



**Faculty of Electrical Engineering**  
**Universiti Teknikal Malaysia Melaka**

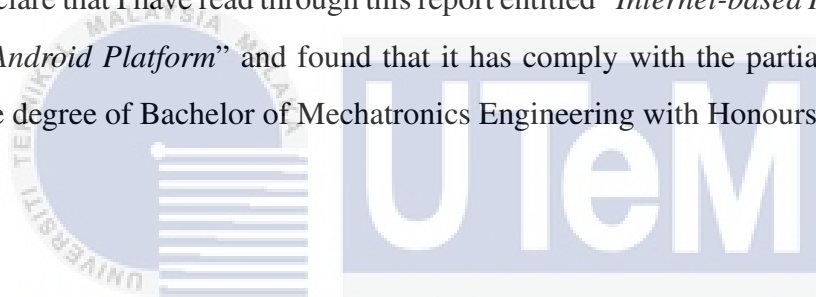


**LOR SHENG QIN**

**Bachelor of Mechatronics Engineering with Honours**

**2017**

“I hereby declare that I have read through this report entitled “*Internet-based Robotic Control System via Android Platform*” and found that it has comply with the partial fulfilment for awarding the degree of Bachelor of Mechatronics Engineering with Honours.”



Signature : .....  
اونيور سیتی تیکنیکل ملیسیا ملاک

Supervisor's Name : DR MUHAMMAD HERMAN BIN JAMALUDDIN  
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Date : .....

**INTERNET-BASED ROBOTIC CONTROL SYSTEM VIA ANDROID  
PLATFORM**

**LOR SHENG QIN**



**A report submitted in partial fulfilment of the requirements for the degree of  
Bachelor of Mechatronics Engineering with Honours**

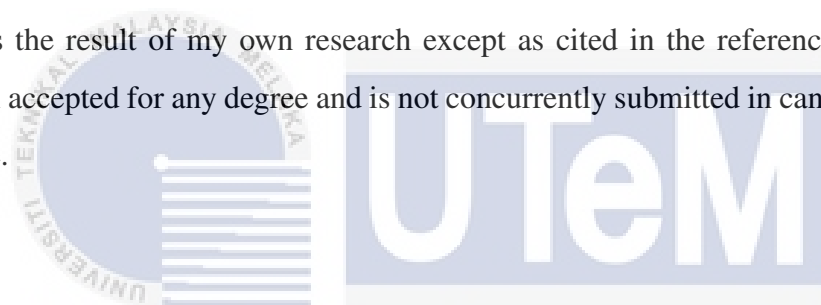
**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**Faculty of Electrical Engineering**

**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**2017**

I declare that this report entitled “*Internet-based Robotic Control System via Android Platform*” is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



Signature : اونيور سيتي تيكنيكل ميسيا ملاك

Name : UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Date : .....



## ACKNOWLEDGEMENT

Prima facie, I am grateful and thankful to the Almighty God our heavenly Father, for His grace and wisdom with good health that granted unto me throughout the completion of this Final Year Project.

With great pleasure, I would like to express utmost gratitude to Dr Muhammad Herman bin Jamaluddin, my supervisor of Final Year Project and whom also the Head of Department of Mechatronics Engineering for providing me with all the information, valuable guidance and encouragement extended to me despite being busy with his works. Dr Herman exhibited his generosity, guidance and encouragement on me throughout this project; I have also gain adequate knowledge, skills, supervision and training from him in various part from this project.

Next, I am also indebted to my family especially parents for their unceasing encouragement and attention. Getting through four years of my Degree was not an easy task, their financial and morality supports is the main pillar throughout my university life. Without my parents, this Degree which is not deemed possible for me to made it through such smooth and easy. Also not forgetting my relatives for their backing, optimistic comments and concern in my studies.

Upon ending this, I would take the opportunity to give my sincere gratitude to Teow Boon Pei that lent a helping hand in this project, my fellow coursemates, housemates and ex-colleagues for their constructive comment and unselfish idea which enlighten me through this project. Their benevolent assist and well-reasoned opinions made me a better person and improved my thinking mindset. There are also many people I would like to give thanks but seems impossible with the provided space here. Thus, I would like to acknowledge their unconditional contributions and assistance in all form gratefully.

## ABSTRACT

As Internet of Things (IoT) emerges to become popular and rapidly growing in this era, it allows people to connect all sensors that interact in our daily life for various purpose. Aside from real time data monitoring, data collection and exchanging, this concept aimed to transfer data without human interaction with machine. Nowadays, with the combination of low-cost components that can be easily obtain from electronic stores, IoT can be made possible by self learning and hands-on projects. There are problems of accessing places or areas that are unfit for human such as collapse building, radioactive factory, explosive device area or even disaster area. Human are prone to life-threatening diseases and injuries while mobile robot does not, the parts can be replaced whenever damaged while human cannot. Hence, this project proposed a Arduino powered Omni-wheel mobile robot that can be control with Android device through internet wirelessly. Objectives involved are design and develop a mobile robot that can be controlled by Android platform device wirelessly, develop an Android application for mobile robot control and analyse the distance and obstacle detection using sensor. This system involves two interrelated components which are Android platform application and mobile robot, as objectives both will communicate and interact with each others through cloud server. Obstacle detection system is added to the mobile robot, where it can detects obstacle ahead using multiple sensors. Reliability study of the mobile robots which includes the translational motion, rotational motion, data transmission test and obstacle detection test had been carried out. Each experiment on the mobile robot will be repeated 10 times to increase accuracy as well as consistency, results are tabulated with average value. Expected results would be successful control of mobile robot using Android application and obstacle detection system is successfully deployed. These analysis will make way for future researcher in making IoT mobile robot more successful.

## ABSTRAK

*Objek Rangkaian Internet (IOT) muncul untuk menjadi popular dan berkembang pesat dalam era ini, ia membolehkan orang ramai menyambung semua sensor yang berinteraksi dalam kehidupan seharian kita untuk pelbagai tujuan. Selain daripada memantau data dan mengumpul data, konsep ini bertujuan untuk permindahan data yang tanpa interaksi manusia dengan mesin ataupun komputer. Pada masa kini, dengan gabungan komponen kos rendah yang mudah didperolehi daripada kedai-kedai elektronik, projek IOT boleh dibuat oleh seseorang dengan pembelajaran sendiri dan praktikal. Projek ini boleh membantu untuk menyelesaikan masalah mengakses ke tempat atau kawasan-kawasan yang tidak sesuai untuk manusia seperti bangunan runtuh, kilang radioaktif, kawasan letupan atau kawasan bencana. Manusia terdedah kepada penyakit dan kecederaan manakala robot tidak, bahagian-bahagian yang rosak boleh diganti bila-bila masa manakala manusia tidak boleh. Oleh itu, projek ini bercadang satu robot mudah alih Arduino yang boleh dikawal oleh peranti Android secara wayarless melalui internet. Objektif yang terlibat adalah mereka bentuk dan membina sebuah robot mudah alih yang boleh dikawal oleh peranti Android secara wayarless, membina sebuah aplikasi Android untuk mengawal robot mudah alih dan menganalisis pengesanan jarak dan halangan yang menggunakan sensor. Sistem ini melibatkan aplikasi platform Android dan robot mudah alih, sebagai objektif kedua-duanya akan berkomunikasi dan berinteraksi antara satu sama lain melalui pelayan awan. Ciri pengesanan halangan objek ditambah kepada robot mudah alih, di mana ia boleh mengesan halangan pada depan menggunakan sensor yang ada. Kajian kebolehpercayaan robot mudah alih yang termasuk gerakan translasi, gerakan putaran, ujian penghantaran data dan ujian mengesan halangan telah dijalankan. Keputusan yang dijangka daripada projek ini adalah kejayaan mengawal robot melalui aplikasi Android dan sistem mengesan halangan berjaya dilaksanakan. Analisis ini akan memberi laluan kepada penyelidik masa depan dalam membuat IOT robot mudah alih yang lebih berjaya.*



## TABLE OF CONTENTS

<b>Acknowledgement</b>	v
<b>ABSTRACT</b>	vi
<b>ABSTRAK</b>	vii
<b>TABLE OF CONTENTS</b>	viii
<b>List of Tables</b>	xi
<b>List of Figures</b>	xii
<b>CHAPTER 1 INTRODUCTION</b>	1
1.1 Overview	1
1.2 Problem Statement	2
1.3 Motivation	3
1.4 Objectives	4
1.5 Scope of Study	5
1.6 Summary	5
 <b>CHAPTER 2 LITERATURE REVIEW</b>	
2.1 Introduction	6
2.2 Related Works	7
2.3 Wireless Technology	8
2.4 Controller Platform	10
2.5 Android Platform	14
2.5.1 Android Stack	15
2.6 Lightweight Communication Protocol for Robotic Application	18

2.6.1	Message Queuing Telemetry Transport (MQTT)	19
2.6.2	Constrained Application Protocol (CoAP)	20

### CHAPTER 3 METHODOLOGY

3.1	Introduction	21
3.2	Project Overview	22
3.3	System Overview	24
3.4	Components of Mobile Robot	25
3.4.1	Robot Base	25
3.4.2	Omni wheel	27
3.4.3	Miniature DC gear motor	28
3.4.4	L293D DC Motor Driver	29
3.4.5	ESP-8266 Wifi Module	31
3.4.6	Ultrasonic Sensor	32
3.4.7	Arduino Mega 2560 microcontroller	33
3.5	Mobile Robot Hardware Design	35
3.6	Obstacle Detection System Design	36
3.6.1	Flow Chart of obstacle detection system	37
3.7	Android Application Design	38
3.8	Research Methodology	38
3.8.1	Transmission test between Arduino and Android application	39
3.8.2	Hardware Reliability	40
3.8.2.1	Translational Motion Test	40
3.8.2.2	Rotational Motion Test	42
3.8.3	Obstacle Detection Test	43

### CHAPTER 4 RESULT AND DISCUSSION

4.1	Introduction	45
4.2	Hardware Design and Fabrication	46
4.2.1	Mobile Robot Hardware with Obstacle Detection System	48
4.3	Android application	49
4.4	Data Transmission between Arduino and Android application	52
4.5	Hardware Reliability	54
4.5.1	Translational Motion Test	56
4.5.2	Rotational Motion Test	58
4.6	Obstacle Detection of mobile robot	60
<b>CHAPTER 5</b>	<b>CONCLUSION AND RECOMMENDATION</b>	
5.1	Conclusion	62
5.2	Recommendation	64
<b>REFERENCES</b>		65
<b>APPENDICES</b>	<b>UNIVERSITI TEKNIKAL MALAYSIA MELAKA</b>	<b>68</b>
<b>APPENDIX A</b>	<b>ANDROID APPLICATION JAVA CODING</b>	69
<b>APPENDIX B</b>	<b>ANDROID APPLICATION LAYOUT CODING</b>	80
<b>APPENDIX C</b>	<b>ARDUINO CODING</b>	86
<b>APPENDIX D</b>	<b>GANTT CHART FOR FINAL YEAR PROJECT 1</b>	94
<b>APPENDIX E</b>	<b>GANTT CHART FOR FINAL YEAR PROJECT 2</b>	95

## LIST OF TABLES

<b>TABLE</b>	<b>TITLE</b>	<b>PAGE</b>
Table 2.1	Comparison of Bluetooth and Wi-Fi specifications.	9
Table 2.2	Comparison of data rates for fixed and mobile standards.	9
Table 3.1	Table of comparison between ABS and PLA.	26
Table 3.2	Specifications of Omni wheel.	27
Table 3.3	Specifications of miniature DC gear motor.	29
Table 3.4	Specifications of L293D Motor Driver module.	30
Table 3.5	Table of ESP-8266 pin description.	31
Table 3.6	Specifications of HC-SR04 module.	32
Table 3.7	Table of Arduino Mega 2560 microcontroller Specifications	34
Table 4.1	Packet delivery time for each command to be send from Android application to Arduino or vice versa.	52
Table 4.2	Hardware reliability based on the communication between the mobile robot and Android application test done.	54
Table 4.3	Comparison of mobile robot's center point deviation angle from initial starting point at two stopping distance.	56
Table 4.4	Comparison of Rotational motion test with 3 combinations of Pulse Width Modulation (PWM) and Run Time for left and right rotation.	58
Table 4.5	Comparison of stopping distance of mobile robot when encountered obstacle ahead.	60

## LIST OF FIGURES

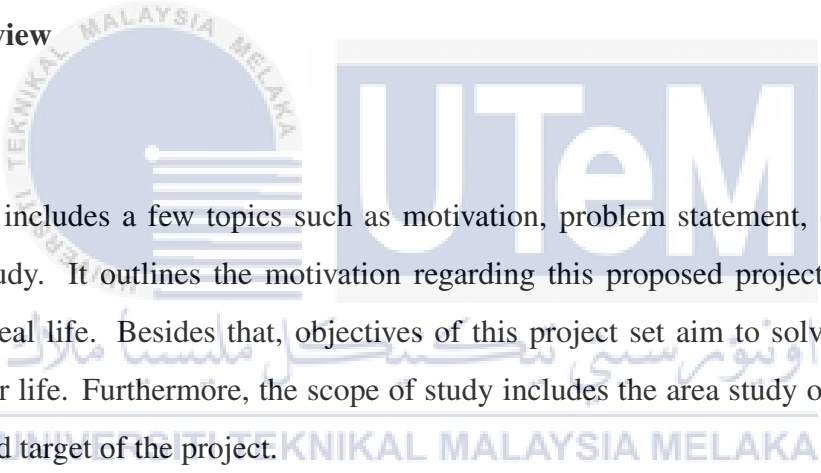
<b>FIGURE</b>	<b>TITLE</b>	<b>PAGE</b>
Figure 2.1	Arduino Mega 2560 microcontroller board	11
Figure 2.2	Raspberry Pi 2	12
Figure 2.3	Beaglebone Black.	12
Figure 2.4	Udoo x86 Basic.	13
Figure 2.5	Android Logo.	14
Figure 2.6	The software stack of Android platform [1].	15
Figure 3.1	The overall flow chart for this project.	23
Figure 3.2	The overall system in block diagram.	24
Figure 3.3	Design of mobile robot base using Solidworks.	26
Figure 3.4	A pair of Omni wheel.	27
Figure 3.5	Miniature DC gear motor.	28
Figure 3.6	Dimension of the miniature DC gear motor.	28
Figure 3.7	L293D 4 H-bridge motor driver module.	30
Figure 3.8	ESP-8266 Wifi module revision 1.	31
Figure 3.9	Ultrasonic ranging sensor, HC-SR04.	32
Figure 3.10	Arduino or Genuino Mega 2560 microcontroller.	34
Figure 3.11	Isometric view of the mobile robot hardware.	35
Figure 3.12	Slot for the placement of ultrasonic sensors mounting on the mobile robot base.	36
Figure 3.13	Flow chart of the obstacle detection system on the mobile robot.	37

Figure 3.14	Setup of test with (a) center point of mobile robot with 4 bamboo skewer bottom view, and (b) angle marked on the floor.	<b>41</b>
Figure 3.15	Bamboo skewer is attached to front edge of the mobile robot.	<b>42</b>
Figure 3.16	Mobile robot (a) is set-up according to 60cm and 90cm, and (b) bamboo skewer is point to the starting line.	<b>44</b>
Figure 4.1	Fabricated mobile robot base using 3D printer.	<b>46</b>
Figure 4.2	Fabricated Omni wheel's shaft using 3D printer.	<b>47</b>
Figure 4.3	Shaft fabricated (a) combine with DC motor and Omni wheel, and (b) the combination attached to fabricated robot base.	<b>47</b>
Figure 4.4	Fully assembled Omni wheel mobile robot.	<b>48</b>
Figure 4.5	OmniRobot apps with customized logo for mobile robot control.	<b>49</b>
Figure 4.6	Information that is generated and assigned by CloudMqtt borker website.	<b>50</b>
Figure 4.7	Application interface which are (a) disconnected from the mobile robot, and (b) connected to the mobile robot with update status obtained.	<b>50</b>
Figure 4.8	Home internet performance before test is carried out.	<b>53</b>
Figure 4.9	Angle obtained from the 2 center point of mobile robot.	<b>57</b>

## CHAPTER 1

### INTRODUCTION

#### 1.1 Overview



This section includes a few topics such as motivation, problem statement, objectives and scopes of study. It outlines the motivation regarding this proposed project with existing problem in real life. Besides that, objectives of this project set aim to solve the problem existed in our life. Furthermore, the scope of study includes the area study of experiments, limitation and target of the project.

## 1.2 Problem Statement

Internet of Things (IoT) application in real life works on wireless interconnectivity of devices and cloud wirelessly, where the type of wireless communication method is important for data exchange and transfer, without it data are not able to go from one end to another. Nowadays, there are wide range of wireless communication to control mobile robot without physically present at a location. In order to effectively transfer and exchange data with cloud, a reliable and effective type of wireless communication is required, so that the limitation of range is able to overcome without affecting the performance of the system.

Next, the controllability of mobile robot have to be flexible rather than can be control using laptop with user interface due to its weight and size that directly affects its mobility. With the rise of smartphone era, task no longer restrain to be perform only by laptop, mobile applications is created by developers to diversify the functionality of smartphone and to overcome restrain. An application should be develop to substitute the task performed by laptop and for the control of mobile robot. Application of the device platform should be programmable, have wirelessly connectivity, mobility and user-friendly.

A mobile robot that situated a place that user hard to reach or estimate surrounding area can be said as blindfolded and hence is one of the challenge. This is because obstacle ahead or around the mobile robot have the possibility to deal damages on it as mobile robot is unable to avoid on itself and lack of self-awareness on surrounding hazard. Camera on-board might be one of the effort to solve this issue and it requires machine vision technique but the camera is unable to rotate and monitor the surrounding of mobile robot at a time. Thus, obstacle detection feature must be implement to any of the mobile robot system.



### 1.3 Motivation

Remote controlled mobile robot nowadays are everywhere using Bluetooth and WiFi technology as a communication channel between the user and mobile robot. Using method above are common in the research area because both proves to provide effectiveness in controlling the mobile robot. This project was inspired by the advancement of internet connectivity with every WiFi-enabled devices, sensors or machines and introduction of Internet of Things (IoT).

As internet has become part of human life, it revolutionised human way of living and now with a simple touch anyone could access the world. Internet does make impossible things to possible in every way such as getting in touch with people around the world, gathering information without camping in the library, and knowing news from every corner of the globe. With introduction of WiFi, the rate of American went online has increased drastically according to Pew Study from 29% to 71% since 2000 until as of today's date [2].

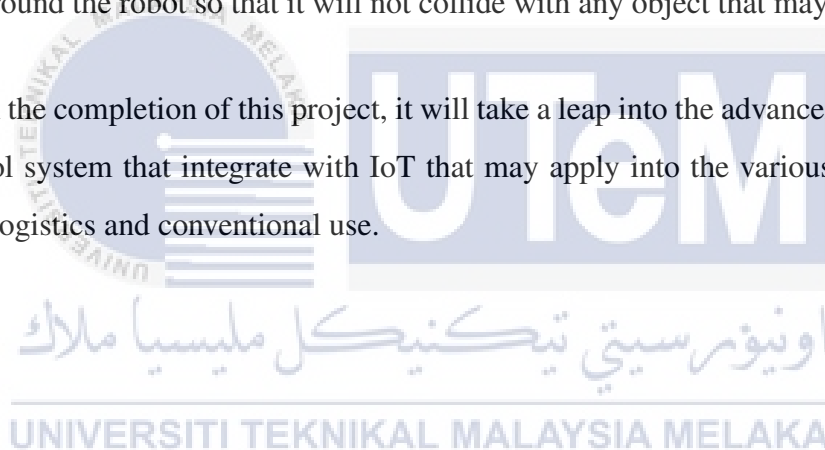
Internet of Things (IoT) are making a buzz these days, it is built based on the network of sensors and cloud computing with application. Data are collected into the cloud in real time and work around the clock in order to provide flexibility and accessibility, it is like a medium of exchanging data between sensors and user with further features like analysing, leveraging or interpreting [3]. A research conducted by Juniper predicted that there will be 38.5 billions of devices connected to each other and also the internet. It is all about making everything around us smart, easier and efficient [4].

Next, Android is an open source platform aims to introduce more innovative and successful product that could improve users' experience on mobile. It provide all developers the freedom to customize to ones need, however every legitimate developers or users are required to join in the Android Compatibility Program that would require developers to achieve the status of "Android Compatibility" as one of the effort to maintain a common ecosystem [5]. Android provides unlimited resources that one needs to create an application, its flexibility and customizability are much easy to fuse in any projects with existing library and back-end

support. Using Android platform in this projects also due to large numbers of user with similar device platform, increasing chances of application being notice and hence increase feedback on the development of application based on their needs.

This project will implement a system that able to control the mobile robot wirelessly and remotely. A mobile robot with holonomic movement are used as conventional wheeled robot could not move in any direction instantaneously without performing sort of motion to change the robot heading. Then, an application are needed to control the movement and interaction of mobile robot where it is able to communicate with the controller on mobile robot itself wirelessly, so that every command sent would be receive and execute by the controller. Thirdly, user that control the mobile robot may not able to estimate the distance between obstacle and robot, therefore an obstacle detection ability is aim to provide a safe perimeter around the robot so that it will not collide with any object that may damage itself.

With the completion of this project, it will take a leap into the advancement of mobile robot control system that integrate with IoT that may apply into the various fields such as warehouse logistics and conventional use.



#### **1.4 Objectives**

The objectives set for this project are:-

1. To design and develop a mobile robot that can be controlled by Android platform device wirelessly.
2. To develop an Android User Interface application for mobile robot.
3. To analyse the distance and obstacle detection using ultrasonic sensor.

## 1.5 Scope of Study

The scopes of this projects are:-

- (a) Experiments of mobile robot are conducted on flat and even surface such as brick, ceramic, concrete, limestone, mosaic and etc.
- (b) The activity of mobile robot is limited to control by Android platform device that only installed with self-developed application.
- (c) The activity area of mobile robot is limited to Wireless Local Area Network (Wireless LAN) environment.
- (d) Wireless communication module and device which comes with specifications that user should not operate beyond the limit.
- (e) Android application to control and monitor the activity of mobile robot is designed and developed using Android Studio.



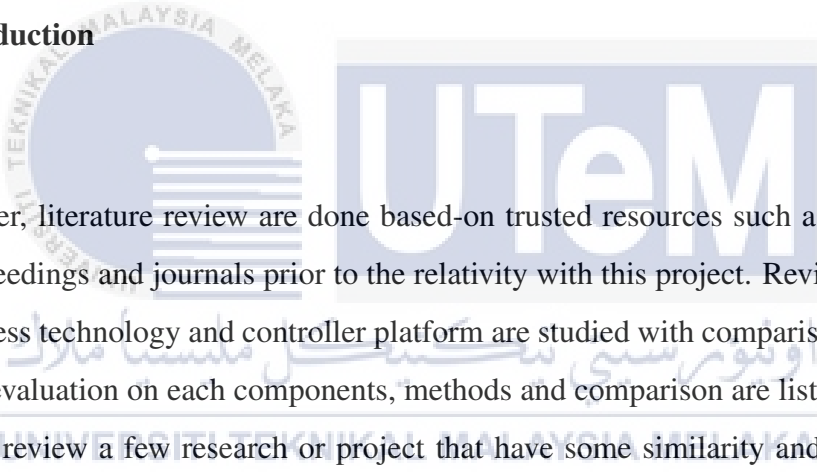
## 1.6 Summary

Problem statements were outlined in this section with the rise of motivation towards those problem and objectives of study were clearly listed out above. The scopes of this projects are set accordingly to fulfil the objectives of study. The following chapter would be literature review on this project.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Introduction



In this chapter, literature review are done based-on trusted resources such as articles, conference proceedings and journals prior to the relativity with this project. Reviews on related works, wireless technology and controller platform are studied with comparison within each others. The evaluation on each components, methods and comparison are listed below. This section is to review a few research or project that have some similarity and has been carried out by other researchers. These projects utilized different approach of communication method within robot and controller, different robot platform or microcontroller and controller to robot interacting method.

## 2.2 Related Works

G. Ersahin and H. Sedef has studied and developed a tablet computer controlled mobile robot that communicate using Wi-Fi. The mobile robot is equipped with on-board web camera that is used to obtain real-time video feed and send to controller wirelessly. The mobile robot itself is controlled with Raspberry Pi that used ARM processor and programmable by using Linux operating system. The Raspberry Pi consists of general purpose input/output (GPIO) ports with PWM enabled are responsible to control the wheel movement through L293D motor driver that supported with 12V DC power supply. Besides that, Raspberry Pi is used to process video feed from the camera, communicate and transmit data to the controller wirelessly through Wi-Fi device.

Surveillance Robot is proposed and studied using communication of Android platform smartphone and remote user over the internet. The Android smartphone on the mobile robot served for two purposes, one acts as a communicating bridge between Arduino UNO microcontroller and the controller, where it receives command from the remote host and relay them to the microcontroller using USB On-the-go cable. The second is it used for live video feed transmission to the remote host over the Internet using internet sockets [6]. The MATLAB Graphical User Interfaces is designed and created using GUIDE toolset to control the mobile robot remotely, however the Android application is coded using Android Software Development Kit (SDK) in Java language [7].

Development of smartphone remote control robot conducted in reference [8] are based on Android operating device that coded in Java Language. The communication between robot host with 802.11N compatible wireless network card and smartphone with Wi-Fi functions are using Wi-Fi network. There two schemes of indoor Internet sharing, point-to-point network and Wireless Hub. For this system, it used the Ad-Hoc mode which belongs to the first scheme that does not require through a wireless router. TCP protocol is used due to its reliability, order and sequence of data transmission.

### 2.3 Wireless Technology

Nowadays, there are various wireless technology that can be choose to implement in Internet of Things (IoT) and Machine-to-Machine (M2M) projects communication method [9]. IoT-based robotic system requires a medium for interchanging data and information whether its within an area or remote from far. In this project, wireless communication is required in order to communicate, control and interface with the mobile robot through Arduino. Reviews under Wireless Technology includes Wi-Fi, Bluetooth/BLE and WiMAX.

Wi-Fi or known as Wireless Fidelity is one of the technology that connecting two or more device with similar radio waves wirelessly without the connection of cables. It can be said as the most notable technology as it have achieved an enormous growth in these few years [10]. This type of wireless network that operate based on the 802.11 standards [11] that developed by the Institute of Electrical and Electronics Engineers (IEEE) which available at 2.4GHz UHF and 5GHz SHF ISM bands [12].

Bluetooth or Bluetooth Low Energy (BLE) are technologies that used to transmit data over a limited distance which mostly used at hand-held or mobile devices such as smartphone and tablet. It is a UHF radio waves that used to be standardized under IEEE 802.15.1. [12]. Today's Bluetooth connections basically are supported by most of the mobile devices which give it a benefit of convenience over others device. However, the disadvantage of using Bluetooth are high and jittering on latency with instability issues [13].

Table 2.1 below shows the table of comparison of frequency, nominal range, signal rate, typical current absorbed and output power between Bluetooth and Wi-Fi technology [14]. Next, World Interoperability for Microwave Access stands for WiMAX in short that

Table 2.1: Comparison of Bluetooth and Wi-Fi specifications.

Specification	Bluetooth	Wi-Fi
Frequency	2.4 GHz	2.4 & 5 GHz
Nominal range	10 m	100 m
Signal rate	1 Mbps	54 Mbps
Typical current absorbed	1-35 mA	100-350 mA
Typical output power	1-10 mW	30-100 mW

runs at IEEE 802.16 standards is able to deliver low cost yet high speed wireless broadband with larger coverage compared to WiFi. Although WiFi and WiMAX serve the same purpose of interchanging data but both are designed to suit two complete different needs. Table 2.2 shows the corresponding spectrum, theoretical data rates and typical data rates of WiMAX fixed and mobile standards [15]. Wi-Fi proved itself to be a better choice over

Table 2.2: Comparison of data rates for fixed and mobile standards.

Standards	Spectrum	Data Rates	Typical Data Rates
Fixed	20 MHz	75 Mbps	20-30 Mbps
Mobile	10 MHz	30 Mbps	3-5 Mbps

the rest where it is able to provides faster rate of data transmission, more secure network (properly configured) and greater signal range compared to Bluetooth technology [16]. As to control the robot remotely at two different location and in the case as Wireless network are presence, Wi-Fi technology is chosen over WiMAX for implementation in this project due to the multiple reason. Broadcasting range of WiMAX requires more energy and high cost infrastructure to sustain whereby Wi-Fi can be installed easily at low cost providing access point to users. Most of the smart device or electronic gadgets with WiFi enabled nowadays mostly are compliant to Wi-Fi 802.11 standards rather than WiMAX 802.16 standards [15].

## 2.4 Controller Platform

In order to receive, transmit and execute command from the remote host, this project requires a controller to communicate with the remote host using Wi-Fi module in order to control the movement of mobile robot. The controller also requires to process data or information that obtain through the sensors that attach to it and send to the remote host for monitoring and controlling purpose, thereby it is important to choose a controller with specifications within the allocated budget that are suitable with the purpose and operation of the system, such as the complexity of whole system, number of task, repetitive of task and complicated decision making.

Arduino Mega 2560 as shown in Figure 2.1 is a board-based open source microcontroller using ATmega2560 chip that designed and manufactured by Atmel Corporation consists a total of 70 General Purpose Input/Output (GPIO) that includes 54 Digital I/O and 16 Analog I/O, clock speed of 16 MHz, 256 kb of flash memory, a reset button, ICSP header, power jack and 4 UARTs. With the price of \$35, this microcontroller operates at 5V DC with the maximum input voltage of 20V and can be connected to computer using USB cable for programming or power up. It is also compatible to most of the modules or shields that are designed for the former version of Arduino. Arduino microcontroller can be coded with C or C++ language using Arduino integrated development environment (IDE). It is well-known and widely use by beginners, professionals, researchers, hobbyists and engineers for completing various research and projects due to its low cost, open-source, cross-platform microcontroller and easy-to-use programming environment [17].



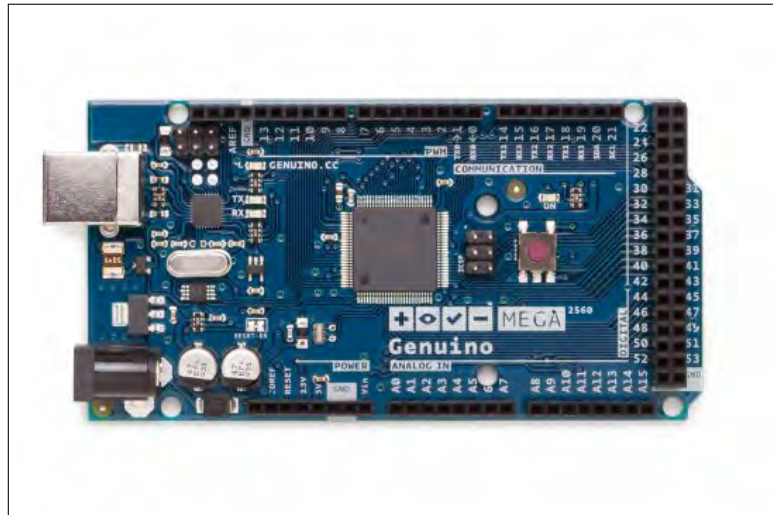


Figure 2.1: Arduino Mega 2560 microcontroller board

Raspberry Pi 2 as shown in Figure 2.2 above is System-on-chip (Broadcom SoC) board that includes an ARMv7 processor that runs on 900 MHz and comes with 1 GB of RAM, 250 MHz GPU and Digital Signal Processor (DSP). With price tagged at \$42, it consists of 40 GPIO pins, 4 USB ports, HDMI port, 3.5 mm jack, microSD card slot and Ethernet network port [18]. This credit card-sized tiny-computer that runs on Linux operating system is a powerful processor that mainly uses Python and Scratch as the programming language. There are research that use the combination of Raspberry Pi and Arduino, Arduino as the middleman of the system to interact between Raspberry Pi and the sensors or actuators. It is responsible to execute the control system of the robot, retrieve data from sensors and execute command for robot's movement while Raspberry Pi is used to process the data obtain from Arduino and interface with remote host through wireless connection [19].



Figure 2.2: Raspberry Pi 2

Beagleboard shown in Figure 2.3 above is Beaglebone Black microcomputer that has a 1 GHz ARM Cortex-A8 processor that comes with 512 MB of RAM, 4GB on-board expandable storage, double 32-bit PRU microcontroller and 3D graphics accelerator. This upgraded version of Beaglebone at a price of \$54.95, consists of 65 GPIO pins, 1 USB port, micro HDMI port, microSD card slot, micro USB port and Ethernet network port [20]. Beaglebone black is one of the Beagleboard that widely use around the world that due to its famous 10 second Linux boot speed and the board compatibility with software such as Android, Debian, Ubuntu and Cloud9 IDE on Node.js with BoneScript library [21].

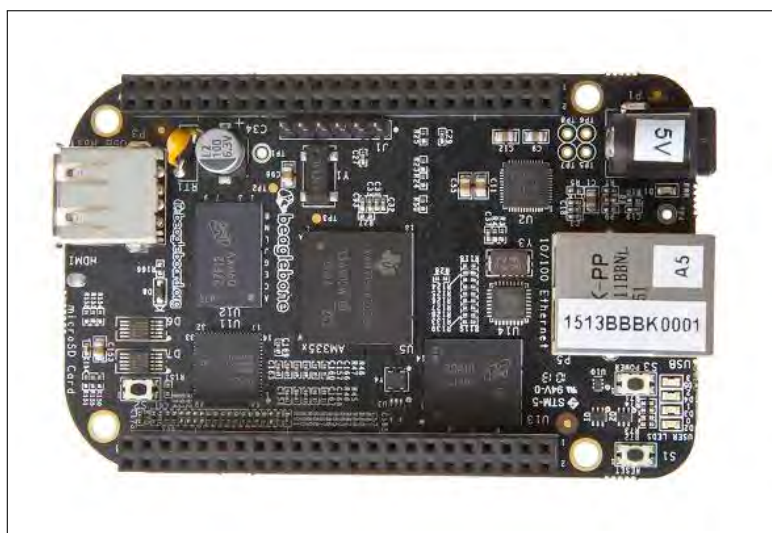


Figure 2.3: Beaglebone Black.

Udoo x86 Basic is a powerful x86 maker board with Arduino 101-compatible platform. This board retailed at price of \$125 armed with CPU 2GHz Quad core Intel Atom X5-E8000 and Intel HD Graphics. It has 2GB DDR3L RAM, 20 GPIO which is Arduino 101-compatible, HDMI port, miniDP++ connectors, ethernet port, touchscreen port, audio and mic jack connectors, camera connector, micro sd slot (boot device) and SATA connector. Udoo x86 is able to run Windows operating system, any x86 linux distribution and Android platform [22].

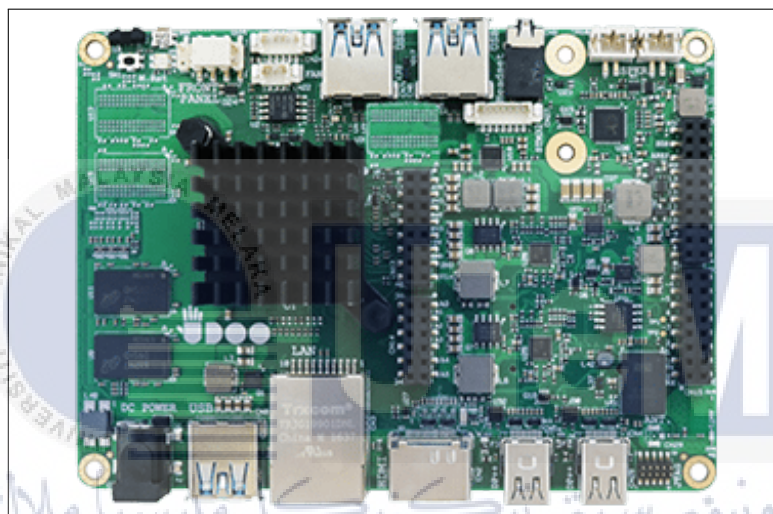


Figure 2.4: Udoo x86 Basic.

## 2.5 Android Platform

Being the world's most famous smartphone platform which capable of powering hundreds of millions mobile devices of the world, Android is acquired by Google in 2005, currently with global partnership and huge installed base that contributed by the community of open source Linux. Since 2008, Google has rolled out 24 versions of Android Operating System (OS) in market with substantial improvement in order to develop, grow and also improve the operating system.

Being an open-source platform, it is completely opposite to close-source software which allowed source code to be view, download, modify, enhance and redistribute by anyone without requesting any sort of charges or royalties. Users that using its platform can fully customize their homescreens' launchers and widgets, or even reskin their devices at their satisfaction. It also has compelling development framework and complete Java Integrated Development Environment (IDE) which provides developer to develop, create and deploy the application with endless creativity and possibility over a broad range of existing devices. With Google Play, application developers can put their program to publish and sell with predefined placement based on its popularity [23].



Figure 2.5: Android Logo.

## 2.5.1 Android Stack

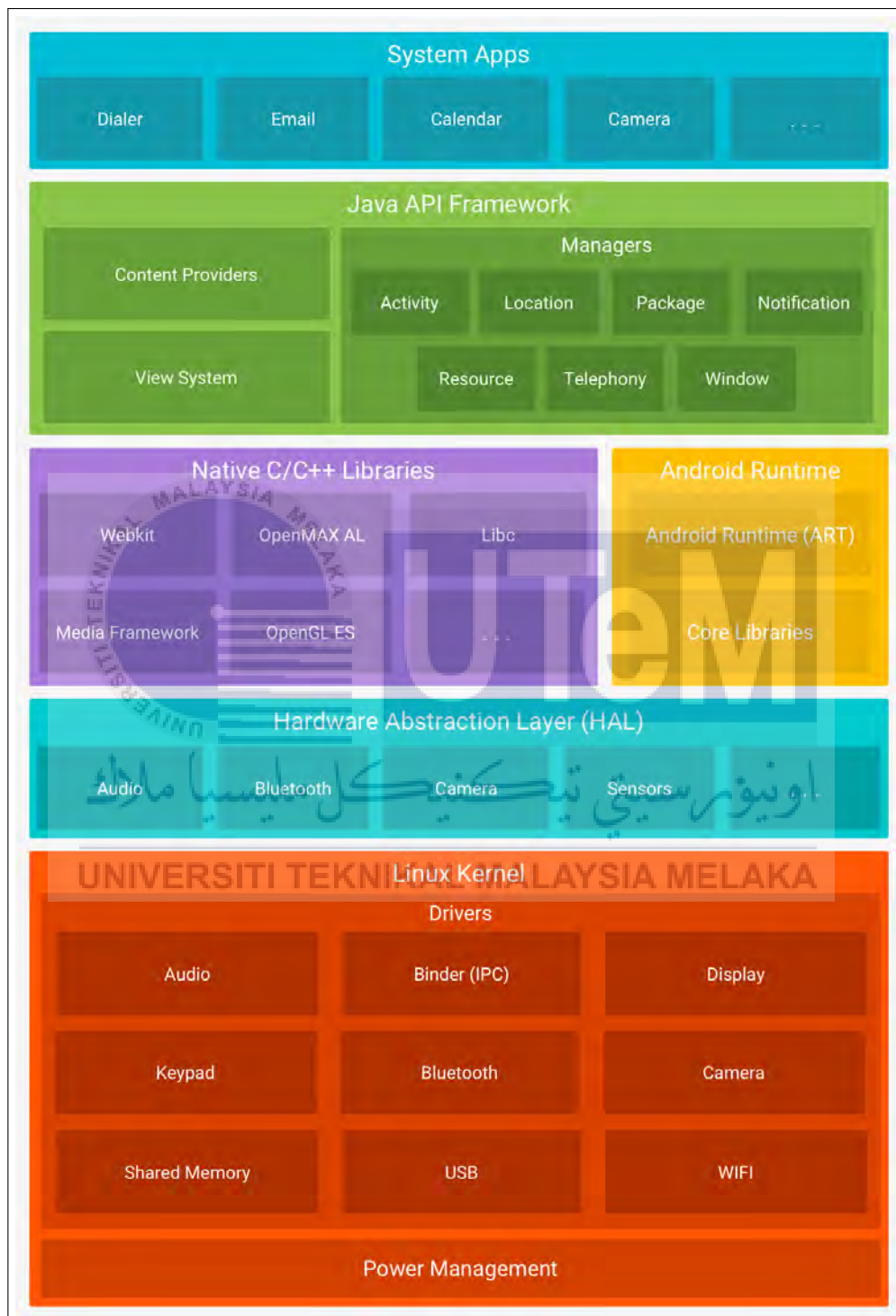


Figure 2.6: The software stack of Android platform [1].



There consisted six major components of the Android:-

1. System Apps
2. Java API Framework
3. Native C/C++ Libraries
4. Android Runtime
5. Hardware Abstraction Layer (HAL)
6. Linux Kernel

System applications is a set of core apps that come with the platform such as clock, calendar, contacts, messaging, dialer, camera, and others. These apps has no different with third-party app that a user choose to install, instead user can decide the default app to perform certain functionalities. The system apps play a role as the core capabilities for third-party app, as such to capture photo from WhatsApp, WhatsApp does not require to have the camera function but instead it can invoke the camera that come with the system.

Under Java API Framework there are content providers, view system and managers. Content providers is a feature that enable one apps to access data from another apps and view system is rich with extensive of user interface (UI) for apps development. Managers present under the framework includes activity, location, package, resource, notification, telephony and window of the system [24].

- Activity manager monitors, controls and manages the activity of applications that running in the system.
- Location manager access to the location services of the system to allow apps to request updates on the device's location periodically.
- Package manager use to retrieve any data or information from packages that installed with the application on the device.
- Resource manager keeps track of all the non-code assets that relate to an application.

- Notification manager manages all the custom alerts from all application and notify the system such as flashing LED, vibrating and playing notification tone.
- Telephony manager handles all the telephony services such as mobile networks status and information.
- Windows manager manages the switching layout, window and interface of the apps to system.

For Native C/C++ Libraries, it consist of system's core components and also services that is written in C/C++ language that cannot access directly unless with the assist of Java API framework. Android Runtime (ART) is built to execute multiple process or application through virtual machine (VM) using DEX files with the purpose of minimizing the memory footprint. There are a few features of ART which are Just-in-time (JIT) and ahead-of-time (AOT) compilation, garbage collection (GC) optimizing, and refined debugging assistant.

Hardware Abstraction Layer (HAL) introduces a higher level of framework APIs to the hardware integrate in the device. It consists of various hardware module that comply with the components, when the framework API wanted to access the device component, the library module is being extracted from the system [1].

Lastly, Linux Kernel is the bedrock of Android platform where others layer would rely or its service such as security, power and memory management. It also provide the communication between the software and hardware by binding and encrypting each others [25].

## 2.6 Lightweight Communication Protocol for Robotic Application

Communication is important in few robot application, one of it is robot control. Robot that needs to be control remotely requires minimal delay in response to operate properly. Such delay in remotely controlled robot's communication may result in an undesired output and could be hazardous when involves with industrial or rapid motion robots. There are several protocols that serve on non-mobile and non-battery powered robots are not optimized for inverse of the stated robots [26].

In order to serve such purpose, an optimized protocol must be created to suit the application. There are two new standard lightweight application protocols that emerged and gained much attention which are IBM-designed Message Queuing Telemetry Transport (MQTT) [27] and Constrained Application Protocol (CoAP) designed by internet Engineering Task Force (IETF) [28]. These two protocols have very different principle in designing, CoAP is a protocol that designed to provide HTTP like request-response mechanisms in a constrained environments while MQTT is a messaging application protocol optimized for pub/sub-based systems. Furthermore, CoAP has been equipped with mechanism to push resource representations from servers to relevant clients, which makes it is also appropriate to implement on pub/sub applications [29].



### 2.6.1 Message Queuing Telemetry Transport (MQTT)

As this section title depicts, Message Queuing Telemetry Transport (MQTT) is an open source with royalty-free license protocol that is approved its membership as one of the OASIS standard at 2014 on Version 3.1.1. It is a very simple messaging protocol that operates on publish or subscribe mechanism that designed to use on constrained device such as limited capabilities on system processing and memory, or required to transmit data over low-bandwidth networks.

Before transmission of data, it is important to setup the connection between the publisher (clients that publish data to the broker) to the broker, and also the broker to subscriber (clients that interested on selective topics of interest and thereby receive the related published data). After connection is successful, the publisher will publish any data/message to the broker, then it will be sent to all subscribers that subscribed to the particular topic. To boost reliability on MQTT, it offered three types of Quality of Service (QoS) modes, QoS 0, QoS 1 and QoS 2. For QoS 0, messages are sent at once without requiring ACK, while for QoS 1, ensuring message delivery by requiring ACK and lastly QoS 2, ensured the transaction is delivered once by requiring the publisher to do the following sequence (PUBLISH>PUBACK>PUBREL>PUBCOMP) [26, 29].

### 2.6.2 Constrained Application Protocol (CoAP)

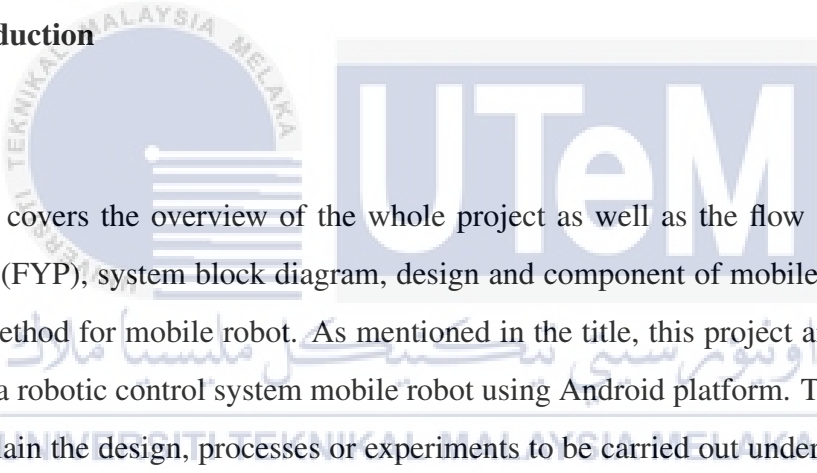
Constrained Application Protocol (CoAP) is developed for application on device that have low capabilities on processing and power. CoAP uses UDP as an underlying transport layer and Representational State Transfer (REST) method, which is using request and response that doesn't required a complex translator.

There are two modes that CoAP support, which are reliable and non-reliable. Reliability in CoAP is overcome by using Confirmable (CON) messages. For each CON message by the server, this will be replied by Acknowledge (ACK) packages to the source of data. Besides that, it also support "Piggybacked Response" ACK message that can be included in the response message of another message to reduce and optimized communication. CoAP also have major design objectives which are reduce message overhead where defined as the memory utilize by the broker to store information of a message, and limit the packet fragmentation which can be said that packet size that larger than the Maximum Transmission Unit (MTU) size require to be divided into smaller fragment in order to enable them for transmission; it includes a short fixed-length compact binary header of 4 bytes followed by compact binary options [26, 29].

## CHAPTER 3

### METHODOLOGY

#### 3.1 Introduction



This chapter covers the overview of the whole project as well as the flow chart for Final Year Project (FYP), system block diagram, design and component of mobile robot, and the evaluation method for mobile robot. As mentioned in the title, this project aimed to design and develop a robotic control system mobile robot using Android platform. This chapter are going to explain the design, processes or experiments to be carried out under this project in order to evaluate the mobile robot at certain aspects.

### 3.2 Project Overview

A set of experiments are will be carried out to evaluate, analyse and determine the performance of the mobile robot. This project are required to be accomplished within two semester of the course, therefore for the first semester, literature review of various journal, article and related information and overall system design; second semester emphasize on the system implementation and research of the system.

This project can be divided into four phases which are literature review, system design, system implementation and experiment of the system. System design is where the concept of the project is drafted and revised, system implementation is phase where the designed system in put into use with the mobile robot and lastly research of the system according to the method propose in the methodology.

The Figure 3.1 shows the flow chart for this Final Year Project. For first phase of FYP, literature review on materials related to this topic has to be conducted before proceed to the motivation and problem statement. Then, motivation indicated the desire and reason or conducting this title, and problem statement shows the challenge faced by previous research or improve of previous research. Next, the study on Android UI and application development, design of the mobile robot, and study on the obstacle detection mechanism on mobile robot.

The second phase which is FYP 2 includes the assembly of mobile robot, develop and design Android application for control purpose with the setup of communication channel between application and mobile robot. Lastly, when system is successfully implemented, experiment is conducted to analyse on various performance of the mobile robot such as movement limit, speed, connectivity, response rate, and object detection.

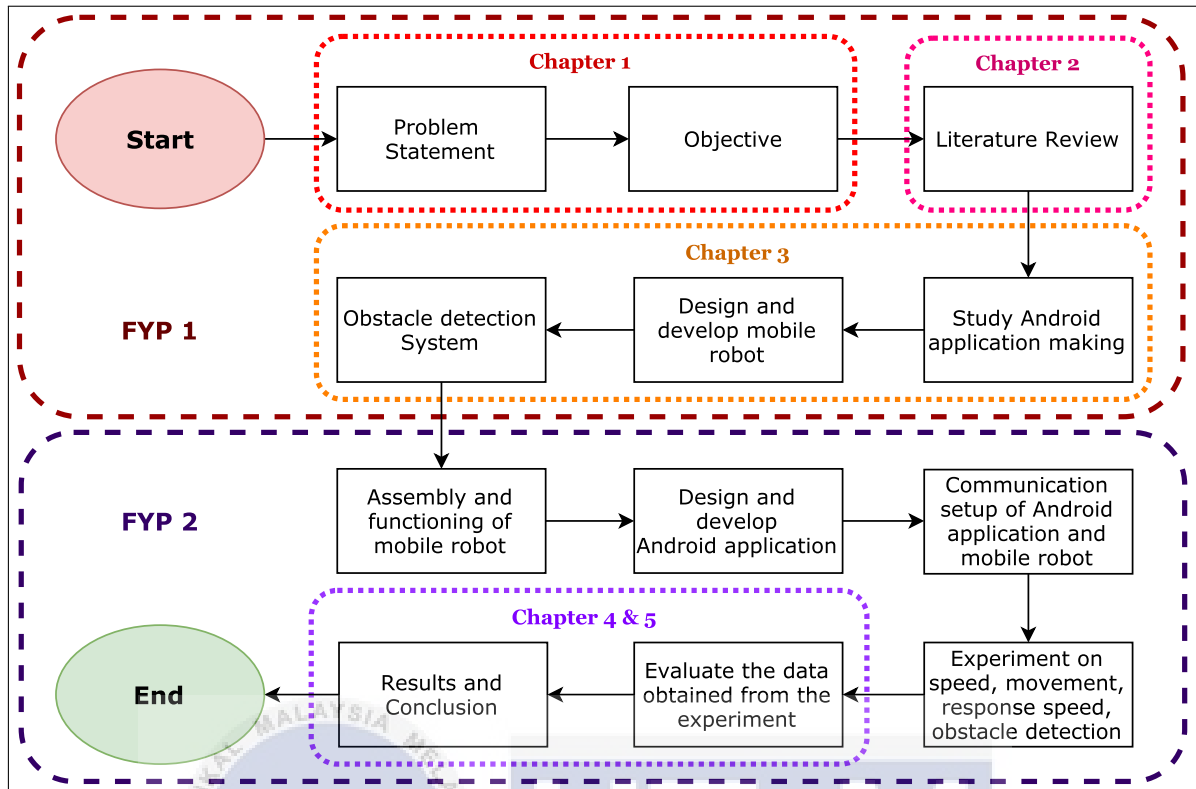


Figure 3.1: The overall flow chart for this project.

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

### 3.3 System Overview

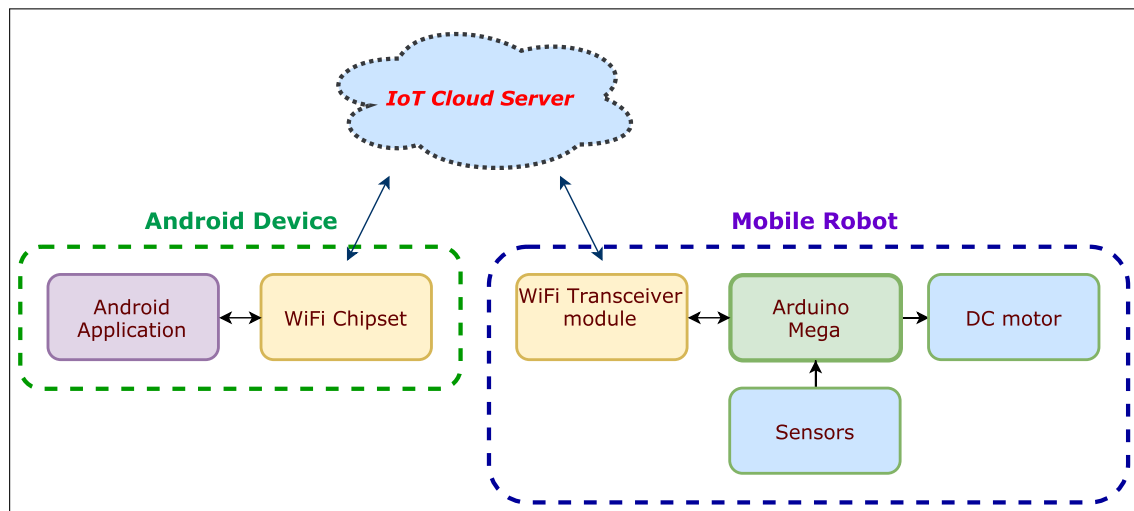


Figure 3.2: The overall system in block diagram.

The system of project is projected on Figure 3.2. It is explained by portrait the system with block diagram and cloud server; it is divided into 2 different components, Android device and mobile robot. From left to right, the first component which is Android device shows the combination of Android application and built-in Wifi chipset, data is transmit to mobile robot by Android Wifi through IoT cloud server. Next, the Wifi transceiver module that integrate with Arduino will prompt the IoT cloud server for data and information updated by Android application. Arduino Mega that received command in the form of data will execute accordingly to DC motor and listen signal from sensors.

### 3.4 Components of Mobile Robot

In this section, the hardware that make up the mobile robot will be listed here with its specifications and details. The components involved robot base, omni wheels, miniature DC geared motor, L293D DC motor driver module, ESP-8266 Wifi module, ultrasonic sensor and Arduino Mega2560.

#### 3.4.1 Robot Base

The mobile robot for this project is not complicated, it only requires 1 base or chassis to mount the DC motor with Omni wheel and carry the microcontroller. It is difficult to look for robot base chassis in the market as customized component would cost more, so 3D printed base with self-designed is more preferable. Figure 3.3 below shows the board that was self-customized using Solidworks and the material for chassis is Poly Lactic Acid (PLA). After design using Solidworks, the file is saved to STL format for 3D printing. The design of mobile robot base without the mounting of DC motor and omniwheel is attached below.

There are two choices of materials for 3D printing which are Acrylonitrile-Butadiene Styrene (ABS), an oil based plastic and Poly Lactic Acid (PLA), a biodegradable plastic. Table 3.1 was built below to add a comparison between the 2 materials. Therefore, PLA is more preferable than ABS to used as printing material.

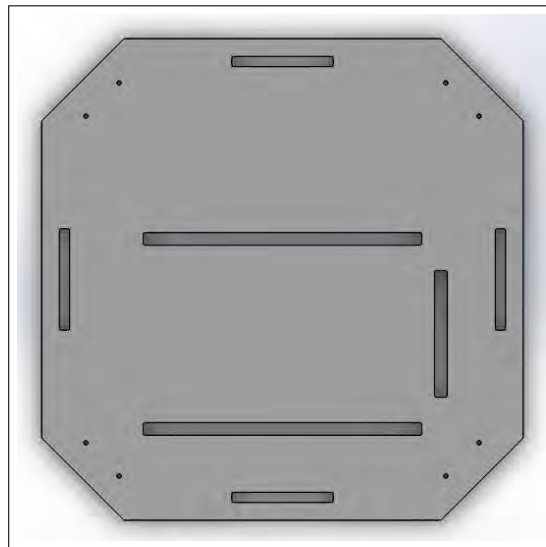


Figure 3.3: Design of mobile robot base using Solidworks.

Table 3.1: Table of comparison between ABS and PLA.

Properties	ABS	PLA
General Properties	Strong, mild-flexible and temperature resistance plastic	Decomposable, Rigid than ABS, various colour & opacity
Degradability	Non-biodegradable	Biodegradable
Smell when printing	Plastic odour	Sweet smell
Part Accuracy	Prone to Warping (Need hotbed)	Less warping (Hotbed not needed)
Long-term Storage	Reduced Quality	Discolouration



### 3.4.2 Omni wheel

Omni wheel is a different concept of wheels that attached with small cylindrical discs along the circumference of the wheel and perpendicular to the rotating axis. It can be rotate as normal wheel and slide laterally freely, this features enable holonomic movement to be employ at various mobile projects. The mobile robot movement can be change instantaneously in any direction without going through additional movement to change the direction. Figure 3.4 below shows the pair of orange colour Omni wheel and Table 3.2 shows the specifications of the Omni wheel.

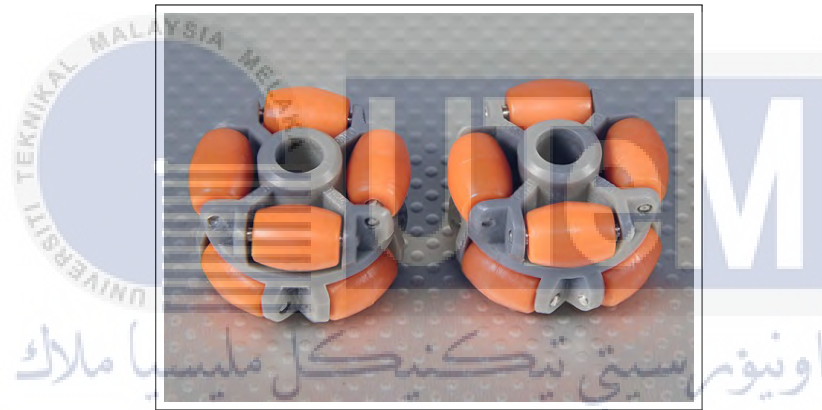


Figure 3.4: A pair of Omni wheel.

Table 3.2: Specifications of Omni wheel.

Parameter	Specification
Material	Plastic
Shaft Diameter	15 mm
Dimension (L*W*H)	75 x 75 x 40 mm

### 3.4.3 Miniature DC gear motor

For the movement of holonomic, the mobile robot needs 2 pairs of DC motor to drive 4 omni wheels. Each motor can be drive independently relates to others motor using motor driver, this would enable the robot to move in any direction by rotating different pair of wheel at a time. The Figure 3.5 and Figure 3.6 below shows the actual item and its dimension respectively whereas Table 3.3 listed the specifications for this type of motor.



Figure 3.5: Miniature DC gear motor.

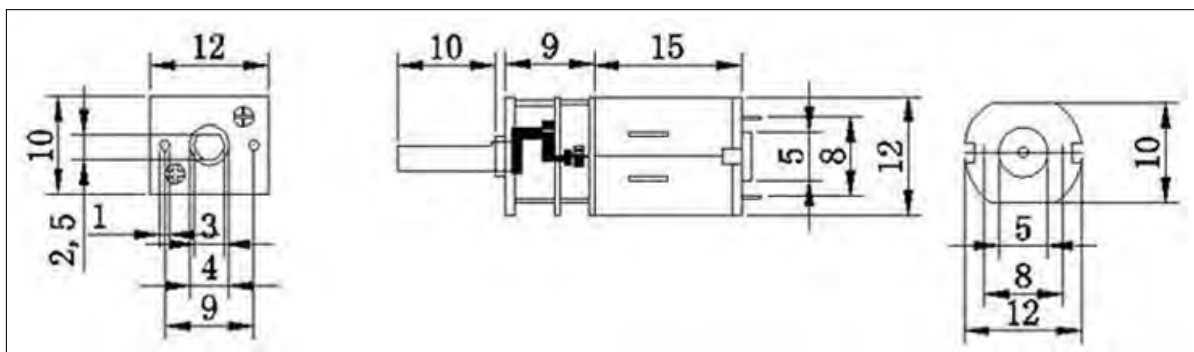


Figure 3.6: Dimension of the miniature DC gear motor.

Table 3.3: Specifications of miniature DC gear motor.

Parameter	Specification
Rated Voltage	6V DC
Rated Current	160 mA
Rated Speed	150 rpm
Loaded Speed	120 rpm
Rated Torque	0.3 kg/cm
Stall Torque	2.4 kg/cm
Dimension (L*W*H)	35 x 12 x 10mm

#### 3.4.4 L293D DC Motor Driver

Motor driver is used to control the direction of rotation and the amount of power supply to drive the motor. This module will ease the user to change the rotation of motor only by feeding signal into different port. Inside this module contained of two L293D motor drivers and one 74HC595 shift register. The L293D chip is a type of H-bridge that can be say as a circuit that enables the switching of motor direction either clockwise or anticlockwise by alternate the input of voltage. There are some features which includes:-

- Provides up to 2 connections for 5V hobby servo motor.
- Support up to 2 5V DC servo motors or 2 stepper motors or 4 DC motors.
- Support individual 8-bit speed selection for 4 bi-directional DC motors.
- Each of 4 H-bridge could output 600mA (Max 1.2A) and thermal shutdown protection.
- Pull-down resistors for motor to maintain off during module power up.
- 2 external terminal power interface, to separate power supplies for logic and motor.
- Arduino compatible module.

Figure 3.7 shows the H-bridge motor driver module and Table 3.4 depicts the specifications of the driver stated.

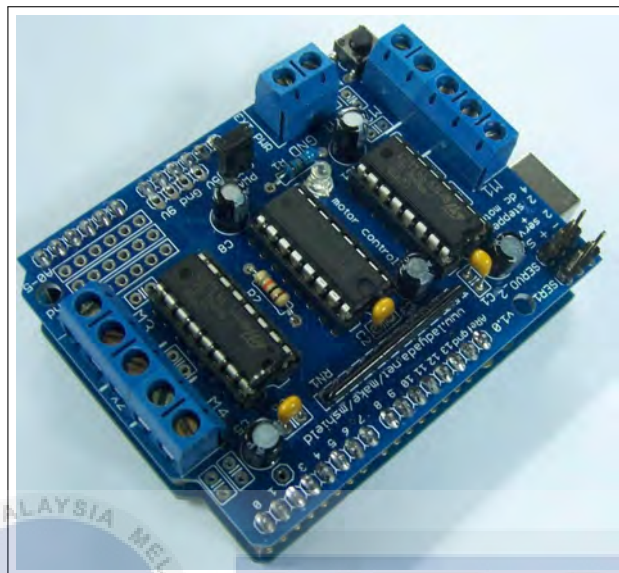


Figure 3.7: L293D 4 H-bridge motor driver module.

Table 3.4: Specifications of L293D Motor Driver module.

Parameter	Specification
Voltage Supply	4.5V - 36V DC
Output Current	600 mA
Peak Output Current	1.2 A
Channel	4 H-Bridges
Dimension (L*W*H)	70 x 53 x 20mm

### 3.4.5 ESP-8266 Wifi Module

ESP-8266 Wifi module is constructed with built-in System on Chip (SoC) that integrated with TCP/IP protocol, TR switch, LNA, balun, matching network, temperature sensors, power amplifier and etc. that support 802.11 bgn protocol, Wi-fi direct (P2P), soft Access Point (AP), and antenna diversity. ESP-8266 module enable any microcontroller the access to Wifi network seamlessly with this cost-effective method. So, in order to enable Arduino to interact with Android application, it requires to transmit and receive data to the IoT cloud through ESP-8266 because Arduino did not comes with an Wifi features on the board. This module is comes with a pre-programmed firmware that use AT commands. It contains of 8 pins which are Vcc, GND, RST, CH\_PD, GPIO0, GPIO2, TXD and RXD. The Figure 3.8 shows the hardware of the Wifi module revision 1 and Table 3.5 that shows the label of each pin on the module.

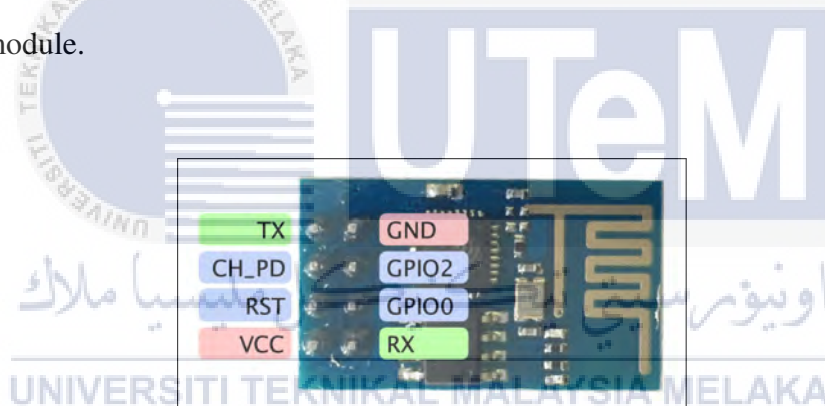


Figure 3.8: ESP-8266 Wifi module revision 1.

Table 3.5: Table of ESP-8266 pin description.

Pin	Description
Vcc	Voltage supply of 3.3V
GND	Ground
RST	Reset (Active LOW)
CH_PD	Chip power down
RXD	Receive data pin at 3.3V
TXD	Transmit data pin at 3.3V
GPIO 0	General Purpose I/O pin 0
GPIO 2	General Purpose I/O pin 2

### 3.4.6 Ultrasonic Sensor

HC-SR04 is an economic ultrasonic ranging sensor that is able to detect range from 2cm to 400cm and with accuracy tolerance up to 3mm. The HC-SR04 module is equipped with a receiver, ultrasonic transmitter and a control circuit. This module only comes with four pins which are VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground). Figure 3.9 shows the ultrasonic sensor module and Table 3.6 shows the specifications.

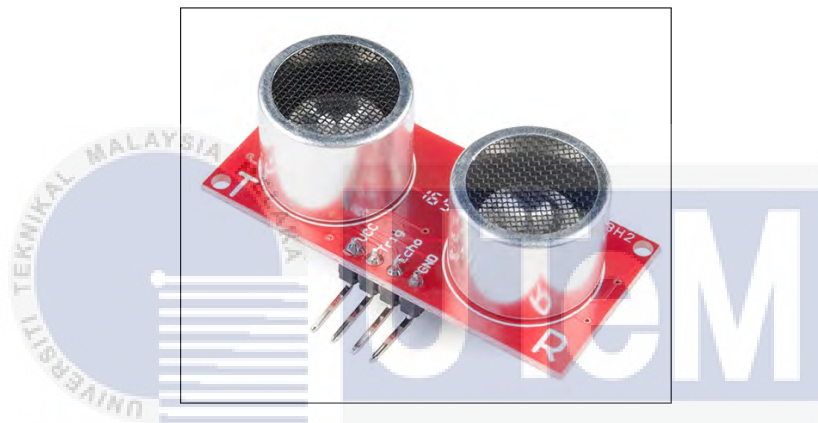


Figure 3.9: Ultrasonic ranging sensor, HC-SR04.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table 3.6: Specifications of HC-SR04 module.

Parameter	Specification
Operating Voltage	5V DC
Operating Current	15 mA
Measure Angle	15°
Ranging Distance	2cm - 4m

### 3.4.7 Arduino Mega 2560 microcontroller

Arduino is a well-known microcontroller board with various range and design to suit every project by hobbyists, researchers or even students. It used Atmel AVR microcontroller chip on its board and specifically ATmega8, ATmega168, ATmega328, ATmega1280 and ATmega2560. This are a few benefit of this type of microcontroller which are ease of use, user-friendly IDE, less expensive compare to others development board and various module that made compatible with Arduino.

This project required 12 pins to control the Omni wheel independently, each motor requires 2 pins for bi-directional rotation and 1 pin for PWM control. With the obstacle detection system, it required more pins, therefore Arduino Mega2560 is selected for this projects as Arduino Uno only consists 14 digital I/O pins and 6 analog input pins but the latter consists of 54 digital I/O pins and 16 analog input pins which give more than enough I/O pins. It is better to have excess pins than to worry for insufficient pins. The Figure 3.10 below shows Arduino Mega2560 microcontroller.



Figure 3.10: Arduino or Genuino Mega 2560 microcontroller.

Table 3.7: Table of Arduino Mega 2560 microcontroller Specifications

Parameter	Specification
Microcontroller	Atmel ATmega2560
Operating Voltage	5V
V <sub>in</sub> (recommended)	7-12V
V <sub>in</sub> (limit)	6-20V
Digital I/O Pins	54 (includes 16 PWM output pins)
Analog Input Pins	17
Clock Speed	16 MHz
Flash Memory	256 KB



### 3.5 Mobile Robot Hardware Design

This section explains about the hardware design of the mobile robot, part of the design is drawn and drafted using Solidworks which is a 3D Computer Aided Design software. Figure 3.11 below shows isometric view of the design in an assembly file. This view enable to see the design clearer with dimension and the combination of x-axis, y-axis and z-axis position. The designed robot base is 4mm thick with 20cm width on each side. For mounting of the Omni wheel with the miniature DC gear motor, each motor and Omni wheel is mounted on an angle of  $45^\circ$  at the corner of base. The DC gear motor is then secured with a cap that used M1.6 nut.

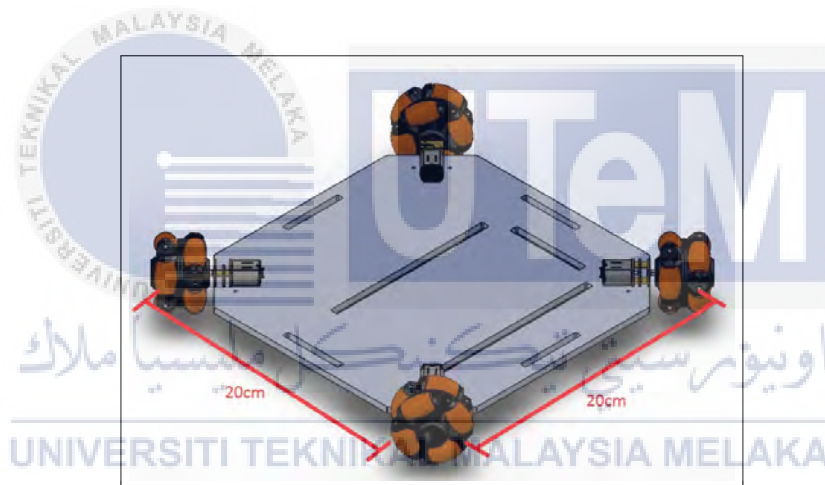


Figure 3.11: Isometric view of the mobile robot hardware.

### 3.6 Obstacle Detection System Design

This system comprises of multiple ultrasonic sensor to detect the presence of an obstacle around and its distance from the mobile robot. One ultrasonic sensor is placed on each side of the mobile robot (marked with red circle) which gives a total number of 4 sensors on it. Every sensor on each side will be faced outwards perpendicular from the side edge of mobile robot base as shown in the Figure 3.12 and will be calibrated according to its effective area and distance of detection.

This obstacle detection system will always active to detects any nearby obstacle of mobile robot and will display warning on the Android application once the distance between obstacle and mobile robot less than safety distance. When the distance detected is less than 30cm between the mobile robot and obstacle, a warning message will be publish to the Android application immediately. The mobile robot will be programmed to reverse in its opposite direction from the detected obstacle until reaches a safe distance once system detects the object.

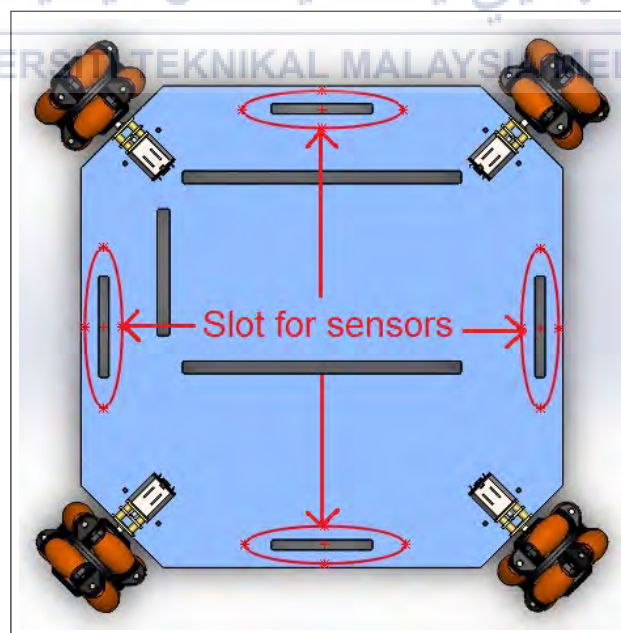


Figure 3.12: Slot for the placement of ultrasonic sensors mounting on the mobile robot base.

### 3.6.1 Flow Chart of obstacle detection system

The flow chart in Figure 3.13 below shows the operation flow of obstacle detection system, it will run in loop continuously when the mobile robot is switched on.

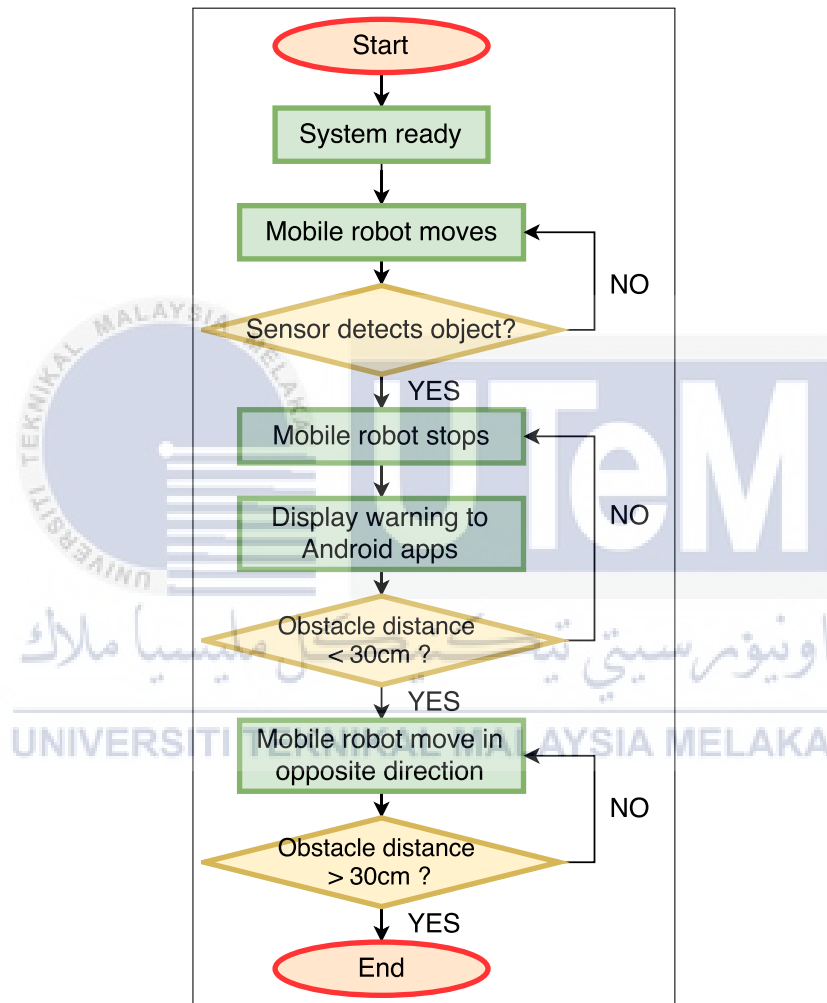


Figure 3.13: Flow chart of the obstacle detection system on the mobile robot.

### 3.7 Android Application Design

Mobile robot will be controlled by user through the communication between Arduino and Android application. An application for Android platform is required to be design in order to let user command the activity of mobile robot. An Android application will be developed using Android official IDE which is Android Studio, the developed application will be able to communicate, control and monitor the mobile robot through Arduino wirelessly with required internet connection.

The Android application will consists of connection switch (Connect/Disconnect), command button for movement (Forward, Backward, Left, Right, Rotate) and text bar for incoming subscribed reply from Arduino. Whole Android application is self-develop and customize to meet the objective of this project, the application should be able to install in Android platform devices.

### 3.8 Research Methodology

In this section, it is necessary to test or experiment on the mobile robot hardware and also software to evaluate its functionality as per objectives. There are a few experiments that were going to carry out in order to fulfil the objectives listed before in this thesis.

For the first and second objective, these objectives are indirectly related to each others, so there are 2 experiments will be conducted to evaluate hardware reliability on command received from Android application and the communication between both mobile robot and Android application using internet connection. Lastly, obstacle detection test will be carry out in line with the last objective of this project. This is to verify the obstacle detection system programmed in the mobile robot when encounter with nearby obstacle.

### 3.8.1 Transmission test between Arduino and Android application

The aim of this test is to validate the data transmission through IoT MQTT hosted broker from Android application to mobile robot (Arduino). This test will utilize a complete constructed mobile robot, a self-developed application and steady internet connection.

First of all, the connection between both device should be established and connected to carry on with the test. Arduino with Wifi capability using ESP8266-01 module will try to publish and subscribe to MQTT broker (CloudMQTT), websockets on the hosted site will be use to verify the transmission of data between them to make sure command send from broker is received by Arduino, the data will be print on Serial Monitor. Then, Android application will be developed and connect to the MQTT broker. The same method as the above will be use to testify publish and subscribe function of the application before further develop and design.

After both Android application and Arduino are successfully connected to each others, a communication test will be implement to get the transmission time for one command to travel from Arduino to Android application and vice versa. Experiment set-up for the test is pretty simple, that is using Android emulator and Arduino Serial Monitor with a set of coding is designed to trigger when a specific command is send from Android. When Arduino receive the command, it will sent an acknowledgement to Android and at the same time, it saves the time (`startMillis`) when reply is sent. Then, Android will send back a reply when acknowledgement is received, the received time (`millis`) is taken to get `durationMillis`. The transmission time for one way is obtained by divide the `durationMillis` by 2.

### 3.8.2 Hardware Reliability

This test is to determine the mobility and constant performance of mobile robot controlled by user. A good mobile robot will determine by consistency and reliability of its performance aside from selection of mobile robot hardware. For both hardware and software, it is important to validate the motion of mobile robot such as translational and rotational motion, obstacle detection systems, communication test between Android application and Arduino, and the consistency of mobile robot to perform same set of test. A table of successful attempt of test will be tabulate at the result and discussion chapter.

#### 3.8.2.1 Translational Motion Test

This test is to determine whether the mobile robot is capable of moving in every direction (front, left, right, back and rotate) and straight line. The set-up of this experiment environment is done at any empty space in home.

For straight line experiment, the mobile robot will be placed at a point that mark with white tape as starting line. Then, mobile robot will be command to move forward for 1 meter and 2 meters respectively, the angle of deviation from the starting point will be measure and record in table. In order to increase accuracy of angle between starting point and mobile robot, a bamboo skewer will be attached to the front edge of the mobile robot. The bamboo skewer will pointed down to the floor surface to indicate the exact point for starting and ending, angle will calculate and data is tabulate in a table based on the location of both points.

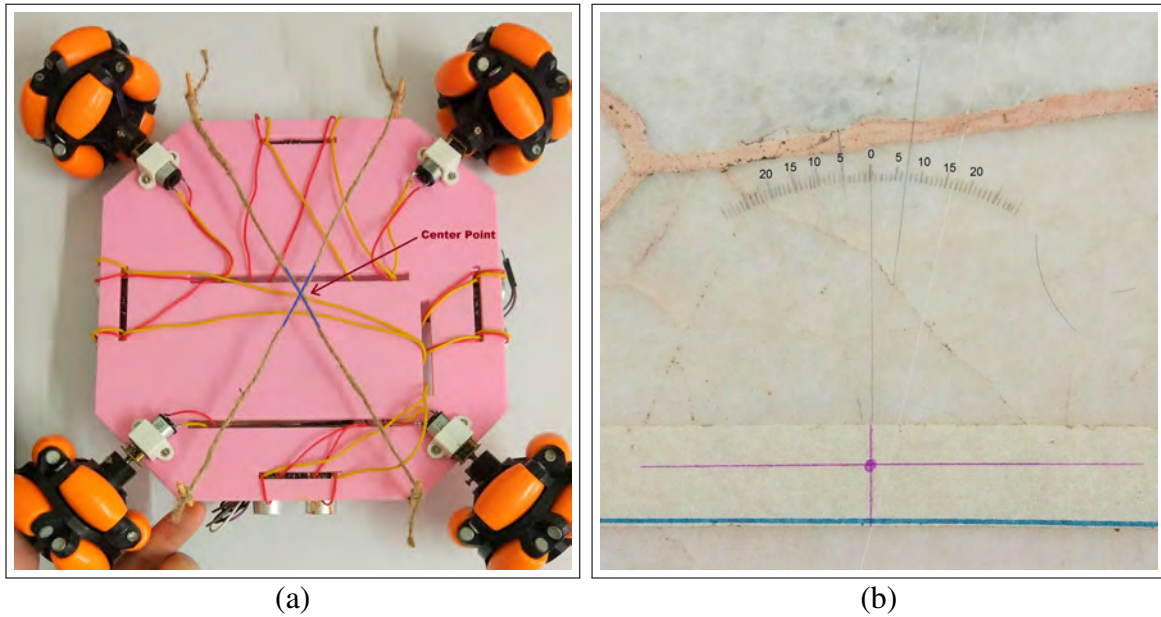


Figure 3.14: Setup of test with (a) center point of mobile robot with 4 bamboo skewer bottom view, and (b) angle marked on the floor.

Figure 3.14(a) shows 4 skewer are fixed at 4 points (top left, top right, bottom left and bottom right) of the mobile robot base, line is tied crossing each others oppositely to form a intersection to indicate as a center point. Figure 3.14(b) shows the angle marked on the floor using protractor tool and pencil to ease out when taking measurement reading on the floor. A metal pull ruler is used to connect 2 initial and final points in order to measure the angle of deviation.



### 3.8.2.2 Rotational Motion Test

With the same purpose as above translational motion test, this test will be use to determine the effective angle that mobile robot rotate with the given PWM and executing time. For rotational motion, mobile robot will rotate with the help of all 4 wheels, unlike the 2-wheel drive robot that use 2 wheels to rotate for an effective turning radius.

Figure 3.15 shows the mounting of bamboo skewer at the edge of mobile robot, 2 bamboo skewer will be mount on the front and back of the mobile robot pointing downwards. The robot will be placed on top of a clean paper board in order to mark each rotating point. First, 2 points (front and back) will be mark on the paper board, then after rotate at certain angle, another 2 points will be marked. Line is formed to connect each 2 points marked at the board respectively, a protractor is used to measure the angle between these 2 lines with refer to the intersection point (centre point). The lines will be form as accurate as possible to minimize the error during obtaining angle from the rotation of mobile robot. Different PWM is loaded to the mobile robot for a time limit and determine the PWM with the required execute time to make the mobile robot rotates at  $90^\circ$ . Each combination of PWM and execute time will be repeated 10 times to get average angle and standard deviation of those angle.

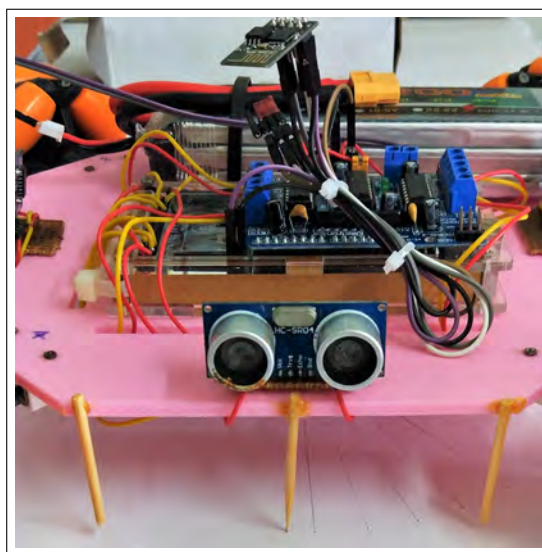


Figure 3.15: Bamboo skewer is attached to front edge of the mobile robot.



### 3.8.3 Obstacle Detection Test

This test will investigate the ability of mobile robot to detect or sense the obstacle ahead or around it and display warning to Android application when exceeds safety distance. The mobile robot is controlled by a user and move towards an object, then the sensors on mobile robot will be examine whether it can detect obstacles and stop effectively, and display warning to the Android application when exceeded a certain distance from obstacle.

Firstly, the mobile robot's obstacle detection system will be experiment based on 2 different range (60cm and 90cm) as Figure 3.16(a) from the obstacle. When test starts, a command will be send to the mobile robot to move towards the obstacle ahead. Bamboo skewer as in Figure 3.16(b) is mounted on the edge of the mobile robot pointing down. When system detects the obstacle in range, the mobile robot would stop immediately, an alert will be send to the Android application and at the same time it will reverse in opposite direction from the obstacle until it reaches a safe distance.

For data collection and analysis, the effective stopping distance of mobile robot from obstacle is marked and measured immediately once it stop. The detection range between the mobile robot and obstacle is set according to the effective stopping distance based on the experiments done above.

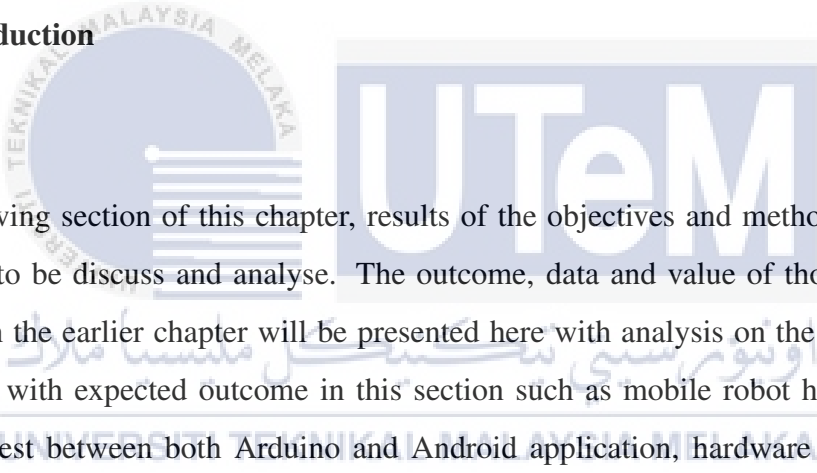


Figure 3.16: Mobile robot (a) is set-up according to 60cm and 90cm, and (b) bamboo skewer is point to the starting line.

## CHAPTER 4

### RESULT AND DISCUSSION

#### 4.1 Introduction



In this following section of this chapter, results of the objectives and methodology are reflected here to be discuss and analyse. The outcome, data and value of those experiment mentioned in the earlier chapter will be presented here with analysis on the obtained data. Experiments with expected outcome in this section such as mobile robot hardware, communication test between both Arduino and Android application, hardware reliability and repeatability experiments that include of mobile robot translational and rotational movement test, and lastly obstacle detection experiment of the mobile robot using sensors.

## 4.2 Hardware Design and Fabrication

Mobile robot is designed and fabrication by each of its part and component. There are 2 parts off the mobile robot needed to be fabricate using 3D printer, the robot base/chassis and the Omni wheel shaft. Both of the components are drawn and designed using Solidworks and before sending for 3D printing, STL format file are required for Cura which is a 3D printing slicing software to slice the model into thin parts and produce Gcode for the printer to read. The printing of robot base in which size of square 20cm x 20cm and the wheel's shaft takes around 14 hours and 2 hours to fabricate respectively.

The outcome of item fabrication are shown in respective figure below, colour of the item are chosen at random by the queue of 3D printer. Mobile robot base comes in pink colour in Figure 4.1 and the Omni wheel shaft in black colour in Figure 4.2.



Figure 4.1: Fabricated mobile robot base using 3D printer.

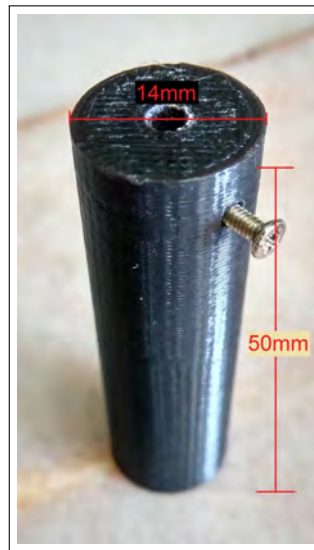


Figure 4.2: Fabricated Omni wheel's shaft using 3D printer.

Next, the wheel's shaft is attached to the Omni wheel and act as a shaft coupler to DC motor's shaft. The connection is made at the red circle in Figure 4.3(a) below. After coupling of Omni wheel with DC motor's shaft, the motor is mounted on the mobile robot base as shown in Figure 4.3(b) below using M1.6 bolt and nut on a motor mounting cap.



Figure 4.3: Shaft fabricated (a) combine with DC motor and Omni wheel, and (b) the combination attached to fabricated robot base.

#### 4.2.1 Mobile Robot Hardware with Obstacle Detection System

The mobile robot is fully fabricated and assembled by hand and also with the help of 3D printer provided by faculty to fulfil the objectives mentioned in this thesis.

Figure 4.4 shows the view of mobile robot from top, which consists of Arduino Mega2560 as the microcontroller, ESP8266-01 as the external Wifi module, PLA fabricated piece as the robot base, 4 ultrasonic sensors as the obstacle sensor, 4 omni wheels as the mobile robot primary moving wheels and Lithium polymer battery as the main power supply for mobile robot.

The obstacle detection system that requires 4 ultrasonic sensors are mounted on each side of the mobile robot base between wheels. The ultrasonic sensors are pointed perpendicular outwards from the centre of mobile robot and at  $90^\circ$  parallel to the horizontal floor level.

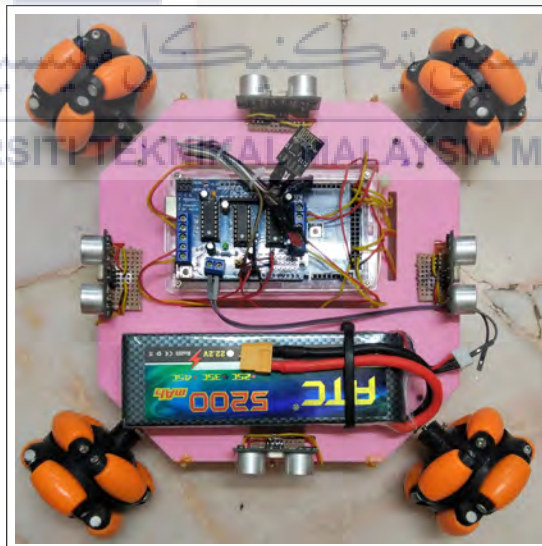


Figure 4.4: Fully assembled Omni wheel mobile robot.



### 4.3 Android application

As proposed in methodology, the Android platform application is designed and built using Android Studio that mainly used Java for application coding and XML for interface coding. The APK is built and transferred into Android smartphone to be install for controlling the mobile robot. This application is fully self-customized with the help of MQTT library obtained from CloudMqtt website's documentation. Figure 4.5 shows Android application installed in smartphone and is named as OmniRobot.



Figure 4.5: OmniRobot apps with customized logo for mobile robot control.

The connection of Android application to MQTT broker website is done by inserting the unique information such as API key, username, password and port number that is generated and assigned by the broker website as Figure 4.6. The unique API key, username and password for MQTT account is important as when a device requires to publish and subscribe to a channel when transmitting or receiving data. This is to make sure that every user of the broker website did not went overboard and interfere others user. The connection at the Arduino to MQTT broker website is similar to steps above.

CloudMQTT Console	
Server	m12.cloudmqtt.com
User	vyczugd
Password	4TDGeUj_rn8f
Port	11927
SSL Port	21927
Websockets Port (TLS only)	31927
Connection limit	10

Figure 4.6: Information that is generated and assigned by CloudMqtt borker website.

Next, Figure 4.7 shows the application interface when it is opened. Figure 4.7(a) shows the application is not connected to the mobile robot as the Connect toggle button on the top right corner is not switch to the right (greyed out). Figure 4.7(b) shows the application is connected with status obtained from the mobile robot as "Connected to OmniRobot!" and the Connect toggle button is switched to the right (green colour).



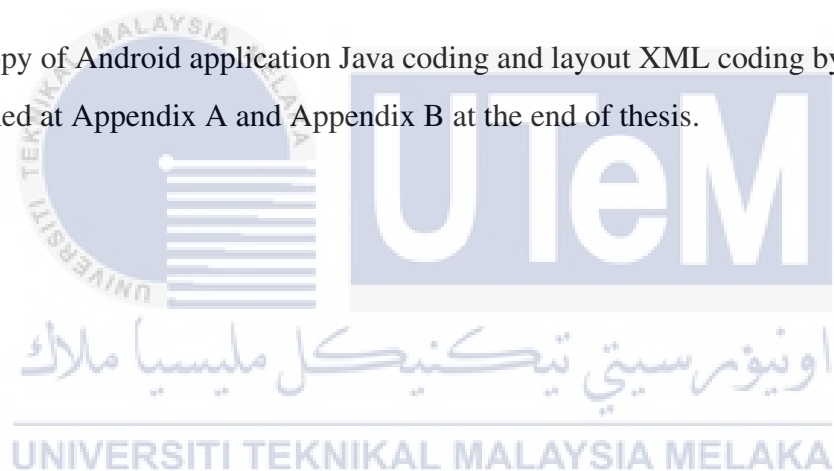
Figure 4.7: Application interface which are (a) disconnected from the mobile robot, and (b) connected to the mobile robot with update status obtained.



Steps to control the mobile robot using OmniRobot application:

- i Install the built APK file that transferred from Android Studio.
- ii Open the OmniRobot application in the phone menu.
- iii Connect the application to mobile robot by switching the toggle button at top right corner to the right.
- iv Wait for the application to connect and get confirmation from mobile robot.
- v After received "Connected to OmniRobot!" at the status box, you are ready to command the mobile robot.

A copy of Android application Java coding and layout XML coding by Android Studio is attached at Appendix A and Appendix B at the end of thesis.



#### 4.4 Data Transmission between Arduino and Android application

Based on Table 4.1, results of packet delivery time are taken for 10 times on each command sent to Arduino, standard deviation and mean time are shown for each respective command. Equation 4.1 is used to calculate the mean time while Equation 4.2 is used to calculate the standard deviation for each command's packet delivery time.

Table 4.1: Packet delivery time for each command to be send from Android application to Arduino or vice versa.

Command Test Count	frwd	back	left	right	stop	rttr	rttl
1	168.0	225.0	225.0	168.0	168.0	225.0	167.0
2	225.0	168.0	168.0	168.0	167.0	168.0	225.0
3	168.0	168.0	168.0	168.0	225.0	168.0	167.0
4	225.0	168.0	225.0	225.0	168.0	225.0	168.0
5	225.0	168.0	168.0	168.0	225.0	225.0	168.0
6	167.0	225.0	225.0	225.0	168.0	167.0	225.0
7	168.0	225.0	225.0	225.0	225.0	225.0	167.0
8	225.0	168.0	168.0	168.0	225.0	168.0	225.0
9	225.0	225.0	225.0	225.0	168.0	167.0	225.0
10	168.0	225.0	168.0	168.0	225.0	168.0	168.0
Std. dev, $\sigma$ (ms)	30.15	30.04	30.04	29.43	30.15	29.61	29.70
Total Std. dev (ms)	29.87						
Mean time, $\bar{x}$ (ms)	196.4	196.5	196.5	190.8	196.4	190.6	190.5
Total Mean time (ms)	194.0						

Notes: frwd = Forward, rttr = Rotate Right, rttl = Rotate Left

$$\bar{x} = \frac{\sum x}{n} \quad (4.1)$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (4.2)$$

where  $x$  = time in milliseconds,  $n$  = number of time and  $N$  = size of sample time obtained.

This test's data is obtained by using residential WiFi provided by local Internet Service Provider (TM UniFi) through optical fiber core network. Speedtest on the network is done before carry out test on data transmission performance, the test is carried out at late night due to less usage by users and surrounding residents. Figure 4.8 shows the download and upload speed is up to 50Mbps and 20Mbps respectively.

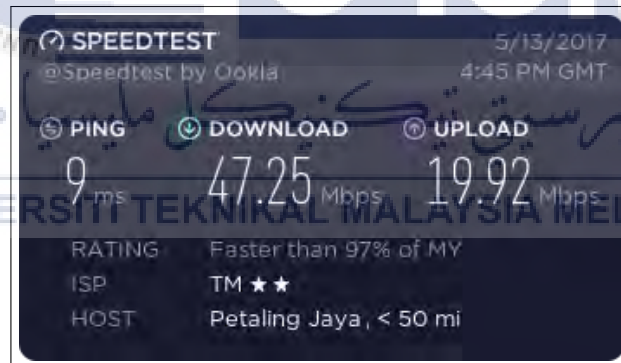


Figure 4.8: Home internet performance before test is carried out.

The total mean time for a command to send from Arduino to Android application (one-way) or vice versa is 194.0 ms and the standard deviation of transmission time is 29.87 ms with the percentage of is 15.4%. The results shows packet delivery time obtained for each test count on each command is not very consistent as it output on a few repeat value such as 167.0, 168.0 and 225.0 due to affected by various factor such as processing delay and buffer size.

#### 4.5 Hardware Reliability

Hardware reliability is done based on the experiment and test carried out by the mobile robot by receiving command from Android application. Those experiments includes translational and rotational motion, and communication test between application and mobile robot. The data extracted for each set of experiment are from a population of results obtained, 15 times of test are tabulated in Table 4.2.

Table 4.2: Hardware reliability based on the communication between the mobile robot and Android application test done.

Test Count	Command						
	frwd	back	left	right	stop	rttr	rttl
1	✓	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓	✓
9	✓	✓	✓	✓	✓	✓	✓
10	✓	✓	✓	✓	✓	✓	✓
11	✓	✓	✓	✓	✓	✓	✓
12	✓	✓	✓	✓	✓	✓	✓
13	✓	✓	✓	✓	✓	✓	✓
14	✓	✓	✓	✓	✓	✓	✓
15	-	✓	✓	✓	✓	-	✓
No. of ticks (/15)	14	15	15	15	15	14	15
Success rate (%)	93.3	100.0	100.0	100.0	100.0	93.3	100.0

Notes: frwd = Forward, rttr = Rotate Right, rttl = Rotate Left

$$\%success = \frac{Tick_{numbers}}{Test_{total}} \times 100 \quad (4.3)$$

The Table 4.2 shows results for 15 attempt of data transmission between both device. Out of 15 attempts, only forward (frwd) and rotate right (rttr) achieve 14 times success which is equivalent to 93.3%. For back, left, right, stop and rotate left (rttl), it achieve 100% success rate which is 15 out of 15 attempts. The success rate of communication test is calculated by using the Equation 4.3 as above, which  $Tick_{numbers}$  is number of ticks and  $Test_{total}$  is total number of attempt done.



#### 4.5.1 Translational Motion Test

Table 4.3 shows the results collected for translational motion test of the mobile robot, the deviation angle obtained are measured from initial position and final position of mobile robot at respective stopping range. In order to improve accuracy of angle taken, the initial and final position of mobile robot are determine by center point of the robot by using 4 bamboo skewer (2 at the front and 2 at the back).

Table 4.3: Comparison of mobile robot's center point deviation angle from initial starting point at two stopping distance.

Stopping Range (cm)		100.0				200.0			
Test Count	Command	frwd	back	left	right	frwd	back	left	right
	1		16.0	14.0	8.0	2.0	5.0	8.0	8.0
2		8.0	14.0	1.0	1.0	5.0	4.0	9.0	4.0
3		10.0	7.0	5.0	1.0	6.0	2.0	9.0	7.0
4		16.0	12.0	3.0	1.0	2.0	7.0	5.0	4.0
5		12.0	8.0	7.0	3.0	3.0	1.0	3.0	2.0
6		7.0	14.0	7.0	1.0	4.0	5.0	3.0	3.0
7		10.0	10.0	5.0	3.0	0.0	5.0	6.0	1.0
8		13.0	12.0	10.0	3.0	1.0	5.0	4.0	4.0
9		11.0	9.0	6.0	4.0	6.0	3.0	7.0	2.0
10		9.0	11.0	7.0	2.0	3.0	5.0	4.0	5.0
Std. dev, $\sigma$ (°)		3.08	2.56	2.56	1.10	2.07	2.12	2.35	2.21
Mean angle, $\bar{x}$ (°)		11.2	11.1	5.9	2.1	3.5	4.5	5.8	4.0

Notes: frwd = Forward

According to the data measured above, the angle for stopping range 100 cm and 200 cm is different to each others. For forward motion, the angle obtained at stopping distance of 100 cm is  $11.2^\circ$  and 200 cm is  $3.5^\circ$ . The is also similar to backward motion, the angle obtained at stopping distance of 100 cm is  $11.1^\circ$  and 200 cm is  $4.5^\circ$ . The outcome of deviation angle shows that the mobile robot moving with slight curve from the perpendicular straight line.

Next, for left motion, the results obtained for 100 cm stopping range is  $5.9^\circ$  and 200 cm is  $5.8^\circ$  while for right motion, the results obtained are  $2.1^\circ$  and  $4.0^\circ$  respectively. Results for both translational motion shows mobile robot move at near to straight line.

Figure 4.9 shows the angle is obtained from the final center point of mobile robot by using metal pull ruler (does not appear in figure). The angle of deviation from perpendicular straight line does not consider positive and negative sign as the aim of this test is to determine the translational motion performance of mobile robot.



Figure 4.9: Angle obtained from the 2 center point of mobile robot.

#### 4.5.2 Rotational Motion Test

Rotational motion test is done with three combinations of PWM and run time (ms), which are PWM 120 with 100ms, PWM 180 with 100ms and PWM 180 with 550ms. Table 4.4 shows angle measured and collected with respect to those combination, data is taken 10 times for each command under every combination and mean angle with standard deviation is calculated.

Table 4.4: Comparison of Rotational motion test with 3 combinations of Pulse Width Modulation (PWM) and Run Time for left and right rotation.

PWM (/255)	120		180		180	
Run Time (ms)	100		100		550	
Test Count	R. Left	R. Right	R. Left	R. Right	R. Left	R. Right
1	8.0	9.0	12.0	13.0	91.5	89.0
2	8.0	7.0	13.0	14.0	89.0	90.0
3	8.0	8.0	15.0	13.0	90.0	88.5
4	9.0	8.5	13.0	12.0	88.5	90.5
5	8.5	9.0	14.5	16.0	92.0	89.0
6	7.0	7.0	14.5	14.5	90.5	90.0
7	9.0	7.0	12.0	13.0	89.5	92.0
8	9.0	8.0	15.0	14.0	88.0	88.0
9	8.5	7.5	13.0	12.0	91.0	90.5
10	9.0	7.0	13.5	15.0	90.5	88.0
Std. dev, $\sigma$ (°)	0.658	0.823	1.141	1.292	1.301	1.279
Mean angle, (°)	8.4	7.8	13.6	13.7	90.1	89.6



$$\bar{x} = \frac{\sum x}{n} \quad (4.4)$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (4.5)$$

$$\%Error = \left| \frac{Actual - Experiment}{Actual} \right| \times 100 \quad (4.6)$$

where  $x$  = time in milliseconds,  $n$  = number of time and  $N$  = size of sample time obtained.

Based on the Table 4.4, combination 1 which PWM is 120 (equivalent to 47% duty cycle) and 100ms run time, gives an average rotate angle of  $8.4^\circ$  for left rotation and  $7.8^\circ$  for right rotation. From the sample taken from rotational motion test, left rotation gives a standard deviation of  $0.658^\circ$  while right rotation results in  $0.823^\circ$ .

For 2nd combination, PWM used is 180 (equivalent to 71% duty cycle) and 100ms run time, result in  $13.6^\circ$  for left rotation and  $13.7^\circ$  for right rotation with a standard deviation of  $1.141^\circ$  and  $1.292^\circ$  respectively. Supposingly, setting PWM 180 for mobile robot is to turn  $15^\circ$ , however the result obtained results in minor error when compared to the theoretical angle. By using Equation 4.6, the error resulted from calculation are 9.33% for left rotation and 8.66% for right rotation.

For 3rd combination, PWM used is 180 (equivalent to 71% duty cycle) and 550ms run time, result obtained from experiment is  $90.1^\circ$  for left rotation and  $89.6^\circ$  for right rotation with a standard deviation of  $1.301^\circ$  and  $1.279^\circ$  respectively. The expected result of this combination test is  $90^\circ$ , the results shown with an error of 0.11% for left rotation and 0.44% for right rotation which is almost precise for the mobile robot. Summing up of results obtained, the error existed in the data above may due to various factors acting on the mobile robot, such as centre of mass, power supply, speed of each geared dc motor and slippage of wheel with floor surface.

#### 4.6 Obstacle Detection of mobile robot

Table 4.5 shows comparison of stopping distance of mobile robot with starting range of 60 cm and 90 cm, total of 10 sets test are done for each directional command with refer to the starting range. The stopping distance programmed at Arduino is 30 cm between the ultrasonic sensor and obstacle.

Table 4.5: Comparison of stopping distance of mobile robot when encountered obstacle ahead.

Starting Range (cm)	60.0				90.0			
Command	frwd	back	left	right	frwd	back	left	right
Test Count								
1	11.80	18.70	11.30	10.70	12.10	7.80	9.30	8.50
2	14.00	8.90	9.70	10.80	11.50	11.60	8.70	10.30
3	14.80	13.40	8.90	8.60	9.20	6.30	11.40	11.60
4	12.90	12.30	13.30	9.10	12.70	10.20	10.90	12.30
5	13.00	6.60	15.70	8.90	8.30	16.80	11.60	9.60
6	12.80	10.90	13.10	13.10	12.20	9.80	9.70	10.40
7	8.40	10.30	10.70	12.50	7.20	13.10	14.20	12.80
8	10.20	9.50	8.20	11.90	8.10	11.30	12.50	9.40
9	9.90	11.80	7.90	8.60	13.50	8.90	17.10	17.60
10	8.70	9.30	12.50	11.10	14.90	15.70	7.50	7.50
Std. dev, $\sigma$ (cm)	2.226	3.274	2.530	1.665	2.595	3.320	2.819	2.843
Total Std. dev	2.659 cm							
Mean distance, $\bar{x}$ (cm)	11.65	11.17	11.13	10.53	10.97	11.15	11.29	11.00
Total Mean distance	11.07 cm							

Notes: frwd = Forward

$$\%Error = \left| \frac{Preset - Result}{Preset} \right| \times 100 \quad (4.7)$$

For 60 cm starting range of mobile robot, the mean distance collected from the experiment are 11.65 cm for forward, 11.17 cm for backward, 11.13 cm for left and 10.53 cm for right. The standard deviation for each command at starting range of 60cm are as follows, 2.226 cm for forward, 3.274 cm for backward, 2.530 for left and 1.665 cm for right.

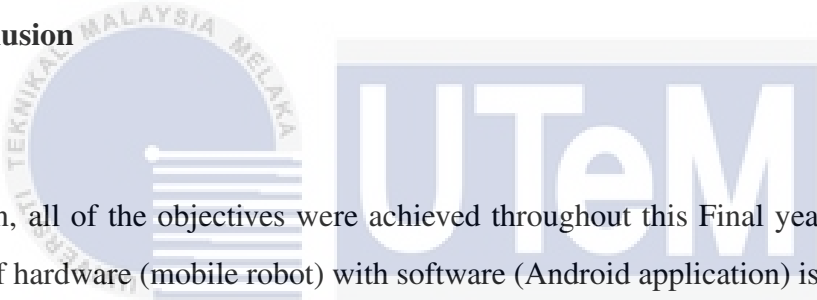
For starting distance of 90 cm, the mean distance collected from the experiment are 10.97 cm for forward, 11.15 cm for backward, 11.29 cm for left and 11 cm for right. The standard deviation for each command at starting range of 90cm are as follows, 2.595 cm for forward, 3.320 cm for backward, 2.819 for left and 2.843 cm for right.

These data give a total mean stopping distance of 11.07 cm for mobile robot when ultrasonic sensor detects an obstacle ahead while the total standard deviation of distance obtained is 2.659 cm. These results are totally different from distance set in Arduino program, the error from data obtained is 63.1% by using Equation 4.7 above. The error of data obtained from mobile robot with obstacle detection system when compared to the expected stopping distance is due to processing speed of Arduino due to length of system code in loop, the Arduino is programmed to scan all ultrasonic sensor in loop which outcome with slow responding time for mobile robot to stop when encountered an obstacle.

## CHAPTER 5

### CONCLUSION AND RECOMMENDATION

#### 5.1 Conclusion



In conclusion, all of the objectives were achieved throughout this Final year Project. The integration of hardware (mobile robot) with software (Android application) is complete. For the first objective, there are two experiment done to prove the working of the mobile robot which are translational and rotational motion test. With the results obtained from test conducted, the mobile robot is not able to move at a perfectly straight line with zero error, the deviation of the mobile robot from straight line is due to multiple factors that affecting it. These factor include centre of mass of mobile robot, reliability and synchronous speed of all geared DC motor, slippage of Omni wheels due to material used and maximum contact area of Omni wheels. For rotational motion test, although the outcome is promising but the results do varies according on Pulse Width Modulation (PWM) speed, power supplied to mobile robot, and slippage of Omni wheels.

For second objectives, an Android application is successfully developed using Android Studio with the main objectives to command mobile robot for its movement and receive reply from mobile robot through online hosted server. The transmission and reliability test is done to prove the feasibility of application towards mobile robot. However, success rate in reliability test is due to the buffer and flooding of ESP8266 when receiving data and there are small amount of error in applications when certain condition of buttons are pressed, it is due to error handling of application.

Lastly, obstacle detection system is successfully implemented in the mobile robot to detect any obstacle ahead using ultrasonic sensors installed at four edge on the robot base. The detection range error existed in the results is due to the mobile robot response rate which includes processing speed and multitasking of Arduino board in collaborate with the mobile robot.



## 5.2 Recommendation

With refer to the source of error surfaced in the research results in each experiments above, recommendation is important to be suggested for future research work in this area to be more comprehensive and error-free. For experiments to complete first objective, the designed of the mobile robot should be balanced in term of mass distribution, the allocation of Arduino board and battery which gives the most in weight that affects the mobile robot. A better geared DC motor is suggested to ensure the stable performance and quality throughout the project. Omni wheel of the mobile robot shall be a rubber finished wheel and metal rim for improve rigidity, as for shaft to hold the wheel, metal shaft is more recommended.

For second experiment, alternative of ESP8266 for Arduino can be use to provide more stable transmission of results, the transmission between both device can be improve by using short strings or command in coding. For third experiment, others microcontroller can be used to substitute Arduino for better processing speed and RAM. This can avoid delay in response of mobile robot when making decision based on sensors reading value. It is more recommend to use microcontroller with WiFi features for more reliable and cut off all the hassle when connecting both devices for data transmission.

## REFERENCES

- [1] Google Inc., “Platform Architecture | Android Developers.” [Online]. Available: <https://developer.android.com/guide/platform/index.html>. [Accessed 2016-11-28].
- [2] Purple, “How WiFi has changed the world,” 2014. [Online]. Available: <http://purple.ai/wifi-changed-world/>. [Accessed 2016-11-22].
- [3] B. Daniel, “The Internet of Things Is Far Bigger Than Anyone Realizes | WIRED,” 2014. [Online]. Available: <https://www.wired.com/insights/2014/11/the-internet-of-things-bigger/>. [Accessed 2016-10-31].
- [4] P. David, “The Internet of Things Is Everywhere, But It Doesn’t Rule Yet | WIRED,” 2015. [Online]. Available: <https://www.wired.com/2015/12/this-year-was-almost-the-year-of-the-internet-of-things/>. [Accessed 2016-10-22].
- [5] Android, “The Android Source Code | Android Open Source Project.” [Online]. Available: <https://source.android.com/source/index.html>. [Accessed 2016-10-31].
- [6] Oracle, “Lesson: All About Sockets (The Java Tutorials),” 2015. [Online]. Available: <http://docs.oracle.com/javase/tutorial/networking/sockets/>. [Accessed 2016-10-27].
- [7] C. Kulkarni, S. Grama, P. G. Suresh, C. Krishna, and J. Antony, “Surveillance Robot Using Arduino Microcontroller, Android APIs and the Internet,” in *SIMS '14 Proceedings of the 2014 First International Conference on Systems Informatics, Modelling and Simulation*, 2014, pp. 83–87.
- [8] X. Lu, W. Liu, H. Wang, and Q. Sun, “Robot control design based on smartphone,” *2013 25th Chinese Control and Decision Conference (CCDC)*, pp. 2820–2823, 2013.
- [9] C. Jim, “The Evolution of the Internet of Things,” Texas Instrument, Tech. Rep., 2013. [Online]. Available: <http://www.ti.com/lit/ml/swrb028/swrb028.pdf>
- [10] V. Wekhande, “Wi-Fi Technology : Security Issues,” *RIVIER ACADEMIC JOURNAL*, vol. 2, no. 2, pp. 1–17, 2006.
- [11] IEEE, *IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2012th ed. New York, USA: IEEE, 2012, no. March.
- [12] Link Labs, “5 Types of Wireless Technology For The IoT - Link Labs,” 2015. [Online]. Available: <https://www.link-labs.com/types-of-wireless-technology/>. [Accessed 2016-10-28].

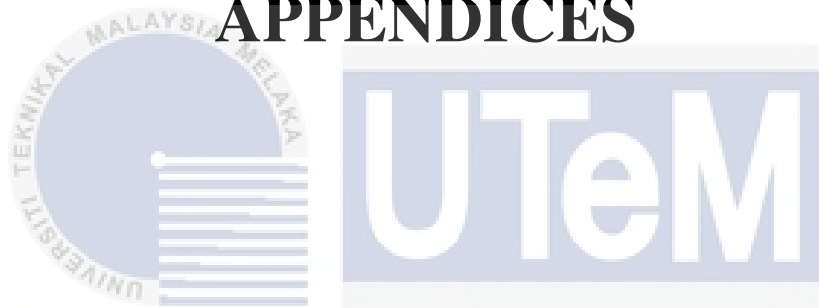
- [13] S. Göbel and R. Jubeh, "Using the Android Platform to control Robots," *Proceedings of 2nd International Conference on Robotics in Education (RiE 2011)*, pp. 135–142, 2011.
- [14] E. Ferro and F. Potortì, "Bluetooth and Wi-Fi wireless protocols: A survey and a comparison," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12–26, 2005.
- [15] S. Banerji and R. S. Chowdhury, "Wi-Fi & WiMAX : A Comparative Study," *Indian Journal of Engineering*, vol. 2, no. 5, pp. 1–5, 2013.
- [16] M. Lutovac Banduka, "Remote Monitoring and Control of Industrial Robot based on Android Device and Wi-Fi Communication," *Automatika – Journal for Control, Measurement, Electronics, Computing and Communications*, vol. 56, no. 3, pp. 281–291, 2015.
- [17] Arduino.cc, "Arduino - ArduinoBoardMega2560," 2016. [Online]. Available: <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>. [Accessed 2016-10-31].
- [18] RPi3B, "Raspberry Pi 2 Model B," 2015. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>. [Accessed 2016-10-31].
- [19] R. Krauss, "Combining Raspberry Pi and Arduino to Form a Low-Cost , Real-Time Autonomous Vehicle Platform," *American Control Conference (ACC)*, pp. 6628–6633, 2016.
- [20] BeagleBoard.org, "BeagleBoard.org - black," 2016. [Online]. Available: <http://beagleboard.org/black>. [Accessed 2016-11-02].
- [21] A. Nayyar and V. Puri, "A review of Beaglebone smart board's-a Linux/android powered low cost development platform based on ARM technology," in *Proceedings - 9th International Conference on Future Generation Communication and Networking, FGCN 2015*. IEEE, nov 2016, pp. 55–63.
- [22] SECO USA Inc, "UDOO x86 Basic," 2016. [Online]. Available: <https://shop.udoo.org/x86/udoo-x86-basic.html>. [Accessed 2017-06-09].
- [23] Google Inc, "Android, the world's most popular mobile platform | Android Developers." [Online]. Available: <https://developer.android.com/about/index.html>. [Accessed 2016-11-28].
- [24] P. Kaur and S. Sharma, "Google Android a mobile platform: A review," *Recent Advances in Engineering and Computational Sciences (RAECS)*, 2014, vol. 10, no. 5, pp. 1–5, mar 2014.
- [25] Kirandeep and A. Garg, "Implementing Security on Android Application," *The International Journal Of Engineering And Science (IJES)*, vol. 2, no. 3, pp. 56–59, 2013.
- [26] M. H. Amaran, N. A. M. Noh, M. S. Rohmad, and H. Hashim, "A Comparison of Lightweight Communication Protocols in Robotic Applications," *Procedia Computer Science*, vol. 76, pp. 400–405, 2015.
- [27] International Business Machines Corporation (IBM) and Eurotech, "MQTT V3.1 Protocol Specification," 2010. [Online]. Available: <http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html>. [Accessed 2016-12-01].



- [28] Z. Shelby, ARM, K. Hartke, and C. Bormann, “RFC 7252 - The Constrained Application Protocol (CoAP),” Universitaet Bremen TZI, Bremen, Germany, Tech. Rep., 2014.
- [29] S. Bandyopadhyay and A. Bhattacharyya, “Lightweight Internet protocols for web enablement of sensors using constrained gateway devices,” in *2013 International Conference on Computing, Networking and Communications, ICNC 2013*, 2013, pp. 334–340.



# APPENDICES



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## APPENDIX A

## ANDROID APPLICATION JAVA CODING

```
package lsq.fyp.mqtttest;

import android.graphics.Color;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Vibrator;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
```

```

import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public class MainActivity extends AppCompatActivity implements View
    ↪ .OnTouchListener {
    static String MQTTHOST = "tcp://m12.cloudmqtt.com:11927";
    static String Username = "yyczugjd";
    static String Password = "4TDGeUt_nn8f";
    String pubtopic = "Command";
    String subtopic = "Arduino";

    MqttAndroidClient client;
    TextView subtext;
    MqttConnectOptions options;
    Vibrator vibrates;
    Ringtone myringtone;

    ImageButton Stopbtn;
    ImageButton Fwdbtn;
    ImageButton Leftbtn;
    ImageButton Rightbtn;
    ImageButton Bwdbtn;
    ImageButton RttLBtn;
    ImageButton RttRBtn;

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);

        MenuItem onoffSwitch = menu.findItem(R.id.myswitch);

```

```

onoffSwitch.setActionView(R.layout.switch_layout);

final Switch ConnectSwitch = (Switch) menu.findViewById(R.id.
    ↪ myswitch).getActionView().findViewById(R.id.switchCD)
    ↪ ;
ConnectSwitch.setOnCheckedChangeListener(new CompoundButton
    ↪ .OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView,
    ↪ boolean isChecked) {
        if (isChecked){
            try {
                IMqttToken token = client.connect(options);
                token.setActionCallback(new
    ↪ IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken
    ↪ asyncActionToken) {
                    Toast.makeText(MainActivity.this, "
    ↪ Connected!", Toast.LENGTH_LONG
                    ↪ ).show();
                    subscription();
                    String topic = pubtopic;
                    String message = "Hi_Omni~";
                    try {
                        client.publish(topic, message.
    ↪ getBytes(), 0, false);
                    } catch (MqttException e) {
                        e.printStackTrace();
                    }
                }}
            @Override
            public void onFailure(IMqttToken
    ↪ asyncActionToken, Throwable
    ↪ exception) {

```

```

        Toast.makeText(MainActivity.this, "
            ↳ Connection_Failed!", Toast.
            ↳ LENGTH_LONG).show();
    });
} catch (MqttException e) {
    e.printStackTrace();
}
} else {
    try {
        boolean connected = client.isConnected();
        if (connected) {
            Toast.makeText(MainActivity.this, "Now_
                ↳ disconnecting!", Toast.
                ↳ LENGTH_LONG).show();
            IMqttToken token = client.disconnect();
            token.setActionCallback(new
                ↳ IMqttActionListener() {
                    @Override
                    public void onSuccess(IMqttToken
                        ↳ asyncActionToken) {
                        Toast.makeText(MainActivity.
                            ↳ this, "Disconnected!",
                            ↳ Toast.LENGTH_LONG).show()
                            ↳ ;
                    }
                @Override
                public void onFailure(IMqttToken
                    ↳ asyncActionToken, Throwable
                    ↳ exception) {
                        Toast.makeText(MainActivity.
                            ↳ this, "Fail_to_disconnect
                            ↳ !", Toast.LENGTH_LONG).
                            ↳ show();
                    }
            }
    }
}

```

```

        }); }
    } catch (MqttException e) {
        e.printStackTrace();
    }}}});

    return true;
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    subtext = (TextView) findViewById(R.id.subtext);
    vibrates = (Vibrator) getSystemService(VIBRATOR_SERVICE);
    Uri uri = RingtoneManager.getDefaultUri(RingtoneManager.
        ↪ TYPE_NOTIFICATION);
    myringtone = RingtoneManager.getRingtone(
        ↪ getApplicationContext(), uri);

    String clientId = MqttClient.generateClientId();
    client = new MqttAndroidClient(this.getApplicationContext()
        ↪ , MQTTHOST, clientId);
    options = new MqttConnectOptions();
    options.setUsername(Username);
    options.setPassword>Password.toCharArray());

    Stopbtn = (ImageButton) findViewById(R.id.Stopbtn);
    Stopbtn.setOnClickListener(this);
    Fwdbtn = (ImageButton) findViewById(R.id.Fwdbtn);
    Fwdbtn.setOnClickListener(this);
    Leftbtn = (ImageButton) findViewById(R.id.Leftbtn);
    Leftbtn.setOnClickListener(this);
    Rightbtn = (ImageButton) findViewById(R.id.Rightbtn);
    Rightbtn.setOnClickListener(this);

```

```

Bwdbtn = (ImageButton) findViewById(R.id.Bwdbtn);
Bwdbtn.setOnClickListener(this);
RttLBtn = (ImageButton) findViewById(R.id.RttLBtn);
RttLBtn.setOnClickListener(this);
RttRBtn = (ImageButton) findViewById(R.id.RttRBtn);
RttRBtn.setOnClickListener(this);

client.setCallback(new MqttCallback() {
    @Override
    public void connectionLost(Throwable cause) {
    }
    @Override
    public void messageArrived(String topic, MqttMessage
    ↪ message) throws Exception {
        subtext.setText(new String (message.getPayload()));
        vibrates.vibrate(500);
        myringtone.play();
    }
    @Override
    public void deliveryComplete(IMqttDeliveryToken token)
    ↪ {
    }
});

}

@Override
public boolean onTouch(View v, MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        switch (v.getId()) {
            case R.id.Stopbtn:
                if (client.isConnected()) {
                    Stopbtn.setBackgroundResource(R.drawable.
                    ↪ button_shape_blue1);
                    String topic = pubtopic;

```



```

String message = "stop";
try {
    client.publish(topic, message.getBytes
        ↪ (), 0, false);
} catch (MqttException e) {
    e.printStackTrace();
}
} else if (!client.isConnected()){
    Toast.makeText(MainActivity.this, "Please_
        ↪ CONNECT_to_MQTT!", Toast.LENGTH_LONG)
        ↪ .show();
}
break;
case R.id.Fwdbtn:
    if (client.isConnected()) {
        Fwdbtn.setBackgroundResource(R.drawable.
            ↪ button_shape_orange1);
        String topic = pubtopic;
        String message = "frwd";
        try {
            client.publish(topic, message.getBytes
                ↪ (), 0, false);
        } catch (MqttException e) {
            e.printStackTrace();
        }
    } else {
        Toast.makeText(MainActivity.this, "Please_
            ↪ CONNECT_to_MQTT!", Toast.LENGTH_LONG)
            ↪ .show();
    }
    break;
case R.id.Leftbtn:
    if (client.isConnected()) {

```

```

Leftbtn.setBackgroundResource(R.drawable.
    ↪ button_shape_orange1);
String topic = pubtopic;
String message = "left";
try {
    client.publish(topic, message.getBytes
        ↪ (), 0, false);
} catch (MqttException e) {
    e.printStackTrace();
}
} else {
    Toast.makeText(MainActivity.this, "Please_
        ↪ CONNECT_to_MQTT!", Toast.LENGTH_LONG)
        ↪ .show();
    }
    break;
case R.id.Rightbtn:
    if (client.isConnected()) {
        Rightbtn.setBackgroundResource(R.drawable.
            ↪ button_shape_orange1);
        String topic = pubtopic;
        String message = "right";
        try {
            client.publish(topic, message.getBytes
                ↪ (), 0, false);
        } catch (MqttException e) {
            e.printStackTrace();
        }
    } else {
        Toast.makeText(MainActivity.this, "Please_
            ↪ CONNECT_to_MQTT!", Toast.LENGTH_LONG)
            ↪ .show();
    }
}
break;

```

```

case R.id.Bwdbtn:
    if (client.isConnected()) {
        Bwdbtn.setBackgroundResource(R.drawable.
            ↪ button_shape_orange1);
        String topic = pubtopic;
        String message = "back";
        try {
            client.publish(topic, message.getBytes
                ↪ (), 0, false);
        } catch (MqttException e) {
            e.printStackTrace();
        }
    } else {
        Toast.makeText(MainActivity.this, "Please_
            ↪ CONNECT_to_MQTT!", Toast.LENGTH_LONG)
            ↪ .show();
    }
    break;
case R.id.RttlBtn:
    if (client.isConnected()) {
        RttlBtn.setBackgroundResource(R.drawable.
            ↪ button_shape_orange1);
        String topic = pubtopic;
        String message = "rttl";
        try {
            client.publish(topic, message.getBytes
                ↪ (), 0, false);
        } catch (MqttException e) {
            e.printStackTrace();
        }
    } else {
        Toast.makeText(MainActivity.this, "Please_
            ↪ CONNECT_to_MQTT!", Toast.LENGTH_LONG)
            ↪ .show();
    }

```

```

    }
    break;
case R.id.RttRBtn:
    if (client.isConnected()) {
        RttRBtn.setBackgroundResource(R.drawable.
            ↪ button_shape_orange1);
        String topic = pubtopic;
        String message = "rttr";
        try {
            client.publish(topic, message.getBytes
                ↪ (), 0, false);
        } catch (MqttException e) {
            e.printStackTrace();
        }
    } else {
        Toast.makeText(MainActivity.this, "Please_
            ↪ CONNECT_to_MQTT!", Toast.LENGTH_LONG)
            ↪ .show();
    }
    break;
}
} else if (event.getAction() == MotionEvent.ACTION_UP){
    switch (v.getId()){
        case R.id.Stopbtn:
            if (client.isConnected()){
                Stopbtn.setBackgroundResource(R.drawable.
                    ↪ button_shape_blue);
            } else break;
        case R.id.Fwdbtn:
            if (client.isConnected()){
                Fwdbtn.setBackgroundResource(R.drawable.
                    ↪ button_shape_orange);
            } else break;
        case R.id.Leftbtn:

```

```

        if (client.isConnected()){
            Leftbtn.setBackgroundResource(R.drawable.
                ↪ button_shape_orange);
        } else break;
    case R.id.Rightbtn:
        if (client.isConnected()){
            Rightbtn.setBackgroundResource(R.drawable.
                ↪ button_shape_orange);
        } else break;
    case R.id.Bwdbtn:
        if (client.isConnected()){
            Bwdbtn.setBackgroundResource(R.drawable.
                ↪ button_shape_orange);
        } else break;
    case R.id.RttLBtn:
        if (client.isConnected()){
            RttLBtn.setBackgroundResource(R.drawable.
                ↪ button_shape_orange);
        } else break;
    case R.id.RttRBtn:
        if (client.isConnected()){
            RttRBtn.setBackgroundResource(R.drawable.
                ↪ button_shape_orange);
        } else break;
    }}
    return true;
}

private void subscription(){
    try{
        client.subscribe(subtopic,0);
    } catch (MqttException e){
        e.printStackTrace();
    }
}
}
}

```

## APPENDIX B

### ANDROID APPLICATION LAYOUT CODING

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
↳ android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="lsq.fyp.mqtttest.MainActivity">

    <TextView
        android:id="@+id/subtext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="16dp"
        android:layout_marginStart="16dp"
        android:layout_toEndOf="@+id/Status">

```

```

android:layout_toRightOf="@+id/Status"
android:gravity="center"
android:text="Waiting..."
android:textAlignment="textStart"
/>

```

```
<ImageButton
```

```

    android:id="@+id/Stopbtn"
    android:layout_width="@android:dimen/
        ↳ notification_large_icon_width"
    android:layout_height="64dp"
    android:layout_alignLeft="@+id/Bwdbtn"
    android:layout_alignStart="@+id/Bwdbtn"
    android:layout_centerVertical="true"
    android:background="@drawable/button_shape_blue"
    android:elevation="4dp"
    android:hapticFeedbackEnabled="false"
    app:srcCompat="@drawable/ic_fullscreen_exit_black_48dp"
/>

```

```
<ImageButton
```

```

    android:id="@+id/Fwdbtn"
    android:layout_width="@android:dimen/
        ↳ notification_large_icon_width"
    android:layout_height="@android:dimen/
        ↳ notification_large_icon_height"
    android:layout_marginBottom="28dp"
    android:background="@drawable/button_shape_orange"
    android:elevation="4dp"
    app:srcCompat="@drawable/ic_expand_less_black_48dp"
    android:layout_above="@+id/Stopbtn"
    android:layout_alignLeft="@+id/Stopbtn"
    android:layout_alignStart="@+id/Stopbtn"
/>

```

```

<ImageButton
    android:id="@+id/Leftbtn"
    android:layout_width="@android:dimen/
        ↳ notification_large_icon_width"
    android:layout_height="@android:dimen/
        ↳ notification_large_icon_height"
    android:background="@drawable/button_shape_orange"
    android:elevation="4dp"
    app:srcCompat="@drawable/ic_chevron_left_black_48dp"
    android:layout_below="@+id/Fwdbtn"
    android:layout_alignLeft="@+id/RttLBtn"
    android:layout_alignStart="@+id/RttLBtn"
/>
<ImageButton
    android:id="@+id/Rightbtn"
    android:layout_width="64dp"
    android:layout_height="@android:dimen/
        ↳ notification_large_icon_height"
    android:background="@drawable/button_shape_orange"
    android:elevation="4dp"
    app:srcCompat="@drawable/ic_chevron_right_black_48dp"
    android:layout_below="@+id/Fwdbtn"
    android:layout_alignLeft="@+id/RttRBtn"
    android:layout_alignStart="@+id/RttRBtn"
/>
<ImageButton
    android:id="@+id/Bwdbtn"
    android:layout_width="@android:dimen/
        ↳ notification_large_icon_width"
    android:layout_height="@android:dimen/
        ↳ notification_large_icon_height"

```



```

android:layout_marginTop="28dp"
android:background="@drawable/button_shape_orange"
android:elevation="4dp"
app:srcCompat="@drawable/ic_expand_more_black_48dp"
android:layout_below="@+id/Stopbtn"
android:layout_centerHorizontal="true"
/>

```

<ImageButton

```

android:id="@+id/RttLBtn"
android:layout_width="@android:dimen/
    ↳ notification_large_icon_width"
android:layout_height="@android:dimen/
    ↳ notification_large_icon_height"
android:layout_above="@+id/Fwdbtn"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_marginBottom="10dp"
android:layout_marginLeft="28dp"
android:layout_marginStart="28dp"
android:background="@drawable/button_shape_orange"
android:cropToPadding="false"
android:elevation="4dp"
app:srcCompat="@drawable/ic_rotate_left_black_48dp"
/>

```

<ImageButton

```

android:id="@+id/RttRBtn"
android:layout_width="@android:dimen/
    ↳ notification_large_icon_width"
android:layout_height="@android:dimen/
    ↳ notification_large_icon_height"
android:background="@drawable/button_shape_orange"
android:cropToPadding="false"

```

```

android:elevation="4dp"
app:srcCompat="@drawable/ic_rotate_right_black_48dp"
android:layout_alignTop="@+id/RttLBtn"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true"
android:layout_marginRight="28dp"
android:layout_marginEnd="28dp"
/>

```

```
<TextView
```

```

android:id="@+id/Status"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Status:"
android:layout_alignParentTop="true"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
/>

```

```
<TextView
```

```

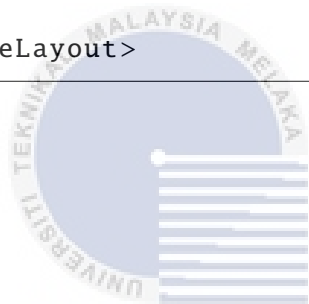
android:id="@+id/FYP"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_marginBottom="16dp"
android:fontFamily="sans-serif-smallcaps"
android:text="Final_Year_Project_2017"
android:textAllCaps="false"
android:textColor="@color/colorPrimaryDark"
/>

```

```
<TextView
```

```
android:id="@+id/FYPname"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_alignParentBottom="true"  
android:layout_alignParentLeft="true"  
android:layout_alignParentStart="true"  
android:layout_marginBottom="0dp"  
android:fontFamily="sans-serif-smallcaps"  
android:text="Lor_Sheng_Qin_[B011310051]"  
android:textAllCaps="false"  
android:textColor="@color/FYP"  
/>
```

```
</RelativeLayout>
```



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## APPENDIX C

## ARDUINO CODING

```

#include <WiFiEsp.h>
#include <WiFiEspClient.h>
#include <WiFiEspServer.h>
#include <WiFiEspUdp.h>
#include <PubSubClient.h>
#include <AFMotor.h>

char ssid[] = "<Wifi_SSID>";
char pass[] = "<Wifi_Password>";
const char* mqtt_server = "<Server_Path>";
const char* username = "<Server_Username>";
const char* password = "<Server_Password>";
int status = WL_IDLE_STATUS;    // the Wifi radio's status

const int trigPin1 = 22; //front ultrasonic sensor
const int echoPin1 = 23;
//and so on for back, left and right

String inputmsg;
long duration1, duration2, duration3, duration4;
int distance1, distance2, distance3, distance4;
byte* payload;

```

```

AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);

//void callback(char* topic, byte* payload, unsigned int length);
WiFiEspClient espClient;
PubSubClient client(espClient);

void setup() {
  pinMode(trigPin1, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin1, INPUT); // Sets the echoPin as an Input
  //and so on for back, left and right
  Serial.begin(<Baudrate>); // Starts the serial communication
  Serial3.begin(<Baudrate>); // initialize serial for ESP module
  WiFi.init(&Serial3); // initialize ESP module
  setup_wifi();
  client.setServer(mqtt_server, <Server Port>);
  client.setCallback(callback);
  client.publish("Command", "");
}

void setup_wifi() {
  if (WiFi.status() == WL_NO_SHIELD) { // check for the
    ↪ presence of the shield
    Serial.println("WiFi_shield_not_present"); //
    while (true);
  } // attempt to connect to WiFi network
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to :");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network
  }
}

```

```

Serial.println("You're_connected_to_the_network");
Serial.print("IP_address:");
Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  char message_buff[19];
  int i=0;
  Serial.print("Message_arrived_from_");
  Serial.print(topic);
  Serial.print("]->");
  for(i=0; i<10; i++) {
    message_buff[i] = payload[i];
  }
  message_buff[i] = '\0';
  inputmsg = String(message_buff);
  Serial.println(inputmsg);
  omnimotor();
  memset(payload, 0, 20);
}

void omnimotor(){
  if (inputmsg == "stop"){
    client.publish("Arduino", "Stop_Received");
    mstop();
  } else if (inputmsg == "frwd" ){
    client.publish("Arduino", "Forward_Received");
    forward();
  } else if (inputmsg == "back" ){
    client.publish("Arduino", "Backward_Received");
    backward();
  } else if (inputmsg == "right" ){
    client.publish("Arduino", "Right_Received");
    right();
  }
}

```

```

} else if (inputmsg == "left" ){
  client.publish("Arduino", "Left_Received");
  left();
} else if (inputmsg == "rttr"){
  client.publish("Arduino", "Rotate_Right_Received");
  rotate_right();
  delay(550);
  mstop();
} else if (inputmsg == "rttl"){
  client.publish("Arduino", "Rotate_Left_Received");
  rotate_left();
  delay(550);
  mstop();
} else if (inputmsg == "Hi_Omni~ob" || inputmsg == "Hi_Omni~"){
  client.publish("Arduino", "Connected_to_OmniRobot!");
}
}

void mstop(){
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
};

void forward(){
  motor1.setSpeed(255);
  motor2.setSpeed(255);
  motor3.setSpeed(255);
  motor4.setSpeed(255);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}

```

```

};

void backward(){
//set speed and functions
};

void left(){
//set speed and functions
};

void right(){
//set speed and functions
};

void rotate_right(){
//set speed and functions
};

void rotate_left(){
//set speed and functions
};

int us_sensor1(){
    digitalWrite(trigPin1, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin1, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin1, LOW);
    duration1 = pulseIn(echoPin1, HIGH);
    distance1= duration1*0.034/2;
// Serial.print("Distance1: ");
// Serial.println(distance1);
    if (inputmsg == "frwd" && distance1 < 30){
        client.publish("Arduino", "Front_distance_<30cm!");
    }
}

```



```

        backward();
        delay(300);
        mstop();
    } else
return 0;
}

int us_sensor2(){
    digitalWrite(trigPin2, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin2, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin2, LOW);
    duration2 = pulseIn(echoPin2, HIGH);
    distance2= duration2*0.034/2;
// Serial.print("Distance2: ");
// Serial.println(distance2);
    if (inputmsg == "back" && distance2 < 30){
        client.publish("Arduino", "Back_distance<30cm!");
        forward();
        delay(300);
        mstop();
    } else
return 0;
}

int us_sensor3(){
    digitalWrite(trigPin3, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin3, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin3, LOW);
    duration3 = pulseIn(echoPin3, HIGH);
    distance3= duration3*0.034/2;

```

```

// Serial.print("Distance3: ");
// Serial.println(distance3);
if (inputmsg == "right" && distance3 < 30){
    client.publish("Arduino", "Back_distance_<_30cm!");
    left();
    delay(300);
    mstop();
} else
return 0;
}

int us_sensor4(){
    digitalWrite(trigPin4, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin4, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin4, LOW);
    duration4 = pulseIn(echoPin4, HIGH);
    distance4= duration4*0.034/2;
// Serial.print("Distance4: ");
// Serial.println(distance4);
if (inputmsg == "left" && distance4 < 30){
    client.publish("Arduino", "Left_distance_<_30cm!");
    right();
    delay(300);
    mstop();
} else
return 0;
}

void reconnect() { // Loop until we're reconnected
    while (!client.connected()) {
        Serial.println("Attempting connection to MQTT...");
        if(client.connect("Connect1", username, password)){

```

```
Serial.println("Connected to CloudMQTT.");
client.publish("Arduino", "OmniRobot Online!");
client.subscribe("Command");
} else {
Serial.print("Failed to connect MQTT, error");
Serial.println(client.state());
Serial.println("Try again in 1 seconds");
delay(1000);    // Wait 1 seconds before retrying
}
}
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  delay(100);
  us_sensor1();
  us_sensor2();
  us_sensor3();
  us_sensor4();
}
```



