ID08

INVESTIGATION OF ROS BASED VEHICLE STATE ESTIMATION WITH UNCERTAINTIES

BACHELOR OF ELECTRICAL ENGINEERING (ELECTRONICS) WITH HONOURS

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : SITI NURAINI BINTI CHE HUHAIMI

Date of Birth : 06/10/1999

Title : INVESTIGATION OF ROS BASED VEHICLE STATE ESTIMATION WITH UNCERTAINTIES

Academic Session : SEMESTER II 2021/2022

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:

Aini

(Student's Signature)

991006035854
New IC/Passport Number
Date: 26.6.2022

(Supervisor's Signature)

Assoc.Prof.Ts.Dr.Hamzah Ahmad
Name of Supervisor
Date: 26.6.2022

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
Thesis Title

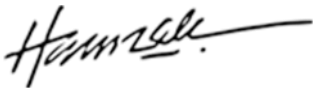
Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,



(Supervisor's Signature)

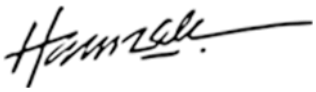
Date: 26.6.2022

Stamp:
DR. HAMZAH BIN AHMAD
ASSOCIATE PROFESSOR
FACULTY OF ELECTRICAL & ELECTRONICS ENGINEERING
UNIVERSITI MALAYSIA PAHANG
26600 PEKAN
PAHANG DARUL MAKMUR
TEL : 09-424 6024 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.

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 Bachelor of Electrical Engineering (Electronics) with Honours.



_____

(Supervisor's Signature)

Full Name : Prof. Madya Dr. Hamzah Bin Ahmad

Position : Senior lecturer

Date : 26.6.2022

_____

(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 University Malaysia Pahang or any other institutions.

Aini

_____

(Student's Signature)

Full Name : Siti Nuraini Binti Che Huhaimi

ID Number : EA18120

Date : 26.6.2022

INVESTIGATION OF ROS BASED VEHICLE STATE ESTIMATION WITH UNCERTAINTIES

SITI NURAINI BINTI CHE HUHAIMI

Thesis submitted in fulfillment of the requirements
for the award of the
Bachelor of Electrical Engineering (Electronics) with Honours

College of Engineering

UNIVERSITI MALAYSIA PAHANG

JUNE 2022

ACKNOWLEDGEMENTS

In the Name of Allah, the Merciful, the Compassionate.

First and foremost, I want to express my heartful appreciation to Prof Madya Dr. Hamzah bin Ahmad, my beloved supervisor, for providing me with invaluable assistance and regular suggestions as a supervisor during this project. Working and studying under his direction was a wonderful honors and privilege. From the first day I met him till now, I appreciate his unwavering support. The route to complete the Final Year Project will be more difficult without his help.

My family, especially my parents, deserve thanks for their love, support, prayers, and sacrifices throughout my life. I owe my parents a debt of gratitude for their patience and constant care for me. Thank you for assisting me in surviving all the stress and for not allowing me to give up. Also, thank you to my siblings for constantly brightening and delighting my days.

My profound gratitude also goes to my friends for assisting me in finishing the thesis when I became overwhelmed and confused, as well as for their unwavering support.

Finally, I would want to express my gratitude to everyone who was directly or indirectly involved in the completion of my Final Year Project, I much appreciate your efforts. Thank you very much.

ABSTRAK

Tesis ini dicadangkan untuk mengkaji anggaran keadaan kenderaan berasaskan ROS (Sistem pengendalian Robot) dengan mengambil kira ketidakpastian. ROS ialah perisian tengah antara perisian dan perkakasan. Ia juga berurusan dengan “Kalman Filter” (KF) melalui kajian kes yang berbeza di mana pendekatan ini bertujuan untuk memberikan anggaran yang lebih baik untuk robot mudah alih. Berdasarkan kajian kes yang berbeza ROS memainkan peranan penting untuk menilai prestasi robot mudah alih merujuk kepada persekitarannya. ROS menyediakan beberapa pakej mudah dengan nod ROS dan algoritma SLAM yang menjadikan masalah pembentukan lebih mudah untuk diselesaikan. Penderia pengesanan dan julat cahaya (LiDAR) digunakan untuk memberi maklumat untuk analisis ini bagi mendapatkan maklumat daripada persekitaran yang membantu turtlebot3 mengelak halangan. Sensor LiDAR 360 darjah juga dilengkapi dalam burger turtlebot3 yang digunakan dalam persediaan makmal. Simulasi itu juga melibatkan simulator 3D gazebo dan pemetaan dalam visualisasi Rviz untuk penjanaan peta. Selain itu, algoritma SLAM digunakan untuk memberi gambaran robot untuk dipetakan secara serentak semasa mengesan dirinya. Keputusan awal eksperimen ini menggambarkan bahawa Burger Turtlebot3 dapat mengelak halangan dalam perjalanan dengan berkesan dari mana-mana titik permulaan sehingga burger Turtlebot3 boleh sampai ke destinasi dengan selamat dan dapat mengelakkan sebarang jenis halangan.

ABSTRACT

This thesis proposed to examine ROS (Robot operating system) based vehicle state estimation considering uncertainties. ROS is a middleware between software and hardware. It also deals with the Kalman filter (KF) via different case studies where This approach is aimed to provide better estimation for mobile robots. Based on different case study ROS play an important role to evaluate mobile robot performance referring to its environment. ROS provides some convenient packages with ROS nodes and the SLAM algorithms that make the formation problem easier to solve. The light detection and ranging (LiDAR) sensor as obstacle detection are used to give information for this analysis in order to gain information from the surrounding where it helps to avoid obstacles. The 360-degree LiDAR sensor also was equipped in the turtlebot3 burger that was used in the laboratory setup. The simulation also involved a gazebo 3D simulator and mapping in Rviz visualization for map generation. Besides, the SLAM algorithm is used to give a picture of the robot to simultaneously map while locating itself. The preliminary result of this experiment illustrates that the Turtlebot3 Burger can avoid the obstacle on its way effectively from any starting point until that Turtlebot3 burger can reach its destination safely and it is expected the turtlebot3 burger can avoid any type of obstacle.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

ROS	Robot Operating System
OSRF	Open-Source Robotics Foundation
KF	Kalman Filter
SLAM	Simultaneous localization and mapping
LiDAR	Light Detection and Ranging
LDS	Laser Distance Sensor
AI	Artificial Intelligence

CHAPTER 1

INTRODUCTION

1.1 Project Background

Nowadays, with the innovation of artificial intelligence technology, the varieties of intelligent robotics technology have become growingly widely available, and their autonomy has been continuously enhanced, from the first market floor cleaning robot and delivery service robot, escorting robots, education robot, rehabilitation robot, grocery store robot, and other direction advancement, service areas, and service objects. The majority of service robots are mobile because they can support humans or machines. The mobility of the robots enables them to perform the tasks for which they are intended without difficulty.

A mobile robot is a machine that is driven by software and detects and moves around its environment using sensors and other technology. Mobile robots work by mixing AI and physical robotic parts such as wheels, tracks, and legs. A mobile robot also is an autonomous agent capable of navigating intelligently anywhere. One of the most important roles of mobile robots is autonomous navigation. The control system is the major element of a mobile robot since it controls the performance of movements such as accuracy, stability, and ability. Autonomous navigation involves localization, path planning, and motion control of the mobile platform. Autonomous vehicles have seemed to be the next big thing in the vehicle industry.

The fast development of self-driving or autonomous technology necessitates new strategies for slowing down the rate of accidents. The most challenging aspect of autonomous vehicles is the potential to react accurately and safely to potential hazards while driving. As a necessity, autonomous vehicles must be designed with sensors that can respond immediately and a processing unit that can understand all this data in real-time.

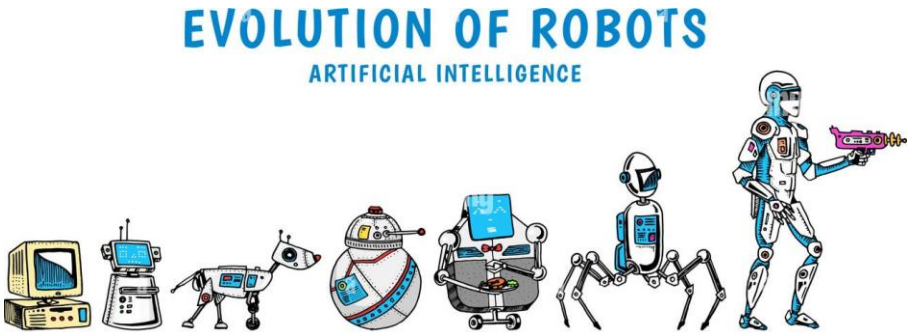


Figure 1 Evolution of Robot

1.2 Problem Statement

Mobile robot nowadays is worldwide application to perform many tasks. Therefore, its capability in detecting and then avoiding from any obstacle is one of the limitations. Then, it is an issue when many sensors have a very limited coverage, detection distance and its complexity. Thus, it provides limits for the mobile robot to predict collision-free paths automatically in various condition. Next, by using this sensor, mobile robot then needs to be able to identify its exact position. This issue can be solved by implementing a good estimation to ensure mobile robot obtain better performance.

A mobile robot should be effectively navigating to any location, however, without the perfect system it may cause a problem. Thus, a suitable design with a good system can reduce hazard for both human and the robot. This would reduce the losses cost of maintenance for mobile robot. Thus, a mobile robot without the obstacle avoidance technique is unsafe for both human and the robot itself. Which may lead to a critical accident that may cause major losses.

1.3 Objective

The aim of this project is:

1) To design the navigation of mobile robot with obstacle via ROS in different environment.

2) To develop the state estimation for better performance of turtlebot3 in MATLAB simulation.

1.4 Scope of Work

There are several project scopes that covers most of the project objectives to accomplish this project. It includes:

1) Set up the ROS Version based on Ubuntu Linux that provided in WikiRos.org websites in the PC. ROS systems are used to design the navigation of turtlebot3 burger to move from one point to another point in different environment

2) The turtlebot3 burger able to avoid the obstacles on the flat surface, from any starting point by using the obstacle avoidance behaviour.

3) The solving method used in this project is using a Kalman filter to decrease the influences or measurement noise on the final error level reached to quickly estimate the true value, position, velocity of the turtlebot3 in simulation via MATLAB and ROS (Robot Operating System) with the using of Turtlebot3 Burger in real time experiment to analyse the turtlebot3's movement in different condition.

1.5 Thesis Outline

This thesis has five chapters which include Introduction, Literature Review, Methodology, Result and Discussion, and Conclusion.

Chapter 1 is about the introduction of the research. In this chapter will covered on the project background, problem statement, objective, and the scope of this project.

Chapter 2 is on literature review of the research topic. This chapter explored our topic of focus by comparing it to what other researchers had mentioned in their study publications. The results of this discussion will be utilised as the framework for our project.

Chapter 3 is about the methodology for this project. In this chapter, the components and tools used will be discussed in detail with the implementation process of the turtlebot3 navigation.

Chapter 4 discussed on the result, finding and outcome of the navigation system. The data of the navigation system are collected and recorded before discussion being made about the system performance.

Chapter 5 is on conclusion of the project. In this chapter, the limitation of the system will be discussed. Further improvement will then be proposed for the upcoming research related to this topic.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter briefly discusses the literature review and conclusions from earlier studies that other researchers conducted that related to my topic. This research will be more focusing on ROS (Robot Operating System), an implementation of Kalman filter in mobile robot estimation, SLAM (Simultaneous Localization and Mapping) Algorithm, System Design based on ROS, Lidar Sensor, and others. All these elements will be utilized in my project that needs to develop the system via ROS to analyse the mobile robot movements.

2.2 Flow Of Literature Review

2.2.1 Robot Interface Using ROS

(Péter Fankhauser 2017) stated in his paper that ROS (Robot Operating System) was originally developed in 2007 at the Stanford Artificial Intelligent Laboratory then after 2008 it was promoted and maintained by Willow Garage. Since 2013, ROS were managed and supported by the OSRF (Open-Source Robotics Foundation). In Open Robotics, they work with industry, academics, and government to create and support open software and hardware for use in robotics, from research and education to product development.

ROS is not a real operating system it is more like an environment. The big strength of ROS is the ability to connect nodes together and nodes are the pieces of software that can be written in C++, Python language, and others. In (Li and Shi 2018) research, they stated that ROS features point-to-point design, can support multiply programming languages, simplification, and integration is easy to test, application for large-scale program development, and for open-source code which greatly shortens the development cycle of robot software and reduces the development cost of robots. ROS can take care of a small subset of tasks for example reading a sensor or controlling a servo, and those nodes connect together thanks to a publisher or subscriber protocol what does it mean, mean that if a node has a piece of information to share it will share this information using a topic and if another node is interested in that information, its subscribers to that topic and reads the information. (Yu 2019) in their research paper stated that instead of a single ROS process, numerous ROS processes are running in a robot. These ROS processes are referred to as nodes. ROS communicates messages between nodes via a network. Because ROS allows several robots to collaborate, a centralized PC is used to communicate between nodes on the same or separate machines. The nodes are classified into three types which are master, publisher, and subscriber. Figure 2 and figure 3 below depicts an example of message communication between ROS nodes and schematic diagram of ROS messages delivery respectively.

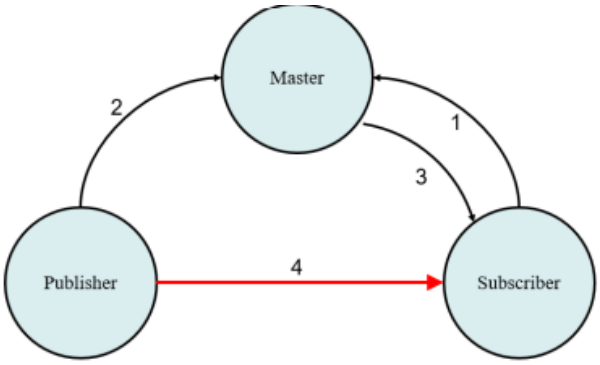


Figure 2 Message Communication between ROS nodes

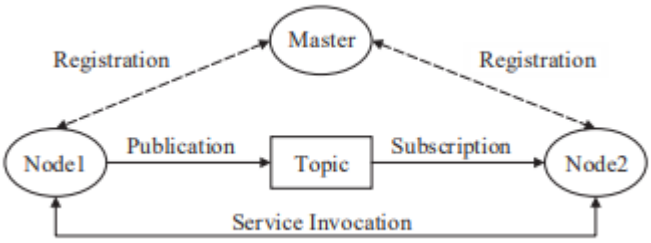


Figure 3 Schematic diagram of ROS message delivery

In the research of (Thale 2020), the author stated that Instead of developing the entire system in hardware, ROS is used to construct a virtual environment, produce a robot model, execute the algorithms, and view it in the virtual world. Gazebo and RVIZ are used to create the virtual environment.

ROS also includes several established feature packages that may be utilised directly, such as the cartography software package for robotic positioning and mapping, as well as navigation kit includes for autonomous navigation and obstacle avoidance. In the paper of (Takaya 2016, October) the author stated that the advantage of using ROS is ROS allows effective development of the new robotic system and when used together with a simulation middleware like Gazebo (co-Simulink of ROS) it produces reliable development and high performance. The Gazebo has high-quality graphics, programmatic and graphical interfaces. Now there are multiple versions of ROS, there is ROS 1 and ROS 2 and a lot of the distribution of ROS will be using a kinetic. In my project, I will use ROS 2 because it is the latest version and has more advanced functions compared to ROS 1, which ROS 1 does not have. The goal of the ROS 2 project is to adapt to these changes by leveraging what functions well in ROS 1 and improving what doesn't.

2.2.2 An Implementation of Kalman Filter

The robot localization problem is critical in the development of totally autonomous robots. It might be difficult to identify what to do next when a robot does not know where it is. A robot can localise itself using relative and absolute measurements that provide input on its driving actions and the state of the environment around it. The robot must identify its location as precisely as possible using this information. What makes this difficult is the existence of uncertainty in both the driving and the sensing of the robot. Thus, the Kalman filter will be used to estimate the system mobile robot state. The Kalman filter has been widely used for mobile robot navigation and system integration. Kalman filter is the iterative mathematical process that uses a set of equations and consecutive data inputs to quickly estimate the true value, position, velocity of any object or mobile robot that is being measured when the measured value contains unpredicted or variation or random error that is also known as uncertainty. Kalman filter proved to have a smarter way to integrate measurement data into an estimate by recognizing that measurements are noisy and that sometimes they should be ignored or have only a small effect on the state estimate which has been stated in the research of (Suliman 2009)**.** It smooths out the effects of noise in the state variable being estimated by incorporating more information from reliable data than from unreliable data. In that research paper, they also proved the result that they obtained by simulating the Kalman filter case that they were implementing in MATLAB. They show a very simple case where a robot follows a path obtained from the system model and figure 4 below presents the estimated path of the mobile robot compared with the real path. The Kalman filter performed very well and as the result, the estimates path with the help of the Kalman filter is closer to the real path.

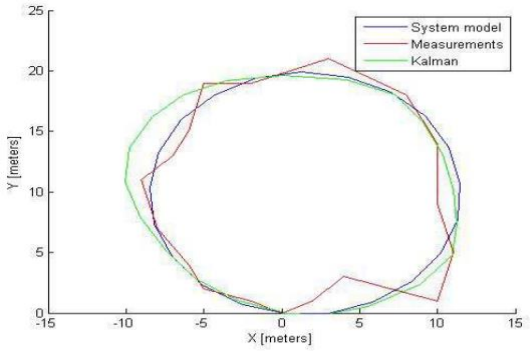


Figure 4 Trajectory estimation with the Kalman Filter

2.2.3 Simultaneous Localization and Mapping (SLAM)

Navigation and control systems are the most discussed topics in autonomous mobile robots. For the mobile robot, the control system is the most major part, which can decide the performance of the movement. On the flip side, autonomous navigation is one of the crucial functions of mobile robots. SLAM (Simultaneous Localization and Mapping) is a rudimentary part of the navigation, which is used to construct the map of the environment for the mobile robot. In the paper of (Meng 2020, September) the author stated that SLAM is mainly used to solve the problem of localization and mapping when the robot moves in an unknown environment. By using SLAM robot can understood to determine its position and posture in an uncertain environment by scanning the surroundings with its sensors.

(Zhi 2018) had investigated the performance of navigation and control systems of mobile robots based on ROS. In their paper which is using the G-mapping package to construct the map with the data of laser scan, encoder, and IMU sensor.

First, in their result based on laser radar SLAM based on ROS, they claimed that autonomous mobile technology can produce mobile robots to become more intelligent. In the unmapped environment, the robot will move from an unknown position. From the action of moving, the robot will be positioned according to the position estimation and map. Simultaneously, an incremental map is constructed based on its own positioning, which can be used to notify the autonomous localization and navigation of the robot. Sensors like laser-scanner, sonar sensors, and cameras were used most for constructing the map of the environment. They use laser scans to obtain the message of the environment or surroundings. The function of laser scan is to detect the location of the environment or obstacles by transmitting laser beams. They claimed that SLAM used the data of the scanning laser. Hence for the result, they get incremental map data of their place in figure 5 below.

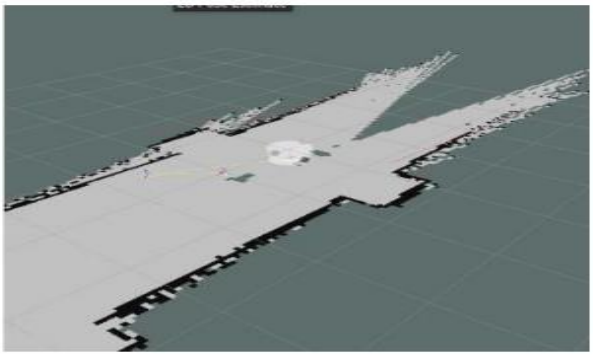


Figure 5 Incremental Map

2.2.4 System Design Based on ROS and Lidar Sensor

A laser radar sensor estimates distance without making contact. When a laser beam is reflected onto a target, it is processed to generate an environmental image. Traditional laser sensor systems include a distance sensor that measures a feature point, a two-dimensional scanning plane of lidar, and a three-dimensional lidar [2,3]. (Zhu 2019) in their paper research used the two-dimensional laser radar RPLIDAR A2 that was developed by Siwei Technology Co. Ltd, which uses laser triangulation technology, is equipped with RPVision 2.0 high-speed visual ranging engine, and can rotate clockwise to 8000 times per second within a radius of 18m in the surrounding environment 360degree detection to obtain an environmental image or information. In their system software design, they use ROS which includes the Cartographer software package for robotic positioning and obstacle avoidance. Cartographer is an open-source SLAM library that contains the ROS interface which was developed by Google. In the Cartographer's ROS framework there are cartogrpaher_node which is the main node where /scan is the data obtained by the lidar scanning and /imu is the data released from IMU.

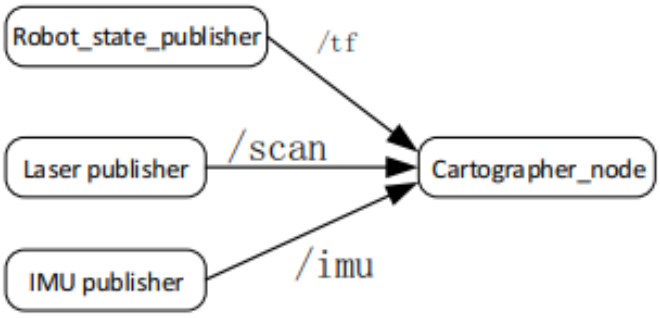


Figure 6 Cartographer' ROS framework

For closed-loop optimization, the cartography algorithm employs another technique known as Branch-and-bound scan matching. In their research they claimed that using the brand and bound can optimize the search, improve efficiency, and achieve real-time loopback. In that paper, they prove that their experimental result is successful as based on the identified obstacle information, the robot will calculate a possible path and avoid the obstacles as shown in figure 7 below.

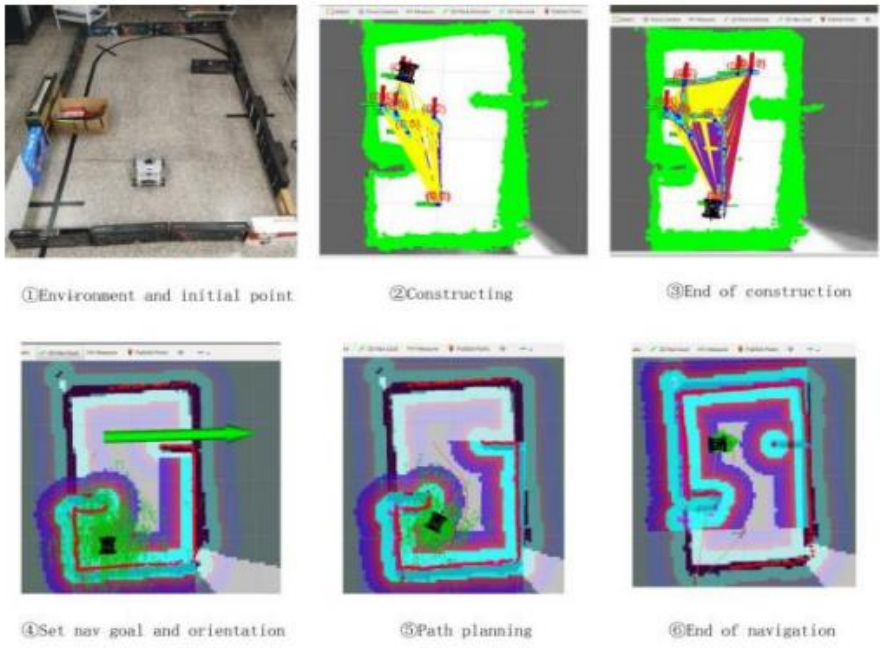


Figure 7 Result Experimental

2.2.5 Mobile Robot Localization Using the Kalman Filter and a Single fixed camera (capable of detect the landmarks)

A mobile robot is vulnerable by its own ability to move all on sides (from all directions or everywhere). The most important and basic functionalities are needed to know where exactly that mobile robot moves. Poor localization will cause the vehicle or robot to hit a wall or nosedive over a drop. Global localization or finding position relative to some non- robot reference frame is essential for any strategy that requires movement to a specific location even if the vehicle or robot can avoid these obstacles and others in the local area around that robot. Global position estimation is "the ability to determine the robot's position in an a priori or previously learned map, given no other information than that the robot is somewhere on the map. Once a robot has been localized in the map. Local tracking is the problem of keeping track of that position over time" [Dalleart 1999 (ICRA)]. Kalman filters have been used for navigation, tracking, and many other applications because compared to other filtering techniques that require less processing power to implement, Kalman filters provide better performance at a lower computational cost which has been stated in the research (Burnett 2006). The Kalman filter also estimates the future state of a system using its internal transition model. In addition, the filter replicates a measurement in that future condition. It updates the state based on the difference between the predicted and actual measurements.

2.2.6 Lidar-based Obstacle Avoidance for the Autonomous Mobile Robot

In some dangerous sites, such as chemical industries, polluted environments, and mining areas that can put the worker and residents at the risk of harm around them. These will have an impact on both monetary and human losses. Therefore, the usage of robots can be the solution to these issues. Nowadays, mostly all autonomous mobile robots were equipped with several sensors that can investigate in a room or in an open area, as well as in industrial fields, and also were equipped with obstacles avoidance features as one type of intelligent robot. (Hutabarat 2019, July) in his paper stated that the obstacle avoidance behaviour ensures and guarantees that the robot never collides with an obstacle in its way. In his research, he uses light detection and ranging (Lidar) sensors. As we know, Lidar technology is widely used as an obstacle avoidance system on autonomous mobile robots. Lidar has been developed to avoid obstacles and was equipped with a motor so that it can rotate 360-degree as we know to provide a view of the environment and it can measure and find distance and other information from target so that the robot can avoid the obstacle. Figure 8 and figure 9 below show the experiments by this author which is the mobile robot navigation in a room without obstacle and with obstacle respectively.

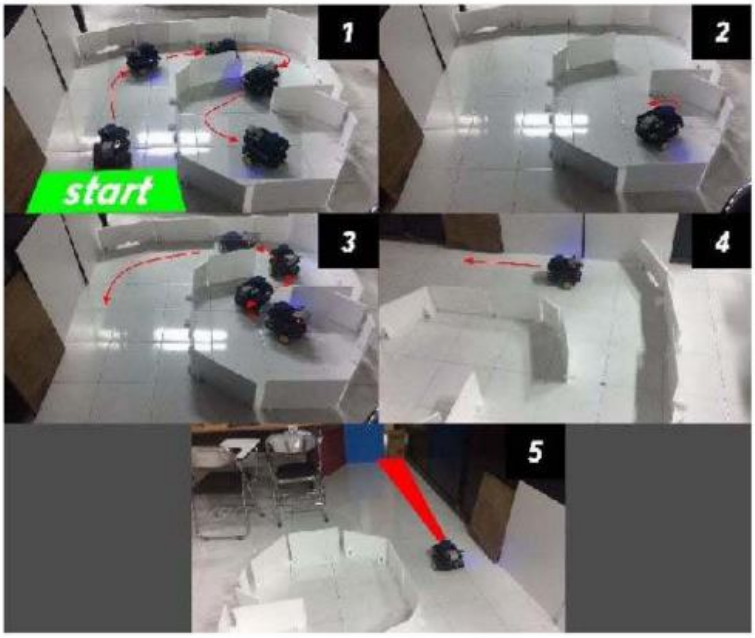


Figure 8 The mobile robot navigation in a room without obstacle

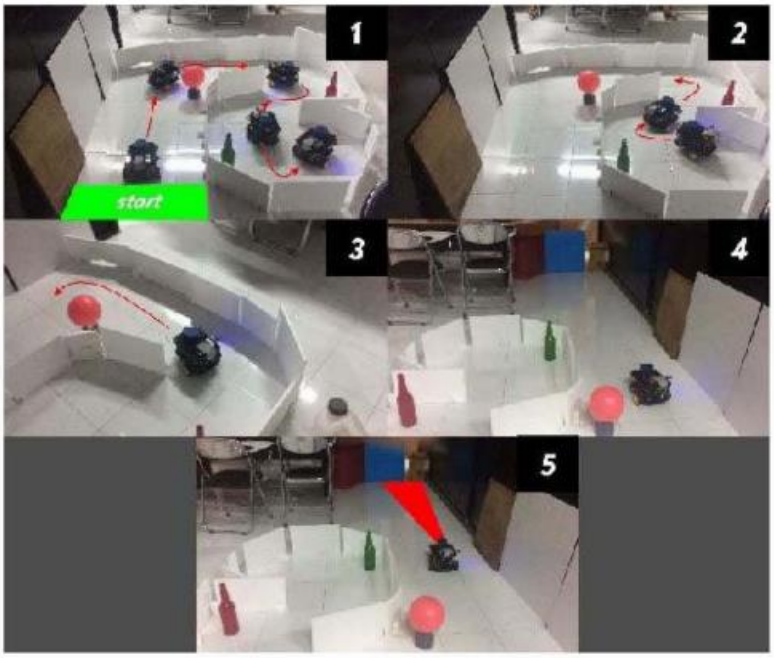


Figure 9 The mobile robot navigation in a room with obstacle

2.2.7 Autonomous Obstacle Avoidance Using Lidar

From the paper of (Nikolaos Baras, May 2019), the author stated that self-driving cars appear to be the next big thing in the automobile business. The constant advancement of self-driving technology necessitates new strategies for reducing the frequency of accidents. One of the most difficult aspects of autonomous cars is the potential to react accurately and safely to potential hazards while driving. As a result, these vehicles must be equipped with sensors that can respond fast and a processing unit that can analyse all this data in real time. In his paper the author used the LiDAR sensor among the other sensors. He said the LIDAR carries a high implementation complexity that makes it particularly suitable for autonomous driving. The fundamental technology for achieving safety in Autonomous Vehicles is Obstacle Avoidance (OA). In his experiment, the author considers two separate Obstacle Avoidance methods which are first based on the already known environment (offline processing algorithms) and the second based on the gathered sensors information in real time. For the experiment, the two main hardware components in his research in his implementation for an autonomous vehicle are Raspberry Pi and RPLidar A2. In his experiments, the author setup with the environment with the multiple obstacles of various sizes located in random places. The vehicle managed to navigate from the starting point (SP) to the goal point (GP) in 5 minutes and 31 second.

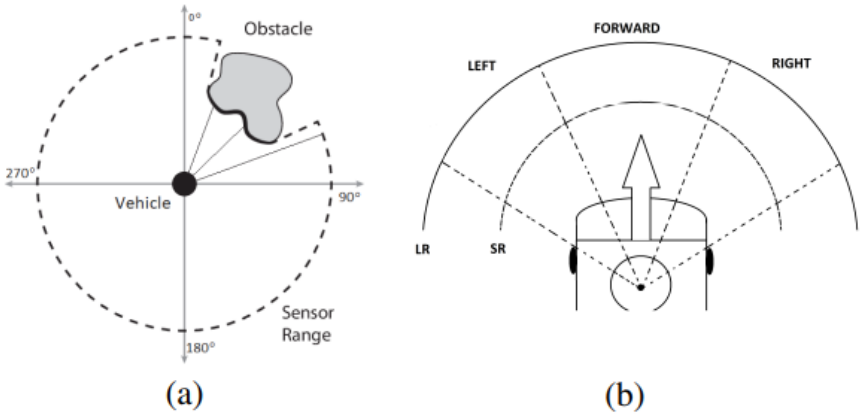


Figure 10 In (a) LiDAR detecting an obstacle. In (b) collected data based on measurement location

2.3 Summary

In this part we can conclude that from the research review stated that the Kalman Filter is a technique from estimation theory that by using the Kalman filter the output not too far from the measurement of the system model. The filter has been successfully applied in wide range of applications. The Kalman Filters estimates the state of a noisy system using noisy measurements. The KF makes a few assumptions on the system, measurements and different noises that are involved in the estimation problem. the Kalman Filters calculates the belief by performing two alternating steps which are prediction step and update step. From the research paper also stated the most autonomous mobile robot nowadays has been developed were equipped with Light Detection and Ranging (LiDAR) sensor to avoid obstacle. Many kinds of sensor also can be implemented on a mobile robot to determine its environment conditions. The sensors commonly used other than LiDAR sensor are camera, ultrasonic and others.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter has shown the procedure for this project. it will be explained by using the method and technique that was proposed to accomplish the project's objective and the hardware setup. As stated in chapter 1, the objectives of this project are to design the navigation of mobile robot with uncertainties (obstacle) via ROS in different environment and to develop the state estimation for better performance of turtlebot3 in MATLAB simulation. In general, Robot Operating System (ROS) software was being used for the simulation and hardware also Gazebo is a co-Simulink of ROS that has a high-quality graphics, programmatic and graphical interfaces that provide 3D dynamic simulator. For the mobile robot in our project, we used the Turtlebot3 Burger as it is equipped with LiDAR that can rotate 360-degree to give a view of the environment. Simultaneous Localization and Mapping (SLAM), G-Mapping, and the control node which is python will be implemented in our project. SLAM is a method used for autonomous vehicles that give us to create a map and localize our vehicle on that map at the same time which will be discussed in this chapter.

Below is the flowchart for the technique that will be used in this project to build the simulation environment, starting with ROS and Gazebo as the operating system and 3D simulation environment. In Gazebo, a Turtlebot3 Burger will be created in the virtual world as the mobile robot in this project. Then, the SLAM will be implemented to provide mapping in that virtual world. The Turtlebot3 will move and scan every corner in the virtual world or the real room and create the map using g-mapping. Once the map generation is complete, the map will be saved for further progress in this project. The

main Turlebot3 will be programmed to avoid any obstacle detected by its sensor. All details will be discussed in this chapter.

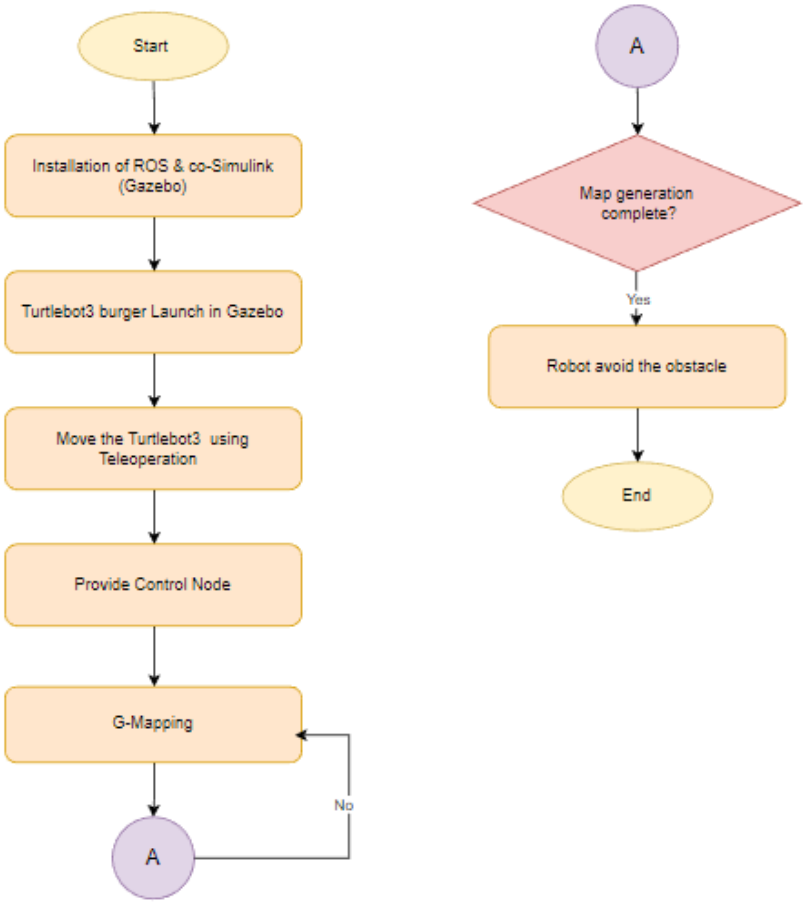


Figure 11 Progression Flowchart for Simulation

Next, for the hardware part, the first important procedure will start with the Personal Computer (PC) setup as the ROS master and follow with Single-Board Computer (SBC) setup for raspberry pi. After the SBC setup, then the OpenCR setup (Arm Cortex-M7). The PC setup will start by installing the correct Ubuntu and ROS versions for example the Ubuntu OS 16.04 for ROS Kinetic, Ubuntu 18.04 for ROS Melodic, Ubuntu 20.04 for ROS Noetic, and more types of ROS. In our project, we will use ROS Kinetic and ROS Noetic. Then all dependent ROS package and turtlebot3 burger package is installed. We can refer to all the steps from PC setup until Bringup the turtlebot3 burger on ROBOTIS e-Manual website. In the step of PC setup, it will have the network configuration which is to connect the PC to a Wi-Fi device and find the assigned IP address of ROS_MASTER_URI which is the Ip address of the PC itself, and IP address ROS_HOSTNAME which is the Ip address of the Raspberry Pi with the command ifconfig. In the SBC setup, we need to download Turtlebot3 SBC Image and install the Raspberry Pi Imager to burn the image file to a microSD card that will be inserted into Raspberry Pi on the Turtlebot3 Burger robot and follow with other setups on the SBC setup until all it has done.

Then follow with the OpenCR setup by connecting the OpenCR to the Raspberry Pi using the microUSB cable. Then all the required packages on the Raspberry Pi were installed to upload the OpenCR firmware. After all the setup was done and the firmware uploaded successfully, the OpenCR Test will be done to make sure the Turtlebot3 Burger robot has been properly assembled. Next, we will have the hardware assembly and then follow with Bringup Turtlebot3 Burger. RViz will be launched to detect the environment of the turtlebot3 burger in the RViz visualization. The obstacle avoidance node will be applied, and we can see the turtlebot3 burger robot can avoid the obstacle in front of it from starting point until its destination goal. We will have a data analysis and after that, we will apply Kalman filter node to test the performance of turtlebot3 burger by applying the Kalman filter technique to it in different environment.

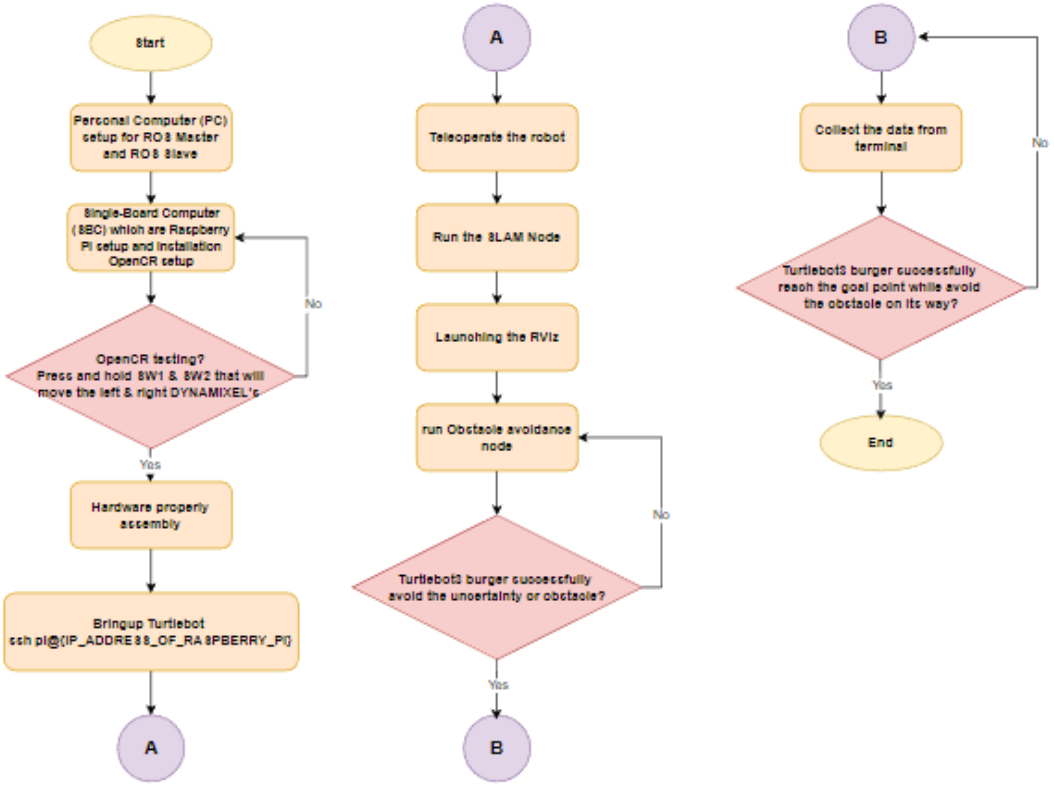


Figure 12 Progression Flowchart on Hardware

3.2 Kalman Filter's implementation in Mobile Robot

In this thesis, The Kalman Filter was used as the solution to the navigation of mobile robots combining the measurement and the prediction to find the optimal estimate of the mobile robot's position. There are three steps (procedure) of Kalman Filter based on figure of block diagram below which are:

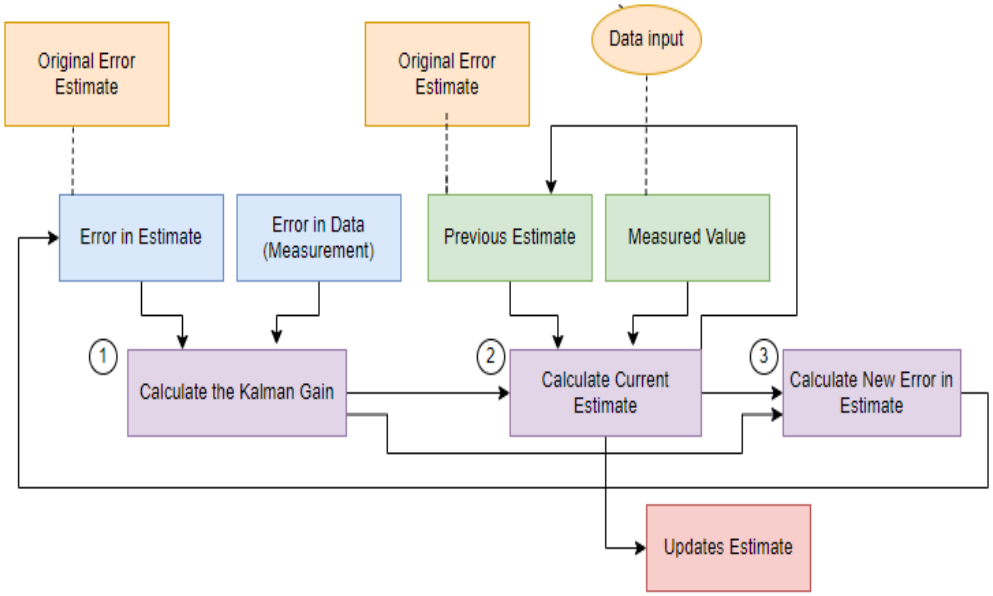


Figure 13 Block Diagram Process of Kalman Filter

1) Calculate the Kalman Gain

2) Calculate Current Estimate (Update estimate)

3) Calculate New Error in Estimate

These three equations or calculations are iterative that might have to happen repeatedly for the estimate to zoom into the actual correct value. Hence, we need to calculate the Kalman Gain. Next, calculate the Current Estimate and must follow with calculating the New Error in Estimate (uncertainty or noise like obstacle or other).

Above in the figure 13 of the block diagram gives a picture of Kalman Filters. Kalman Gain aims to figure out the weightage of measured value and estimation in order to use the update from the previous value of the estimate. Hence, this value will be considered as a new value. Next, the current estimate is also called an update estimate. To obtain the current estimate it refers to the previous estimate and measured value where the previous estimate is an original estimate, and the measured value comes from the data input as shown in the diagram above. In the last step, the value from the new error in the estimate is repeated as an iteration until it narrows down to the true value. As we know that the Kalman filter can be used to estimate the system when it cannot be measured directly. At the initial time step k-1, the actual mobile robot position can be anywhere around the estimate $\boldsymbol{x}k-1$ and this uncertainty is described by this probability density function.

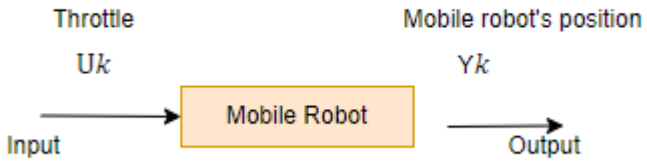
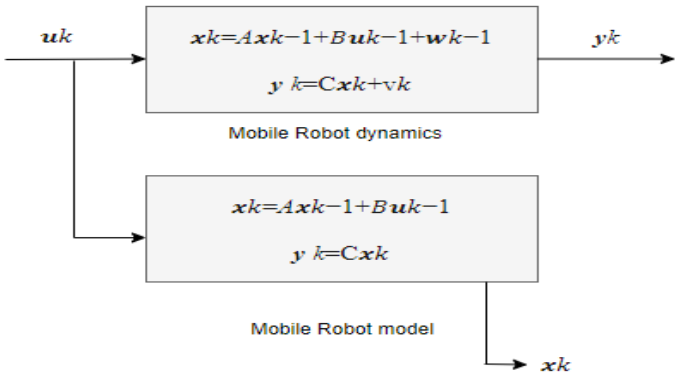


Figure 14 System Mobile Robot



Equation 1 Kalman Filter Equation

In equation 1 above explain that the Kalman Filter combine the measurement and the prediction to find the optimal estimate of the mobile robot's position. This is where the Kalman filters comes into play, it combines these two pieces of information to come up with the best estimate of the mobile robot's position in the presence of process and measurement noise. This formula was stated in the research of (Maqsood 2018).

For "standard" Kalman filtering, everything must be linear and below the specified variable of Kalman Filter equation that can refer as:

$$\boldsymbol{x}k=A\boldsymbol{x}k-1+B\boldsymbol{u}k-1+\boldsymbol{w}k-1 \quad (1)$$

A	The matrix A is the state transition matrix which is applied to the previous state vector $\boldsymbol{x}k-1$
B	The matrix B is the control input matrix which is applied to the control vector $\boldsymbol{u}k-1$
$\boldsymbol{w}k-1$	Unknown system noise or disturbance in process model
Q	Covariance or Process noise or uncertainties

Table 1 Variable Kalman filter System Model of Equation (1)

$$\boldsymbol{y}\,k=C\boldsymbol{x}k+vk \quad (2)$$

$\boldsymbol{y}\,k$	Measurement model
C	Matrix C is measurement matrix
vk	Measurement noise vector

Table 2 Variable Kalman filter Measurement Model of Equation (2)

Equations 1 the system model and equation 2 show the measurement model (sensor) data. From table 1 and table 2 above, all variables and their own description. State variable, x as the input value. So, this input value of x will be predicted by this filter. In simple words, the predicted process by the filter is to determine the location of the model or mobile robot using information about that model or mobile robot's position, velocity, or angular acceleration. While the state variable, y as the measurement model that will be measured. Equation 2 is the measurement model for external sensors that can detect the location of the mobile robot or environment landmarks.

Kalman filter algorithm have a two-step process which are the prediction part and the update part. Every part has their own equation like figure below:

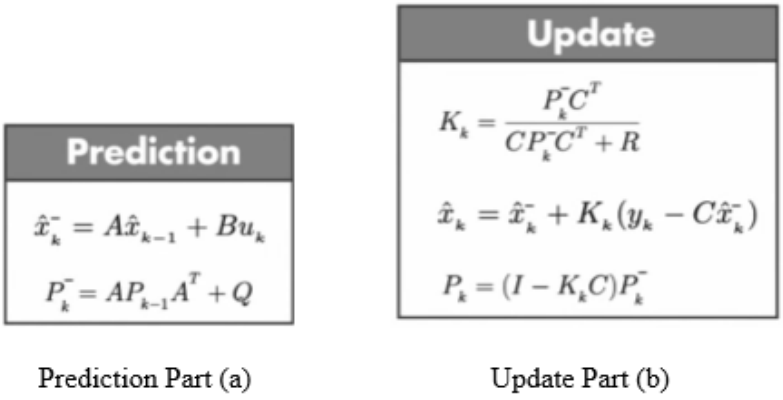


Figure 15 Prediction Part (a) and Update Part (b)

Below are the variable and their own description:

Prediction Part	Update Part
$\hat{x}_k^-$ as a prior state estimate	$\hat{x}_k$ as the state estimate
P_k^- as the Error covariance	P_k as the Posteriori error covariance
	K_k as the Kalman Gain

Table 3 Variable Predict and Update

Hence, the system model is used to calculate a prior state estimate, xk^- and the error covariance, Pk^-. In the single-state system, P is the variance of the prior estimate, and it can be thought of a measure of uncertainty in the estimate state. This variance comes from the process noise and propagation of the uncertain, $\boldsymbol{x}k-1$. At the very start of the algorithm, $k-1$ values for $\boldsymbol{x}$ and P comes from their initial estimates.

The second step of the algorithm uses a prior estimate calculated in the prediction step and update step to find a posteriori estimates of the state and error covariance. In the prediction step, the Kalman Filter predicts the new values from the original value and then predicts the uncertainty or error based on the prior process noise appearing in the system, which is also known as the state error covariance, Q.

While in update step, the actual measured value comes from the device system. The difference between the predicted value and the measured value was decided by calculating the Kalman Gain. From this point, the new value of estimation error can be calculated. As a result, the output from the update step is employed back in the prediction stage, and the process is repeated until there are differences between the predicted and measured values, which are likely to be reduced to zero. Hence, the calculated value is the predicted guess or estimation done by the Kalman Filter.

3.3 Robot Operating System (ROS)

3.3.1 What are ROS and Gazebo?

As stated in chapter 2, The literature review on Robot Operating systems (ROS) is a Linux-based and open-source meta operating system. However, ROS is not an operating system, but middleware for robotics which makes ROS such an adjustable framework for building robot software and aims to make the difficult task of developing complex and robust robot solutions easier. Linux needs an operating system which is Ubuntu Operating System to allow itself to run. This is accomplished via the use of diverse libraries and toolsets built by combining the knowledge and experience of people from various backgrounds. ROS provides a variety of services such as controlling low-level devices, hardware abstraction, implementing commonly used functionality, transferring messages between many processes, and control packages. ROS allows us to use pre-created packages such as G-mapping and teleop_key that can reduce development time. In this paper, ROS used to publish messages in the form of topics between different nodes. So rather than building the entire system in hardware, ROS is used to construct a virtual environment, produce robot models, apply algorithms, and view them in the virtual world. The virtual environment is built with Gazebo and ROS visualization.

The Gazebo is a co-Simulink of ROS and 3D dynamic simulator that will be used in this project for simulating the complex mobile robot which is turtlebot3 burger. The environment or location may be selected between indoor and outdoor. The Gazebo simulator also can connect directly to the ROS via packages. These packages include the interfaces required to simulate a Gazebo robot including ROS messages, services, and dynamic reconfiguration. Gazebo offers a capability that allows you to evaluate the performance of mobile robots in any environment, including severe environments. This is a useful feature since mobile robots may be tested before being used in real-world applications. ROS requires a description of the robot's kinematics to function properly with it. Trajectories, navigation, and other tasks can be designed and carried out in this technique. ROS way of describing a robot is by specifying its properties in URDF (Universal Robot Description Format) files. URDF supports XML and xacro (XML macro) languages. To operate properly with Gazebo, several additional simulation-specific tags must be added to use a URDF file in Gazebo.

The relationship between ROS and Gazebo is the same as the relationship between ROS and real-world robot hardware. The controller in ROS collects information from both models on one topic and publishes data on other topics to both models. Simulation and real robot control can be performed at the same time. Their behaviours and performance may so be easily compared. In fact, ROS makes no difference whether it controls real robot hardware or a simulation model of a robot. The biggest concern here is to implement the relevant nodes and have the proper communication topics. The figure below shows the 3D model turtlebot3 burger simulated in a gazebo and the figure shows the turtlebot3 burger also as a mobile robot model that was tested in the indoor environment.

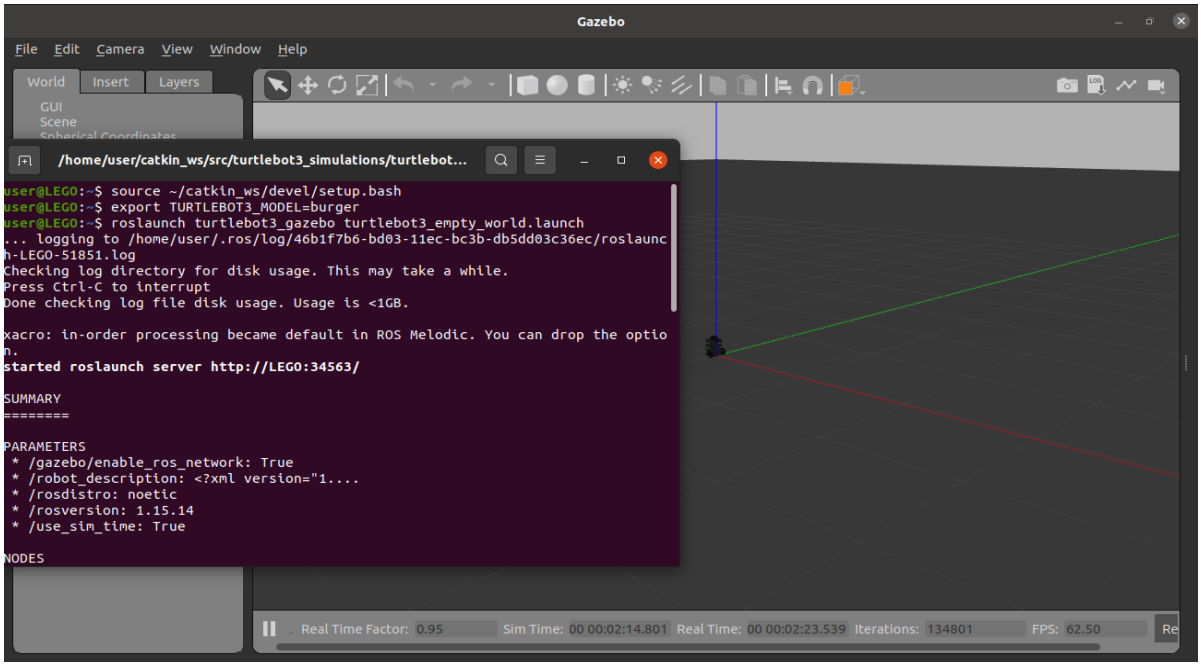


Figure 16 Command to launch the turtlebot3 burger in empty world

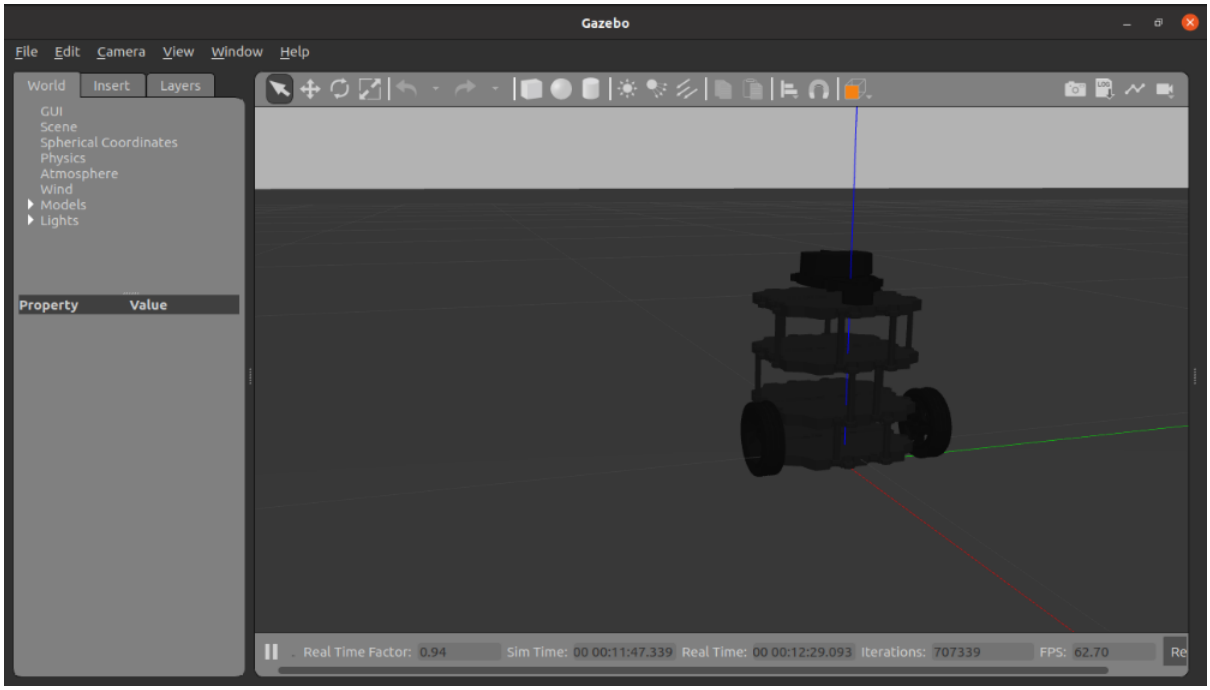


Figure 17 Turtlebot3 Burger simulation in empty world

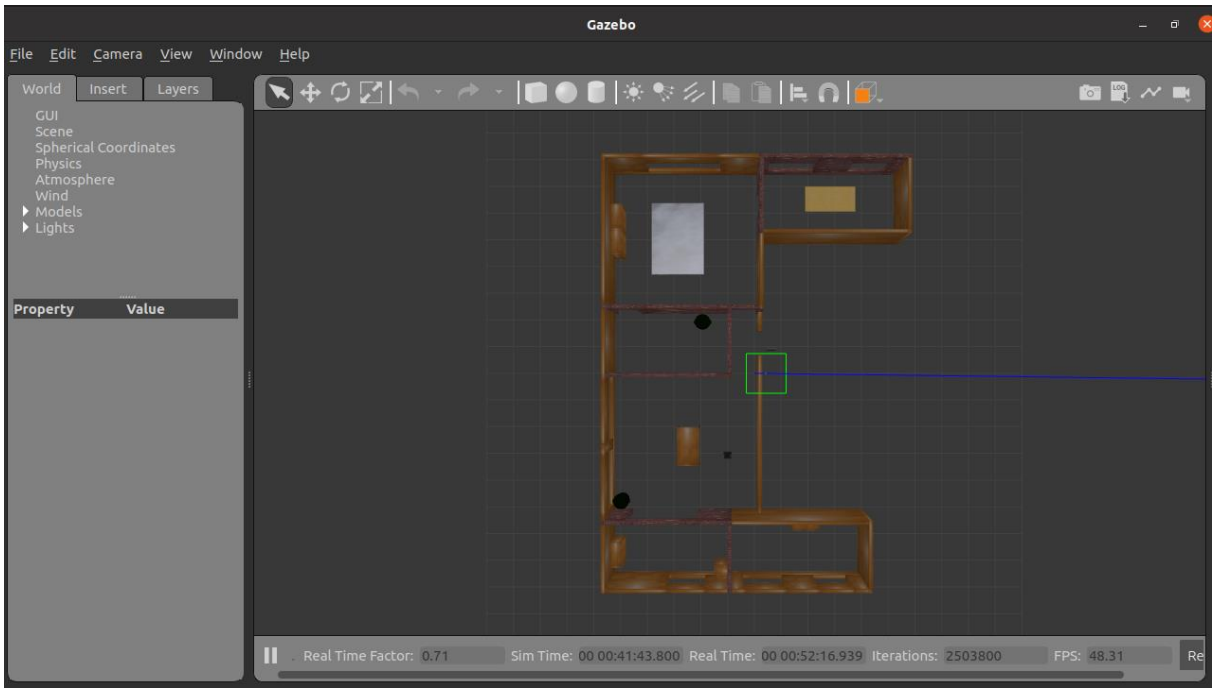


Figure 18 Turtlebot3 Burger in house environment in the gazebo

3.3.2 Installation of Robot Operating System (ROS)

We can find or download on Wiki Ros.org website as stated in figures below and to install we need to change our computer system from window to Ubuntu Operating System.

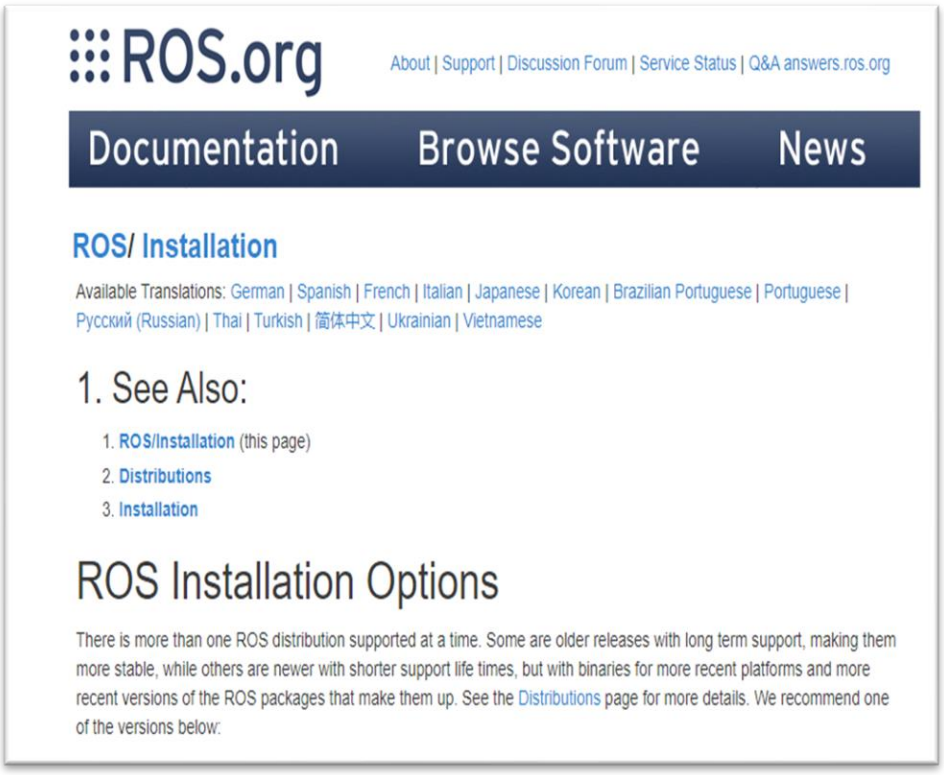


Figure 19 ROS.org website

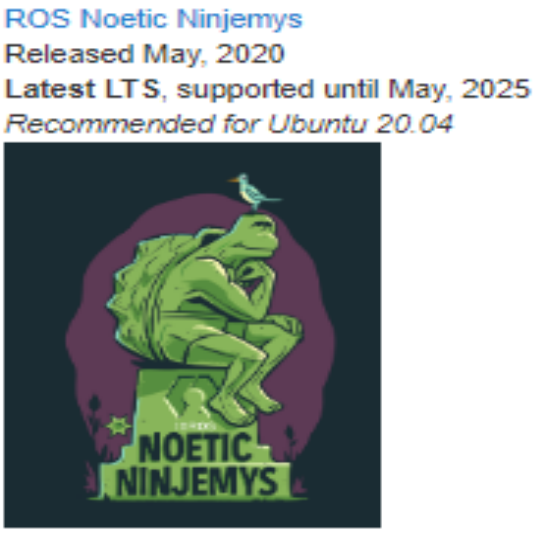


Figure 20 ROS Noetic Ninjemys

3.3.3 ROS Command

This section contains a list of the commands required to communicate with ROS as well as their often-used parameters.

- **roscore** bringup a ROS Master, which when the TurtleBot is not online, is useful for experimenting with ROS on one's workstation. However, to utilize this master instead of the TurtleBot3s, the following must be done in each relevant shell: $ ROS MASTER URI=http://localhost:11311 which is the IP address of the Remote PC (PC).

- **roscreate-pkg** <**package**> [**dependencies**] creates package folders containing the source code for one or more modules. The package directory structure will be generated within the current folder in a new subdirectory named package, which must have been in the $ROS_PACKAGE _PATH.

- **rosmsg** <**verb**> <**arguments**> shows display information about the message kinds that are presently specified and may be passed over topics. When a verb is shown and the inputs are <**package**>/<**message-type**>, for example, an "API reference" of the type and names of the elements in the struct hierarchy corresponding to the message are displayed.

- **rosrun** <**package**> <**node**> is the command that is simply used to run or execute a node once the package has been merged with make.

- **rostopic** <**verb**> <**topicpath**> acts as a link to presently advertised topics over messages that already passed.

3.4 Design Development (Turtlebot3 Burger)

For this project, a Turtle Bot 3 Burger is used to determine the proposed technique performance (Kalman Filter) to guarantee the performance. The Turtle Bot is a small ROS based on a mobile robot and since it is highly customizable and fully open-source hardware, and low-cost it is very suitable for product prototyping. The Turtle Bot 3 Burger can be customised in many ways depending on how we reconstruct the mechanical parts and use optional parts such as computer and sensor. The TurtleBot 3's core technology is SLAM and can run a SLAM algorithm to build a map and can drive around our room, house, and others. It can be controlled remotely from a laptop or Pc and Android-based smart phone. Below is the figure of the Turtle Bot 3 Burger and all the package contents in that Turtle Bot.

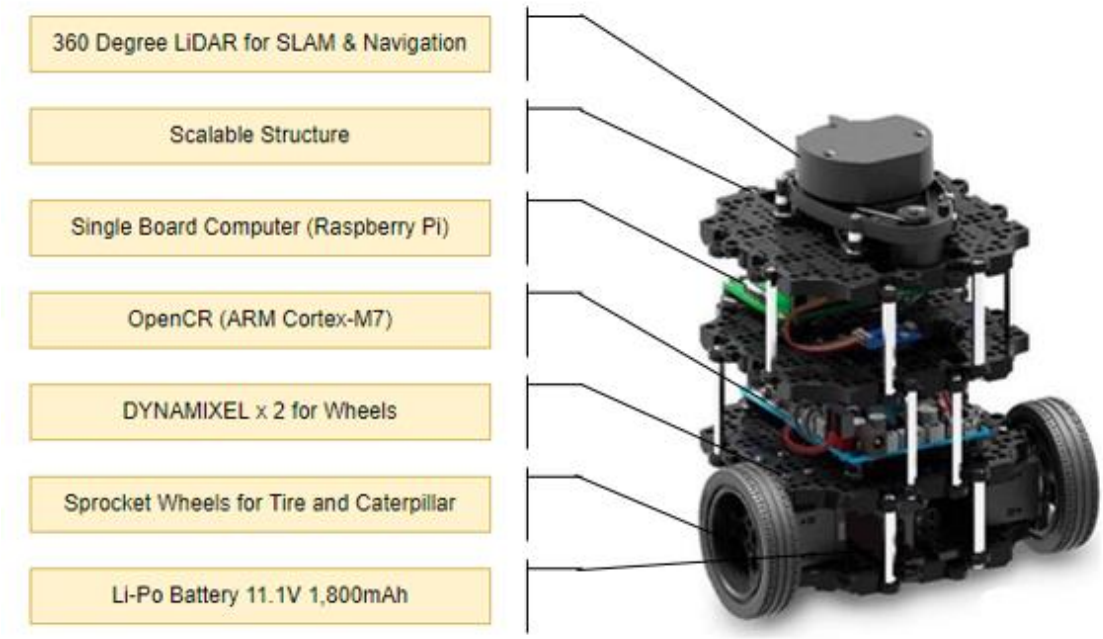


Figure 21 Turtlebot3 Burger

The Turtlebot3 Burger has a Raspberry Pi 3 and a LiDAR sensor. The Raspberry Pi 3 is a single-board computer that is one of the important parts of the hardware components in the implementation of an autonomous vehicle. The Raspberry Pi 3, also known as a data collector, from the lidar sensor and communicates with the lidar sensor to find the position or location of the turtlebot3 burger and receive measurements regarding the surrounding environment of that turtlebot3. The Raspberry Pi supports many operating systems, including Raspbian, Ubuntu, and Windows IoT.

When a lidar sensor scans in all directions (360 degrees) and counts reflected pulses at all angles using a sensor. The distance to the nearest obstacle is calculated using the time and wavelength of the received pulse. After that, all the measurements must be processed to detect obstacles and navigate. As a result, a Raspberry Pi 3 board was used to perform the processing.

3.4.1 Turtlebot3 Burger specification

In the figure below, the details for the turtlebot3 burger specifications. Turtlebot3 burger is a small ROS- based mobile robot. Since it fully open-source hardware, it is very suitable for product prototyping. TurtleBot is the most popular open-source robot for education and research. TurtleBot3 is the most affordable robot among the SLAM-able mobile robots equipped with a 360° Laser Distance Sensor LDS-01.

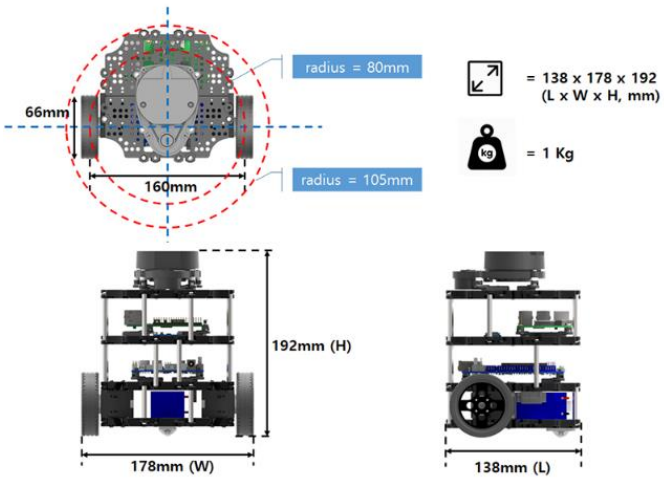


Figure 22 General specification of Turtlebot3 Burger

TurtleBot3 Burger Specifications	
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 3 Model B and B+
MCU	32-bit ARM Cortex-M7 with FPU (216 MHz, 462 DMIPS)
Remote Controller	-
Actuator	XL430-W250
LDS (Laser Distance Sensor)	Raspberry Pi 3 360° Laser Distance Sensor LDS-01 (360°LIDAR)
Camera	-
IMU	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 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 23 General specification of turtlebot3 model type burger

3.4.2 Block diagram of Turtlebot3 Burger system

The turtlebot3 hardware system consists of four major components which are the Data Communication Unit, the Data Processing Unit, the Data Acquisition Unit, and the Robot Actuator Unit. The data communication unit is essentially a PC with Python nodes, with the ROS system serving as the master. It will power and connect to the ROS slave in the data processing unit, which is a turtlebot3 equipped with a raspberry pi. This data processing process will determine the component's behaviour in acquisition unit data. The actuator unit, which is a Dynamixel XL 430-W250 actuator motor, will be the output of this input.

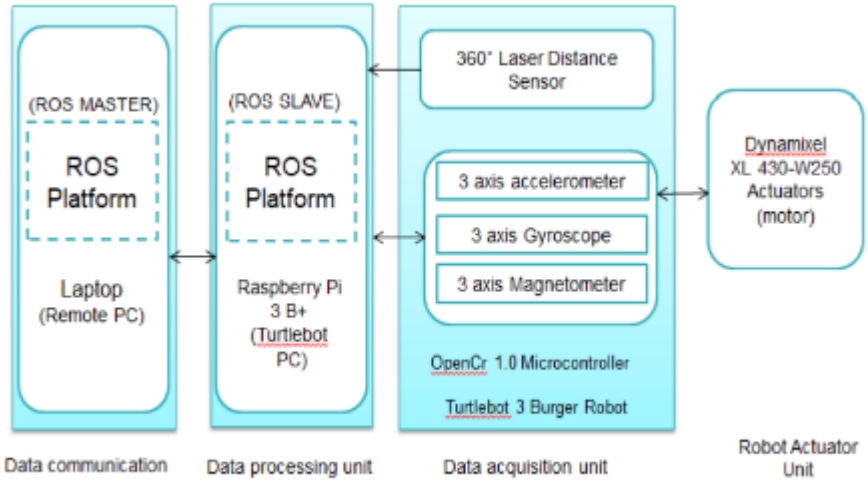


Figure 24 Block Diagram of the Turtlebot3 system

3.4.3 Turtlebot3 Burger Setup

Since the turtlebot3 is equipped with a raspberry pi 3 and OpenCR, both must be properly setup to ensure turtlebot3 functionality. The Raspberry Pi setup, also known as the SBC setup, requires an SD card to burn the downloaded recovery image. This file must be the correct image file version for the hardware and ROS version. The Raspbian OS downloaded will be unzipped and saved to the local disc. The image file will then be burned using the Raspberry Pi imager. The whole step to setup this SBC and OpenCR can be found in Emanual Robotics websites.

After burn the image file in microSD card finish, the SD card need to insert into the Raspberry pi slot in the turtlebot3 burger. Then need to follow with boot up the raspberry pi by start with connect the HDMI cable of the monitor to HDMI port of raspberry pi. The monitor will be used to find the Ip address of that raspberry pi. Then, follow by connect the input devices such as a keyboard and mouse to the USB port of the raspberry pi 3. After that, the power supply will be connected to turn on the raspberry pi. It can be either USB or by OpenCR which using the battery. The raspberry pi needs to configure by connecting with the same Wi-Fi network that connect to the PC. For both PC and monitor that connected to raspberry pi, need to determine with the command $ ifconfig to know the Ip address for both. Usually, the Ip address for raspberry pi can be found under wlan0 section. Then, from the PC, new terminal will be open and connect with the raspberry pi with its Ip address. The default password is set as “turtlebot”. Its command is $ ssh pi@{IP_ADDRESS_OF_RASPBERRY_PI}. After getting both Ip address for PC and raspberry pi 3, the new terminal will be open, and its command is $ nano ~/. bashrc then, the Ip address for both PC and raspberry pi 3 need to copy then paste in nano file under the section export ROS_MASTER URI and export ROS_HOSTNAME respectively. Then apply changes by using the $ source ~/. bashrc command.

The OpenCR is connected to the Raspberry Pi via a micro-USB cable for the OpenCR setup. The necessary packages are installed on the Raspberry Pi to upload the OpenCR firmware. The firmware then was uploaded to the OpenCR. The OpenCR test can be used to validate a successful firmware setup. In this test, the button SW1 and SW2 need to push to see whether the left and right DYNAMIXEL’s wheels can be move. Next, the red power led will be turn on or blinking after the turn on the power of OpenCR. The

turlebot3 need to place on the flat surface in a wide area for testing. Press and hold button sw1 and sw2 for a few second to command the robot to move about 12 inches forward and rotate 180 degrees in place respectively.

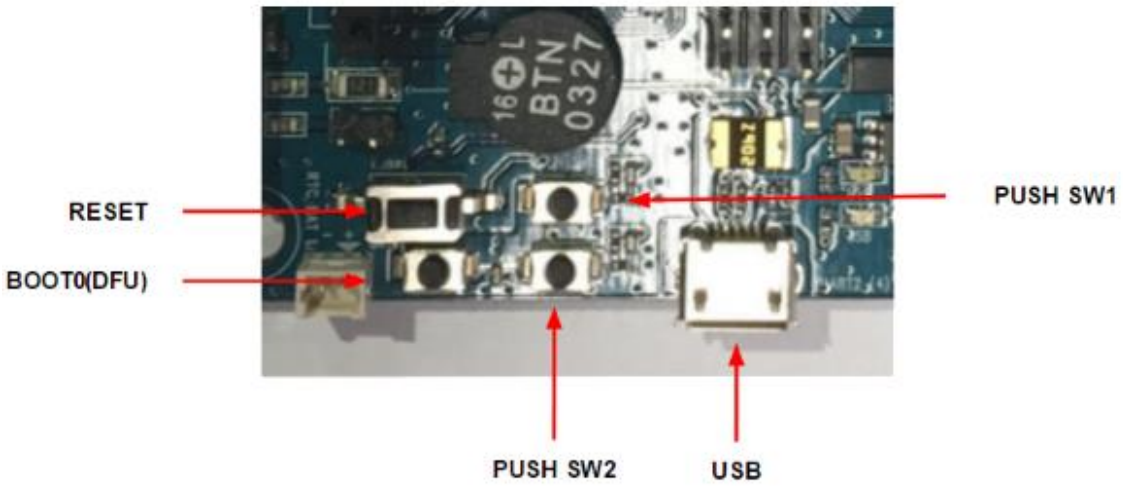


Figure 25 OpenCR Board

3.4.4 Turtlebot3 Burger Bringup

The $roscore command in the terminal is the first step in bringing up the turtlebot3 burger. This command runs from the PC. Using the $ifconfig command to get the Ip address and the $nano /. bashrc command to set the Ip address PC as ROS_MASTER_URI and the turtlebot3 burger also known as raspberry pi Ip address as ROS_HOSTNAME like in SBC setup before. Connect the Raspberry Pi to the PC by typing $ ssh pi@{IP_ADDRESS_OF_RASPBERRY_PI} into a new terminal. Then, using the command $ roslaunch turtlebot3_bringup turtlebot3_robot.launch, bring up all the turtlebot3 packages to start the Turtlebot3 application. A successful connection will be depicted in the figure below.

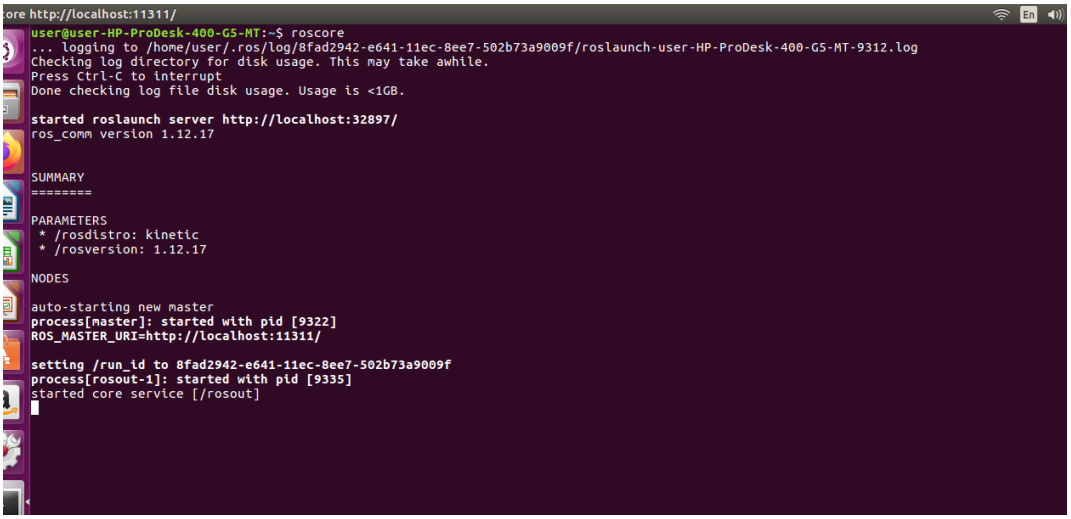


Figure 26 Run roscore from PC

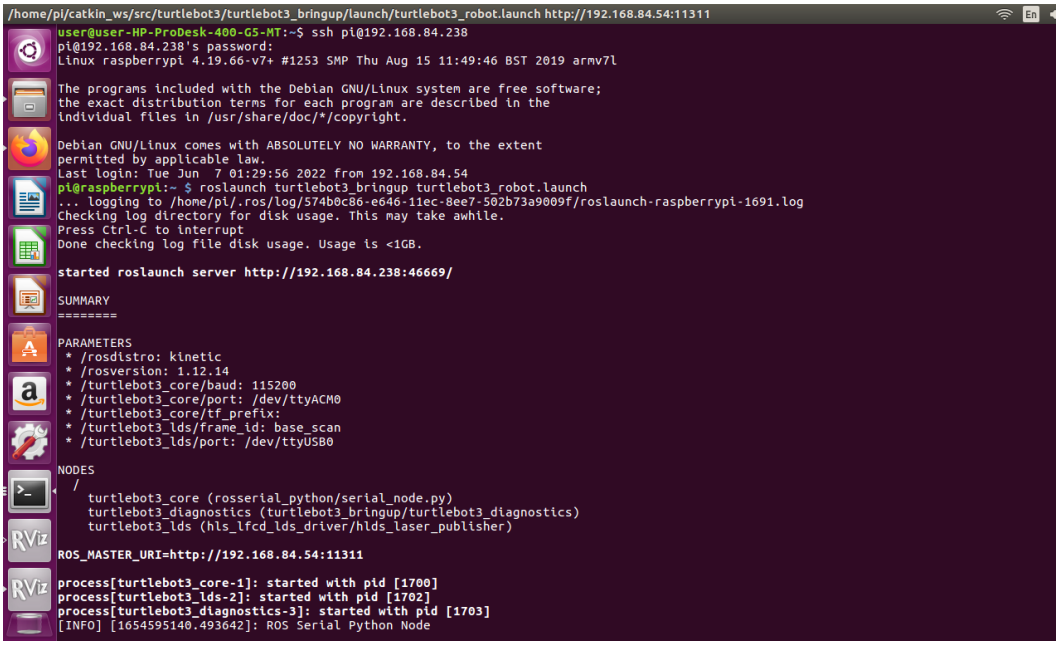


Figure 27 Bringup Turtlebot3

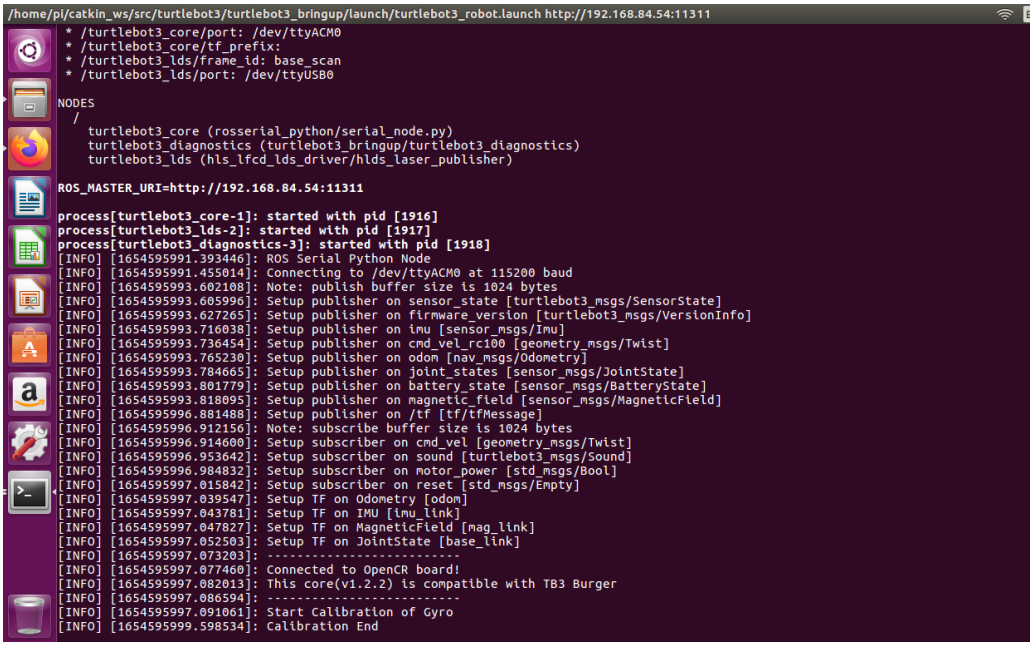


Figure 28 Visualization on the terminal of the successful turlebot3 bringup

3.4.5 Launching the obstacle avoidance node to Turlebot3 burger

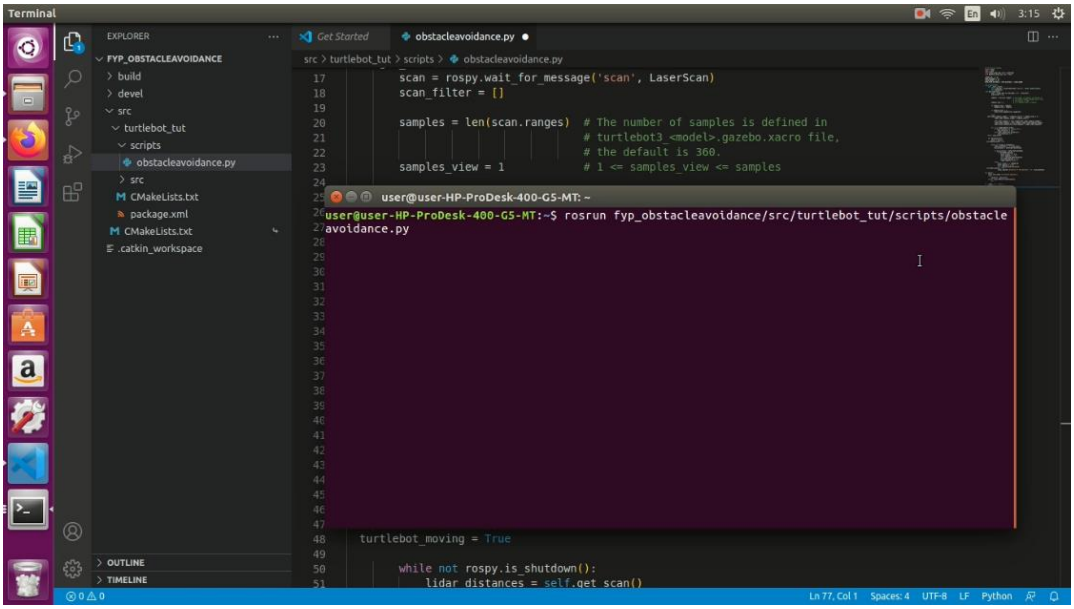


Figure 29 Turtlebot3 node run

3.4.6 Slam Algorithm of the Turlebot3 burger

The PSM Laboratory room was chosen as a place to run the SLAM of the hardware part. This part shows initial and final map generation of turtlebot3 of the Lab room before setup with any obstacle. The first figure of the map is the visual of the room right after launching the SLAM Node to scan the map and before the turtlebot3 burger is running and scanning using the LDS or Lidar sensor. The second figure is the result after 30 minutes the turtlebot3 burger runs to scan the map of the environment in the room.

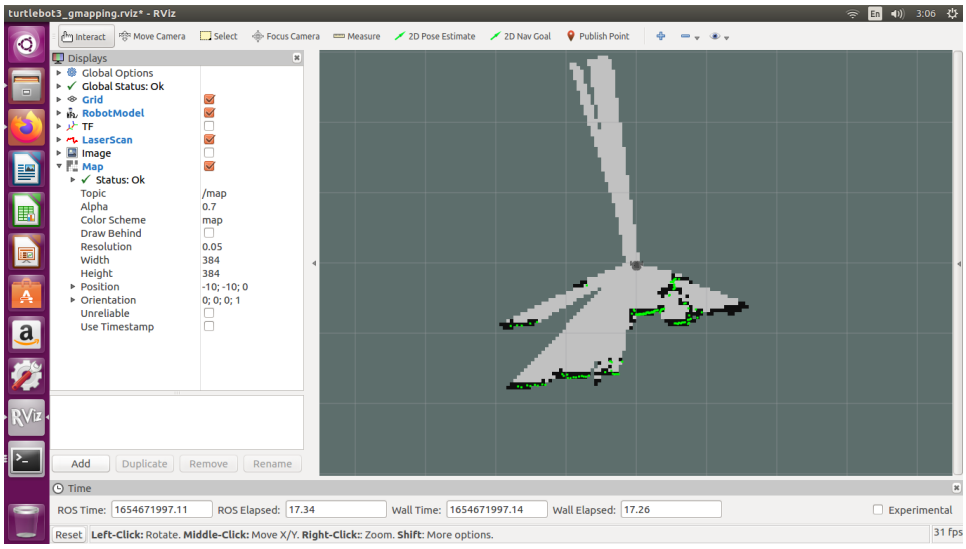


Figure 30 The map after launch RViz for the first time before scanning process

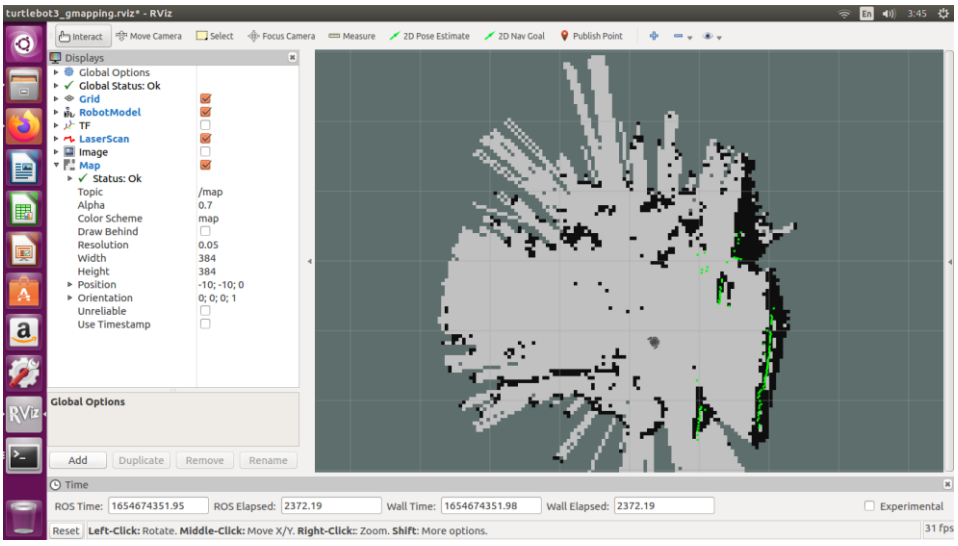


Figure 31 The map after 30 minutes the turtlebot3 runs

CHAPTER 4

RESULT AND DISCUSSIONS

4.1 Introduction

In this chapter, the result for this thesis will be discussed. It is to actively demonstrate the expected outcomes for this thesis where the design of a mobile robot will be based on the simulation by applying the Kalman filter to the system model in MATLAB. Another scope of this thesis is to analyse the mobile robot's movement in different conditions using ROS, Gazebo and Rviz and the output compared by applying Kalman filter and without Kalman filter of turtlebot3's performance effects. Next, the effectiveness of lidar sensor also were proved.

4.2 Result from the Hardware Part

4.2.1 Detection of obstacle and avoidance data

Table 4 below shows the minimum and maximum data taken from the reading of the lidar sensor from turtlebot3. The minimum distance measurement was set to be 0.25 meters, which means that it will never touch below 0.25 meters and the maximum values is 4.5 meters and the average error of lidar sensor measurements of 0.4%. The tests are performed with varied environmental different object colour and type to verify the trustworthiness of the LiDAR data.

Distance (m)	Read Distance (m)	Error (%)
0-0.24	0	-
0.25	0.251	0.40
0.26	0.261	0.38
0.30	0.301	0.33
0.56	0.561	0.18
0.82	0.827	0.85
1.5	1.450	0.33
2.5	2.509	0.36
3.5	3.517	0.48
4.5	4.531	0.68
Average		0.4

Table 4 Lidar measurement

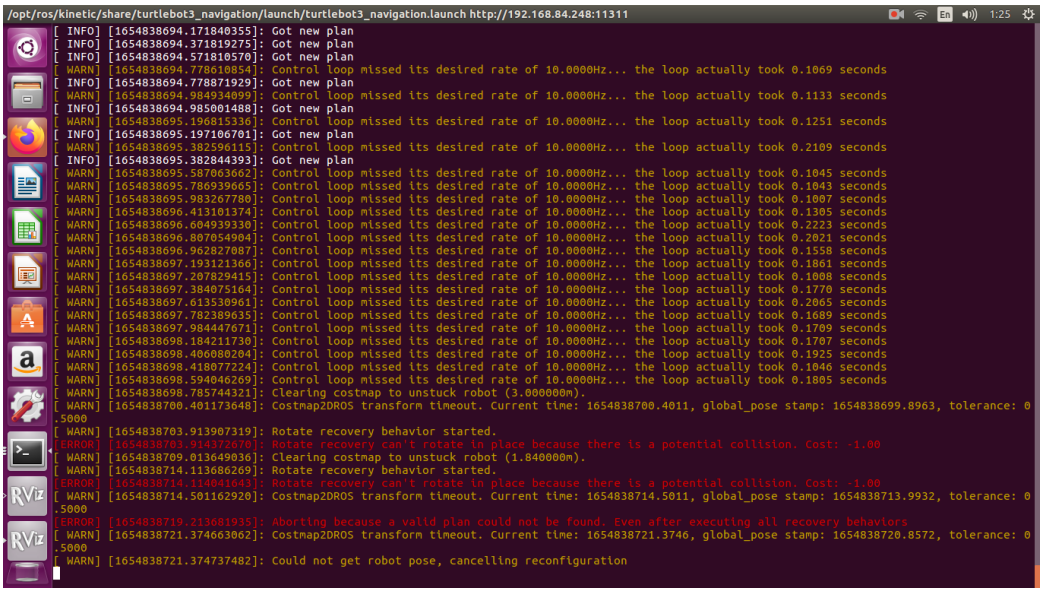


Figure 32 Data error collision

In the figure of 32 above show the data error collision that get from the terminal. In the terminal it will display the info, warning, and error. If the turtlebot3 collide with any obstacle on its ways in the terminal will show the error that stated, the robot cannot move anymore due to presence of obstacle in front of it. When the lidar sensor detect the obstacle, in the terminal will stated warn.

4.2.2 Cases study

4.2.2.1 SLAM Algorithm of Turtlebot3 in Environment 1

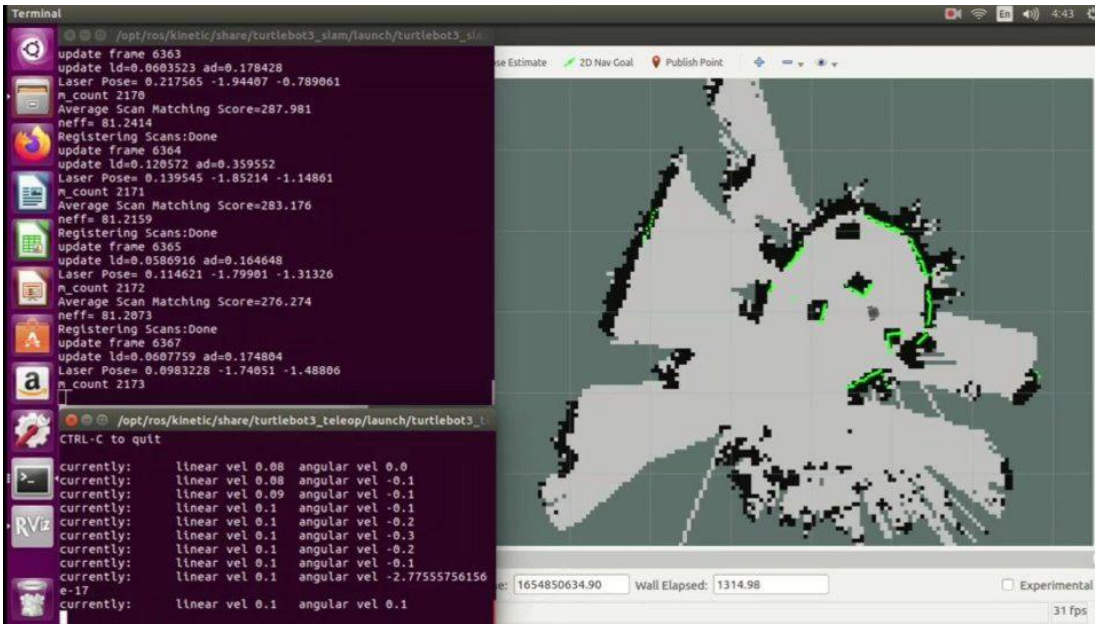


Figure 33 Final map generation from Lidar sensor for Environment 1

In figure 33 above shows the final map of the room for environment 1 with the obstacle. This is the visual of the room after 20 minutes the turtlebot3 providing velocity and avoid any obstacle around it. This is a good result from the turtlebot3 as it proven that the turtlebot3 can move around the room in environments 1 without any problem. This is also proved data from the Lidar sensor is fully functioning during turtlebot3 is moving around in this environment.

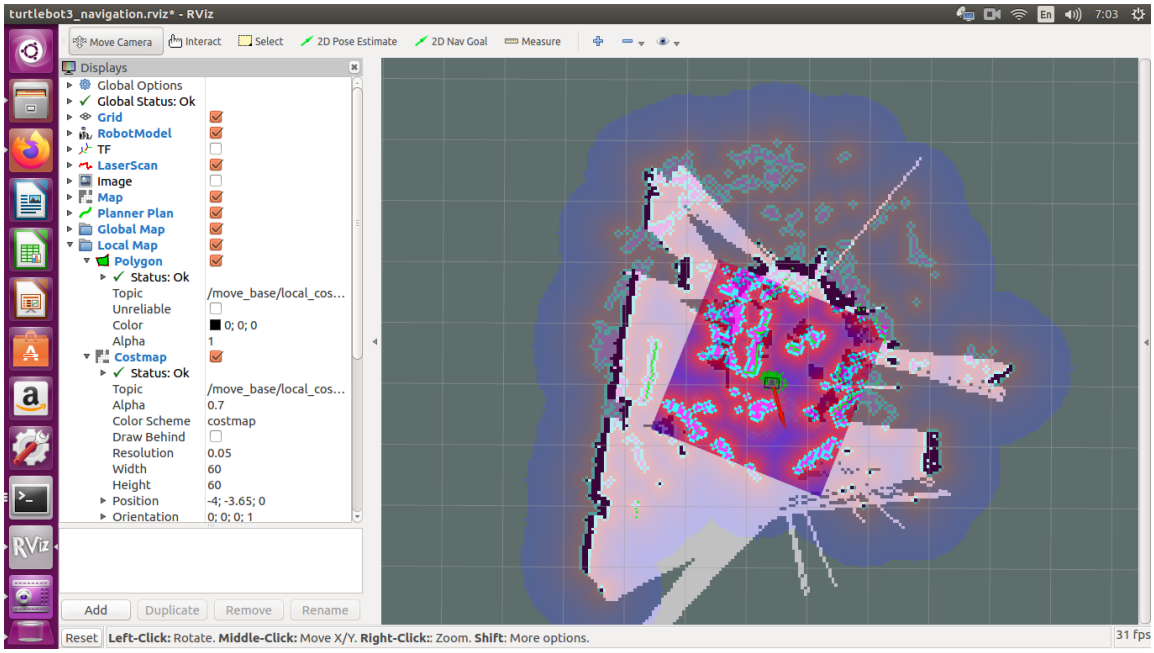


Figure 34 2D Navigation in Rviz of Environment 1

Figure 34 above shows the 2D navigation in Rviz of environment 1 after launching the command of the run navigation nodes. The purpose of navigation is to move the turtlebot3 from one point to the set destination as clarified on Emanual Robotics websites. In Rviz, generating a map also consists of information such as furniture, object, and wall in an environment. As described in the previous SLAM section in figure 33 above, the map was created with the distance information obtained by the lidar sensor and the pose information of the turtlebot3 itself.

4.2.2.2 Environment 1 with Kalman filter and without Kalman filter

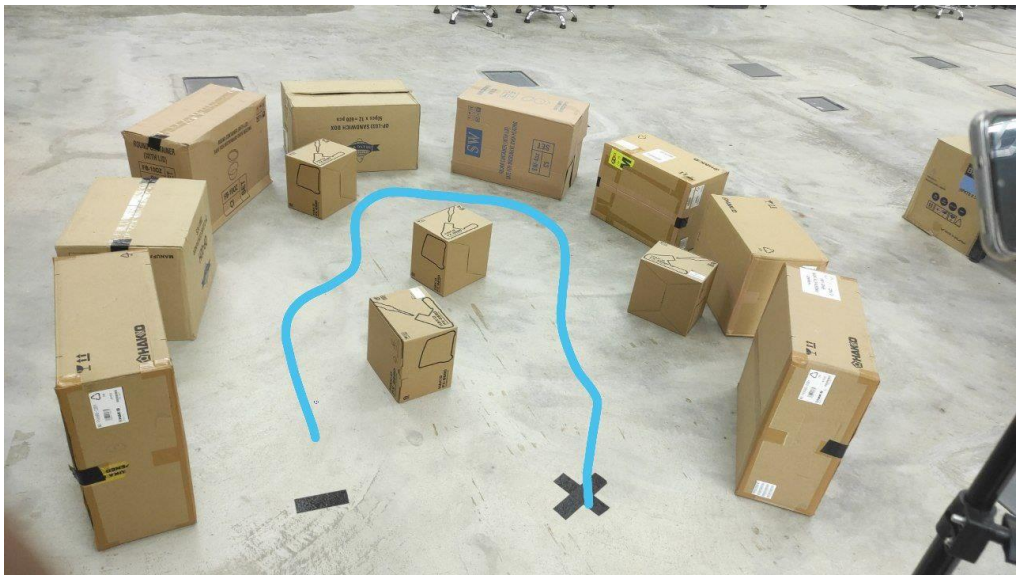


Figure 35 Environment 1 with Kalman filter

Figure 36 Environment 1 without Kalman filter

	With Kalman Filter	Without Kalman Filter
Obstacles	4	4
Time (s)	121s	138s
Path Length (m)	4.20m	4.20m

Table 5 Comparison with Kalman Filter and without Kalman Filter

In figure 35 and figure 36 above show the environment 1 setup, both with the same condition. In both figures also have a same number of obstacles were setup. Then, table 5 showing a result of Turlebot3 Burger move to avoid the obstacle by applying the Kalman filter and without applying Kalman filter into it, which is considered in terms of distance and time taken. As illustrated in table 5, when the Kalman filter is applied, the turtlebot3 taken only 121 second to reach its goal without any collisions. While, without the Kalman filter, the turtlebot3 takes longer to reach the goal because the mobile robot seems confused about its next path and tend to lose its way to get there without colliding with any obstacles in its path.

4.2.2.3 Environment 2 with different type of obstacle

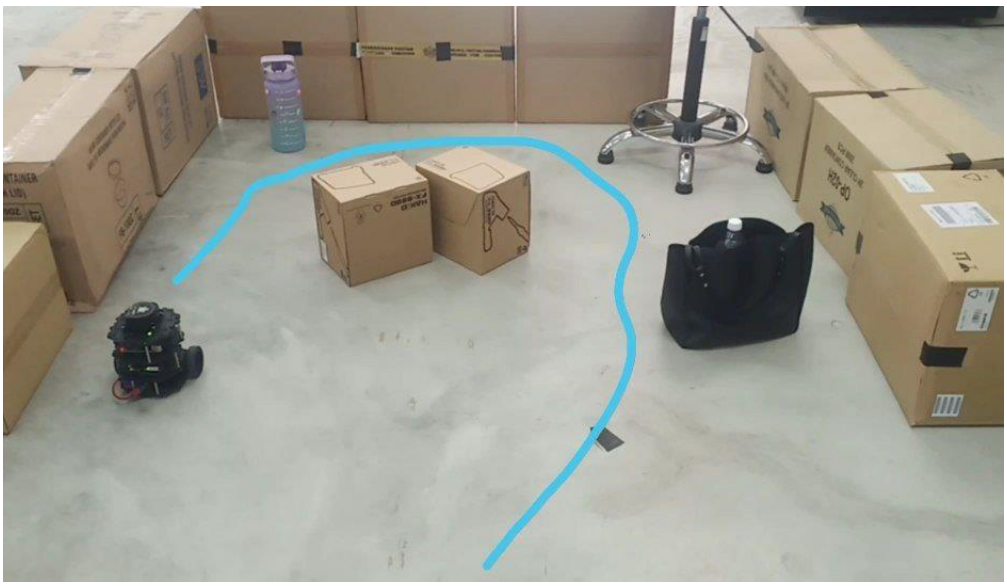


Figure 37 Environment 2

Figure 37 above show environment 2, in this environment four different types of obstacles were set up to see whether the turtlebot3 still can detect different type of obstacle with different colours. After the experiment was done, it was found that the turtlebot3 also can reach the goal point while avoiding the obstacle on its way even though with various types of obstacles in front of it.

4.2.2.4 Environment 3

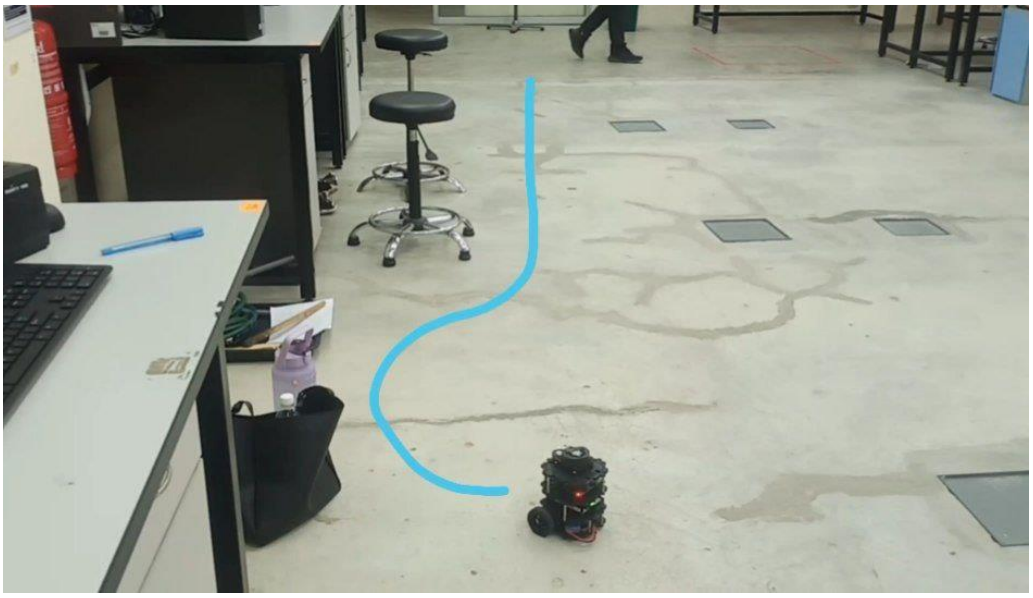


Figure 38 Environment 3

While figure 38 above shows the environment 3. In this environment, it is to show a different motion of turtlebot3 burger movement. This section is only to show a variation motion for the robot movement other than the other 2 environments.

4.2.3 Summary of Turtlebot3 Burger Performance

No	Environment	Running time (s)	Path length (m)
1	1	121s	4.20m
2	2	115s	4m
3	3	74s	2.52m

Table 6 Summary of Turtlebot3 Performance

Table 6 above show the summary of the turtlebot3 performance compared in all environments. For environment 1, the time taken for turtlebot3 burger to running from starting point to goal point while avoiding the obstacle is 121 second with the path length 4.20 meters. While for environment 2, the running time for turtlebot3 to reach its goal is 115 second and the path length is 4 meters. Last but not least, the time taken for the turtlebot3 running from start point to goal point is only 74 second with the path length 2.52 meters in the environment 3. We can see from the summary of environment results 1, 2, and 3 above that the complex path takes the longest time for the turtlebot3 to reach its goal. Because the complicated route restricted the robot's movement

4.2.4 Result Comparison

In this section, the effectiveness of the LiDAR sensor to detect the different types of obstacles ahead of the turtlebot3 burger was proven by supporting our results with another research paper result which is from the paper (Hutabarat, 2019).

Figure 39 Comparison Environment 2 with different type of obstacle

Type of obstacle	Collisions
Box	No
Chair	No
Bag	No
Bottle	No

Table 7 Type of Obstacle in environment 2 setup

Object	Object distance: 25 cm		Object distance: 50 cm	
	Collisions		Collisions	
	Yes	No	Yes	No
Acrylic	✓		✓	
Polycarbonate Bottle	✓		✓	
Glass Bottle		✓		✓
Jerry cans		✓		✓
Cardboard		✓		✓

Figure 40 Table of comparison from other research paper

Table 7 above shows the different type of obstacles that were used in the experiment which is a box, chair, bag, and bottle. From the result of the environment 2, turtlebot3 burger can reach its destination while avoiding the different types of obstacles, it proved that there are none of these four types of the obstacle was hit by the turtlebot3.

(Hutabarat 2010) stated that the LiDAR sensor is a scanner technology that detects the properties of emitted light to determine distance and other information from a target. The author support that by doing an experiment and showing the result in figure 40 above, his mobile robot with a lidar sensor can avoid different object, of different size namely glass bottle, jerry can, and cardboard. However, his result shows, that the transparent objects, such as acrylic and polycarbonate bottles, cannot be avoided by the robot. The author said it can be caused by the laser pulses emitted by lidar do not reflect to the object or obstacle.

4.3 Result from the Simulation Part

4.3.1 Simulation based Obstacle Avoidance in Gazebo world

In the Gazebo world, test conditions are created. The turtlebot3 burger is positioned at (0,0) and was ordered to move to its destination at (4,0). (Meter distance). The turtlebot3's linear and angular velocity are limited to 0.15m/s and 5 rad/s, respectively.

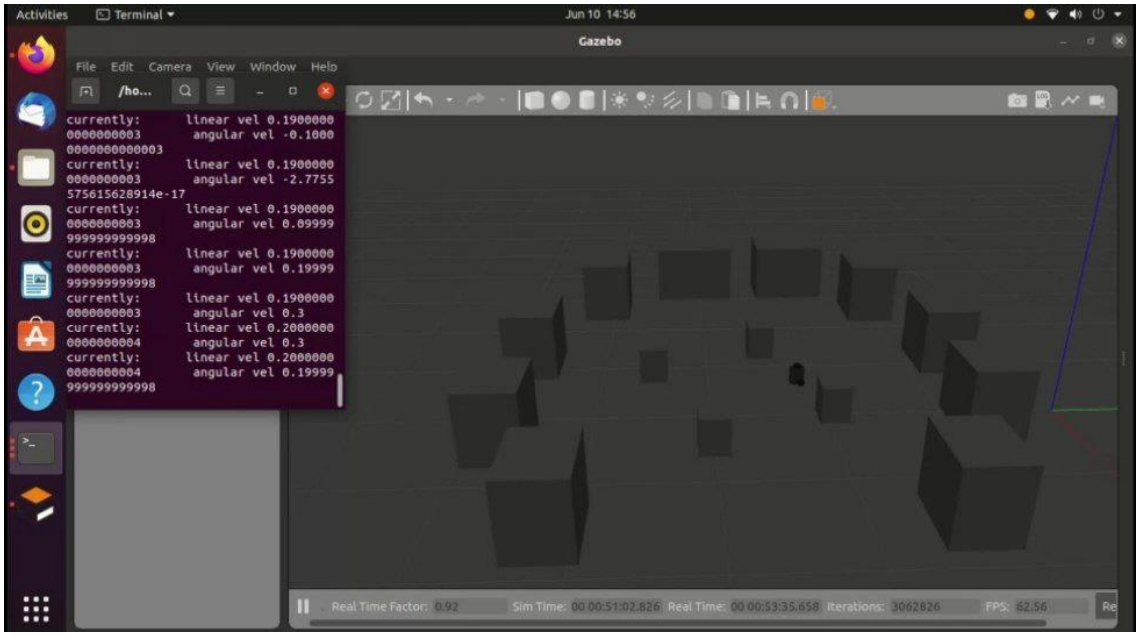


Figure 41 Simulation of turtlebot3 in Gazebo world

Simulation environment with cluttered-like obstacles also was developed in Gazebo as shown in figure 41 above. Due to the existence of obstacles between the starting and finishing point, the robot is unable to proceed with the straight line, it needs to avoid the obstacle on its ways to reach the goal point.

4.3.2 Simulation for State Estimation by applying Kalman Filter in Turtlebot3 model

This result from the simulation on state estimation using Kalman filter that were implemented on MATLAB. This result is to estimate the position and velocity of a turtlebot3 in the x and y directions. This estimate based on noisy position measurement from the sensor measurements.

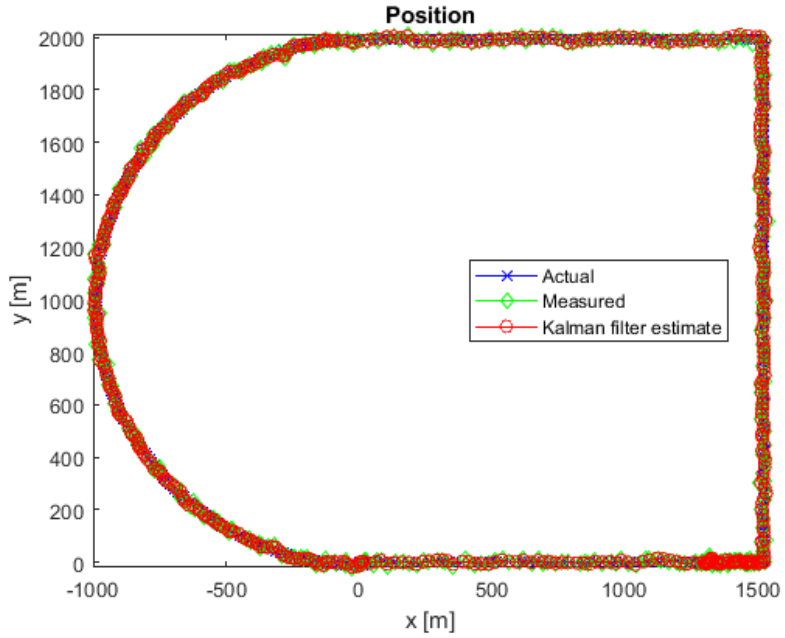


Figure 42 x-y Position of the Turtlebot3 with help of Kalman filter

The figure 42 above shows the x-y position of the turtlebot3 from the MATLAB simulation. The blue color represented the actual position of the turtlebot3, while the green color indicates the measurement result that get from the sensor. The red color represented the estimation position of the turtlebot3 by using Kalman filter.

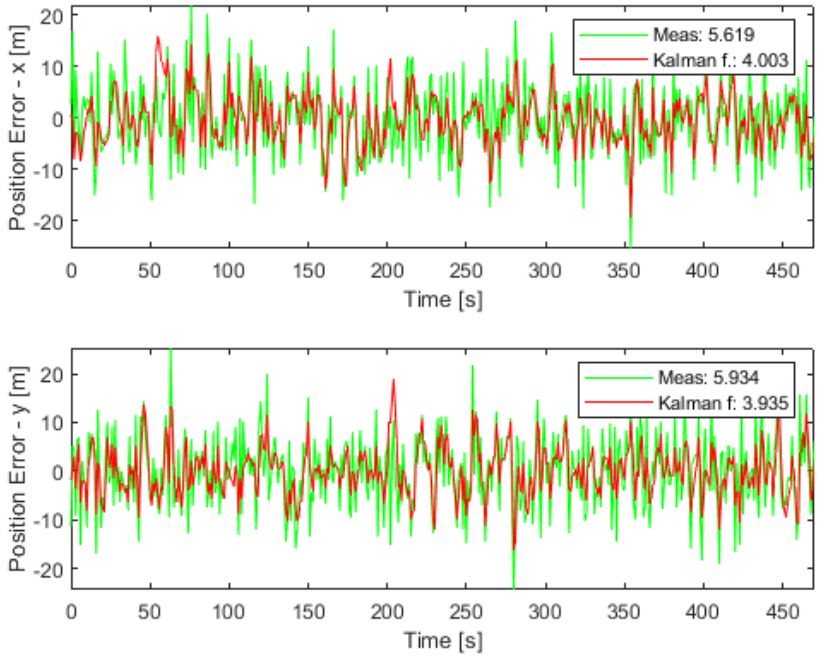


Figure 43 Error between Kalman filter estimate & measured

While the above figure 43 above shows the result of error between the Kalman filter estimate and measured. The graph was plotted to show the position measurement and estimation error.

		Kalman filter (KF)	Without Kalman filter (WKF)	Error (%)
Position	x(m)	2.40	4.35	28
	y(m)	2.17	3.26	20

Table 8 Percentage error between cases with KF and WKF in x and y position

From the graph in figure 43 above, we extract into comparison the percentage % of error between cases with Kalman filter and without Kalman filter in position x and y coordinates (m) that show in table 8 above. In x coordinates or position it shows that without Kalman filter produces larger error compared between Kalman filter where when implementing the Kalman filter, it produced 2.40 and without Kalman filter produced 4.35 in x position. While, in y position, without Kalman filter also produced a larger error which is 3.26 compared with the Kalman filter produced 2.17. To conclude, in x position the error calculated between Kalman filter and without Kalman filter is 28% error and in y position the error that calculated between KF and WKF is 20% error. In Kalman filter-based navigation, the main concern is to access the estimation errors. Because the small error will provide a better estimation. Thus, it ensures better performance especially in terms of robot navigation. Kalman filter performance also gives the efficiency of the estimation, it has been mentioned in the previous research paper of (Hamzah Ahmad 2021, November).

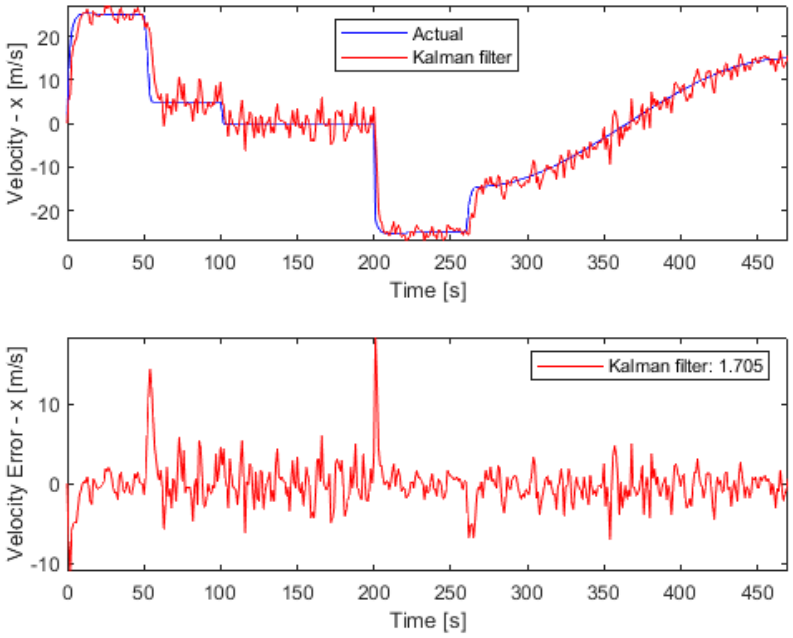


Figure 44 Velocity Estimation Error of x position

The graph in figure 44 above shows the x velocity estimation error, the Kalman filter estimations accurately match the actual velocity patterns. The noise level decreases when the turtlebot3 is move at a higher velocity. There are two larges' spikes at time (second) equal to 50 and 200. These occur when the turtlebot3 makes a sharp turn. The velocity changes at these points are hugely more than predicted by the Kalman filter. Unfortunately, after a few seconds, the filter estimations finally match again the actual velocity.

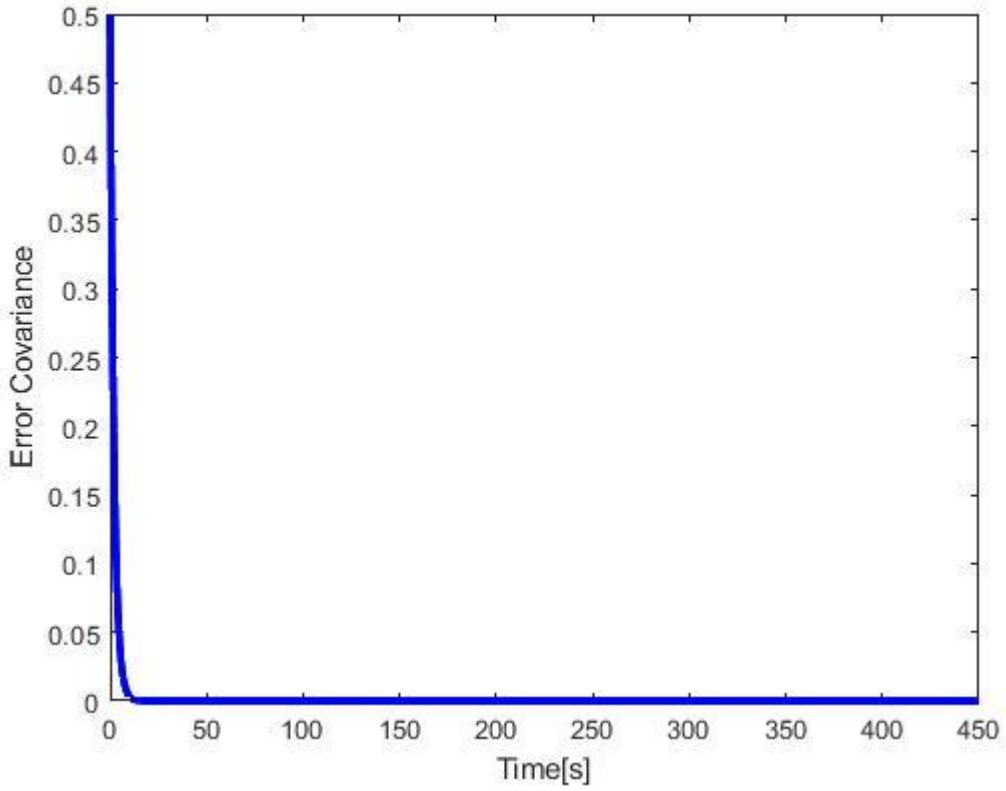


Figure 45 Covariance matrix

Figure 45 above shows the covariance matrix. This covariance matrix was plotted to determine the accuracy of the filter. The graph is converged to a value referred to in this case is 0. Then, we can determine that the Kalman filter provides a better estimation for the mobile robot as the graph does not diverge. As clarified in the paper (Casanova, O.L. 2008). They claimed that divergence in the covariance matrix indicates that the Kalman filter does not work properly. Hence, this simulation has proof that the Kalman filter gives a better estimation.

CHAPTER 5

CONCLUSION

5.1 Conclusion

Taking everything into account, this project was focused on evaluating performance mobile robot using ROS. Considering Kalman filter estimation to provide better estimation. The result has been presented by simulation in MATLAB to show state estimation of mobile robot by applying Kalman filter algorithm. In addition, in ROS we had setting up turtlebot3 burger also in Gazebo and Rviz. In simulation depicts that using Kalman filter and without Kalman filter has around 28% less error in x position and 20% error in y position. In this work, Lidar was equipped in turtlebot3 to avoid the obstacle. It shows that using Lidar can avoid many types of obstacles during observation where it shows that mobile robot does not collide with any of the obstacle. Next, it also illustrates that by using Kalman filter the turtlebot3 can reach its goal much faster without involve in any collision compared between without applying Kalman filter. In addition, all of the simulation and hardware setup result shows the similar output with theoretical analysis from previous study. All in all, it is compulsory for mobile robot to apply a good technique to solve any issues especially in navigation.

5.2 Future Recommendations

In the future work of Artificial Intelligence in an autonomous vehicle, through more software development, hardware implementation, and analysis, the existing effort can attain greater precision in navigation. Adjustments to the algorithm are also important and might potentially be made to detect various obstacle movements, and motions, more complex routes and navigate more efficiently in dynamic areas. In the future, we also aim to merge fuzzy logic and lidar sensor with the filters proposed in this research to improve position estimation. Furthermore, in order to accomplish more accuracy in motion tracking, smooth movement, and precise positions, other systems including a speed controller and an inertial sensor must be included. This is recommended for future research on this topic.

REFERENCES

1) Li, Y. and C. Shi (2018). Localization and Navigation for Indoor Mobile Robot Based on ROS. 2018 Chinese Automation Congress (CAC).

2) Yu, Y.-S., Chih-Heng Ke, Yeong-Sheng Chen, and Pin-Yuan Y (2019). "Development of Following Vehicle Prototype Using Robot Operating System." In 2019 8th International Conference on Innovation, Communication and Engineering (ICICE): 142-144.

3) Suliman, C., Cruceru, C., & Moldoveanu, F. (2009). "Mobile Robot Position Estimation Using the Kalman Filter." Acta Marisiensis. Seria Technologica, 6,: 75.

4) Zhi, L., & Xuesong, M. (2018). "Navigation and control system of mobile robot based on ROS." In 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC): 368-372.

5) Zhu, J., and Li Xu (2019). "Design and Implementation of ROS-Based Autonomous Mobile Robot Positioning and Navigation System." In 2019 18th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES): 214-217.

6) Burnett, J. R. (2006). "Mobile robot localization using a Kalman filter and relative bearing measurements to known landmarks

7) Maqsood (2018). "Kalman Filters."

8) Hutabarat, D., Rivai, M., Purwanto, D., & Hutomo, H. (2019, July). "Lidar-based Obstacle Avoidance for the Autonomous Mobile Robot." In 2019 12th International Conference on Information & Communication Technology and System (ICTS): 197-202.

9) Péter Fankhauser, D. J., Martin Wermelinger, Marco Hutter (2017). Programming for Robotics–Introduction to ROS.

10) Takaya, K., Asai, T., Kroumov, V., & Smarandache, F. (2016, October). "Simulation environment for mobile robots testing using ROS and Gazebo." In 2016 20th International Conference on System Theory, Control and Computing (ICSTCC). IEEE.: 96-101.

11) Caro, G. A. D. (2010). Introduction to ROS. Introduction to Robotics.

12) Panich, S. (2010). "Indirect Kalman Filter in Mobile Robot Application." J. Math. Stat, 6(381384.43).

13) Bersani, M., Vignati, M., Mentasti, S., Arrigoni, S., & Cheli, F. (2019, July). Vehicle state estimation based on Kalman filters. In 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE) (pp. 1-6). IEEE.

14) Shepelev, D., & Ustyuzhanin, A. (2015, April). Towards development of reliable mobile robot navigation system. In 2015 2nd International Conference on Information Science and Control Engineering (pp. 1006-1010). IEEE.

15) Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

16) Meng, Z., Wang, C., Han, Z., & Ma, Z. (2020, September). "Research on SLAM navigation of wheeled mobile robot based on ROS." In 2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE). IEEE.: pp. 110-116.

17) Thale, S. P., Prabhu, M. M., Thakur, P. V., & Kadam, P. (2020). "ROS based SLAM implementation for Autonomous navigation using Turtlebot." In ITM Web of conferences (Vol. 32, p. 01011). EDP Sciences.

18) Kwon, S. J., Yang, K. W., Park, S. D., & Ryuh, Y. S. (2005). A Kalman Filter Localization Method for Mobile Robots. pp 973-978

19) Kang, Y., Roh, C., Suh, S. B., & Song, B. (2012). A lidar-based decision-making method for road boundary detection using multiple Kalman filters. IEEE Transactions on Industrial Electronics, 59(11), 4360-4368.

20) Negenborn, R. (2003). Robot localization and Kalman filters. Utrecht Univ., Utrecht, Netherlands, Master's thesis INF/SCR-0309.

21) Al Khatib, E. I., Jaradat, M. A. K., & Abdel-Hafez, M. F. (2020). Low-cost reduced navigation system for mobile robot in indoor/outdoor environments. IEEE Access, 8, 25014-25026.

22) Elnagar, A. (2001, July). Prediction of moving objects in dynamic environments using Kalman filters. In Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (Cat. No. 01EX515) (pp. 414-419). IEEE.

23) Mahmud, M. A., Aman, M. S., Jiang, H., Abdelgawad, A., & Yelamarthi, K. (2016, March). Kalman filter based indoor mobile robot navigation. In 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 1949-1953). IEEE.

24) Budiharto, W., Santoso, A., Purwanto, D., & Jazidie, A. (2011, October). A navigation system for service robot using stereo vision and Kalman filtering. In 2011 11th International Conference on Control, Automation and Systems (pp. 1771-1776). IEEE.

25) Smith, C. M., Feder, H. J. S., & Leonard, J. J. (1998, December). Multiple target tracking with navigation uncertainty. In Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171) (Vol. 1, pp. 760-761). IEEE.

26) Baras, N., Nantzios, G., Ziouzios, D., & Dasygenis, M. (2019, May). Autonomous obstacle avoidance vehicle using lidar and an embedded system. In 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST) (pp. 1-4). IEEE.

27) Rozsa, Z., & Sziranyi, T. (2018). Obstacle prediction for automated guided vehicles based on point clouds measured by a tilted LIDAR sensor. IEEE Transactions on Intelligent Transportation Systems, 19(8), 2708-2720.

28) Ahmad, H., Peeie, M. H., Ramli, M. S., Shafie, A. A. B., & Rahiman, M. H. F. (2021, November). Investigation of ROS Based Environment Modelling and Mobile Robot Position Estimation with Dead Reckoning and Uncertainties. In *2021 IEEE Industrial Electronics and Applications Conference (IEACon)* (pp. 19-24). IEEE.

29) Casanova, O. L. (2008). Robot Position Tracking Using Kalman Filter.

APPENDIX A
SAMPLE APPENDIX 1

Gantt Chart

No	Title	Week													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Literature Review														
2	Hardware development														
3	Personal computer setup														
4	Simulation on MATLAB														
5	Simulation environment of Turlebot3 in Gazebo														
6	G-mapping on RVIZ														
7	Hardware assembly														
8	Python nodes developing and testing														
9	Bringup operation														
10	Data analysis														
11	Thesis writing														
12	Submission slide for PSM 2 Ekselen														
13	PSM 2 Ekselen														
14	PSM 2 thesis and logbook submission														