SLAM Robot for Drug Delivery in Hospitals

BY

SUPISSARA BOONCHUAY 62011253

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN BIOMEDICAL ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2022

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	SLAM Robot for Drug Delivery in Hospitals
Student Name	Miss Supissara Boonchuay
	Student ID. 62011253
Degree	Bachelor of Engineering in Biomedical Engineering

Project Advisor — Signed: _____
(Assoc. Prof. Dr. Chuchart Pintavirooj)

Committee — Signed: _____
(Assoc. Prof. Dr. Wibool Piyawattanamatha)

Committee — Signed: _____
(Asst. Prof. Dr. Treesukon Treebupachatsakul)

Committee — Signed: _____
(Asst. Prof. Dr. Kasama Srirussamee)

Head of Department — Signed: _____
(Assoc. Prof. Dr. Sarinporn Visitsattapongse)

Project Title	SLAM Robot for Drug Delivery in Hospitals
Student Name	Miss Supissara Boonchuay
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Assoc. Prof. Dr. Chuchart Pintavirooj International Program, Ph.D.
Academic Years	2022

ABSTRACT

Currently, Thailand is facing a shortage of medical staff. Thus, reducing tasks is productive, especially drug delivery to each department in a hospital. Delivery requires a lot of staff to process. However, errors may occur due to human errors. For instance, unpunctual, incorrect, damaged, and lost. From the drug distribution center to each department, drug distribution procedures are routine, pattern-based, low-complexity, and time-consuming. With simultaneous mapping and localization, or SLAM, mobile robots use SLAM to map their environments and locate themselves in real time. SLAM robots can drive themselves and adjust their direction based on data or recently encountered obstacles in an unknown environment without prior knowledge of the location, which typically functions indoors. In order to facilitate the drug delivery process, this project aims to develop an indoor SLAM robot for use in an outdoor environment with unchanged route conditions.

ACKNOWLEDGEMENTS

For the acknowledgements, I would like to give my sincerest appreciation to everyone who has supported and encouraged me during my educational life.

Initially, I wish to express my heartfelt appreciation to my thesis advisor, Associate Professor Dr. Chuchart Pintavirooj, for his invaluable guidance, insights, and patience throughout the research process. Without his/her constant support, this thesis would not have been possible.

I am also grateful to the faculty and staff of the Biomedical Engineering Department and the School of Engineering at King Mongkut's Institute of Technology Ladkrabang for their unwavering support and for providing me with the necessary resources and opportunities to pursue my research interests.

To my friends and family, thank you for your unwavering encouragement, motivation, and love throughout this challenging yet rewarding journey. Your unwavering support has been a source of inspiration for me and has helped me persevere through the toughest of times.

Finally, I would like to express my heartfelt gratitude to the open-source community, in particular, has made a significant impact on the advancement of knowledge in many fields, and I am deeply grateful for their contributions.

Sincerely,

Supissara Boonchuay

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
AMCL	Adaptive Monte Carlo Localization
AMD	Advanced Micro Devices
API	Application Programming Interface
ARM	Advanced RISC Machines
AVR	Alf and Vegard's RISC processor
CLI	PlatformIO Core
CPU	Central processing Unit
GCC	GNU Compiler Collection
GNOME	GNU Network Object Model Environment
GNU	An operating system
GPL	General Public License
Hg	Mercurial
IDE	An integrated development environment
IMU	Inertial Measurement Unit
LAN	Local Area Network
LiDAR	A laser
MPU6050	A gyroscope and accelerometer sensor
ODSL	Open Source Development Labs
OS	Operating System
PC	Personal Computer
PhD	Doctor of Philosophy
PWM	Pulse Width Modulation
ROS	Robot Operating System
RViz	ROS visualization
SBC	Single Board Computer
SDK	Software Development Kit
SLAM	Simultaneous Localization Mapping
SVN	Subversion
VCS	Version Control System

WAN	Wide Area Network

CHAPTER 1
INTRODUCTION

This chapter begins with the background and significance of the research, which introduces the theory of Robot Operating System (ROS), 2D LiDAR Simultaneous Localization Mapping (SLAM), and port forwarding. Subsequently, the research objectives are described, and followed by the hypothesis.

1.1 Background and significance of the Research

Simultaneous Localization and Mapping (SLAM) is a method utilized to gather data from an unfamiliar environment in real-time. Therefore, when the SLAM method is applied to a robot, it can sense any obstacle and localize while moving simultaneously as an autonomous robot [1]. SLAM robots are widely used indoors. However, this research aims to develop a SLAM robot to be used outdoors to expand the accessible area and increase the ability of services.

A helpful framework of software libraries and tools is the Robot Operating System (ROS). ROS provides sophisticated open-source tools for developing autonomous robots, such as hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and implementing state-of-the-art algorithms for robotic applications under the BSD license.

SLAM algorithms require a device to scan or measure the distance between obstacles. This research is A 2D LiDAR-based SLAM system, meaning a lidar laser sensor used for sensing. When the robot can notice and compute the distance of the object, the robot uses this data to generate a map, resulting in localization so that the robot can locate itself on the collected map. For the purpose of self-driving, the robot can be navigation to any ordered destination within the map because of localization.

Generally, SLAM robots usually employ indoors. With the aim of outdoor services, port forwarding is a method to improve the limit of internet conditions. Port forwarding

purpose to allow public computers in WAN to access private LANs as known as remote access.

1.2 Research Objectives

1.2.1. To facilitate delivery in an outdoor environment.

1.2.2. To decrease the probability of loss during transportation to enhance the opportunity cost.

1.3 Research Hypothesis

Since many SLAM robots assist humans in different services, this study aims to improve the accuracy of the SLAM robot. The calibration method is hypothesized to provide more accuracy for the service-SLAM robot. And many SLAM robot is an indoor robots. Port forwarding is hypothesized to use, control, and communicate with robots from anywhere in outdoor environments.

CHAPTER 2
THEORY

2.1 SLAM

Simultaneous Localization and Mapping or SLAM is a process that allows a mobile robot to create a map of its environment while also determining its location. In SLAM, the trajectory of the platform as well as the placement of all landmarks are repeatedly calculated and updated based on incoming data from the new surrounding area, without any prior experience of the location being required. [1]

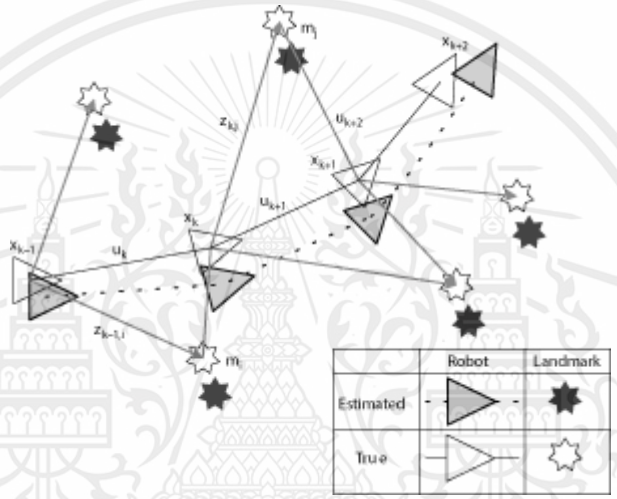

Figure 1 The Illustration of a robot reaching landmark or target using SLAM [1]

Regarding a mobile robot travelling somewhere in an environment, as depicted in Figure 1, the robot's sensor makes relevant evaluations of a variety of unacquainted landmarks. At instant k, the following values are established:

- x_k: the state vector explaining the vehicle's location and orientation.
- u_k: the control vector used to operate the automobile to state x_k at time k-1.
- m_i: a vector expressing the i th landmark's true position, which is supposed to be time invariant.
- z_{ik}: a vehicle-based perspective of the location of the i landmark at time k. When many landmark observations are made at the same time, or when the exact landmark is irrelevant to the conversation, the observation is basically reported as z_k.

Furthermore, the following sets are defined:

- $X_{0:k} = \{x_0, x_1, \cdots, x_k\} = \{X_{0:k-1}, x_k\}$: the history of vehicle locations
- $U_{0:k} = \{u_1, u_2, \cdots, u_k\} = \{U_{0:k-1}, u_k\}$: the history of control inputs
- $M = \{m_1, m_2, \cdots, m_n\}$ the set of all landmarks
- $Z_{0:k} = \{z_1, z_2, \cdots, z_k\} = \{Z_{0:k-1}, z_k\}$: the set of all landmark observations

[1]

2.2 Software

2.2.1 Linux

Linux, the kernel at the heart of a free operating system, was developed and released by Linus Benedict Torvalds in 1991. Torvalds, a former PhD student at the University of Helsinki in Finland who is now a Fellow at the Linux Foundation, created Linux in 1991. (www.linuxfoundation.org). Before joining Open Source Development Labs (ODSL), a consortium founded by numerous high-tech enterprises to support Linux development, he worked as an engineer for Transmeta, Inc., a CPU design and manufacturing business, from which he departed in 2003 to devote his whole attention to the Linux kernel. Thankfully, Torvalds chose to release Linux under a free software license known as the GNU General Public License, which is used by the vast majority of Linux users (GPL).[2]

The GNU General Public License (GPL) was developed by Free Software Foundation founder Richard M. Stallman. In order to guarantee that every software licensed under the GPL would always be freely available in source code form, the GPL was developed by Stallman, the famous author of the emacs editing environment and the GCC compiler system. Linux's ownership, distribution, and copyright are all spelled out under the GNU General Public License (GPL). Although Torvalds owns the Linux trademark, the Internet, hundreds of programmers throughout the world, GNU software, and the GNU General Public License (GPL) will ensure that Linux is always available without cost or licensing fees.[2]

There is a consistent numbering system for Linux kernels, kernel updates, and Ubuntu kernel releases. These values have nothing to do with the version of Ubuntu Linux you're using. Linux kernel version numbers are typically assigned by Linus Torvalds and his team of developers, whereas the Ubuntu distribution's version numbers are decided by the Ubuntu team. Ubuntu is based on the Debian Linux distribution, however there are many other Linux distributions available from a wide variety of manufacturers. [2]

Figure 2 Ubuntu logo [3]

Figure 3 Debian logo [4]

Levels and Layers

The Linux system is divided into three levels. Figure 3 depicts these levels as well as some of the components inside each level. The hardware is located at the bottom. Memory is included in hardware, as are one or more central processing units (CPUs) for computation and reading and writing to memory. Hardware also includes devices such as drives and network interfaces. [5]

The kernel, which is the operating system's core, is the next step up. The kernel is memory-resident software that instructs the CPU on where to look for its next task. The kernel handles the hardware (particularly main memory) and serves as the primary interface between the hardware and any running program. [5]

The system's higher level, known as user space, is made up of Users Processes, which are the running programs that the kernel manages. User process is more detailed than just "process" because it implies that a user is involved in the procedure. In particular, all web servers are implemented as user processes..[5]

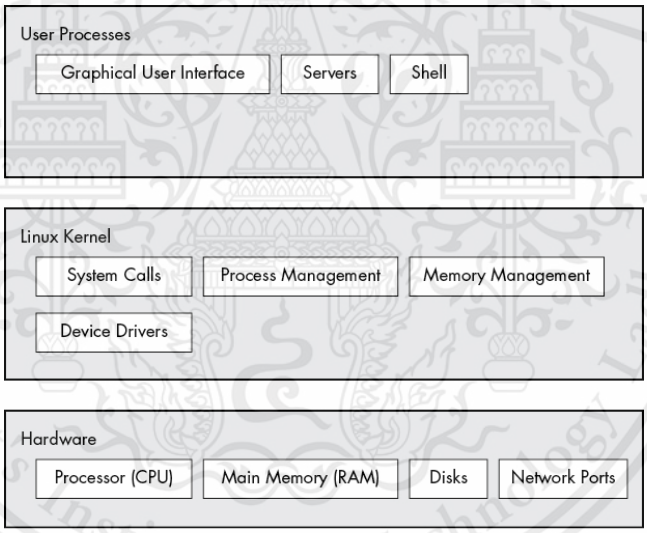

Figure 4 Linux System Architecture [5]

There is a relevant difference between how the kernel and user processes operate: the kernel operates in kernel mode, while user processes operate in user mode. In kernel mode, code has complete access to the processor and main memory. This is a strong but dangerous privilege that allows the kernel to quickly corrupt and crash the system. Kernel space refers to the memory area that only the kernel has access to. [5]

In contrast, user mode restricts access to a (typically restricted) subset of memory and safe CPU activities. The areas of main memory that user processes can access are referred to as user space. If a process fails and crashes, the ramifications are limited and can be cleaned up by the kernel. This means that even if your web browser crashes, the scientific computation that has been operating in the background for days is unlikely to be affected. [5]

Kernel

When a process running in user mode requires more memory, pages are allocated from the kernel's list of free page frames. This list is often built using a page-replacement technique and most likely comprises free pages distributed throughout physical memory. Remember that if a user process seeks a single byte of memory, the process will be granted a full-page frame, resulting in internal fragmentation. [6]

Kernel memory, on the other hand, is frequently assigned from a different free-memory pool than the one used for user-mode activities.. This is due to two fundamental reasons:

1. The kernel needs memory for numerous data structures, and some of these are smaller than a page. As a consequence, the kernel must utilize memory sparingly and strive to minimize fragmentation-related waste. This is crucial because the majority of operating systems do not expose kernel code or data to the paging system. [6]
2. There is no requirement that pages assigned to operating system be physically coincident. Specified hardware components, on the other hand, establish a connection with physical memory even with aid of a virtual memory interface, and thus may require memory to be stored in continuous pages. [6]

2.2.2 Ubuntu

Ubuntu is a Linux-based operating system developed, maintained, refined, and provided by the Ubuntu Community at www.ubuntu.com. Ubuntu is an open source project sponsored by Canonical Ltd. (www.canonical.com) and backed by a global community of software developers. [2]

Figure 5 Canonical Logo [7]

Ubuntu's initial version was published in October 2004. It immediately earned a reputation for its ease of installation and usage, as well as the slightly bizarre code names assigned to each edition. However, Ubuntu is built on Debian, a much older distribution that is well-regarded in the Linux community. Debian is the foundation upon which Ubuntu is built. [2]

Ubuntu is compatible with both desktop computers and servers, with over a thousand available software packages, including the Linux kernel and the GNOME/KDE desktop environment. Accordingly, Ubuntu also involves various programming languages, word processing, spreadsheets, a web browser, a web server, an integrated development environment (IDE), and PC games are known as "conventional desktop apps". It is compatible with Intel x86, AMD64, ARMv7, and ARMv8 architectures (ARM64). [8]

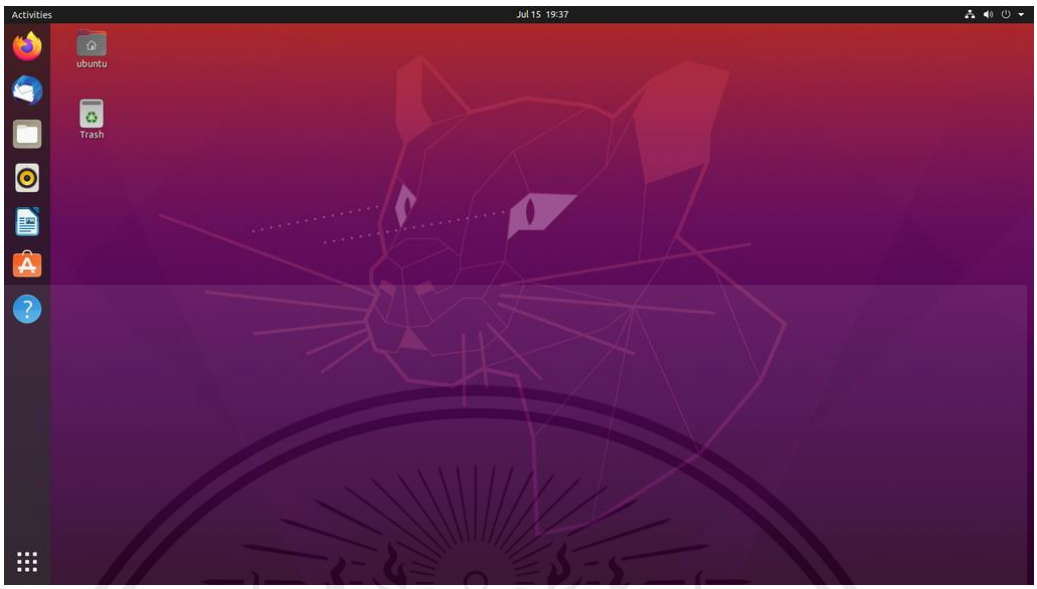

Figure 6 Desktop Ubuntu 20.04 [9]

2.2.3 ROS

The Robot Operating System (ROS) is a versatile framework that includes a variety of tools and libraries for developing robotic software. It has a number of powerful features that can assist developers with tasks as with forwarding messages, distributed computing, reusing code, and designing cutting-edge algorithms for robotic applications. The goal of ROS was to create a standardized method for programming robots, as well as to provide software components that could be easily integrated into custom robotic applications. There are several causes consider when choosing ROS as a programming framework, including the following:

- **Advance capabilities**: ROS includes pre-installed functionality. ROS tools such as SLAM and AMCL can be utilized providing moving robots with autonomous navigation. In out robot software can immediately utilize these capabilities without any complications. For many common robotics tasks across multiple platforms, these programs are all that is required. In addition, these features are highly configurable through a variety of settings.
- **Tools**: Among the mainly potent open-source applications are rqt gui, RViz, and Gazebo. Consequently, these are examples of debugging, visualizing, and simulation tools in the ROS ecosystem.
- **The usage opportunity of advanced sensors**: To expand the boundaries of applications, ROS enables the use of multiple interface packages and device drivers for a variety of robotic sensors and actuators. 3D LIDAR, laser scanners, motor actuators, and other high-end sensors are examples. We can easily integrate with ROS.
- **Various integrated languages**: Using the ROS message-passing middleware, applications can communicate regardless of the underlying hardware. ROS refers to these components as "nodes." These nodes can be created in any language that has ROS client libraries. Nodes can be written in numerous languages, such as Python, Java, and C++.

- **Modularity**: The majority of autonomous robotic systems are susceptible to having one of their major code threads become unstable, which may result in the application for the robot as a whole becoming inoperable. The situation is different in ROS; we write individual nodes for each process, and even if one of those nodes fails, the system can still function normally.
- **Coexistence code handling**: When using ROS, you can write the code so that it operates in both a multithreaded and a single-threaded code for coexistence. When more than two features are added to threads, the performance of the application deteriorates and becomes difficult to debug. In contrast, ROS can communicate with hardware through ROS topics exposed by ROS drivers. The picture message from the ROS camera driver can be subscribed to by any number of ROS nodes, and each node might have distinct capabilities. This can minimize computing complexity while simultaneously improving the whole system's debugging ability. [10]

2.2.3.1 Concepts

2.2.3.1.1 ROS Filesystem Level

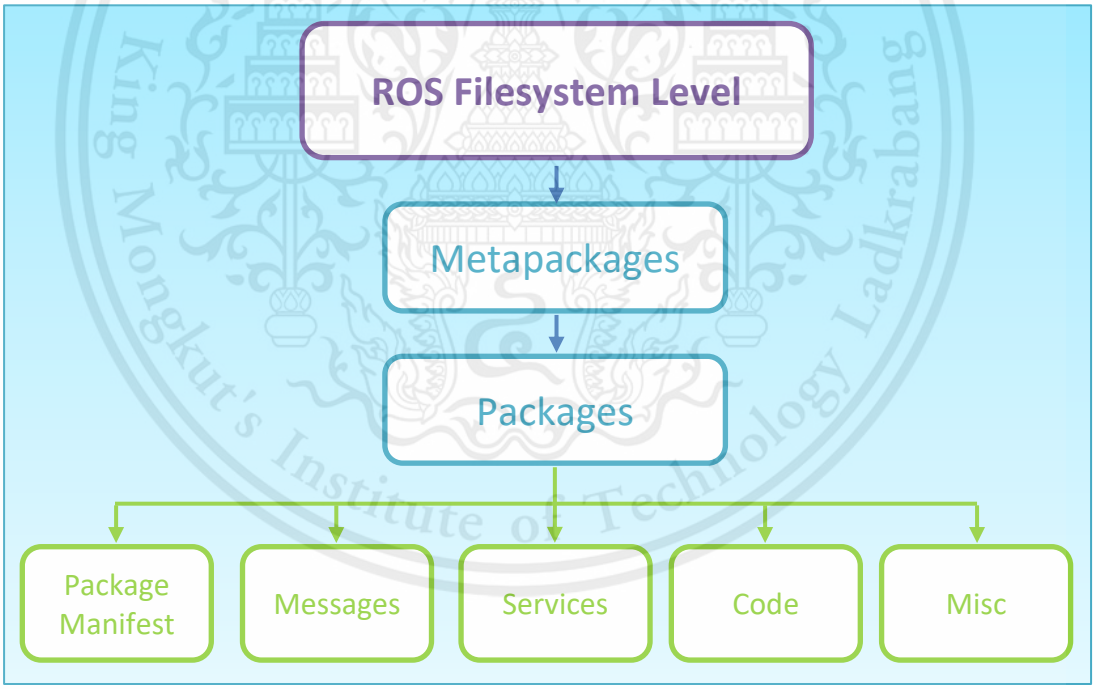

Figure 7 Architecture of ROS Filesystem

- **Packages**: Software in ROS is organized primarily into packages. Anything that can benefit from being clustered together and distributed as a whole can be contained in a package, including ROS runtime processes (nodes), a library that relies on ROS, data sets, and configuration files. In ROS, packages are the smallest possible unit for both the build process and the release process. In other words, a package represents the smallest unit that can be designed and constructed and released.

- **Metapackages**: Packages whose entire objective is to stand in for a collection of related packages. It is frequent practice to use metapackages to temporarily store converted rosbuild Stacks in a way that is compatible with older versions of the software.
- **Package manifest**: The package's author, dependencies, compilation flags, license, and other metadata can be found in its manifest file, which is located within the package itself. The ROS package contains the package.xml manifest file.
- **Metapackages manifest**: The metapackage manifest has the same capabilities as the package manifest, except that it cannot contain packages as runtime dependencies or define an export tag.
- **Messages (.msg)**: In the msg folder of a package, custom messages can be declared.(for example, my package/msg/MyMessageType.msg). The file type for a message is.msg.
- **Services (.srv)**: The srv folder within a package is where the response and request data types are defined (my package/srv/MyServiceType.srv).
- **Repositories**: ROS packages are managed in repositories from which a Vesion Control System (VCS) like Git, Subversion (SVN), or Mercurial (hg). Any repository is a collection of files stored in a version control system. [10]

2.2.3.1.2 ROS Computation Graph Level

The ROS communication middleware packages are contained within the Ros comm stack; these packages, when grouped together, are referred to as the ROS graph layer: ROS makes use of a network of ROS nodes in order to perform computations. The term "computation graph" refers to this network of computational nodes. The computation graph is comprised of several key concepts, the most important of which are ROS nodes, the master, the parameter server, messages, topics, services, and bags. The representation of each concept in the graph was accomplished through a variety of means. [10]

A stack known as Ros comm contains all of the ROS communication-related packages. This includes core client libraries like roscpp and rospython, as well as the implementation of concepts like topics, nodes, parameters, and services (http://wiki.ros.org/ros_comm). [10]

Tools such as rostopic, rosparam, rosservice, and rosnode are included in this stack as well. These tools are used to investigate the concepts that came before.

The ROS communication middleware packages are contained within the ros_comm stack: these packages, when grouped together, are referred to as the ROS graph layer:

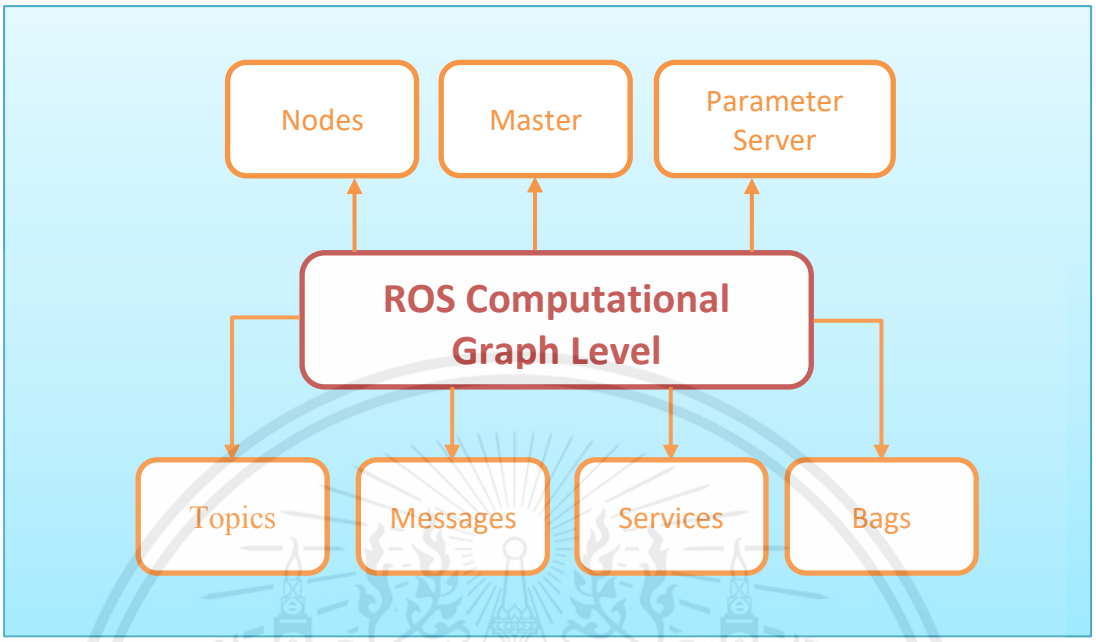

Figure 8 Architecture of ROS Computational Graph Level

- **Nodes**: Nodes represent computational processes. ROS nodes are written by using various ROS client libraries that are related to their function. For the ROS nodes, their architecture is less complex due to the communication method. Because the node can easily construct any single process instead of a large group with all the needed capabilities, ROS nodes are easier to troubleshoot because of their basic architecture.
- **Master**: A ROS master works as a manager and communication hub. The ROS master registers the headline name and monitors all interrelated functioning nodes. Accordingly, a lack of a registered master will affect communication between nodes because nodes will be unable to locate and communicate with others. Therefore, the process of exchanging messages, requesting services, etc., will not occur in the system.
- **Parameter Server**: A parameter server is an element of the ROS master. The parameter server enables saving information in a central location that is accessible and modifiable.
- **Topics**: Each message in ROS is conveyed by means of identified buses known as topics. When a node is registered it will start message transmitting through a topic, known as publishing a topic. After the node receives a message from a topic, meaning the node subscribes to the subject. The publishing node and the subscribing node have no knowledge of each other's existence. Even if there is no publisher for a subject, we can subscribe to it. Information creation and consumption are disconnected, to put it briefly. Each topic has a unique name, and any node that has the appropriate message type can transmit data over it and access it.
- **Logging**: For a logging system in ROS is required for storing raw data, such as sensor data, that can be difficult to obtain but is required for creating and testing

robot algorithms. Bagfiles are referred to as such. When working with sophisticated robot mechanisms, bagfiles are incredibly helpful tools. [10]

The graph that follows illustrates how the nodes communicate with one another through the use of topics:

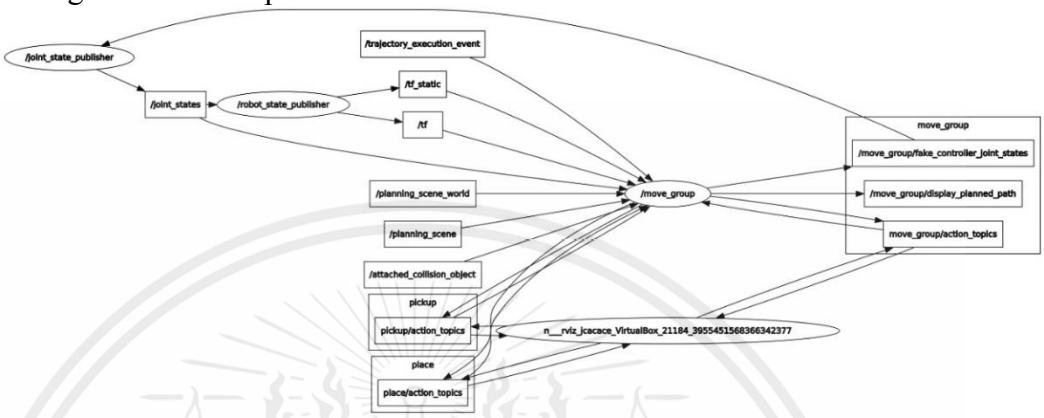

Figure 9 Communication of nodes while using topics [10]

2.2.3.1.3 ROS Community Level

These ROS alternatives facilitate for a newly formed community inside ROS to share software and information with each other. The following is an inventory of the numerous resources that are available in these communities:

- **Distributions**: A ROS distribution is a collection of installable metapackages, much like Linux distributions are. Installing and compiling ROS software is a simplicity owing to the ROS distributions. Further, they ensure that all of the programs in a package are running the same version.
- **Repositories**: ROS is dependent on a decentralized network of code repositories called a "federated network of code repositories." Within this network, several institutions are able to build and release their own individual robot software components. [10]

2.2.3.2 Higher-Level Concepts

2.2.3.2.1 Coordinate Frames

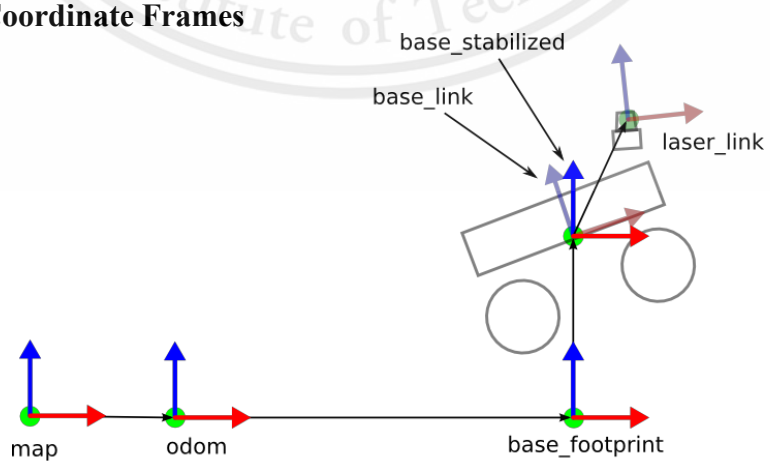

Figure 10 The illustration of the relationship between each coordinate frame [11]

In figure 10 there are 3 colors representing different axes for coordinate frames including: red arrows are the x-axis (yaw), blue arrows are the z-axis (roll), and green dots are the y-axis (pitch) in ROS coordinates.

- **map:** The coordinate frame of the map is a world-fixed frame that the Z-axis pointing upward. A mobilizing part's position with respect to the map's frame must not change considerably by the time. The map frame is discontinuous, so the position of a mobilizing part within the map frame will be varied in discrete moves at any time. [12]
- **odom:** The odom coordinate frame is a world-fixed frame. A moving platform's orientation in the odom frame might change continuously throughout time. The odom frame can no longer be used as a permanent global reference because of this shift. However, a robot's odom frame position is always certain to be continuous, meaning that the odom frame pose of a moving platform never changes suddenly. [12]
- **base_footprint:** It originates directly beneath the robot's center. It is the robot's 2-dimension posture. This coordinate frame moves along with the robot. [13]
- **base_link:** The base link coordinate frame is linked to the base of the mobilizing vehicle. The base link may be joined to the base in any placment of the mobility platform, meaning that a different reference point will exist on the base. [12]
- **laser_link:** The position of laser link is issued from the laser sensor's core. This coordinate frame is static (fixed) with relation to the base link. [13]
- **sensor_link:** The sensor link coordinate frame is defined when another sensor, such as an IMU sensor, is applied to the vehicle. The origin of this coordinate frame is the core of the sensor and fixed frame. The coordinate frame that is linked to the base of the mobilzing vehicle (base_link). [13]

2.2.3.2.2 TF

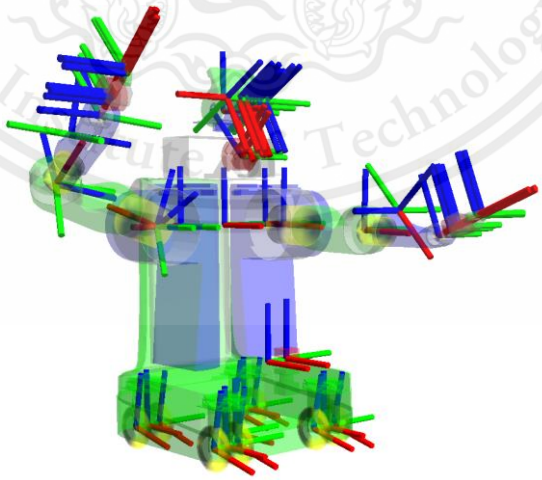

Figure 11 The illustration of TF on a robot [14]

In order to monitor coordinate frames, the TF library is necessary to work with. The TF library not only tracks the coordinate data for the system but also transfers data for the whole system. As robotic systems get more demanding, the ability to narrow in on the

task frame and only the relevant coordinate frames become significant. The majority of robotic systems integrate sensor data from several coordinate frames. [15]

Static Transform Publisher

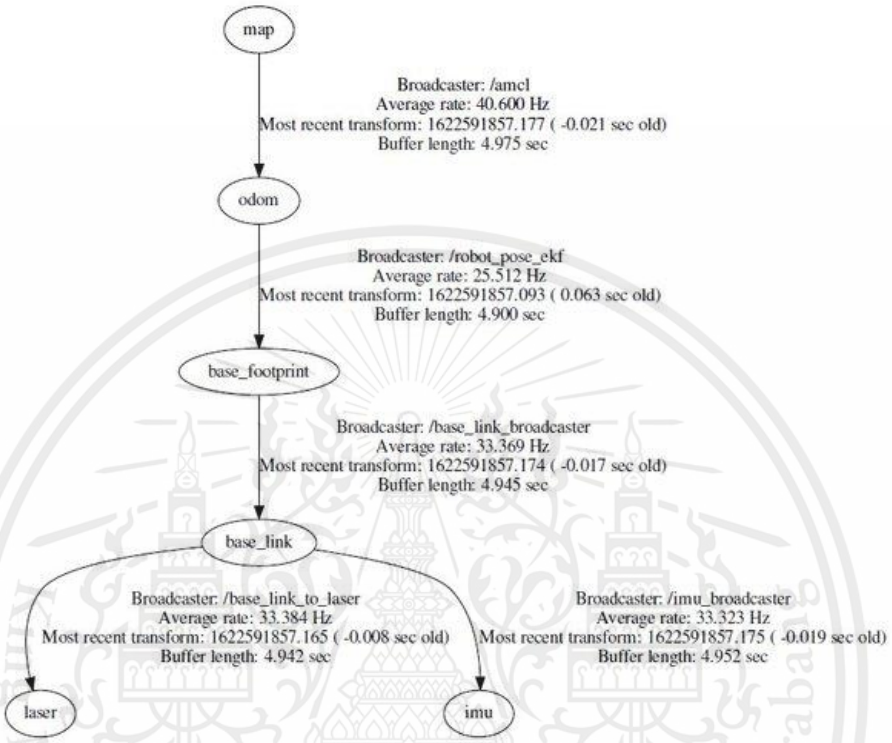

Figure 12 Map of the static transform publishers of all coordinate frames [13]

- **map to odom:** The map-to-odom transformation reveals the placement and orientation of the robot's initial position (the odom coordinate frame, which represents the child) within the map's coordinate frame. [13]
- **odom to base_footprint:** The data transformation from odom to base_footprint is not static according to the robot's global movement. As the wheels rotate, the base footprint coordinate frame will change continuously. [13]
- **base_footprint to base_link:** Because both coordinate frames are fixed, the data transformation between base footprint and base link is a static transform. When the robot is in motion, the footprint's coordinate frame will therefore move. [13]
- **base_link to laser_link:** The transformation of data from base_link to laser_link yields the position and orientation of the laser within the coordinate frame of base link. Therefore, this transformation is also static. [13]
- **base_link to sensor_link:** the base link to sensor link data transformation method returns the position and orientation of the sensor within the base link's coordinate frame. Consequently, transition occurs statically. [13]

2.2.4 PlatformIO

For the embedded market, PlatformIO takes a unique approach by providing developers with a state-of-the-art integrated development environment (Cloud & Desktop IDEs)

that is cross-platform, supports a wide variety of software development kits (SDKs) or frameworks, and features advanced debugging (Debugging), unit testing (Unit Testing), automated code analysis (Static Code Analysis), and remote management (Remote Management) (Remote Development). Engineered to maximize developer freedom, it supports both graphical and command line editors (PlatformIO Core (CLI)). PlatformIO is a system that lets you write code for your Arduino board and then compile and upload it to the board in either the standard Arduino format used by the Arduino IDE or in PlatformIO's own format. [16]

The PlatformIO package comes in two flavors:

- **PlatformIO Core**: command-line applications intended for installation as the purpose. It is a one-of-a-kind, custom-built build system that frees developers from the restrictions of any given SDK or sample embedded application by handling the tedium of software integration, packaging, and library dependencies. [17]
- **PlatformIO IDE**: Which sets up an integrated development environment (IDE) plugin for the Atom and Visual Studio Code editors. Similar IDE donot technically support some plugins, the pio command can create project files for those programs so that AVR code can be developed in a user-friendly IDE. [17]

2.3 Hardware

2.3.1 Raspberry Pi

Raspberry Pi is a class of credit card-sized Single Board Computers (SBCs) designed by the Raspberry Pi Foundation in the United Kingdom. A single-board computer (SBC) is a completely working on the computer system based on a single printed circuit board. A Single-Board Computer (SBC) is a little computer that has the essential components for basic computing tasks including a microprocessor(s), memory, input/output, and other similar components. Most SBCs, in contrast to desktop PCs, do not provide expansion slots for adding new peripherals or functionality. Because the SBC's processor(s), memory(s), graphics processing unit(s), etc. are all built into a single PCB, it is not possible to upgrade it. [18]

Raspberry Pi is used in a wide variety of settings, from education to manufacturing. They are used to teach children to code, to create various hardware projects, to make their homes smarter, and to power Edge computing. The Raspberry Pi is an afforable Linux computer with General purpose input / output ports (GPIO) for experimenting with physical computing and the Internet of Things (IoT).[19]

In 2009, the Raspberry Pi Foundation was established. By providing a cost-effective computing platform, Raspberry Pi aims to boost computer science education in schools and underdeveloped countries. The Raspberry Pi was released in 2012 by the Raspberry Pi Foundation. The Raspberry Pi has altered subsequent versions of the board. They also released more Pi additions. [18] There are various Raspberry Pi variants applicable, as shown below:

- Pi 1 Model B (2012)
- Pi 1 Model A (2013)

- Pi 1 Model B+ (2014)
- Pi 1 Model A+ (2014)
- Pi 2 Model B (2015)
- Pi Zero (2015)
- Pi 3 Model B (2016)
- Pi Zero W (2017)
- Pi 3 Model B+ (2018)
- Pi 3 Model A+ (2019)
- Pi 4 Model A (2019)
- Pi 4 Model B (2020)
- Pi 400 (2021)

[19]

The features of the Raspberry Pi that are used the most frequently are listed in the table that can be found below. The following is a comparison of those functions:The features of the Raspberry Pi that are used the most frequently are listed in the table that can be found below. The following is a comparison of those functions:

Table 2.1 Comparison of Raspberry Pi's Feature [20]

Features	Raspberry Pi Model B+	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B	Raspberry Pi Zero
SOC	BCM2835	BCM2836	BCM2837	BCM2835
CPU	ARM11	Quad Cortex A7	Quad Cortex A53	ARM11
OPERATING FREQ.	700 MHz	900 MHz	1.2 GHz	1 GHz
RAM	12 MB SDRAM	1 GB SDRAM	1 GB SDRAM	512 MB SDRAM
GPU	250 MHz Videocore IV	250MHz Videocore IV	400 MHz Videocore IV	250MHz Videocore IV
STORAGE	micro-SD	Micro-SD	micro-SD	micro-SD
ETHERNET	Yes	Yes	Yes	No

There have been several generations of Raspberry Pi released so far, from the Pi 1 to the Pi 4, and even a Pi 400. Typically, there have been two distinct variations: Model A and Model B. Model A is considerably less priced but also has less RAM and limited ports such as USB and Ethernet ports. The Pi Zero is a diminutive and inexpensive variant of the original (Pi 1).[19]

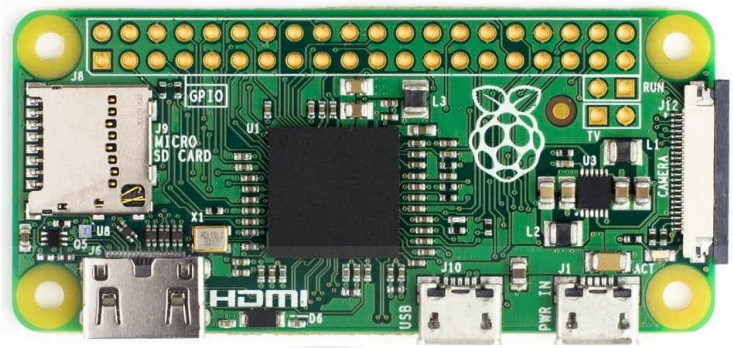

Figure 13 Raspberry Pi Zero (2015) [20]

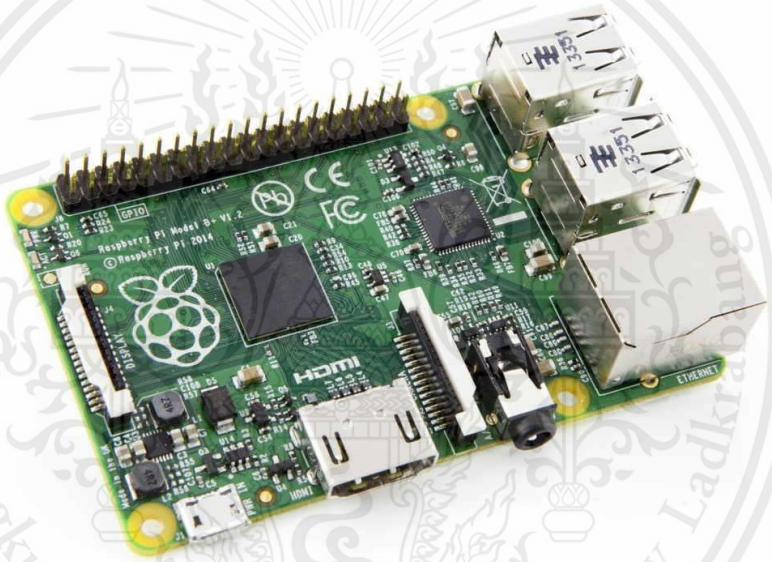

Figure 14 Raspberry Pi 3 Model B+ (2018) [20]

Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B is the newest iteration of the widely used Pi computer line. Compared to its predecessor, it boasts unprecedented improvements in processing speed, multimedia performance, memory, and connection. Raspberry Pi 3 Model B+ is upgraded while keeping compatibility and power use the same as previous models. Users can expect desktop performance from a Raspberry Pi 4 Model B that is on par with budget x86 PCs. [21]

Some of the features that set this product apart from the competition include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and Power over Ethernet (PoE) capability (via a separate PoE HAT add-on).[22]

With certifications for dual-band wireless LAN and Bluetooth compliance, the board can be integrated into final devices with significantly less compliance testing, hence lowering costs and shortening development times.[22]

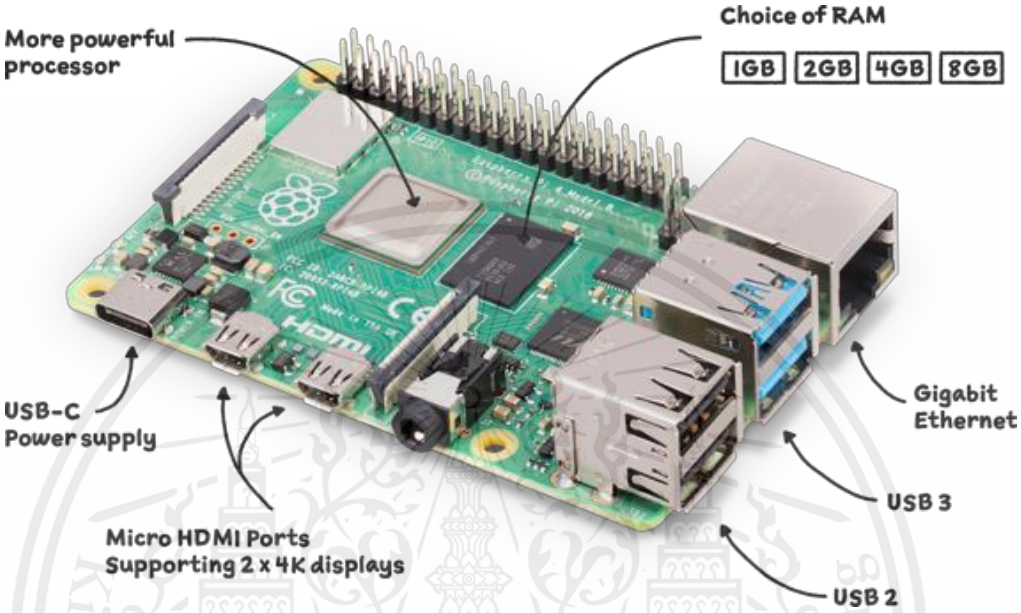

Figure 15 Raspberry Pi 4 Model B (2020) [23]

Peripherals

- **Display Parallel Interface (DPI)**: The GPIOs offer a standard RGB parallel interface (DPI). This parallel interface supports a secondary display with up to 24 bits. [24]
- **SD/SDIO Interface**: The Pi4 model B's SD card connection supports 1.8V, DDR50 mode (with a maximum bandwidth of 50 Megabytes per second). In addition, the GPIO pins offer a classic SDIO interface. [24]
- **Camera and Display Interfaces**: The Pi4 model B includes a 2-lane MIPI CSI Camera and a 2-lane MIPI DSI Display connection. These connections are compatible with older Raspberry Pi boards and support all camera and display peripherals for the Raspberry Pi. [24]
- **USB ports**: The Pi4 model B includes two USB2 and two USB3 type-A connections. Total downstream USB current is limited to around 1.1A across the four sockets. [24]
- **HDMI**: The two micro-HDMI ports of the Pi 4 model B are compatible with CEC and 4Kp60 at resolutions up to HDMI 2.0. [24]
- **Audio and Composite**: The Pi4 model B has a composite TV-output and an audio output that is nearly CD quality through a 4-ring TRS 'A/V' connection. The 32 Ohm headphones are natively supported by the analog audio output. [24]
- **General-Purpose I/O (GPIO)**: The Pi4 model B's 40-pin header has 28 BCM2711 GPIOs. This header is compatible with prior 40-way Raspberry Pi boards. [24]

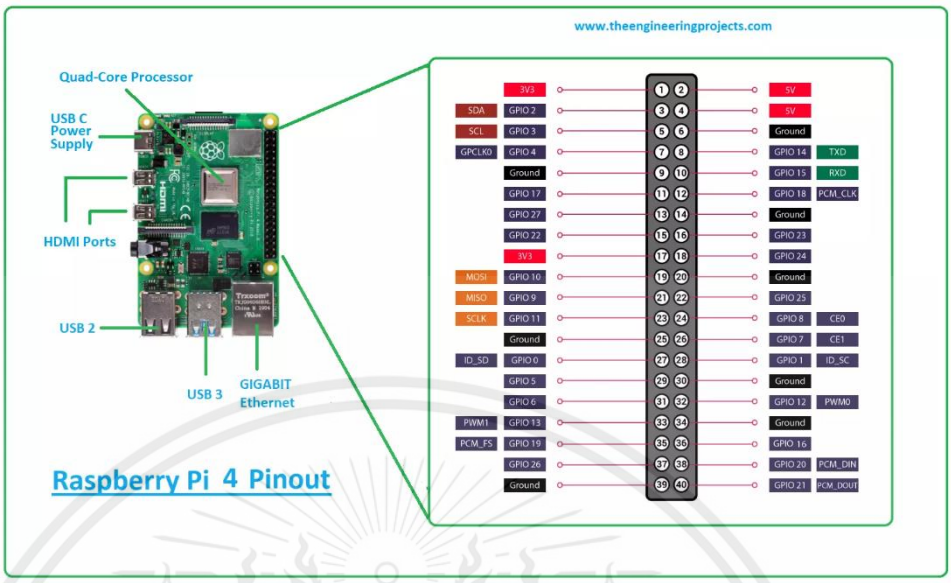

Figure 16 Illustration of Raspberry Pi 4 Pinout [25]

Table 2.2: GPIO Alternative Functions for Raspberry Pi 4 [24]

GPIO	DEFAULT PULL	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_ MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6

23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

2.3.2 Sensors

2.3.2.1 RPLiDAR

Figure 17 RPLiDAR Model A2M8 [26]

SLAMTEC has created the RPLIDAR A2, a 360-degree, low-cost, 2D laser scanner (LiDAR) solution. Laser range finding samples at high speeds can be taken at up to 4000 per second. In addition, it is built with SLAMTEC's unique OPTMAG technology, which allows it to outlast the reliability issues that plague conventional LIDAR systems. [27]

Within a range of 6 meters, the device is capable of performing a 2D full-circle scan. The 2D point cloud data that is produced can be utilized for several purposes, including mapping, localization, and environment simulation. [27]

The RPLIDAR A2 typically operates at a scanning frequency of 10hz (600rpm). There will be a 0.9° resolution if these conditions hold. In addition, customers have complete control over the actual scanning frequency, which may be set anywhere from 5 to 15 Hz to best suit their needs. [27]

As a result of using SLAMTEC's low-cost laser triangulation measuring technology, the RPLIDAR A2 performs admirably in any interior setting and under non-direct sunlight conditions when used outside. Meanwhile, all RPLIDAR A2s are placed through rigorous testing to guarantee their laser output powers are up to FDA Class I requirements before they are released. [27]

Mechanism

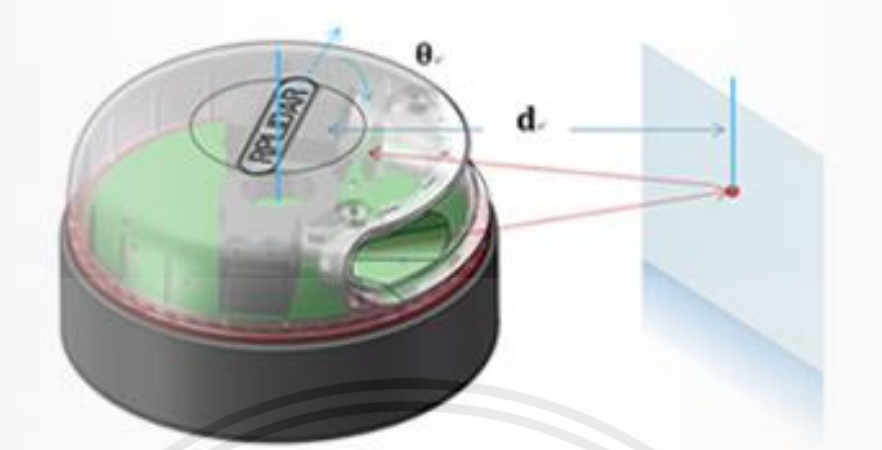

Figure 18 RPLiDAR Operation [26]

The RPLIDAR A2 utilizes the high-speed vision acquisition and processing technology that was created by SLAMTEC. It functions on the laser triangulation approach ranging as shown on the figure 18, which was invented by SLAMTEC. The system iterates at a rate of almost 4000 times every single second. [27]

The RPLIDAR sends out a modulated infrared laser signal at each step of the ranging process, which is subsequently reflected by the target object. This allows the target object to be located. The returning signal is then sampled by the RPLIDAR's vision acquisition system, and the DSP that is embedded within the RPLIDAR begins administering the sampled data. Over the communication link, it then transmits the measured distance and angle between the obstacle and the RPLIDAR.[27]

The range scanner core will do a 360-degree scan for the present surroundings while it is being driven by the motor system, which will cause it to revolve in a clockwise direction. [27]

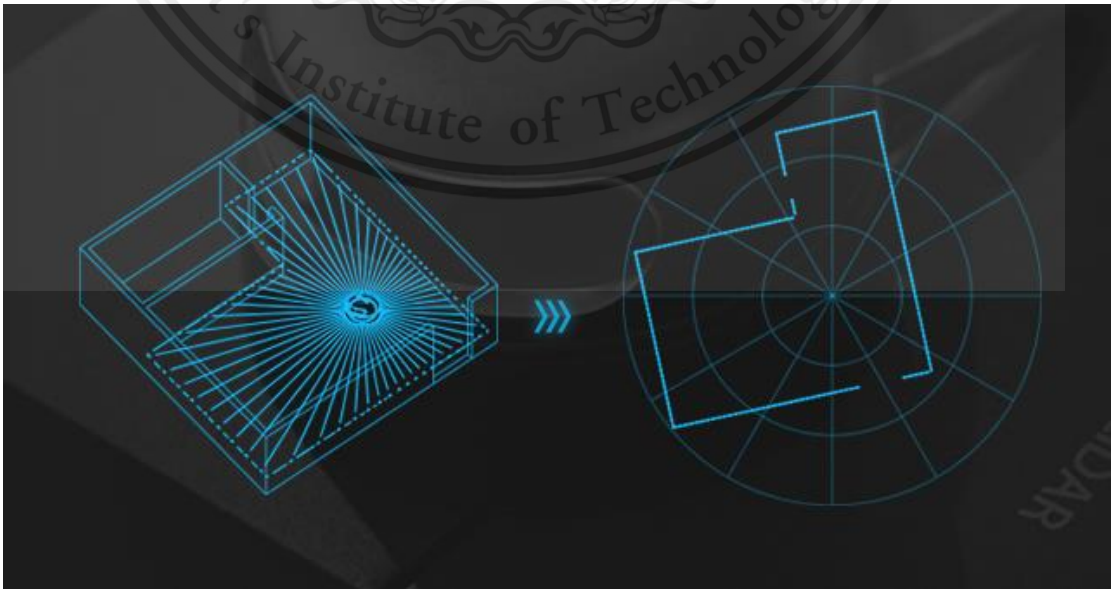

Figure 19 Simulation of environment mapping that was obtained through RPLiDAR scanning [26]

2.3.2.2 MEMS sensor

Microelectromechanical systems, more often known as MEMS, are a success of the twenty-first century and have transformed the semiconductor industry by integrating the science of microelectronics with micromachining. Despite the fact that the word "mechanical" is commonly used to describe MEMS, true to its meaning, micromachined devices need not necessarily incorporate mechanical components. However, the issue of miniaturization, which MEMS technology was designed to solve, still has to be dealt with. MEMS, or microelectromechanical systems, is essentially a platform that enables the fabrication of a variety of typical small mechanical elements, such as channels, holes, cantilevers, membranes, cavities, and other structures. [28]

Despite the fact that MEMS are often connected with micromachining, the assemblies themselves are not molded. Instead, microfabrication technology that is suited for batch processing of integrated circuits are the manufacturing process. This allows for a more streamlined production process. [28]

The manufacture of MEMS might be done in batches, which is a potential advantage. Batch processing is extremely cost-effective; hence, MEMS contributes to cost-effectiveness in the microelectronics sector. [28]

When selecting MEMS, the focus is placed more on their mechanical qualities than than their electrical properties. Although it is dependent on the application in question, microelectromechanical systems (MEMS) have been shown to exhibit mechanical features such as high stiffness, high fracture strength, fracture toughness, high-temperature stability, and chemical inertness. [28]

2.3.2.2.1 Accelerometer and Gyroscope

Firstly, begin with accelerometers. These are, as the name indicates, designed to measure acceleration. Therefore, anytime there is acceleration, accelerometers must detect the acceleration and provide the relevant data. [29]

A gyroscope is a sensor that uses Earth's gravity to precisely detect the robot's orientation. An accelerometer is a device designed to measure accelerations that are not caused by gravity. When acceleration occurs, the crystals in the accelerometer become excited and provide a voltage proportional to the acceleration. [29]

The gyroscope, in contrast, can detect rotation, although the accelerometer cannot. The accelerometer can, in a sense, determine the orientation of a stationary object in regard to the Earth's surface. When speeding in a specific direction, the accelerometer cannot differentiate between that acceleration and the acceleration caused by the Earth's gravitational attraction. The gyroscope monitors the rotational speed around a certain axis. The accelerometer, in contrast, detects linear acceleration. [29]

Accelerometer

Accelerometers are sensitive enough to detect not just gravitational acceleration but also displacement and instantaneous linear accelerations, as well as other forms of background noise and disturbance. The following diagram illustrates the accelerometer model. [30] :

$$\alpha_a = r(\alpha_{accel} - g) + d_a + n_a$$

- α_a : measured accleration
- α_{accel} : external acceleration
- r : orientation of the sensor
- g : gravitational acceleration
- d_a : drift of accelerometer
- n_a : noise of acclerometer

Gyroscope

Commonly employed for the determination of angular velocity, a gyroscope gives the rate of change in orientation angle and the quality of rotary motion of the item throughout x,y, and z-axis in the object coordinate system and gyroscope model is as follows. [30] :

$$\omega_{gyro} = \omega + d_g + n_g$$

- ω_{gyro} : measured angular velocity from the gyroscope
- ω : latent ideal angular velocity
- d_g : drift of gyroscope which changes with time and other factors
- n_g : noise of gyroscope

MPU6050

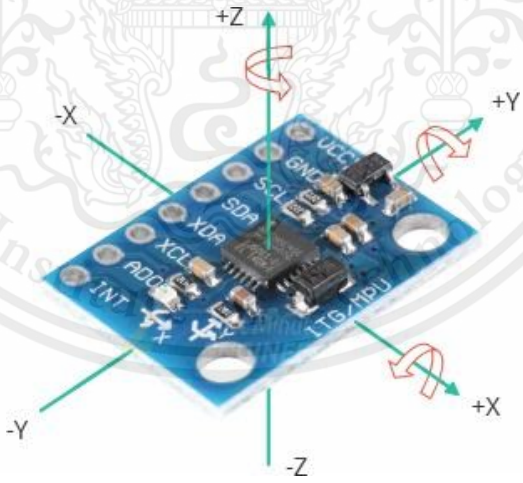

Figure 20 MPU6050 with three dimensions of movement [31]

The MPU-6050 sensor module, in order to ascertain the position of an angle within a three-dimensional space. The inertial measurement unit (IMU) was employed in a combination of two sensors (accelerometer and gyroscope). [30]

A three-dimensional vector, in meters per second squared, indicating the acceleration of the device less the acceleration of free fall, g, is provided by an accelerometer sensor.

The gyroscope delivers readings in radians per second that represent the object's angular velocity along all three axes. [30]

Get a precise angle using the MPU-6050 MEMS sensor's 6-DOF (3-axis gyroscope and 3-axis accelerometer) sensors on a single chip. The gyroscope provides an indication of rotational velocity (angular displacement in time), whereas the accelerometer provides information about linear displacement in time. [30]

This auxiliary device is an excellent illustration of an economically viable semiconductor that provides high quality orientation and acceleration data. To the standard set of pins, the breakout circuit adds a whole bunch more. An additional chip selection pin (AD0) allows the I2C address to be changed, and an interrupt pin allows for on-demand reading from the MPU6050, much like the GPIO expander. [32]

MPU6050 Module

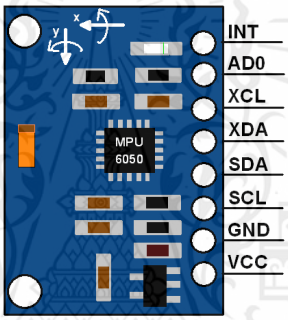

Figure 21 Illustration of MPU6050 pins [33]

- **INT**: Interrupt digital output pin.
- **AD0**: The least important bit of the I2C slave address. This is the lowest bit of the 7-bit slave address for the device. If connected to VCC, it is read as logic one, and the slave's address is reset.
- **XCL**: A port for the serial clock's spare input. This connector connects the MPU-6050 to sensors that use the I2C bus and have a SCL jack.
- **XDA**: Pin for additional serial data. The SDA pin of other I2C-enabled sensors can be connected to the MPU-6050 using this connector.
- **SCL**: Serial Clock input. Connect this pin to the SCL pin of the microcontroller.
- **SDA**: Serial Data pin. Connect this pin to the SDA pin of the microcontroller.
- **GND**: Ground terminal. Connect this terminal to the ground.
- **VCC**: Pin for power supply. Connect this pin to a +5V DC power source. [33]

2.3.3 LiPo Battery

Lithium Polymer batteries (henceforth "LiPo" batteries) are utilized in many consumer electronics gadgets. They've gained prominence in the RC business in recent years and are currently the top choice for extended run periods and great power. [34]

Clearly, lithium is a component in lithium-polymer batteries. As an alkali metal, lithium interacts with water and ignites. Lithium reacts similarly to oxygen in that it burns when heated. Extra Oxygen and Lithium atoms are created at either end of the battery (the

cathode or the anode) as a result of the sometimes rough treatment batteries receive in the R/C world. This can cause the anode or cathode to accumulate Lithium Oxide (Li2O). The two most common types of corrosion are iron oxide, also known as "rust," and lithium oxide, which is essentially corrosion of the lithium sort. The addition of Li2O raises the internal resistance of the battery. The internal resistance of a circuit is the resistance encountered by an electric current within the circuit. Higher internal resistance has the practical effect of causing the battery to heat up more during operation. [34]

Example

Figure 22 LiPo Battery 11.1 V 3S RC 60C 2250 mAh [35]

Qualifications

- Brand: SUNPADOW
- Battery Chemistry: Lithium polymer
- Configuration: 3S1P
- Voltage: 11.1V
- Capacity: 2250mAh
- Constant Discharge: 60C
- Peak Discharge: 120C
- Dimension: 107×35×28 mm
- Connector Type: XT60

Features

LiPo battery contains materials of selected premium grade, low resistance, high dependability, and good consistency. In addition, in terms of its physical features, it is both lightweight and compact. Which concerns the most perfect form of power supply for the Hobbyking series.

CHAPTER 3
METHODOLOGY

3.1 Introduction

This chapter provides a comprehensive description of the methodology, including an explanation of how the overall processes will be carried out and the reasons why these processes are relevant. Initially, to clarify the steps for this research, the plan and working procedures are organized along with the schedule in the section 3.2. Accordingly, both hardware and software will be listed in section 3.3 for all used materials. Lastly, the section 3.4 concludes with a demonstration and explanation of the robot assembly, software installation, port forwarding, calibrations, G-mapping, and navigation procedures.

3.2 Action Plan

	2022															
	Aug				Sep				Oct				Nov			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Study the operation of SLAM robot and study on related research																
Study on the circuit and hardware																
Assemble robot model A																
Software installation																
Calibration																
Collect the map indoor																
Navigation indoor																
	2023															
	Jan				Feb				Mar				Apr			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Assemble robot model B																
Software installation																
Calibration																
Collect the map indoor																
Navigation indoor																

3.3 Materials

The materials utilized for this project are divided into two categories: hardware equipment and software component. And hardware equipment is classified into three classes based on their respective functions which are robot body, central processing unit and motor control.

3.3.1 Hardware equipment

- Robot Body
  - WT-600 tank car chassis
  - LiPo battery 2250 mAh, 3S, 11.1V, 60C
- Central Processing Unit
  - Raspberry Pi 4 model B
  - RPLiDAR
- Motor Controller
  - Arduino Mega2560
  - Motor Driver
  - Encoder
  - MPU6050

3.3.2 Software component

- SD card
- Ubuntu MATE 20.04

3.4 Methods

3.4.1 Hardware Design

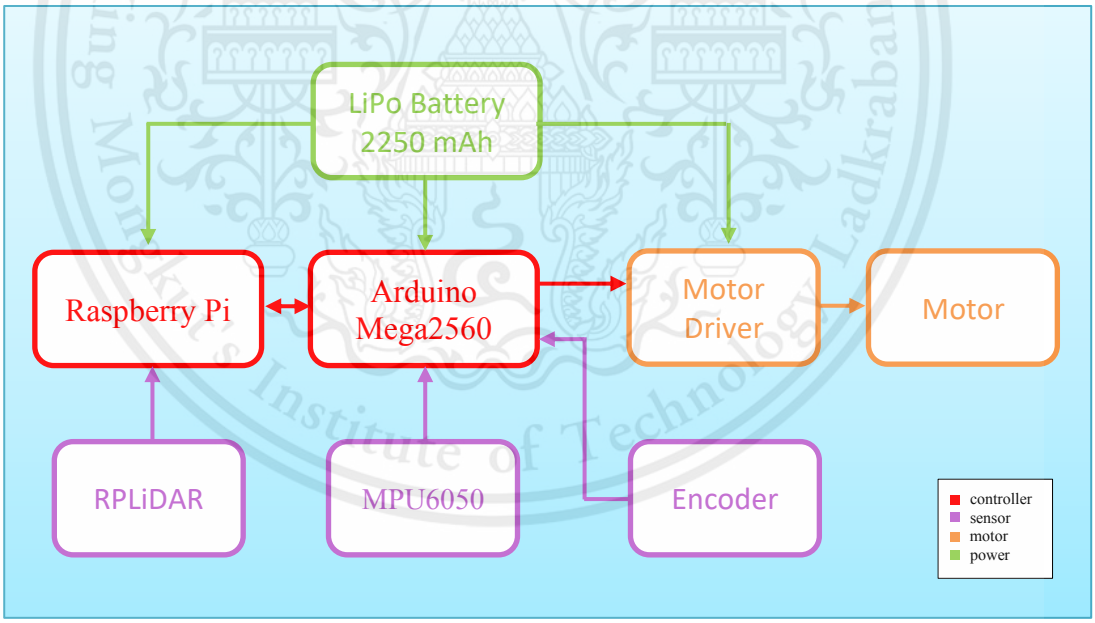

Figure 23 Control system of the robot

3.4.1.1 Circuit Design

Figure 23 in red describes the two main parts for the controller. Firstly, the Raspberry Pi, a microprocessor, is assigned as a central processing unit. Secondly, the Arduino Mega2560, a microcontroller, is assigned as a motor controller. Each section is powered by a LiPo battery. The Raspberry Pi is connected to the RPLiDAR, which transmits real-time data about its surroundings to the Raspberry Pi, allowing it to detect its environment and track itself on a map. When the Raspberry Pi sends a transmission

order to the Arduino Mega2560, the transportation begins. After that the Arduino Mega2560 drives the motor with the motor driver and tracks and limits the amount of distance to reach the respective destination by utilizing the motor encoder. And real-time data from the motor encoder is transmitted to the Arduino Mega2560 and sent to the Raspberry Pi to track the amount of distance on the user interface.

3.4.1.2 Drawer Design

The drawer runner (figure 24) and the drawer container(figure 25) were design to store the drug in order to process the drug delivery.

3.4.1.3 Drawer runner part

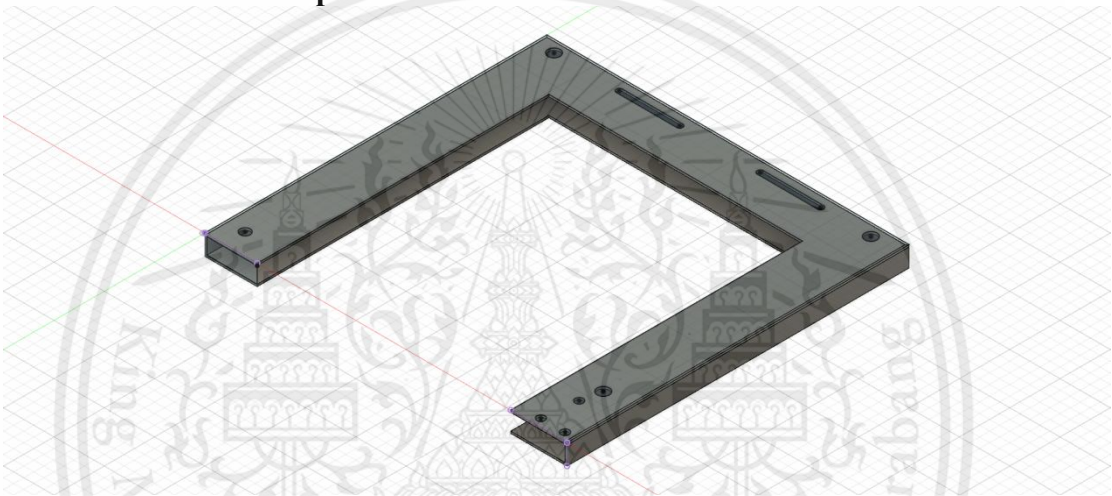

Figure 24 The design of drawer runner

3.4.1.4 Container part

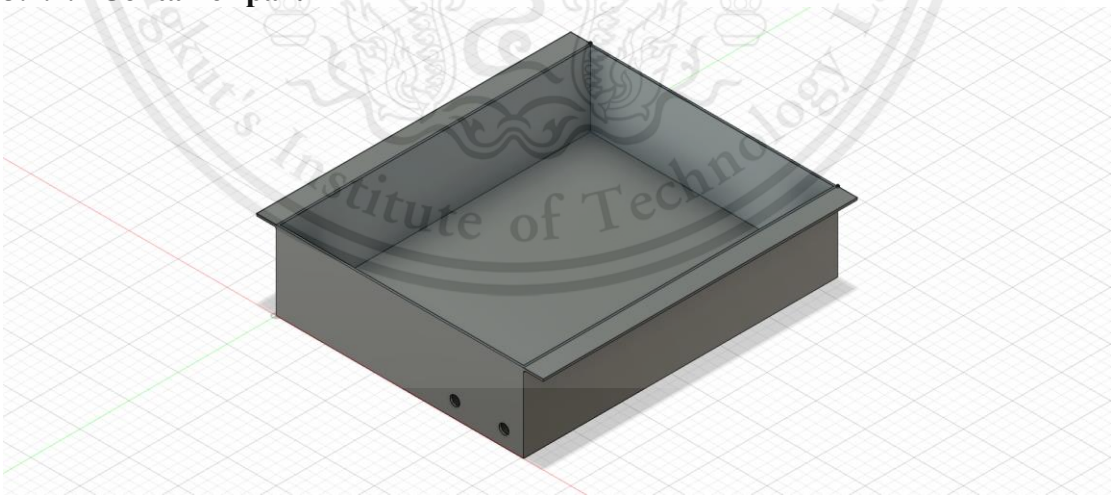

Figure 25 The design of drawer container

3.4.2 Software Process

3.4.2.1 Initial System Setting

Firstly, an operating system (OS) is required, so install Ubuntu Mate 20.04 on the SD card to be used as an OS for the Raspberry Pi. After launching the SD card in the

Raspberry Pi to start the OS, configure the Wi-Fi settings in order to work remotely only for the coding by utilizing ssh-server and net-tools. Then, use PuTTY and the SSH connection to connect to the Raspberry Pi from a PC using the same Wi-Fi connection and the IP address of the Raspbeery Pi.

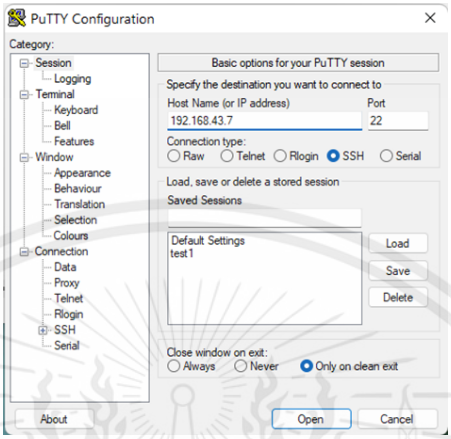

Figure 26 PuTTY Configuration

However, for the visualization work, the remote desktop connection is required. Therefore, xrdp, a remote desktop package for the Raspberry Pi, needs to be installed.

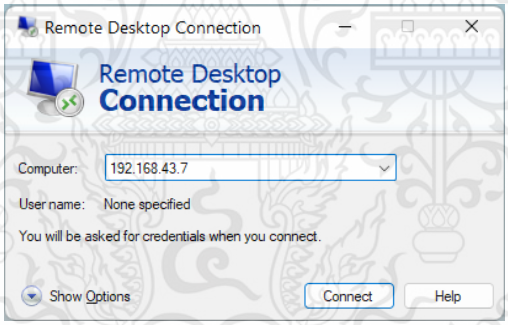

Figure 27 Remote desktop Connection

Install Robot Operating System (ROS), which is compatible with Ubuntu mate 20.04, ROS1 Noetic, after the OS has been set. There are necessary packages for the robot's performance including navigation, teleop, gmapping, gazebo, location, control system, sensor, and rviz. Then, create a catkin workspace to aid in the execution of the ROS code, and install Hello Robot for the robot's basic functionality. Lastly, install platform.io to be used as a headless Arduino IDE.

3.4.2.2 Component Inspections

3.4.2.2.1 Encoder and Motor

Beginning the inspection with the encoder. A bringup.launch file must be executed in order to investigate the encoder's performance. As shown in Figure 28, the bringup monitor displays the values for each encoder, including the front left, front right, rear left, and rear right. Therefore, the inspection process can begin by manually rotating wheels to examine encoder data. Finally, review that all encoder values are roughly identical.

```
[INFO] [1668072829.155869]: Encoder FrontLeft  : 1779
[INFO] [1668072829.164069]: Encoder FrontRight : 1772
[INFO] [1668072829.171324]: Encoder RearLeft   : 0
[INFO] [1668072829.178285]: Encoder RearRight  : 0
```

Figure 28 The bringup monitor shows data of encoders

When encoder values are acceptable, configure an approximation in the lino_base config.h file to improve the robot's performance. Moreover, configure the wheel diameter, pulse width modulation, distance between front and rear wheels, and distance between left and right wheels.

*lino_base_config.h

```
27
28 //define your robot' specs here
29 #define MAX_RPM 120                    // motor's maximum RPM
30 #define COUNTS_PER_REV 1775            // wheel encoder's no of ticks per rev
31 #define WHEEL_DIAMETER 0.035           // wheel's diameter in meters
32 #define PWM_BITS 8                     // PWM Resolution of the microcontroller
33 #define LR_WHEELS_DISTANCE 0.250       // distance between left and right wheels
34 #define FR_WHEELS_DISTANCE 0.450       // distance between front and rear wheels.
   Ignore this if you're on 2WD/ACKERMANN
35 #define MAX_STEERING_ANGLE 0.415       // max steering angle. This only applies to
   Ackermann steering
```

Figure 29 The lino_base_config file

After passing encoder inspection, the next step is checking motor performance. To test the motor performance, a teleop_twist_keyboard is required, as shown on the left side of figure 23. The teleop_twist_keyboard is used for driving and controlling the speed and direction of the vehicle. To verify motor performance, ensure that the robot's movement corresponds to the order from the teleop_twist_keyboard.

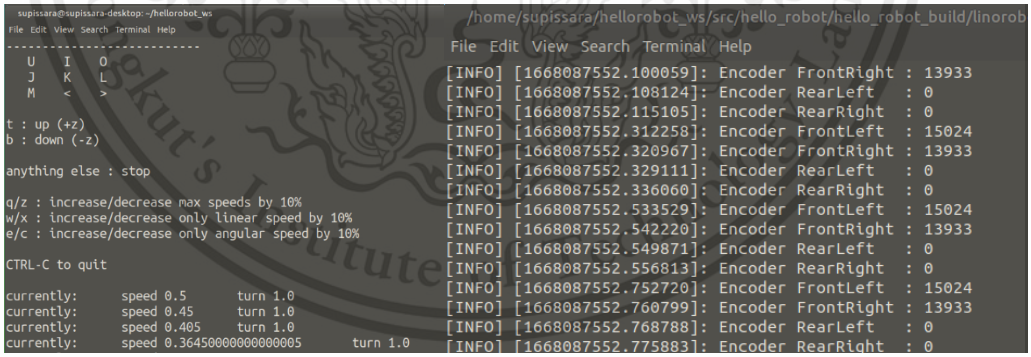

Figure 30 The Teleop monitor for driving robot with the Bringup monitor

3.4.2.2.2 IMU

To inspect IMU we will check with the Arduino IDE to check the result of pitch roll and yaw. The example of MPU6050_raw is required to test as shown in figure 31.

MPU6050_raw | Arduino 1.8.19

File Edit Sketch Tools Help

MPU6050_raw

```
    */

    // configure Arduino LED pin for output
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // these methods (and a few others) are also available
    //accelgyro.getAcceleration(&ax, &ay, &az);
    //accelgyro.getRotation(&gx, &gy, &gz);

    #ifdef OUTPUT_READABLE_ACCELGYRO
        // display tab-separated accel/gyro x/y/z values
        Serial.print("a/g:\t");
        Serial.print(ax); Serial.print("\t");
        Serial.print(ay); Serial.print("\t");
        Serial.print(az); Serial.print("\t");
        Serial.print(gx); Serial.print("\t");
        Serial.print(gy); Serial.print("\t");
        Serial.println(gz);
    #endif

    #ifdef OUTPUT_BINARY_ACCELGYRO
        Serial.write((uint8_t)(ax >> 8)); Serial.write((uint8_t)(ax & 0xFF));
        Serial.write((uint8_t)(ay >> 8)); Serial.write((uint8_t)(ay & 0xFF));
        Serial.write((uint8_t)(az >> 8)); Serial.write((uint8_t)(az & 0xFF));
        Serial.write((uint8_t)(gx >> 8)); Serial.write((uint8_t)(gx & 0xFF));
        Serial.write((uint8_t)(gy >> 8)); Serial.write((uint8_t)(gy & 0xFF));
        Serial.write((uint8_t)(gz >> 8)); Serial.write((uint8_t)(gz & 0xFF));
    #endif

    // blink LED to indicate activity
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
}
```

Figure 31The example of MPU6050 test in Arduino

3.4.2.2.3 RPLiDAR

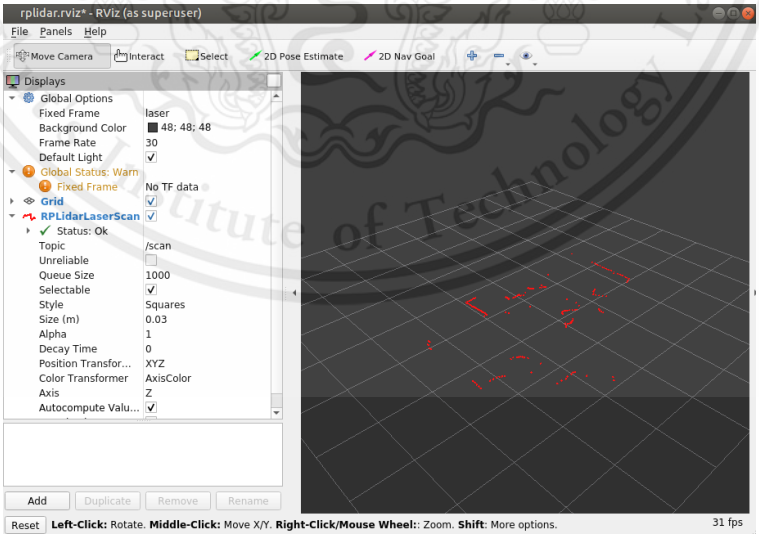

Figure 32 RPLiDAR scan topic from rviz monitor

This step verifies that the RPLiDAR can detect the environment. The red line, as shown on the figure 32, means the laser in the RPLiDAR works well because it can echo its surroundings and gather data to visualize as a red quick map.

3.4.2.3 Calibration

3.4.2.3.1 IMU Calibration

IMU calibration is a step to correct the bias of the MPU6050. In order to improve the robot movement, the robot needs to calibrate the MPU6050 because this sensor measure velocity and acceleration.

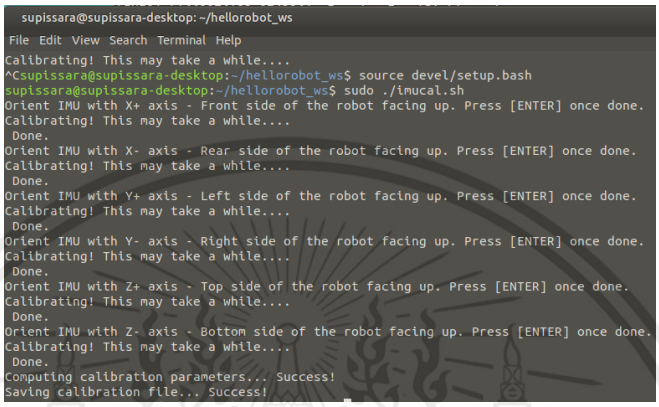

Figure 33 The procedure of IMU calibration

When the calibration process is complete, the set of IMU bias must be applied to the ROS system to improve the transport precision of the robot.

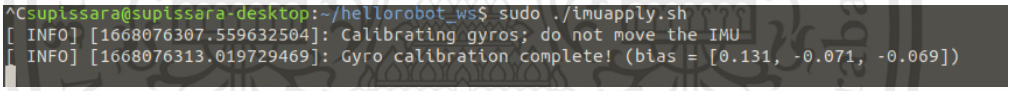

Figure 34 The procedure of applying IMU bias

3.4.2.3.2 Linear Calibration

Lastly, the movement of the robot is verified when the linear calibration is accurate. According to the figure 29, the calibration process can be set for a specific testing distance and the speed of the robot.

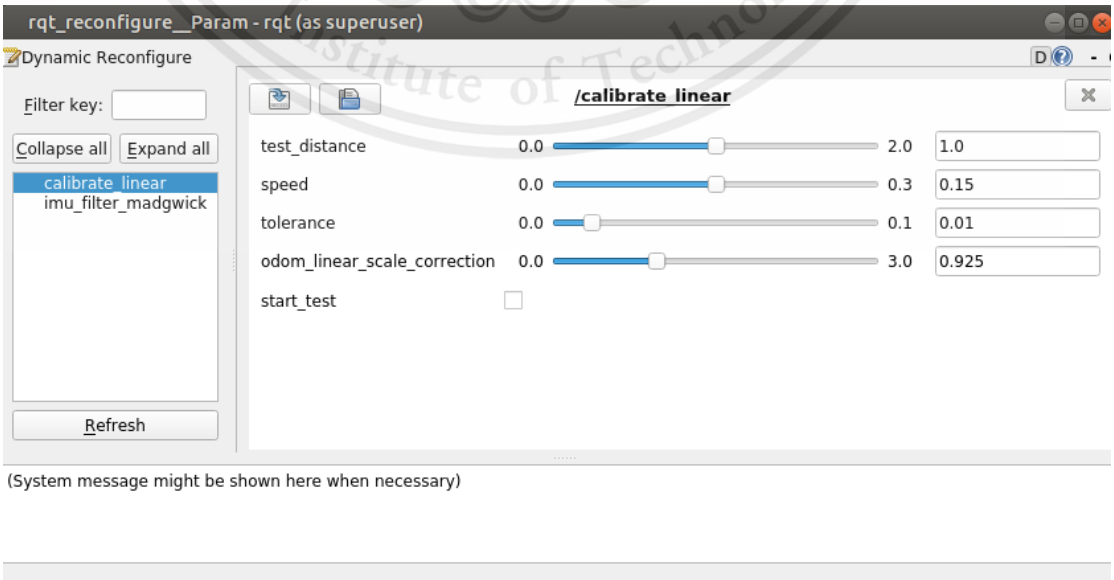

Figure 35 The monitor of linear calibration

3.4.2.4 G-Mapping

Mapping will be performed to collect the environment and generate a map of the robot's current location.

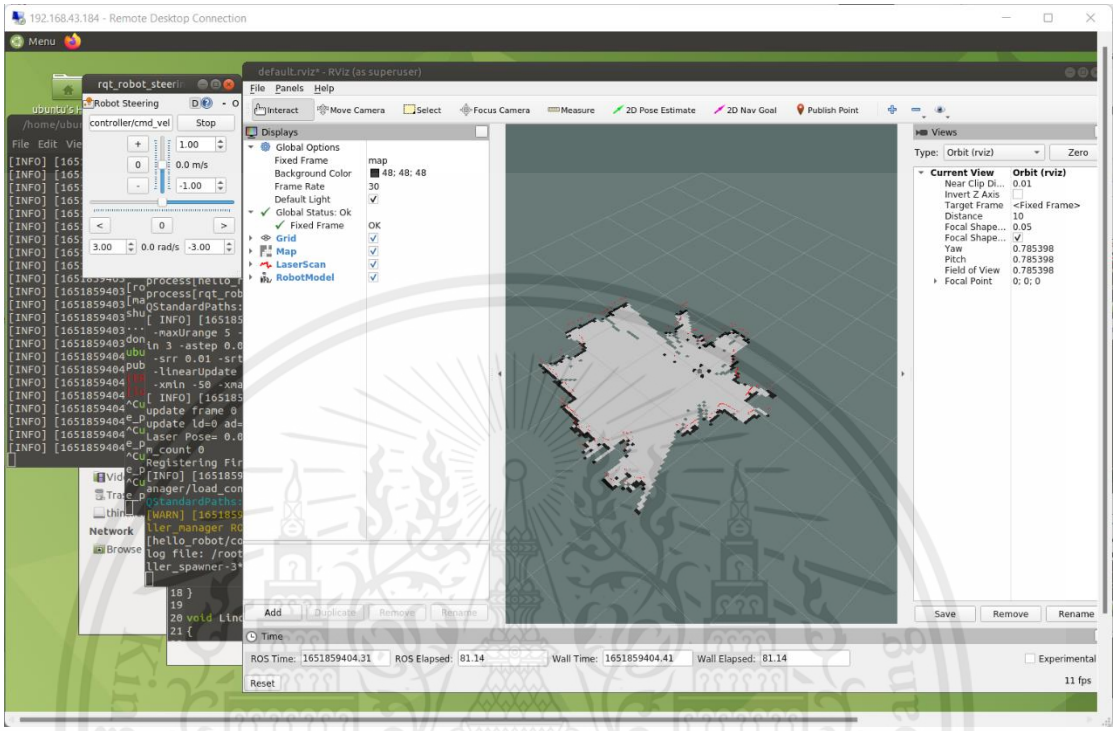

Figure 36 G-mapping in rviz

CHAPTER 4
EXPERIMENTAL RESULTS

4.1 Introduction
There are two distinct robot models, designated as A and B. The experiment illustrates the outcomes of circuit design, component inspections, performance tests, linear calibration test, and comparison summary for two distinct robot models.

4.2 Hardware design
This topic illustrates the exterior and interior of two robot models, including their appearances and internal circuits.

4.2.1 Model A
The appearance of the model A shows in figure 34 and the circuit shows in figure 35 and the dimension of the robot is 20x50x10 centimeters (widthxlengthxheight).

Figure 37 Appearance of the model A

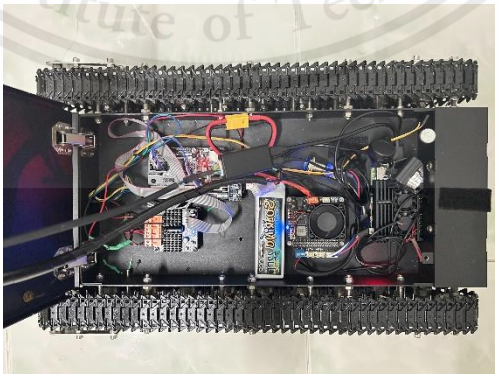

Figure 38 Circuit of the model A

4.2.2 Model B
The appearance of the model B shows in figure 36 and the circuit shows in figure 37 and the dimension of the robot is 26.5x31.5x16 centimeters (widthxlengthxheight).

Figure 39 Appearance of the model B

Figure 40 Circuit of the model B

4.3 Component inspections

4.3.1 Model A

4.3.1.1 Motor and Encoder

Two-wheel drive, or 2WD, is the type of motor utilized by model A. Due to significant friction in the robot's two motors, the robot cannot move in a straight line or turn as instructed. Thus, this can impede the precision with which a robot can move.

4.3.1.2 IMU

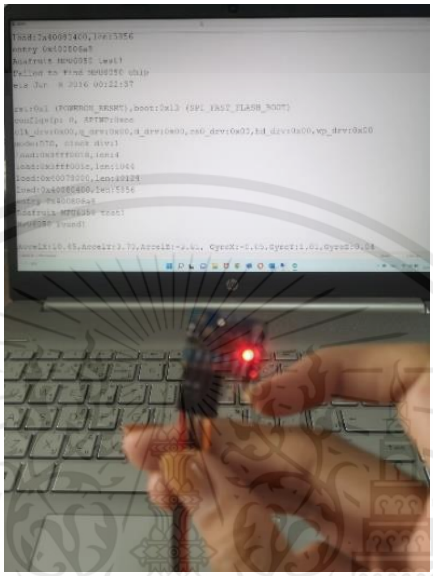

Figure 41 The result of IMU sensor

IMU sensor can be used as the light shows in figure 41. Also, it can read the pitch roll, and yaw values accurately.

4.3.1.3 RPLiDAR

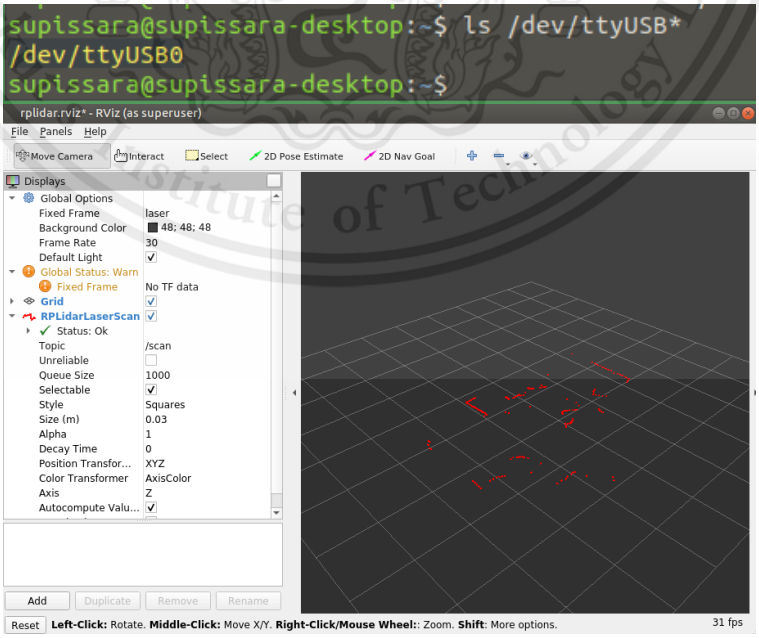

Figure 42 RPlLiDAR in Rviz

Figure 42's red line indicates that RPLiDAR's performance is ready for use.

4.3.2 Model B

4.3.2.1 Motor and Encoder

The motor utilized by the model A is a 4WD, or four-wheel drive, motor. The motors can move forward in a straight line or turn depending on the order.

4.3.2.2 IMU

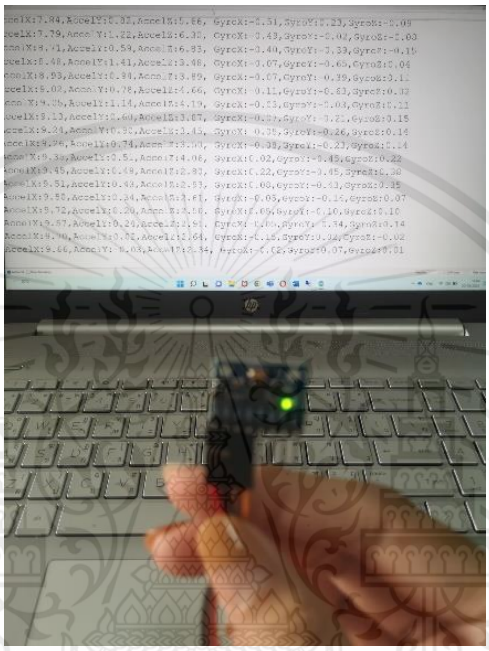

Figure 43 The result of IMU sensor

The IMU sensor can be utilized as shown in figure 43. Additionally, it can accurately read the pitch, roll, and yaw values.

4.3.2.3 RPLiDAR

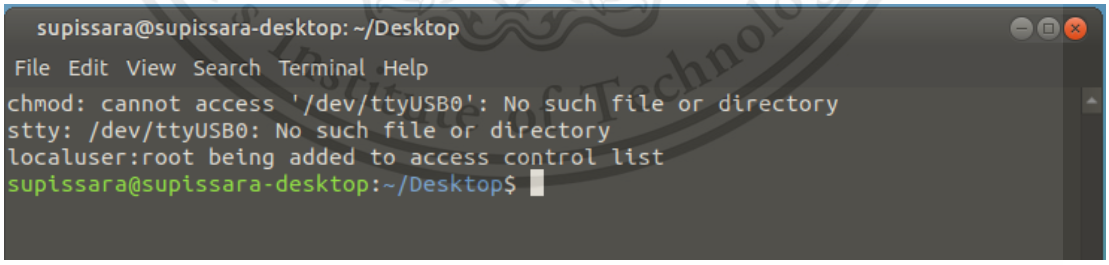

Figure 44 RPLiDAR was not found by the Rarpberryp Pi

Figure 44 shows that the RPLiDAR is not ready to be used due to the disappearance of the RPLiDAR signal from the Raspberry Pi.

4.3.2.4 Ultrasonic

```
supissara@supissara-desktop:/dev$ chmod og+rwx gpio*
chmod: changing permissions of 'gpiochip0': Operation not permitted
chmod: changing permissions of 'gpiochip1': Operation not permitted
chmod: changing permissions of 'gpiomem': Operation not permitted
supissara@supissara-desktop:/dev$ su - root
Password:
su: Authentication failure
supissara@supissara-desktop:/dev$ su - root
Password:
su: Authentication failure
supissara@supissara-desktop:/dev$ chmod o+w root
chmod: cannot access 'root': No such file or directory
supissara@supissara-desktop:/dev$ cd
supissara@supissara-desktop:~$ su - root
Password:
su: Authentication failure
supissara@supissara-desktop:~$
```

Figure 45 Ultrasonic is not permitted

Figure 45 shows that the ultrasonic is not ready to be used due to the unpermitted ultrasonic from the Raspberry Pi.

4.4 Performance tests

4.4.1 Text remote connection (PuTTy)

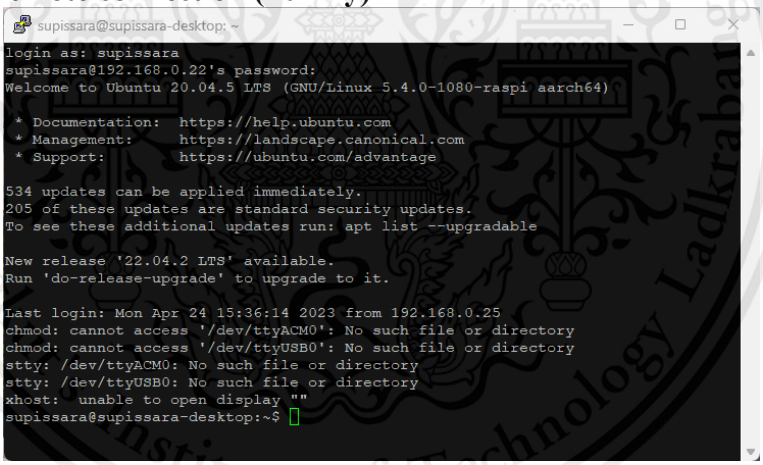

Figure 46 The command window of PuTTy

Both model A and model B can be accessed remotely by PuTTy for text commands with the same result according to the same raspberry Pi and memory.

4.4.2 Visual remote connection (Remote desktop)

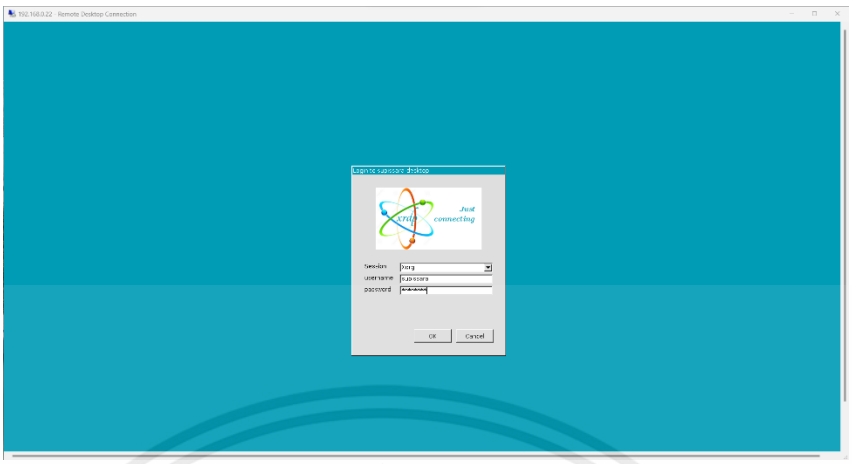

Figure 47 The command window of Remote desktop

Both model A and model B can be accessed remotely by PuTTy for visual command with the same result according to the same raspberry Pi and memory.

4.4.3 Robot model in Rviz world

4.4.3.1 Model A

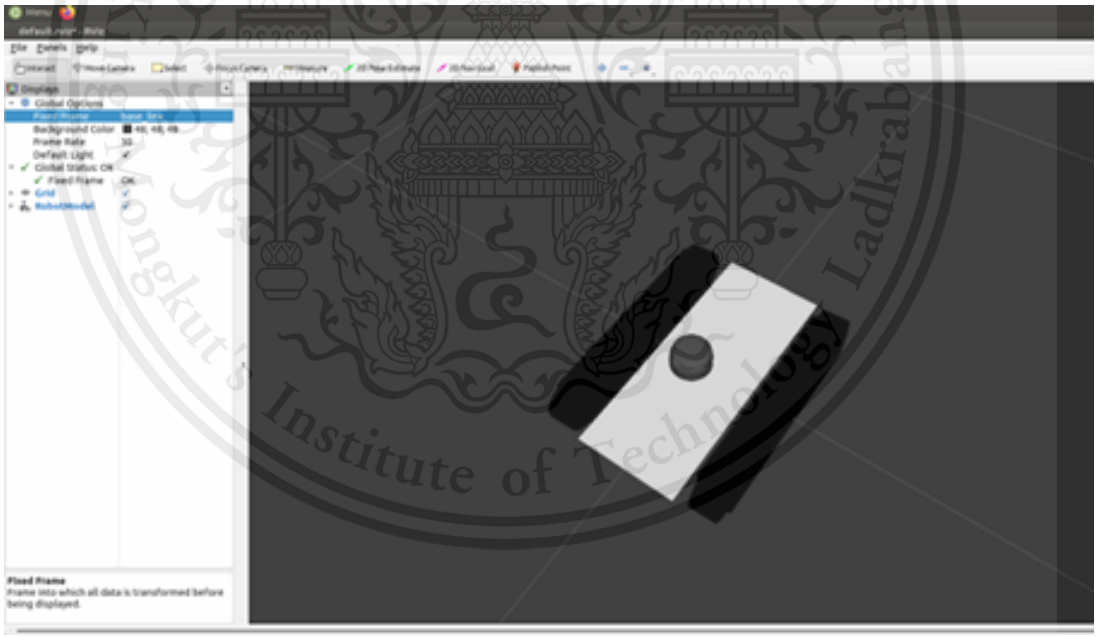

Figure 48 URDF robot model of the model A in rviz world

4.4.3.2 Model B

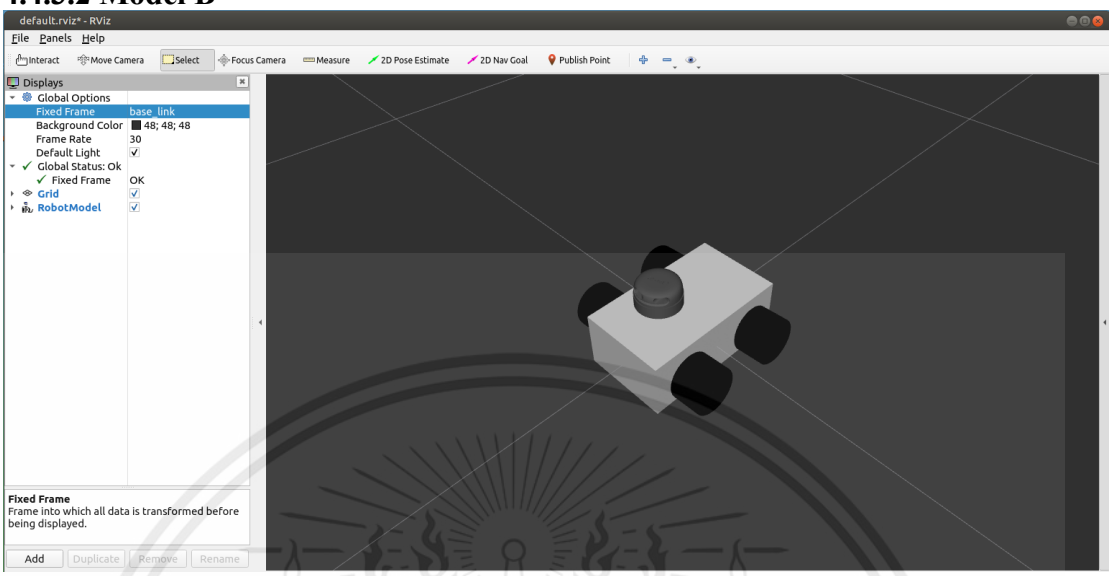

Figure 49 URDF robot model of the robot B in rviz world

4.4.4 G-mapping

4.4.5 Model A

Model A cannot perform the mapping because of wheel friction and motor problems.

4.4.6 Model B

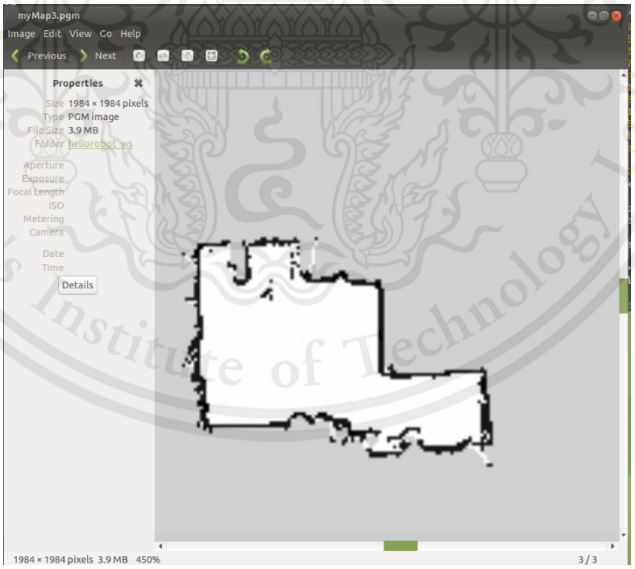

Figure 50 myMap3 from G-mapping

Figure 40 shows the results of myMap3, the map that was collected from the G-mapping process by the robot model B.

4.5 Linear calibration tests

4.5.1 Model A

Model A cannot perform the mapping because of wheel friction and motor problems.

4.5.2 Model B

Table 4.5.2.1: Results of linear calibration

Experimental Round	Distance (centimeters)		
	100	200	300
1	95	197	290
2	98	202	295
3	102	201	304
4	100	195	306
5	104	189	292
Average	99.8	196.8	297.4

From the table, the experiment was attempted five times for 100 centimeters, 200 centimeters, and 300 centimeters. And the average results are 99.8 centimeters, 196.8 centimeters, and 297.4 centimeters, respectively.

4.6 Comparison summary

Table 4.6.1: Comparison summary between model A and model B

Orders	***Model A***	***Model B***
		
Component inspections		
dimension		26.5x31.5x16 cm
Motor type	2WD	4WD
Motor test	Failed	Passed
IMU test	Passed	Passed
RPLiDAR test	Passed	Failed
Ultrasonic test	N/A	Falied
Performance test		
PuTTy (text)	Passed	Passed
Remote Desktop (visual)	Passed	Passed
Robot model	Passed	Passed
Mapping	Falied	Passed
Linear Calibration	Falied	Passed

CHAPTER 5
CONCLUSION AND DISCUSSION

5.1 Conclusion

According to the chapter 4, this project has presented a comparison between two different robot models, Model A and Model B, in terms of their performance, features, and applications.

The research has demonstrated that both models have their unique strengths and weaknesses Model A excels in a variety of ways that allow it to run well on rough terrain; the wheel is attached to the body in a prefabricated manner, resulting in a high percentage of wheel accuracy. It has ample space for easily organizing the circuit. And the material allows it to absorb more shock. While Model B outperforms Model A in Lighter weight, lighter sound, the materials used make it easy to modify, four-wheel drive makes it easy to turn around. and grip the road better than 2-wheel drive (suitable for outdoor), move faster, uses less power to drive, and finally, the wheels are not contain friction.

However, Model A is the wheels are only driven by two 2WD motors, which may not be adequate for the size of the vehicle, make a wide turn, the raft wheels are challenging to replace, and the wheel has a high friction. Because the gear is utilized to rotate the wheel's raft. Therefore, it is extremely power-hungry, difficult to modify, loud, and slow.and for the Model B you must install the wheel yourself, which may result in a less-than-ideal wheel position, less impact resistance due to the thin material, and a mirror.

In conclusion, this project has provided valuable insights into the comparison of two different robot models and can serve as a useful reference for researchers, engineers, and practitioners who are involved in the design and development of robots for various applications.

5.2 Discussion

Since the purpose of this study is to improve an outdoor SLAM robot, it is hypothesized that the calibration method will increase the outdoor SLAM robot's accuracy. In addition, port forwarding is hypothesized to enable users to utilize, control, and communicate with the robot via the internet from any location. This project achieves online communication with the robot. However, the robot must acquire greater accuracy and precision to be used in an outdoor environment. To improve the sensing accuracy, the robot should be equipped with a second eye or an ultrasonic sensor to enhance the surrounding environment.

REFERENCES

[1] D.-W. Hugh and B. Tim, “Simultaneous localization and mapping: part I,” *IEEE Robotics & Automation Magazine,* vol. XIII, no. 2, pp. 99-110, June 2006.

[2] M. Helmke, Ubuntu Linux Unleashed 2021 Edition, 14th ed., Addison-Wesley Professional, 2021.

[3] 2019 Canonical Ltd., “Ubuntu,” 2019. [Online]. Available: https://design.ubuntu.com/brand/ubuntu-logo/. [Accessed 25 October 2022].

[4] Debian, “Debian,” 1999. [Online]. Available: https://www.debian.org/logos/. [Accessed 25 October 2022].

[5] B. Ward, How Linux Works, 3rd ed., No Strach Press, 2021.

[6] A. Silberschatz, P. B. Galvin and G. Gagn, Operating System Concepts, 8th ed., John Wiley & Sons, 2008.

[7] 2019 Canonical Ltd., “Ubuntu,” 2019. [Online]. Available: https://design.ubuntu.com/brand/canonical-logo/. [Accessed 25 October 2022].

[8] L. Joseph and A. Johny, Robot Operating System (ROS) for Absolute Beginners: Robotics Programming Made Easy, 2nd ed., Apress, 2022.

[9] Giorgosarv18, “wikimedia,” 15 July 2020. [Online]. Available: https://commons.wikimedia.org/wiki/File:Desktop_Ubuntu_20.04.png. [Accessed 25 October 2022].

[10] L. Joseph and J. Cacace, Mastering ROS for Robotics Programming, 3rd ed., Packt Publishing, 2021.

[11] Creative Commons, "ROS," 31 August 2021. [Online]. Available: http://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot. [Accessed 5 March 2023].

[12] W. Meeussen, "ROS," 27 October 2010. [Online]. Available: https://www.ros.org/reps/rep-0105.html#coordinate-frames. [Accessed 5 March 2023].

[13] A. Sear-Collins, "Automatic Addison," 16 June 2021. [Online]. Available: https://automaticaddison.com/coordinate-frames-and-transforms-for-ros-based-mobile-robots/. [Accessed 5 March 2023].

[14] Creative Commons, "ROS," 2 October 2017. [Online]. Available: http://wiki.ros.org/tf. [Accessed 5 March 2023].

[15] T. Foote, "tf: The transform library," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 2013.

[16] PlatformIO, “PlatformIO,” 2014. [Online]. Available: https://docs.platformio.org/en/latest/what-is-platformio.html. [Accessed 30 October 2022].

[17] N. Dundar, Arduino Software Internals: A Complete Guide to How Your Arduino Language and Hardware Work Together, Apress, 2020.

[18] A. Pajankar, Raspberry Pi Supercomputing and Scientific Programming : MPI4PY, NumPy, and SciPy for Enthusiasts, Apress, 2017.

[19] Red Hat Inccorporated, “Opensource,” 2022. [Online]. Available: https://opensource.com/resources/raspberry-pi. [Accessed 31 October 2022].

[20] ElectronincWings, “ElectronicWings,” 2022. [Online]. Available: https://www.electronicwings.com/raspberry-pi/raspberry-pi-introduction. [Accessed 31 October 2022].

[21] Flyability, “Flyability,” 2022. [Online]. Available: https://www.flyability.com/simultaneous-localization-and-mapping. [Accessed 14 October 2022].

[22] Raspberry Pi Trading Limited Liability Company, "RaspberryPi," June 2019. [Online]. Available: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf. [Accessed 1 November 2022].

[23] Raspberry Pi Trading Limited Liability Company, “RaspberryPi,” [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. [Accessed 31 October 2022].

[24] Raspberry Pi Trading Limited Liability Company, “RaspberryPi,” June 2019. [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf. [Accessed 1 November 2022].

[25] A. Shaw, “The Engineering Projects,” 11 March 2021. [Online]. Available: https://www.theengineeringprojects.com/2021/03/what-is-raspberry-pi-4-pinout-specs-projects-datasheet.html. [Accessed 3 Novenber 2022].

[26] Shanghai Slametic Limited Company, “SLAMTEC,” 28 April 2016. [Online]. Available: https://www.slamtec.com/en/Lidar/A2. [Accessed 3 November 2022].

[27] Shanghai Slamtec Company Limited, “Generation Robots,” 28 April 22016. [Online]. Available: https://www.generationrobots.com/media/robopeak_2d_lidar_brief_en_A2M4.pdf. [Accessed 3 Novenber 2022].

[28] S. Roy and C. K. Sarkar, MEMS and Nanotechnology for Gas Sensors, CRC Press, 2015.

[29] D. Vaish, Python Robotics Projects, Packt Publishing, 2018.

[30] Z. M. Naing, S. Anatolii, H. S. Paing and L. V. Thang, “Evaluation of Microelectromechanical System Gyroscope and Accelerometer in the Object Orientation System Using Complementary Filter,” in *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, 2021.

[31] LastMinuteEngineers.com, “Last Minute Engineers,” [Online]. Available: https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/. [Accessed 4 November 2022].

[32] R. Portales, Android Things Quick Start Guide, Packt Publishing, 2018.

[33] ElectronicWings, “ElectronicWings,” 2022. [Online]. Available: https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module. [Accessed 4 November 2022].

[34] Hyperion, “Hyperion-world,” [Online]. Available: https://www.robotshop.com/media/files/pdf/hyperion-g5-50c-3s-1100mah-lipo-battery-User-Guide.pdf. [Accessed 4 November 2022].

[35] SunpadowBattery, “Sunpadow,” 2022. [Online]. Available: https://www.sunpadowmall.com/products/sunpadow-11-1v-3s-rc-lipo-battery-60c-2250mah-with-xt60-plug-for-rc-airplane-quadcopter-helicopter-drone-fpv-racing-hobby?pr_prod_strat=use_description&pr_rec_id=d0a68b18c&pr_rec_pid=6142876909727&pr_ref_pid=614287681. [Accessed 4 November 2022].

[36] ROS, “ROS,” 20 May 2022. [Online]. Available: https://wiki.ros.org/. [Accessed 12 October 2022].