



# WAREHOUSE INVENTORY SYSTEM USING MESSAGE QUEUING TELEMETRY TRANSPORT

This report is submitted in accordance with requirement of the Universiti Teknikal Malaysia Melaka (UTeM) for Bachelor Degree of Manufacturing Engineering (Hons.)



**GOH SHE HUI**

FACULTY OF MANUFACTURING ENGINEERING

2022

## DECLARATION

I hereby, declared this report entitled “Warehouse Inventory System Using Message Queuing Telemetry Transport” is the result of my own research except as cited in references.

Signature



Author's Name : GOH SHE HUI

Date : 26<sup>TH</sup> JAN 2022



## APPROVAL

This report is submitted to the Faculty of Manufacturing Engineering of Universiti Teknikal Malaysia Melaka as a partial fulfilment of the requirement for Degree of Manufacturing Engineering (Hons). The member of the supervisory committee is as follow:



## ABSTRAK

Kepintaran sistem inventori gudang merupakan salah satu penyelesaian untuk meningkatkan kecekapan penjejakan inventori kerana kebanyakan kesilapan manusia dalam menjejak inventori secara manual menyebabkan kesilapan salah ambil data dan kerugian syarikat. Oleh itu, kajian ini berkaitan dengan sistem IoT dalam inventori gudang menggunakan MQTT protokol dicadangkan sebagai sesuatu penyelesaian untuk menggantikan kerja menjejak inventori secara manual. *Message Queuing Telemetry Transport (MQTT)* protokol mempunyai sifat automatik dalam menghantar data terkini yang dapat dijejak melalui telefon bimbit. Keupayaan Seni bina MQTT protokol dikaji menggunakan aplikasi berlainan dalam mempapar mesej dari MATLAB ke Raspberry Pi dan MATLAB ke ThingSpeak. Kedua-dua seni bina MQTT mempunyai keupayaan kerana data mempapar adalah sama dengan data diterima. Selain itu, ThingSpeak juga boleh menghantar amaran kepada pengguna semasa terdapat kekurangan kuantiti inventori yang tetap. Amaran ini dapat mengelakkan kekurangan inventori dalam kilang. Kajian ini juga akan melaksanakan kecekapan analisi dan kos analisi dengan menggunakan MQTT protokol.



## ABSTRACT

Smart warehouse inventory management has become a solution to maximize the efficiency of keeping track of inventory as there is a lot of human error which is prone to the inefficiency of manual records and sharing of paper records which can lead to loss of a company. Hence, the IoT-based warehouse inventory will be presented using MQTT protocol to replace manual inventory tracking. The Message Queuing Telemetry Transport (MQTT) protocol offers automated features which can publish real-time data and can be tracked by using a mobile device. The MQTT Protocol architecture is study the capability in using different applications such as publishing data from MATLAB to Raspberry Pi and MATLAB to ThingSpeak. Both MQTT architecture are capable as the data publish is the same as the data received. Besides, the engineer will be notified of the low inventory of the stock through the E-mail from ThingSpeak will keep update to ensure the stock is top up. On the other hand, this paper will also present the efficiency analysis and cost analysis of implementing MQTT protocol.

## ACKNOWLEDGEMENT

I would like to express my gratitude to my supervisor, Dr. Mohd Nazrin bin Muhammad to provide me with valuable insights and encouragement on this project. I would also like to acknowledge my panels which are Ir. Dr.-Ing. Azrul Azwan Bin Abdul Rahman, En. Mahasan Bin Mat Ali and Ir. Dr. Lokman Bin Abdullah for the valuable suggestions and comments on my project. A special thanks to Ir. Dr. Lokman Bin Abdullah that willing to share the technique on the report writing skills. I would also like to thank the coordinators of *Projek Sarjana Muda* (PSM), Mr. Nor Akramin bin Mohamad and Dr. Chang Siang Yee to invite different speakers to provide us the guideline for the presented PSM report.

اونيورسيتي تيكنيكل مليسيا ملاك  
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

# TABLE OF CONTENTS

<b>ABSTRAK</b> .....	<b>i</b>
<b>ABSTRACT</b> .....	<b>ii</b>
<b>ACKNOWLEDGEMENT</b> .....	<b>iii</b>
<b>TABLE OF CONTENTS</b> .....	<b>iv</b>
<b>LIST OF FIGURES</b> .....	<b>viii</b>
<b>LIST OF TABLES</b> .....	<b>xi</b>
<b>LIST OF ABBREVIATIONS</b> .....	<b>xii</b>
<b>LIST OF SYMBOLS</b> .....	<b>xiv</b>
<b>CHAPTER 1</b> .....	<b>1</b>
<b>INTRODUCTION</b> .....	<b>1</b>
1.1 Research Background .....	1
1.2 Problem Statement.....	2
1.3 Objectives.....	3
1.4 Scopes of the Research .....	4
1.5 Rational of Research.....	4
1.6 Organization of the Study.....	5
1.7 Summary.....	5
<b>CHAPTER 2</b> .....	<b>6</b>
<b>LITERATURE REVIEW</b> .....	<b>6</b>
2.0 Warehouse Inventory System .....	6
2.1 Internet of Think (IoT).....	8
2.2 IoT based Inventory System .....	9
2.2.1 Sensor In Inventory Tracking Systems.....	9
2.2.1.1 Barcode.....	9
2.2.1.2 Computer Vision .....	11
2.2.1.3 Radio Frequency Identifier (RFID).....	12
2.2.1.4 Conclusion.....	14
2.2.2 IoT Gateway .....	15
2.2.2.1 NodeMCU .....	15

2.2.2.2	Arduino.....	16
2.2.2.3	Raspberry Pi.....	17
2.2.2.4	Conclusion.....	18
2.2.3	Network Communication .....	18
2.2.3.1	Bluetooth.....	19
2.2.3.2	Wi-Fi.....	19
2.2.3.3	Bluetooth vs Wi-Fi.....	20
2.2.3.4	Conclusion.....	21
2.2.3.5	Communication Protocol.....	21
2.2.3.6	HTTP.....	22
2.2.3.7	CoAP.....	22
2.2.3.8	MQTT.....	23
2.2.3.9	MQTT vs Other Communication Protocol.....	25
2.2.3.10	Conclusion.....	27
2.2.4	Cloud Service Application .....	27
2.2.4.1	MIT App Inventor.....	27
2.2.4.2	Blynk.....	28
2.2.4.3	ThingSpeak.....	29
2.2.4.4	Conclusion.....	30
2.3	Relevant Works.....	31
2.4	Summary.....	32
<b>CHAPTER 3</b>		<b>33</b>
<b>METHODOLOGY .....</b>		<b>33</b>
3.0	Project Development Model .....	33
3.1	Flow Chart of General Project development .....	34
3.2	Design of Proposed Inventory System Framework with MQTT.....	35
3.3	MQTT Protocol Architectural.....	35
3.4	Requirement Specification.....	36
3.5	Technologies in Implementation of MQTT protocol .....	37
3.6	Implementation of IoT-based Inventory System Simulation.....	38
3.7	Raspberry Pi Implemented with MATLAB using MQTT.....	40
3.7.1	MQTT Protocol Architectural .....	40
3.7.2	MQTT Publisher.....	41
3.7.3	MQTT Broker.....	42
3.7.4	MQTT Subscriber.....	42



3.8	ThingSpeak Implemented with MATLAB using MQTT .....	44
3.8.1	MQTT Protocol Architecture .....	44
3.8.2	MQTT Publisher.....	44
3.8.3	MQTT Subscriber.....	45
3.8.3.1	ThingSpeak.....	46
3.8.3.2	ThingView.....	48
3.9	Alert System in ThingSpeak .....	48
3.10	Efficiency Analysis.....	49
3.11	Cost Analysis .....	51
<b>CHAPTER 4.....</b>		<b>52</b>
<b>RESULT AND DISCUSSION .....</b>		<b>52</b>
4.0	Implementation of IoT-based Inventory System .....	52
4.1	MQTT Protocol Architecture with Raspberry Pi.....	56
4.1.1	Setup Raspberry Pi .....	56
4.1.2	Capability of MQTT Publisher.....	56
4.1.3	Capability of MQTT Subscriber.....	59
4.2	MQTT Protocol Architecture with ThingSpeak .....	63
4.2.1	Capability of MQTT Publisher.....	65
4.2.2	Capability of MQTT Subscriber.....	68
4.2.2.1	ThingSpeak.....	68
4.2.2.2	ThingView.....	69
4.3	Send E-mail Alerts from ThingSpeak.....	70
4.3.1	Implementation of Alert System .....	70
4.3.2	Performance of the Alert system .....	70
4.3.3	The Efficiency of the Alert System .....	72
4.4	Efficiency Analysis.....	73
4.5	Cost Analysis .....	74
<b>CHAPTER 5.....</b>		<b>76</b>
<b>CONCLUSION AND RECOMMENDATIONS .....</b>		<b>76</b>
5.0	Conclusion .....	76
5.1	Recommendation .....	77
5.2	Improvement of the System.....	78
5.3	Sustainability .....	78
<b>REFERENCES .....</b>		<b>79</b>
<b>APPENDIX .....</b>		<b>85</b>

Appendix A .....	85
Appendix B.....	86
Appendix C.....	87
Appendix D .....	89
Appendix E .....	91
Appendix F .....	91



## LIST OF FIGURES

Figure 2.1 Warehouse Management System (Vatumalae et al., 2020) .....	6
Figure 2.2 Technology roadmap for Internet of Things (IoT) (Tejesh & Neeraja, 2018).....	7
Figure 2.3 Barcode .....	9
Figure 2.4 The Case diagram of the operation of Barcode Inventory Control System (Muyumba & Phiri, 2017) .....	10
Figure 2.5 Automatically object and human detection and classification (Khan, A. I. and Al-Habsi, S. (2020)) .....	11
Figure 2.6 Detection process of the object or image (Robotics Tomorrow) .....	12
Figure 2.7 The components of RFID system (Tejesh & Neeraja, 2018).....	13
Figure 2.8 RFID tag (Adcbarcode,2017).....	13
Figure 2.9 General feature of IoT Gateway (Kang et al., 2017).....	15
Figure 2.10 NodeMCU (Paveesha et al., 2020).....	15
Figure 2.11 Arduino Uno Board and L293D Motor Driver Shield (Pololu Corporation, Open Circuit).....	16
Figure 2.12 Raspberry Pi (Seth Kenlon,2021) .....	17
Figure 2.13 ZF & Bluetooth® Low Energy (ZF Friedrichshafen AG) .....	19
Figure 2.14 Wi-Fi (Autodesk).....	19
Figure 2.15 Percentage of successful activity of the attacks (Susila et al., 2017).....	20
Figure 2.16 Communication Protocols in different layer (B. Mishra & Kertesz, 2020).....	21
Figure 2.17 Architectural of HTTP (Yokotani & Sasaki, 2017) .....	22
Figure 2.18 Comparison between architecture of CoAP and HTTP (Devopedia, 2019) ....	23
Figure 2.19 The MQTT development over time (B. Mishra & Kertesz, 2020) .....	23
Figure 2.20 Architecture of MQTT Protocol (Pham & Hoang, 2021).....	24
Figure 2.21 MQTT Protocol on tracking shipment .....	25
Figure 2.22 Request and Respond model used by HTTP and CoAP .....	26
Figure 2.23 Publish and Subscribe model used by MQTT .....	26
Figure 2.24 Graphical interface editor of MIT App Inventor (Mir & Lluca, 2020).....	27
Figure 2.25 Code editor of MIT App Inventor (Mir & Lluca, 2020) .....	28
Figure 2.26 Switch ON button in Blynk app.....	28
Figure 2.27 Schematic of the Blynk's component.....	29



Figure 2.28 Sensor output on ThingView Free APP (Tapakire & Patil, 2019).....	29
Figure 3.1 Waterfall Model .....	33
Figure 3.2 Flow Chart of the project overview .....	34
Figure 3.3 Hardware inventory system framework .....	35
Figure 3.4 MQTT Protocol architectural .....	35
Figure 3.5 Implementation of MATLAB in replacing the experimental in lab. ....	36
Figure 3.6 Machine Type-B (MTB) robot.....	38
Figure 3.7 Algorithm of the simulation in CoppeliaSim .....	39
Figure 3.8 MQTT Protocol between MATLAB and Raspberry Pi .....	40
Figure 3.9 Algorithm of the coding in MATLAB connected to Raspberry Pi.....	41
Figure 3.10 Set up Raspberry Pi connect to MQTT in Matlab.....	42
Figure 3.11 Scanning IP address of Raspberry Pi .....	43
Figure 3.12 PuTTY Configuration .....	43
Figure 3.13 Interface of the command-line of Raspberry Pi .....	43
Figure 3.14 MQTT Protocol between MATLAB and ThingSpeak.....	44
Figure 3.15 Algorithm of the coding in MATLAB connected to ThingSpeak .....	45
Figure 3.16 Creating channel in ThingSpeak .....	46
Figure 3.17 Get MQTT under 'Device' .....	46
Figure 3.18, Add a new device under MQTT Device .....	47
Figure 3.19 Fill up the details in the following column .....	47
Figure 3.20 Specific MQTT credentials for the channel. ....	47
Figure 3.21 Add channels to allow publish and subscribe .....	47
Figure 3.22 ThingView in subscribing channels .....	48
Figure 3.23 Algorithm of the alert system in ThingSpeak .....	49
Figure 3.24 'Run and Time' in Matlab (Editor) .....	50
Figure 4.1 IoT-based inventory system .....	52
Figure 4.2 Coding in MATLAB to calculate the angle $\theta_1$ and $\theta_2$ .....	53
Figure 4.3 Part of the MTB language .....	54
Figure 4.4 Initial environment of simulation (left) and end environment of simulation (right).....	54
Figure 4.5 Stock is scanned and loaded onto conveyor belt.....	55



Figure 4.6 The RFID active area for reading RFID .....	55
Figure 4.7 Wi-Fi set-up of Raspberry Pi .....	56
Figure 4.8 Properties of Raspberry Pi in Matlab .....	57
Figure 4.9 Connection between Raspberry Pi and Matlab using MQTT .....	57
Figure 4.10 The inventory quantities update in MATLAB .....	57
Figure 4.11 Real-time inventory update for four stock in Raspberry Pi .....	58
Figure 4.12 Data Received of Topic A in Raspberry Pi.....	60
Figure 4.13 Data Received of Topic B in Raspberry Pi.....	61
Figure 4.14 Data Received of Topic C in Raspberry Pi.....	62
Figure 4.15 Data Received of Topic D in Raspberry Pi.....	63
Figure 4.16 Interface of the ThingSpeak with 4 fields.....	64
Figure 4.17 MQTT credential.....	64
Figure 4.18 Successfully connected between MATLAB and ThingSpeak via MQTT.....	65
Figure 4.19 The inventory quantities update in MATLAB.....	65
Figure 4.20 Real-time Inventory Update for Four Stock in ThingSpeak.....	66
Figure 4.21 Time-Table generated by function call ThingspeakRead in Matlab.....	67
Figure 4.22 Graph generation by updating the data from MATLAB to ThingSpeak in different fields.....	68
Figure 4.23 Front page interface of ThingView .....	69
Figure 4.24 Screenshot from the phone.....	69
Figure 4.25 Profile of the ThingSpeak.....	70
Figure 4.26 MATLAB code successfully run .....	71
Figure 4.27 Notification sent to e-mail when low inventory.....	71
Figure 4.28 Last update of the Stock A's quantities.....	72
Figure 4.29 ThingSpeak Alerts in every 5 minutes.....	72
Figure 4.30 Total time taken of the MQTT protocol with Raspberry Pi.....	73
Figure 4.31 Total time taken of the MQTT protocol with Raspberry Pi.....	73
Figure 4.32 Comparison of 'Free version' and 'Standard version' of ThingSpeak (MathWorks, 2022b) .....	74
Figure 4.33 Features of the 'Standard Version' which cost USD 710.00/year .....	75
Figure 4.34 Features of the 'Standard Version' which cost USD 4,150.00/year .....	75

## LIST OF TABLES

Table 2.1 Comparison Between Manual, Semi-automated, and Fully-automated system (Abugabah et al., 2020; Tejesh & Neeraja, 2018).....	7
Table 2.2 Pros and Cons of Barcode Sensor .....	10
Table 2.3 Pros and Cons of Computer Vision.....	12
Table 2.4 Pros and Cons of RFID .....	14
Table 2.5 Different types of Raspberry Pi with either support or no support MATLAB ...	18
Table 2.6 Bluetooth vs Wi-Fi compare between (Autodesk) .....	21
Table 2.7 Type of QoS (B. Mishra & Kertesz, 2020; Pham & Hoang, 2021) .....	24
Table 2.8 Advantages of MQTT.....	25
Table 2.9 MQTT Compared With Others Communication Protocol (Jaloudi, 2019).....	26
Table 2.10 List of research that related with IoT Warehouse Inventory System .....	31
Table 3.1 Requirement specification of the project.....	36
Table 3.2 Technologies in implementation of MQTT protocol .....	37
Table 3.3 Initially quantities of stock in Raspberry Pi implementation .....	42
Table 3.4 Initially quantities of stock in ThingSpeak implementation .....	45
Table 3.5 Variables in comparison of the performance of Raspberry Pi and ThingSpeak .	49
Table 3.6 Result of the performance of Raspberry Pi and ThingSpeak.....	50
Table 3.7 Parameter of the efficiency.....	50
Table 3.8 Implementation Cost .....	51
Table 4.1 True coordinate on x-axis and y-axis. ....	53
Table 4.2 Stocks quantities updated of Raspberry Pi for 20 Times .....	58
Table 4.3 Stocks quantities updated of Thingspeak for 20 Times .....	66
Table 4.4 Result of the performance and efficiency of Raspberry Pi and ThingSpeak .....	73

## LIST OF ABBREVIATIONS

AC	-	Alternative Current
API	-	Application Programming Interface
APK	-	Android Package
APP	-	Application
ASRS	-	Automated Storage and Retrieval System
BLE	-	Bluetooth Low Energy
CoAP	-	Constrained Application Protocol
Covid-19	-	Coronavirus Disease 2019
CPU	-	Central Processing Unit
DC	-	Direct Current
DOF	-	Degree-Of-Freedom
DoS	-	Denial-Of-Service
GPIO	-	General-Purpose Input/Output
HTTP	-	Hypertext Transfer Protocol
ID	-	Identity Document
IoT	-	Internet Of Things
IP	-	Internal Protocol
JIT	-	Just-In-Time
JSON	-	JavaScript Object Notation
M2M	-	Machine To Machine
MAC	-	Media Access Control
MCU	-	Microcontroller
MITM	-	Man-In-The-Middle
MQTT	-	Message Queuing Telemetry Transport
MTB	-	Machine Type-B
NPO	-	Non-Profit Organization
OASIS	-	Organization For the Advancement of Structured Information Standards
PC	-	Personal Computer

QoS	-	Quality Of Service
QR	-	Quick Response
RF	-	Radio Frequency
RAM	-	Random Access Memory
RFID	-	Radio Frequency Identification
ROI	-	Return of Investment
ROM	-	Read-Only Memory
SBC	-	Single Board Computer
SCADA	-	Supervisory Control and Data Acquisition
SSL	-	Secure Sockets Layer
SSP	-	Secure Simple Pairing
TCP	-	Transmission Control Protocol
TLS	-	Transport Layer Security
TTL	-	Time-To-Live
UDP	-	User Datagram Protocol
UHF	-	Ultrahigh Frequency
USB	-	Universal Serial Bus
Wi-Fi	-	Wireless Fidelity
WIP	-	Work-In-Progress
WMS	-	Warehouse Management System





## LIST OF SYMBOLS

cm	-	Centimeter
GHz	-	Giga Hertz
m	-	Meter
mm	-	Millimeter
Mbps	-	Megabits per second
RM	-	Ringgit Malaysia
s	-	Second
USD	-	United State Dollar
%	-	Percentage




# CHAPTER 1

## INTRODUCTION

Chapter 1 will establish the research background on the Radio Frequency Identification (RFID) and Message Queuing Telemetry Transport (MQTT) in this coming trend and identify the problems exist in the Warehouse Inventory Management. Then determine the objective, scope and structural of the project.

### 1.1 Research Background



In this age of globalism, the transfer of goods has expanded much more than ever before. As such, inventory management has become a crucial part of the industry. For example, small retailer or e-commerce shop rely heavily on inventory management for planning and decision making. The transparency in the warehouse inventory between the buyers or customers have to be maintained. (Laxmi & Mishra, 2018) Darya Plinere et al (2015) stated that production scheduling can be more effective through inventory management which helps in forecasting and purchasing of stock. Hence, efficient asset utilization is achieved and issues such as over stocking or out-of-stock can be avoided. A well-known inventory system is the Just-In-Time (JIT) inventory system by Toyota Motor Corporation which utilizes data of inventory to react quickly and efficiently on the trend of demand. Besides, Syed Mohamad et al. (2016) also mentioned that the inventory management is directly influence the company's performance so it plays an important role in financial performance of company. Hence, company must keep track on the amount of the inventory to maintain a proper inventory level to maximize profitability.

In conjunction to the mass availability of electronics such as sensors, Internet of Things (IoT) has become an emergent technology that would revolutionize inventory management in the industry. Radio Frequency Identification (RFID) technology offers effective inventory tracking by exposing the physical placement of inventory to digital systems (Aishwarya Raj Laxmi, 2018). For example, real time visualization of work-in-progress (WIP) or finished products can be achieved by attaching it with RFID tags which identifies its location.

Therefore, an inventory system is proposed in this research which utilizes Message Queuing Telemetry Transport (MQTT) protocol to facilitate the communication between PC and mobile devices through the cloud. The publishing devices will publish data through broker to subscribing devices. The overall architecture involves inventory data gathered from RFID sensors which is published to the cloud through a Wi-Fi gateway using MQTT protocol, and subsequently subscribed by PC or mobile phone for monitoring purposes. Cloud database is utilized as a scalable pay-per-use data storage solution.

The outbreak of Corona Virus Disease 2019 (COVID-19) has devastated economic sector especially supply chain. (Sube Singh et al, 2020). Thus, highlighting the importance of communication between companies and suppliers to coordinate the transfer of goods, which can be made more effective through a cloud-based inventory management system. Decisions on logistics can be made more efficiently by accessing data on the availability and demands of supplies and products with the tap of a mobile device through the cloud.

## 1.2 Problem Statement

The task of inventory management is to define the quantity of current inventories that will fulfill the demand (Darya Plinere et al, 2015). However, the impact of supply issues such as overstock or out-of-stock will be exacerbated by a small inventory. One of the solutions to alleviate the risks in a supply chain is by improving the inventory tracking system. Current widely used paper-based manual inventory counting system is inefficient and prone to human error, which sometimes results in improper inventory control and missing items. Besides, the paper records are shared between departments which may lead to further delays and errors.

Therefore, it is obvious that there is a need to revolutionize warehouse inventory tracking system by computerization of data and automation of inventory counting operations (Thomas Muyumba, 2017). Currently, web-based inventory tracking system using barcode is available on the market. Although it is simpler and cheaper to use it, but it is less durable and less secure compared to RFID-based system.

In addition, the mass availability of mobile phones presents an opportunity to expand the accessibility of inventory tracking system. Adopting mobile devices in asset monitoring eliminates the need for human operators to physically be present on site. This is important as human travel time significantly contributes to the loss of efficiency in a production line.

Furthermore, manual inventory counting brings about the risk of human error. This is especially apparent during the COVID-19 outbreak, where labour shortage lead to overworked workers which are prone to mistakes (Singh et al., 2021). Reduction of labour requirements in inventory tracking will improve efficiency as workers can then be tasked with more impactful duties.

### 1.3 Objectives

The objectives of the project are as below:

- (a) To investigate current IoT-based Warehouse Inventory Management System in using a suitable technology to track the inventory quantities.
- (b) To develop IoT-based Warehouse Inventory Management System with MQTT protocols that can track inventory quantities by various application and mobile devices.
- (c) To evaluate the efficiency and performance of the MQTT protocol architectural and notification system of ThingSpeak when the inventory is low.



## 1.4 Scopes of the Research

The project scopes of research are shown as below:

- (a) Research will be conducted on the IoT-based Warehouse Inventory Management with compared between 3 types of inventory system and how the IoT integrated to increase the efficiency of the Warehouse Inventory System.
- (b) The study on the IoT-based Warehouse Inventory Management system with MQTT protocol in the aspects of architectural and component. Comparison is made between different type of sensors, network connection and IoT protocols.
- (c) Appropriate method to develop MQTT protocol is studied and the comparison is made between different type of Cloud service provider.
- (d) Extend the existing warehouse management system by enhancing the system into Cloud Based service that can accessed by different users in a software-based.



## 1.5 Rational of Research

The rational of project research are shown as below:

- (a) There is different type of inventory system that need to be studied to define the different on the pro and cons of each system and the integration of the IoT to help in giving a bigger picture on the history.
- (b) Different type of IoT components have different system requirement. characteristic and working principle of each components have to be weighed

when deciding a suitable component in implementing an IoT system that can connect to the mobile phone with suitable Cloud Service provider.

- (c) To construct a reliability alert system of ThingSpeak to notify the user when low inventory occurred.

## 1.6 Organization of the Study

An IoT based RFID inventory tracking system using MQTT is proposed in this thesis. The findings in this report are divided into 5 Chapters. Chapter 1 is Introduction which establishes the background study of the title and giving an overall idea about the project objective, scope of research, and rational of research. Next, Literature Review is presented in Chapter 2 which comprises of previous studies to highlight the work done on automated inventory control system and its components. Research methodology is presented in Chapter 3 which includes the Software and hardware to be used and the detailed architecture of the proposed IoT system. Preliminary Results in Chapter 4 is analyzed to ensure the project developed is capable and efficient. Lastly, conclusion and recommendation on this research are summarized in Chapter 5.

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## 1.7 Summary

In short, background on current inventory tracking and control system is presented in this chapter. The problem statement is also highlighted which revolves around the reliance of current systems on manual human labour. Then, the objectives of this research which emphasizes on automation of inventory tracking system are proposed and justified with the respective rationales. In addition, the scope of this research is staged to set limitations and expectations on the proposed project.

## CHAPTER 2

### LITERATURE REVIEW

This chapter is a compilation of work done by other researchers on topics related to the proposed project. Information that are relevant to inventory tracking and management, along with Internet of Things (IoT) using Message Queuing Telemetry Transport (MQTT) protocol is presented and discussed.

#### 2.0 Warehouse Inventory System

Warehouse Management System (WMS) is Software that is designed for optimizing inventory flow between warehouses and distribution centers. (Rana, 2020)



Figure 2.1 Warehouse Management System (Vatumalae et al., 2020)

The main goal of WMS is to manipulate the movement of products within the facility to minimize the loss of time. By integrating semi-automated and automated



warehouse management system, companies can reduce effort and improve efficiency to produce more consistent results compared to the manual system. Bar code technology is considered as a type of semi-manual system that uses optical scanner to read labels and transmit data into computer systems. However, it is expensive and susceptible to security issues. To improve upon the flaws of bar code systems, RFID technology is introduced as a fully-automated system that is capable of larger data capacity. (Tejesh & Neeraja, 2018) The role of RFID in localization, tracking and positioning applications within the IoT architecture is shown in Figure 2.2, along with future directions for expansion of the architecture.

Table 2.1 Comparison between Manual, Semi-automated, and Fully-automated system (Abugabah et al., 2020; Tejesh & Neeraja, 2018)

Inventory Tracking System	Manual	Semi-automated (Barcode)	Fully-automated (RFID)
Efficiency	Low	High	High
Consistency of result	Low	High	High
Data Storage	Paper-Work	System with Low Capacity	System with High Capacity
Challenges	Human Error	Unreadable Label That Is Damaged	Low-Cost effectiveness (maintenance & testing)

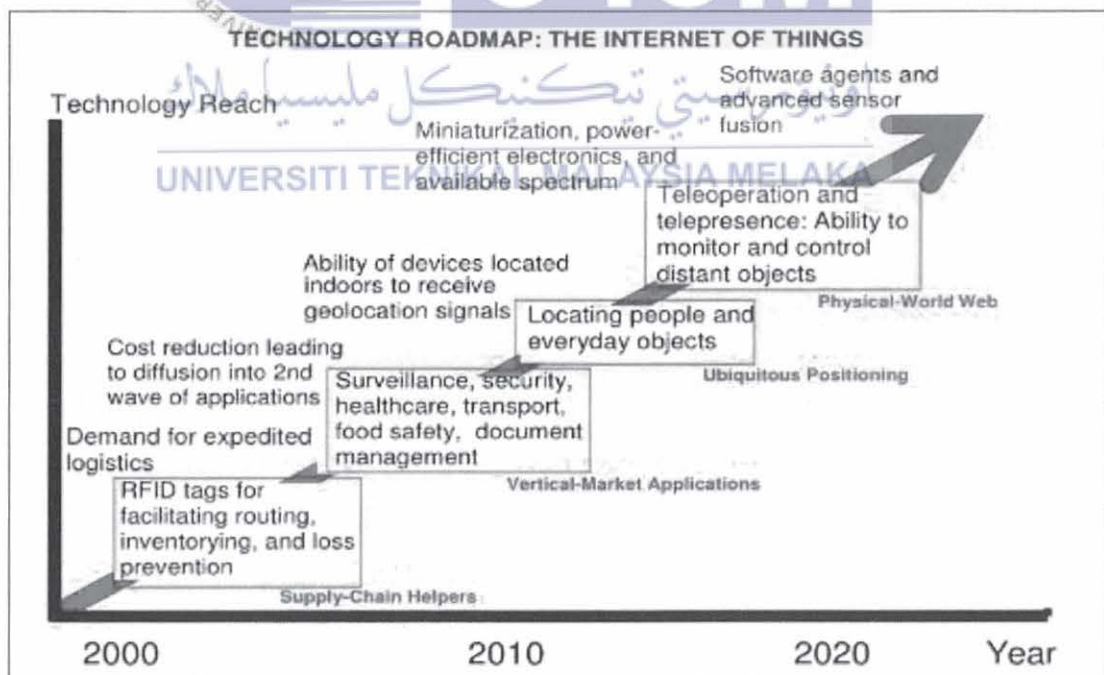


Figure 2.2 Technology roadmap for Internet of Things (IoT) (Tejesh & Neeraja, 2018)

Besides, Mathaba et al (2017) states the IoT technology able to connect between massive number of things and sensors. This helps to detect the location of the products and level of inventory. The using of Internet technology can increase the productivity and reduce cost. Functional requirement in IoT inventory is more to complete task manually while non-functional requirements are focus on the system properties that give a short feedback time, high security and scalability.

## 2.1 Internet of Think (IoT)

The use of Internet of Things (IoT) architecture is rising in the sector of industrial automation and also in consumer products. It able to help different experts such as business experts or production engineer. (Nath, 2017)

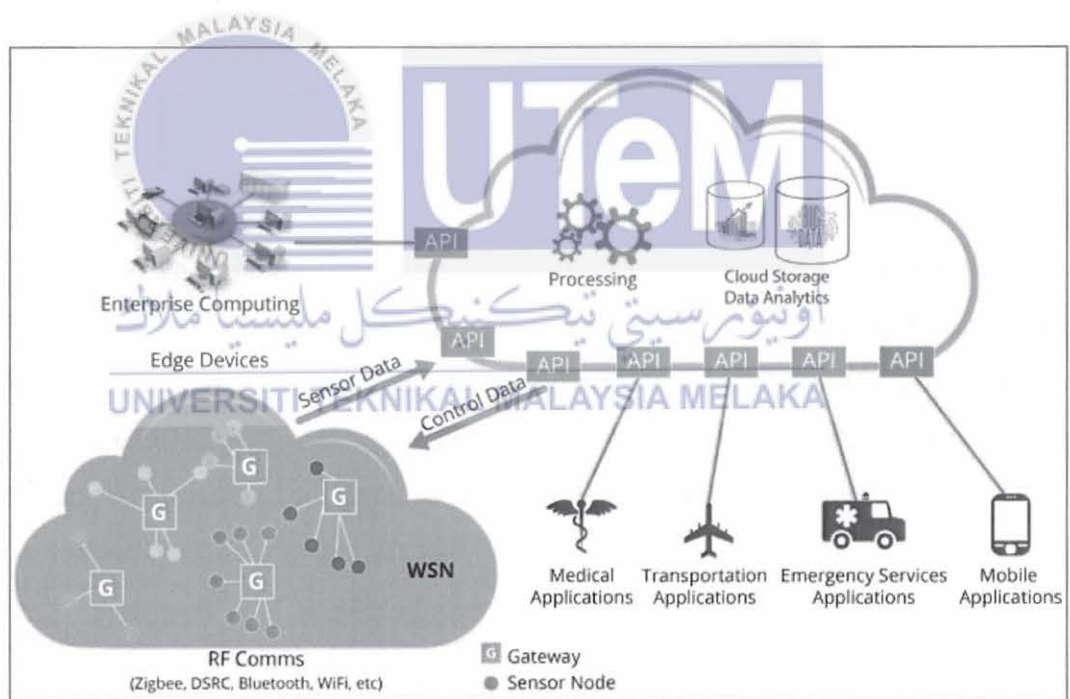


Figure 2.3 Internet of Things (IoT) and the connected sensor world (Tabassum et al., 2020)

In short, it is a giant network that connects and coordinates between sensing and actuating devices. Data is collected and shared between the connected embedded devices which allows remote monitoring and control in a wireless local area network (Yüksel, 2020) or over the Internet (Paveesha et al., 2020) Data integration is a key component of the complex data structure of an IoT-based warehouse that served with main components such as configuration, databasing and transmission. (Aamer & Sahara, 2021)

## 2.2 IoT based Inventory System

High complexity is involved in the processes within warehouse management systems (WMS), especially in the tracking of inventory. Hence, the propagation of IoT architecture has provided manufacturers and wholesalers with a better way to track and monitor inventory by ensuring a smooth and efficient operation (Rana, 2020). Common inventory tracking technologies include computer vision, bar code, and Radio Frequency Identifier (RFID) (Octaviani & Ce, 2020). In conjunction with Industrial Revolution 4.0, network communication systems such as Bluetooth and Wi-Fi have become commonplace in industrial applications. Therefore, the stated technologies will be reviewed for their respective strength and weaknesses.

### 2.2.1 Sensor In Inventory Tracking Systems

#### 2.2.1.1 Barcode

Barcode is a method to encode information using black and white vertical bars with varying thickness and spacings, as shown in Figure 2.3. Information can be retrieved from bar code tags by scanning and decoding it through a bar code scanner, and performing a look up of the code in a registered database. (Putra Yudha et al., 2018) Barcode recognition process is easily influenced by environmental conditions such as light intensity and humidity. A significant step involved in barcode recognition is noise detection which filters noise produced in data transmission circuits within the scanner. (Bing & Yang, 2019)

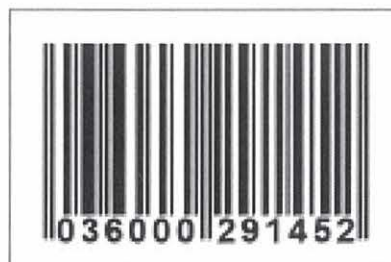


Figure 2.3 Barcode

In bar code inventory system, the interaction between the users and the system is shown in Figure 2.5, which consists of middleware, barcode reader, and application users.



The data is scanned by the barcode reader and sent to the middleware. Then, the backend database manages the exchange of information between the reader and user. (Muyumba & Phiri, 2017)

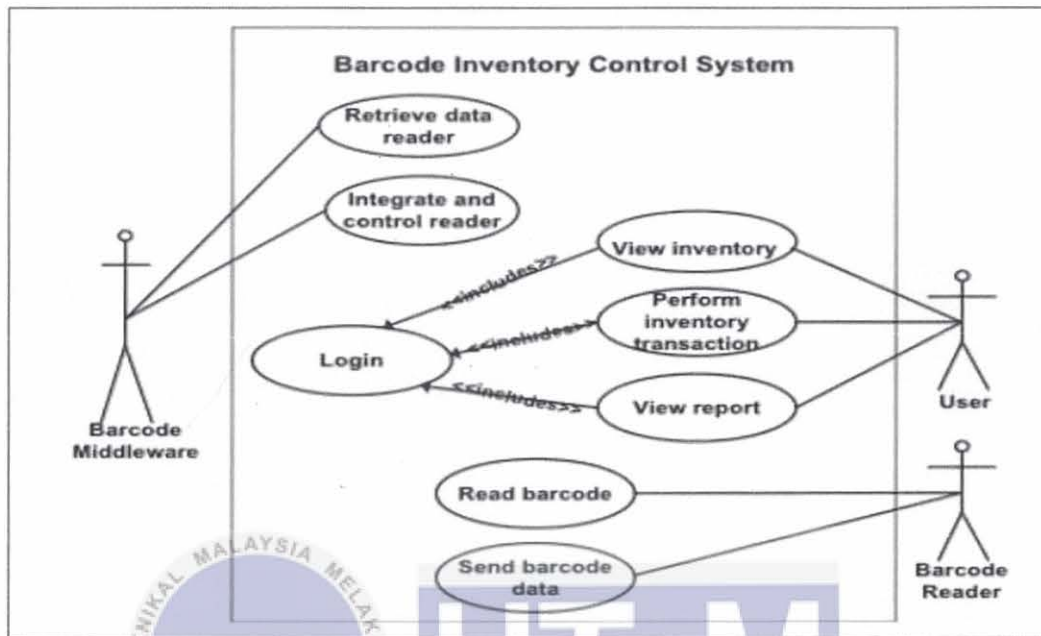


Figure 2.4 The Case diagram of the operation of Barcode Inventory Control System (Muyumba & Phiri, 2017)

اونيورسيتي تكنولوجيكا مليسيا ملاك  
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table 2.2 Pros and Cons of Barcode Sensor

Concerns	Pros	Cons
Cost concern	<ul style="list-style-type: none"> <li>It is inexpensive as it can be designed and print, extremely flexible to implement, and can be used in other aspects.</li> </ul>	<ul style="list-style-type: none"> <li>High Overhead cost as the inventory to be tracked is a lot.</li> </ul>
Time concern	<ul style="list-style-type: none"> <li>Time efficiency in less training time needed for operator to familiar.</li> </ul>	<ul style="list-style-type: none"> <li>Low time efficiency in finding the inventories as it needs to be referred to on the spreadsheet manually.</li> </ul>
Performance concern	<ul style="list-style-type: none"> <li>Can automatically send data to cloud-based.</li> <li>Almost no privacy issue involved in use.</li> </ul>	<ul style="list-style-type: none"> <li>Short range</li> <li>Does not have read or write capabilities</li> <li>Human error would happen</li> <li>Damaged / Blurred barcode over time.</li> </ul>

### 2.2.1.2 Computer Vision

Computer vision can be defined as the extraction of rich information from a computer image input. Computer images are inputs to computer vision systems that either represents color information such as pixels and voxels, or it can be vectors consisting of size, edge, color , texture distribution (Strain et al., 2020) Computer vision is used to detect patterns and recognizing them in images.(Khan & Al-Habsi, 2020)

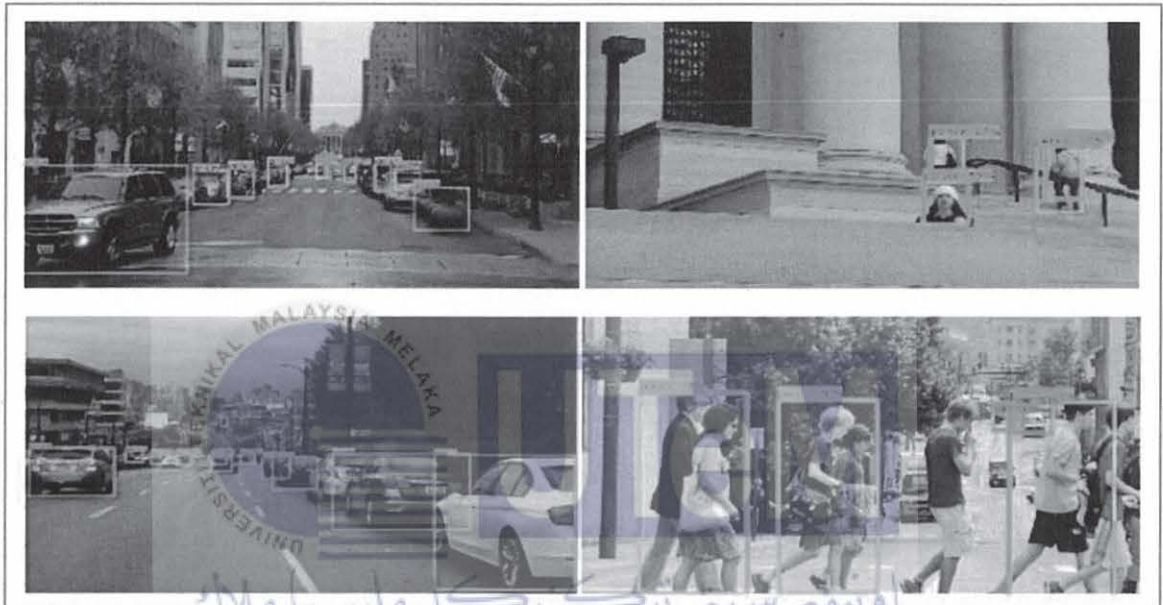


Figure 2.5 Automatically object and human detection and classification (Khan, A. I. and Al-Habsi, S. (2020))

Another application of computer vision is reconstructing 3-dimensional scene from a 2-dimensional image. This is achieved through models and algorithms such as using camera and performing geometric transformations on the image illustrate in Figure2.6. Computer vision system provides the solution to automated identification, tracking and measurement of objects using a small number of sensors. (Tian et al., 2020) From the project of Saha & Agarwal (2019), computer vision tools are developed to increase the efficiency of inventory management in a warehouse by identifying Quick Response codes (QR-codes) on goods that are matched to alphanumeric codes on shelves where the goods should be placed. Open-source computer vision (OpenCV) and ZBar libraries are used to identify the QR code while deep learning text detection is performed using Tesseract library.



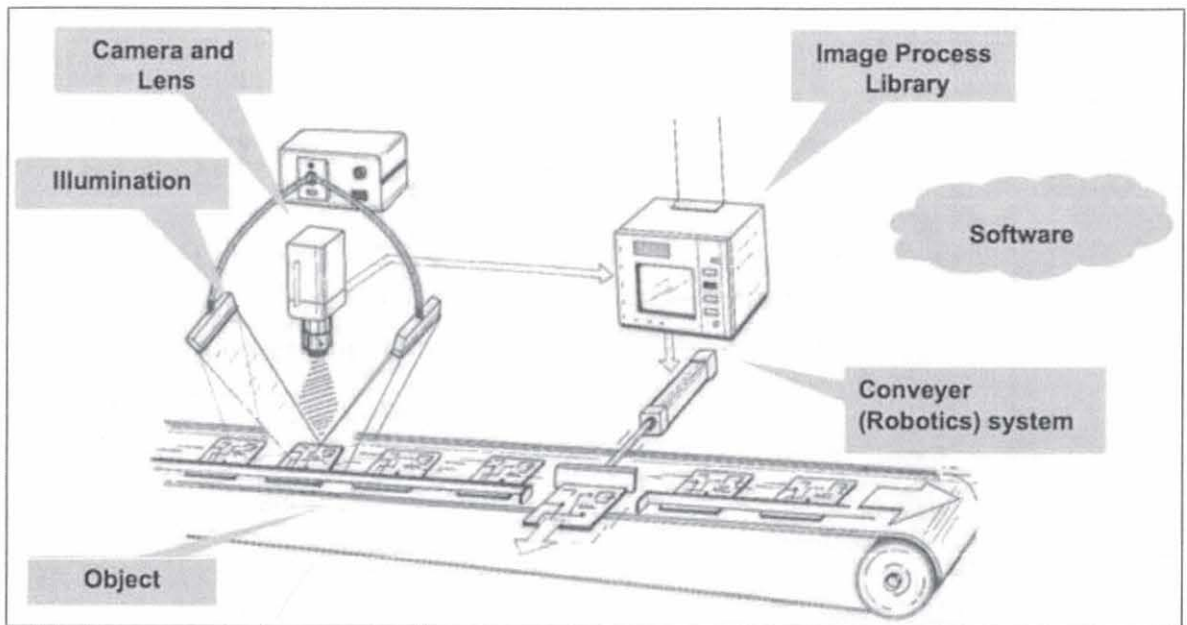


Figure 2.6 Detection process of the object or image (Robotics Tomorrow)

Table 2.3 Pros and Cons of Computer Vision

Concerns	Pros	Cons
Cost concern	High-cost efficiency because it is independent as it does not require a transponder like RFID tags or any printed object to attach on the target product.	High-cost implementation for high technology infrastructure.
Time concern	Time efficiency in reducing operator training time as it is easy to use.	Time needed in recheck of the undefined picture that captured by the camera vision.
Performance concern	<ul style="list-style-type: none"> <li>• Accuracy as it does not have limitation like as human perception.</li> <li>• Simplicity infrastructure that can upgrade and replace easily.</li> </ul>	<ul style="list-style-type: none"> <li>• Short range</li> <li>• Bad Picture Quality reduce the performance and accuracy in collect the data.</li> <li>• Cannot perform when it got obstacle block of the camera.</li> </ul>

### 2.2.1.3 Radio Frequency Identifier (RFID)

Radio Frequency Identifier (RFID) technology is ubiquitous in its implementation within inventory management systems to identify products or goods by the unique tag attached. The chipless RFID tag offers a wireless, cheap and high accuracy localization (Fathi et al., 2020). According to Alwadi et al.(2017), RFID technology is easily scalable for use with general development of algorithms in smart inventory systems. The electromagnetic

field is implemented in the RFID and RFID Reader to track the RFID tag which store the product information electronically as shown in Figure 2.7.

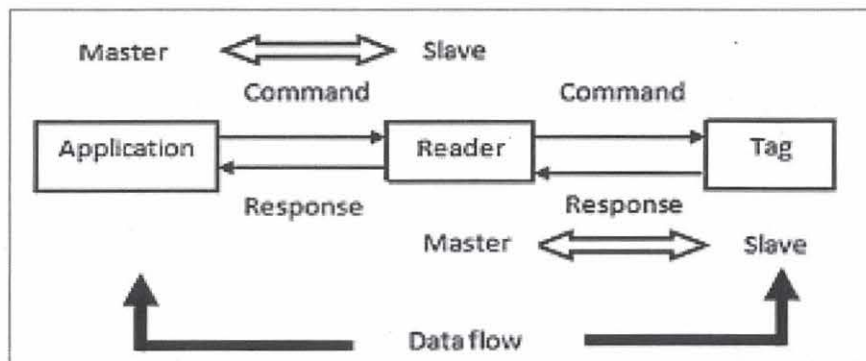


Figure 2.7 The components of RFID system (Tejesh & Neeraja, 2018)

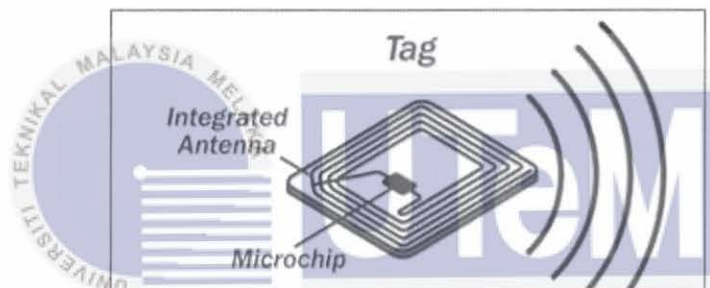


Figure 2.8 RFID tag (Adcbarcode,2017)

The Figure 2.8 shows the antenna attached to the microchip plays a role in transmitting information from chip to reader via radio waves. There is two type of RFID tags which are active and passive. Passive RFID sensor tag does not contain battery so it needs to activate by collecting the energy that interrogated from the radio waves of RFID reader. (Byondi & Chung, 2019) Active RFID sensors have onboard power supplies (such as batteries) to power integrated circuit (IC) of sensors. The IC carries out all the communication between the tag and the reader as Figure 2.8 shown. Battery and IC in these types of tags provide high data capacity handling and a wide range of operations at the expense of high complexity. (Tejesh & Neeraja, 2018)

In a metal environment, a special tag is used instead of normal tags due to some of the factors.(Valente & Neto, 2017) Firstly, the metal will detune the tag antennas and make it difficult to be read. Second, it is the bounced off of the radio waves from metal that will

led to incorrect of reading data from others unwanted tags. For the steel that freshly come with high temperature, the tag should be made to withstand heat.

Table 2.4 Pros and Cons of RFID

Concerns	Pros	Cons
Cost concern	Low operating costs as RFID tags can be reuse.	High investment cost needed as it need new Infrastructure and equipment.
Implementation concern	<ul style="list-style-type: none"> <li>• Less labour needed for replace manual keeping track work into automated provide real-time data.</li> <li>• Less error occurs as it is a automated system with high accuracy of read rate.</li> <li>• High Efficiency with the high accuracy given in real-time and user can monitor all inventory details that keep in RFID code.</li> </ul>	<ul style="list-style-type: none"> <li>• Difficulties in getting the information in metal or liquid environment.</li> <li>• Collision happens between tags when respond at a same time.</li> <li>• Collision happens between reader when the signal is overlap and tag unable to respond.</li> <li>• Smaller tag will be hidden or missing.</li> </ul>
Security concern	Secure than Bar code	DoS Attack against RFID Reader. Encryption of information and authentication of RFID

#### 2.2.1.4 Conclusion

The uses and working principle of sensor type is studied. The use of RFID is more suitable than computer vision and barcode. In cost, the barcode has a lower price for implementation than RFID. However, the overhead cost for barcodes is higher due to the higher load needed to be scanned manually. (Valente & Neto, 2017) And by using a manual spreadsheet map, the operator will waste time finding the location of the inventory and find it hard to keep track back on the stock when the barcode gets blurred.

According to Alfian et al. (2017), the bar code system has been replaced by RFID technology to identify the inventory wirelessly without limited distance. It can be utilized in tracking the item and controlling the stock as it is low cost by offering easy integration into any type of application. (Tejesh & Neeraja, 2018) Based on the studies, it was decided to go for RFID technology as the sensor in tracking the inventory data.



## 2.2.2 IoT Gateway

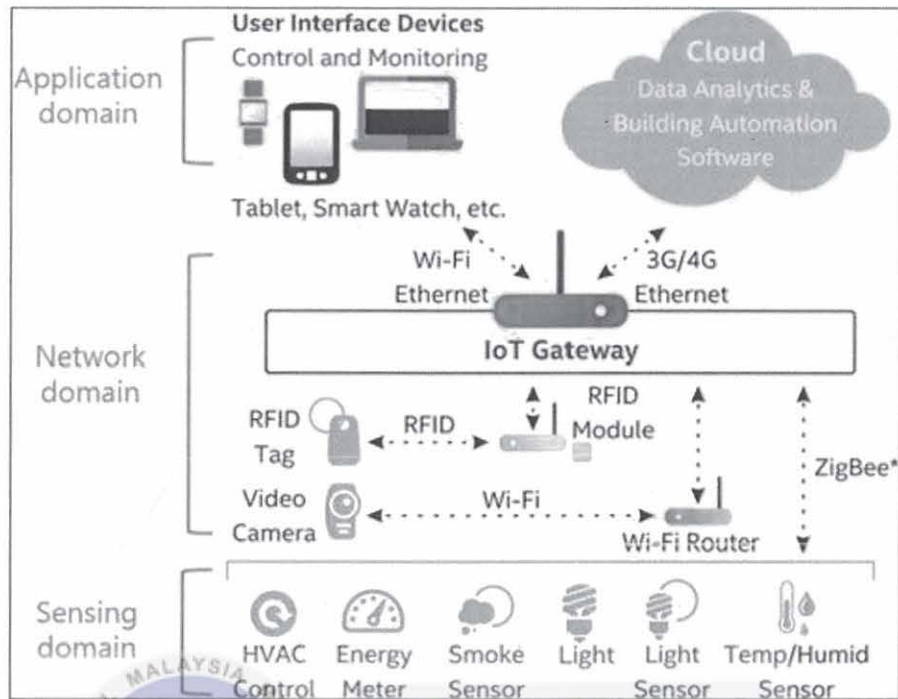


Figure 2.9 General feature of IoT Gateway (Kang et al., 2017)

The IoT gateway acts as a communication protocol that enables massive sensors and machine can be connected to transmit data or message illustrated in Figure 2.9. They integrate networking protocols for managing storage, edge analytics, and data flow in a secure and reliable manner between the devices and the cloud. (Kang et al., 2017)

### 2.2.2.1 NodeMCU

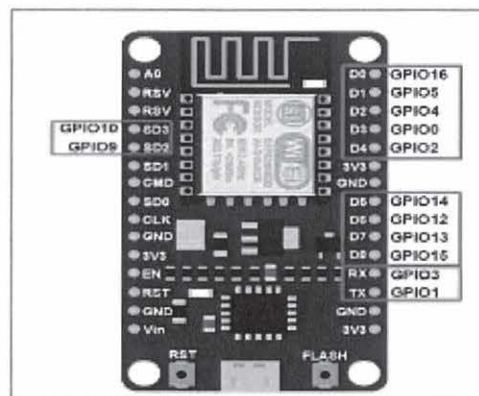


Figure 2.10 NodeMCU (Paveesha et al., 2020)

The Wi-Fi communication in an IoT based project is processing by using NodeMCU and it can be said as IoT gateway router. It can support about 2.4GHz Network connection and the power required is 3.3V DC. (Durani et al., 2018) NodeMCU can use in measuring and storing data of the inventory with an open-source platform. It also can act as a microcontroller to upload a program into MCU in order to control the GPIO ports. The CP2102 (works up to Windows-10) IC on NodeMCU that offer USB to TTL functionality. The NodeMCU is programmed by using the Arduino software that installed in certain package. (Paveesha et al., 2020)

### 2.2.2.2 Arduino

Arduino is an open hardware development board that can be used to interact with electronic components. While there are many types of Arduino boards, generally it contains digital I/O pins and analog pins, a power connector, a microcontroller and a serial connector.



Figure 2.11 Arduino Uno Board and L293D Motor Driver Shield (Pololu Corporation, Open Circuit)

Arduino has the pins that arranged in a specific pattern. The reason is to fit the compatible shield like L293D motor driver shield etc. The digital I/O pins can read and write a single state signal whereas the analog pins can read a range of values for a finer control. The power connector provides voltage to the board itself and some low voltage components like LED. The source can be either AC or DC voltage. Besides, the microcontroller is the primary chip of the Arduino which executes the program written. The chip varies depending on the type of Arduino boards. Moreover, the serial connector allows the communication of the board to computer. It can be used to load new programs onto the

device. On top of that, it also replaces the power connector in some models in the newer versions. (Open Source)

### 2.2.2.3 Raspberry Pi

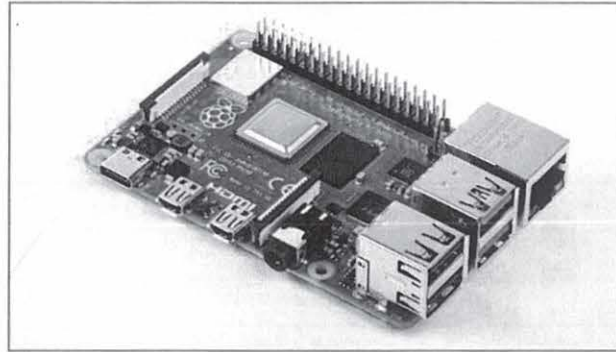


Figure 2.12 Raspberry Pi (Seth Kenlon,2021)

Raspberry Pi is known as a single-board computers. The objective of this association is to educate and create easier access to computing education towards the public. The series is launched in 2012 and constantly updating newer versions. The Raspberry Pi runs on Linux and Pi OS as its operating system. Besides that, it comes with a set of GPIO (General Purpose Input/Output) pins which allows the control of electronic components for physical computing. (Open Source) It based portable and can work with various machine learning models or system. Moreover, it can extend the previous setup by implementing new system. (Dhillon et al., 2021)

In warehouse management system, Raspberry Pi 3 is a single board computer (SBC) that enables quick prototyping of Software that keeps track of details of goods in a database. Raspberry Pi 3 can also be programmed to act as a web server. A common programming language for Raspberry Pi 3 development is Python. Raspberry Pi 3 can be powered by 5 V power supply and operates at 3.3V logic level for its General-Purpose Input/Output (GPIO) pins. Raspbian is that the Linux-based software package that is running on Raspberry Pi 3. (Tejesh & Neeraja, 2018)

In MATLAB, the raspberry pi support package offers two type of programming which are Interactive communication and Standalone execution. The Interactive



communication is acquiring the data from sensors and analyzing or visualizing it in MATLAB. While the standalone performance creates a MATLAB algorithm in generating a C code to deploy into Raspberry Pi as a standalone application. The Raspberry Pi models supported by MATLAB are limited, summarized into the Table 2.5.

Table 2.5 Different types of Raspberry Pi with either support or no support MATLAB

Raspberry Pi Model	MATLAB Releases Supported	Supported in MATLAB Online?
Raspberry Pi 1 Model B (discontinued)	R2014a - Current	No
<u>Raspberry Pi 1 Model B+</u>	R2014b - Current	No
<u>Raspberry Pi 2 Model B</u>	R2014b - Current	Yes
<u>Raspberry Pi 3 Model B</u>	R2016a - Current	Yes
<u>Raspberry Pi 3 Model B+</u>	R2018b - Current	Yes
<u>Raspberry Pi 4 Model B</u>	R2020a - Current	Yes
Raspberry Pi 1 Model A,	No support	No support

#### 2.2.2.4 Conclusion

In conclusion, the overall features and criteria of Raspberry Pi, Arduino and Node MCU is studied to determine the best component that suit in the project. As it is more flexibility as it can run on Pi OS. Furthermore, Raspberry Pi can handle variety of changes and support for the further development.

#### 2.2.3 Network Communication

Machine to Machine (M2M) is a communication system between machineries that is widely used in the industry. Different M2M suppliers each have their own proprietary Software and hardware that are not cross-compatible. In addition, there are various challenges in implementing M2M systems: security and performance issues such as network congestion, limited bandwidth, and transmission cost. (Ouaissa & Rhattoy, 2019)

### 2.2.3.1 Bluetooth



Figure 2.13 ZF & Bluetooth® Low Energy (ZF Friedrichshafen AG )

Bluetooth offers a short-range wireless technology that is designed to replace wired technology. It is an established technology that is currently embedded in almost all electronic devices. Through Bluetooth technology, IoT architecture can be empowered with high range and transmission speed to be adapted in asset tracking applications. (Zeadally et al., 2019) Bluetooth Low Energy (BLE) is introduced as an update to the Bluetooth technology that uses a low amount of energy, which further enhances compatibility with IoT and M2M applications. (Octaviani & Ce, 2020) Some security threats of Bluetooth technology include Pin cracking, MAC spoofing, Man-In-The-Middle (MITM) Attack. MITM attack is an exploit that interrupts the connection during Secure Simple Pairing (SSP) to steal information from the connected device. (Zeadally et al., 2019)

### 2.2.3.2 Wi-Fi



Figure 2.14 Wi-Fi (Autodesk)



Wi-Fi is a low power wireless network technology with many advantages such as high data transmission rate, mobility, inbuilt IP-network compatibility, and simple integration with existing infrastructure. Basically, Wi-Fi embeds in IoT applications where high communication bandwidth is required. (Yüksel, 2020) Wi-Fi can work in 2 different bands which are UHF band with 2.4GHz (100MHz) and SHF band with 5GHz (150MHz). For a gadget to connect with Wi-Fi it needs to embed a Wi-Fi module on a microcontroller or a wireless adapter that suits the input adapter of the machine or other interface.

Mobile phones are pretty helpful nowadays, the mobile phone communication technologies such as 2G, 3G, and 4G are recommended to provide logistics management services. Despite the fact that 5G technologies are still in development and have been proposed to give high-speed communications and decrease the power in response. (Di, 2020) It is widely used in massive M2M communication and IoT, especially in an intelligent sensor environment. (Alonso-Zarate & Dohler, 2016) There is some threat when Wi-Fi hotspot is activated, such as sniffing, DNS Spoofing and Hijacking. (Susila et al., 2017)

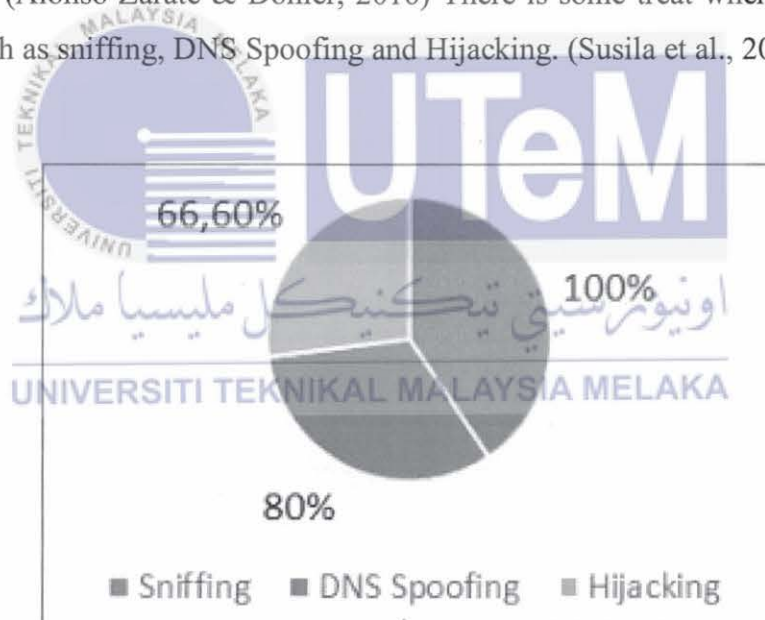


Figure 2.15 Percentage of successful activity of the attacks (Susila et al., 2017)

### 2.2.3.3 Bluetooth vs Wi-Fi

By comparing the characteristics of Bluetooth and Wi-Fi, it can clearly define which is the suitable network communication in this project.

#

Table 2.6 Bluetooth vs Wi-Fi compare between (Autodesk)

Characteristics	Bluetooth	Wi-Fi
Frequency	2.4GHz	2.4GHz or 5GHz
Cost	Low	High
Power Consumption	Low	High
Effective Range	30 m	100m
Data transfer	25Mbps (Bluetooth 4.0)	250Mbps
Connection	Up to 7 devices only can be connected.	Based on the bandwidth of the router that is connected.

### 2.2.3.4 Conclusion

Through the study, Wi-Fi has been chosen to give the connectivity between the raspberry Pi and mobile phone. It has a wider range that can cover the monitoring area, higher speed transmission and can be connected by many users.

### 2.2.3.5 Communication Protocol

Communication protocol is a system of rules authorizing two or more communications system entities to transmit information while ensuring Quality of Service (QoS). Typically, communication protocol can be defined as a data transmission asynchronously or synchronous from one device to another device through a same network. (Gorry Fairhurst, 2020) There is several type of communication protocols in IoT have to be studied to suit different layers, (Maha Kavya Sri et al., 2019) the rate of transmission data and the respond time. (Jaloudi, 2019)

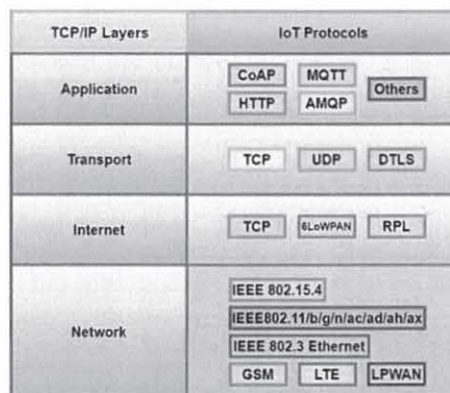


Figure 2.16 Communication Protocols in different layer (B. Mishra & Kertesz, 2020)

### 2.2.3.6 HTTP

The Hypertext Transfer Protocol (HTTP) is an protocol in application level to exchange information through the internet between client and server. (Mathaba et al., 2017) The clients in this protocol are the one who send the request while servers are the one that provide service. There is a specific language used between the clients and servers. (Kumar, 2019) The HTTP connect via TCP/IP to deliver a massive of tiny package (Yokotani & Sasaki, 2017) that can be the form of hypertext, pictures, media player etc. (MDN contributors, 2021) Nowadays, there have the combination of HTTP and REST, HTTP with JSON. The client can use POST, GET, PUT and DELETE to perform the operation.(Dizdarević et al., 2019)

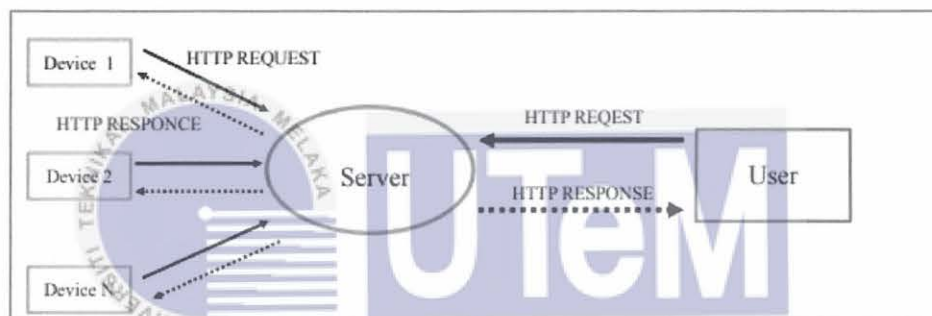


Figure 2.17 Architectural of HTTP (Yokotani & Sasaki, 2017)

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

### 2.2.3.7 CoAP

Constrained application protocol (CoAP) is designed to serve the devices by limiting the devices' capability in processing. It required a low power consumption to exchange data through a simple network, UDP. CoAP can communicate in one-to-one to exchange the information between the client and server. It is alike to the HTTP, which both of the systems are 'request and response'. While the difference of CoAP and HTTP is CoAP is depending on the second level of the structure (message layer) that use to resending the missing package. Besides, it had added in a function for the clients to receive the information when changing in resource. This makes the CoAP tend to be 'publish and subscribe' pattern. (Dizdarević et al., 2019)



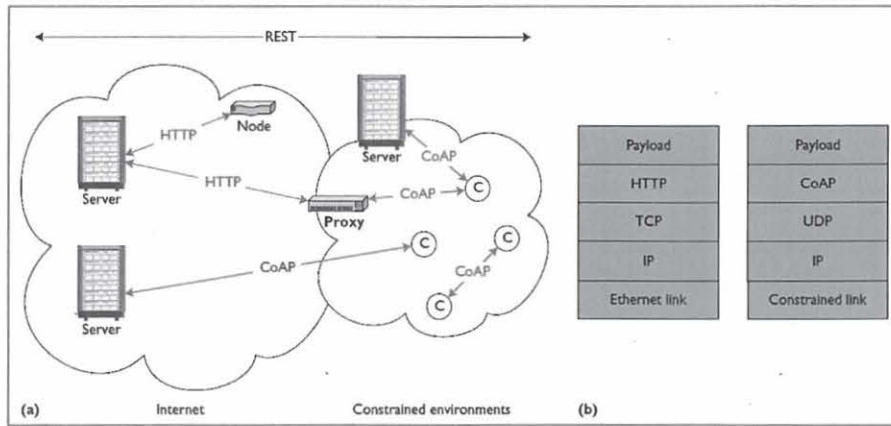


Figure 2.18 Comparison between architecture of CoAP and HTTP (Devopedia, 2019)

### 2.2.3.8 MQTT

The Message Queuing Telemetry Transport (MQTT) was releasing in 1999 and it has modified into many versions throughout the years and the latest is MQTT v5 which got OASIS standard in 2019 by upgrading its performance in terms of simplicity to implement. (B. Mishra & Kertesz, 2020) OASIS is a Non-profit organization (NPO) that approves standards for advanced projects.

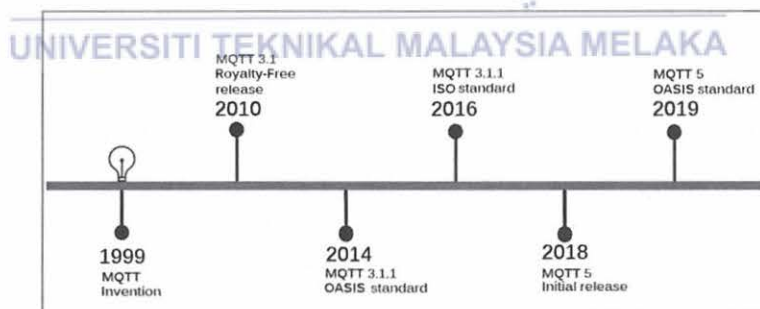


Figure 2.19 The MQTT development over time (B. Mishra & Kertesz, 2020)

MQTT is a lightweight and low-cost protocol to exchange data via MQTT server. (D. Mishra et al., 2019) It has low bandwidth with high latency that can ensure reliability in delivering of large data between machine and machine. (Mishra Devesh et al., 2019; Pham & Hoang, 2021) The sensor can connect between each other through the satellite, SCADA etc. In MQTT broker, it will process the information collected by the publisher (MQTT



Client) and send it to the subscriber (MQTT Client) in the topic generated which also called data block. (Laxmi & Mishra, 2018; Yokotani & Sasaki, 2017))

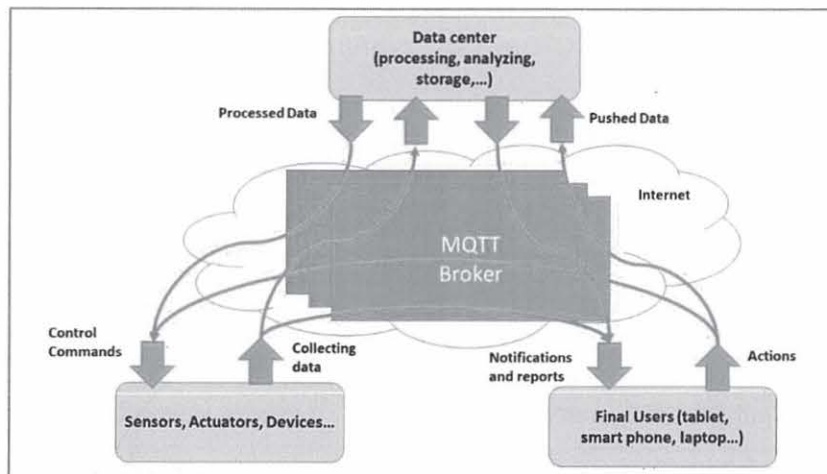


Figure 2.20 Architecture of MQTT Protocol (Pham & Hoang, 2021)

MQTT broker can communicate in many-to-many that allow can exchange data between several clients by several publisher (Valente & Neto, 2017) and also in different type of communication. (Jaloudi, 2019) It is connected via a TCP which can transmit the data continuously with lightweight overhead. There are 3 types of Quality of Service (QoS) can be offered by the MQTT which shown in the Table 2.5.

Table 2.7 Type of QoS (B. Mishra & Kertesz, 2020; Pham & Hoang, 2021)

Quality of Service (QoS)	Description
Maximum of one (QoS 0)	Deliver live data to the reader just in once and without acknowledged. This will cause some missing message happens.
Minimum of one (QoS 1)	Deliver the data is acknowledged to reach publisher. This will cause duplication of the message happens. It can send the message in a shorter time than QoS 2.
Only One (QoS 2)	Deliver all data once without missing any data, even offline, but it will cause data overhead.

It can run on a computer or be hosted in the cloud supported by a different type of MQTT Broker which serve in Clouds based, sharing network, and own hosted network. The Cloud-based MQTT broker provides global access via the Internet. (Laxmi & Mishra, 2018) Next, the MQTT have high security due to the MQTT brokers need to sign in with username and password authentication from clients to connect. (Kayal & Perros, 2017) The

information that the client used the system would not be published. (Rebecca Deprey, 2020)  
 Overall, the advantages of MQTT can be summarized into:

Table 2.8 Advantages of MQTT

Advantages of MQTT	References
Competently allocation of data	Mishra Devesh et al., 2019
Growth in scalability.	
Ease of bandwidth depletion	
Update information within seconds.	
Suitable in remote sensing.	
Exceptionally lightweight overhead.	Kayal & Perros, 2017; Mishra Devesh et al., 2019
High Secure	Kayal & Perros, 2017; Rebecca Deprey,2020

Nowadays, there is a lot of industry using MQTT as their communication protocol. One of the examples using MQTT in tracking the goods is shown as Figure 2.20.

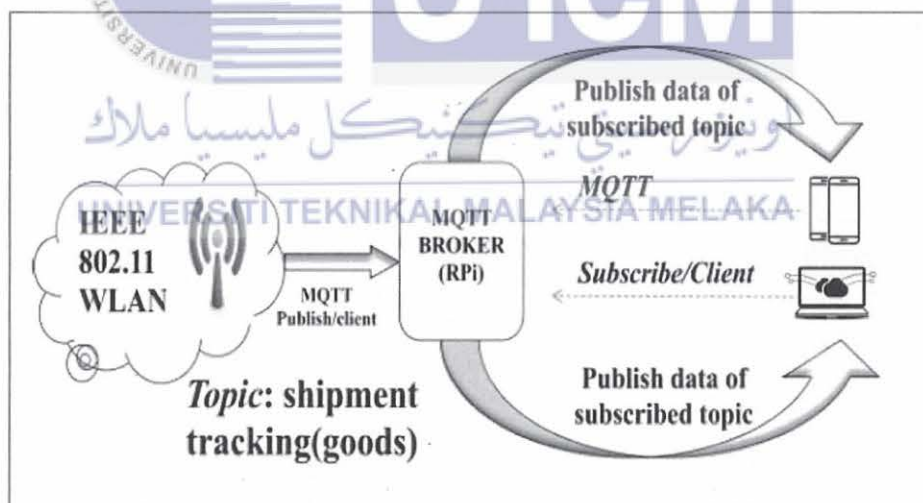


Figure 2.21 MQTT Protocol on tracking shipment

### 2.2.3.9 MQTT vs Other Communication Protocol

HTTP and CoAP has a similar pattern which is request and respond/reply that can see the relationship clearly as shown in Figure2.22. While, the Publish-Subscribe model used

by MQTT is shown in Figure 2.23. The overall comparison of HTTP, CoAP and MQTT by Jaloudi (2019) is shown in the Table 2.8.

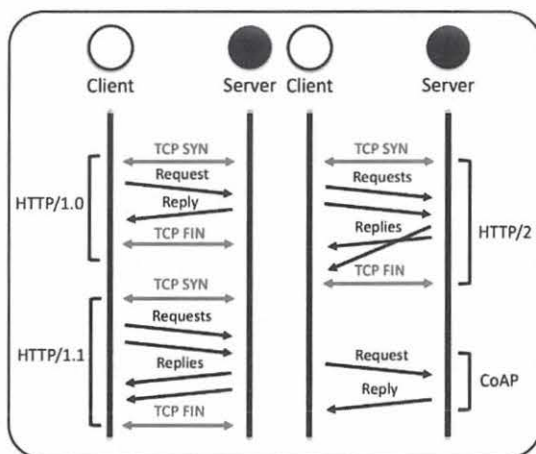


Figure 2.22 Request and Respond model used by HTTP and CoAP

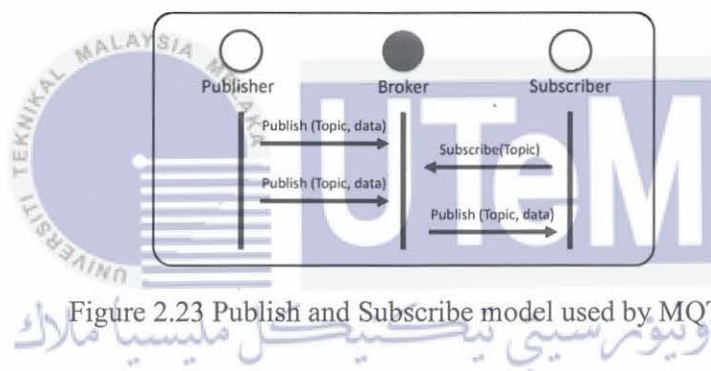


Figure 2.23 Publish and Subscribe model used by MQTT

Table 2.9 MQTT compared with others Communication Protocol (Jaloudi, 2019)

Features	HTTP	CoAP	MQTT
Infrastructure	Ethernet, Wi-Fi	6LoWPAN	Ethernet, Wi-Fi
Transport Layer	TCP	UDP	TCP/IP
Model	Synchronous	Asynchronous	Asynchronous
Mechanism	one-to-one	one-to-one	one-to-many
Pattern	Request-Request	Both	Publish-Subscribe
Methodology	Document-oriented	Document-oriented	Message-oriented
Security	SSL, TLS	DTLS	SSL, TLS

From the studied of Dizdarević et al.(2019), it also stated that ‘publish and subscribe’ model can give more advantages:

1. High security with client’s information will not be published.
2. Get data from many resource (one-to-many /many-to many)
3. Exchange data can be done offline, as it has MQTT broker to store data)



### 2.2.3.10 Conclusion

The MQTT which is a light-weight protocol is preferred in this prototype. Besides, using a Cloud MQTT broker can easily be accessed by the internet globally. In addition, the study of Yokotani & Sasaki, 2017 in comparing the bandwidth required in HTTP and MQTT shows that the MQTT lower bandwidth than HTTP which can minimize the delays in transmission. Although, MQTT has the same transport layer which use TCP/IP offers longer connection to a broker is better than UDP. Even though the HTTP has used the same TCP as MQTT but it only needs to operate in synchronous and it only can communicate in one-to-one. So, MQTT which is a one-to-many protocol is selected to reach the objective we need to connect to different users.

## 2.2.4 Cloud Service Application

### 2.2.4.1 MIT App Inventor

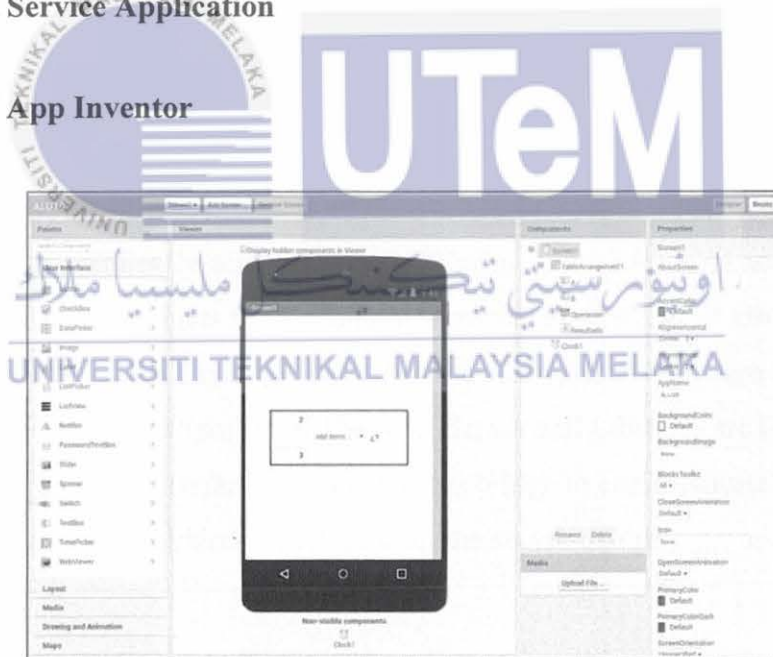


Figure 2.24 Graphical interface editor of MIT App Inventor (Mir & Lluca, 2020)

From Mir & Lluca (2020), the MIT App Inventor is a visual programming environment featuring that enables developers to create fully functional apps on mobile and tablet that it only support on Android and iOS. Those apps able to link with the sensor in mobile phone and analyze the data by providing user a visualize on the application's graphical user interface and its code using Block Editor. The application is compiled into



debug gable APK file once the design finished. (Adiono et al., 2019)The MIT App Inventor can work well with JSON data in a project. (Mir & Lluca, 2020)

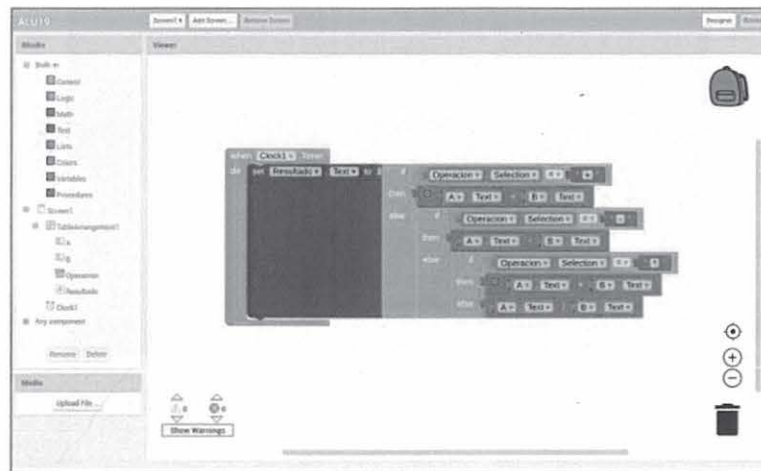


Figure 2.25 Code editor of MIT App Inventor (Mir & Lluca, 2020)

#### 2.2.4.2 Blynk

Blynk is a web platform that enables users to control various electronic devices remotely. It works seamlessly across various platforms such as Android and iOS by creating a layout using different widgets such as button shown in Figure 2.27. It also supports storing and displaying sensor data. Blynk provides libraries for various hardware problems such as Arduino, Raspberry Pi, and SparkFun. The App, Server and Libraries are the components of the Blynk. App provides interface, sever act as the bridge to communicate between app and hardwires and libraries provide interaction with the server using commands. (Durani et al., 2018)



Figure 2.26 Switch ON button in Blynk app

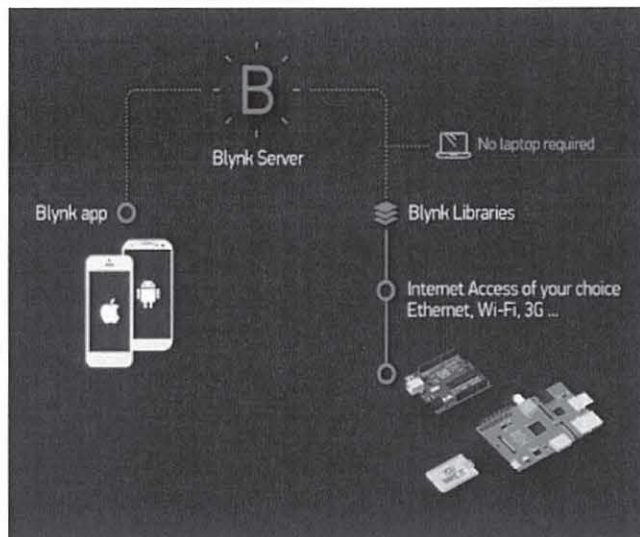


Figure 2.27 Schematic of the Blynk's component

### 2.2.4.3 ThingSpeak

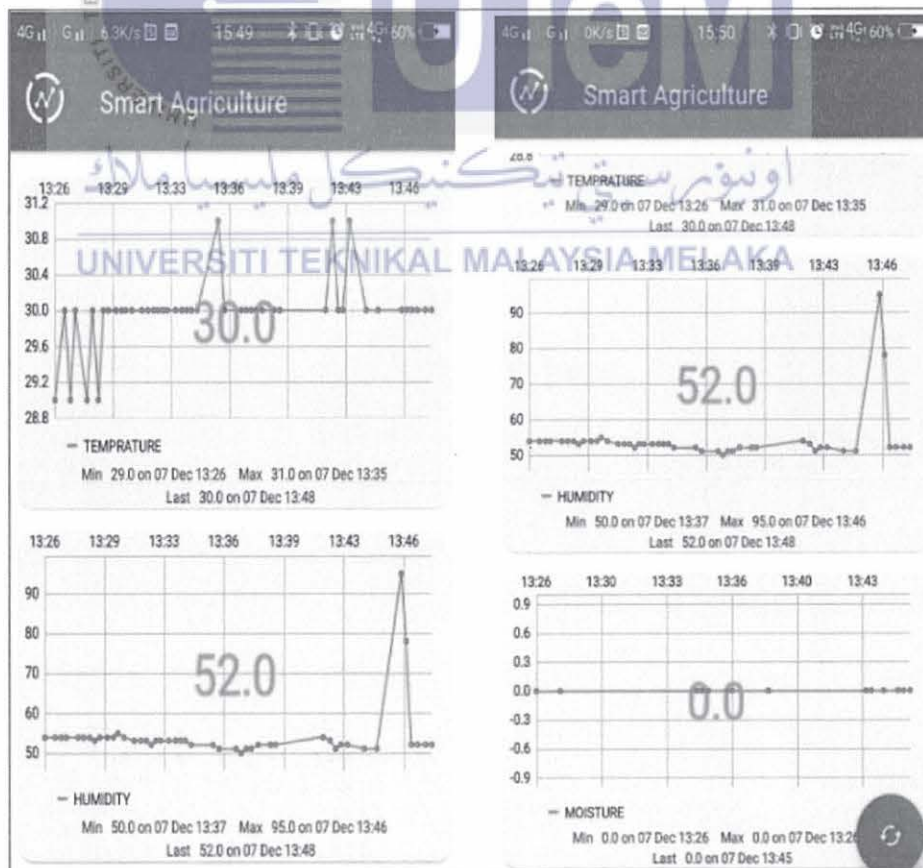


Figure 2.28 Sensor output on ThingView Free APP (Tapakire & Patil, 2019)

ThingSpeak is IoT platform developed by MathWorks, the developer of MATLAB. It allows the user to store and collect any data into their cloud and server. (Et.al, 2021) It offers visualization and analysis of the real-time data that can also send alerts via web service such as Twitter® and Twilio®. The analysis in ThingSpeak is done by MATLAB® analytics with the coding for the processing, visualizations, and analyses. (ThingSpeak, 2020) The data from the sensor can be viewed on the mobile phones by using ThingView Free application in the play store as shown in Figure X.

#### 2.2.4.4 Conclusion

ThingSpeak is chosen as it has a free server and app that can implement in the system with various of the gateway. Besides, it offers visualization and analysis of the real-time data. It can be viewed neither on PC or mobile phone through ThingView application.





## 2.3 Relevant Works

Table 2.10 List of research that related with IoT Warehouse Inventory System

Author and year	Research title	Method used	Result obtained
Saha D, Agarwal P (2019)	Warehouse Management Using Real-Time QR-Code and Text Detection	<ul style="list-style-type: none"> <li>QR code and numerical code.</li> <li>Open-source computer vision</li> </ul>	<ul style="list-style-type: none"> <li>Able to perform text detection using Tesseract OCR.</li> <li>Save cost by using paper for the label.</li> </ul>
Bing & Yang, (2019)	Design and development of inventory system based on barcode scanning technology	<ul style="list-style-type: none"> <li>Barcode with Digital image processing technology</li> </ul>	Able to increase the scanning speed and accuracy of the barcode recognition.
Putra Yudha et al. (2018)	Applications of goods inventory system that uses barcode scanner on Android	<ul style="list-style-type: none"> <li>Sensor type: Barcode</li> <li>Android</li> <li>phone reader</li> </ul>	<ul style="list-style-type: none"> <li>Able to get information and status of the inventory.</li> <li>Able to exchange data using phone by connecting to the sever.</li> </ul>
Alfian et al. (2017)	Application of RFID and Computer Vision for the Inventory Management System	<ul style="list-style-type: none"> <li>Computer Vision and RFID</li> <li>IP Camera</li> <li>PC core i3</li> </ul>	<ul style="list-style-type: none"> <li>Able to avoid misreading and ghost reading problem by combination of 2 sensors.</li> </ul>
Lakshmi Narayan, S. P., Kavinkartik, E., & Prabhu, E. (2019).	IoT based food inventory tracking system. In Communications in Computer and Information Science	<ul style="list-style-type: none"> <li>Load cell</li> <li>Arduino</li> <li>NodeMCU</li> <li>MQTT protocol</li> <li>Android</li> </ul>	<ul style="list-style-type: none"> <li>Able to track real time and history inventory through windows and Android application.</li> <li>Able to develop analytic tool for analyzing and predict the usage/consumption patterns.</li> </ul>
Laxmi & Mishra (2018)	RFID based Logistic Management System using Internet of Things (IoT)	<ul style="list-style-type: none"> <li>RFID RC522 module</li> <li>Raspberry Pi</li> <li>Cloud MQTT broker</li> <li>MQTT Subscriber supported by GUI</li> </ul>	<ul style="list-style-type: none"> <li>Able to detect shipment at different warehouse locations.</li> <li>Able to develop data analytic tool.</li> <li>Able to implement an user friendly MQTT broker with the Cloud-based MQTT broker.</li> </ul>
Valente & Neto (2017)	Intelligent Steel Inventory Tracking with IoT / RFID	<ul style="list-style-type: none"> <li>RFID</li> <li>MQTT</li> <li>iBeacon</li> </ul>	<ul style="list-style-type: none"> <li>Able to come out a special RFID tag for metal product</li> </ul>
Mathaba, S., Adigun, M., Oladosu, J., & Oki, O. (2017).	On the use of the Internet of Things and Web 2.0 in inventory management.	<ul style="list-style-type: none"> <li>Sensor type: RFID module</li> <li>Arduino</li> <li>HTTP</li> <li>Beachcomber</li> <li>Twitter</li> </ul>	<ul style="list-style-type: none"> <li>Able to detect misplaced stock and low stock levels on shelves</li> <li>Able to received notification on real time inventory status.</li> </ul>
Gopi Krishna et al., 2018	Design and development of bi directional IoT gateway using ZigBee and Wi Fi technologies with MQTT p rotocol	<ul style="list-style-type: none"> <li>Raspberry Pi</li> <li>NodeMCU</li> <li>Mosquito – MQTT broker</li> </ul>	<ul style="list-style-type: none"> <li>Able to provide inventory data</li> <li>Able to analyze data.</li> </ul>



## 2.4 Summary

The basic components and technologies have been studied on their characteristics, performance advantages and carry out comparison to support the decision making.

To sum up, the RFID is chosen as the IoT sensor as its advantages offered which can save time, lower overhead cost and it can perform without limited by wired connection are suitable to implement in this project.

Next, the Raspberry Pi is chosen as it is a microprocessor that have high flexibility that it could extend the capability to enhance the connection to future development such as connect with other sensor to give other features. Besides, by using Raspberry Pi, engineer do no need to depends on the factory's PC to develop or upgrade the system. They just need to setup up a small-scaled experiment at home with their ow PC.

In additional, to own a wider range of monitoring, Wi-Fi is chosen instead of Bluetooth as the network connection in this prototype. It can be easily to set up with the wireless Lan built in in Raspberry Pi. Wi-Fi also can provide more people to connect on to a same sever. In the studies on various type of communication protocol, the MQTT is chosen as rather than the benefits in fast transmission, the pattern of one-to-many etc. It is also well-known in IoT application as its mature technology and high standard controlled by OASIS.

Last but not least, ThingSpeak as a chosen as a Cloud Service Application that act as Cloud Based MQTT broker to connect with the Raspberry Pi and Mobile device. It is a free platform that suitable in the small scaled project in this prototype.

# CHAPTER 3

## METHODOLOGY

Chapter 3 discusses the methodology of this project in extend the previous project by implementing the MQTT protocol and to evaluate the overall system by different test.

### 3.0 Project Development Model

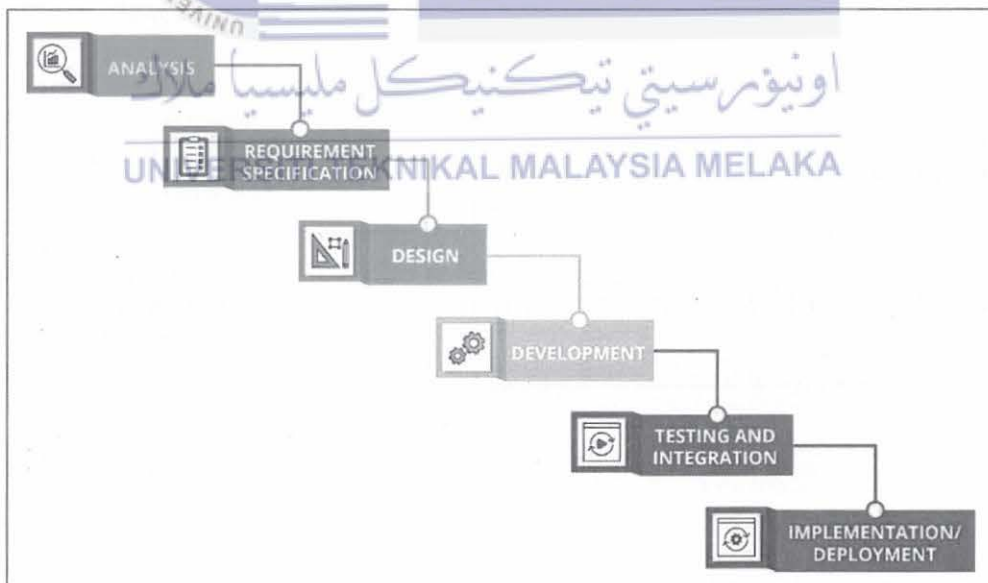


Figure 3.1 Waterfall Model

The Waterfall model helps visualize and explain the various steps in the project that shown in Figure3.1 with the sequence of the steps and their description. The analysis stages which is literature review helps in identifying the suitable methods, parameter and components for the project. In this project, it can be differed into two main applications which are Raspberry

Pi and ThingSpeak. The test and analysis will be conducted for further integration and look for potential implementation development.

### 3.1 Flow Chart of General Project development

This Flow chart is developed based on the flow of Waterfall model to achieve the objectives.

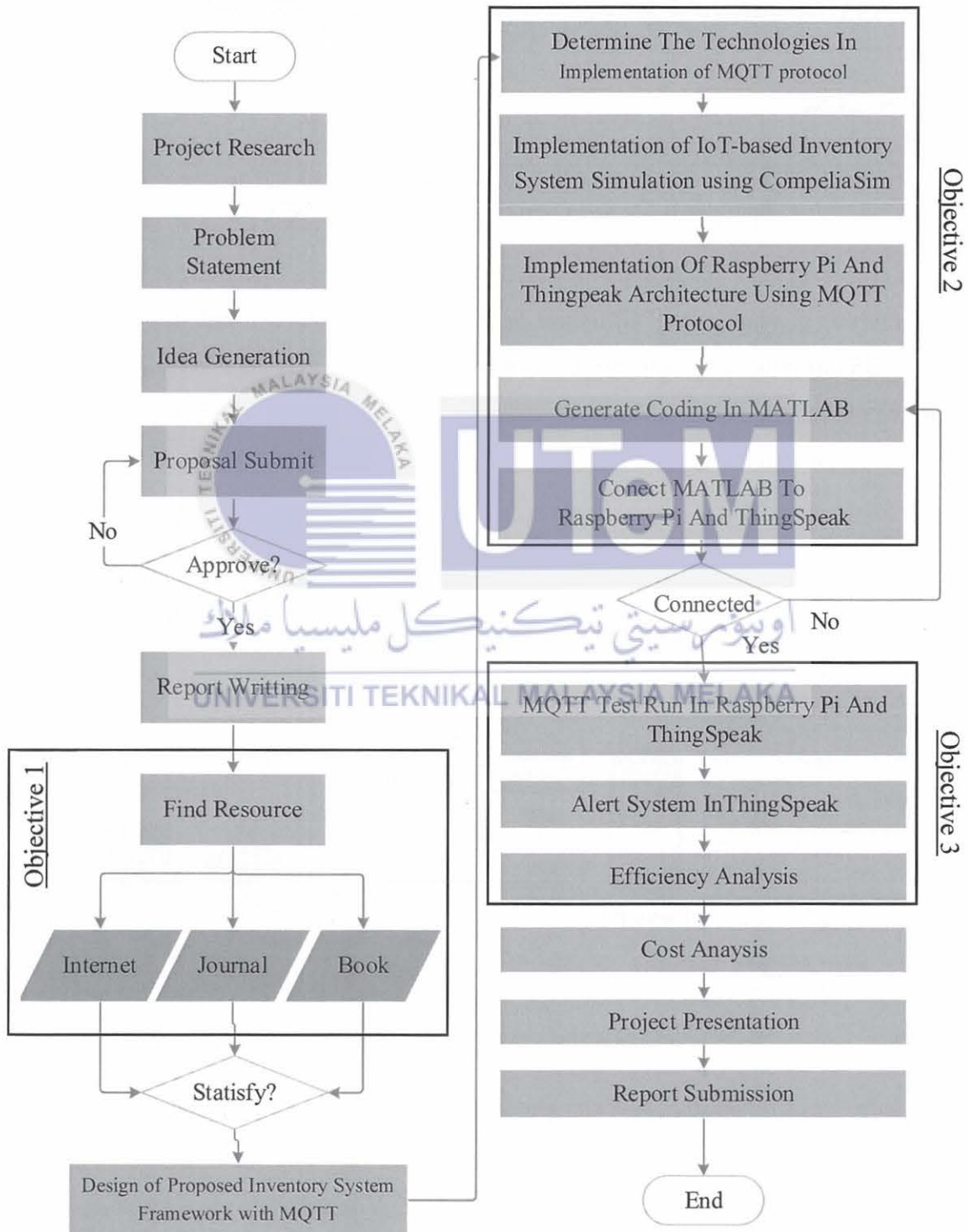


Figure 3.2 Flow Chart of the project overview

### 3.2 Design of Proposed Inventory System Framework with MQTT

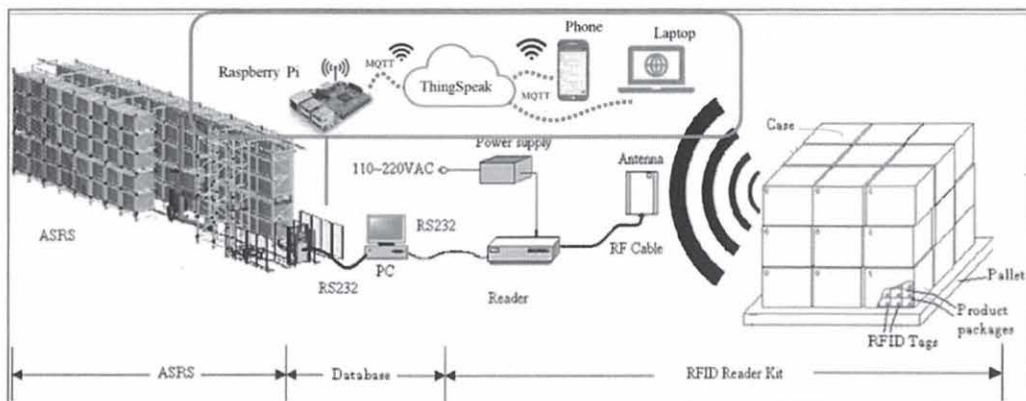


Figure 3.3 Hardware inventory system framework

The framework of the system is illustrated in Figure 3.3. This project is a from the previous project that had already have built in the RFID Technology in retrieving storage automatically. The MQTT Protocol Architectural will be plugged in after the PC. A USB 3.0 Male to Male USB Cable is connected to the Raspberry Pi's USB port. Set up the Wi-Fi on Raspberry Pi to connect to the ThingSpeak. The user can monitor the data in ThingSpeak and get notified from ThingSpeak.

### 3.3 MQTT Protocol Architectural

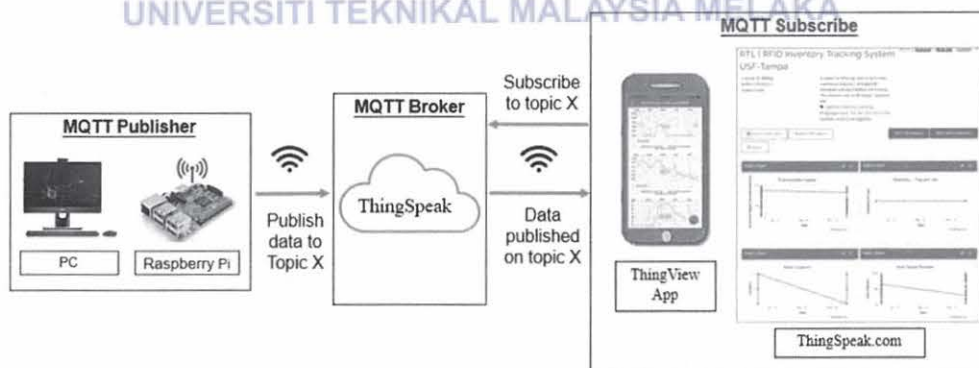


Figure 3.4 MQTT Protocol architectural

The overall system is a publish-subscribe method is shown in Figure 3.4. The Raspberry Pi will publish the data from the PC to the topics in ThingSpeak and to the subscribing device such as mobile device (ThingView APP) and PC (ThingSpeak.com). ThingSpeak can generate analytic results from the data collected and send notifications to the user when they meet the conditions.



### 3.4 Requirement Specification

However, in this project, the framework of the inventory system in the lab does not been carried out due to some of the reasons. First, the machine of this system has not been used for a long time, so it needs to be reset or maybe replaced with new components. Second, it will use up quite a long time that will exceed the timeframe in a semester as the lab can only be accessed after November. Third, since the Covid-19 pandemic hits, people have started to adapt to the Work from Home culture by getting the jobs done at home. The culture has brought up the software-based to carry on the research and development of the company in testing out the new system such as IoT things.

Hence, a software-based experiment using MATLAB acts as a vital program that assists the engineer in performing the design of Software using mathematical calculation, then carrying out analysis in generating graphs to evaluate. Figure 3.5 shows that MATLAB is used as a simulation tool that can publish the data to Raspberry Pi and ThingSpeak via MQTT by designing a set of coding that is similar to the real-time scenario.

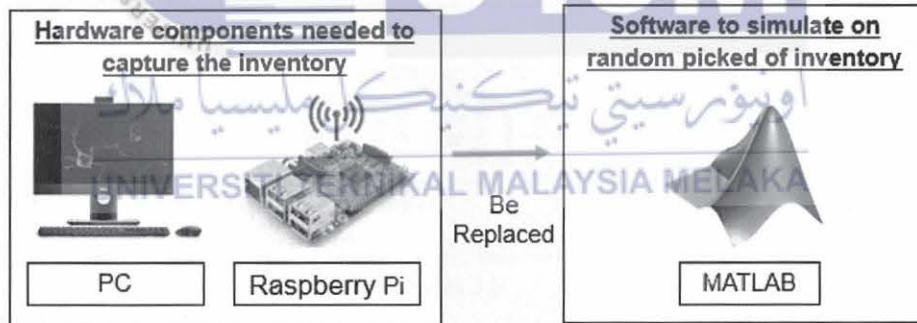


Figure 3.5 Implementation of MATLAB in replacing the experimental in lab.

Therefore, the prototype will focus on functional and non-functional requirements shown in Table 3.1 to ensure the data can be updated correctly and instantly by using MQTT protocol. Besides, it can also give alertness to the clients.






Table 3.1 Requirement specification of the project



Functional	Non-Functional
Automated publish data to the clients and users	Accuracy
Automated Update inventory level	Effectiveness
Automated notify the clients and users.	Alertness

### 3.5 Technologies in Implementation of MQTT protocol

This section will brief on the technologies required in different layers to build up the warehouse inventory connection using MQTT shown in Table 3.2 which consists of 4 layers with either software or hardware.

Table 3.2 Technologies in implementation of MQTT protocol

Level	Description	Components	Function
1	Controllers	MATLAB 	<ul style="list-style-type: none"> <li>It is used to write a code set to stimulate the MQTT protocol in publishing data to Raspberry Pi and ThingSpeak.</li> </ul>
2	Connectivity	Gigabit Ethernet Cable 5 meters 5 Meter 	<ul style="list-style-type: none"> <li>It is used to connect the router with Raspberry Pi to receive the WIFI.</li> </ul>
		Wi-Fi 	<ul style="list-style-type: none"> <li>It is used to connect the Matlab to ThingSpeak</li> </ul>
3	Data Accumulation	Raspberry Pi 3 B 	<ul style="list-style-type: none"> <li>It can be connected to the WIFI using it a special IP address.</li> <li>It is the version that the MATLAB can support.</li> <li>It is a platform to receive data from the publisher.</li> </ul>
		Cloud Service – Thingspeak 	<ul style="list-style-type: none"> <li>It is a platform for the users to monitor the data.</li> <li>It offers visualization and analysis of the real-time data</li> <li>It can also send alerts vis e-mail to the user on the inventory level.</li> <li>Every 15 seconds of updating time.</li> </ul>

		<p>MATLAB Analysis</p> 	<ul style="list-style-type: none"> <li>• It is used to write a set of MATLAB code to notify the user from ThingSpeak Alerts.</li> </ul>
4	Data Abstraction	<p>Mobile Device</p> 	<ul style="list-style-type: none"> <li>• It is a user's application to subscribe or publish topics to ThingSpeak.</li> <li>• It shows the data of the inventory and alert message.</li> </ul>

### 3.6 Implementation of IoT-based Inventory System Simulation

Due to this project being Software-based, CoppeliaSim offers a better illustration in understanding the proposed IoT-based Inventory Management system. The environment in CoppeliaSim will be set up with four different shelves that contain stocks, Machine Type-B (MTB) robot as shown in Figure 3.6, conveyor belt, and PC. The MTB is used to pick up those stocks from the shelves and place them on to the conveyor, and the RFID reader will scan the stocks.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Figure 3.7 shows the general idea of the algorithm to construct the IoT-based Inventory System using a Machine Type-B (MTB) robot in pick and place the stocks and scanned by the limit switch to control the conveyor speed. Besides, it is assumed that there is an RFID Reader to read the tag of the stocks in a close range and a PC that is connected with ThingSpeak to get the data.

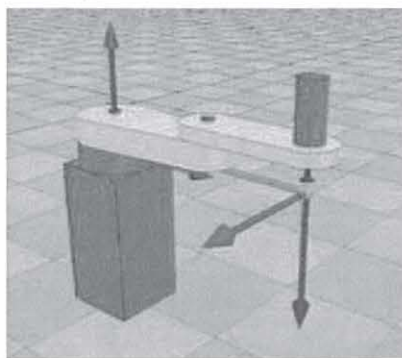


Figure 3.6 Machine Type-B (MTB) robot



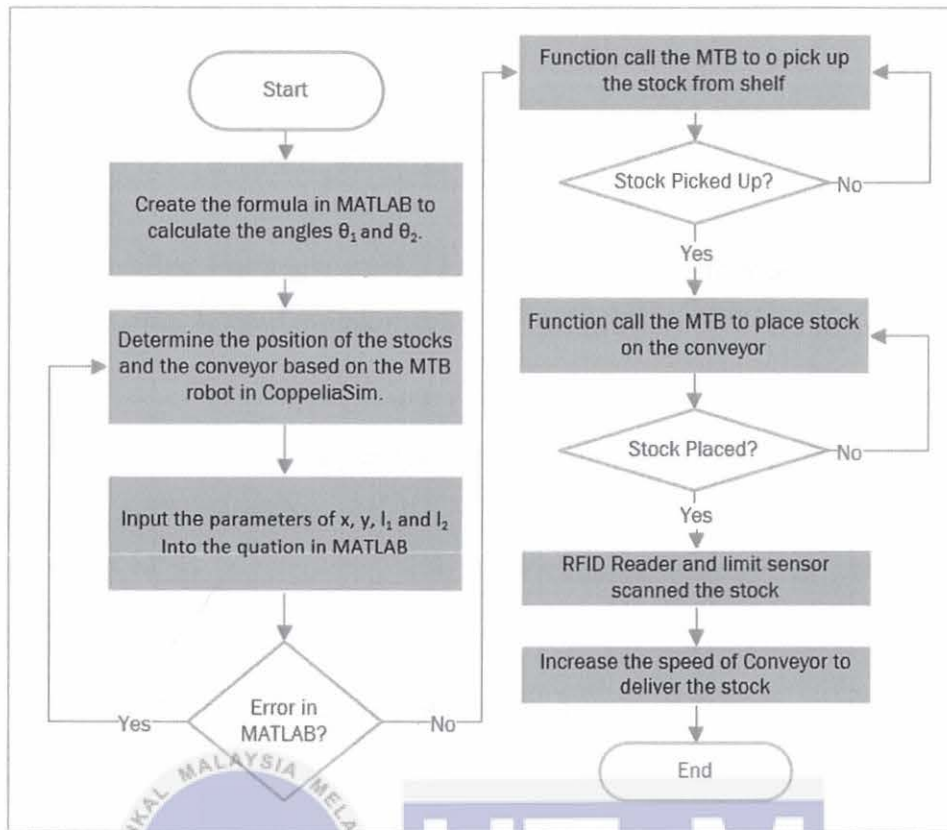


Figure 3.7 Algorithm of the simulation in CoppeliaSim

MTB robot is a two-degree-of-freedom (DOF) robot. The angle of rotation of the joints to reach the desired place is determined by the inverse kinematics, which is the position and orientation of the end-effector. The end-effector at  $Z_3$  is static. The equation of the cosine  $\theta_2$  ( $c_2$ ) of the axis at the second joint is given as Equation 3.1:

$$c_2 = \frac{x^2 + y^2 + l_1^2 + l_2^2}{2l_1l_2} \quad \text{Equation 3.1}$$

Where  $x$  and  $y$  refer to the  $x$ -axis and  $y$ -axis at  $Z_3$ , respectively,  $l_1$  and  $l_2$  are the lengths of the first and second links of the MTB robot. By using the trigonometric equivalent value, we obtain the equation of sine  $\theta_2$  ( $s_2$ ) Equation 3.2:

$$c_2^2 + s_2^2 \equiv 1$$

$$s_2^2 = \pm \sqrt{1 - c_2^2} \quad \text{Equation 3.2}$$



The angle of  $\theta_2$  can be obtained from the arctangent function in MATLAB with the Equation 3.3:

$$\theta_2 = \text{atan2}(s_2, c_2) \quad \text{Equation 3.3}$$

In order to find the angle of  $\theta_1$ , 2 parameters are introduced,  $k_1$  and  $k_2$ . The new parameters is Equation 3.4 and Equation 3.5:

$$k_1 = l_1 + l_2 c_2 \quad \text{Equation 3.4}$$

$$k_2 = l_2 s_2 \quad \text{Equation 3.5}$$

The angle of  $\theta_1$  is calculated with the arctangent function in MATLAB with the Equation 3.6:

$$\theta_1 = \text{atan2}(y, x) - \text{atan2}(k_2, k_1) \quad \text{Equation 3.6}$$

As the angle of  $\theta_1$  and  $\theta_2$  are determined, the end-effector's position can be determined. The Equations 3.1 to Equation 3.6 are created in MATLAB to calculate the angles  $\theta_1$  and  $\theta_2$ . (MathWorks, 2016) Those inputs will be inserted to get the position that MTB robot can reach.

### 3.7 Raspberry Pi Implemented with MATLAB using MQTT

This is to simulate how the MQTT work between MATLAB and Raspberry pi, which can offer better analytics of the data sent between publisher and subscriber using MQTT.

#### 3.7.1 MQTT Protocol Architectural

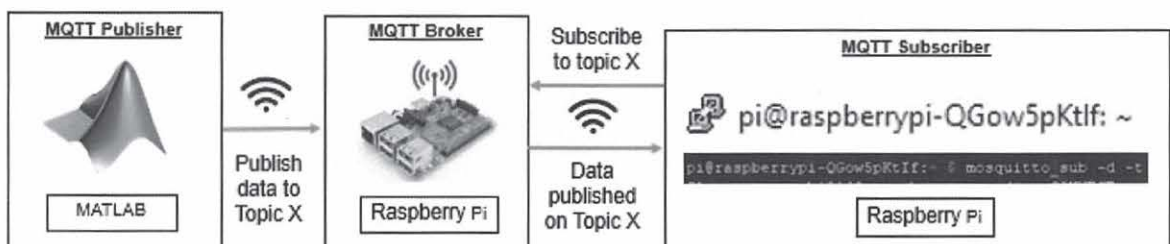


Figure 3.8 MQTT Protocol between MATLAB and Raspberry Pi

MQTT protocol will be set up between MATLAB and Raspberry pi using MATLAB coding. Figure 3.8 shows that raspberry pi will be connected to MATLAB by MQTT using its IP address. Then, MATLAB is configured as an MQTT publisher to publish the real-time inventory quantities of each stock under four topics named regarding the stock label. For example, publish stock A's quantities under Topic A. Numerous specific coding needs to present the MQTT function in constructing this function.

### 3.7.2 MQTT Publisher

The warehouse inventory quantities update is stimulated by MATLAB act as MQTT Publisher to publish the inventory quantities to a raspberry pi. Figure 3.9 shows the algorithm of the coding generated (Appendix A) in MATLAB to illustrate the scenario of the worker hand-pick stocks randomly from four different shelves with the different initial number of stocks as in Table 3.3. The stocks will randomly decrease by one from stock labeled A, B, C or D for 20 times and publish to the MQTT subscriber in receiving the updated quantities.

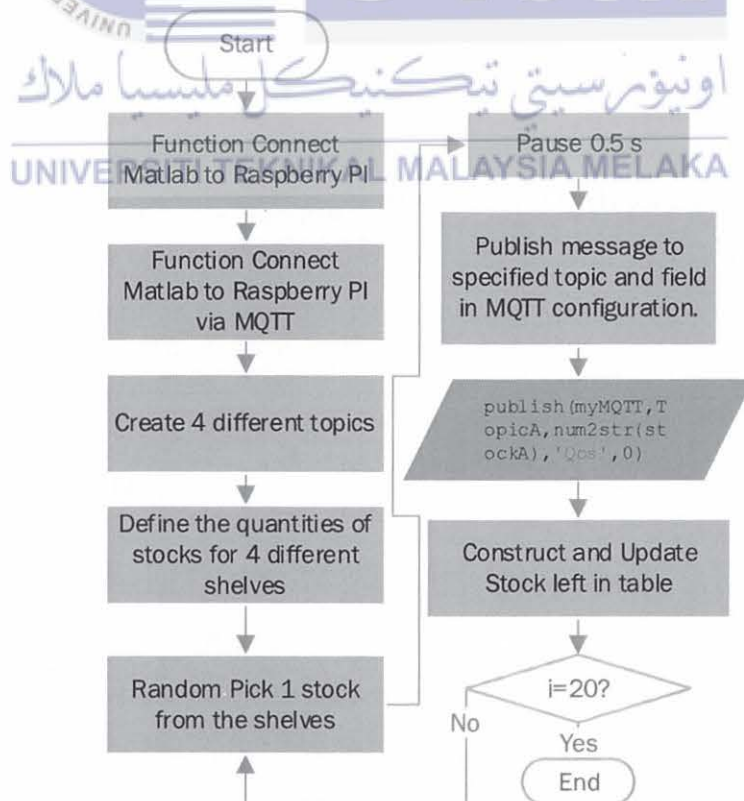


Figure 3.9 Algorithm of the coding in MATLAB connected to Raspberry Pi

Table 3.3 Initially quantities of stock in Raspberry Pi implementation

Stock Label	Initially Quantities
A	3
B	6
C	20
D	11
Total	40

### 3.7.3 MQTT Broker

Raspberry pi is the MQTT broker which offers MQTT clients to communicate. It is a media between publisher and subscriber to receive a message from the publisher, filter it and post it to subscribers.

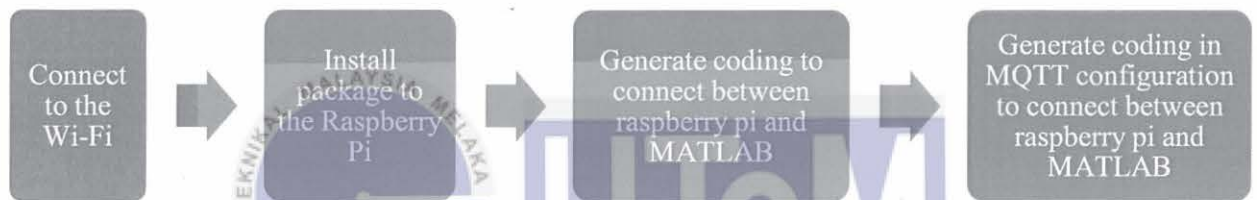


Figure 3.10 Set up Raspberry Pi connect to MQTT in Matlab

To connect between MATLAB and Raspberry Pi, several steps are carried out, as in Figure 3.10 shown. First, Raspberry Pi has to be activated and connected to the Wi-Fi. Then, the installation of a support package of Raspberry pi in MATLAB for the libraries of MATLAB functions. Next, function call to connect the raspberry pi with MATLAB and generate the coding in MQTT configuration to send the message using MQTT.

### 3.7.4 MQTT Subscriber

PuTTY is used to access the Raspberry pi that can run on the laptop with a command-line interface. The IP address of the raspberry pi is the key to access the PuTTY. Raspberry pi needs to be activated to determine the IP address using the network scanner, as shown in Figure 3.11. Then, insert the IP address (192.168.15) with the Port '22' using SSH, as in Figure 3.12 shown.

As there are 4 topics published, so it is necessary to open 4 PuTTY configurations to subscribe the topic. Then, sign in using 'pi' as the username and 'raspberry' for the password as in Figure 3.13.



Figure 3.11 Scanning IP address of Raspberry Pi

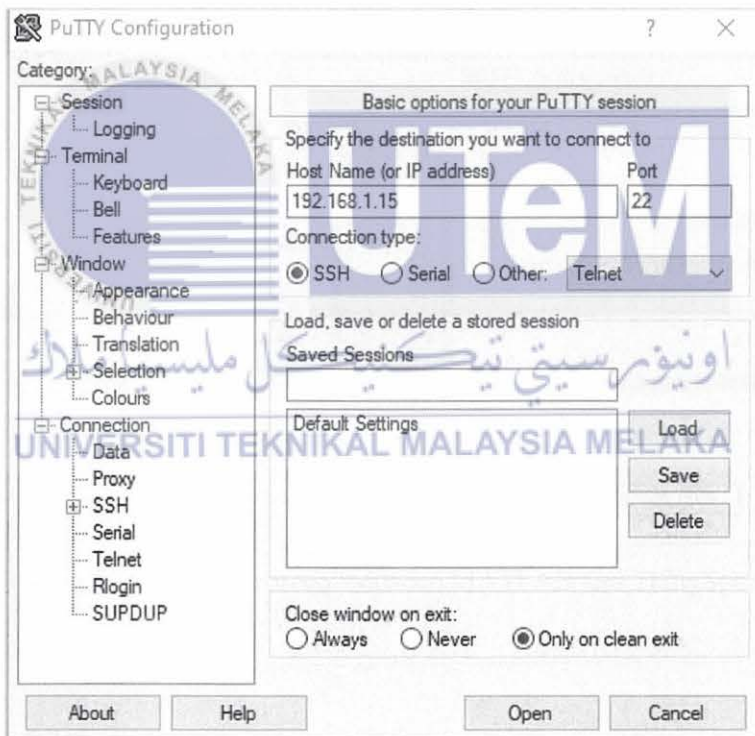


Figure 3.12 PuTTY Configuration



Figure 3.13 Interface of the command-line of Raspberry Pi



### 3.8 ThingSpeak Implemented with MATLAB using MQTT

This architecture stimulates how the MQTT works for the small industry that uses a basic IoT implementation that requires only Wi-Fi connectivity and the ThingSpeak platform.

#### 3.8.1 MQTT Protocol Architecture

MQTT protocol will be constructed between MATLAB and ThingSpeak. Figure 3.14 shows the MATLAB directly connecting to ThingSpeak in sending and receiving messages. MATLAB is configured as MQTT publisher to publish the real-time inventory quantities of each stock under four topics named in MQTT configuration.

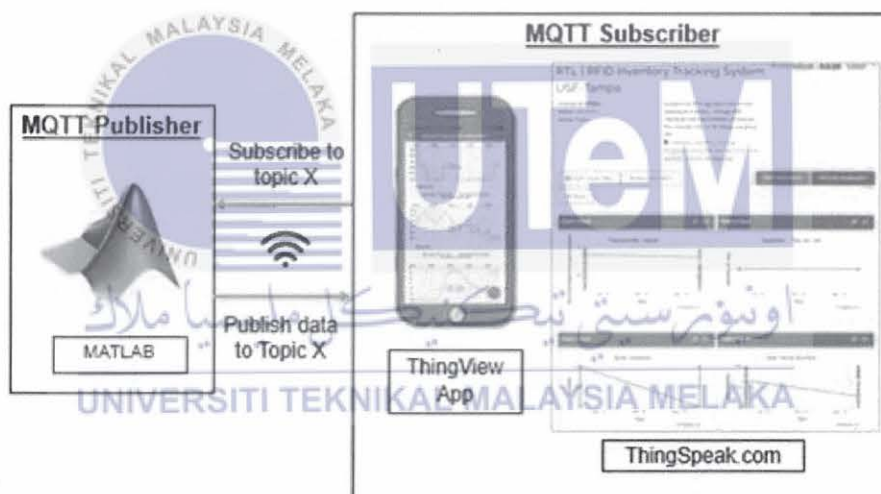


Figure 3.14 MQTT Protocol between MATLAB and ThingSpeak

#### 3.8.2 MQTT Publisher

The warehouse inventory quantities update is stimulated by MATLAB act as MQTT publisher to publish the inventory quantities to ThingSpeak. Figure 3.15 shows the algorithm of the coding generated ([Appendix B](#)) will be set up using MATLAB in MQTT configuration using the Read API Key to publish the inventory data of the stocks after the stocks decrease by one from random stock for 20 times with the initial quantities as in Table 3.4. This illustrates the scenario of the worker to hand-pick stocks randomly in the real-time.

Table 3.4 Initially quantities of stock in ThingSpeak implementation

Stock Label	Initially Quantities
A	6
B	9
C	20
D	11
Total	46

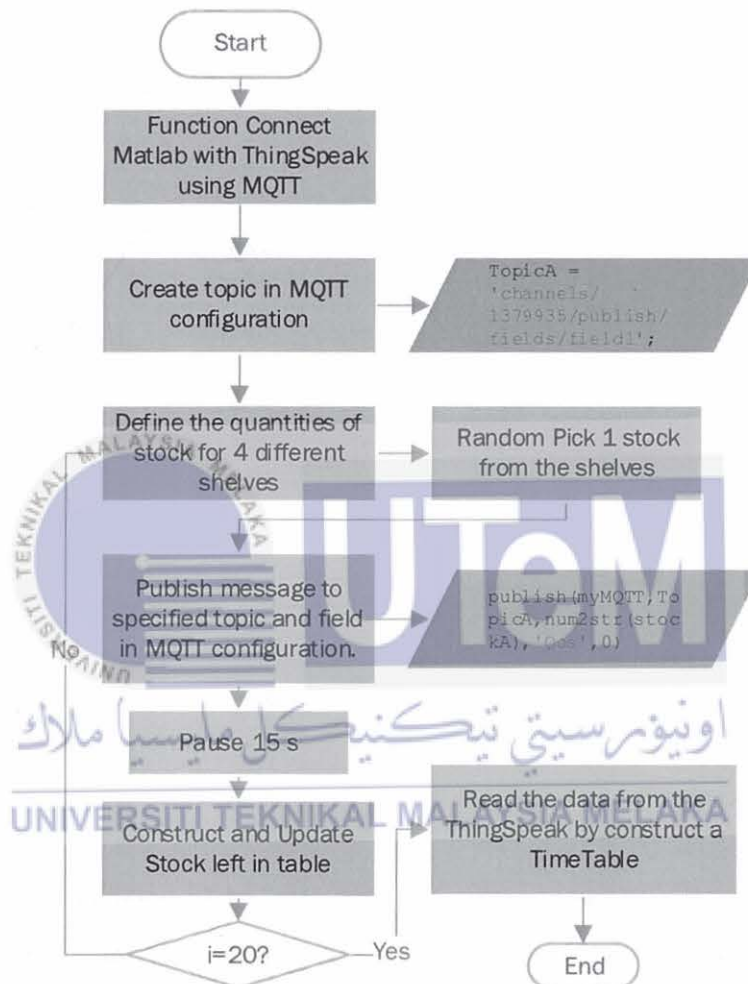


Figure 3.15 Algorithm of the coding in MATLAB connected to ThingSpeak

### 3.8.3 MQTT Subscriber

ThingSpeak and ThingView are MQTT subscribers that receive the data published from MATLAB and generate it into an analytic graph. ThingSpeak is accessed via the online browser (ThingSpeak.com) while ThingView accesses the data using the application (APP) that can get it free from ‘Google Play Store’.

### 3.8.3.1 ThingSpeak

The steps to set up the ThingSpeak channel are shown as below:

1. Register E-mail address in ThingSpeak
2. Create a new channel with a title and description and 4 fields in the channel represent the Stock A, B, C and D as in Figure 3.16.
3. Save the Channel.

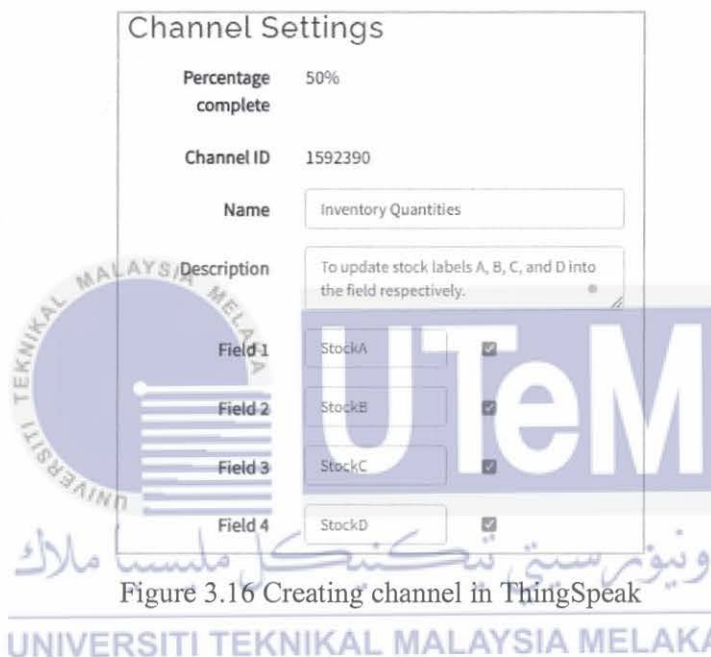


Figure 3.16 Creating channel in ThingSpeak

The setting connecting MATLAB and ThingSpeak via MQTT is shown step by step from Figure 3.17 to Figure 3.21. Figure 3.19 shows that ThingSpeak has offered the MQTT properties in 'Devices'. Figure 3.20 shows the MQTT Credentials consisting of their unique client ID, username, and password. Figure 3.21 shows that the channels entitle with 'Inventory Quantities' are given authority to access and publish data.



Figure 3.17 Get MQTT under 'Device'

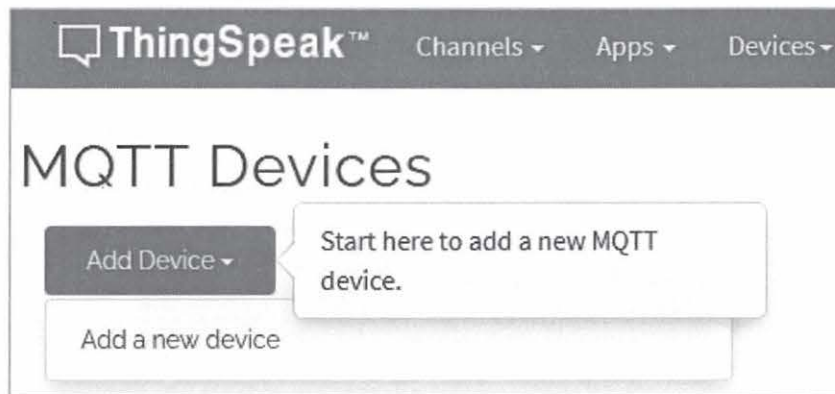


Figure 3.18, Add a new device under MQTT Device

**Device Information**

**Name**

**Description**

Figure 3.19 Fill up the details in the following column

**MQTT Credentials**  
Use these MQTT credentials to publish and subscribe to ThingSpeak channels. [Learn More](#)

**Client ID**

**Username**

**Password**

Figure 3.20 Specific MQTT credentials for the channel.

**Authorize channels to access**

-- Select a Channel --

...

Authorized Channel	Allow Publish	Allow Subscribe
Inventory Quantities (1592390)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 3.21 Add channels to allow publish and subscribe



### 3.8.3.2 ThingView

The data can be accessed more easily by using the ThingView APP in mobile phone. Fill in the channel ID that wants to subscribe as in Figure 3.22 shown.



Figure 3.22 ThingView in subscribing channels

### 3.9 Alert System in ThingSpeak

ThingSpeak has implemented an alert system that can notify the user by sending the E-mail to avoid the out of stock happening. The code ([Appendix C](#)) will be generated to alert the user. The process flow to set up Alert System in MATLAB Analysis:

1. Get the ThingSpeak Alert API key from my Profile.
2. Then go to Apps then MATLAB Analysis and generate the code based on the algorithm in Figure 3.23 by connecting to the Time Control.
3. Click Save and Run, and the notification can be received after a few moments.

Figure 3.23 shows the data is getting from Field 1 and Field 2 in a channel. When the inventories level is lower than 5, it will send the data to the E-mail. After 5 minutes, if the inventory is still lower than 5 quantities, it will send the alert message again.

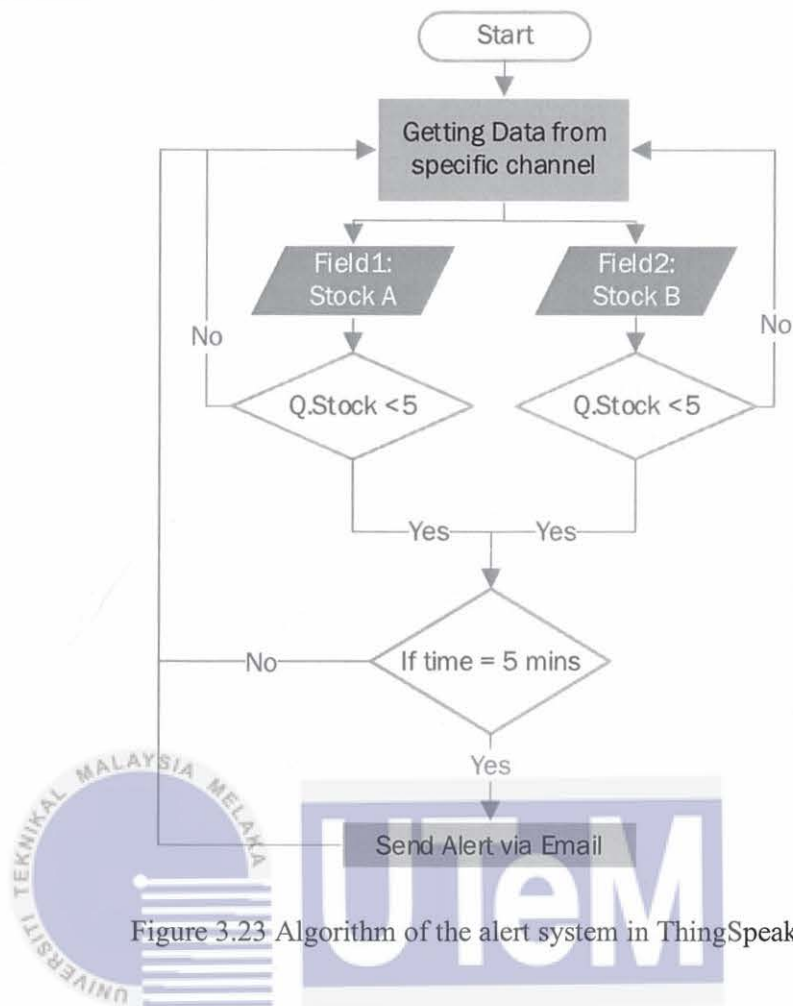


Figure 3.23 Algorithm of the alert system in ThingSpeak

### 3.10 Efficiency Analysis

Then, a comparison of the performance of Raspberry Pi and ThingSpeak on the time taken in receiving the data is carried out to determine which applications (Raspberry Pi or ThingSpeak) use lesser time is preferred. The variables will be tabulated in Table 3.5, and the result will be recorded in Table 3.6.

Table 3.5 Variables in comparison of the performance of Raspberry Pi and ThingSpeak

Variables	Description
Independent	1. Raspberry Pi is connected using Ethernet. 2. ThingSpeak is connected wireless.
Dependent	1. Time interval between 2 messages. 2. Total time taken in receiving all the data.
Controlled	1. WIFI 2.4 GHz is used 2. MATLAB platform in publish data 3. 20 message is published

Table 3.6 Result of the performance of Raspberry Pi and ThingSpeak

Result	MQTT in Raspberry Pi	MQTT in ThingSpeak
Total time used		
Time interval between 2 messages		
Efficiency		

The total time taken of completing the transmission of data using MQTT protocol is taken by using the ‘run and time’ in the MATLAB’s editor, as Figure 3.24 shows.

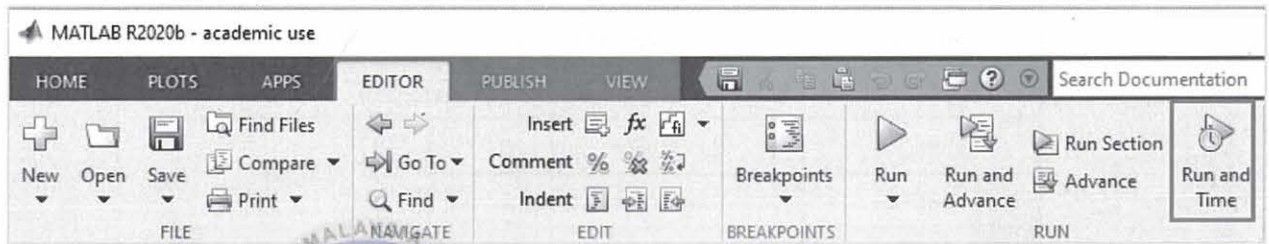


Figure 3.24 ‘Run and Time’ in Matlab (Editor)

The Efficiency Analysis of the MQTT protocol of Raspberry Pi and ThingSpeak is carried out to ensure the data sent by the publisher is all received by the subscriber without any missing or different value. The formula to calculate the value of efficiency is given as below:

$$\frac{\text{Data output}}{\text{Data input}} \times 100\% = \text{Efficiency}$$

The data output only collected when the data is tallied with the data input. Next, the parameter of this study is defined as in Table 3.7 to determine the efficiency of the system based on the value obtained.

Table 3.7 Parameter of the efficiency

Parameter	Result
100%	Efficient
<100%	No efficient and need to be revised

### 3.11 Cost Analysis

To give investors the best idea of how much the project will cost and whether it can fit into their own budgets. Table 3.5 shows the cost of implementation MQTT needed. The Price of the component is taken from cytron (<https://my.cytron.io/>). Hence, the total price is RM167.90.

Table 3.8 Implementation Cost

Items		Cost (RM)
Gigabit Ethernet Cable 5 meters		12.90
Raspberry Pi 3 Model B		155
MQTT	Broker: ThingSpeak	0(Free)
	API: ThingView	0(Free)
Total		RM167.90





## CHAPTER 4

### RESULT AND DISCUSSION

This chapter is going to discuss about the result of the developing of IoT-based Inventory System using MQTT protocol and the result of analyses in evaluating the performance and efficiency. The final consequence of this project is measured to assure the system achieves the objectives. Besides, the video simulation can be viewed in [Appendix F](#) for better understanding on the algorithm of the coding interpreted based on the result get.

#### 4.0 Implementation of IoT-based Inventory System

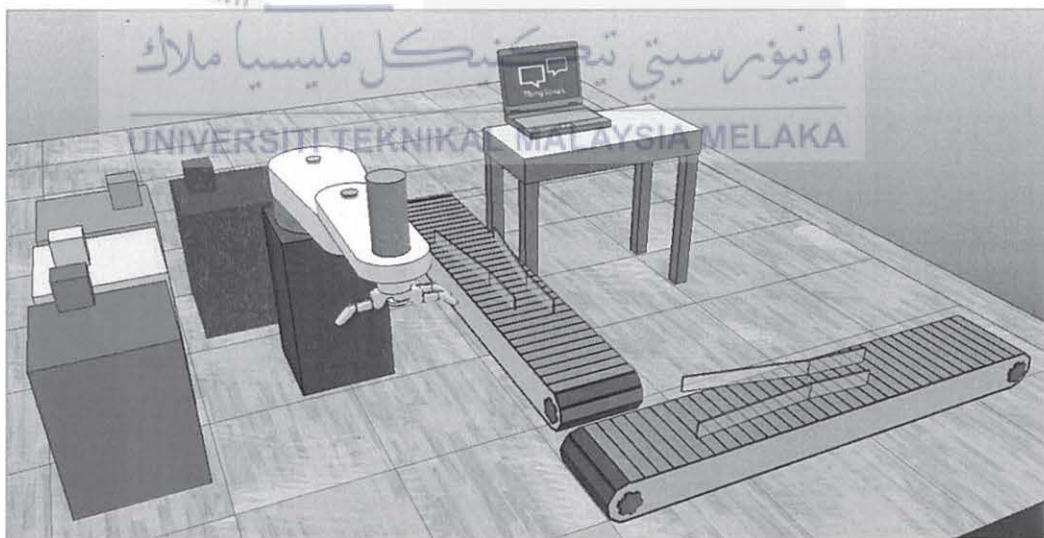


Figure 4.1 IoT-based inventory system

Developing an IoT-based Inventory System with MQTT protocol is one of the key objectives in this project by implementing MTB robot in collecting stock. Figure 4.1 illustrates 4 blocks representing 4 shelves with a different color. The stock will be carried out randomly by the robotic arm to the conveyor belt.

The MTB robot is not located on the origin of the simulation world but in a coordinate of (-0.8, 0). Therefore, the release location needs to be determined considering the offset of the MTB robot. Otherwise, the release location will be inaccurate. Thus, Table 4.1 indicates the correct coordinate of x and y.

Table 4.1 True coordinate on x-axis and y-axis.

Stock	Coordinate	World	MTB Robot	(MTB Robot) – (World) = (True)
A	x-axis	-1	-0.8	$(-0.8) - (-1) = 0.2$
	y-axis	0.8	0	$(0) - (0.8) = -0.8$
B	x-axis	-0.7	-0.8	$(-0.8) - (-0.7) = -0.1$
	y-axis	0.8	0	$(0) - (0.8) = -0.8$
C	x-axis	-0.25	-0.8	$(-0.8) - (-0.25) = -0.55$
	y-axis	0.6	0	$(0) - (0.6) = -0.6$
D	x-axis	-0.21	-0.8	$(-0.8) - (-0.21) = -0.59$
	y-axis	0.325	0	$(0) - (0.325) = -0.325$

The equations to calculate the angle of joint 1,  $\theta_1$  and joint 2,  $\theta_2$  are generated in MATLAB. Figure 4.2 shows the example of the picking position of Stock A. The values x, y,  $l_1$  and  $l_2$  are inputted into the equation to get the 'theta1deg',  $\theta_1$  and 'theta2deg',  $\theta_2$  easily. However, the original calculated by MATLAB is in radian units while CoppeliaSim uses degree. Thus, the answer is changed to radian with the line 'rad2deg'.

```

clear; close all ; clc

% Pick up position in stock A
x = -0.8;
y = -0.8;
l1 = 0.467;
l2 = 0.4005;

c2 = (x^2+y^2-l1^2-l2^2) / (2*l1*l2);
s2 = sqrt(1-c2^2);

theta2rad = atan2(s2,c2);
theta2deg = rad2deg(theta2rad)

k1 = l1+l2*c2;
k2 = l2*s2;

theta1rad = atan2(y,x) - atan2(k2,k1);
theta1deg = rad2deg(theta1rad)

theta2deg =
    55.7098

theta1deg =
    -82.6698

```

Figure 4.2 Coding in MATLAB to calculate the angle  $\theta_1$  and  $\theta_2$

Figure 4.3 shows the 'theta1deg',  $\theta_1$  (-92.6696) and 'theta2deg',  $\theta_2$  (36.2898) are substituted into MTB language to run the MTB robot in picking and placing the stock A as in Appendix D.

```

CLEARBIT 1
SETROTVEL 90
SETLINVEL 0.1
MOVE -92.6696 36.2898 0.025 -151 (Pick up Stock A)
REM CLOSE GRIP:
SETBIT 1
WAIT 1500
MOVE 77.8691 95.9058 0 -151 (Place Stock A at conveyor)
REM open the gripper:
CLEARBIT 1
WAIT 1500

```

Figure 4.3 Part of the MTB language

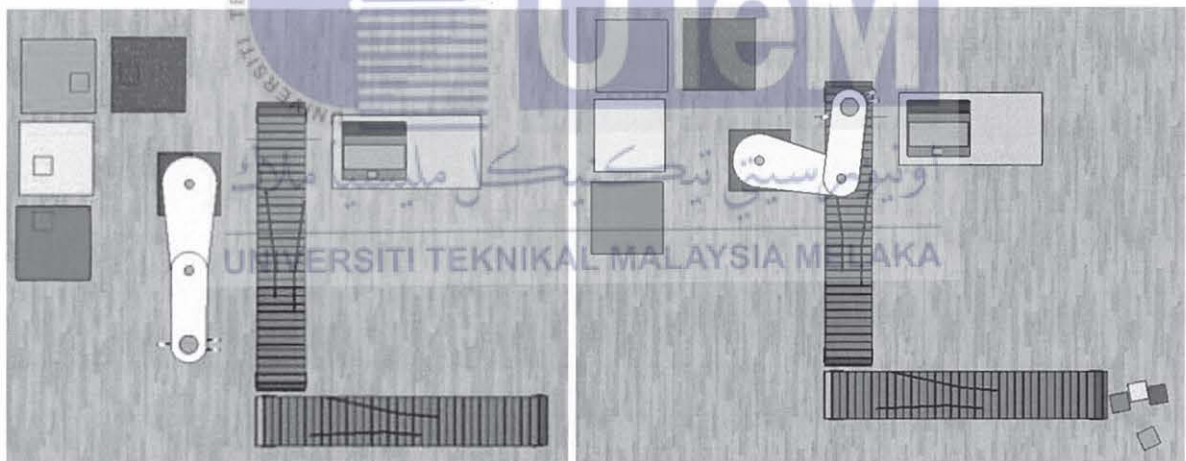


Figure 4.4 Initial environment of simulation (left) and end environment of simulation (right)

Figure 4.4 shows the environment of the start and end. At the start, the 4 different stocks with different color representing the label A, B, C and D are placed in respectively rectangular. Once the stimulation begins, the MTB robot is move to pick up the stock A (red) and placed on the conveyor. This process is repeated on stock B, C and D. Lastly, they will accumulate at the floor. The sequence of the MTB robot is predefined as it does not have any technology to take the stock randomly. Many industries prefer the inverse kinematics as the solution is very direct. All in all, CoppeliaSim is very helpful for students and engineers to simulate the projects and it is also a user-friendly application



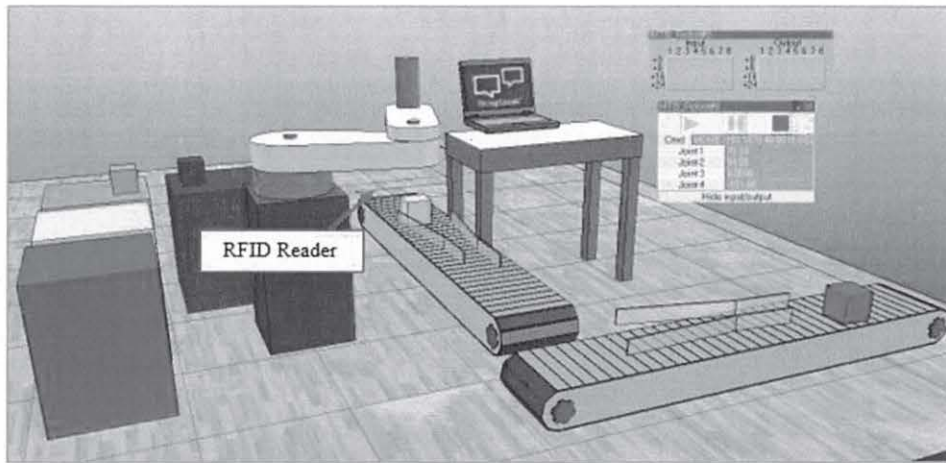


Figure 4.5 Stock is scanned and loaded onto conveyor belt

Figure 4.5 shows there is a RFID reader at the front side of conveyor belt that can read the RFID tags that attached at the stocks. Then, the data will be updated to the ThingSpeak from the raspberry pi via MQTT. Then, inventory can be tracked automatically and access by mobile devices using Thingspeak application.

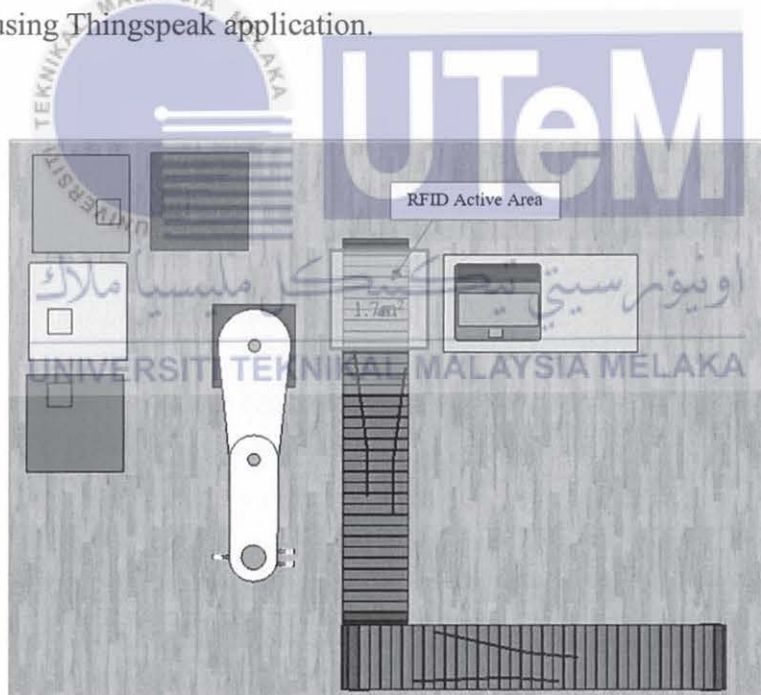


Figure 4.6 The RFID active area for reading RFID

The RFID system has a long bandwidth, so it will be easily scanned the stock around that area. Hence, the RFID system needs to be defined to secure the only stock on the conveyor to be scanned. Based on the references in [Appendix E](#), it should be allocated in a 2.89-meter square (1.7m x 1.7m) as in Figure 4.6 shown. The limit of the range is to ensure the stock can be scanned and to prevent it from the disturbances of the database system of RFID.



## 4.1 MQTT Protocol Architecture with Raspberry Pi

To achieve objective 2, Raspberry Pi is used as one of the applications in tracking the inventory quantities via MQTT protocol.

### 4.1.1 Setup Raspberry Pi

The MQTT Hand-Pick Stock Randomly Scenario outlined in the previous chapter has been implemented to connect to the raspberry pi through the MATLAB in Software-based. The raspberry pi is set up by connecting the Wi-Fi router via Ethernet cable wire and is activated by the power supply as shown in Figure 4.7.



Figure 4.7 Wi-Fi set-up of Raspberry Pi

### 4.1.2 Capability of MQTT Publisher

The capability of the MQTT publisher (MATLAB) is defined as the ability to publish the message to raspberry pi via MQTT. Once the coding ([Appendix A](#)) run, the result is interpreted as shown in Figure 4.8 to 4.11, and they will be discussed.

Figure 4.8 shows the connection between raspberry pi and Matlab has successfully implemented using the MQTT configuration by showing 'Raspberry Pi 3 Model B' connected using the device address (192.168.1.15).

```

rpi =

raspi with properties:

    DeviceAddress: '192.168.1.12'
    Port: 18734
    BoardName: 'Raspberry Pi 3 Model B'
    AvailableLEDs: {'led0'}
    AvailableDigitalPins: [4,5,6,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
    AvailableSPIChannels: {'CE0','CE1'}
    AvailableI2CBuses: {'i2c-0','i2c-1'}
    AvailableWebcams: {}
    I2CBusSpeed:

<a href="matlab:raspi.internal.helpView('raspberrypilo','RaspiSupportedPeripherals')">Supported peripherals</a>

```

Figure 4.8 Properties of Raspberry Pi in Matlab

Figure 4.9 shows MQTT is successfully called out. It shows there is only one device (raspberry pi) connected to the raspberry pi using the transport layer of Transmission Control Protocol (TCP).

```

myMQTT =

Mqtt with properties:
    BrokerAddress: "tcp://192.168.1.12"
    Port: 1883
    ClientID: "myClient"
    Timeout: 5
    Connected: 1
    KeepAlive: 60

```

Figure 4.9 Connection between Raspberry Pi and Matlab using MQTT

```

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ans =
    3
    No. of stock being picked

ans =
    1
    35
    Total number of stock left

stockA =
    3

stockB =
    6

stockC =
    19

stockD =
    11

```

Figure 4.10 The inventory quantities update in MATLAB

Figure 4.10 shows the **blue circle** indicates the answer for the stock label being chosen, **yellow circle** indicates the quantity of stocks being picked and **red circle** indicates the total number of stock left, and the rest is the latest update of the inventory quantities of Stock A, Stock B, Stock C and Stock D. The function in Figure 4.10 is continued for another 19 times.

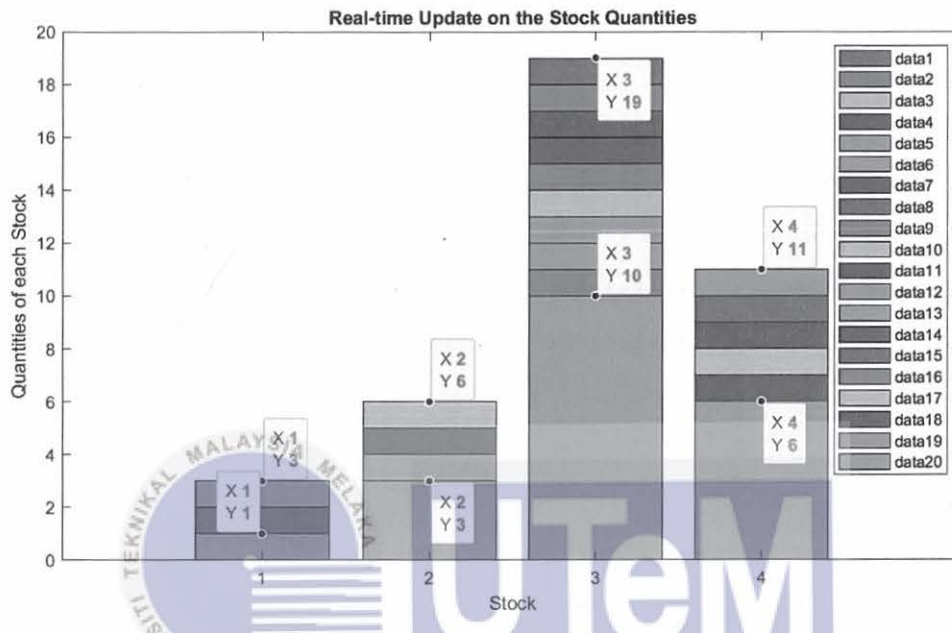


Figure 4.11 Real-time inventory update for four stock in Raspberry Pi

Table 4.2 Stocks quantities updated of Raspberry Pi for 20 Times

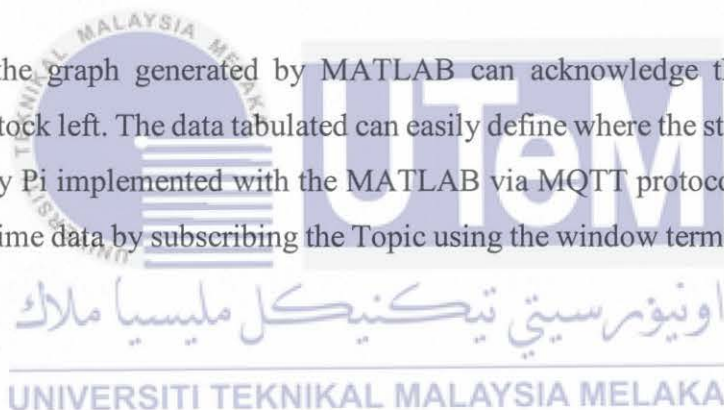
Pick up Cycle	Stock A	Stock B	Stock C	Stock D	Accumulate quantities
1	3	6	19	11	39
2	3	6	18	11	38
3	3	6	17	1	37
4	3	5	17	11	36
5	3	5	16	11	35
6	2	5	16	11	34
7	2	5	16	10	33
8	2	5	15	10	32
9	2	5	15	9	31
10	2	5	14	9	30
11	2	5	13	9	29
12	2	5	13	8	28
13	2	5	12	8	27
14	2	5	11	8	26
15	2	4	11	8	25
16	1	4	11	8	24
17	1	4	10	8	23
18	1	4	10	7	22
19	1	4	10	6	21
20	1	3	10	6	20

The stock that has picked up.

Figure 4.11 shows the bar chart of 'Real-Time Update on the Stock Quantities' generated from the MATLAB for better visualization of the stock's quantities updated for 20 times in four different stocks: **Stock 1, Stock 2, Stock 3 and Stock 4**, which represent **Stock A, Stock B, Stock C and Stock D**. The legend consists of 20 data which means the stock quantities decreased by one for 20 times. The data from the graph can be tabulated in Table 4.2 for better analysis on the tally with the data received in Raspberry Pi.

By comparing the data between MQTT publisher and MQTT subscriber, the number of data received by Raspberry Pi from Topic A, B, C, and D is the same as the number of data published (20 times) by MATLAB. Besides, the quantity of the stock updated by MATLAB each time is tally with Raspberry Pi. Hence, both the MQTT publisher and MQTT subscriber are capable as they can update the stocks' quantities successfully via MQTT.

Besides, the graph generated by MATLAB can acknowledge the engineer on the quantities of the stock left. The data tabulated can easily define where the stock has been picked up. The Raspberry Pi implemented with the MATLAB via MQTT protocol is ease for user to view on the real-time data by subscribing the Topic using the window terminal in laptop or PC.



#### 4.1.3 Capability of MQTT Subscriber

The capability of the MQTT subscriber (Raspberry Pi) is defined as the ability to receive the message from MATLAB via MQTT for 4 different topics. In getting 4 messages from 4 different topics, 4 PuTTY configures are opened and subscribed to 4 different topics respectively using 'mosquitto\_sub -d -t InventoryA' and changed the 'InventoryA' into InventoryB, C and D in the other 3 PuTTY configurations.

Figure 4.12 till Figure 4.15 show **4 topics are successfully subscribed and received the update of stock's quantities 20 times without any missing data**. In short, the MQTT publisher and subscriber are both capable.



```

pi@raspberrypi-QGow5pKtIf: ~
pi@raspberrypi-QGow5pKtIf:~$ mosquitto sub -d -t InventoryA
Client mosqsub|6819-raspberryp sending CONNECT
Client mosqsub|6819-raspberryp received CONNACK (0)
Client mosqsub|6819-raspberryp sending SUBSCRIBE (Mid: 1, Topic: InventoryA, QoS
: 0)
Client mosqsub|6819-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', .
.. (9 bytes))
StockA 3
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', .
.. (9 bytes))
StockA 3
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 3
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 3
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 2
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 2
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 2
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 2
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 2
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 2
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 2
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 2
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 1
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 1
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 1
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 1
Client mosqsub|6819-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryA', ... (9 bytes))
StockA 1

```

Figure 4.12 Data Received of Topic A in Raspberry Pi





```
pi@raspberrypi-QGow5pKtIf: ~
pi@raspberrypi-QGow5pKtIf:~ $ mosquitto_sub -d -t InventoryC
Client mosqsub|6821-raspberryp sending CONNECT
Client mosqsub|6821-raspberryp received CONNACK (0)
Client mosqsub|6821-raspberryp sending SUBSCRIBE (Mid: 1, Topic: InventoryC, QoS
: 0)
Client mosqsub|6821-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 19 Invention quantity
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 18
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 17
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 17
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 16
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 16
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 15
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 15
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 14
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 13
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 13
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 12
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 11
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 11
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 11
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 10
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 10
Client mosqsub|6821-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryC', .
.. (10 bytes))
StockC 10
```

Figure 4.14 Data Received of Topic C in Raspberry Pi

```

pi@raspberrypi-QGow5pKtIf: ~
pi@raspberrypi-QGow5pKtIf:~ $ mosquitto_sub -d -t InventoryD
Client mosqsub|6822-raspberryp sending CONNECT
Client mosqsub|6822-raspberryp received CONNACK (0)
Client mosqsub|6822-raspberryp sending SUBSCRIBE (Mid: 1, Topic: InventoryD, QoS: 0)
Client mosqsub|6822-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (10 bytes))
StockD 11 Invention quantity
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (10 bytes))
StockD 11
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (10 bytes))
StockD 11
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (10 bytes))
StockD 11
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (10 bytes))
StockD 11
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (10 bytes))
StockD 11
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (10 bytes))
StockD 10
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (10 bytes))
StockD 10
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 9
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 9
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 9
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 8
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 8
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 8
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 8
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 8
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 8
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 7
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 6
Client mosqsub|6822-raspberryp received PUBLISH (d0, q0, r0, m0, 'InventoryD', ... (9 bytes))
StockD 6

```

Figure 4.15 Data Received of Topic D in Raspberry Pi

## 4.2 MQTT Protocol Architecture with ThingSpeak

To achieve the objective 2, the ThingSpeak introduced as an application that can be used as one of the applications in tracking the inventory quantities via MQTT protocol. The ThingSpeak account is registered and successfully creating an interface of the channels with the ID (1592390) as shown in Figure 4.16. It consists of 4 Fields which represents Stock A, B, C and D. Next, the MQTT devices is successfully created as shown in Figure 4.17 to get the credential that consists of client ID, username, password and port are the keys to accessed the data via MQTT.



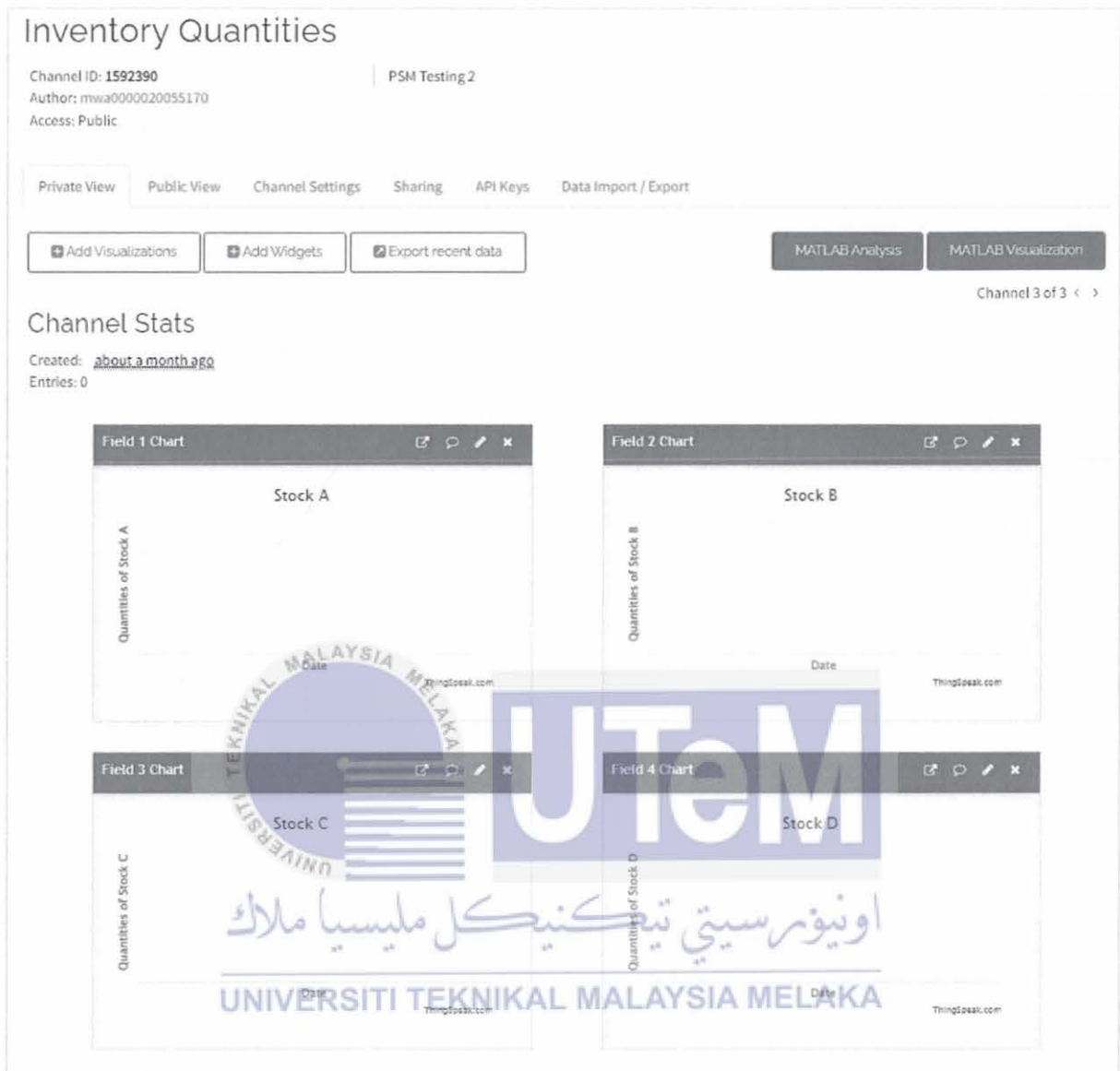


Figure 4.16 Interface of the ThingSpeak with 4 fields

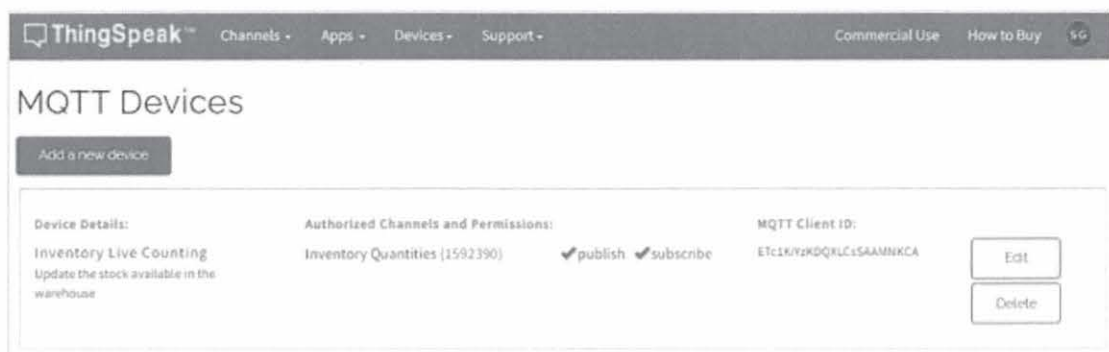


Figure 4.17 MQTT credential

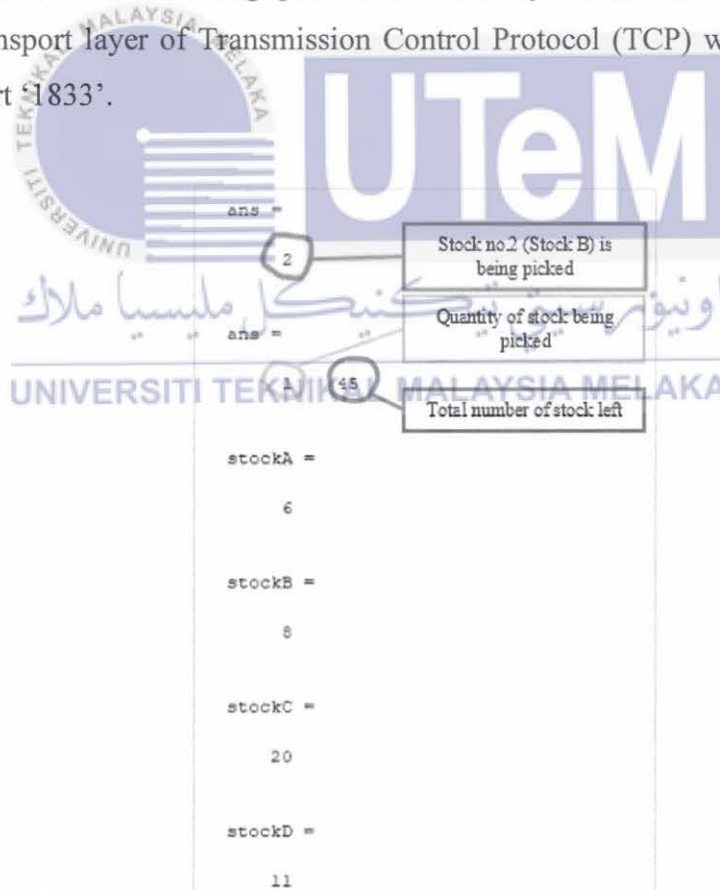
#### 4.2.1 Capability of MQTT Publisher

The capability of the MQTT publisher (MATLAB) is defined as the ability to publish the message to Thingspeak via MQTT. Once the coding ([Appendix B](#)) run, the result is interpreted as shown in Figure 4.18 to 4.20 and will be discussed.

```
myMQTT =  
  
Mqtt with properties:  
  BrokerAddress: "tcp://mqtt3.thingspeak.com"  
  Port: 1883  
  ClientID: "ETclKiYzKDQXLCsSAAMNKCA"  
  Timeout: 5  
  Connected: 1  
  KeepAlive: 60
```

Figure 4.18 Successfully connected between MATLAB and ThingSpeak via MQTT

Figure 4.18 shows the ThingSpeak is successfully connected to the MATLAB via MQTT using transport layer of Transmission Control Protocol (TCP) which connect to the ClientID with port '1883'.



The image shows a MATLAB console window with a large 'UTeM' watermark. The console output includes several lines of code and their results. Three callout boxes are present: one pointing to the number '2', another pointing to the number '45', and a third pointing to the text 'Total number of stock left'. The console output is as follows:

```
ans =  
2  
ans =  
45  
Total number of stock left  
  
stockA =  
6  
  
stockB =  
8  
  
stockC =  
20  
  
stockD =  
11
```

Figure 4.19 The inventory quantities update in MATLAB

Figure 4.19 indicates the meaning of the result presented in the MATLAB. The function in Figure 4.19 is continued for another 19 times.

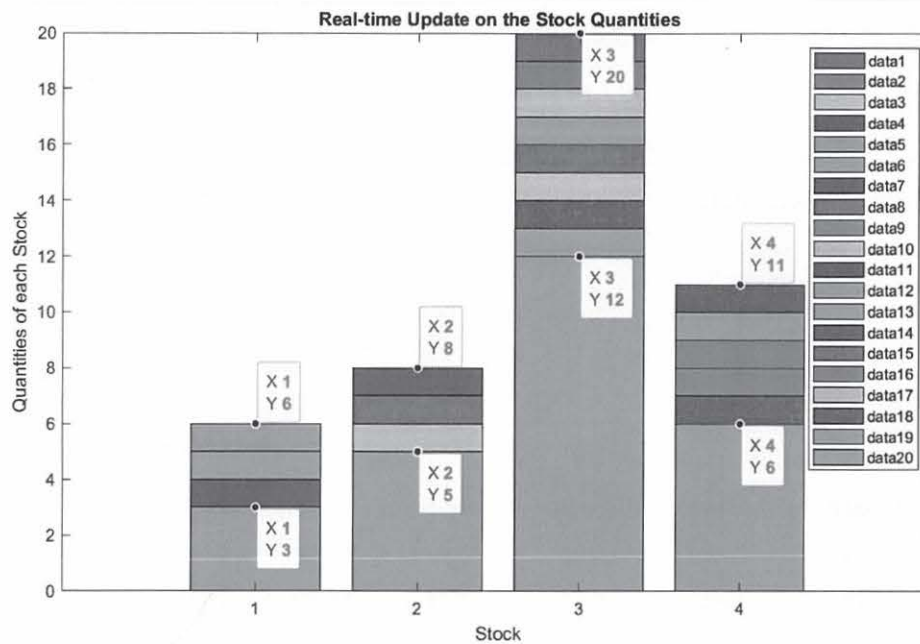


Figure 4.20 Real-time inventory update for four stock in ThingSpeak

The Figure 4.20 shows the bar chart of 'Real-Time Update on the Stock Quantities' generated from the MATLAB for 20 times in four different stocks: **Stock 1, Stock 2, Stock 3 and Stock 4** which represent **Stock A, Stock B, Stock C and Stock D**. The legend consists of 20 data which means the quantities of stock decreased by one for 20 times. The data from the graph can be tabulated in Table 4.3 for tallying the data received in ThingSpeak.

Table 4.3 Stocks quantities updated of Thingspeak for 20 Times

Pick up Cycle	Stock A	Stock B	Stock C	Stock D	Accumulate quantities
1	6	8	20	11	45
2	6	8	19	11	44
3	6	8	18	11	43
4	6	8	17	11	42
5	6	8	17	10	41
6	6	8	17	9	40
7	6	8	16	9	39
8	6	7	16	9	38
9	6	7	15	9	37
10	6	7	15	8	36
11	6	7	14	8	35
12	6	7	13	8	34
13	5	7	13	8	33
14	4	7	13	8	32
15	3	7	13	8	31
16	3	6	13	8	30
17	3	6	13	7	29
18	3	5	13	7	28
19	3	5	13	6	27
20	3	5	12	6	26

The stock that has picked up.



From the Table 4.3, it shows the highest picked up rate of the Stock is Stock C and the lowest is Stock A. Besides, Figure 4.21 present the result of the ‘ThingSpeakRead’ which can easily to determine when is the stock decrease at which type of stocks. The data start to read after 20 data has been updated and due to some latency, the nearest data 8<sup>th</sup> cycle is the first one to be read.

```
data =
```

39x4 timetable

Timestamps	StockA	StockB	StockC	StockD
25-Jan-2022 17:40:31	NaN	7	NaN	NaN
25-Jan-2022 17:40:47	NaN	NaN	14	NaN
25-Jan-2022 17:41:06	NaN	NaN	NaN	8
25-Jan-2022 17:41:17	6	NaN	NaN	NaN
25-Jan-2022 17:41:31	NaN	7	NaN	NaN
25-Jan-2022 17:41:46	NaN	NaN	13	NaN
25-Jan-2022 17:42:05	NaN	NaN	NaN	8
25-Jan-2022 17:42:16	5	NaN	NaN	NaN
25-Jan-2022 17:42:33	NaN	7	NaN	NaN
25-Jan-2022 17:42:46	NaN	NaN	13	NaN
25-Jan-2022 17:43:01	NaN	NaN	NaN	8
25-Jan-2022 17:43:16	4	NaN	NaN	NaN
25-Jan-2022 17:43:32	NaN	7	NaN	NaN
25-Jan-2022 17:43:46	NaN	NaN	13	NaN
25-Jan-2022 17:44:04	NaN	NaN	NaN	8
25-Jan-2022 17:44:16	3	NaN	NaN	NaN
25-Jan-2022 17:44:34	NaN	7	NaN	NaN
25-Jan-2022 17:44:46	NaN	NaN	13	NaN
25-Jan-2022 17:45:02	NaN	NaN	NaN	8
25-Jan-2022 17:45:16	3	NaN	NaN	NaN
25-Jan-2022 17:45:32	NaN	6	NaN	NaN
25-Jan-2022 17:45:46	NaN	NaN	13	NaN
25-Jan-2022 17:46:01	NaN	NaN	NaN	8
25-Jan-2022 17:46:16	3	NaN	NaN	NaN
25-Jan-2022 17:46:31	NaN	6	NaN	NaN
25-Jan-2022 17:46:46	NaN	NaN	13	NaN
25-Jan-2022 17:47:01	NaN	NaN	NaN	7
25-Jan-2022 17:47:16	3	NaN	NaN	NaN
25-Jan-2022 17:47:31	NaN	5	NaN	NaN
25-Jan-2022 17:47:46	NaN	NaN	13	NaN
25-Jan-2022 17:48:01	NaN	NaN	NaN	7
25-Jan-2022 17:48:16	3	NaN	NaN	NaN
25-Jan-2022 17:48:31	NaN	5	NaN	NaN
25-Jan-2022 17:48:46	NaN	NaN	13	NaN
25-Jan-2022 17:49:03	NaN	NaN	NaN	6
25-Jan-2022 17:49:16	3	NaN	NaN	NaN
25-Jan-2022 17:49:31	NaN	5	NaN	NaN
25-Jan-2022 17:49:46	NaN	NaN	12	NaN
25-Jan-2022 17:50:03	NaN	NaN	NaN	6

Figure 4.21 Time-Table generated by function call ThingspeakRead in Matlab

## 4.2.2 Capability of MQTT Subscriber

The capability of the MQTT subscriber (ThingSpeak and ThingView) is defined as the ability to receive the message from MATLAB via MQTT for 4 different topics or fields.

### 4.2.2.1 ThingSpeak

Once the coding is run, the quantities of stocks are published to the ThingSpeak without any missing of data. Each of the topics has been updated 20 times, hence the total entries are 80 as shown in Figure 4.22.

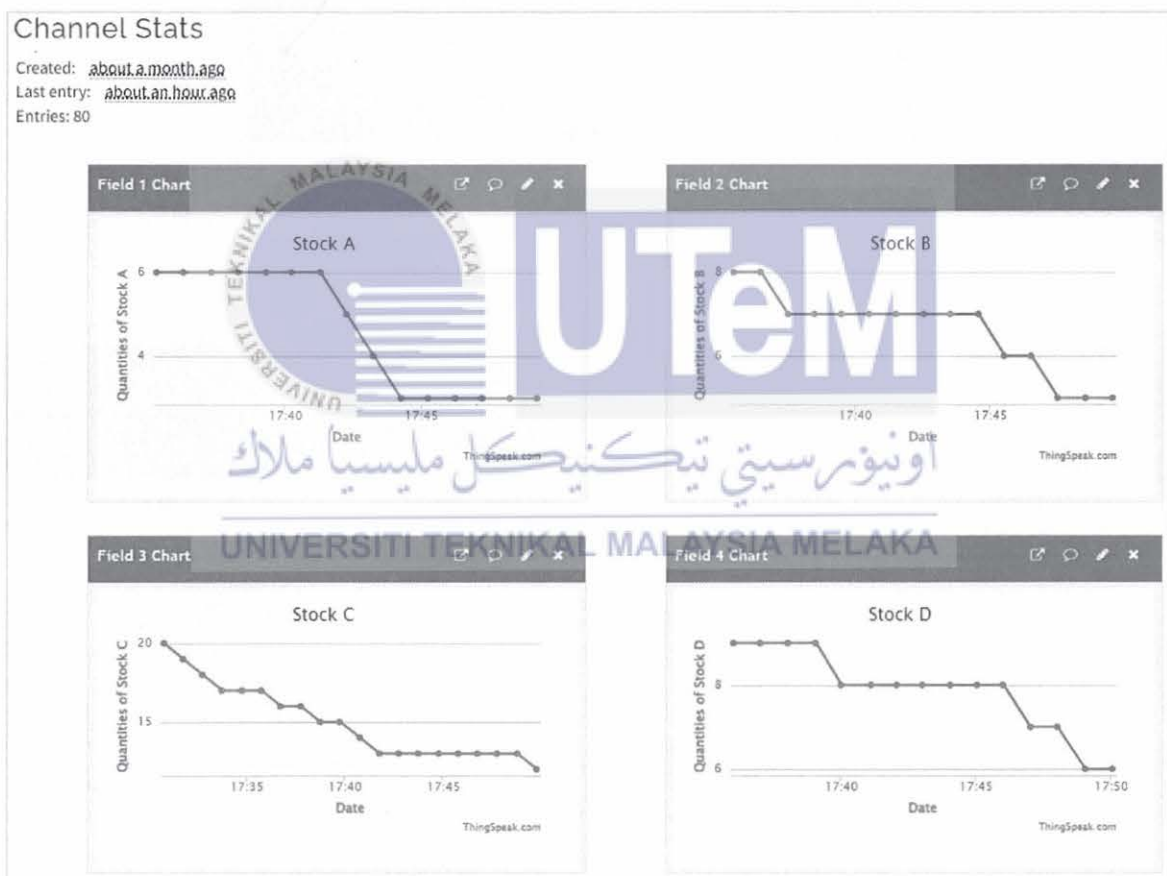


Figure 4.22 Graph generation by updating the data from MATLAB to ThingSpeak in different fields

Figure 4.22 shows the Stock A in Field 1 has decreased from 6 to 3, Stock B in Field 2 has decreased from 8 to 5, Stock C in Field 3 has decreased from 20 to 12, and Stock D in Field 4 has decreased from 11 to 6. The data updated in ThingSpeak are the same as the data stimulated in MATLAB. Hence, the MQTT publisher and MQTT subscriber are capable as the data published and received are accurate.

### 4.2.2.2 ThingView

To achieve objective 2, the ThingView is used to access the real-time update of the data using a phone that offer ease in tracking the inventory. After subscribing to the Channel, it will show the respective channel on the front page of the ThingView as shown in Figure 4.23. Figure 4.24 shows the view of Fields 1 (Stock A), Field 2 (Stock B), Field 3(Stock C) and Field 4 (Stock D) are same as a result shown in ThingSpeak. Hence, ThingView is also a capable subscriber to use for IoT applications.



Figure 4.23 Front page interface of ThingView



Figure 4.24 Screenshot from the phone



### 4.3 Send E-mail Alerts from ThingSpeak

The sending of E-mail alert from ThingSpeak is one of the objectives as the notification system can be sent out when low inventory.

#### 4.3.1 Implementation of Alert System

From the Figure 4.25, the ThingSpeak of the Alert system is collected and the E-mail shown is where the alert message publishes to. The alert system is limited for 2 alerts for this free trial. Hence in generate the coding, two variables have been set. If more than 2 requests made in once, the system will shoe 'Error' as the requests will be limited (MathWorks, 2022a).

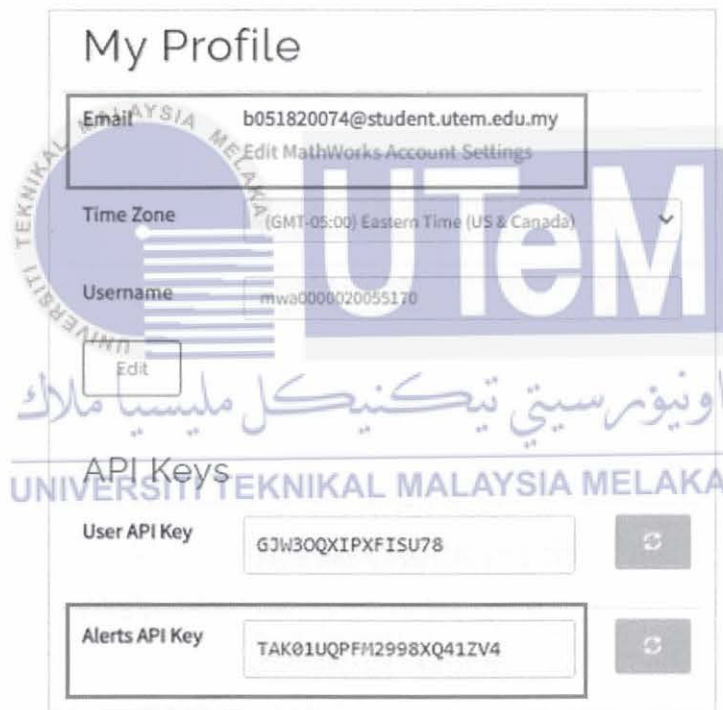


Figure 4.25 Profile of the ThingSpeak

#### 4.3.2 Performance of the Alert system

After coding ([Appendix C](#)) is successfully run as in Figure 4.26, the E-mail notification is received as in Figure 4.27 after a few moments due to there is no any further update of the quantity of stock A after the last update as in Figure 4.28. The alert system is highly dependent for the engineer in instant decision making to cope with the low inventory problem



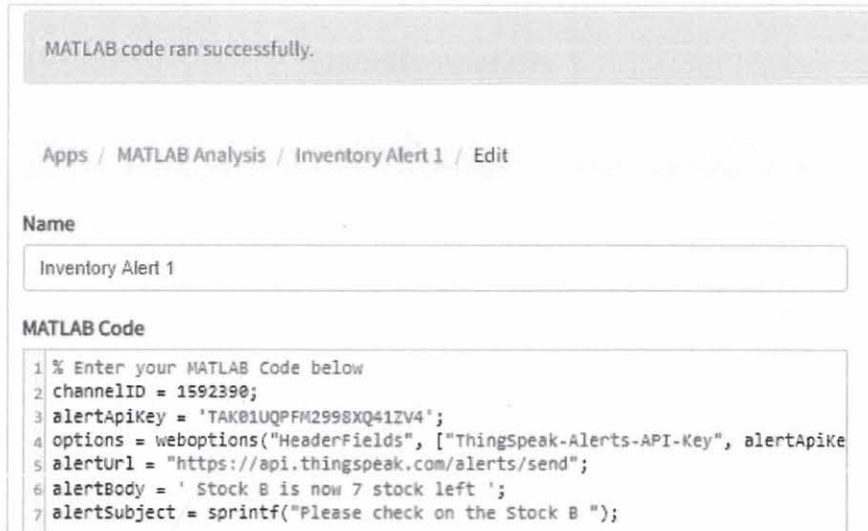


Figure 4.26 MATLAB code successfully run



Figure 4.27 Notification sent to e-mail when low inventory

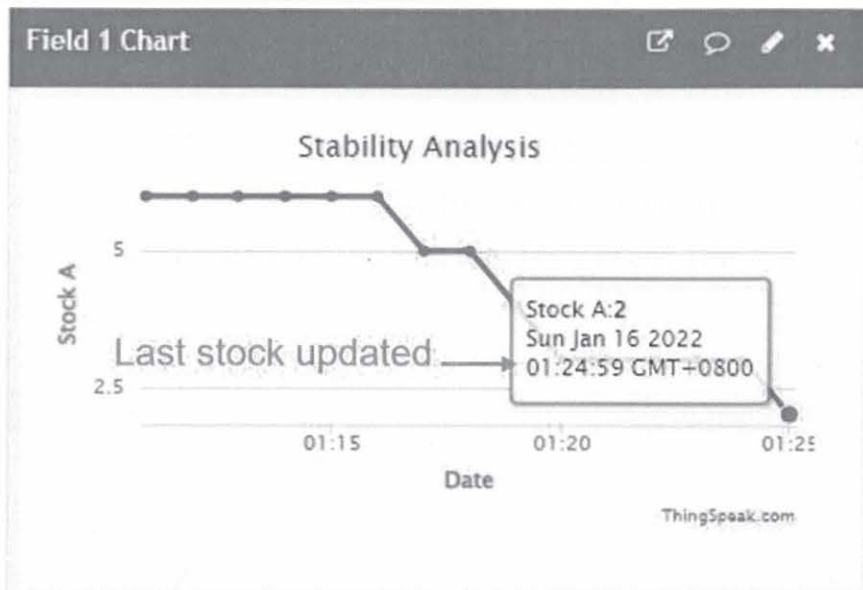


Figure 4.28 Last update of the Stock A's quantities

### 4.3.3 The Efficiency of the Alert System

The notification is sent with a time interval as short as 5 minutes as shown in Figure 4.29, by setting 'Time Control'. This feature can **replace manual reminders frequently**, so the worker can focus more on another task. Lastly, produce higher productivity of the workers.

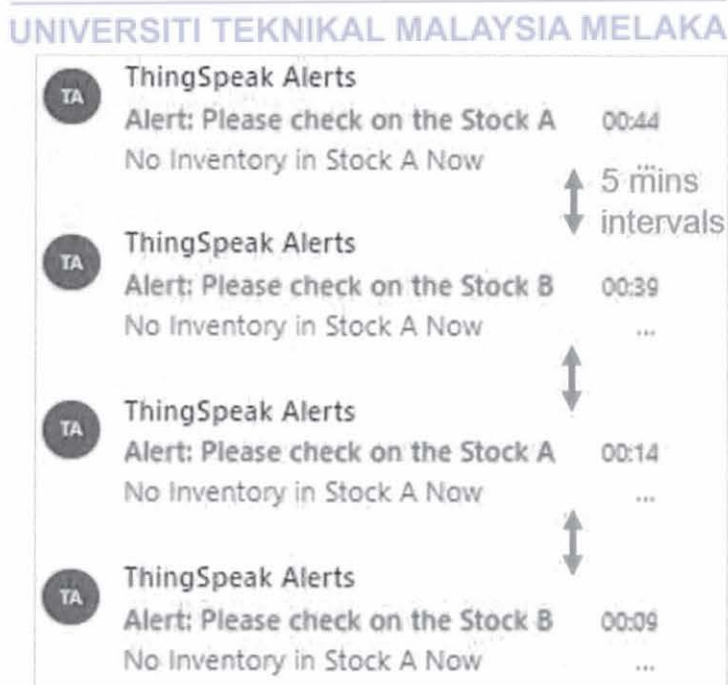
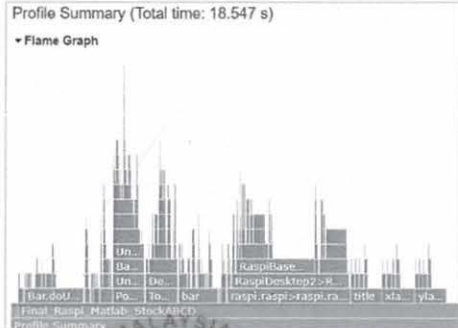
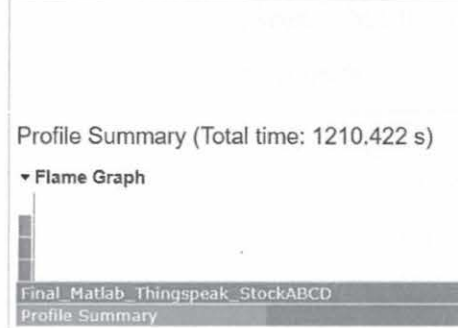


Figure 4.29 ThingSpeak Alerts in every 5 minutes

#### 4.4 Efficiency Analysis

Both of the MQTT protocols implemented with Raspberry Pi and ThingSpeak are capable; hence this analysis analyses the efficiency of both of the MQTT architectural. Table 4.4 shows the comparison between MQTT with Raspberry Pi and ThingSpeak.

Table 4.4 Result of the performance and efficiency of Raspberry Pi and ThingSpeak

Result	MQTT with Raspberry Pi	MQTT with ThingSpeak
Total Time used	 <p>Profile Summary (Total time: 18.547 s) ▼ Flame Graph</p> <p>Figure 4.30 Total time taken of the MQTT protocol with Raspberry Pi</p> <p>18.547s</p>	 <p>Profile Summary (Total time: 1210.422 s) ▼ Flame Graph</p> <p>Figure 4.31 Total time taken of the MQTT protocol with Raspberry Pi</p> <p>1210.422s</p>
Time interval between 2 messages	0.5 s	15 s
Efficiency	$\frac{20}{20} \times 100\% = 100\%$	$\frac{20}{20} \times 100\% = 100\%$

The MQTT with sending the data from MATLAB to Raspberry Pi takes 18.547 seconds which is shorter than the MATLAB to ThingSpeak as it takes 1210.422 seconds (20.17minutes). The total time used included the time connected to the application respectively (Raspberry Pi and ThingSpeak).

In MQTT with Raspberry Pi, it takes almost 18 seconds in connecting to Raspberry Pi, and sending data as short as 0.5s for each data update. While, MQTT with ThingSpeak, it takes a longer time due to the ThingSpeak has limit the free user to update data every 15 seconds.

Both of the implementation of MQTT is efficient as the data publish is all same as the data received, hence, this implementation is success and reliable.

## 4.5 Cost Analysis

This cost analysis aims to determine whether the implementation of this demonstration is worth with the value cost. Even though the project is software-based, there is still involved of the hardware setup in Raspberry Pi and the free applications used: ThingSpeak and ThingView.

The total cost to set up the Raspberry Pi with MQTT protocol is RM167.90 which is more than enough on constructing low-cost computing of Raspberry Pi using MQTT protocol. Besides, a lot of money can be saved by using the free version of ThingSpeak, but the time taken to receive the data is longer as the data update is limited by every 15 seconds.


In running real-time inventory tracking, the 'Standard version' is needed as it can update the data every second and more channels can be created. The comparison of 'Free version' and 'Standard version' of ThingSpeak is shown in Figure 4.32. The cost of the 'Standard Version' is USD 710.00/year (RM2,974.19/year) by improving the updating time into 10 seconds with adding 7 more channels added as shown in Figure 4.33. The upgrading of the updating time into 1 second with adding 7 more channels is cost USD 4150/year (RM 17,384.35) as Figure 4.34 shows. The ROI is needed to be calculated before implementing the system.

	FREE <sup>1</sup> For time-limited commercial evaluation of the service	STANDARD For all commercial, government and revenue generating activities
Scalable for larger projects	✗ No. Annual usage is capped.	✓
Number of messages	3 million/year (~8,200/day) <sup>(2)</sup>	33 million/year per unit (~90,000/day per unit) <sup>(2)</sup>
Message update interval limit	Every 15 seconds	Every second
Number of channels	4	250 per unit
MATLAB Compute Timeout	20 seconds	60 seconds
Private channel sharing	Limited to 3 shares	Unlimited
Technical Support	Community Support	Standard MathWorks support

Figure 4.32 Comparison of 'Free version' and 'Standard version' of ThingSpeak (MathWorks, 2022b)



### Pricing calculator




How many channels?

Currently: **3**

To be added:

Calculated number of channels needed: **10**



How often will they collect data?

Every

### Purchase

License type: **Standard**


ThingSpeak units:

x USD 710.00 price/unit/year

**Total: USD 710.00/year**

Figure 4.33 Features of the 'Standard Version' which cost USD 710.00/year

### Pricing calculator




How many channels?

Currently: **3**

To be added:

Calculated number of channels needed: **10**



How often will they collect data?

Every

### Purchase

License type: **Standard**

ThingSpeak units:

x USD 415.00 price/unit/year (6-49 units)

**Total: USD 4,150.00/year**

Figure 4.34 Features of the 'Standard Version' which cost USD 4,150.00/year

## CHAPTER 5

### CONCLUSION AND RECOMMENDATIONS

This chapter is to give an overall significant of study that the conclusion is made to fulfil the objective in Chapter 1.

#### 5.0 Conclusion

There are numerous of issues in current inventory tracking. The involvement of human is non-efficiency caused by the consumption of time in completing task is longer, and also the lack of awareness on the accuracy of data collected. These kinds of problems indicate that the issues must be handled properly with an appropriate method. Hence, the literature review in studying the IoT-based Warehouse inventory management system using MQTT protocol is carried out which fulfilled the Objective 1.

Due to some limitation in using lab in faculty, the project is successfully carried out in software-based. The IoT warehouse inventory management is successfully developed by simulated by CoppeliaSim which replacing the manual work into MTB robot in carry out the pick-and -place for a better understanding on the real-time scenario. Next, the, inventory system using MQTT protocol in between of MQTT subscriber (Raspberry Pi, ThingSpeak) and MQTT publisher (MATLAB) has successfully designed and run by using generation of code and setup of hardware. The system is capable as the data published from MATLAB are the same as the data received by Raspberry Pi and ThingSpeak. Besides, ThingView also offers the accessibility to track the inventory quantities using mobile phone. Hence, the Objective 2 is achieved.

In this project, it is not only providing the solution in using MQTT protocol but the efficiency and performance of the MQTT protocol in sending all the data of stock A, B, C and D for 20 times is evaluated and the objective 3 is achieved. The efficiency of both of MQTT architecture are 100% regarding to the number of updating and receiving is tallied. While the sending of data to ThingSpeak has about 15 seconds latency which the data only publish after 15 seconds. Those data publish within the 15 seconds will not be received. The notification has successfully sent to the E-mail address when low inventory happens. It also offers an efficient way in informing the user in every 5 minutes when there is still no update in the Stock.

## 5.1 Recommendation

This MQTT protocol has successfully connect with different type of applications in this study. Therefore, this system can be fully implemented in the real-time inventory system to figure out its full benefits. The integration of IoT with MQTT protocol are not a strange thing anymore. It just needs more time in implemented the entire system. The recommendation of the system for the future work can be modified as below:

1. The IoT-based inventory system with MQTT protocol is suggested to construct using a QoS 2 in sending the data even it is offline, the data still can be stored and it will be published after connected to Wi-Fi.
2. Raspberry Pi is suggested to become a gateway in interact with the RFID sensor and ThingSpeak as a more flexible methods for big factory in connecting with different type of protocol such as Bluetooth, Lora-Wan etc.
3. ThingSpeak is suggested to upgrade to be more reliable while the cost is too high. Hence, it will be great in trying to implement with Google Cloud Platform which offers a lot of features and variety of interface.
4. From the inventory quantities collected, generate an analysis to forecast the inventories that need to be buy in or giving promotion regarding on the market strategy.

## 5.2 Improvement of the System

Although the MQTT Protocol with IoT in tracking the inventory is done in the software-based, this study also proves that the using of IoT-based inventory warehouse management is efficient to track the inventory through mobile device is very convenient. However, there is some improvement that can be made which are:

1. Install the features of reconnect the Wi-Fi automatically when lost connect in Raspberry Pi. Hence, when the Wi-Fi is not stable and might be disable, the connection can be renewed in few seconds.
2. The Micro SD card is suggested to have at least 32GB for install of the libraries and more capacity to let the program run.

## 5.3 Sustainability

MQTT protocol is a low power consumption application which can give a greater sustainability of cost in a long term by saving a little from every month. By tracking of the inventory using IoT, the paper-based record can be eliminated, thus replacing with the interphase from the online. This reduces the impact on the environment from logging.



## REFERENCES

### Article in journal

- Aamer, A. M., & Sahara, C. R. (2021). Real-time data integration of an internet-of-things-based smart warehouse: a case study. *International Journal of Pervasive Computing and Communications*. <https://doi.org/10.1108/IJPCC-08-2020-0113>
- Abugabah, A., Nizamuddin, N., & Abuqabbeh, A. (2020). A review of challenges and barriers implementing RFID technology in the Healthcare sector. *Procedia Computer Science*, 170, 1003–1010. <https://doi.org/10.1016/j.procs.2020.03.094>
- Adiono, T., Anindya, S. F., Fuada, S., Afifah, K., & Purwanda, I. G. (2019). Efficient Android Software Development Using MIT App Inventor 2 for Bluetooth-Based Smart Home. *Wireless Personal Communications*, 105(1). <https://doi.org/10.1007/s11277-018-6110-x>
- Alwadi, A., Gawanmeh, A., Parvin, S., & Al-Karaki, J. N. (2017). Smart solutions for RFID based inventory management systems: A survey. *Scalable Computing*, 18(4), 347–360. <https://doi.org/10.12694/scpe.v18i4.1333>
- Bing, L., & Yang, L. (2019). Design and development of inventory system based on barcode scanning technology. *IOP Conference Series: Materials Science and Engineering*, 563(4). <https://doi.org/10.1088/1757-899X/563/4/042092>
- Dhillon, N. S., Sutandi, A., Vishwanath, M., Lim, M. M., Cao, H., & Si, D. (2021). A raspberry pi-based traumatic brain injury detection system for single-channel electroencephalogram. *Sensors*, 21(8). <https://doi.org/10.3390/s21082779>
- Di, H. (2020). Logistics management inventory model based on 5G Network and Internet of Things system. *Microprocessors and Microsystems*. <https://doi.org/10.1016/j.micpro.2020.103429>
- Dizdarević, J., Carpio, F., Jukan, A., & Masip-Bruin, X. (2019). A survey of communication

protocols for internet of things and related challenges of fog and cloud computing integration. In *ACM Computing Surveys* (Vol. 51, Issue 6).  
<https://doi.org/10.1145/3292674>

Durani, H., Sheth, M., Vaghasia, M., & Kotech, S. (2018). Smart Automated Home Application using IoT with Blynk App. *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, 393–397.  
<https://doi.org/10.1109/ICICCT.2018.8473224>

Et.al, S. A. (2021). Safety Features and Inventory Management for Vending Machine Using IoT. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(3).  
<https://doi.org/10.17762/turcomat.v12i3.1174>

Fathi, P., Karmakar, N. C., Bhattacharya, M., & Bhattacharya, S. (2020). Potential Chipless RFID Sensors for Food Packaging Applications: A Review. In *IEEE Sensors Journal* (Vol. 20, Issue 17, pp. 9618–9636). Institute of Electrical and Electronics Engineers Inc.  
<https://doi.org/10.1109/JSEN.2020.2991751>

Gopi Krishna, P., Sreenivasa Ravi, K., Hari Kishore, K., Krishna Veni, K., Siva Rao, K. N., & Prasad, R. D. (2018). Design and development of bi-directional IoT gateway using ZigBee and Wi-Fi technologies with MQTT protocol. *International Journal of Engineering and Technology(UAE)*, 7(2), 125–129.  
<https://doi.org/10.14419/ijet.v7i2.8.10344>

Jaloudi, S. (2019). Communication protocols of an industrial internet of things environment: A comparative study. *Future Internet*, 11(3). <https://doi.org/10.3390/fi11030066>

Kang, B., Kim, D., & Choo, H. (2017). Internet of Everything: A Large-Scale Autonomic IoT Gateway. *IEEE Transactions on Multi-Scale Computing Systems*, 3(3).  
<https://doi.org/10.1109/TMSCS.2017.2705683>

Khan, A. I., & Al-Habsi, S. (2020). Machine Learning in Computer Vision. *Procedia Computer Science*, 167, 1444–1451. <https://doi.org/10.1016/j.procs.2020.03.355>

Kumar, A. (2019). Hypertext Transfer Protocol (HTTP). In *Web Technology*.  
<https://doi.org/10.1201/9781351029902-4>

Laxmi, A. R., & Mishra, A. (2018). RFID based Logistic Management System using Internet of Things (IoT). *2018 Second International Conference on Electronics, Communication*

and *Aerospace Technology (ICECA)*, 556–559.

<https://doi.org/10.1109/ICECA.2018.8474721>

Maha Kavya Sri, J., Narendra, V. G., & Pai, V. (2019). Implementing and Testing of Internet of Things (IoT) Technology in Agriculture and Compare the Application Layer Protocols: Message Queuing Telemetry Transport (MQTT) and Hyper Text Transport Protocol (HTTP). *Communications in Computer and Information Science*, 1076. [https://doi.org/10.1007/978-981-15-0111-1\\_29](https://doi.org/10.1007/978-981-15-0111-1_29)

Mathaba, S., Adigun, M., Oladosu, J., & Oki, O. (2017). On the use of the Internet of Things and Web 2.0 in inventory management. *Journal of Intelligent and Fuzzy Systems*, 32(4), 3091–3101. <https://doi.org/10.3233/JIFS-169252>

Mir, S. B., & Lluca, G. F. (2020). Introduction to Programming Using Mobile Phones and MIT App Inventor. *Revista Iberoamericana de Tecnologías Del Aprendizaje*, 15(3), 192–201. <https://doi.org/10.1109/RITA.2020.3008110>

Mishra, B., & Kertesz, A. (2020). The use of MQTT in M2M and IoT systems: A survey. *IEEE Access*, 8. <https://doi.org/10.1109/ACCESS.2020.3035849>

Mishra, D., Yadav, R. S., Agrawal, K. K., Abbas, A., & Singh, R. K. (2019). Study of Application Layer MQTT Protocol with Assessment & Comparison of Hardware to Enable IOT. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.3444099>

Muyumba, T., & Phiri, J. (2017). A Web based Inventory Control System using Cloud Architecture and Barcode Technology for Zambia Air Force. *International Journal of Advanced Computer Science and Applications*, 8(11). <https://doi.org/10.14569/ijacsa.2017.081117>

Nath, S. V. (2017). IoT architecture. In *Internet of Things and Data Analytics Handbook*. <https://doi.org/10.1002/9781119173601.ch14>

Octaviani, P., & Ce, W. (2020). Inventory Placement Mapping using Bluetooth Low Energy Beacon Technology for Warehouses. *2020 International Conference on Information Management and Technology (ICIMTech)*, 354–359. <https://doi.org/10.1109/ICIMTech50083.2020.9211206>

Ouaissa, M., & Rhattoy, A. (2019). A secure group based authentication protocol for machine to machine communications in LTE-WLAN interworking architecture. *Indonesian*



*Journal of Electrical Engineering and Computer Science*, 16(2), 848–859.

<https://doi.org/10.11591/ijeecs.v16.i2.pp848-859>

- Pham, L. M., & Hoang, X.-T. (2021). An Elasticity Framework for Distributed Message Queuing Telemetry Transport Brokers. *VNU Journal of Science: Computer Science and Communication Engineering*, 37(1). <https://doi.org/10.25073/2588-1086/vnucsce.267>
- Putra Yudha, I. P. A., Sudarma, M., & Arya Mertasana, P. (2018). PERANCANGAN APLIKASI SISTEM INVENTORY BARANG MENGGUNAKAN BARCODE SCANNER BERBASIS ANDROID. *Jurnal SPEKTRUM*, 4(2). <https://doi.org/10.24843/spektrum.2017.v04.i02.p10>
- Rana, M. E. (2020). Recommendations for Implementing an IoT based Inventory Tracking and Monitoring System. *International Journal of Psychosocial Rehabilitation*, 24(5). <https://doi.org/10.37200/ijpr/v24i5/pr202099>
- Singh, S., Kumar, R., Panchal, R., Manoj, & Tiwari, K., & Tiwari, M. K. (2021). Impact of COVID-19 on logistics systems and disruptions in food supply chain. *International Journal of Production Research*, 59(7). <https://doi.org/10.1080/00207543.2020.1792000>
- Strain, T., Wilson, R. E., & Littleworth, R. (n.d.). *Computer Vision for Rapid Updating of the Highway Asset Inventory*. <https://doi.org/10.1177/0361198120928348>
- Susila, A., Riadi, I., & Prayudi, Y. (2017). Wi-Fi Security Level Analysis for Minimizing Cybercrime. *International Journal of Computer Applications*, 164(7). <https://doi.org/10.5120/ijca2017913667>
- Tabassum, T., Hossain, S. K. A., Rahman, M. A., Alhamid, M. F., & Hossain, M. A. (2020). An efficient key management technique for the internet of things. *Sensors (Switzerland)*, 20(7). <https://doi.org/10.3390/s20072049>
- Tejesh, B. S. S., & Neeraja, S. (2018). Warehouse inventory management system using IoT and open source framework. *Alexandria Engineering Journal*, 57(4). <https://doi.org/10.1016/j.aej.2018.02.003>
- ThingSpeak. (2020). IoT Analytics - ThingSpeak Internet of Things. In *ThingSpeak*.
- Tian, H., Wang, T., Liu, Y., Qiao, X., & Li, Y. (2020). Computer vision technology in agricultural automation —A review. In *Information Processing in Agriculture* (Vol. 7, Issue 1). <https://doi.org/10.1016/j.inpa.2019.09.006>



- Valente, F. J., & Neto, A. C. (2017). Intelligent steel inventory tracking with IoT / RFID. *2017 IEEE International Conference on RFID Technology and Application, RFID-TA 2017*. <https://doi.org/10.1109/RFID-TA.2017.8098639>
- Vatumalae, V., Rajagopal, P., Pandiyan, V., & Sundram, K. (2020). Warehouse Management System of a Third Party Logistics Provider in Malaysia. *International Journal of Economics and Finance*, 12(9). <https://doi.org/10.5539/ijef.v12n9p73>
- Yokotani, T., & Sasaki, Y. (2017). Comparison with HTTP and MQTT on required network resources for IoT. *ICCEREC 2016 - International Conference on Control, Electronics, Renewable Energy, and Communications 2016, Conference Proceedings*. <https://doi.org/10.1109/ICCEREC.2016.7814989>
- Yüksel, M. E. (2020). Power consumption analysis of a Wi-Fi-based IoT device. *Electrica*, 20(1), 62–70. <https://doi.org/10.5152/ELECTRICA.2020.19081>
- Zeadally, S., Siddiqui, F., & Baig, Z. (2019). 25 years of bluetooth technology. *Future Internet*, 11(9). <https://doi.org/10.3390/fi11090194>



Company / Organization Website

اونيورسيتي تيكنيكل  
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

- Pololu Corporation. (n.d) Arduino Uno R3. Retrieved from:  
<https://www.pololu.com/product/2191>
- Open Circuit (n.d) Motor Driver Shield L293D. Retrieved from:  
<https://opencircuit.shop/Product/Motor-Driver-Shield-L293D>
- ZF Friedrichshafen AG (n.d) ZF & Bluetooth® Low Energy. Retrieved from:  
<https://switches-sensors.zf.com/asia-en/zf-bluetooth-low-energy/>
- Open Source (n.d) What is a Raspberry Pi? Retrieved from:  
<https://opensource.com/resources/raspberry-pi>
- Open Source (n.d) What is an Arduino? Retrieved from:  
<https://opensource.com/resources/what-arduino>

## Article from internet/ website

Len Calderone (2019, December 17) What is Machine Vision? Retrieved from:

<https://www.roboticstomorrow.com/article/2019/12/what-is-machine-vision/14548>

Gorry Fairhurst (2020, October 1) Communication Protocol. Retrieved from:

<https://erg.abdn.ac.uk/users/gorry/course/intro-pages/protocols.html>

Adcbarcode (2017, November 22) RFID-Tag-Structure. Retrieved from:

<https://adcbarcode.com/2017/11/22/rfid-and-inventory-control/rfid-tag-structure/>

Rebecca Deprey (2020, Sep 8) Use MQTT to Share Sensor Data Among Multiple Devices.

Retrieved from: <https://medium.com/swlh/use-mqtt-to-share-sensor-data-among-multiple-devices-bc82d1a44ea4>

MathWorks. (2016). *Modeling Inverse Kinematics in a Robotic Arm - MATLAB & Simulink*

*Example.* <http://uk.mathworks.com/help/fuzzy/examples/modeling-inverse-kinematics-in-a-robotic-arm.html>

MathWorks. (2022a). *Send Alert.* The MathWorks, Inc.

[https://blogs.mathworks.com/iot/2020/01/10/send-email-alerts-from-thingspeak/?from=cn&doing\\_wp\\_cron=1643110804.0599510669708251953125](https://blogs.mathworks.com/iot/2020/01/10/send-email-alerts-from-thingspeak/?from=cn&doing_wp_cron=1643110804.0599510669708251953125)

MathWorks. (2022b). *Why Buy a Standard License ?* (pp. 0–3).

[https://thingspeak.com/prices/thingspeak\\_standard?license\\_name=Standard&number\\_of\\_devices=10&intervals=10&interval\\_type=Seconds](https://thingspeak.com/prices/thingspeak_standard?license_name=Standard&number_of_devices=10&intervals=10&interval_type=Seconds)

MDN contributors (2021, June 14) An overview of HTTP. Retrieved from:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

# APPENDIX

## Appendix A

Coding to publish data to Raspberry Pi from MATLAB.

```
clc;
clear;
close all;

% Connect to Rasp Pi ?192.168.1.8?374C)
rpi=raspi("192.168.1.15", "pi", "raspberrypi");
%Connect to MQTT
myMQTT=mqtt('tcp://192.168.1.15', 'ClientID', 'myClient')

%SDefine Topic's Name
TopicA = 'InventoryA';
TopicB = 'InventoryB';
TopicC = 'InventoryC';
TopicD = 'InventoryD';

a=[1,1,1];
b=[2,2,2,2,2,2];
c=[3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3];
d=[4,4,4,4,4,4,4,4,4,4,4];
all=[a,b,c,d];

% Random take from 4 shelf 25times
for i=1:20
    r = randi([size(all)], 1);
    all(r)
    all(r) = [];
    size(all)

    name1='StockA';
    name2='StockB';
    name3='StockC';
    name4='StockD';

%Categories
stockA = size(find(all > 0 & all < 2), 2);
stockB = size(find(all > 1 & all < 3), 2);
stockC = size(find(all > 2 & all < 4), 2);
stockD = size(find(all > 3 & all < 5), 2);
x=1:4;

pause(0.5)

% Result
A=[name1, ' ', num2str(stockA)];
B=[name2, ' ', num2str(stockB)];
C=[name3, ' ', num2str(stockC)];
D=[name4, ' ', num2str(stockD)];

% To publish to Raspberry Pi
publish(myMQTT,TopicA,A, 'Qos',0)
publish(myMQTT,TopicB,B, 'Qos',0)
publish(myMQTT,TopicC,C, 'Qos',0)
publish(myMQTT,TopicD,D, 'Qos',0)

%Construct Table in Figure
bar(x, [stockA,stockB,stockC,stockD]),xlabel('Stock'),ylabel('Quantities of
each Stock'),title('Real-time Update on the Stock Quantities');
hold on
end
```



## Appendix B

Coding to publish data to ThingSpeak from MATLAB.

```
clc;
clear;
close all;

%Connect to MQTT
myMQTT=mqtt('tcp://mqtt3.thingspeak.com','ClientID','ETclKiYzKDQXLCsS
AAMNKCA','Username','ETclKiYzKDQXLCsSAAMNKCA','Password','148W0U7Uy74
xjPL0+A9wf4yD','Port',1883)

%SDefine Topic's Name
TopicA = 'channels/1592390/publish/fields/field1';
TopicB = 'channels/1592390/publish/fields/field2';
TopicC = 'channels/1592390/publish/fields/field3';
TopicD = 'channels/1592390/publish/fields/field4';

a=[1,1,1,1,1,1];
b=[2,2,2,2,2,2,2,2,2];
c=[3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3];
d=[4,4,4,4,4,4,4,4,4,4,4];
all=[a,b,c,d];

% Random take from 4 shelf 25times
for i=1:20
    r = randi([size(all)], 1);
    all(r)
    all(r) = [];
    size(all)

%Categories
stockA =size(find(all > 0 & all < 2), 2);
stockB =size(find(all > 1 & all < 3), 2);
stockC =size(find(all > 2 & all < 4), 2);
stockD =size(find(all > 3 & all < 5), 2);

x=1:4;

% To publish to ThingSpeak
publish(myMQTT,TopicA,num2str(stockA),'Qos',0)
pause(15.0)
publish(myMQTT,TopicB,num2str(stockB),'Qos',0)
pause(15.0)
publish(myMQTT,TopicC,num2str(stockC),'Qos',0)
pause(15.0)
publish(myMQTT,TopicD,num2str(stockD),'Qos',0)
pause(15.0)

%Construct Table in Figure
bar(x,[stockA,stockB,stockC,stockD]),xlabel('Stock'),ylabel('Quantiti
es of each Stock'),title('Real-time Update on the Stock Quantities');
hold on
end

data = ThingSpeakRead(1592390,'Fields',[1,2,3,4],'NumMinutes',10,
'OutputFormat','TimeTable')
```

## Appendix C

Coding to send alert notification to E-mail via ThingSpeak's MATLAB Analysis.

```
% Enter your MATLAB Code below
channelID = 1592390;
alertApiKey = 'TAK01UQPFM2998XQ41ZV4';
options = weboptions("HeaderFields", ["ThingSpeak-
Alerts-API-Key", alertApiKey ]);
alertUrl = "https://api.thingspeak.com/alerts/send";
alertBody = ' Stock B is now 7 stock left ';
alertSubject = sprintf("Please check on the Stock B
");

[timestamps,StockA]=
thingSpeakRead(channelID,'NumMinutes',12,'Fields',1);
[timestamps,StockB] =
thingSpeakRead(channelID,'NumMinutes',12,'Fields',2);

%Set the outgoing message
if StockA < 5

    alertBody = sprintf ('Your Stock has less than 5
stocks now. Please top up the stock A. ');
    alertSubject = sprintf("Please check on the Stock A
");
    webwrite(alertUrl , "body", alertBody, "subject",
alertSubject, options);
    try
        webwrite(alertUrl , "body", alertBody, "subject",
alertSubject, options);
    catch someException
        fprintf("Failed to send alert: %s\n",
someException.message);
    end
end

if StockB<5
    alertBody = ' Your Stock has less than 5 stocks
now. Please top up the stock B. ';
    alertSubject = sprintf("Please check on the
Stock B ");
    try
        webwrite(alertUrl , "body", alertBody, "subject",
alertSubject, options);
    catch someException
        fprintf("Failed to send alert: %s\n",
someException.message);
    end
end
```

```

if isempty(StockA)
    alertBody = ' No Inventory in Stock A Now ';
    alertSubject = sprintf("Please check on the Stock
A ");
try
    webwrite(alertUrl , "body", alertBody, "subject",
alertSubject, options);

    catch someException
        fprintf("Failed to send alert: %s\n",
someException.message);
    end
end

if isempty(StockB)
    alertBody = ' No Inventory in Stock B Now ';
    alertSubject = sprintf("Please check on the Stock B
");
try
    webwrite(alertUrl , "body", alertBody, "subject",
alertSubject, options);

    catch someException
        fprintf("Failed to send alert: %s\n",
someException.message);
    end
end

```

اوتنورسیتی تکنیکل ملیسیا ملاک


UNIVERSITI TEKNIKAL MALAYSIA MELAKA



## Appendix D

Coding of MTB robot in defining the pick and place positions of the stock from different shelves red, yellow, green and purple.

```
REM 1ST
PROGRAM_BEGIN_LABEL
CLEARBIT 1
SETROTVEL 90
SETLINVEL 0.1
MOVE -92.6696 36.2898 0.025 -151 REM Pick from shelf red
REM CLOSE GRIP:
SETBIT 1
WAIT 1500
MOVE 77.8691 95.9058 0 -151 REM Place on conveyor
REM open the gripper:
CLEARBIT 1
WAIT 1500
REM 2ND
PROGRAM_MIDDLE_LABEL
CLEARBIT 1
SETROTVEL 90
SETLINVEL 0.1
MOVE -117.1058 43.4617 0.025 -151 REM Pick from shelf yellow
REM CLOSE GRIP:
SETBIT 1
WAIT 1500
MOVE 77.8691 95.9058 0 -151 REM Place on conveyor
REM open the gripper:
CLEARBIT 1
WAIT 1500
```



REM 3TH  
 CLEARBIT 1  
 SETROTVEL 80  
 SETLINVEL 0.1  
 MOVE -151.1870 40.6018 0.025 -151 REM Pick from shelf green  
 REM CLOSE GRIP:  
 SETBIT 1  
 WAIT 1500  
 MOVE 77.8691 95.9058 0 -151 REM Place on conveyor  
 REM open the gripper:  
 CLEARBIT 1  
 WAIT 1500  
 REM 4TH  
 CLEARBIT 1  
 PROGRAM\_MIDDLE\_LABEL  
 CLEARBIT 1  
 SETROTVEL 90  
 SETLINVEL 0.1  
 MOVE -186.7732 78.3970 0.025 -151 REM Pick from shelf purple  
 REM CLOSE GRIP:  
 SETBIT 1  
 WAIT 1500  
 MOVE 77.8691 95.9058 0 -151 REM Place on conveyor  
 REM open the gripper:  
 CLEARBIT 1  
 WAIT 1500



	Delta degree 1, $\theta_1$
	Delta degree 2, $\theta_2$

## Appendix E

The study of distance analysis of the RFID system from the previous project is illustrated as in Table B1 and Figure B1. The Table B1 shows the RFID can be read by the antenna in a range of -120 cm to 160 cm.

Table E Result of Data Analysis

Distance (cm)/ Test	Outdoor									Centre	Indoor								
	-180	-160	-140	-120	-100	-80	-60	-40	-20	0	20	40	60	80	100	120	140	160	180
Test 1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Test 2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
Test 3	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

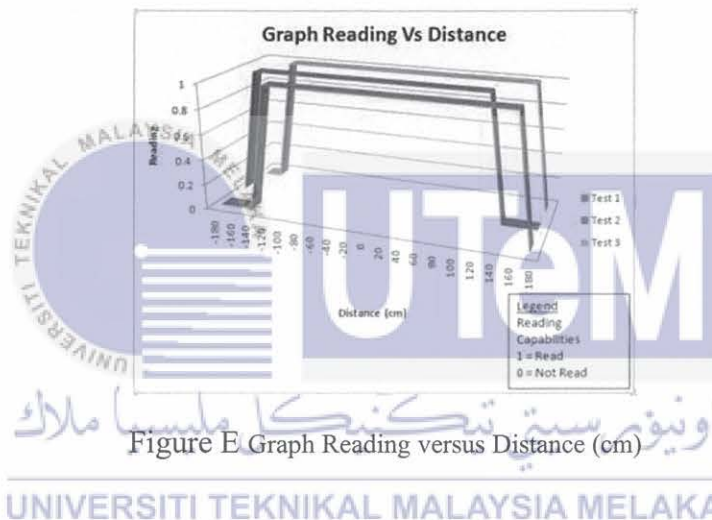


Figure E Graph Reading versus Distance (cm)

## Appendix F

Here is the video simulation of implementing IoT-based Inventory Warehouse Management System using CoppeliaSim and the Implementation of MQTT Protocol with Raspberry Pi and ThingSpeak in sending and receiving data. ( <https://youtu.be/8Kb3xRgEN9A> )