

**INTEGRATION OF MAPPING AND NEURAL NETWORK FOR
MOBILE ROBOT WITH LASER DISTANCE RANGE SENSOR
(LDS)**

NUR FATINI BINTI KAMARUZAMAN @KAMARULDIN

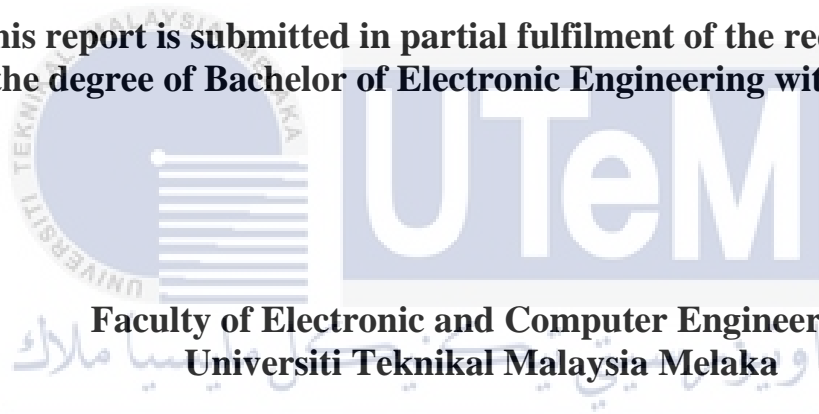


UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**INTEGRATION OF MAPPING AND NEURAL NETWORK FOR
MOBILE ROBOT WITH LASER DISTANCE RANGE SENSOR
(LDS)**

NUR FATINI BINTI KAMARUZAMAN @KAMARULDIN

**This report is submitted in partial fulfilment of the requirements
for the degree of Bachelor of Electronic Engineering with Honours**



**Faculty of Electronic and Computer Engineering
Universiti Teknikal Malaysia Melaka**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2021

DECLARATION

I declare that this report entitled “**Integration of Mapping and Neural Network for Mobile Robot with Laser Distance Range Sensor (LDS)**” is the result of my own work except for quotes as cited in the references.



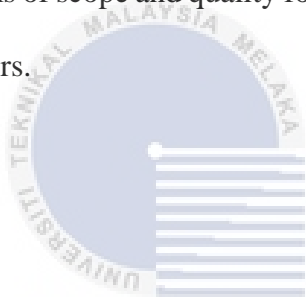
Signature : اونیورسیتی تکنیکل مالایا ملاک

Author : Nur Fatini Binti Kamaruzaman @Kamaruldin

Date : 24 June 2021

APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours.



اونيورسيتي تيكنيكل مليسيا ملاك

Signature : _____

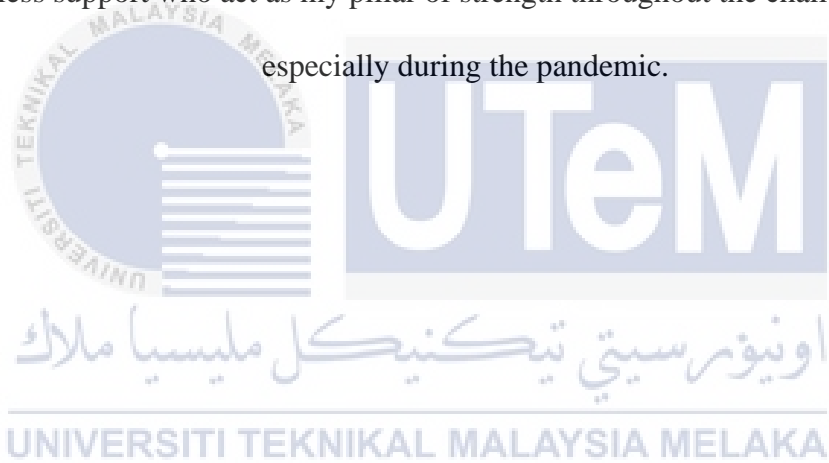
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Supervisor Name : Dr. Norhidayah Mohamad Yatim
.....

Date : 25 June 2021
.....

DEDICATION

I would like to thank my family members, my supervisor, and friends for their endless support who act as my pillar of strength throughout the challenging time especially during the pandemic.



ABSTRACT

Mobile robots are becoming increasingly popular. They have already moved out of laboratories. For instance, they are widely used not only in industry but also at home. The ability to map the surrounding environment is one of the most common tasks in autonomous navigation for a mobile robot. However, the problem is the accuracy of the build map depends on sensor measurements. Thus, noisy sensor measurements would produce a grid map that prone to error when using low-cost sensors. Therefore, machine learning algorithm integration using a neural network approach has been introduced to improve the quality of the map created and to evaluate the accuracy of the environmental grid map using a low-cost laser distance sensor (LDS). In this work, the neural network and the occupancy grid map algorithm will be experimented using the Turtlebot3 mobile robot. The process of conducting project is to collect sensor measurements, train the neural network, and test a neural network to build an occupancy grid map. Throughout the project, the software that will be used are Ubuntu, ROS, Gazebo, RViz while the hardware is Turtlebot3 and LDS sensor. At the end of the project, a mobile robot able to build a high-accuracy environmental map.

ABSTRAK

Robot mudah alih menjadi semakin popular. Mereka sudah berpindah dari makmal. Sebagai contoh, mereka digunakan secara meluas bukan hanya di industri tetapi juga di rumah. Keupayaan untuk memetakan persekitaran di sekitarnya adalah salah satu tugas yang paling biasa dalam navigasi autonomi untuk robot bergerak. Namun, masalahnya adalah ketepatan peta binaan bergantung pada pengukuran sensor. Oleh itu, pengukuran sensor yang bising akan menghasilkan peta grid yang terdedah kepada kesalahan ketika menggunakan sensor kos rendah. Oleh itu, integrasi algoritma pembelajaran mesin menggunakan pendekatan rangkaian neural telah diperkenalkan untuk meningkatkan kualiti peta yang dibuat dan untuk menilai ketepatan peta grid persekitaran dengan menggunakan sensor jarak laser kos rendah (LDS-01). Dalam karya ini, rangkaian saraf dan algoritma peta grid penghunian akan dieksperimen menggunakan robot mudah alih Turtlebot3. Proses menjalankan projek adalah untuk mengumpulkan pengukuran sensor, melatih jaringan saraf, dan menguji jaringan saraf untuk membangun peta grid penghunian. Sepanjang projek ini, perisian yang akan digunakan adalah Ubuntu, ROS, Gazebo, Rviz sementara perkakasannya adalah sensor Turtlebot3 dan LDS-01. Pada akhir projek, robot bergerak dapat membina peta persekitaran dengan ketepatan tinggi.

ACKNOWLEDGEMENTS

First of all, I am grateful to Allah for allowing me to finally complete this thesis because I believe I managed to finish this project is due to His endless blessing in helping me to go through this journey.

I would like to give appreciation and thank my supervisor, Dr. Norhidayah Binti Mohamad Yatim for her continuous guide and support for me while doing this project. I thank her for always making time and full of enthusiasm in assisting me in writing the thesis.

Furthermore, I would also have to appreciate the guidance given by panels who share their expertise and recommendation during the project presentation and give me the right direction toward the completion of my project.

I would like to give honor and gratitude to my parents for their endless support, encouragement and motivation in doing this project as I can never be where I am not without their support and love. Last but not least, I would like to give appreciation to everyone who always being beside me and motivate me to go through this bitter-sweet journey.

TABLE OF CONTENTS

| | |
|--|------------|
| Declaration | |
| Approval | |
| Dedication | |
| Abstract | i |
| Abstrak | ii |
| Acknowledgements | iii |
| Table of Contents | iv |
| List of Figures | vii |
| List of Tables | ix |
| List of Symbols and Abbreviations | x |
| List of Appendices | xi |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Project Background | 2 |
| 1.2 Problem Statements | 2 |
| 1.3 Objectives | 3 |
| 1.4 Scope of Work | 3 |

| | | |
|-----------------------------------|--|-----------|
| 1.5 | Significant of Project | 3 |
| 1.6 | Thesis Outline | 4 |
| CHAPTER 2 BACKGROUND STUDY | | 5 |
| 2.1 | Mapping | 6 |
| 2.2 | Robot Platforms | 6 |
| 2.2.1 | Type of Robot | 6 |
| 2.2.2 | Characteristic of Sensor | 8 |
| 2.3 | Machine Learning Algorithm | 9 |
| 2.4 | Research Gap | 10 |
| 2.5 | Summary | 10 |
| CHAPTER 3 METHODOLOGY | | 12 |
| 3.1 | Introduction to ROS | 14 |
| 3.2 | Simulation and Hardware Setup | 16 |
| 3.2.1 | Robot Platform | 16 |
| 3.2.2 | Single Board Computer (SBC) Setup | 18 |
| 3.2.3 | 360 Laser Distance Sensor (LDS-01) | 20 |
| 3.3 | Occupancy Grid Map | 23 |
| 3.3.1 | Data Preparation | 24 |
| 3.3.2 | Map Updating | 25 |
| 3.4 | Correction of Nonlinearity using Neural Networks | 26 |

| | | |
|--|--|-----------|
| 3.4.1 | Accuracy | 27 |
| CHAPTER 4 RESULTS AND DISCUSSION | | 29 |
| 4.1 | Build Test Environment | 30 |
| 4.1.1 | The environment in Gazebo Simulator | 30 |
| 4.2 | Data Measurement of Sensor | 33 |
| 4.2.1 | Input and Output Data for Training and Testing | 33 |
| 4.2.2 | Nonlinearity of Sensor Model | 35 |
| 4.3 | Machine Learning Algorithm | 38 |
| 4.3.1 | Training and Testing Data Sensor Measurement | 38 |
| 4.4 | Accuracy of Training | 40 |
| 4.4.1 | Root Mean Square Error | 40 |
| CHAPTER 5 CONCLUSION AND FUTURE WORKS | | 41 |
| 5.1 | Conclusion | 42 |
| 5.2 | Future Works | 43 |
| REFERENCES | | 44 |
| APPENDICES | | 46 |

LIST OF FIGURES

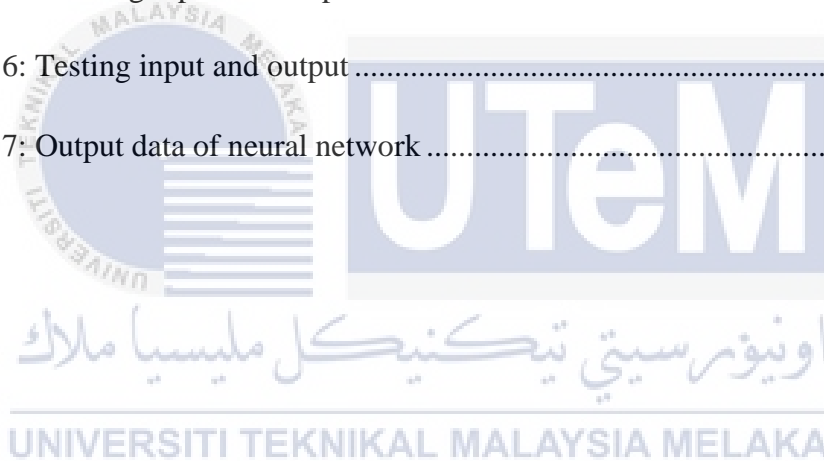
| | |
|---|----|
| Figure 3.1: Flow Chart of Project | 13 |
| Figure 3.2: An illustration of the hierarchical structure of ROS | 14 |
| Figure 3.3: The communication process between nodes is 1-2-3-4 | 15 |
| Figure 3.4: TurtleBot3 Burger | 17 |
| Figure 3.5: Raspberry Pi 3 B+ | 18 |
| Figure 3.6: OpenCR | 19 |
| Figure 3.7: Laser Distance Range Sensor (LDS-01) | 20 |
| Figure 3.8: Polar Graph | 22 |
| Figure 3.9: Measuring the distance by adjusting the position of mobile robot | 24 |
| Figure 3.10: The position of mobile robot to the cylinder in Gazebo simulator | 24 |
| Figure 3.11: Neural network input-output configuration | 27 |
| Figure 4.1: Commands to build test environment in Gazebo | 30 |
| Figure 4.2: Test environment in Gazebo simulator | 30 |
| Figure 4.3: Scanning area | 31 |
| Figure 4.4: Occupancy grid map | 31 |
| Figure 4.5: The rqt graph | 32 |
| Figure 4.6: Nonlinearity of the sensor model (0.2m) | 35 |
| Figure 4.7: Nonlinearity of the sensor model (0.4m) | 36 |

| | |
|--|----|
| Figure 4.8: Nonlinearity of the sensor model (0.6m) | 36 |
| Figure 4.9: Nonlinearity of the sensor model (0.8m) | 37 |
| Figure 4.10: Nonlinearity of the sensor model (1.0m) | 37 |
| Figure 4.11: Neural network with random weight | 38 |
| Figure 4.12: Neural network with weight after training | 39 |



LIST OF TABLES

| | |
|---|----|
| Table 1: Summary of Literature Review..... | 11 |
| Table 2: General Specifications LDS-01 | 21 |
| Table 3: Measurement Performance Specifications LDS-01..... | 21 |
| Table 5: Training input and output | 33 |
| Table 6: Testing input and output..... | 34 |
| Table 7: Output data of neural network..... | 40 |



LIST OF SYMBOLS AND ABBREVIATIONS

- ROS : Robot Operating System
- SLAM : Simultaneous Localization and Mapping
- LDS : Laser Distance Range Sensor
- RMSE : Root Mean Square Error



LIST OF APPENDICES

| | |
|---|----|
| Appendix A: Python codes for Neural Network | 46 |
| Appendix B: Output data of Neural Network | 48 |
| Appendix C: Project planning | 53 |
| Appendix D: Cost of project | 54 |



CHAPTER 1

INTRODUCTION



This chapter explains the introduction of the project for background, problem statements, and objectives of this project. Besides, the scope of work and structure of the thesis is also included in this section.

1.1 Project Background

The mobile robot is one of the most recent research topics that gained popularity and focus on either academic or commercial researchers. In the era of modernization, many people are busy working and their activities are also limited. Do you ever believe that a mobile robot can help people in their work? Since people's activity is limited and the mobile robot has a broad range of applications, it can help them such as cleaning, and mowing. In mobile robot applications, apart from the variety of sizes, prices, and functionalities, there are three main tasks or common problems that have to be solved for a robot to learn the environment which are localization, mapping, and path planning.

One of the challenging parts of robotics is mapping the environment of the mobile robot with high accuracy. In the consumer market, they must be considered cheap solutions targeting. Typically, mobile robots only have simple features, because of the poor accuracy of the low-cost sensors. However, this issue can be overcome using the machine-learning algorithm approaches to artificial intelligence while maintaining the use of low-cost sensors. Thus, the accuracy of the mapping algorithm will be increased by using the machine learning method which can reduce the noisy characteristic of a low-cost sensor.

1.2 Problem Statements

The quality of the map depends on the characteristics of the sensor. From previous research, different ranges and a low-cost sensor for mapping were evaluated. There is an error between the actual distance and the observed distance, mostly due to nonlinearity and a random error in the readings for those type of sensors used. This research work focuses on mapping problems by using a low-cost sensor. The noisy

sensor measurement will be used to create a grid map that is vulnerable to error when using high variance sensors.

1.3 Objectives

1. To propose neural network configurations that interpret noisy sensor measurement into the grid map.
2. To analyze the accuracy of the grid map using a low-cost sensor.

1.4 Scope of Work

To ensure that the analysis is carried out within its expected boundary, multiple scopes are identified. This project focuses on the mapping mobile robot, TurtleBot3 Burger using laser distance sensor-low cost which is 360 Laser Distance Sensor-01 (LDS-01) that capable of sensing 360 degrees and its maximum distance measurement is up to 3500mm. In order to map the indoor environment, an occupancy grid map is used and Neural Network had been selected using lower-cost sensor features to increase the efficiency of map construction.

1.5 Significant of Project

This project is environmentally friendly because laser scanning technology promises faster-scanning speed, longer ranges, and better precision and resolution. Besides, the use of low-cost sensor directly will be able to reduce total cost of mobile robots. The task performed using a mobile robot more quickly compared to the manual method. Mapping robots can be applied to indoor environments such as libraries, office buildings, and restaurants. These innovations lead to new possibilities concerning the performance of work, including flexible working time and flexible

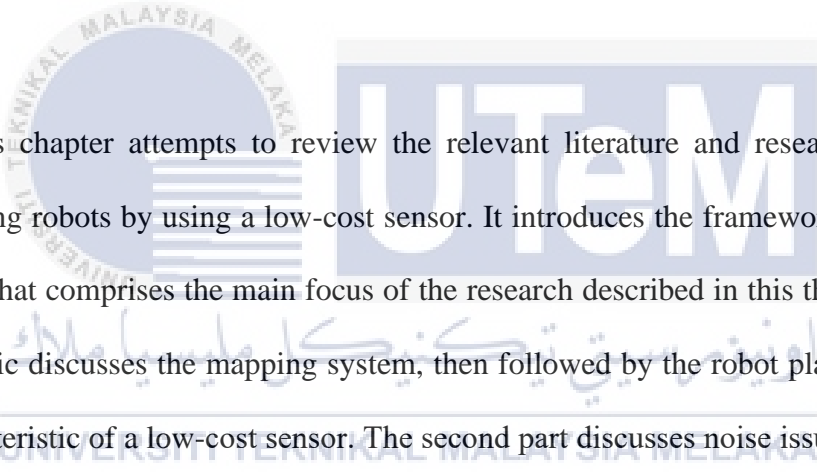
working areas. Besides, the task can also be done by robotics platforms with higher number of repetitions compared to human capability, who can concentrate only on more creative work. Therefore, it is necessary to use mobile robot devices anywhere and at any time. This project also uses open-source software such as Ubuntu and Robot Operating System (ROS) which contribute to open-source community. Open-source community is a better option because it is from community to community for free. Thus, the project is more sustainable, long lasting and not depends on expensive software.

1.6 Thesis Outline

The thesis is organized and separated into five major chapters. In Chapter 1, the overview of robot mapping using a low-cost sensor and machine learning algorithm is discussed in the project background. Moreover, the problem statement, objective, and scope of the project are outlined clearly in this section as well. Chapter 2 discusses the past studies related to robot mapping using a low-cost sensor and machine learning algorithm. In Chapter 3, all relevant experiments and techniques used in the project are explained. In Chapter 4, the results of the project are presented and interpreted in this section. The collected data are analyzed carefully to verify whether the objectives have been achieved or not. In the final chapter, a conclusion is drawn from the project. Besides, the recommendation for future work related to the project is made in this section.

CHAPTER 2

BACKGROUND STUDY



This chapter attempts to review the relevant literature and research related to mapping robots by using a low-cost sensor. It introduces the framework for the case study that comprises the main focus of the research described in this thesis. The first subtopic discusses the mapping system, then followed by the robot platform and the characteristic of a low-cost sensor. The second part discusses noise issues in sensors' measurements. The third part is focused on the identification of the machine learning algorithm to interpret the noise issues.

2.1 Mapping

Mapping the environment of robots has always been a challenge to researchers and several researchers have previously accomplished this. In those works, most of them used an occupancy grid map representation. Occupancy-grid map is one of the most popular mapping approaches in mobile robotics and it was first introduced by Moravec and Elfes (1985). Occupancy grid mapping belongs to the class of probabilistic robotics computer algorithms for mobile robots that address the issue of developing maps from noisy and unknown sensor measurement data, assuming that the location of the robot is known. Therefore, the occupancy grid map is more flexible due to no assumption on the features of the environment and no data association needed for landmark identification. However, the drawback is it has high memory consumption. Grid-based map representation is a metric map, where the map of the environment is divided into grid cells. Each cell of the occupancy grid map has a probability value showing its state (filled or empty) which depends on the measured distance. In order to update the map, the Bayes filter is used in the occupancy grid map [1], [5]. The Bayes filter is a general method of probabilistic estimation of uncertain probability. It is a recursive technique that uses the latest measurement and former estimation to estimate the current state.

2.2 Robot Platforms

2.2.1 Type of Robot

Nowadays a broad variety of handheld robotic systems can be seen on the market and in robotic laboratories worldwide. A lot of researchers have used some of them. Most of the researchers chose to use a small robot that capable of moving through narrow corridors for mapping the environment. Among the small indoor robots, the

Khepera III is one platform that has achieved widespread acceptance [6]. Based on literature studies, researchers have experimented with using Khepera III [5]. Released in 2006, Khepera III has sold more than 600 items to 150 universities worldwide and has been used for hundreds of publications. Khepera III meets the requirements of being compact with a diameter of approximately 12 cm and is fitted with two low-cost sensor types which are five ultrasonic sensors and nine infrared sensors. The newer version of the Khepera IV robot developed by K-Team1 launched in January 2015, the successor to Khepera III [7]. Khepera IV is a 14 cm diameter differential wheeled mobile robot. It is equipped with 12 infrared sensors, five ultrasound sensors, two microphones, and a camera. However, this type of robot requires high costs. The authors in the literature [1] used the KOMA mobile robot, which wheel type self-contained. It has rotary encoders to detect the angular velocity of the wheel rotation. Both wheels are driven by 20W DC motors. Attached to the front of the body is a PB9-01 and an ultrasonic sensor array to recognise the environment. The KOMA controller consists of two components: the main controller for navigation, motion control and positioning, and the auxiliary controller for assisting the main controller measure the amount of data in a complicated way.

2.2.2 Characteristic of Sensor

Developing a high-precision environmental map tends to be one of the most challenging tasks in the field of mobile robots. Therefore, analyze the characteristic of low-cost sensors is crucial because it is closely related to getting a high-quality map. Previous studies have shown, one of the most commonly used for distance measurement is the infrared sensor. It can be used in robotic to prevent obstacles. Furthermore, it has a cheaper cost and faster time response rather than an ultrasonic sensor [8]. Using Khepera III mobile robot in Webots robot simulators, there are two types of low-cost sensors which are ultrasonic sensors and infrared sensors. The array of infrared sensors is used in the research work [5] because the infrared sensor has a smaller beam characteristic compared to the sound wave by the ultrasonic sensor. According to the physical structure of the robot, the array of infrared sensors is not equally separate. The infrared sensor measurement in the Webots simulator depends on the reflectance on the colour properties of the object's surface. It has a maximum range of 30 cm. They adjusted the position of a white plate in front of the sensor installed on the mobile robot to identify the characteristics of the sensor. Measurements for distances from 0 to 30 cm were taken every 1 cm away from the sensor. The robot requires several loops in the test environment to achieve optimum observation due to the limited distance of the infrared sensor.

In the research work [1], the sensor used is an infrared range finder, PB9-01 manufactured by HOKUYO AUTOMATIC CO., LTD, which uses an infrared LED modulated at 870nm to generate signals. This sensor has a compact size and lightweight, but it also provides distant and directional information to an object for one-time scanning. While it is not recommended for use in public spaces due to its limited range of detection and sunlight interference. A series of tests were carried out by measuring

the distance while switching the location of the plate in front of a mobile robot with PB9-01 in the range of 10 to 300 cm, similar to the previous study. However, all these sensors have non-linear characteristics, often referred to as measurement noise and they depend on the reflectance properties of the object surfaces. In other words, how the surface scatters, reflects, and absorbs infrared energy is required to interpret the performance of the sensor as a distance measure [9]. More significantly, most robot sensors are subject to strict range limitations. Nevertheless, the nonlinearity error and a random error of reading need to be corrected by using machine learning approaches.

2.3 Machine Learning Algorithm

Machine learning is the study of computer algorithms that improve automatically through experience (Tom & Mitchell, 1998). Regarding the previous section, it can be seen the problem of construct a high precision of an environmental map for a mobile robot is extremely related to the characteristic of the low-cost sensor such as infrared range finder sensor [1], and infrared sensor [5]. It happens due to having significant noise in their sensors' measurements. Consequently, the machine learning method is beneficial for the analysis of noise sensor measurements and it has been proven by some researchers. To generate a more precise map, a neural network learner [1] was used with an infrared rangefinder. Apart from the Neural Network, there are also several algorithms built in the machine learning field. For example, in research work [5], Naïve Bayes, Decision Tree, AdaBoost, and Neural Network learners were experimented with to learn the sensor input. The resulting map obtained by the Neural Network has much more precision than the other three learners, after comparing the output of each learner who integrates with the occupancy grid map algorithm.

Furthermore, a map in multiple robot simultaneous localization and mapping (SLAM) also needs to be learned, thus neural networks were a successful solution [10]. Upon learning a map, due to the extremely reduced dimensionality, any integration with other maps would be much quicker and easier. Therefore, it can be concluded that neural network algorithms are efficient approaches to artificial intelligence that can be implemented for any learning or training issue.

2.4 Research Gap

By conducting this literature review, we see that most of the literature on robot mapping above focuses mainly on the aspect of low-cost sensor issues that may affect the quality of maps.

2.5 Summary

This chapter has reviewed all the relevant theoretical literature used in the study of robot mapping with low-cost sensors include reviews of the machine learning method for interpreting noisy sensor measurements. These reviews led to the development of a conceptual framework. The following chapter discusses hypotheses development and measurement development based on the proposed conceptual model.

Table 1: Summary of Literature Review

| TITLE, YEAR, AND AUTHOR | MAPPING | ROBOT PLATFORM | LOW-COST SENSOR | MACHINE LEARNING |
|--|---|---------------------------|--|---|
| Environmental map building for a mobile robot using infrared range-finder sensors (2004) Y. S. Ha and H. H. Kim [1] | <ul style="list-style-type: none"> • Occupancy grid map. • Bayes filter used to update the map. | KOMA mobile robot | <ul style="list-style-type: none"> • An infrared range finder sensor. • The distance range is 10 to 300 cm. | Neural network learner to produce a more accurate map. |
| Indoor mapping with machine learning algorithm using Khepera III mobile robot (2016) N. M. Yatim and N. Buniyamin [5] | <ul style="list-style-type: none"> • Occupancy grid map. • Bayes filter used to update the map. | Khepera III mobile robot. | <ul style="list-style-type: none"> • Consist of an ultrasonic and infrared sensor • An array of infrared sensors is used. • The maximum range of 30 cm. | <ul style="list-style-type: none"> • Naïve Bayes, Decision Tree, AdaBoost, and Neural Network • Neural Network has more accuracy. |
| Using Occupancy Grids for Mobile Robot Perception and Navigation (1989) A. Elfes [2] | Occupancy grid map is one of the most popular mapping approaches in mobile robotics. | | | |

CHAPTER 3

METHODOLOGY



This chapter will explain the technique or implementation adopted by this research. The instrument such as a sensor that was used for data collection is also described and the procedures that were followed to carry out this study are included. The procedure and methods for completing the project are illustrated clearly in the charts. Each stage of the level chart will be discussed in detail.

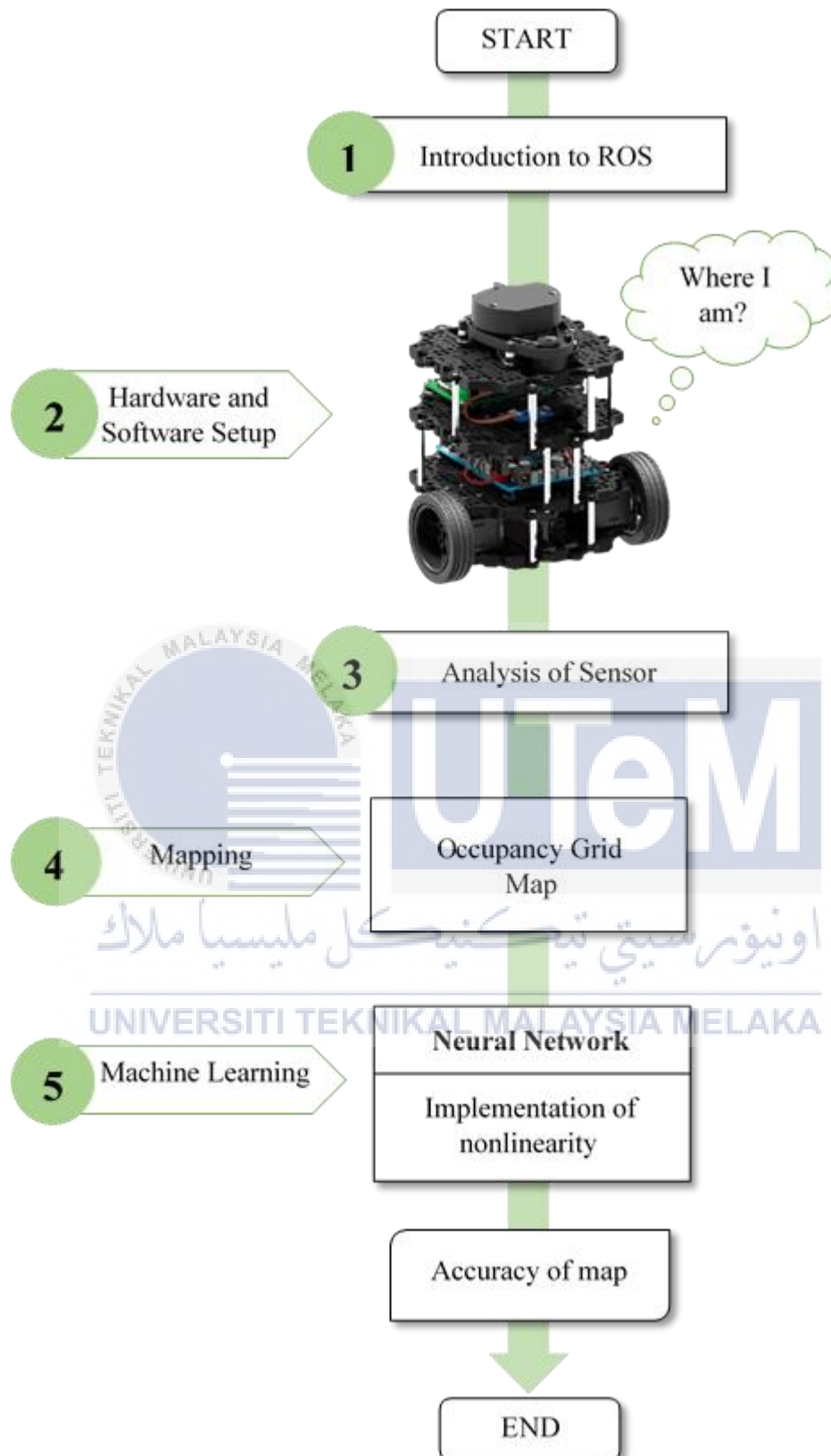


Figure 3.1: Flow Chart of Project

3.1 Introduction to ROS

Robot operating system (ROS) is a free and open-source framework that shows particular usefulness in the robotics and automation fields around the world. ROS is an open-source development framework designed to support another generation of individual robots. In a human environment, individual robots travel about and interfere with a person's use of the same objects. These robots are called administration robots or multipurpose controllers that are often switched off and on again. Instead of designing a whole program for all functions, the benefit of ROS allows users to design each function as a node and run these nodes simultaneously. Figure 3.2 shows a ROS framework is as an algorithm diagram comprising of a set of nodes corresponding with each other. ROS provides instruments for evaluating the diagram and testing what is the subject said by node Figure 3.3 shows the communication process between nodes.

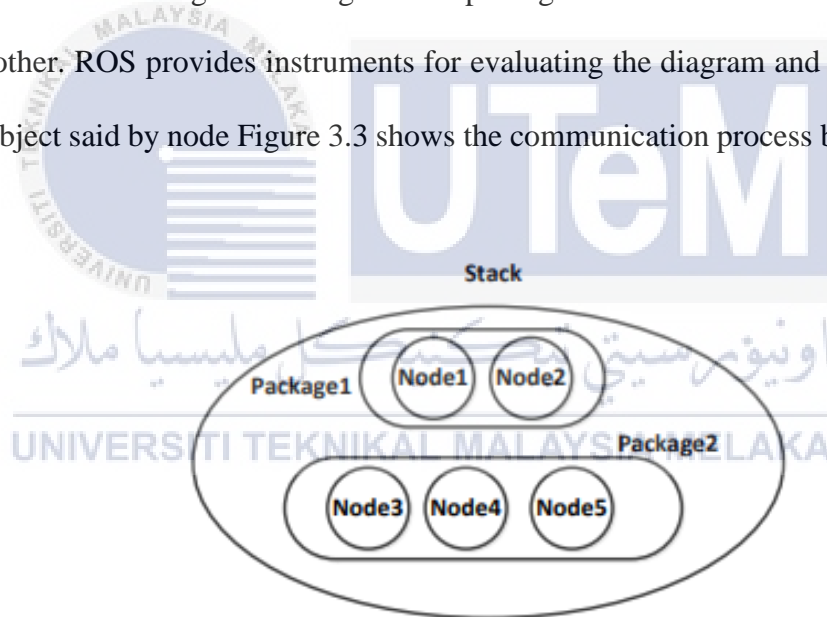


Figure 3.2: An illustration of the hierarchical structure of ROS

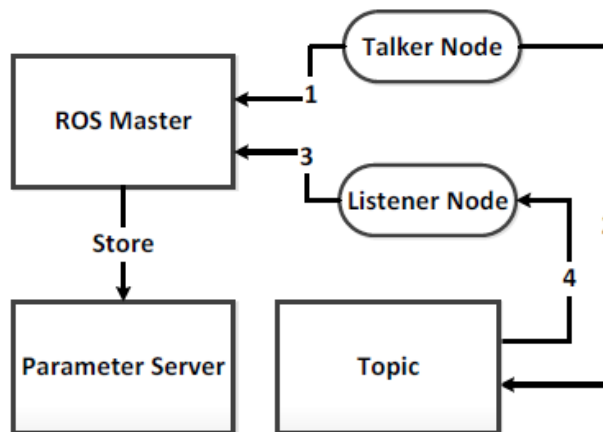


Figure 3.3: The communication process between nodes is 1-2-3-4

The ROS Master provides the rest of the nodes with name registration and lookup. Without a ROS Master, Talker Node (1) and Listener Node (3) would not be able to locate each other, exchange messages, or invoke services. By connecting with the master, other remote nodes will locate each other while running the master on one device.

Each message in ROS is transported using named buses called topic. When a Talker Node sends a message through a topic (2), the node publishing the topic. When a Listener Node receives a message through a topic (4), the node subscribing to a topic.

In ROS, configuration information is normally stored on the Parameter server. The server parameter is a list of values that can be accessed through the command prompt, nodes, or launch files upon request. Parameters are meant to be fairly static variables, such as integers, floats, strings, or bool values that are globally accessible.

To build an autonomous mobile robot in this project, ROS is the most preferred over the robotic platforms such as Player, YARP, and so on. It is due to high-end capabilities, inter-platform, and support for the high-end sensors, and actuators. ROS

also is packed with a ton of tools for debugging, visualizing, and performing a simulation.

In the simulation, the tools used such as Gazebo and Rviz which is for developing the test area in Gazebo through the `gazebo_ros` package and the robot runs randomly while collecting the data respectively. ROS program contains python file configuration, compilation files, simulation files, launch files, and parameter files. The main language used for writing the ROS coding is Python and C++.

3.2 Simulation and Hardware Setup

3.2.1 Robot Platform

The indoor mobile robot used in this study is the TurtleBot3 Burger. TurtleBot3 is a new generation mobile robot that was released in May 2017 and it is made up of modular plates that users can customize the shape and compact. There are three types of TurtleBot3 which are Burger, Waffle, and Waffle Pi. However, TurtleBot3 Burger is selected as a mobile robot platform in this project because it satisfies the criteria of being small in size compared to others and has lower the price of the platform without scarifying capability, functionality, and quality. Other than that, it is equipped with a low-cost sensor (a 360 degrees LIDAR), TurtleBot3 consists of a base, two Dynamixel motors for driving, a 1,800mAh battery pack, and a hardware mounting kit attaching everything.

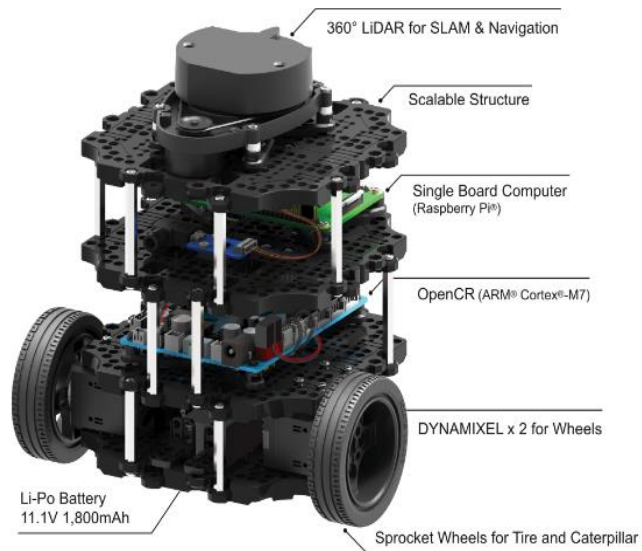


Figure 3.4: TurtleBot3 Burger



3.2.2 Single Board Computer (SBC) Setup

There are several types of Raspberry Pi. One of the electronic components attached to the TurtleBot3 Burger platform is Raspberry Pi 3 B+. The main function of the raspberry pi is to connect nodes to the subscriber or publishing the messages to the laptop in order to implement ROS for mapping and navigation. Firstly, prepare the microSD card which is to burn the TurtleBot3 Raspbian image using Ubuntu in order to use the raspberry pi. To boot up the Raspberry Pi, connect the HDMI port of raspberry pi to the laptop HDMI port, and connect the input devices to the USB port of Raspberry Pi. Next, insert the microSD card that has been prepared into the Raspberry Pi SD card slot and it will turn on once it is connected to the power with USB or OpenCR. After the Raspbian OS is running, connect the raspberry pi to the router network that is connected to the PC. Through the PC, the assigned IP address of Raspberry Pi can be found with the `$ ifconfig` and `$ ssh pi@{IP_ADDRESS_OF_RASPBERRY_PI}`.

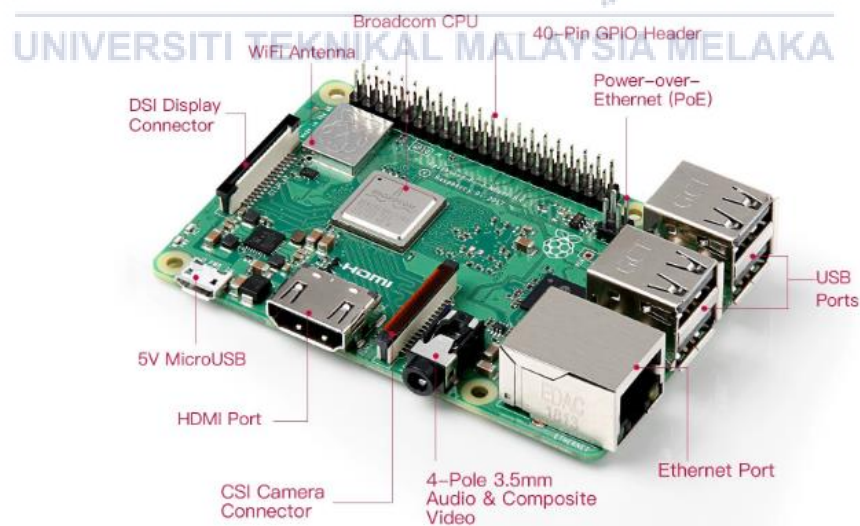


Figure 3.5: Raspberry Pi 3 B+

The OpenCR for the ROS is well suited. This is because it has a very large main chip (ARM Cortex-M7 with floating point unit) and both digital and analog input/output pins (UART, SPI, I2C). Consequently, it helps to monitor and run robots in precise periods with various sensor data. Apart from that, it also supports power output at 3.3V, 5V, and 12V. Therefore, it is so flexible for users to use any power output for the robot and sensor, and helps to operate of SBC. Third, using ROS Serial, it may publish or subscribe to topics. This key feature allows ROS users to quickly use the ROS library for robots.

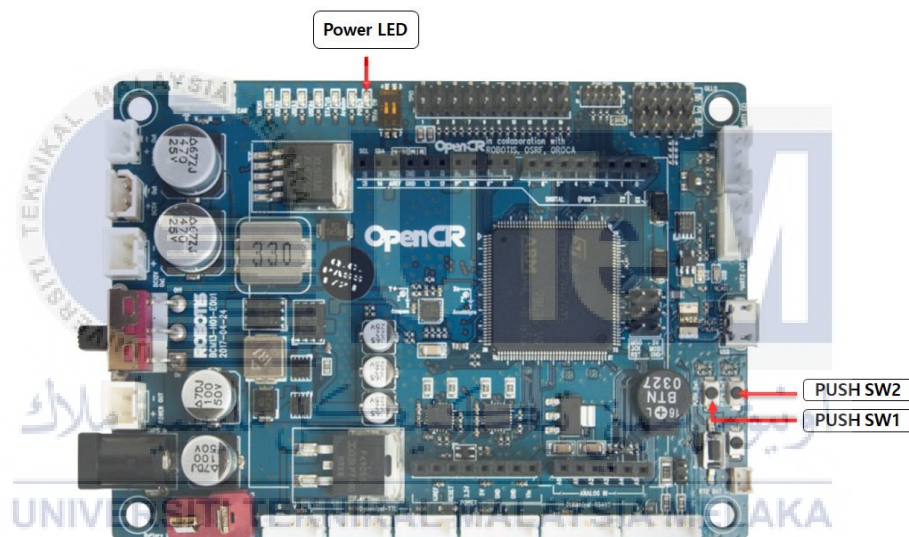


Figure 3.6: OpenCR

3.2.3 360 Laser Distance Sensor (LDS-01)

3.2.3.1 Structure and Specifications

The sensor adopted for this project is a laser distance range sensor (LDS-01) as shown in Figure 3.7. Objects located between 120 to 3,500 mm away can be measured by this 2D laser distance sensor. It is mounted on a rotating head, allowing it to map its environment at 360° and collects a set of data around the robot to use for SLAM. A technology commonly used in embedded mobile robotics is SLAM, which stands for simultaneous localization and mapping. Therefore, it is possible to mount a SLAM distance sensor such as the LDS-01 on a robotics platform capable of avoiding obstacles or exploring its environment and a robotic monitoring device. This sensor also high precision and USB interface makes it a perfect attachment for robotics projects, enabling easy installation on a device. It also has a complete ROS that allows you to benefit from all its features to the fullest. Table 2 and Table 3 give the details of its specifications.



Figure 3.7: Laser Distance Range Sensor (LDS-01)

Table 2: General Specifications LDS-01

| Items | Specifications |
|--------------------------|--|
| Operating supply voltage | 5V DC $\pm 5\%$ |
| Light source | Semiconductor Laser Diode($\lambda=785\text{nm}$) |
| LASER safety | IEC60825-1 Class 1 |
| Current consumption | 400mA or less (Rush current 1A) |
| Detection distance | 120mm ~ 3,500mm |
| Interface | 3.3V USART (230,400 bps) 42bytes per 6 degrees, Full Duplex option |
| Ambient Light Resistance | 10,000 lux or less |
| Sampling Rate | 1.8kHz |
| Dimensions | 69.5(W) X 95.5(D) X 39.5(H)mm |
| Mass | Under 125g |

Table 3: Measurement Performance Specifications LDS-01

| Items | Specifications |
|---|-------------------|
| Distance Range | 120 ~ 3,500mm |
| Distance Accuracy (120mm ~ 499mm) | $\pm 15\text{mm}$ |
| Distance Accuracy (500mm ~ 3,500mm) | $\pm 5.0\%$ |
| Distance Precision (120mm ~ 499mm) | $\pm 10\text{mm}$ |
| Distance Precision (500mm ~ 3,500mm) | $\pm 3.5\%$ |
| Scan Rate | 300 \pm 10 rpm |
| Angular Range | 360 $^\circ$ |
| Angular Resolution | 1 $^\circ$ |

3.2.3.2 Analysis of Sensor Characteristics

In the occupancy grid map algorithm, the quality of a map is closely related to the sensor's characteristics. Therefore, it is necessary to analyze the characteristic of the sensor based on data obtained through experiments to enhance the accuracy of the environmental map. However, before analyzing the sensor characteristic, the first thing that needs to be done is testing the LDS-01 for the functionality of the sensor and the distance range that the sensor can sense. To do that, the command uses as shown below:

1. For installing the LDS-01:

```
$ sudo apt-get install ros-kinetic-hls-1fcd-lds-driver
```

2. Run hlds_laser_publisher Node:

```
$ roslaunch hls_1fcd_lds_driver hlds_laser.launch
```

3. Run hlds_laser_publisher Node with Rviz:

```
$ roslaunch hls_1fcd_lds_driver view_hlds_laser.launch
```

When run the driver of LDS-01 with a command (`./lds_driver`), the raw data will appear in the terminal (Ubuntu) and built Qt Creator to get LDS-01 graph in the form of a polar graph.

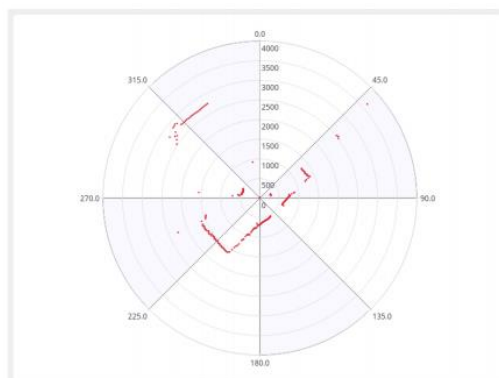
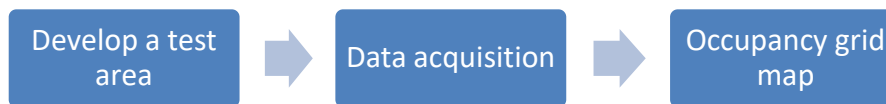


Figure 3.8: Polar Graph

3.3 Occupancy Grid Map



The gmapping package is an algorithm named SLAM, which uses laser scan data and odometry to generate a 2D occupancy grid map.

1) Build a map using SLAM

The ROS Gmapping package is an open-source SLAM implementation extension called OpenSLAM. A node named `slam_gmapping`, which is the implementation of SLAM is included in the package and helps create a 2D occupancy grid map from the laser scan data and the pose of the mobile robot. In this project, LDS-01 is the basic hardware requirement for doing SLAM which is horizontally mounted on the top of the robot. Then, it can generate the 2D map of the environment by using the gmapping package in ROS. Before start operating with Gmapping, there need to install that package using the command below:

```
$ sudo apt-get install ros-kinetic-gmapping
```

2) Create a launch file for gmapping

The main task is to set the parameters for the `slam_gmapping` node and the `move_base` node while creating a launch file for gmapping. Inside the ROS Gmapping package, there consist of the `slam_gmapping` node as the core node. From the `slam_gmapping` node, it subscribes to the sensor data. Finally, it publishes the occupancy grip map as output (`nav_msgs/OccupancyGrid`). This node is highly configurable and to enhance the mapping precision, we can fine-tune the parameters.

3.3.1 Data Preparation

The actual distance (output target) between the mobile robot and the cylinder was adjusted in the range of 0.2m to 1 m as shown in Figure 3.9. Distance by the sensor measured of 185 was collected for each space of 0.2m and those were averaged. Since the sensor senses the environment at 360° and gives the measurements at each angle, therefore the python code is created and executed and measurements were taken only at 0° which is towards the cylinder.

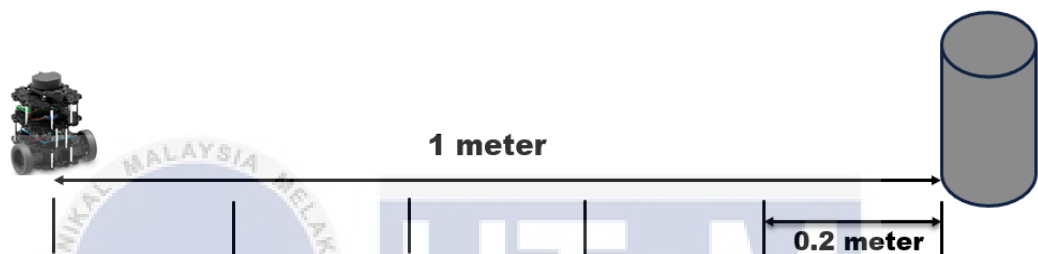


Figure 3.9: Measuring the distance by adjusting the position of mobile robot



Figure 3.10: The position of mobile robot to the cylinder in Gazebo simulator

3.3.2 Map Updating

To update the occupancy grid map, the Bayesian filter is used. Here, the probability form has been converted for computational efficiency into the log odds notation defined in the following Equation 3.1. Note that the $p(m_i|z_t, x_t)$ probability is derived from the machine learning algorithm and uses as a probability value.

$$l(x) = \log\left(\frac{p(x)}{1-p(x)}\right) \quad (3.1)$$

using this substitution, Equation 3.2 was used to calculate the log odd notation form of each cell in the grid map. $l(m_i|z_{1:t}, x_{1:t})$ represent the log odd value of m_i given the observation of all robots, $z_{1:t}$, and the $x_{1:t}$ for all robot's state. Equation 3.2 is the cell's initial value, m_i . Next, set 0.5 as an initial probability. The log odds value, $l(m_i|z_{1:t}, x_{1:t})$ is then converted back to probability using Equation 3.3, which is the inverse of Equation 3.1, to get the occupancy probability.

$$l(m_i|z_{1:t}, x_{1:t}) = l(m_i|z_t, x_t) + l(m_i|z_{1:t-1}, x_{1:t-1}) - l(m_i) \quad (3.2)$$

$$p(x) = 1 - \left(\frac{1}{1 + e^{l(x)}}\right) \quad (3.3)$$

3.4 Correction of Nonlinearity using Neural Networks

A neural network is integrated to cope with an error resulting from the nonlinearity of LDS-01 due to its noisy sensor characteristic that has been analyzed. It is beneficial to use the machine learning process. Instead of interpreting the range measurement of the sensor separately, this work interpreted multiple simultaneous reading sensors. Besides, using data sensor measurement previously, it needs the training to train the learner. To do that, a simple environment is create in Gazebo simulator which use as a test area and let the robot operate by changing the position of robot manually while collecting sensed data.

The first thing to solve the problem of mapping is visualizing the data by classifying it into two-state; filled (1) or empty (0). In order to visualise, the input data of the sensor measurements need to reduce. The regression approach is then used where the four nearest sensors input a regression feed to achieve four parameters (i.e. $\alpha_1, \alpha_2, \alpha_3$, and α_4) leading to a single result, z_t . Next, z_t was calculated using Equation 3.4, where c is a fixed value. With sensor regression output, $z_t, d_{x,y}$, and $\theta_{x,y}$ the training data will be visualized in three dimensions.

$$z_t = c + \sum_i^4 \alpha_i \tilde{z}_t^i \quad (3.4)$$

The input and output configuration for implementing the Neural Network as seen in Figure 3.11. The probability of cell occupation is denoted as $p(m_i|z_t, x_t)$, which defines the probability that cell m_i is occupied provided the current location of the robot, x_t and recent measurements of the sensor, z_t . Here, m_i on the map is the i th cell, m . Function (\cdot) is the regression function in Equation 3.4.

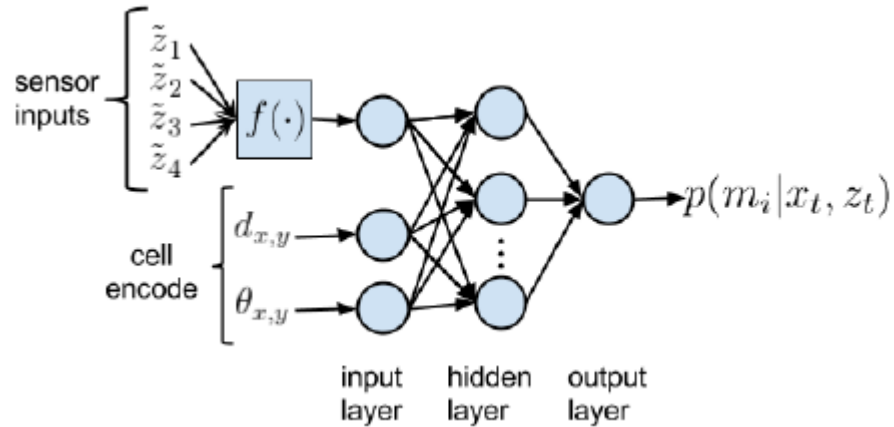


Figure 3.11: Neural network input-output configuration

Finally, in Equation 3.5, a fitness function is used to measure or rate the efficiency of each learner's output compared with the occupancy grid map algorithm.

$$f(M, M_{\text{truth}}) = 1 - \frac{\sum_{c \in \text{cells}} |M(c) - M_{\text{truth}}(c)|}{N_{\text{cells}}} \quad (3.5)$$

3.4.1 Accuracy

The Root Mean Square Error (RMSE) is a useful metric for determining how well a model fits a dataset. The greater the RMSE, the greater the range between predicted and actual values, showing that a model does not fit the data well. Conversely, the smaller the RMSE, the better a model can fit the data. Comparing the RMSE of two different models to identify which one fits the data better is extremely beneficial. To calculate the RMSE value is based on the value between target output and output data after training.

$$RMSE = \sqrt{\left[\frac{\sum (d_i - p_i)^2}{N} \right]} \quad (3.6)$$

where:

d_i = the predicted value (output target) for the i^{th} observation

p_i = the observed value (training output) for the i^{th} observation

N = the sample size



CHAPTER 4

RESULTS AND DISCUSSION



This chapter will present the test environment of the robot mapping and the data sensor measurement analysis. In part 4.3, it will show the results from the training and testing data using the neural network algorithm and from data output is used to determine the root mean square error value.

4.1 Build Test Environment

4.1.1 The environment in Gazebo Simulator

Building the test environment is the most significant element to create the occupancy grid map. By executing the commands shown in Figure 4.1, the Turtlebot3 world was launched as a test environment in the Gazebo simulator in Figure 4.2 and created the occupancy grid map as shown in Figure 4.4.

```
Terminal 1
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

Terminal 2
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```

Figure 4.1: Commands to build test environment in Gazebo



Figure 4.2: Test environment in Gazebo simulator

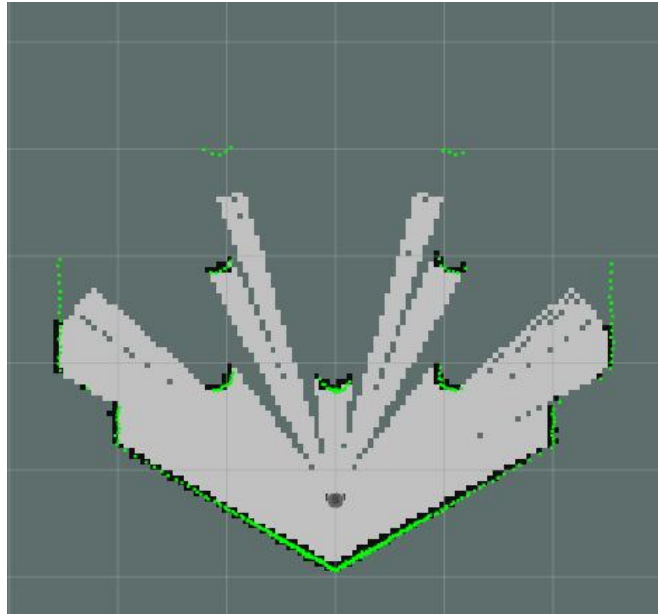


Figure 4.3: Scanning area

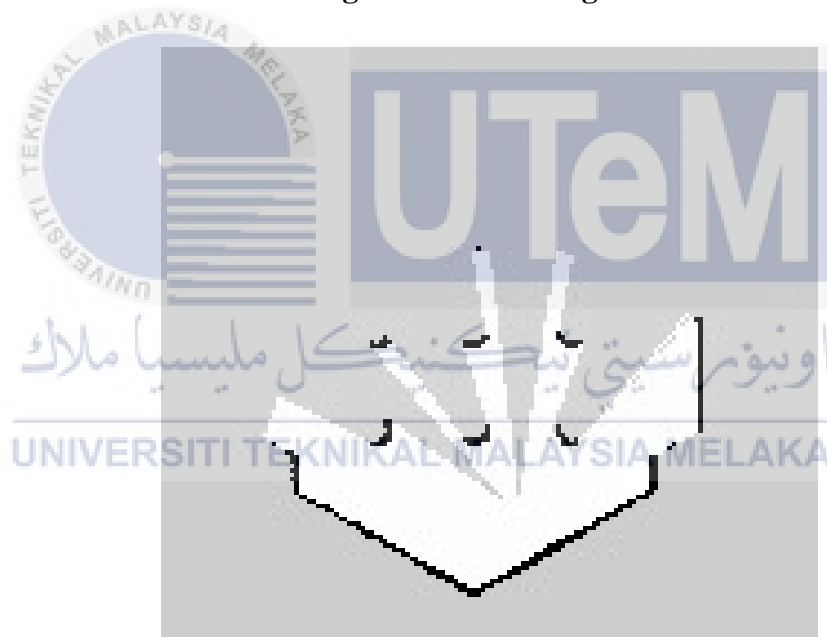


Figure 4.4: Occupancy grid map

Figure 4.5 shows the 'rqt graph' which is a tool that displays the correlation between active nodes and messages sent over the ROS network [11]. The starting node is /gazebo that provides the test environment for the robot simulation. The other node which is /turtlebot3_teleop_keyboard is launched to allow the user to control the mobile robot move forward and go around in the environment in the Gazebo simulator.

Besides, using gmapping SLAM package which uses for the occupancy grid mapping algorithm and the /map_server node was launched to provide the saved map that was created in the RViz simulation.

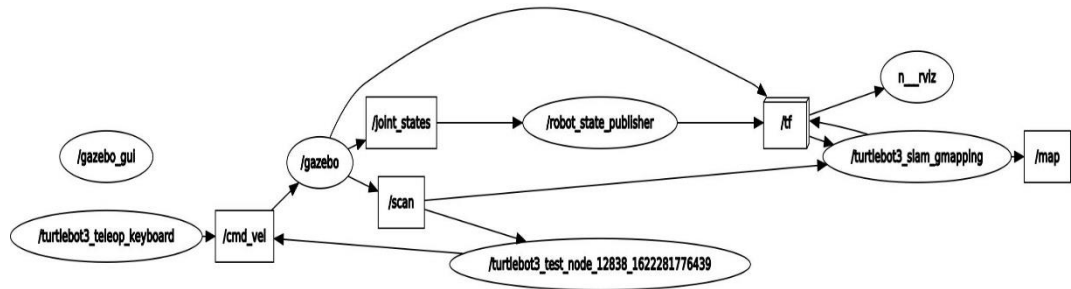


Figure 4.5: The rqt graph



4.2 Data Measurement of Sensor

4.2.1 Input and Output Data for Training and Testing

All distance data that has been collected is divided into two files of .csv format. The first file is for training input data consist of 155 reading each 0.2m and another file with different values from training data is for testing input data which consist of 30 reading each 0.2m. Table 5 and Table 6 below show the distance data for training data and testing data respectively.

Table 4: Training input and output

| No. of Reading | The distance measured between sensor and cylinder (m) | Actual Distance/ Output Target (m) |
|----------------|---|------------------------------------|
| 1 | 0.232621 | 0.2 |
| 2 | 0.251257 | 0.2 |
| : | : | 0.2 |
| 156 | 0.418543 | 0.4 |
| 157 | 0.412008 | 0.4 |
| : | : | 0.4 |
| 311 | 0.6198 | 0.6 |
| 312 | 0.615589 | 0.6 |
| : | : | 0.6 |
| 466 | 0.791495 | 0.8 |
| 467 | 0.802637 | 0.8 |
| : | : | 0.8 |
| 621 | 1.032365 | 1.0 |
| 622 | 1.041709 | 1.0 |
| : | : | 1.0 |
| 775 | 1.024512 | 1.0 |

Table 5: Testing input and output

| No. of Reading | The distance measured between sensor and cylinder (m) | Actual Distance/ Output Target (m) |
|----------------|---|------------------------------------|
| 1 | 0.2326205373 | 0.2 |
| : | : | 0.2 |
| 31 | 0.4178821445 | 0.4 |
| : | : | 0.4 |
| 61 | 0.6301306486 | 0.6 |
| : | : | 0.6 |
| 91 | 0.8265252709 | 0.8 |
| : | : | 0.8 |
| 121 | 1.9992797971 | 1.0 |
| : | : | 1.0 |
| 150 | 1.0141673088 | 1.0 |



4.2.2 Nonlinearity of Sensor Model

The use of low-cost sensors has had an impact on the nonlinearity of the sensor model. What is a nonlinear model sensor in this context? Nonlinear sensor means the sensor measurements deviate from the actual distance and give the random value. For example, the distance measurement data were collected 155 readings every 0.2m by the sensor. Figure 4.6 shows the histogram distribution reading at 0.2m which the values are random error readings of sensor measurement but still it is in the average of actual distance. Similarly, this occurs to other distances as shown Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10 because it uses the same model sensor.

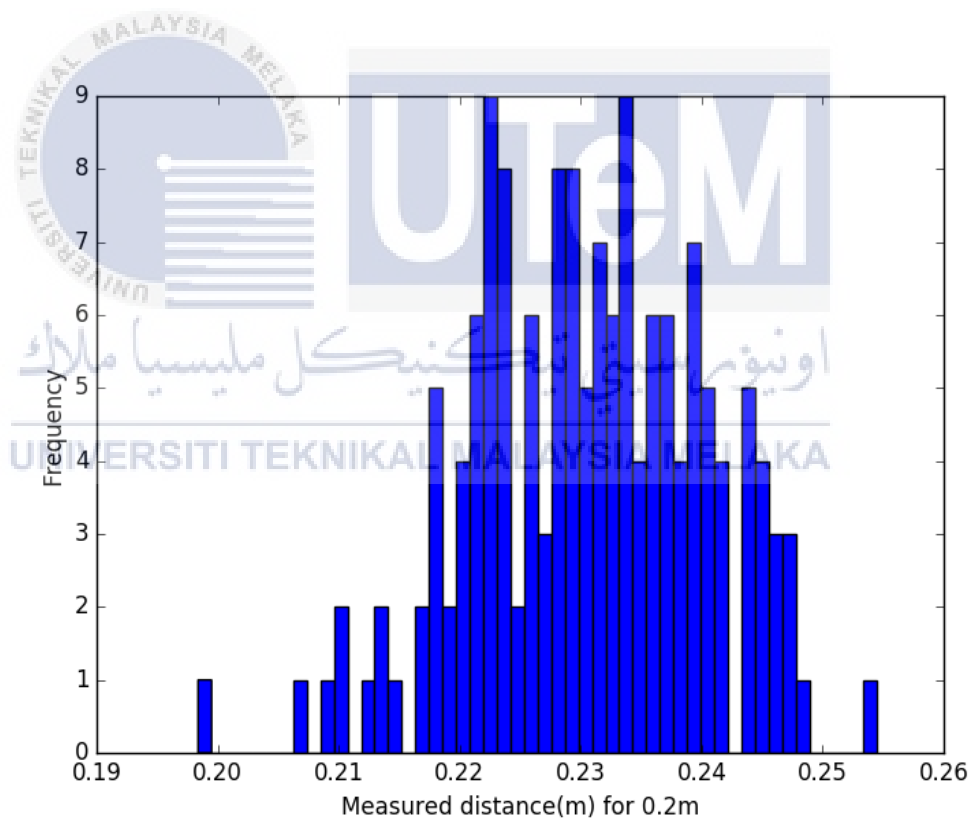


Figure 4.6: Nonlinearity of the sensor model (0.2m)

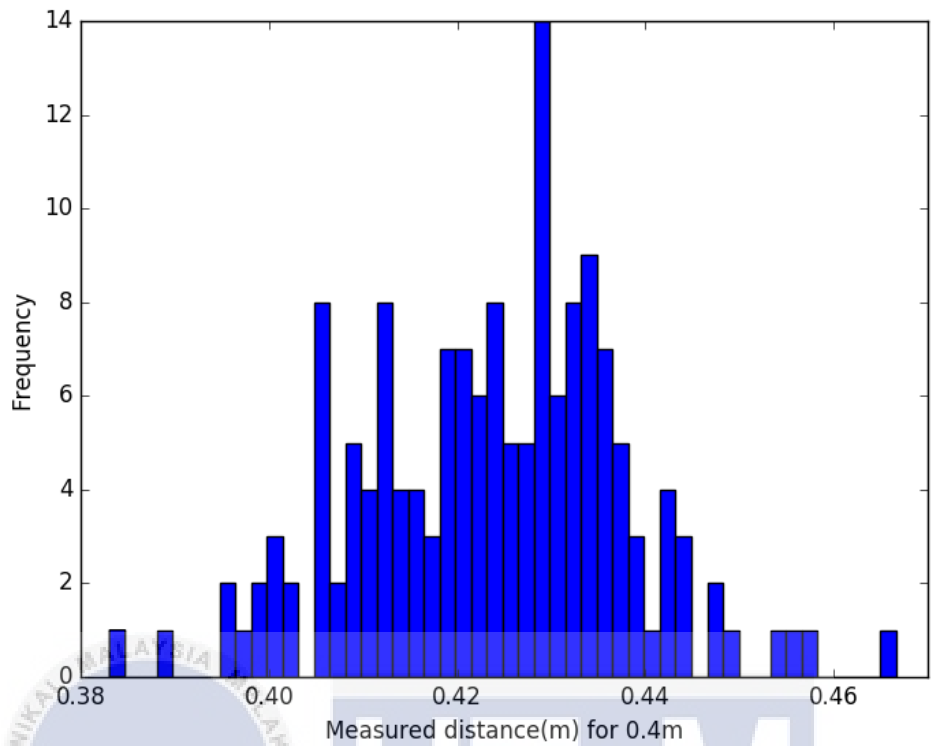


Figure 4.7: Nonlinearity of the sensor model (0.4m)

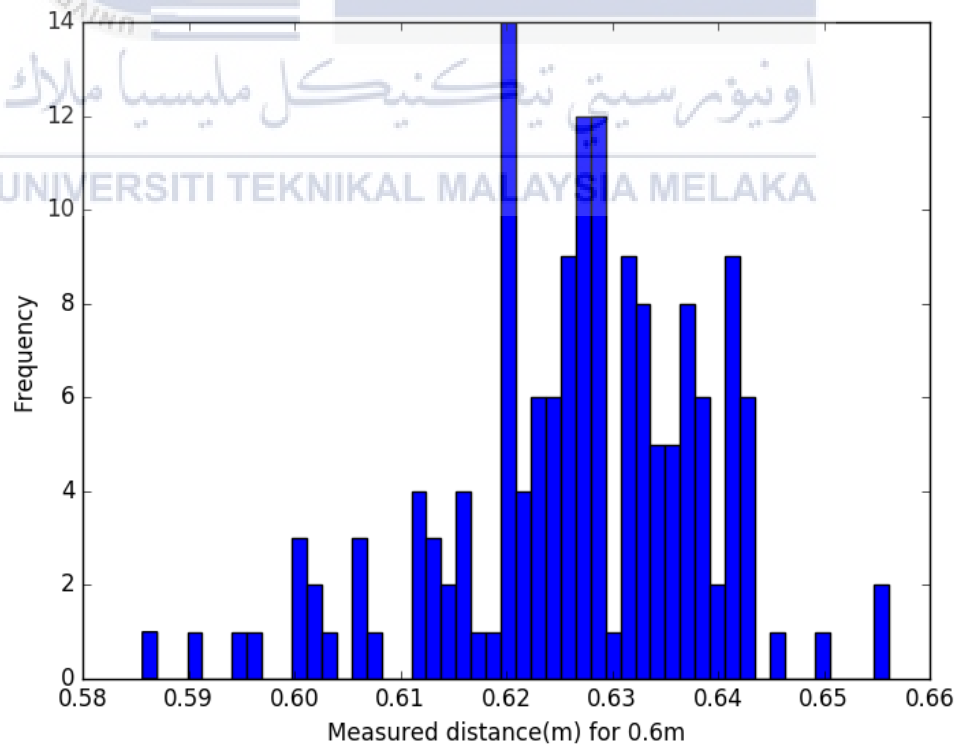


Figure 4.8: Nonlinearity of the sensor model (0.6m)

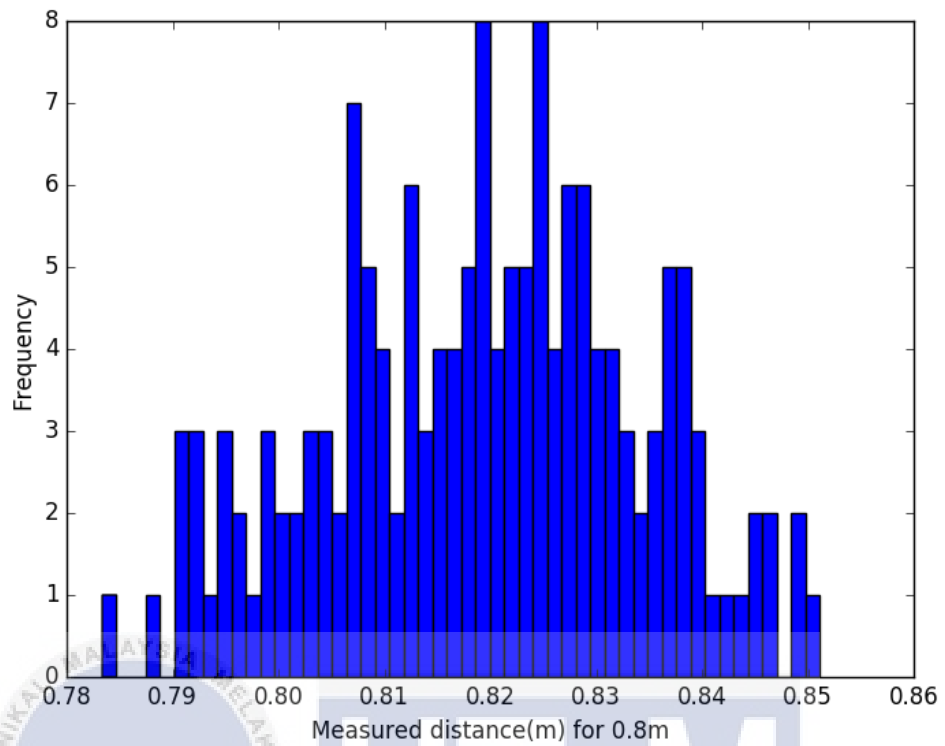


Figure 4.9: Nonlinearity of the sensor model (0.8m)

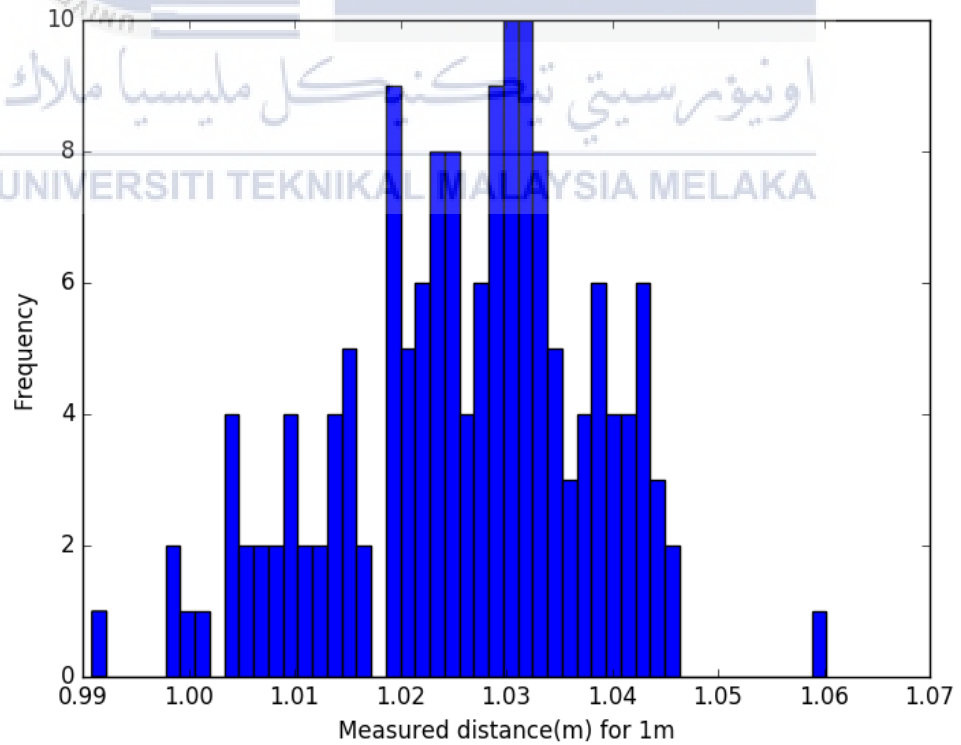


Figure 4.10: Nonlinearity of the sensor model (1.0m)

4.3 Machine Learning Algorithm

4.3.1 Training and Testing Data Sensor Measurement

The neural network has an architecture of 2 layers network. Layer 1 consists of four neurons with one input and layer 2 create as one neuron with four inputs. The random starting synaptic weights and new synaptic weights after training the data have been evaluated as below.

Stage 1: Random starting synaptic weights

- Layer 1 (4 neurons, each with 1 input):

$[[[-0.16595599 \quad 0.44064899 \quad -0.99977125 \quad -0.39533485]]]$

- Layer 2 (1 neuron, with 4 inputs):

$[[[-0.70648822] \quad [-0.81532281] \quad [-0.62747958] \quad [-0.30887855]]]$

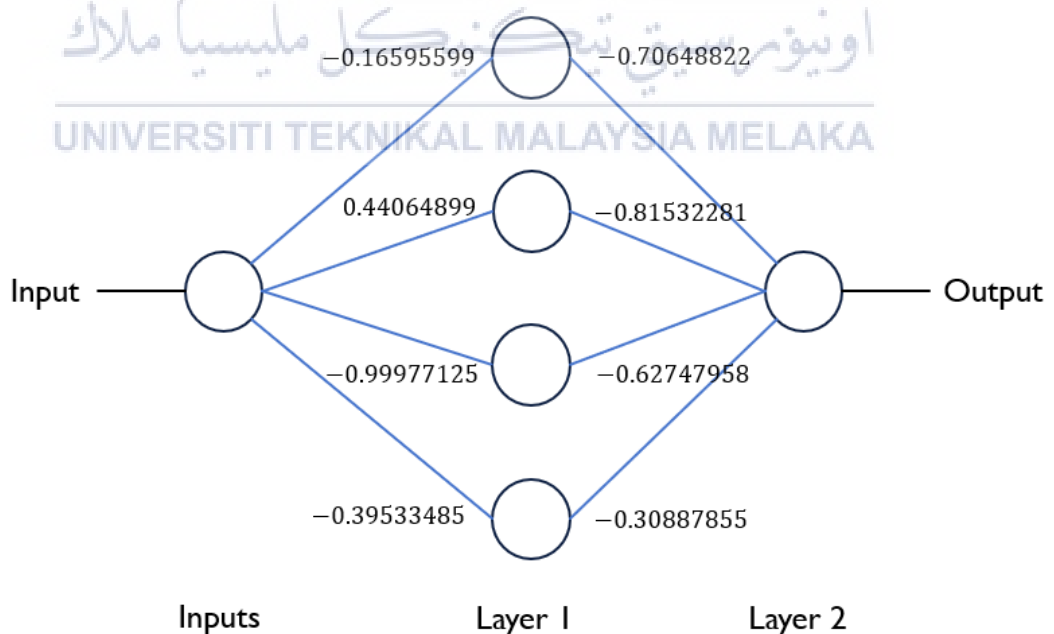


Figure 4.11: Neural network with random weight

Stage 2: New synaptic weights after training

- Layer 1 (4 neurons, each with 1 inputs):

$[-19.19837616 \quad -19.2546827 \quad -19.11644858 \quad -11.72461929]$

- Layer 2 (1 neuron, with 4 inputs):

$[[45.4681033 \quad] \quad [52.14494676 \quad] \quad [36.11792163 \quad] \quad [-49.64325098]]$

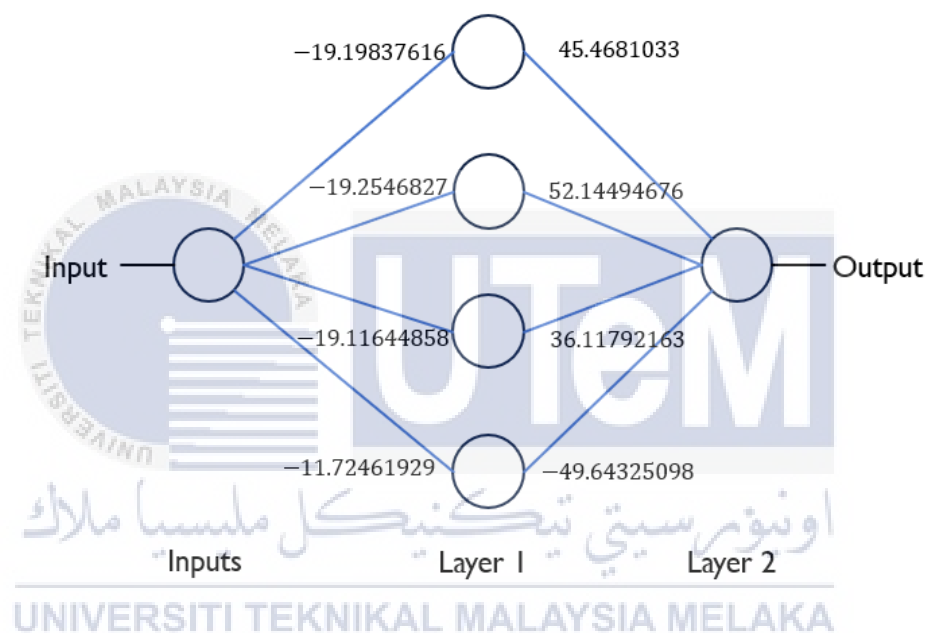


Figure 4.12: Neural network with weight after training

After training the data sensor measurement, testing input data is required to evaluate the accuracy of neural network learner. This is considered a new situation. The output data is calculated by using neural network learner with the synaptic weight and bias in the training session. Table 7 shows the output of the neural network from testing data along with the target output data. The complete testing output is included in the Appendix B.

Stage 3: Considering a new situation

Table 6: Output data of neural network

| Line | Output Data (Matric 50x1) |
|------|---------------------------|
| 1. | [[0.17841534] |
| 2. | [0.17841534] |
| 3. | [0.19597195] |
| 4. | [0.18086985] |
| 5. | [0.18432288] |
| 6. | [0.17908762] |
| 7. | [0.17934008] |
| 8. | [0.17457509] |
| 9. | [0.17738718] |
| 10. | [0.17204118] |
| 11. | [0.17754166] |
| 12. | [0.17447898] |
| 13. | [0.18169514] |
| 14. | [0.17532408] |
| 15. | [0.18969734] |

4.4 Accuracy of Training

4.4.1 Root Mean Square Error

In this case, the Root Mean Square Error (RMSE) value which is 0.265800109993 was obtained. By comparing to the author [1], the RMSE is 2.5×10^{-5} which is smaller than in this project. It happens due to the author used neural network configuration of 4 layered network consist six nodes and two nodes in the first and second hidden layers, respectively. However, the error conducted in this project can be improved by inserting more input and the neurons during training the data using a neural network algorithm.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

This chapter will discuss and summarize the results of this project. The recommendation for future development and works will be discussed to enhance the quality of the project.

5.1 Conclusion

This project has a problem creating the map with high accuracy for mobile robots using low-cost sensors due to the nonlinearity of data measured by the sensor. It has been proven based on the histogram of the distribution of the distance readings in subtopic 4.2.2. The reading results in the range are quite different compare to the actual distance. Thus, the various nonlinear value can be enforced using machine learning algorithms and chose neural network architectures to carry out this work.

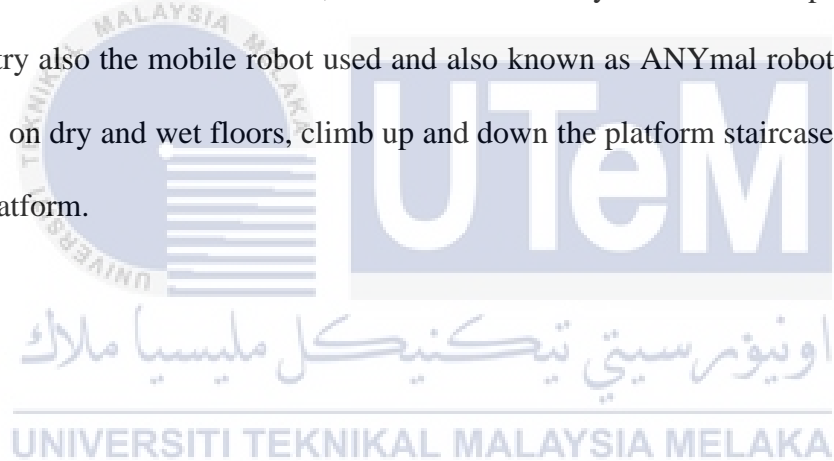
The neural network configurations that have been proposed were implemented which can interpret multiple noisy sensor measurements and corrected the reading error. In this stage, training and testing data is managed to get perfect weight, bias, and output data from the neural network. Consequently, machine learning is a good strategy to train universal data at one time.

Furthermore, the accuracy of using a low-cost sensor was analyzed using the root mean square error (RMSE). The value of RMSE determined how good the data collected is and if the RMSE is small, then the data fit well. However, it could not get lower because of the noisy sensor measurements and the need to train more input data. It also because it has a limitation in time to collect and train a large number of input data.

Lastly, the occupancy grid map could not be generated from the output data in the neural network. It happens due to some issues with the plugin or inserts the topic node in the RViz simulator such as `ground_truth/state` topic. If this part is successfully created, the quality of the grid map can be determined based on the occupied and unoccupied cells.

5.2 Future Works

The accuracy of the sensor measured using a neural network can be improved which adding more data input for training. Next, to create the occupancy grid map can amend by creating new codes that include publisher and subscriber which is published scan and subscribe gmapping. In the future, there may be robotic discoveries that are beyond the dreams such as mobile robot can measure the circumference of the object and measure curved object. As for now, everyone knows the existence of drone functions especially for photography, and the military. But then, not many people know robotic (drone) in the oil and gas industry used for security surveillance, measure the curved surface of each tank, and material delivery on an offshore platform. In that industry also the mobile robot used and also known as ANYmal robot that can walk stably on dry and wet floors, climb up and down the platform staircase, and manning the platform.



REFERENCES

- [1] Y. S. Ha and H. H. Kim, "Environmental map building for a mobile robot using infrared range-finder sensors," *Adv. Robot.*, vol. 18, no. 4, pp. 437–450, 2004, doi: 10.1163/156855304773822509.
- [2] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer (Long Beach, Calif.)*, vol. 22, no. 6, pp. 46–57, 1989, doi: 10.1109/2.30720.
- [3] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," *Proc. Natl. Conf. Artif. Intell.*, vol. 3, pp. 2024–2030, 2012.
- [4] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artif. Intell.*, vol. 99, no. 1, pp. 21–71, 1998, doi: 10.1016/S0004-3702(97)00078-7.
- [5] N. M. Yatim and N. Buniyamin, "Indoor mapping with machine learning algorithm using Khepera III mobile robot," *J. Telecommun. Electron. Comput. Eng.*, vol. 8, no. 9, pp. 61–66, 2016.
- [6] A. Prorok, A. Arfire, A. Bahr, J. R. Farserotu, and A. Martinoli, "Indoor navigation research with the Khepera III mobile robot: An experimental

- baseline with a case-study on ultra-wideband positioning,” *2010 Int. Conf. Indoor Position. Indoor Navig. IPIN 2010 - Conf. Proc.*, 2010, doi: 10.1109/IPIN.2010.5647880.
- [7] J. M. Soares, I. Navarro, and A. Martinoli, *The khepera IV mobile robot: Performance evaluation, sensory data and software toolbox*, vol. 417. 2016.
- [8] T. Mohammad, “Using Ultrasonic and Infrared Sensor for Distance Measurement,” 2009, [Online]. Available: <http://www.waset.org/journals/waset/v27/v27-51.pdf>.
- [9] N. J. F. M Novotny, “Using infrared sensor and the Phong illumination model to measure distances.”
- [10] S. Saeedi, L. Paull, M. Trentini, and H. Li, “Neural network-based multiple robot simultaneous localization and mapping,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 22, no. 12, pp. 880–885, 2011, doi: 10.1109/IROS.2011.6048300.
- [11] T. L. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, *Robot Programming.*, vol. 23, no. 4. 1984.
- [12] J. C. Lentin Joseph, *Mastering ROS for Robotics Programming.* .
- [13] W. D. S. Morgan Quigley, Brian Gerkey, *Programming Robots with ROS: A Practical Introduction to the Robot Operating System.* .

APPENDICES

Appendix A

Python codes for Neural Network

```

from numpy import exp, array, random, dot
import numpy as np

class NeuronLayer():
    def __init__(self, number_of_neurons, number_of_inputs_per_neuron):
        self.synaptic_weights = 2 * random.random((number_of_inputs_per_neuron, number_of_neurons)) - 1

class NeuralNetwork():
    def __init__(self, layer1, layer2):
        self.layer1 = layer1
        self.layer2 = layer2

    # The Sigmoid function, which describes an S shaped curve.
    # We pass the weighted sum of the inputs through this function to
    # normalise them between 0 and 1.
    def __sigmoid(self, x):
        return 1 / (1 + exp(-x))

    # The derivative of the Sigmoid function.
    # This is the gradient of the Sigmoid curve.
    # It indicates how confident we are about the existing weight.
    def __sigmoid_derivative(self, x):
        return x * (1 - x)

    # We train the neural network through a process of trial and error.
    # Adjusting the synaptic weights each time.
    def train(self, training_set_inputs, training_set_outputs, number_of_training_iterations):
        for iteration in xrange(number_of_training_iterations):
            # Pass the training set through our neural network
            output_from_layer_1, output_from_layer_2 = self.think(training_set_inputs)

            # Calculate the error for layer 2 (The difference between the desired output
            # and the predicted output).
            layer2_error = training_set_outputs - output_from_layer_2
            layer2_delta = layer2_error * self.__sigmoid_derivative(output_from_layer_2)

```

```

        # Calculate how much to adjust the weights by
        layer1_adjustment = training_set_inputs.T.dot(layer1_delta)
        layer2_adjustment = output_from_layer_1.T.dot(layer2_delta)

        # Adjust the weights.
        self.layer1.synaptic_weights += layer1_adjustment
        self.layer2.synaptic_weights += layer2_adjustment

    # The neural network thinks.
    def think(self, inputs):
        output_from_layer1 = self.__sigmoid(dot(inputs, self.layer1.synaptic_weights))
        output_from_layer2 = self.__sigmoid(dot(output_from_layer1, self.layer2.synaptic_weights))
        return output_from_layer1, output_from_layer2

    # The neural network prints its weights
    def print_weights(self):
        print "   Layer 1 (4 neurons, each with 1 inputs): "
        print self.layer1.synaptic_weights
        print "   Layer 2 (1 neuron, with 4 inputs):"
        print self.layer2.synaptic_weights

    # To calculate the root mean square error
    def rmse(self, predictions, target):
        return np.sqrt(((predictions - target) ** 2).mean())

if __name__ == "__main__":

    #Seed the random number generator
    random.seed(1)

    # Create layer 1 (4 neurons, each with 1 inputs)
    layer1 = NeuronLayer(4, 1)

    # Create layer 2 (a single neuron with 1 inputs)
    layer2 = NeuronLayer(1, 4)

    # Combine the layers to create a neural network
    neural_network = NeuralNetwork(layer1, layer2)

    print "Stage 1) Random starting synaptic weights: "
    neural_network.print_weights()

    # The training set. We have 775 examples, each consisting of 1 input values
    # and 1 output value.
    inputs = np.genfromtxt('newtrain.csv', delimiter=',')
    training_set_inputs = inputs[:,0]
    training_set_inputs.shape = (775, 1)
    training_set_outputs = inputs[:,1]
    training_set_outputs.shape = (775, 1)

    # Train the neural network using the training set.
    # Do it 60,000 times and make small adjustments each time.
    neural_network.train(training_set_inputs, training_set_outputs, 60000)

    print "Stage 2) New synaptic weights after training: "
    neural_network.print_weights()

    test= np.genfromtxt('newtest.csv', delimiter=',')
    testing_inputs = test[:,0]
    testing_inputs.shape = (150, 1)
    target_outputs = test[:,1]
    target_outputs.shape = (150, 1)

    # Test the neural network with a new situation.
    print "Stage 3) Considering a new situation -> ?:"
    hidden_state, output = neural_network.think(testing_inputs)
    print output

    d = target_outputs #actual/predictions
    p = output #output data
    rmse_val = neural_network.rmse(np.array(d), np.array(p))
    print("The rms error is: " + str(rmse_val))

```

Appendix B

Output data of Neural Network

| Line | Output Data (Matric 50x1) |
|------|---------------------------|
| 1. | [[0.17841534] |
| 2. | [0.17841534] |
| 3. | [0.19597195] |
| 4. | [0.18086985] |
| 5. | [0.18432288] |
| 6. | [0.17908762] |
| 7. | [0.17934008] |
| 8. | [0.17457509] |
| 9. | [0.17738718] |
| 10. | [0.17204118] |
| 11. | [0.17754166] |
| 12. | [0.17447898] |
| 13. | [0.18169514] |
| 14. | [0.17532408] |
| 15. | [0.18969734] |
| 16. | [0.17878401] |
| 17. | [0.18159867] |
| 18. | [0.17281794] |
| 19. | [0.18893698] |
| 20. | [0.17395106] |
| 21. | [0.18487418] |
| 22. | [0.17283385] |
| 23. | [0.17633342] |
| 24. | [0.17648447] |
| 25. | [0.17809337] |
| 26. | [0.17815696] |
| 27. | [0.18002999] |
| 28. | [0.17544320] |

| | |
|-----|---------------|
| 29. | [0.17510481] |
| 30. | [0.19154380] |
| 31. | [0.41988101] |
| 32. | [0.43664127] |
| 33. | [0.42145747] |
| 34. | [0.42620355] |
| 35. | [0.43419979] |
| 36. | [0.42708777] |
| 37. | [0.40934830] |
| 38. | [0.41710180] |
| 39. | [0.41924103] |
| 40. | [0.42183939] |
| 41. | [0.42817683] |
| 42. | [0.41614855] |
| 43. | [0.41012578] |
| 44. | [0.41676626] |
| 45. | [0.43284187] |
| 46. | [0.42352975] |
| 47. | [0.42291955] |
| 48. | [0.41781213] |
| 49. | [0.43368679] |
| 50. | [0.41690272] |
| 51. | [0.42449495] |
| 52. | [0.43392682] |
| 53. | [0.43116039] |
| 54. | [0.42501785] |
| 55. | [0.41829003] |
| 56. | [0.43388637] |
| 57. | [0.42847033] |
| 58. | [0.42351734] |
| 59. | [0.42919741] |
| 60. | [0.42901972] |

| | |
|-----|---------------|
| 61. | [0.49251437] |
| 62. | [0.49251437] |
| 63. | [0.49073774] |
| 64. | [0.49295188] |
| 65. | [0.49286269] |
| 66. | [0.49345127] |
| 67. | [0.49331137] |
| 68. | [0.49257241] |
| 69. | [0.49131592] |
| 70. | [0.49340855] |
| 71. | [0.49122178] |
| 72. | [0.49282492] |
| 73. | [0.49185649] |
| 74. | [0.48995618] |
| 75. | [0.49211612] |
| 76. | [0.49274264] |
| 77. | [0.49219183] |
| 78. | [0.49263743] |
| 79. | [0.49299032] |
| 80. | [0.49289202] |
| 81. | [0.49120310] |
| 82. | [0.49375504] |
| 83. | [0.49201299] |
| 84. | [0.48956221] |
| 85. | [0.49260660] |
| 86. | [0.49112941] |
| 87. | [0.49247740] |
| 88. | [0.49211799] |
| 89. | [0.49278672] |
| 90. | [0.49292663] |
| 91. | [0.49923666] |
| 92. | [0.49922821] |

| | |
|------|---------------|
| 93. | [0.49932173] |
| 94. | [0.49939474] |
| 95. | [0.49940630] |
| 96. | [0.49921115] |
| 97. | [0.49923138] |
| 98. | [0.49918269] |
| 99. | [0.49916574] |
| 100. | [0.49921709] |
| 101. | [0.49932182] |
| 102. | [0.49927378] |
| 103. | [0.49931317] |
| 104. | [0.49927127] |
| 105. | [0.49925687] |
| 106. | [0.49925771] |
| 107. | [0.49923142] |
| 108. | [0.49933700] |
| 109. | [0.49922832] |
| 110. | [0.49903675] |
| 111. | [0.49905059] |
| 112. | [0.49917469] |
| 113. | [0.49936011] |
| 114. | [0.49915363] |
| 115. | [0.49927663] |
| 116. | [0.49912695] |
| 117. | [0.49921511] |
| 118. | [0.49919504] |
| 119. | [0.49920509] |
| 120. | [0.49925681] |
| 121. | [0.49989888] |
| 122. | [0.49992765] |
| 123. | [0.49992129] |
| 124. | [0.49993215] |

| | |
|------|---------------|
| 125. | [0.49992225] |
| 126. | [0.49991421] |
| 127. | [0.49989862] |
| 128. | [0.49990900] |
| 129. | [0.49992112] |
| 130. | [0.49991148] |
| 131. | [0.49991952] |
| 132. | [0.49991972] |
| 133. | [0.49991293] |
| 134. | [0.49992062] |
| 135. | [0.49993003] |
| 136. | [0.49991891] |
| 137. | [0.49992366] |
| 138. | [0.49991225] |
| 139. | [0.49990784] |
| 140. | [0.49990176] |
| 141. | [0.49992145] |
| 142. | [0.49990928] |
| 143. | [0.49990725] |
| 144. | [0.49990421] |
| 145. | [0.49990578] |
| 146. | [0.49991633] |
| 147. | [0.49991804] |
| 148. | [0.49991155] |
| 149. | [0.49991193] |
| 150. | [0.49991506] |

Appendix D

Cost of project

| ITEM | DESCRIPTION | QUANTITY | UNIT PRICE (RM) | AMOUNT (RM) |
|-----------------|--|----------|-----------------|-------------|
| 1. | TurtleBot3 Burger | 1 | 1,175.00 | 1,175.00 |
| 2. | Laser Distance Range Sensor (LDS-01) including postage fee | 1 | 102.54 | 102.54 |
| 3. | Corrugated Board (300mm X 420mm X 30mm) | 20 | 0.90 | 18.00 |
| Subtotal | | | | RM 1,295.54 |

The table above shows the cost of this project that needs to spend is about RM 1,295.54. However, the project conducted is fully used the software due to pandemic, it does not require any cost.