Research and Design Motion Planner for AGV 2022

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

FACULTY OF MECHANICAL – MECHATRONICS ENGINEERING

Student's name : VO HOANG QUOC BAO 1852269

TRINH PHUONG HIEU 1710085

Instructor : Le Thanh Hai

HO CHI MINH CITY, 2022

ACKNOWLEGDEMENT

Final thesis is an opportunity for students to apply what we have learnt at the University in order to solve practical problems. It is also an overview to approach and understand the characteristics of the tasks we will face after graduated. From our point of view, this thesis is a milestone in our journey to fulfill the theoretical knowledge and a necessary experience for me to prepare myself for working in an industrial environment.

We would like to express my deepest thank to Ho Chi Minh City University of Technology, the Faculty of Mechanical Engineering, and the Department of Mechatronics Engineering for allowing me to do the thesis and providing we with the best conditions to complete the thesis.

We would also like to express our appreciation to Le Thanh Hai – our instructor, for helping me for all these times to complete the project, as well as giving us constructive advice and comments to help me finish the project successfully.

Due to the lack of some important knowledge while learning and practicing, therefore errors made during the thesis are inevitable. We are looking forward to the sympathy of the instructor in charge, and we also look forward to receiving constructive comments from other instructors.

My sincere gratitude.

THESIS SUMMARY

This thesis aims to design fully system of Automated Guided Vehicle with perform duties for transport items in inhouse factories, warehouse, etc.

For AGV systems with multiple units, extensive guide path routes and the need to interface with your plant operating system. The AGV each communicate with this base station via Wi-Fi. The base station manages all of the units in the system and each guide path intersection to ensure the system operates efficiently. Movement ordered are buffered as required and the AGVs routed in the most safe and efficient manner.

Base station was built with typically feature as a screen for system visualization as well as an administrative interface (keyboards, mouses ....).

Every unit could be access by touchscreen, the plant associate responsible and easy customize from this location. Screen also show the operating parameter likely map and review the direction, battery and unit’s errors.

CONTENT

TABLE OF FIGURE

TABLE

CHAPTER 1: INTRODUCTION

1.1 Overall, about the automatic factories and AGV's fundamental role

1.1.1 Introduce about the category of factory

Nowadays, with the advancement of science and technology and competition in the market, the problems of transporting, loading and unloading and storing materials and goods become more and more important. Therefore, many companies and enterprises mechanize and automate their factories/warehouses. The appearance of automated mechanical machines such as pick-up elevators, automated robotic systems operating 24/7, management and storage systems help increase productivity and greatly reduce storage time and costs. store. [first]

Currently, an automated factory consists of many components, but they all have the following essential components (a factory can have one or more of the following components depending on the economic situation and the size of the factory) [2]:

- Automated storage/Retrieval system (AS/RS)
- Transport conveyor system
- Classification system
- Identification and inspection system
- Management software system
- A system of transport robots (AGVs)

1.1.2 AGV in the factory

An AGV (Automated Guided Vehicle) is a type of mobile robot that follows markers, paths on the floor, or uses a camera or laser. Today, AGVs are present in almost all industries and are applied in many fields, especially in manufacturing plants.

Figure 1 KIVA system in Amazons' warehouse

Figure 2 LEVIO in Toyota manufactured factory

- AGVs are used in a distribution center such as Amazon's Kiva system of AGVs (Figure 1.1)

- AGVs are used to transport goods and components from storage to predefined return locations. Toyota factory is one of the factories that apply AGVs to the logistics system (Figure 1.2).

- AGVs are used to transport materials between the supply station and the production station in the Daxner dry food factory.

Figure 3 AGV in Danxer factory

- AGVs are used in automatic or manual product assembly: at the BMW factory, 10 ROBOS AGVs Assembly are used in engine assembly lines with loads up to 2500kg.

Figure 4 AGV in BMW factory

1.2 Genre of AGV

There are so many ways to distribute AGVs; however, AGVs are normally classified according to the way they are loaded. Based on the above criteria, there are 3 fundamental genres: Tugger AGV/Towing AGV, Forklift AGV, Unit load AGV.

1.2.1 Towing AGV

Figure 5 Towing AGV of Casun

The current traction AGV is still being widely used, good support in pulling goods, conveying, ... in order to finish manual work, transport materials in the process of work.

The towing AGV is suitable for carts with wheels, by using the trailers, the towing AGV can tow multiple carts at the same time. However, their main disadvantage is that they cannot hook up the trolley automatically (or the automation is quite complicated) so an operator is required to perform the splicing/unloading operations.

1.2.2 Forklift AGV

Figure 6 Forklift AGV by CEIT

The lifting AGV helps to meet the requirements of transporting goods at high altitudes, with some special features

Points of self-propelled lift trucks:

- Ability to lift goods
- Suitable for goods placed on shelves, platforms or pallets
- Usually more expensive than other types of AGV due to more complex design

1.2.3 Unit load AGV

Figure 7 Unit load AGV by CASUN

Cargo AGVs usually carry goods after coming off the conveyor, or under the shelves/pallets and directly bear the weight of the heavy objects, so the carrier AGVs are often used for transporting goods. Compact, light-weight goods such as: electronic components, automotive components, ... AGVs in this form are more flexible than pulling and lifting because they save space for AGVs.

1.3 Mandatory equipment of AGV

An AGV is mainly composed of parts such as:

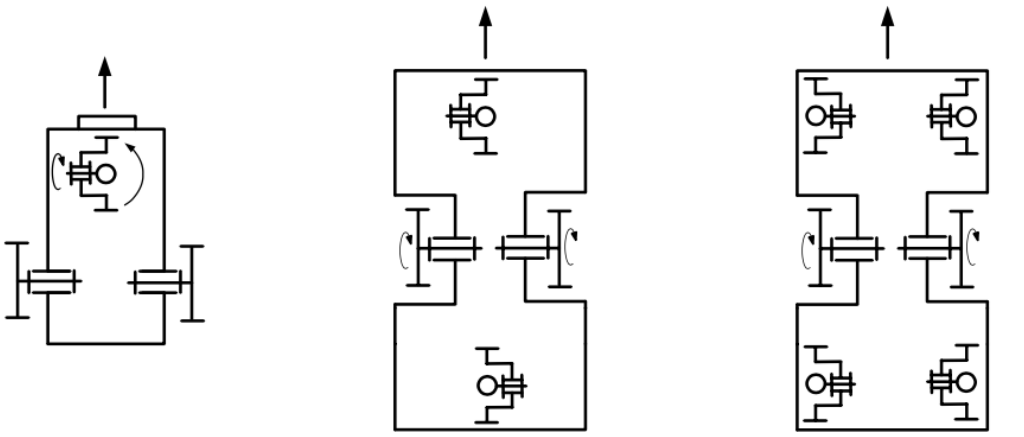
- Driving system: provides the main movement for the AGV
- Positioning system (road tracking system): to determine the route for AGV

- The module performs additional functions (safety sensors, locating row, ...)
- Electric control system
- Power

Below are studies on the most necessary parts to be able to build a complete AGV.

1.4 Driving system of AGV

An AGV has all the components such as the drive motors, sensors, towing unit, power supply and controls all housed into a single chassis. The structures of an AGV are:



a. 3-wheel principle b. 4-wheel principle c. 6-wheel principle

Figure 8 Some popular principle of AGVs' structures

1.4.1 Three-wheel principle

Figure 9 3-wheel principle of AGV

This structure of the AGV consists of three wheels in which the driving wheel (usually the front wheel) has two degrees of freedom: both pulling and rotating, the other two wheels are free to rotate. The front-wheel drive wheel both creates a straight-line movement of the vehicle and creates a steering motion that helps the vehicle follow a predetermined line.

The advantage of this structure is that the three wheels are always co-planar and at the same time create a stable base plane, suitable for applications where AGVs are used for lifting or loading on the body of the vehicle.

The disadvantage of this structure is that when the AGV operates, especially when cornering, the inertia generated for the drive wheel is large. In addition, this structure is not suitable for towing AGV because when towing, the load of the cart will act through the traction mechanism, creating a flipping torque that tends to lift the front wheel of the vehicle, causing the drive wheel to be damaged. loss of traction makes the AGV unable to run.

1.4.2 Four-wheel principle/ Six-wheel principle

Figure 10 4-wheel principle of AGV

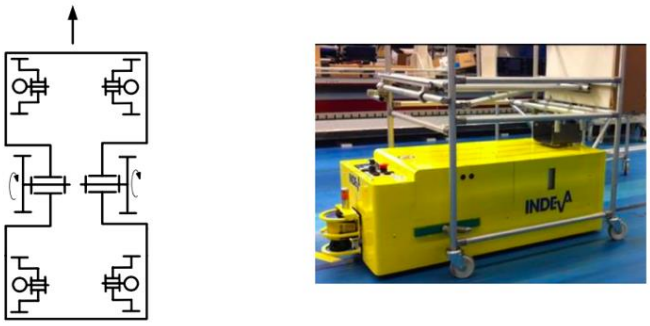


Figure 11 6-wheel principle of AGV

The structure includes two drive wheels placed in the middle and 1-2 self-selecting wheels placed at the top and bottom of the vehicle to help balance the AGV. The AGV works by controlling the speed of the two driving wheels.

Thanks to the coordination of the two drive wheels, the AGV can run straight, reverse, even rotate in place (in the case of two wheels running at the same speed and

in opposite directions). In addition, the four-wheel / six-wheel structure also creates a wide base (rhomboid) so the load capacity is greater.

The disadvantage of this structure is that a coplanar mechanism is required to compress the two drive wheels to the ground to create traction. This structure also has a disadvantage that when the AGV turns a corner, the two driving wheels must bear a large moment of inertia, so for large loads or large sized AGVs, the AGV is prone to losing line due to fluctuations due to inertia. caused by AGV.

1.5 Navigation system and safety sensor

The positioning system helps AGV determine the route to go, its location in the factory. AGVs are guided using one or more tracking mechanisms such as electrical wires, magnetic guide tapes, microdots, lasers, etc.

1.5.1 Navigation system (path tracking system)

1.5.1.1 Route tracking system with electrical conductor

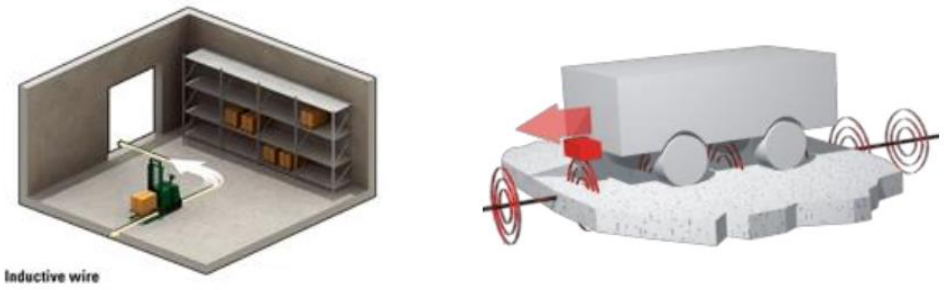


Figure 12 Route tracking system with electrical conductor

- Principles

  An electrical conductor placed in a groove in the floor of the factory emits a magnetic field. This magnetic field interacts with an antenna placed on the AGV and this antenna converts that interaction into an electrical signal to drive the AGV. This system is AGV's first navigation system.

- Advantages

  • The system is simple, low cost and unaffected by dirt and oil.

  • High applicability to autonomous robot applications in agriculture and autonomous trains.

- Defect

- The system is inflexible because the floor has to be slotted at the factory to install the conductors.
- Magnetic field signals are susceptible to interference causing errors during the operation of the AGV.

1.5.1.2 Magnetic tape navigation

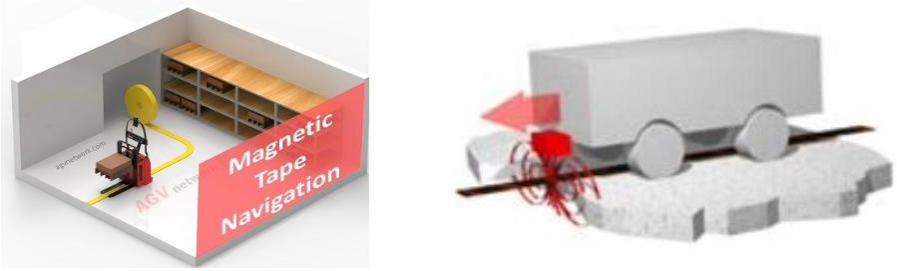


Figure 13 Magnatic tape navigation

- Principles

Magnetic tape is essentially a strip of tape coated with magnetic material to emit a magnetic field. An array of magnetic sensors - usually a Hall sensor, will convert the magnetic field strength into a voltage signal, combined with a processing algorithm that will return the deviation from the line of the AGVs. Currently, this is the most commonly used method and the magnetic sensors are also prefabricated according to industry standards, making it very convenient for manufacturers.

- Advantages
  - Simple installation line system, unaffected by dirt and grease in industrial environments.
  - The system is flexible, easy to change and expand upon request.
  - The magnetic line is relatively flexible, so it is easy to bend into many curvature radii.
  - The sensor system is pre-fabricated according to industry standards, so it works stably and reliably.
- Defect
  - Relatively high cost

1.5.1.3 Magnetic spot navigation

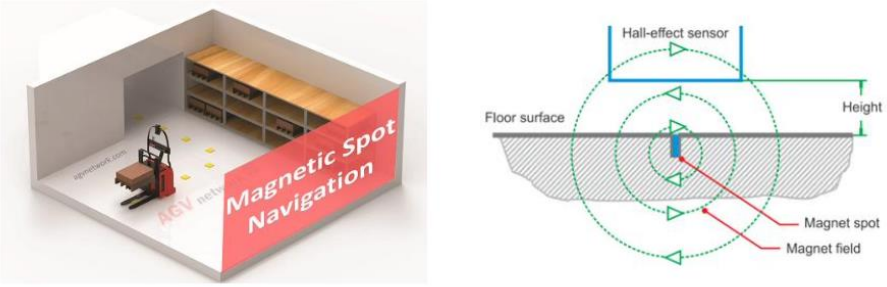


Figure 14 Magnetic spot navigation

- Principles

This positioning system is similar to the magnetic line positioning system but here the small magnetic dots are permanent magnets that will be placed in a grid and specified by the installer.

Magnetic dot navigation is based on the Hall effect. When a conductive flat plate (Hall element) is placed in a magnetic field, the magnetic force will attract electrons and protons to either side of the plate. A Hall sensor measures the magnetic flux by measuring the voltage across the flat plate [7].

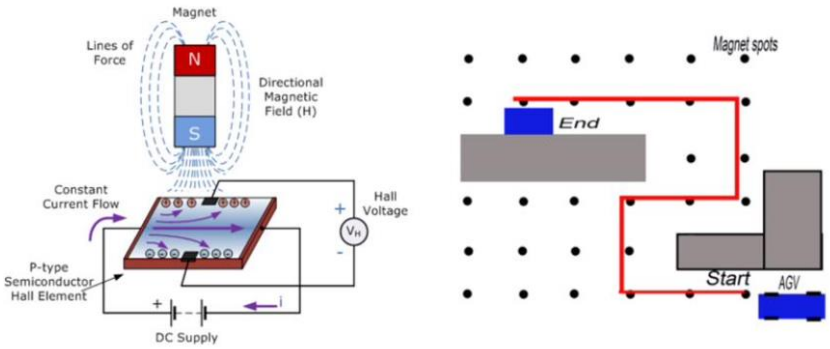


Figure 15 Small magnetic dots navigation

The magnets are arranged on the warehouse floor as a coordinate system, when the AGV determines a point on the coordinate grid, the AGV will continue to make a turn request in the direction specified by the programmer.

- Advantages

• Simple installation line system, unaffected by dirt and grease in industrial environments.

• The system is flexible, easy to change and expand upon request.

- The sensor system is pre-fabricated according to industry standards, so it works
- •. Stable and very reliable.

- Defect
  - The accuracy of positioning depends very much on how the calibration as well as the data processing algorithm.
  - More complex positioning algorithms.
  - Simple installation, but still more complicated than the magnetic line positioning method.

1.5.1.4 Ultra-wide band route tracking system

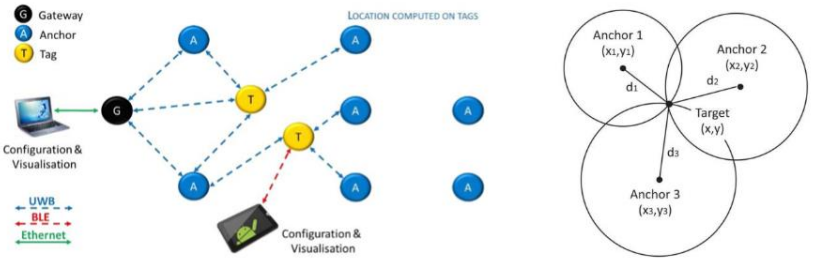


Figure 16 Ultra wide-band route tracking system

- Principles

Ultra wide-band (UWB) works based on electromagnetic waves, Anchors transmitting signals are fixed at different locations in the warehouse (from 3 Anchors or more), a receiver will be mounted on the AGV to receive signal, the system will calculate the distance from the AGV to the Anchors, thereby determining the position of the AGV [4].

- Advantages
  - The system can be changed and expanded upon request.
- Defect
  - The accuracy of positioning depends greatly on the data processing algorithm.
  - More complex positioning algorithms.

1.5.1.5 Laser navigation system

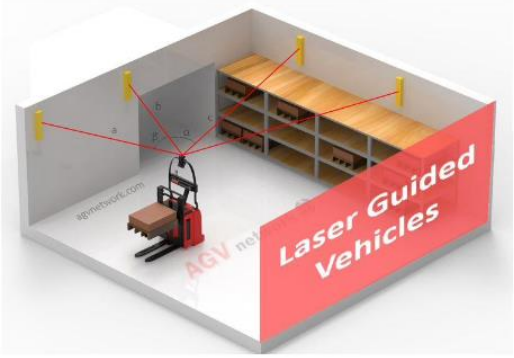


Figure 17 Laser guided system

- Principles

A map in digital format will be stored in the memory of the AGVs controller, AGVs will locate themselves on the real map by scanning reflective sheets pasted at landmarks specified by the programmer. Usually, the reflective panels as a mold are usually glued on the wall, at the location of the store shelves.

The Laser scanner sensor will emit a laser and rotate that laser with a fixed rotational speed, the laser will reflect back when it meets the reflective panels mounted on the wall where the AGVs work. Based on the calculation of the phase delay of the emitted and reflected rays as well as the angle of deviation of the reflected rays, the processor of the AGVs will calculate the position and predict the next position of the AGVs based on the 3-point algorithm.

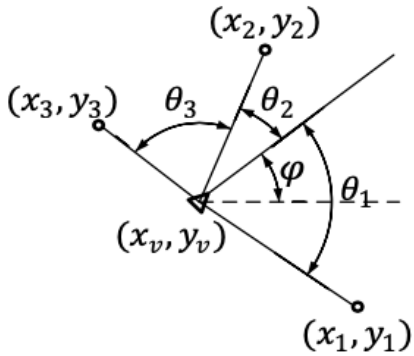


Figure 18 Principle of 3-point positioning of the laser positioning system

- Advantages
  - Simple installation system, no need to break the structure of the place of installation.
  - Flexibility to change and expand.
- Defect
  - Maintenance of reflective panels should be done regularly.
  - Complex processing algorithms.

1.5.1.6 Natural feature navigation

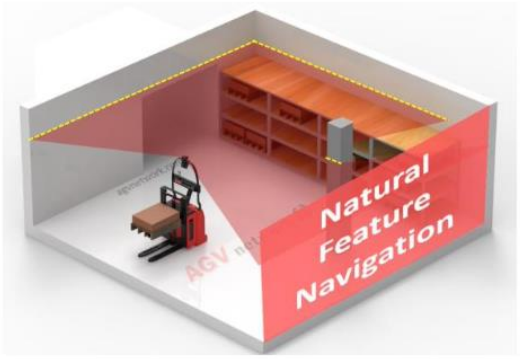


Figure 19 Natural feature navigation

- Principle

Similar to the laser navigation system, a digital map is also stored in the memory of the controller, but the difference here is that this navigation system does not need to use reflective panels for positioning. location, the AGV maps the environment using cameras, lidar sensors, lasers, ... make a reference map, compares it with the map stored in memory, thereby determining its position in the digital map. This algorithm is called SLAM (Simultaneous Localization and Mapping). This method helps the robot adapt itself to the new environment without having to reprogram the map or avoid obstacles, reducing the number of safety sensors. Currently, this method of positioning is being developed very strongly thanks to the application of artificial intelligence to improve the positioning ability of AGVs. The systems of companies like Kollmorgen are very applicable, especially in the medical field where it is difficult to change the existing structures of the hospital.

- Advantages

- No need to spend time installing navigation systems, without disrupting the structure where AGVs operate.
- Flexibility: easily add new paths or modify existing ones.
- In the environment with interference factors such as the movement of people, goods, pallets, ... The system is less affected by environmental noise because the system uses fixed elements such as walls, columns.
- Precise positioning system and stable operation.

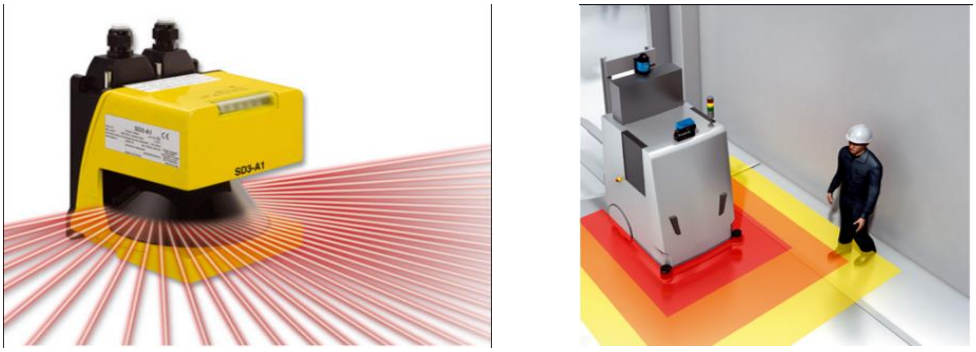
- Defect
  - The processing algorithm is complex, so it requires research and development time, this system is only suitable for environments with relatively noisy factors.

1.5.2 Safety sensor

Safety sensors are used to ensure the safety of AGVs as well as people in the warehouse. Usually, two types of sensors are used at the same time: non-contact safety sensors (Laser) and contact type safety sensors.

1.5.2.1 Safety sensor using laser

Laser safety sensor is a non-contact safety sensor that creates an invisible safe area around the AGV when the AGV is in operation. If someone or an obstacle enters this safe zone, the AGV will issue a warning, slow down or stop completely when the obstacle is closer than the allowed distance of the AGV.



a)Wention Laser scanner *b) The safe zone has many levels generated by the sensor*

Figure 20 Safety laser sensor from Wention Technology

1.5.2.2 Safety sensor using contact type

Contact sensors in AGVs or bumpers are composed of a rigid plastic rod connected to a damper cylinder and connected to a switch. When the bumper is impacted, it will act on the switch creating an interrupt signal that causes the AGVs to stop working. Bumper is a safety sensor that helps protect AGVs from unexpected impacts during operation.

1.6 Motion planning

The path planning is normally known as motion planning, navigation problem or piano movers' issue. It is the computational algorithm help the robot determine a continuous, deadlock-free path, with little congestion delay for the AGV from the initial state to the target position. Hence, AGV can be autonomously navigated throughout locations, potentially within a large swarm. This algorithm can be implemented for 2D and 3D workspace which is linked to coordinate system, also called as configuration space (C_{space}). An AGV often chooses to create new unique path or using the generated path before, that depends on which method is approached. Constraints of static and dynamic obstacles, feasible curvature, robot size, lane dimensions, and speed are included to determine the approximately optimal path with single or combined objectives.

There are 2 methods of motion planning for single AGV or swarm. Nevertheless, due to the aim of this thesis – apply algorithm to 1 vehicle, swarm method is exception.

There are plenty of criteria to specify path planning classification. Taking account to the environment, it can be separated into global path planning and local path planning in order to deal with the static and dynamic environments thoroughly. In the static environments, the path planning performance is implemented only once; nevertheless, in the dynamic environments, it is repletely required to generate a new path multiple times which is collision-free, for the AGV to bypass or to remove the obstacles. In robot kinematic constraint, the path planning can also be divided into holonomic path planning and non-holonomic path planning. Finally, in decomposition method, it can be classified into deterministic and probabilistic planners.

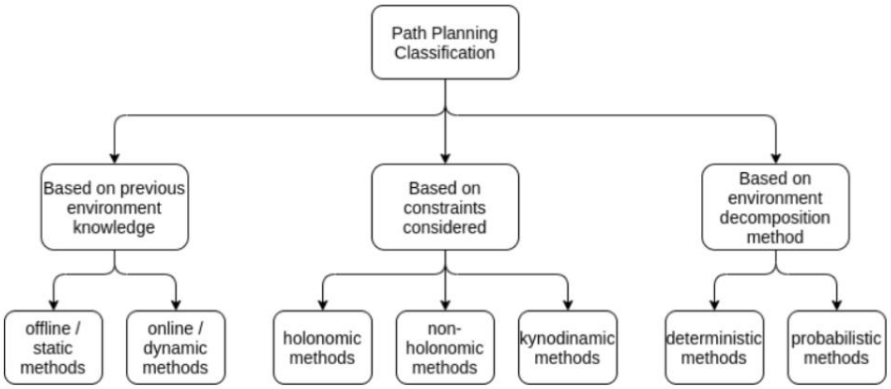
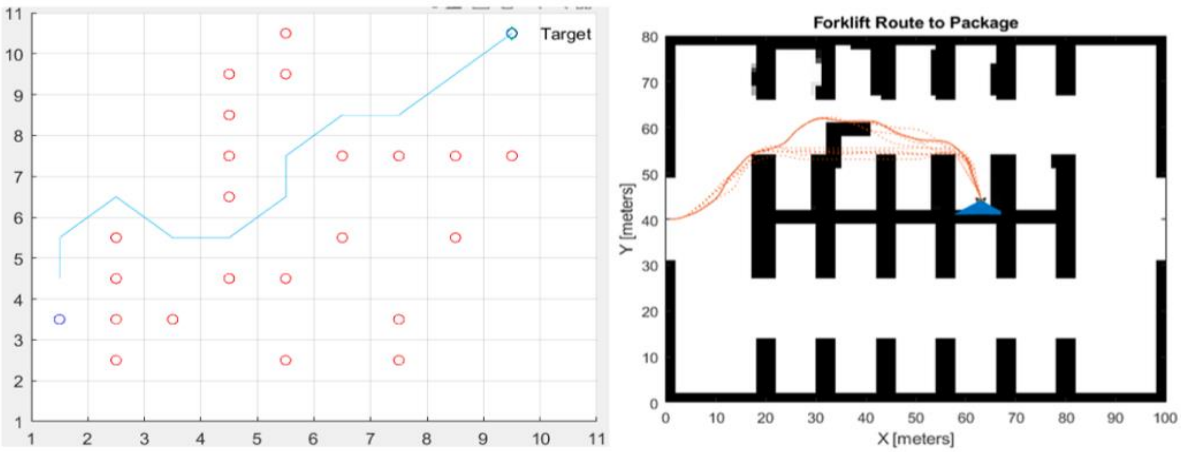


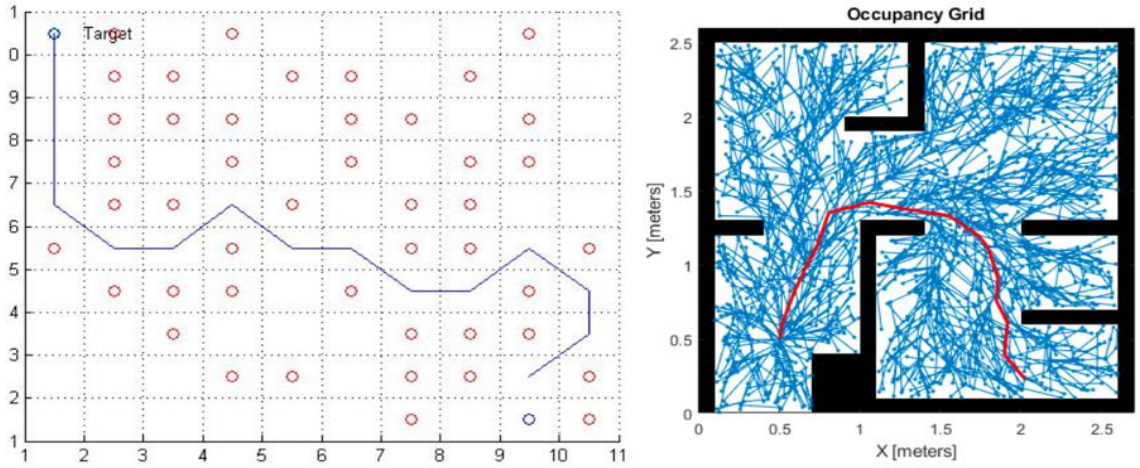
Figure 21 Trajectory planning distribution

In order to navigate in the dynamic environment, there are 2 main tasks needed to handle by by the algorithm. First of all, the global path algorithm is computed in the static environment and link initial with target position based on provided map from SLAM method. Then, while operating, a range of sensors may detect some dynamic obstacles. Consequently, the robot has to deal with them by making the priority of avoiding collision – which is generated from the local path planner – before it comebacks to track the global trajectory. In conclusion, this method can help the robot deal with some unknown obstacles appearing accidently in the map.



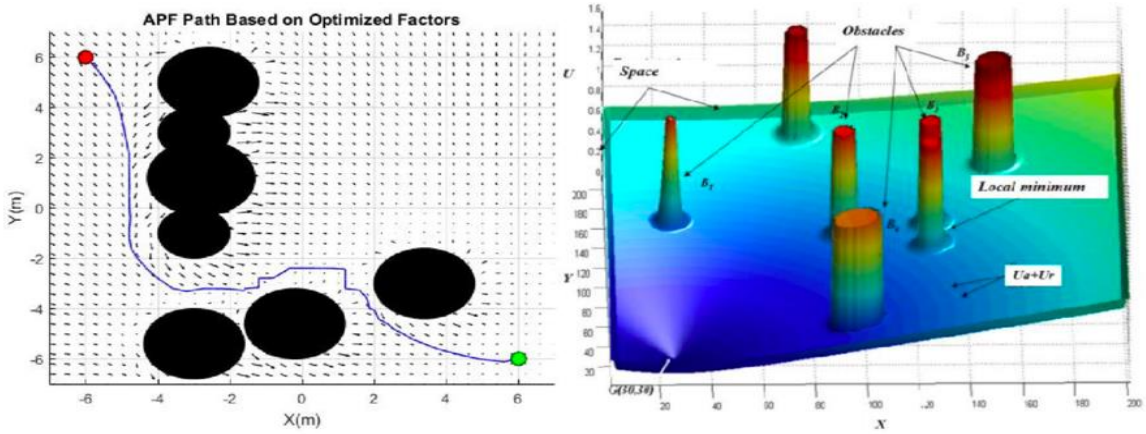
a)Static environment *b)Dynamic environment*

Figure 22 Path planning algorithm using MATLAB



a)Grid – base planner *b)Sampling – base planner*

Figure 23 Path planning: Grid – base and sampling – base process



a)Algorithm view *b)3D view*

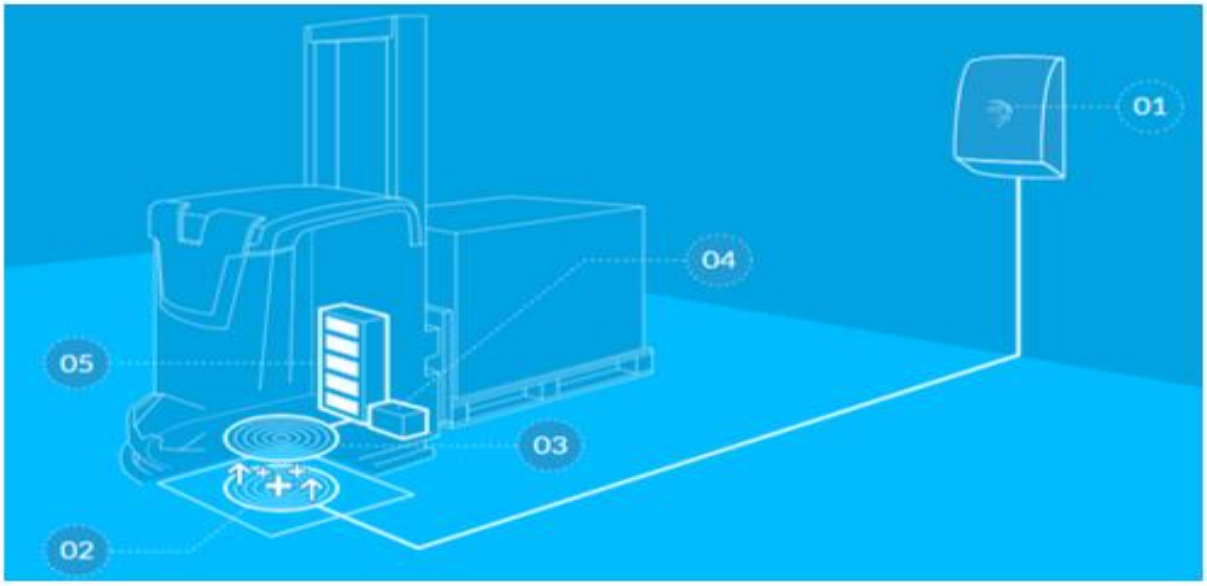
Figure 24 Path planning: Potential view process

There are various of motion planning algorithm and its variant has been developed; however, it can be classified into 3 fundamental genre such as Grid – base planner, Sampling – base planner, and Potentials field method. In addition, at the end, the algorithm not considering physical constrain, the holonomic and non-holonomic robot should be examined.

1.7 Battery

Traction and undercarriage AGVs usually use lithium-ion batteries with rechargeable function, voltage of 24VDC or 48VDC, battery capacity to meet working time of 8-10 hours, corresponding to this type of battery is manual charging mode.

Many production facilities / warehouses require AGVs to work 24 hours a day continuously, the AGVs for these facilities often use continuous rechargeable batteries at wireless charging stations in the factory, the charging method is described as follows:



(01) Wall box, (02) Charging pad, (03) Receiver coil, (04) Mobile charging unit, (05) Energy storage

Figure 25 Continuous charging station

The charging base (2) is fixed on the floor, when there is an electromagnetic field provided by the source (1), a magnetic field will be generated. When the AGV moves to position on the charging base (2), this magnetic field acts on a conductive coil (3) inside the AGV that generates current from which to charge the battery (5).

Some battery models of Delta Industrial EV Power Solutions company:

MODEL*	INPUT	OUTPUT	AIR GAP
1kW	110V / 240V	24V	10 to 20mm
7.4kW	400V / 480V	48V	80 to 120mm
30kW			100 to 150mm

Charging modules of some brands such as B&PLUS for communication distances from 0-40mm.

The continuous use of the charging station should pay attention to the position error of the AGV.

1.8 Objective

1.8.1 Design target

Design a AGV for the towing goods in the manufactory. This robot is able to have a safe performance when dealing with the dynamic environment and track the optimal path to the target position

1.8.2 The design solution

Mechanical and electrical design

Control algorithm design

Improve development of simulation or experimental test for motion path planning on prototype

1.8.3 Robot requirement

Normal mode – the robot is able to find the optimal path to carry goods and turn back to the pick-up place

Traveling different setup places in a trip

Each shelf has already numbered

The robot can communicate with users via UI on the web sever

Criteria	Content
Speed	Maximum speed: 1 m/s Average speed: 0.5 m/s
Load capacity	Maximum: 200kg
Maximum dimension	
Operating time	6-8 hours
Charging mode	Auto charge

Table 1 Robot requirement

1.8.4 Operation environment requirements

This design AMR arm to design a vehicle can work in several types of places such as: warehouse, assembly factory, etc. Which are assume that have clean environment and flag floors. In working environment, AMR would have ability to scan map, towing carts, move independently and avoid obstacles randomly.

1.8.5 Thesis structure organization

Base on the goals of this project, the thesis will consist of some fundamental chapter:

Chapter 1: Introduction about robot and navigation system.

Chapter 2: Provide various design method for selecting the most suitable one.

Chapter 3: Design and compute mechanical parts, with motor selection.

Chapter 4: Calculate and deploy electrical elements.

Chapter 5: Design the robot controls' algorithm.

Chapter 6: Indoor operation: apply navigation algorithm.

Chapter 7: Deploy design into the real system.

Chapter 8: Conclusion and future goal.

CHAPTER 2: DESIGN METHODOLOGY

2.1 Mechanical selection

2.1.1 Mechanical diagram

No	Criteria	Definition
1	Cornering radius	Minimum radius (or width) of available space required for that vehicle to make a circular turn
2	Co-planarity	Wheels contact points constitute a flat plane
3	Load capacity	The maximum carried load distribute on each wheel
4	Design simplicity	Mechanical design (assembly, manufacturability) Electrical design (Wiring)

Table 2 Selection criteria for implementing principle

The required environment of this thesis is indoor and flat floor. To achieve that goal, there are some principle diagrams which is considerably satisfied below:

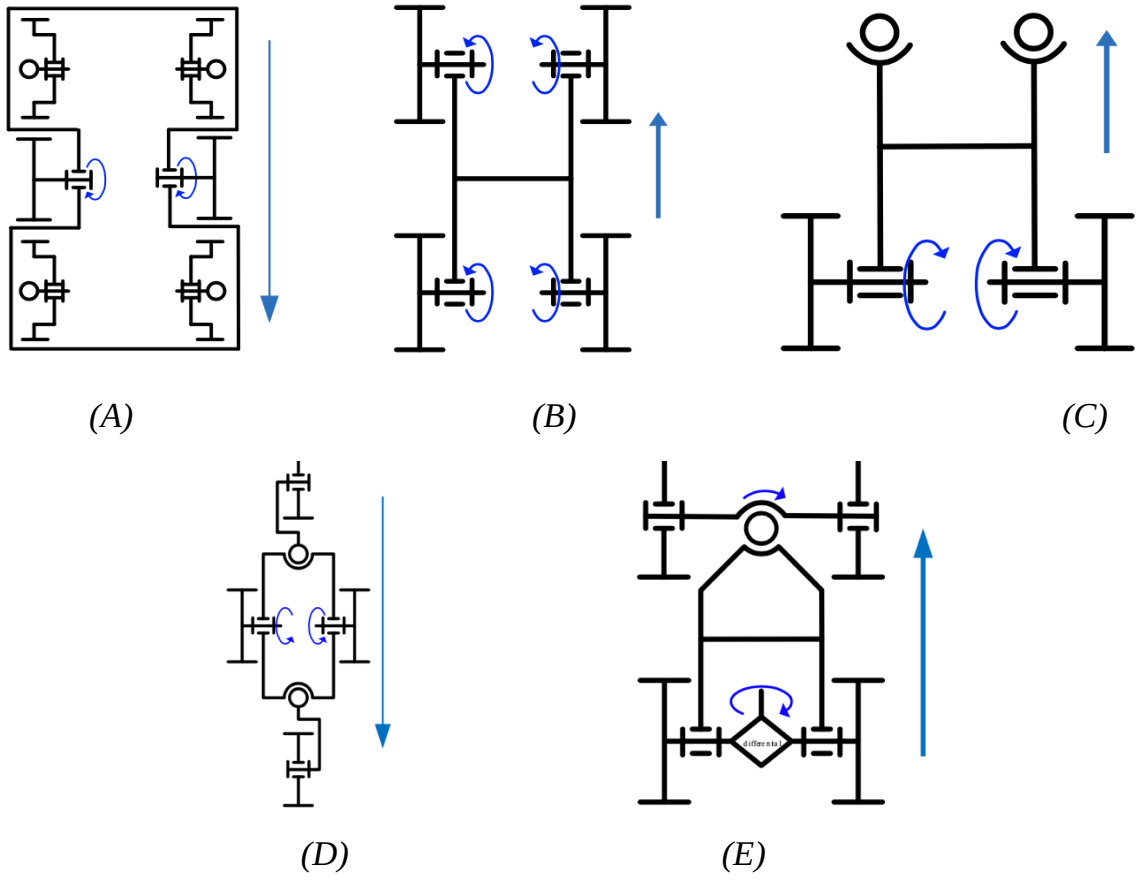


Figure 26 Satisfied models

Criteria	Weight	Model A		Model B		Model C		Model D		Model E	
Cornering radius	0.4	4	1.6	3	1.2	2	0.8	4	1.6	1	0.4
Co – planar	0.1	1	0.1	3	0.3	3	0.3	2	0.2	2	0.2
Load capacity	0.2	4	0.8	3	0.6	2	0.4	2	0.4	3	0.6
Design simplicity	0.3	2	0.6	1	0.3	3	0.9	2	0.6	1	0.3
Sum	1		3.1		2.4		2.4		2.8		1.5

Table 3 Weight matrix selection

According to the above table, consequently, it can be easily observed that Model A is clearly satisfied for this project. With 6 wheels in total, however, this system requires higher co-planar for all wheels. Additionally, with 2 middle-driving wheels, the dynamic capability as well as the controller complication can be adjusted.

2.1.2 Suspension system

According to the floors' characteristic of the manufactory, there are several incline angle. In other word, the selected mechanical diagram require co-planar capability of all wheels. Thus, the suspension system is chosen to play a vital role reducing the system cost and complication of suspension controller. Additionally, there are 2 genres of passive suspension system as below

Type	Independent suspension	Dependent suspension
Description	Permit wheels moving up and down independently	Wheels are linked with each other by a common rail/beam and work in tandem
Advantage	Superior tackling and cornering Great balance/stability and steering Lower weight	Immense load solving Uncomplicated when maintaining Great off-road capabilities Clearance and articulation
Disadvantage	Complicated design Expensive maintenance cost Small overall strength	Impoverished solving Finite improvement Instability when high-speed operation

Table 4 Passive suspension system

And base on the specification above, the criteria of this project is illustrated below

No	Criteria	Definition
1	Stability	Robot operation moving through the bump and inclined plane needs to be stable
2	Traction	Wheels contact points constitute a flat surface
3	Maintenance cost	The less elements, the cheaper cost to handle

Table 5 Criteria selection

Criteria	Weight	Independent suspension		Dependent suspension	
Stability	0.6	3	1.8	1	0.6
Traction	0.3	2	0.6	2	1.2
Maintenance cost	0.1	1	0.1	3	0.3
Total	1		2.5		2.1

Table 6 Weight matrix selection of suspension system

According to the above table, the independent suspension system is the most satificated principal diagram for this project. Nevertheless, this system rises up the complication in co-axial of 2 driving wheels

There are 3 fundamental genre of passive independent suspension system consist of Macpherson strut suspension, double wishbone suspension, trailing arm suspension [5]. Based on the maintenance cost criteria, the trailing arm model [6] is chosen due to the less required component

2.2 Electrical selection

2.2.1 Power supply

The requirement of power supply consists of mobile capacity, high life cycle and discharge rate

DC power is considerably chosen as the fundamental power supply due to the flexibility – better than AC power supply. By using energy container in form of battery, robot is able to move freely in a distance. Taking high life cycle and discharge rate into account, the acid battery is less handy than acid Lithium-ion battery [7].

2.2.2 Motor

Because of the above selection of DC power, the genre of DC motor is elected to reduce the DC to AC converter cost

Type	Advantage	Disadvantage
Brushless motor	High speed implement Large starting torque Great efficiency Immense power density	Complicated in motor driver Expensive (permanent magnet) Short constant power field
Brushed motor	Small speed operation Uncomplicated in motor driver Great torque at low speed	Heat generated by brushed Bulky structure Inefficiency
Stepper motor	Low overshoot Favorable positioning control in moment dimension	Miss step under high-torque operation

Table 7 Motor comparison

The criteria of this project is great load carry, close – loop control system in order to enhance accuracy and maintenance cost.

Thus, according to the specification of the above table, brushless motor is completely chosen. And, additionally, flange mounting genre is chosen due to the suitable arrangement with suspension and wheel.

2.3 Sensor

2.3.1 Sensory system

- **LIDAR**

Sensor is going to generate laser pulse train, which is aimed to send to the surface/object forward and measure the distance after an interval time when signal return. With the mounted motor, sensor is able to scan at maximum angle - 360 degrees. In addition, there are 2 fundamental of Lidar sensor: 2D Lidar and 3D Lidar. First one, taking 2D Lidar into account, it can get the 2D planar data, except the obstacle out of sensor range. The collected data from sensor are processed using specially designed computer software (LIDAR Point Cloud Data Processing Software). Moreover, the Lidar sensor is efficient for mapping the environment with SLAM algorithm. Base on reference [8]:

$$Distance = \frac{Speed\ of\ light\ \times\ Time}{2}$$

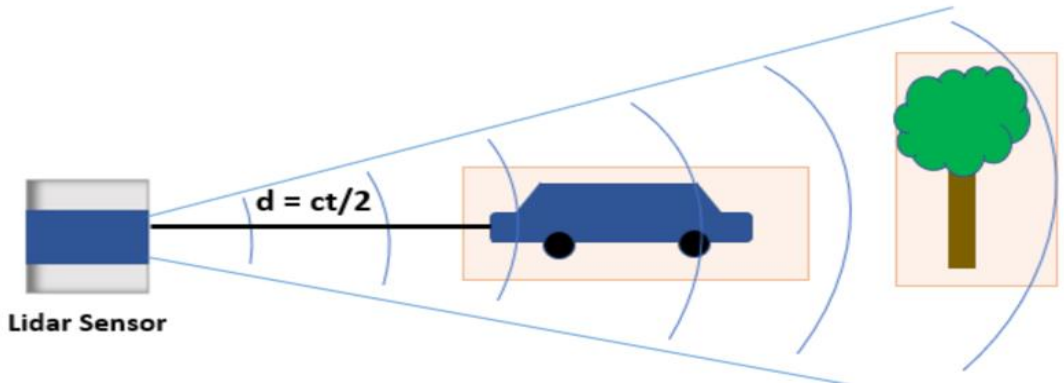


Figure 27 Lidar principle

- Advantages:
  - Detect obstacles' distance.
  - Mapping.
- Disadvantages:
  - Noisy when dealing with transparent and rough surface such as glass – which cannot reflect laser signal back directly to the receiver of Lidar sensor.

- **Camera:**

Camera is an optical instrument which captures a visual image of the physical world. Moreover, it can inform the vast details of the boundary, shape, and color of every object in 3D space. Nevertheless, comparing to the Lidar sensor, cameras' data gives a great density collection. There are both monocular and stereo camera are satisfactory for mapping the environment with VSLAM algorithm.

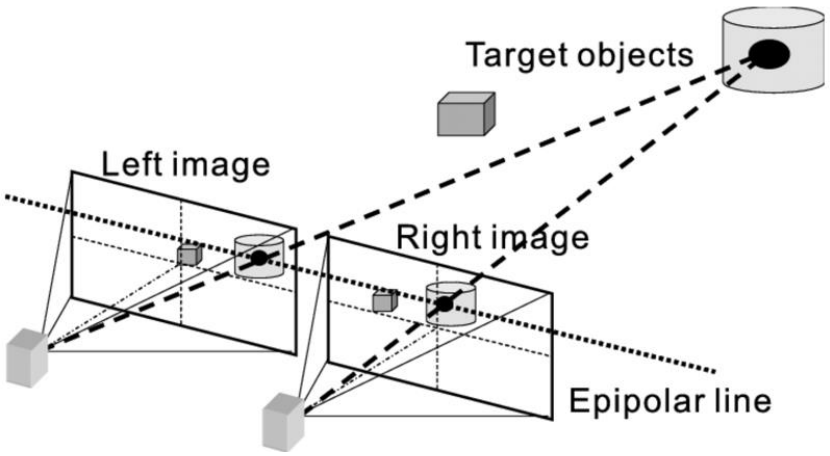


Figure 28 Stereo camera

The principle of stereo camera is using 2 cameras with triangular configuration scheme can the depth of the image [9]. The depth of a point in 3D space is inversely ratio to the disparity of that point between the right and left cameras

- Advantages:
  - Detect obstacles' characteristic in 3D space.
  - Mapping.
- Disadvantages:
  - Depend on light condition.

- **Ultrasonic:**

This device is able to measure the distance of an aim object by emitting ultrasonic soundwaves, and converts the reflect sound into an electrical signal. This sensor has 2 fundamental elements: the transmitter (trigger the sound emitter using piezoelectric crystal) and receiver. Base on reference [10]:

$$Distance = \frac{Speed\ of\ light\ \times\ Time}{2}$$

- Advantages:
  - Detect transparent obstacle.
- Disadvantages:
  - Inefficient when handling several rough surfaces such as foam, cotton lead to flaw.

- **IMU:**

This sensor is aimed to illustrate the orientation - in Roll, Pitch, Yaw – and acceleration by combining Accelerometer, Gyroscope and Magnetometer into one large component. Moreover, this sensor is able to inform the data measurement of state of the robot

- Advantages:
  - Detect orientation of the robot
- Disadvantages:
  - Quite noisy with accumulated errors, so that it must be approached with filter algorithm

2.3.2 Mapping sensor

According to the above information about navigated sensors, from the criteria: wide range, high accuracy and the condition of environment, the Lidar sensor is considerably chosen to deal with the generation of 2D map.

2.4 Control structure

There are 2 fundamental control structure genres: centralized control and distributed control.

Type	Centralized Control	Distributed Control
Controller number	1	Must be at least 2
Working principle	1 controller will handle all tasks: receive signal from the various sensors, apply and compute algorithm, transmit the signal to control modules	-1 master controller will process and compute sensor data, approach algorithm and 1 slave controller will control the actuator thanks to the signal from master
Advantage	-Low cost -Less space needed -Great response time due to no communication among controllers	-Uncomplicated when debugging the program -Improve, enhance the program simply
Disadvantage	-Simply detect and maintain the issue, but complicated when updating	-High cost -Space requirement

Table 8 Control structure analysis

No	Criteria	Definition
1	Response time	The time needed for MCU to execute the control program. The faster, the better
2	Scalability, simple when debugging	Ability to edit, expand and upgrade programs in a teamwork industrial setting. The easier, the better
3	Specialization	Ability of MCU in a system to perform separate functions. The more specialization, the better
4	Demand of resources	Number of controller board. The less, the better

Table 9 Criteria for control structure

Criteria	Weight	Centralized control		Distributed control	
Response time	0.5	3	1.5	1	0.5
Scalability, simple when debugging	0.2	2	0.4	4	0.8
Specialization	0.2	4	0.8	3	0.6
Demand of resources	0.1	1	0.1	2	0.2
Total	1		2.8		2.1

Table 10 Weight matrix selection

According to the weight matrix selection, the centralized control is more reasonable for this project. Inspire of the tough scalability and debugger with various complicated algorithm for motion planning and mapping, the ROS system is used to enhance this drawback by dividing plenty of tasks into several nodes. The slow response time can be boosted by using embedded computer unit, because of high-speed frequency and resolution (approximately 32bit).

2.5 Controller board

From the requirement, the controller unit should have enough GPIO to connect the actuator and sensor module, great memory storage to contain complicated algorithms (path planning, mapping, indoor positioning), and can implement with ROS system

According to those criteria, the embedded computer board is considerably chosen due to the high performance of ARM codex core – which can implement with Operating system.

Nevertheless, MCU consume less power, has immense interrupt processing speed, and has low cost compare to the others [11].

2.6 Control method

PID controller: Can be used to calculate both motors speed and trajectory following. The algorithm enhances the input control value by error (the different between desired and feedback measurement data output). Nevertheless, only valuable results in linear systems, noise happens at the differential stage, and the consequence is immense errors in the output.

Fuzzy controller: From the data measurement returned from sensors, the controller computes how much or little the robot deviates from the desired path and generates the velocity values of the corresponding motors to track the expected line. Error based

significantly on fuzzy laws, so it is easily observed that the more accuracy the laws computation, the less errors it is. Nevertheless, it requires a knowledge-based approach of the system response.

Lyapunov stability: This controller takes care of 3 errors of the robot: in the tangent direction, normal direction, and deviation angle between the robot and the path to control angular velocity and normal velocity of the robot.

From the specification above, with the expected criteria, the method of control system is:

- Tracking path control: LQR controller
- Control motor: PID controller

2.7 Location and mapping

According to the SLAM algorithm, it is separated into 2 genres base on the input sensor data, comprise: camera – based SLAM (VSLAM) and laser – based SLAM (SLAM)

In other word, when considering the VSLAM, this method establishes the 3D data measurement via 3D point cloud data. This operation can cause the immense of space complexity of embedded computer, while the trajectory planning algorithm of mobile robot only generate node on 2D plane and the requirement of robot operation is on 2D surface. Thus, in this project, the laser – based SLAM will be chosen

Additionally, there are plenty of algorithm's base on laser – based SLAM principle, such as HectorSLAM, Gmapping, Cartographer. According to the experiment in [28], Gmapping is selected due to the simplicity and acceptable error.

Environments	Algorithms	Measurement(m)	Truth(m)	Error(m)
Room	Gmapping	5.52	5.90	0.38
	Hector SLAM		Invalid	
	Cartographer	5.65	5.90	0.25
Corridor	Gmapping	61.28	67.98	6.70
	Hector SLAM		Invalid	
	Cartographer	67.40	67.98	0.58

Figure 29 Experiment of 3 algorithm

The Hector SLAM is invalid because there are several overlaps and drifts of the map generated by Hector SLAM – which leading to the impossible map established.

2.8 Trajectory planning

Type	Grid-based planner	Sampling-based planner
Algorithm	Dijkstra, A*, Hybrid A*	PRM, RRT, RRT*
Description	Generate and handle the collision – free path via an obstacle grid map	Generate and handle the collision – free path base on sampling distance
Application	Low degree of freedom Small map	High degree of freedom Large map

Table 11 Overview of path planning method

CHAPTER 3: MECHANICAL DESIGN

Mechanical design can improve robot's capability, performance efficiency and reduce the complexity of controller. Hence, the suspension and battery removable feature is added.

3.1 Suspension design

This type of trailing arm suspension system has been chosen in Chapter 2 .

The wheel tire is a rigid body, due to indoor wheel type so assume that is tubeless type made by PU material, so only chassis of AGV could be considered in dynamic model.

System have criteria: Maximum overshoot: $OS \leq 5\%$; settling time: $Ts \leq 2s$.

Mathematical model of suspension system:

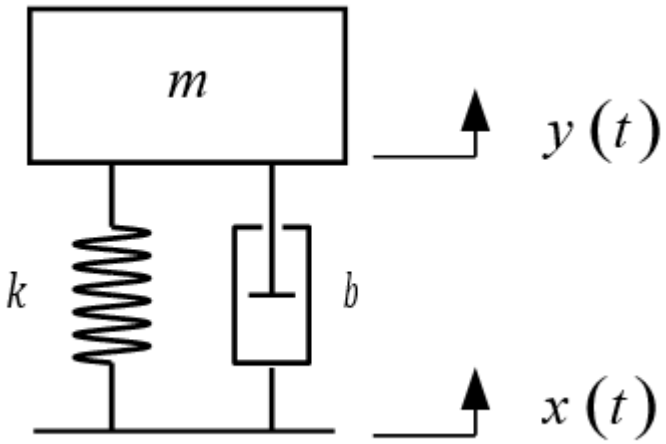


Figure 30 Suspension modelling

The differential equation for the movement of the vehicle:

$$m\frac{d^2y(t)}{dt^2} + b\frac{dy(t)}{dt} + ky(t) = b\frac{dx(t)}{dt} + kx(t) \qquad (3.1)$$

Let assume that weight of AGV is balance, thus that the gravity force of total weight is divide onto both wheels (m = 22.5kg).

Where m: half of robot mass.

b, k: the damping coefficient of damper and elastic coefficient of spring, respectively.

Transfer function of suspension system:

$$G(s) = \frac{y(s)}{x(s)} = \frac{bs + k}{ms^2n + bs + k} = \frac{2\xi\omega s + \omega^2}{s^2 + 2\xi\omega s + +\omega^2}$$

Where: ξ is damping coefficient: $\xi = \frac{b}{2m\omega}$

ω is angular frequency of oscillation: $\omega = \sqrt{\frac{k}{m}}$

Consider that in the working environment of AGV, there is a small protruded terrain separate 2 areas, which robot has to across. Therefore, input response in this situation is the step response. The suspension system must satisfy requirements in these cases.

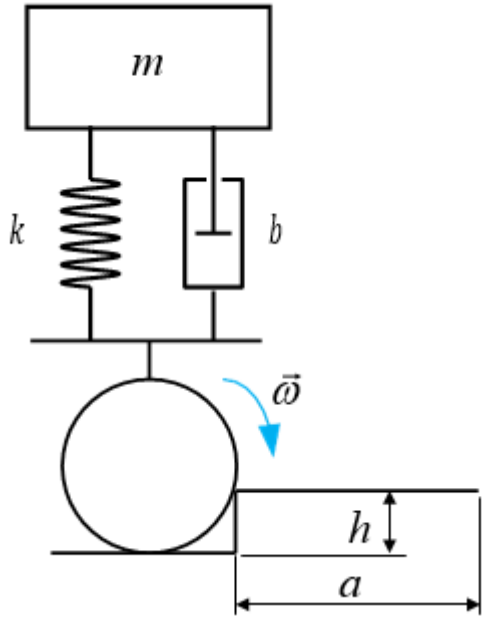
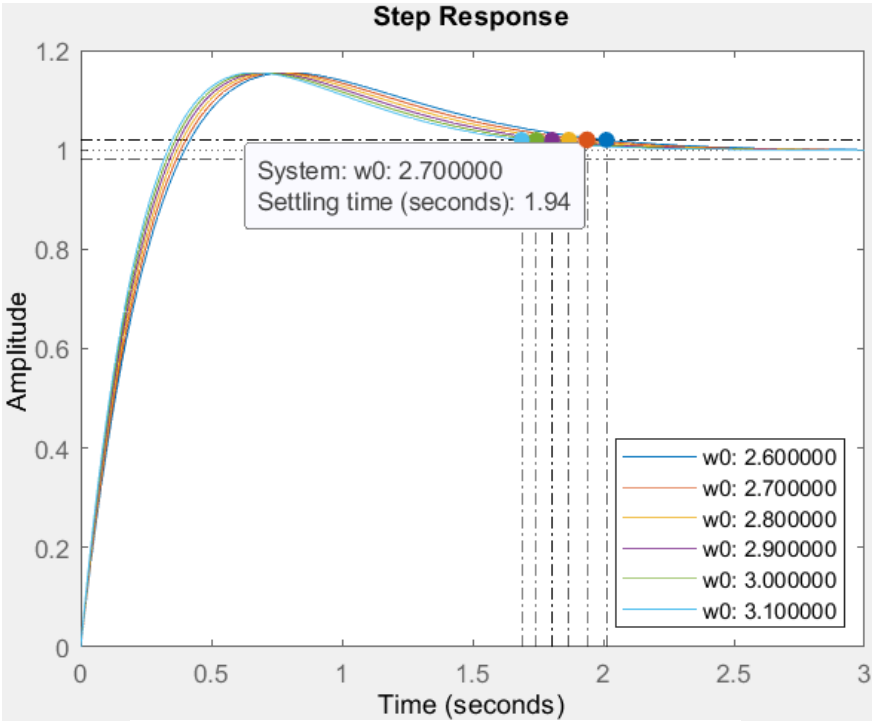


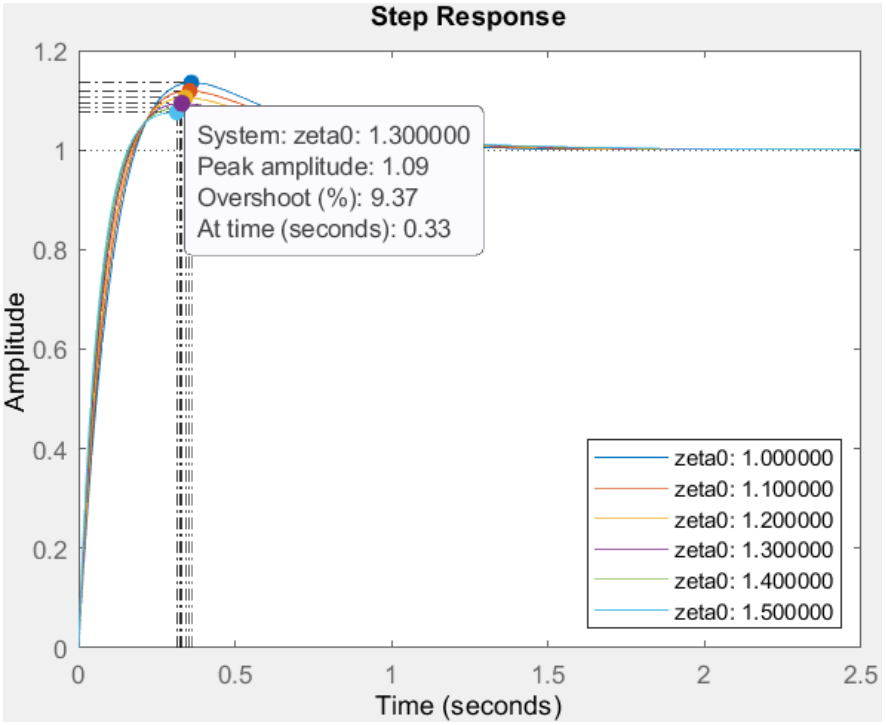
Figure 31 Step response of suspension model

Base on given requirements, chosen variable are ξ = 1.3; ω = 2.7.

Substitute the data in to 2 mentioned equation to 3. And 3. To find the coefficient of system, so $k = 731.025\ N/m; b = 333.45\ N.s/m$



(a) Settling time of different ξ



(b)Overshoot of different ω

Figure 32 Step response of different variable suspension system

Due to the limited of the chassis space, the chosen length of shock absorber is 150 mm with adjustable reload parameter (the load capability) and compression parameter (compression speed), in order to achieve *k* and *b* variable. Besides, under different required environment and load condition, the shock absorber can be adjusted to improve the stability of the system.

Figure 33 TWISTED shock absorber

Parameter	Value	Parameter	Value
Expected damping coefficient	731.025 N/m	Eye to eye distance	150 mm
Expected elastic coefficient	333.45 N.s/m	Maximum load	150 kg

Table 12 Shock absorber parameter

3.2 Motor selection

The AGV with 2 drive-wheels at sides and 4 caster-wheels, so assumed that total loading of AGV distribute equally on 6 wheels. However, 2 drive-wheels provides torque to move AGV, there are rolling resistance and aerodynamic drag acting on robot when moving.

The Towing AGV will tow a cart a with mass of cart at 200kg. Hence, robot parameters would be:

Parameter	Value
Vehicle mass	45 kg
Wheel outer diameter	200 mm
Wheel mass	2
Number of wheels	2
Rolling friction coefficient	0.03
Continuous speed	0.5 m/s
Safety factor	1.5

Table 13 Input vehicle parameter

Additional container's parameter:

Parameter	Value
Vehicle mass	200 kg
Wheel outer diameter	75 mm
Wheel mass	1
Number of wheels	4
Rolling friction coefficient	0.03
Maximum speed	0.05 m/s
Safety factor	1.5

Table 14 Input CART parameter

Base on the selected driving wheels, so maximum speed of driving wheels

In the design, the total weight is mainly distributed on 4 driven wheels, while the 2 driving wheels only bear the weight of chassis plate and suspension structure, as well as traction force from shock absorber.

So, consider force diagram acting on robot structure on flat surface as shown below

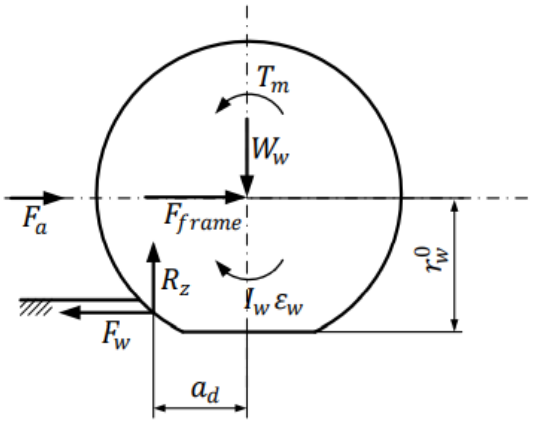


Figure 34 Model of the drive wheel

Where: T_m: driving torque

W_W: components of vehicle gravity acting on the driving wheel and gravity of its wheel

F_{frame}: horizontal reaction force exerted by the chasis on the driving wheel

F_a: specific inertia

R_z: reaction force exerted by the ground on the wheel

F_W: traction from motor torque

$I_W \varepsilon_W$: rotational inertia component of its wheel

a_d: the setpoint displacement of the reaction Rz caused by the elasticity of the tire

r_W^0: the driving wheels' radius

Apply the equilibrium equation of force for the above dynamic system:

$$\begin{cases} F_w = F_{frame} + F_\alpha \\ W_w = R_z \\ T_m = r_w^0.\left(F_\alpha + F_{frame}\right) + R_z.a_d + I_w \varepsilon_w \end{cases}$$

In fact, the driving-wheel made by PU almost undeformed ($a_d \approx 0$) thus that, r_w^0 could be same to the driving-wheel's radius. Apply these values into 3.2 equation, we have

$$T_m = r_w^0.\left(m_w.a_w + F_{frame}\right) + \frac{1}{2} m_w ({r_w}^2 - {r_h}^2)\varepsilon_w$$

Where: m_W: mass of driving wheel

a_W: acceleration of the center of the driving wheel

r_W: the driving wheels' radius

r_h: radius of the mounting hole of wheel's shaft

ε_W: angular acceleration of the driving wheel

Because of the driving-wheels are assembly in base, so acceleration of robot is $a_w = 0.5\ m/s^2$. Thus that, angular acceleration ε_w

$$\varepsilon_w = \frac{a_w}{r_w} = \frac{0.5}{0.1} = 5(rad/s^2)$$

The diagram have force apply onto the driving-wheels is the AGV frame's force: F_{frame}

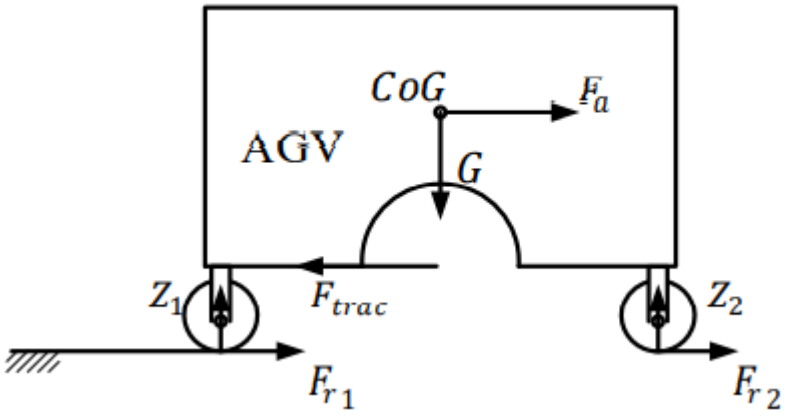


Figure 35 Force acting on robot

Where: F_a: inertia of AGV system – goods,

$$F_a = (M_{load} + M_{AGV})a_W$$

F_{trac}: the combined traction generated by the two driving wheels, considering that AGVs move on a straight line, then: $F_{trac} = 2F_{frame}$

G: mass of AGV - goods

F_{r_i}: rolling resistance acting on the i wheel, suppose that mass is equally distributed, so: $F_{r_i} = Z_i \times C_{roll}$

Z_i: reaction due to the ground acting on the i wheel, suppose that mass is equally distributed, so: $Z_i = \frac{1}{6} \times (M_{load} + M_{AGV}) \times g$

C_{roll}: coefficient of rolling resistance

$$F_{trac} = F_a + \sum_{i=1}^{6} F_{r_i} = (M_{load} + M_{AGV}).a_w + (M_{load} + M_{AGV}).C_{roll}.g$$

The elements of the above equation is calculated below:

$(M_{load} + M_{AGV}) = 250kg$, $a_w = 0.5m/s^2$, $C_{roll} = 0.03$

$$F_{trac} = 250 \times 0.5 + 0.03 \times 9.81 \times 250 = 198.575$$

$$F_{frame} = \frac{1}{2}.F_{trac} = 99.2875$$

$$T_m = 10.02675 \text{ (Nm)}$$

In this project, choice safe factor at 1.5. Thus, that angular velocity and minimum motor moment require.

$$\omega_{m_r} = k_n.\omega_m = 1.5 \times 5 = 7.5 \left(\frac{rad}{s}\right) = 71.62\ (RPM)$$

$$T_{m_r} = k_s.T_m = 1.5 \times 10.02675 = 15.040125\ (Nm)$$

The power of the driving wheel

$$P_{m_r} = \omega_{m_r} \times T_{m_r} = 112.800\ (W)$$

Parameter	Value
Continuous wheel velocity	72 RPM
Minimum power	112.8 W
Minimum torque	15.04 Nm

Table 15 Required motor parameter

MPW86 – 185V48 – 1000 – X020 wheel drive is selected for this project [12]:

Figure 36 Driving motor wheel: MPW86 – 185V48 – 1000 – X020

Criteria	Value	Unit
Peaked power	451	Watt
Peaked torque	35	Nm
Rated speed	150	RPM
Rated torque	20	Nm
Rated power	314	Watt
Wheel diameter	200	mm
Material	Aluminum Hub with PTMEG POLYETHER Shore A 70 (Red)	
Gear ratio	20:1	

Table 16 Wheel drive motor mechanical parameter

3.3 Driven wheel selection

Requirement: quick turning, high traction capability and co – planar of 4 wheels.

The caster wheel having shock absorber addition is considered to guarantee the co – planar and high traction capability requirement. This addition feature also improves the smooth movement when robot at emergency brake situation.

4 caster shock absorbing wheel (GDS – 75 – ASF) is selected [15].

Figure 37 Driven wheel: GDS – 75 – ASF

Parameter	Value
Wheel diameter	75 mm
Height	130 mm
Bolt hole spacing	73 × 73
Load capacity	90 kg
Spring defection	10 mm
Wheel type	Cast PU (A95)

Table 17 Caster wheel in series GASD

3.4 Components' material selection and strength analysis

Problem: Need to stress test due to several of the extra load are put on the AGV chassis, as a results of component weight (Motor, Battery, Controller, etc.) and towing carts behind.

Solution: SOLIDWORK analyze software to compare and simulation the structure of AGV.

The stress and strength of material could be exanimated by SOLIDWORK software (at static condition). The results will show the stress condition, deformed and strength

of material which are using to build AGV.

There are several symbols is added by software such as:

- Green arrow: the fix reference plane
- Purple arrow: force and torque

3.4.1 Base chassis

The base chassis of AGV:

In stress exanimate: the structure must experience only elastic deformation under maximum payload. So, if $\sigma_{maximum} < \sigma_{yield}$, the structure is safety.

The base chassis of AGV is structure which carried all load of vehicle.

In this case, the examinate the ability of this part which 2 main forces:

- 450N downforce distribute by inertial weight of AGV
- 2000N force is assumed the maximum force that vehicle can tow

The results show that structure can be work in this case, which material chosen is aluminum 6061. The stress condition is satisfied as figure.

$$\sigma_{maximum} = 1{,}348{,}926.875 N/m^2 < [\sigma_{yield}] = 275{,}000{,}000 N/m^2$$

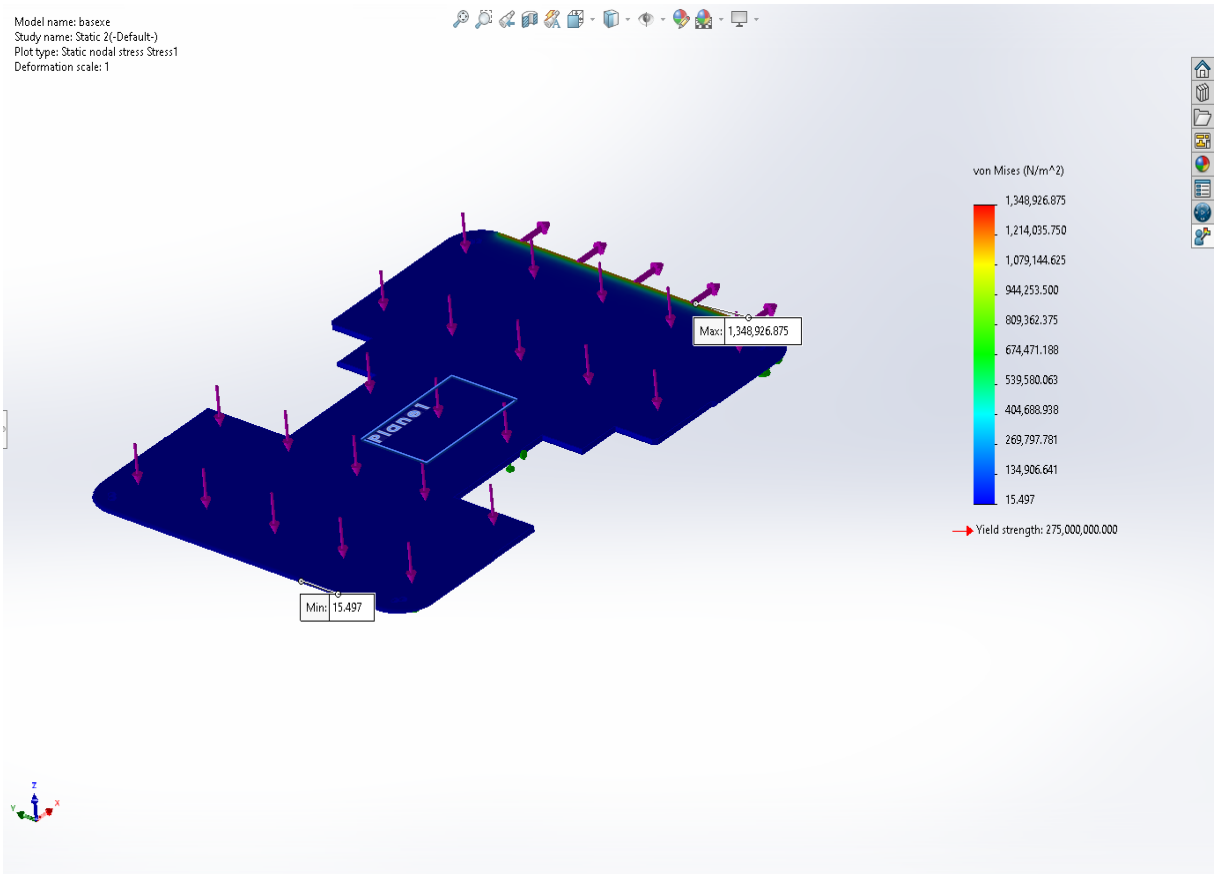


Figure 38 Yield stress of base chasis frame

3.4.2 Motor flange

In this case of examinate, this part was applied force such as:

- Downforce from frame 400 N
- The moment force which created by motor 80 N.m
- The moment force use to connect motor and flange by bolt 8N.m

The results show that structure can be work in this case, which material chosen is AISI 4340 Steel (normalized). The stress condition is satisfied as figure.

$$\sigma_{maximum} = 535{,}089{,}984 N/m^2 < [\sigma_{yield}] = 710{,}000{,}000 N/m^2$$

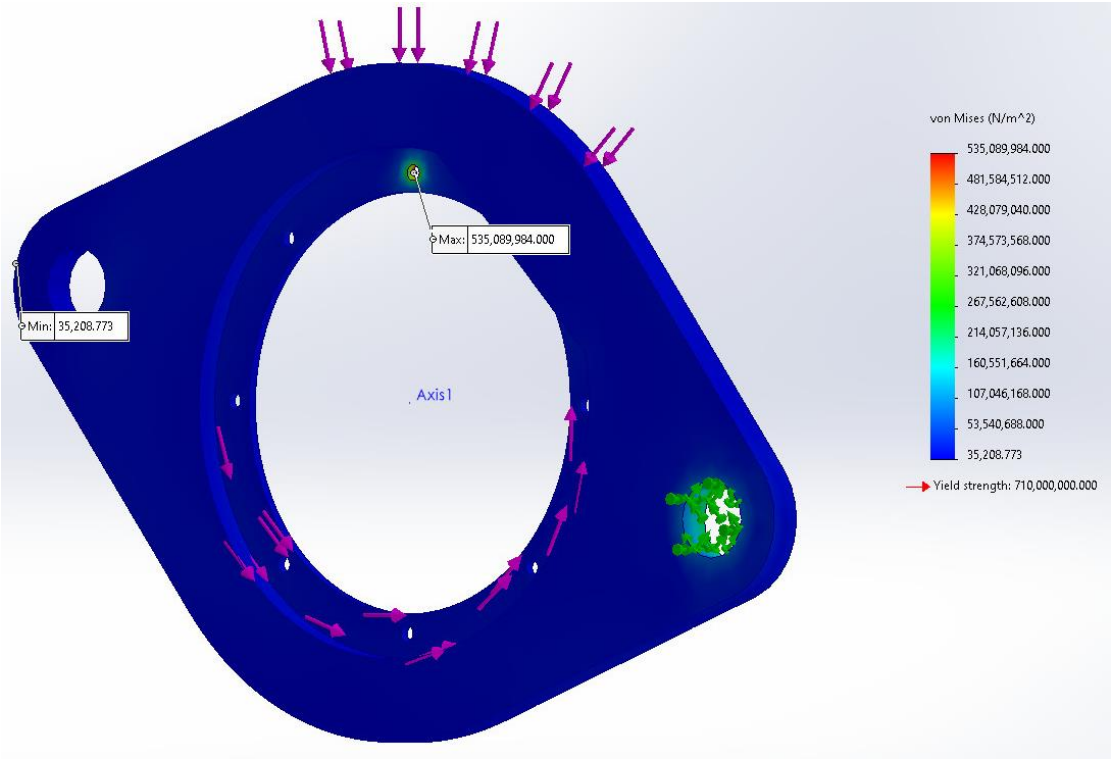


Figure 39 Yield stress of motor flange

3.4.3 Lock carts

In this case of examinate, this part was applied force such as:

- Downforce from cart 40 N
- The force be applied when vehicle tow several carts behind vehicle 600 N
- The moment force use to connect motor and flange by bolt 8N.m

The results show that structure can be work in this case, which material chosen is AISI 4340 Steel (normalized). The stress condition is satisfied as figure.

$$\sigma_{maximum} = 191{,}967{,}008 N/m^2 < [\sigma_{yield}] = 710{,}000{,}000 N/m^2$$

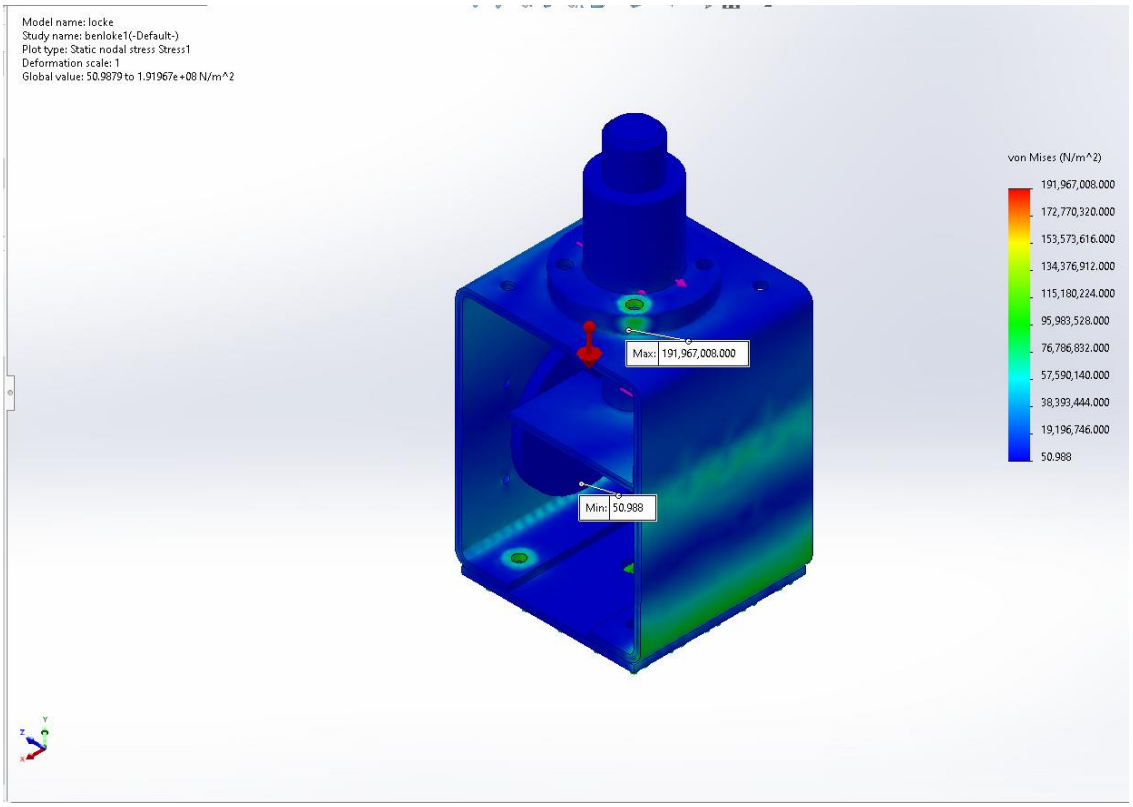


Figure 40 Yield stress of lock carts part

3.4.4 Suspension flange

In this case of examinate, this part was applied force such as:

- Downforce from frame.
- Suspension force.

The results show that structure can be work in this case, which material chosen is AISI 4340 Steel (normalized). The stress condition is satisfied as figure.

$$\sigma_{maximum} = 7{,}299{,}739.000 N/m^2 < \left[\sigma_{yield}\right] = 710{,}000{,}000 N/m^2$$

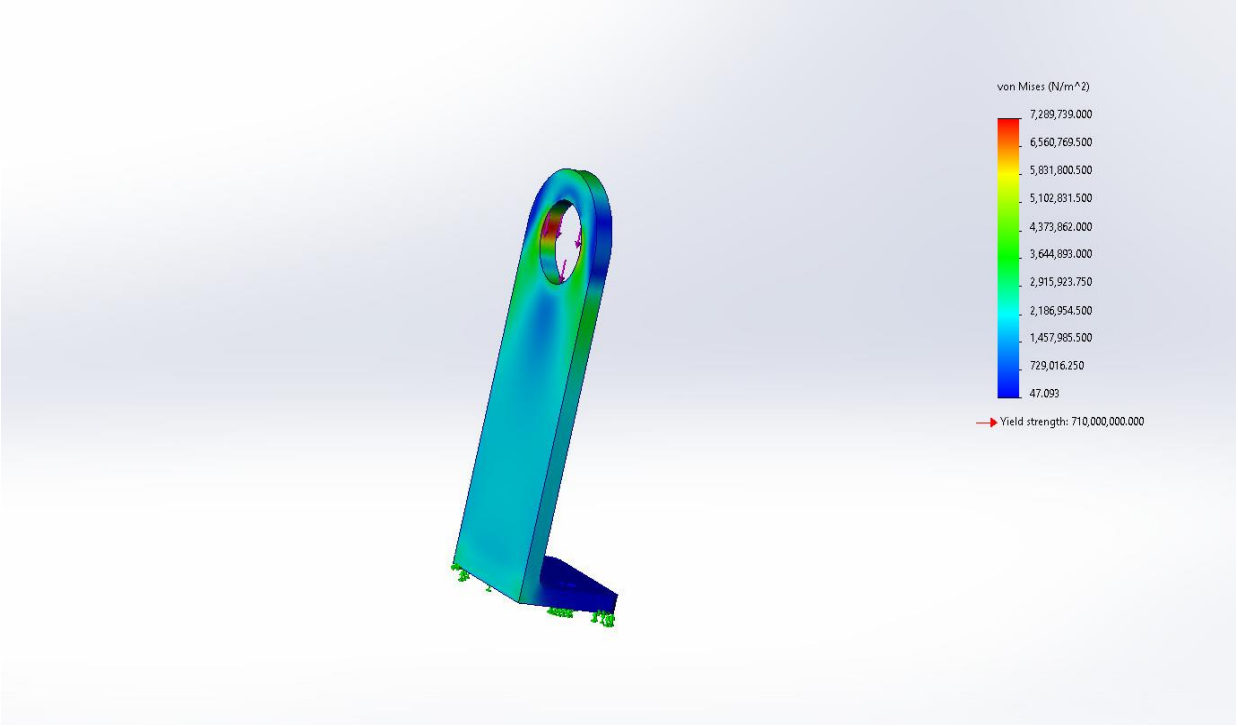


Figure 41 Yield stress of suspension flange part

CHAPTER 4: ELECTRICAL DESIGN

Electrical design can optimize the continuous operating range, period range and executive speed of robot. The electrical device experiment and PCB design for auxiliary board is shown at Appendix A.

4.1 System diagram

The electrical circuit is design based on distributed control structure.

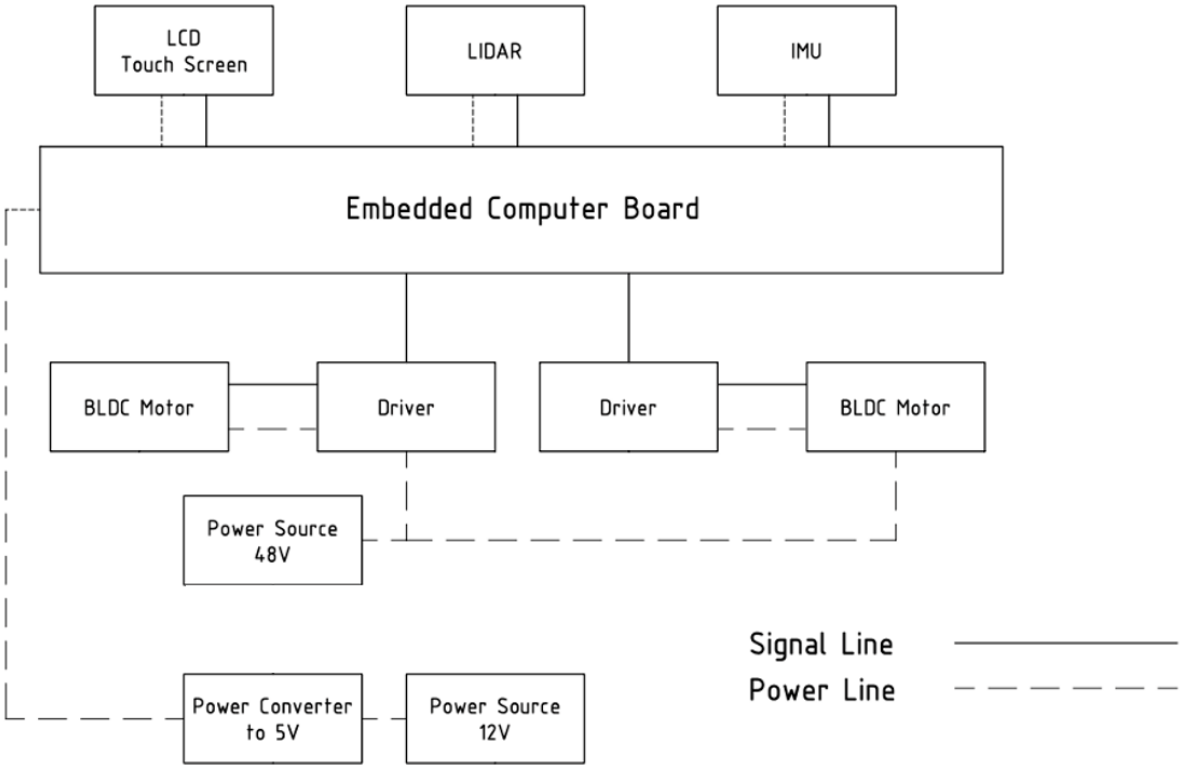


Figure 42 Communication block

4.2 Driver motor selection

Voltage	Peaked current	Stall current	Continuous current at rated speed
24V	37A	23A	14A

Table 18 Wheel drive motor electrical parameter

Based on motor electrical requirement [12], CPP – A24V80A – SA – USB driver motor [16] is chosen, and it has USB communication protocol for set – up and tuning on Window – based software.

Figure 43 CPP – A24V80A driver motor

Input voltage	Current		PWM frequency	Efficiency
	Continuous	Peaked		
12 – 80V	24A	60A	20, 40, 80 kHz	98%

Table 19 Motor driver parameter

This driver has already embedded firmware. The mode can be initialized by computer through USB connection. The Control Loops mode allows the user to adjust the tuning values for current, speed, position and position with speed control modes.

This project requires speed control of 2 wheels to control the speed and direction of robot. The firmware of driver also supports PID control method, but the control gain need to be calculate and initialize first.

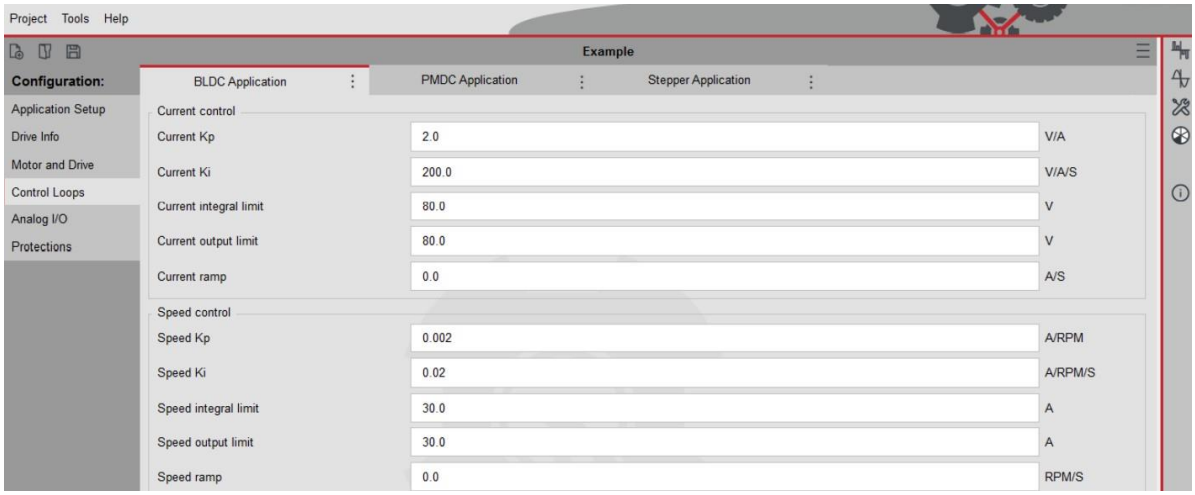


Figure 44 Setup PID gain for speed control in driver firmware

4.3 Sensor

4.3.1 LIDAR sensor

The chosen LIDAR sensor for mapping is Hokuyo UTM-30LX A1. Its parameter is shown

Figure 45 Hokuyo YVT-35LX

Operating parameter	Value	Device parameter	Value
LIDAR type	3D	Sound level	59dB
Scan angle	Horizontal: 210 Vertical:40	Angular resolution	0.25º (360º/1,440 step)
Light source	Laser diode Wavelength = 905mm	Accuracy	15m:±50mm 15m~: ±100mm
Communication	Ethernet (TCP/IP)	Power source	12VDC
Data spots	2590 spots (20 fps) 518000 spots (0.1 fps)	Current consumption	Max – 1.5A Normal – 0.8A

Table 20 Hokuyo YVT-35LX parameter

However, beside choosing Hokuyo LIDAR for the mobile robot, RPLIDAR A1 from SLAMTECH is considerably selected due to similar characteristic, lack of cost and handy experiment.

Its parameter is shown in a table below.

Figure 46 RPLIDAR A1

Operating parameter	Value	Device parameter	Value
LIDAR type	2D	Number of channels	1
Angular range	360^o	Sampling rate	8000 sample/second
Distance range	0.15 – 6m	Scan rate	5.5 Hz
Communication	UART	Sample frequency	≥ 2000Hz
Baud rate	115200	Laser wave length	785 nm

Table 21 RPLIDAR parameter

This sensor uses 3.3V – TTL serial port (UART communication protocol) to send point cloud data to controller. Data frame for this protocol include 4 main sections [17].

Data Type	Unit	Description
Start flag	Boolean	Flag of a new scan
Distance	mm	Current measured distance value between the rotating core of the RPLIDAR A1 and the sampling point
Heading	Degree	Current heading angle of the measurement
Quality	Level	Quality of the measurement

Table 22 LIDAR data protocol

LIDAR sensor experiment:

Target: this experiment is used to determine origin angle to determine the mounting orientation of LIDAR sensor; determine the minimum distance from sensor to robot outer surface.

The procedure is described in detail in Appendix Orientation result:

Figure 47 Origin angle of LIDAR

Therefore, the movement of robot is the origin angle direction.

Distance result: The minimum distance of LIDAR is 11 cm

4.3.2 IMU

The selected sensor for reference the state of the mobile robot is WT901C IMU sensor

Device parameter	Information
Working voltage	3.3 – 5V
Current	< 40mA
Output frequency	0.2 – 200Hz
Interface	TTL
Baud rate	9600 (default, could be changed)

Table 23 WT901C parameter

With this sensor, generated data measurement can minimize the error from LIDAR measurement thanks to 9 DOF (accelerometer, gyroscope, magnetometer, angle/inclinometer) [19]

Figure 48 WT901C IMU sensor

4.4 Controller board

Requirement: High speed processor, has wireless communication, has I/O pins and flash memory, USB communication protocol, HDMI connection

Raspberry Pi embedded computer board is chosen for this project, due to open-source hardware design a detail datasheet, immense speed ARM processor (cortex A72) and suitable cost. Among Raspberry Pi series, the edition 4th of Rasp is satisfied those criteria. This board is compatible with Ubuntu Operating System – which is familiarly supported ROS platform.

Nevertheless, the amount of USB communication port is really few, so the extended USB hub is significantly required to connect LIDAR, 2 Drivers

Figure 49 Raspberry Pi 4 wireless board

Parameter	Value
Processor	Quad core Cortex-A72 (ARM v8)
RAM	2-4-8 GB LPDDR4
PRU	32bit
External ROM	microSD Card Slot
I/O Pins	40
Wi – Fi	2.4 GHz and 5.0 GHz IEEE 802.11ac
Bluetooth	5.0, BLE
USB	4
Communication interface	HDMI, UART, SPI, I2C, CAN

Table 24 Raspberry Pi 4 parameter

4.5 LCD interface

LCD interface is used to indicate the operating process of robot, as well as interact with customer and staff to track the robot and set location for towing goods

Requirement: Touch screen capability to skip a physical keyboard, size is not smaller than 6 inch, has HDMI communication protocol

Figure 50 Waveshare LCD

The Waveshare 7inch LCD touch screen is considerably selected for this project, LCD hardware requires USB host interface and micro HDMI connection embedded computer board. Additionally, software requires Angstrom image firmware file setup to suitable with Raspberry Pi 4

Parameter	Value
Resolution	1024 × 600
Touch point	5 points
Touch type	Capacitive
Viewing angle	170º
Display port	HDMI
Touch port	USB

Table 25 Waveshare LCD parameter

4.6 Power block

4.6.1 Power converter

The switching regulator is chosen to supply energy for controller circuit. This type of regulator has higher noise than linear regulator, but it has better efficiency in comparison. By adding the capacitor filter on board, the noise of switching regulator can be acceptable.

Figure 51 LM2596

Buck converter LM2596 is selected with current load capacity is 3A.

4.6.2 Power source

In order to prevent noise from the back electromagnetic force affect controller board when motor starting, the power source and power circuit is divided into power circuit and control circuit.

4.6.2.1 Power circuit

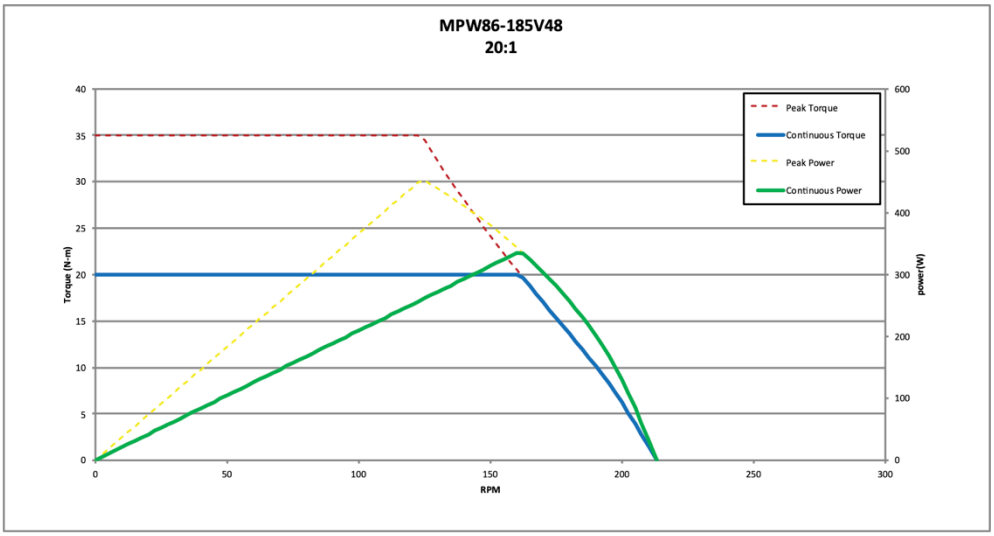


Figure 52 Power consumption of motor

The motor speed is 72 RPM and the power of each motor is 112.8W. So that, Lithium of etaSTORE LTO is chosen due to acceptable criteria [18]

Figure 53 etaSTORE LTO lithium battery

Parameter	Value
Capacity (nominal)	22Ah
Voltage (nominal)	25.3V
Cycles	<17000
Charge current	125A
No of batteries per system	Max 2
IP – Protection	IP53
Charge rate	Bis zu 5C
Size	W x H x D: 19 x 16.8 x 25 (cm)

Table 26 etaSTORE LTO lithium battery parameter

The operating period:

$$T = \frac{2.2 \times 10^3}{(120 + 10) \times 2} \approx 8.46\ hours$$

4.6.2.2 Control circuit

Device	Quantity	Current (mA)	Voltage (V)	Power (W)
LIDAR scanner	1	800	12	9.6
Controller board	1	1010	5	5.1
LCD board	1	30	5	0.15
IMU sensor	1	25	5	0.125
Total		1865		14.975

Table 27 Power consumption of control circuit

For control circuit, GTK Lipo4 Rechargeable Batteries is selected with nominal voltage: 12V capacity: 2200mAh, rechargeable

The operating period:

$$T = \frac{12 \times 2.2}{15} = 17.6\ hours$$

CHAPTER 5: CONTROLLER DESIGN

5.1 Motor modelling

In order to control the velocity of two motors as expectation, the trajectory controller design is implemented to handle the expected criteria. Hence, the robot can track the desired trajectory in the journey – which is generated from the motion path planning algorithm.

5.1.1 Motor transfer

To simplify the driving wheel motor controller design, the transfer function is needed. From the catalogue [12], the characteristic parameters of the motor is:

Parameter	Value
Motor Rotor Inertia	J = 1.949 g.cm^2 = 1.949 x 10^{-4} kg.m^2
Continuous Torque at Rated Speed	τ = 20 Nm
Continuous Rated Wheel Speed	ω = 150 RPM = 15.71 rad/s
Gear Motor Voltage Constant	Ke = 213 V/kRPM = 2.034 V/rad/s
Gear Motor Torque Constant	Kt = 2.04 Nm/Amp
Motor Winding Resistance	R = 0.23 Ω
Motor Winding Inductance	L = 1.36 mH = 1.36 x 10^{-3}H

Table 28 Motor characteristic

Motor viscous friction constant:

$$b = \frac{\tau}{\omega} = \frac{20}{15.71} = 1.273\ N.m.s \quad (5.1)$$

System identification is chosen, additionally, in white box method, because of the fully supplicant of motor parameters.

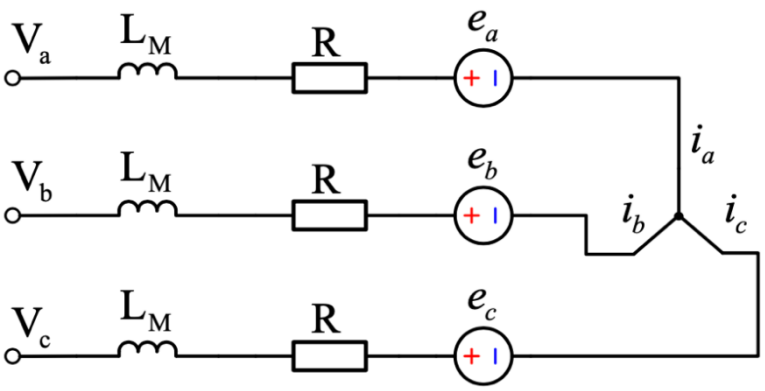


Figure 54 BLDC motor circuit diagram

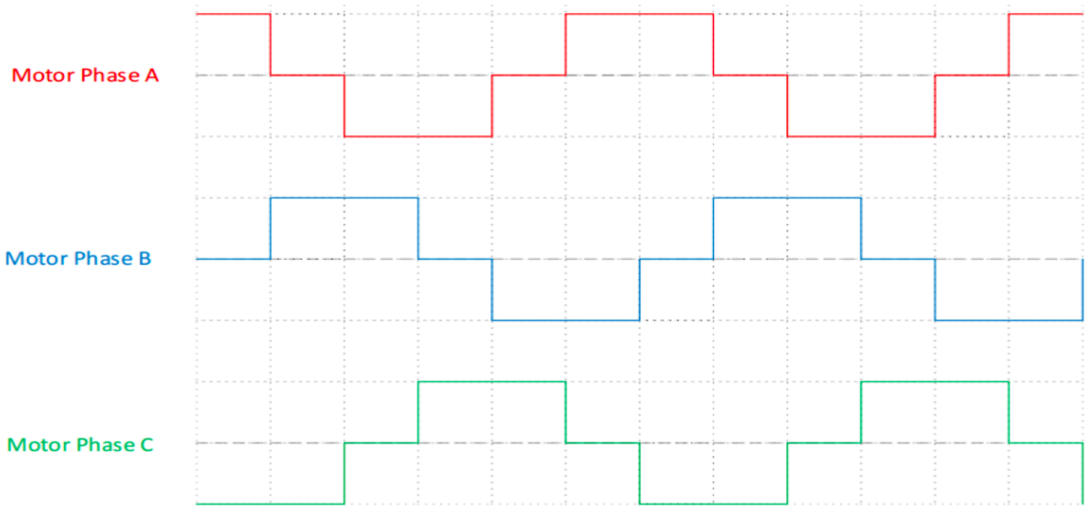


Figure 55 BLCD motor phase

According to figure 2 via catalogue [16] and figure 2.1 in research [20], the BLDC's operating driver uses trapezoidal commutation with hall sensor feedback is at 120 degree mode conducts current in two stator phases. Consequently, only two phases are exited AB, CA or BC, at a time, so the equivalent circuit is illustrated as below:

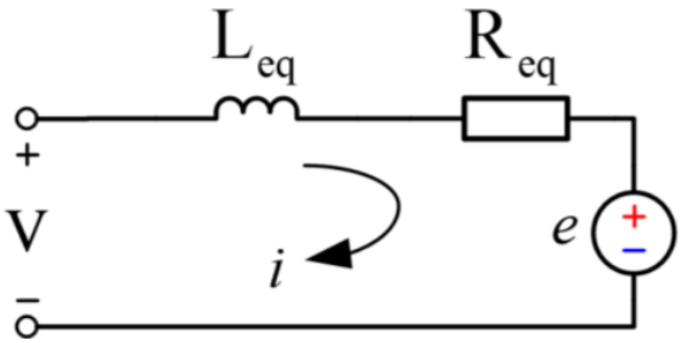


Figure 56 Equivalent circuit diagram of BLDC

Let suppose that magnetic field created from permanent magnet is constant. Besides, the motor control method is armature – controlled motor

Rotor torque: $\tau = K_t i$ (5.2)

Back EMF: $e_b = K_e \dot{\theta}$ (5.3)

Where: i: electrical current flow in armature

θ: rotor angle

Newtons'2nd law:

$$J\ddot{\theta} + b\dot{\theta} = K_t i \rightarrow i = \frac{J\ddot{\theta} + b\dot{\theta}}{K_t} = \frac{J\dot{\omega} + b\omega}{K_t} \quad (5.4)$$

Kirchhoff voltage law:

$$L_{eq}\frac{di}{dt} + R_{eq} i = V - K_e \dot{\theta} \rightarrow V = L_{eq}\frac{di}{dt} + R_{eq} i + K_e \omega \quad (5.5)$$

Where: V: total voltage applied on two phases

ω: angular velocity of rotor

L_{eq}: equivalent inductance $L_{eq} = 2L_M$

R_{eq}: equivalent resistance $R_{eq} = 2R$

Substitute formular (6.4) into (6.5):

$$V = L_{eq}\frac{d}{dt}\left(\frac{J\dot{\omega} + b\omega}{K_t}\right) + R_{eq}\frac{J\dot{\omega} + b\omega}{K_t} + K_e\omega$$

The mathematical model:

$$V = \frac{L_{eq}J}{K_t}\ddot{\omega} + \frac{L_{eq}b + R_{eq}J}{K_t}\dot{\omega} + \left(\frac{R_{eq}b}{K_t} + K_e\right)\omega \quad (5.6)$$

Laplace transform:

$$\frac{\omega(s)}{V(s)} = \frac{K_t}{L_{eq}Js^2 + (R_{eq}J + L_{eq}b)s + (R_{eq}b + K_eK_t)} \quad (5.7)$$

Motor transfer function:

$$G(s) = \frac{1}{2.597 \times 10^{-8}s^2 + 2.137 \times 10^{-4}s + 2.321} \quad (5.8)$$

5.1.2 Motor controller

Design criteria: Maximum overshoot: $OS < 5\%$, steady state error: $e_{ss} < 2\%$, settling time: $t_s < 0.1s$.

From the design criteria, PI controller is implemented due to the reductant of steady state error its bring during satisfied response time. Approach PID controller to deal with PI controller gain.

Figure 57 Block diagram for motor controller

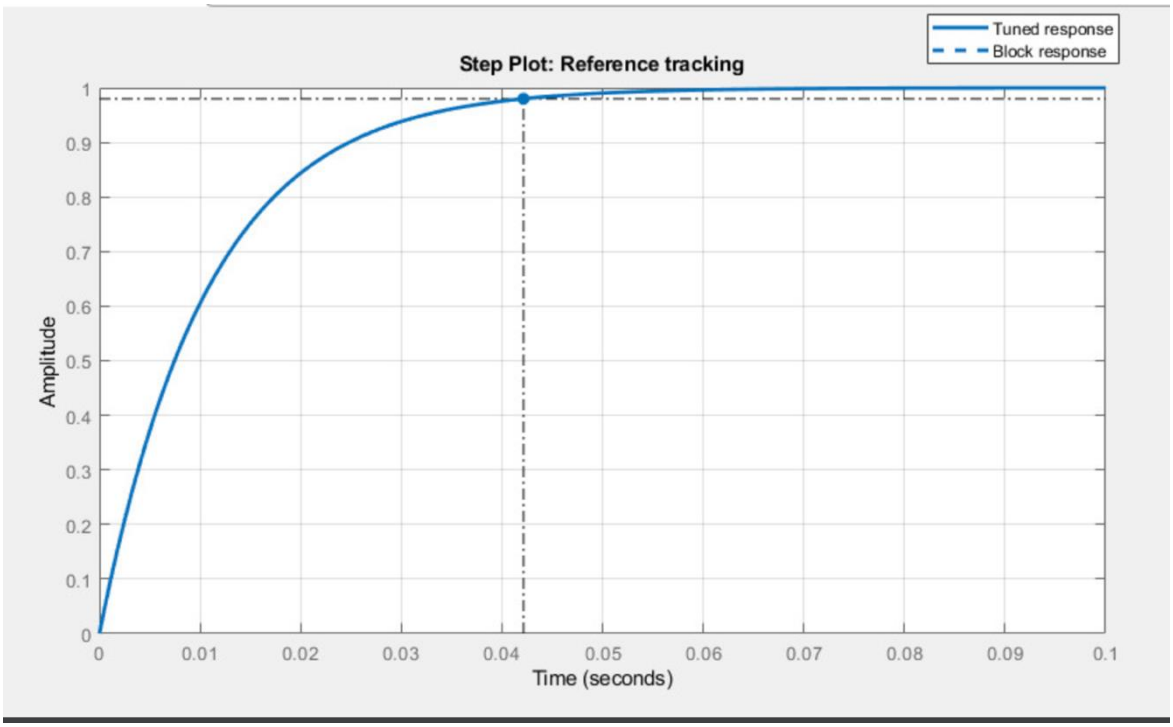


Figure 58 PI controller for motor response

The new response system parameters include: maximum overshoot: $OS = 0\%$, steady state error: $e_{ss} = 0\%$, settling time: $t_s = 0.0421s$

In conclusion, controller gain for motor controller is:

$$K_P = 0.081$$

$$K_i = 215.46$$

5.2 Motion planning

5.2.1 Kinetic model

The 6-wheel robot is design in form of differential drive robot, its model can be simply converted to 2D model as below:

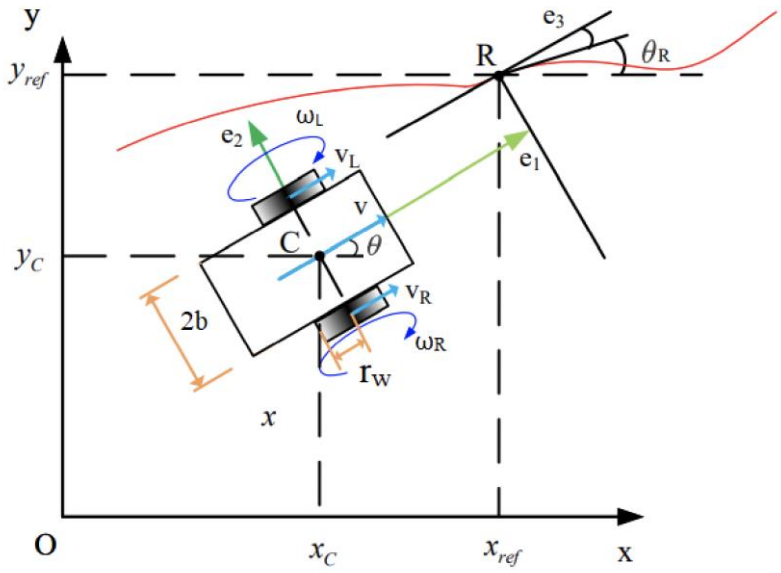


Figure 59 Kinetic model of robot

The velocity and angular equation of left and right wheel is illustrated in the formula as below:

$$v = \frac{(\omega_R + \omega_L) \times r_W}{2} \quad (6.1)$$

$$\omega = \frac{(\omega_R - \omega_L) \times r_W}{2b} \quad (6.2)$$

With: r_W: radius of each wheel

ω_R, ω_L: angular velocity of right and left wheel respectively

From the above kinetic model, the position of mobile robot is achieved from the integral equation:

$$\dot{x}(t) = v cos\varphi \rightarrow x(t) = \int v cos\varphi dt \quad (6.3)$$

$$\dot{y}(t) = v sin\varphi \rightarrow y(t) = \int v sin\varphi dt \quad (6.4)$$

$$\dot{\varphi}(t) = \omega \rightarrow \varphi(t) = \int \omega(t) dt \quad (6.5)$$

In conclusion, the kinetic equation at point C – which is the mid-point between two driving wheel – can be described as:

$$\begin{bmatrix} \dot{x}_C \\ \dot{y}_C \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} cos\varphi & 0 \\ sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (6.6)$$

By determining v and w, tangenal velocity of each wheel can be described as:

$$v_L = v - \omega b. (6.7)$$

$$v_R = v + \omega b. (6.8)$$

For trajectory tracking of the mobile robot, suppose that there is a reference point R at the reference line, with the moving linear velocity v_{ref} and angular velocity ω_{ref}. The kinetic equations at point R can be described as:

$$\begin{bmatrix} \dot{x}_{ref} \\ \dot{y}_{ref} \\ \dot{\varphi}_{ref} \end{bmatrix} = \begin{bmatrix} cos\varphi_{ref} & 0 \\ sin\varphi_{ref} & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} \quad (6.9)$$

The reference trajectory can be calculated from the following equation:

$$v_{ref} = \pm\sqrt{\left(\dot{x}_{ref}\right)^2 + \left(\dot{y}_{ref}\right)^2} \quad (6.10)$$

$$\omega_{ref} = \frac{\dot{x}_{ref} \times \ddot{y}_{ref} - \dot{y}_{ref} \times \ddot{x}_{ref}}{\left(\dot{x}_{ref}\right)^2 + \left(\dot{y}_{ref}\right)^2} \quad (6.11)$$

$$\dot{\varphi}_{ref} = atan2\left(\frac{\dot{y}_{ref}}{\dot{x}_{ref}}\right) + k\pi, k = 0; 1 \quad (6.12)$$

Where: k = 1: forward drive dimension

k = 0: reverse drive dimension

At time k, the state of the mobile robot is: $x_k = [x_k \quad y_k \quad \varphi_k]^T$, and at time (k-1), $x_{k-1} = [x_{k-1} \quad y_{k-1} \quad \varphi_{k-1}]^T$. Therefore, the kinetic equation of the mobile system can be described as follow formula [13]:

$$\begin{bmatrix} x_k \\ y_k \\ \varphi_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \varphi_{k-1} \end{bmatrix} + \begin{bmatrix} cos\varphi_{k-1} & 0 \\ sin\varphi_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} v_{k-1} \\ \omega_{k-1} \end{bmatrix} \quad (6.13)$$

5.2.2 Kalman filter

The Kalman filter model assumes the true state at time k is transformed from the state at time (k-1) according to the equation below [25]:

$$x_k = F_k x_{k-1} + B_k u_k + w_k \quad (6.14)$$

Where: F_x: the state trasition matrix for obtaining the current state x_k by the previous state x_{k-1}

B_k: the control matrix applied to calculate the variable quantity of state

w_k: the process noises

At time k, the measured value of the system is calculated as follow:

$$y_k = C_x x_k + v_k \quad (6.15)$$

Where: C_k: the observation matrix – turning the true state vector into the observed vector

y_k: the observation vector of the system

v_k: the measurement noise vector, its covariance is R_k – Gaussian white noise

Corresponding to the covariance of x_k, can use the equation below to illustrate:

$$P_k = F_k P_{k-1} F_k^T + Q_k \quad (6.16)$$

Where: Q_k: the white noise of the system process

P_k: the forecast estimation matrix

Base on the combined predictions and measured values, so can get the optimal state $x_{k \mid k}$ of the current state:

$$x_{k \mid k} = x_k + K_k(y_k - C_k x_k) \quad (6.17)$$

Where: K_k: the Kalman gain

$$K_k = P_k C_k^T / (C_k P_k C_k^T + R_k) \quad (6.18)$$

Until now, the optimal state $x_{k \mid k}$ in k state has been obtained. But to achieve to keep the Kalman filter running till the end of the system process, the covariance of $x_{k \mid k}$ should be updated:

$$P_{k \mid k} = (I - K_k C_k) \times P_{k-1} \quad (6.19)$$

5.2.3 Error model of system

When the robot is tracking the reference trajectory, several errors will occur in x, y and φ. Hence, these oscillations can be described as:

$$\begin{bmatrix} e_x \\ e_y \\ e_\varphi \end{bmatrix} = \begin{bmatrix} x_{ref} - x_C \\ y_{ref} - y_C \\ \varphi_{ref} - \varphi_C \end{bmatrix} \quad (6.20)$$

However, these parameters are referenced in the base frame coordinate. Consequently, the transformation of them has to be implemented from the base frame coordinate to the robot coordinate. And the transformation will be converted below:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} cos\varphi & sin\varphi & 0 \\ -sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\varphi \end{bmatrix} \quad (6.21)$$

The error model of this system is derived from its position error and robot kinematic model [21]:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} v_{ref} cose_3 \\ v_{ref} sine_3 \\ \omega_{ref} \end{bmatrix} + \begin{bmatrix} -1 & e_2 \\ 0 & -e_1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} v_C \\ \omega_C \end{bmatrix} \quad (6.22)$$

In this model, the consolidation of feedforward control and linear feedback control based on LQR approach is proposed. In addition, linearization approach around robot's operation point is implemented to because this is the non-linear system. (Operating point: $e_1 = e_1 = e_1 = 0$) [21]. The linear model of error is shown as bellow:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & \omega_{ref} & 0 \\ -\omega_{ref} & 0 & v_{ref} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{cl} \\ \omega_{cl} \end{bmatrix} \quad (6.23)$$

With nonlinear transformation of velocity inputs:

$$v_{cl} = v_{ref} cose_3 - v_C \quad (6.24)$$

$$\omega_{cl} = \omega_{ref} - \omega_C \quad (6.25)$$

Hence, the closed-loop system has three state variables e_1, e_2 and e_3, and two inputs v_{in} and ω_{in}

5.2.4 Controller design

Design criteria: maximum error 2 is 15mm, maximum error 3 is 10º, error 1 is 0 – the mobile robot runs with unchangeable velocity.

LQR controller is chosen to make those errors of system to become 0. The controller model is: $u_{LQR} = -K_{LQR}x$

Kalman filter is selected to determine the real state of the system and make many errors of system to become almost 0. The controller model is $u_{Kal} = -K_{Kal}x$

5.2.4.1 Controllability

Since the linearized system is time-varying, the full-state feedback controller required system has to be controllable. Thus, the trajectory controllability must be achieved with the non-holonomic constraint vehicle [22]:

$$C = [B \quad AB \quad A^2B] = \begin{bmatrix} 1 & 0 & 0 & 0 & -\omega_{ref}^2 & v_{ref}\omega_{ref} \\ 0 & 0 & -\omega_{ref} & v_{ref} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.26)$$

When v_{ref} and w_{ref} is constant and nonzero, and the whole system is clearly satisfied with the controllability condition: $rank(C) = 3$

Hence, the kinematic system can be locally stabilized by linear feedback about trajectories executed with constant velocity

5.2.4.2 LQR optimal

LQR method is chosen to take the various location of pole in the way such that the quadratic cost function: $J_{LQR} = \int_0^\infty [x(t)^T Q x(t) + u(t)^T R u(t)]\, dt$ is minimized with the aims to optimal the control law: $u(t) = -K_{LQR}x(t)$

Where $K_{LQR} = R^{-1}B^T P$ is the optimal state feedback gain matrix, which is evaluated by handling the Ricattis' equation: $PA + A^T P + Q - PBR^{-1}B^T P = 0$

Hence, in addition, in order to get the inputs of closed-loop equation, these parameters are calculated as below:

$$\begin{bmatrix} v_{cl} \\ w_{cl} \end{bmatrix} = -\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \quad (6.21)$$

Therefore, it can be observed that the tracking performance of the system depend on the weight of matrix Q and R via optimal gain K_{LQR}. Moreover, trial and error method are often implemented to estimate the gain K_{LQR} base on the system performance [23]. By using MATLAB simulation, from matrix (6.23), the value of each matrix is:

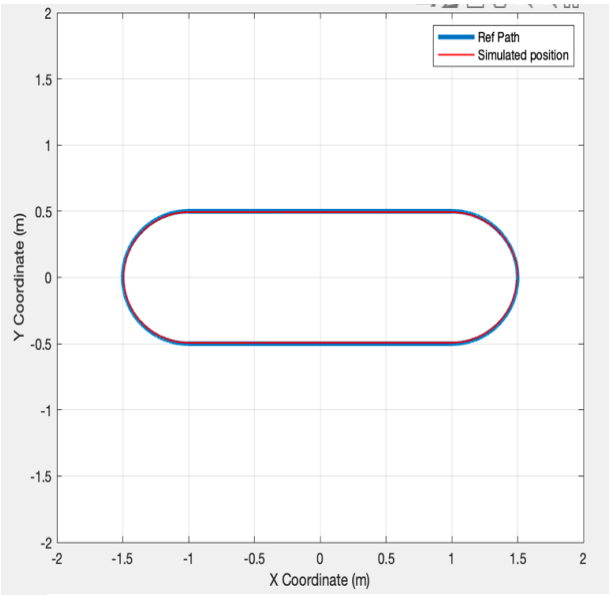
$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 28 \end{bmatrix}; R = 0.01 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

From the value of matrix Q, it can be easily observed that three value in the main diagonal illustrate the value of variables e_1, e_2 and e_3. Thus, by modifying these components, the various sensitivity of the system is fluctuated. The elements of the main diagonal of matrix R, belong to the control input v_{in} and w_{in}

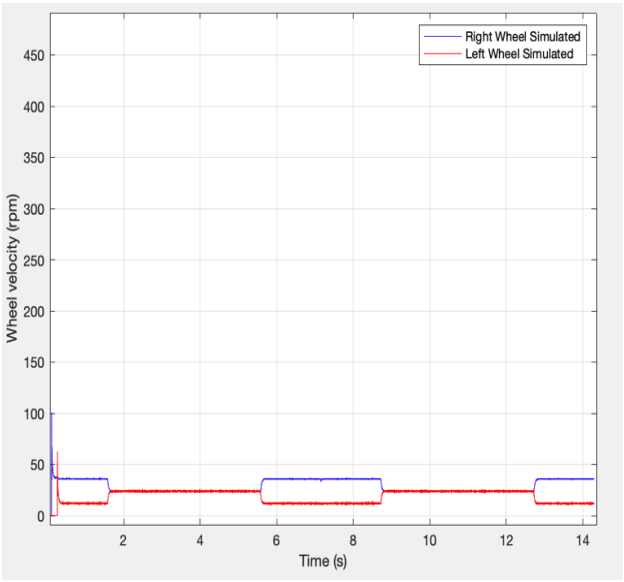
The control of LQR controller:

$$K_{LQR} = \begin{bmatrix} 30.6094 & -55.9408 & -1.5013 \\ -1.5013 & 5.5574 & 53.0511 \end{bmatrix}$$

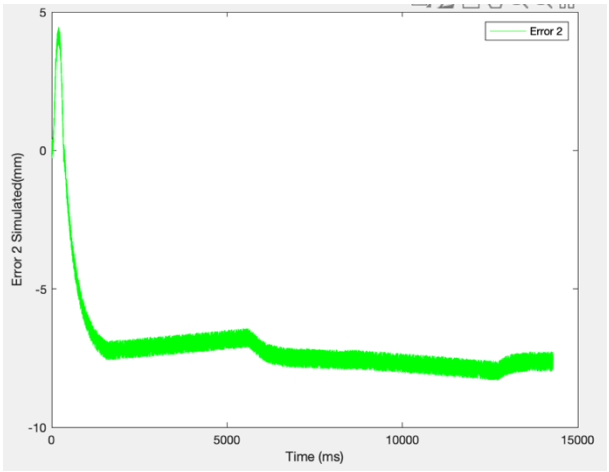
5.3 Simulation result



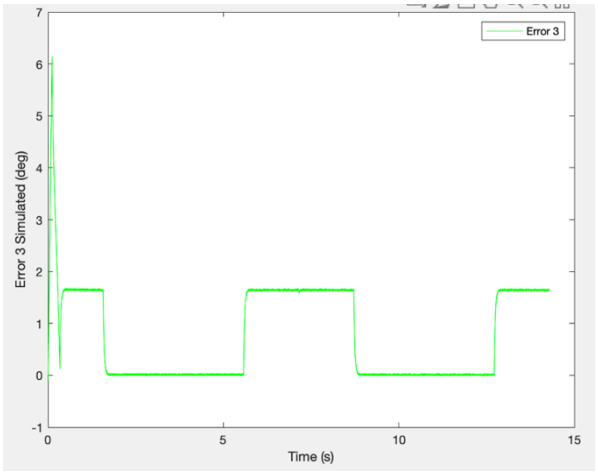
(a)Reference map



(b) Motor speed



(c) Tracking error 2



(d) Tracking error 3

In the experiment, there is a plenty of trajectories which robot can follow; however, this simulation just only takes curve and straight path into account – which is popular with practicing. Hence, the controller gain can adjust the robot performance relatively to the desired criteria.

The simulation path (red path) in figure 6.2a has tracked very well compare to the reference trajectory. The tracking error 2 and error 3 are adjusted random noise to test

(IMU sensor has significant noise in practice), but the desired performance is still acceptable refer to the requirement.

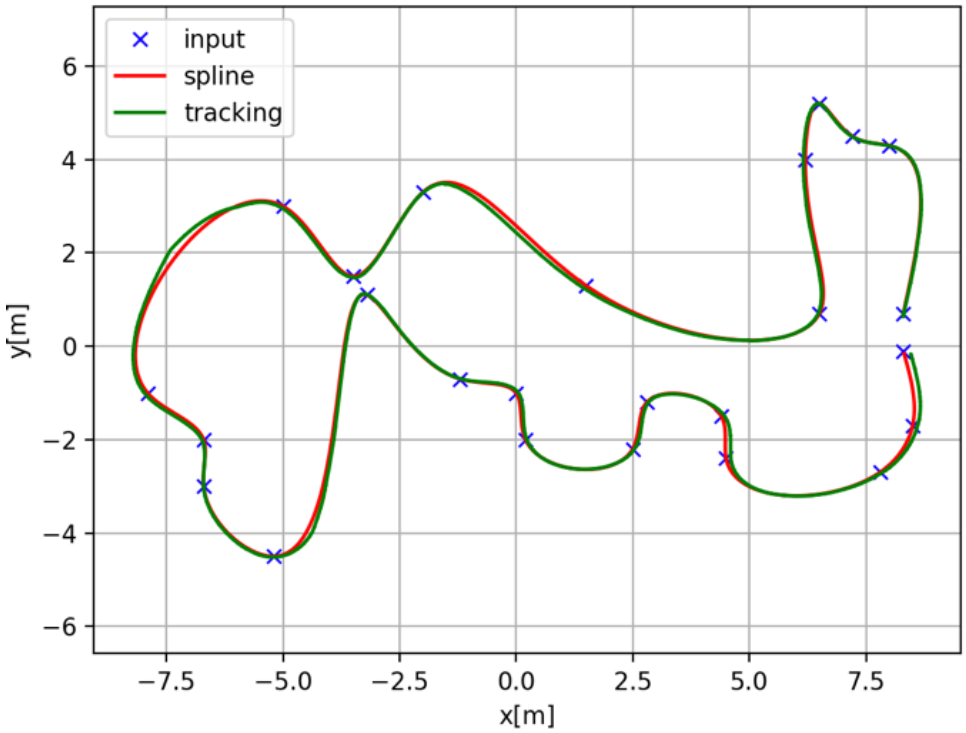


Figure 60 Simulation in complicated environment and unpredicted noise

5.4 Control model scheme

The control model of this system consists of 3 fundamental parts: in the first part (green box), coordinates and angle of the mobile robot will be obtained from input angular velocity of wheels, then in the second part (yellow box), angular velocities required for the first part of the system with the goals of the best possible tracking of reference trajectory will be generated, and the last is the red one (third part), the state of the robot will be calibrated with the sensors and be feedbacked to compare with the desired state after applying the filter.

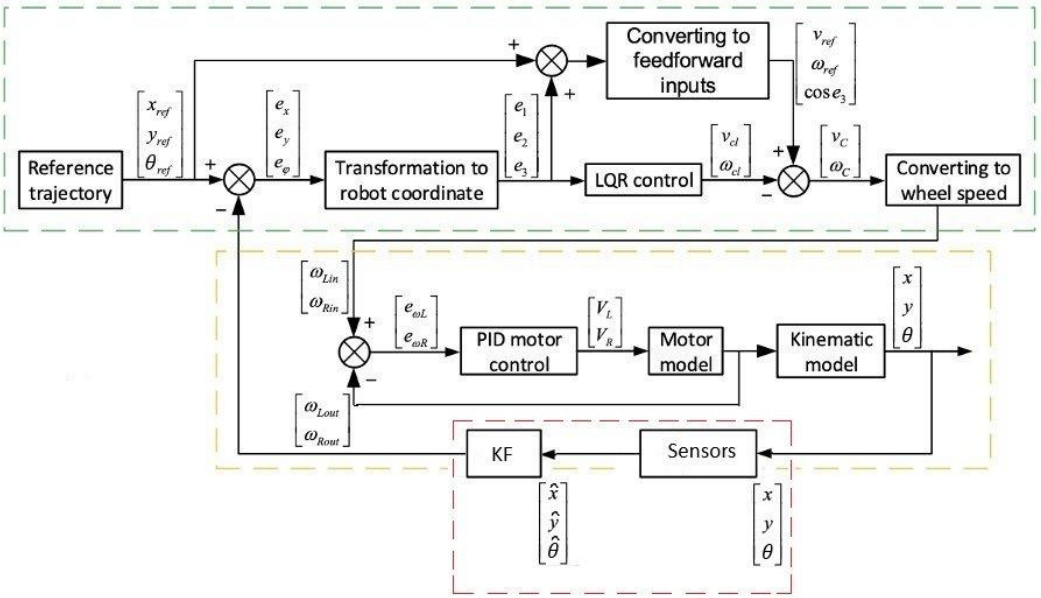


Figure 61 The schematic diagram of robot controller

The reference path is generated from path planning algorithm, based on a given map from SLAM algorithm

CHAPTER 6: PATH PLANNING ALGORITHM

At first, before applying algorithm, position in the map is the most significant criteria that need to be considered. Therefore, seeking that point is able to be found by SLAM algorithm (Simultaneous Localization and Mapping).

Secondly, the factory environment is classified into dynamic environment because of shifting obstacles (such as human, movable goods,…). Thus, dynamic path finding has to be glanced at considerably.

6.1 Localization and mapping

The basis of SLAM method is to estimate the locolization of desired object into a map and generate a map rely on its position and orientation synchronously. In addition, the purpose of laser sensor is use to generate a 2D occupancy grid map. This term is for helping mobile robots tackle some noisy and uncertain measurement of sensor data

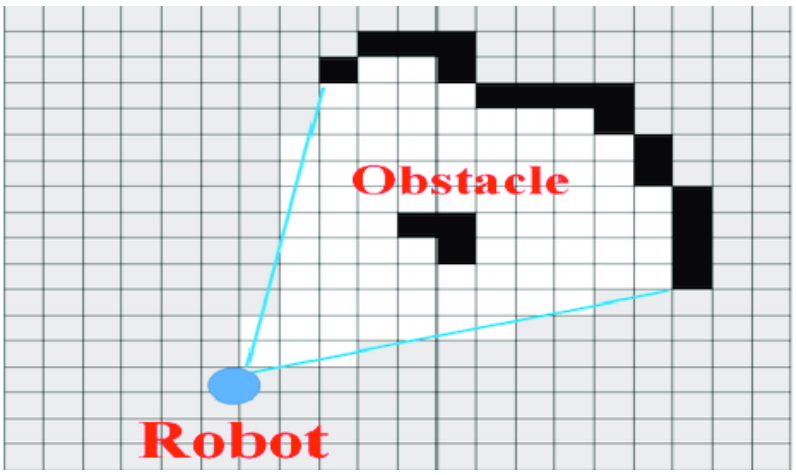


Figure 62 Example of Occupancy grid map

Where: Unknown (gray scale): unexplored area

Occupied (black scale): must-avoided area

Empty (white scale): free area for moving

Gmapping algorithm is one of the laser-based SLAM algorithms using position with the combination of orientation in order to create grid map. The main point is using Rao–Blackwellized particle filters (RBPFs) to forecast the state transition function.

There are 3 fundamental input data below which is put into this algorithm with the intention of lessening the ambiguous and increasing the accuracy of the outcome:

- LiDar sensor: provides scan data, which can witness the surrounding environment for estimating the position in the desired map.
- Hall sensors: located in BLDC motor provides moving orbit, which is used to calculate the distance from the original location
- IMU: provides angular orientation of the robot

The algorithm parameters in the period time 1 to t:

- Robot control signal: $u_{1:t} = \{u_1, u, u_3, \dots, u_t\}$
- Observations from sensor: $z_{1:t} = \{z_1, z_2, z_3, \dots, z_t\}$

The desired outcome parameters:

- The grid map denoted by m
- Robot trajectory parameter (robot pose): $x_{1:t} = \{x_1, x_2, x_3, \dots, x_t\}$

However, in practice, some various errors are not evaded from the measurement of sensors data. Consequently, the output cannot match with the desired value – position. Thus, the probability approach is implemented to determinate the probability distribution of each pose: $p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$

With: p: probability distribution

$x_{1:t}$: robot motion

m: map

$z_{1:t}$: observation data from sensor

$u_{1:t}$: control signal

Applying Bayes' theorem, the distribution is transformed as below:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) = p(x_{1:t} \mid z_{1:t}, u_{1:t})\, p(m \mid z_{1:t}, u_{1:t})$$

With: $p(x_{1:t} \mid z_{1:t}, u_{1:t})$: is the position issue

$p(m \mid z_{1:t}, u_{1:t})$: is the map issue

The principle of Gmapping algorithm:

- Sampling: each sampling time (base on LIDAR sensor frequency) the robot pose x_t is sampled into a list and a new propose distributions are generated.

- Weight: each of current robot pose x_t^i is get a weight calculation:

  $w_t^i = p(z \mid x_t^i)$
- Resampling: depending on the weights, particles with smaller weights are discarded and replaced by resampled particles, but the total number of particles in the resampled particle set is unchanged.
- Map updating: the map update is implemented by the pose represented by each particle in combination with the current observation.

According to the method of this algorithm, in process, the robot should be hand-operated control with feedback signal and slow velocity around the desired workspace. Thus, the resampling and map updating can avoid severally uncertain errors and get more misallocation pose.

6.2 Path planning

The factory is a dynamic environment. Therefore, the robot has to face with some problems below:

- Unpredictable obstacle avoidance during operation
- Optimal path while moving to expected goal

Thus, using 2 algorithms is required to handle those problems. The mission of global planner is smooth path calculation from starting point to target position and local planner is used for evading some abrupt obstacles exist in the planning path in the map

6.2.1 Configuration space

Although the robot implements on the practice environment, it also deals with in the configuration space (C_{space}). The C_{space} consider all of position of robot and also including the obstacle, in order to make the collision free. The A-star and Dijkstra algorithm bases on the grid (based-planner technique) so the C_{space} can be discretized. The workspace in C_{space} is denoted by number of dimensions: $W = R^m$. The set of obstacle region is $O \in W$ and the robot configuration is $A(q)$, $q \in W$

The workspace contains free space and obstacle regions:

$$C_{free} = \{q \in C \mid A(q) \cap O = \varnothing\}$$

$$C_{obstacle} = C / C_{free}$$

6.2.2 Global planner

6.2.2.1 Dijkstra algorithm

The Dijkstra algorithm is named after a computer scientist - Edsger W. Dijkstra discovered in 1956 and published 3 years later, this is an algorithm that solves the problem of finding the shortest path between two nodes in term of positive weighted graphs.

- **Principle of Dijkstra algorithm:**

Dijkstra is formulated in terms of weighted graphs: from a initial node, as its purpose is finding the shortest path to target node (minimum distance, minimum travelled time,…). This can be done by maintaining a tree of paths from the starting node and extends the paths along the edges until the criteria to the goal is satisfied.

At each of its main loops, it is essential to determine which path to the next node should be chosen. Specifically, for each current node under consideration (current node), we calculate the costs $g(n)$ from the starting node to the neighbors (neighbors) of the 'current node' and then select the node with $g(n)$ minimum, this node will also be the 'current node' of the next loop. With 'start node' as first 'current node', keep repeating the above process until we get 'current node' followed by destination node then stop, then we get the shortest path equal to how to go back to the nodes before the 'current nodes'.

The value $g(n)$ of a node is calculated by the below formula:

$$g(n) = g(c) + d(c,n)$$

Where: $g(n)$: cost from ‘start node’ to ‘neighbor’ of ‘current node’

g(c): cost from ‘start node’ to ‘current node’

d(c,n): cost from ‘current node’ to neighbor’

A typical implementation of this algorithm is to use a priority queue. At each step of the algorithm, the node with the least cost g is removed from the queue, and the value $g(n)$ of the neighboring 'neighbors' is updated and added to the queue. The algorithm continues until it reaches a target node whose cost g is less than any other node in the queue (or until the queue is empty).

- **The advantages and disadvantages:**

The Dijkstra algorithm is easy to implement in any programming language because of its simplicity, so it is also the basis for creating many variations for different applications. However, this is not an optimal algorithm in terms of processing speed and takes up a large amount of memory at runtime.

- **Algorithm structure:**

```
function Dijkstra(Graph, source):
    for each vertex v in Graph:           //Initialization
        dist[v] := infinity               //Initial distance from source to vertex v is set to infinite
        previous[v] := undefined          //Previous node in optimal path from source
    dist[source] := 0                     //Distance from source to source
    Q := the set of all nodes in Graph    //All nodes in the graph are unoptimized – thus they are in Q
    while Q is not empty:
    u := node in Q with smalles dist[]
    remove u from Q
    for each neighbor v of u:             //Where v has not yet been removed from Q
        alt := dist[u] + dist_between(u,v)
        if alt < dist[v]                  //Relax (u,v)
            dist[v] := alt
            previous[v] := u
    return previous[]
```

Figure 63 Dijkstra pseudocode

1. Initialization of all nodes with distance “infinite”; initialization of the starting node with 0
2. Marking of the distance of the starting node as permanent, all other distances as temporarily
3. Setting of starting node as active
4. Calculation of the temporary distances of all neighbor nodes of the active node by summing up its distance with the weights of the edges
5. If such a calculated distance of a node is smaller as the current one, update the distance and set the current node as antecessor. This step is also called update and is Dijkstra’s central idea
6. Setting of the node with the minimal temporary distance as active. Mark its distance as permanent.

7. Repeating of steps 4 to 7 until there aren't any nodes left with a permanent distance, which neighbors still have temporary distances

6.2.2.2 A* algorithm

Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research Institute (now SRI International) first published the algorithm in 1968. It can be considered an extension of Edsger W. Dijkstra's algorithm. A* achieves better performance by using heuristics to guide the search.

The A* algorithm is the path searched algorithm. In addition, A* algorithm only finds the shortest path from a specified source to a specified goal, and not the shortest-path tree from a specified source to all possible goals. This is a essential trade-off for using a specific-goal-directed heuristic.

- **Principle of A* algorithm:**

Similar to Dijkstra's algorithm. The most important difference between these two algorithms is the calculation of the total cost used to select the next node of the 'current node', which is also the 'current node' of the next loop. A* has the additional cost of guessing $h(n)$ in the formula, which evaluates how far away the node is from the target node and gives preference to the node closer to the target, thereby greatly reducing the number of iterations to access the target node. output the shortest path. Thus, the path to the next node selected must have the minimum cost $f(n)$ calculated by the below formula:

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

Where: $g(n)$: cost from 'start node' to 'neighbor' of 'current node'

$h(n)$ - heuristic function: estimate cost of nearest path from 'neighbor node' to 'target node'

The implementation of this algorithm also uses a priority queue like Dijkstra's algorithm. At each step of the algorithm, the node with the least cost f is removed from the queue, and the value $f(n)$ of the neighboring 'neighbors' is updated and added to the queue. The algorithm continues until

it reaches a target node whose cost f is less than any other node in the queue (or until the queue is empty).

+ **Heuristic cost**:

The cost of $h(n)$ would ideally equal the exact cost of reaching the target. However, it is impossible because of the unknown path. According to reference [29], there are 2 methods to calculate heuristic cost, such that improve the algorithm performance:

.The Manhattan Distance Heuristic:

$$h = |x_{start} - x_{destination}| + |y_{start} - y_{destination}|$$

.The Euclidean Distance Heuristic:

$$h = \sqrt{(x_{start} - x_{destination})^2 + (y_{start} - y_{destination})^2}$$

The Euclidean Distance Heuristic approach has more accuracy but it slower than its counterpart because of exploring a larger area condition, but the Manhattan Distance Heuristic is used for A* algorithm due to its acceptable errors.

- **The pros and cons:**

Because of the same principle as Dijkstra's algorithm, the A* algorithm is also very easy to implement and gives faster processing speed if the appropriate heuristic function is chosen. Although somewhat improved in terms of resources and speed compared to Dijkstra's algorithm, the choice of heuristic function also affects the efficiency of the algorithm that the implementer wants. Specifically, in order to choose a heuristic function to make the algorithm faster, there may be a trade-off that the path found may not be the shortest (in terms of cost), and vice versa to be sure to find the shortest-cost path. The best algorithm is accepted, the algorithm takes longer. Notice that when $h(n)$ = 0 in every loop - $f(n)$ = $g(n)$, the A* algorithm turns into Dijkstra.

- **Algorithm structure:**

```
Put node_start in the OPEN list:
f(node_start) = h(node_start)          //Initialization
while the OPEN list is not empty:
      f(node_current) = g(node_current) + h(node_current)    //Take the node_current lowest from open list
      if node_current is node_goal    break;
      Generate each state node_successor that come after node_current
      For each node_successor of node_current
            Set successor_current_cost = g(node_current) + ω(node_cureent, node_successor)
      if node_successor in OPEN list:
      if g(node_sucessor) ≤ sucessor_current_cost
      Add node_current to CLOSED list
      else if node_successor in CLOSED list
      if g(node_sucessor) ≤ sucessor_current_cost
      Add node_current to CLOSED list
      Move node_successor from CLOSED list to OPEN list
      else   add node_successor to OPEN list
                  set h(node_successor) to be the heuristic distance to node_goal
      Set g(node_successor) = successor_current_cost
      Set the parrent of node_successor to node_current
Add node_current to CLOSED list
if node_current != node_goal   exit with error (the OPEN list is empty)
```

Figure 64 A algorithm pseudocode*

With: Open list consists on nodes that have been visited but not expanded, and it is the list of pending tasks. Close list consists on nodes that have been visited and expanded (successors have been explored already and included in the Open list). Node_successor is the one of 8 node surround node_current (not include the node in obstacle region).

6.2.3 Local planner

6.2.3.1 DWA algorithm

With the criteria to make the robot capable of autonomous obstacle avoidance, dynamic window approach (DWA) algorithm is chosen to become the local path planning algoritßhm. In each section of following global path, if obstacles are detected which don't exist in the known map, they can block the path planning robot. In addition, the robot should apply DWA to handle those obstacles – avoid them and generate the new path such that it is collision free. Then, after passing the obstacles, following the optimal path created by global planner is implemented again.

However, the DWA algorithm cannot generate the optimal path compare to global planner algorithm, and problem of falling local minimum can be occurred. Hence, it is suitable for local planner.

This aim of this algorithm is to transform the path planning issue into a constrained optimization problem on the velocity vector space. Multiple sets of velocities in the velocity space (v, ω), where: v: linear velocity of robot and ω: angular velocity of robot.

- **Algorithm formula:**

The limitation of maximum and minimum velocity of robot [24]:

$$V_m = \{(v, w) \mid v_{min} \le v \le v_{max}, w_{min} \le w \le w_{max}\}. \quad (6.1)$$

The limitation of acceleration and deceleration of robot. This value base on torque and performance of motor characteristic:

$$V_d = \left\{(v, w)\middle|\begin{matrix} v \in [v_c - \dot{v}_b \Delta t, v_c + \dot{v}_a \Delta t], \\ w \in [w_c - \dot{w}_b \Delta t, w_c + \dot{w}_a \Delta t] \end{matrix}\right\} \quad (6.2)$$

With: v and w: the current linear velocity and angular velocity

$\dot{v}_b$ and $\dot{w}_b$: the maximum deceleration of the linear velocity and angular velocity

$\dot{v}_a$ and $\dot{w}_a$: the maximum acceleration of the linear velocity and angular velocity

In order to improve safety requirement, the robot must stop before encountering an obstacle.

$$V_a = \left\{(v, w)\middle|\begin{matrix} v \le \sqrt{2 \times dis(v, w) \times \dot{v}_b}, \\ w \le \sqrt{2 \times dis(v, w) \times \dot{w}_b} \end{matrix}\right\} \quad (6.3)$$

With: $dis(v,\omega)$: the shortest distance from the robot to the obstacle on the corresponding trajectory of the velocity (v, ω).

The final sampling velocity of robot speed is the intersect of the above formulas:

$$V = V_m \cap V_d \cap V_a \quad (6.4)$$

The path planning performance results depending on the sampling velocity resolution. In the sampled speed group, there are several feasible trajectories. Therefore, it is required to evaluate the corresponding trajectory by evaluation function below (equation 6.5).

The greater this variable, the greater probability of a collision, so the trajectory move around obstacle is priority:

$$H(v,w) = \alpha \times ang(v,w) + \beta \times vel(v,w) + \gamma \times dis(v,w) \quad (6.5)$$

With: $ang(v, \omega)$: the azimuth deviation between the end direction of the simulated trajectory and the goal:

$$ang(v,w) = 1 - \frac{\theta}{\pi} \quad (6.6)$$

With: θ: the angle between car and the target point

$vel(v,w)$: is the forward velocity of the car and supports fast movements

$$vel(v,w) = \frac{v}{v_{max}} \quad (6.7)$$

$$dis(v,w) = \begin{cases} \frac{1}{L}, 0 \le l \le L \\ 1, L \ll l \end{cases} \quad (6.8)$$

With:

L: the distance between robot position and the target one at the moment. The less the value, the likely to be collision

- **Algorithm structure:**

```
function DWA(robotPose, robotGoal, robotModel)
    desiredV = calculateV(robotPose, robotGoal)
    laserscan = readScanner()
    allowable_v = generateWindow(robotV, robotModel)
    allowable_ω = generateWindow(robotW, robotModel)
    for each v in allowable_v
        for each ω in allowable_ω
            dist = find_dist(v, ω, laserscan, robotModel)
            breakDist = calculateBreakingDistance(v)
            if (dist > breakDist)          //can stop in time
                heading = hDiff(robotPose, goalPose, v, ω)
                clearance = (dist – breakDist)/(dmax – breakDist)
                cost = costFunction(heading, clearance, abs(desired_v – v))
                if (cost > optimal)
                    best_v = v
                    best_ ω = ω
                    optimal = cost
    Set robot trajectory to best_v, best_ ω
```

Figure 65 DWA algorithm pseudocode

First of all, the desired velocity to the goal being calculated based on our current position, and the destination

Secondly, the allowable velocities are selected following by formula (6.2). Each allowable velocity sets can create each trajectory.

Thirdly, for each velocity, determine the closest obstacle for the proposed robot velocity following by formula (6.8)

Fourthly, determine if the distance to the closest obstacle is within the robots breaking distance. If the robot will not be able to stop in time, disregard this proposed robot velocity following by formula (6.3). Otherwise, the velocity is 'admissible', so the required value is calculated (heading and clearance value)

Finally, the evaluation function is calculated for the proposed velocity, to selected the optimal path.

6.3 Fusion 2 path planning

Each global and local path will be generated by the global and local path planning algorithm. Therefore, a fusion step is required to combine those 2 paths into a final path.

Consequently, the mobile robot will be implemented to follow the optimal path while avoiding the dynamic obstacle function could be applied

This fusion step is employed on DWA algorithm instead of choosing another fusion algorithm. In each achieved trajectory, DWA will deal with global path by Evaluation function as shown:

$$H'(v,w) = \alpha \times hea(v,w) + \beta \times vel(v,w) + \gamma \times dis(v,w) \quad (6.9)$$

With: h$ea(v, \omega)$:the azimuth deviation between the end direction of the simulated trajectory and the current target

$vel(v, \omega)$: the set of velocity vector space

$dis(v, \omega)$: is the shortest distance from the robot to the obstacle on the corresponding trajectory of the velocity $vel(v, \omega)$.

The current target is the sequence point in the global optimal path which is the nearest to the current specific of the robot. This evaluation function makes the local path planning tracking the integral contour of the global optimal path. Hence global optimality is guaranteed effectively.

Finally, the final path will become the desired path that the mobile robot system must track. The trajectory control algorithm (LQR) computed in chapter 5 will be implemented to ensure the whole system will move truthfully.

CHAPTER 7: SIMULATION AND EXPERIMENT RESULTS

Come to the experiment chapter, due to shortage of cost, the mobile robot is able to be equipped few components. Nevertheless, in addition, the device still has enough capability to be the simple edition of the real AGV – which can handle some desired criteria and simulate the motion path planning on ROS platform.

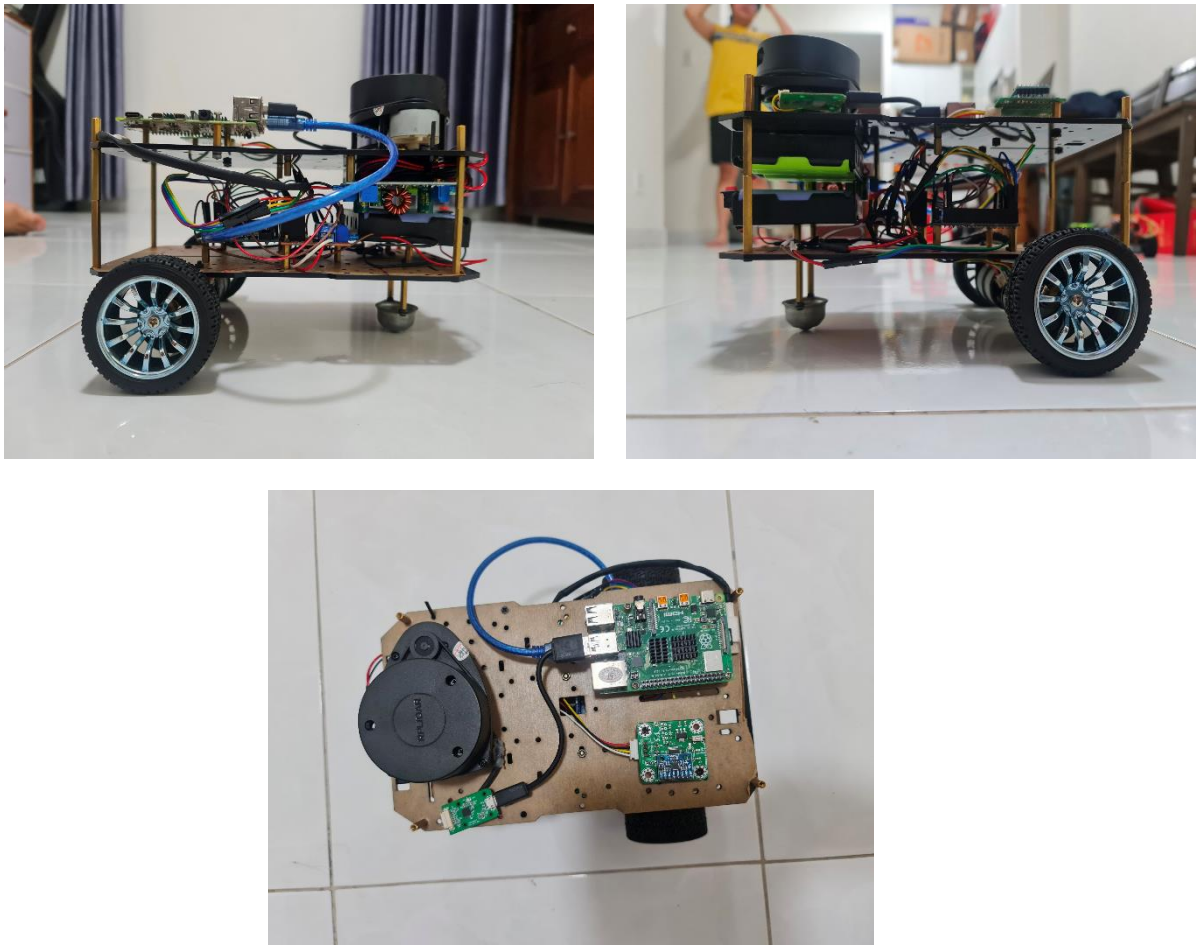


Figure 66 Experimental mobile robot

Moreover, this autonomous mobile robot will be used to handle some experiments due to its suitable functionality. In addition, the experiment will be implemented such as create map and vehicle run onto this map by several navigation guide. After that, at the end, the conclusion is going to be analyze specifically from the comparison of 2 motion planning algorithms when operating in the environment.

7.1 Components experiment

7.1.2 LIDAR

Using node concept from ROS platform to illustrate the lidar experiment. In addition, this transmitted node is implemented by TOPIC communication – Publisher and Subscriber.

Figure 67 TOPIC communication diagram

Moreover, the developed ROS node for testing LIDAR sensor is programmed by Python language:

```
#!/usr/bin/env python
import rospy
from sensor_msgs.msg import LaserScan

def scan_callback(scan_data):
    print("Angle 0 degree: " + str(scan_data.ranges[0]))
    print("Angle 30 degree: " + str(scan_data.ranges[30]))
    print("Angle 60 degree: " + str(scan_data.ranges[60]))
    print("Angle 90 degree: " + str(scan_data.ranges[90]))


if __name__ == '__main__':
    rospy.init_node('lidar', anonymous=True)
    rospy.Subscriber("scan", LaserScan, scan_callback)
    rospy.spin()

```

Figure 68 LIDAR experiment code

Finally, the result is considerably used to determine LIDARs' maximum, minimum range and its orientation mounting direction

```
Angle 0 degree: 0.5090000033378601
Angle 30 degree: 3.440000057220459
Angle 60 degree: 2.436000108718872
Angle 90 degree: 2.24399995803833
Angle 0 degree: 0.9620000123977661
Angle 30 degree: 3.447999954223633
Angle 60 degree: 2.4040000438690186
Angle 90 degree: 2.24399995803833
```

Figure 69 LIDAR experiment result

7.1.2 IMU

This sensor is attached to microcontroller via UART protocol. Its data measurement is often sliding due to the noisy. In addition, this element will transmit data to MCU and send to Raspberry via rosserial protocol and will be handled the sliding phenomenon on ROS platform. And the transmitted parameter of this sensor is degree

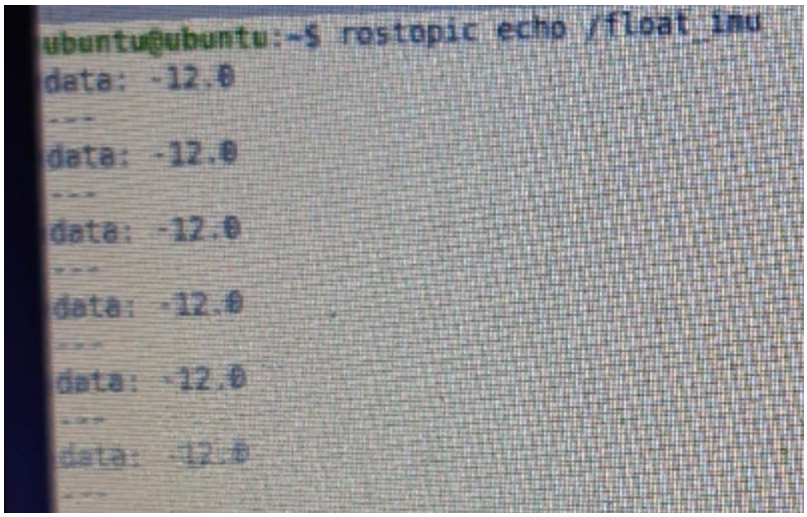


Figure 70 IMU experiment result

7.1.3 Motor

In this section, due to shortage of cost and available components, DC servo motor is selected for the experiment.

After getting the data measurement from Arduino IDE with PLX-DAQ application, the result is plotted in MATLAB as below:

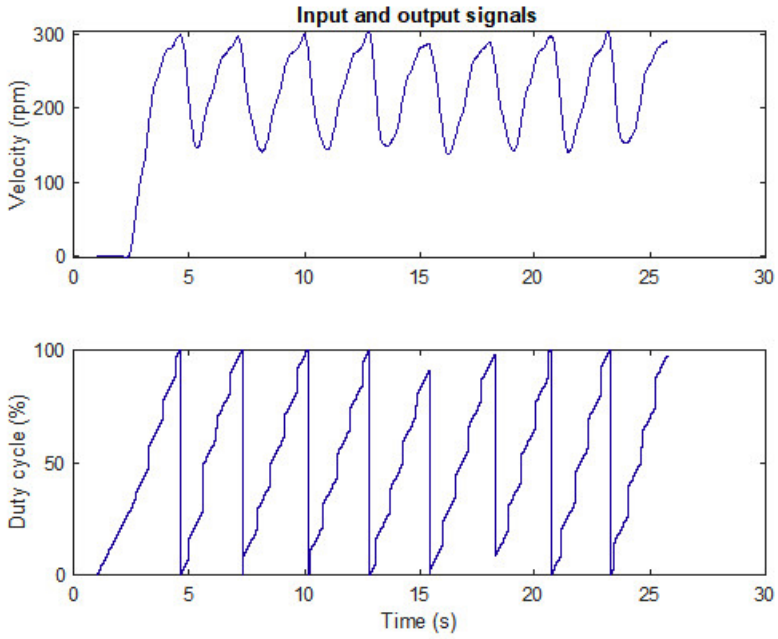


Figure 71 The response of DC Servo motor

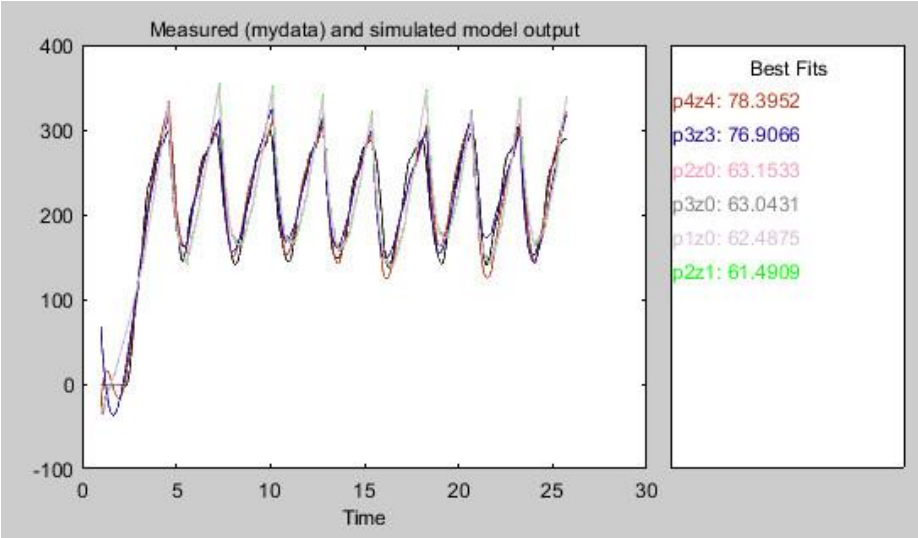


Figure 72 Estimate Transfer Function from System Identification tool

```
p4z4 =

  From input "u1" to output "y1":
  0.1041 z^-1 - 0.2998 z^-2 + 0.2877 z^-3 - 0.09197 z^-4
  -------------------------------------------------------
  1 - 3.906 z^-1 + 5.725 z^-2 - 3.732 z^-3 + 0.9129 z^-4

Name: p4z4
Sample time: 0.025 seconds
Discrete-time identified transfer function.

Parameterization:
   Number of poles: 4   Number of zeros: 4
   Number of free coefficients: 8
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "mydata".
Fit to estimation data: 72.33% (stability enforced)
FPE: 455.8, MSE: 448.5
>>
```

Figure 73 Transfer Function of motor

According to the above infomations, the Transfer Function of DC Servo motor is computed and predicted. Consequently, it illustrates that the result is just approximately 80%. Hence, the behaviour of the motor may not response as the expected outputs' parameter precisely.

From the predicted Transfer Function, applying some criteria for the PID controller, the parameter is shown as below

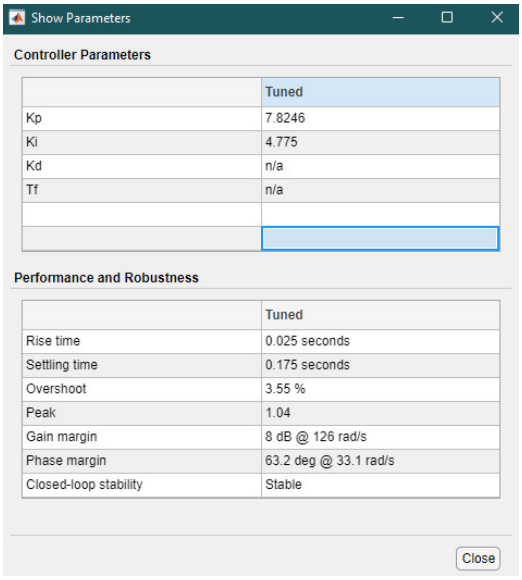


Figure 74 Parameter of PID controller

7.1.4 Serial communication

With the aim to manipulate available components optimally and expand the experience of communication protocol, rosserial package from ROS platform is

selected to support our experiment on connecting Arduino board with Raspberry through UART protocol. This concept will generate a message from Raspberry and Arduino will be responsible for computing the linear and angular velocity for 2 motors after receiving.

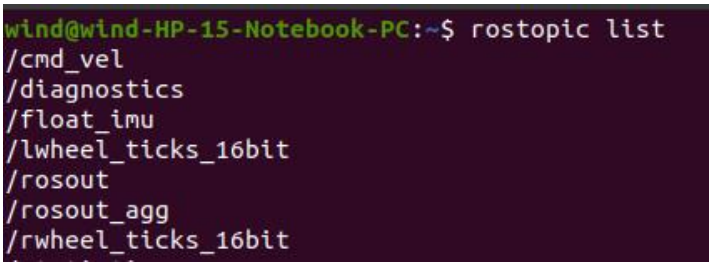


Figure 75 TOPIC infomation for communication

Moreover, according to the graphic above, it can be observed that the slave unit feedback the encoder ticks in order for the master unit to compute the odometry of the mobile robot.

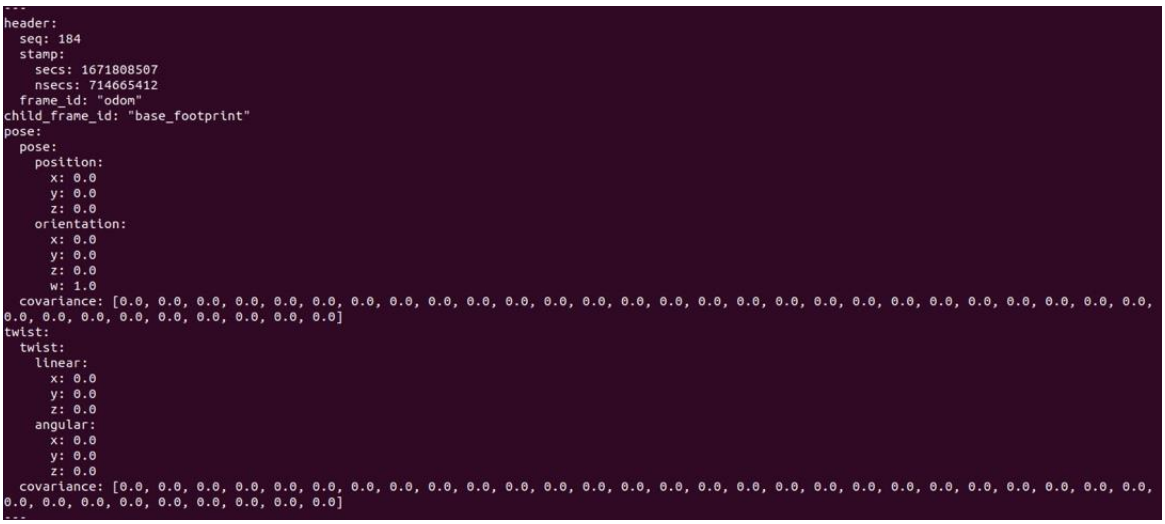


Figure 76 Interraction with Master by Odometry

7.2 Map building

The simulation manufactorys' environment model based on BMW warehouse. This map is used for path planning simulation by Python language.

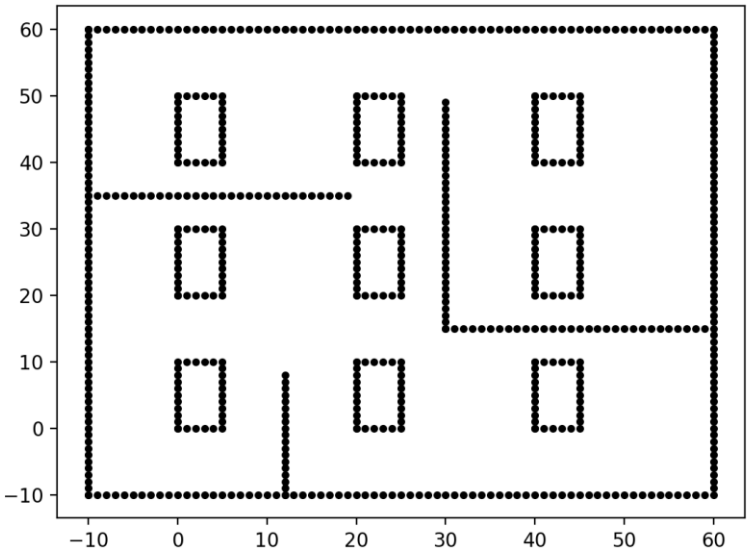


Figure 77 The simulated manufactorys' environment

After the experiment handling by Gmapping algorithm, the generated map is illustrated as below with the detail of characteristic.

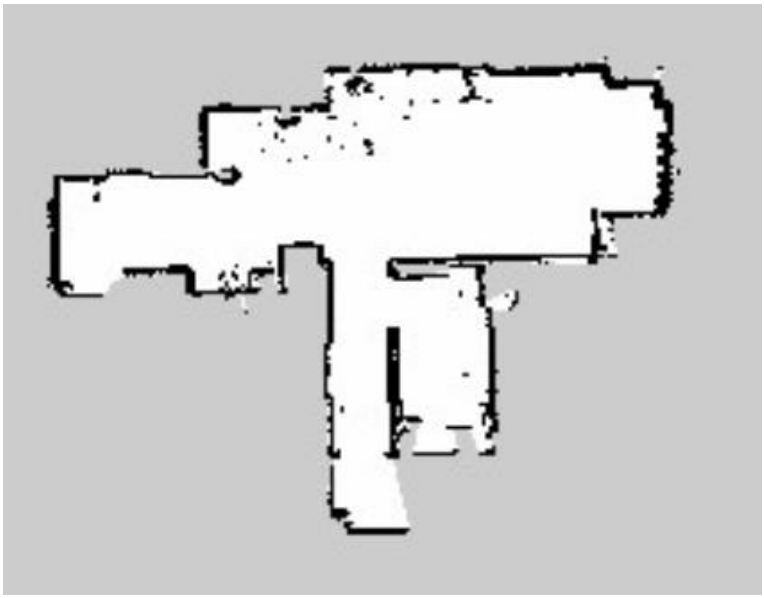


Figure 78 Generated map by Gmapping algorithm

```
image: hieu_house.pgm
resolution: 0.050000
origin: [-15.400000, -15.400000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

Figure 79 Characteristic of the map

According to the firgure above, the resolution of this map is shown as 0.05m/pixel, and the *occupied_thresh* with *free_thresh* illustrate that the pixel with occupancy probility greater/smaller than these threshold value are considered completely occupied/free

7.3 Motion path planning

In this section, using 2 maps for simulation and experiment. In addition, the simulated map is hand-generated from ROS platform via Gazebo – App for creating 3D map – and the another one is manually generated from artificial robot in Hieu's aparment.

The experiment map has been shown in Figuree 78, and the simulated map is illustrated below:

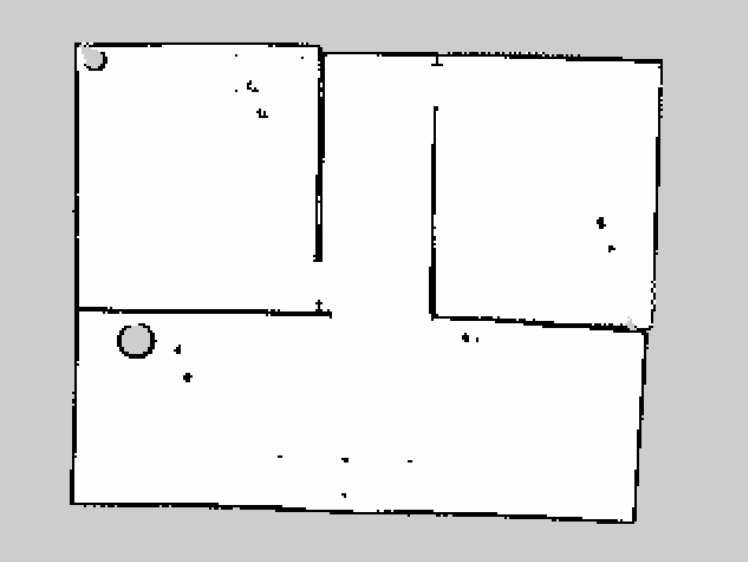


Figure 80 Simulated map

```
image: hieu_final.pgm
resolution: 0.050000
origin: [-10.000000, -10.000000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

Figure 81 Characteristic of simulated map

7.1.1 A* simulation experiment

In the preparation step, the start node is completely fixed and the desired goal is selected with 3 position simultaneously. Consequently, the algorithm will be computed and then generate the estimated path below:

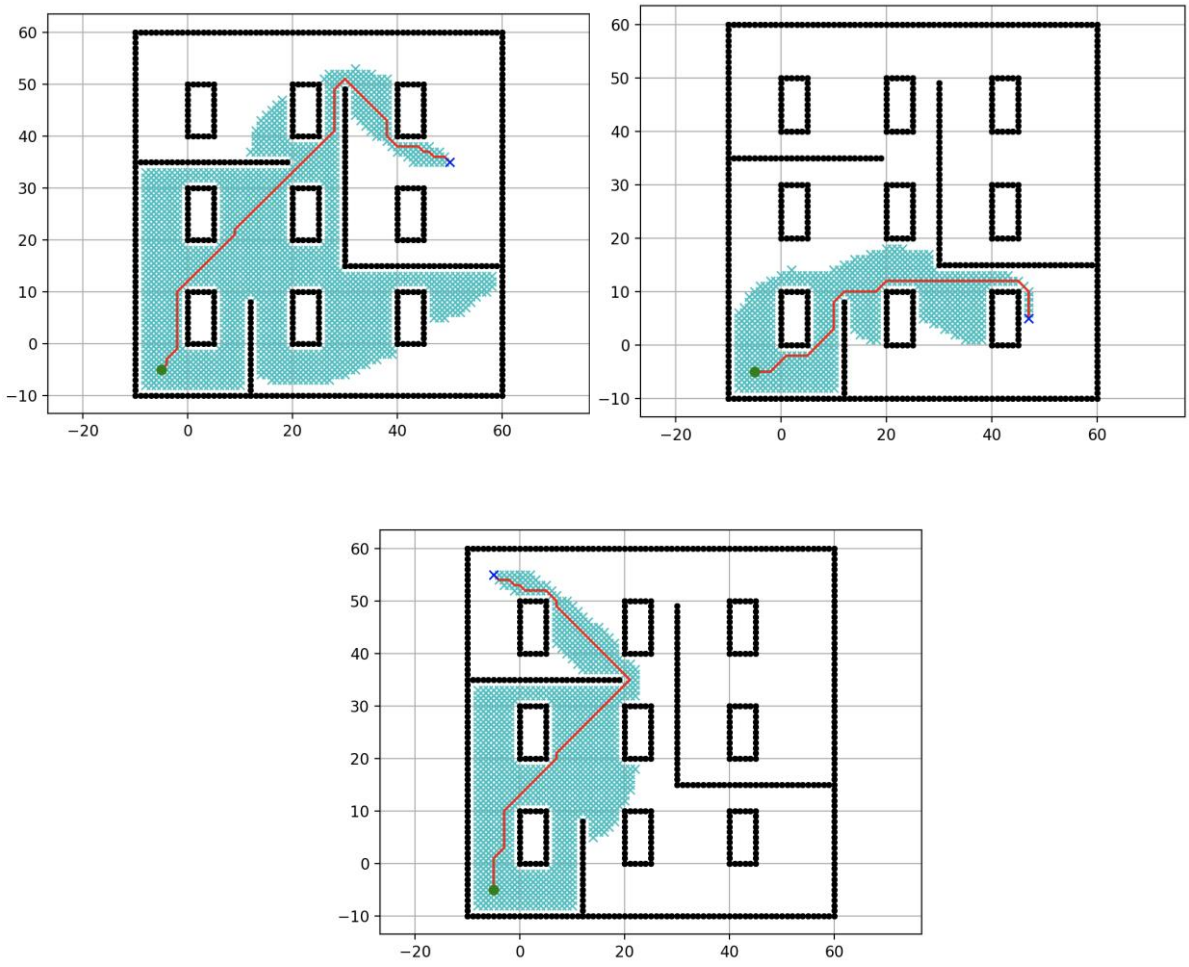


Figure 82 A simulation algorithm*

Considering all of the above figure, the execution time after estimating the algorithm is 50, 3 and 8 second simultaneously – with Grid cell = 1. Overall, the path can reach to the desired position in the approximately short path – not be always the shortest path – while the searching area is not too immense and the behavior is quite fast due to less computation – best for time criteria.

7.1.2 Dijkstra simulation experiment

In the preparation step, the start node is completely fixed and the desired goal is selected with 3 positions simultaneously. Consequently, the algorithm will be computed and then generate the estimated path below:

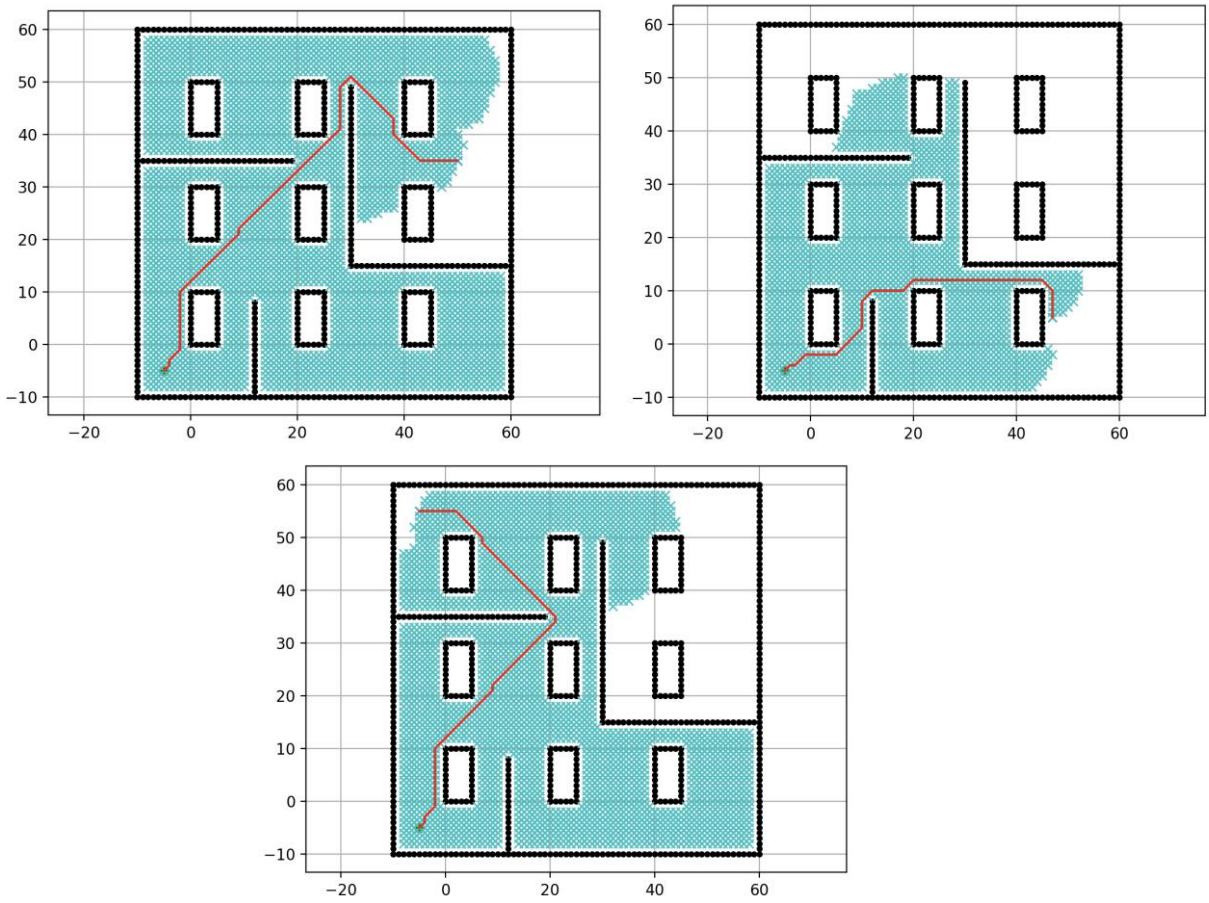


Figure 83 Dijkstra simulation algorithm

This algorithm predicts the desired goal via points following their neighbors. Then, finally, it computes the optimal path thanks to having the large data measurements.

From the excitation time, the cost is 2m20s, 16 second and 1m44s simultaneously. Overall, the generated path is optimal but the space complexity cost is quite immense and so does the time complexity when dealing with the huge area.

7.1.3 DWA simulation experiment

Experiment about local planning:

Taking account to this algorithm, the map is manually created based on the simulation of Hieu's apartment and setting the desired goals' position and expected starts' position, with simulated randomly obstacles which not occurred in the server map. This algorithm will play a vital role in handling the randomly obstacle – DWA is used in the estimated system, which predicts the collision of the robot may happen or not in the fundamental function.

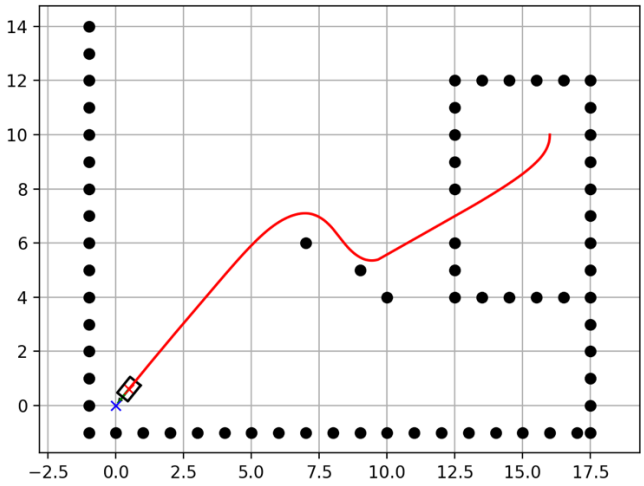


Figure 84 DWA simulation algorithm

According to the above figure, this algorithm is smoothly generated path which supports mobile robot have abilities avoiding randomly obstacle (when avoiding obstacle from the mobile robot). Nevertheless, this is not the optimal one, just activated when robot scan different obstacle in planted plan. Moreover, its behavior also depends on the yaw_rate_resolution variable.

7.1.4 Simulation path planning

In this section, the simulation will combine the global path planner and the local path planner with the aim of analyzing the behavior of the mobile robot

Fusion Dijkstra and DWA

As the above comment on partial simulation section, Dijkstra will generate the optimal path for the mobile robot with the immense of the time and space complexity's cost. Moreover, this algorithm will be simulated as the below data measurement with these prepared parameters:

Parameters	Value
Linear velocity	Max: 0.22 m/s Min: 0.01 m/s
Angular velocity	Max: 2.75 rad/s Min: 1.37 rad/s
Tolerance of goal	0.015 m and 0.17 rad/s

Table 29 Parameters of the mobile robot

When simulating, there are many unexpected factor which can impact to the result of the simulation, such as the network communication, the noise from updated frequency between the core functions and the feedback signals, and cannot simulate the dynamic obstacles significantly,… Consequently, these drawbacks can cause the delay and incorrection of the results.

Simulation link:

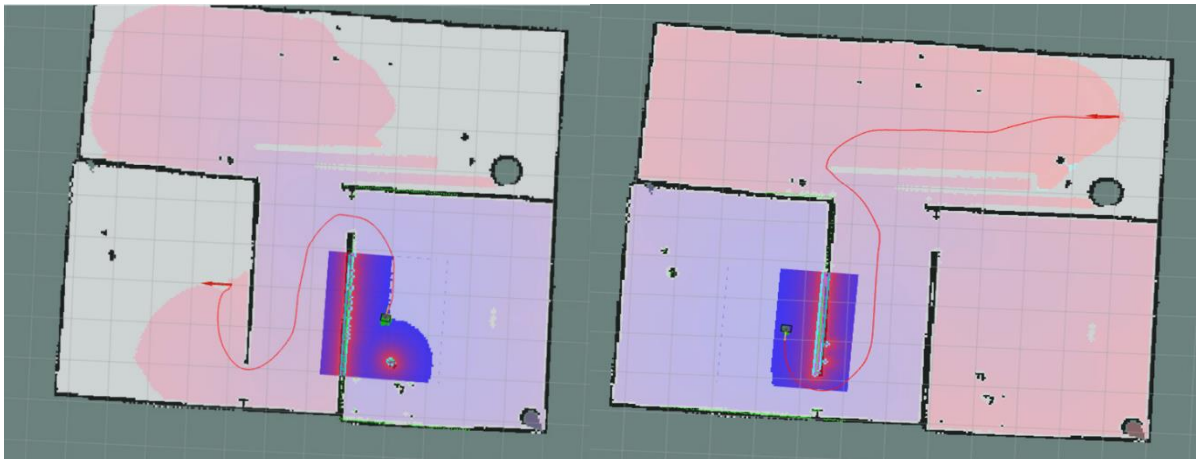


Figure 85 Simulation of Dijkstra and DWA

From the result of this simulation, the behavior of the algorithm supports the robot move from start position to goal position considerably. In addition, the excitation time of simultaneous picture is: 1m41s and 2m14s.

Fusion A* and DWA

The parameters of the robot and the preparation has been described on the above section. Comparing Dijkstra, A* will take less time complexity due to heuristic function. However, its computation may cause some curved path and the mobile robot have to steer in some case.

Simulation link:

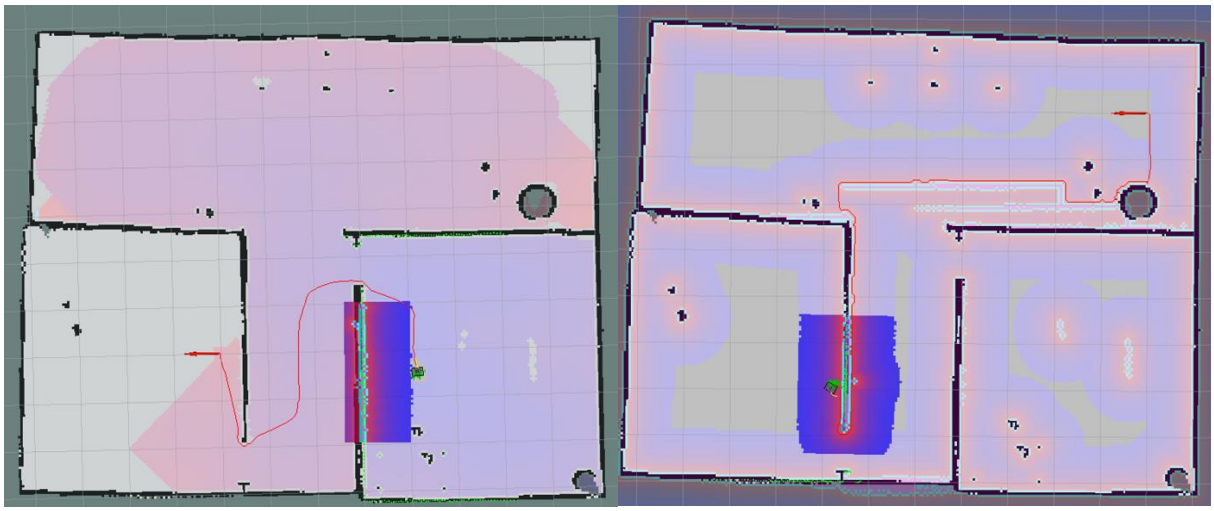


Figure 86 Simulation of A and DWA*

From the result of this simulation, the behavior of the algorithm almost likely A* in some cases. However, as shown in this simulation, in the 2nd case, the robot has to rotate many times to reach the desired goal. Moreover, that's reaction is caused due to the unoptimal path, the sliding of the wheel – created from inertia and the generated line is so close compare to the wall. In addition, the excitation time of simultaneous picture is: 3m15s and 5m2s

7.1.5 Experiment path planning

When implementing the experiment, the expected WIFI should be in acceptable condition and the dynamic or randomly obstacles for experiment has to be further than 11cm due to the characteristic of RPLIDAR device equipped on the mobile robot. However, this experiment will have large error due to the characteristic of the floor – which causes the wheels sliding and the stuck phenomenon of multi-dimension wheel.

Fusion Dijkstra and DWA

This part will operate the combination of Dijkstra and DWA algorithm on Hieus' apartment with the acceptable condition.

Moreover, the mobile robot is located in the bedroom and the final position is outside the room – which next to the bookshelf. And a obstacle is a carton box, which will be randomly located in the generated path from Dijkstra algorithm.

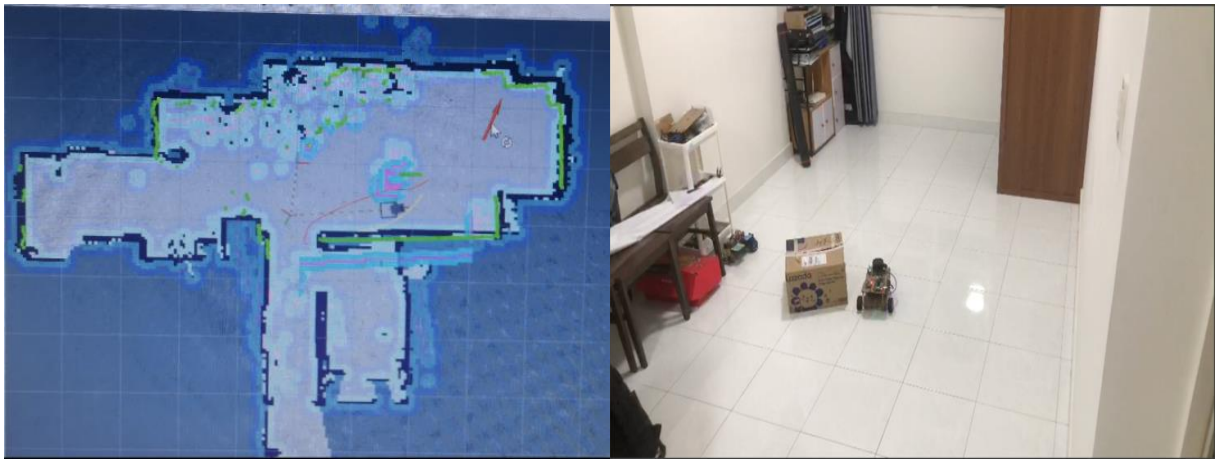


Figure 87 Dijkstra and DWA experiment

Experiment link:

Output: 1m8s

Fusion A* and DWA

This section will operate the combination of A* and DWA algorithm on Hieus' apartment with the acceptable condition.

Moreover, the mobile robot is located in the bedroom and the final position is outside the room – which next to the bookshelf. And a obstacle is a carton box, which will be randomly located in the generated path from A* algorithm.

Experiment link:

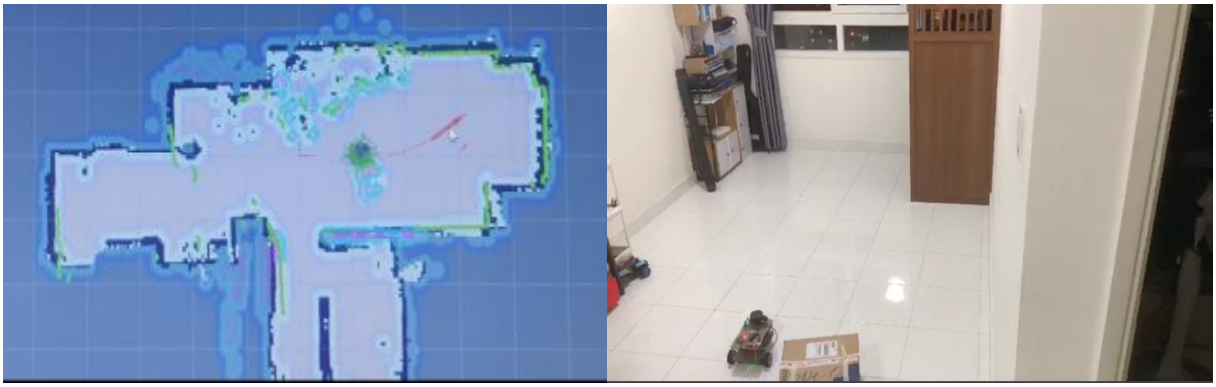


Figure 88 Experiment A with DWA*

Output:2p10s

7.3.6 Analyze the result

From the results of these experiment, the table below will be clearly illustrated once again:

CHAPTER 8: CONCLUSION

8.1 Results reached in this thesis:

This project has already cover to the mechanical, electrical, trajectory and path planning design to improve the robot performance:

- Design a model of TOWING AGVs with mentioned critical.
- Research and apply a few local planning and glocal-planing use to render the future route which can navigate AMRs.
- Applying the research in reality, build a model to examinate the theorycal for navigate a AMR and multiple AMR
- Sample ROBOT can avoid random and dynamic obstacles

8.2 Remaining imperfection

There are some remaining issues which are not able to handle, such as:

- Have already developed MCU in STM32; however, the serial communication can not adapts due to unexpected error.
- The path planning algorithm has many unexpected errors due to noise accumulation from sensors and updated frequency.
- The scale of the environment is quite small, leading to the unsatisfied result (However, only the path planning is exam in practice through the small scale protocol model)
- Error from unsuitable friction coming from the floor, noisy from imu and transform data from odometry.
- The route which was render by algorithm independent => The route of the robot will randomly, so not realistic in real world.

8.3 Future development

In order to improve advantages of robot for many restaurant conditions, there are some research need to be developed. ß

- Narrow the boundary of the robots' working space
- Build a manual communication protocol via C++ code (not depend in ROS)ß
- Apply RGB depth camera for mapping 3D environment instead of TCRT5000 sensor to detect obstacle better.

- The circuit should be custom into PCB board to reduce the noise between the connector
- The API protocol and server – client model should be use to replace ROS in order to improve the security.

APPENDIX A. AUXILIARY BOARD DESIGN

The auxiliary board is a custom PCB design board. This board is used to put all individual electrical components into 1 place to reduce the space consumption. There are 2 fundamental functions include power converter and Analog to Digital converter

A.1 Power converter

There are LM2596 adjustable power converter on the market. However, this board is used for general power purpose. This project requires only 1 power source types – 5V. According to LM2596 datasheet, the efficiency of this IC is shown:

5V converter	Output voltage		Condition (°C)	
	Min	Max	Min	Max
80%	4.8	5.2	-40	125

Table 30 LM2596 efficiency base on converter output type

Therefore, the specific output voltage version is chosen to improve efficiency. Power converter schematic is designed as below:

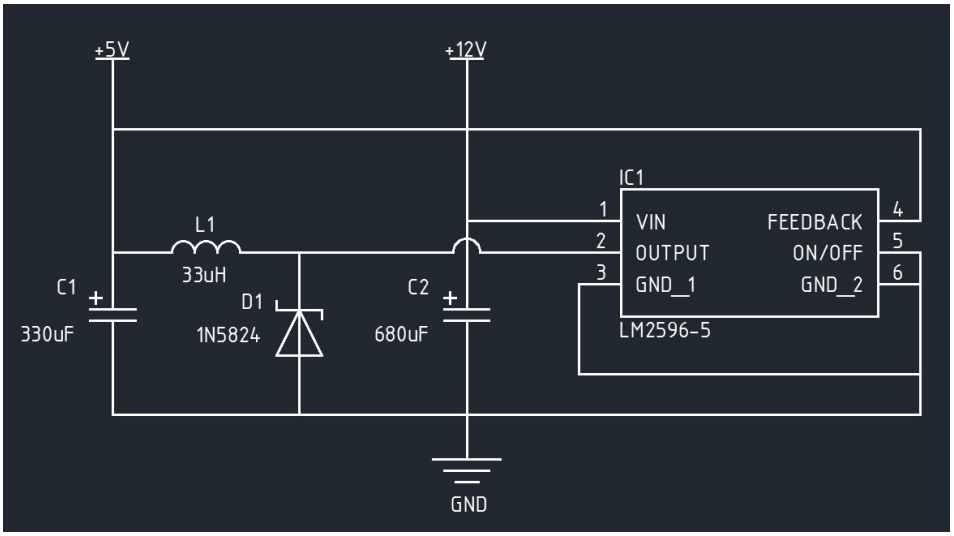


Figure 89 Buck converter

A.2 ADC

The ADC MCP3008 from Microchip manufactory is familiar with Raspberry Pi module. It has some fundamental features such as 8 – channel, 10bit, and using SPI interface – which available in Raspberry. However, this module has already popular in the market but in this project, the aim is to design a combination board in order to lessen the space consumption. And the schematic of this IC is wiring as below:

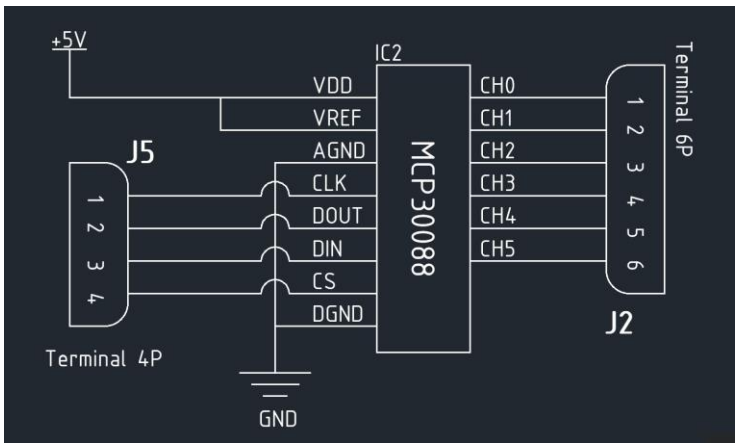


Figure 90 ADC converter

A.3 PCB design

Using ALTIUM software to design PCB board from the schematic above

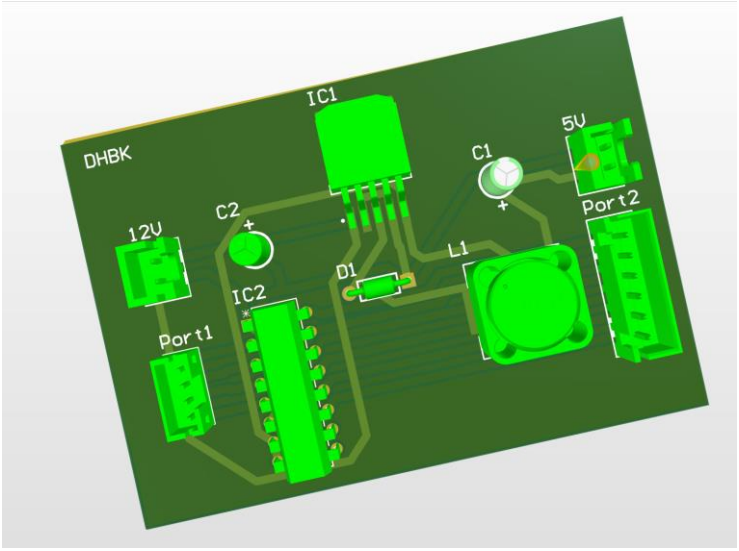


Figure 91 Auxiliary board design in 3D

REFERENCE

[1] “CREFORM STANDARD BOLT – ON AGV DRIVERS catalogue “

[2] http://www.jracking.com/news/automated-storage-and-retrieval-system-history-10640475.html

[3] UWB Time-of-arrival triangulation of ranges to determine location, available: https://jis-eurasipjournals.springeropen.com/articles/10.1186/1687-417X- 2014−3, (18.12.2017).

[4] M. Kladivová, J. Ziman, Influence of the Hall effect on domain basic solid state physics wall mobility in a cylindrical sample with circumferential easy axis, Physica Status Solidi (B) Basic Research, Vol. 246, Iss. 10, 2009, pp. 2341−2345.

[5] S. Palla, "Types of suspensions," 1 12 2021. [Online]. Available: https://www.spinny.com/blog/index.php/types-of-suspensions/.

[6] M.Sridharan, Dr.S.Balamurugan, "Design and Analysis of Lower Control ARM," International Journal of Innovative Research in Science, Engineering and Technology, 2016.

[7] M. mobbs, "Lead acid vs lithium-ion battery comparison," Sustaineble projects.

[8] T. Agarwal, "All You Know About LIDAR Systems and Applications," [Online]. Available: https://www.elprocus.com/lidar -light- detection-and-ranging-working-application/.

[9] M. M. T.-H. H. M.-S. W. Yi-Chun Du, "Stereo Vision-Based Object Recognition and Manipulation by Regions with Convolutional Neural Network," Electronics, 2020.

[10] D. Jost, "What is an Ultrasonic Sensor?," 8 10 2019. [Online]. Available: https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor.

[11] "GSAS micro system," [Online]. Available: https://gsasindia.com/newsroom/cortex-a-cortex-r-and-cortex-m/.

[12] "MPW86 MobilePower catalogue".

[13] M Arens, A Ottlik, H H Nagel. Using behavioral knowledge for situated prediction of movements[C].In Annual Conference on Artificial Intelligence.Berlin，Springer Berlin Heidelberg，2015，141-155.

[15] "The Pride of Top Quality Caster & Wheels catalogue".

[16] ElectroCraft, "CPP-A24V80A-SA-USB technical reference".

[17] SLAMTEC, "RPLIDAR A1 catalogue".

[18] Wiferion, “etaSTORE LTO catalogue”

[19] WitMotion, “WT901C AHRS IMU Sensor catalogue”

[20] Areej Bint Sajid, Aqsa Marryam, Musayyab Ali, "Modelling and Control of Brushless DC Motor," 2021.

[21] D. M. S. B. Gregor Klancar, "Mobile Robot Control on a Reference Path," Cyprus, 2005.

[22] G. O. M. V. Alessandro De Luca, "Control of Wheeled Mobile Robots: An Experimental Overview".

[23] A. J. Moshayedi, "Trajectory Tracking of Two-Wheeled Mobile Robots, Using LQR Optimal Control Method, Based On Computational Model of KHEPERA IV," 2017.

[24] J. X. Z. w. Zeya Zhu, "Global Dynamic Path Planning Based on Fusion of A* Algorithm and Dynamic Window Approach," 2019.

[25] Xu, B., & Wang, D. (2018). Magnetic Locating AGV Navigation Based on Kalman Filter and PID Control. 2018 Chinese Automation Congress (CAC).