

LEADER FOLLOWER PATTERN FORMATION
CONTROL USING ROS FOR MOBILE ROBOT
APPLICATION

AHMAD HAKIMI BIN HASNAN

B.ENG (HONS.) ELECTRICAL
ENGINEERING (ELECTRONICS)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : AHMAD HAKIMI BIN HASNAN
Date of Birth : 1ST JULY 1998
Title : LEADER FOLLOWER PATTERN FORMATION
CONTROL USING ROS FOR MOBILE ROBOT
APPLICATION
Academic Session : 2020/2021

I declare that this thesis is classified as:

- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*
 RESTRICTED (Contains restricted information as specified by the organization where research was done)*
 OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:



(Student's Signature)

980701-03-5449
New IC/Passport Number
Date: 27th JANUARY 2022



PROF. DR. HAMZAH BIN AHMAD
Name of Supervisor
Date: 27th JANUARY 2022

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

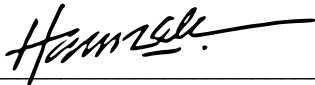
Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name	AHMAD HAKIMI BIN HASNAN
Thesis Title	LEADER FOLLOWER PATTERN FORMATION CONTROL USING ROS FOR MOBILE ROBOT APPLICATION

Reasons	(i)
	(ii)
	(iii)

Thank you.

Yours faithfully,



(Supervisor's Signature)

Date: 11.2.2022

Stamp: DR. HAJWAN BINTI AWANG
ASSOCIATE PROFESSOR
FACULTY OF ELECTRICAL & ELECTRONICS ENGINEERING
UNIVERSITI MALAYSIA PAHANG
39000 PEKAN
PAHANG DARUL MAJLIS
TEL : 09-526 0224 FAX : 09-424 6111

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

MAKLUMAT PANEL PEMERIKSA PEPERIKSAAN LISAN

(only for Faculty of Computer's student)

Thesis ini telah diperiksa dan diakui oleh

This thesis has been checked and verified by

Nama dan Alamat Pemeriksa Dalam :

Name and Address Internal Examiner

Nama dan Alamat Pemeriksa Luar :

Name and Address External Examiner

Nama dan Alamat Pemeriksa Luar :

Name and Address External Examiner

Disahkan oleh Penolong Pendaftar IPS :

Verified by Assistant Registrar IPS


Tandatangan :
Signature

Tarikh :
Date

Nama :
Name

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of the Bachelor of Electrical Engineering (Electronics) with Honours.



(Supervisor's Signature)

Full Name : PROF. DR. HAMZAH BIN AHMAD

Position : Senior Lecturer

Date : DR. HAMZAH BIN AHMAD
ASSOCIATE PROFESSOR
FACULTY OF ELECTRICAL & ELECTRONICS ENGINEERING
UNIVERSITI MALAYSIA PAHANG
36600 PEKAN
PAHANG DARUL MAKMUR
TEL : 09-424 6574 FAX : 09-424 6111

(Co-supervisor's Signature)

Full Name :

Position :

Date :



STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

A handwritten signature in black ink, appearing to read 'AHMAD HAKIMI BIN HASNAN', is written over a horizontal line.

(Student's Signature)

Full Name : AHMAD HAKIMI BIN HASNAN

ID Number : EA18160

Date : 27 JANUARY 2022

LEADER FOLLOWER PATTERN FORMATION CONTROL USING ROS FOR
MOBILE ROBOT APPLICATION

AHMAD HAKIMI BIN HASNAN

Thesis submitted in fulfillment of the requirements
for the award of the
B.Eng (Hons.) Electrical Engineering (Electronics)

College of Engineering
UNIVERSITI MALAYSIA PAHANG

FEBRUARY 2022

ACKNOWLEDGEMENTS

First and foremost, I would like to express sincere gratitude toward my project supervisor, Prof. Dr. Hamzah Bin Ahmad for giving me an amazing opportunity to do this final year project on the topic “Leader Follower Pattern Formation Control using ROS for mobile robot application”. His superb supervision really helped me to do this project and create a lot of good idea to specify the discussed idea and continuous support for making this research possible to be done. I am appreciating all the time and effort given by him to me while doing this final year project.

Other than that, my appreciation also goes to my parents who have helped me a lot until I have become an adult and final year student now. They always support me in every single time when at my best and also at my worst. I really appreciate all the advice that have been given to me to keep me stay motivated to made this task becomes much easier to be done.

Nevertheless, I also want say thank you to all my friends, housemates, classmates and those who involve directly or indirectly in this project. My final word, I hope all the knowledge and experience that I have obtain from this project can be shared back with all the community. Thank you.

ABSTRAK

Dalam projek ini, robot mudah alih dengan kawalan pembentukan ketua-pengikut sedang dibuat. Masalah pembentukan ditukar kepada masalah pengesanan trajektori dan trajektori rujukan dijana untuk setiap pengikut. Selepas itu, pengawal penjejakan daripada literatur digunakan untuk memacu robot ke arah trajektori yang dikehendaki dan kemudian pembentukan akan terbentuk kerana pengikut juga akan bergerak. Sasaran dan objektif kami dalam projek ini adalah untuk mereka bentuk pengikut pemimpin robot mudah alih dengan mengelakkan halangan sambil menganalisis kawalan pembentukan robot berdasarkan kedudukan yang dikehendaki dalam persekitaran yang berbeza. Di samping itu, kami juga akan mereka bentuk strategi kawalan pembentukan robot mudah alih dengan mengelakkan halangan. Sebenarnya, ini adalah bahagian utama untuk memastikan semua kejadian yang tidak diingini dapat dielakkan selepas digunakan kemudian. Untuk mencapai semua objektif yang dikehendaki, rangka kerja baharu dicadangkan untuk melaksanakan undang-undang kawalan pembentukan ke atas robot mudah alih bukan holonomi berdasarkan ROS (Robot Operating System). ROS akan digunakan untuk menjadi sistem utama dalam mengawal perkakasan dan juga untuk mewujudkan persekitaran maya, menjana model robot yang dipanggil Turtlebot dan melaksanakan algoritma seperti SLAM. ROS menyediakan beberapa pakej mudah dengan nod ROS dan algoritma SLAM yang menjadikan masalah pembentukan lebih mudah untuk diselesaikan. Penciptaan bahagian perkakasan menggunakan raspberry pi 3b+ dan OpenCR sebagai komponen utama manakala penciptaan persekitaran maya akan dilakukan dengan menggunakan Gazebo dan Rviz (ROS Visualization). Cadangan kerja kami dalam projek ini adalah untuk mereka bentuk burger turtlebot3 yang dilengkapi dengan sensor Lidar sebagai robot mudah alih utama untuk mempunyai pengelakan halangan dan kawalan pembentukan. Persekitaran kawalan pembentukan yang berbeza akan diwujudkan dan akan dianalisis dalam projek ini. Sebagai tambahan, Kami akan mencipta trek dan peta persekitaran dalaman untuk mensimulasikan semua ujian dalam projek ini. Kami akan menyiasat keupayaan Turtlebot untuk mengekalkan formasi dengan halaju yang diingini. Selepas itu, tugas mengelak halangan akan digunakan pada Turtlebot yang sama untuk mengelakkan sebarang keadaan yang menghalang laluan. Turtlebot ini tidak lama lagi akan menjadi peneraju untuk berbilang robot mudah alih yang akan mengikutinya sebagai pengikut dengan bantuan LDS untuk mengesan jarak antara mereka dengan ketua mereka dan menyediakan maklumat untuk trajektori rujukan.

ABSTRACT

In this project, the mobile robots with the leader-follower formation control is being created. The formation problem is converted to a trajectory tracking problem and a reference trajectory is generated for each follower. After that, a tracking controller from the literature is applied to drive the robots towards their corresponding desired trajectories and then the formation will be formed as the follower will also move. Our target and objective in this project are to design a mobile robot leader follower with obstacle avoidance while analyze the robot formation control based on the desired position in different environment. In addition, we are also to design a control strategy of the mobile robot formation with obstacle avoidance. In facts, this is the major parts to ensure that all unwanted incident can be avoid after be applied in later. To achieve all the desired objective, a new framework is proposed for implementing the formation control laws on nonholonomic mobile robots based on ROS (Robot Operating System). ROS will be used to be the main system in controlling the hardware and also to create a virtual environment, generate robot model called Turtlebot and implement the algorithms such as SLAM. ROS provides some convenient packages with ROS node and the SLAM algorithms that make the formation problem easier to solve. The creation of the hardware parts is using the raspberry pi 3b+ and the OpenCR as the main component while the creation of the virtual environment will be done by using Gazebo and Rviz (ROS Visualization). Our propose work in this project is to design for the turtlebot3 burger equipped with Lidar sensor as the main mobile robot to have the obstacle avoidance and the formation control. The environment of different formation control will be created and will be analyze in this project. As addition, We will create the track and the indoor environment map to simulate all the testing in this project. We will investigate Turtlebot ability to keep the formation with desired velocity. After that, the obstacle avoidance task will be applied to the same Turtlebot to avoid any circumstances that block its way. This Turtlebot soon will be a leader for multiple mobile robots that will follow it as follower with the help of LDS to detect the distance between them with their leader and provide the information for reference trajectory.

TABLE OF CONTENT

DECLARATION	
TITLE PAGE	
ACKNOWLEDGEMENTS	ii
ABSTRAK	iii
ABSTRACT	iv
TABLE OF CONTENT	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF SYMBOLS	xi
LIST OF ABBREVIATIONS	xii
CHAPTER 1 INTRODUCTION	1
1.1 GENERAL	1
1.2 PROJECT BACKGROUND AND CASE STUDY	2
1.3 PROBLEM STATEMENT	4
1.4 OBJECTIVE	4
1.5 SCOPE	4
1.6 PROJECT OUTLINE	5
CHAPTER 2 LITERATURE REVIEW	6
2.1 Vision based leader-follower formation control for mobile robots	6
2.2 ROS-Based Multi-Robot System Simulator	10
2.3 Behaviour-Based Formation Control for Multirobot Teams	12

2.4	The Approach	13
CHAPTER 3 METHODOLOGY		14
3.1	Introduction	14
3.2	ROS Background and Gazebo	16
3.3	Simultaneous localization and mapping (SLAM)	19
3.4	TURTLEBOT	19
3.4.1	Turtlebot3 specification	20
3.4.2	Block diagram of Turtlebot3 system	23
3.4.3	Turtlebot3 setup	23
3.4.4	Turtlebot bringup	25
3.5	CONTROL NODE	26
3.6	LAUNCHING OBSTACLE AVOIDANCE TO LEADER TURTLEBOT3	26
3.7	LAUNCHING FORMATION CONTROL FOR FOLLOWER TURTLEBOT3	28
3.8	MAP GENERATION	29
CHAPTER 4		33
RESULT AND DISCUSSION		33
4.1	INTRODUCTION	33
4.2	RESULT FROM LEADER TURTLEBOT3	33
4.2.1	Obstacle detection and avoidance data	33
4.2.2	SLAM algorithm of leader turtlebot3	36
4.2.3	Simulation of obstacle avoidance	38
4.3	RESULT FROM FOLLOWER TURTLEBOT3	39

4.3.1	Formation control of follower turtlebot3	39
4.3.2	Formation control with different environment	43
CHAPTER 5 CONCLUSION		50
5.1	PROJECT CONCLUSION	50
REFERENCES		51

LIST OF TABLES

Table 4.1 significant value based on reading untuk distance between robot dan obstacle	34
Table 4.2 Leader position value from follower	46

LIST OF FIGURES

Figure 1.1 example of trolley in industries	1
Figure 1.2 Example of Unmanned Ground Vehicle	2
Figure 1.3 example of TURTLEBOT	3
Figure 2.1 robot system overview	7
Figure 2.2 Vision Processing Flow Diagram.	8
Figure 2.3 Multiple Robot Formation	9
Figure 2.4 The architecture of MRS simulator	10
Figure 2.5 MRS simulation	11
Figure 2.6 (a) line, (b) column, (c) diamond, and (d) wedge	12
Figure 3.1 research progression Flowchart for simulation	15
Figure 3.2 Research progression flowchart on Hardware	16
Figure 3.3 Turtlebot simulation in Gazebo	18
Figure 3.4 Turtlebot in the Gazebo indoor environment	18
Figure 3.5 Component in Tutlebot	20
Figure 3.6 general specification of turtlebot3	21
Figure 3.7 general specification of tutebot3 model type Burger	22
Figure 3.8 block diagram of the main system	23
Figure 3.9 OpenCR switch location	25
Figure 3.10 visualization on the terminal of the successful bringup	26
Figure 3.11 visual of terminal of teleoperation node control	27
Figure 3.12 example of leader turtlebot3	27
Figure 3.13 example of follower turtlebot3	28
Figure 3.14 follower turtlebot3 in circle trajectory	29
Figure 3.15 follower turtlebot3 in square trajectory	29
Figure 3.16 Before and After map scanning using gmapping	30
Figure 3.17 Turtlebot is travelling around the map	30
Figure 3.18 3D generated map	31
Figure 3.19 Teleoperation command	31
Figure 4.1 Data collection of LiDar sensor with 0.7 meter range of obstacle detection	34
Figure 4.2 Data collection of LiDar sensor with 0.4 meters range of obstacle detection	35
Figure 4.3 Message on terminal with 0.4 meters range	36
Figure 4.4 initial map generation from Lidar sensor	37

Figure 4.5 Final map generation from Lidar sensor	38
Figure 4.6 simulation of leader turtlebot	39
Figure 4.7 detail explanation on data collection of follower turtlebot3	40
Figure 4.8 linear velocity of follower turtlebot3	41
Figure 4.9 angular velocity of follower turtlebot3	41
Figure 4.10 Lidar value on detecting the leader	42
Figure 4.11 Lidar value on detecting the leader	42
Figure 4.12 Square-shape path	43
Figure 4.13 linear velocity of follower turtlebot3 with square-shape environment	44
Figure 4.14 angular velocity of follower turtlebot3 with square-shape environment	44
Figure 4.15 Circle-shape track	46
Figure 4.16 the visual from the terminal on follower turtlebot3 in circle-shape environment	47
Figure 4.17 Linear velocity of follower turtlebot with circle-shape environment	48
Figure 4.18 angular velocity of follower turtlebot with circle-shape environment	48

LIST OF SYMBOLS

LIST OF ABBREVIATIONS

SLAM	Simultaneous localization and mapping
ROS	Robot Operating Software
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Control Processing Unit
MRS	Mechanical Reliability Simulation
URDF	Universal Robotic Description Format
AMCL	Adaptive Monte-Carlo Localization
LDS	Laser Distance Sensor
UGV	Unmanned Ground Vehicle

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Mobile robot systems have been chosen as it is a way to apply certain tasks that are better performed using several low-cost robots compared to single or complex ones. Mobile robot can be applied into multi-robot system may be required to travel over large distances in order to reach a site related to a mission or task. While traversing the distances, it may be desirable for the robots to move in a rigid formation with fixed inter-robot distances and being called as leader-follower. Leader-follower formation control is based on a group of mobile robots to group or move together but his gives rise to the formation control problem. As the world now are trying to move to industrial 4.0, which is required advanced technologies in industries, this project has a good potential application for the future. Site application such as automatic trolley, traveller or conveyer can be applied in the industrial soon. in fact, most of the industrial company still using manual, human-based trolley to deliver the item into the line production. the advantages such systems offer over a single robot including greater flexibility, adaptability, scalability, and affordability.



Figure 1.1 example of trolley in industries

Having a group of robots move in formation would be beneficial. It is possible for one user to control an entire group of robots without having to specify explicitly the commands for each one. However, the formation problem has been set as a major problem in designing this multi-robot systems where the objective is to make a team of mobile robot to move toward and maintain a desired geometric pattern, while maintaining a featured motion. In this thesis, we propose a new framework for all the mobile robots based on ROS so that real experiments and simulation for the formation control problem can be conducted effectively.

1.2 PROJECT BACKGROUND AND CASE STUDY

Having a group of robots move in formation have been applied in many real-world applications, such as search and rescue, demining in military missions, transporting large objects, and convoying. Leader follower technology is really needed in this generation as the world now are focusing on the advanced technologies that makes the world better. This leader follower can cut cost and the most important can safe life. For example, leader follower has been applied in the advanced technology for military mission vehicle called UGV (Unmanned Ground Vehicle). systems that has been used in this technique are able to navigate to waypoints, avoid hazards and keep formation at the same time. The formation behaviours were implemented as motor schemas as steering and speed behaviours within the Unmanned Ground Vehicle (UGV). This vehicle only used the leader follower technology to move without using any driver to reach the desired place. This is used in vehicle war as we know the risk of losing life in the war is really high.



Figure 1.2 Example of Unmanned Ground Vehicle

Now, we try to make the scope smaller and try to focusing on the industrial revolution 4.0. Real technology is needed in the line production to make it faster on delivering item from the store to the line production when needed. In ROS, we are using TUTLEBOTS as a simulation for the leader and follower robot which estimated about 3 or 4 robots will be used. Given a leader robot that moves about in a desired trajectory, an attempt was made to maintain the robots following the leader at a certain distance behind, by using only visual information about the position of the leader robot. This is the main step of investigation on leader follower formation control mobile robot before being applied on real situation.



Figure 1.3 example of TURTLEBOT

1.3 PROBLEM STATEMENT

Some of the problem that we see that in production line in industrial sector right now transferring the packaging process from one point to one point is slower and limited. Many big companies are still using manual transferring the into the assembly line and really took time as it was handled manually by person in charge. By replacing or upgrading this system by using robots, we can overcome this problem. The second one is more labour cost and more people involvement in dangerous situation. We know in 2021, we have a new normal because of the pandemic. So, people involvement in workplace especially in industrial or factory is really need to avoid. In fact, 57% covid cases in Malaysia came from the industrial clusters. we really need to reduce this risk in the future.

1.4 OBJECTIVE

Below is the following objective for this project:

1. To design a mobile robot leader follower with obstacle avoidance.
2. To analyze the robot formation control based on the desired position in different environment.

With ROS will be the main part in this project as the main system, Obstacle avoidance can be created by using the LDS data. LDS data provide the vision as it is a sensor that sending the data to Host for the simultaneous localization and mapping (SLAM)

1.5 SCOPE

With ROS will be the main part in this project as the main system, Obstacle avoidance can be created by using the LDS data. LDS data provide the vision as it is a sensor that sending the data to Host for the simultaneous localization and mapping (SLAM). Simultaneously the detecting obstacle data can also be sent to Host. When the mobile robot moves, it stops when it detects an obstacle ahead. To avoid obstacle and create formation between each robot, an algorithm needs to be develop based on priorities. For example, the leader which is robot 1 will always have the priority so he just need to focus on the trajectory and listen to the command from the user that it will

follow. It also will be equipped with the command that if it detects a possible collision, the robot will move to avoid that. Follower robot 2 will just be aware of leader robot 1, so if there is a possible collision, robot 2 will also move to avoid that.

1.6 PROJECT OUTLINE

This paper consists of 5 chapters which are chapter 1 until chapter 5. The thesis focuses on the discussion and progress of development and design that be carried out. The construction of all chapters is as follow

Firstly, chapter 1 represent the general point of the review, Project Background, Problem Statement, Objective, and Project Scope will be analysed in this chapter. In this chapter it will explain the introduction of this project.

Chapter 2 is viewing the literature reviews of this project based on some journals, research papers and various references. In this scope it will explain about the technique that has been use and the achievement by every research paper.

Chapter 3 reviews the techniques that have been used to run the project and all method used. This chapter includes two sections which are the development of hardware and software. In addition, the methods and techniques used will also be described in detail in this project, including the method and steps required.

Chapter 4 presents the results acquired after the experiment is done. The results are collected and recorded. The results are recorded by using graph and table.

Chapter 5 will talk about the conclusion for the whole project and references are also included in this chapter.

CHAPTER 2

LITERATURE REVIEW

Multi-robot formation control really has been studied extensively before by many authors. However, when considered in simulation in ROS before the application in the real world, the problem becomes much more challenging as extra understanding and works like the programming, create new environment, simulation experiment, and etc. To create something that can be controllable to the waypoint and have a good velocity to provide good time delivery with multiple robots as followers, expert design and idea is needed as references. We also need to overcome all the problem that may be faced in progress of this project. Among the problems are how the followers detects their leader on vision system, how the formation is determined and how to create the multi-Robot system simulator. In this chapter, a depth survey of the existing techniques on related topics, including formation control for the multi robot is presented. Three major references as been listed on detail to generate idea in this project.

2.1 Vision based leader-follower formation control for mobile robots

Each robot in a team is capable of localizing and following the leader robot using vision. The system structure of such a robot is shown in Figure 2.1. It consists of six units: main unit, power unit, locomotion, sensing, communication and the host computer units. The main unit, an ARM7 CPU. The power unit is a single 8V lithium-polymer battery along with 3.3V and 5V voltage regulators for powering the main control board. The drive motors run directly on the 8V. An onboard CMOS camera captures JPEG images and sends them over serially to the ARM7 controller. However, in this research review, a USB camera was used due to camera firmware issues. A Zigbee radio handles communication between the ARM7 controller and the host computer. All the image

processing is accomplished by the host computer and control commands are sent back to the robots locomotion module. (Sequeira, 2007)

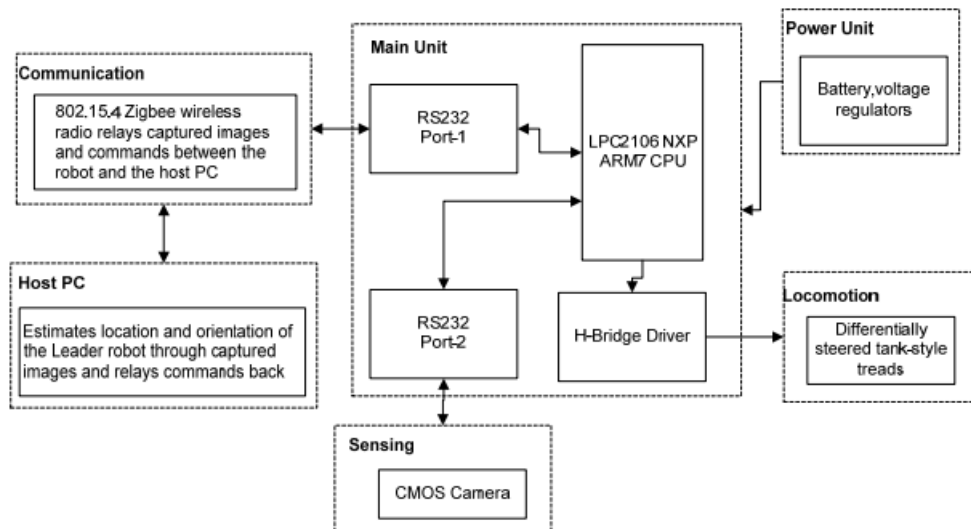


Figure 2.1 robot system overview

For the software, All code used in this work was developed using GNU open source tools. Code for the ARM7 microcontroller was written in C/C++ using an ARM port of GCC, which is a GNU compiler. OpenCV, an open source computer vision library developed by Intel was used to process images. This library provides functions for image capture and tracking, as well as processing. OpenCV is an image processing library developed by Intel™ specifically for their processors. It makes use of both the multimedia and streaming Single Instruction, Multiple Data (SIMD) extensions (MMX and Streaming SIMD Extensions) that Intel have introduced into their Pentium range, resulting in image manipulation speeds of up to 25fps. Figure 2.2 is a flow diagram illustrating how captured images are sent wirelessly to a host computer for processing. Features are then extracted and used by the visual servoing algorithm for generating velocity feedback commands. These commands are then sent back to the robots via the same radio link.

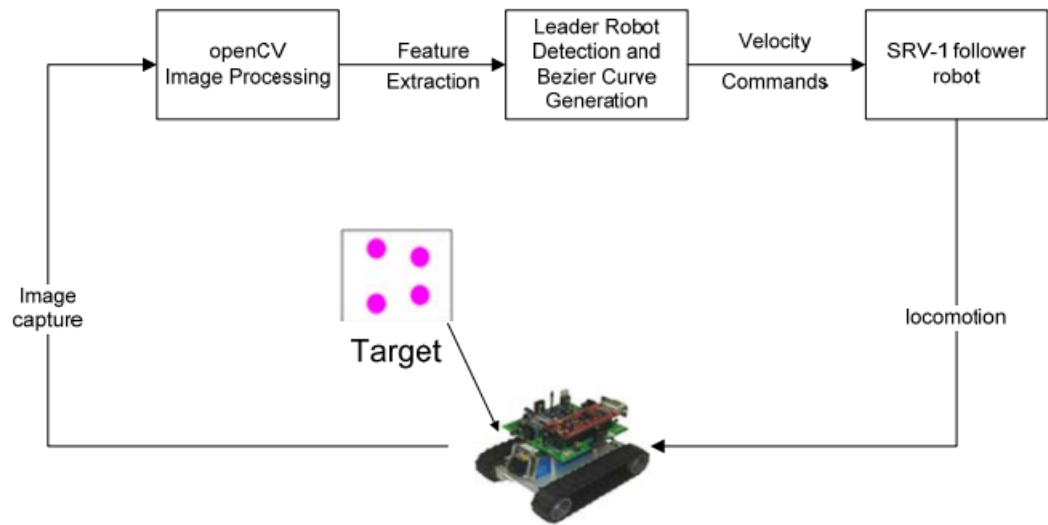


Figure 2.2 Vision Processing Flow Diagram.

In this research, the follower robot maintains a position relative to a leader robot. Using the same leader-follower philosophy, multiple robots can be made to follow the same leader with the help of virtual points. These points are the position of the leader displaced a certain distance away. The only drawback of such a formation is that every follower robot has to have the leader in view. Virtual destinations are assigned to each follower to maintain a geometric formation. These points are the position of the leader robot moved perpendicular to and a certain distance away from the leader's y-axis. Then, cubic Bézier trajectories are defined between the follower and these virtual destinations to allow the robots to follow the leader. The trajectory is updated in real-time because the virtual destination varies as the leader robot moves. Figure 2.3 illustrates two follower

robots tracking virtual points V1 and V2 displaced by certain amount in different directions from the leader.

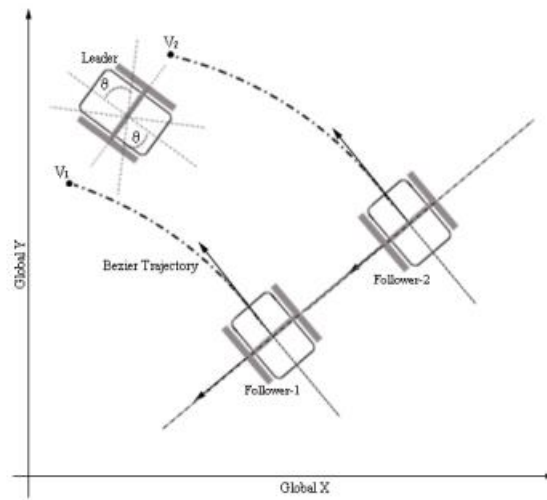


Figure 2.3 Multiple Robot Formation

2.2 ROS-Based Multi-Robot System Simulator

In this paper, an effective ROS-based MRS simulator with a simple and easy to use yet powerful GUI are developed. The GUI is developed using Qt as it has built-in support for ROS libraries and the software modules which link the GUI to the various parts of simulation are designed according to the ROS framework. The simulator is ideal for simulation of coordination behaviour of MRS as it can port directly to real robots, requiring little change to run real life testing.

In parallel with these developments, a new flexible framework for robot applications has emerged in the last few years, taking a strong position in the academic world: the Robot Operating System (ROS) environment. As an open source robot operating system, ROS provides a communication layer and a data and programming structure. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. This system is being widely adopted by the scientific community, which allows for an unprecedented direct access to state-of-the-art developments in the robotics field. (Ma, Zhu, Wang, & Zhao, 2019)

The MRS simulator is developed based on the open source ROS environment. As is shown in Figure 2.4, the simulation system consists of three layers including display, application function and data & communication. Parts of the features of the simulator are shown as follows:

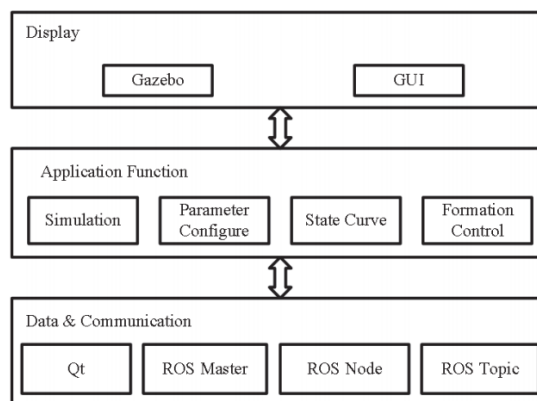


Figure 2.4 The architecture of MRS simulator

In the display layer, all of the simulation process are displayed with Gazebo and GUI. Gazebo is a powerful 3D physics simulation platform with a powerful physics

engine, some of the messages of robots are published through the open-source gazebo ros package. And most of the simulation processes are presented in GUI and user can interact with the simulation system with GUI.

The application layer contains almost all of the function of the simulator. The simulation of MRS can be executed through the shortcuts and the simulation process can be presented in the GUI as is shown in Figure 2.5. The trajectories of all the robots can be set to display or not. The number of robots in the MRS simulation can be configured directly with GUI. And the initial position of robots can be edited and the number of text edit widgets is dynamic change according to the number of robots.

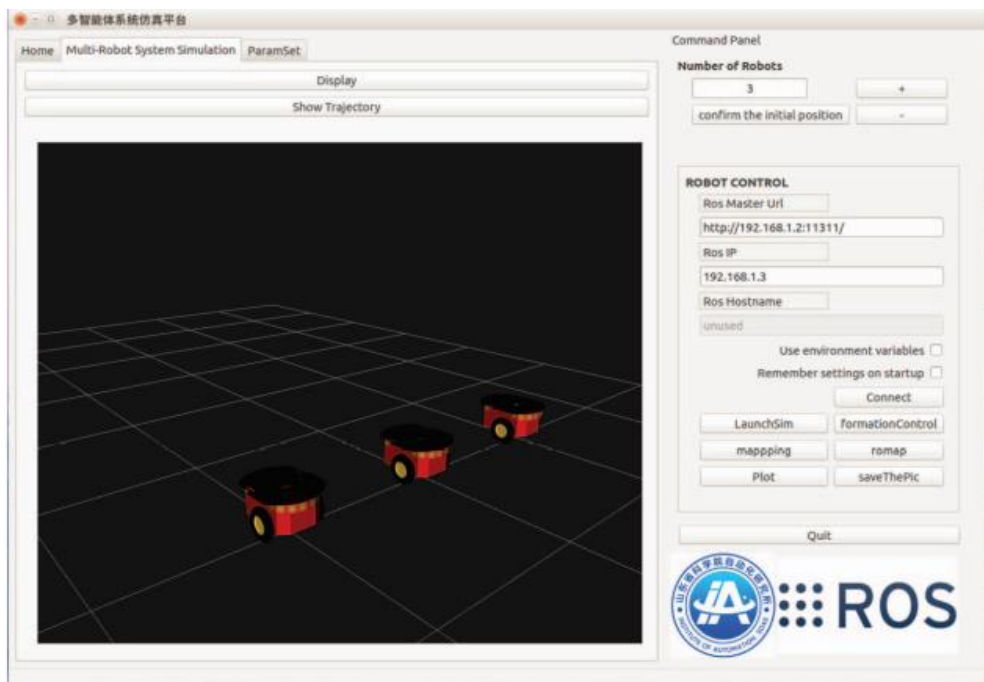


Figure 2.5 MRS simulation

2.3 Behaviour-Based Formation Control for Multirobot Teams

Since behaviour-based systems integrate several goals oriented behaviours simultaneously, systems using this technique are able to navigate to waypoints, avoid hazards and keep formation at the same time. The dynamics and stability of multi-robot formations have drawn recent attention. The sources developed a strategy for robot formations where individual robots are given specific positions to maintain relative to a leader. Sensory requirements for these robots are reduced since they only need to know about a few other robots. The analysis centered on feedback control for formation maintenance and stability of the resulting system. It did not include integrative strategies for obstacle avoidance and navigation. In work by the references article of formation generation by distributed control is demonstrated. Large groups of robots are shown to cooperatively move in various geometric formations. In the approach forwarded in this article, geometric formations are specified in a similar manner, but formation behaviors are fully integrated with obstacle avoidance and other navigation behaviors. (Balch & Arkin, 1998)

Several formations for a team of four robots are considered as shown in figure 8. line for robots travel line-abreast, column for robots travel one after the other, diamond for robots travel in a diamond and wedge robots travel in a “V.”

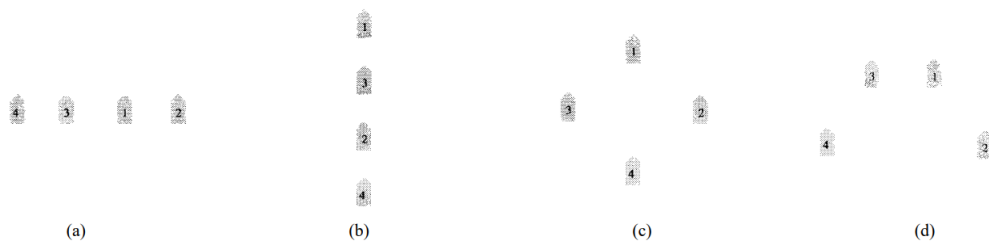


Figure 2.6 (a) line, (b) column, (c) diamond, and (d) wedge

For each formation, each robot has a specific position based on its identification number. Active behaviours for each of the four robots are identical, except in the case of Robot 1 in leader-referenced formations. The task for each robot is to simultaneously move to a goal location, avoid obstacles, avoid colliding with other robots and maintain

a formation position, typically in the context of a higher-level mission scenario. Formation maintenance is accomplished in two steps: first, a perceptual process, detect-formation-position, determines the robot's proper position in formation based on current environmental data; second, the motor process maintain formation, generates motor commands to direct the robot toward the correct location.

2.4 The Approach

After getting some literature review, we got some understanding and makes comparison between the existing project by the expert and with this project. ROS is an opensource robot operating system that provides a communication layer and a data and programming structure and can be used as the main program in this project. But instead we use MRS simulator like in second reference, we prefer to use Gazebo simulator as Gazebo will provide more easier program to create the environment. Besides, MRS simulator are focusing more on transferring data directly to real robots. In our project, we will try to create something that is incomplex and more simple program to understand for the virtual world simulation. For the pattern in the leader-follower, we will try to create one pattern which only in same column or in straight line to be more efficient after being apply in the real world instead of using many formations. This pattern formation will save the space when they are moving as we are targeting to design multi robot formation in production line.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, we are discussing about the technique and methodology that will be used for the application of multi robot in ROS and also the hardware setup. In general, Robot Operating software is being used for the main system in this simulation and hardware also Gazebo is used to provide 3D dynamic simulator. For the mobile robot, we used TurtleBot 3 as it equipped with 360 laser pointer that will be used in this project. SLAM algorithm, G mapping and control node (python) will be implemented in this project for the control purpose and creating the map environment which will be discuss in this chapter. There will be two different turtlebots with different control nodes which will be used in this project, leader turtlebot and follower turtlebot. Both will have their own application in this project later. All the experiment from implement SLAM to generate multiple mobile robots mostly will be test in the track map first to test their functions and controls before be apply to the indoor map environment which have more obstacles to test the obstacles avoidances features.

Below are the flowchart for the full technique that will be used in this project to create the simulation environment, starting from ROS and Gazebo as operating system and 3D simulation environment. Then, a single Turtlebot will be create in the virtual world as the main robot in this project. After that, SLAM implementation will be used to provide mapping in the virtual world. To provide velocity in Turlebot, the control node(Teleoperation node) will be program into the Turlebot so the Turtlebot can be control by using the keyboard command. The Turtlebot will be move and scan every corner in the virtual world or the real room and create the map using gmapping technique. Once the map generation complete, the map will be saved for further progress in this project. The main Turlebot will be program to avoid any obstacle detected by its sensor.

Once this program is complete, we will create multiple mobile robots as followers to create leader follower program. All details will be discussed in this chapter.

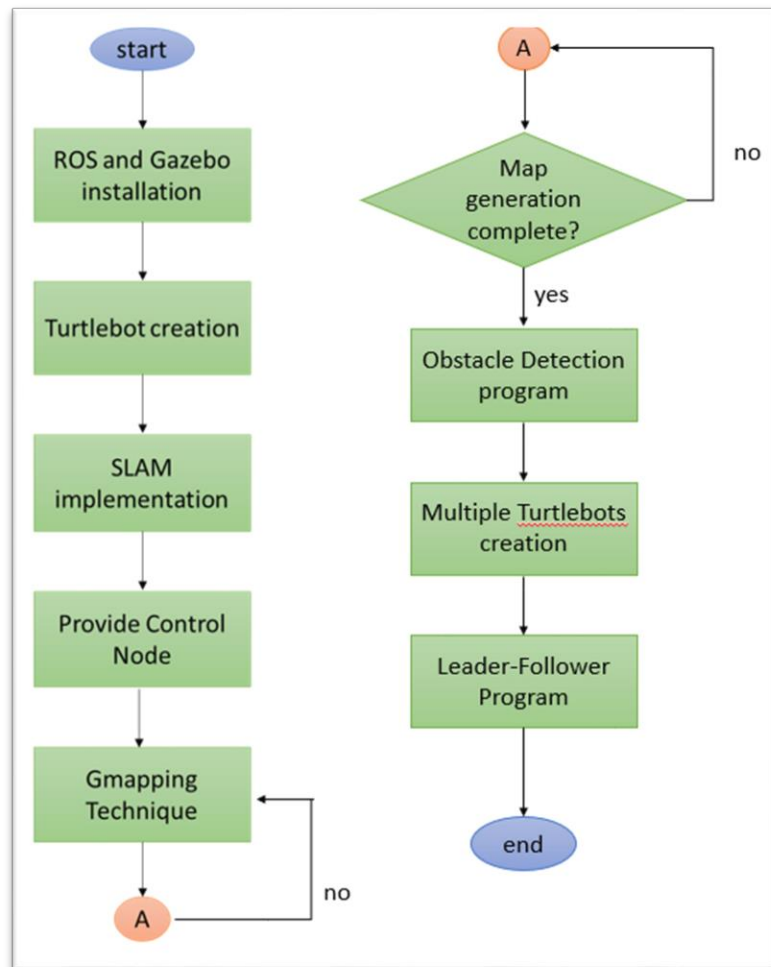


Figure 3.1 research progression Flowchart for simulation

For the hardware part is starting with setup of Personal Computer(PC) as the ROS master and setup of the raspberry pi for the ROS slave(will be explained later). The PC setup starting by installing correct Ubuntu and ROS version. Then all dependent ROS package and turtlebot package is installed. Later that, it will have a network configuration to apply in the connection of hotspot. Then we need to install OS on raspberry pi called Raspberry imager and all setup has been done. Then the OpenCR is setup by connecting the OpenCR to the Raspberry Pi using micro USB cable. Then all the required packages on the Raspberry Pi is installed to upload to the OpenCR firmware. After all is done and the firmware upload is successful, the OpenCR test will be done to make sure all is properly installed. Then, we will have the hardware assembly and bringup the turtlebot.

The Rviz will be launch to detect the turtlebot in the Rviz visualization. The obstacle avoidance node will be applied and we can see the turtlebot is successfully avoid the obstacle. We will have a data analysis and after that we will apply follower node. we can see the turtlebot will follow the leader through velocity and angle and we will have data analysis after that.

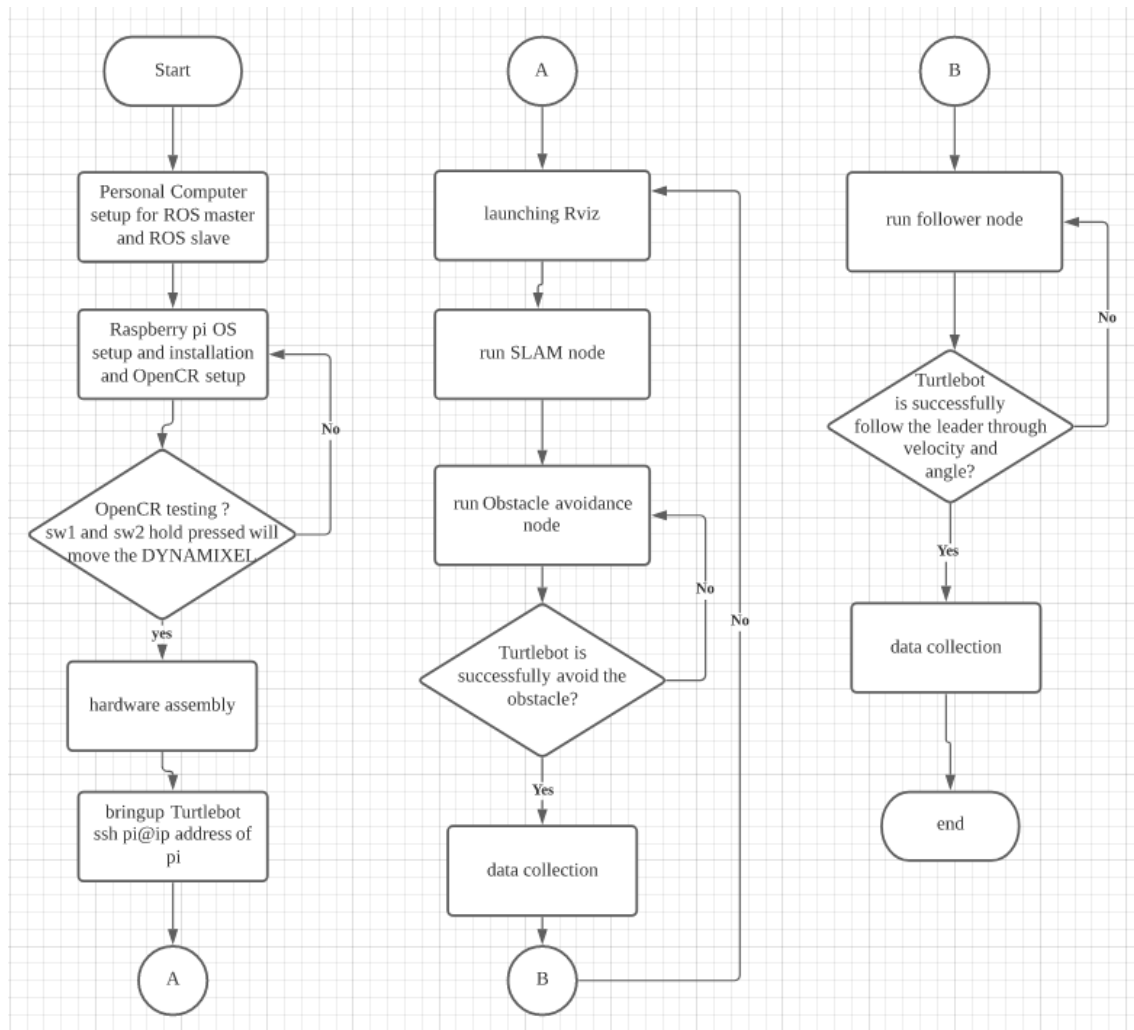


Figure 3.2 Research progression flowchart on Hardware

3.2 ROS Background and Gazebo

ROS is a Linux-based, an open-source software package that provides a software framework to aid in the development of complex robotic applications. It required an operating system called Ubuntu from Linux to run. ROS provide the services designed for a computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes,

and package management. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor data, control, state, planning, actuator, and other messages. A node is an executable program that performs computation. Nodes need to communicate with each other to complete the whole task. The communicated data are called messages. ROS provides an easy way for passing messages and establishing communication links between nodes, which are running independently.

ROS enables us to use already created packages like Gmapping and teleop key which reduces development time. In this paper ROS is used to publish messages in the form of topics between different nodes. ROS is also used to create a virtual environment, generate robot model, implement the algorithms and visualize it in the virtual world rather than implementing the whole system in the hardware itself. The creation of the virtual environment is done using Gazebo and ROS visualization.

Gazebo is a 3D dynamic simulator that will be used in this project for simulating complex mobile robots. The environment can be chosen whether in indoor and outdoor environments. It provides with the ability to test the mobile robots in the simulated environment and incorporate the data from the sensors. Gazebo has a feature that can allows to test the performance of the mobile robots in any conditions especially in extreme environment. This is a good features as mobile robot can be test before be apply into the real application. It uses an physical engine for illumination, inertia, gravity and many more. An XML file called the (URDF) Universal Robotic Description Format is used to describe different elements of the robot. The figure 3.1 shows the 3D model of turtlebot simulated in gazebo. The mobile robot model is tested in the environment shown in the Figure 3.2

In this project framework, services are essentially used in communication between robots where the control node solicits the communication node to transmit postures to another robot. The presented formation implementation relies on several operational assumptions to narrow the implementation goal to a specific scope which is about testing the formation control laws with real experiments. First, it is assumed an occupancy grid representation of the static map is available to all robots. This removes the requirement for multi-robot SLAM and map merging, which are outside the scope of this work. Second, each robot knows its initial position and possesses the required sensors

in order to maintain an accurate estimation of its pose within the two dimensional map using the navigation stack. A wireless communication network is assumed to be available over the entire coverage region.

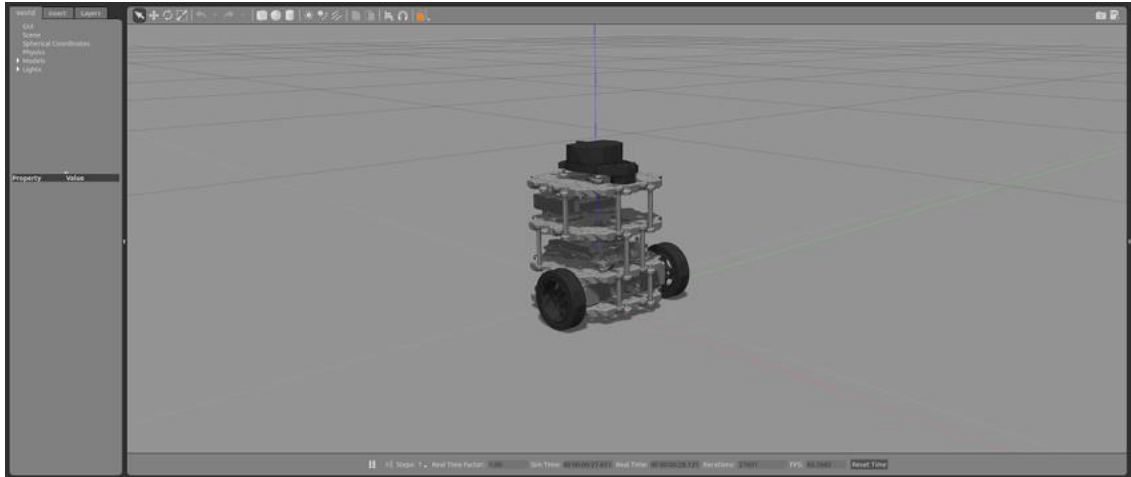


Figure 3.3 Turtlebot simulation in Gazebo

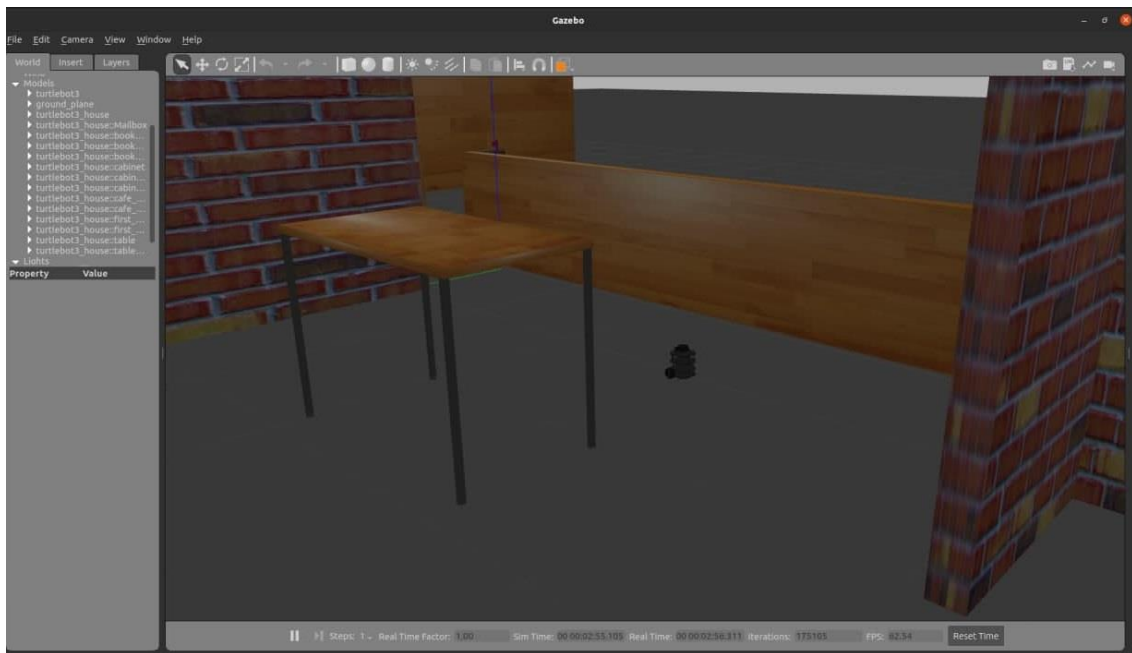


Figure 3.4 Turtlebot in the Gazebo indoor environment

3.3 Simultaneous localization and mapping (SLAM)

SLAM algorithms are tailored to the available resources, hence not aimed at perfection, but at operational compliance. The objective of the SLAM in mobile robotics is constructing and updating the map of an unexplored environment with help of the available sensors attached to the robot which is will be used for exploring. The main part of SLAM is a range measurement device which is used for observing the environment. The range measurement device depends on different variables. The robot uses landmarks to determine its location using measuring devices and sensors. When the robot has sensed a landmark it extracts the input and identifies the environment. Among example that using SLAM in real life right now is self-driving cars, unmanned aerial vehicles, autonomous underwater vehicles and newer domestic robots. Engineers often use the SLAM or map information to carry out tasks such as path planning and obstacle avoidance. we will try to implement SLAM by using MATLAB as MATLAB is a powerful software that capable to provide SLAM applications for our target system and addressing many of the countermeasures to known technical challenges. MATLAB also can deploy standalone ROS nodes and communicate with the ROS-enabled Turtlebot from MATLAB and Simulink using ROS Toolbox.

3.4 TURTLEBOT

TurtleBot is a personal robot kit with open-source software. It is a standard platform robot in ROS. It consists of two drive wheels mounted on a common axis that can independently be driven either forward or backward. The TurtleBot's core technology is SLAM, Navigation and Manipulation, making it suitable for home service robots. The TurtleBot can run in simultaneous localization and mapping or SLAM algorithms to build a map and can drive around the room. This is a perfect functionality in our project. Also, it can be controlled remotely from a laptop, joypad or Android-based smart phone. The TurtleBot can also follow a person's legs as they walk in a room. SLAM algorithm really helps TurtleBot to solve the computational problem by constructing and mapping an unknown environment while simultaneously keeping track of an agent's location within it. Basically, this will be applied in our project as the follower TurtleBot

will try to mapping the environment while keeping track of the leader which being controlled by the user.



Figure 3.5 Component in Tutlebot

The Turtlebot 3 Burger complete with Raspberry pi3 and LiDAR sensor. The Raspberry pi is a pretty stable computer, and can function without any trouble for weeks. The alternative must have similarly stable operation, support from the community and resources. LiDAR sensor in this project is to measure the distance and angle. This sensor measures the distance by measuring the time emitted the laser light hits the object which is the leader vehicle and return. This LiDAR sensor has their maximum and minimum distance which is between 120mm to 3500mm. It also can measure the angle in 360° even in the dark place.

3.4.1 Turtlebot3 specification

Below are the details for the turtlebot3 specifications. TurtleBot is the most popular open-source robot for education and research. The new generation TurtleBot3 is a small, low cost, fully programmable, ROS based mobile robot. It is intended to be used for education, research, hobby and product prototyping. The dimension of TurtleBot3 Burger is only 138mm x 178mm x 192mm (L x W x H). Its size is about 1/4 of the size of the predecessor which is Turtlebot2. TurtleBot3 also was developed to meet the cost-conscious needs of schools, laboratories and companies. TurtleBot3 is the most

affordable robot among the SLAM-able mobile robots equipped with a 360° Laser Distance Sensor LDS-01.

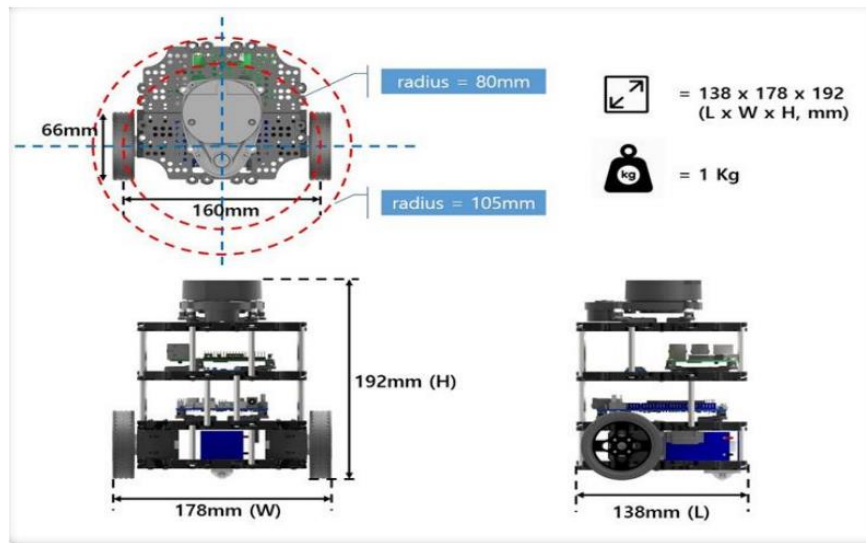


Figure 3.6 general specification of turtlebot3

Items	Burger
Maximum translational velocity	0.22 m/s
Maximum rotational velocity	2.84 rad/s (162.72 deg/s)
Maximum payload	15kg
Size (L x W x H)	138mm x 178mm x 192mm
Weight (+ SBC + Battery + Sensors)	1kg
Threshold of climbing	10 mm or lower
Expected operating time	2h 30m
Expected charging time	2h 30m
SBC (Single Board Computers)	Raspberry Pi
MCU	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
Remote Controller	-
Actuator	XL430-W250
LDS(Laser Distance Sensor)	360 Laser Distance Sensor LDS-01
Camera	-
IMU	Gyroscope 3 Axis Accelerometer 3 Axis
Power connectors	3.3V / 800mA 5V / 4A 12V / 1A
Expansion pins	GPIO 18 pins Arduino 32 pin
Peripheral	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
DYNAMIXEL ports	RS485 x 3, TTL x 3
Audio	Several programmable beep sequences
Programmable LEDs	User LED x 4
Status LEDs	Board status LED x 1 Arduino LED x 1 Power LED x 1
Buttons and Switches	Push buttons x 2, Reset button x 1, Dip switch x 2
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
PC connection	USB
Firmware upgrade	via USB / via JTAG
Power adapter (SMPS)	Input : 100-240V, AC 50/60Hz, 1.5A @max Output : 12V DC, 5A

Figure 3.7 general specification of tutlebot3 model type Burger

3.4.2 Block diagram of Turtlebot3 system

Four major parts for the turtlebot in the hardware system, which are Data Communication Unit, Data Processing Unit, Data Acquisition Unit, and Robot Actuator Unit. Data communication unit basically a PC with python nodes and the ROS system function as the master. It will supply and connect with the ROS slave in data processing unit part which is a turtlebot3 equipped with raspberry pi to. This data processing process will determine the behavior of the component in acquisition unit data. By this input, the actuator unit which is a Dynamixel XL 430-W250 actuators motors will be the output.

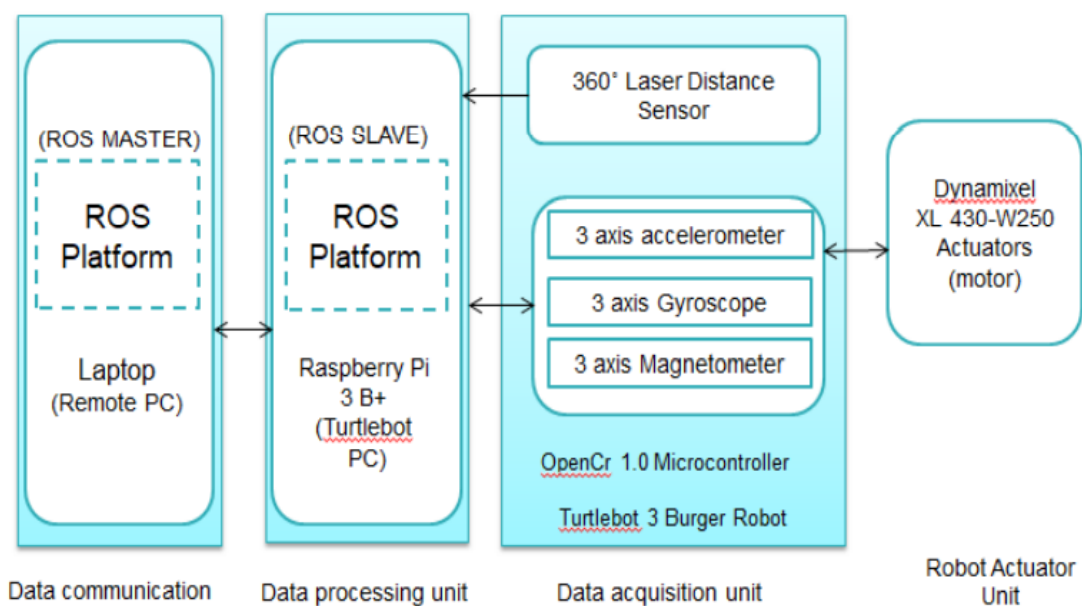


Figure 3.8 block diagram of the main system

3.4.3 Turtlebot3 setup

As the turtlebot3 is equipped with raspberry pi 3b+ and OpenCR, both need to have a proper setup to make sure the functionality of turtlebot3. The setup of Raspberry pi or being called SBC setup need to have a SD card to burn the recovery image that has been download. This file must be the correct version of image file with the hardware and ROS version. The OS called Raspbian OS downloaded will be unzip and save in the local disk. Then the image file will be burn using Raspberry pi imager. Full step by step can be refer in the https://emanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/#sbc-setup .

After the SBC setup is done, the microSD card will be inserted into the raspberry pi and the HDMI cable will be connect to the monitor and HDMI port of raspberry pi . all other input devices such as keyboard and mouse also will be connected in the USB port of the Raspberry pi. Then, the power supply will be connected to turn on the raspberry pi. It can be either USB or by OpenCR which using battery. The raspberry pi need to configure by connecting the WIFI network which is the same one with the WIFI that connect to the PC. It need to determine the IP address for raspberry pi with the command `$ ifconfig` . usually the wireless IP address for raspberry Pi can be found under the wlan0 section. From the PC, the terminal is opened and the raspberry pi is connected with its IP address. The default password is set as “turtlebot”. For the first time setup, the Raspberry pi interface need to configure by using `$ sudo raspi-config` command and network configuration for ROS by using `$ nano ~/.bashrc` command. In this section, both ROS master and ROS hostname need to configure by putting the IP address of remote PC and IP address of Raspberry pi . then, apply changes by using `$ source ~/.bashrc` command.

For the OpenCR setup, the OpenCR is connected to the raspberry Pi using the micro USB cable. The required packages is installed on the Raspberry Pi to upload the OpenCR firmware. Then the firmware also has been upload to the OpenCR. A successful setup of firmware can be tested by using the OpenCR test. In this test, we can use push sw1 and push sw2 button to see whether the robot is properly installed. This process tests the left and right DYNAMIXEL’s wheel and the OpenCR board. After assembling Turtlebot3, the power is connected to OpenCR and turn on the power switch of OpenCR. The red power LED will be turned on. Then, press and hold the sw1 for a few second to command the robot to move 30 centimeters forward. Press and hold sw2 for a few seconds to command the robot to rotate 180 degree in place. Before testing, make sure the robot is in the safe area such as in flat ground in a wide area.

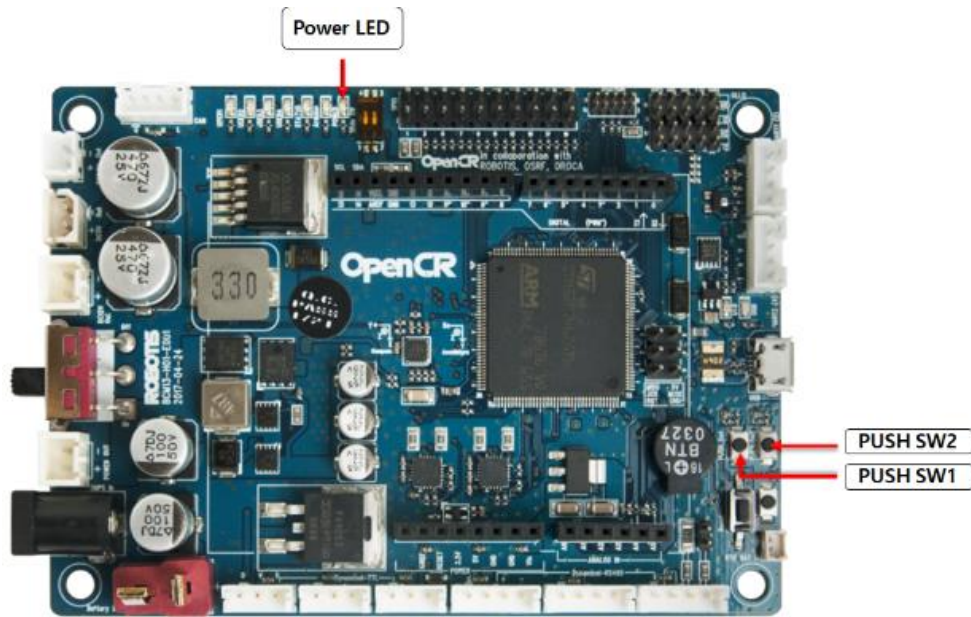


Figure 3.9 OpenCR switch location

3.4.4 Turtlebot bringup

The first step in getting bringup of the turtlebot is by running the `$roscore` command in the terminal. Checking the IP address of ROS master and ROS slave is correct one by using the `$ifconfig` command to get the hotspot IP address and use `$nano~/`. `Bashrc` command to set the PC as ROS master and turtlebot IP address as ROS hostname. By opening new terminal, connect the raspberry Pi to the PC by using command `$ ssh pi@{IP_ADDRESS_OF_RASPBERRY_PI}`. Then, bringup all the package of turtlebot to start the Turtlebot3 application by using command `$ roslaunch turtlebot3_bringup turtlebot3_robot.launch`. A successful connection will be shown as below in figure.

```

pi@raspberrypi:~$
[INFO] [1642436441.039764]: Setup publisher on firmware_version [turtlebot3_msgs/VersionInfo]
[INFO] [1642436441.044151]: Setup publisher on imu [sensor_msgs/Imu]
[INFO] [1642436441.048633]: Setup publisher on cmd_vel_rc100 [geometry_msgs/Twist]
[INFO] [1642436441.053128]: Setup publisher on odom [nav_msgs/Odometry]
[INFO] [1642436441.057521]: Setup publisher on joint_states [sensor_msgs/JointState]
[ERROR] [1642436441.061799]: Creation of publisher failed: Checksum does not match: 476f837fa6771f6e10e3bf4ef96f8770_4dda7f048e32fda22cac764685e3974
[INFO] [1642436441.066365]: Setup publisher on magnetic_field [sensor_msgs/MagneticField]
[INFO] [1642436441.070825]: Setup publisher on /tf [tf/tfMessage]
[ERROR] [1642436441.088519]: Tried to publish before configured, topic id 131
[ERROR] [1642436441.161490]: Tried to publish before configured, topic id 131
[ERROR] [1642436441.215176]: Tried to publish before configured, topic id 131
[INFO] [1642436441.346037]: Setup publisher on sensor_state [turtlebot3_msgs/SensorState]
[INFO] [1642436441.351114]: Setup publisher on firmware_version [turtlebot3_msgs/VersionInfo]
[INFO] [1642436441.356312]: Setup publisher on imu [sensor_msgs/Imu]
[INFO] [1642436441.361643]: Setup publisher on cmd_vel_rc100 [geometry_msgs/Twist]
[INFO] [1642436441.366830]: Setup publisher on odom [nav_msgs/Odometry]
[INFO] [1642436441.371874]: Setup publisher on joint_states [sensor_msgs/JointState]
[ERROR] [1642436441.376187]: Creation of publisher failed: Checksum does not match: 476f837fa6771f6e10e3bf4ef96f8770_4dda7f048e32fda22cac764685e3974
[INFO] [1642436441.381527]: Setup publisher on magnetic_field [sensor_msgs/MagneticField]
[INFO] [1642436441.386242]: Setup publisher on /tf [tf/tfMessage]
[ERROR] [1642436441.402654]: Tried to publish before configured, topic id 131
[ERROR] [1642436441.447390]: Tried to publish before configured, topic id 131
[ERROR] [1642436441.488632]: Tried to publish before configured, topic id 131
[INFO] [1642436441.638205]: Setup publisher on sensor_state [turtlebot3_msgs/SensorState]
[INFO] [1642436441.642667]: Setup publisher on firmware_version [turtlebot3_msgs/VersionInfo]
[INFO] [1642436441.648156]: Setup publisher on imu [sensor_msgs/Imu]
[INFO] [1642436441.652886]: Setup publisher on cmd_vel_rc100 [geometry_msgs/Twist]
[INFO] [1642436441.657471]: Setup publisher on odom [nav_msgs/Odometry]
[INFO] [1642436441.661871]: Setup publisher on joint_states [sensor_msgs/JointState]
[ERROR] [1642436441.665897]: Creation of publisher failed: Checksum does not mat

```

Figure 3.10 visualization on the terminal of the successful bringup

3.5 CONTROL NODE

Control node can be considered as the important parts in control system. In this project, we are using python for developing the control node code. the algorithms differ when it comes to a leader or a follower robot control. Meaning that, leader and follower robots control have different path. after the program start, the program goes into the control loop. The nodes will read to their corresponding command and provide velocity where the robot's task has been set. The robot velocities and direction also being transmit and published by the communication node to get the leader's postures and the user's order. As for leader, the main task set by is avoid the obstacle while moving in desired direction while for the follower, it will have a range detection on the front robot(leader) and maintain the distance value of range between them. The python node for this control node is as same in the Appendix A and appendix B.

3.6 LAUNCHING OBSTACLE AVOIDANCE TO LEADER TURTLEBOT3

With the setup of the bring up of the turtlebot3 and the preparation of python node for the obstacle avoidance, the turtlebot can be launch to avoid any obstacle that has been detected along the path while moving. But before launching the python node, we will test

the connectivity of the ROS master and ROS slave first by applying the teleoperation node. The teleoperation node can be done by using command `$export turtlebot3_model=burger` and `$roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch`. From here, we control the turtlebot velocity and movement by using the keyboard. The terminal will show as below.

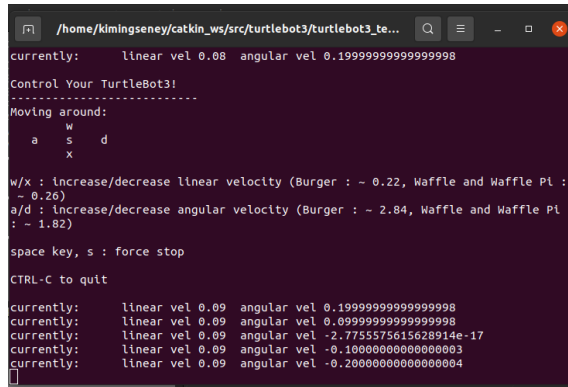


Figure 3.11 visual of terminal of teleoperation node control

After the teleoperation is successful, we can close the process by click the CTRL + C and the terminal will be neutral back. Now we will run the python node for the leader turtlebot3 as Appendix A by using command `$roslaunch [name of the package] [name of file]`. So from here, the command `$roslaunch fyp-leader-follower obstacle.py` and the turtlebot3 will be run by moving autonomously and avoid any obstacle ahead.

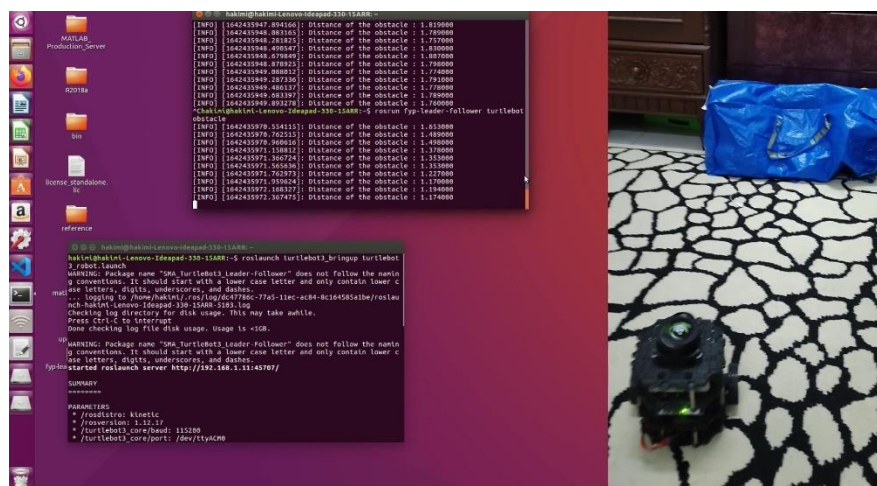


Figure 3.12 example of leader turtlebot3

3.7 LAUNCHING FORMATION CONTROL FOR FOLLOWER TURTLEBOT3

After all data for leader turtlebot3 has been collected, we will again neutral back the turtlebot3 by using CTRL + C to close all the running program on turtlebot3. Now, we will run the formation control on the turtlebot3 by using command \$roslaunch fyp-leader-follower follower.py and the turtlebot3 will run by detecting the leader and have a range detection between them.



Figure 3.13 example of follower turtlebot3

For different environment, the experiment will be illustrate in different path of shape which is square-shape and the circle-shape environment. So the leader will follow the path of shape while the follower will follow the leader. In this condition, the movement and value of Lidar sensor from the follower will be recorded for analysis purpose.



Figure 3.14 follower turtlebot3 in circle trajectory



Figure 3.15 follower turtlebot3 in square trajectory

3.8 MAP GENERATION

The map generation is carried out by creating the environment and importing turtlebot in Gazebo. The mobile robot consists of two wheels and a kinect sensor is mounted inside the frame. The mapping process in this project is implemented using the gmapping package. To use gmapping the robot model must provide odometry data and should be equipped with a horizontally-mounted, fixed, laser range-finder. The turtlebot used already have 360 laser distance sensors as a range finder. The kinect captures the depth image of the environment using Infrared sensors and acts as a range finder device. It takes 16 bits per pixel infrared data with a resolution of 640 x 480 as an color image format. This depth image is converted into a 3d point cloud using the depth image proc package in ROS. Further this 3d point cloud data is converted into 2d laserscan using

pointcloud to laserscan package. By providing all the required parameters to the packages in gmapping, a map is created.

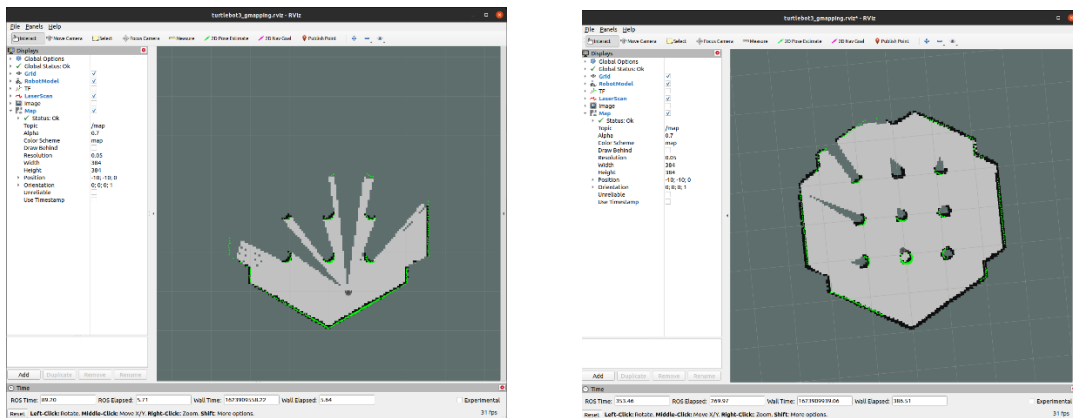


Figure 3.16 Before and After map scanning using gmapping

Figure 4.1 shows the difference between the map scanning in Rviz. On the left side, the initial vision of the map before being scan by Turtlebot. On the right side, the vision of the map data is drawn in the Rviz windows after the Turtlebot is travelling around the map and scanning all the information in the area. So, after generating the map is complete, the map will be save to the local drive for the further work.

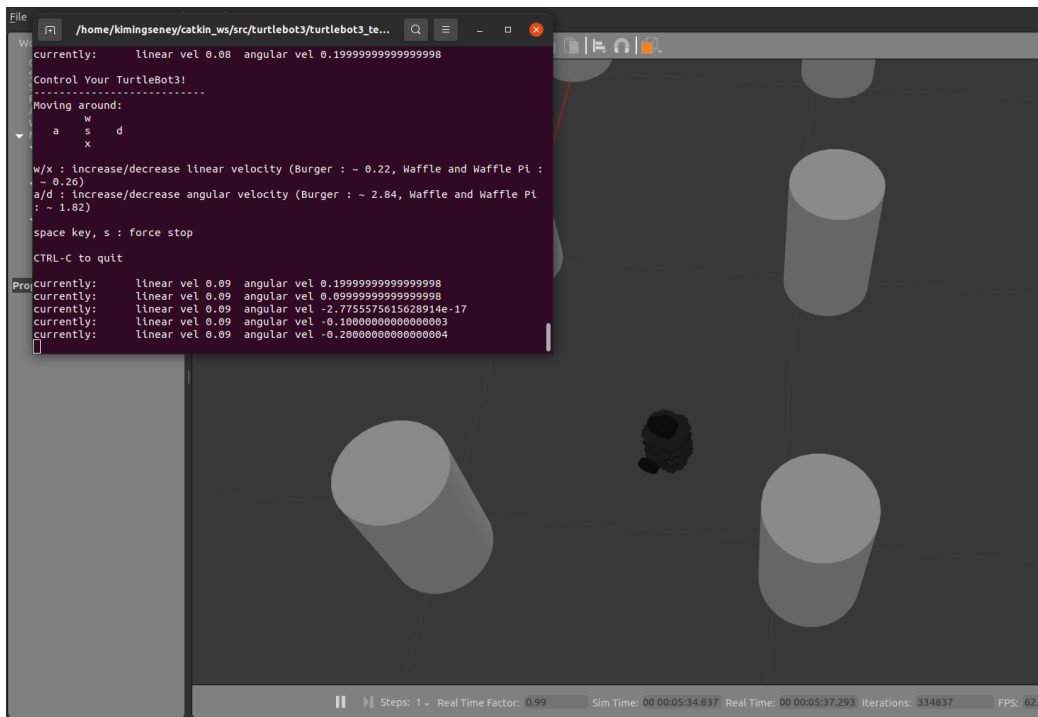


Figure 3.17 Turtlebot is travelling around the map

For the figure 3.15, the progression of gmapping technique to scan the map is shown. The terminal shows the teleoperation nodes and the keyboard command is used to provide the movement or velocity to the Turtlebot to ensure the Turtlebot can travel around the map. In this picture, we use default map from the wiki ROS as demonstration to this progress.

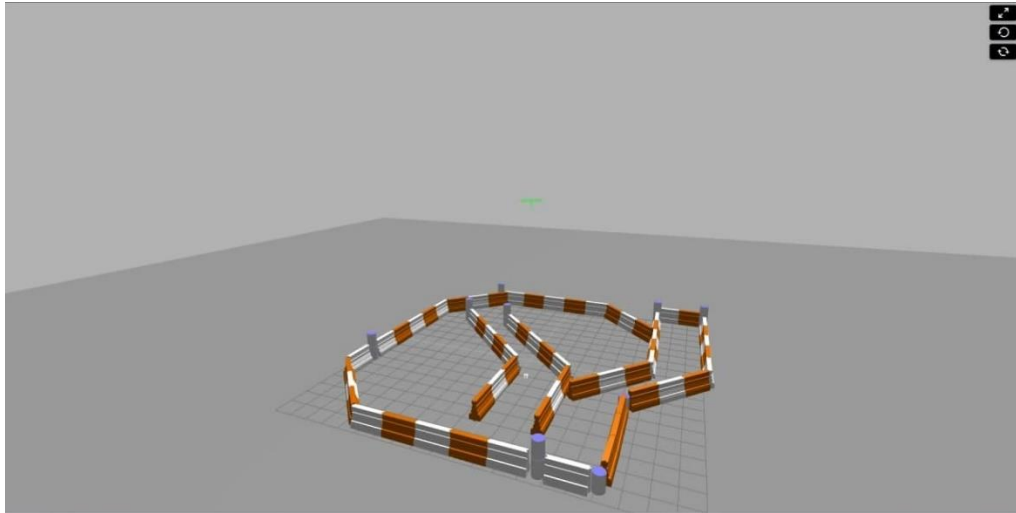


Figure 3.18 3D generated map

```
Control Your TurtleBot3!  
-----  
Moving around:  
    w  
  a   s   d  
    x  
  
w/x : increase/decrease linear velocity  
a/d : increase/decrease angular velocity  
space key, s : force stop  
  
CTRL-C to quit
```

Figure 3.19 Teleoperation command

The figure 4.3 shows the generated map. Initially the robot is moved in the simulated environment using the teleoperation key package in ROS by sending velocity commands by using the keyboard. The figure 4.4 show the example of command that will be use to provide velocity for the mobile robots. The gmapping package provides /map

topic which can be used by the map server package to save the generated map. The map is now ready to be utilized for navigation and obstacle avoidance for the multi robot.

CHAPTER 4

RESULT AND DISCUSSION

4.1 INTRODUCTION

This chapter presents the discussion about the result based on the step that has been used from the methodology. The result was taken for the value of Lidar sensor reading on turtlebot with obstacles detection and also the accuracy of formation control by the turtlebot3 with different environment.

4.2 RESULT FROM LEADER TURTLEBOT3

4.2.1 Obstacle detection and avoidance data

In the table below shows the data taken from the reading of the LiDar sensor from leader turtlebot3. The minimum range set for the obstacle detection is 0.7 meter means that it will never touch below than 0.7 meter. As if it detects the obstacle below than the value, the leader turtlebot3 will have a command to stop and changing it angular direction to anti clockwise until it no longer detects any obstacle along the path. For analysis purpose, all data has been shown in the graph created in the matlab.

value based on reading untuk distance between robot and obstacle in meter				
1.653	Stop!	1.27	0.767	0.919
1.489	Stop!	1.236	Stop!	Stop!
1.498	0.778	1.193	Stop!	Stop!
1.378	0.862	1.148	0.771	2.136
1.353	0.805	1.116	0.849	3.78
1.353	0.836	Stop!	0.825	3.74
1.227	Stop!	1.04	0.805	2.558
1.17	0.773	1.04	0.75	2.479
1.194	0.831	1.001	Stop!	2.468
1.174	Stop!	0.971	Stop!	2.434

1.11	Stop!	0.927	0.878	2.395
1.14	0.767	0.895	1.089	2.364
1.079	Stop!	0.86	1.062	2.3
1.143	Stop!	0.82	1.004	2.36
1.136	Stop!	0.798	1.054	2.323
1.081	Stop!	Stop!	1.013	2.334
1.072	1.638	Stop!	0.976	2.346
1.155	1.621	Stop!	0.99	2.354
1.094	1.598	0.767	0.911	2.379
1.102	1.577	0.79	0.866	2.355
1.067	1.555	0.761	0.773	2.358
0.944	1.505	Stop!	Stop!	2.348
0.954	1.498	Stop!	Stop!	2.345
0.952	1.463	Stop!	1.011	2.357
0.976	1.441	0.832	1.3	2.326
0.831	1.413	0.91	1.169	2.332
0.815	1.353	0.867	0.94	2.359
0.798	1.324	0.802	0.974	
Stop!	1.302	0.783	0.832	

Table 4.1 significant value based on reading untuk distance between robot dan obstacle

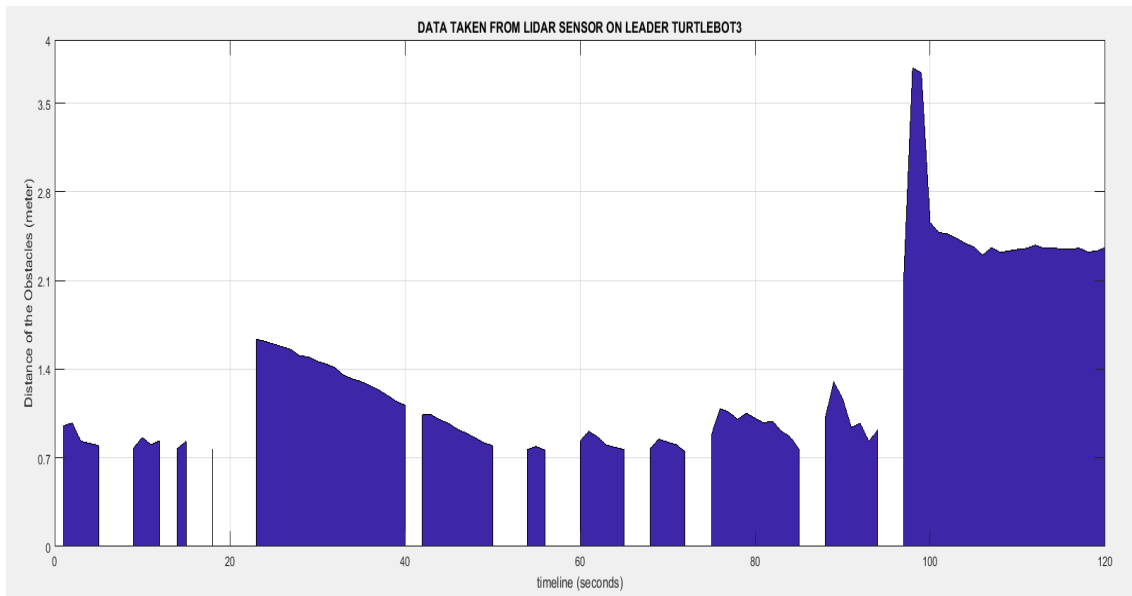


Figure 4.1 Data collection of LiDar sensor with 0.7 meter range of obstacle detection

The area under graph has been used for clear vision purpose. All the movement of the Turtlebot3 is recorded in the graph when the value of y-axis is visualized which is the Lidar sensor is functioning to detect the distance of the obstacle ahead. From the graph above, we can see there are many parts of blank spaces (without any velocity) is occur. This is because the Lidar sensor is detecting the range of less than 0.7 meter and the turtlebot3 currently stop and move it angular velocity to find another path. In this situation, Lidar sensor is currently on hold until the turtlebot3 move again. So, all the blank spaces in the graph is actually refer to the turtlebot3 is stop and changing its direction and that is why after the blank spaces, there are more reading has been recorded as the velocity is provided in another path.

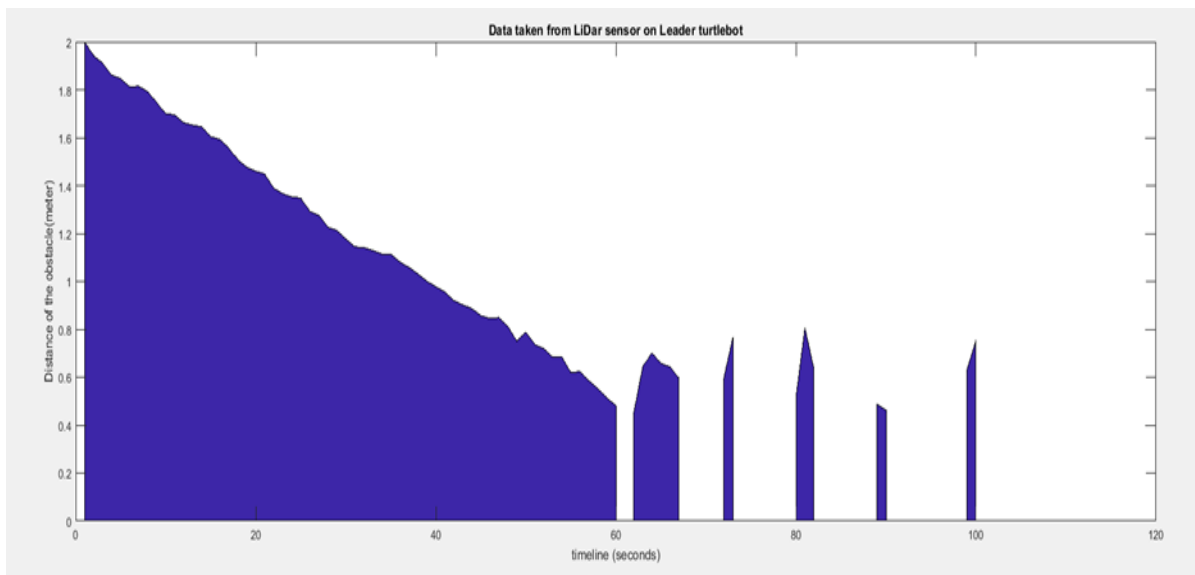


Figure 4.2 Data collection of LiDar sensor with 0.4 meters range of obstacle detection

In this figure above is the visualization of collection of data from LiDar sensor on detection of obstacle when the turtlebot3 is moving but the difference between this data and the previous collection data as in figure 4.1 is the range of obstacle detection on this situation is set by only 0.4 meter. By this value, there are some error on the progress on providing the velocity to the turtlebot3 as the Dynamixel wheel is stuck on the obstacles and turtlebot3 have a failure in getting the angular velocity to changing its direction. That is why in this figure, the value of obstacle detection is getting disappear after 80 seconds of the progress as it cannot continue to get the angular velocity.

```
hakimi@hakimi-Lenovo-Ideapad-330-15ARR: ~
[INFO] [1643037277.171982]: Distance of the obstacle : 0.701000
[INFO] [1643037277.371290]: Distance of the obstacle : 0.658000
[INFO] [1643037277.569068]: Distance of the obstacle : 0.643000
[INFO] [1643037277.778263]: Distance of the obstacle : 0.597000
[INFO] [1643037277.976578]: Stop!
[INFO] [1643037278.175209]: Stop!
[INFO] [1643037278.374101]: Stop!
[INFO] [1643037278.573951]: Stop!
[INFO] [1643037278.773868]: Distance of the obstacle : 0.597000
[INFO] [1643037278.982274]: Distance of the obstacle : 0.766000
[INFO] [1643037279.180561]: Stop!
[INFO] [1643037279.379123]: Stop!
[INFO] [1643037279.578212]: Stop!
[INFO] [1643037279.776604]: Stop!
[INFO] [1643037279.985436]: Stop!
[INFO] [1643037280.183803]: Stop!
[INFO] [1643037280.382077]: Distance of the obstacle : 0.513000
[INFO] [1643037280.581457]: Distance of the obstacle : 0.805000
[INFO] [1643037280.789833]: Distance of the obstacle : 0.634000
[INFO] [1643037280.989716]: Stop!
[INFO] [1643037281.189000]: Stop!
[INFO] [1643037281.398007]: Stop!
[INFO] [1643037281.596460]: Stop!
[INFO] [1643037281.794471]: Stop!
[INFO] [1643037281.992827]: Stop!
[INFO] [1643037282.191855]: Distance of the obstacle : 0.488000
[INFO] [1643037282.390714]: Distance of the obstacle : 0.463000
[INFO] [1643037282.590585]: Stop!
[INFO] [1643037282.798904]: Stop!
[INFO] [1643037282.998180]: Stop!
[INFO] [1643037283.226886]: Stop!
[INFO] [1643037283.394930]: Stop!
[INFO] [1643037283.593896]: Stop!
```

Figure 4.3 Message on terminal with 0.4 meters range

By this situation, this is the visual from the terminal as the message is always have “stop!” message as it cannot get the proper angular velocity to avoid the obstacle and changing its direction. So the LiDar is always detect the same obstacle ahead. That’s why after some experiment, the value of 0.7 meter on range of detection of the obstacle is determined.

4.2.2 SLAM algorithm of leader turtlebot3

While the turtlebot3 is moving around while avoiding any obstacle, the SLAM algorithm also has been implemented by opening the new terminal and using command `$ roslaunch turtlebot3_slam turtlebot3_slam.launch` . by this command, it will open the Rviz software to show the current location of the turtlebot3 while moving around the room. In this experiment, my private room is chosen as it is in medium size place and can provide a good obstacle to the environment for the turtlebot3. This technique is very important to ensure the functionality of the LiDar sensor during the experiment and also generate the environment.

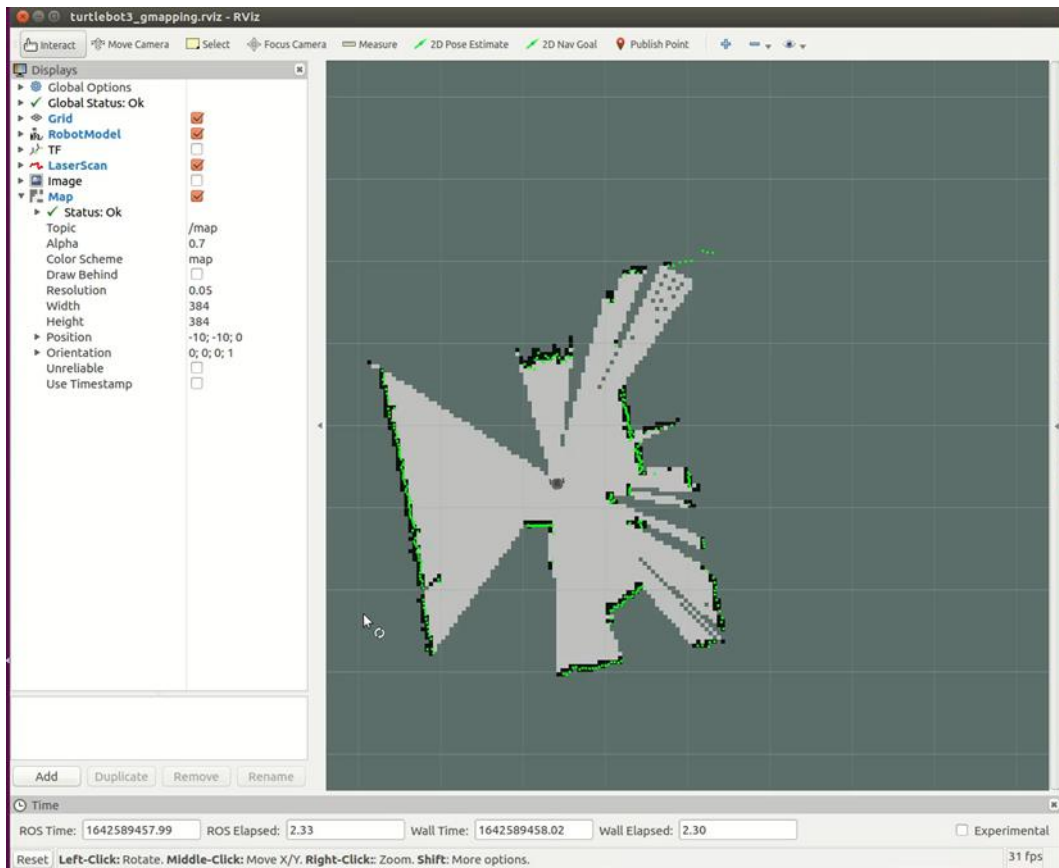


Figure 4.4 initial map generation from Lidar sensor

In the figure above shows the initial map of the room. This is the visual of the room before the robot is running and scanning using the LiDar sensor. From here we can see that only a small map is generated as the Lidar is only gaining the information in the static places. As the turtlebot3 providing its velocity, the Lidar will gain more and more information of the map to be generated.

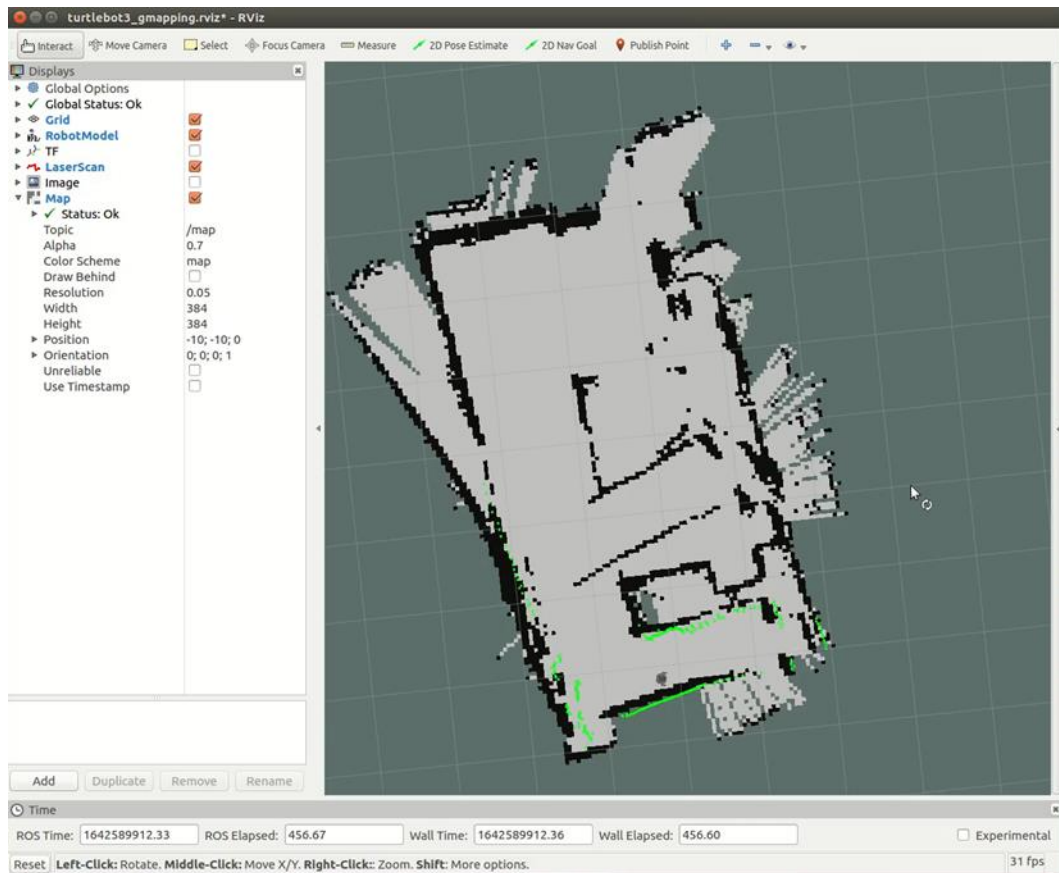


Figure 4.5 Final map generation from Lidar sensor

After 20 minutes the leader turtlebot3 is providing velocity and avoid any obstacle around, we can see the map generation from the leader turtlebot3 is almost done. This is a good result from the turtlebot3 leader as it is proven that the turtlebot3 can travel around the room without any problem. This is also a proven data that the Lidar sensor is fully functioning during the turtlebot3 leader is moving around the room.

4.2.3 Simulation of obstacle avoidance

The simulation of the leader turtlebot equipped with obstacle avoidance also is created. In this simulation running in the Gazebo, the condition is same with the hardware as the turtlebot3 is moving around while avoiding any obstacle around. In this video we can see the turtlebot3 is moving while the terminal is showing the distance of obstacle detected by the Lidar and after it meet the barrier, the turtlebot3 will stop and changing its directions. The full simulation can be watch in this youtube video link, <https://youtu.be/oR53ZEHO7ho>

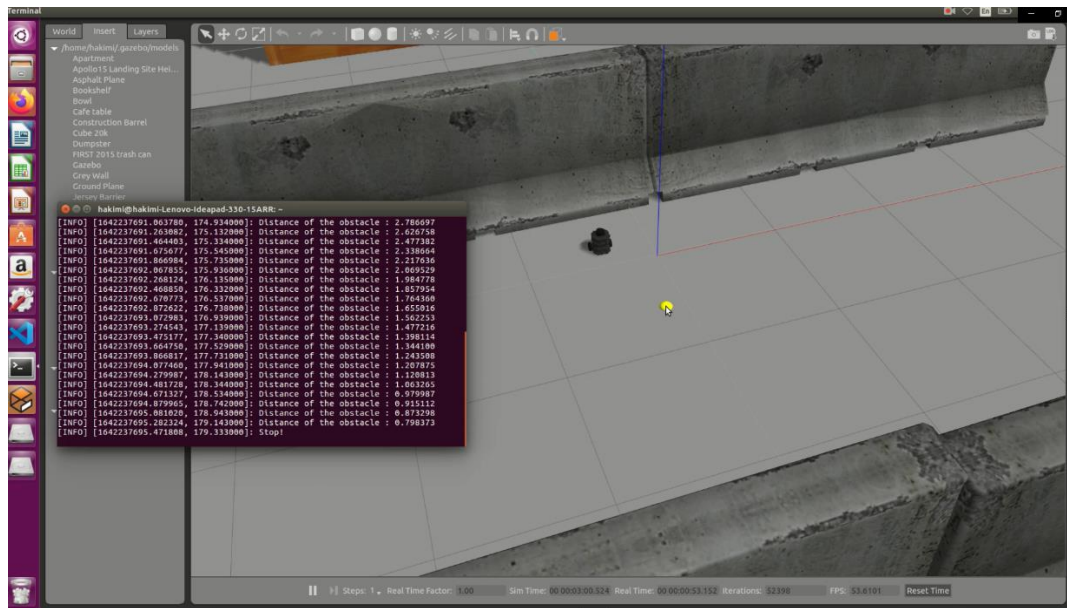


Figure 4.6 simulation of leader turtlebot

4.3 RESULT FROM FOLLOWER TURTLEBOT3

4.3.1 Formation control of follower turtlebot3

For the follower turtlebot3, we run the follower to follow the leader based on the trajectory that has been plan. In this experiment, the trajectory of the follower will be measured and recorded to see if it success to follow the leader or not. The leader will move straight, to the right , to the left while the follower is in behind. In appendix D shows the data taken from the follower turtlebot. From the data, we can see the value of different data is collected. This is because of the nodes that perform on the follower turtlebot3. The data collection such as '30_l' , '30_r' , '45_l' and '45_r' have their own value on translational velocity and angular velocity. We can see the detail in the figure below.

```

def follow(self):
    while not rospy.is_shutdown():
        check = self.check_people()
        if check == 1:
            x = self.laser_scan()
            twist = Twist()
            ## Do something according to each position##
            if x == ['30_0']:
                twist.linear.x = 0.13;           twist.angular.z = 0.0;
            elif x== ['30_l']:
                twist.linear.x = 0.10;           twist.angular.z = 0.4;
            elif x== ['30_r']:
                twist.linear.x = 0.10;           twist.angular.z = -0.4;
            elif x== ['45_0']:
                twist.linear.x = 0.13;           twist.angular.z = 0.0;
            elif x== ['45_l']:
                twist.linear.x = 0.10;           twist.angular.z = 0.3;
            elif x== ['45_r']:
                twist.linear.x = 0.10;           twist.angular.z = -0.3;
            elif x== ['15_0']:
                twist.linear.x = 0.0;            twist.angular.z = 0.0;
            elif x== ['empty']:
                twist.linear.x = 0.0;            twist.angular.z = 0.0;
            else:
                twist.linear.x = 0.0;            twist.angular.z = 0.0;

            self.pub.publish(twist)
            rospy.loginfo(x)

```

Figure 4.7 detail explanation on data collection of follower turtlebot3

In this figure, this is the general command of the follower turtlebot3 while maintaining its formation with the leader. The value of x is determined by the Lidar sensor that scanning the leader behaviour in front. the self labelling such as '30_0', '30_l', '30_r', '45_0', '45_l', '45_r', '15_0', and 'empty' have their own velocity on linear x and angular z. for example, if the value is '30_l' from the leader behaviour, the follower turtlebot3 will have a command on linear x which is 0.10 meter per second and angular z 0.4 radian per second. Logically, the follower turtlebot will move 30 degree to the left. Same with if the value is '45_r' from the leader, the follower turtlebot will move its linear by 0.10 meter per second and angular value of -0.3 radian per second which is to the right.

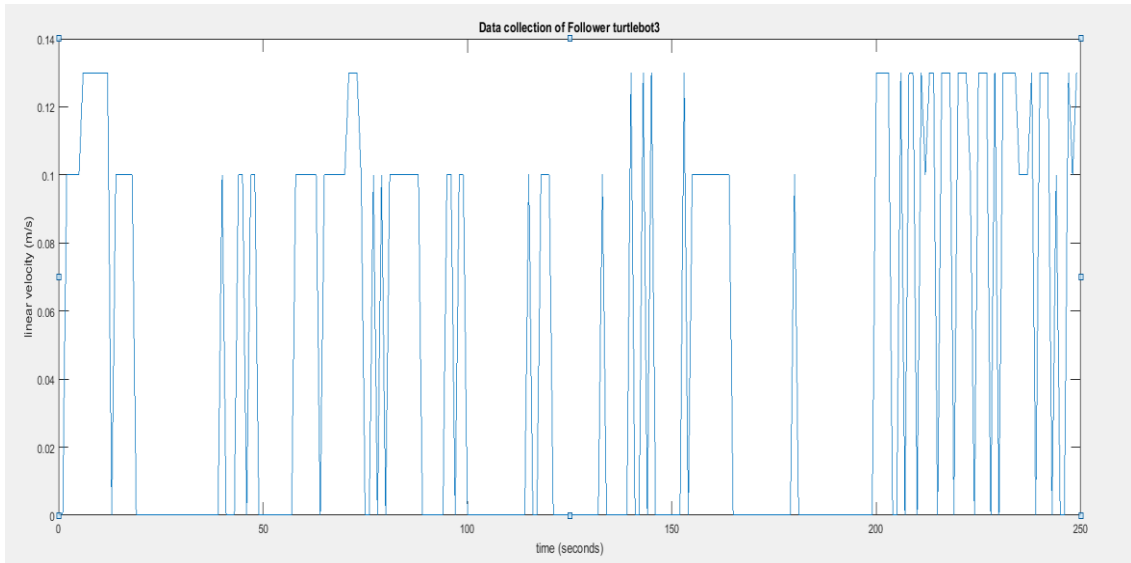


Figure 4.8 linear velocity of follower turtlebot3

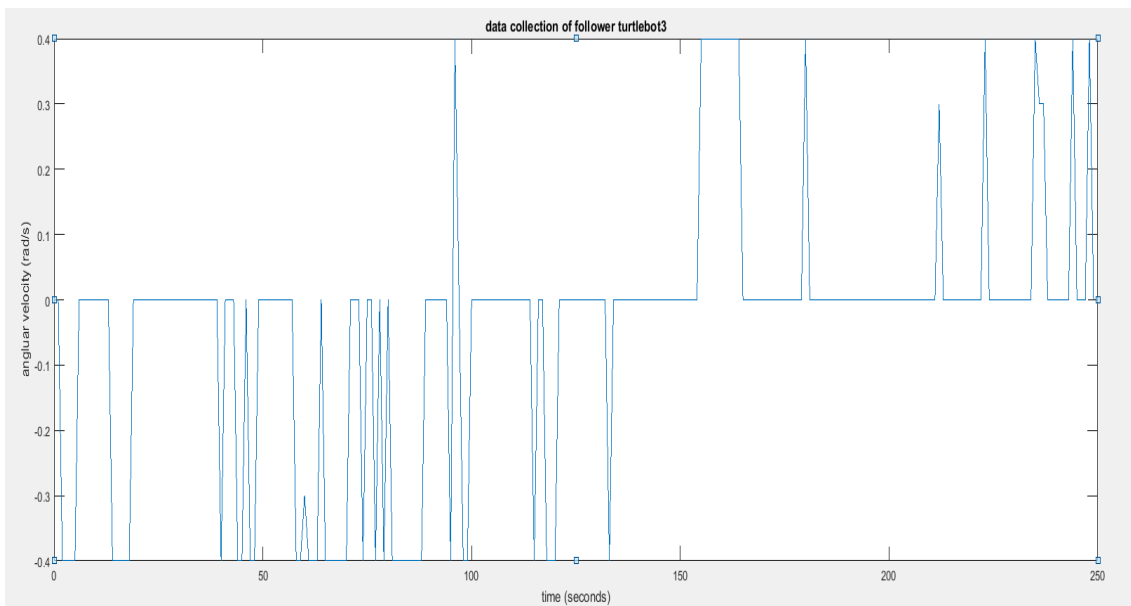


Figure 4.9 angular velocity of follower turtlebot3

This line graph is being implemented from the result taken on data collection of follower turtlebot3. Both are in same value of time in seconds. In figure 4.6, the linear velocity recorded is in between 0 until 0.13 m/s as it already be used in the python nodes. In figure 4.7, the angular velocity here is up and down which means that the follower is moving to the left and right. If the line graph is moving down, the follower is moving to the right while if the line is moving up, the follower is moving to the left.

To know that the Lidar of follower turtlebot is always scanning the leader, the `rospy.loginfo` is being used here to display on the information of Lidar that scanning if it detect the leader or not. The value of display here should be 1 if the Lidar is functioning and detect the leader in front. Else, it should be 0.

```
[INFO] [1643036420.524565]: 1
[INFO] [1643036420.923676]: 1
[INFO] [1643036421.331474]: 1
[INFO] [1643036421.733284]: 1
[INFO] [1643036422.133695]: 1
[INFO] [1643036422.528171]: 1
[INFO] [1643036422.931859]: 1
[INFO] [1643036423.334711]: 1
[INFO] [1643036423.728258]: 1
[INFO] [1643036424.131118]: 1
[INFO] [1643036424.535435]: 1
[INFO] [1643036424.946634]: 1
[INFO] [1643036425.474362]: 1
[INFO] [1643036425.965905]: 1
[INFO] [1643036426.151136]: 1
[INFO] [1643036426.542496]: 1
[INFO] [1643036426.951428]: 1
[INFO] [1643036427.342224]: 1
[INFO] [1643036427.746141]: 1
[INFO] [1643036428.154007]: 1
[INFO] [1643036428.546079]: 1
[INFO] [1643036428.950079]: 1
[INFO] [1643036429.353768]: 1
[INFO] [1643036429.765307]: 1
[INFO] [1643036430.156498]: 1
[INFO] [1643036430.556984]: 1
[INFO] [1643036430.968575]: 1
[INFO] [1643036431.554923]: 1
[INFO] [1643036431.954707]: 1
[INFO] [1643036432.360382]: 1
[INFO] [1643036432.769606]: 1
[INFO] [1643036433.165899]: 1
[INFO] [1643036433.565840]: 1
[INFO] [1643036433.968018]: 1
[INFO] [1643036434.369478]: 1
[INFO] [1643036434.780119]: 1
[INFO] [1643036435.383613]: 1
[INFO] [1643036435.796113]: 1
[INFO] [1643036436.177793]: 1
[INFO] [1643036436.570512]: 1
[INFO] [1643036436.973856]: 1
[INFO] [1643036437.375181]: 1
[INFO] [1643036437.781097]: 1
[INFO] [1643036438.184097]: 1
[INFO] [1643036438.590036]: 1
[INFO] [1643036438.984696]: 1
[INFO] [1643036439.388202]: 1
[INFO] [1643036439.780089]: 1
[INFO] [1643036440.190801]: 1
[INFO] [1643036440.585573]: 1
[INFO] [1643036440.990059]: 1
[INFO] [1643036441.397430]: 1
[INFO] [1643036441.794601]: 1
[INFO] [1643036442.194628]: 1
[INFO] [1643036442.606236]: 1
[INFO] [1643036443.051645]: 1
[INFO] [1643036443.393103]: 1
[INFO] [1643036443.795662]: 1
[INFO] [1643036444.195235]: 1
[INFO] [1643036444.606807]: 1
[INFO] [1643036445.038087]: 1
```

Figure 4.10 Lidar value on detecting the leader

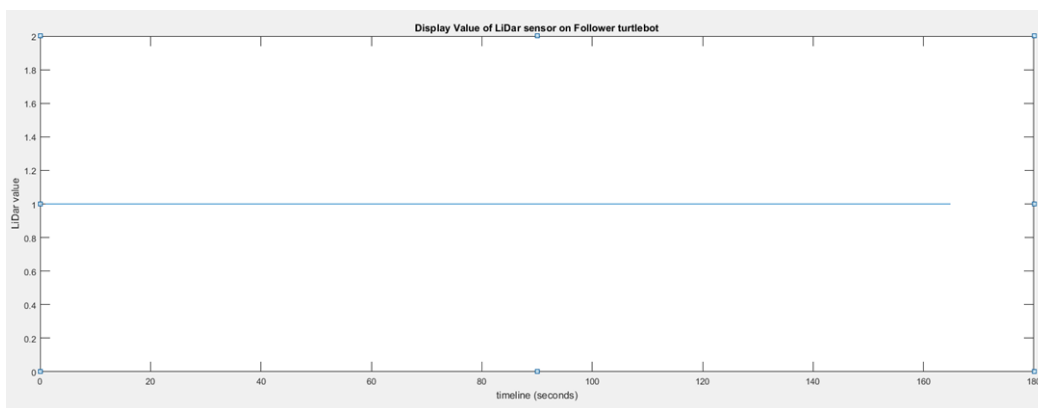


Figure 4.11 Lidar value on detecting the leader

4.3.2 Formation control with different environment

In getting more result from the follower turtlebot3, different environment has been setup for the follower to run its trajectory. The square-shape and circle-shape environment is setup to implement the leader follower formation control. With the leader moving first in the certain environment, the follower turtlebot will follow and the data of the follower turtlebot3 will be collected for future analysis.

4.3.2.1 Square-shape environment

In this situation, there are a track for the follower to move around which is the square shape. The follower turtlebot3 will be in the behind to follow the leader. By running in this environment, the terminal from the ROS master will collect the data of Lidar sensor from the follower turtlebot3.



Figure 4.12 Square-shape path

Below are the result taken after the experiment is held. Basically, the follower turtlebot should make a square pattern by providing velocity to go straight for a certain second and turn to the the right for about three times. Because of the nodes didn't create a 90 degree instruction, the follower will moving to right with the angle of 30 degree and 45 degree only.

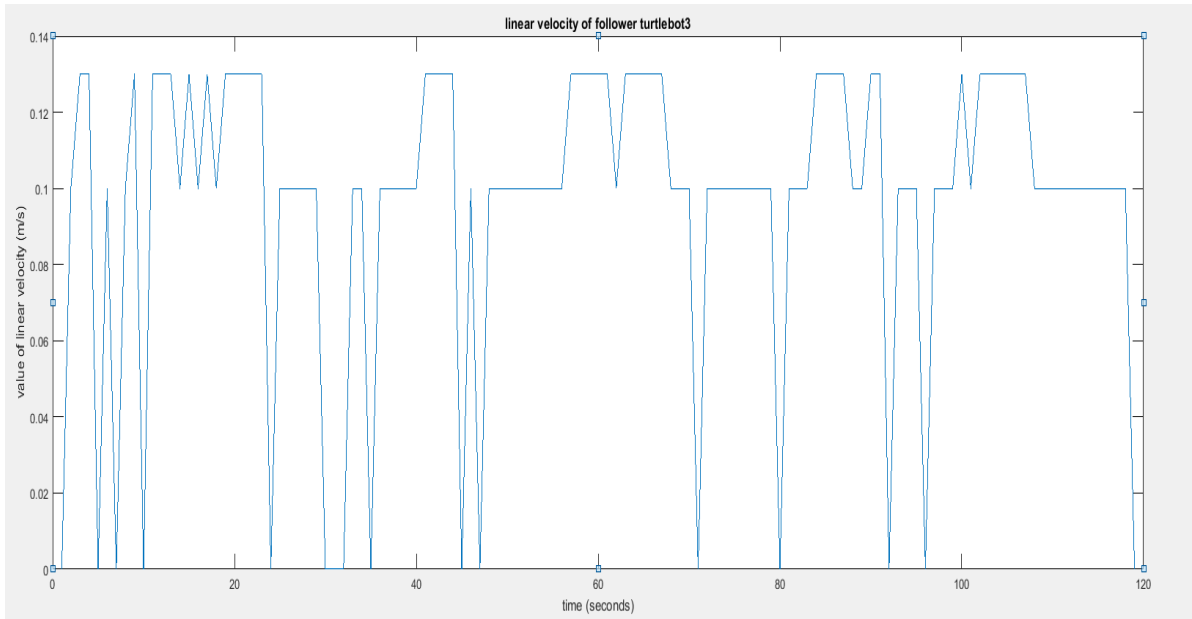


Figure 4.13 linear velocity of follower turtlebot3 with square-shape environment

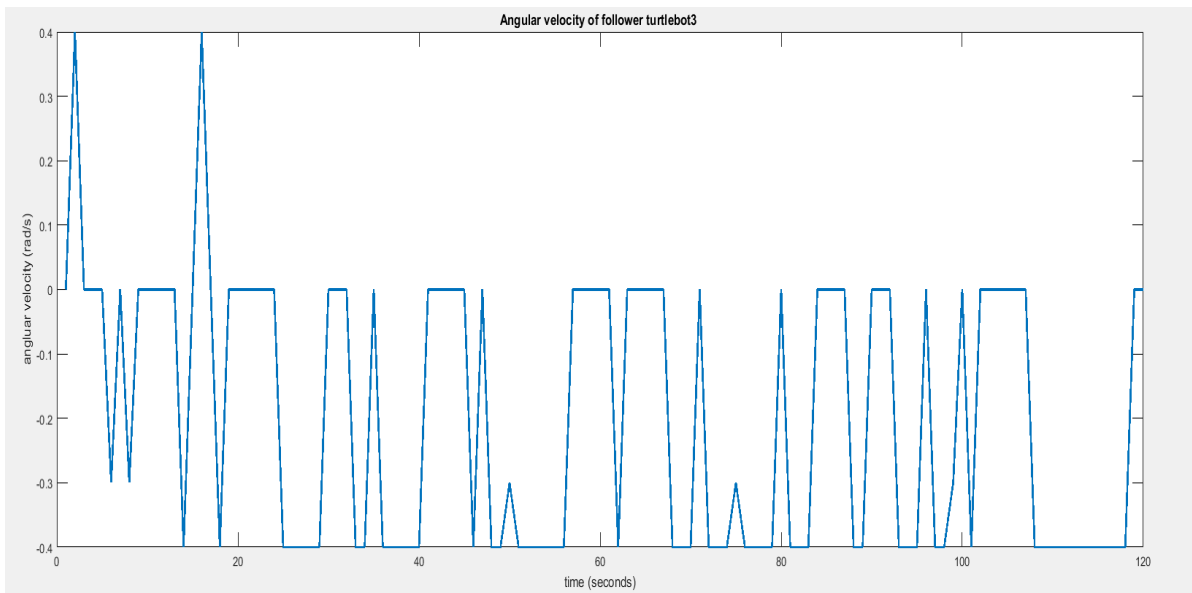


Figure 4.14 angular velocity of follower turtlebot3 with square-shape environment

From this line graph, we can see that in the linear velocity, the follower turtlebot has a good velocity in following the leader which has a non-stop velocity. Only a certain of time, there is a zero value in the linear velocity and basically, the follower is searching of the leader which changing it direction on that time. It is about 15 times the follower has a zero value of velocity and this is quite acceptable. But unfortunately, in the angular velocity, the follower turtlebot is always measuring and getting the input on the different

direction of the leader. As same in the explanation before, the follower turtlebot didn't have a good value of 90 degree in changing of the direction to make the corner of the square. That is why the value in the line graph, we can see the value is always present which is in 30 degree and 45 degree. Basically, the square-shape is failed to be created in this formation control and as this is only the experiment, this can be concluded that the path of square-shape is not really suitable with the current nodes on the follower turtlebot3. This can be develop by adding the 90 degree selection for the future development.

4.3.2.2 Circle-shape environment

In this situation, a circle-shape track for the follower to move around is being setup. The follower turtlebot3 will be in the behind to follow the leader. By running in this environment, the terminal from the ROS master will collect the data of Lidar sensor from the follower turtlebot3. Theoretically, the data should be always moving to the right with certain angular and linear velocity as it is a form of a circle. In the table below is the detection of leader pose from the follower turtlebot. In this different self.labels type of data, the output is different between each data depends on the input detected which is being shown in the figure 4.8.

time (ns)	leader pose	time (ns)	leader pose	time (ns)	leader pose
[1644396145.316062]:	['30_0']	[1644396155.347643]:	['30_r']	[1644396165.377178]:	['30_r']
[1644396145.718016]:	['30_0']	[1644396155.748362]:	['30_r']	[1644396165.787617]:	['empty']
[1644396146.118028]:	['30_0']	[1644396156.149316]:	['empty']	[1644396166.189090]:	['30_r']
[1644396146.517961]:	['empty']	[1644396156.553178]:	['30_r']	[1644396166.590769]:	['30_r']
[1644396146.919035]:	['empty']	[1644396156.954503]:	['30_r']	[1644396167.195890]:	['30_r']
[1644396147.323603]:	['30_r']	[1644396157.356197]:	['30_r']	[1644396167.595446]:	['30_r']
[1644396147.730992]:	['empty']	[1644396157.756376]:	['30_r']	[1644396167.991100]:	['empty']
[1644396148.121704]:	['empty']	[1644396158.157825]:	['30_0']	[1644396168.394881]:	['45_0']
[1644396148.521095]:	['30_r']	[1644396158.621294]:	['empty']	[1644396168.798674]:	['30_r']
[1644396148.934106]:	['empty']	[1644396159.022249]:	['30_r']	[1644396169.202742]:	['30_r']
[1644396149.335215]:	['empty']	[1644396159.366411]:	['empty']	[1644396169.597681]:	['30_r']
[1644396149.734889]:	['empty']	[1644396159.770843]:	['45_r']	[1644396170.007065]:	['30_r']
[1644396150.128608]:	['30_r']	[1644396160.165833]:	['30_r']	[1644396170.402123]:	['empty']
[1644396150.537947]:	['30_r']	[1644396160.568757]:	['30_r']	[1644396170.796791]:	['empty']
[1644396150.941695]:	['30_r']	[1644396160.971131]:	['empty']	[1644396171.197486]:	['30_r']
[1644396151.338965]:	['30_r']	[1644396161.373027]:	['30_r']	[1644396171.603752]:	['30_r']
[1644396151.740137]:	['30_r']	[1644396161.773594]:	['30_r']	[1644396172.003912]:	['30_r']
[1644396152.141793]:	['empty']	[1644396162.175646]:	['45_r']	[1644396172.412011]:	['30_r']
[1644396152.541497]:	['30_0']	[1644396162.582060]:	['empty']	[1644396172.812657]:	['45_0']
[1644396152.940296]:	['30_r']	[1644396162.972890]:	['empty']	[1644396173.214397]:	['45_0']
[1644396153.350992]:	['30_r']	[1644396163.376978]:	['30_r']	[1644396173.613889]:	['30_r']
[1644396153.741664]:	['30_r']	[1644396163.777522]:	['empty']	[1644396174.012308]:	['30_r']
[1644396154.153042]:	['empty']	[1644396164.182160]:	['empty']	[1644396174.421380]:	['30_r']
[1644396154.545270]:	['45_0']	[1644396164.584725]:	['45_r']	[1644396174.816728]:	['30_r']
[1644396154.957876]:	['30_r']	[1644396164.978245]:	['45_r']	[1644396175.214902]:	['30_r']

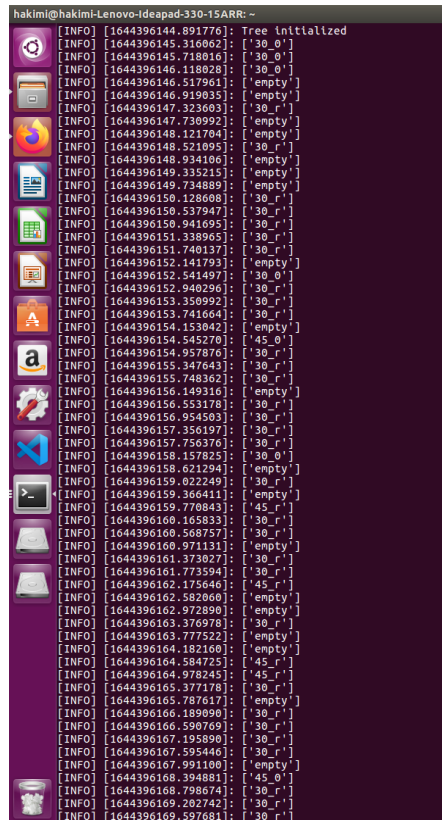
time (ns)	leader pose	time (ns)	leader pose
[1644396175.617891]:	['30_r']	[1644396186.059007]:	['empty']
[1644396176.020789]:	['45_0']	[1644396186.450189]:	['30_r']
[1644396176.420372]:	['30_r']	[1644396186.851003]:	['30_r']
[1644396176.824244]:	['30_r']	[1644396187.253341]:	['30_r']
[1644396177.227065]:	['30_r']	[1644396187.658186]:	['30_r']
[1644396177.630702]:	['30_r']	[1644396188.053907]:	['30_r']
[1644396178.022802]:	['30_r']	[1644396188.460601]:	['30_r']
[1644396178.427637]:	['30_r']	[1644396188.854933]:	['45_r']
[1644396178.831530]:	['45_0']	[1644396189.262180]:	['45_r']
[1644396179.232023]:	['empty']	[1644396189.663160]:	['30_r']
[1644396179.635026]:	['30_r']	[1644396190.064801]:	['empty']
[1644396180.026534]:	['30_r']		['empty']
[1644396180.434896]:	['45_0']		
[1644396180.839134]:	['45_0']		
[1644396181.352081]:	['30_r']		
[1644396181.633617]:	['30_r']		
[1644396182.138361]:	['30_r']		
[1644396182.643380]:	['30_r']		
[1644396183.264141]:	['30_r']		
[1644396183.647167]:	['30_r']		
[1644396184.060617]:	['30_r']		
[1644396184.445831]:	['30_r']		
[1644396184.850765]:	['30_r']		
[1644396185.253763]:	['45_0']		
[1644396185.648768]:	['30_r']		

Table 4.2 Leader position value from follower



Figure 4.15 Circle-shape track

Below are the result taken after the experiment for the circle-shape environment is held. Basically, the follower turtlebot should make a circle pattern by providing velocity and angular velocity of 30 degree to go around.



```
hakimi@hakimi-Lenovo-Ideapad-330-15ARR: ~
[INFO] [1644396144.891776]: Tree initialized
[INFO] [1644396145.316082]: ['30_0']
[INFO] [1644396145.718016]: ['30_0']
[INFO] [1644396146.118028]: ['30_0']
[INFO] [1644396146.517961]: ['empty']
[INFO] [1644396146.919035]: ['empty']
[INFO] [1644396147.323093]: ['30_r']
[INFO] [1644396147.720992]: ['empty']
[INFO] [1644396148.121704]: ['empty']
[INFO] [1644396148.521095]: ['30_r']
[INFO] [1644396148.934106]: ['empty']
[INFO] [1644396149.335215]: ['empty']
[INFO] [1644396149.734889]: ['empty']
[INFO] [1644396150.128608]: ['30_r']
[INFO] [1644396150.537947]: ['30_r']
[INFO] [1644396150.941695]: ['30_r']
[INFO] [1644396151.338905]: ['30_r']
[INFO] [1644396151.740137]: ['30_r']
[INFO] [1644396152.141793]: ['empty']
[INFO] [1644396152.541497]: ['30_0']
[INFO] [1644396152.940296]: ['30_r']
[INFO] [1644396153.350992]: ['30_r']
[INFO] [1644396153.741664]: ['30_r']
[INFO] [1644396154.153042]: ['empty']
[INFO] [1644396154.545270]: ['45_0']
[INFO] [1644396154.957876]: ['30_r']
[INFO] [1644396155.347643]: ['30_r']
[INFO] [1644396155.748302]: ['30_r']
[INFO] [1644396156.149316]: ['empty']
[INFO] [1644396156.553178]: ['30_r']
[INFO] [1644396156.954503]: ['30_r']
[INFO] [1644396157.350197]: ['30_r']
[INFO] [1644396157.750376]: ['30_r']
[INFO] [1644396158.157825]: ['30_0']
[INFO] [1644396158.621294]: ['empty']
[INFO] [1644396159.022249]: ['30_r']
[INFO] [1644396159.366411]: ['empty']
[INFO] [1644396159.770843]: ['45_r']
[INFO] [1644396160.165833]: ['30_r']
[INFO] [1644396160.568757]: ['30_r']
[INFO] [1644396160.971131]: ['empty']
[INFO] [1644396161.373027]: ['30_r']
[INFO] [1644396161.773394]: ['30_r']
[INFO] [1644396162.175646]: ['45_r']
[INFO] [1644396162.582060]: ['empty']
[INFO] [1644396162.972890]: ['empty']
[INFO] [1644396163.376978]: ['30_r']
[INFO] [1644396163.777522]: ['empty']
[INFO] [1644396164.182160]: ['empty']
[INFO] [1644396164.584725]: ['45_r']
[INFO] [1644396164.978245]: ['45_r']
[INFO] [1644396165.377178]: ['30_r']
[INFO] [1644396165.787617]: ['empty']
[INFO] [1644396166.189090]: ['30_r']
[INFO] [1644396166.590769]: ['30_r']
[INFO] [1644396167.195890]: ['30_r']
[INFO] [1644396167.595446]: ['30_r']
[INFO] [1644396167.991100]: ['empty']
[INFO] [1644396168.394881]: ['45_0']
[INFO] [1644396168.798674]: ['30_r']
[INFO] [1644396169.202742]: ['30_r']
[INFO] [1644396169.597681]: ['30_r']
```

Figure 4.16 the visual from the terminal on follower turtlebot3 in circle-shape environment

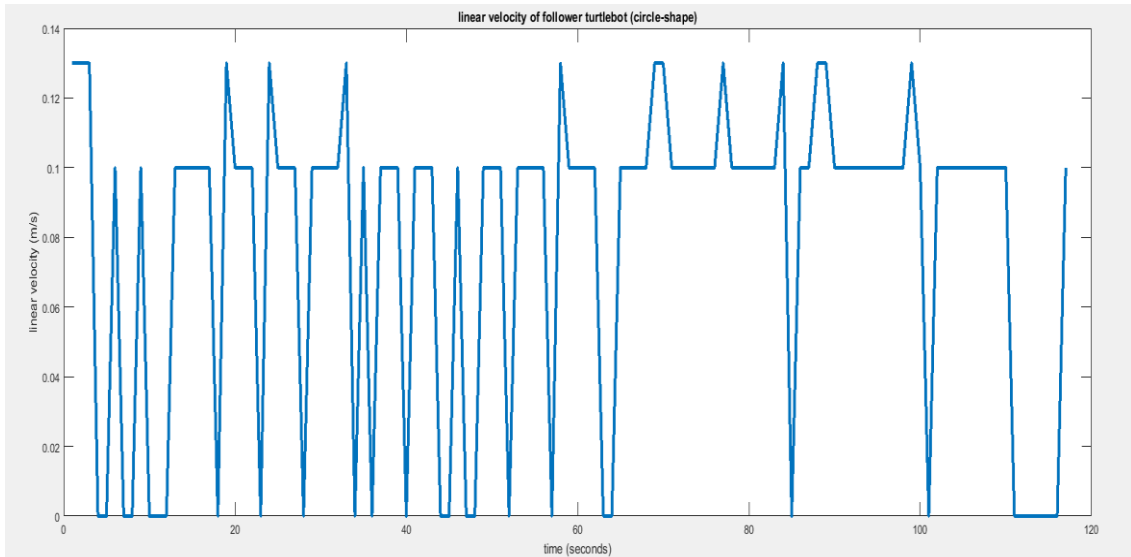


Figure 4.17 Linear velocity of follower turtlebot with circle-shape environment

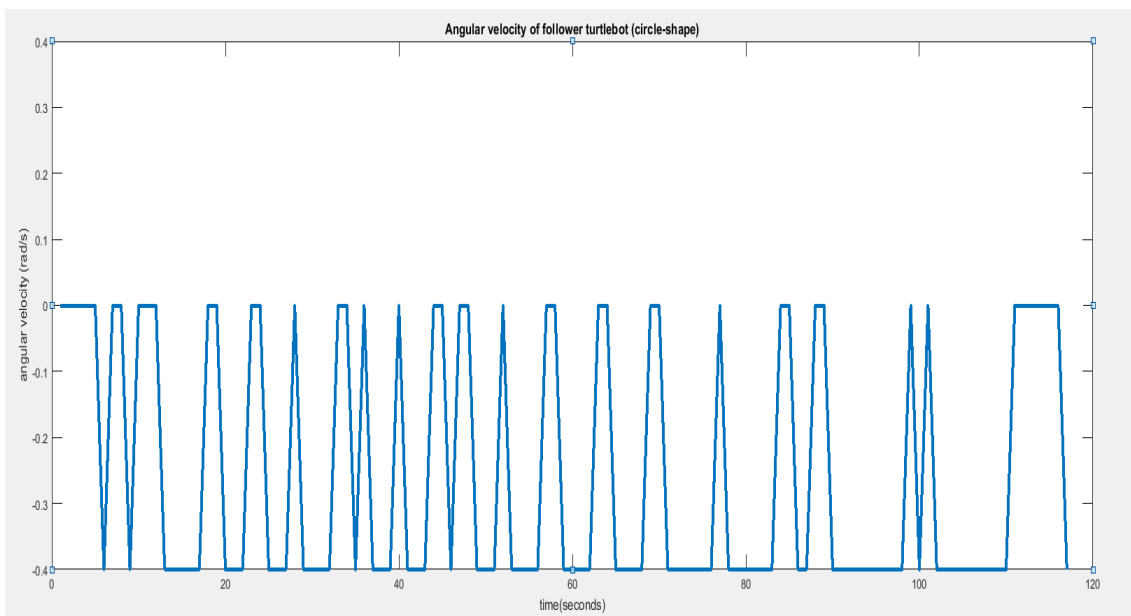


Figure 4.18 angular velocity of follower turtlebot with circle-shape environment

In this result taken, we can see that the velocity in linear direction is maintain in 0.1 m/s with some value of 0.13 m/s. there is also a zero value in velocity as it is in static for about 1 second only and start moving back as this is the process in waiting and scanning the leader trajectory in front. basically, the value of linear is acceptable as it can maintain the velocity while following the leader in front which has a non-stop velocity on the circle-shape environment. In the angular velocity illustrated in figure 4.16, we

already know that the value of positive term in radian per second is to the left and the negative value is to the right. So in this figure , we can see that the follower turtlebot is successfully recorded a value of negative 0.4 rad/s which is always detecting the leader moving to the right and it successfully maintain the direction and follow the leader. This is also a good result as it has zero percent of error on moving to the left. the direction of follower is always to the right with certain time to stop when the linear velocity also stop. So, the circle-shape environment seem can be implemented in this follower turtlebot in getting the formation control as in generate a circle , it only need a 30 degree or 45 degree while maintain this angular position until reach the initial place.

CHAPTER 5

CONCLUSION

5.1 PROJECT CONCLUSION

As a conclusion, ROS framework is very powerful tools to simulate or running real robot for multi-purpose. The final year project really teach me on learn new things especially about robotics and this will be explore more and more in my future career. for the final one, The objective on this project which is To design a mobile robot leader follower with obstacle avoidance and to analyze the robot formation control based on the desired position in different environment is achieved. The leader can have a movement by putting the LiDar ON to ensure the obstacle is detected and be avoid. For the follower, the desired geometric pattern is determined by the leader although it has some error and delay on the practical of the hardware. This project really have a big opportunity to be developed in the future development. The impact in society from this application is really promising as it can help people to facilitate their daily lives such as helping in transferring packaging from one point to another with heavy objects or item that require many vehicles to move in formation and this application. By having the obstacle avoidance in the application, the user didn't have to worry about incident that might occur as it will avoid any possibility on getting the incident happen.

REFERENCES

- [1] Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6), 926–939. <https://doi.org/10.1109/70.736776>
- [2] Ma, Z., Zhu, L., Wang, P., & Zhao, Y. (2019). *ROS-Based Multi-Robot System Simulator*. 4228–4232.
- [3] Sequeira, G. (2007). Vision Based Leader-follower Formation Control for Mobile Robots. *Evolution*. Retrieved from http://scholarsmine.mst.edu/thesis/pdf/Sequeira_09007dcc804429d4.pdf
- [4] Baquero-Suárez, Mauro & Mas, Ignacio & Giribet, Juan. (2020). Robust Tracking Control for Non-holonomic Wheeled Mobile Robots in a Leader Follower Formation with Time gap Separation.
- [5] Qian, D.W., Tong, S.W. and Li, C.D., 2017. Observer-based leader-following formation control of uncertain multiple agents by integral sliding mode. *Bulletin of the Polish Academy of Sciences. Technical Sciences*, 65(1), pp.35-44.
- [6] A. Alfaro and A. Morán, "Leader-Follower Formation Control of Nonholonomic Mobile Robots," 2020 IEEE ANDESCON, 2020, pp. 1-6, doi: 10.1109/ANDESCON50619.2020.9272048.
- [7] Andre, T., Neuhold, D. and Bettstetter, C. (2014) Coordinated multi-robot exploration: Out of the box packages for ROS. In Globecom Workshops (GC Wkshps), pp. 1457-1462, IEEE.
- [8] Guanghua, W., Deyi, L., Wenyan, G. and Peng, J. (2013) Study on formation control of multi-robot systems. In Intelligent System Design and Engineering Applications, 2013 3rd Int. Conf on pp. 1335-1339, IEEE.
- [9] Hennes, D., Claes, D., Meeussen, W. and Tuyls, K. (2012) Multi-robot collision avoidance with localization uncertainty. In Proc. 11th Int. Conf. on Autonomous Agents and Multiagent Systems-Volume 1, pp. 147-154.
- [10] Muddu, R.S.D., Wu, D. and Wu, L. (2015) A frontier based multi-robot approach for coverage of unknown environments. In Robotics and Biomimetics (ROBIO), 2015 IEEE Int. Conf on pp. 72-77, IEEE.
- [11] Quigley M., Gerky B. and Smart W.D. (2015) Programming Robots with ROS, a Practical Introduction to the Robot Operating System, First. O'Reilly Media, Inc.
- [12] Reis, J.C., Lima, P.U. and Garcia, J. (2013) Efficient distributed communications for multi-robot systems. In Robot Soccer World Cup, pp. 280-291, Springer Berlin Heidelberg.
- [13] Tiderko, A., Hoeller, F. and Röhling, T. (2016) The ROS Multimaster Extension for Simplified Deployment of Multi-Robot Systems, chapter in Robot Operating System (ROS) Volume 625 of the series Studies in Computational Intelligence pp. 629-650.
- [14] Cherubini, A., Spindler, F. and Chaumette, F., 2014. Autonomous visual navigation and laser-based moving obstacle avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 15(5), pp.2101-2110.
- [15] Turnage, D.M., 2016, December. Simulation results for localization and mapping algorithms. In 2016 Winter Simulation Conference (WSC) (pp. 3040-3051). IEEE.

- [16] Omara, H.I.M.A. and Sahari, K.S.M., 2015, August. Indoor mapping using kinect and ROS. In 2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR) (pp. 110-116). IEEE.
- [17] P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 4, 1998, pp. 2864–2869.
- [18] . E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in Proceedings of the IEEE Conference on Decision and Control (CDC), vol. 3, 2001, pp. 2968–2973.
- [19] H. Su, X. Wang, and Z. Lin, "Flocking of multi-agents with a virtual leader," IEEE Transactions on Automatic Control, vol. 54, no. 2, pp. 293–307, 2009.
- [20] H. Chen, D. Sun, J. Yang, and J. Chen, "Localization for multirobot formations in indoor environment," IEEE/ASME Transactions on Mechatronics, vol. 15, no. 4, pp. 561–574, 2010.
- [21] Sakai, H. Fukushima, and F. Matsuno, "Leader-follower navigation in obstacle environments while preserving connectivity without data transmission," IEEE Transactions on Control Systems Technology, vol. 26, no. 4, pp. 1233– 1248, 2017.
- [22] C. B. Low, "A flexible leader-follower formation tracking control design for nonholonomic tracked mobile robots with low-level velocities control systems," in Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC), 2015, pp. 2424–2431.

APPENDIX A

OBSTACLE AVOIDANCE FOR LEADER TURTLEBOT3

```
#!/usr/bin/env python

import rospy

import math

from sensor_msgs.msg import LaserScan

from geometry_msgs.msg import Twist

LINEAR_VEL = 0.22

STOP_DISTANCE = 0.7

LIDAR_ERROR = 0.05

SAFE_STOP_DISTANCE = STOP_DISTANCE + LIDAR_ERROR

class Obstacle():

    def __init__(self):

        self._cmd_pub = rospy.Publisher('cmd_vel', Twist, queue_size=1)

        self.obstacle()
```

```

def get_scan(self):

    scan = rospy.wait_for_message('scan', LaserScan)

    scan_filter = []

    samples = len(scan.ranges) # The number of samples is defined in

                                # turtlebot3_<model>.gazebo.xacro file,

                                # the default is 360.

    samples_view = 1          # 1 <= samples_view <= samples

    if samples_view > samples:

        samples_view = samples

    if samples_view is 1:

        scan_filter.append(scan.ranges[0])

    else:

        left_lidar_samples_ranges = -(samples_view//2 + samples_view % 2)

        right_lidar_samples_ranges = samples_view//2

```



```
left_lidar_samples = scan.ranges[left_lidar_samples_ranges:]  
  
right_lidar_samples = scan.ranges[:right_lidar_samples_ranges]  
  
scan_filter.extend(left_lidar_samples + right_lidar_samples)
```

```
for i in range(samples_view):
```

```
    if scan_filter[i] == float('Inf'):
```

```
        scan_filter[i] = 3.5
```

```
    elif math.isnan(scan_filter[i]):
```

```
        scan_filter[i] = 0
```

```
return scan_filter
```

```
def obstacle(self):
```

```
    twist = Twist()
```

```
    turtlebot_moving = True
```

```
while not rospy.is_shutdown():
```

```
lidar_distances = self.get_scan()

min_distance = min(lidar_distances)

if min_distance < SAFE_STOP_DISTANCE:

    if turtlebot_moving:

        twist.linear.x = 0.0

        twist.angular.z = 0.7

        self._cmd_pub.publish(twist)

        turtlebot_moving = True

        rospy.loginfo('Stop!')

    else:

        twist.linear.x = LINEAR_VEL

        twist.angular.z = 0.1

        self._cmd_pub.publish(twist)

        turtlebot_moving = True

        rospy.loginfo('Distance of the obstacle : %f', min_distance)

def main():
```

```
rospy.init_node('turtlebot3_obstacle')

try:

    obstacle = Obstacle()

except rospy.ROSInterruptException:

    pass

if __name__ == '__main__':

    main()
```

APPENDIX B

FOLLOWER TURTLEBOT3 PYTHON NODE

```
#!/usr/bin/env python

import rospy

import os

import pickle

from sensor_msgs.msg import LaserScan

from geometry_msgs.msg import Twist

import numpy as np

class follower:

    def __init__(self):

        rospy.loginfo('Follower node initialized')

        self.config_dir = os.path.join(os.path.dirname(__file__))

        self.config_dir = self.config_dir.replace('nodes', 'config')

        self.pub = rospy.Publisher('cmd_vel', Twist, queue_size = 1)

        self.clf = pickle.load(open(self.config_dir + '/clf', "rb"))

        self.clf2 = pickle.load(open(self.config_dir + '/clf2', "rb"))
```

```
self.labels = {'30_0':0, '30_l':1, '30_r':2, '45_0':3, '45_l':4, '45_r':5, '15_0':6, 'empty':7}
```

```
rospy.loginfo('Tree initialized')
```

```
self.follow()
```

```
def check_people(self):
```

```
    laser_data=[]
```

```
    laser_data_set=[]
```

```
    result=[]
```

```
    ret = 0
```

```
    self.msg = rospy.wait_for_message("scan_filtered", LaserScan)
```

```
    for i in range(70,-2,-1) + range(359, 289,-1):
```

```
        if np.nan_to_num( self.msg.intensities[i] ) != 0 :
```

```
            laser_data.append(np.nan_to_num(self.msg.intensities[i]))
```

```
            elif (i+1) in range(70,-2,-1) + range(359, 289,-1) and (i-1) in range(70,-2,-1) + range(359, 289,-1) and np.nan_to_num(self.msg.intensities[i]) == 0:
```

```
laser_data.append((np.nan_to_num(self.msg.intensities[i+1])+np.nan_to_num(self.msg.intensities[i-1]))/2)
```

```
else :
```

```
    laser_data.append(np.nan_to_num(self.msg.intensities[i]))
```

```
laser_data_set.append(laser_data)
```

```
[x for (x , y) in self.labels.iteritems() if y == self.clf2.predict(laser_data_set) ] ##  
Predict the position
```

```
if result == ['empty']:
```

```
    ret = 0
```

```
else:
```

```
    ret = 1
```

```
return ret
```

```

def laser_scan(self):

    data_test=[]

    data_test_set=[]

    self.msg = rospy.wait_for_message("scan_filtered", LaserScan)

    for i in range(70,-2,-1) + range(359, 289,-1):

        if np.nan_to_num( self.msg.ranges[i] ) != 0 :

            data_test.append(np.nan_to_num(self.msg.ranges[i]))

            elif (i+1) in range(70,-2,-1) + range(359, 289,-1) and (i-1) in range(70,-2,-1) +
range(359, 289,-1) and np.nan_to_num(self.msg.ranges[i]) == 0:

data_test.append((np.nan_to_num(self.msg.ranges[i+1])+np.nan_to_num(self.msg.ranges[i-1]))/2)

        else :

            data_test.append(np.nan_to_num(self.msg.ranges[i]))

```

```
data_test_set.append(data_test)
```

```
return [x for (x , y) in self.labels.iteritems() if y == self.clf.predict(data_test_set) ]
```

```
def follow(self):
```

```
    while not rospy.is_shutdown():
```

```
        check = self.check_people()
```

```
        if check == 1:
```

```
            x = self.laser_scan()
```

```
            twist = Twist()
```

```
            ## Do something according to each position##
```

```
            if x == ['30_0']:
```

```
                twist.linear.x = 0.13;          twist.angular.z = 0.0;
```

```
            elif x == ['30_1']:
```

```
                twist.linear.x = 0.10;          twist.angular.z = 0.4;
```

```
            elif x == ['30_r']:
```

```
                twist.linear.x = 0.10;          twist.angular.z = -0.4;
```



```
elif x== ['45_0']:

    twist.linear.x = 0.13;          twist.angular.z = 0.0;

elif x== ['45_1']:

    twist.linear.x = 0.10;          twist.angular.z = 0.3;

elif x== ['45_r']:

    twist.linear.x = 0.10;          twist.angular.z = -0.3;

elif x== ['15_0']:

    twist.linear.x = 0.0;           twist.angular.z = 0.0;

elif x== ['empty']:

    twist.linear.x = 0.0;           twist.angular.z = 0.0;

else:

    twist.linear.x = 0.0;           twist.angular.z = 0.0;

self.pub.publish(twist)

rospy.loginfo(x)

elif check == 0:
```

```

x = self.laser_scan()

twist = Twist()

if x== ['empty']:

    twist.linear.x = 0.0;          twist.angular.z = 0.0;

else:

    twist.linear.x = 0.1;          twist.angular.z = 0.0;

self.pub.publish(twist)

rospy.loginfo(laser_scan)

def main():

    rospy.init_node('follower', anonymous=True)

    try:

        follow = follower()

    except rospy.ROSInterruptException:

```

```
pass #print("Shutting down")
```

```
if __name__ == '__main__':
```

```
    main()
```

APPENDIX D

DATA COLLECTION OF FOLLOWER TURTLEBOT3

[INFO] [1642953498.024087]: ['empty']	[INFO] [1642953504.245828]: ['30_r']
[INFO] [1642953498.618344]: ['30_r']	[INFO] [1642953504.651094]: ['30_r']
[INFO] [1642953499.022471]: ['30_r']	[INFO] [1642953505.044246]: ['30_r']
[INFO] [1642953499.426673]: ['30_r']	[INFO] [1642953505.448006]: ['empty']
[INFO] [1642953499.827487]: ['30_r']	[INFO] [1642953505.852374]: ['empty']
[INFO] [1642953500.231990]: ['45_0']	[INFO] [1642953506.248112]: ['empty']
[INFO] [1642953500.637717]: ['30_0']	[INFO] [1642953506.654346]: ['empty']
[INFO] [1642953501.032185]: ['30_0']	[INFO] [1642953507.057713]: ['empty']
[INFO] [1642953501.432454]: ['30_0']	[INFO] [1642953507.455013]: ['empty']
[INFO] [1642953501.832354]: ['45_0']	[INFO] [1642953507.861526]: ['empty']
[INFO] [1642953502.234753]: ['30_0']	[INFO] [1642953508.259567]: ['empty']
[INFO] [1642953502.640715]: ['30_0']	[INFO] [1642953508.866487]: ['empty']
[INFO] [1642953503.039785]: ['empty']	
[INFO] [1642953503.441471]: ['30_r']	
[INFO] [1642953503.841513]: ['30_r']	

[INFO] [1642953509.469724]:
['empty']

[INFO] [1642953514.685230]:
['empty']

[INFO] [1642953509.865291]:
['empty']

[INFO] [1642953515.074498]:
['empty']

[INFO] [1642953510.264647]:
['empty']

[INFO] [1642953515.478597]:
['empty']

[INFO] [1642953510.665582]:
['empty']

[INFO] [1642953515.876822]: ['30_r']

[INFO] [1642953511.073892]:
['empty']

[INFO] [1642953516.287913]: ['30_r']

[INFO] [1642953511.468513]:
['empty']

[INFO] [1642953516.687519]:
['empty']

[INFO] [1642953511.872470]:
['empty']

[INFO] [1642953517.089810]: ['30_r']

[INFO] [1642953512.266115]:
['empty']

[INFO] [1642953517.490558]: ['30_r']

[INFO] [1642953512.677109]:
['empty']

[INFO] [1642953517.889276]:
['empty']

[INFO] [1642953513.080507]:
['empty']

[INFO] [1642953518.282736]:
['empty']

[INFO] [1642953513.474179]:
['empty']

[INFO] [1642953518.694797]:
['empty']

[INFO] [1642953513.878408]:
['empty']

[INFO] [1642953519.094489]:
['empty']

[INFO] [1642953514.281764]: ['30_r']

[INFO] [1642953519.497933]:
['empty']

[INFO] [1642953519.896828]:
['empty']

[INFO] [1642953520.298637]: ['empty']	[INFO] [1642953526.721991]: ['45_0']
[INFO] [1642953520.703098]: ['empty']	[INFO] [1642953527.116428]: ['45_0']
[INFO] [1642953521.104286]: ['empty']	[INFO] [1642953527.726200]: ['45_0']
[INFO] [1642953521.501724]: ['30_r']	[INFO] [1642953528.325558]: ['30_r']
[INFO] [1642953521.903247]: ['30_r']	[INFO] [1642953528.727784]: ['empty']
[INFO] [1642953522.303040]: ['45_r']	[INFO] [1642953529.126484]: ['empty']
[INFO] [1642953522.708443]: ['30_r']	[INFO] [1642953529.530567]: ['30_r']
[INFO] [1642953523.107134]: ['30_r']	[INFO] [1642953529.932710]: ['empty']
[INFO] [1642953523.506727]: ['30_r']	[INFO] [1642953530.328914]: ['30_r']
[INFO] [1642953523.909773]: ['empty']	[INFO] [1642953530.733977]: ['empty']
[INFO] [1642953524.314307]: ['30_r']	[INFO] [1642953531.132950]: ['30_r']
[INFO] [1642953524.714441]: ['30_r']	[INFO] [1642953531.532766]: ['30_r']
[INFO] [1642953525.114466]: ['30_r']	[INFO] [1642953531.945474]: ['30_r']
[INFO] [1642953525.517591]: ['30_r']	[INFO] [1642953532.343722]: ['30_r']
[INFO] [1642953525.917470]: ['30_r']	[INFO] [1642953532.741268]: ['30_r']
[INFO] [1642953526.320560]: ['30_r']	[INFO] [1642953533.136322]: ['30_r']

[INFO] [1642953533.543519]: ['30_r']	[INFO] [1642953539.152166]: ['empty']
[INFO] [1642953533.941081]: ['30_r']	[INFO] [1642953539.560763]: ['empty']
[INFO] [1642953534.341845]: ['empty']	[INFO] [1642953539.960628]: ['empty']
[INFO] [1642953534.742405]: ['empty']	[INFO] [1642953540.358363]: ['empty']
[INFO] [1642953535.153562]: ['empty']	[INFO] [1642953540.765460]: ['empty']
[INFO] [1642953535.552954]: ['empty']	[INFO] [1642953541.169982]: ['empty']
[INFO] [1642953535.954833]: ['empty']	[INFO] [1642953541.573522]: ['empty']
[INFO] [1642953536.354101]: ['empty']	[INFO] [1642953541.963442]: ['empty']
[INFO] [1642953536.756604]: ['30_r']	[INFO] [1642953542.373957]: ['empty']
[INFO] [1642953537.156327]: ['30_l']	[INFO] [1642953542.778054]: ['empty']
[INFO] [1642953537.559065]: ['empty']	[INFO] [1642953543.176094]: ['empty']
[INFO] [1642953537.950613]: ['30_r']	[INFO] [1642953543.578263]: ['empty']
[INFO] [1642953538.352241]: ['30_r']	
[INFO] [1642953538.765476]: ['empty']	

[INFO] [1642953543.977728]:
['empty']

[INFO] [1642953549.593620]:
['empty']

[INFO] [1642953544.376440]:
['empty']

[INFO] [1642953550.001861]:
['empty']

[INFO] [1642953544.777986]: ['30_r']

[INFO] [1642953550.398514]:
['empty']

[INFO] [1642953545.180182]:
['empty']

[INFO] [1642953550.809656]:
['empty']

[INFO] [1642953545.583602]:
['empty']

[INFO] [1642953551.201761]:
['empty']

[INFO] [1642953545.982353]: ['30_r']

[INFO] [1642953551.608838]:
['empty']

[INFO] [1642953546.385005]: ['30_r']

[INFO] [1642953551.996842]: ['30_r']

[INFO] [1642953546.784830]: ['30_r']

[INFO] [1642953547.183548]:
['empty']

[INFO] [1642953552.396731]:
['empty']

[INFO] [1642953547.591239]:
['empty']

[INFO] [1642953552.807073]:
['empty']

[INFO] [1642953547.991574]:
['empty']

[INFO] [1642953553.205408]:
['empty']

[INFO] [1642953548.396207]:
['empty']

[INFO] [1642953553.605661]:
['empty']

[INFO] [1642953548.787076]:
['empty']

[INFO] [1642953554.008976]:
['empty']

[INFO] [1642953549.196105]:
['empty']

[INFO] [1642953554.407230]:
['empty']

[INFO] [1642953554.806702]: ['30_0']	[INFO] [1642953560.435362]: ['empty']
[INFO] [1642953555.208967]: ['empty']	[INFO] [1642953560.832022]: ['30_1']
[INFO] [1642953555.615635]: ['empty']	[INFO] [1642953561.434809]: ['30_1']
[INFO] [1642953556.015441]: ['30_0']	[INFO] [1642953561.840041]: ['30_1']
[INFO] [1642953556.413594]: ['empty']	[INFO] [1642953562.238964]: ['30_1']
[INFO] [1642953556.854244]: ['30_0']	[INFO] [1642953562.643440]: ['30_1']
[INFO] [1642953557.291650]: ['empty']	[INFO] [1642953563.044613]: ['30_1']
[INFO] [1642953557.620165]: ['empty']	[INFO] [1642953563.448445]: ['30_1']
[INFO] [1642953558.025045]: ['empty']	[INFO] [1642953563.845875]: ['30_1']
[INFO] [1642953558.426115]: ['empty']	[INFO] [1642953564.250599]: ['30_1']
[INFO] [1642953558.826384]: ['empty']	[INFO] [1642953564.650024]: ['30_1']
[INFO] [1642953559.229004]: ['empty']	[INFO] [1642953565.047096]: ['empty']
[INFO] [1642953559.626110]: ['empty']	[INFO] [1642953565.449213]: ['empty']
[INFO] [1642953560.023367]: ['30_0']	[INFO] [1642953565.855008]: ['empty']
	[INFO] [1642953566.250834]: ['empty']

[INFO] [1642953566.652753]:
['empty']

[INFO] [1642953572.273058]:
['empty']

[INFO] [1642953567.054049]:
['empty']

[INFO] [1642953572.677883]:
['empty']

[INFO] [1642953567.456082]:
['empty']

[INFO] [1642953573.072864]:
['empty']

[INFO] [1642953567.858549]:
['empty']

[INFO] [1642953573.478521]:
['empty']

[INFO] [1642953568.257407]:
['empty']

[INFO] [1642953573.880329]:
['empty']

[INFO] [1642953568.664629]:
['empty']

[INFO] [1642953574.282660]:
['empty']

[INFO] [1642953569.065763]:
['empty']

[INFO] [1642953574.685649]:
['empty']

[INFO] [1642953569.472229]:
['empty']

[INFO] [1642953575.088791]:
['empty']

[INFO] [1642953569.865277]:
['empty']

[INFO] [1642953575.480084]:
['empty']

[INFO] [1642953570.266144]:
['empty']

[INFO] [1642953575.893932]:
['empty']

[INFO] [1642953570.670947]:
['empty']

[INFO] [1642953576.288193]:
['empty']

[INFO] [1642953571.273784]: ['30_1']

[INFO] [1642953576.692943]:
['empty']

[INFO] [1642953571.870519]:
['empty']

[INFO] [1642953577.084457]:
['empty']

[INFO] [1642953577.489776]:
['empty']

[INFO] [1642953578.088652]:
['empty']

[INFO] [1642953578.491319]:
['empty']

[INFO] [1642953578.891425]:
['empty']

[INFO] [1642953579.292986]:
['empty']

[INFO] [1642953579.695537]: ['30_0']

[INFO] [1642953580.095143]: ['30_0']

[INFO] [1642953580.499595]: ['30_0']

[INFO] [1642953580.906216]: ['30_0']

[INFO] [1642953581.306914]:
['empty']

[INFO] [1642953581.706912]:
['empty']

[INFO] [1642953582.108109]: ['30_0']

[INFO] [1642953582.510578]:
['empty']

[INFO] [1642953582.910551]: ['45_0']

[INFO] [1642953583.316903]: ['45_0']

[INFO] [1642953583.713302]:
['empty']

[INFO] [1642953584.108156]: ['45_0']

[INFO] [1642953584.507572]: ['45_1']

[INFO] [1642953584.915039]: ['30_0']

[INFO] [1642953585.316525]: ['30_0']

[INFO] [1642953585.719734]:
['empty']

[INFO] [1642953586.123712]: ['30_0']

[INFO] [1642953586.527432]: ['30_0']

[INFO] [1642953586.920010]: ['30_0']

[INFO] [1642953587.321356]:
['empty']

[INFO] [1642953587.725011]: ['30_0']

[INFO] [1642953588.128280]: ['30_0']

[INFO] [1642953588.537584]: ['30_0']

[INFO] [1642953588.929401]: ['30_1']

[INFO] [1642953589.331533]: ['empty']	[INFO] [1642953595.750774]: ['30_0']
[INFO] [1642953589.733545]: ['30_0']	[INFO] [1642953596.154564]: ['30_0']
[INFO] [1642953590.142549]: ['30_0']	[INFO] [1642953596.553262]: ['30_0']
[INFO] [1642953590.531131]: ['30_0']	[INFO] [1642953596.962960]: ['empty']
[INFO] [1642953590.934356]: ['empty']	[INFO] [1642953597.353999]: ['30_1']
[INFO] [1642953591.340661]: ['30_0']	[INFO] [1642953597.752643]: ['empty']
[INFO] [1642953591.734193]: ['empty']	[INFO] [1642953598.159772]: ['empty']
[INFO] [1642953592.135640]: ['45_0']	[INFO] [1642953598.559797]: ['30_0']
[INFO] [1642953592.539529]: ['45_0']	[INFO] [1642953598.960300]: ['30_1']
[INFO] [1642953592.938262]: ['30_0']	[INFO] [1642953599.362464]: ['30_0']
[INFO] [1642953593.351207]: ['30_0']	[INFO] [1642953599.764583]: ['30_1']
[INFO] [1642953593.749905]: ['30_1']	[INFO] [1642953600.170537]: ['30_1']
[INFO] [1642953594.147317]: ['45_1']	[INFO] [1642953600.578399]: ['empty']
[INFO] [1642953594.551534]: ['45_1']	[INFO] [1642953600.978019]: ['empty']
[INFO] [1642953594.940428]: ['30_0']	[INFO] [1642953601.368045]: ['empty']
[INFO] [1642953595.343575]: ['empty']	

[INFO] [1642953601.772789]:
['empty']

[INFO] [1642953606.796034]:
['empty']

[INFO] [1642953602.174362]:
['empty']

[INFO] [1642953607.190549]:
['empty']

[INFO] [1642953602.580748]:
['empty']

[INFO] [1642953607.594859]: ['30_0']

[INFO] [1642953602.974678]:
['empty']

[INFO] [1642953607.994820]: ['30_1']

[INFO] [1642953603.380365]:
['empty']

[INFO] [1642953608.598285]:
['empty']

[INFO] [1642953603.784453]:
['empty']

[INFO] [1642953608.999520]:
['empty']

[INFO] [1642953604.186585]:
['empty']

[INFO] [1642953609.398222]:
['empty']

[INFO] [1642953604.781452]:
['empty']

[INFO] [1642953609.799981]: ['30_r']

[INFO] [1642953605.191922]:
['empty']

[INFO] [1642953610.198872]:
['empty']

[INFO] [1642953605.586652]:
['empty']

[INFO] [1642953610.605134]:
['empty']

[INFO] [1642953605.992323]:
['empty']

[INFO] [1642953610.999268]:
['empty']

[INFO] [1642953606.385965]:
['empty']

[INFO] [1642953611.416818]:
['empty']

[INFO] [1642953611.799729]:
['empty']

[INFO] [1642953612.203753]:
['empty']

[INFO] [1642953612.603069]: ['30_0']

[INFO] [1642953613.015103]: ['30_0']

[INFO] [1642953613.407194]: ['30_0']

[INFO] [1642953613.807939]: ['30_0']

[INFO] [1642953614.215045]:
['empty']

[INFO] [1642953614.613460]:
['empty']

[INFO] [1642953615.016289]:
['empty']

[INFO] [1642953615.421223]: ['45_0']

[INFO] [1642953615.813643]: ['30_0']

[INFO] [1642953616.211739]: ['30_0']

[INFO] [1642953616.624444]:
['empty']

[INFO] [1642953617.015786]: ['30_0']

[INFO] [1642953617.428663]: ['45_0']

[INFO] [1642953617.831971]: ['30_0']

[INFO] [1642953618.229875]: ['30_1']

[INFO] [1642953618.631085]: ['30_0']

[INFO] [1642953619.034760]: ['45_0']

[INFO] [1642953619.440811]: ['30_0']

[INFO] [1642953619.835347]: ['30_0']

[INFO] [1642953620.238657]: ['30_1']

[INFO] [1642953620.639552]: ['30_0']

[INFO] [1642953621.033086]: ['30_0']

[INFO] [1642953621.443205]: ['30_0']

[INFO] [1642953621.846055]: ['45_1']

[INFO] [1642953622.241927]: ['30_0']

[INFO] [1642953622.641414]: ['30_0']

[INFO] [1642953623.048299]: ['45_0']

[INFO] [1642953623.451520]: ['30_0']

[INFO] [1642953623.845138]: ['45_0']

[INFO] [1642953624.248986]: ['45_0']

[INFO] [1642953624.651702]:
['empty']

[INFO] [1642953625.054634]:
['empty']

[INFO] [1642953625.458283]: ['30_r']	[INFO] [1642953632.259814]: ['30_0']
[INFO] [1642953625.847245]: ['45_0']	[INFO] [1642953632.671958]: ['45_0']
[INFO] [1642953626.253158]: ['45_0']	[INFO] [1642953633.076603]: ['30_1']
[INFO] [1642953626.651817]: ['empty']	[INFO] [1642953633.477542]: ['45_1']
[INFO] [1642953627.056899]: ['empty']	[INFO] [1642953633.878932]: ['45_0']
[INFO] [1642953627.462088]: ['empty']	[INFO] [1642953634.277876]: ['30_0']
[INFO] [1642953627.855441]: ['45_1']	[INFO] [1642953634.686706]: ['30_0']
[INFO] [1642953628.262381]: ['30_0']	[INFO] [1642953635.083765]: ['30_0']
[INFO] [1642953628.661761]: ['empty']	[INFO] [1642953635.483618]: ['empty']
[INFO] [1642953629.060795]: ['45_0']	[INFO] [1642953635.889946]: ['empty']
[INFO] [1642953629.462287]: ['45_0']	[INFO] [1642953636.287626]: ['empty']
[INFO] [1642953629.863306]: ['30_0']	[INFO] [1642953636.700083]: ['empty']
[INFO] [1642953630.300257]: ['30_0']	[INFO] [1642953637.090348]: ['empty']
[INFO] [1642953630.670257]: ['30_0']	[INFO] [1642953637.495524]: ['empty']
[INFO] [1642953631.365404]: ['45_1']	[INFO] [1642953637.897115]: ['empty']
[INFO] [1642953631.670219]: ['30_0']	[INFO] [1642953637.897115]: ['empty']

[INFO] [1642953638.297594]:
['empty']

[INFO] [1642953643.109822]:
['empty']

[INFO] [1642953638.697448]:
['empty']

[INFO] [1642953643.515405]:
['empty']

[INFO] [1642953639.098245]:
['empty']

[INFO] [1642953643.917052]: ['45_I']

[INFO] [1642953639.503880]:
['empty']

[INFO] [1642953644.315996]: ['45_I']

[INFO] [1642953644.717151]: ['45_I']

[INFO] [1642953639.905291]:
['empty']

[INFO] [1642953645.118833]:
['empty']

[INFO] [1642953640.306649]:
['empty']

[INFO] [1642953645.518560]:
['empty']

[INFO] [1642953640.706159]:
['empty']

[INFO] [1642953645.924247]:
['empty']

[INFO] [1642953641.107973]:
['empty']

[INFO] [1642953646.326186]:
['empty']

[INFO] [1642953641.509690]:
['empty']

[INFO] [1642953646.737695]:
['empty']

[INFO] [1642953641.910272]:
['empty']

[INFO] [1642953647.121864]:
['empty']

[INFO] [1642953642.311217]:
['empty']

[INFO] [1642953647.530418]:
['empty']

[INFO] [1642953642.709689]:
['empty']

[INFO] [1642953647.931121]:
['empty']

[INFO] [1642953648.331594]:
['empty']

[INFO] [1642953653.146339]:
['empty']

[INFO] [1642953648.731759]:
['empty']

[INFO] [1642953653.551320]:
['empty']

[INFO] [1642953649.134302]:
['empty']

[INFO] [1642953653.953614]:
['empty']

[INFO] [1642953649.541560]:
['empty']

[INFO] [1642953654.345508]:
['empty']

[INFO] [1642953649.932559]:
['empty']

[INFO] [1642953654.748293]:
['empty']

[INFO] [1642953650.343156]:
['empty']

[INFO] [1642953655.150762]:
['empty']

[INFO] [1642953650.733897]:
['empty']

[INFO] [1642953655.552734]:
['empty']

[INFO] [1642953651.137933]:
['empty']

[INFO] [1642953655.955627]:
['empty']

[INFO] [1642953651.548505]:
['empty']

[INFO] [1642953656.355032]:
['empty']

[INFO] [1642953651.940698]:
['empty']

[INFO] [1642953656.757495]:
['empty']

[INFO] [1642953652.344387]:
['empty']

[INFO] [1642953657.159368]:
['empty']

[INFO] [1642953652.745421]:
['empty']

[INFO] [1642953657.558897]:
['empty']

[INFO] [1642953657.963133]:
['empty']

[INFO] [1642953662.778664]:
['empty']

[INFO] [1642953658.365857]:
['empty']

[INFO] [1642953663.188914]:
['empty']

[INFO] [1642953658.762437]:
['empty']

[INFO] [1642953663.582362]:
['empty']

[INFO] [1642953659.162529]:
['empty']

[INFO] [1642953663.986812]:
['empty']

[INFO] [1642953659.564647]:
['empty']

[INFO] [1642953664.380198]:
['empty']

[INFO] [1642953659.970438]:
['empty']

[INFO] [1642953664.788158]:
['empty']

[INFO] [1642953660.378144]:
['empty']

[INFO] [1642953665.190698]:
['empty']

[INFO] [1642953660.766081]:
['empty']

[INFO] [1642953665.592294]:
['empty']

[INFO] [1642953661.179862]:
['empty']

[INFO] [1642953665.983660]:
['empty']

[INFO] [1642953661.575440]:
['empty']

[INFO] [1642953666.385730]:
['empty']

[INFO] [1642953661.980025]:
['empty']

[INFO] [1642953666.786463]:
['empty']

[INFO] [1642953662.374738]:
['empty']

[INFO] [1642953667.188346]:
['empty']

[INFO] [1642953667.593495]:
['empty']

[INFO] [1642953673.408168]:
['empty']

[INFO] [1642953667.994421]:
['empty']

[INFO] [1642953673.807776]: ['30_r']

[INFO] [1642953668.395565]:
['empty']

[INFO] [1642953674.212052]:
['empty']

[INFO] [1642953668.795271]:
['empty']

[INFO] [1642953674.612395]:
['empty']

[INFO] [1642953669.198746]:
['empty']

[INFO] [1642953675.019092]:
['empty']

[INFO] [1642953669.798445]:
['empty']

[INFO] [1642953675.428947]:
['empty']

[INFO] [1642953670.197804]:
['empty']

[INFO] [1642953675.819766]:
['empty']

[INFO] [1642953670.599986]:
['empty']

[INFO] [1642953676.225618]:
['empty']

[INFO] [1642953671.005720]:
['empty']

[INFO] [1642953676.637113]:
['empty']

[INFO] [1642953671.605951]: ['45_0']

[INFO] [1642953677.119286]: ['45_r']

[INFO] [1642953672.202661]: ['30_r']

[INFO] [1642953677.427298]: ['45_r']

[INFO] [1642953672.600869]:
['empty']

[INFO] [1642953677.828251]: ['45_0']

[INFO] [1642953673.005549]: ['45_r']

[INFO] [1642953678.231918]: ['30_0']

[INFO] [1642953678.633745]: ['30_0']

[INFO] [1642953679.032509]: ['30_0'] [INFO] [1642953684.645333]:
['empty']

[INFO] [1642953679.433663]: [INFO] [1642953685.046669]: ['30_0']
['empty']

[INFO] [1642953679.836375]: ['30_0'] [INFO] [1642953685.451310]: ['30_0']

[INFO] [1642953680.239573]: [INFO] [1642953685.858306]: ['30_1']
['empty']

[INFO] [1642953680.639352]: [INFO] [1642953686.260522]:
['empty']

[INFO] [1642953681.044737]: [INFO] [1642953686.667441]: ['45_1']
['empty']

[INFO] [1642953681.439189]: [INFO] [1642953687.088836]: ['45_1']
['empty']

[INFO] [1642953681.843431]: [INFO] [1642953687.462513]:
['empty']

[INFO] [1642953682.240327]: [INFO] [1642953687.865500]:
['empty']

[INFO] [1642953682.645787]: ['45_0'] [INFO] [1642953688.268236]:
['empty']

[INFO] [1642953683.041263]: ['45_0'] [INFO] [1642953688.662078]: ['45_0']

[INFO] [1642953683.443971]: [INFO] [1642953689.064045]:
['empty']

[INFO] [1642953683.843772]: ['15_0'] [INFO] [1642953689.465552]: ['45_0']

[INFO] [1642953684.253541]: ['30_0'] [INFO] [1642953689.871842]: ['45_0']

[INFO] [1642953690.277776]: ['45_0']

[INFO] [1642953690.880647]: ['45_0'] [INFO] [1642953697.288516]: ['45_0']

[INFO] [1642953691.283049]: ['45_0'] [INFO] [1642953697.689166]:
['empty']

[INFO] [1642953691.676088]:
['empty'] [INFO] [1642953698.093756]:
['empty']

[INFO] [1642953692.074961]: ['45_0'] [INFO] [1642953698.496048]: ['45_1']

[INFO] [1642953692.480097]: ['45_0'] [INFO] [1642953698.900370]: ['30_1']

[INFO] [1642953692.882500]: ['45_0'] [INFO] [1642953699.303627]:
['empty']

[INFO] [1642953693.281978]: ['30_r'] [INFO] [1642953699.695531]:
['empty']

[INFO] [1642953693.684011]: ['30_0'] [INFO] [1642953700.106892]:
['empty']

[INFO] [1642953694.089066]: ['45_0'] [INFO] [1642953700.506465]:
['empty']

[INFO] [1642953694.485060]: ['45_0'] [INFO] [1642953700.907635]: ['30_0']

[INFO] [1642953694.886762]: ['30_r'] [INFO] [1642953701.310792]:
['empty']

[INFO] [1642953695.287479]: ['30_0'] [INFO] [1642953701.704305]: ['30_0']

[INFO] [1642953695.687726]:
['empty'] [INFO] [1642953702.107128]:
['empty']

[INFO] [1642953696.090710]:
['empty'] [INFO] [1642953702.511144]: ['45_0']

[INFO] [1642953696.484884]:
['empty']

[INFO] [1642953696.897608]: ['30_1']

[INFO] [1642953702.916605]: ['30_r']	[INFO] [1642953708.927808]: ['empty']
[INFO] [1642953703.315728]: ['30_r']	[INFO] [1642953709.325940]: ['empty']
[INFO] [1642953703.718583]: ['30_r']	[INFO] [1642953709.731487]: ['empty']
[INFO] [1642953704.118480]: ['30_r']	[INFO] [1642953710.133468]: ['30_r']
[INFO] [1642953704.518033]: ['30_r']	[INFO] [1642953710.534366]: ['30_r']
[INFO] [1642953704.927045]: ['30_r']	[INFO] [1642953710.931230]: ['empty']
[INFO] [1642953705.319730]: ['30_r']	[INFO] [1642953711.332357]: ['empty']
[INFO] [1642953705.720022]: ['30_0']	[INFO] [1642953711.742094]: ['empty']
[INFO] [1642953706.123478]: ['30_r']	[INFO] [1642953712.142359]: ['30_r']
[INFO] [1642953706.522969]: ['empty']	[INFO] [1642953712.538714]: ['empty']
[INFO] [1642953706.923431]: ['empty']	[INFO] [1642953712.943937]: ['empty']
[INFO] [1642953707.324654]: ['empty']	[INFO] [1642953713.342929]: ['empty']
[INFO] [1642953707.723188]: ['empty']	[INFO] [1642953713.752837]: ['empty']
[INFO] [1642953708.124825]: ['empty']	[INFO] [1642953714.145505]: ['30_r']
[INFO] [1642953708.526503]: ['empty']	

[INFO] [1642953714.542381]:
['empty']

[INFO] [1642953720.564311]:
['empty']

[INFO] [1642953714.950304]:
['empty']

[INFO] [1642953720.972049]: ['45_l']

[INFO] [1642953715.352239]:
['empty']

[INFO] [1642953721.377114]: ['30_l']

[INFO] [1642953715.748036]:
['empty']

[INFO] [1642953721.769679]:
['empty']

[INFO] [1642953716.153030]:
['empty']

[INFO] [1642953716.551963]:
['empty']

[INFO] [1642953716.947040]:
['empty']

[INFO] [1642953717.356724]:
['empty']

[INFO] [1642953717.760462]:
['empty']

[INFO] [1642953718.156742]: ['30_0']

[INFO] [1642953718.959161]:
['empty']

[INFO] [1642953719.370703]: ['30_r']

[INFO] [1642953719.963460]:
['empty']