HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY OF INTERNATIONAL EDUCATION

**GRADUATION PROJECT**

# RESEARCH ON INDOOR AUTONOMOUS MOBILE ROBOT SYSTEMS USING 2D LIDAR

**VU XUAN HIEP**

**Student ID**: **20145413**

**NGUYEN CHI THANH**

**Student ID**: **20145435**

**Major**: **AUTOMOTIVE ENGINEERING TECHNOLOGY**

**Supervisor**: **CAI VIET ANH DUNG, PhD.**

Ho Chi Minh City, DECEMBER 2024

**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF INTERNATIONAL EDUCATION**

HCMUTE

**GRADUATION PROJECT**

# RESEARCH ON INDOOR AUTONOMOUS MOBILE ROBOT USING 2D LIDAR

**VU XUAN HIEP**

**Student ID**: **20145413**

**NGUYEN CHI THANH**

**Student ID**: **20145435**

**Major**: **AUTOMOTIVE ENGINEERING TECHNOLOGY**

**Supervisor**: **CAI VIET ANH DUNG, PhD.**

Ho Chi Minh City, DECEMBER 2024

HOCHIMINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION
fie.hcmute.edu.vn

THE SOCIALIST REPUBLIC OF VIETNAM
**Independence – Freedom– Happiness**
--------

*Ho Chi Minh City, December 23, 2024*

# GRADUATION PROJECT ASSIGNMENT

| | |
|---|---|
| Student name: Vu Xuan Hiep | Student ID: 20145413 |
| Student name: Nguyen Chi Thanh | Student ID: 20145435 |
| Major: Automotive Engineering Technology | Class: 20145CLA |
| Supervisor: Cai Viet Anh Dung | Phone number: ________________ |
| Date of assignment: 15/09/2024 | Date of submission: 22/12/2024 |

1. Project title: RESEARCH ON INDOOR AUTONOMOUS MOBILE ROBOT SYSTEMS USING 2D LIDAR

2. Initial materials provided by supervisor: Computer, 2D LIDAR

3. Content of the project: Research on SLAM algorithm using 2D LIDAR.

4. Final product: Algorhithm for robot’s autonomous navigation

| | |
|---|---|
| **CHAIR OF THE PROGRAM** | **SUPERVISOR** |
| *(Sign with full name)* | *(Sign with full name)* |

HOCHIMINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION
fie.hcmute.edu.vn

THE SOCIALIST REPUBLIC OF VIETNAM
**Independence – Freedom– Happiness**
**--------**
**----**

# SUPERVISOR'S EVALUATION SHEET

| | |
|---|---|
| Student name:Vu Xuan Hiep...................................... | Student ID: 20145413................ |
| Student name: Nguyen Chi Thanh.................................. | Student ID: 20145435................ |

Major: ........................................................................................
Project title:.........................................................................................................
Supervisor: Cai Viet Anh Dung, Ph.D

**EVALUATION**

1. Content of the project:
   ..........................................................................................................................
   ..........................................................................................................................
   ..........................................................................................................................
2. Strengths:
   ..........................................................................................................................
   ..........................................................................................................................
   ..........................................................................................................................
3. Weaknesses:
   ..........................................................................................................................
   ..........................................................................................................................
   ..........................................................................................................................
4. Approval for oral defense? *(Approved or denied)*
   ..........................................................................................................................

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

..........................................................................................................................

6. Mark:……………….(*in words*:..........................................................................)

*Ho Chi Minh City, December , 2024*

**SUPERVISOR**
*(Sign with full name)*

HOCHIMINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION
fie.hcmute.edu.vn

THE SOCIALIST REPUBLIC OF VIETNAM
**Independence – Freedom– Happiness**
--------

# PRE-DEFENSE EVALUATION SHEET

Student name:Vu Xuan Hiep...................................... Student ID: 20145413

Student name: Nguyen Chi Thanh.................................. Student ID: 20145435

Major: ..........................................................................................
Project title:.........................................................................................................
Supervisor: Cai Viet Anh Dung, Ph.D
Name of Examiner: .............................................................................................................

**EVALUATION**

1. Content and workload of the project

...................................................................................................................................
...................................................................................................................................
...................................................................................................................................
...................................................................................................................................

2. Strengths:

...................................................................................................................................
...................................................................................................................................
...................................................................................................................................

3. Weaknesses:

...................................................................................................................................
...................................................................................................................................
...................................................................................................................................

4. Approval for oral defense? *(Approved or denied)*

...................................................................................................................................

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

...................................................................................................................................

6. Mark:……………….(*in words*:..........................................................................)

*Ho Chi Minh City, December , 2024*

**EXAMINER**
*(Sign with full name)*

HOCHIMINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY OF INTERNATIONAL EDUCATION
fie.hcmute.edu.vn

THE SOCIALIST REPUBLIC OF VIETNAM
**Independence – Freedom– Happiness**
--------

# EVALUATION SHEET OF DEFENSE COMMITTEE MEMBER

Student name:Vu Xuan Hiep...................................... Student ID: 20145413................

Student name: Nguyen Chi Thanh.................................. Student ID: 20145435................

Major: ..............................................................................................
Project title:...........................................................................................................
Supervisor: Cai Viet Anh Dung, Ph.D
Name of Defense Committee Member:
..........................................................................................................................................

**EVALUATION**

1. Content and workload of the project

............................................................................................................................................
............................................................................................................................................
............................................................................................................................................
............................................................................................................................................

2. Strengths:

............................................................................................................................................
............................................................................................................................................
............................................................................................................................................

3. Weaknesses:

............................................................................................................................................
............................................................................................................................................
............................................................................................................................................

4. Overall evaluation: *(Excellent, Good, Fair, Poor)*

............................................................................................................................................

5. Mark: ……………… (*in words*:.........................................................................................)

*Ho Chi Minh City, , 2025*

**COMMITTEE MEMBER**
*(Sign with full name)*

## Disclaimer

## Acknowledgements

This Graduation Thesis Report on the topic **“RESEARCH ON INDOOR AUTONOMOUS ROBOT SYSTEMS USING 2D LIDAR”** signifies the culmination of our team’s extensive research endeavors and technical commitment. Its completion has been made possible through the invaluable mentorship and expertise provided by the faculty at the **Automotive Engineering Technology Department** and **Mechanical Engineering Department**, Ho Chi Minh University of Technology and Education.

Firstly, we wish to convey our profound gratitude to our principal supervisor, **Dr. Cai Viet Anh Dung**, for his steadfast support, expert insights, and meticulous guidance, which played a pivotal role in bringing this project to fruition. We also wish to express heartfelt thanks to our friends and colleagues at the **RAI Lab**, who generously shared their knowledge and provided assistance during the project. Their encouragement and expertise significantly contributed to our learning and progress.

Finally, we express our sincere appreciation to our families for their unwavering encouragement, valuable advice, and enduring support throughout the entire development of this thesis.

# Table of Contents

## Abbreviations

| **Abbreviation** | **Full Term** |
|---|---|
| 2D LiDAR | Two-Dimensional Light Detection and Ranging |
| SLAM | Simultaneous Localization and Mapping |
| ROS | Robot Operating System |
| IMU | Inertial Measurement Unit |
| FOV | Field of View |
| A* | A-star (Pathfinding Algorithm) |
| RRT* | Rapidly-exploring Random Tree Star |
| D* | Dynamic A* |
| LIDAR | Light Detection and Ranging |
| RTK | Real-Time Kinematic |
| EKF | Extended Kalman Filter |
| VSLAM | Visual Simultaneous Localization and Mapping |
| UAV | Unmanned Aerial Vehicle |
| GPS | Global Positioning System |
| UWB | Ultra-Wideband |
| GPU | Graphics Processing Unit |
| CPU | Central Processing Unit |
| MCU | Microcontroller Unit |

| | |
|---|---|
| GPIO | General Purpose Input/Output |
| ADC | Analog-to-Digital Converter |
| DAC | Digital-to-Analog Converter |
| PWM | Pulse Width Modulation |
| PCB | Printed Circuit Board |
| VRM | Voltage Regulator Module |
| IC | Integrated Circuit |
| MOSFET | Metal-Oxide-Semiconductor Field-Effect Transistor |
| I2C | Inter-Integrated Circuit |
| SPI | Serial Peripheral Interface |
| UART | Universal Asynchronous Receiver-Transmitter |
| LED | Light Emitting Diode |
| VCC | Voltage at the Common Collector |
| GND | Ground |
| VDD | Voltage at the Drain or Source |
| RTC | Real-Time Clock |
| SMPS | Switched-Mode Power Supply |
| LDO | Low Dropout Regulator |
| R/C | Resistor/Capacitor |

**List of figures**

## List of Tables

## Abstract

This thesis explores the design and implementation of an indoor autonomous robot using a 2D LiDAR sensor for navigation and mapping in dynamic environments. The primary objective of the research is to develop an efficient localization and mapping strategy for the robot, utilizing Simultaneous Localization and Mapping (SLAM) algorithms. By integrating the 2D LiDAR with a microcontroller and real-time data processing, the robot is able to generate an accurate map of its surroundings while continuously updating its position within that map.

The research investigates the performance of various SLAM techniques in indoor environments, focusing on optimizing the robot's ability to avoid obstacles and navigate autonomously. Key challenges addressed include improving sensor accuracy, handling sensor noise, and minimizing computational load. Additionally, the study evaluates the integration of path planning algorithms, such as A* and RANSAC, to ensure efficient movement and collision-free navigation. The results demonstrate that the proposed system significantly improves the robot's navigation precision and responsiveness, with a smoother and more reliable performance in cluttered indoor environments.

Keywords: Autonomous Robot, 2D LiDAR, SLAM, Path Planning, Obstacle Avoidance, Localization

# Chapter 1: INTRODUCTION

## 1.1 Background

### 1.1.1 Overview

Development Trends:

**Improved Automation**: Recent research emphasizes advancing robot automation to minimize human involvement and achieve higher precision in task execution. These improvements are designed to boost the productivity of industrial processes and machinery.

**Broad Applications**: With the rapid evolution of the industrial revolution, autonomous robots are increasingly essential, not only in manufacturing but also in sectors such as agriculture, transportation, and logistics.

Sensors and Technology:

**Sensor Technologies**: Devices like 2D and 3D LiDAR are instrumental for mapping and avoiding obstacles, while IMUs and encoders facilitate simultaneous localization and odometry.

**Integration of Multiple Sensors**: Emerging trends focus on combining different types of sensors to enhance detection accuracy and environmental awareness. Incorporating sensors such as RGB-D cameras can improve environmental understanding and landmark recognition, leading to more precise localization. Additionally, GPS and Wi-Fi sensors offer enhanced positioning capabilities.

Localization and Mapping:

**Advanced SLAM Techniques**: Research has yielded SLAM methodologies that leverage deep learning to achieve superior object recognition and localization in complex scenarios.

Planning and Control:

**Innovative Planning Algorithms**: Efforts are concentrated on developing dynamic algorithms like RRT* and D* to optimize path planning in challenging environments.

**Adaptive Control Mechanisms**: Self-learning approaches, including reinforcement learning, are being implemented to refine robotic movement and decision-making.

### 1.1.2. Research progress in domestic.

In Vietnam, there has been growing research on autonomous robots, especially in the context of 2D LiDAR technology for navigation, mapping, and path planning. Several studies have made significant strides in this domain, particularly focusing on the integration of 2D LiDAR with other sensors and algorithms to enhance robot autonomy.

**Nguyễn Hải Anh (2023)** explored the use of 2D LiDAR for autonomous vehicles in e-commerce warehouses. The study aimed to optimize navigation within confined spaces using 2D LiDAR sensors to map the environment. The research highlighted the importance of integrating 2D LiDAR with localization algorithms to ensure accurate positioning and obstacle avoidance in dynamic warehouse environments. This approach significantly improved the operational efficiency of autonomous robots, reducing errors and improving path planning.

**Trần Quốc Tuấn and Nguyễn Thành Tâm (2020)** developed path planning algorithms for mobile robots, with a particular focus on the application of 2D LiDAR data for navigation in indoor environments. They optimized algorithms to process the 2D LiDAR data efficiently, addressing challenges such as local minima and deadlock avoidance. This research contributed to improving the overall navigation capabilities of mobile robots, ensuring safer and more reliable movement in environments with complex layouts.

**Phạm Thị Lan Hương and Nguyễn Thị Duyên (2023)** examined the use of 2D LiDAR for local path planning on ROS platforms, specifically in agricultural settings such as greenhouses. Their study focused on using 2D LiDAR to build accurate local maps, which helped to guide autonomous robots through narrow aisles and obstacles within the agricultural environment. The research provided insights into how local planners can be optimized using LiDAR-based data to ensure safe navigation and effective task completion in real-world scenarios.

**Trần Trọng Tiến et al. (2022)** investigated the integration of 2D LiDAR with SLAM techniques for mobile robots in confined indoor environments. While their work primarily focused on 3D SLAM, it also highlighted the significant role of 2D LiDAR in the initial stages of mapping and localization. Their work demonstrated how 2D LiDAR could provide real-time data for creating precise 2D maps, which served as a foundation for more complex 3D mapping in cluttered and dynamic spaces.

**Phan Hoàng Anh et al. (2020)** provided a comprehensive analysis of simultaneous localization and mapping (SLAM) on ROS platforms, with a focus on the use of 2D LiDAR. The study showed how 2D LiDAR can be integrated into ROS-based systems to achieve accurate and efficient SLAM. Their research emphasized the importance of data fusion techniques, combining 2D LiDAR with other sensor data, to improve the overall performance and reliability of autonomous robots in real-world applications.

These studies highlight the increasing importance of 2D LiDAR in the development of autonomous robots, especially in the context of navigation, mapping, and path planning. The integration of 2D LiDAR with advanced algorithms and ROS platforms is central to improving the efficiency and safety of autonomous systems in a variety of applications, from warehouses to agriculture.

### 1.1.3. Research progress in foreign

Recent advancements in the field of autonomous robots, particularly those utilizing 2D LiDAR sensors for navigation and path planning, have resulted in significant contributions to the development of robust, real-time systems. These contributions encompass various aspects, including navigation algorithms, SLAM technology, and the integration of 2D LiDAR with dynamic environments. Below is a broad review of the current research trends related to 2D LiDAR autonomous robots.

**Navigation Systems and Behavior Tree Integration** Steve Macenski et al. (2020) in their work "The Marathon 2: Navigation System" (Open Navigation LLC) focused on enhancing the navigation capabilities of autonomous robots using behavior trees and integrated planning components. The research incorporated a **global planner**, **local planner**, and **recovery server** to manage different levels of navigation tasks. In particular, the local planner uses real-time sensor data, such as from 2D LiDAR, to help the robot navigate through its immediate environment, while the global planner takes care of higher-level path planning. The **recovery server** is essential in dynamic environments, ensuring that the robot can recover from navigation failures. The integration of behavior trees helps optimize the decision-making process in dynamic conditions, making the navigation system both adaptable and robust.

**ROS2 Framework for Autonomous Navigation** In "Robot Operating System 2: Design, Architecture, and Uses in the Wild" (Macenski et al., 2022), the authors emphasized the significance of the **ROS2** framework in the development of autonomous robots. ROS2 provides a highly modular environment where 2D LiDAR can be seamlessly integrated into various components of the robot's control system. It facilitates the development of navigation stacks, where 2D LiDAR data is used in combination with **local planners** and **global planners** to improve decision-making and path execution. ROS2's **real-time capabilities** and support for distributed computing enable better synchronization between sensors, actuators, and planning algorithms, making it an ideal choice for real-world applications in dynamic environments.

**SLAM Algorithms and LiDAR Integration** In their 2021 study, Steve Macenski and Ivona Jambrecic (SLAM Toolbox: SLAM for the Dynamic World), discussed the integration of **SLAM (Simultaneous Localization and Mapping)** algorithms with dynamic sensor data, including 2D LiDAR. SLAM is essential for creating accurate maps of an environment, particularly when the environment is subject to change. The authors focused on the **dynamic world SLAM** problem, where robots using 2D LiDAR sensors continuously update their maps to reflect new obstacles and environment changes. The ability to use **LiDAR data** in dynamic environments provides significant advantages for autonomous robots, allowing them to navigate safely and efficiently while updating their spatial awareness in real-time.

**Kinematic Models and Dynamic Navigation** The study by Muhammad Umair Shafiq et al. (2024) on the **real-time navigation of a mecanum wheel-based mobile robot** in a dynamic environment discussed the application of both **forward kinematics** and **inverse kinematics** in navigation. This work focuses on the dynamics of a robot equipped with a **mecanum wheel drive system**, utilizing 2D LiDAR for obstacle detection and path planning. The paper presents methods for integrating kinematic models with real-time navigation, where 2D LiDAR plays a crucial role in perceiving the robot's surroundings and enabling precise motion control. Real-time feedback from LiDAR data allows the robot to adapt its path to avoid obstacles while following a pre-defined trajectory.

**Path Planning for Autonomous Ground Vehicles (AGVs)** Roman Fedorenko and Boris Gurenko (2016) in their study "Local and Global Motion Planning for Unmanned Surface Vehicle" focused on path planning algorithms for **Autonomous Ground Vehicles (AGVs)**. While this research was geared toward surface vehicles, the principles of **local motion planning** and **global path planning** are applicable to 2D LiDAR-based mobile robots. The study highlights the importance of real-time decision-making and how 2D LiDAR data aids in obstacle detection and avoidance, providing a dynamic response to environmental changes. By utilizing both **local** and **global motion planning**, AGVs can efficiently navigate complex environments, making this approach highly relevant for 2D LiDAR-based autonomous robots.

The integration of **2D LiDAR** with advanced **navigation algorithms**, **SLAM**, and **kinematic models** has enabled the development of highly efficient and reliable autonomous robots. Research across various domains, such as warehouses, agriculture, and unmanned surface vehicles, demonstrates the versatility and importance of 2D LiDAR in improving the autonomy, safety, and efficiency of mobile robots in real-world applications. These advancements will continue to drive the field forward, particularly as new technologies and frameworks like ROS2 emerge.

## 1.2. Research objectives

Research efficient Mapping Algorithms for analyzing and designing robust mapping algorithms tailored for autonomous mobile robots, ensuring accurate and scalable environment representation.

Investigate Simultaneous Localization and Mapping (SLAM) Techniques for exploring advanced SLAM methodologies and evaluating their performance in diverse operational scenarios, focusing on improving localization accuracy and real-time processing efficiency.

Integrate Multi-Sensor Systems for Enhanced Operational Capabilities for studying and implementing effective sensor integration strategies for real-time data fusion, optimizing the decision-making processes of autonomous mobile robots.

Examine Forward and Inverse Kinematics for Motion Controlling, theoretically analyzing and programmatically solving forward and inverse kinematics problems, aiming to improve the precision and adaptability of robot movements.

Research Path Planning Algorithms for investigating and developing advanced path planning algorithms that ensure efficient, obstacle-free navigation in dynamic and unstructured environments.

### 1.3. Research subjects

- Mecanum wheels
- IMU, encoder, Lidar sensor.
- Raspberry Pi 4B, ESP32
- ROS2
- Linux ubuntu

### 1.4. Research limitation

- Using open sources packages
- Budget for mechanical and electronic component
- PID controller
- Weak processor to run autonomous task
- Not having simulation works

### 1.5. Research methods

- Read and refer to theoretical and practical documents, research sources and robots that use the same technologies.
- Simulate and evaluate the use of Gazebo(possibly)
- Survey opensource codes of individuals and organizations with the same content as the topic
- Operate and test the autonomous vehicle robot system

Research scope:

- Environment: in a warehouse with walls to scan maps, with little human presence
- Using a small-sized mecanum-wheeled robot

### 1.6. Research contents

| No. | Main tasks and activities | Expected outcomes |
|---|---|---|

| | | |
|---|---|---|
| 1 | Collect necessary data (nav2, lidar, mapping, etc.). Draw a diagram of the workflow (odometry → mapping → navigation). | System diagram and project implementation plan |
| 2 | Research algorithms for robot control and map creation. | Map generated by the robot |
| 3 | Solve the autonomous navigation problem for the robot using the ROS platform:<br>- Odometry: motor encoder, IMU sensor.<br>- Mapping: lidar, odometry, mapping algorithm.<br>- Navigation:<br>• Input: map of environment, odometry, sensor data.<br>• Processor: local planner, global planner, recovery server, behavior tree navigation server.<br>• Output: command velocity → reverse kinematics → PWM output for each motor. | Robot can autonomously navigate to the final destination |
| 4 | Test and optimize the system. | Final project deliverable |
| 5 | Final report, presentation slides. | Final report, presentation slides. |

## 1.7. Research layout

**These are the overall chapters of the thesis:**

Chapter 1: Introduction

Chapter 2: Literature review

Chapter 3: Systems design

Chapter 4: Experiment results

Chapter 5: Conclusion and recommendations

# Chapter 2: LITERATURE REVIEW

## 2.1. Kinematics

When Mecanum wheels are actuated, the angled peripheral rollers translate a portion of the force in the rotational direction of the wheel to a force normal to the wheel direction. Depending on each individual wheel direction and velocity, the resulting combination of all these forces produce a total force vector in any desired direction thus allowing the platform to move freely in the direction of the resulting force vector, without changing of the wheels themselves

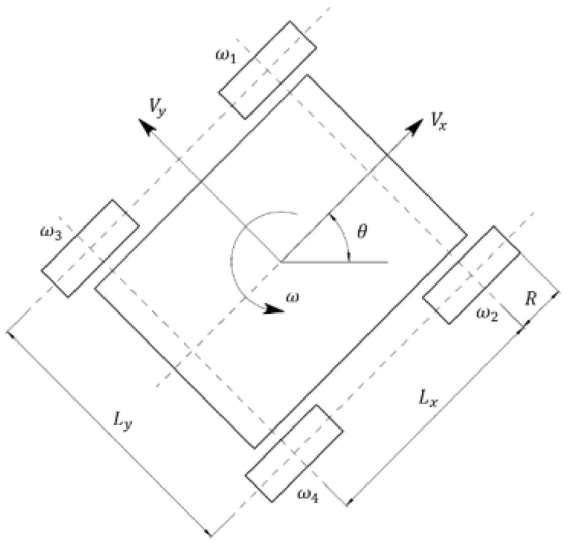


***Figure 2.1**: Calculation scheme of the mathematical model [23]*

Notation on the calculation scheme of the mathematical model as seen in [Eq. 2.1]:

- $v_x, v_y$ – mobile robot linear velocity
- $\omega_z$ – mobile robot angular velocity

- $\theta$ – mobile robot orientation (the yaw angle);
- $l_x + l_y$ – the platform overall dimensions (the distance between the geometric
- centers of the wheels);
- $\omega_1, \dots, \omega_4$– mobile robot wheels' angular velocities.

If we consider the frame attached to the robot chassis [Fig 2.1], we can write the body speed equations as follow:

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \frac{R}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{2}{l_x+l_y} & \frac{2}{l_x+l_y} & -\frac{2}{l_x+l_y} & \frac{2}{l_x+l_y} \end{bmatrix} * \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix}$$

*Equation 2.1 [23]*

Where: R is the wheel radius; $\omega_i$ is the angular velocity of the wheel i (i = 1…4); $l_1, l_2$ are the distances between wheel axis and body center.

Longitudinal Velocity:

$$v_x(t) = (\omega_1 + \omega_2 + \omega_3 + \omega_4) * \frac{R}{4}$$

Transversal Velocity:

$$v_y(t) = (-\omega_1 + \omega_2 + \omega_3 - \omega_4) * \frac{R}{4}$$

Angular velocity:

$$\omega_z(t) = (-\omega_1 + \omega_2 - \omega_3 + \omega_4) * \frac{R}{2(l_x + l_y)}$$

Mobile robot full velocity:

$$v_{full} = \sqrt{{v_x}^2 + {v_y}^2}$$

The direction of the mobile robot's motion vector:

$$\theta = \frac{1}{\tan(v_x + v_y)}$$

The inverse kinematics problem is defined by the following dependencies:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & -1 & -\frac{(l_x + l_y)}{2} \\ 1 & 1 & \frac{(l_x + l_y)}{2} \\ 1 & 1 & -\frac{(l_x + l_y)}{2} \\ 1 & -1 & \frac{(l_x + l_y)}{2} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix}$$

The angular speed of each wheel of the mobile robot:

$$\begin{cases} \omega_1 = \frac{1}{R}\left(v_x - v_y - \left(\frac{(l_x + l_y)}{2}\right) * \omega\right) \\ \omega_2 = \frac{1}{R}\left(v_x + v_y + \left(\frac{(l_x + l_y)}{2}\right) * \omega\right) \\ \omega_3 = \frac{1}{R}\left(v_x + v_y - \left(\frac{(l_x + l_y)}{2}\right) * \omega\right) \\ \omega_4 = \frac{1}{R}\left(v_x - v_y + \left(\frac{(l_x + l_y)}{2}\right) * \omega\right) \end{cases}$$

## 2.2. Odometry

### 2.2.1. Localization

In automated driving applications, localization is the process of estimating the pose of a vehicle in its environment. Localization algorithms use sensor and map data to estimate the position and orientation of vehicles based on sensor readings and map data. It is a critical component of autonomous navigation, enabling the robot to understand where it is in relation to its surroundings, so it can plan and execute movements effectively.

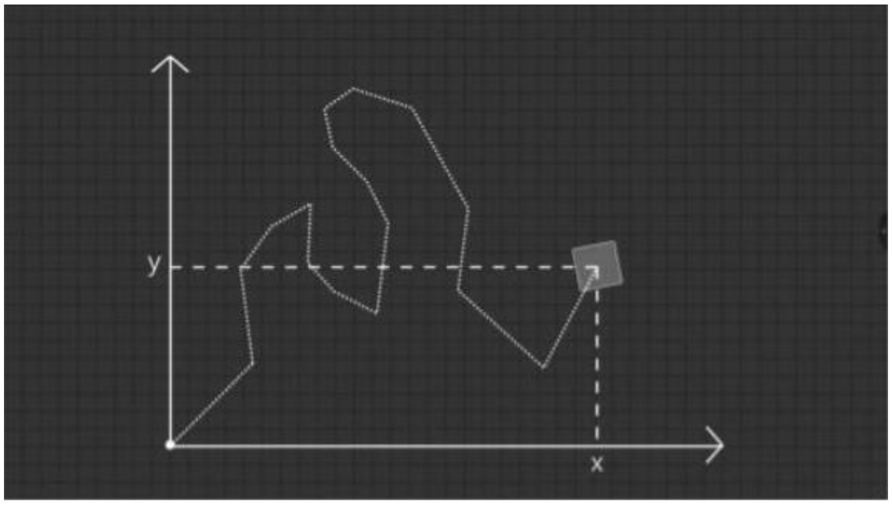


***Figure 2.2****: Point tracking [24]*

### 2.2.2 Pose

Pose is referred to the position of the something in a certain environment, in the context two-dimensional space, as the movement of the robot is generally constrained to a single plane. A pose contains two entries: the robot's position and heading "Position" is generally in Cartesian coordinates, so the pose can be represented with x, y and θ, "Heading" is a term for the direction towards which the front of the robot is facing. Because of this, the robot's coordinate frame is set up such that the global x-axis is lined up with the 0 heading.

We can refer to the current pose as $\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$. The change in pose over some very small amount of time $(\Delta S)$ The difference in time between the current pose and the last pose should be as small as possible to improve the approximations for the math. The updating of the robot frame should be implemented every cycle of the control loop.

Updating the pose is as simple as adding the transformed change to the previous pose:

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ \theta_0 \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{pmatrix}$$

Now the mathematic problem close to finding the displacement of x, y and θ for every period of time $\Delta t$. In order to determine the current location of the robot and update its pose, the change must be calculated using data read from the sensors.

## 2.3. Mapping

Mapping is the process of generating the map data used by localization algorithms. System can obtain map data by collecting data from various sources of measuring distance sensors(2D,3D lidar, Kinect camera, ultrasonic sensor,…). Alternatively, use simultaneous localization and mapping (SLAM) algorithms to incrementally build a map and estimate the location of a vehicle at the same time. The goal of mapping is to generate a map, which is typically a 2D or 3D representation, that captures the spatial layout, obstacles, and other relevant features of the surroundings.

Mapping is a fundamental capability for robots to navigate and operate autonomously in their environment. By building a map, a robot can understand where it is located, plan paths, avoid obstacles, and make intelligent decisions based on its surroundings.

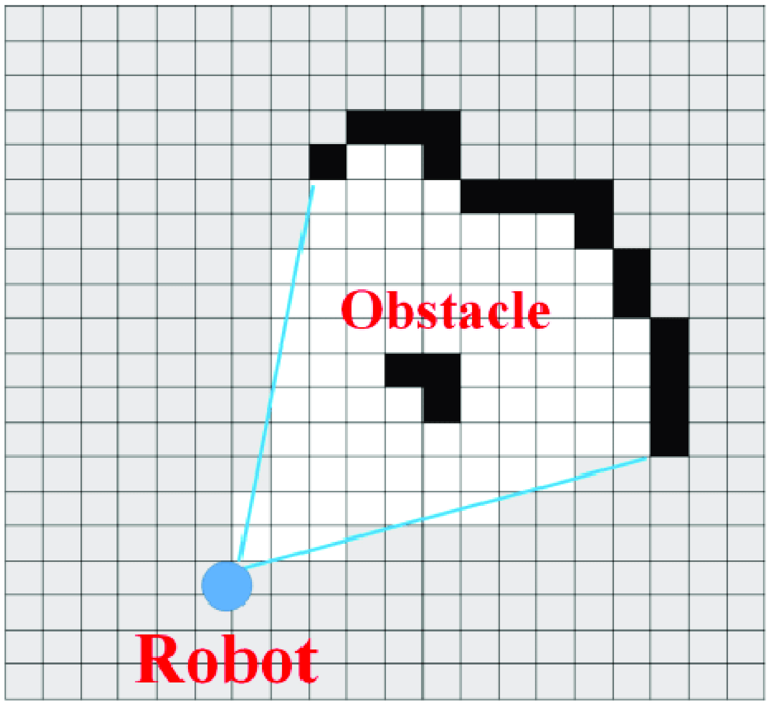


***Figure 2.3**: Occupancy grid map [15]*

An occupancy grid map is a representation of a robot's environment divided into a grid, where each cell contains a probability value indicating whether that cell is occupied, free, or unknown. Occupancy grid is the final result for 2D mapping. The grid structure is simply made of a 2D array with each unit described as a square cell. Each cell represents a small, uniform area of the real world. The array is always set all blocks of cell at the unknown value at the beginning.

An occupancy grid map contains with 3 main values in each unit:

- Occupied: The cell is occupied by an obstacle (e.g., value near 1 or 100%).
- Free: The cell is empty and traversable (e.g., value near 0 or 0%).
- Unknown: The state of the cell is not yet determined (e.g., a value of 0.5 or 50%).

When it comes to operation, the map is dynamically updated as the robot receives new sensor data. Sensors like LiDAR, sonar, or depth cameras detect objects and distances in the robot's surroundings are sensor input for updating of the cell’s state.

## 2.4 Navigation

### 2.4.1 Path Planning

Path planning is the process of determining a feasible and efficient route for a robot or autonomous system to travel from a starting point to a goal. Simply understand that find a way from point A to point B with the least effort. One of the way for efficient path planning is that A* algorithm. Unlike other conventional algorithm, A* has the least time and consume less resource to calculate than the others.

Application of path planning in industries:

- Robotics: Autonomous navigation for mobile robots, robotic arms, and drones.
- Autonomous Vehicles: Lane navigation, obstacle avoidance, and parking.
- Logistics: Route optimization for delivery robots and warehouse operations.

### 2.4.2 Path Following

Path following is a method used by robots and autonomous systems to track and adhere to a predefined path. Unlike path planning, which determines the optimal route, path following ensures accurate execution of the planned path. It involves reducing deviations between the robot's position and the desired path using real-time feedback. Sensors like GPS, IMU, and encoders are crucial for providing data about the system's position and orientation. Controllers, such as PID, Pure Pursuit, and Stanley, adjust the robot’s motion to follow the path accurately.

Paths can be represented as discrete waypoints or smooth curves, such as splines or Bézier curves. Path following ensures smooth navigation while respecting physical constraints like turning radius and acceleration limits. Dynamic adaptability allows robots to adjust for disturbances like slippage or environmental changes. This feature is essential for applications requiring precise motion, such as autonomous vehicles and robotic arms. Real-time operation ensures the system continually corrects its trajectory for accurate path tracking.

Path following systems optimize navigation by balancing speed, safety, and energy efficiency. Advanced algorithms enable smooth transitions between straight paths and curves, ensuring comfort and stability. Real-time feedback integration minimizes errors and enhances path adherence in dynamic environments.

Obstacle avoidance can be integrated, enabling adjustments to avoid collisions while maintaining the path. Multi-objective optimization ensures paths meet diverse requirements, from precision to safety. These systems are scalable, supporting configurations like UAVs, marine vehicles, and multi-jointed robots. Autonomous vehicles use path following for lane navigation, parking, and merging into traffic. Drones rely on it for aerial surveys and video capturing along specific routes. Mobile robots follow paths for logistics tasks in warehouses and factories. Path following is also essential for robotic arms, guiding precise movements for tasks like welding or assembly.

## 2.5 ROS (Robot operating system)

### 2.5.1 Overview about ROS

ROS (Robot Operating System) is an open-source framework for building robotic systems. Despite its name, it is not a traditional operating system but rather a set of software libraries and tools that help developers create robot applications. ROS is designed to work on top of a traditional operating system like Linux, providing a set of services to manage the complexities of robotics development. From drivers and state-of-the-art algorithms to powerful developer tools, ROS has the open source tools you need for many robotics project.

There are 2 version of ROS:

- ROS 1: The original version of ROS, which is widely used for research and development but has certain limitations when it comes to real-time performance and support for distributed systems.

- ROS 2: A more recent iteration that addresses some of ROS 1's shortcomings. ROS 2 provides better support for real-time systems, improved security, and enhanced communication protocols, allowing it to work in more varied industrial and production environments.

***Figure 2.4****: ROS2 humble distro*

ROS is a highly versatile and powerful tool for developing robotics systems, from small educational robots to large industrial machines. It simplifies robot development by providing an open-source ecosystem for communication, hardware abstraction, and a wide range of libraries for robot functionality. With the development of ROS 2, the platform has become more suitable for real-time, robust, and scalable industrial applications, ensuring its relevance for the future of robotics.

### 2.5.2 Nodes

In ROS (Robot Operating System), a node is a fundamental concept representing an independent process that performs a specific task within a robot system. Each node in ROS should be responsible for a single, modular purpose, e.g. controlling the wheel motors or publishing the sensor data from a laser range-finder. Each node can send and receive data from other nodes via topics, services, actions, or parameters.

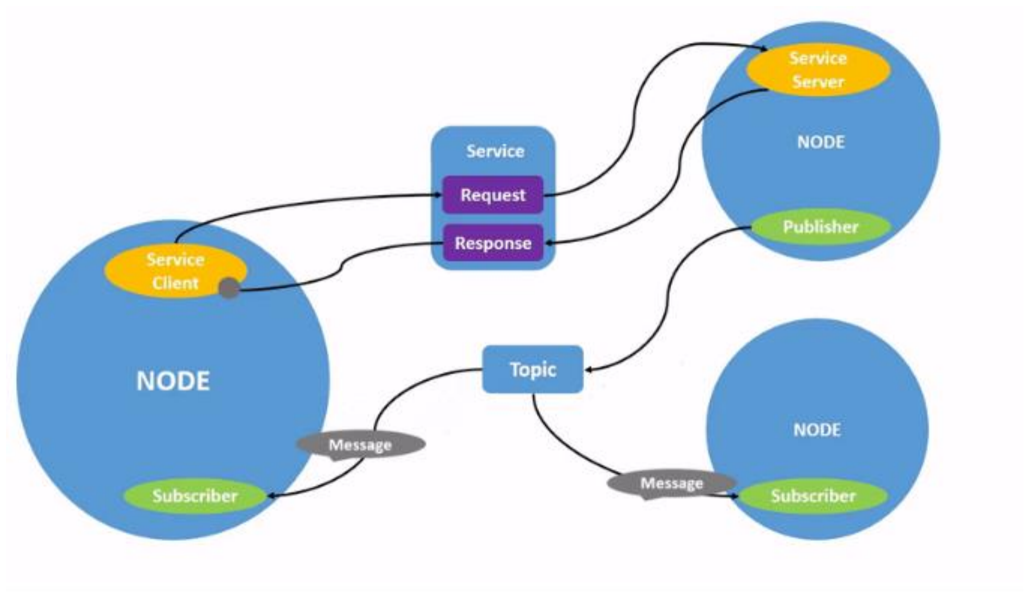


***Figure 2.5*** *Nodes block [26]*

In ROS a node could easily understand as a program in the form of subscriber, publisher or service and client. Each node communicates through topic and service using a message. With this advantage nodes in ROS have a flexible feature because of a whole systems is made by multiple nodes, every node now is an subsystem or a feature to the big program. Then it is so easy for maintain or replace any features or subsystem because the nodes are only represent for a specific feature.

### 2.5.3 Publisher and Subscriber

In ROS (Robot Operating System), Publisher and Subscriber are fundamental communication concepts used for message exchange between nodes. They operate on the publish/subscribe model, which allows one node to publish data to a topic, and other nodes to subscribe to that topic to receive the data. This is a decoupled way of communication, meaning that the publisher and subscriber do not need to be aware of each other directly. Simply Explain, publishers are nodes which produce data and subscribers are nodes which consume data.

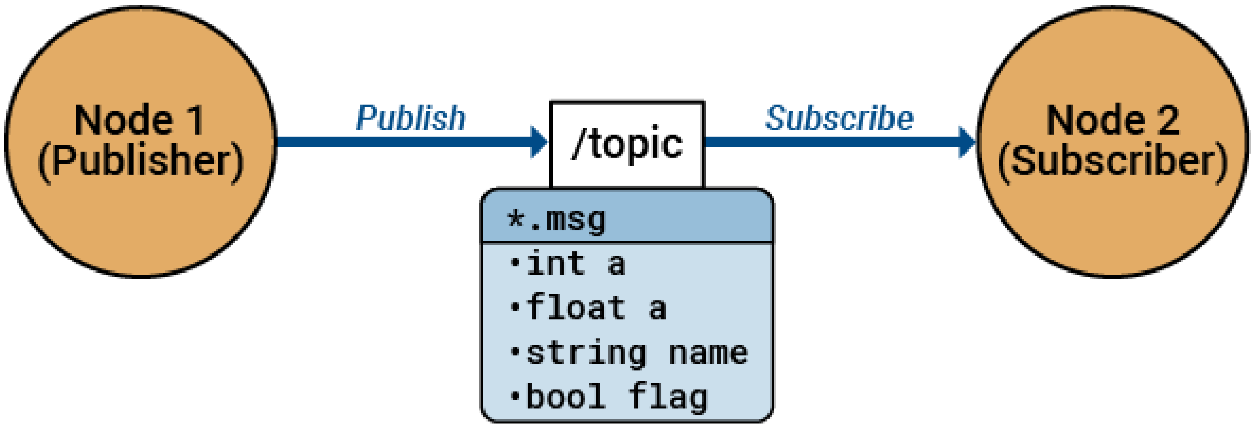


***Figure 2.6**: Publisher and Subscriber [27]*

Publisher could represent as a speaker which anyone can hear through "/topic" or think of the way they communicate is using topic. For a Publisher to public a data it needs to initialize and create a topic and inside the topic there is a message type, which could be any kind of data types.

Subscriber could represent as a listener which hear message every time it sent. For a Subscriber to work it need to have a "/topic" to hear. Beside a for one Subscriber can only hear to one Publisher, but one Publisher can speak to multiple Subscriber. In a node, it is possible to create many subscriber publisher in the same nodes as long as these nodes work in a proper way.

Topic could be seen as a station which store the information. A topic existed a form of communication between Subscriber and Publisher.

### 2.5.4. Service and Client

Services are another method of communication for nodes in the ROS graph. Services are based on a call-and-response model versus the publisher-subscriber model of topics. While topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client. A node can be exist in the form of service and client for various feature and function. In ROS, Services and Clients provide a synchronous communication model that allows nodes to request and respond to specific actions or data, unlike the Publish/Subscribe model, which is asynchronous. Services are useful when a node needs to request data or trigger an action from another node and wait for a response. This is more appropriate for scenarios where the client needs a direct result or confirmation before continuing with other tasks.

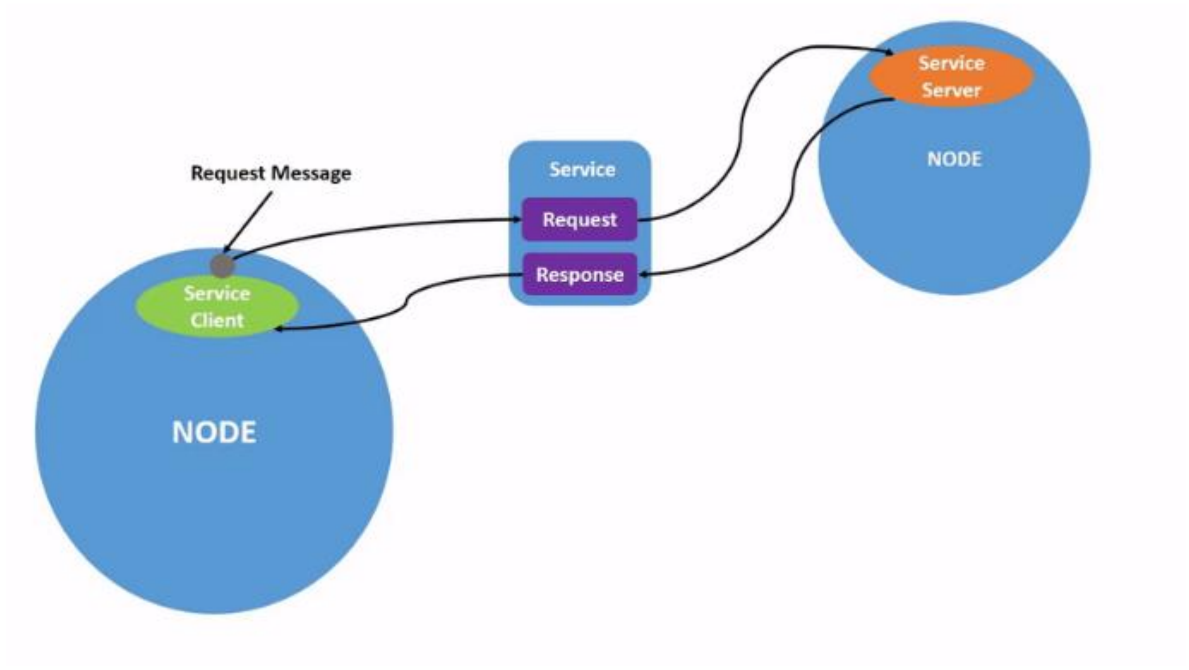


***Figure 2.7***: *Service and Client [28]*

A Service in ROS 2 is a method that a node offers to provide some functionality or data upon request. It works on a request-response model, where the client sends a request to the service, and the service processes that request and sends back a response.

A Service Client is a node that sends a request to a service and waits for a response. The client can send a request containing parameters to the service and then handle the response once the service finishes its processing.

# Chapter 3: SYSTEM DESIGN

## 3.1 Hardware Design:

### 3.1.1 Hardware planning

The robot platform is designed for autonomous operation in indoor environments, utilizing a 2D LiDAR sensor for navigation and mapping. Its construction focuses on the following key features:

- Compact Dimensions: Optimized for easy maneuvering in confined indoor spaces.

- Modular Design: Simplifies component integration and replacement.

- Lightweight, Durable Frame: Built with aluminum extrusion to balance weight and strength.

The robot's frame comprises two 2040 aluminum extrusion pieces on the sides and two 2020 aluminum extrusion pieces in the center, connected by four 90-degree corner keels for stability. The frame dimensions are 30 cm in length and 23 cm in width, forming a compact yet durable base.

Frame Materials and weight:

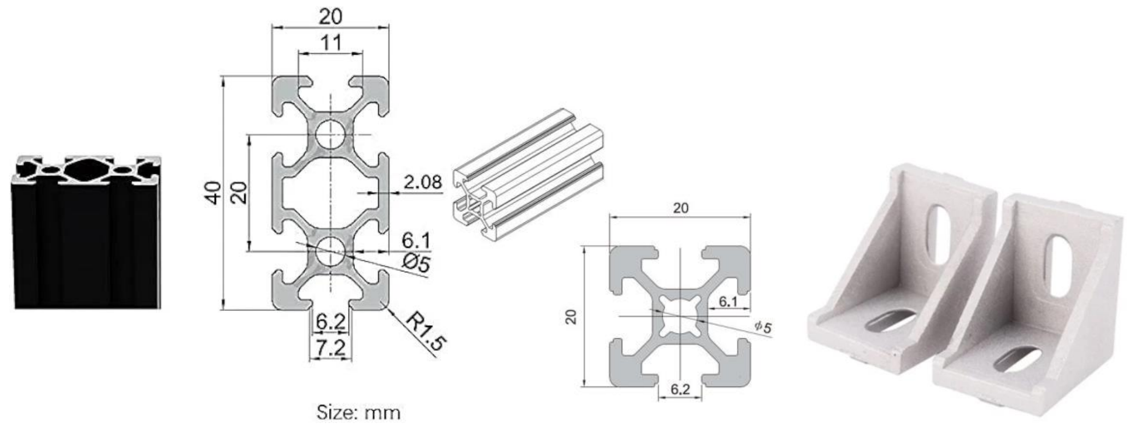


***Figure 3.1:*** *2020& 2040 Extrusion aluminum and keel*

- Two 2040 Aluminum extrusions pieces (Approx. 400 g total)

- Two 2020 aluminum extrusions pieces (approx. 250 g total)

- Four 90-degree corner keels (approx. 60 g total)

The complete frame weighs approximately 710 g, offering a robust yet lightweight structure suitable for indoor use. This design ensures a balance of maneuverability, load capacity, and structural integrity, making it ideal for various autonomous applications. The profile of Extrusion as illustrated in [fig 3.1

Mecanum wheel:

The Mecanum wheel enables omnidirectional movement, allowing robots or vehicles to move forward, backward, sideways, diagonally, and rotate without changing orientation. Its unique design features rollers angled at 45°, enabling smooth and precise motion. Widely used in robotics and automation, Mecanum wheels excel in tight spaces and precise positioning but require advanced control and perform best on flat surfaces.

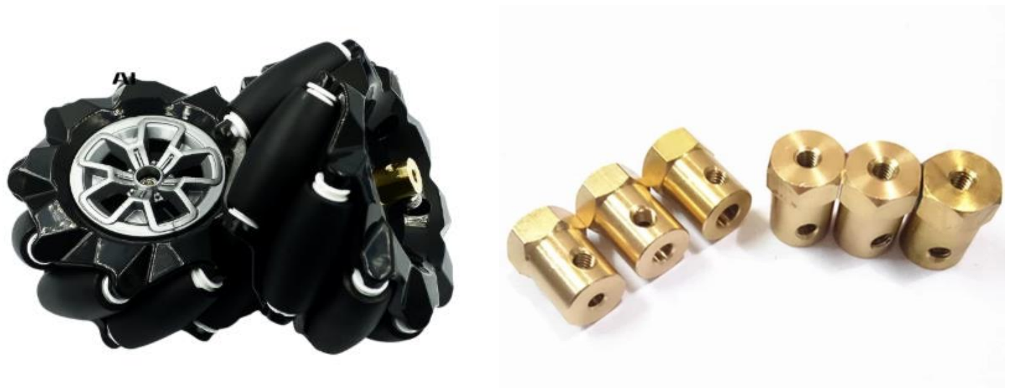


***Figure 3.2****: Mecanum wheel& Hex nut*

Specifications:

- Material: ABS plastic + soft rubber wheel.

- Diameter: 97mm

- Hexagonal shaft size: 8.4mm (compatible with 8.3mm hexagonal shafts)

## 3.1.2. Designing Robots chassis

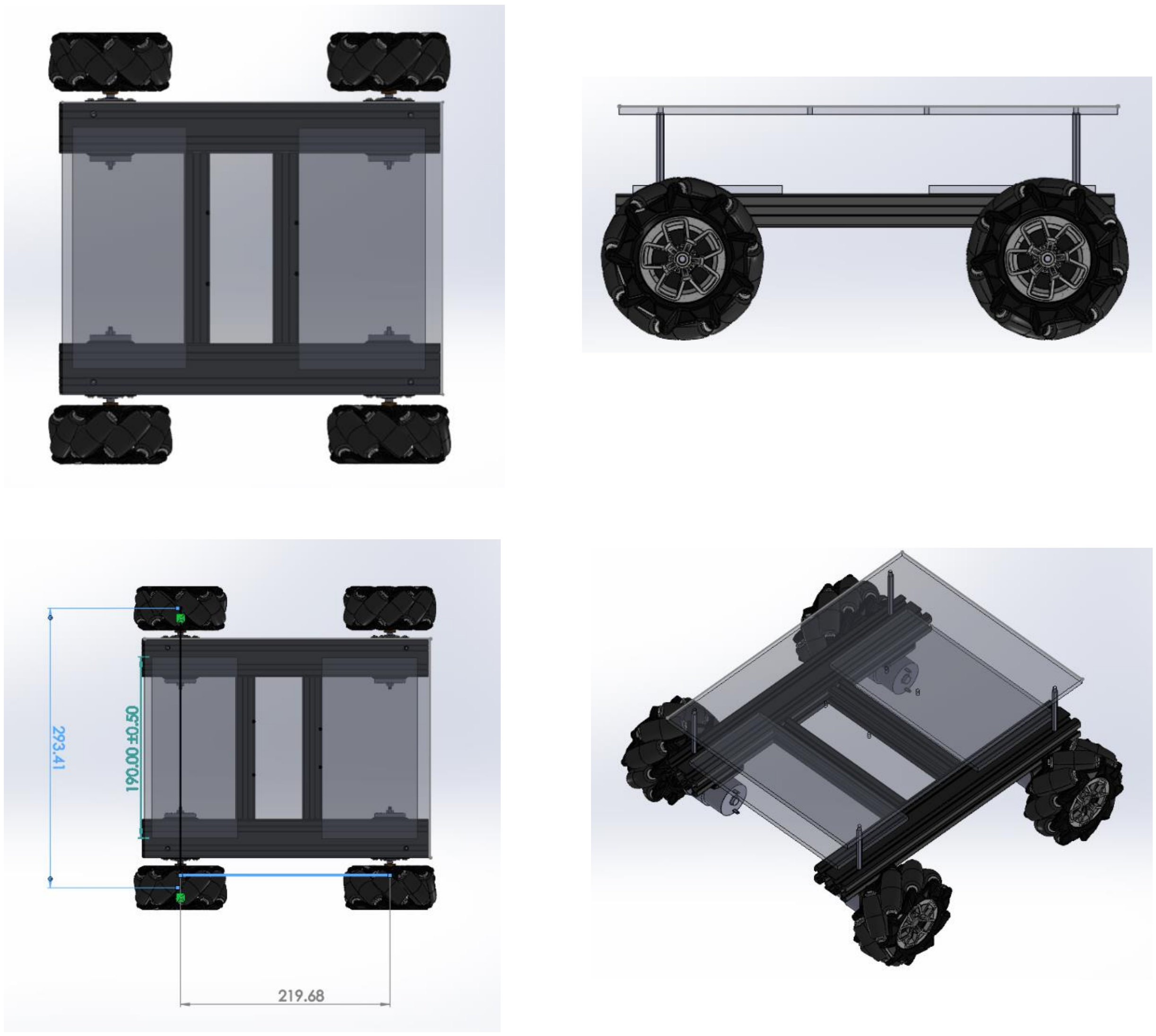


***Figure 3.3***: *3D model of mobile robot*

The robot chassis was designed in SolidWorks to achieve a compact and modular structure, suitable for indoor environments. The design process began with a conceptual sketch, which was translated into a 3D model using parametric modeling techniques. The chassis was modeled using 2020 and 2040 aluminum extrusions, with Mecanum wheel mounts integrated for precise alignment.. Additional features such as mounting holes for the LiDAR and motor drivers were incorporated into the design, streamlining the assembly process. The dimension of the Robot as illustrated in [ fig 3.3] and [ Table 3.1]

*Table 3.1: Chassis dimension*

| | |
|---|---|
| **Wheel base** | 219.68mm |
| **Wheel track** | 239.41mm |
| **Ground clearance** | 22mm |

### 3.1.3 Overall electrical components block diagram

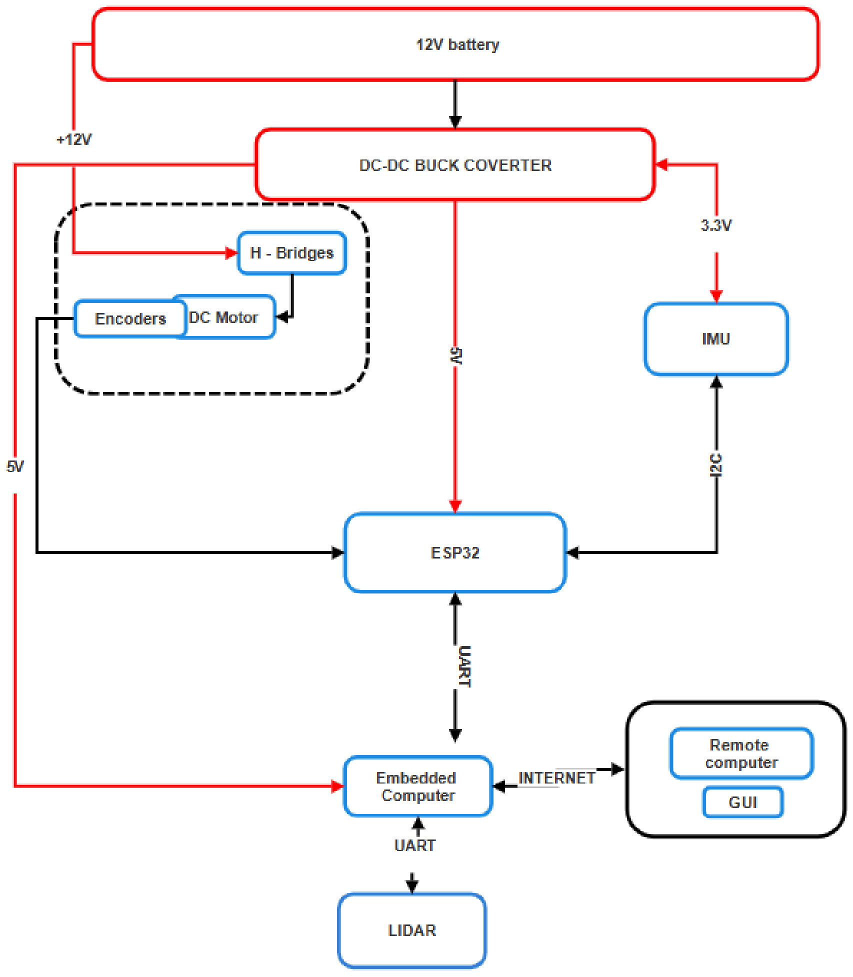


***Figure 3.5:*** *Electrical block diagram*

**- Power Supply:** A 12V 6-cell 18650 Li-ion battery offering 6Ah capacity ensures uninterrupted operation.

-

***Figure 3.6:*** *12V 6S 18650 battery with BMs*

+Operating voltage: 10,8-12,6 V

+Max current: 10A

**- Motor Drivers (L298N):** For driving motors with adequate power and control.

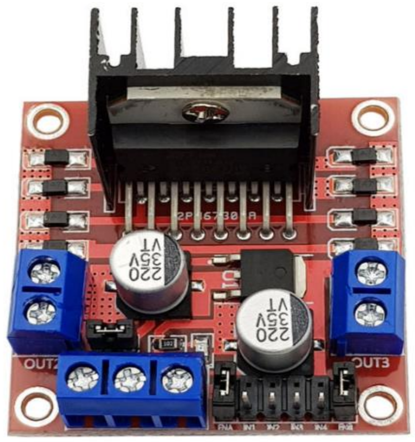


***Figure 3.7****: L298N module*

+ Operating Voltage: 5~30VDC

+ Maximum Current for Each H-Bridge: 2A

+ Logic Voltage Levels:

- Low: -0.3V~1.5V
- High: 2.3V~Vss

+ Dimensions: 43x43x27mm

**- Motors (JGB37-520):** Delivering reliable and controlled movement.

***Figure 3.8**: JGB37-520 Motor*

+Rated voltage: 12V DC

+Reduction ratio: 1:168

+No-load current: 120 mA

+No-load speed: 60 RPM

+Rated speed: 46 RPM

+Rated torque: 19 kg.cm

+Max torque: 31 kg.cm

+Rated current: 1 A

+Stall current: 2.3 A

+Hall speed sensor: 11 pulse/round

**- MPU6050:** for motion sensing and orientation detection

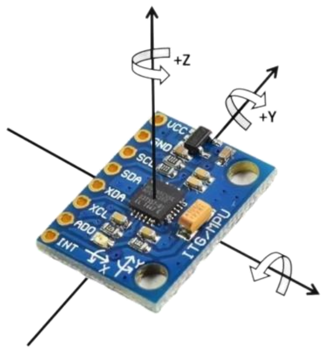


***Figure 3.9***: *MPU6050 axis*

+ 16-bit ADC resolution.
+ I²C Communication
+ Built-in DMP (Digital Motion Processor)
+ Dimension: 21mmx16mm

**- ESP32:** The ESP32, by Espressif Systems, is a versatile microcontroller for IoT, embedded systems, and smart applications, featuring dual-core performance, Wi-Fi, and Bluetooth connectivity. Physical of ESP32 as can be seen as fig.

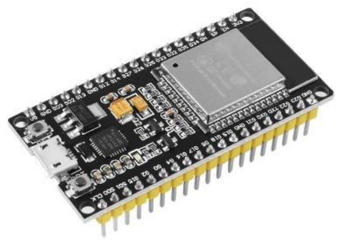


***Figure 3.7***: *ESP32 dev module*

+ Duo core processor: Two Xtensa® LX6 processors, running at up to 240 MHz

+ Wireless Connectivity: Wi-Fi (802.11 b/g/n), Bluetooth 4.2

+ Operating Voltage: 3.3V.

+ Flash Memory: Up to 16 MB external flash and 8 MB PSRAM.

+ Temperature Range: -40°C to 125°C.

+ 38 GPIO Pins**:** Supports multiple peripherals such as ADC, DAC, SPI, I²C, UART, and PWM.

+ Analog and Digital Support**:** Includes a 12-bit ADC and a 2-channel 8-bit DAC for precision input/output

+ Compatible with popular frameworks like Arduino IDE, ESP-IDF, and MicroPython.

+ Dimension 55mmx24mm.

**- Raspberry Pi 4B**: for running computationally intensive tasks like SLAM, image processing, and AI-based navigation in autonomous robots.

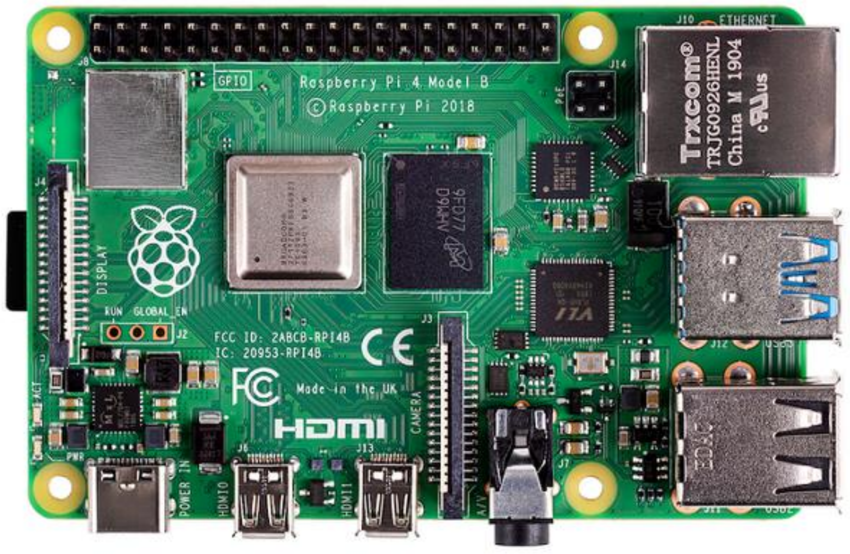


***Figure 3.8****: Raspberry Pi 4B*

+ Processor: Quad-core Cortex-A72 (ARM v8) 64-bit CPU running at 1.5GHz.

+ Memory: Equipped with 4GB LPDDR4 RAM.

+ Connectivity: Features Gigabit Ethernet, dual-band Wi-Fi (2.4GHz/5GHz), and Bluetooth 5.0.

+ USB Ports: Includes 2 USB 3.0 ports and 2 USB 2.0 ports.

+ Video and Display: Supports 4K resolution at 60Hz through dual micro-HDMI ports.

+ Storage Options: Boots from a microSD card or external USB storage.

+ GPIO: Offers a 40-pin GPIO header for connecting peripherals such as sensors and motors.

+ Power Supply: Requires a 5V/3A input via USB-C.

## 3.1.4 Motor controlling circuit.

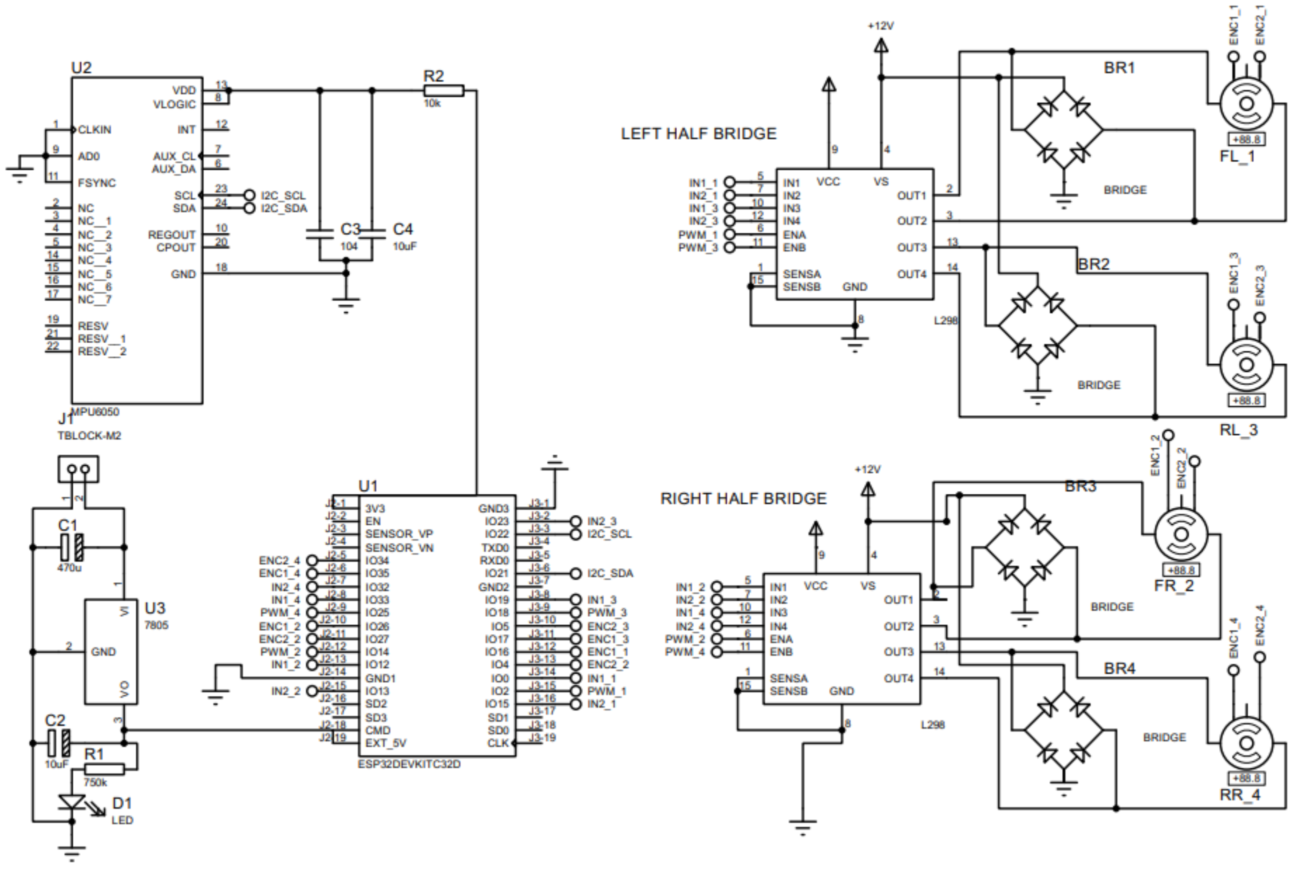


***Figure 3.9a**: Circuit schematic*

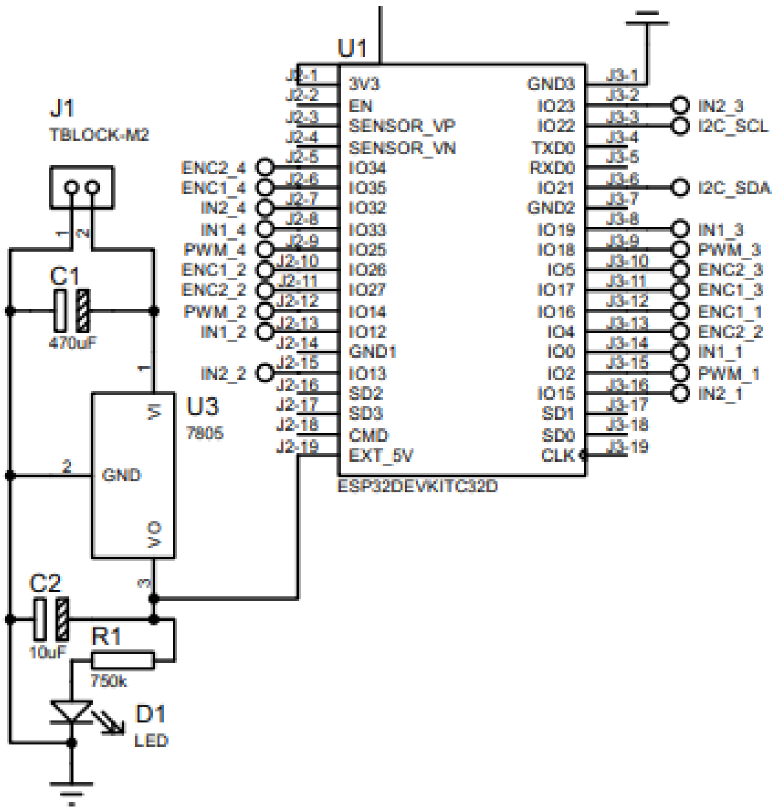


***Figure 3.9b:*** *Power supply for ESP32 schematic*

The ESP32 is powered by a voltage regulator LM7805, which consume 12VDC from the 12V battery. This component also comes with two capacitors for filtering and smoothing 5VDC supplying to the main controller. The used pins for peripherals of the controller can be seen in [ Fig 3.9b]

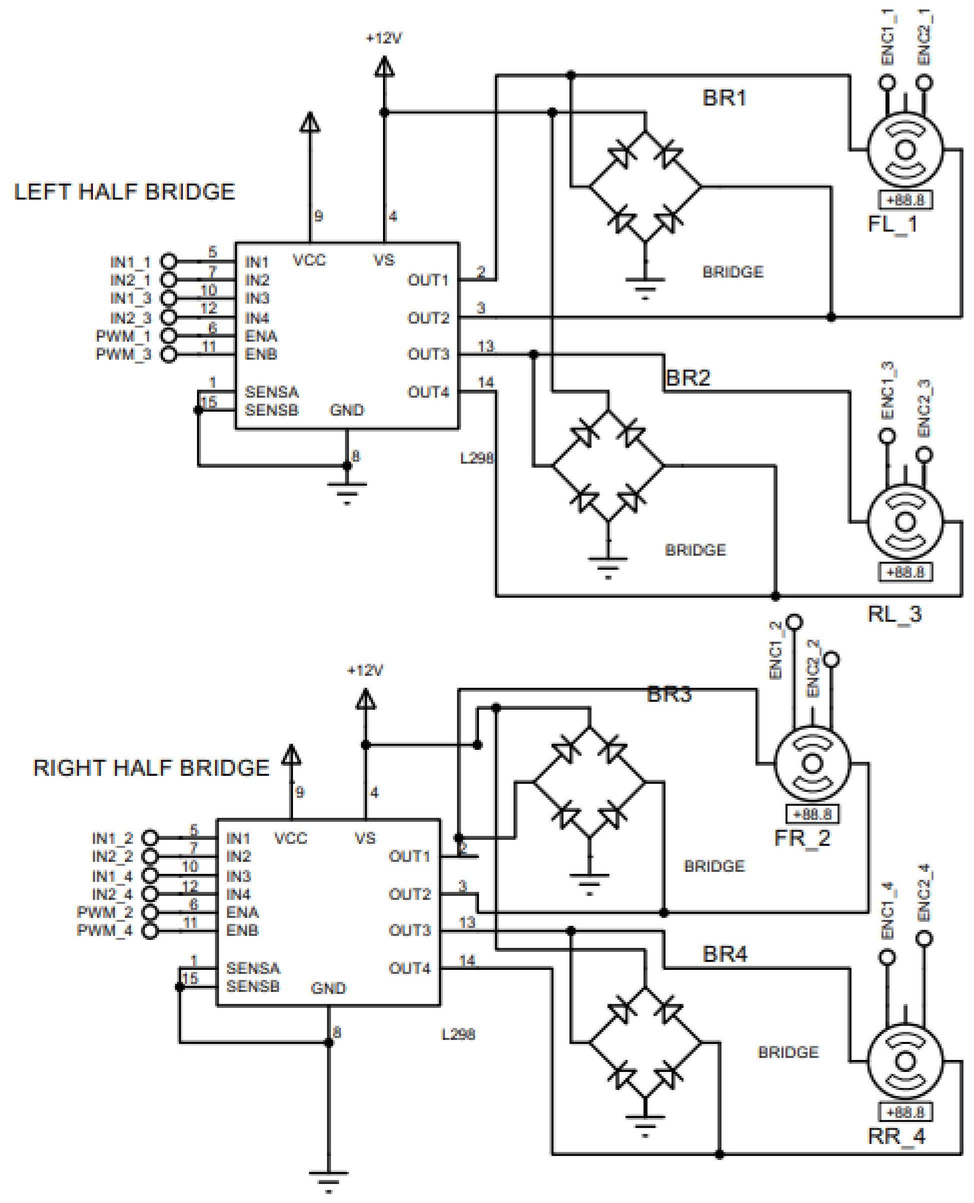


***Figure 3.9c:*** *H bridge Motor driver schematic*

The circuit diagram illustrates the use of dual L298N H-bridge motor driver modules to control four JGB37-520 DC motors for a 4WD robot. Below is a detailed description:

L298N H-Bridge Modules:

- Each L298N module is divided into two halves (left and right), allowing independent control of two motors per module.

- The IN1–IN4 pins control the motor direction, while the ENA and ENB pins regulate motor speed through PWM signals.

Power Supply:

- A 12V DC power source is connected to the VS pin of the L298N modules to power the motors.

Motor Connections:

- Each motor is connected through its respective half-bridge (BR1, BR2, BR3, BR4).

- Forward Left Motor (FL_1) and Rear Left Motor (RL_3) are controlled by the left half-bridge of the left half bridge as illustrated in [fig 3.9b].

- Forward Right Motor (FR_2) and Rear Right Motor (RR_4) are controlled by the right half-bridge of the right half bridge as illustrated in [fig 3.9b].

Encoders:

- Each motor features an encoder (ENC1, ENC2, etc.) for speed and direction feedback, allowing precise control and monitoring. These encoders connected to ESP32 via ADC / IO pin as illustrated in [fig 3.9a, fig 3.9c]

Control Logic:

- The IN1–IN4 pins control the motor direction, while the ENA and ENB pins regulate motor speed through PWM signals as illustrated in [fig 3.9a, fig 3.9c]

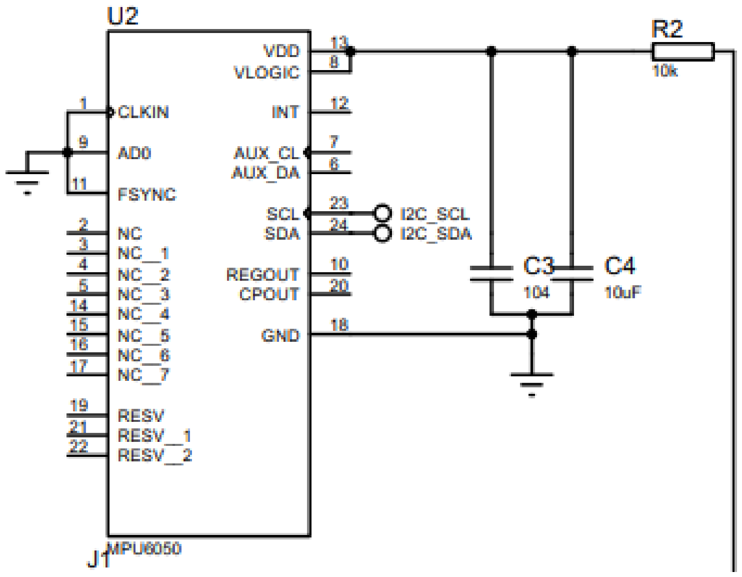


***Figure 3.9d**: MPU6050 schematic*

Meanwhile MPU6050 powered by ESP32 via 3.3V pin and have I2C communicate with ESP32 SCL & SCA pin as can be seen as [Fig 3.9a, Fig 3.9d]

## 3.2 Software architecture

### 3.2.1. Motion Control

#### 3.2.1.1 Mecanum wheels

The mecanum wheel is based on the principle of a central wheel with placed rollers inclined at an angle to the circumference of the wheel . The angle between the roller axis and the central wheel axis is can have any value, but in the case of common mecanum it is 45.

The Mecanum wheel is a form of tireless wheel, with a series of rubberized external rollers obliquely attached to the whole circumference of its rim. The rollers are mounted at a 45-degree angle around the circumference of the wheel, allowing simultaneous longitudinal and lateral force generation. Each Mecanum wheel is an independent non-steering drive wheel with its own powertrain, and when spinning generates a propelling force perpendicular to the roller axle, which can be vectored into a longitudinal and a transverse component in relation to the vehicle.

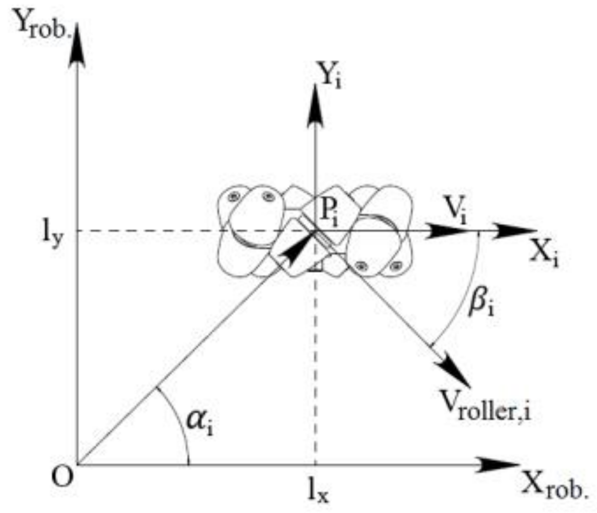


***Figure 3.12**: Mecanum wheel [29]*

The design of mechanum wheel help the system move in any direction with a very little space require, this the design highly use in some narrowed environment like industrial plant or small warehouse. The velocity of the wheel is nearly assemble to the y velocity direction(yr), in the mean time xr is trivial or have little impact on velocity's direction.

Depending on each individual wheel direction and speed, the resulting combination of all these forces produce a total force vector in any desired direction thus allowing the platform to move freely in the direction of the resulting force vector, without changing of the wheels themselves.

The number of the roller also effect to the motion and kinematics of the wheel and the wheel could have some sliding condition during operation which is a factor affect to the desired output and localization of the system.

#### 3.2.1.2. Wheel Configuration

Mecanum wheels are equipped with rollers set at a 45-degree angle to the wheel axis, enabling the decomposition of forces into longitudinal and lateral components. This unique design allows the robot to achieve omnidirectional movement, which is particularly advantageous in constrained or dynamic environments. The typical configuration involves four wheels(Front-left, Front-right, Rear-left, Rear-right)

Each wheel is independently powered by a motor, and its rotation direction and speed contribute to the robot's overall motion. By coordinating the velocities of the wheels, the robot can achieve complex movements such as lateral translation, diagonal motion, and rotation in place. The wheels of system ought to be placed in parallel position in order to perform multi-directional velocity, each of the wheel should placed in the way the picture show.

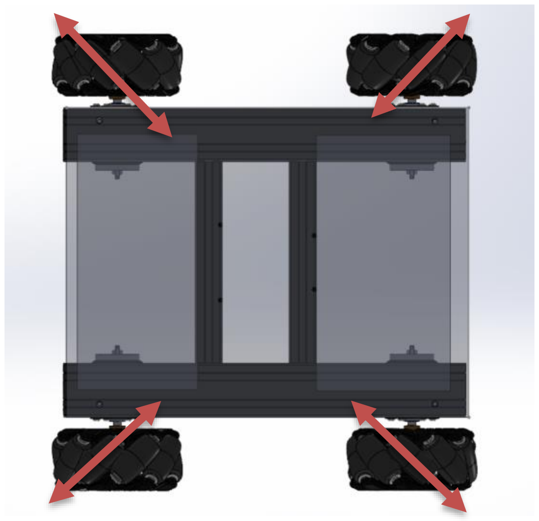


***Figure 3.14**: wheel configuration*

### 3.2.1.3 Motion Control

The robot is able to translate on any direction, forward/backward but also sideways left/right, and turning on the spot, thanks to its special wheels. This is especially helpful when having to maneuver in a tight environment such as a factory floor.

When Mecanum wheels are actuated, the angled peripheral rollers translate a portion of the force in the rotational direction of the wheel to a force normal to the wheel direction. - - Depending on each individual wheel direction and velocity, the resulting combination of all these forces produce a total force vector in any desired direction thus allowing the platform to move freely in the direction of the resulting force vector, without changing of the wheels themselves.

In the Kinematics section, the velocity function for the robot could be seen as:

$$\begin{cases} \omega_1 = \frac{1}{R}\left(v_x - v_y - \left(\frac{(l_x + l_y)}{2}\right) * \omega\right) \\ \omega_2 = \frac{1}{R}\left(v_x + v_y + \left(\frac{(l_x + l_y)}{2}\right) * \omega\right) \\ \omega_3 = \frac{1}{R}\left(v_x + v_y - \left(\frac{(l_x + l_y)}{2}\right) * \omega\right) \\ \omega_4 = \frac{1}{R}\left(v_x - v_y + \left(\frac{(l_x + l_y)}{2}\right) * \omega\right) \end{cases}$$

To control the 4 motors, we need to convert those velocities in angular velocities for each wheel:

front_left = (x_velocity - y_velocity – rot_ velocity * WHEEL_GEOMETRY) / WHEEL_RADIUS

front_right = (x_velocity + y_velocity + rot_velocity * WHEEL_GEOMETRY) / WHEEL_RADIUS

back_left = (x_velocity + y_velocity - rot_velocity * WHEEL_GEOMETRY) / WHEEL_RADIUS

back_right = (x_velocity - y_velocity + rot_velocity * WHEEL_GEOMETRY) / WHEEL_RADIUS

This allows the program to transfrom from the desired movement (x_velocity, y_velocity and rotational velocity) to specific wheel speeds for each of the robot's four mecanum wheels. Then we convert each of the angular velocity into PWM input for each motor driver. The corresponding wheel speeds are calculated using the motion control algorithm.  The resulting wheel speeds are then passed to a function "Set_Single_Motor".

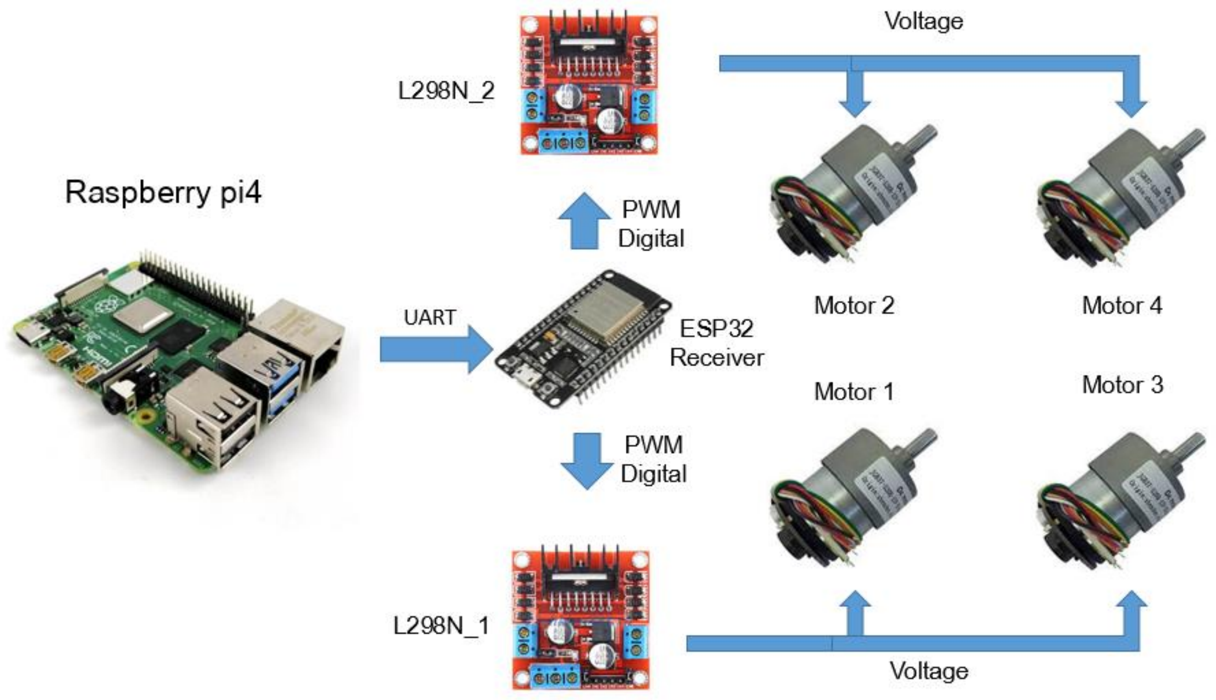


***Figure 3.15***: *Motion control diagram*

The system separate into two main parts, the controller and the receiver. First, the controller getting order from the remote keyboard, then the processor of Raspberry Pi4 decode and standardize the received data. Then, it send the data to via UART serial communication. The receiver now getting the data through UART and check weather that data is right or not. After checking process, the data is transfer to motion control function and then the ESP32 Receiver send out the control signal toward 2 motor driver "H-bridge" and make the motor to rotate in a logical way.

### 3.2.2. Odometry

### 3.2.2.1. Definition

Simply understand, Odometry is finding “where am i” compare to the original position, the process of determining the position and orientation of a robot (or vehicle) relative to a starting point. By continuously tracking how far the robot has traveled, odometry helps estimate "where" the robot is in relation to its original position. However, it often involves some degree of error accumulation over time due to factors like wheel slippage or sensor inaccuracies.

Odometry is the use of data from motion sensors to estimate change in position over time. It is used in robotics by some legged or wheeled robots to estimate their position relative to a starting location. This method is sensitive to errors due to the integration of velocity measurements over time to give position estimates. Rapid and accurate data collection, instrument calibration, and processing are required in most cases for odometry to be used effectively.

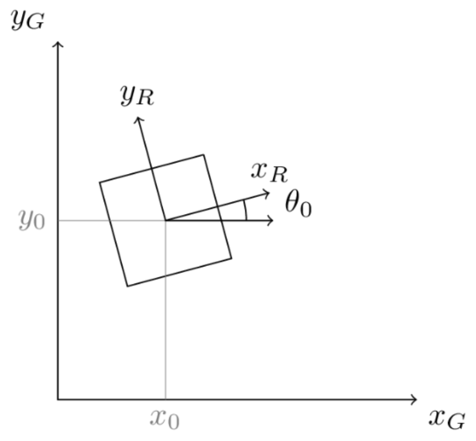


***Figure 3.16**: Odometry [28]*

Tracking each the changes in each step the robot take is the main mission of the subsystem “Odometry”. Each step, or movement, typically provides data about how far the robot has moved (distance traveled) and any change in its orientation (rotation). By combining these incremental changes, the odometry subsystem estimates the robot's current position relative to its starting point. The key challenge is to do this accurately over time, as small errors in each step can accumulate, leading to drift from the true position.

In this project, the two sensors used for odometry are IMU and encoder. Although this 2 sensors have a high error over time during long operation, but they are more accessible and affordable for using and getting data from it.

The microcontroller utilizes the I2C (Inter-Integrated Circuit) communication protocol to establish a connection and exchange data with the IMU (Inertial Measurement Unit) sensor, specifically the MPU6050. The microcontroller acts as the master device, initiating data requests and sending commands, while the MPU6050 functions as a slave device, responding to these requests and providing measurements such as acceleration and angular velocity.

Encoder also use for the odometry perpose and because encoder have various error odometry from the environment like road surface, sliding wheels, ... For that reason it more efficient to fuse encoder and IMU sensor to create more accurate data for odometry. In our case, the project use robot_localization package for fusing sensors with the resultant data is odometry.

#### 3.2.2.2 Wheel Odometry

Hall sensor is the kind of sensor that can detect the magnetic field of the magnet, not only the hall sensor can detect the appearance of magnetic field but also it can measure the amplitude of the magnetic field. Hall sensor have 2 different types: hall sensors have analog output and Hall sensors have digital output.

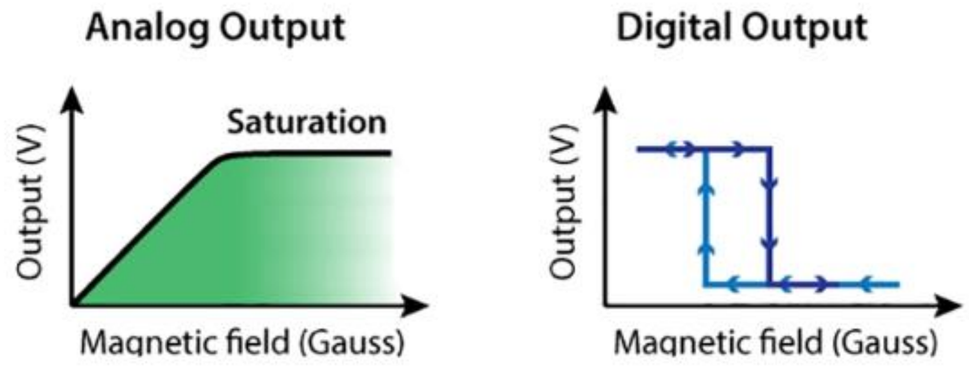


***Figure 3.17***: *Two types of hall sensor [29]*

The output of an analog sensor is proportional to the magnetic field strength or the material of the hall sensors. Since the output of this type of sensor is linear, it is often used for distance measurement. Meanwhile, the digital output type of sensor only provides two output states (On/Off) and has an additional element called Schmitt Trigger- which is capable of providing hysteresis or two different thresholds for the output to be high or low.

When an electric current flows through a metal plate in a magnetic field perpendicular to the direction of the current, then the electron and proton would to two opposite side of the metal or semiconductor bar. The ratio between the hall voltage and the current running through the hall bar is called the hall resistance, which is characteristic of the material that makes up the hall bar. Beside, hall sensor could wrongly read by several effect in some cases like getting expose to metal dust or strong magnetic field environmet.

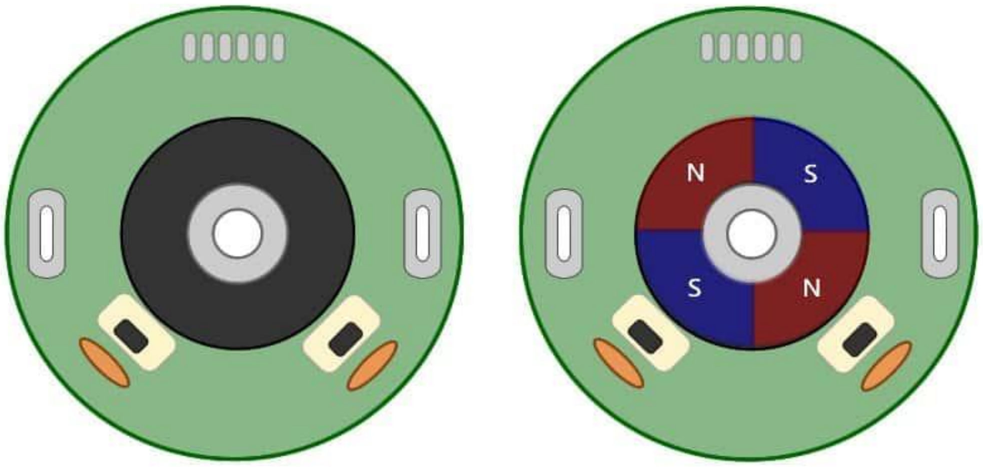


***Figure 3.18****: Hall-sensor mount on DC motor [29]*

The picture above is a simple diagram of the hall-effect encoder found on the back of our motors. The dark center mass is a round magnet that is fixed to the rear shaft of the motor. As it spins the sensors pass the symmetrically placed hall-effect sensors. As the north pole passes the sensor, it will trip and connect that channel output to ground.

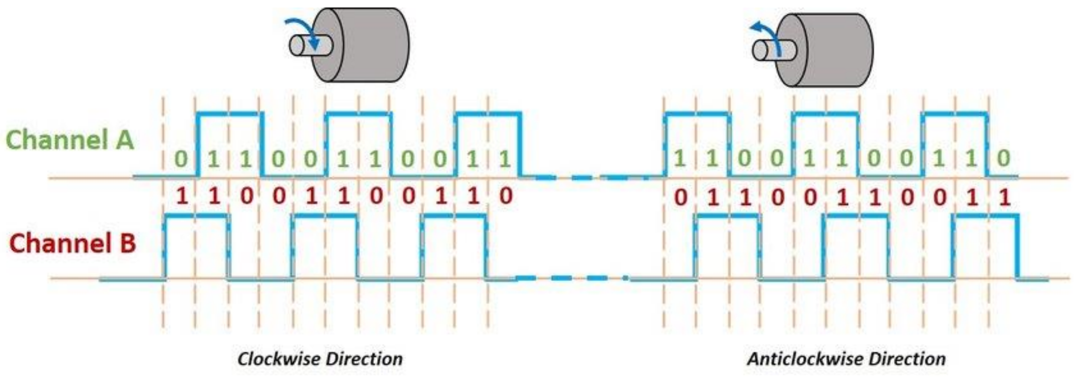


***Figure 3.19:*** *Channel of hall sensor [30]*

From these pulses we can calculate movement by watching the changes in the voltage levels of each encoder channel. The importance of having two channels that are offset about 90 degrees is that we can infer the direction of the motor by which channel is leading (goes from low to high first). This allows us increment or increment our encoder counts properly.

Velocity measure could be implement by counting how much pulse the hall sensor make every a certain period of time. To counting encoder, the processor could make a trigger flag for interrupt everytime the hall effect activates, and the direction is define the reading the other channel of the encoder.

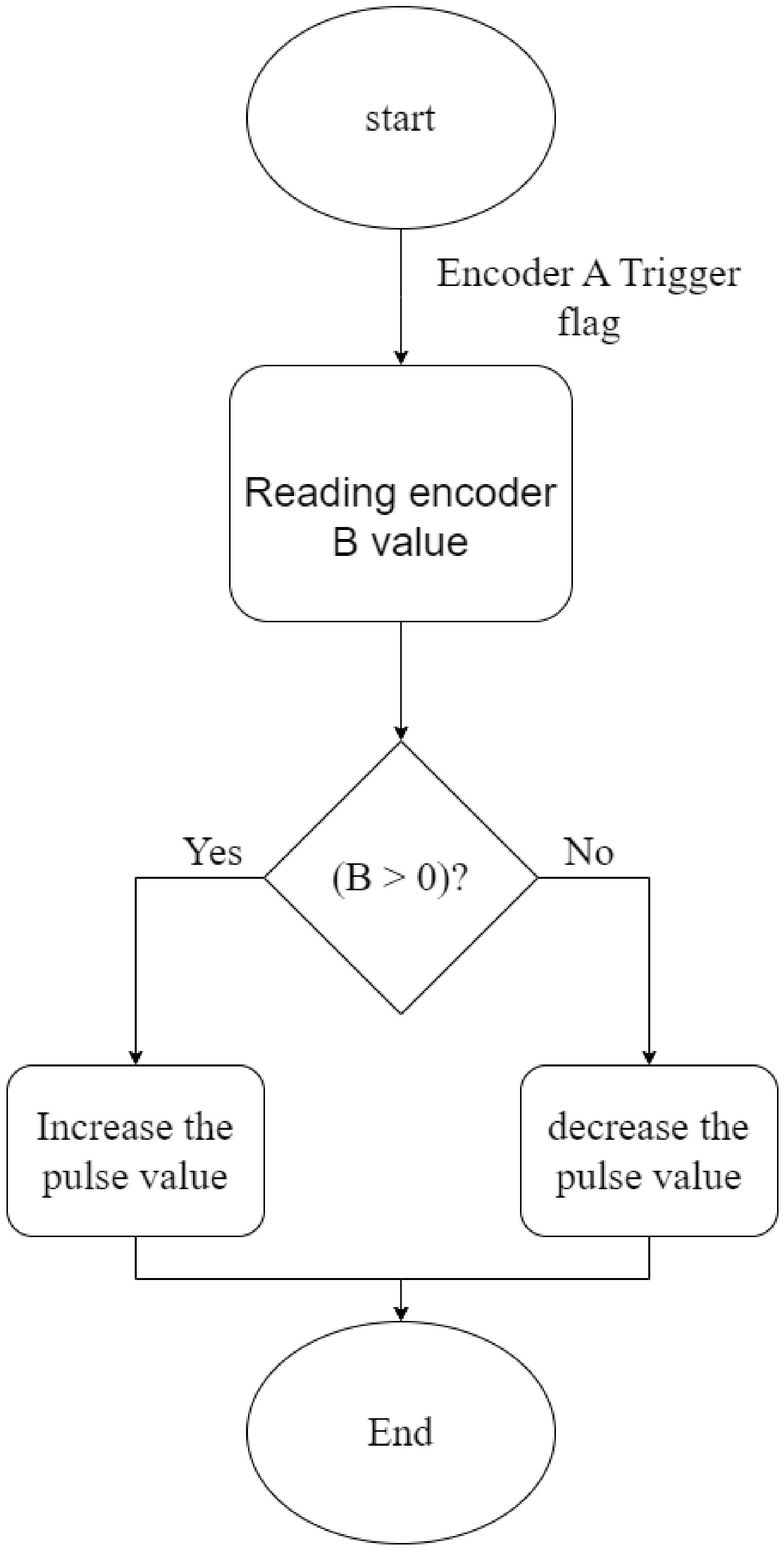


***Figure 3.20****: Encoder flow chart*

Each encoder generates pulses with **11** revolution per rotation and the transmission ratio of the gear box is **1:168**, then the total revolution for one rotation of the wheel is **1868**. The angular velocity of the wheel can be calculated as:

$$\omega = \frac{\Delta\theta}{\Delta t}$$

$$\omega = \frac{pulse * 2\pi}{\Delta t * 1868}$$

Where pulse is show as the number of **pulse** for a period of time($\Delta t$)

From equation [2.1] the robot velocity calculated by the encoder could be written as:

Longitudinal Velocity:

$$v_x(t) = (pulse_1 + pulse_2 + pulse_3 + pulse_4) * \frac{R}{4} * \frac{2\pi}{\Delta t * 1868}$$

Transversal Velocity:

$$v_y(t) = (-pulse_1 + pulse_2 + pulse_3 - pulse_4) * \frac{R}{4} * \frac{2\pi}{\Delta t * 1868}$$

Angular velocity:

$$\omega_z(t) = (-pulse_1 + pulse_2 - pulse_3 + pulse_4) * \frac{R}{4} * \frac{2\pi}{\Delta t * 1868 * (l_1 + l_2)}$$

Encoders are widely used in odometry for robots, but they come with certain disadvantages that may affect accuracy and reliability. These equation is not without error due to the lack of certainty of encoder. During operation, wheel slippage and skidding makes the encoder readings will not accurately represent the robot's actual motion. Small inaccuracies in the encoder readings accumulate over time, leading to significant errors in position estimation making long-term odometry less reliable without additional corrections from other sensors. There are also some impact of environmental factors, dust, dirt, or debris can interfere with optical encoders, reducing accuracy or causing signal loss. Magnetic encoders might be affected by strong magnetic fields.

Then to address these disadvantages, the robot combine with combine encoder data with imu sensors to filter out the errors of the two errors. By the application of sensor fusion, the mecanum robot can significantly enhance odometry accuracy and robustness.

#### 3.2.2.3 IMU Odometry

The IMU consists of two main sensors: Accelerometer and Gyroscope. These two sensors can be placed in many locations on the robot to detect movement in time and handle problems immediately.

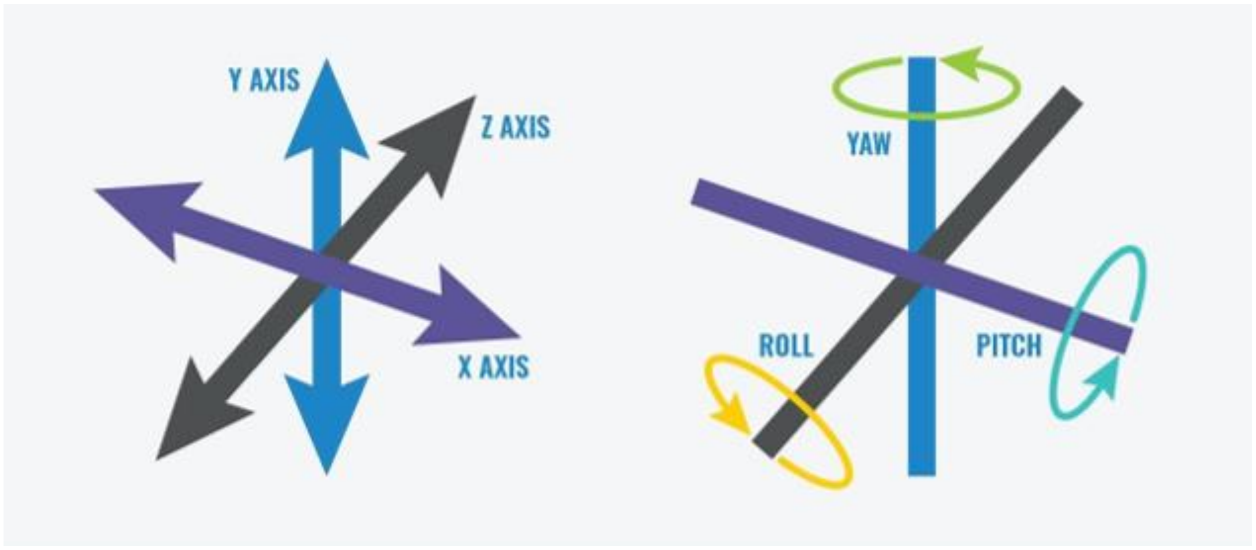


***Figure 3.21**: Six degree of freedom in IMU*

In which, Accelerometer is responsible for measuring its own acceleration sensor, and also measuring the acceleration of gravity, so the actual value when measured will include this factor. In addition, Gyroscope (abbreviated as gyro) plays the role of a sensor that measures the rotational speed when it rotates around an axis.

GY-521 6DOF IMU MPU6050 sensor is used to measure 6 parameters: 3 axis Rotation angle (Gyro), 3 axis acceleration direction (Accelerometer) and this sensor is used on the project.

The IMU sensor has some limitations typically noise sensitivity, cumulative errors, vibrations, … These limitations could be solved by fuse the data sensor to other types of sensor and in our case the IMU and encoder are selected for odometry mission.

#### 3.2.2.4. Sensor fusion

Sensor fusion is the process of combining data from multiple sensors to achieve a more accurate and robust representation of an environment or system state than any individual sensor could provide alone. It leverages the complementary strengths and mitigates the weaknesses of each sensor.

Sensor fusion combines data from multiple sensors to achieve improved accuracy, robustness, coverage, and reduced ambiguity. By integrating sensor data, noise and errors are minimized as each sensor's strengths compensate for others' weaknesses, ensuring precise measurements. Redundancy enhances robustness, allowing the system to remain operational even if a sensor fails or becomes unreliable. Additionally, combining sensors with complementary characteristics extends coverage, enabling effective operation across a range of scenarios. Cross-referencing data from multiple sources also reduces uncertainty and resolves conflicting measurements, providing a reliable and comprehensive understanding of the environment.

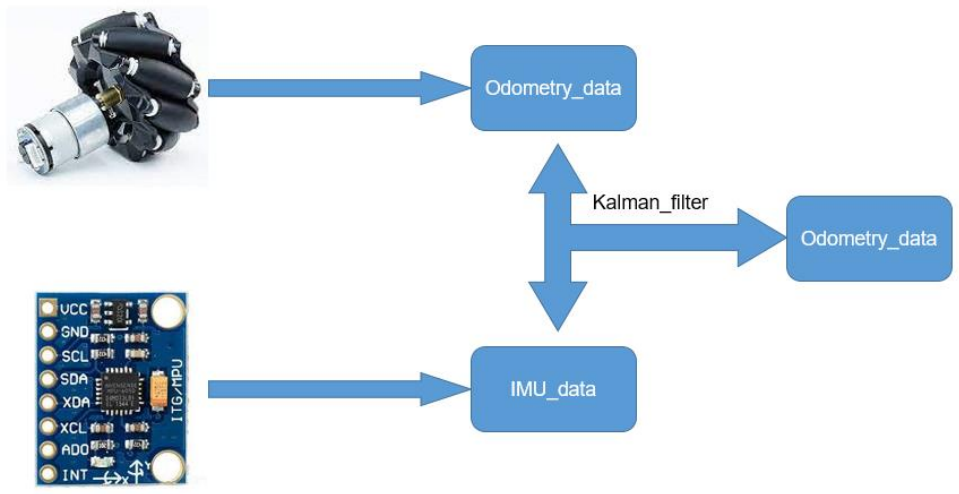


***Figure 3.22**: Logic table of odometry*

## 3.3. Mapping

### 3.3.1 Lidar

LiDAR (an acronym for Light Detection and Ranging) is a technology that allows to scan and map any environment by sending light beams and measuring the time it takes them to return to the source. Light Detection and Ranging (LiDAR) is a remote sensing technology that uses laser light to measure distances. LiDAR has gained significant attention due to its high precision and versatility in applications such as robotics, autonomous vehicles, mapping, and environmental monitoring.

- Advantages:

+ High Accuracy: Provides precise distance measurements, often within millimeter-level accuracy.

+ Fast Scanning Rates: Capable of capturing hundreds or thousands of data points per second.

+ Compact and Lightweight: Easily integrated into mobile and compact systems, such as robots and drones.

-Limitations:

+ Susceptible to Environmental Factors: Performance can be affected by dust, rain, fog, or reflective surfaces.

+ Power Consumption: Requires a stable power source, which can be challenging for battery-operated systems.

+ High cost: While prices have decreased, high-quality LiDAR systems can still be expensive.

- RPLIDAR A1 contains a range scanner system and a motor system. After power on each sub-system, RPLIDAR A1 start rotating and scanning clockwise. User can get range scan data through the communication interface (Serial port/USB).

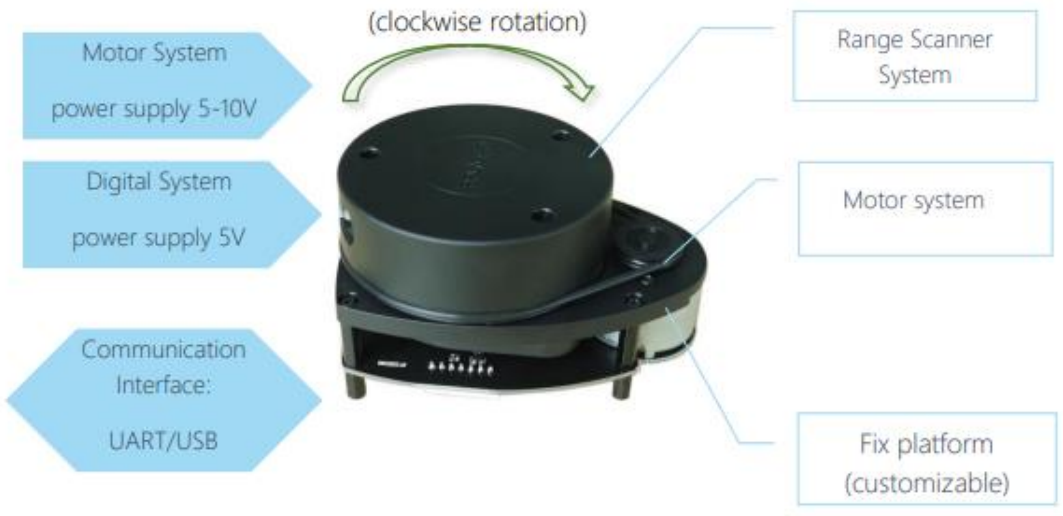
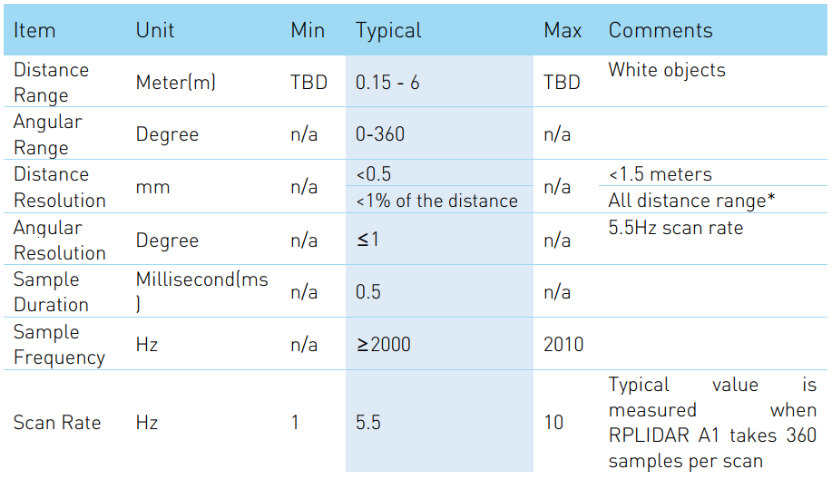


***Figure 3.23***: *RPLidarA1[31]*

- RPLIDAR A1 comes with a speed detection and adaptive system. The system will adjust frequency of laser scanner automatically according to motor speed. And host system can get RPLIDAR A1's real speed through communication interface.

***Table 3.2:*** *RPLIDAR A1 specifications [31]*



| Item | Unit | Min | Typical | Max | Comments |
|---|---|---|---|---|---|
| Distance Range | Meter(m) | TBD | 0.15 - 6 | TBD | White objects |
| Angular Range | Degree | n/a | 0-360 | n/a | |
| Distance Resolution | mm | n/a | <0.5 | n/a | <1.5 meters |
| | | | <1% of the distance | | All distance range* |
| Angular Resolution | Degree | n/a | ≤1 | n/a | 5.5Hz scan rate |
| Sample Duration | Millisecond(ms) | n/a | 0.5 | n/a | |
| Sample Frequency | Hz | n/a | ≥2000 | 2010 | |
| Scan Rate | Hz | 1 | 5.5 | 10 | Typical value is measured when RPLIDAR A1 takes 360 samples per scan |

- RPLIDAR is based on laser triangulation ranging principle and uses high-speed vision acquisition and processing hardware developed by SLAMTEC. The system measures distance data in more than 2000 times' per second and with high resolution distance output (<1% of the distance). RPLIDAR emits modulated infrared laser signal and the laser signal is then reflected by the object to be detected. The returning signal is sampled by vision acquisition system in RPLIDAR A1 and the DSP embedded in RPLIDAR A1 start processing the sample data and output distance value and angle value between object and RPLIDAR A1 through communication interface.

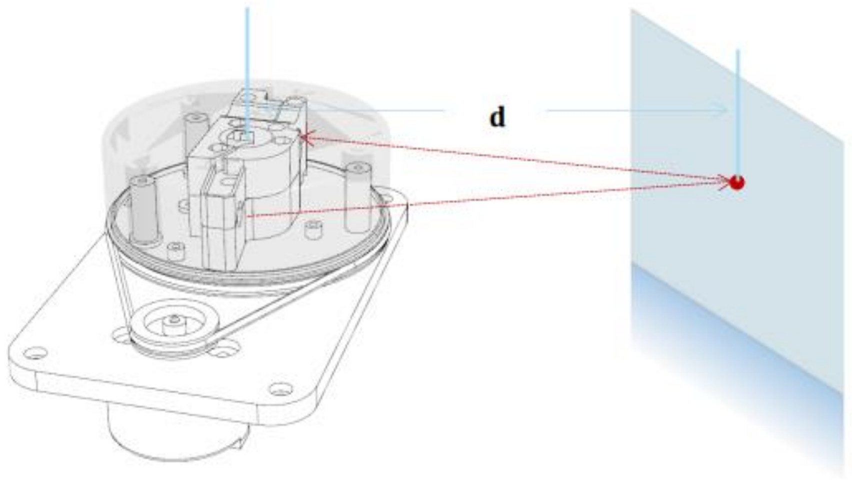


***Figure 3.24:*** *RPLidar A1 working schematic [31]*

***Emission***: The laser is shotted from the emitter every period of time. This pulse lasts for an extremely short duration, often in the range of nanoseconds.

***Reflection***: These pulses hit objects in the environment (such as walls, trees, or vehicles) and reflect back to the LIDAR sensor.

***Time of Flight (TOF)***: The time taken for the laser pulse to return is recorded by the lidar, then now by detect how long the laser have gone, the lidar can find out how much distance the laser have traveled

Using the speed of light($c = 3 \times 10^8 \ m/s$) and time for calculation:

$$d(distance) = \frac{c \times t}{2} \qquad [43]$$

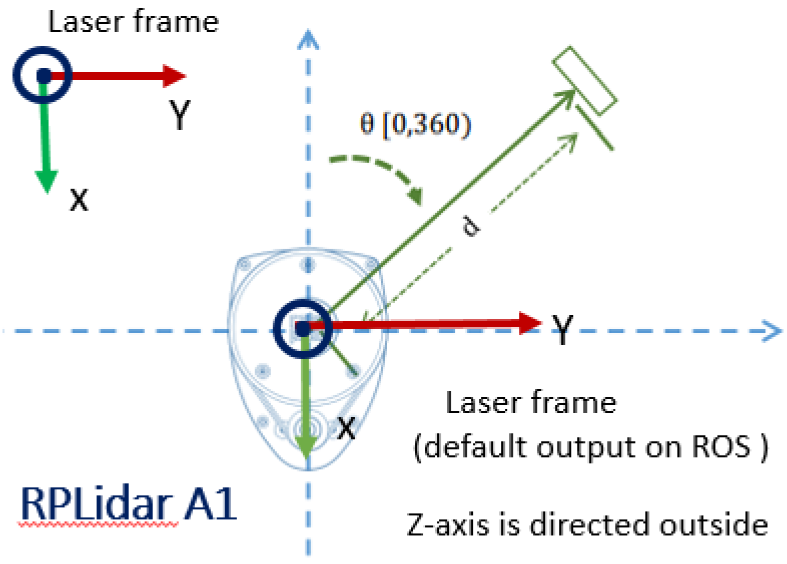


***Figure 3.25****: Data description of Lidar [31]*

The scanner mechanism rotates the laser, allowing measurements at different angles. This can provide a 360° or a specific field of view. With the combination of angular postioning in Lidar, the 2D lidar can easily scanning the surounding environment in 2 sided form. Eventually, the processor combines distance and angular data to make a 2D map.

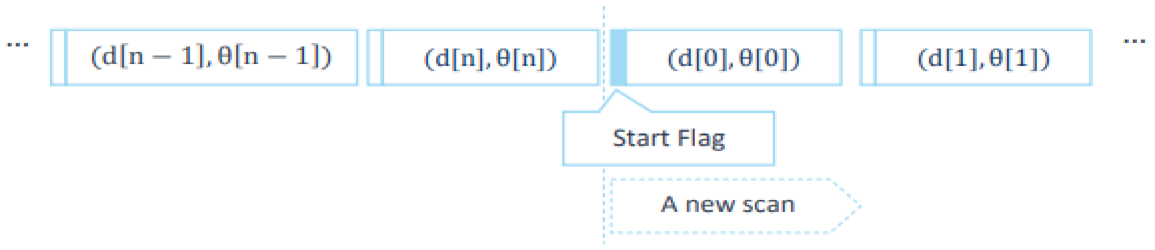


***Figure 3.26*** *: The RPLIDAR A1 Sample Point Data Frames [31]*

***Table 3.3:*** *RP LIDAR A1 specs [31]*

| Data types | Unit | Description |
|---|---|---|
| Distance | mm | Current measured distance value between the rotating core of the RPLIDAR A1 and the sampling point |
| Angle | degree | Current heading angle of the measurement |
| Start Flag | (Boolean) | Flag of a new scan |

### 3.3.2 Occupancy grid

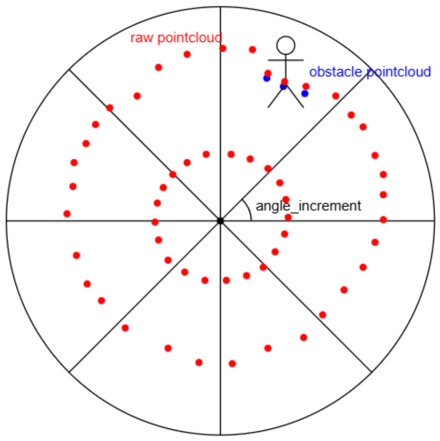


***Figure 3.27****: Lidar scanning [32]*

First of all, each of the pointcloud on the picture represent for an input data of lidar, the pointcloud are transformed into the polar coordinate centered around the center (lidar frame) and divided int circular bins per angle_increment respectively. At this time, each point belonging to each bin is stored as range data. Then by using these data, the processor could easily find out the position and width of the obstacle. Each of the point contain the following information:

- range data from origin of ray trace
- x (position of lidar frame) on map coordinate
- y (position of lidar frame) on map coordinate

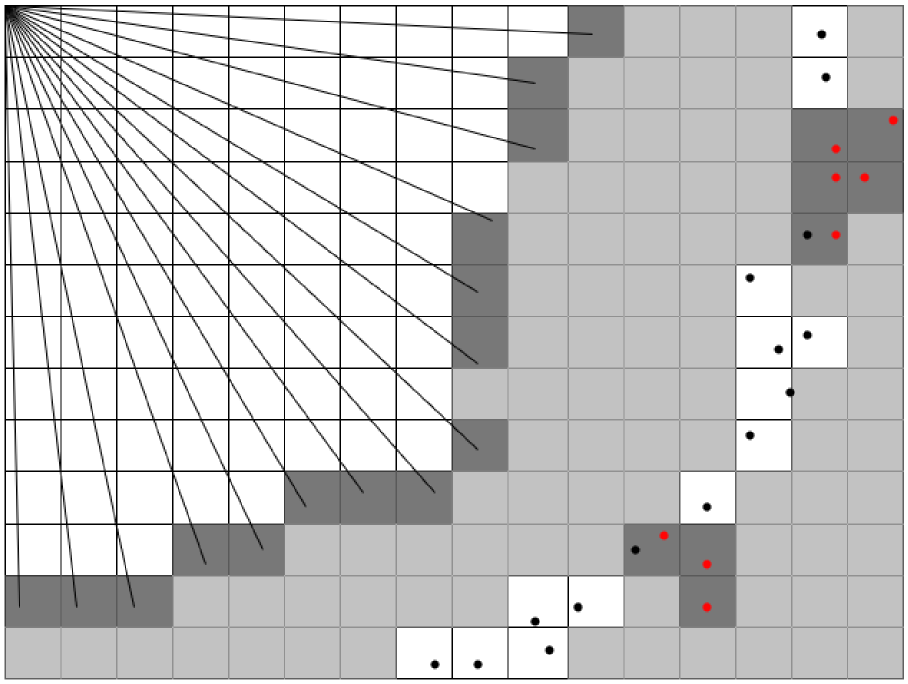


***Figure 3.28:*** *Occupancy grid map [32]*

Occupancy grid is a data structure commonly uses for robotics and specially in the area of autonomous mobile robot. Occupancy grid is design to represent the represent a working  environment of the robots as a structured grid of cells, where each cell's value reflects the likelihood of it being occupied, free, or unknown. Using occupancy grid the robot can understand the exact position of them on the map, then now they can know where can they go.

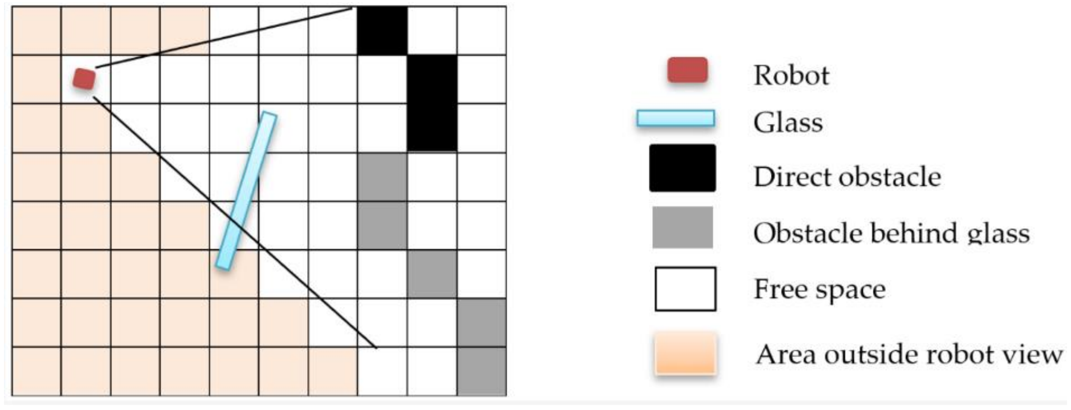


***Figure 3.29***: *State of the cells in occupancy grid [33]*

The figure State of the cells in occupancy grid as illustrated [fig 3.29] illustrates how conventional grid mapping algorithms work in a glass environment. The orange square represents where the robot poses, and the blue rectangle represents the presence of glass in the environment. Obstacles that are directly hit by the LiDAR lasers are represented by black boxes, whereas the grey boxes represent obstacles behind the glass. All grid areas found where the glass is located should be mapped as occupied space to obtain an accurate representation of the environment and avoid potential collisions. However, traditional occupancy grid mapping fails to detect glass and identify its location.

### 3.3.3. Mapping

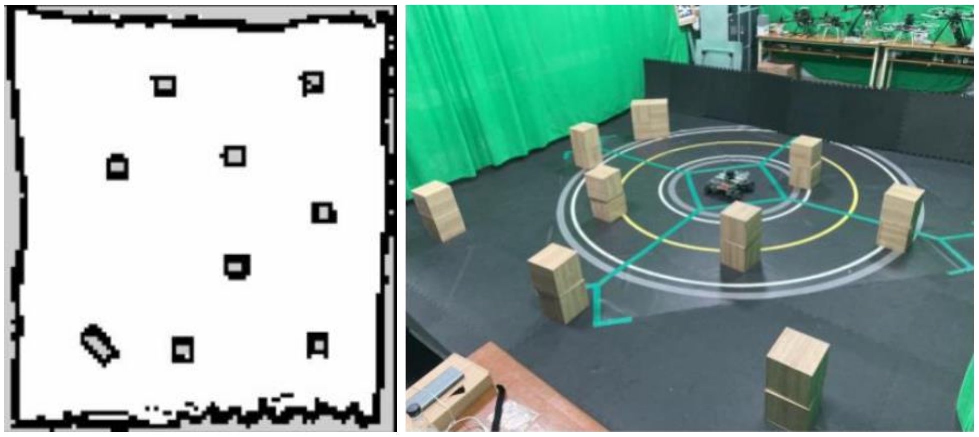
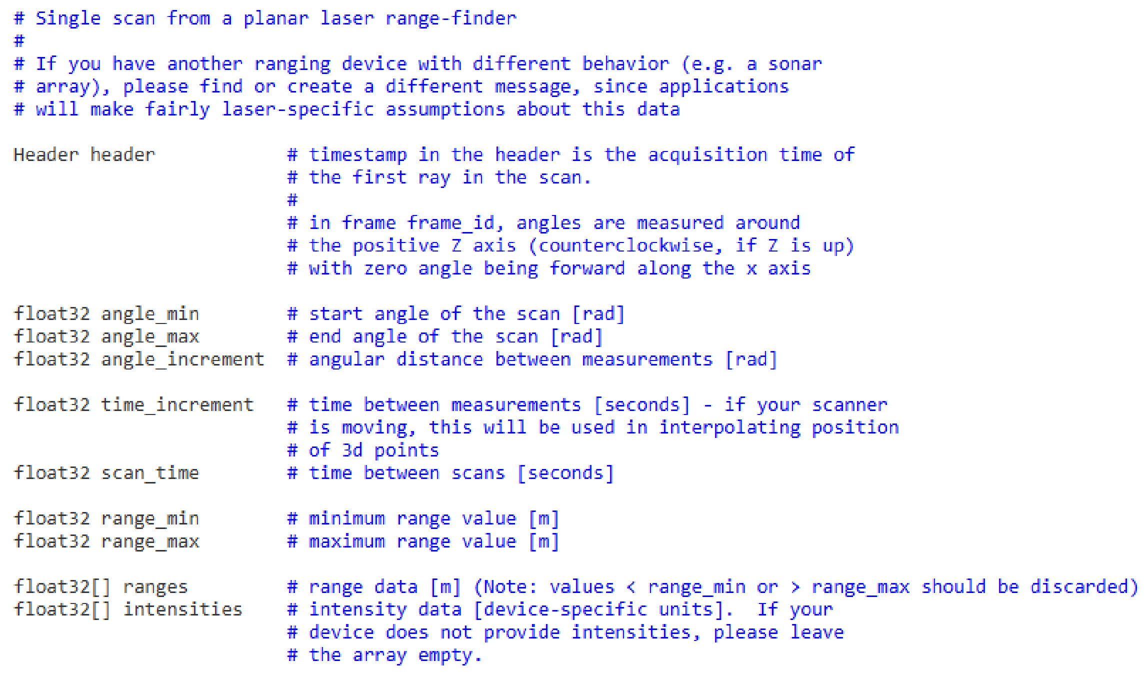


***Figure 3.30***: *Example of a map*

Mapping is the process of building a spatial representation of an environment, which robots or autonomous systems use for navigation, obstacle avoidance, and planning. It involves collecting data from sensors and converting it into a model of the environment, often in the form of a map. Simply understand, Mapping is the combination of Occupancy grid and lidar.

For a robot to be autonomous, an environmental map is a must.



```
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header            # timestamp in the header is the acquisition time of
                         # the first ray in the scan.
                         #
                         # in frame frame_id, angles are measured around
                         # the positive Z axis (counterclockwise, if Z is up)
                         # with zero angle being forward along the x axis

float32 angle_min        # start angle of the scan [rad]
float32 angle_max        # end angle of the scan [rad]
float32 angle_increment  # angular distance between measurements [rad]

float32 time_increment   # time between measurements [seconds] - if your scanner
                         # is moving, this will be used in interpolating position
                         # of 3d points
float32 scan_time        # time between scans [seconds]

float32 range_min        # minimum range value [m]
float32 range_max        # maximum range value [m]

float32[] ranges         # range data [m] (Note: values < range_min or > range_max should be discarded)
float32[] intensities    # intensity data [device-specific units].  If your
                         # device does not provide intensities, please leave
                         # the array empty.
```

***Figure 3.31***: *Lidar message [34]*

Every second the scanning message is sent back to the embedded computer ten times. The figure up there show how the lidar senser message represent. For every time the robot get the scanning message, the program would take, process and update the current map.

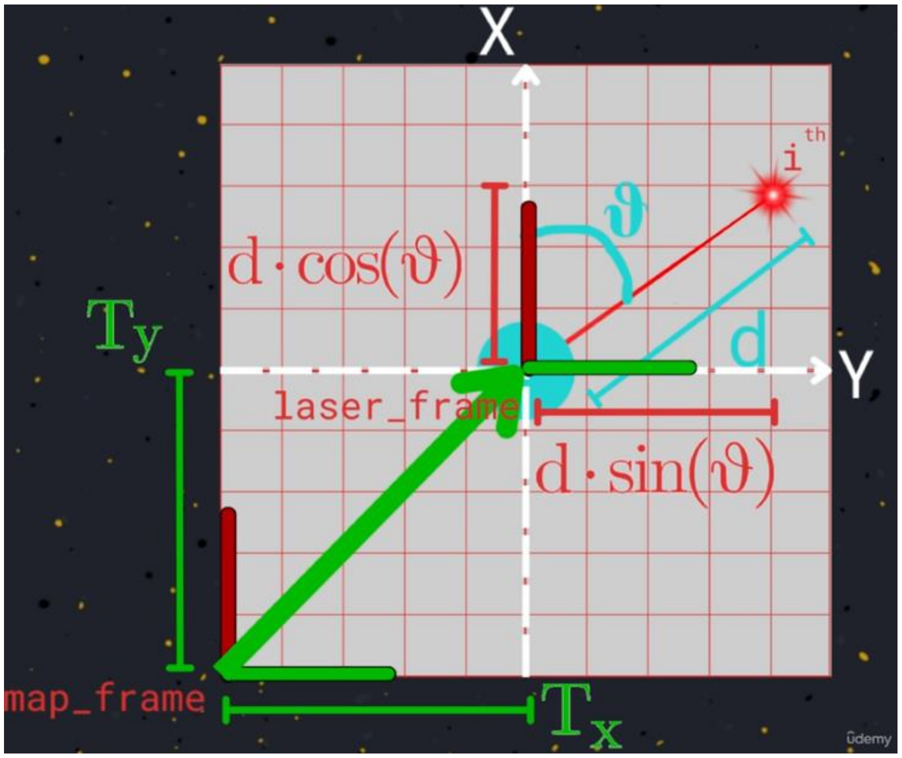


***Figure 3.32****: Position of lidar frame [35]*

$$px = d \times \cos(\vartheta + \varphi) + Tx$$

$$py = d \times \sin(\vartheta + \varphi) + Ty$$

Now the robot needs to find the position of the scanning obstacles, which receive by the lidar. With each point the robot would do the calculation to find out "where that point is". Then the program would make a loop until the last point of the scan messege complete. The sample would take 10 times every 1 second to ensure the legitimate of the mapping data. Besides, the taking sample frequency could enhance to make the map more accurate, but that should come with the right hardware and a faster processor.

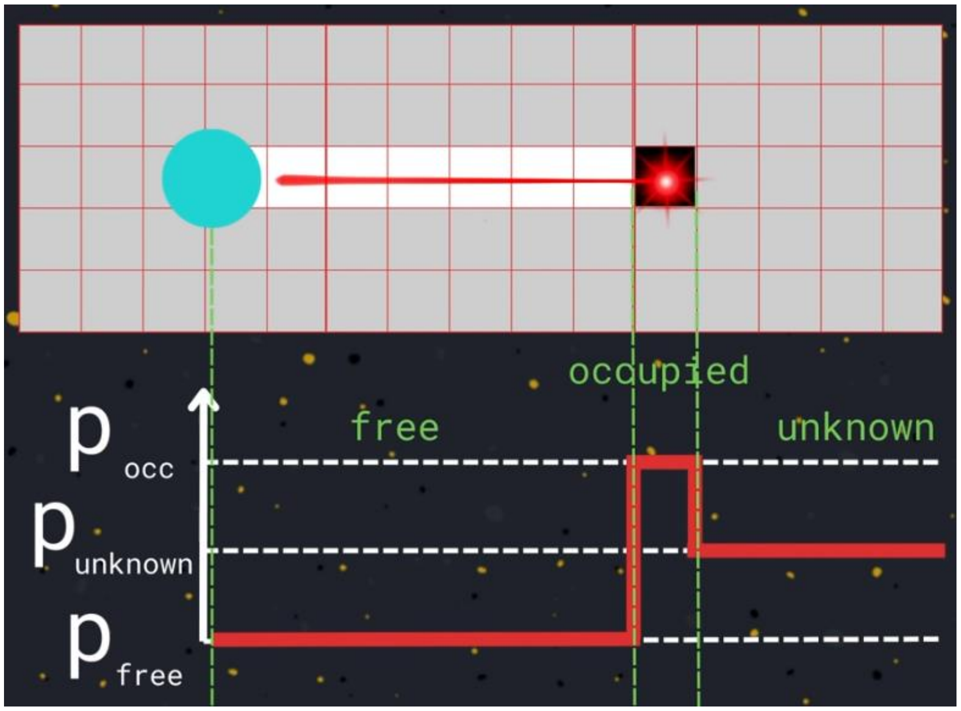


***Figure 3.36**: Marking and Clearing cells [35]*

Marking: LIDAR scans provide distance measurements. Any detected object within a certain range can be marked as an **occupied cell** on the map. Then, the program converts the LIDAR scan data into the map's coordinate system using the robot's current pose. Mark the cells along the detected obstacle's location in the occupancy grid.

Clearing: LIDAR rays that reach their maximum range indicate free space except for the last cells. Update these cells in the occupancy grid to mark them as free.

Cells can be marked as:

- Free (0): Indicates no obstacle.

- Occupied (100): Indicates an obstacle.

- Unknown (-1): Indicates unmapped areas

Bresenham's line algorithm is a line drawing algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. It calculates the intermediate points of a line between two given points using only integer arithmetic, making it ideal for systems with limited computational power.

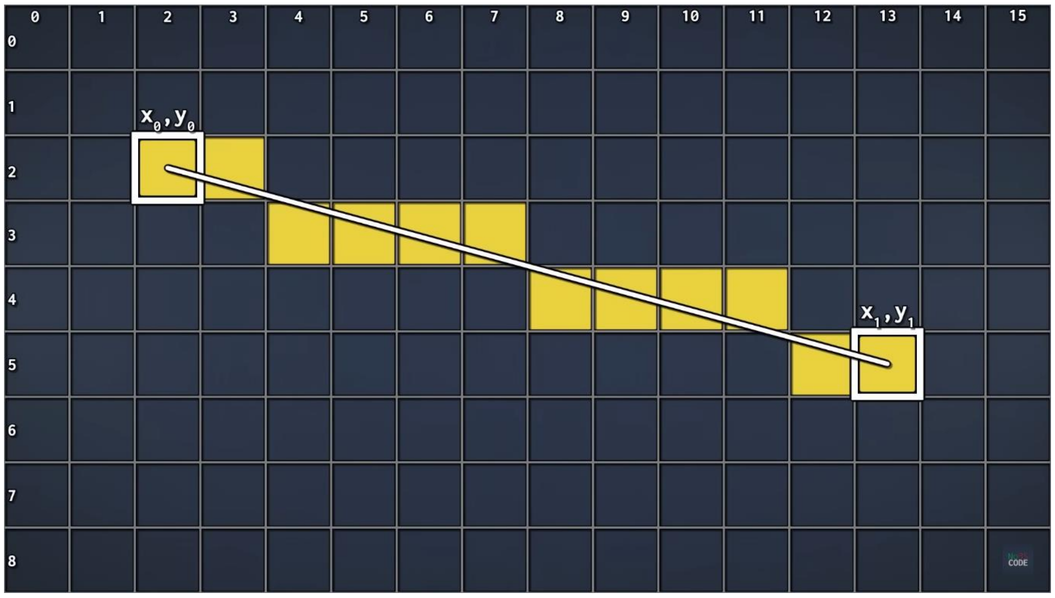


***Figure 3.37:*** *Bersenham's line [36]*

This simply could understand as how to draw a line when mapping. With the input is start point($x_0$, $y_0$) and end point($x_1$, $y_1$), the program would draw a line with in it. In mapping, the application of Bersenham's line algorithm is so vital because along with scanning data it will define how precise the map would be created.

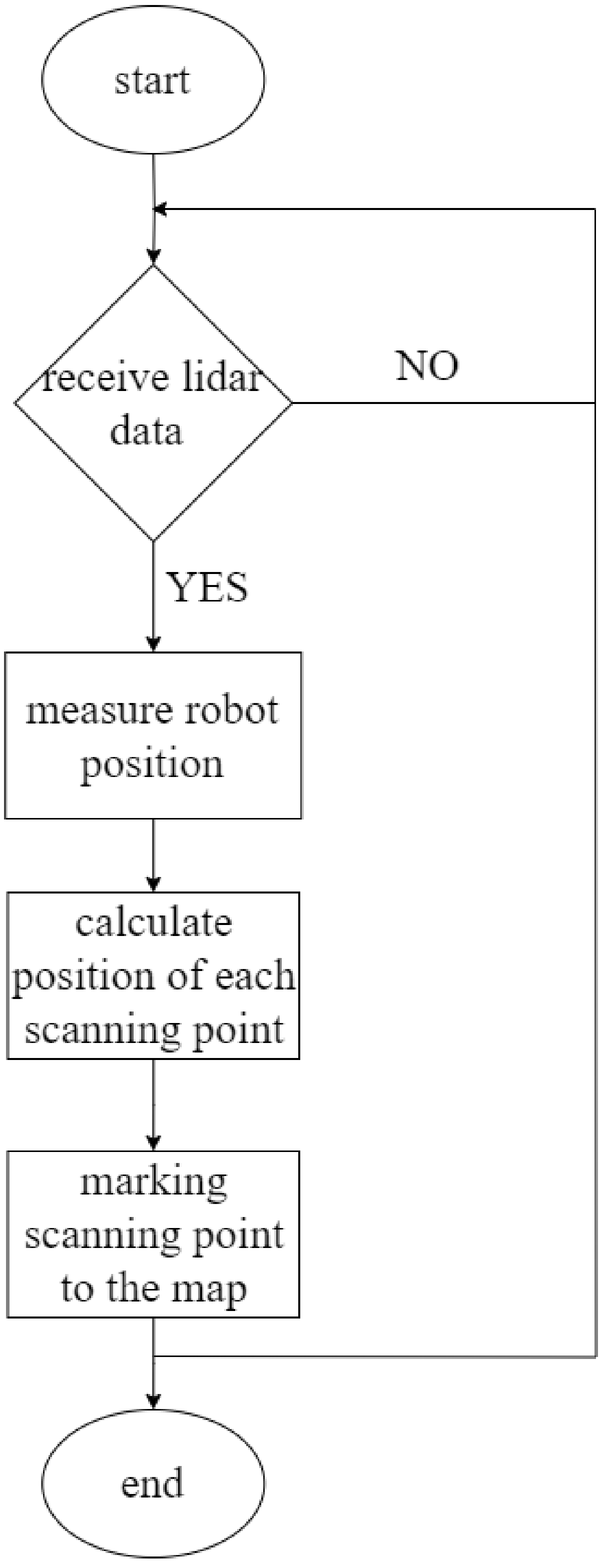

start
receive lidar data
NO
YES
measure robot position
calculate position of each scanning point
marking scanning point to the map
end

***Figure 3.38**: Mapping flow chart*

*Step 1- Receive data:*

- Objective: The program waits for incoming data from the LIDAR sensor before starting the mapping process.

- LIDAR sensors typically output data at a fixed frequency (7Hz). This data contains a series of distance measurements from the sensor to nearby obstacles, arranged radially. The program needs to subscribe to the sensor's topic and store this data as illustrated [fig 3.31]for processing.

*Step 2- Measure robot position:*

- Objectives: Determine the robot's pose (position and orientation) relative to the map frame as illustrated [fig 3.32].

- The robot's pose is crucial for mapping because it defines the reference point for all LIDAR scan measurements.

- The robot's position is described by three components:

+x, y: The robot's coordinates in the map frame.
+θ (theta): The robot's orientation (yaw angle) relative to the map frame.

*Step 3- Measure robot position:*

- Objective: The program would convert LIDAR scan data from the sensor's local frame as illustrated[fig 3.32]..

- To determine the global (map frame) position of each point scanned by the LIDAR sensor. This step converts raw LIDAR data from the sensor's local frame into coordinates that can be directly used to update the map.

*Step 4- Marking the scanning point to map:*

- Objective: Update the occupancy grid to reflect the presence of obstacles at the scanning points as illustrated [fig 3.28].

- To update a 2D occupancy grid map based on the positions of the scanned points. Each scanned point is used to indicate the presence of obstacles on the map, while free space along the LIDAR rays is optionally cleared.

*Step 5-Continuous Updates:*

- Objective: Repeatedly process new LIDAR data and robot poses to refine the map.

- The entire process (receiving data, measuring pose, transforming points, and marking the map) is executed in a loop at the LIDAR data frequency (10 Hz). This ensures the map is constantly updated as the robot moves and scans its environment.

## 3.4. Navigation

### 3.4.1 Path planning

Path planning is a cornerstone of robotics, enabling autonomous systems to navigate from a starting point to a target destination efficiently, safely, and reliably. Path planning can simply be understanding as finding the path from point A to point B with the shortest, most optimal or the least obstacle path. The path could be performed in a waypoint form and making the robot to follow the route during operation. Path planning require a map (2D or 3D) of the environment to define the starting point to the goal point. The map in this project performed as a occupancy grid.

After getting a map from the robot, the program now would make the calculation to find the path to the setting point. Path planning is used in various application from robotics, AI to video games. One of the most optimal ways for path planning is A* algorithm.

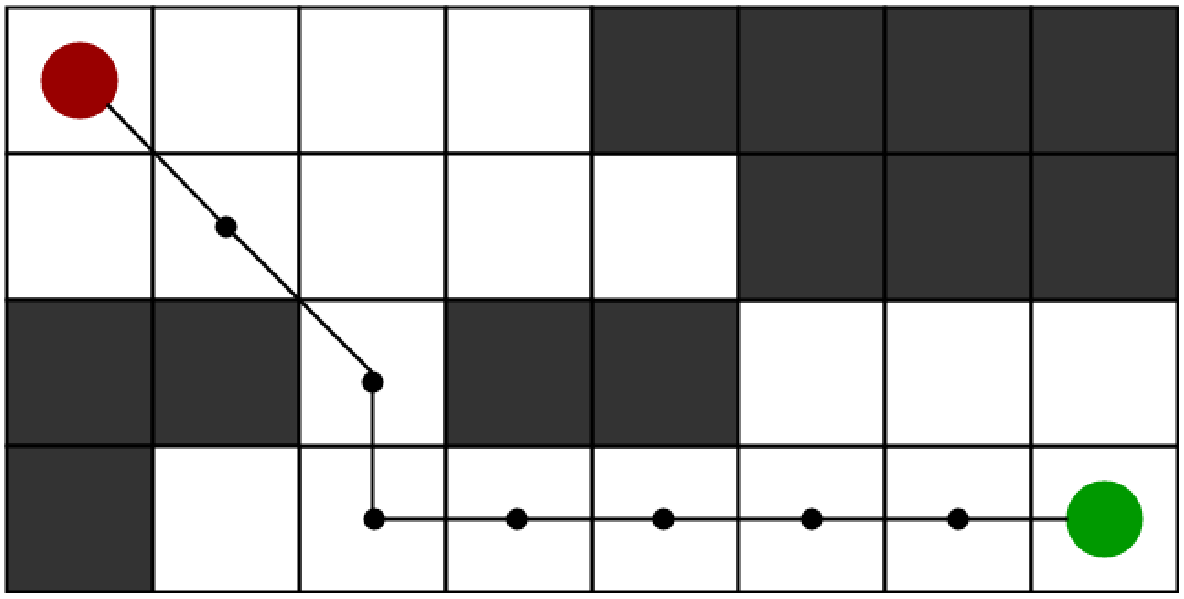


***Figure 3.39****: Path planning [37]*

A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals. In other word, A* algorithm has a better speed and smarter than others conventional path finding algorithm. The A* search algorithm is also known as a global optimization and state space heuristic algorithm. It can be seen as an improved version of the Dijkstra algorithm with the addition of an evaluation function [1].

The A* algorithm is typically used for pathfinding in 2D environments. In the A* algorithm, each grid point (or node) contains two main pieces of information: its position in 2D space (usually represented by x and y coordinates) and a status indicating whether the node is passable or blocked.

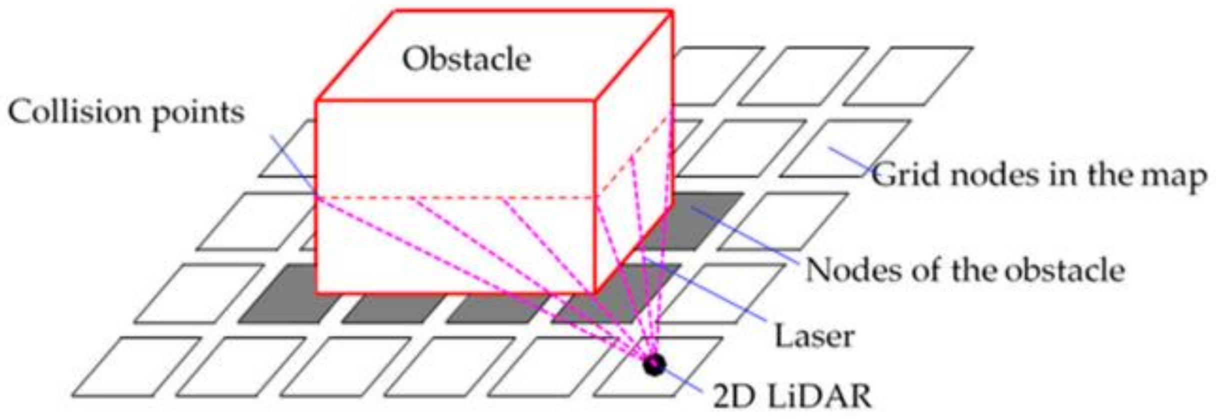


***Figure 3.40****; Obstacle judgment of the A* algorithm ([38])*

When using the A* algorithm for navigation in 2D, a 2D map is required, which is often generated using 2D LiDAR data. The LiDAR sensor scans the environment in two dimensions, providing distance measurements that are used to create a map that marks areas as either free space or obstacles. This map is then utilized by the A* algorithm to find the shortest path from a start location to a destination.

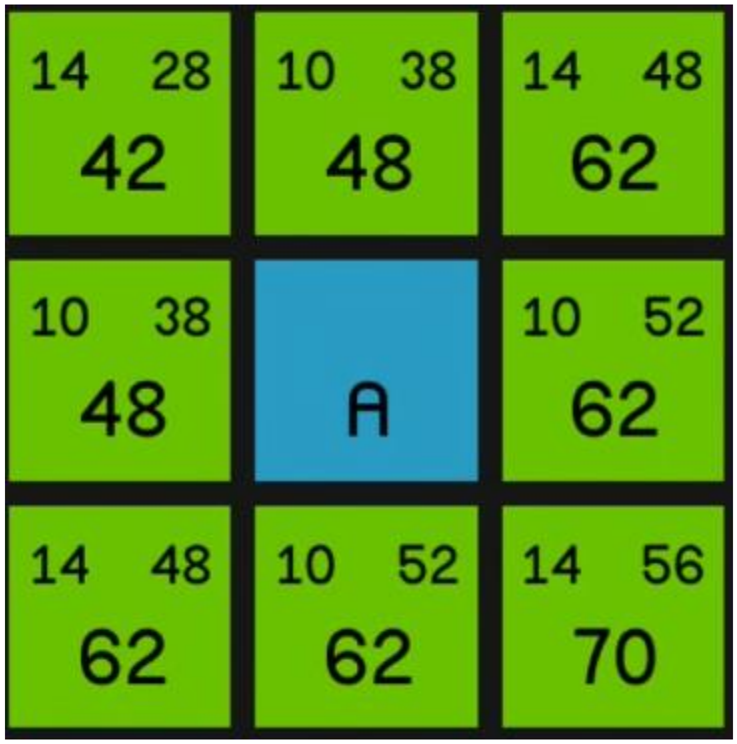


***Figure 3.41a**: Evaluation of the next cells [39]*

Take block with the number o

**The evaluation function of the A* algorithm is as follows:**

$$f(n) = g(n) + h(n)$$

where f(n) is the estimated cost of arriving at the target node from the initial node through node n. g(n) is the actual cost for travelling from the initial node to node n in the state space. h (n) represents the cost of estimating the optimal path from node n to target node [1]. The algorithm would choose the block that have the least f(n) value, if there are 2 blocks with the same value of f(n), the program would pick the one which has the least value of h(n). Besides, the more h(n) close to the real value, the better the efficiency of the algorithm.

In the *figure 3.41a* take the block with the number 42 as an example. The block A (the starting point) would look around the find the block for its next move with the least value. The value of the block defined by the combination of the travelling cost of that block from point A and the travelling cost of that block from point B. The function could define as:

$$f(1)\text{-}42 = g(1)\text{-}14 + h(1)\text{-}28$$

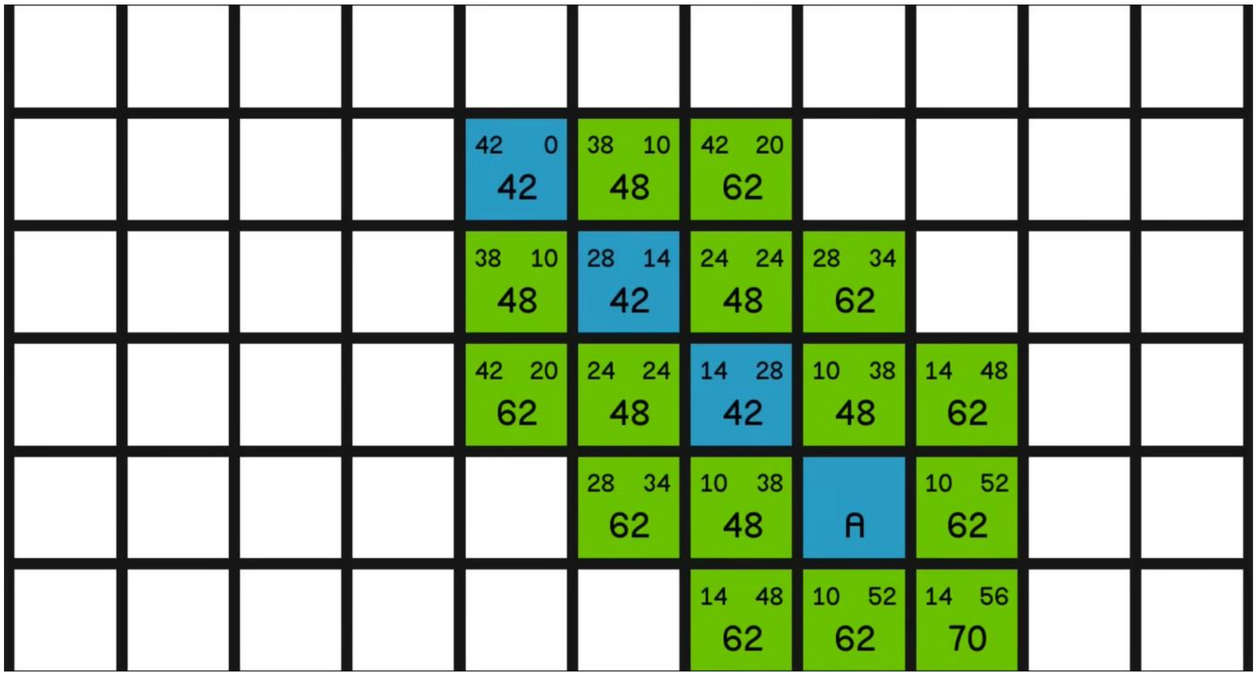


***Figure 3.41b**: Example of how A* algorithm work [39]*

In other next movement of the algorithm the program would keep picking up the least travelling cost block until it reach to point B(the final point).

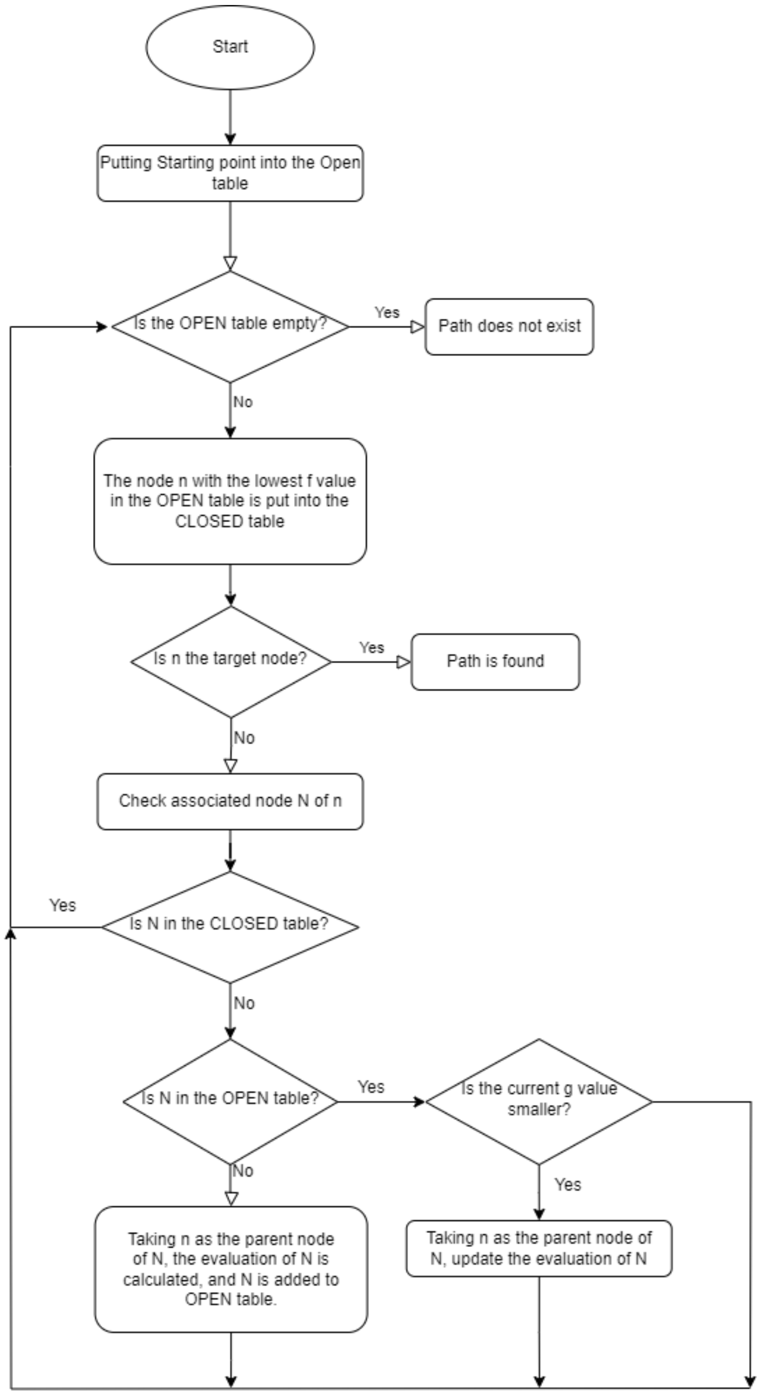

Start
Putting Starting point into the Open table
Is the OPEN table empty?
Yes
Path does not exist
No
The node n with the lowest f value in the OPEN table is put into the CLOSED table
Is n the target node?
Yes
Path is found
No
Check associated node N of n
Yes
Is N in the CLOSED table?
No
Is N in the OPEN table?
Yes
Is the current g value smaller?
No
Yes
Taking n as the parent node of N, the evaluation of N is calculated, and N is added to OPEN table.
Taking n as the parent node of N, update the evaluation of N

***Figure 3.42****: A* algorithm flowchart [40]*

### 3.4.2 Path Following

If mobile robot attempts to precisely follow the planned path, the path following can be accomplished by setting many numbers of waypoints on the path. However, in this case, it take a long to reach the target point because the increased number of waypoint leads to the acceleration/ deceleration of robot speed[2]. For a robot to move efficiently, the precision to following the path and the time it takes to reach the waypoint need to be take into consideration.

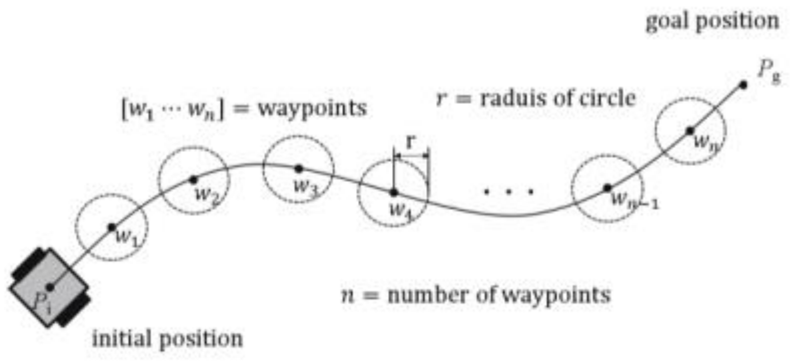


***Figure 3.43:*** *Path following using way point [41]*

For a robot to stick to a path waypoints are needed to be set. In mobile robotics, achieving precise path following necessitates balancing travel time and tracking error. While increasing the number of waypoints can enhance accuracy by providing more reference points, it may also extend travel time due to frequent adjustments and potential reductions in speed.

In waypoint-based navigation, a mobile robot sequentially traverses a series of waypoints from an initial position $p_i$ to the goal postion $p_g$. Each waypoint is typically associated with a circular region of radius $r$. Upon entering this region, the robot updates its target to the next waypoint.

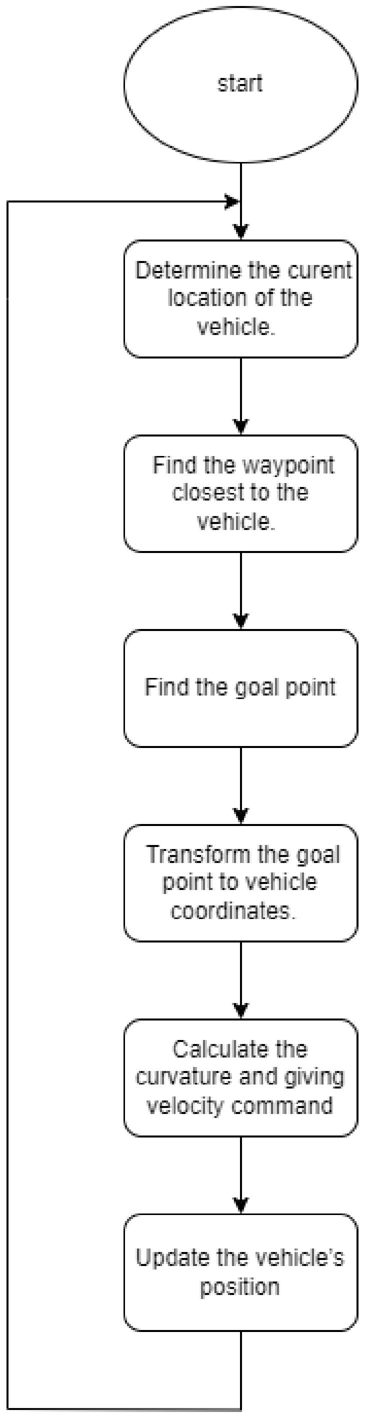

start
Determine the curent location of the vehicle.
Find the waypoint closest to the vehicle.
Find the goal point
Transform the goal point to vehicle coordinates.
Calculate the curvature and giving velocity command
Update the vehicle's position

***Figure 3.44***: *Path flowing flow chart*

### Determine the Current Location of the Vehicle

Obtain the vehicle's current position as (x, y, θ) in global coordinates.

This position is referenced from the initialization point, which serves as the global reference frame.

**Find the Path Point Closest to the Vehicle**

- Identify the point on the path nearest to the vehicle’s current position.

- This point serves as the starting reference for finding the next "goal point."

- Only consider points within the predefined *lookahead distance*, as these are within reach for steering.

**Find the Goal Point**

- Move forward along the path starting from the closest point.

- Identify the first path point that is approximately one *lookahead distance* away from the vehicle’s current location in global coordinates.

- The goal point defines the target position for the vehicle to steer toward.

**Transform the Goal Point to Vehicle Coordinates**

- Convert the global coordinates of the goal point into the vehicle's local coordinate frame.

- This ensures that calculations for curvature and steering commands align with the vehicle’s orientation.

**Sending Velocity command**

- Then the robot making the calculation to tracking to the waypoint

- Sending velocity commands to a mobile robot, specifically controlling its linear velocities in the X and Y directions and its angular velocity around the Z-axis.

**Update the Vehicle’s Position**

- Simulate the movement by updating the vehicle’s position and heading based on the steering command and the vehicle's kinematic model.

- This step is essential for simulation and real-world application to assess the effectiveness of the commands.

### 3.4.3 Navigation2 Stack

Nav2 is the professionally-supported successor of the ROS Navigation Stack deploying the same kinds of technology powering Autonomous Vehicles brought down, optimized, and reworked for mobile and surface robotics. This project allows for mobile robots to navigate through complex environments to complete user-defined application tasks with nearly any class of robot kinematics. Not only can it move from Point A to Point B, but it can have intermediary poses, and represent other types of tasks like object following, complete coverage navigation, and more. Nav2 is a production-grade and high-quality navigation framework trusted by 100+ companies worldwide.

It provides perception, planning, control, localization, visualization, and much more to build highly reliable autonomous systems. This will compute an environmental model from sensor and semantic data, dynamically path plan, compute velocities for motors, avoid obstacles, and structure higher-level robot behaviors

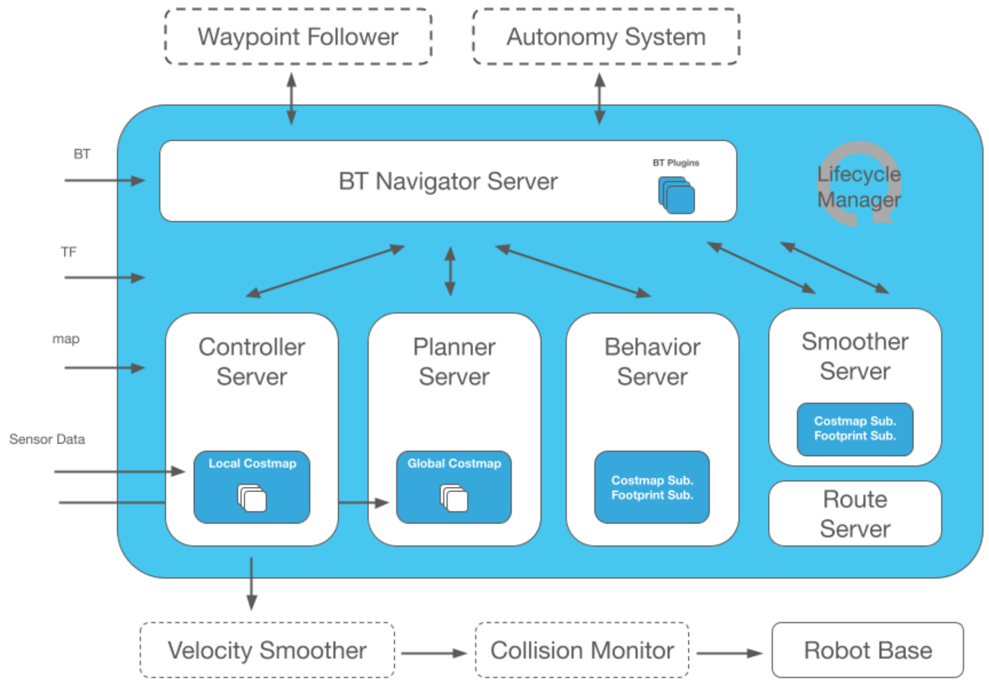


***Figure 3.45*** *Nav2 architecture [26]*

For the **Nav2** package to work effectively in autonomous robot navigation, several key input data sources are essential. The robot needs a **map** (either static or dynamic) that represents the environment, which is used for localization and path planning. **TF data** tracks spatial relationships between different parts of the robot and its environment, ensuring accurate positioning and movement. **Sensor data**, such as from **LiDAR, cameras**, and **IMUs**, provides real-time environmental information, enabling obstacle detection and the creation of a cost map.

Nav2 processes this input data to perform tasks like **localization**, where it determines the robot's position using algorithms like **AMCL. Path planning** is then performed to calculate the optimal route, taking into account obstacles and the robot's destination. **Obstacle avoidance** is continuously updated with real-time sensor data to prevent collisions. Finally, **velocity commands** are generated to control the robot's movement along the planned path.

To achieve optimal performance, the input data must be **accurate, timely**, and **consistent**. Inaccurate or delayed data can result in poor navigation, while inconsistent data can cause errors in robot movement. Proper data handling ensures that **Nav2** can navigate autonomously and safely in dynamic environments.

In this project, the Nav2 is use to perform several kinds of tasks for autonomous operation

# Chapter 4: RESULT

## 4.1 Mapping

When do mapping experiment, the first thing to do in the process is setup the environment, there are 2 kind of of environment in our experiment. One is a blank environment, the other is map with obstacles. In this experiment, group uses **slam toolbox** *[42]* as an algorithm for mapping

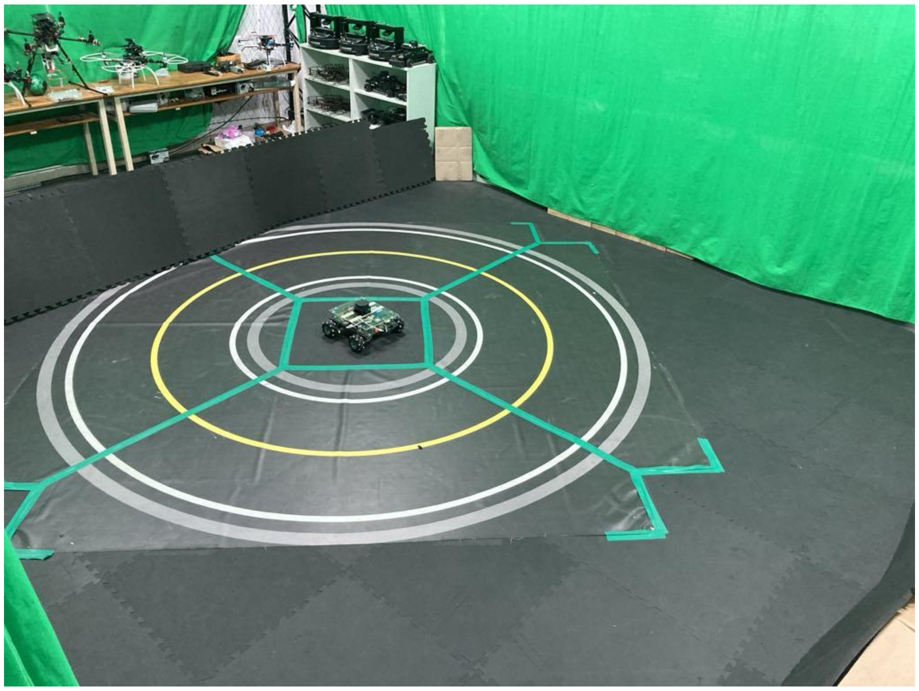


***Figure 4.1a****: Environment without obstacle*

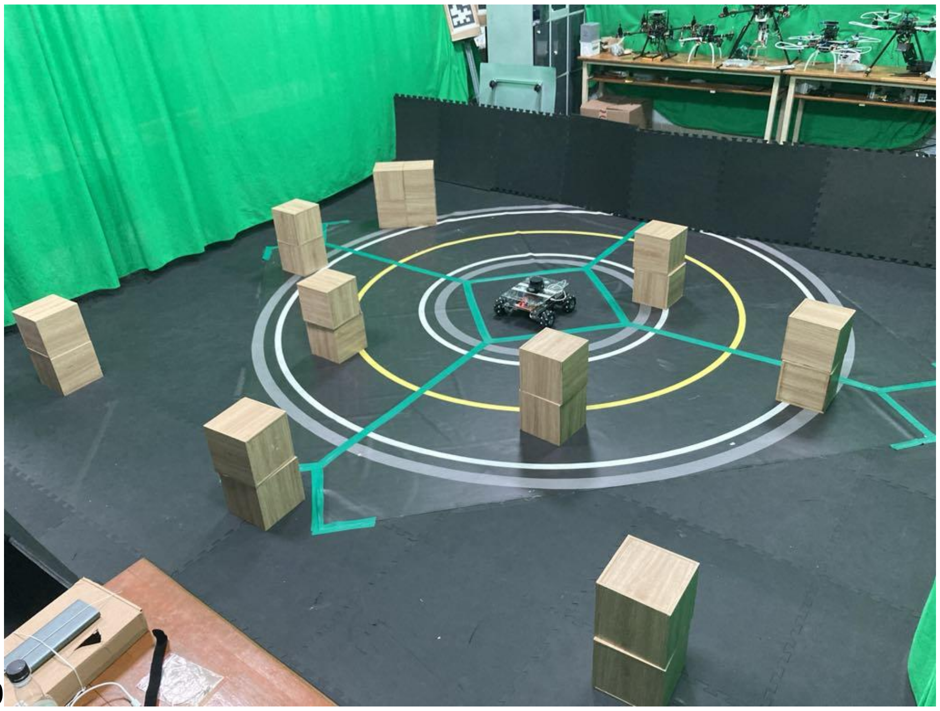
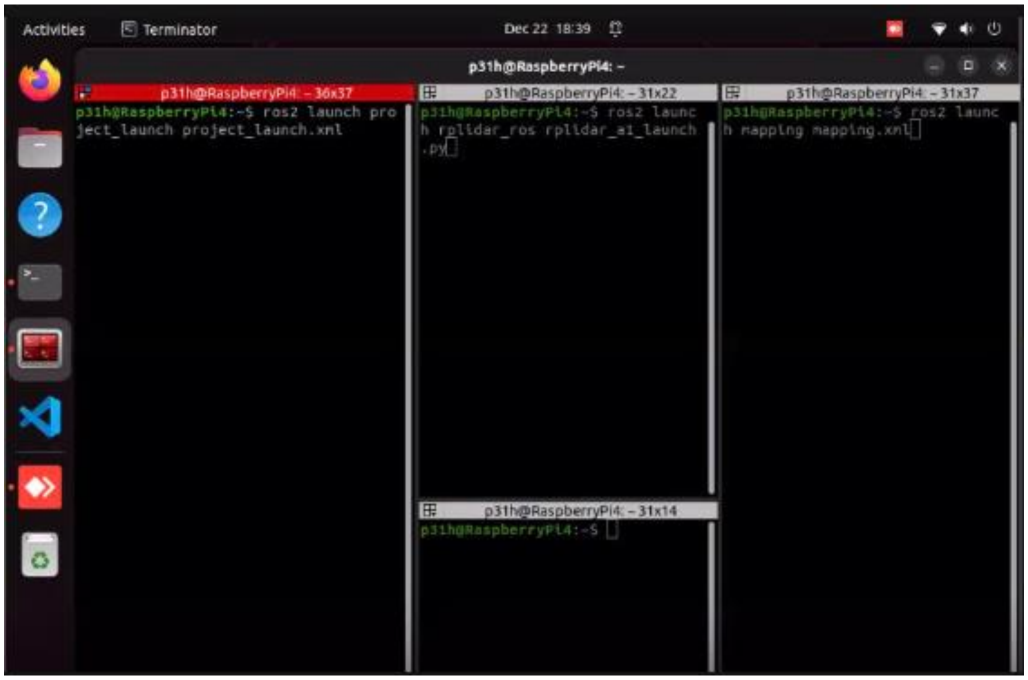
+99

***Figure 4.1b**: Environment with obstacle*

Then connect to the raspberry pi 4B by remote to initialize from the left to right and start the program. In the first table of terminal, the first order is to start the program and make the robot to spawn. In the second table, give the order to the program to activate the lidar sensor a long with reading it. In The last table, Starting the process for mapping.



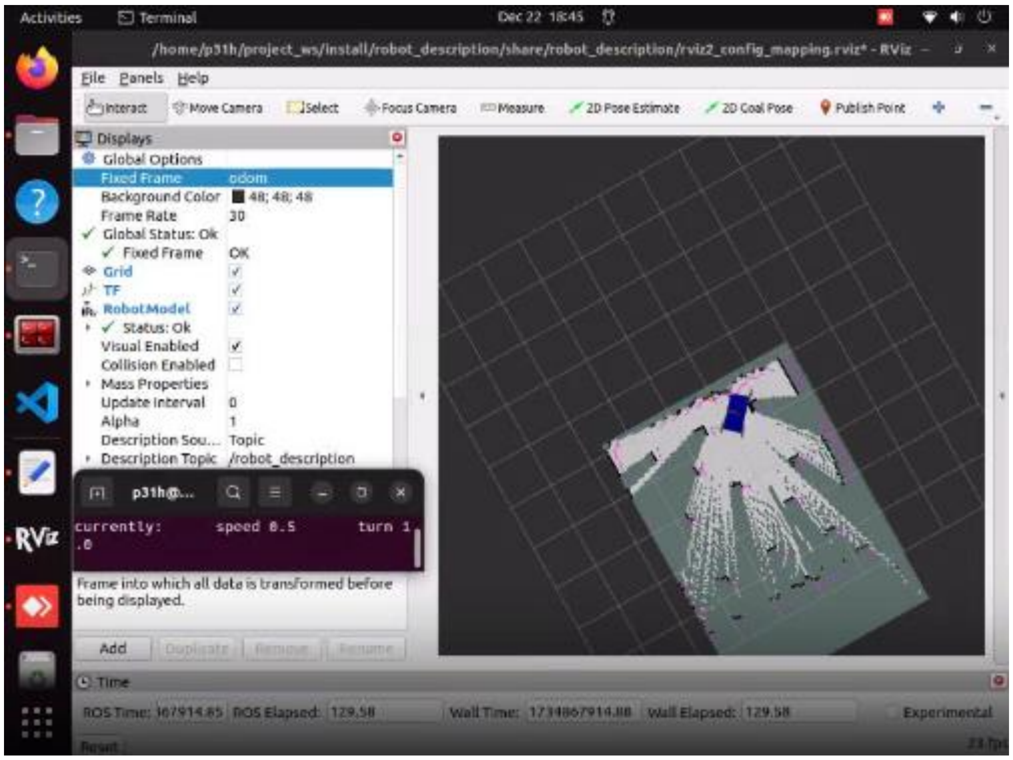
```
p31h@RaspberryPi4:~$ ros2 launch pro
ject_launch project_launch.xml
```

```
p31h@RaspberryPi4:~$ ros2 launc
h rplidar_ros rplidar_a1_launch
.py
```

```
p31h@RaspberryPi4:~$
```

```
p31h@RaspberryPi4:~$ ros2 launc
h mapping mapping.xml
```

***Figure 4.2a***: *Computer setup for mapping*



| | |
|---|---|
| Global Options | |
| Fixed Frame | odom |
| Background Color | 48; 48; 48 |
| Frame Rate | 30 |
| ✓ Global Status: Ok | |
| ✓ Fixed Frame | OK |
| Grid | ✓ |
| TF | ✓ |
| RobotModel | ✓ |
| ✓ Status: Ok | |
| Visual Enabled | ✓ |
| Collision Enabled | |
| Mass Properties | |
| Update Interval | 0 |
| Alpha | 1 |
| Description Sou... | Topic |
| Description Topic | /robot_description |

```
currently:      speed 0.5       turn 1
.0
```

Frame into which all data is transformed before being displayed.

ROS Time: 167914.85 ROS Elapsed: 129.58 Wall Time: 1734867914.88 Wall Elapsed: 129.58

***Figure 4.2b***: *Map when the robot spawn*

Then, when the robot is spawn, user would use keyboard to control the robot to move around. When moving around the robot would process and receive sensor from various sources from lidar to encoder and IMU in order to perform task like odometry, mapping. Then when complete save the map to folder.

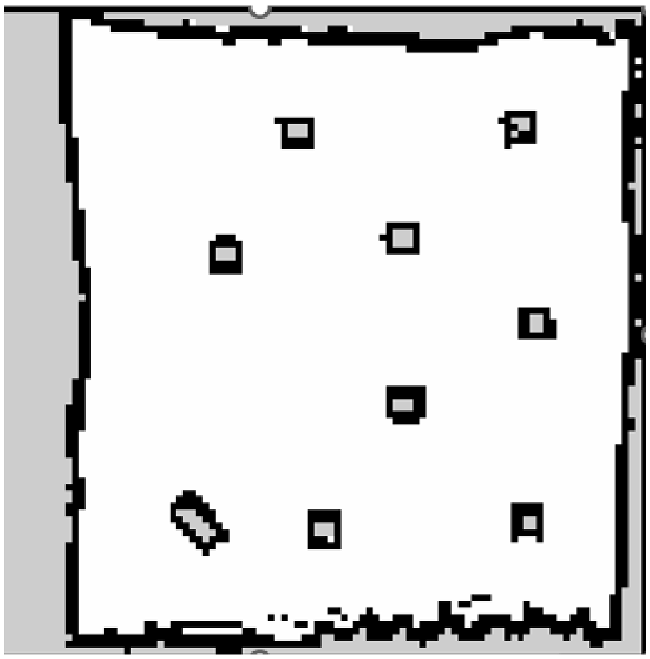


***Figure 4.2c**: The map with obstacle after finish mapping process*

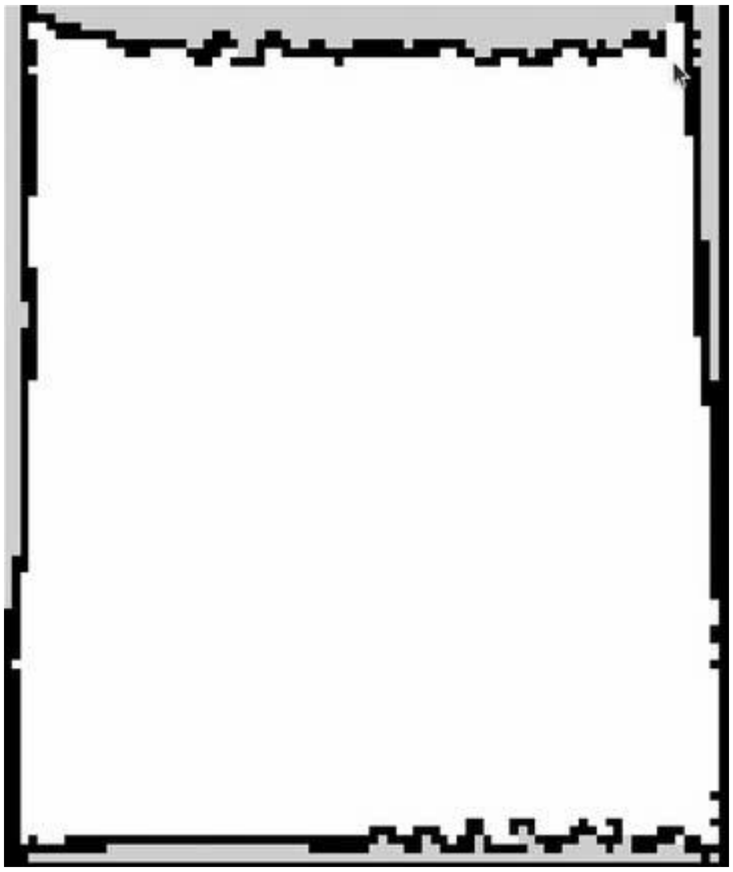


***Figure 4.2d**: The map without obstacle after finish mapping process*

The map after mapping process is an occupancy grid with 100cm$^2$ for each block so the revolution of each cell is 0.1 m$^2$. The map performed in 3 value, Free cell, occupied cell and unoccupied cell.

## 4.2 Navigation

To perform navigation task, mapping (4.1) must be done beforehand. So for a robot to be autonomous in an indoor or in a warehouse. The robot must know about the surrounding environment, meaning where it should go, or where should not.

At first when before starting the program, it is necessary to make the set up for both the environment. In this experiment, we choose a blank environment to perform navigation task.

Then we set up for the program to run. The first to do is initialization of the robot and ensure everything go in the right way. Then, Starting the lidar sensor to scan the surrounding environment. Lastly, when the previous is right, now it time for starting the nav2 Stack to perform navigation tasks.

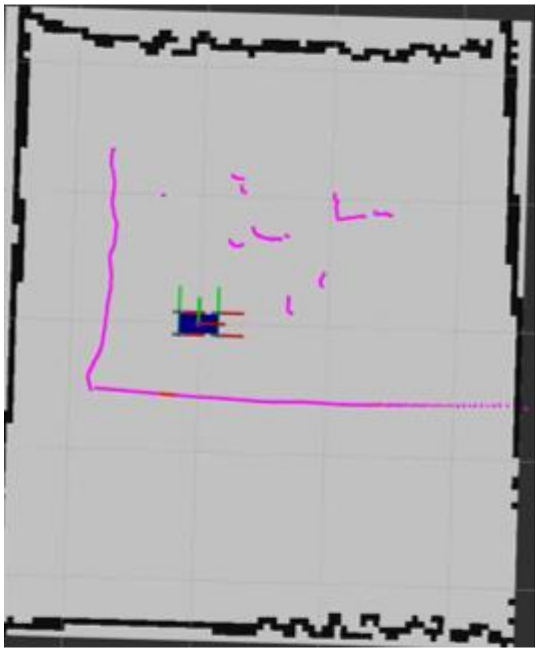


***Figure 4.3a****: Setting up the program*

Now we to provide or give the robot about the information of “where it is” on the map. This would allow the robot to know where exact it is on the map.

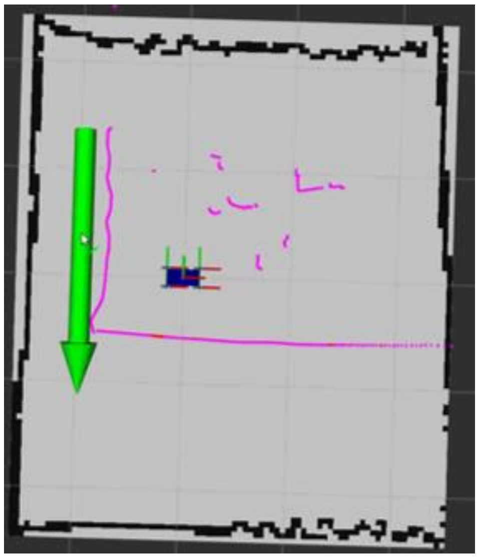


***Figure 4.3b**: Setting the original point*

Then, give the robot the order to move from point A (starting point) to point B. The given information includes the position and orientation of the robot. Next, the program would plan the path base on A* algorithm.

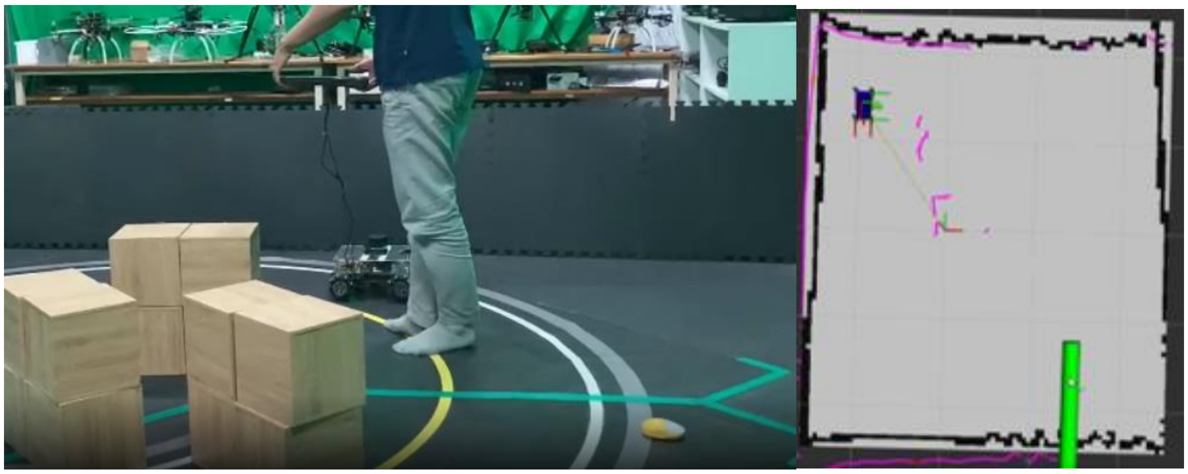


***Figure 4.3c**: Setting the goal point*

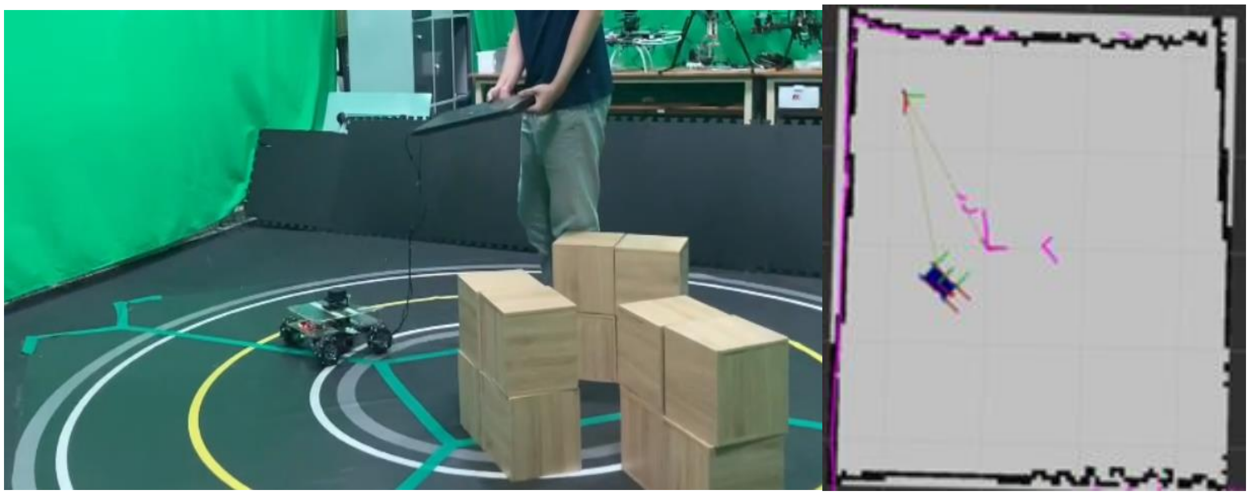


***Figure 4.3d:*** *On Running*

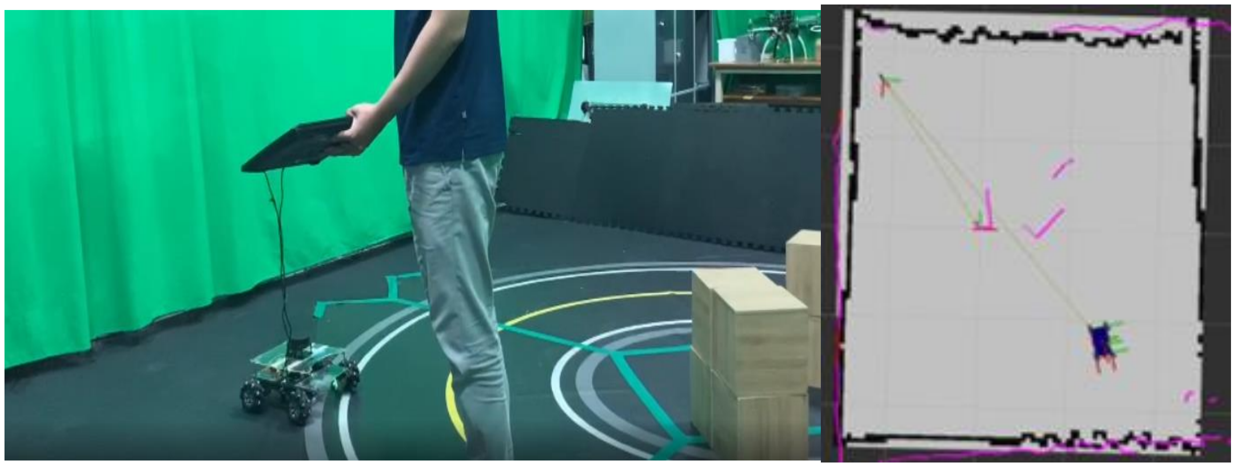


***Figure 4.3e****: Reach the goal*

Eventually, the robot hit the goal point. Then, it stops itself and sending to the computer that it has reach to the goal and wait for the next order to be implemented.

# Chapter 5: CONCLUSION AND IMPROVEMENT

## 5.1 Conclusion

**Achievement Summary**:

- Developed a fully functional indoor autonomous robot using 2D LiDAR.

- Integrated SLAM techniques for efficient mapping and localization in dynamic environments.
- Demonstrated effective navigation and obstacle avoidance using A* algorithm and Nav2 stack.

**Key Findings**:

- The robot successfully created accurate 2D occupancy grid maps and navigated through complex environments.
- Sensor fusion using encoders, and IMU significantly improved localization accuracy.

**Limitations**:

- Performance constraints due to hardware limitations (e.g., processor speed).
- Motion controller

## 5.2 Recommendations for Improvement

**Hardware Enhancements**:

- Upgrade the processor to handle more computationally intensive tasks.
- Consider integrating for enhanced obstacle detection and spatial awareness.

**Algorithm Optimization**:

- Improve SLAM algorithms to reduce computational load and latency.
- Experiment with dynamic planning algorithms like RRT* for better adaptability.
- Apply PIDs controller into motion control
- Self-building algorithms like path planning, path following, behavior tree and obstacle avoiding.

**System Scalability**:

- Implement a modular architecture for easier hardware and software upgrades.
- Explore real-time cloud-based processing to offload computational tasks.

**Future Research**:

- Investigate the integration of machine learning techniques for adaptive navigation.
- Expand testing to include diverse indoor environments such as crowded warehouses or complex office layouts.

**Reference:**

[1] Nguyễn, H. A. (2023). Ứng dụng xe tự hành vận chuyển hàng tự động trong kho thương mại điện tử. *Tạp chí Công Thương*

*[2] Trần, Q. T., & Nguyễn, T. T. (2020). Nghiên cứu và phát triển giải thuật lập kế hoạch đường đi cho robot di động. Tạp chí Khoa học và Công nghệ.*

*[3] Phạm, T. L. H., & Nguyễn, T. D. (2023). Xây dựng quỹ đạo cục bộ cho robot tự hành trong nhà lưới nông nghiệp trên nền tảng hệ điều hành ROS. Tạp chí Khoa học Nông nghiệp Việt Nam*

*[4] Trần, T. T., Trần, B. H., Đoàn, N. M., Sái, T. H., Đoàn, Q. K., & Trương, T. B. L. (2022). Thiết kế SLAM địa hình 3D cho robot di động di chuyển trong môi trường kín. Sinh viên nghiên cứu khoa học*

*[5] Phan, H. A., Bùi, D. N., Trần, H. Q. Đ., Vũ, T. Đ., & Nguyễn, T. T. V. (2020). Nghiên cứu phát triển robot xây dựng bản đồ và định vị đồng thời trên nền tảng ROS. Đại học Quốc gia Hà Nội*

*[6] Macenski, S., Martín, F., White, R., & Clavero, J. G. (2020). The Marathon 2: Navigation System. Open Navigation LLC*

*[7] Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics*

*[8] Shafiq, M. U., Imran, A., Maznoor, S., Majeed, A. H., Ahmed, B., Khan, I., & Mohamed, A. (2024). Real-time navigation of mecanum wheel-based mobile robot in a dynamic environment. Heliyon, 10(2), e02860*

*[9] Fedorenko, R., & Gurenko, B. (2016). Local and Global Motion Planning for Unmanned Surface Vehicle. EDP Sciences*

*[10] Salih, J. E. M., Rizon, M., Yaacob, S., Adom, A. H., & Mamat, M. R. (2006). Designing omni-directional mobile robot with Mecanum wheel. American Journal of Applied Sciences, 3(5), 1831–1835*

*[11] Juhari, M. R., Salih, J. E. M., & Abdul Hamid, M. H. B. (n.d.). Control of omni-directional Mecanum wheel mobile robot and analysis of artificial muscle. Pusat Pengajian Kejuruteraan Mekatronik, Kolej Universiti Kejuruteraan Utara Malaysia, 01000 Kangar, Perlis, Malaysia.*

*[12] Taheri, H., Qiao, B., & Ghaeminezhad, N. (2015). Kinematic model of a four Mecanum wheeled mobile robot. International Journal of Computer Applications, 113(3), 6–10*

*[13] Ioan Doroftei, Victor Grosu and Veaceslav Spinu (2007). Omnidirectional Mobile Robot - Design and Implementation, Bioinspiration and Robotics Walking and Climbing Robots, Maki K. Habib (Ed.), ISBN: 978-3- 902613-15-8, InTech*

*[14] RoboPeak Team, & Shanghai Slamtec Co., Ltd. (2016). RPLIDAR A1: Low cost 360-degree laser range scanner: Introduction and datasheet (Rev. 1). Shanghai Slamtec Co., Ltd.*

*[15] Nam, T. H., Shim, J. H., & Cho, Y. I. (2017). A 2.5D map-based mobile robot localization via cooperation of aerial and ground robots. Sensors, 17(11), Article 2612*

*[16] Alghodhaifi, H., & Lakshmanan, S. (2016). Autonomous vehicle evaluation: A comprehensive survey on modeling and simulation approaches. [IEEE], Volume 4, [10-14].*

*[17] Wang, S.-L. (2018). Motion control and the skidding of Mecanum-wheel vehicles. IJISET - International Journal of Innovative Science, Engineering & Technology, 5(5), [77-79]*

*[18] Tripicchio, P., Satler, M., Avizzano, C. A., & Bergamasco, M. (2014, September 15–17). Autonomous navigation of mobile robots: From basic sensing to problem solving. 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing, Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy. Scuola Superiore Sant'Anna*

*[19] Alhanov, D. S., & Rubtsov, V. I. (2020). Development of the laboratory work: "Modeling of a mobile robot on Mecanum wheels kinematics." ITM Web of Conferences, 35, 04001*

*[20] Fedorenko, R., & Gurenko, B. (201). Local and global motion planning for unmanned surface vehicle. MATEC Web of Conferences, 201, 06001*

*[21] Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (n.d.). Path planning and trajectory planning algorithms: A general overview. DIEGM – Dip. di Ingegneria Elettrica, Gestionale e Meccanica – University of Udine.*

*[21] Shafiq, M. U., Imran, A., Maznoor, S., Majeed, A. H., Ahmed, B., Khan, I., & Mohamed, A. (2024). Real-time navigation of mecanum wheel-based mobile robot in a dynamic environment. Elsevier Ltd.*

*[22] Röhrig, C., Heß, D., & Künemund, F. (2017). Motion controller design for a Mecanum wheeled mobile manipulator. Proceedings of the 2017 IEEE Conference on Control Technology and Applications (CCTA 2017), 27–30 August 2017, Kohala Coast, Hawai'i, USA.*

Link:
[23] https://www.itm-conferences.org/articles/itmconf/pdf/2020/05/itmconf_itee2020_04001.pdf

[24] https://www.youtube.com/watch?v=vxSK2NYtYJQ)

[26] https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html)

[27] https://www.mathworks.com/help/ros/gs/ros2-topics.html)

[28] https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Services/Understanding-ROS2-Services.html)

[29] https://sdrobots.com/ig3242-52-encoder-interfacing-cpr-calculation/

[30] https://www.researchgate.net/figure/Quadrature-encoding-with-hall-effect-sensors-Both-channels-will-output-square-waves-that_fig4_342577004)

[31] *RPLidarA1M8 datasheet*

*[32]* https://autowarefoundation.github.io/autoware.universe/pr-6268/perception/probabilistic_occupancy_grid_map/laserscan-based-occupancy-grid-map/

[33] *https://www.mdpi.com/1424-8220/21/7/2263)*

*[34]* https://docs.ros.org/en/noetic/api/sensor_msgs/html/msg/LaserScan.html)

[35] https://www.udemy.com/course/self-driving-and-ros-2-learn-by-doing-map-localization/?couponCode=ST12MT122624)

[36] *https://www.youtube.com/watch?v=CceepU1vIKo )*

*[37] https://www.geeksforgeeks.org/a-search-algorithm/*

*[39] https://www.youtube.com/watch?v=-L-WgKMFuhE*)

[40] *https://www.mdpi.com/1424-8220/19/13/2976)*

*[41] https://www.researchgate.net/figure/Path-following-method-of-mobile-robot-using-waypoints_fig1_289084212*

*[42] https://github.com/SteveMacenski/slam_toolbox*

*[43] https://felix.rohrba.ch/en/2015/an-introduction-to-lidar/#:~:text=Using%20the%20measured%20time%20t,%3D%20c%20*%20t%20%2F%202**.***