

**General platform for  
Internet of Things Application**

**LEONG ZHE HAO**

**A project report submitted in partial fulfilment of the  
requirements for the award of Bachelor of Engineering  
(Hons.)Electrical & Electronics Engineering**

**Faculty of Engineering and Science  
UniversitiTunku Abdul Rahman**

**January 2016**

## DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : \_\_\_\_\_

Name : Leong Zhe Hao

ID No. : 1301001

Date : \_\_\_\_\_

**APPROVAL FOR SUBMISSION**

I certify that this project report entitled “**General platform for Internet of Things Application**” was prepared by **LEONG ZHE HAO** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Electrical and Electronics at Universiti Tunku Abdul Rahman.

Approved by,

Signature : \_\_\_\_\_

Supervisor : \_\_\_\_\_

Date : \_\_\_\_\_

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2016, Leong Zhe Hao. All right reserved.

## **ACKNOWLEDGEMENTS**

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Mr. See Yuan Chark for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement.

## **GENERAL PLATFORM FOR INTERNET OF THINGS APPLICATION**

### **ABSTRACT**

In this proposed application, the concept of Internet of Thing is used to develop the framework for user application. The system is designed to collect and submit data to the internet, control and monitor by the users remotely. A server is developed to supply services to the healthcare devices and users, users can also access to the server by using web browser and mobile application. The communication protocol between the devices and server, the structure of the server framework and services of the server are designed with the concept of the Internet of Thing to utilize the resources. The protocol in this proposed application between the devices and server is MQTT, the devices communicate with the server through MQTT packet. The data published by the devices is stored in the server database and the database can be accessed by using the web browser. The performance of the proposed framework is also tested by using two different methods.

## TABLE OF CONTENTS

<b>DECLARATION</b>	<b>ii</b>
<b>APPROVAL FOR SUBMISSION</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>LIST OF FIGURES</b>	<b>xi</b>
<b>LIST OF SYMBOLS / ABBREVIATIONS</b>	<b>xiii</b>
<b>LIST OF APPENDICES</b>	<b>xiv</b>

### CHAPTER

<b>1</b>	<b>INTRODUCTION</b>	<b>15</b>
	1.1 Background	15
	1.2 Aims and Objectives	16
	1.3 Problem statement	16
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>18</b>
	2.1 Internet of Things	18
	2.2 Network connectivity of devices	18
	2.2.1 Wi-Fi	19
	2.2.2 Zig-Bee	19
	2.2.3 Bluetooth	19
	2.3 Software	20
	2.3.1 FreeRTOS	21
	2.3.2 Contiki	21

2.4	Hardware	22
2.4.1	ESP8266	22
2.5	IOT protocol	22
2.5.1	MQTT	22
2.5.2	CoAP	24
2.6	LAMP server	24
<b>3</b>	<b>METHODOLOGY</b>	<b>25</b>
3.1	System Framework	25
3.2	Hardware	26
3.2.1	SoC ESP8266EX	27
3.3	Cloud	30
3.3.1	HTTP server	30
3.3.2	Database	31
3.3.3	MQTT broker/ Web services	31
3.4	Block diagram	32
<b>4</b>	<b>Result and Discussion</b>	<b>34</b>
4.1	Testing and Background setting	34
4.1.1	ESP8266 Module	34
4.1.2	ESP8266 Firmware test	35
4.1.3	LAMP server configuration and testing	35
4.2	User Interfaces	41
4.2.1	ESP8266 Interfaces	41
4.2.2	Server Interface	43
4.2.3	Performance	45
4.3	Discussion	47
4.3.1	HTTP	47
4.3.2	Authentication and Authorisation	48
<b>5</b>	<b>Conclusion and Recommendations</b>	<b>49</b>
5.1	Conclusion	49
5.1.1	Personal Breakthrough	50



5.2	Recommendations	50
5.2.1	MQTT in endpoint devices	50
5.2.2	MQTT security protocol	51
5.2.3	Email identification	51
5.2.4	AP mode of ESP8266	51
<b>REFERENCES</b>		<b>52</b>
<b>APPENDICES</b>		<b>54</b>

**LIST OF TABLES**

<b>TABLE</b>	<b>TITLE</b>	<b>PAGE</b>
Table 2.1:	Comparison between Wi-Fi, Zig-Bee and Bluetooth	20
Table 2.2	Control packet types	23
Table 4.1	Predefine order to enter flash mode	35
Table 4.2	Page Loading Benchmark	45
Table 4.3	Time taken and throughput benchmark	47
Table 4.4	Difference between GET and POST method	48

## LIST OF FIGURES

<b>FIGURE</b>	<b>TITLE</b>	<b>PAGE</b>
Figure 3-1:	Proposed IOT framework	26
Figure 3-2:	Embedded device system block diagram	27
Figure 3-3:	ESP8266EX block diagram (Inc, 2016)	28
Figure 3-4:	System architecture of Xtensa LX (Tensilica Xtensa CS451, 2005)	29
Figure 3-5:	Framework of Cloud	30
Figure 3-6:	Procedure in block diagram	33
Figure 4-1	Schematic diagram ESP8266-01	34
Figure 4-2	Screenshot of UART terminal	35
Figure 4-3	Screenshot of Ping test	36
Figure 4-4	Screenshot of info.php	37
Figure 4-5	snippet of invalid protocol	37
Figure 4-6	Screenshot of MQTT version	37
Figure 4-7	Screenshot of PPA website	38
Figure 4-8	Screenshot of Python script result	38
Figure 4-9	Screenshot of MQTT server connection result	39
Figure 4-10	Screenshot of database tables and data	40
Figure 4-11	Database block diagram	41
Figure 4-12	Screenshot of initialization of ESP8266	42

Figure 4-13 Screenshot of ESP8266 Interface	42
Figure 4-14 Snippet of Python script with MQTT	43
Figure 4-15 Screenshot of login page	43
Figure 4-16 Screenshot of registration form	44
Figure 4-17 Screenshot of user webpage	45
Figure 4-18 Screenshot of benchmark tool running in CMD	46

**LIST OF SYMBOLS / ABBREVIATIONS**

IOT	Internet of Thing
CoAP	Constraint Application Protocol
RTOS	Real Time Operating System
LAMP	Linux, Apache, MySQL, PHP/Python
PPA	Personal Package Archives
MQTT	MQ Telemetry Transport
HTTP	Hypertext Transfer Protocol
HTML	Hypertext Mark-up Language
RFID	Radio-frequency Identification
NFC	Near Field Communication
IP	Internet Protocol
REST	Representational Safe Transfer

**LIST OF APPENDICES**

<b>APPENDIX</b>	<b>TITLE</b>	<b>PAGE</b>
APPENDIX A:	Graphs	54
APPENDIX B:	Computer Programme Listing	56

## CHAPTER 1

### INTRODUCTION

#### 1.1 Background

Internet now not only connects people but things. Internet of Things (IOT) is a device with computational intelligence which comes with network connection. Computational intelligence can be an embedded system that consists of electronics, micro-controller/processor, software and sensors. With the help of these components and network connectivity, data are connected and exchanged.

In the recent years, there was an enormous demand for portable health monitoring devices. These devices are come in the form of wrist band, watch and necklace which are able to measure heart rate, blood pressure, glucose level and so on. With these devices, daily behaviour and health status can be tracked and monitored by installing certain Apps in the smart phone. The Apps will collect and store the data in the memory and form statistical information to users.

With the Internet of Things, devices can be connected and communicated over the internet. The devices can be connected to a router which connected to the internet or form a network among themselves with the help of sensor network like Zig-Bee, RFID, Bluetooth or NFC. Data that produced by these sensor network will be sent to the local server to become information and then to the internet. This information can be accessed from a desktop, smart devices or any equipment that is connected to the internet in real time.

By applying the Internet of Things, health monitoring system, home automation system can be implemented. The health monitoring devices are connected to the internet to reduce the efforts of doctor in rural areas; patients' status can be directly monitored from the central hospital. The improved system can reduce the virtual distance between urban area and rural area, the patients in rural area can get better healthcare services with the help of Internet of Things devices. The embedded devices can be implemented into smart gadget to monitoring home status, home automation and luminaire control. The power consumption, security switch, camera and a personal status can be monitored through the system. With the help of internet of thing concept, it can become a solution of smart home.

## **1.2 Aims and Objectives**

The objectives of this project are

1. To identify the most suitable wireless module for IoT.
2. To design services and setup server to handle data and monitor, manage information from the IOT devices.
3. To test the reliability and the performance of the proposed system.

## **1.3 Problem statement**

In the recent years, the concept of internet of things is going into the technology vision in the whole wide world due to the convergence of multiple technologies such as RFID, low energy wireless IP networks, advance computational power in embedded system, light-weighted internet protocol and so on. Vast number of devices are enabled and connecting to the internet and the number is rapidly growing in next few years.



The IoT development is affected by two aspects, which are the market and the technology. Several countries have started to embark and develop the IoT system implementation into the projects that are focus in water, energy, building, transportation and so on. The IoT opportunities bring an enormous global economic value to the countries. Malaysia is just starting to take part in the IoT solution implementation.

However, several challenges are stated to unleash the full potential of IoT besides leveraging on existing technology strength in Malaysia. From the view of technology, Malaysia has challenges to embark the IoT implementation with high technology complexity, difficulty to replace the legacy systems that being used in Malaysia and data accessibility and knowledge sharing availability in recent system. ("National Internet of Things (IoT) Strategic Roadmap: A Summary", 2015)

With the presence of wireless devices that available in market, robust and complex machine to machine communication protocols that available in open source community and complex operating software for wireless devices. An IoT system framework is formed to lower down the barrier for IoT implementation in Malaysia which can provide a complex system, compatible or easy to replace for legacy system and increase the data accessibility and availability.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Internet of Things**

The internet of things (known as IOT) is the network formed by physical objects such as embedded devices on vehicles, buildings and other applications. The devices must have some basic requirements to be considered as IOT devices which are software, sensors and network connectivity. These enable the objects to collect and exchange data, share the data to the network.

The internet of things allows objects to be accessed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer systems to improve accuracy, efficiency and economic benefit.(Vermesan & Friess, 2013)

#### **2.2 Network connectivity of devices**

The IOT devices must have an ability to connect to network or form a personal area network (PAN) by themselves, this feature is usually performed by using wireless communications. In market, the devices are designed to come with a wireless module and the modules have different spectrum and protocols such as Wi-Fi, Zig-Bee, Blue-tooth and others.

Due to the limitation of the power supply of the IOT devices, high power consumption wireless protocol such as Wi-Fi and Bluetooth are not suitable and not allow the device to run in long term. Low power versions of the protocols are then developed to compete in the IOT market.

### **2.2.1 Wi-Fi**

Wi-Fi is a wireless technology that intends for computer to computer communication. It has a higher data rate compare to others. It allows maximum number of 2007 devices to form a network. It provides a stable connection and complexity for different devices and applications. The disadvantage is it has a large power demand and not suitable for power limit devices.(Lee, Su, & Shen, 2007)

### **2.2.2 Zig-Bee**

Zig-Bee is a wireless protocol that special designed for the internet of thing devices that have high mobility and monitoring or measuring simple parameter. It has lower data rate compare to Wi-Fi. It allows large number of devices connected to the network, the Zig-Bee protocol provides the devices can form different network topology such as mesh, star and cluster tree topologies, the connection is not as stable as Wi-Fi. The Zig-Bee is widely used for internet of thing devices application due to its extremely low power consumption.(Lee et al., 2007)

### **2.2.3 Bluetooth**

Bluetooth is frequently used as wireless protocol for file transfer between two mobile phones; its data rate is between Wi-Fi and Zig-Bee. Bluetooth can only allow 8 devices to form a network and has a stable connection. It provides the Piconet and

scatternet topologies to the devices which are working in master slave mode. The power consumption of Bluetooth is as low as Zig-Bee. It is widely used as wearable devices with IOT features that can be connected to mobile phone. Bluetooth provides a large number of RF channels that allow many users in same place without interference. (Lee et al., 2007)

Table 2.1: Comparison between Wi-Fi, Zig-Bee and Bluetooth

<b>Standard</b>	<b>Wi-Fi</b>	<b>Zig-Bee</b>	<b>Bluetooth</b>
<b>IEEE spec.</b>	<b>802.11a/b/g</b>	<b>802.15.4</b>	<b>802.15.1</b>
<b>Max Signal rate</b>	<b>54Mb/s</b>	<b>250Kb/s</b>	<b>1Mb/s</b>
<b>Number of RF channels</b>	<b>14</b>	<b>16</b>	<b>79</b>
<b>Topology</b>	<b>BSS, ESS</b>	<b>Star, Cluster tree, Mesh</b>	<b>Piconet, Scatternet</b>
<b>Number of cell nodes</b>	<b>2007</b>	<b>65000</b>	<b>8</b>
<b>Nominal range</b>	<b>100</b>	<b>10-100</b>	<b>10</b>

### 2.3 Software

Software is playing a main role for the internet of thing devices, most of the devices are operating in real time and doing more than one task, a light weighted real time operating system (RTOS) is ported into the devices to let them operate multi-task or multi-thread in real time. The IOT devices need a capable system to manage the network protocol, transmitting and receiving information while doing a certain task.

There are a lot of open source RTOS which are suitable for IOT devices that can be obtained from internet. Some elements are considered when choosing a RTOS for the devices such as scalability, modularity, connectivity and reliability. For scalability, a developed software may not only run in a certain type of device, the software must have an ability to run in different devices without a huge modifying in code or effortless when porting to other device.

For modularity, the RTOS must be able to fit in the devices while most of the IOT devices have small RAM and ROM memory. Sometimes, the devices have very low processing power and cannot perform complex calculation. RTOS must have a simple structure in order to schedule the task without giving too much effort to the devices.

The operating system also needs to provide the ability to maintain the network protocol to remain connectivity and provide a stable and safety system in normal condition for reliability. (Milinković, Milinković, & Lazić)

### **2.3.1 FreeRTOS**

FreeRTOS is a real time operating system that is designed to run on microcontroller platform. It has a small ROM and RAM size requirement and has few operating modes. It is supported by different communities and able to port and run in various devices. It is widely used as RTOS of MCU devices. The disadvantage is FreeRTOS does not have a mechanism to avoid priority inversion which may increase the effort for software development. (Milinković et al.)

### **2.3.2 Contiki**

Contiki OS is not a real time operating system, due to its system structure, it is able to perform similar performance compared with RTOS. The simple system structure is also used in MCU platform and slowly become dominant in market. The OS has implemented IPv6 stack for IEEE 802.15.4 specification and support different type of application protocol such as Constrained Application Protocol (CoAP) and light weighted M2M protocol (Lwm2m). The simplicity to form IOT structure and hardware network simulation are the main reason to make it popular. (Österlind, 2006)

## **2.4 Hardware**

### **2.4.1 ESP8266**

ESP 8266 is a system on chip mounted with Wi-Fi function. The chip is special designed for internet of thing applications. It is usually function as a Wi-Fi module; it is flashed with AT command firmware while the chips are manufactured. Users can directly use the module by giving the AT-command through serial communication. The AT-command has provided a simple function to become a Wi-Fi module and perform connection with host or client.

The ESP 8266 is designed that it can be flashed in firmware easily; users can flash in specific firmware into the module and become a standalone device, which is independent from AT commands. The open sourced code for internet of thing application can be obtained from the communities, which have implemented the IOT protocols such as MQTT, CoAP, and Lwm2m.

## **2.5 IOT protocol**

The IOT protocol provide the devices and cloud in the platform able to communicate to each other. Using the standard protocol may increase the interoperability to the devices, the devices can exchange data without any translator or interpreter in between. Various IOT specific communication protocols are designed to overcome the interoperation problem.

### **2.5.1 MQTT**

MQTT is called as Message Queue Telemetry Transport, it is a messaging framework that designed by the IBM and Eurotech. The MQTT is designed to be light-weighted for the client and there is a MQTT broker which is running as a service provider. The client can subscribe or publish data with a reference topic. Once a client publishes a data to MQTT broker, the broker copies the data to the subscribed client with a certain topic. So it is a publish/subscribe model where it distinguishes the client with topic.

The protocol of MQTT is working behind with a series of exchanging MQTT control packet in a pre-defined way. The MQTT control packet consist of three elements which are fixed header, variable header and payload. The fixed header is the specific header that is used to distinguish type of MQTT control packet. The variable header is used to send MQTT packet identifier to identify the packet types with other packets. The payload is used to store data, payload may not necessary in some situation. ("MQTT Version 3.1.1", 2015) The type of control packets for fixed header is listed in table 2.2.

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

**Table 2.2 Control packet types**

### **2.5.2 CoAP**

CoAP is called as constraint application protocol, the protocol is transferring messages with UDP layer. The protocol is designed for machine to machine applications, the devices can directly communicate to each other by using one of their URL. Clients can access to the resources or data by using the REST model like method such as GET, PUT, POST and DELETE.

### **2.6 LAMP server**

LAMP is a software bundle that is used to become the web service solution stack. The LAMP is named as an acronym of four open source components, which is Linux operating system, Apache web server, MySQL database and PHP server side scripting language. The combination of the software is used to build a dynamic website and web applications.



## **CHAPTER 3**

### **METHODOLOGY**

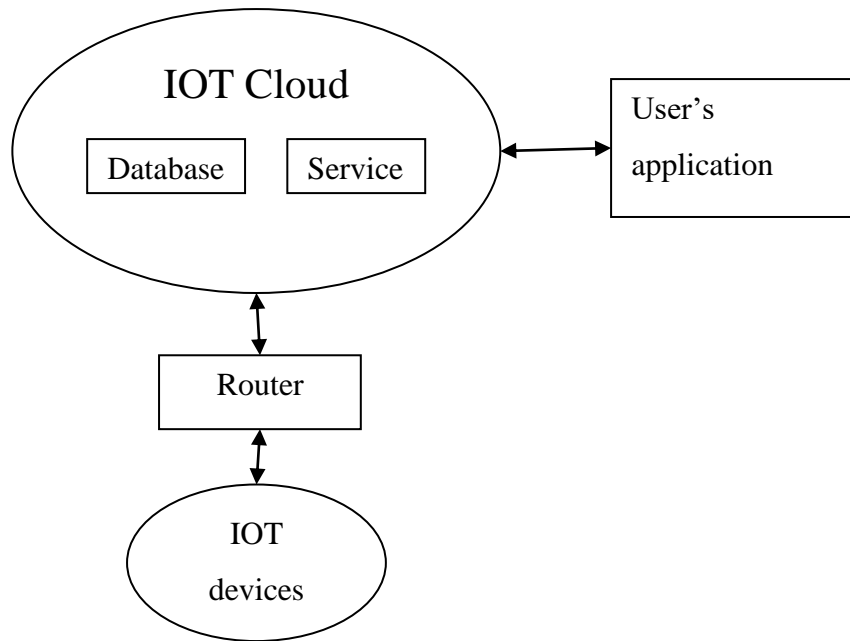
#### **3.1 System Framework**

The framework of the Internet of Things is decided by the types of application, such as REST, MQTT, XMPP architectures. The proposed application framework of IOT consists of a main server that keeps subscribing the devices information, the router act as an interpreter between the server and IOT devices. The proposed IOT framework is shown in figure 3.1.

In the proposed application, the IOT devices are connected to the server with the help of router. The router acts as a gateway to connect the IOT devices to the Internet and allows these devices to communicate with the server.

The server which is usually called cloud provides services to the users. The obtained data from the devices are stored in the database of the server.

Users can access the cloud server through the user application; the application provides an interface between the cloud and the user. This means the application acts as a middleware between the cloud and user.



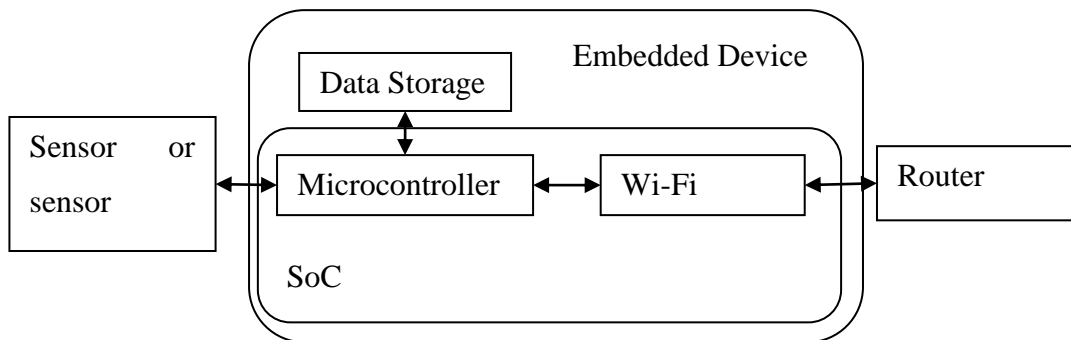
**Figure 3-1: Proposed IOT framework**

### 3.2 Hardware

In order to achieve a power efficient and feasible monitoring system, an intergrated Wi-Fi chip is selected. The chip should interpret with Wi-Fi module, micro-controller, filters, amplifiers and power management modules. The chip should small enough to fit into the proposed system. Based on these criteria, ESP8266EX by Espressif system is selected. This chip will be further discussed in section 3.2.1.

In the proposed application, the ESP8266 chip collects data from sensor and processed by the microcontroller. In order to ensure devices to communicate efficiently, a light weighted protocol is used in the application. The light weighted implemented protocol is MQTT. For example, the ESP chip is designed to publish the data periodically to the broker. The broker will send the data to the related topic subscribed client.

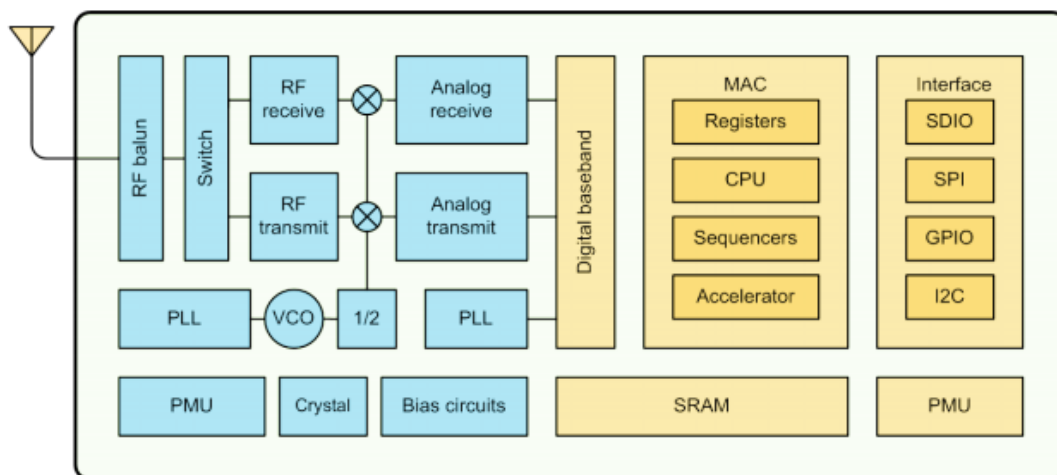
The ESP chip and other components are used to form the IOT devices. The components consist of LED, USB to UART converter, button, switches and EEPROM. The EEPROM flash acts as a memory device to the ESP chip and used to store the firmware of IOT devices. The IOT device in block diagram is shown as figure 3.2.



**Figure 3-2: Embedded device system block diagram**

### 3.2.1 SoC ESP8266EX

The system on chip (SoC) ESP8266EX is selected as the core element of the embedded device. The ESP8266EX provides a microcontroller (3.2.1.1) and Wi-Fi (3.2.1.2) on a single chip. The system block diagram of ESP8266EX is shown as figure 3.3.



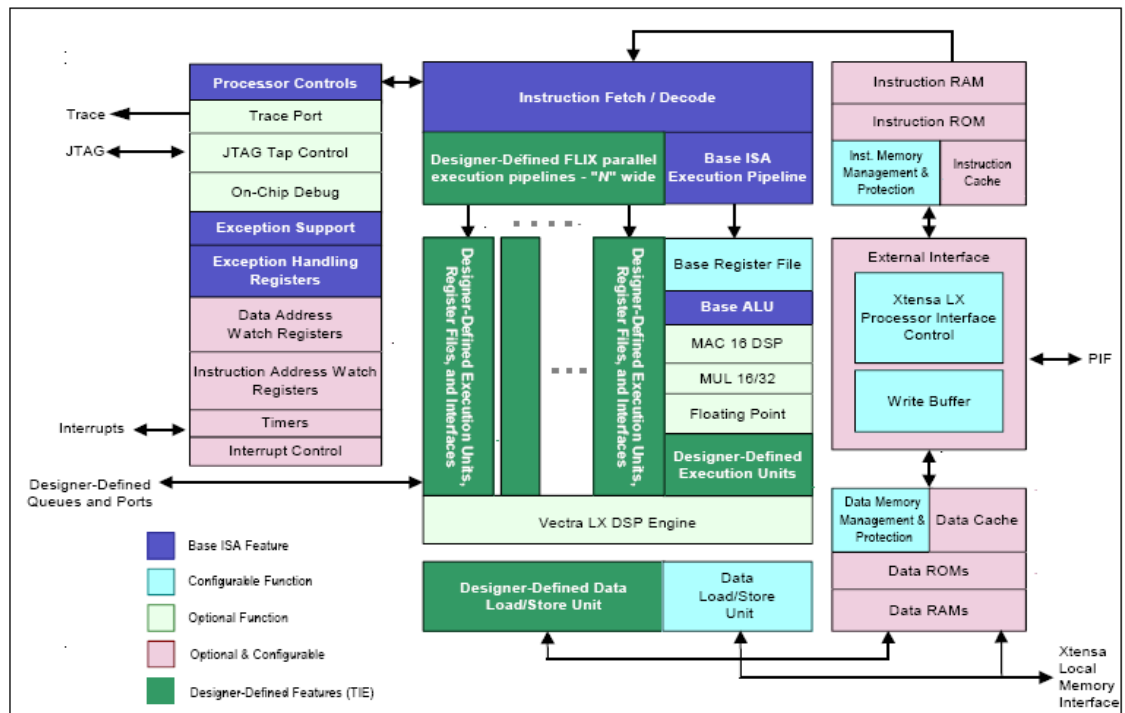
**Figure 3-3:ESP8266EX block diagram (Inc, 2016)**

The ESP8266EX integrates the enhanced version of Tensilica's LX106 diamond series 32 bit MCU with on chip SRAM which is the MAC part on the block diagram. The ESP8266EX has 32 pins, the interfaces of the SoC are 16 GPIO pins, I2C, SPI, on pin I2S, UART and a 10bit ADC. The ESP8266EX does not have flash memory for user and operate at 3.3V. External flash and power regulator is required to operate this chip.

The ESP8266EX also interpret the antenna switches, RF filter, power amplifier, low noise receive amplifier filter, power management modules in the single chip with the micro-controller.

### 3.2.1.1 Microcontroller

The ESP8266EX integrate TensilicaXtensa LX106, 32bit micro-controller. The controller is using 16bit RISC with Harvard Architecture and operates at 80MHz frequency. The design architecture of the Xtensa LX MCU is shown as figure 3.4.



**Figure 3-4: System architecture of Xtensa LX (Tensilica Xtensa CS451, 2005)**

The system has 80 core instructions set and the instructions are in 16bit and 24bit. It contains 64 general purpose physical registers and 6 special purpose register. The data memory and instruction memory are separated and the memory sizes are 96KiB and 64KiB respectively.

### 3.2.1.2 Wi-Fi module

The Wi-Fi module is integrated in the chip and using 2.4GHz band with WPA/WPA2 support. The specifications are 802.11 b/g/n, integrated TCP/IP protocol stack, TR switch, balun, LNA, power amplifier and matching network. It also has phase locked loop (PLL), regulators and power management unit (PMU) for wireless signal capture. It needs an external antenna to capture wireless signal.

### 3.3 Cloud

Cloud computing provides a platform to perform the computer task for the users which is connected to internet. It is the setup to provide services to the user and devices. So, the cloud and internet of thing are inseparable. It gives a great advantages when involve large amounts of data, where cloud computing has virtually unlimited storage.

On the cloud side, a PC which is connected to the internet, it is setup to provide service to the user and devices. The PC is running with Linux OS and executing some software, the software is web server software, database software, python server and so on. The framework of the cloud is shown in figure 3.5.

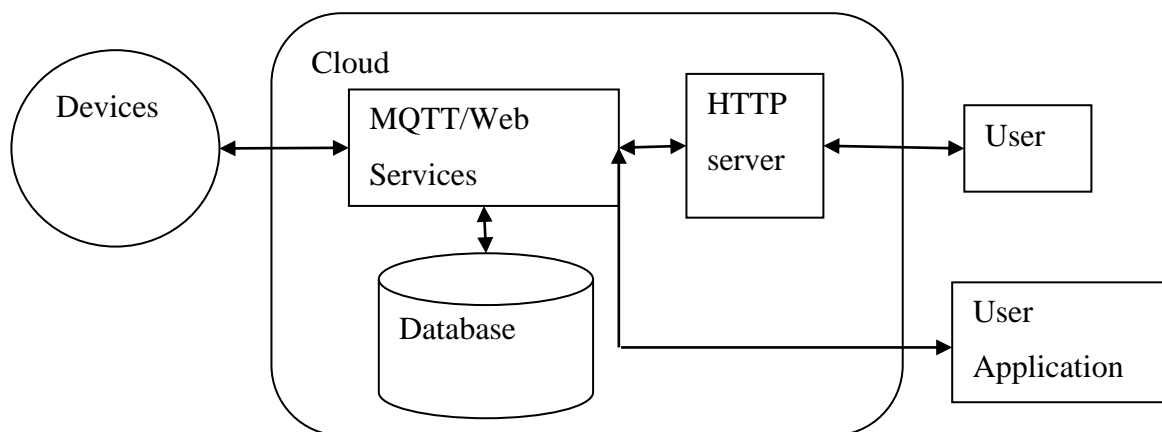


Figure 3-5: Framework of Cloud

#### 3.3.1 HTTP server

HTTP server provides an interface between user and cloud, the software is Apache web server. The software allows the PC to work as a HTTP server, handle request from the user and send back webpage, the user can visit the cloud through a web browser.

### **3.3.2 Database**

Database will be used to store the data upload from devices and store the analytical data for user. The database software is MariaDB. The MariaDB is an open source database framework that provide the feature same as MySQL. The client software like phpMyAdmin is used to configure, setting and organize the structure of the database.

### **3.3.3 MQTT broker/ Web services**

The web services in cloud are done by using python and MQTT broker, the web service will become an interpreter between database, devices and HTTP server. The web service also processes the data into analytical data and become statistical information for user.

In this proposed application, the MQTT is chosen between various lwm2m protocols such as Co-AP, REST and so on. In this project, the IoT devices are assumed to be run on a stable environment, the devices are able to connect to the internet with a Wi-Fi router through internet service provider. The Wi-Fi system used in normal operation is not lossy network available. So, the MQTT that rely on TCP is more reliable than CoAP (UDP), the TCP provide a safer and loss-proof environment to the transferred packet.

Besides that, the architecture of the MQTT is publish-subscribe model, while the CoAP is request response or resource observe model. The publish-subscribe model has more advantages on this proposed application as the user application and other devices can subscribe to same topic to get the certain message, the message queue structure is well developed as a part of MQTT function. While using CoAP, the message queue structure is needed to be developed in the python services.

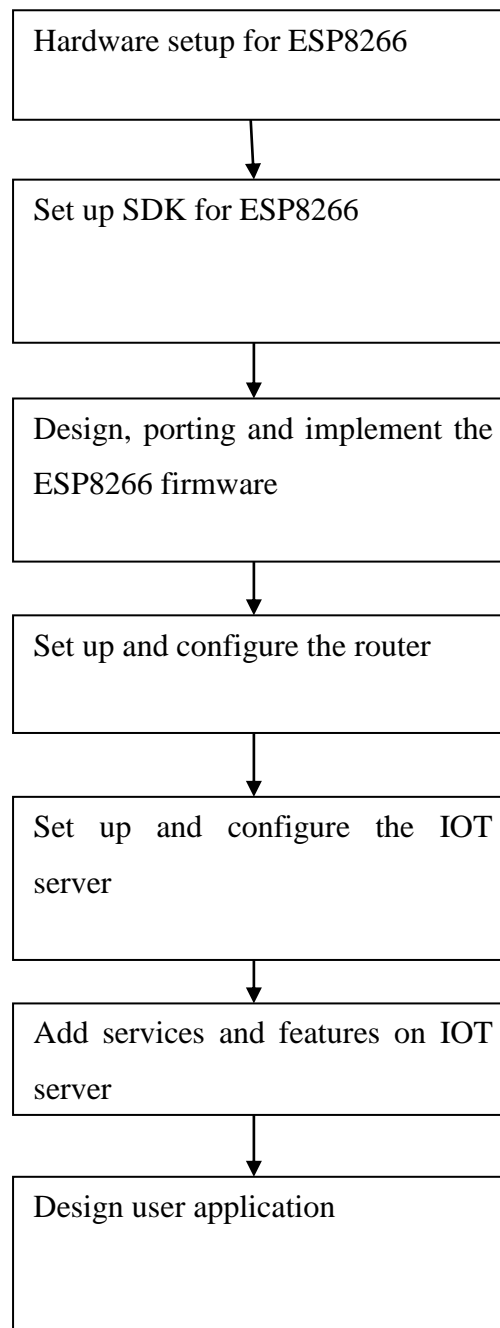
### **3.4 Block diagram**

The interface of ESP8266 is designed with LED, a few push buttons and USB to serial converter. With these components, the ESP8266 chip is able to communicate with the PC. This is followed by the deployment of the SDK in PC. Here, the firmware is designed and developed.

On the server side, the router is setup to direct the incoming request to the server port. The LAMP server is used to run IOT services with python script. The database, http server and PHP services are activated in the LAMP server. After setting up, IOT services can be designed using python accordingly.

An android app is developed to access through the database of the server.





**Figure 3-6: Procedure in block diagram**

## CHAPTER 4

### Result and Discussion

#### 4.1 Testing and Background setting

##### 4.1.1 ESP8266 Module

In order to test the functionality of ESP8266 module, the testing circuit is constructed. The testing circuit is shown in figure 4-1. The circuit consists of two push buttons which use in firmware flashing in a predefined order as shown in table 4.1.

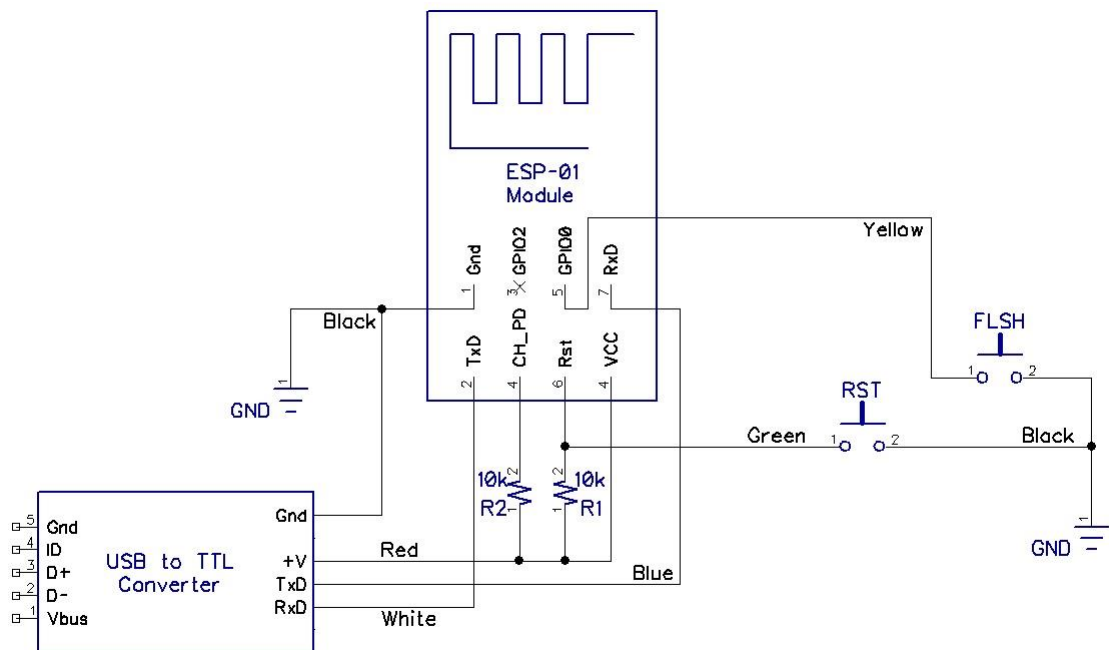


Figure 4-1 Schematic diagram ESP8266-01

Stages	1	2	3	4
Button order	Hold RST	Hold FLSH	Release RST	Release FLSH

**Table 4.1 Predefine order to enter flash mode**

### 4.1.2 ESP8266 Firmware test

The ESP8266 is flashed by using the communication through USB port. A USB to UART converter is used to convert the signal between UART and USB. The UART function is tested with windows' serial COM software. Other wireless functions are also tested such as Wi-Fi host, Wi-Fi client, HTTP get and post. The screenshot of ESP8266 with scan network function and the messages were shown on the UART terminal is shown in the figure below.

```
Scanning available networks...
** Scan Networks **
number of available networks:4
0) 3338 Signal: -36 dBm Encryption: Auto
1) AP_502773602 Signal: -88 dBm Encryption: WPA2
2) jasonyee Signal: -93 dBm Encryption: WPA2
3) 3338 Signal: -65 dBm Encryption: WPA
```

**Figure 4-2 Screenshot of UART terminal**

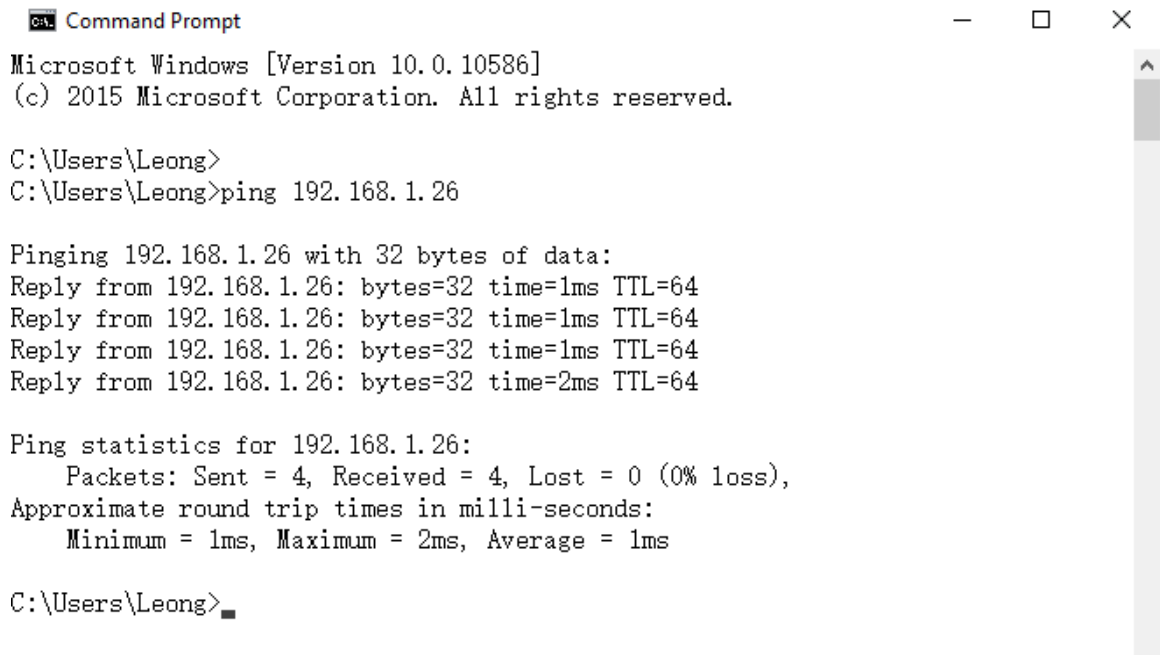
### 4.1.3 LAMP server configuration and testing

The operating system Linux Ubuntu is chosen and configured as LAMP server. There are several packages needed for the LAMP server configuration process. For example, apache web server package and MariaDB package. The python and Php are installed in the system and act as communication tools between the LAMP software.

#### 4.1.3.1 Server Ping Test

To ensure the Apache server is working well, a server ping test is done from another PC in LAN. The ping function in another PC is used to test the connection of

the server, the PC is in the same local area network with the server, the PC will receive the reply from server. The server is assigned an address of 192.168.1.26 by the router. The screenshot of ping test is shown as below.

The image shows a screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt". The text inside the window shows the following output:

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Leong>
C:\Users\Leong>ping 192.168.1.26

Pinging 192.168.1.26 with 32 bytes of data:
Reply from 192.168.1.26: bytes=32 time=1ms TTL=64
Reply from 192.168.1.26: bytes=32 time=1ms TTL=64
Reply from 192.168.1.26: bytes=32 time=1ms TTL=64
Reply from 192.168.1.26: bytes=32 time=2ms TTL=64

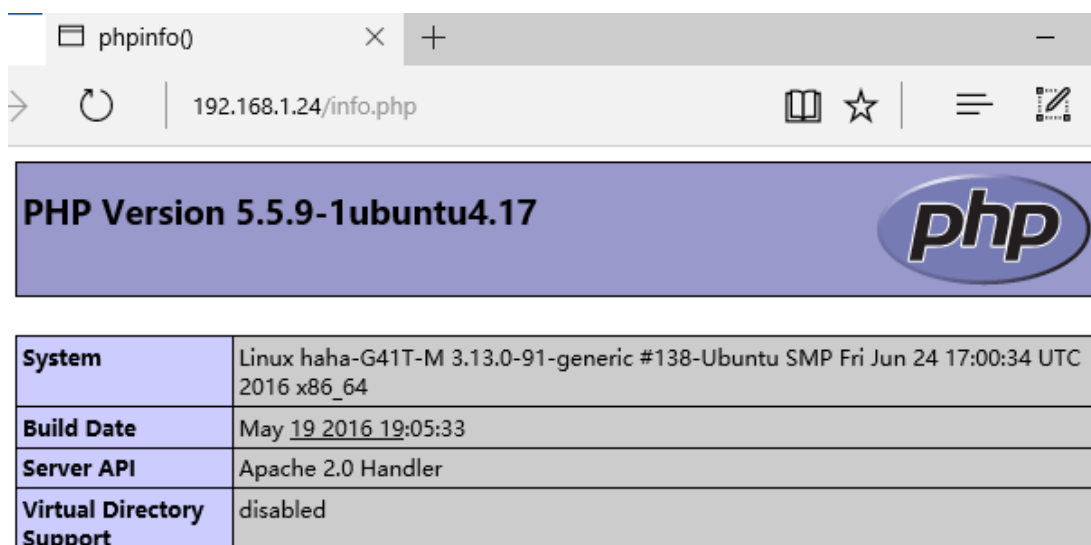
Ping statistics for 192.168.1.26:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms

C:\Users\Leong>
```

**Figure 4-3 Screenshot of Ping test**

#### 4.1.3.2 PHP test

A PHP webpages are created and stored in the server webpages directory. The webpage was accessed from another PC with the web browser. The PHP server read the PHP script in the webpage, transcript the script to HTML type and then sends to the request address. If the PHP is not working, the web browser will receive a blank webpage. A function of `phpinfo()` is used in the PHP script for testing purposes. The working screenshot of PHP (`info.php`) is shown in figure 4-4.



**Figure 4-4 Screenshot of info.php**

### 4.1.3.3 MQTT broker setup and testing

In order to verify the compatibility of the python MQTT protocol, ESP8266 protocol and MQTT broker protocol, a MQTT broker service is run in frontend mode. This is shown in figure 4-5.

```
mosquitto[383]: New connection from 127.0.0.1.
mosquitto[383]: Invalid protocol "MQTT" in CONNECT from 127.0.0.1.
mosquitto[383]: Socket read error on client (null), disconnecting.
```

**Figure 4-5 snippet of invalid protocol**

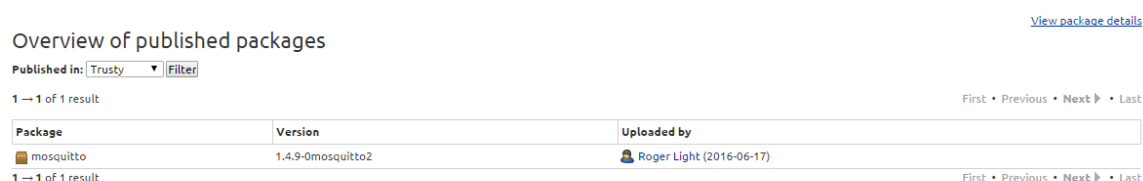
The version of MQTT can be shown by using the Linux command in terminal window. The screenshot of the installed MQTT info is shown in Figure 4-6.

```
$ apt-cache show mosquitto
Package: mosquitto
Priority: optional
Section: universe/net
Installed-Size: 190
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Roger A. Light <roger@atchoo.org>
Architecture: amd64
Version: 0.15-2ubuntu1
```

**Figure 4-6 Screenshot of MQTT version**

In the server side, a python MQTT script is used to test the MQTT server. This is done by connecting to the server and subscribe MQTT topic. The snippet result is shown in figure 4-5. The results indicated incompatible protocol error.



To solve the protocol compatibility problem, the MQTT broker has to be latest version. The “apt-get update” and “upgrade” command is called to update and install all the outdated software that are recorded in the repo. After upgrade, the MQTT broker still remain the old version, it is because the admin of the Linux repo has not updated the PPA to the latest version. From the official MQTT broker website, the latest version is 1.4.9 and there is no binary installation for Linux platform. The only way to get the latest version MQTT broker is to build from its source code. Before building the software, Mosquitto PPA website is visited to ensure the package has been built in the current Linux version of the Server. The figure 4-7 show that the source code of 1.4.9-0 Mosquitto is built by the maintainer of MQTT.



Overview of published packages [View package details](#)

Published in: Trusty

1 → 1 of 1 result First • Previous • Next • Last

Package	Version	Uploaded by
 mosquitto	1.4.9-0mosquitto2	 Roger Light (2016-06-17)

1 → 1 of 1 result First • Previous • Next • Last

**Figure 4-7 Screenshot of PPA website**

After that, the source code of MQTT server is “git clone” from the official Github, and then the code is built by using the “make” command in the source code directory. After compiling, the compiled software is installed to root directory by “make install” command.

The Mosquitto broker server is opened again in the frontend mode and the python script is opened to test the MQTT server connection. The screenshot of the test results is shown in figure 4-8 and figure 4-9.

```
haha@haha-G41T-M:~$ python test.py
Connected with result code 0
```

**Figure 4-8 Screenshot of Python script result**

```
haha@haha-G41T-M:~$ mosquitto
1468322363: mosquitto version 1.4.9 (build date 2016-07-10 19:46:04+0800) starting
1468322363: Using default config.
1468322363: Opening ipv4 listen socket on port 1883.
1468322363: Opening ipv6 listen socket on port 1883.
1468322384: New connection from 127.0.0.1 on port 1883.
1468322384: New client connected from 127.0.0.1 as paho/0F47898F28EC5A9CDD (c1, k60).
```

**Figure 4-9 Screenshot of MQTT server connection result**

From the figure 4-8, the testing python script is connecting successfully to the MQTT broker and a debug message is dumped into the terminal. In figure 4-9, the MQTT broker run in frontend mode is showing that a new client is successfully connected from localhost with port 1833 as a specific client ID.

#### **4.1.3.4 Database configuration**

In this proposed project, a database is created using MySQL command in Linux terminal, the created database is named as secure\_login. Once the database is created, a new database account is created with a limited privilege for web user access. The tables are formed and the data is stored inside the tables, the data includes user information, devices name, topic and collected data. The database and tables are shown in the figure 4-10.

```

+-----+
| Tables_in_secure_login |
+-----+
| data                    |
| devices                 |
| login_attempts         |
| members                 |
+-----+
4 rows in set (0.00 sec)

MariaDB [secure_login]> select * from devices;
+-----+-----+-----+-----+
| device_id | user_id | device_name | topic      |
+-----+-----+-----+-----+
|          1 |         2 | test_device | test_topic |
+-----+-----+-----+-----+
1 row in set (0.01 sec)

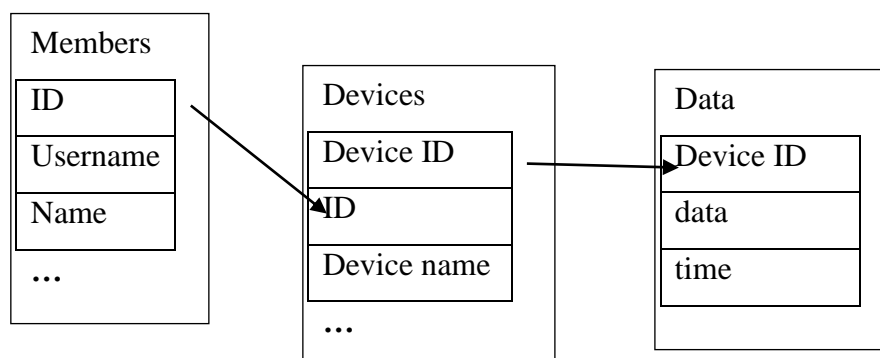
MariaDB [secure_login]> select * from data;
+-----+-----+-----+
| device_id | data | time                |
+-----+-----+-----+
|          1 | 1985 | 2016-07-26 23:44:48 |
|          1 | 1985 | 2016-07-26 23:45:18 |
|          1 |   30 | 2016-07-31 05:58:46 |
|          1 |   30 | 2016-07-31 05:59:16 |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

**Figure 4-10 Screenshot of database tables and data**

From the figure 4-10, the tables in the database are data, devices, login attempts and members. The table login attempts are users' login record that store the user login status to enhance the server security. The members' table is used to store the information of member, such as username, E-mail and hashed password. The table devices record the devices added by the user and the data is the data submitted by the devices. The tables members, devices and data are normalized from each other which is shown as figure 4-11.





**Figure 4-11 Database block diagram**

## 4.2 User Interfaces

In this proposed project, user can login to the server through the web browser, configure ESP8266 to submit data, connect to internet by using the COM terminal. Once, the device is turned on, the user can control the devices, access the information through the interfaces.

The ESP8266 interfaces, Server interface and their procedures are discussed in the section 4.2.1 and section 4.2.2.

### 4.2.1 ESP8266 Interfaces

When the ESP8266 is plugged in to a PC USB port, the COM terminal of the PC is opened, the ESP8266 will show the available networks to user and waiting for SSID and password to login. The starting status of ESP8266 is shown in figure 4-12, there are four Wi-Fi network available to be connected by ESP8266 after performing network scan in this demonstration.

```
Scanning available networks...
** Scan Networks **
number of available networks:4
0) 3338 Signal: -36 dBm Encryption: Auto
1) AP_502773602 Signal: -88 dBm Encryption: WPA2
2) jasonyee      Signal: -93 dBm Encryption: WPA2
3) 3338 Signal: -65 dBm Encryption: WPA
select ssid to connect
Enter password
```

**Figure 4-12 Screenshot of initialization of ESP8266**

After the user keying the SSID and the password, the ESP8266 is connected to the chosen network. Connecting to the network may need for a few seconds.

When connection is established, the IP address of device is shown and start connecting to the MQTT server and a dummy data (humidity) is published periodically in this demonstration. The interface in the COM terminal is shown in figure 4-13.

```
Connecting to 3338

.....
IP address:
192.168.1.8
Attempting MQTT connection...connected
New humidity:30.00
New humidity:30.00
New humidity:30.00
New humidity:30.00
New humidity:30.00
```

**Figure 4-13 Screenshot of ESP8266 Interface**

At the server side, a python script is running on the front mode to subscribe the topic and print onto the terminal for testing purposes. The data obtained is then

stored into the database. A screenshot of the MQTT python script run on terminal is shown as figure 4-14.

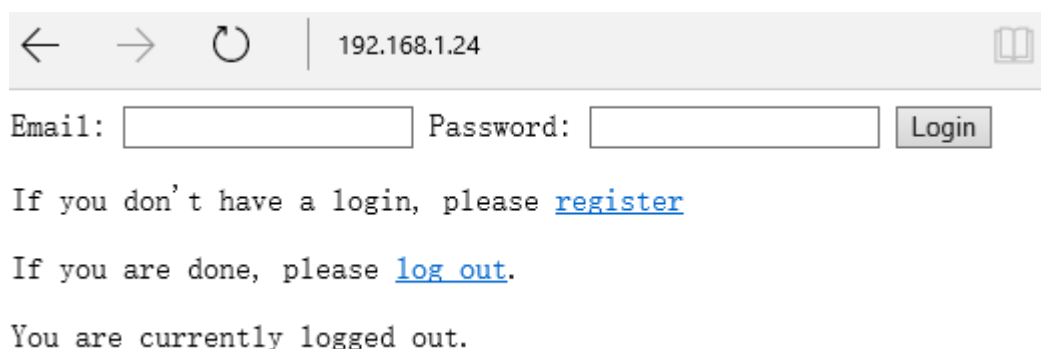
```
haha@haha-G41T-M:~$ python test.py
Connected with result code 0
sensor/humidity 30.00
sensor/humidity 30.00
sensor/humidity 30.00
sensor/humidity 30.00
sensor/humidity 30.00
sensor/humidity 30.00
sensor/humidity 30.00
sensor/humidity 30.00
```

**Figure 4-14** Snippet of Python script with MQTT

#### 4.2.2 Server Interface

In the LAN, a PC is used to browse to the server webpage. The address of the server is entered into the web browser and the login page is appeared. In this project, the PHP hypertext pre-processor is installed above the apache web server to serve the webpages request, therefore all the webpages are PHP files and have the extension of “.php”.

In the main page, there are forms and button to let user to login with their account. Besides that, the account can be registered by clicking the register button. The login status is recorded and is shown on the webpage, user can logout from this webpage. The screenshot of the login page is shown in the figure 4-15.



← → ↻ | 192.168.1.24

Email:  Password:

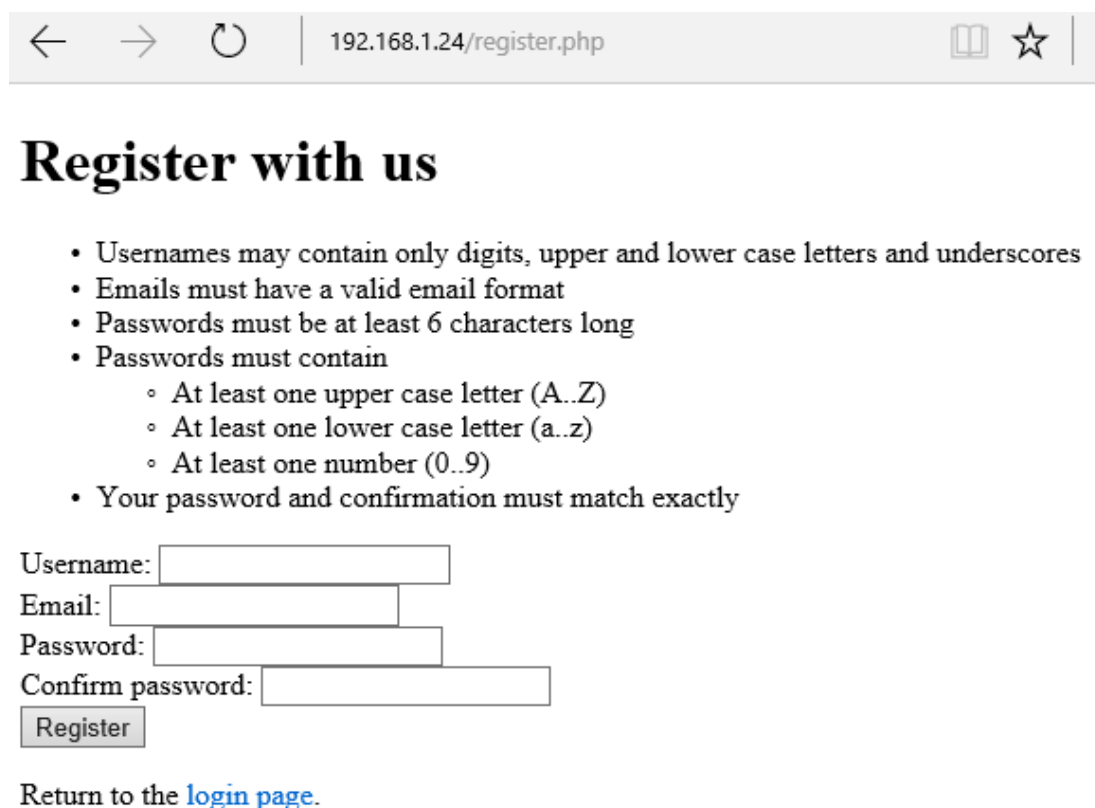
If you don't have a login, please [register](#)

If you are done, please [log out](#).

You are currently logged out.

**Figure 4-15** Screenshot of login page

An account is created in the register page, by clicking the register link in the login page, a register webpage is directed to the user. The screenshot of the registration page is shown in the figure 4-16.



← → ↻ | 192.168.1.24/register.php | 📖 ☆

## Register with us

- Usernames may contain only digits, upper and lower case letters and underscores
- Emails must have a valid email format
- Passwords must be at least 6 characters long
- Passwords must contain
  - At least one upper case letter (A..Z)
  - At least one lower case letter (a..z)
  - At least one number (0..9)
- Your password and confirmation must match exactly

Username:

Email:

Password:

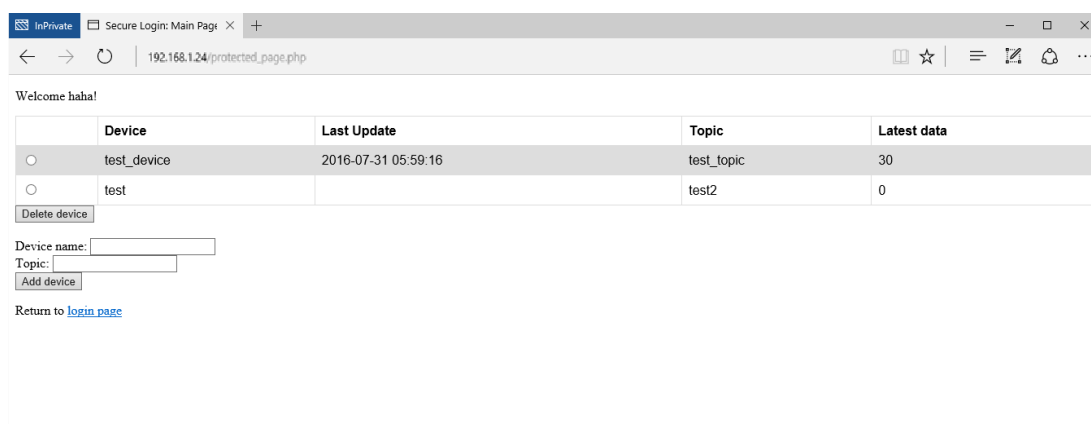
Confirm password:

Return to the [login page](#).

**Figure 4-16 Screenshot of registration form**

An account is used to login to the server, the data is shown in a table of the webpage. The name of device, last updated time, topic and latest submit data is shown in this table. The data is requested from the server database. User can delete the device by clicking the delete button. The user can also add devices by submit the topic and device name to the server.

In figure 4-17, two devices are added and only one topic is submitted by the device, the latest update time and latest data are shown.



**Figure 4-17 Screenshot of user webpage**

### 4.2.3 Performance

The performance of server is tested in this proposed project which includes performance of webpages server and MQTT broker. They are further discussed in section 4.2.3.1 and section 4.2.3.2.

#### 4.2.3.1 Webpages Server

Various factors can affect the performance of a page loading request, one of the important elements to determine the performance is loading time. The time taken of loading the webpages is tested by Firebug. The Firebug is a useful tool for web development which can track the page load time with in depth details of loading process. The data obtained is tabulated in the table 4.2.

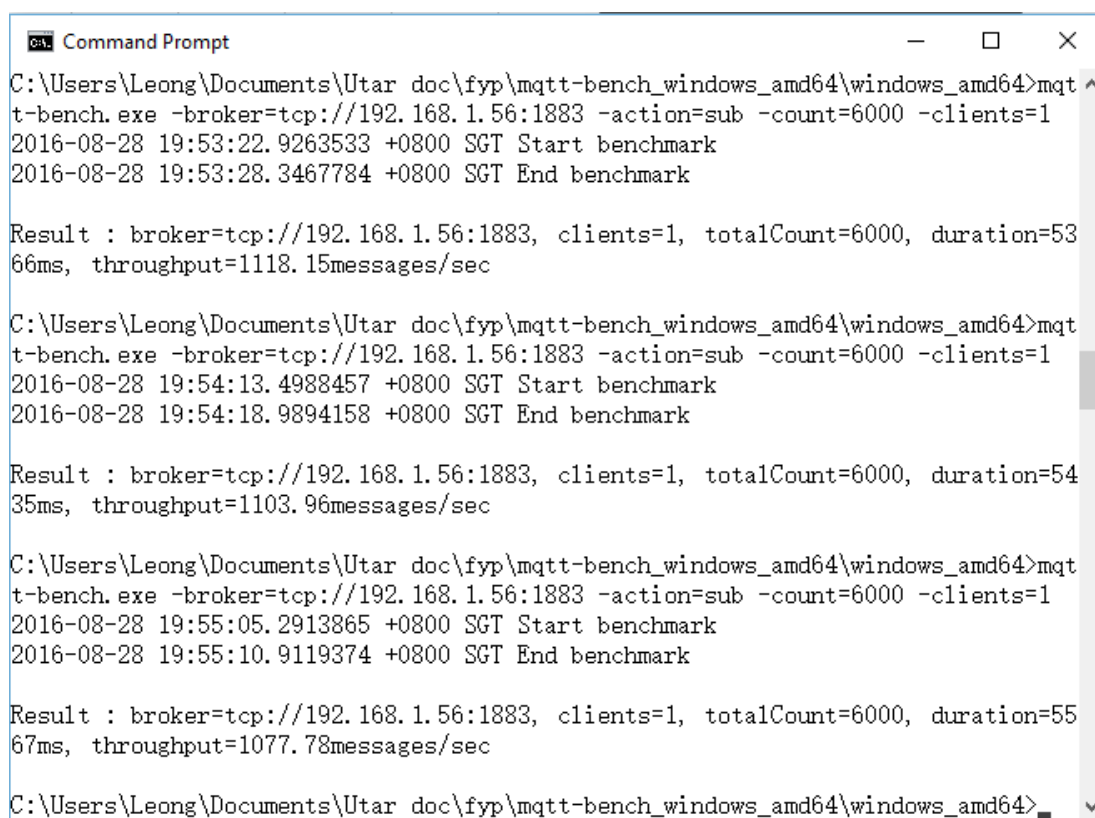
Page	Number of request	Total document size (kB)	Total loading time (ms)
Main page	4	7.1	94
Registration page	4	7.4	49
User page	2	0.651	36
Add devices	1	0	91
Logout	1	0	1

**Table 4.2 Page Loading Benchmark**

From the results in table 4.2, few webpages have zero document size. It is because a PHP technique that perform web service as a webpage is used in this proposed project and the webpage itself is acting like a request. When the browser is requesting for the webpage, the webserver send nothing but a final data and redirecting hyperlink. The loading time used by the zero size webpage is the time taken for the server to process the data and perform services.

### 4.2.3.2 MQTT broker

An open source benchmark tools is used to test the performance of the MQTT broker. The tool is designed to estimate the throughput of the MQTT broker. ("takanorig/mqtt-bench", 2015)



```

ca. Command Prompt
C:\Users\Leong\Documents\Utar doc\fyp\mqtt-bench_windows_amd64\windows_amd64>mqtt-bench.exe -broker=tcp://192.168.1.56:1883 -action=sub -count=6000 -clients=1
2016-08-28 19:53:22.9263533 +0800 SGT Start benchmark
2016-08-28 19:53:28.3467784 +0800 SGT End benchmark

Result : broker=tcp://192.168.1.56:1883, clients=1, totalCount=6000, duration=5366ms, throughput=1118.15messages/sec

C:\Users\Leong\Documents\Utar doc\fyp\mqtt-bench_windows_amd64\windows_amd64>mqtt-bench.exe -broker=tcp://192.168.1.56:1883 -action=sub -count=6000 -clients=1
2016-08-28 19:54:13.4988457 +0800 SGT Start benchmark
2016-08-28 19:54:18.9894158 +0800 SGT End benchmark

Result : broker=tcp://192.168.1.56:1883, clients=1, totalCount=6000, duration=5435ms, throughput=1103.96messages/sec

C:\Users\Leong\Documents\Utar doc\fyp\mqtt-bench_windows_amd64\windows_amd64>mqtt-bench.exe -broker=tcp://192.168.1.56:1883 -action=sub -count=6000 -clients=1
2016-08-28 19:55:05.2913865 +0800 SGT Start benchmark
2016-08-28 19:55:10.9119374 +0800 SGT End benchmark

Result : broker=tcp://192.168.1.56:1883, clients=1, totalCount=6000, duration=5567ms, throughput=1077.78messages/sec

C:\Users\Leong\Documents\Utar doc\fyp\mqtt-bench_windows_amd64\windows_amd64>

```

**Figure 4-18 Screenshot of benchmark tool running in CMD**

From figure 4-18, the benchmark tools application is running in the Windows' CMD console. The command is inserted with the application and the results are shown out after few seconds.

A flood test is performed with one subscribe one publish. The publisher is continuously publishing 1kB data to flood the MQTT broker. On other hand, a subscriber is subscribing all the data from the MQTT broker. The test is performed with an increasing count of data, the data is tested with multiple times and the average data is tabulated in table 4.3.

Count	Publish		Subscribe	
	Time taken (ms)	Throughput (Messages/s)	Time taken (ms)	Throughput (Messages/s)
1000	90.333	11245.4	4217	237.333
2000	242.666	8494.223	4608.333	436.5967
3000	461	6758	4871.333	616.4567
4000	642	6367.037	4787	835.6433
5000	1052	4900.183	5195.333	926.4833
6000	1006.33	5967.83	5456	1099.963

**Table 4.3 Time taken and throughput benchmark**

The obtained data is plotted in graph 5-1 and graph 5-2 as shown in Appendix-A.

### **4.3 Discussion**

#### **4.3.1 HTTP**

GET method and POST method are the most frequently used request method in HTTP. The request method is used to transfer data between server and client. The GET method is requesting data from a specified resource and POST method is submitting data to be processed. The differences between these two methods are shown in the table 4-4.

	GET	POST
Reload/Back button	Harmless	Data will be resubmitted
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Restriction on length	Yes	No
Data type	ASCII only	No restriction
Data visibility	Data is shown in URL	Data is not displayed

**Table 4.4 Difference between GET and POST method**

From the table 4-4, the HTTP GET appends all the data to the end of the request URL, limited length and lower power of security when sending data to the server. The method has limitation when sending important info between the server and the client.

HTTP POST is also used to transmit data from web browser to web server. There is no limitation in transmit length at the browser side. The method is used as the data submit from user to server in this proposed project.

### **4.3.2 Authentication and Authorisation**

In this project, users are required to login by entering the login form in login page. The entered E-mail and password are required, in order to be authenticated by the server. When user load the page, a JavaScript is run on the client side, the JavaScript is used to hash the user password using MD5 algorithm. The hashed string will be sent back to the server for verification.

Once the hash is matched, the server will send the user page to the client, and the login attempt and session of the user is recorded. If the hash string does not match, the server will redirect user back to the login page again.



## CHAPTER 5

### Conclusion and Recommendations

#### 5.1 Conclusion

The goal and objectives of the proposed project are achieved. The ESP8266 wireless module is programmed to become an IOT system endpoint devices and data is submitted to the server. At the server side, software is setup and the services are designed to handle the submitted data from the endpoint devices. User can access to the server through HTML.

In this proposed project, the MQTT protocol is used to send and receive the data between users and devices. The broker is required to run on the server in order to execute the protocol. The MQTT broker is built from the latest source code which is from the Git repository of Mosquitto and the source code version is version 1.4.9, the protocol version is version 3.1 and version 3.1.1. The broker is required to be built from the source code, it is because the repository of the software that provided by the Linux Ubuntu is outdated and the protocol version is not version 3.1 and 3.1.1.

The webpages are designed by using HTML and PHP. Most of the information from the database is pre-processed by the PHP before send to the users. The PHP is also used to manage the user login session and form handler. The interfaces of the webpages are designed by using HTML.

### **5.1.1 Personal Breakthrough**

Software and networking knowledge was gained in the process of completing this project. Database design, computer networking and other programming language than C were not introduced in curriculum activities. Database configuration, router port forwarding configuration, PHP webpages design and python programming were performed.

Besides that, I was given an opportunity to design a firmware for a Wi-Fi module. Conventionally, the firmware in the wireless module is fixed and not open to change the firmware, user need to follow the protocol in order to use the wireless module such as AT commands. By using ESP8266, the firmware of the module can be designed by using provided SDK library.

## **5.2 Recommendations**

There are some additional features not included in this project. The features may be added in the future for further system enhancement. The features are listed in the subsection below.

### **5.2.1 MQTT in endpoint devices**

In this project, the ESP8266 can only submit the data with certain topic to the server. In additional feature, user can also submit the signal with certain topic to ESP8266, in order to control the devices, such as input/output control. The ESP8266 can receive the data with certain topic and react correspondingly to the signal data.

The enhancement can be used as home automation system, the ESP8266 is installed in a dimmer circuit, the dimming signal can be sent from the user to control the dimmer circuit.

### **5.2.2 MQTT security protocol**

In this project, the security mode of MQTT broker is disabled. Any devices can submit data to the MQTT broker without any restriction. To enable the security service, user needs username and password in order to submit data with topic.

In this project, user can subscribe any topic they want, this can violate the user privacy. The topic should be personalized and restricted to certain user. The submitted topic needs to be processed with a header to identify the topic belongs to which user to prevent repeated topics and privacy violations.

### **5.2.3 Email identification**

During the registration session, the E-mail is not verified by the server. User can randomly fill in anything to finish the registration session. In this project, an account can be created by submitting any character in the E-mail form.

To further enhance the system, the server may verify the email to prevent the account from being created directly.

### **5.2.4 AP mode of ESP8266**

In this project, the ESP8266 can only be initialized by using a COM terminal. Without a COM terminal, user cannot modify which network to connect and what topic to publish. To solve the problem, when the ESP8266 is not initialized or fails to initialize, the ESP8266 will change to AP mode.

In AP mode, ESP8266 becomes a Wi-Fi host and user can connect to ESP8266 using Wi-Fi. By accessing the address of the ESP8266 with a browser, user can change the settings and restart the device through HTML interfaces.

## REFERENCES

- Lee, J. S., Su, Y. W., & Shen, C. C. (2007, 5-8 Nov. 2007). *A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi*. Paper presented at the Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE.
- Milinković, A., Milinković, S., & Lazić, L. Choosing the right RTOS for IoT platform.
- National Internet of Things (IoT) Strategic Roadmap: A Summary*. (2015). Mimos. Retrieved 27 April 2016, from [http://mimos.my/iot/National\\_IoT\\_Strategic\\_Roadmap\\_Summary.pdf](http://mimos.my/iot/National_IoT_Strategic_Roadmap_Summary.pdf)
- MQTT Version 3.1.1*. (2015). *Docs.oasis-open.org*. Retrieved 14 April 2016, from <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>
- Organization, W. H. (2012). Malaysia health system review.
- Österlind, F. (2006). A sensor network simulator for the Contiki OS. *SICS Research Report*.
- Vermesan, O., & Friess, P. (2013). *Internet of things: converging technologies for smart environments and integrated ecosystems*: River Publishers.
- Inc, E. (2016, August 10). *ESP8266EX Datasheet*. Retrieved August 22, 2016, from [https://espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)
- Tuan Huynh, Kevin Peek & Paul Shumate (November 15, 2005). **Tensilica Xtensa CS451 - Advanced Processor Architecture**
- PHP 5 Tutorial. (n.d.). Retrieved August 21, 2016, from <http://www.w3schools.com/php/default.asp>
- PHP: Hypertext Preprocessor. (n.d.). Retrieved August 21, 2016, from <http://php.net/>
- MQTT. (n.d.). Retrieved August 21, 2016, from <http://mqtt.org/>
- Mosquitto. (n.d.). Retrieved August 21, 2016, from <https://mosquitto.org/>
- Eclipse/mosquitto. (n.d.). Retrieved August 21, 2016, from <https://github.com/eclipse/mosquitto>

Learn - MariaDB.org. (n.d.). Retrieved August 21, 2016, from <https://mariadb.org/learn/>

MySQL Improved Extension. (n.d.). Retrieved August 21, 2016, from <http://php.net/manual/en/book.mysql.php>

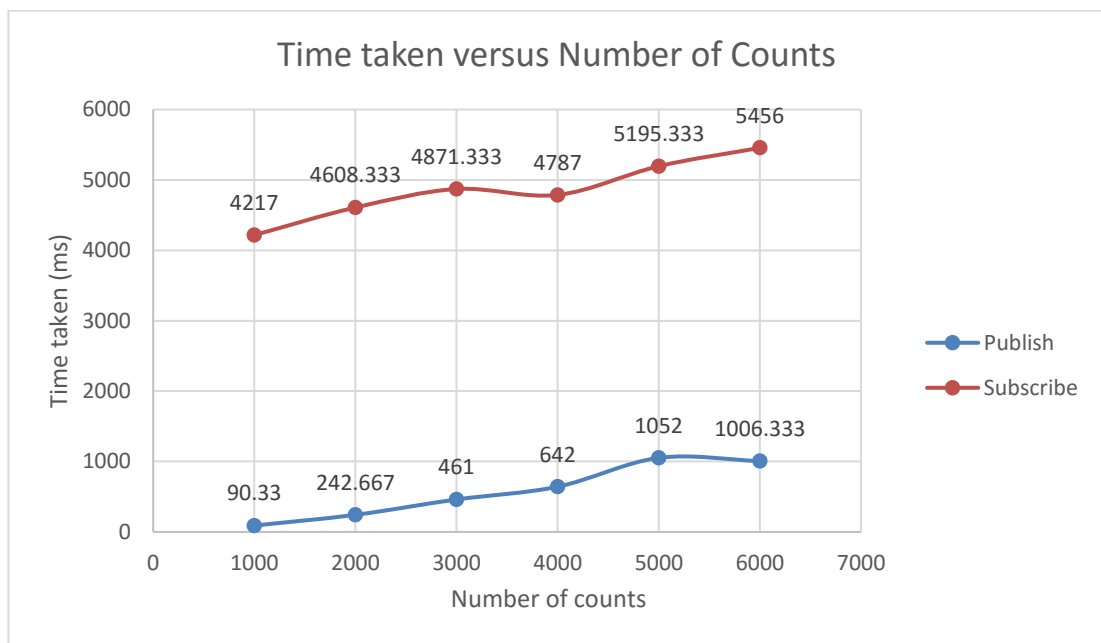
PHP 5 MySQLi Functions. (n.d.). Retrieved August 21, 2016, from [http://www.w3schools.com/php/php\\_ref\\_mysqli.asp](http://www.w3schools.com/php/php_ref_mysqli.asp)

MySQLi. (n.d.). Retrieved August 21, 2016, from <https://en.wikipedia.org/wiki/MySQLi>

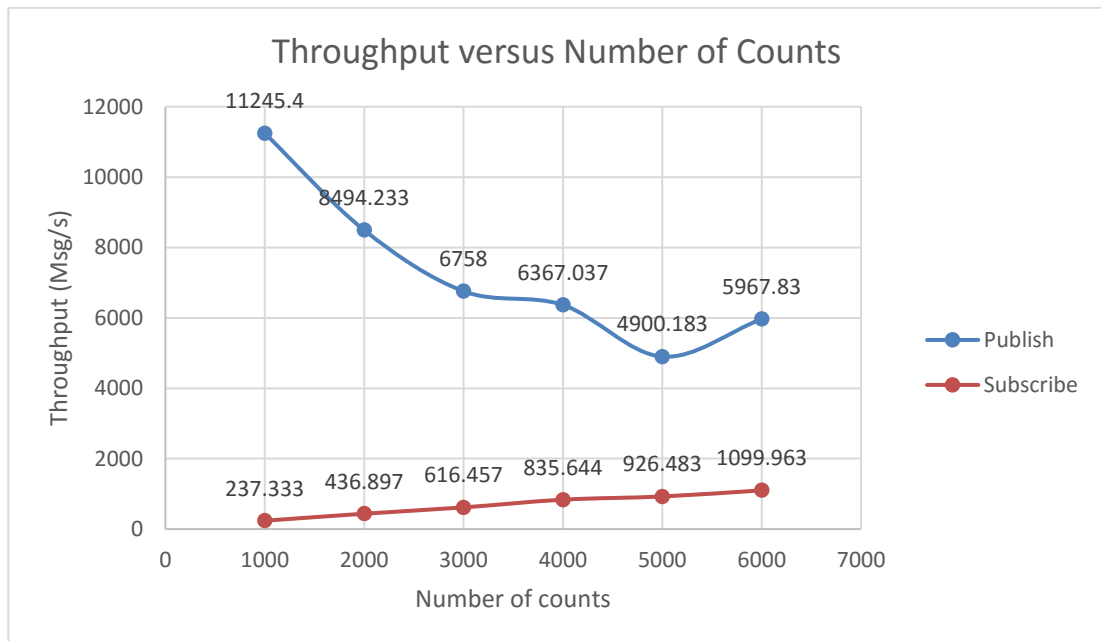
*takanorig/mqtt-bench*. (2015). *GitHub*. Retrieved 14 June 2016, from <https://github.com/takanorig/mqtt-bench>

## APPENDICES

### APPENDIX A: Graphs



**Graph 5-1 Time taken of Publish and Subscribe for MQTT server**



**Graph 5-2 Throughput of Publish and Subscribe for MQTT broker**

## **APPENDIX B: Computer Programme Listing**

The Computer programme is listed and submitted in softcopy

Web server

PHP webpages: file in Html directories

Python service source code: datalogger.py

MQTT mosquito source code

MQTT-bench source code