be used for almost anything, including

autonomous environmental monitoring or recording of laboratory experiments or other activities that use computer technology to develop programs [31].

Raspberry Pi has been used for many IoT projects, including constructing a desktop PC. An example of this research is a novel approach to constructing timed-release encryption. In using the Raspberry Pi to control everything in this research, the Raspberry Pi can still be turned into a MySQL database server to store sensor data [33]. Or using Raspberry Pi to make a wireless solution for developing Raspberry Pi workshops [34].

We used the Raspberry Pi as the primary component in this research, installing Ubuntu and running ROS software. We use the Raspberry Pi to run every single command, such as bring up, teleop, gmapping, navigation, etc. We control the Raspberry Pi via remote control on a laptop.

2.7.2 Rplidar

The RPLidar is a laser device designed by SLAMTEC that utilizes laser triangulation ranging, high-speed vision acquisition, and processing hardware to detect and analyze environments. The RPLidar device emits a modulated infrared laser signal, which is subsequently reflected by the object it is directed at [35].

Figure 2.6 The Obtained Environment Map from RPLIDAR A1 Scanning [35]

RPLidar is commonly utilized in numerous projects, primarily for the purpose of obstacle avoidance. RPLidar utilizes simultaneous localization and mapping (SLAM) to construct a map of an unfamiliar environment [35, 36].

In this research, RPLidar is used to build the map from an unknown environment in the area that we need to make the map by letting it scan the area around it. And RPLIdar is something I will use to avoid obstacles during the navigation part also.

2.7.3 Encoder

An encoder is a sensor that offers feedback. An encoder transforms the movement into an electrical signal that can be interpreted by various control devices in a motion control system. The encoder will transmit a feedback signal that can be utilized to ascertain the position, count, speed, or direction[37]. An encoder transforms physical motion into electrical data by employing a range of technologies. The encoder consists of a photosensor and an LED positioned on opposite sides. The motor shaft is linked to a hub disk, which is accurately positioned to allow its see-through disk to move across the gap between the LED and photo sensor of the module. The presence of engraved lines on the disc hinders the transmission of light from the LED to the sensor. However, this is not the case when it comes to the spaces that divide the lines. The encoder produces a direct current (DC) voltage of 0 volts when the light is obstructed, and 5 volts when the light reaches the sensor [38].

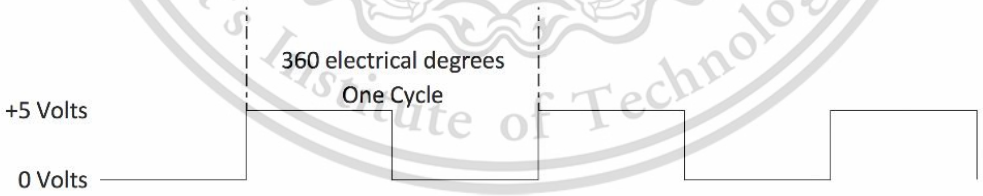


Figure 2.7 Encoder report 5 volts and 0 volts when encoder rotation [38].

Joint torque sensing is a fundamental aspect of contemporary robotic systems [39]. Encoders are widely used in advanced robotics to detect external force and torque and determine the characteristics of the force applied to the manipulator [40].

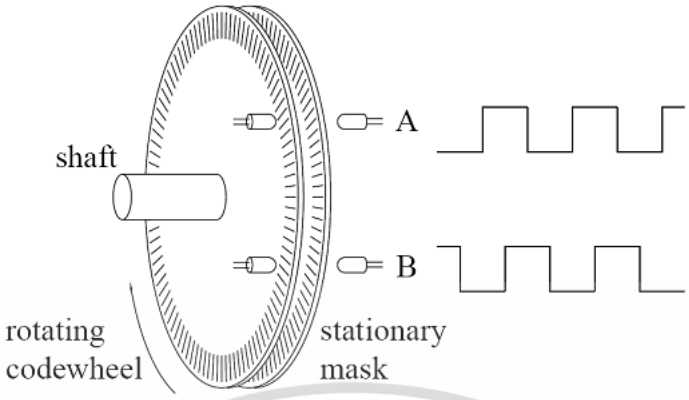


Figure 2.8 A rotary encoder generates pulses [41]

Figure 2.8 shows the device of the encoder, consisting of the disk and the sensor detection. The disk of the encoder will have a small hole, and when the motor rotates, it will force the disk of the encoder to rotate to cut the light of the sensor and let the detector receive the signal in a period that will create the signal into pulses A and B, as shown in figure 2.8 [42].

Pulse A and B tell the motor which way to go. If pulse A comes before pulse B, the motor will move forward [42]. If pulse B comes before pulse A, the motor will move backward. We are able to calculate the motor's rotational distance by measuring the voltage generated by the encoder. For instance, if the encoder records 360 pulses during a single motor round, we can utilize these pulses to compute the motor's rotational distance through using the robot's wheel circumference and the pulses it produces during that complete rotation. The encoder counts the number of pulses per unit of time to determine the motor's speed as well.

An encoder is used for the detection of wheels while they are moving and to collect the value while moving to know the distance when we are moving right now. The value of the encoder is dependent on the direction of the wheelchair, while when the wheelchair goes forward, the value of the encoder will increase, and when the wheelchair moves backward, it will decrease.

2.7.4 MPU 6050

The MPU-650 sensor is equipped with a three-axis accelerometer and gyroscope, as shown. The MPU6050 is frequently employed to detect acceleration on each axis in three-axis MPU accelerometers. If the device is tilted or in a state of weightlessness or excessive

weight, the corresponding measurement will be altered. The acceleration value is determined by converting the reading from the accelerometer's reading range to the measuring range. Three-axis gyroscopes function based on the principle of Coriolis acceleration. When an effort is made to tilt this arrangement, the crystals experience a force that operates in the direction of the inclination. This phenomenon occurs because of the inertia displayed by the moving. Consequently, the crystals produce an amplified current due to the piezoelectric effect [43]. The camera primarily utilizes an MPU, which is less susceptible to hand movements and shocks [44].

The MPU-6050 can detect the movement and rotation of the wheelchair on various axes. This information can be used to estimate the robot's position and movement. It can be used to improve, map, and coordinate robot movements in different environments.

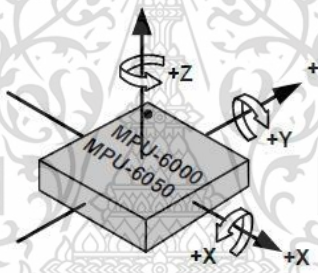


Figure 2.9 Show axis of MPU 6050 [43]

2.7.5 Motor drive

Motor drive is employed to control the orientation, velocity, and rotational force of electric motors, thereby enhancing the functionality and effectiveness of machinery. Motor drive works have a wide range of industrial applications where they enhance motor performance by converting electric energy and adjusting frequency, voltage, or phase [45].

The L298N is a type of motor controller that operates by controlling the direction and speed of two DC motors. The L298N module consists of two channels, which are labeled as A and B in the diagram. The L298N is outfitted with a circuit that is compatible with the Arduino Uno. The L298N module functions as a dual H-bridge motor driver, allowing for precise control over the direction and speed of two DC motors. The H-bridge circuit has the ability to control both the polarity and pulse width. The L298N motor driver controls both the speed and direction of rotation of a DC electric motor. This is achieved by using an

L298N PWM system, which uses square wave pulses to control voltage. As the pulse width increases, the motor will rotate at a higher frequency [46].

This research utilized the L298N motor driver module to control the motors of our wheelchair, receiving commands from an Arduino Mega board. The L298N operates with an H-bridge configuration to enable control of both wheels of the wheelchair. This allows for independent speed control of each wheel, suitable for differential movement.

Figure 2.10 Show L298N [47]

2.7.6 Ultrasonic sensor

For ultrasonic sensors to operate, it is necessary to produce a sound wave that surpasses the human hearing range. Similarly to a microphone, the sensor's transducer receives and transmits ultrasonic sound. Utilizing a single transducer, our ultrasonic sensors produce and receive an echo, similar to various other sensors. The sensor determines the distance to a target by analyzing the time intervals between the sending and receiving of the ultrasonic pulse [46].

This research uses a ultrasonic sensor to help avoid obstacles during navigation. As we know, RPLidar is also used to avoid obstacles, but it can only detect them in the 2D dimension. If an object that lowers RPLidar can't be detected, then we used ultrasonics to help detect that object.

Figure 2.11 Ultrasonic Sensor [47]

2.7.7 Arduino Mega Board

The Arduino Mega Board is a powerful microcontroller capable of working in a variety of applications. The Arduino Mega Board is a suitable platform for data collection, as it features inputs and outputs that can be utilized to process the information gathered from the sensor in use. In order to set up communication with the Arduino Mega Board, it is necessary to build code from our laptop in the Arduino IDE (Integrated Development Environment), which is the software utilized to develop the code [48].

This research used an Arduino Mega Board for receiving the sensor data of the MPU6050 and an encoder that will collect the value while navigation and controlling motor L298N through the command from the Raspberry Pi.

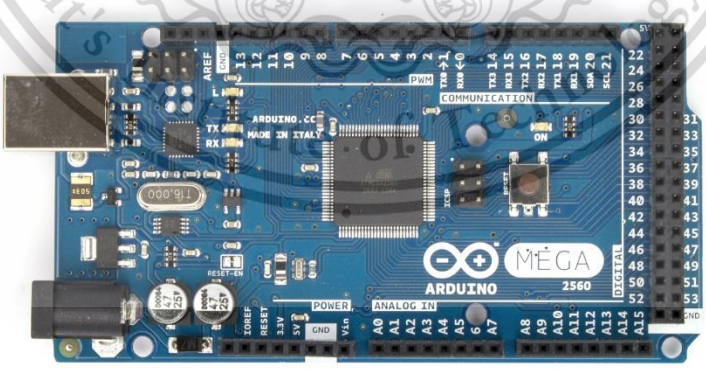


Figure 2.12 Show Arduino Mega Board [49]

CHAPTER 3

METHODOLOGY

3 .1 Introduction

This chapter provides a comprehensive description of the methodology, including the testing function and the overall process of this project. The chapter commences by delineating the operational phase of each process in Section 3.2. The subsequent component of the overall material utilized in this endeavor is hardware. Integrate a complete Raspberry Pi setup, which is the most valuable aspect of this undertaking, to manage every component.

3. 2 Action plan

	2023															
	June				July				Aug				Sep			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Design our wheelchair.																
Building box control for our wheelchair																
Change the motor of the wheelchair																
Connect the RPLidar and encoder.																
Setup software for Ubuntu																
Test software																
Calibration test																
Collect the map and navigation tests.																
	2024															
	Jan				Feb				Mar				Apr			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4

Add an ultrasonic sensor.															
Motor test															
Move base test and set up															
Navigation test															

3.3 Component of Smart Wheelchair

This project consists of two parts: the first part is the control system, and the second is the hardware part, which is the wheelchair. The control system is the system that is used to control the wheelchair to navigate to the destination and avoid obstacles while going there.

- Device part
  - Automatic Wheelchair
  - Raspberry Pi4 8GM RAM
  - RPLidar A2
  - MPU 6050
  - Motor drive
  - Arduino mega board 2560
  - Motor drive 24V
  - Encoder
  - Battery 24 V 12 A
  - Ultrasonic
- Software part
  - Ubuntu 20
  - Linux 20
  - Robot operation system (ROS)
  - simultaneous localization and mapping (SLAM)

3.4 Design a smart wheelchair

Figure 3.1 shows the automatic wheelchair that was designed for patients who had problems walking. To develop a wheelchair, we need to replace the new motor, which can be controlled by a motor drive. Attached encoder to motor of wheelchair for detect while

motor moving. On top of the wheelchair was a RPLidar that was used to detect objects around them. We made a box control that had all the components that were required to control the wheelchair for navigation.

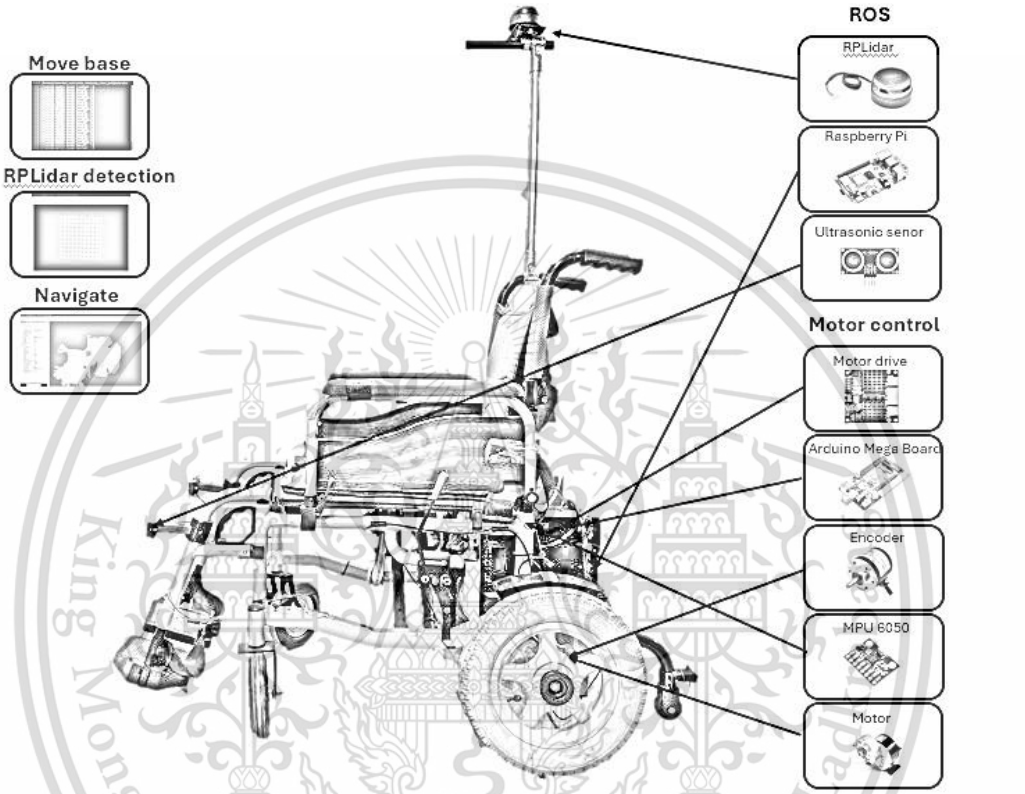


Figure 3.1 Design of smart wheelchair

3.5 Design electronic circuit of control system

The control navigation system had a Raspberry Pi for the control device and controlled every sensor that was used in the wheelchair. The Raspberry Pi worked like a computer, and we remotely controlled it by connecting it to our laptop, which made it easy to control the Raspberry Pi. Figure 3.2 shows the circuit of the control system for all the components that we attach to it. Start at the battery and give the power supply to the Raspberry Pi, motor drive, and Arduino Mega Board. Raspberry Pi, as we know it, received sensor data directly from the RPLidar, Ultrasonic, and Mega Board. The Mega board was used

for the control motor, received sensors from the IMU and encoder, and had a Raspberry Pi control them.

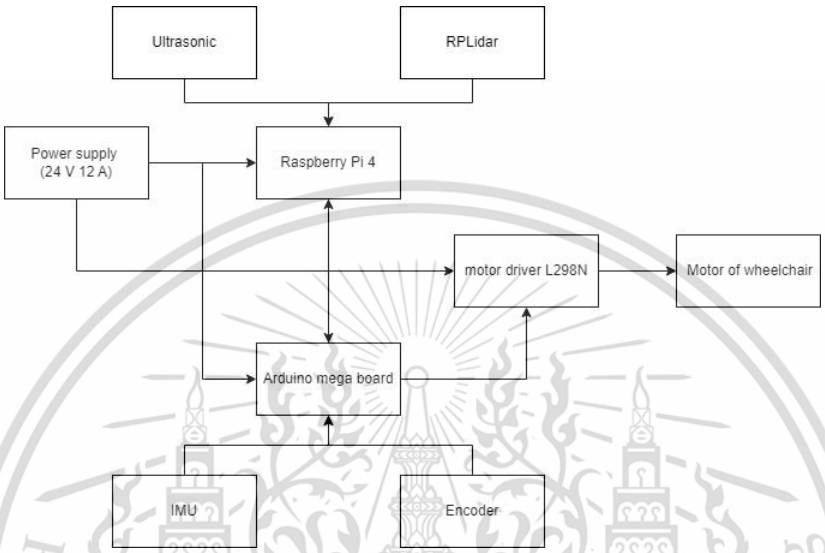


Figure 3.2 Diagram of control system.

3.6 Software Process of control system

3.6.1 ROS setup

We used the Raspberry Pi as the computer to control everything in this project. The Raspberry Pi will run the SD card that we prepared for the interface of Ubuntu that we are using, as shown in Figure 3.3. After When we can operate Ubuntu, we must set up the wifi to make a remote desktop connection (a remote desktop connection is an application in the window that we can use to type an IP address from another computer to control it) by installing ssh-server and nettles. Then we installed ROS and the library that we needed to use.

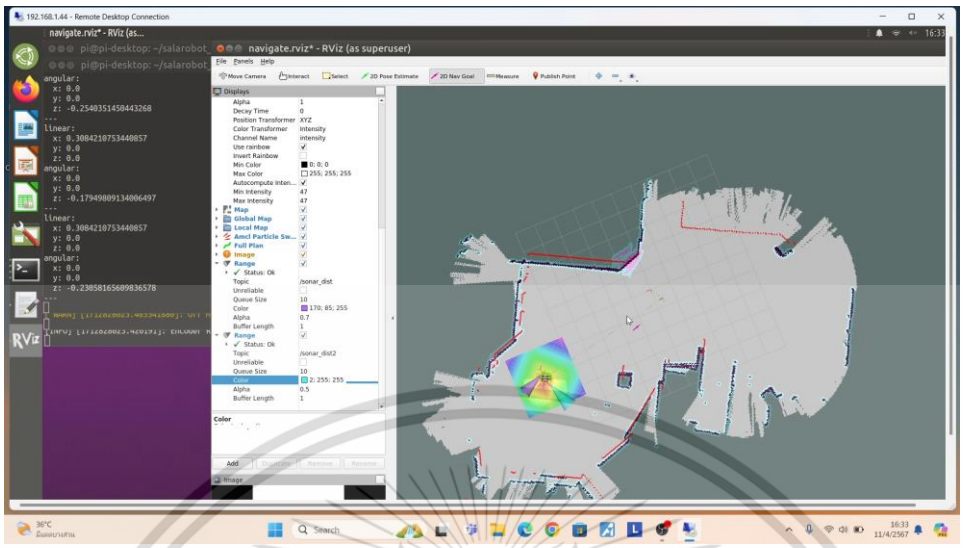


Figure 3.3 Show remote control from laptop

3.6.2 Bringup setup

Bringup is the first part of operating a robot; generally, “Bringup” is the process of starting a module or robot for ready use. Bringup can be used for testing connections between equipment, setting parameters, and controlling other sensors. Bringup is the setup; the first thing that we concern ourselves with in Bringup should show our encoder scale.

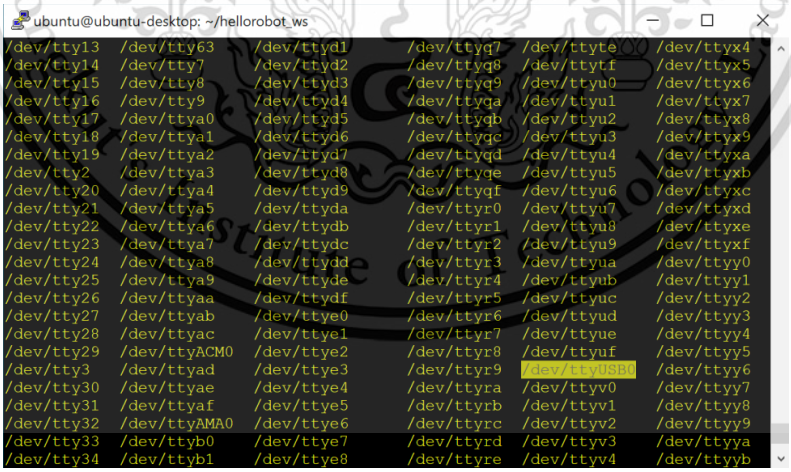


Figure 3.4 Port of Arduino mega board

First of all, we need to make an Arduino Mega Board that can receive the amount of sensor from the imu and encoder sensor that we are using. So we plug in the Arduino Mega Board and find which port of the Arduino Mega Board is shown in Figure 3.4. After we know

which port of the Arduino Mega Board we are able to upload code to, the Raspberry Pi can connect to the Arduino Mega Board. Then we need to upload code that is used on the Raspberry Pi for receiving the amount of encoder by uploading lino_base_config on the Mega Board. Lino_base_config is the configuration of our motor drive. Before we upload, we need to know the pin of the motor. When we do everything, we will run the encoder that is shown in Figure 3.5. That means our Raspberry Pi can receive the amount of value from the encoder when the wheels are rotating.

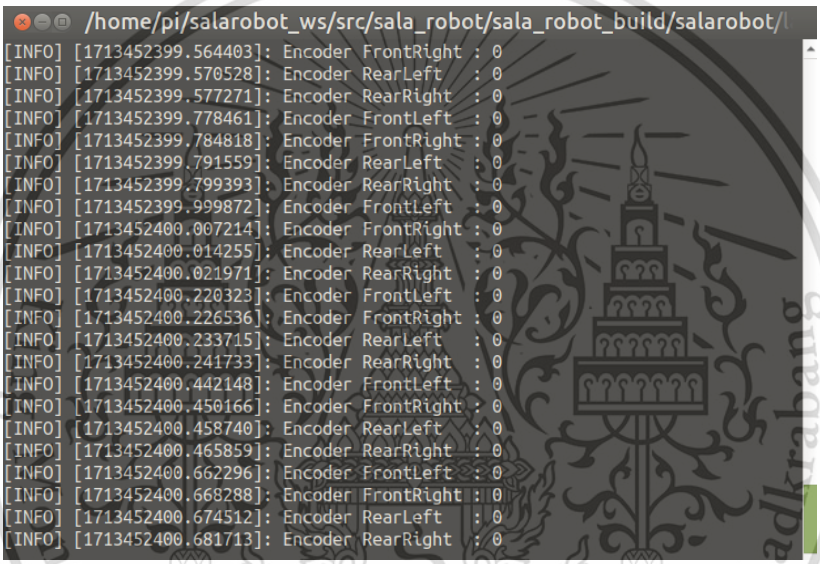


Figure 3.5 Show amount of encoder

So when we can receive the amount of encoder, we need to check if the amount of encoder that we received is correct or not by testing to move the wheel forward and backward. The value of the encoder will increase when the wheel moves forward and decrease when it moves backward. For you to receive the value of the encoder, you need to test how many values there are when the wheels are rotated in one round, because when you set it up, you need to know the correct value.

After we know our encoder is able to be used, we need to check the IMU by running “brinup” and “rostopic echo imu/data_raw." This means we need to check if our IMU is working or not, as shown in Figure 3.6.

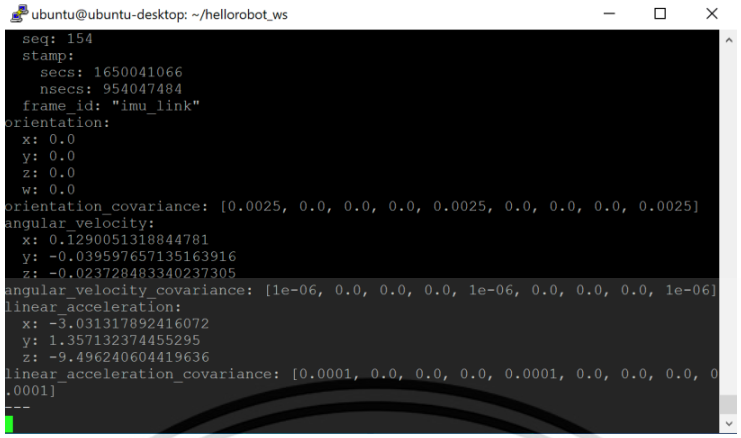


Figure 3.6 Show IMU is working

The next thing that we need to test is RPLidar. First, RPLidar can project the laser around them or not, so if you can connect to RPLidar, you will see in Fig. The red line in Figure 3.7. means RPLidar projects later to it and receives the distance between them and the object. When RPLidar is working, you must check that the configurationof the RPlidar is in the correct position by using your hand cover around RPLidar and observing the red lessor line come to really close to the middle of it.

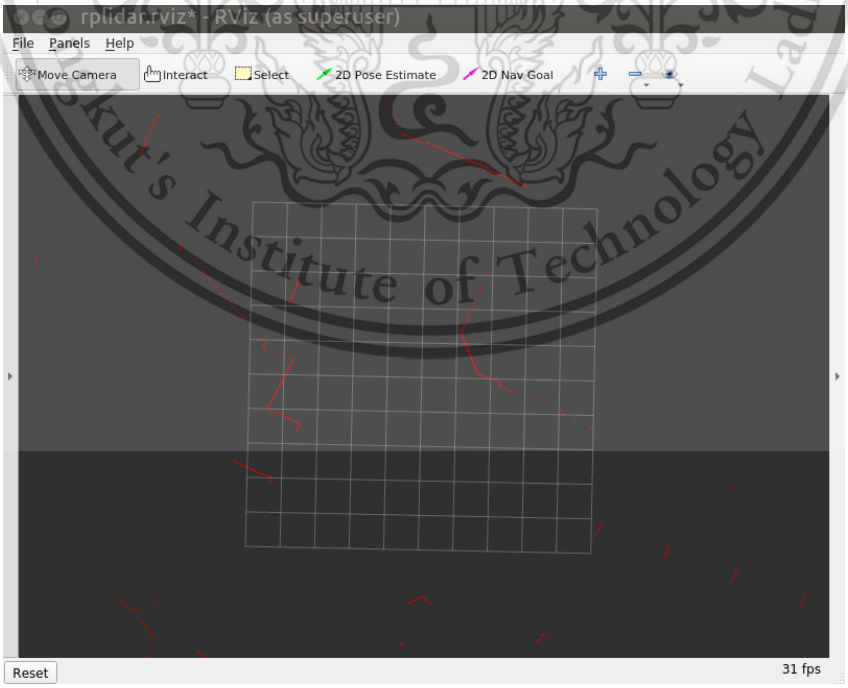


Figure 3.7 Show the RPLidar detection environment around RPLIdar

3.6.3 TF setup

TF set-up is also a very important thing for setting up a wheelchair to know where the robot is right now. The TF that you need to setup are the RPLidar, IMU, ultrasonic, and wheelchair positions that are shown in rviz in Figure 3.8.

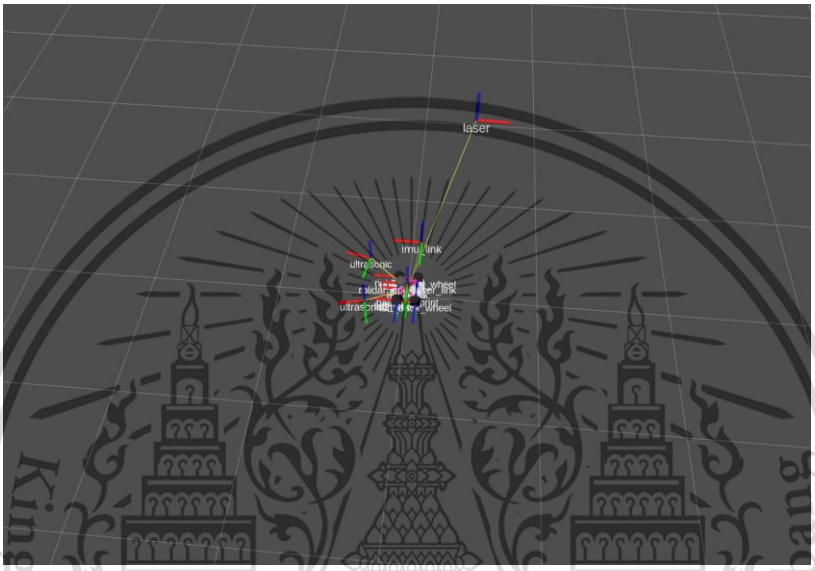


Figure 3.8 Show TF of our robot model and other sensor

3.6.4 Ultrasonic setup

In an ultrasonic setup, you need to be able to connect to the ultrasonic first, as shown in Figure 3.9., which shows the range of objects that our ultrasonic is able to detect.

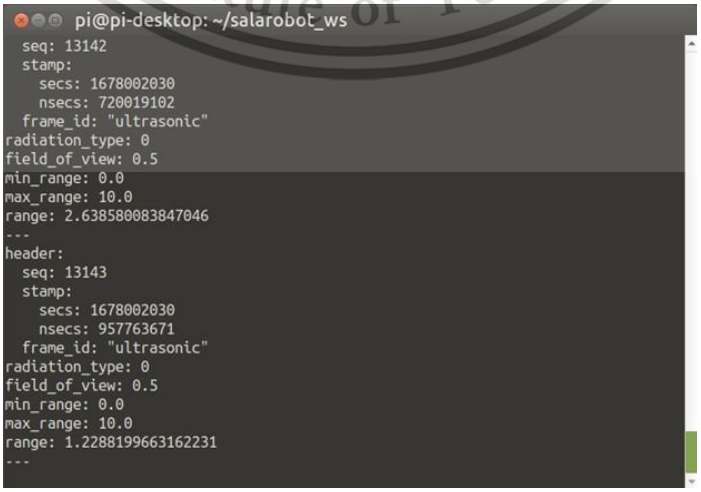


Figure 3.9 Show range of ultrasonic detection

In this Figure 3.10, we show the range of ultrasonic in rviz.

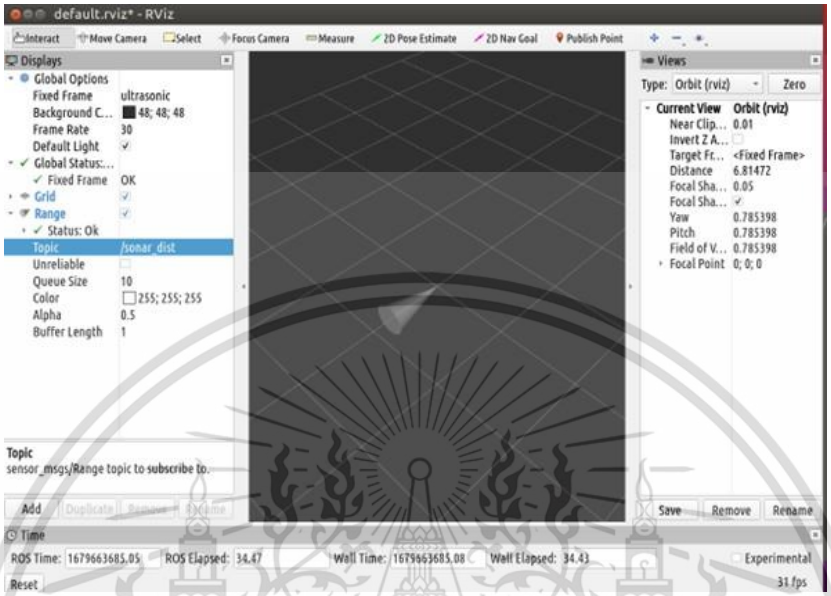


Figure 3.10 Show ultrasonic range in RVIZ

3.6.5 Calibration setup

The next thing that we need to test is the calibration of our wheelchair to know if it has accuracy before we go to correct the map. If we don't do a calibration test when we collect the map, our wheelchair will not be in the correct position on the map, and it will also have an effect on navigation as follows:

Linear test

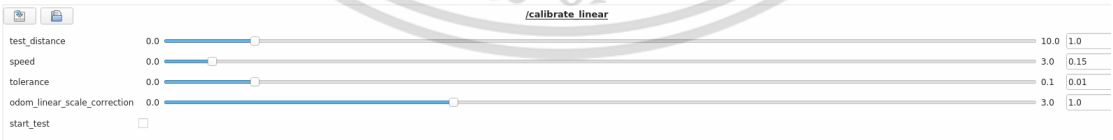


Figure 3.11 Show setup of linear calibration

Angular test

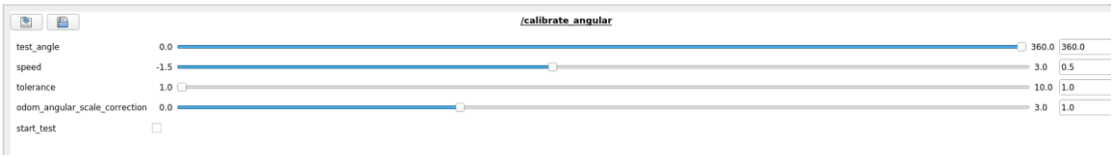


Figure 3.12 Show setup of angular calibration

3.6.6 Gmapping setup

After we calibrate in linear and setup TF, next is what we call "gmapping," which is building the map from an unknown environment by using an RPLidar sensor. When we run the gmapping shown in Figure 3.13, which might show the first scan, we need to manually control our wheelchair around the area that we are interested in first.

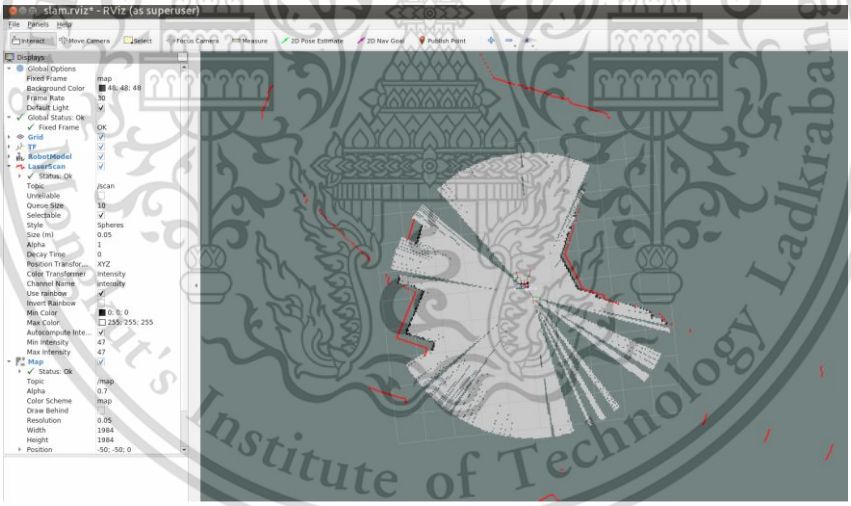


Figure 3.13 Show building map from RPLidar detection environment

3.6.7 Navigation setup

In navigation setup, we need to check that all are working or not. First, RPLidar is able to run correctly, whether we expect it or not, and be aware of objects that are coming close to the wheelchair to avoid them. Second, RPLidar is able to detect objects under the wheelchair to help RPLidar avoid them. The last thing is the speed of our wheelchair,

whether it is able to carry patients to their destination or not. From the Figure 3.14 are show navigation part while we running navigate.

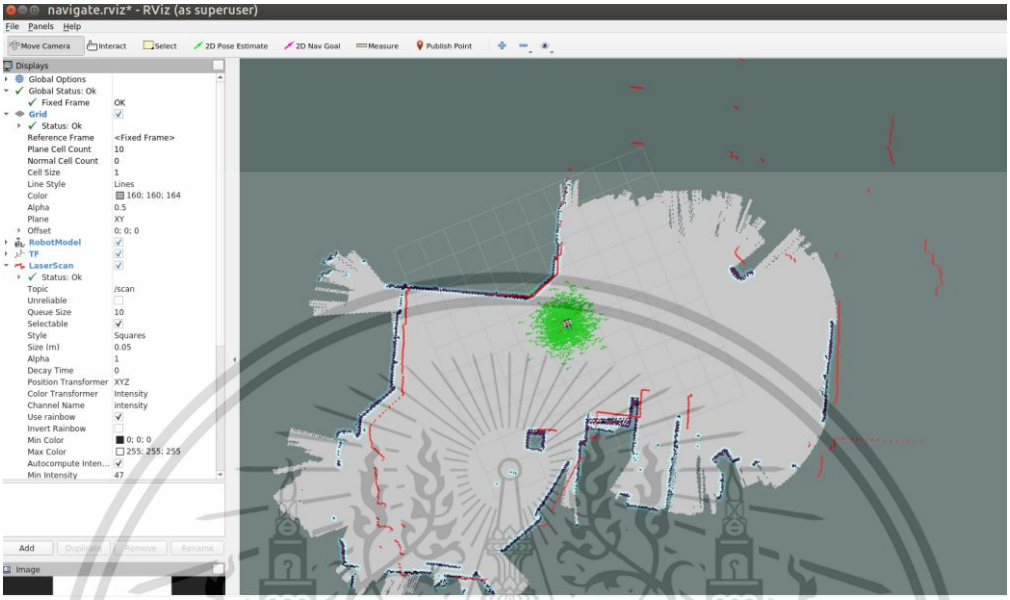


Figure 3.14 Show running navigation

CHAPTER 4
EXPERIMENTAL RESULTS

4.1 Introduction

This project aims to build a smart wheelchair for patients who are unable to control the normal automatic wheelchair. We are using wireless control to control our wheelchair by using a Raspberry Pi through our laptop to control the navigation part of our wheelchair for our destination.

4.2 Motor test

Before we go to test navigation and controlled motors by Raspberry Pi, we need to first test the speed of the motor so that it can go in each direction by uploading code testing to an Arduino mega board for testing the motor and the speed of the wheelchair while having the patient sit on it, which are shown in Table 1,2.

Table 1 Motor testing to measure velocity of wheelchair in 5 meters

RPM	Distance (m)	Time (s)	Velocity (m/s)
20	5	78	0.06410256
150	5	6	0.83333333
255	5	3	1.66666667

Table 2 Motor testing to measure velocity of wheelchair in 10 meters

RPM	Distance (m)	Time (s)	Velocity (m/s)
20	10	153	0.06535948
150	10	13	0.76923077
255	10	6	1.66666667

We also tested the electric current that was used while we tested this motor, as shown in Table 3. We can only test in the range of 20–60 RPM because wheelchairs move really fast, and if we increase them more, it can be dangerous.

Table 3 electric current testing when wheelchair moving forward

RPM	Electric current while loading 0 kg (A)	Electric current while loading 50 kg (A)	Electric current while loading 70 kg (A)	Electric current while loading 100 kg (A)
20	0.5	0.61	0.63	0.66
30	0.87	1.05	1.05	1.05
40	1.28	1.5	1.5	1.5
50	1.69	2.1	2.2	2.4
60	2.1	2.7	2.7	2.9

4.3 Result of control system

From Figure 4.1, we can see the box control of our wheelchair, whose main part is a Raspberry Pi. In our control box is the power supply battery (24 v, 12 Ah) that gives power to the Arduino Mega Board, motor drive, and Raspberry Pi. The Raspberry Pi connects to the RPLidar on the top of our wheelchair and the ultrasonic sensor in front of the wheelchair.

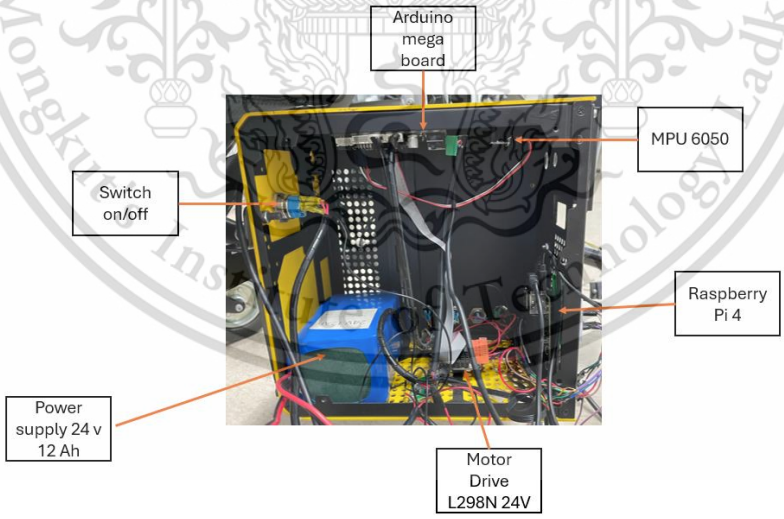


Figure 4.1 Control box of wheelchair

Figure 4.2 shows the encoder while the motor is moving forward and backward to make sure the encoders are working very well.

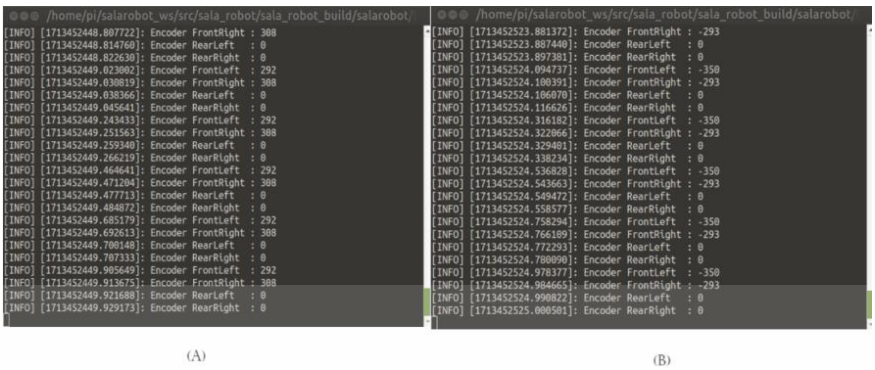


Figure 4.2 Show amount of encoder (A) show while moving forward (B) show while moving backward

4.4 Calibration test

4.4.1 Angular test

In the angular test, we tested a wheelchair for rotation with the patient sitting on it and letting it rotate in 45, 90, 180, and 360 degrees in 10 attempts, as shown in Table 4.

Table 4 Calibration testing of wheelchair in linear distance

Attempt	1 m	2 m	5 m	10 m
1	0.99 m	1.99 m	4.35 m	8.7 m
2	0.99 m	1.95m	4.43m	9 m
3	0.97 m	1.98 m	4.67 m	8.8 m
4	1 m	2 m	4.48 m	9 m
5	1.02 m	2 m	4.47 m	8.8 m
6	1 m	1.97 m	4.59 m	9.1 m
7	1.03 m	1.98 m	4.67m	9.1 m
8	1 m	1.97 m	4.43 m	8.9 m
9	0.98 m	2 m	4.42m	9 m
10	1.05 m	1.99 m	4.39 m	9 m
SD	0.022825	0.015524	0.108167	0.114057
Average	1.003 m	1.983 m	4.49 m	8.941 m

The results from Table 4 of angular testing look great and have fewer errors.

4.4.2 Linear test

In the linear test, we tested the wheelchair for forward motion with the patient sitting on it and allowing it to travel 1, 2, 3,5, and 10 meters, with the results shown in Table 5.

Table 5 Calibration testing of wheelchair in angular

Attempt	45°	90°	180°	360°
1	46° ± 5°	90°± 5°	180°± 5°	350°± 5°
2	45°± 5°	89°± 5°	180°± 5°	360°± 5°
3	46°± 5°	90°± 5°	180°± 5°	355°± 5°
4	45°± 5°	88°± 5°	178°± 5°	350°± 5°
5	45°± 5°	90°± 5°	180°± 5°	350°± 5°
6	45°± 5°	91°± 5°°	177°± 5°	355°± 5°
7	46°± 5°	90°± 5°	180°± 5°	359°± 5°
8	45°± 5°	89°± 5°	180°± 5°	359°± 5°
9	46°± 5°	90°± 5°°	182°± 5	355°± 5°
10	46°± 5°	88°± 5°	180°± 5°	358°± 5°
SD	0.5	0.921	1.268	3.753
Average	45.5°	89.5°	179.7°	355.1°

The result from the linear test has high accuracy in short distance, while for long distances, accuracy will decrease.

4.5 Parameter setup for navigation

4.5.1 costmap common params

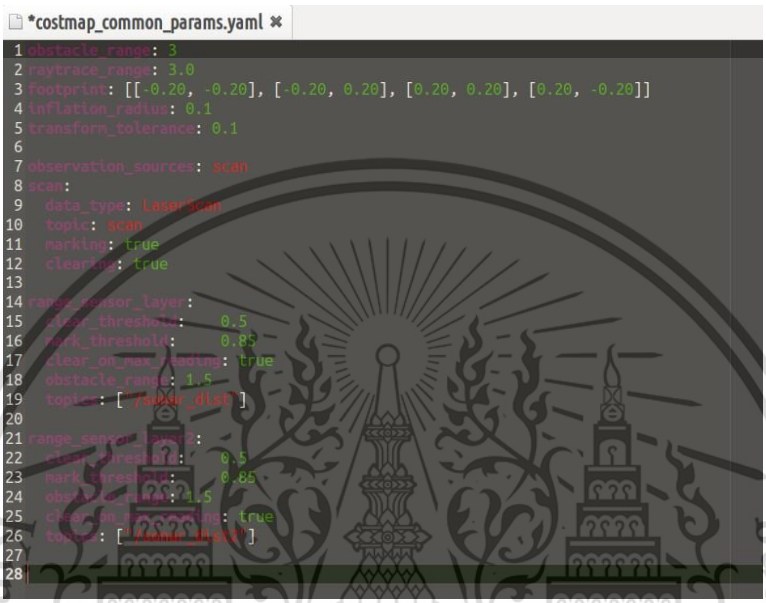
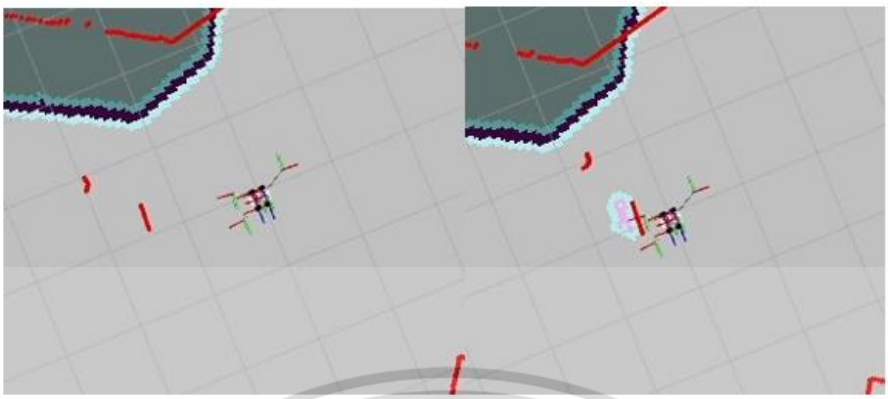


Figure 4.3 Show costmap common parameters

The Costmap Common Parameters specifies the fundamental parameters that the ROS Navigation Stack uses to generate and manage maps. These parameters significantly impact the efficacy of the robotic navigation system.

From the figure 4.3 show the parameter that we using:

- **Obstacl range:** Configures the maximum obstacle detection radius.
- **Raytrace range:** Configure the maximum radius that will be utilized for obstacle detection via ray tracing.
- **Footprint:** Defines the shape of the boundary of the robot. It is a square in this case, with each of its four corners containing the length and width of the boundary.
- **inflation_radius:** Configure the radius of the obstacle's boundary extension. To avoid facing obstacles.
- **transform_tolerance:** The period of time required for the transformation data to become updated.



(a) (b)

Figure 4.4 Show the difference in the obstacle range configuration in 1 m (a). RPLidar can't detect objects that are farther than 1 m (b). RPLidar can detect objects that are closer to 1 m.

In figure 4.4, it demonstrates the difference when we set the parameter. In this figure 4.4, we've set the obstacle range in the common costmap parameters to 1, meaning our RPLidar can only detect objects within a 1-meter radius. As shown in figure 4.4 (a), if the object is farther than 1 meter, it can't detect. Figure 4.4 (b) shows that the RPLidar can detect in the range of 1 m. If we set the obstacle range to a higher value, it will affect navigation calculations performed by the planner.

4.5.2 local_costmap_common_params

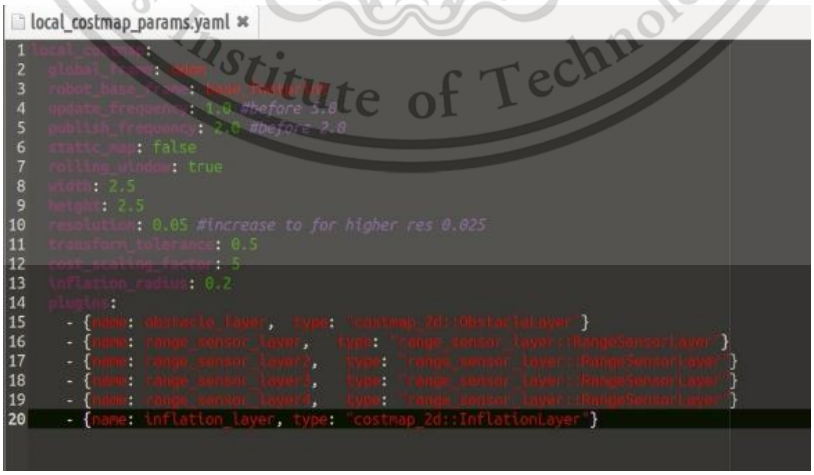
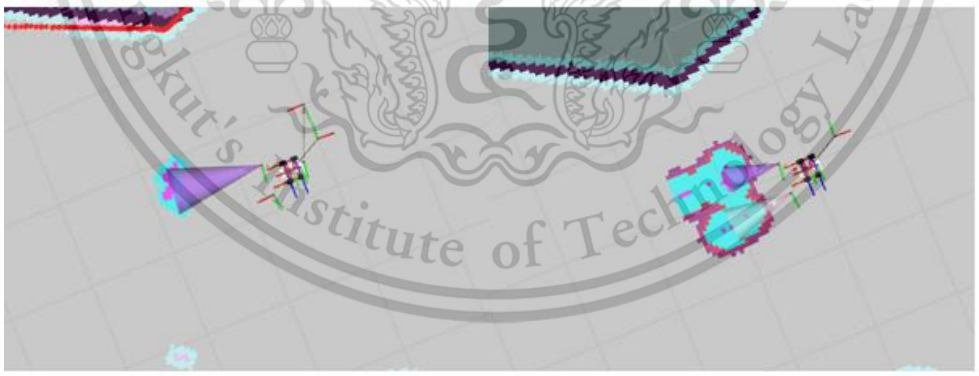


Figure 4.5 Show the local costmap parameter

In ROS navigation, the local_costmap plays a crucial role in enabling safe and efficient robot movement. It's a grid-based representation of the robot's immediate surroundings, incorporating information about obstacles and free space.

From the figure 4.5 show the parameter that we are using:

- **global_frame, robot_base_frame:** Base frame and robot base frame specifications are to be provided. for the transformation system of the robot to correspond with the map.
- **update_frequency, publish_frequency:** Define the publication and update frequency for the local cost map.
- **static_map, rolling_window:** Indicates whether a dynamic or statistical map should be applied for creating the local cost map.
- **Width , High, resolution:** Configure the resolution and size of the local cost map.
- **cost_scaling_factor:** Costmap truth scaling factors.
- **inflation_radius:** Configure the radius of the obstacle's boundary extension. To avoid facing obstacles.
- **Plugins:** This plugin is likely responsible for processing sensor data and marking obstacles in the costmap.



(a) (b)

Figure 4.6 Show the difference in the inflation radius configuration: (a) inflation radius: 0.1; (b) inflation radius: 0.3

In Figure 4.6, we demonstrate the impact of setting different values for the local costmap parameter, specifically by selecting the inflation radius. As shown in Figure 4.6 (a), we set the inflation radius to 0.1. Figure 4.6 (b) sets the inflation radius to 0.3. You can notice

the differences by comparing (a) and (b). Setting the inflation radius to a higher value assists in avoiding obstacles. However, if set higher than necessary, it can adversely affect navigation by making it unable to avoid obstacles during the planning phase.

4.5.3 Global costmap common params

Figure 4.7 Show the globol costmap parameter

A global_costmap plays a crucial role in planning safe and efficient robot trajectories. It's a grid-based representation of the entire environment the robot is navigating in, constructed using sensor data and potentially a pre-loaded map. Similar to the local_costmap, it helps the navigation stack understand the environment and plan paths. The global map is set up almost exactly like the local map, but it focuses on the entre map.

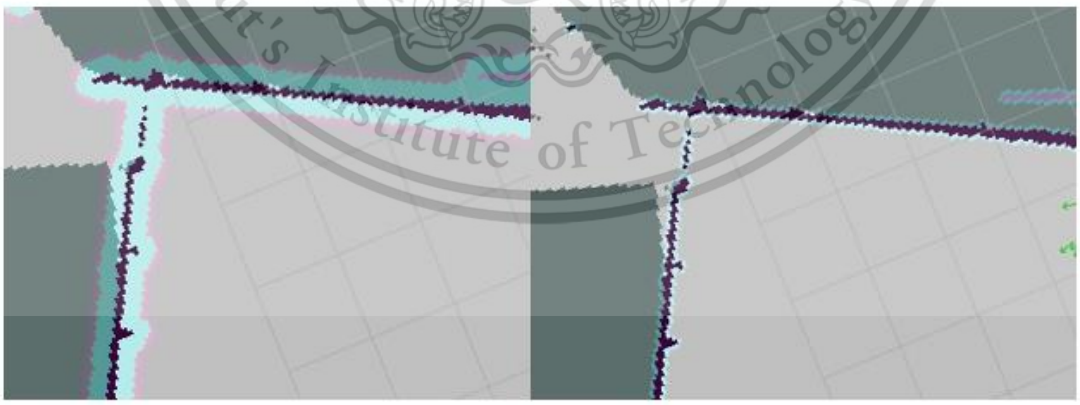


Figure 4.8 Show the difference in the inflation radius configuration: (a) inflation radius: 0.3; (b) inflation radius: 0.05

In Figure 4.8, the difference in inflation radius configuration is shown. In (a), we set the inflation radius to 0.3, while in (b), it's set to 0.05. A notable difference between (a) and (b) can be observed. Setting the inflation radius to a higher value in the global costmap enables the robot to perceive the map boundaries. However, this adjustment also impacts the navigation plan during operation, as the accessible area for the robot diminishes.

4.5.4 base_local_plannce_defult_params

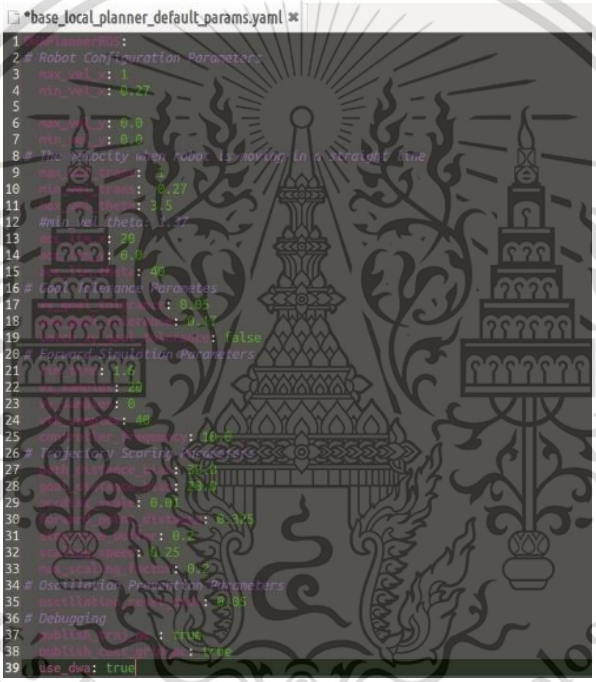


Figure 4.9 Show setup of base local planner default

A base_local_planner_default_params are a set of configuration parameters that define the default behavior of the base local planner within the ROS Navigation Stack. This planner is responsible for generating short-term, obstacle-avoiding motion commands for the robot to follow in order to reach its goal. By adjusting these parameters, you can fine-tune the planner's performance and movement characteristics to suit your specific robot, environment, and task requirements.

From the figure 4.9 show the parameter that we are using:

- **max_vel_x, min_vel_x:** These define the robot's maximum and minimum forward velocities (in meters per second) during navigation.

- **max_vel_trans, min_vel_trans:** The maximum and minimum speeds in the linear direction of the robot.
- **max_vel_theta:** This defines the robot's maximum rotational velocity.
- **acc_lim_x, acc_lim_y, acc_lim_theta:** These define the robot's linear and angular acceleration limits.
- **xy_goal_tolerance:** The acceptable distance (in meters) from the goal pose before the robot considers it "reached" in the X and Y directions.
- **yaw_goal_tolerance:** The acceptable angular difference (in radians) from the goal pose before the robot considers it "reached" in terms of orientation.
- **latch_xy_goal_tolerance:** Whether the robot keeps sending the goal command even after stopping within the tolerance.
- **sim_time:** The time it takes of the simulation to progress to the future location.
- **vx_samples, vy_samples, vth_samples:** Amount of samples used for movement calculations.
- **path_distance_bias, goal_distance_bias, occdist_scale:** Factors included in route estimation
- **forward_point_distance:** The calculation of the direction of movement is based on the distance from the initial point.
- **stop_time_buffer, scaling_speed, max_scaling_factor:** Additional information pertaining to mobile controllers
- **oscillation_reset_dist:** The minimum distance the robot needs to travel in a straight line to reset oscillation detection.
- **publish_traj_pc, publish_cost_grid_pc:** Indicates whether publication of the route points is required.
- **use_dwa:** Confirms the use of the Dynamic Window Approach (DWA) algorithm for trajectory generation.

4.5.5 move_base

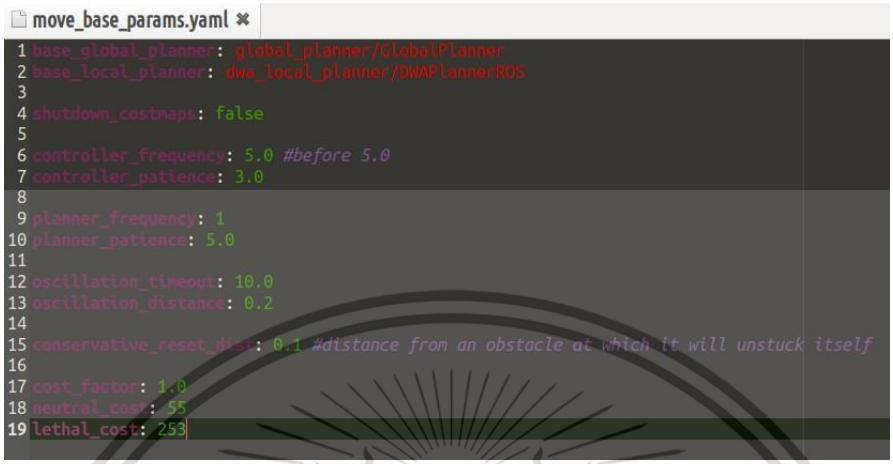


Figure 4.10 Show the setup of move base params

In the ROS Navigation Stack, move_base is a crucial package that acts as the central component for robot navigation. It's responsible for planning and controlling the robot's movement to reach a designated goal location within the environment. move_base is a ROS package that includes base_local_planner and also includes additional components such as a controller and a route planner.

From the figure 4.10 show the parameter that we are using:

- **base_global_planner:** Define the strategic routes that are planned in global planner.
- **base_local_planner:** The strategic plan applied for local route planning should be specified.
- **shutdown_costmaps:** This keeps the costmaps active even when move_base is not actively moving.
- **controller_frequency, controller_patience:** Configure the operating frequency and delay time for the control operation of commands.
- **planner_frequency, planner_patience**: Determine the frequency of route planning and the time required for plans to be completed.
- **oscillation_timeout, oscillation_distance:** Define the time and distance at which duplicate moves are detected and reset if they have not occurred within the allotted time and distance.

- **conservative_reset_dist:** The distance at which the position must be adjusted in order to avoid the obstacle.
- **cost_factor, neutral_cost, lethal_cost:** These are the values utilized for calculating the risks associated with movement in each direction and variables that pertain to the truth values in path planning.

4.5.6 Path planning

Path planning in the setting of ROS is the systematic procedure by which the most efficient route for a robot to traverse from its present location to a designated target location is established, with environmental constraints and obstacle avoidance taken into consideration. Path planning aims to provide the robot with a trajectory that is both secure and efficient to follow. Path planning is a crucial component that works alongside other modules to enable robots to move autonomously.

Our project utilizes the **Dynamic Window Approach (DWA)** in combination with move_base instead of the StaticPlanner. This decision is based on the specific advantages that DWA offers, particularly its dynamic adaptation, ability to generate smooth and efficient paths, and suitability for navigating in dynamic environments. These features align closely with the requirements of our wheelchair, especially the need to navigate around obstacles during movement and the preference for smoother trajectories. By leveraging DWA, our wheelchair navigation system can swiftly adapt to changes in the environment and generate trajectories that optimize both safety and comfort for the user.

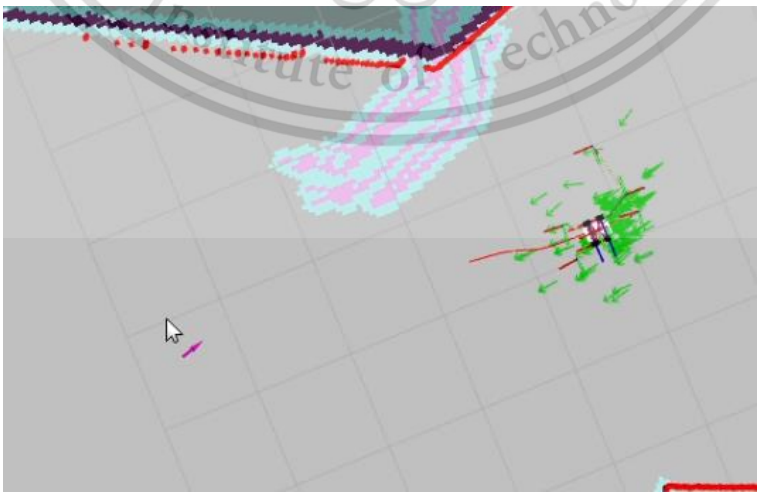


Figure 4.11 Show the path planning of our wheelchair

The path planning of our wheelchair, which incorporates the Dynamic Window Approach (DWA) and a move base for navigation, is shown in Figure 4.11. The mouse cursor shows the destination point in figure 4.11, while the red line in front of our wheelchair model shows the planned path for our robot to reach its destination.

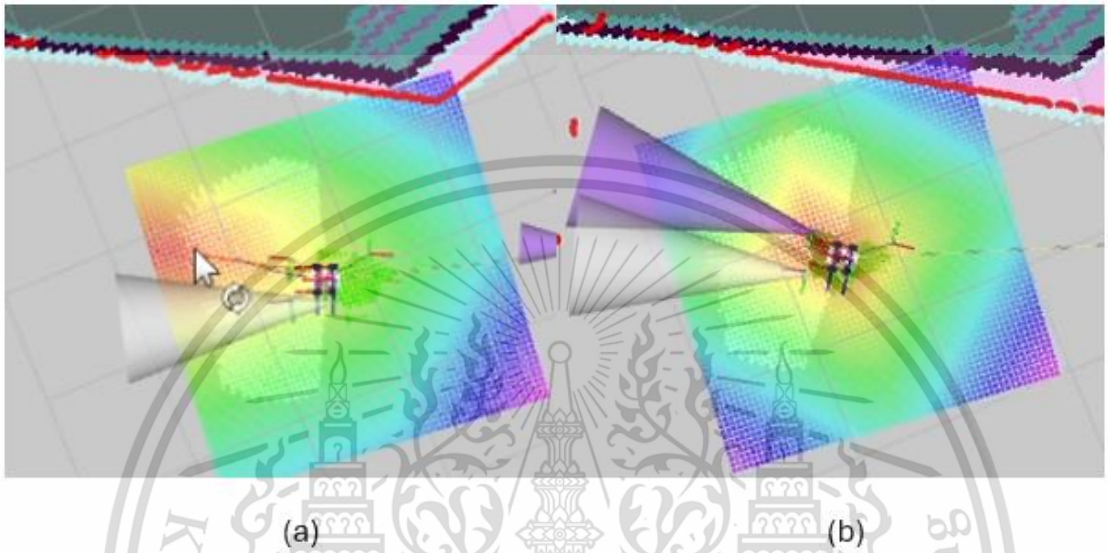


Figure 4.12 Shows the square area of local map working with path planning (a) when going to destination (b) when reach the destination

In Figure 4.12, the square area around the map depicts local area scanning and path planning operations. The range of path planning influences the dimensions of this square area. In Figure 4.12 (a), you can observe the red path planning line and the corresponding red area within the local map, indicating the area where the robot should navigate. In Figure 4.12 (b), it illustrates the wheelchair model reaching the destination, with the red area and the path planning destination displayed at the position of the wheelchair model.

4.5.7 Adaptive Monte Carlo Localization

It is a method for defining the travel path or navigational requirements of tools or robots that are in motion. It utilizes data obtained from environmental sensors or devices in order to optimize the precision of the robot or tool's current location. It combines the most recently sensor data and previous processors in order to incrementally improve the current position.

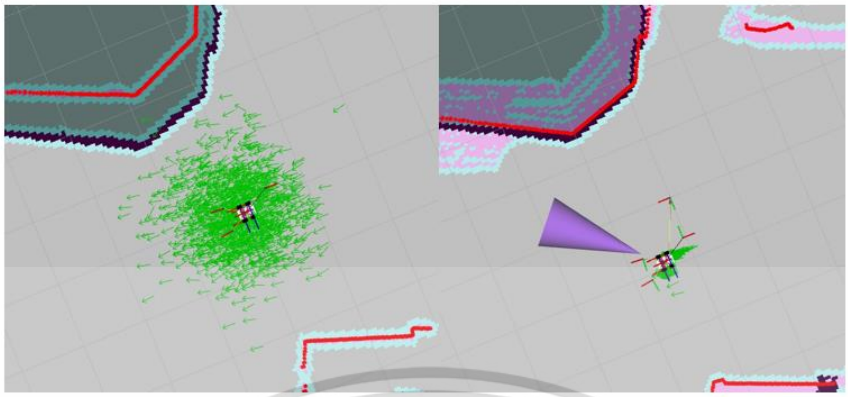


Figure 4.13 Show AMCL Localization, our wheelchair model, on the map (a) it shows the initial run of navigation (b) it show the scenario after moving the wheelchair around to determine its correct position

When we start navigation in figure 4.13 (a), the green arrow around the wheelchair model indicates the robot's possible positions. Each arrow represents the probability that the robot will be at that location. The green arrows' density represents the probability. An area with dense green arrows indicates that there is a high chance that a robot will be in that area. In figure 4.13 (b), the green arrow densely surrounds our wheelchair model, indicating its correct position on the map. You may notice a difference between the RPlidar sensor's detection of the map's boundary in figure 4.13 (a) and figure 4.13 (b). This difference is due to the fact that once AMCL determines the correct position, the boundary should occupy the same position in figure 4.13 (b).

4.5.8 Node subscribe to collect the velocity during navigation

For collecting the velocity when navigation in ros navigation, it has a node called cmd_vel that shows the velocity of our robot during wheelchair movement. We need to subscribe to that node and save it to txt files to use it to plot the velocity graph during navigation.

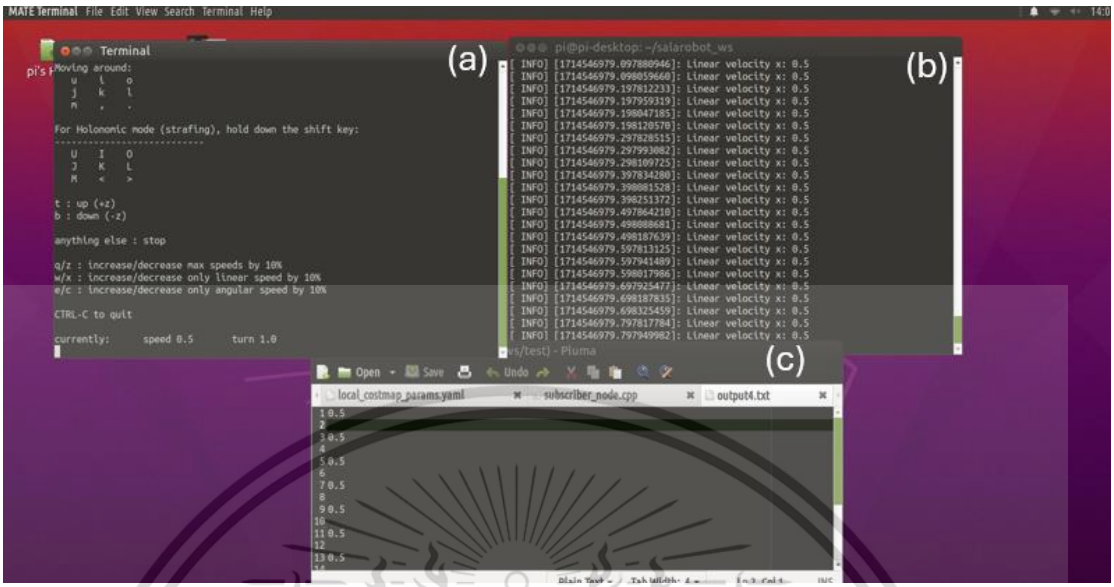


Figure 4.14 Show subscribe the node cmd_vel and collect value to output file (a) show terminal of teleop (b) show node subscribe (c) show files output

In figure. 4.14, we tested by running the teleop at a speed of 0.5 m/s, as shown in figure 4.14 (a), and then running the subscribe node, as shown in figure 4.14 (b), which should receive the amount of velocity from the teleop and save the value every second in the output file, as shown in figure 4.14 (c), for use in plotting the velocity during navigation.

4.6 Navigation test

This is the final part of this wheelchair after we have done all the calibration and testing on all the senses of this wheelchair. In this navigation test, we test two things: first, normal navigation, which doesn't have obstacles on the map. The second will test navigation while the object is on the map.

4.6.1 Navigation test without obstacle

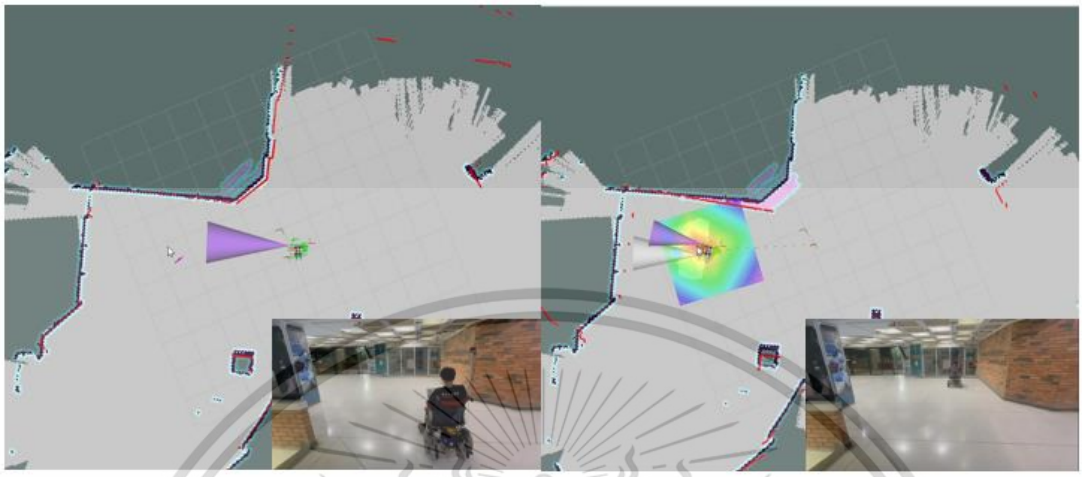


Figure 4.15 Show navigation testing to the first destination (a) is the post position of our wheelchair (b) position of our wheelchair at the first point

Figure 4.15 (a) displays our wheelchair's previous position on the map. After we do adaptive Monte Carlo localization (ACML) to accurately determine the position of our wheelchair on the map, we select the first destination point for our wheelchair using manual input, indicated by the position of my mouse cursor. figure 4.15 (b) shows our wheelchair reaching its destination, both on the map and in real time.

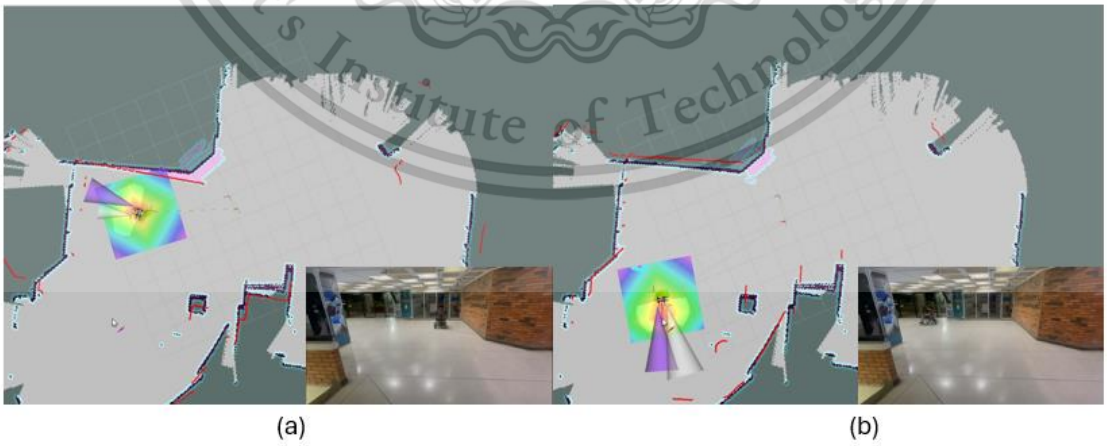


Figure 4.16 Show navigation testing to the Second destination (a) position of wheelchair at the first point (b) position of wheelchair at the second point

As shown in figure 4.16 (a), our wheelchair is displayed at the first position point. We then select the second point, which is represented by the mouse cursor, so that the wheelchair can navigate to that point after it has reached the first point. Our wheelchair can reach at the second point of navigation is depicted in figure 4.16 (b), which depicts the mapping as well as the actual situation.

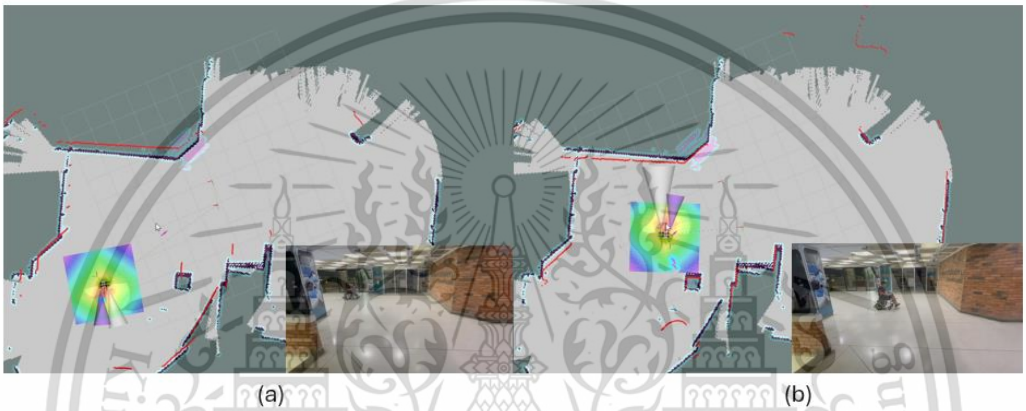


Figure 4.17 Show navigation testing to the Third destination (a) position of wheelchair at the second point (b) position of wheelchair at the third point

As shown in figure 4.17 (a), our wheelchair is able to reach the second destination point on the map, and then we select the third destination point on the map for testing navigation. The destination of the third point is shown on my mouse cursor. In figure 4.17 (b), the position of our wheelchair can reach the third destination on the map and in real time.

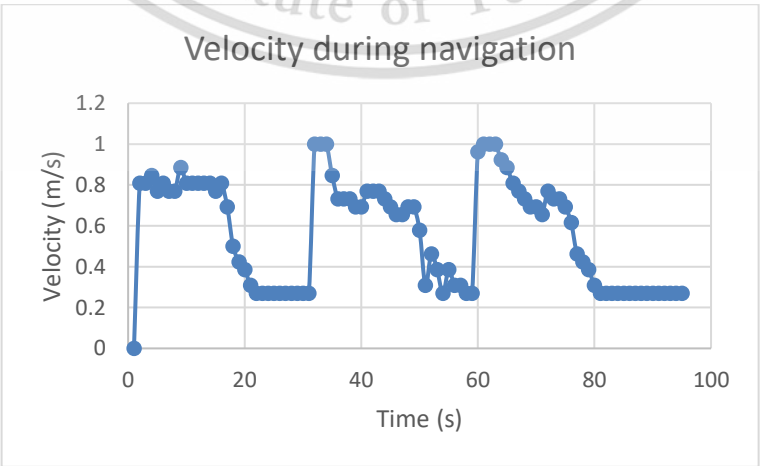


Figure 4.18 Show velocity during navigation

Figure 4.18 shows the velocity during navigation. If we observe it after we choose a destination in the map, the velocity is really high, and when the destination is close to the wheelchair, our wheelchair will decrease the velocity and go to a stop at the destination.

4.6.2 Navigation test with obstacle

Figure 4.19 Show choose destination in map

This research test the navigation wheelchair to the destination and try to avoid the object. From figure 4.19, we put the small box on the area and let the wheelchair detect it as an object and find a way to avoid it.

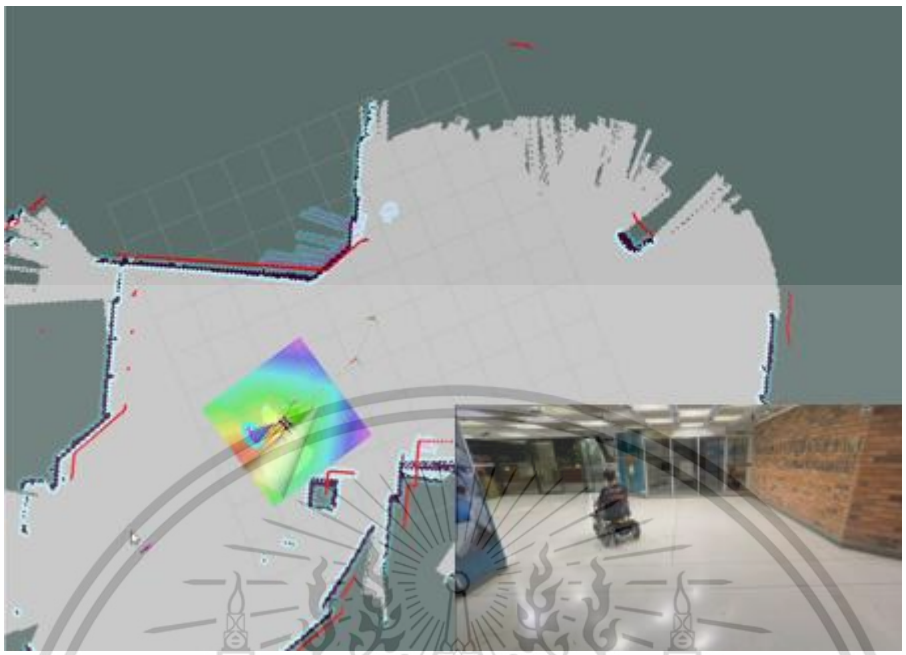


Figure 4.20 Show ultrasonic detection object

Now, from figure 4.20, our ultrasonic right side of the ultrasonic sensor detects the box as an object, as shown in the map. The planning to avoid obstacles should send the command to find another way to avoid this object in front of our wheelchair to go to our destination at the back side of the object.

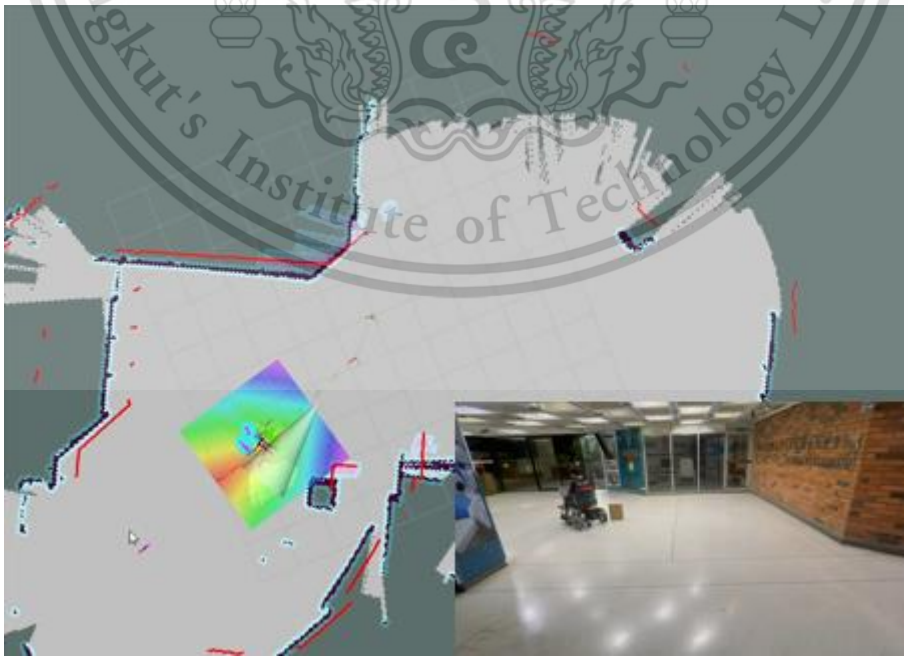


Figure 4.21 Our wheelchair try to avoid the object

In figure 4.21, when the wheelchair detects the object, it blocks the way to the destination, and it is detected by an ultrasonic sensor. Our wheelchair will go to make a plan first, which way we should avoid it. As you can see in figure 4.21, the object is on the front right side of the wheelchair more than the left side, so our wheelchair chose to avoid the left side of it.

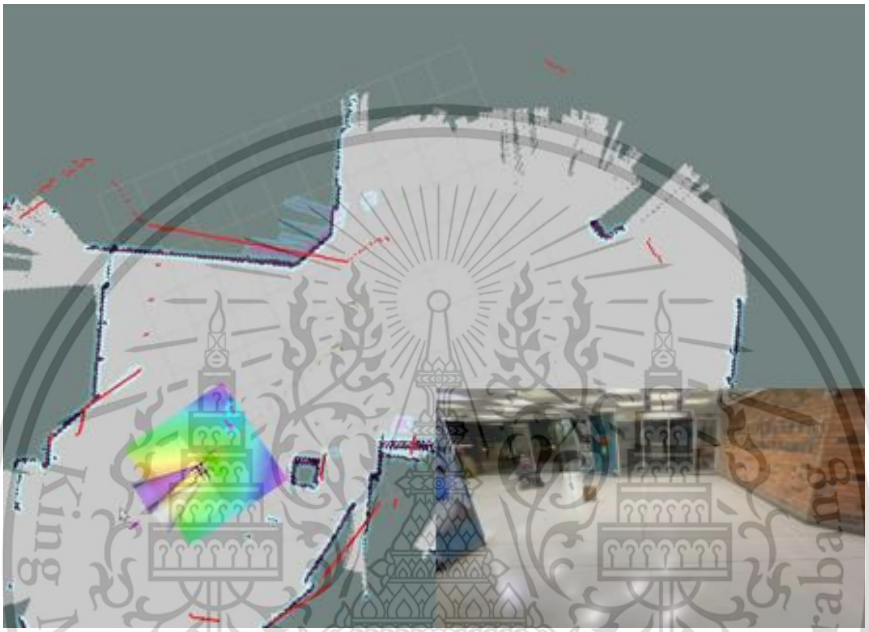


Figure 4.22 Show our wheelchair go to destination

In figure 4.22, we show our wheelchair reaching its destination after avoiding the object from the previous. That means our wheelchair can avoid the object that is detected by the ultrasonic that we attach in front of it and can reach its destination on the map and in real time.

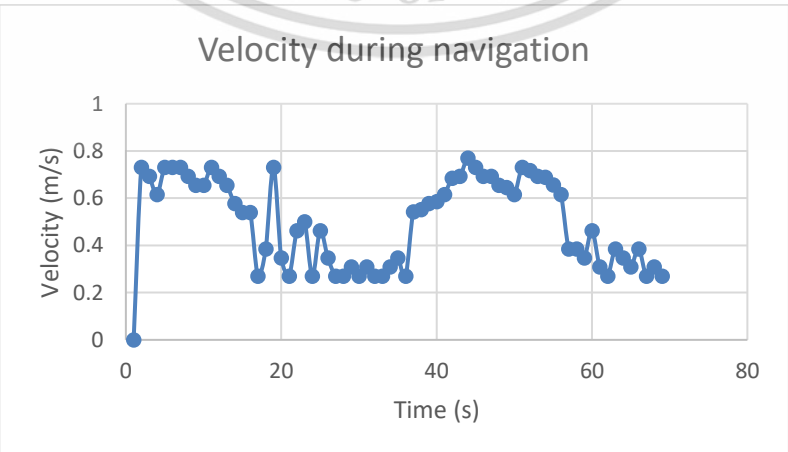


Figure 4.23 Show velocity during navigation with obstacle

In figure 4.23, the graph of velocity during navigation to avoid obstacles at 15–35 seconds shows velocity decreasing while detecting objects, and it might take time to create a new plan for other directions and increase velocity while having another plan to go to the destination.

CHAPTER 5

CONCLUSION

5.1 Introduction

In this chapter, we will discuss and summarize this completed research in each section of Chapter 4, which consists of the design, control, and navigation of a wheelchair. This will provide a better understanding of the development of smart wheelchairs.

5.2 Discussion

Today, there are many people who require wheelchairs to help improve their lives. But there are still some people who have the disease, such as Amyotrophic Lateral Sclerosis (ALS), and a normal wheelchair or electric wheelchair is not suitable for them. So that is the key reason why we need to design the wheelchair for those people. Our wheelchair must be easy to use to bring the patient to their destination by themselves and avoid obstacles during navigation as well.

As we know, our objective is (1) to create a smart wheelchair that improves the quality of life for those with mobility impairments through wireless technology. (2) To create a smart wheelchair that can avoid obstacles during navigation. (3) To create an easily controllable automatic wheelchair for patients. These are our objectives, and we can use the Robot Operation System (ROS) to control our wheelchair. There are many projects that use ROS to control indoor planning robots using lidar, [48] but we improved to using ROS navigation in our wheelchair to create the map by using lidar as a robot and navigating to the destination.

Our wheelchair is working by building the map from an unknown environment or unknown area by using lidar, and while we can collect maps that we are interested in, we can click on the destination in the map that we created, and the wheelchair will go to the destination by themselves. If there is an object while we navigate, our lidar and ultrasonic

can detect it and try to control our wheelchair to avoid it. In the future, we believe that our world will continue to develop, such as today, when we have cars that can drive to bring passengers to their destinations by themselves. We realize and believe that in the future, there must be wheelchairs that can automatically transport patients from hospitals to their desired places. That wheelchair should be able to be controlled by a nurse as to which place the wheelchair should go and can go to different places according to floors. However, this is still difficult to achieve because the space within the hospital and the number of people using the nursing rows every day are high. Therefore, having a wheelchair that can move to different places by itself is still difficult due to safety and the area you want to use. However, we believe that in the future, there will definitely be one. and in our studies, our wheelchair was able to collect maps in various areas. And being able to transport people to different places with just a click should have a greater or lesser effect on the development of wheelchairs that will occur in the future. Including if we can make the wheelchair automatic. A car park within the patient's home that is not as expensive as what we are studying has the opportunity to improve the lives of current patients even further as well.

Furthermore, our wheelchair needs to be connected to the internet if we want to be controlled by a laptop or PC. Our wheelchair is still challenging with the mirror and the not-smooth surface that was able to damage our map during the building

5.3 Conclusion

This project achieved our objective: (1) to create a smart wheelchair that improves the quality of life for those with mobility impairments through wireless technology. (2) To create a smart wheelchair that can avoid obstacles during navigation. (3) To create an easily controllable automatic wheelchair for patients. We can build wheelchairs for patients who are unable to control manual wheelchairs and automatic wheelchairs. Patients only click the destination on the map, and our wheelchairs are able to go to the destination and avoid objects that are also shown in Chapter 4.

The advantage of our wheelchair is being able to navigate to the destination by themselves and avoid objects, but in our project, the disadvantage is some kind of problem while building the map. If we have a mirror as a wall, our RPLidar will go through and have an effect on the map that we are building, and we will also avoid objects during navigation. If

our Raspberry Pi is very hot, it will also be an also be an effect on avoiding objects during navigation.

5.4 Future Study

This project is in the field of biomedical engineering, and I still haven't much to say about the automatic navigation wheelchair that uses a map for control. And hopefully this project will have other things to develop in the future.

In the future, to develop this, we might change other sensors to sensors that have much more accuracy than ultrasonic or add more sensors, but if we combine all sensors to work together, that would have much more effect on creating a plan to control wheelchairs, which might affect precision and velocity in wheelchairs.

REFERENCES

[1] N. A. PRESS, *The Promise of Assistive Technology to Enhance Activity and Work Participation* 2017.

[2] K. Liu *et al.*, "A novel brain-controlled wheelchair combined with computer vision and augmented reality," *Biomedical Engineering Online,* vol. 21, no. 1, pp. 1-20, 2022.

[3] M.-U. A. Sajid, M. F. Mahmud, I. Rahaman, S. Shahriar, and M. N. Rahman, "Design of an intelligent wheelchair for handicap people conducting by body movement," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020: IEEE, pp. 1-5.

[4] N. Wanluk, S. Visitsattapongse, A. Juhong, and C. Pintavirooj, "Smart wheelchair based on eye tracking," in *2016 9th Biomedical Engineering International Conference (BMEiCON)*, 2016: IEEE, pp. 1-4.

[5] W. H. organization. "Health products policy and standards." https://www.who.int/teams/health-product-policy-and-standards/assistive-and-medical-technology/assistive-technology/wheelchair-services#:~:text=Wheelchairs%20provide%20mobility%2C%20postural%20support,life%20on%20their%20own%20terms. (accessed 21 Dec, 2023).

[6] P. Gulla. "Role of the Wheelchair." https://www.physiopedia.com/Role_of_the_Wheelchair#cite_note-:0-1 (accessed 21 dec, 2023).

[7] FreedomMobility. "21 Nov." https://www.freedomhme.com/blog/post/types-of-manual-wheelchairs (accessed 27 Dec 2023).

[8] OrangeBadge. "What are the Different Types of Wheelchairs?" https://orangebadge.co.uk/what-are-the-different-types-of-wheelchairs/#:~:text=Manual%20chairs%20are%20the%20standard,controller%20%E2%80%93%20similar%20to%20a%20joystick. (accessed 27 Dec, 2023).

[9] S. Medical. "Manual Mobility - The Basics." https://www.sunrisemedical.ca/education-in-motion/clinical-corner/december-2013/manual-mobility-the-basics#:~:text=The%20categories%20of%20manual%20wheelchairs,%2C%20rigid%2C%20and%20dynamic%20tilt. (accessed.

[10] a. 2. "Indoor Electric Wheelchairs." https://aspire2.co.uk/collections/indoor-electric-wheelchairs (accessed.

[11] W. B. Dante G. Scarpelli. "disease." https://www.britannica.com/science/disease (accessed 28 Dec, 2023).

[12] K. Claud and J. Sun, "Metabolites and micronutrition in modulating amyotrophic lateral sclerosis," *Neural Regeneration Research,* vol. 19, no. 6, pp. 1183-1184, 2024.

[13] S. Martin, A. Al Khleifat, and A. Al-Chalabi, "What causes amyotrophic lateral sclerosis?," *F1000Research,* vol. 6, 2017.

[14] sakaowrat.tec. "Get to know...spinal cord injury." https://pt.mahidol.ac.th/knowledge/?p=2646#:~:text=%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%9A%E0%B8%B2%E0%B8%94%E0%B9%80%E0%B8%88%E0%B9%87%E0%B8%9A%E0%B8%97%E0%B8%B5%E0%B9%88%E0%B9%84%E0%B8%82%E0%B8%AA%E0%B8%B1%E0%B8%99%E0%B8%AB%E0%B8%A5%E0%B8%B1%E0%B8%87%E0%B8%A1%E0%B8%B5,%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B8%9E%E0%B8%B4%E0%B8%81%E0%B8%B2%E0%B8%A3%20(1%2C2) (accessed 29 Dec, 2023).

[15] L. A. Simpson, J. J. Eng, J. T. Hsieh, Wolfe, and D. L. the Spinal Cord Injury Rehabilitation Evidence Research Team, "The health and life priorities of individuals with spinal cord injury: a systematic review," *Journal of neurotrauma,* vol. 29, no. 8, pp. 1548-1555, 2012.

[16] C. Baecher-Allan, B. J. Kaskow, and H. L. Weiner, "Multiple sclerosis: mechanisms and immunotherapy," *Neuron,* vol. 97, no. 4, pp. 742-768, 2018.

[17] J. Halper, "The psychosocial effect of multiple sclerosis: the impact of relapses," *Journal of the neurological sciences,* vol. 256, pp. S34-S38, 2007.

[18] M. Rohrig. "Mobility and Walking Issues." (accessed 29 Dec, 2023).

[19] H. J. Willison, B. C. Jacobs, and P. A. van Doorn, "Guillain-barre syndrome," *The Lancet,* vol. 388, no. 10045, pp. 717-727, 2016.

[20] Linux.com. "What Is Linux." https://www.linux.com/what-is-linux/ (accessed 2 Jan, 2024).

[21] W. Python, "Python," *Python Releases for Windows,* vol. 24, 2021.

[22] N. G. "What Is Ubuntu? A Quick Beginner's Guide." https://www.hostinger.com/tutorials/what-is-ubuntu#Ubuntu_vs_Linux_Whats_the_Difference (accessed 2 jan, 2024).

[23] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics,* vol. 32, no. 6, pp. 1309-1332, 2016.

[24] ROS.org. "ROS/Introduction." http://wiki.ros.org/ROS/Introduction (accessed 3 Jan, 2024).

[25] ROS.org. "Navigate." http://wiki.ros.org/navigation#Overview (accessed 3 Jan, 2024).

[26] ROS.org. "move_base." http://wiki.ros.org/move_base (accessed 3 Jan, 2024).

[27] ROS.org. "Gmappings." http://wiki.ros.org/gmapping (accessed 3 Jan, 2024).

[28] ROS.org. "teleop_twist_keyboard." http://wiki.ros.org/teleop_twist_keyboard (accessed 3 Jan, 2024).

[29] J. K. Makhubela, T. Zuva, and O. Y. Agunbiade, "A review on vision simultaneous localization and mapping (VSLAM)," in *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, 2018: IEEE, pp. 1-5.

[30] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE robotics & automation magazine,* vol. 13, no. 2, pp. 99-110, 2006.

[31] J. W. Jolles, "Broad‐scale applications of the Raspberry Pi: A review and guide for biologists," *Methods in Ecology and Evolution,* vol. 12, no. 9, pp. 1562-1579, 2021.

[32] M. Donnison. "The Differences Between Raspberry Pi 4 Model B & Raspberry Pi 5." https://kitronik.co.uk/blogs/resources/the-differences-between-raspberry-pi-4-model-b-raspberry-pi-5 (accessed.

[33] C. Bell, *Beginning Sensor Networks with XBee, Raspberry Pi, and Arduino*. Springer, 2020.

[34] B. Trowbridge, F. Calas, Z. Weingarten, J. Husinger, V. Fomitchev, and R. Integlia, "Protomesh, a wireless solution and platform for embedded education and Raspberry Pi workshops," in *2017 IEEE World Engineering Education Conference (EDUNINE)*, 2017: IEEE, pp. 90-94.

[35] L. Shanghai Slamtec.Co. "RPLIDAR A1." https://www.generationrobots.com/media/rplidar-a1m8-360-degree-laser-scanner-development-kit-datasheet-1.pdf (accessed 29 March, 2024).

[36] M. Wu, H. Ma, M. Fu, and C. Yang, "Particle filter based simultaneous localization and mapping using landmarks with RPLidar," in *International Conference on Intelligent Robotics and Applications*, 2015: Springer, pp. 592-603.

[37] E. P. COMPANY. "What is an encoder?" https://www.encoder.com/article-what-is-an-encoder (accessed 4 April, 2024).

[38] usdigital. "How does an (incremental) encoder work?" https://www.usdigital.com/blog/how-does-an-encoder-work?gad_source=1&gclid=CjwKCAjw_LOwBhBFEiwAmSEQAeVyrFWwDsowjNghr7smTgbwlHtrH6o8LYgVmZk-c4qLxel_vbeufxoCJy8QAvD_BwE (accessed 4 April, 2024).

[39] N. Kashiri, J. Malzahn, and N. G. Tsagarakis, "On the sensor design of torque controlled actuators: A comparison study of strain gauge and encoder-based principles," *IEEE Robotics and Automation Letters,* vol. 2, no. 2, pp. 1186-1194, 2017.

[40] S. Mikhel, D. Popov, S. Mamedov, and A. Klimchik, "Advancement of robots with double encoders for industrial and collaborative applications," in *2018 23rd Conference of Open Innovations Association (FRUCT)*, 2018: IEEE, pp. 246-252.

[41] E. Community. "Rotary Encoder Working Principle." https://engineerscommunity.com/t/rotary-encoder-working-principle/6781 (accessed 21 june, 2024).

[42] primusthai. "How is Increment Encoder different from Absolute Encoder?" https://www.primusthai.com/primus/Knowledge/info?ID=157 (accessed 21 june, 2024).

[43] sunfounder. "Accelerometer & Gyroscope Module (MPU6050)." https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/05-component_mpu6050.html (accessed 5 April, 2024).

[44] A. A. Rafiq, W. N. Rohman, and S. D. Riyanto, "Development of a simple and low-cost smartphone gimbal with MPU-6050 sensor," *Journal of Robotics and Control (JRC),* vol. 1, no. 4, pp. 136-140, 2020.

[45] electricity-magnetism. "Motor Drive " https://www.electricity-magnetism.org/motor-drives/ (accessed 9 April, 2024).

[46] Vayuyaan. "Everything You Want to Know About L298N Motor Driver." https://www.linkedin.com/pulse/everything-you-want-know-l298n-motor-driver-vayuyaan#:~:text=The%20L298N%20motor%20driver%20controls,faster%20the%20motor%20will%20rotate. (accessed 9 April 2024).

[47] Robotsiam. "Motor Driver Module L298N." https://www.robotsiam.com/product/4/motor-driver-module-l298n (accessed 9 April, 2024).

[48] S. Gatesichapakorn, J. Takamatsu, and M. Ruchanurucks, "ROS based autonomous mobile robot navigation using 2D LiDAR and RGB-D camera," in *2019 First international symposium on instrumentation, control, artificial intelligence, and robotics (ICA-SYMP)*, 2019: IEEE, pp. 151-154.

BIOGRAPHY

Name	Mr. Nutt Jaturat
Date of Birth	26 August 2001
Address	130/261 M.1 Lam Phak Kut Sub-district, Thanyaburi District, Pathum Thani. 12110
Education	Bachelor of Engineering Biomedical Engineering King Mongkut's Institute of Technology Ladkrabang
Specialized field	1) Medical Instruments 2) Iot application
Publications and Awards	
Present	15th Biomedical Engineering International Conference – BMEiCON2023, held in Tokyo, Japan