USER FRIENDLY IOT SOLUTION FOR MULTIPLE INDUSTRIES BY ADOPTING USB-ENABLED SENSOR/ACTUATOR MODULES AND PRECONFIGURED COMPUTER

BY

CHIA SHUN CHENG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2020

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer

Academic Session: Jan 2020

I CHIA SHUN CHENG

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

[illegible]

(Author's signature)

[illegible]

(Supervisor's signature)

Address:
90, Bagan Sungai Pulai,
45200 Sabak Bernam,
Selangor.

Dr. Cheng Wai Khuen
Supervisor's name

Date: 24 April 2020

Date: 24 April 2020

USER FRIENDLY IOT SOLUTION FOR MULTIPLE INDUSTRIES BY ADOPTING USB-ENABLED SENSOR/ACTUATOR MODULES AND PRECONFIGURED COMPUTER

BY

CHIA SHUN CHENG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2020

DECLARATION OF ORIGINALITY

I declare that this report entitled "**USER FRIENDLY IOT SOLUTION FOR MULTIPLE INDUSTRIES BY ADOPTING USB-ENABLED SENSOR/ACTUATOR MODULES AND PRECONFIGURED COMPUTER**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : [illegible]

Name : Chia Shun Cheng

Date : 24 April 2020

ACKNOWLEGDEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Cheng Wai Khuen and my moderator, Mr Teoh Shen Khang who have given me this golden opportunity to engage in an Internet of Things related project. It is my first step to establish a career in Internet of Things field. A million thanks to you.

Moreover, I would like to thank my project teammate, Ang Cheng Kee, who has given me countless of assistance while developing this project. Although each of us are engaging in different project scope, however he is still helping me to solve difficulties faced in this project.

ABSTRACT

This project is an Internet of Things (IoT) device development project. The proposed solution will enable non-technical users to explore and customise an IoT solution with the least amount of technical knowledge and skill. Nowadays, the Internet of Things technology has become one of the key components that help a company to grow. However, customisation issue has become one of the biggest challenges faced by existing IoT applications. Due to lack of IoT platforms with capabilities to support multiple industries, companies are facing difficulties to develop custom IoT solutions based on their requirements. Besides, most of the existing IoT applications are lack of user-friendly characteristics. Implementation and modification of an IoT solution still required a technician. The proposed solution is to implement plug-and-play features for sensors and actuators by using microcontrollers and Universal Serial Bus (USB) technology. The main purposes of this project are to develop a user-friendly customisable IoT platform with plug-and-play sensors and actuators and to develop USB-enabled sensor/actuator modules by using customised microcontrollers. To realise the proposed solution, master-slave architecture is adopted by using a Raspberry Pi computer as master control board and Digispark ATTINY85 Mini USB Development Boards as slave devices. Raspberry Pi 3 model B+ is chosen to be the brain of the proposed solution due to its ability to handle high-level programs and the availability of Internet connectivity requirements. Besides, Digispark ATTINY85 Mini USB Development Board is suitable to act as a slave device due to its cost-effectiveness and the abilities to integrate with sensors and actuators. Furthermore, Raspberry Pi 3 model B+ computer and Digispark ATTINY85 Mini USB Development Board are also capable to perform data transfer by using USB protocol in order to transmit sensor readings and allow commands to be passed. At the end of the project, a customisable IoT device and USB-enabled sensor/actuator modules will be developed to reduce the learning gap for a user to get familiar with the Internet of Things and enable fast setup for an IoT solution as well as providing ease of modification of an IoT solution.

TABLE OF CONTENTS

TITLE PAGE i
DECLARATION OF ORIGINALITY ii
ACKNOWLEGDEMENTS iii
ABSTRACT iv
TABLE OF CONTENTS v
LIST OF FIGURES viii
LIST OF TABLES xiii
LIST OF SYMBOLS xiv
LIST OF ABBREVIATIONS xv
CHAPTER 1: INTRODUCTION 1
1.1 Problem Statements 1
1.2 Background Information and Motivation 2
1.3 Project Scope 5
1.4 Project Objectives 6
1.5 Impact, Significance and Contribution 7
1.6 Highlights of What Have Been Achieved 8
1.7 Report Organisation 9
CHAPTER 2: LITERATURE REVIEW 10
2.1 IoT in a Box 10
2.1.1 Product Review 10
2.1.2 Strengths 11
2.1.3 Weaknesses 11
2.2 Skydrop Smart Watering Sprinkler Controller 12
2.2.1 Product Review 12
2.2.2 Strengths 13
2.2.3 Weaknesses 13
2.3 VersaSense Wireless Device 14
2.3.1 Product Review 14
2.3.2 Strengths 15
2.3.3 Weaknesses 15
2.4 Ezblock Pi 16
2.4.1 Product Review 16
2.4.2 Strengths 17
2.4.3 Weaknesses 17
2.5 Tuya Smart WIFI Power Strip 18

2.5.1 Product Review 18
2.5.2 Strengths 19
2.5.3 Weaknesses 19
2.6 Comparison Between Proposed IoT Device and Existing Applications 20
2.7 Recommendations and Summary 21
CHAPTER 3: SYSTEM DESIGN 23
3.1 System Architecture Design 23
3.1.1 Google Cloud Architecture 23
3.1.2 Use Case Diagram 25
3.1.3 System Flowcharts 26
3.1.4 System Block Diagram 37
3.1.5 Schematic Diagram 38
3.2 GUI design for Debugging Purposes 42
3.3 Container Design for Small Scale Agriculture 43
CHAPTER 4: SYSTEM IMPLEMENTATION AND TESTING 44
4.1 Methodology 44
4.2 Project Workflow in Iterative Phased Product Development Model 45
4.3 Project Timeline 47
4.4 Technology Involved 49
4.4.1 Raspberry Pi 3 Model B+ 49
4.4.2 Digispark ATTINY85 Mini USB Development Board 51
4.4.3 Micro-USB Charger for Raspberry Pi 52
4.4.4 Type-A-to-Type-B-Micro USB cable 52
4.4.5 DHT11 Temperature & Humidity Sensor 53
4.4.6 LDR Module 53
4.4.7 LED board 54
4.4.8 USB Web Camera 54
4.4.9 Water Pump 55
4.4.10 Raspbian Buster Operating System 55
4.4.11 Google Cloud Platform 55
4.4.12 Firebase Real-Time Database 56
4.4.13 NGINX web server 56
4.4.14 JSON 57
4.4.15 MQTT 57
4.4.16 Linux Shell Scripts 57
4.4.17 Python Programming Language 58

4.4.18 C and C++ Programming Language 58
4.5 System Implementation 59
4.5.1 Cloud Architecture Implementation in Google Cloud Platform 59
4.5.2 Configuration of Raspberry Pi 63
4.5.3 Configuration of Digispark ATTINY85 Mini USB Development Board 66
4.6 Implementation Issues and Challenges 68
4.7 System Verification Test 69
CHAPTER 5: SYSTEM DELIVERABLE 73
5.1 Cloud Architecture in Google Cloud Platform 73
5.2 The Proposed IoT Device (Master Device) 74
5.2.1 The Preconfigured Raspberry Pi Computer 74
5.2.2 Graphic User Interface for Debugging Purposes 80
5.3 The Proposed USB-Enabled Sensor/Actuator Modules (Slave Devices) 81
5.3.1 The Proposed USB-Enabled Actuator Modules 81
5.3.2 The Proposed USB-Enabled Sensor Modules 82
5.4 Container for Use Case Studies 84
5.5 The Proposed Solution 87
CHAPTER 6: CONCLUSION 89
6.1 Project Review and Discussion 89
6.2 Novelties and Contributions 90
6.3 Future Work 91
BIBLIOGRAPHY 92
APPENDICES – WEEKLY LOG A-1
POSTER
PLAGIARISM CHECK RESULT
FYP 2 CHECKLIST

LIST OF FIGURES

Figure Number	**Title**	**Page**
Figure 1.1	Graph of Average Sensor Cost in USD per Year	3
Figure 1.2	Infographic of Internet of Things	4
Figure 2.1	IoT in a Box Plug-and-Play Devices	10
Figure 2.2	IoT in a Box Mobile Apps	10
Figure 2.3	Skydrop smart sprinkler controller	12
Figure 2.4	VersaSense Wireless Device, VersaSense Sensors and Edge Gateway	14
Figure 2.5	Ezblock Pi	16
Figure 2.6	Tuya Smart WIFI Power Strip	18
Figure 3.1	Google Cloud Architecture Diagram	24
Figure 3.2	Use Case Diagram	25
Figure 3.3	Flowchart of Activity in Raspberry Pi after Powered on	27
Figure 3.4	Flowchart of Scheduler in Raspberry Pi	28
Figure 3.5	Flowchart of Adding Modules to Raspberry Pi Computer	30
Figure 3.6	Flowchart of Removing Modules from Raspberry Pi Computer	31
Figure 3.7	Flowchart of Actuation Control via Internet	32
Figure 3.8	Flowchart of Bluetooth Connectivity	33

Figure Number	Title	Page
Figure 3.9	Flowchart of Wi-Fi Configuration	34
Figure 3.10	Flowchart of Activities in USB-Enabled Sensor Module	35
Figure 3.11	Flowchart of Activities in USB-Enabled Actuator Module	36
Figure 3.12	Block Diagram of the Proposed Solution	37
Figure 3.13	Schematic Diagram of Bluetooth Switch	38
Figure 3.14	Schematic Diagram of USB-Enabled Temperature & Humidity Sensor Module	39
Figure 3.15	Schematic Diagram of USB-Enabled Light Sensor Module	39
Figure 3.16	Schematic Diagram of USB-Enabled Sprinkler Module	40
Figure 3.17	Schematic Diagram of USB-Enabled LED Module	41
Figure 3.18	GUI design for Debugging Purposes	42
Figure 4.1	Project Work Flow for First Version of Proposed Solution in Iterative Phased Product Development Model	45
Figure 4.2	Project Work Flow for Second Version of Proposed Solution in Iterative Phased Product Development Model	46
Figure 4.3	Gantt Chart 1 of Project Timeline	47
Figure 4.4	Gantt Chart 2 of Project Timeline	48
Figure 4.5	Raspberry Pi 3 Model B+	50
Figure 4.6	Digispark ATTINY85 Mini USB Development Board	51
Figure 4.7	Micro-USB Charger for Raspberry Pi	52

Figure Number	Title	Page
Figure 4.8	Type-A-to-Type-B-Micro USB cable	52
Figure 4.9	DHT11 Temperature & Humidity Sensor	53
Figure 4.10	LDR Module	53
Figure 4.11	LED Board	54
Figure 4.12	USB Web Camera	54
Figure 4.13	Water Pump	55
Figure 4.14	Registry Details in Cloud IoT Core	59
Figure 4.15	Registration of IoT Device 1	60
Figure 4.16	Registration of IoT Device 2	60
Figure 4.17	Topics Created in Cloud Pub/Sub	61
Figure 4.18	Cloud Functions in Google Cloud Platform	61
Figure 4.19	Data Structure in Firebase Real-Time Database	62
Figure 4.20	RTMP Web Server Installation	62
Figure 4.21	RTMP Module in Web Server	63
Figure 4.22	Commands for Update and Upgrade	64
Figure 4.23	Commands for Libraries Installation	64
Figure 4.24	Commands for Public Key Pairs Generation	64
Figure 4.25	Commands for Google Cloud Connectivity	64
Figure 4.26	Commands for Enable Bluetooth Connectivity	65
Figure 4.27	Changes in Bluetooth Service File	65
Figure 4.28	Settings in Arduino IDE	66

Figure Number	Title	Page
Figure 4.29	Install Digistump AVR Boards in Arduino IDE	67
Figure 4.30	Upload Program to Digispark ATTINY85 Mini USB Development Board	67
Figure 5.1	Final Google Cloud Architecture Diagram	74
Figure 5.2	The Preconfigured Raspberry Pi Computer	74
Figure 5.3	Start-Up Activities with No Sensor Connected	76
Figure 5.4	Start-Up Activities with At Least One Sensor Connected	76
Figure 5.5	Start Background Thread for Reconnection	77
Figure 5.6	USB-Enabled Actuator Module Plugged-in Activity	77
Figure 5.7	Unsupported USB Device Plugged-in Activity	78
Figure 5.8	USB-Enabled Sensor Plugged-in Activity	78
Figure 5.9	Activities of Performing Cloud Commands	79
Figure 5.10	Scheduled Time Reach with At Least One Sensor Connected	79
Figure 5.11	GUI Implemented for Debugging Purposes	80
Figure 5.12	USB-Enabled LED Module	81
Figure 5.13	USB-Enabled Water Sprinkler Module	82
Figure 5.14	USB-Enabled Temperature and Humidity Sensor Module	82
Figure 5.15	USB-Enabled Light Sensor Module	83
Figure 5.16	Container for Use Case Studies	84
Figure 5.17	Front View of Container for Use Case Studies	85

Figure Number	Title	Page
Figure 5.18	Back View of Container for Use Case Studies	85
Figure 5.19	Wooden Box on Top of Container for Use Case Studies	86
Figure 5.20	Under the Cover of The Container	86
Figure 5.21	The Proposed Solution	87
Figure 5.22	The Proposed Solution Implemented in Small Agriculture Industry	88

LIST OF TABLES

Table Number	**Title**	**Page**
Table 2.1	Comparison between proposed IoT device and other existing applications	17
Table 4.1	System Verification Test	69

LIST OF SYMBOLS

%	Percentage
℃	Degree Celsius
&	And

LIST OF ABBREVIATIONS

A	Ampere
app	Application
DC	Direct Current
EEPROM	Electrically Erasable Programmable Read-Only Memory
GPIO	General-Purpose Input Output
I2C	Inter-integrated Circuit
ID	Identity Document
IoT	Internet of Things
IIoT	Industrial Internet of Things
IPSO	Independent Press Standards Organisation
JSON	JavaScript Object Notation
LAN	Local Area Network
LCD	Liquid Crystal Display
LDR	Light Dependent Resistor
LED	Light-Emitting Diode
mA	Milliampere
MQTT	Message Queuing Telemetry Transport
RAM	Random-Access Memory
RH	Relative Humidity
RTSP	Real-Time Streaming Protocol

RTMP	Real-Time Messaging Protocol
USB	Universal Serial Bus
QR	Quick Response
V	Volt
VCC	Voltage Common Collector

CHAPTER 1: INTRODUCTION

1.1 Problem Statements

Nowadays, IoT has been implemented in multiple industries in order to increase productivity and profit. However, each industry required custom IoT solution in order to fulfil their needs. The demand for IoT solutions that perfectly fulfilled the requirements for each industry has negatively affected the sales of existing IoT solution because of the solutions are lack of readiness for customisation (Kamat, 2017). Due to **lack of IoT platform with capability to support multiple industries**, companies tend to visit IoT technician in order to develop an IoT platform that meets their requirements. According to Kurisu (2015), companies are forced to develop their specialised gateways and suffered from development cost and time delays due to lack of customisable gateway solutions that can concurrently support multiple industries in the market. Most of the current existing IoT platform only provides fixed IoT solutions for industries and common users. The provided IoT solutions are not flexible and only exist to fulfil the specification of certain users. Current existing customisable IoT platforms have provided variety of solutions, but most of the solutions are not cost effective or do not provide modification of current IoT solution. For instance, if user no longer requires a temperature sensor in his IoT solution, he is not able to remove the temperature sensor from his current IoT solution as all components have soldered on the IoT devices. Thus, industries and common IoT users are facing difficulties to have a custom IoT solution which can fulfil their needs.

Besides, there is also **lack of user-friendly customisable IoT platforms** in the market. Traditionally, if user tends to develop a custom IoT solution, they will need to learn a lot of technical skills such as programming and wiring. In addition, user also needs to familiar with hardware sensors, processors, controllers, and also the IoT platform they have chosen for his IoT solution. In short, it is not easy for a common IoT user to develop a custom IoT solution as technical skills are required. Even for the user who has the knowledge, customisation of IoT platform still involves a lot of complex configurations to be achieved and a lot of time will be spent. On the other hand, user can use existing user-friendly customisable IoT platforms to develop custom IoT solution. However, the platforms are not cost effective and do not provide ease of modification on the IoT solution.

1.2 Background Information and Motivation

Internet of Things (IoT) is a network of connected electronic devices that have the ability to transfer data between each other via the Internet. Each electronic device in the Internet of Things is called a “thing” and any electronic device can become a “thing” if an IP address can be assigned to it (Rouse, 2019). With an IP address, an IoT device can be identified among other connected IoT devices. Hence, effective communication can be achieved throughout the world. An IoT device is usually embedded with sensors and actuators to provide sensing and actuation. The readings of sensors collected by the IoT device are being transferred to other IoT devices in the network via the Internet for data exchange. Furthermore, commands can be sent to an IoT device in order to control the actuation of its embedded actuator and rules also can be set to achieve automation. By adopting the IoT technology, almost every daily device can have sensors embedded to them and data can be shared between each other via built-in Wi-Fi (Morgan, 2014).

The Internet of Things has provided an opportunity for industries to develop cyber-physical systems for advanced manufacturing. Industrial Internet of Things (IIoT), which is also known as the fourth industrial revolution, is the implementation of the Internet of Things technology in a business setting. IIoT is used to optimise industrial processes by introducing embedded sensors, actuators and antennas in industrial applications (Ranger, 2018). Data collected from embedded IoT sensors are used to monitor industrial processes, big data analysis, data exchange in machine control and automation. Analysis of big data collected by thousands of real-time sensors helps a company to monitor its system processes and manufacturing processes as well as providing a chance for predictive maintenance (Rajkarne, 2019).

With the availability of the Internet and cheap IoT sensing devices, many industries have implemented IoT technology in their businesses in order to maximise their profit. Furthermore, cheap IoT sensing devices also have enabled industries to collect data from a wide area easily and this big amount of data is then being used for big data analysis and environment monitoring. For instance, agriculture industries have used IoT sensors in their crops field to optimise management of the fertilisation process based on the readings of IoT sensors.

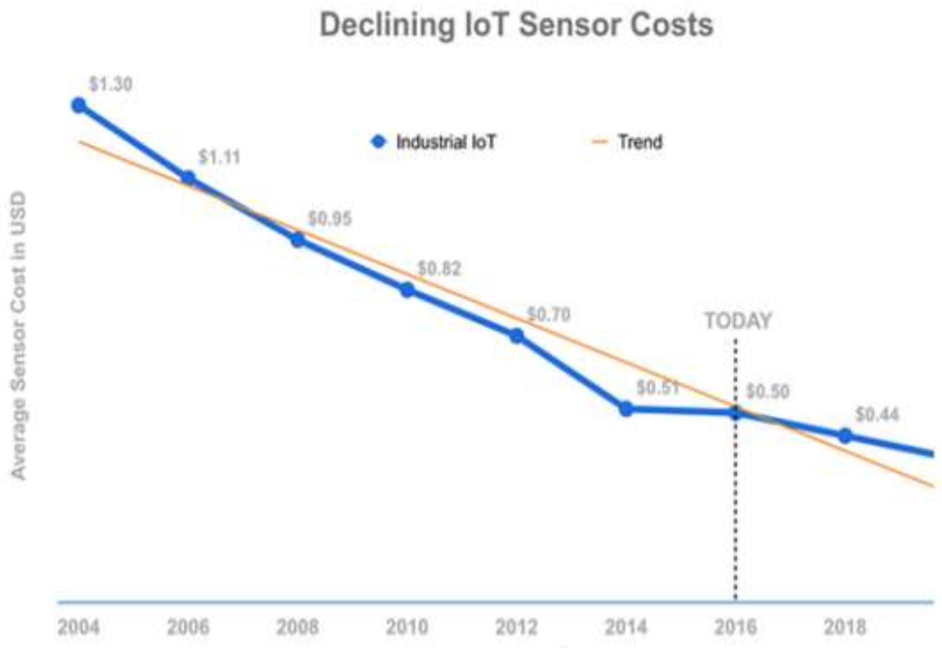


Figure 1.1 Graph of Average Sensor Cost in USD per Year

Considering the advanced growth rate of IoT technology, researcher predicts that the shipment of agriculture IoT device will reach 52 million units in 2018 and the crops production will be increased by 70 percent in the near future (Klubnikin, 2017). Besides, factories are generating approximate 3.7 trillion economic values from the implementation of Internet of Things. It is proven that by using IoT technology, productivity of a business and operational efficiency have greatly improved. According to experts, industrial IoT will reach 110 billion in the market by 2020 (Petrov, 2019).

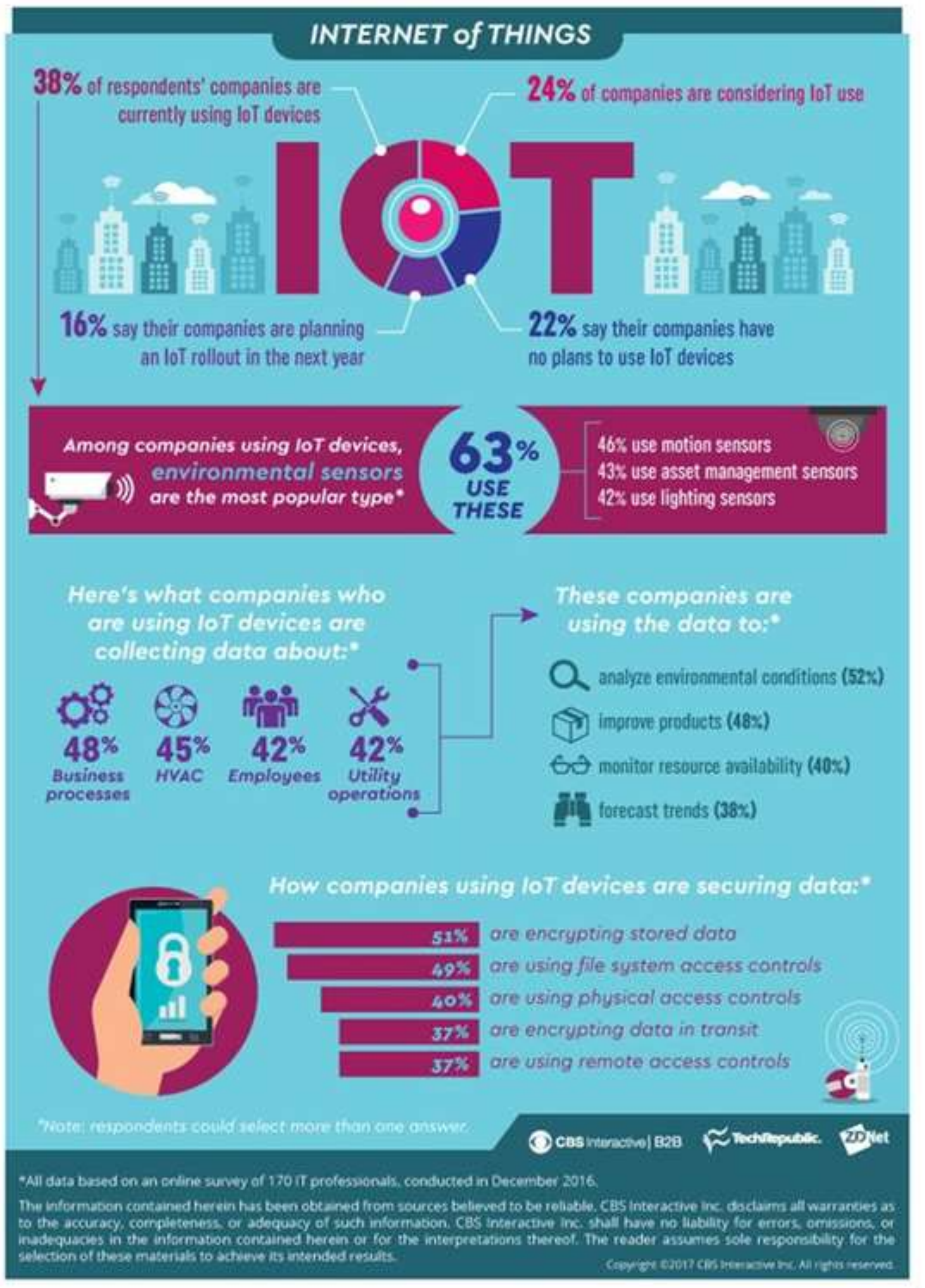


Figure 1.2 Infographic of Internet of Things

1.3 Project Scope

At the end of this project, a preconfigured customisable, scalable IoT device will be developed with plug-and-play USB-enabled sensor/actuator modules provided. The proposed solution will be developed by using a small size computer known as Raspberry Pi computer, because it is suitable to be used as a platform to connect USB-enabled sensors and actuators as well as establish cloud connectivity and data management. Besides, the proposed USB-enabled sensor/actuator module will be developed by using a microcontroller known as Digispark ATTINY85 Mini USB Development Board in order to provide plug-and-play features. The USB-enabled sensor/actuator modules can be added or removed from the proposed solution based on the requirement of an industry in order to provide IoT solution customisation. Even though the proposed solution can be used as an IoT solution for many industries by playing with the combination of USB-enabled sensor and actuator modules, however, the proposed solution currently will only focus on small scale agriculture IoT solution.

The instances of connected USB-enabled sensor/actuator modules will be managed by the main program of the proposed IoT device dynamically and update the status of the proposed IoT solution to the Google Cloud Platform based on how many USB-enabled sensor/actuator modules are added or removed. The proposed solution will be monitored through a monitoring mobile application developed by another student. However, a simple graphic user interface (GUI) will be implemented into the proposed solution for local monitoring and debugging purposes. Besides, a switch and some LEDs will be connected to the GPIO pins of Raspberry Pi computer in order to enable Bluetooth functionality. The switch will enable users to turn on Bluetooth of Raspberry Pi computer for setting Wi-Fi and Internet connectivity during offline and the LEDs will indicate the status of Bluetooth connectivity between the proposed IoT device and the monitoring mobile application. Furthermore, the proposed IoT solution also will be able to detect a USB camera and provide video streaming and photo capturing features. The main program of the proposed IoT device will be designed to detect the same type of microcontroller used in the proposed USB-enable sensor/actuator modules and enable hex file of specified sensor or actuator module to be flashed into the newly bought blank microcontroller in order to integrate the microcontroller into the proposed solution.

1.4 Project Objectives

1. To develop a user-friendly customisable IoT platform with plug-and-play sensors and actuators

By introducing plug-and-play sensors and actuators, user-friendliness issues faced while developing a custom IoT solution can be solved. The objective is to eliminate the need of technical skills required to configure an IoT device in order to customise an IoT solution. For instance, user will not need to perform any programming skill in order to add or remove a sensor from the IoT device. Traditionally, when a sensor need to be added into an IoT device, user are required to perform programming on the IoT device in order to configure input pins for sensor data. For instance, user needs to configure the GPIO pins on a Raspberry Pi computer in order to allow and specify the pins used as data input for sensor readings. However, device configuration can be avoided by introducing a pre-configured IoT platform that supports plug-and-play sensors and actuators. Furthermore, implementation of plug-and-play concept for sensors and actuators also solve customisation problem by allowing the sensors and actuators to be plugged-in or remove from the IoT platform easily. Hence, user can use the proposed solution to develop, customise and modify an IoT solution without technical skills.

2. To develop USB-enabled sensor/actuator modules by using customised microcontrollers

In order to develop plug-and-play sensors and actuators that solve customisation problem and user-friendliness issues, common sensors and actuators will integrate with small in size and cost-effective microcontrollers. The objective is to standardise the way sensors and actuators communicate with IoT device by using pre-configured microcontroller to manage device communication. The proposed solution will enable sensors and actuators to be added or removed from the proposed IoT device by using common Type-A USB cables. The USB-enabled sensor/actuator module will be the integration of a customised microcontroller, a sensor/actuator and a USB cable. When user wants to add a sensor to the proposed IoT device, they only need plug-in a preconfigured USB-enabled sensor module to the proposed IoT device. Traditionally, user will need to perform electronic wiring and soldering in order to add a sensor to an IoT device. Without adopting the proposed solution, sensors and actuators are hard to be removed from an IoT device as they are whether already soldered with iron or have messy wires connected to the IoT device. By introducing sensor/actuator modules that have integrated with customised USB-enabled microcontrollers, the modification of a custom IoT solution will become easy as there are lesser wires and no programming skill required.

1.5 Impact, Significance and Contribution

When developing a custom IoT solution, user will usually search for existing products, consult an IoT solution provider or choose to build the IoT solution from scratch. However, most existing IoT products are rather expensive or do not provide ease of modification on the hardware platform. Besides, the process of developing a complete IoT solution may be complicated based on complexity of design and there are a lot of skills are needed to build an IoT solution. For instance, when a user is developing an IoT solution, the user usually will perform some wiring, soldering, programming as well as the configuration of their preferred cloud IoT platform. The development of an IoT solution also required a lot of time and once the development of their custom IoT solution is completed, it is hard to perform modifications as the process of modification will also involve technical skills. Thus, if a novice user wants to have some modifications on their IoT solution, it is hard for them to not waste time on learning the required skills, implementation and debugging.

The deliverables of this project will help user to develop a custom IoT solution in the least effortful way via plug-and-play feature. This proposed solution will speed up the development of custom IoT solutions for multiple industries by using preconfigured computer and customised USB-enabled sensor/actuator modules. For instance, an agriculture industry can purchase the proposed IoT device, which is the preconfigured computer, together with the USB-enabled plug-and-play sensors and actuators and then a custom IoT solution will be achieved by plugging in the USB-enabled sensor/actuator modules to the preconfigured computer. This is due the preconfigured computer is designed to detect the proposed USB-enabled sensor/actuator modules and trigger the corresponding functionalities automatically based on the type of connected USB-enabled sensor/actuator modules. The sensor readings and the information of the connected USB-enabled sensor/actuator modules will be published to the Cloud, thus the end user can monitor the proposed solution via Internet. By using the proposed solution, user can only focus on what combination of USB sensor/actuator modules is suitable for their custom IoT solution and be able to build their IoT solution by themselves. User will not worry about complex configuration for each IoT device to be connected to the Cloud platform. Besides, user can interact, monitor and control the proposed solution by sending commands to the Cloud via Internet and the monitoring mobile application developed by another team member.

1.6 Highlights of What Have Been Achieved

There are a few highlights have been achieved in order to fulfil the objectives of this project. First and foremost, a Raspberry Pi computer has been configured to become the proposed IoT device with the abilities to manage the communication with Cloud platform and connectivity of the proposed USB-enabled sensor/actuator modules as well as the events happened in its USB environment. Besides, the proposed USB-enabled sensor/actuator modules are developed by using the Digispark ATTINY85 Mini USB Development Boards with sensors and actuators. They have been configured to perform USB communication with the proposed IoT device. Furthermore, the proposed solution has adopted the Google Cloud Platform for publish sensor readings, device status and the details of connected USB-enabled sensor/actuator modules, as well as receiving commands from Google Cloud Platform. For data storage, the Firebase Real-Time Database is used by the proposed solution as it provides services that based on events happened in the database.

1.7 Report Organisation

The report will be organised into 6 chapters. The Introduction will be in chapter 1, which is the first chapter of the report. Chapter 2 is for Literature Review, this chapter will discuss the reviews of 5 existing applications which are similar to the proposed solution. At the end of chapter 2, there will be a table of comparison between the existing applications and the proposed solution followed by a summary and recommendations. Chapter 3 will focus on System Design. This chapter will contain system flow charts, use case diagrams, block diagrams, schematic diagrams and system architecture design. This chapter also includes a container design for use case studies, as well as a GUI design for debugging purposes. Chapter 4 of the report is about System Implementation and Testing. This chapter will mention about the methodology adopted by this project and the project timeline. This chapter also will describe technologies required for project development and explain how the hardware and software are being set up. Besides, this chapter will include implementation challenges and system verification test. System Deliverable will be in chapter 5 and this chapter will discuss the final deliverables of the project. Chapter 6 of the report will be the conclusion of the project. Project review, discussions, conclusions, as well as future improvements will be included in this chapter.

CHAPTER 2: LITERATURE REVIEW

2.1 IoT in a Box

2.1.1 Product Review

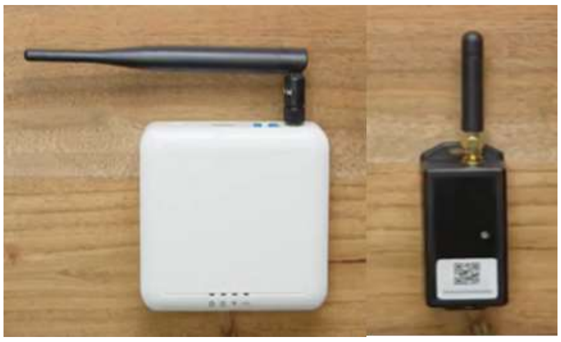


Figure 2.1 IoT in a Box Plug-and-Play Devices

Figure 2.2 IoT in a Box Mobile Apps

IoT in a Box is a collection of pre-packaged IoT solutions. Each package comes with a ready-to-use solution which consisted of a plug-and-play IoT gateway, customisable mobile apps and wireless IoT sensors. IoT in a Box aims to eliminate manual monitoring on facility assets due to frequent human errors. To encourage user to install automated monitoring systems, IoT in a Box minimises the time and effort for user to develop a custom IoT solution, by providing the most user-friendly hardware configuration through plug-and-play features and QR codes (Iotinabox, 2019).

Each sensor of IoT in a Box has its own QR code and ID. QR code and ID is used to identify the type of sensor. Correct sensor configuration is triggered when the QR code on the sensor is scanned by IoT in a Box mobile app. Besides, all the sensors provided by IoT in a Box are standalone and connected wirelessly. The sensors have their own batteries and there are antennas provided for each sensor to connect and send the readings to cloud service through IoT in a Box plug-and-play gateway.

IoT in a Box plug-and-play gateway also required no technical knowledge to be setup. The gateway is pre-configured, user only need to plug in Ethernet cable and antenna to the gateway in order to provide Internet connection. In addition, all ports on the plug-and-play gateway have unique shape and only the correct cable or antenna can fit in each port. Besides, the gateway also has its own QR code and ID. User can simply scan the QR code to connect the gateway to their IoT in a Box mobile app.

2.1.2 Strengths

User can customise their IoT solution by using QR code and simple plug-and-play feature of IoT in a Box. User also can easily expand their IoT solution by adding or removing IoT in a Box sensors or gateways. All IoT in a Box sensors are wireless, user can install the sensors at anywhere within the coverage area of IoT gateway. Readings of sensors are sent to cloud database, thus user can monitor their IoT solution at anywhere and anytime.

2.1.3 Weaknesses

Although IoT in a Box provides hundreds of IoT sensors, but all the sensors are wireless, thus each sensor is very expansive. Besides that, IoT in a Box doesn’t provide any actuator, for example a water pump and a light emitting device. Due to lack of actuators, IoT in a Box cannot provide self-regulating services but only act as a platform that only read the environments. IoT in a Box only supports their own products which are the integrated wireless sensors and doesn’t have any platform to connect common IoT sensors.

2.2 Skydrop Smart Watering Sprinkler Controller

2.2.1 Product Review

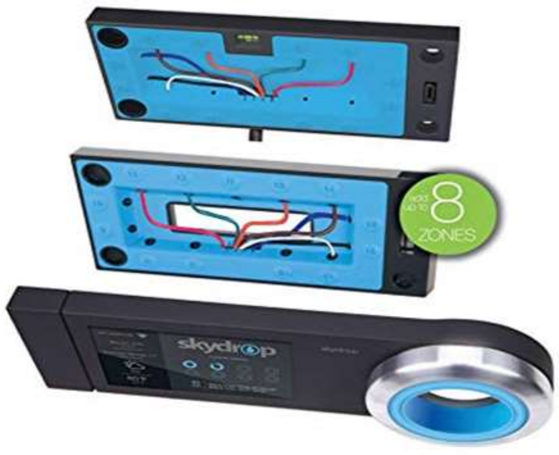


Figure 2.3 Skydrop smart sprinkler controller

Skydrop Smart Watering Sprinkler Controller is a sprinkler controller designed to improve management of watering system by providing IoT features and mobile app, in order to reduce human effort and amount of water used. Skydrop smart sprinkler controller is connected to cloud for weather data in order to adjust amount of water needed for each watering session (Skydrop, n.d.). The controller is also connected to a rain sensor to further improve the watering management.

The back panel of the smart controller is used to connect existing sprinkler system. The back panel has a lot of label and user is provided a short and simple wiring procedure to connect their existing sprinkler system to the back panel. After the wiring of back panel is completed, the smart controller can be plugged into the back panel in order to connect and control the watering sprinkler system. This process use simply plug-and-play feature, it required no extra configuration to connect the smart controller to the sprinkler system. User can manage up to 8 watering zones to the smart controller. To increase available watering zones, a Skydrop Expansion can be added to the smart controller. Skydrop Expansion is similar to the back panel of the smart controller. After the wiring is done, the expansion must be plugged into the back panel and the smart controller is also need to be plugged into the expansion.

Then, the configuration is done through plug-and-play feature and the smart controller is ready to manage the sprinkler system.

2.2.2 Strengths

Skydrop Smart Watering Sprinkler Controller provides plug-and-play feature to configure the existing sprinkler system which connected to back panel of the smart controller. Besides, by adding Skydrop Expansion, available watering zones are increased and no manual configuration is required for the smart controller to connect the extended watering zones. In addition, user can use Skydrop mobile app to monitor the watering system which connected to the smart controller and edit the watering schedules for their lawns. The smart controller will automatically connect to cloud for local weather data in order to perform self-regulation. When Wi-Fi is not available, user also can use Bluetooth to connect the smart controller with their Skydrop mobile app.

2.2.3 Weaknesses

Skydrop Smart Watering Sprinkler Controller also comes with some limitations. First, user cannot pause or resume a watering session via the smart controller (The WiFi Garden, 2016). Besides, the controller also cannot support other type of sensors, for example a light sensor. Basic knowledge of wiring and watering sprinkler system is required to connect the smart controller to the existing sprinkler system.

2.3 VersaSense Wireless Device

2.3.1 Product Review

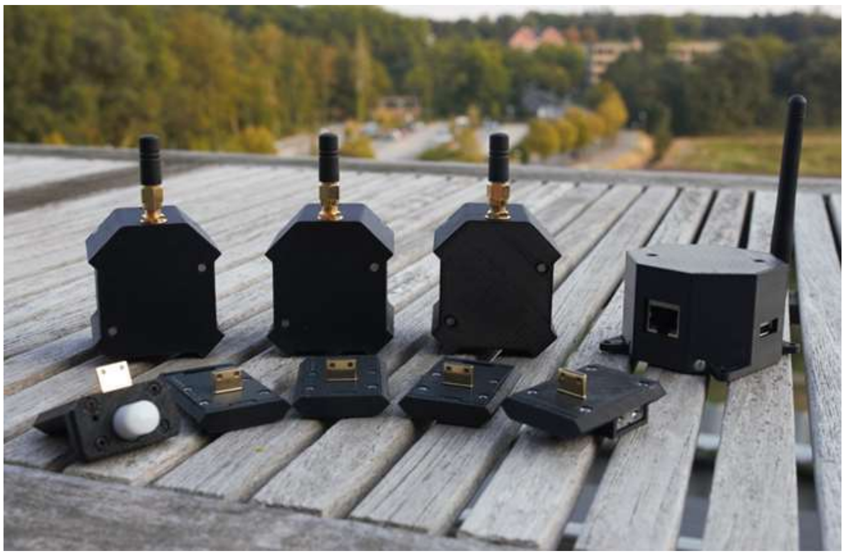


Figure 2.4 VersaSense Wireless Device, VersaSense Sensors and Edge Gateway

VersaSense Wireless Device is one of the most important hardware in MicroPnP platform introduced by VersaSense to automatically identify VersaSense sensors and actuators. VersaSense Wireless Device provides a platform for the communication between multiple VersaSense sensors, actuators and cloud. MicroPnP has successfully solved the complex manual configuration problem to customise an IoT solution and it has won the third place in IPSO challenge in 2015 (Matthys et al., 2016).

VersaSense Wireless Device is a wireless hub that can connect maximum three VersaSense plug-and-play sensors and actuators in order to build a custom IoT solution. Multiple VersaSense Wireless Devices can connect to a VersaSense Edge Gateway in order to expand an IoT solution. VersaSense offers more than 40 types of VersaSense plug-and-play sensors and actuators. The sensors and actuators are identified by using the SmartMesh protocol. When a plug-in is detected on the VersaSense Wireless Device, built-in VersaSense software will install and configure all the required resources for the corresponding sensor or actuator automatically (Versasense, 2019). After that, user can choose to monitor the sensors and control the actuators programmatically or through a web interface.

2.3.2 Strengths

VersaSense Wireless Device provides plug-and-play feature which will automatically identify the type of connected sensor or actuator and indicates them in the cloud. VersaSense Wireless Device minimises user’s effort to develop a custom IoT solution in the hardware platform. An IoT solution can be expanded by adding more VersaSense Wireless Devices to a VersaSense Edge Gateway. User can monitor and control the sensors and actuators via cloud. Besides, VersaSense Wireless Device is more cost effective compared to their competitors.

2.3.3 Weaknesses

VersaSense Wireless Device only supports sensors and actuators with Micro-USB ports. Thus, the solution cannot be scaled up without purchasing another VersaSense Wireless Device. Besides, user cannot connect to VersaSense Wireless Device when it is offline due to lack of offline communication protocol. VersaSense Wireless Device also doesn’t provide self-regulating service, thus user needs to do the technical configuration manually if self-regulation of device is wanted.

2.4 Ezblock Pi

2.4.1 Product Review

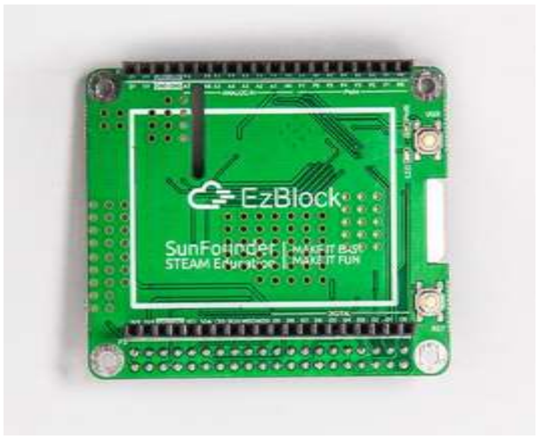


Figure 2.5 Ezblock Pi

Ezblock Pi is introduced to increase the efficiency to configure and customise an IoT device, Raspberry Pi. This application is a portable extension board which can be assembled to all type of Raspberry Pi (Kickstarter, 2019). Moreover, it provides a convenient platform for user to configure Raspberry Pi wirelessly via Bluetooth. User can drag and drop built-in IoT blocks, which are the code blocks for programming, to configure the logic through Ezblock Studio App and then use Bluetooth connection to transfer programs to Ezblock Pi which connected to a Raspberry Pi.

Generally, Ezblock Pi eliminated the messy cables and complex Wi-Fi configuration to access Raspberry Pi. With Ezblock Pi assembled, Raspberry Pi can connect to Ezblock Studio App wirelessly without any manual configuration. User can use Ezblock Studio App to create and test a virtual project. User can drag and drop a component or wire in the app, thus the process of customisation of IoT device in virtual environment is more efficient compared to physical environment. User can realise their project once the development of customised IoT device is successful in virtual environment. User also can develop a control panel in the app in order to control the sensors and actuators.

2.4.2 Strengths

Ezblock Pi provides a platform to increase the speed of process to develop custom IoT solution. By connecting Ezblock Pi to Ezblock Studio App via Bluetooth, user can configure an IoT device, Raspberry Pi, wirelessly without any complex Wi-Fi configuration in Raspberry Pi. Besides, Ezblock Pi can support any kind of common IoT sensors and actuators. By using the control panel built in Ezblock Studio App, user has full control of sensors and actuators which connected to Ezblock Pi.

2.4.3 Weaknesses

Ezblock Pi has no Wi-Fi or Internet connection, so user cannot access to it from a long distance which exceed the range of Bluetooth connection. Besides, the IoT solution cannot be expanded, because each Ezblock Pi can only attach to a Raspberry Pi. In addition, technical knowledge is required for user to perform configuration and customisation of IoT device. Ezblock Pi also doesn’t provide cloud service due to it is a Bluetooth based application. It also doesn’t provide self-regulating service, user needs to perform manual configuration if self-regulation is wanted. Moreover, it is also not cost effective.

2.5 Tuya Smart WIFI Power Strip

2.5.1 Product Review

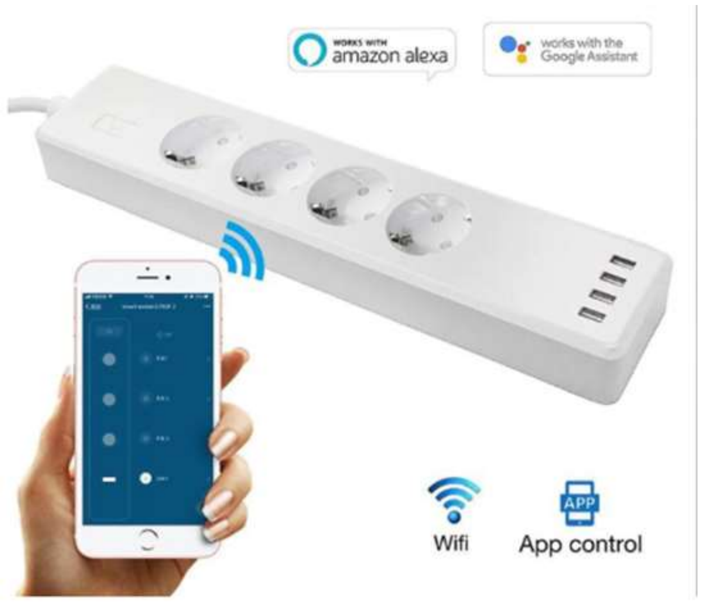


Figure 2.6 Tuya Smart WIFI Power Strip

Tuya Smart WIFI Power Strip is one of the Smart Home IoT devices from Tuya IoT platform which provides power supply for home appliances as well as electronics that charges via USB cables. Basically, Tuya Smart WIFI Power Strip is a universal power strip which has integrated with Wi-Fi and microcontroller in order to let user manages their house appliances remotely via mobile app. This smart power strip provides four USB charging ports and four Alternating current outlets. Each outlet or USB charging port is separated from each other, which means that they can supply electricity independently by pressing on or off button in the mobile app (Aliexpress, n.d.). Tuya Smart WIFI Power Strip also allows user to set schedules for each outlet and USB charging port in order to turn on or off automatically at the specified time. Besides, by pressing the main power button of the smart power strip, the configuration is triggered and it will be pending for user to indicate it into the IoT platform.

2.5.2 Strengths

Tuya Smart WIFI Power Strip can be controlled from a far distance and lets user manages connected home appliances easily. Besides, user can set timers and schedules on the smart power strip in order to perform automation. The smart power strip also provides multiple power supply for normal home appliances and electronics that use USB for charging. In addition, each power supply slot is independent of each other and can be turn on or off based on user's requirement.

2.5.3 Weaknesses

Tuya Smart WIFI Power Strip doesn't provide offline connectivity, once the device is offline, user cannot manually turn on or off a power supply slot. Besides, the smart power strip doesn't provide Cloud connectivity, thus there is no record of device active duration which can be used to calculate the amount of energy being saved or wasted. In addition, the smart power strip doesn't have a built-in temperature sensor, so user cannot be acknowledged the temperature of the power strip when it is active for a long duration.

2.6 Comparison Between Proposed IoT Device and Existing Applications

Table below shows the comparison between the proposed IoT device and existing applications, which are IoT in a Box, Skydrop Smart Watering Sprinkler Controller, VersaSense Wireless Device, Ezblock Pi and Tuya Smart WIFI Power Strip.

	IoT in a Box	Skydrop Smart Watering Sprinkler Controller	VersaSense Wireless Device	Ezblock Pi	Tuya Smart WIFI Power Strip	Proposed Application
Support multiple types of sensors and actuators	No	No	No	Yes	No	Yes
Simple Plug-and-Play	Yes	Yes	Yes	No	Yes	Yes
Expandable solution	Yes	Yes	Yes	No	Yes	Yes
Manual control of actuator	No	No	Yes	Yes	Yes	Yes
Ease of hardware modification	No	No	Yes	No	Yes	Yes
Cloud connectivity	Yes	Yes	Yes	No	No	Yes
Support self-regulating or automation	No	Yes	No	No	Yes	Yes
Offline connectivity	No	Yes	No	Yes	No	Yes
Cost effectiveness	No	No	Yes	No	Yes	Yes

Table 2.1 Comparison between proposed IoT device and other existing applications

2.7 Recommendations and Summary

After studying five types of similar existing applications which related to customisable IoT platform, all of them have served their purposes to provide customisation for IoT solution. However, all of them are rather cost ineffective, have limited type of sensor/actuator for integration, not user-friendly or not providing modifications. In addition, some of the existing applications don't support any actuator, while some of them cannot allow user to manually control the connected actuators.

IoT in a Box can be improved by introducing an IoT device with multiple sensors connected. Thus it will be more cost effective when a user needs two or more sensors in one location. Furthermore, IoT in a Box also needs to include IoT actuators such as light emitting actuator, so user can turn on or off the light whenever they want. Moreover, IoT in a Box can invent an IoT device which common IoT sensors can be added for cost effective customisation purpose.

Skydrop Smart Watering Sprinkler Controller should be designed to give full control of sprinkler system for user. For instance, the smart controller can be designed to have some buttons to let user stops or resumes a watering session, in other words, turns on or off the sprinklers manually. In addition, the smart controller should support other type of sensors to improve watering schedule.

VersaSense Wireless Device can be designed to have the ability to support common Type-A USB interface other than just Micro-USB interface, thus the solution can be scaled up by adding common USB hub. Besides, VersaSense Wireless Device can be more powerful if it can identify multiple same type of sensor are connected and send notification for user to remove unnecessary sensors. Moreover, VersaSense Wireless Device should provide offline connectivity such as Bluetooth connectivity. If there is no Internet or Wi-Fi connection, user still can interact with the IoT devices via Bluetooth or other type of offline communication protocols.

Ezblock Pi should have Internet or Wi-Fi connection, so that user can do configuration or control their IoT project from a far distance. In addition, if Ezblock Pi has Internet connection, it can publish data of connected sensors to the cloud for user to do data analysis. The app for Ezblock Pi is user-friendly, but it is lack of user-friendliness element on Ezblock Pi. Thus, the company should provide a user-friendly hardware platform for user to connect sensors and actuators in order to reduce the learning gap for user to develop a custom IoT project.

Tuya Smart WIFI Power Strip should provide physical on off buttons for each power supply slot in order to let user turn on or off a power supply when it is offline. Furthermore, the smart power strip should have cloud connectivity in order to collect data for user to manage their power on schedule.

Therefore, the proposed solution will consist of an IoT device and some USB-enabled sensor/actuator modules in order to solve user-friendliness problem and provide customisable IoT platform. The proposed solution will be cost-effective, scalable, ease of hardware modification and support multiple sensor and actuators. The proposed IoT device will provide Cloud connectivity, offline connectivity, support self-regulating or automation while the USB-enabled sensor/actuator modules will be designed to provide plug-and-play features.

The proposed IoT device will be able to connect many different types of sensors and actuators, which are USB-enabled plug-and-play sensor/actuator modules. The USB-enabled plug-and-play sensor/actuator modules can be connected to the proposed solution without technical skills in order to reduce learning gap for user to develop custom IoT solution. With Cloud connectivity, the readings of sensors collected by the proposed IoT device from the USB-enabled sensor modules will be published to Cloud and the connected USB-enabled actuator modules can be manually controlled by mobile app wirelessly. By using the proposed IoT device, user can develop their custom IoT solution faster and setup schedulers for sensor readings easily. Besides, by using cheap and powerful microcontrollers to build the USB-enabled sensor/actuator modules, it will be cost effective to develop a custom IoT solution via the proposed IoT solution.

CHAPTER 3: SYSTEM DESIGN

3.1 System Architecture Design

3.1.1 Google Cloud Architecture

In order to turn this Raspberry Pi computer into an IoT device that can publish sensor data to Firebase Real-Time Database as well as receive commands from Google Cloud Platform, it must first register to the Cloud IoT Core for secure device connection with Google Cloud Platform. For global messaging and event ingestion, the IoT device must subscribe to one of the topic generated for specific communication in order to receive or publish a message via Cloud Pub/Sub service. By using topics, telemetry data can be categorised into different categories and enables efficient data management. For instance, sensor readings will be published to topic that only process sensor readings and device status messages will be published to device status message topic.

The sensor readings and device status of the proposed solution will be stored into the Firebase Real-Time Database. The data is then being retrieved by the monitoring mobile application developed by another team member for monitoring purposes. The Google Cloud Function will be triggered by the Firebase Real-Time Database based on the events happened in database. For instance, when the state of a USB-enabled actuator module connected to the proposed IoT device is changed from "off" to "on" through the monitoring mobile application, a Google Cloud Function will be triggered to send the command to turn on the USB-enabled actuator module. The Google Cloud Function will also subscribe to a telemetry topic in Cloud Pub/Sub service in order to receive sensor readings published by the proposed IoT device. The Google Cloud Function will process and store the readings into Firebase Real-Time Database.

In addition, the Cloud Scheduler is used by the proposed solution in order to achieve automation between the proposed IoT devices. The Cloud Scheduler will be configured to trigger specific Cloud Function periodically in order to check whether the conditions are met. By setting some conditions based on sensor readings, the Google Cloud Function will perform corresponding actions set by user via the monitoring mobile application developed by another team member.

Moreover, due to the proposed solution can integrate with a USB web camera, the Cloud Storage is used to receive and store the image captured by the camera. The proposed IoT device will use the connected USB camera to capture some images and then publish the image to the Cloud Storage. Furthermore, a web server will be hosted in the Google Cloud Platform in order to receive live video streams from the proposed IoT device. By using the Compute Engine service provided by the Google Cloud Platform, a RTMP web server can be setup to receive live video streams as well as enable the monitoring mobile application developed by another team member to retrieve the video feed.

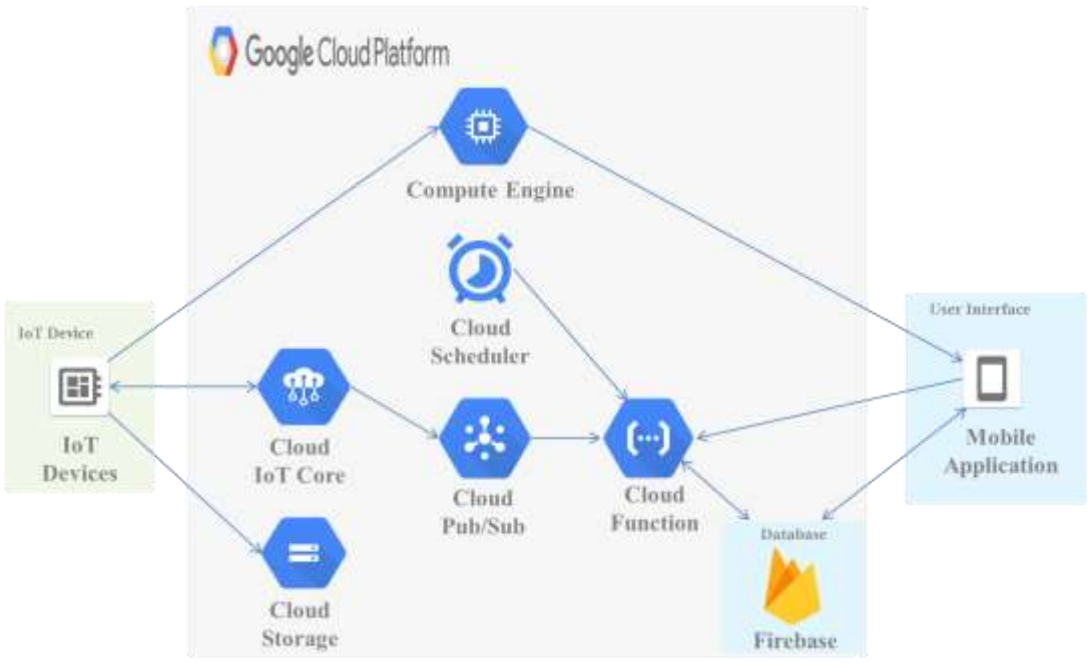


Figure 3.1 Google Cloud Architecture Diagram

3.1.2 Use Case Diagram

Figure 3.2 shows a system use case diagram that represents possible use cases for the proposed solution. User can turn on the Bluetooth connectivity of the proposed IoT device for offline communication. Besides, user can add or remove any number of the proposed USB-enabled sensor/actuator modules from the proposed IoT device. User also can set some thresholds to control the actuations of the proposed USB-enabled actuator modules based on sensor readings collected by the proposed USB-enabled sensor modules. User also can control the actuation of the proposed solution manually. Furthermore, user can add a USB web camera to the proposed solution and use the camera to capture images or streaming live video to web server.

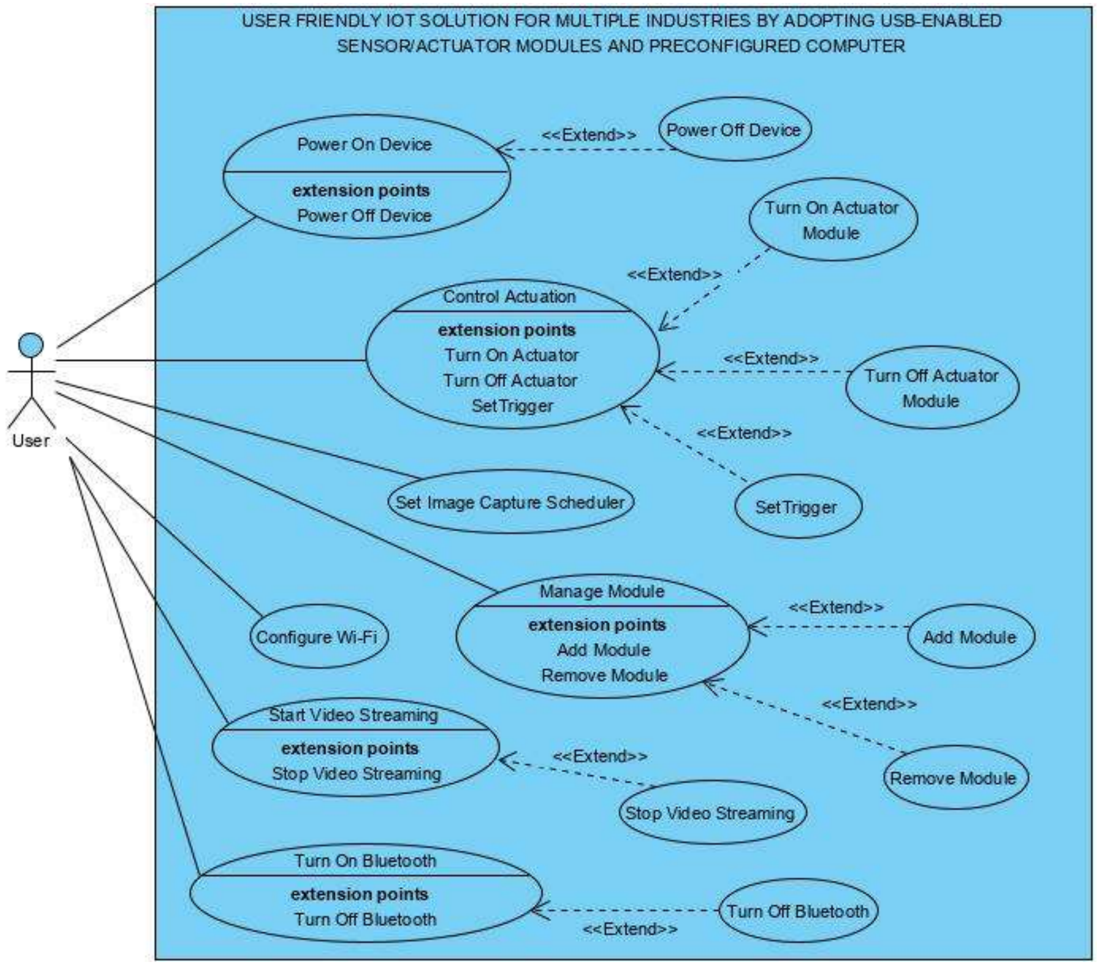
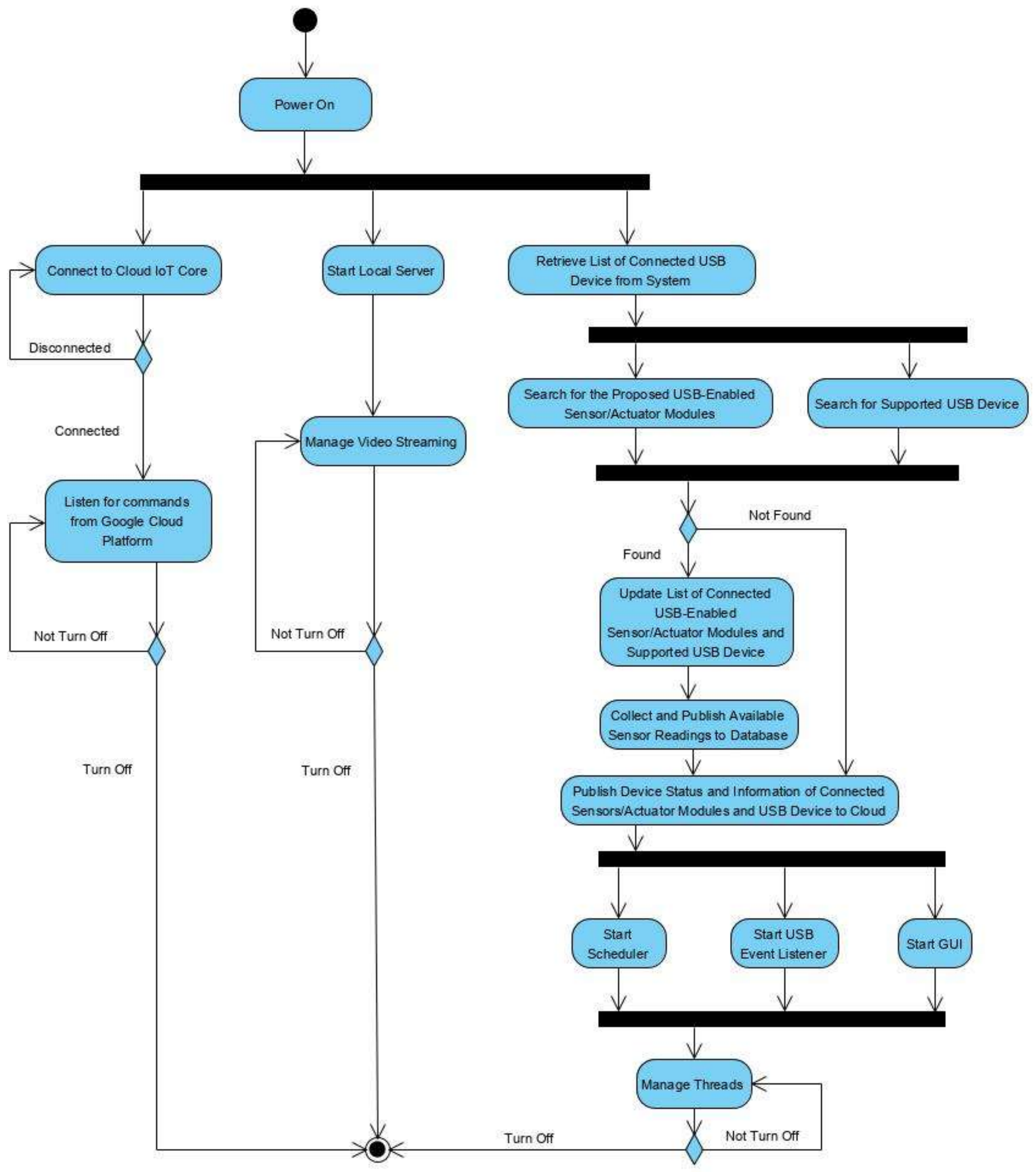


Figure 3.2 Use Case Diagram

3.1.3 System Flowcharts

3.1.3.1 Power on Raspberry Pi Computer (Master Device)

Figure 3.3 shows the activities of main program in Raspberry Pi computer after powered on by user. The main program connects Raspberry Pi computer to Cloud IoT Core. It will try to connect Cloud IoT Core until success. The local server will be started in order to provide video streaming services. Meanwhile, the main program scans the USB ports in search of the proposed USB-enabled sensor/actuator modules. The main program also will search for other supported USB device, such as USB web camera that has plugged-in to the proposed solution. After that, the main program will update the device status and publish the information to Firebase Real-Time Database once it is online and starts to listen for commands from Google Cloud Platform. The main program also will start the scheduler for sensor readings and create a listener to detect USB events, such as USB plug-in or USB unplug events happen in Raspberry Pi computer.



Power On
Connect to Cloud IoT Core
Disconnected
Connected
Listen for commands from Google Cloud Platform
Not Turn Off
Turn Off
Start Local Server
Manage Video Streaming
Not Turn Off
Turn Off
Retrieve List of Connected USB Device from System
Search for the Proposed USB-Enabled Sensor/Actuator Modules
Search for Supported USB Device
Not Found
Found
Update List of Connected USB-Enabled Sensor/Actuator Modules and Supported USB Device
Collect and Publish Available Sensor Readings to Database
Publish Device Status and Information of Connected Sensors/Actuator Modules and USB Device to Cloud
Start Scheduler
Start USB Event Listener
Start GUI
Manage Threads
Turn Off
Not Turn Off

Figure 3.3 Flowchart of Activity in Raspberry Pi after Powered on

3.1.3.2 Scheduler in Raspberry Pi Computer

Figure 3.4 shows the activities in the scheduler that will publish collected sensor readings to Firebase Real-Time Database when the scheduled time is reached. The sensor readings collected from the proposed USB-enabled sensor modules (slave devices) will be appended into an array in JSON format for data structure. Thus, the sensor readings will be published in a batch. In addition, no Internet traffic will be generated if there is no sensor reading being collected.

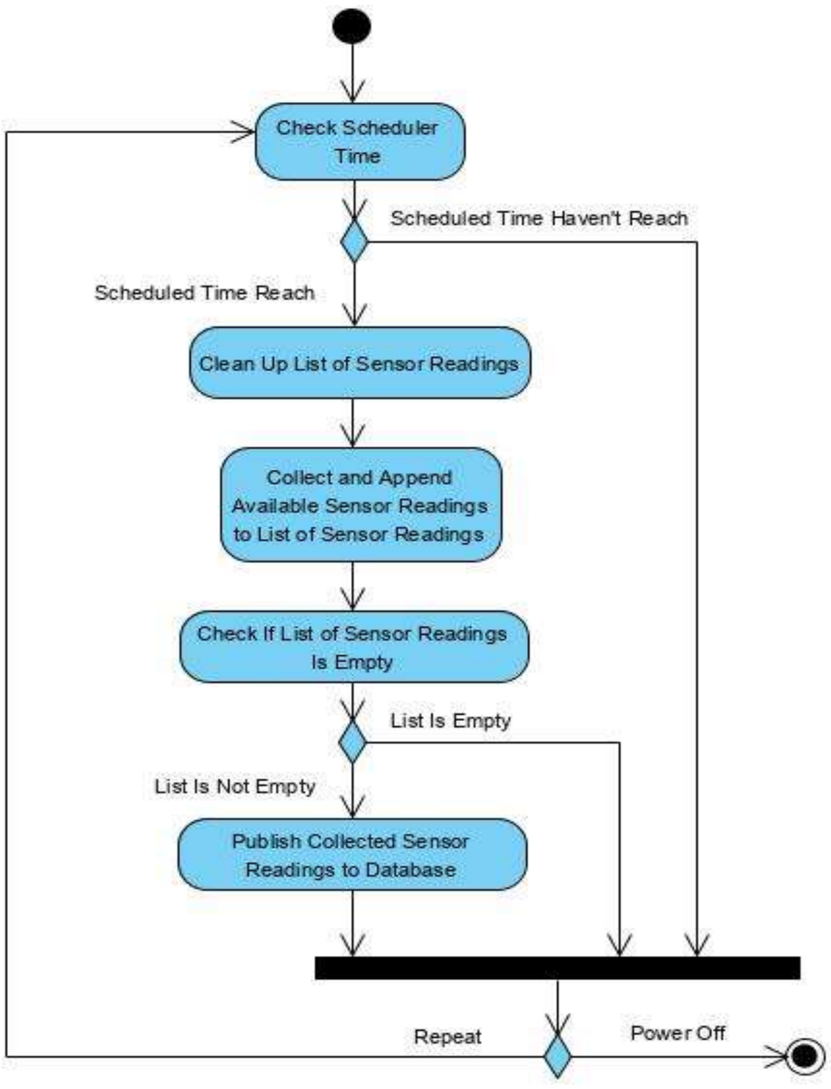

Check Scheduler Time
Scheduled Time Haven't Reach
Scheduled Time Reach
Clean Up List of Sensor Readings
Collect and Append Available Sensor Readings to List of Sensor Readings
Check If List of Sensor Readings Is Empty
List Is Empty
List Is Not Empty
Publish Collected Sensor Readings to Database
Repeat
Power Off

Figure 3.4 Flowchart of Scheduler in Raspberry Pi

3.1.3.3 Add Modules to Raspberry Pi Computer

Figure 3.5 shows the activities of main program while a USB device is plugged-in to the proposed solution. First the main program of Raspberry Pi computer (master device) will verify whether the plugged-in USB device is supported by the proposed solution. If the USB device is the proposed USB-enabled sensor/actuator module, then the main program will send a command to the module for reporting necessary information to enable the module be identified by the main program. After that, the main program will determine whether the connected module is the proposed USB-enabled actuator module or the proposed USB-enabled sensor module by using the information. Finally, the information of the connected USB-enabled sensor/actuator module will be appended to a list, then the device status will be updated and publish to the Firebase Real-Time Database. If it is not the USB-enabled sensor/actuator module but it is supported by the proposed solution, then the main program will collect and update the information of connected USB device to database.

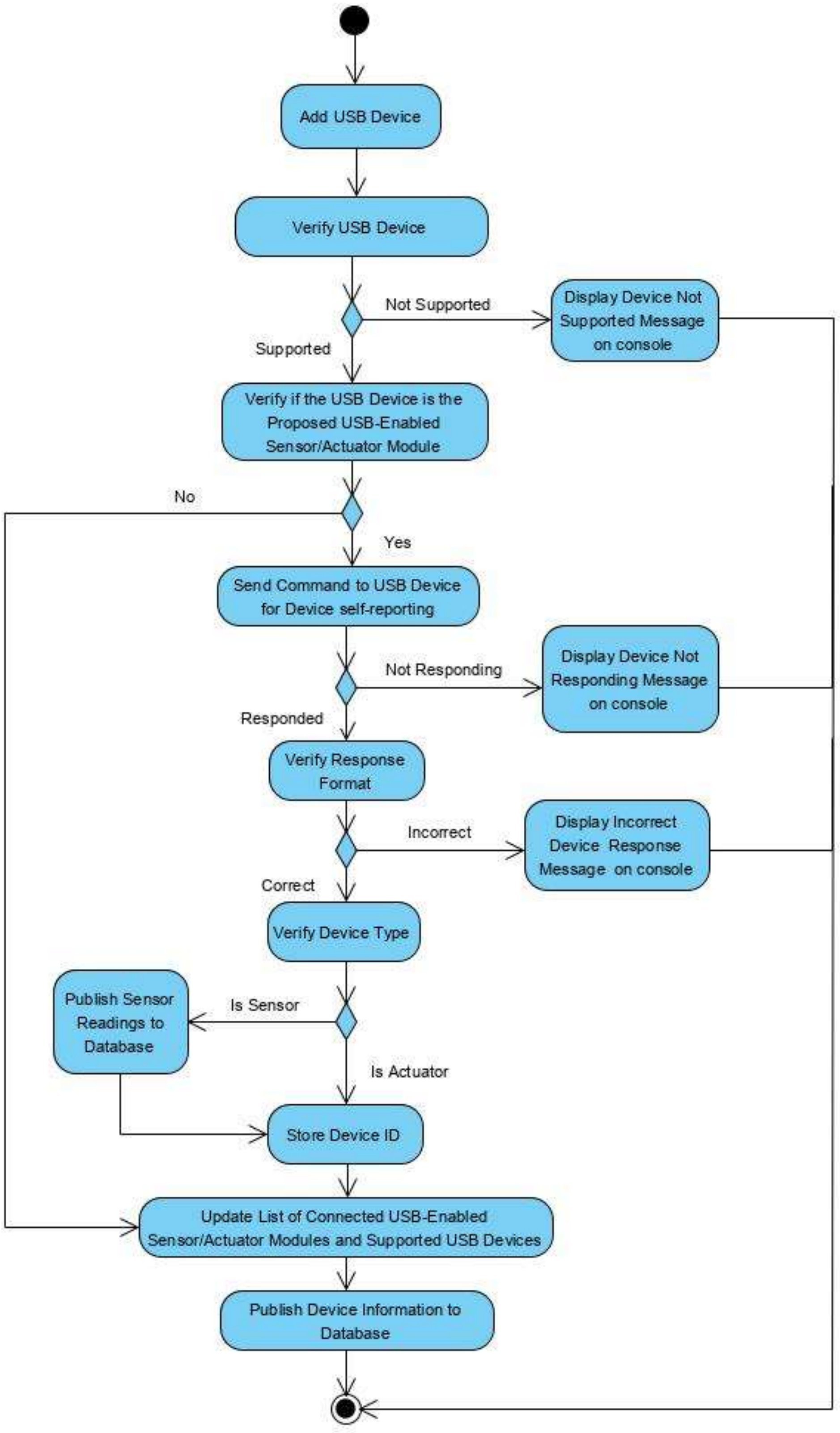

Add USB Device
Verify USB Device
Not Supported
Display Device Not Supported Message on console
Supported
Verify if the USB Device is the Proposed USB-Enabled Sensor/Actuator Module
No
Yes
Send Command to USB Device for Device self-reporting
Not Responding
Display Device Not Responding Message on console
Responded
Verify Response Format
Incorrect
Display Incorrect Device Response Message on console
Correct
Verify Device Type
Publish Sensor Readings to Database
Is Sensor
Is Actuator
Store Device ID
Update List of Connected USB-Enabled Sensor/Actuator Modules and Supported USB Devices
Publish Device Information to Database

Figure 3.5 Flowchart of Adding Modules to Raspberry Pi Computer

3.1.3.4 Remove Modules from Raspberry Pi Computer

Figure 3.6 shows activities of main program when a USB-enabled sensor or actuator is removed from Raspberry Pi computer. When USB removed event is detected by the main program, it first check if the USB device exists in the list of connected USB-enabled sensor/actuator modules and list of supported USB devices. Then the main program will remove the existence of the USB device if it existed in the lists. Finally, update device status to database. If the USB device is not exists in the list, this means the USB device, which is not supported by the proposed solution, was plugged-in to the system.

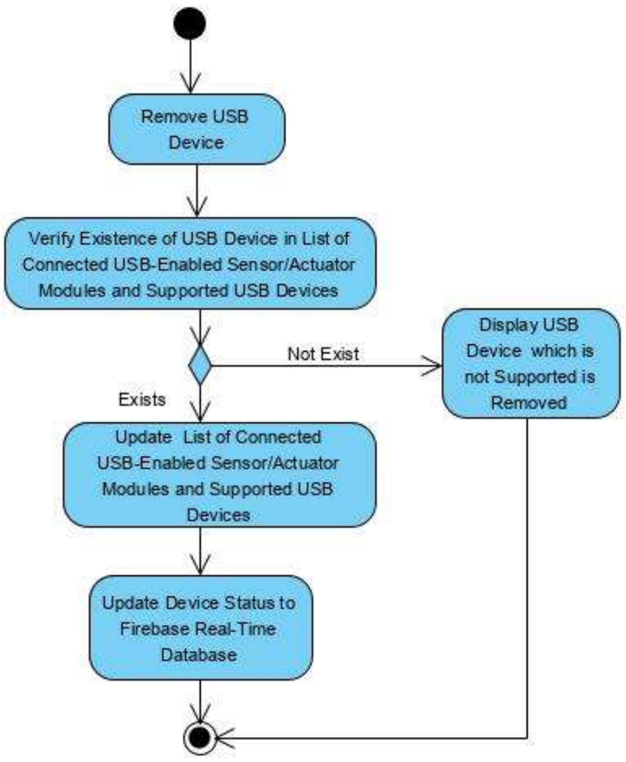

Remove USB Device
Verify Existence of USB Device in List of Connected USB-Enabled Sensor/Actuator Modules and Supported USB Devices
Not Exist
Display USB Device which is not Supported is Removed
Exists
Update List of Connected USB-Enabled Sensor/Actuator Modules and Supported USB Devices
Update Device Status to Firebase Real-Time Database

Figure 3.6 Flowchart of Removing Modules from Raspberry Pi Computer

3.1.3.5 Actuation Control via Internet

Figure 3.7 shows the activities of main program after receiving commands from user via the monitoring mobile application developed by another team member. The commands are sent to the Google Cloud Platform and Firebase Real-Time Database and then passed to the proposed IoT device, which is the customised Raspberry Pi computer. Once the main program in Raspberry Pi computer receives the commands from Cloud, it first check the existence of target USB-enabled actuator module in the list of connected and it will ignore the command if the target actuator module is not exists. If the target USB device exists, the main program will verify the command and transfer the command to the target USB device via USB communication in order to control the actuation. Alternatively, the actuation of the proposed solution can be controlled by using the GUI developed in the proposed IoT device for debugging purposes.

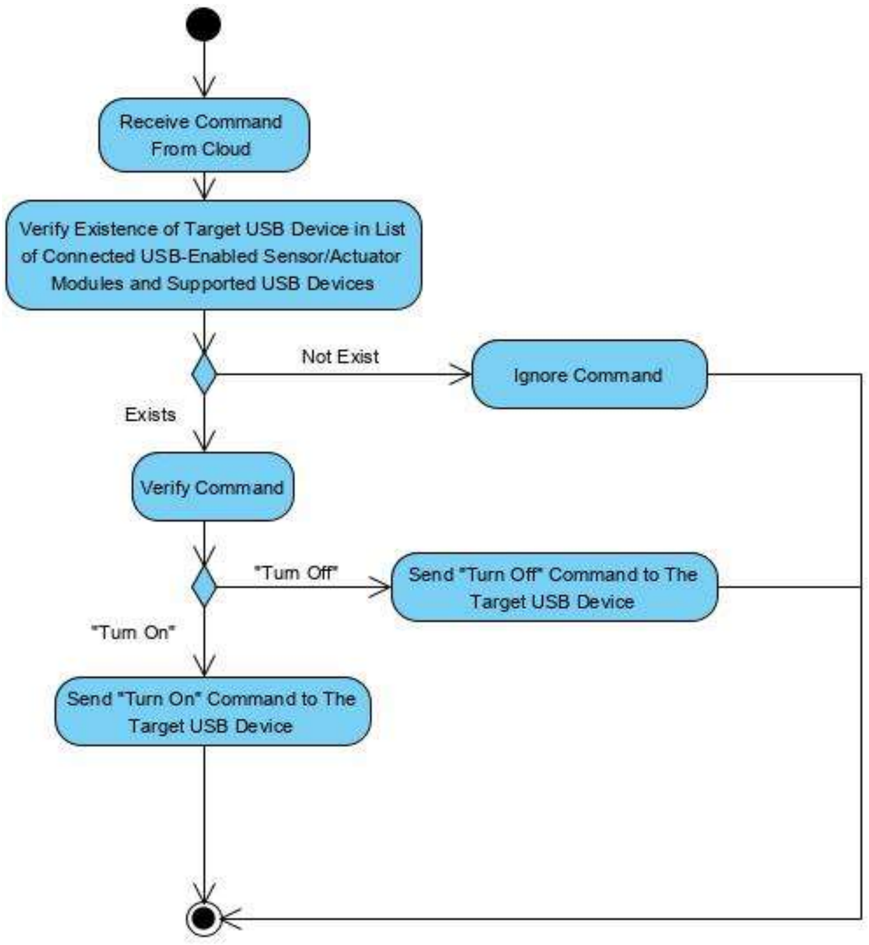

Receive Command From Cloud
Verify Existence of Target USB Device in List of Connected USB-Enabled Sensor/Actuator Modules and Supported USB Devices
Not Exist
Ignore Command
Exists
Verify Command
"Turn Off"
Send "Turn Off" Command to The Target USB Device
"Turn On"
Send "Turn On" Command to The Target USB Device

Figure 3.7 Flowchart of Actuation Control via Internet

3.1.3.6 Turn on Bluetooth Connectivity

Figure 3.8 shows the activities of main program after user has pressed the button implemented on Raspberry Pi computer to turn on Bluetooth connectivity. Bluetooth connectivity will be turn on by the main program of Raspberry Pi computer and the discoverability will be enabled. Then the main program waits for incoming Bluetooth connection from other Bluetooth device, in this case is the monitoring mobile application developed by another student. The Bluetooth communication will be establish if received password is correct. The Bluetooth will be turned off once the user presses again the switch implemented on Raspberry Pi computer.

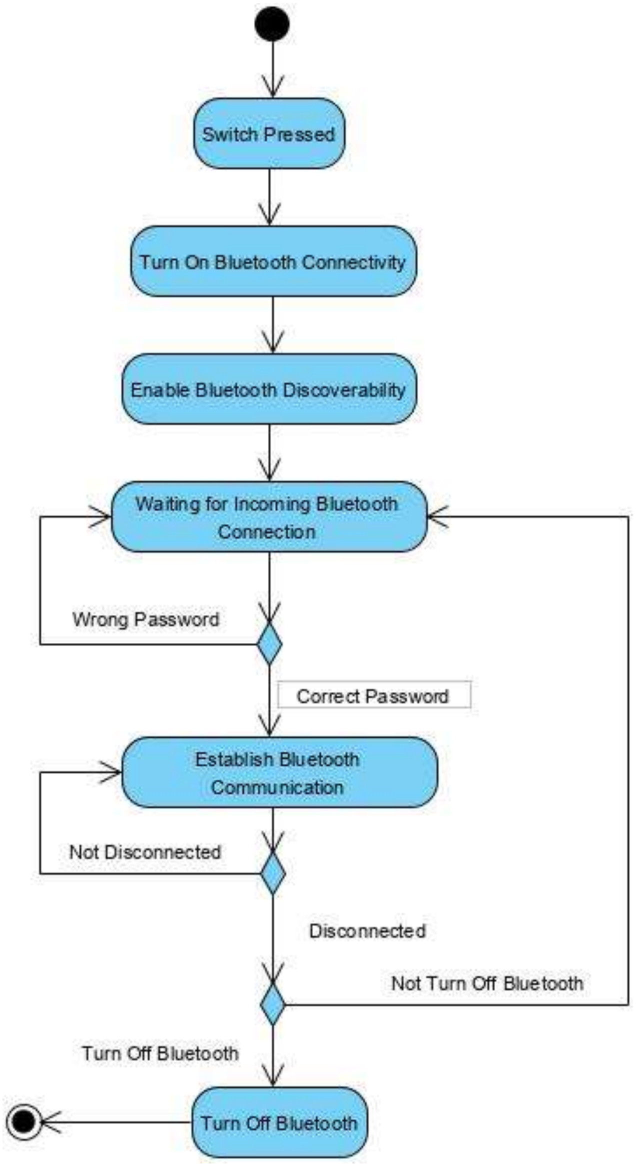

Switch Pressed
Turn On Bluetooth Connectivity
Enable Bluetooth Discoverability
Waiting for Incoming Bluetooth Connection
Wrong Password
Correct Password
Establish Bluetooth Communication
Not Disconnected
Disconnected
Not Turn Off Bluetooth
Turn Off Bluetooth
Turn Off Bluetooth

Figure 3.8 Flowchart of Bluetooth Connectivity

3.1.3.7 Configure Wi-Fi

Figure 3.9 shows the activities of main program after the user passed the Wi-Fi configuration details via Bluetooth. When the main program of Raspberry Pi computer received Wi-Fi configuration information, the main program uses the information to configure and connect to the Wi-Fi. If cannot connect to Wi-Fi and cannot successfully ping Google.com, then it will send failure message to connected Bluetooth device, in this case is the monitoring mobile application developed by another team member. If successfully connected to Wi-Fi and pinged Google.com, then send successful status message to the app.

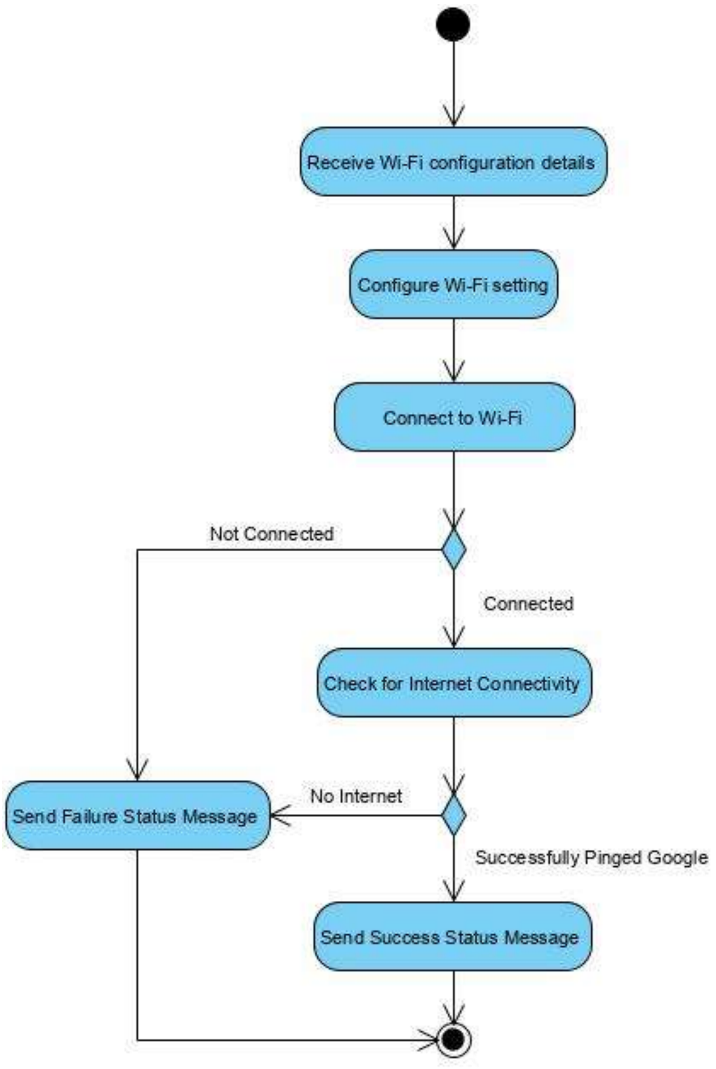

Receive Wi-Fi configuration details
Configure Wi-Fi setting
Connect to Wi-Fi
Not Connected
Connected
Check for Internet Connectivity
No Internet
Send Failure Status Message
Successfully Pinged Google
Send Success Status Message

Figure 3.9 Flowchart of Wi-Fi Configuration

3.1.3.8 Activities of USB-Enabled Sensor Module (Slave Device)

Figure 3.10 shows the activities of Digispark ATTINY85 Mini USB Development Board that integrated with a sensor after being plugged-in to the Raspberry Pi computer. The program will keep refreshing and keep reading its USB buffer periodically to check whether a command is received. Based on the commands received from master device, the Digispark ATTINY85 Mini USB Development Board will first verify whether the commands are valid and then perform corresponding actions such as report sensor readings, reports its information, update its ID, and reset its ID to default ID.

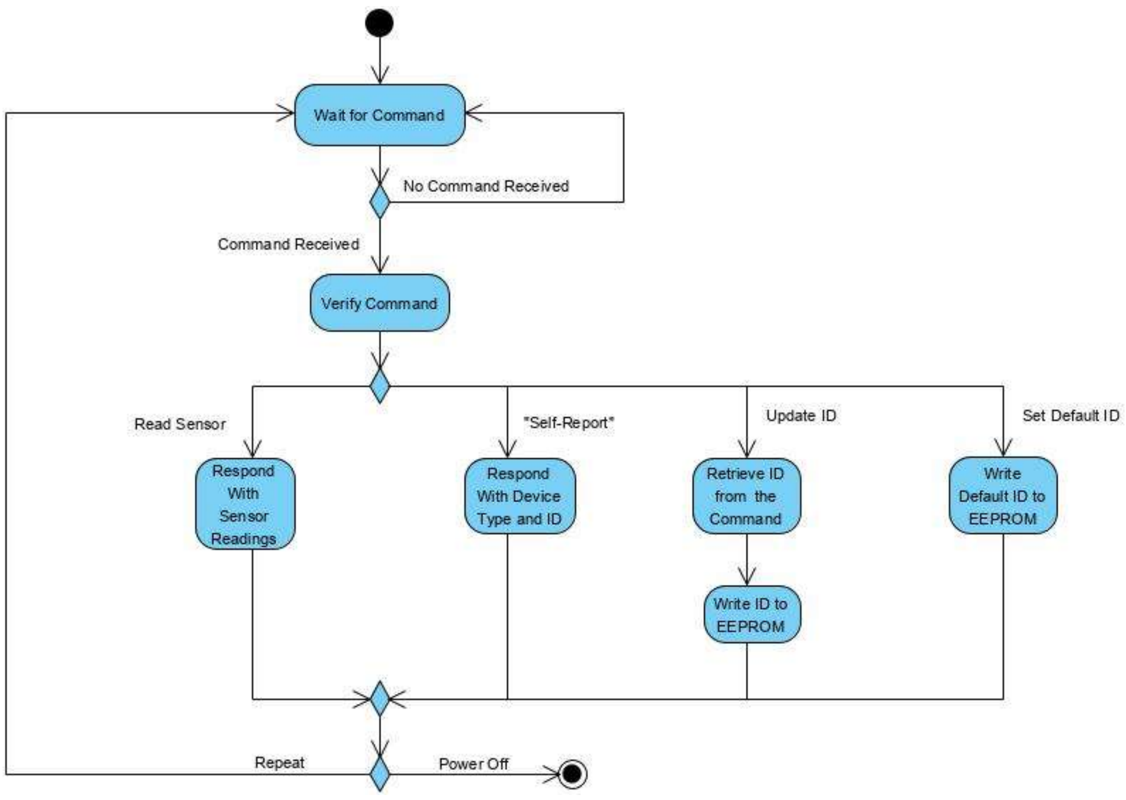


Figure 3.10 Flowchart of Activities in USB-Enabled Sensor Module

3.1.3.9 Activities of USB-Enabled Actuator Module (Slave Device)

Figure 3.11 shows the activities of Digispark ATTINY85 Mini USB Development Board which has integrated with an actuator after being plugged-in to the Raspberry Pi computer. The program will keep refreshing and keep reading its USB buffer periodically to check whether a command is received. Based on the commands received from master device, the Digispark ATTINY85 Mini USB Development Board will first verify whether the commands are valid and then perform corresponding actions such as turning on/off connected actuator, reports its information, update its ID, and reset its ID to default ID.

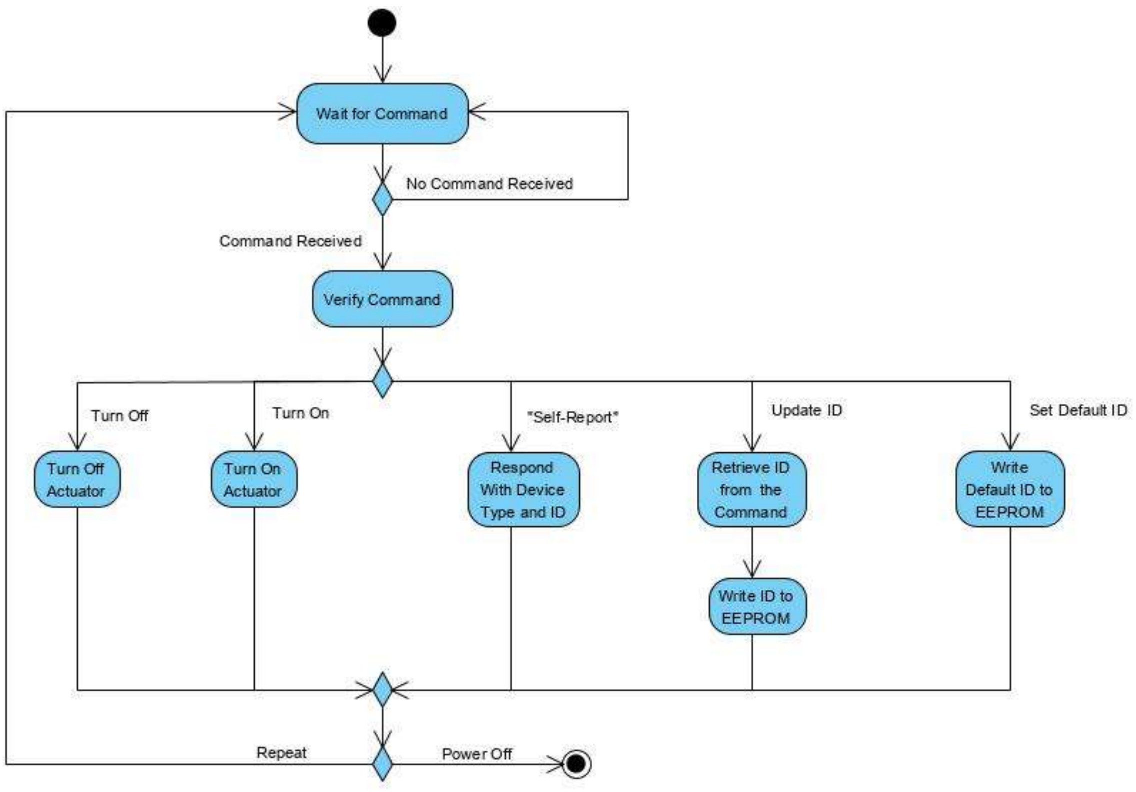


Figure 3.11 Flowchart of Activities in USB-Enabled Actuator Module

3.1.4 System Block Diagram

Figure 3.12 shows the block diagram that represents the connection between Google Cloud Platform, mobile application, Raspberry Pi computer, USB-enabled microcontroller, sensors and actuators. The sensors and actuators are connected to USB-enabled microcontroller, Digispark ATTINY85 Mini USB Development Board, by using jump cables to form the proposed USB-enabled sensor/actuator modules. The USB-enabled sensor/actuator modules are plugged-in to the Raspberry Pi computer via Type-A USB cables for data communication. The Raspberry Pi computer is configured to become the proposed IoT device that performs two way communications with Google Cloud Platform via the Internet provided by Wi-Fi. Besides, the proposed IoT device can use Bluetooth to communicate with the mobile application developed by another team member.

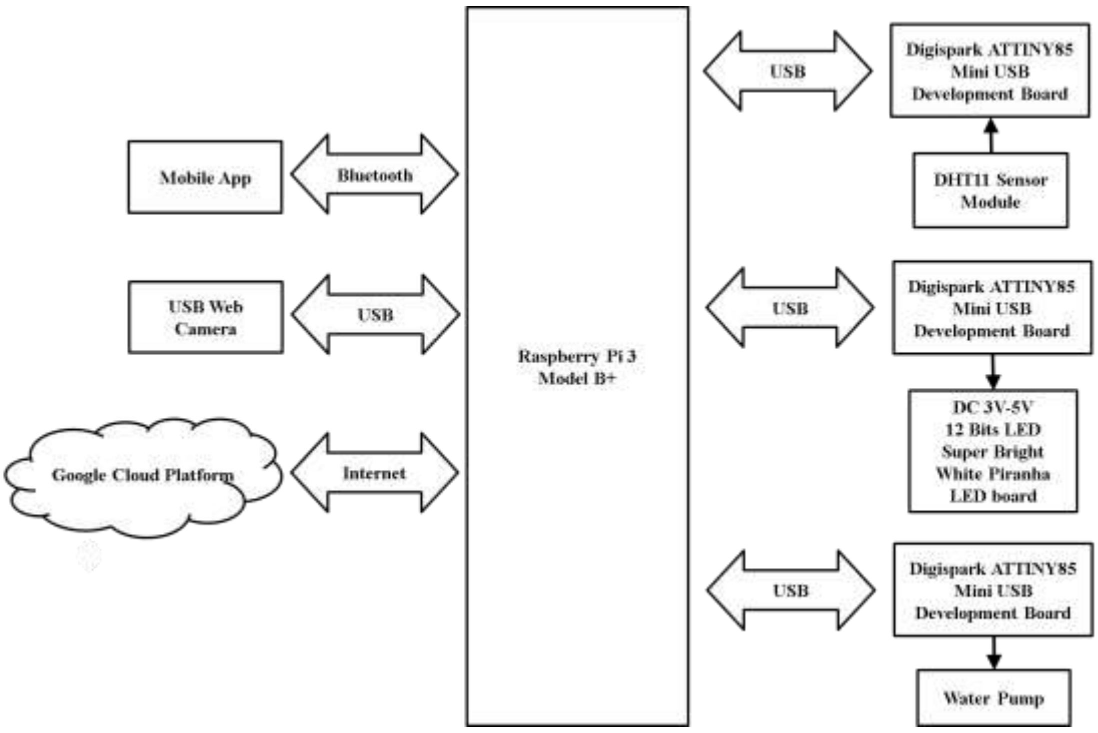

Mobile App
Bluetooth
USB Web Camera
USB
Google Cloud Platform
Internet
Raspberry Pi 3 Model B+
USB
Digispark ATTINY85 Mini USB Development Board
DHT11 Sensor Module
USB
Digispark ATTINY85 Mini USB Development Board
DC 3V-5V 12 Bits LED Super Bright White Piranha LED board
USB
Digispark ATTINY85 Mini USB Development Board
Water Pump

Figure 3.12 Block Diagram of the Proposed Solution

3.1.5 Schematic Diagram

3.1.5.1 Raspberry Pi Computer and Switch (Master Device)

Figure 3.13 shows the wiring of a switch and LEDs implemented to Raspberry Pi computer for turning on Bluetooth connectivity. The LEDs are using source method to connect to GPIO pins of Raspberry Pi computer that provide logic 1 and 0 to turn on/off LEDs for Bluetooth status indication. The switch is connected to GPIO 10 of Raspberry Pi computer and once it is pressed, the Raspberry Pi computer will turn on its Bluetooth.

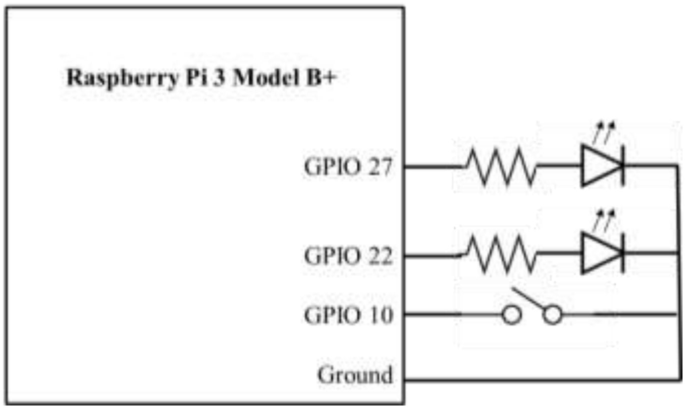


Figure 3.13 Schematic Diagram of Bluetooth Switch

3.1.5.2 USB-Enabled Temperature and Humidity Sensor Module (Slave Device)

Figure 3.14 shows the wiring between Digispark ATTINY85 Mini USB Development Board and DHT11 sensor module. Due to the DHT11 sensor module will transfer sensor readings by using serial communication, thus its data pin is connected to pin 2 of Digispark ATTINY85 Mini USB Development Board which can perform serial communication. Besides, the VCC power input pin of the DHT11 sensor module is connected to 5V pin of the microcontroller for power supply, and the Ground pins are connected to each other.

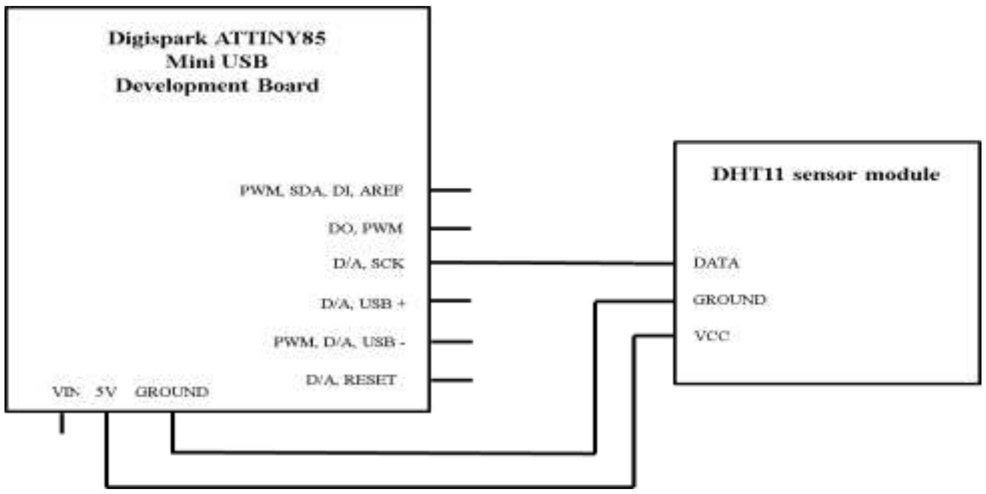


Figure 3.14 Schematic Diagram of USB-Enabled Temperature & Humidity Sensor Module

3.1.5.2 USB-Enabled Light Sensor Module (Slave Device)

Figure 3.15 shows the wiring between Digispark ATTINY85 Mini USB Development Board and LDR module. In order to collect and describe the intensity of light in percentage, analog output pin of the LDR module is used. The analog data of the LDR module is connected to pin 2 of the Digispark ATTINY85 Mini USB Development Board due to the pin can receive analog input. The analog data is then being converted into meaningful brightness value based on the voltage supplied to the LDR module via the microcontroller's power output pin.

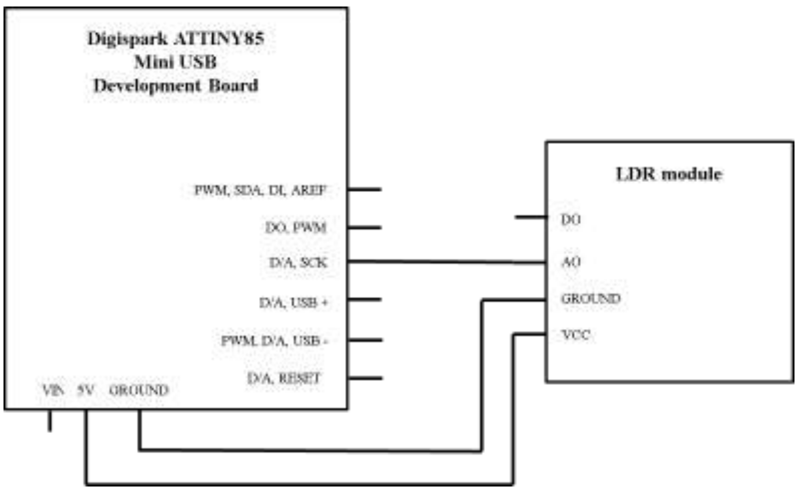


Figure 3.15 Schematic Diagram of USB-Enabled Light Sensor Module

3.1.5.2 USB-Enabled Water Sprinkler Module (Slave Device)

Figure 3.16 shows the wiring between Digispark ATTINY85 Mini USB Development Board and water sprinkler. Due to the water sprinkler is drawing current very effectively, thus an extra circuit is implemented to prevent the microcontroller being damaged. A transistor is used to control the current drawn by the water sprinkler, and the diode is used to control the flow of current. The base input of transistor is connected to pin 1 of the microcontroller to turn on or off the transistor.

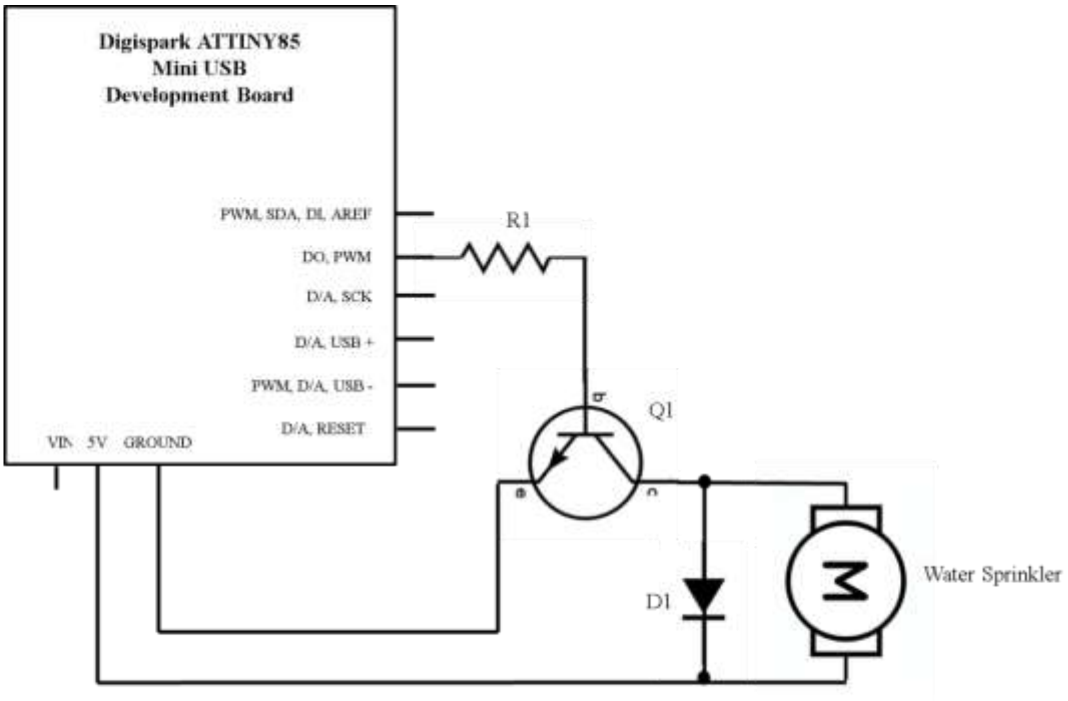


Figure 3.16 Schematic Diagram of USB-Enabled Sprinkler Module

3.1.5.3 USB-Enabled LED Module (Slave Device)

Figure 3.17 shows the wiring between Digispark ATTINY85 Mini USB Development Board and DC 3V-5V 12 Bits LED Super Bright White Piranha LED board. Due to the LED board has built-in resistor, the LED board is directly connected to the microcontroller. The positive pin of the LED board is connected to pin 1 of the Digispark ATTINY85 Mini USB Development Board via a jumper wire due to pin 1 of the microcontroller can provide Logic 1 to for “on” state as well as pulse width modulation. The negative pin of LED board is connected to the Ground of the microcontroller to complete the circuit.

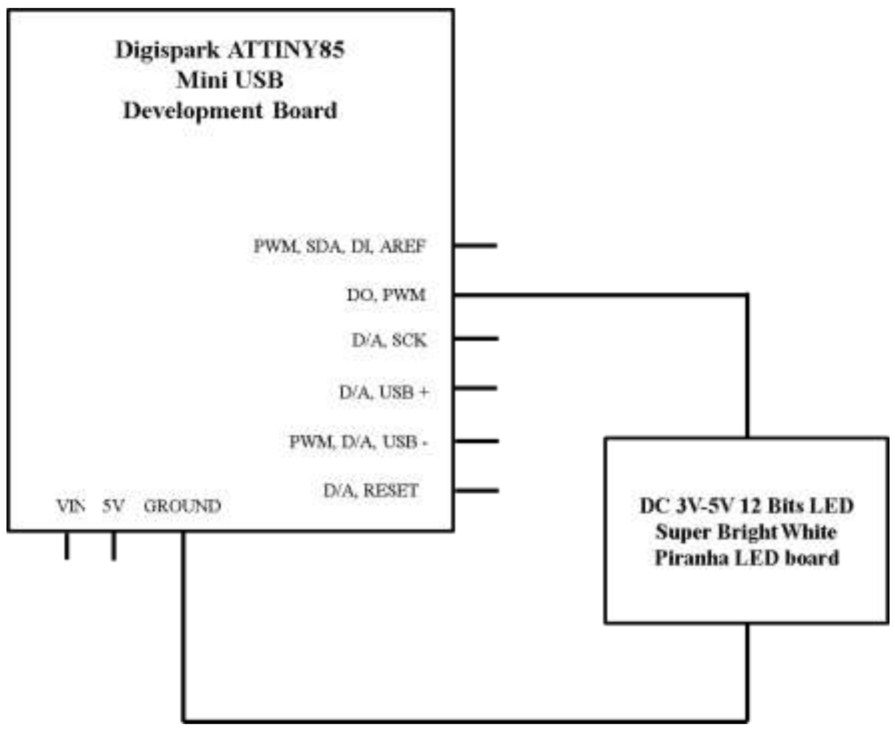


Figure 3.17 Schematic Diagram of USB-Enabled LED Module

3.2 GUI design for Debugging Purposes

The end user of the proposed solution will monitor and control the proposed solution through the monitoring mobile application developed by another team member, thus this Graphic User Interface (GUI) will not be used by end user. However it is useful for developer to debug the system through by using GUI. This GUI is designed to be update its components dynamically based on the event happened in the main program. The main components are the display panel of a list that contains the connected USB-enabled sensor/actuator module and supported USB device such as USB web camera. The panel will create or destroy a component based on the change in content of the list. Besides, this GUI also will provide status of Internet, Bluetooth and Wi-Fi connectivity. The GUI will be designed to enable developer to turn on or off Bluetooth connectivity instead of pressing the button implemented on Raspberry Pi computer. Besides, due to the IP address of RTMP server hosted on Google Cloud Platform will always change, the GUI is designed to let developer to change target IP address for streaming. This GUI will also able to let developer to control the actuator modules via a button in GUI instead of waiting commands from Cloud. Normally, the video streaming is controlled by commands sent by user via the monitoring mobile application. Thus, the GUI is designed to enable developer to stream video directly if a USB camera is connected.

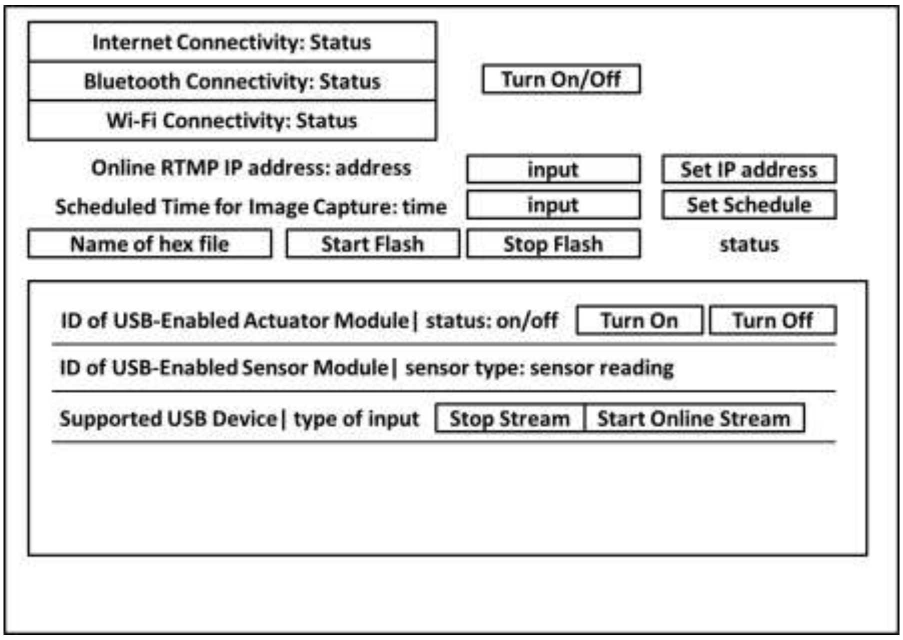

Internet Connectivity: Status
Bluetooth Connectivity: Status
Wi-Fi Connectivity: Status
Turn On/Off
Online RTMP IP address: address
input
Set IP address
Scheduled Time for Image Capture: time
input
Set Schedule
Name of hex file
Start Flash
Stop Flash
status
ID of USB-Enabled Actuator Module | status: on/off
Turn On
Turn Off
ID of USB-Enabled Sensor Module | sensor type: sensor reading
Supported USB Device | type of input
Stop Stream
Start Online Stream

Figure 3.18 GUI design for Debugging Purposes

3.3 Container Design for Small Scale Agriculture

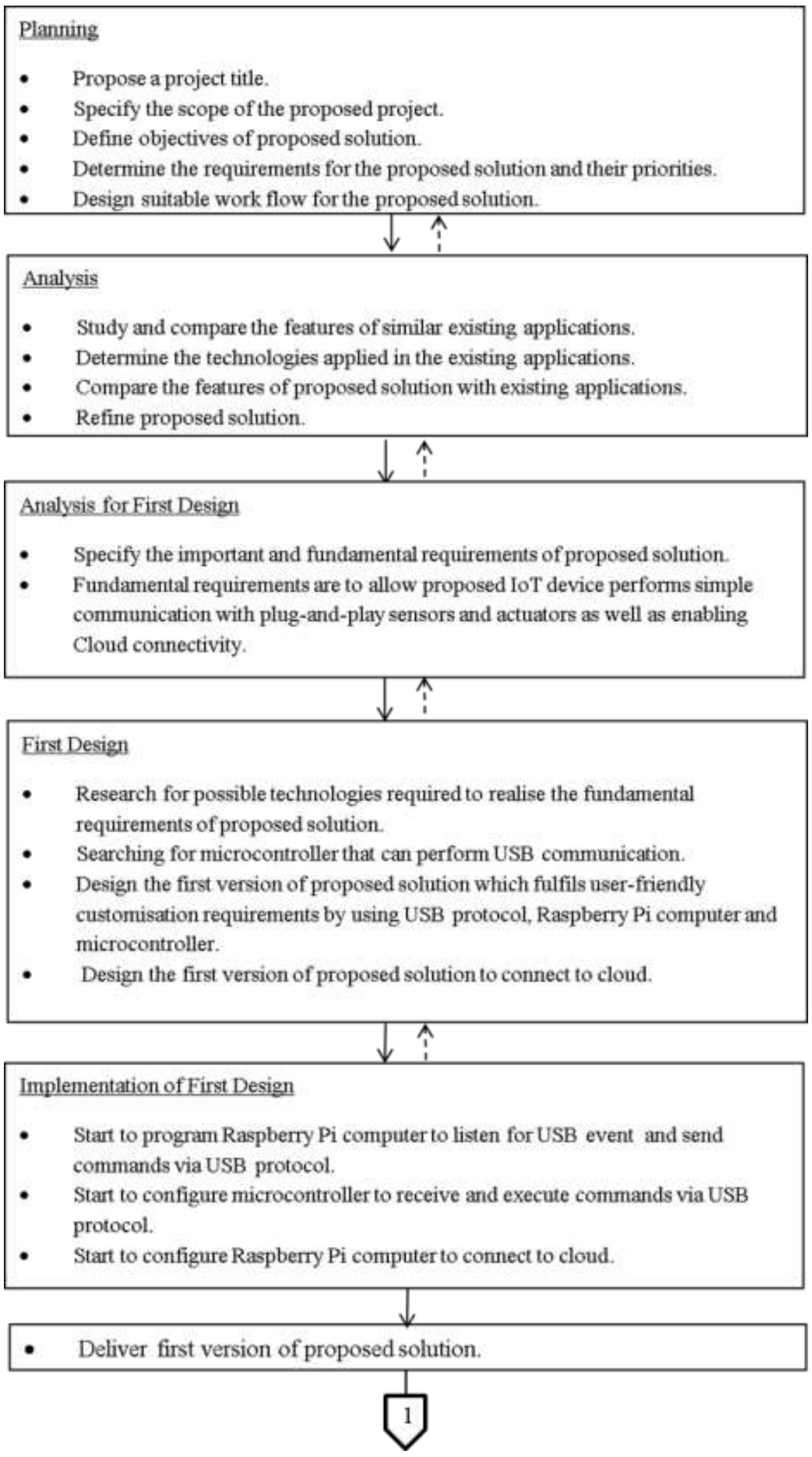
The container will be developed to separate the proposed IoT device from the proposed USB-enabled sensor/actuator modules so that the proposed IoT device can be protected and the proposed USB-enabled sensor/actuator modules are placed near by a high humidity environment. A smaller container will be implemented on top of the cover of the bigger container to place the proposed IoT device. Besides, a special mechanism will be developed under the cover of the bigger container to enable the USB devices to be installed at the most suitable location.

CHAPTER 4: SYSTEM IMPLEMENTATION AND TESTING

4.1 Methodology

In order to develop the proposed solution, Iterative Phased Product Development Model is chosen due to realisation of the proposed solution required multiple phase analysis in order to make sure the final product is working. Iterative Phased Product Development Model divides the requirements of whole project into multiple versions according to the priority of the requirements. Each version of the proposed solution will be developed in order to fulfil several requirements of the final product and the first version must fulfil the most important and fundamental requirements. Before developing the first version of the proposed solution, a planning process must be carried out in order to determine all the requirements for the final product and the most important requirements are chosen to be realised in the first version of the proposed solution. In addition, work flow for developing the proposed solution is carefully planned so that the requirements implemented in each version of the proposed solution are deployed successfully. An analysis will be performed once the first version of the proposed solution is complete, then development of the following version of the proposed solution with lower priority of requirements will be carried out. The processes will repeat by adding more requirements for each version until the final version of the proposed solution is completed. This project consists of two members with each member will achieve different project scope. One member will focus on the project scope related to the development of hardware while another member will focus on the development of mobile application. When both project deliverables are achieved, a complete IoT solution with monitoring mobile application will be delivered.

4.2 Project Workflow in Iterative Phased Product Development Model



Planning

- Propose a project title.
- Specify the scope of the proposed project.
- Define objectives of proposed solution.
- Determine the requirements for the proposed solution and their priorities.
- Design suitable work flow for the proposed solution.

Analysis

- Study and compare the features of similar existing applications.
- Determine the technologies applied in the existing applications.
- Compare the features of proposed solution with existing applications.
- Refine proposed solution.

Analysis for First Design

- Specify the important and fundamental requirements of proposed solution.
- Fundamental requirements are to allow proposed IoT device performs simple communication with plug-and-play sensors and actuators as well as enabling Cloud connectivity.

First Design

- Research for possible technologies required to realise the fundamental requirements of proposed solution.
- Searching for microcontroller that can perform USB communication.
- Design the first version of proposed solution which fulfils user-friendly customisation requirements by using USB protocol, Raspberry Pi computer and microcontroller.
- Design the first version of proposed solution to connect to cloud.

Implementation of First Design

- Start to program Raspberry Pi computer to listen for USB event and send commands via USB protocol.
- Start to configure microcontroller to receive and execute commands via USB protocol.
- Start to configure Raspberry Pi computer to connect to cloud.

- Deliver first version of proposed solution.

Figure 4.1 Project Work Flow for First Version of Proposed Solution in Iterative Phased Product Development Model

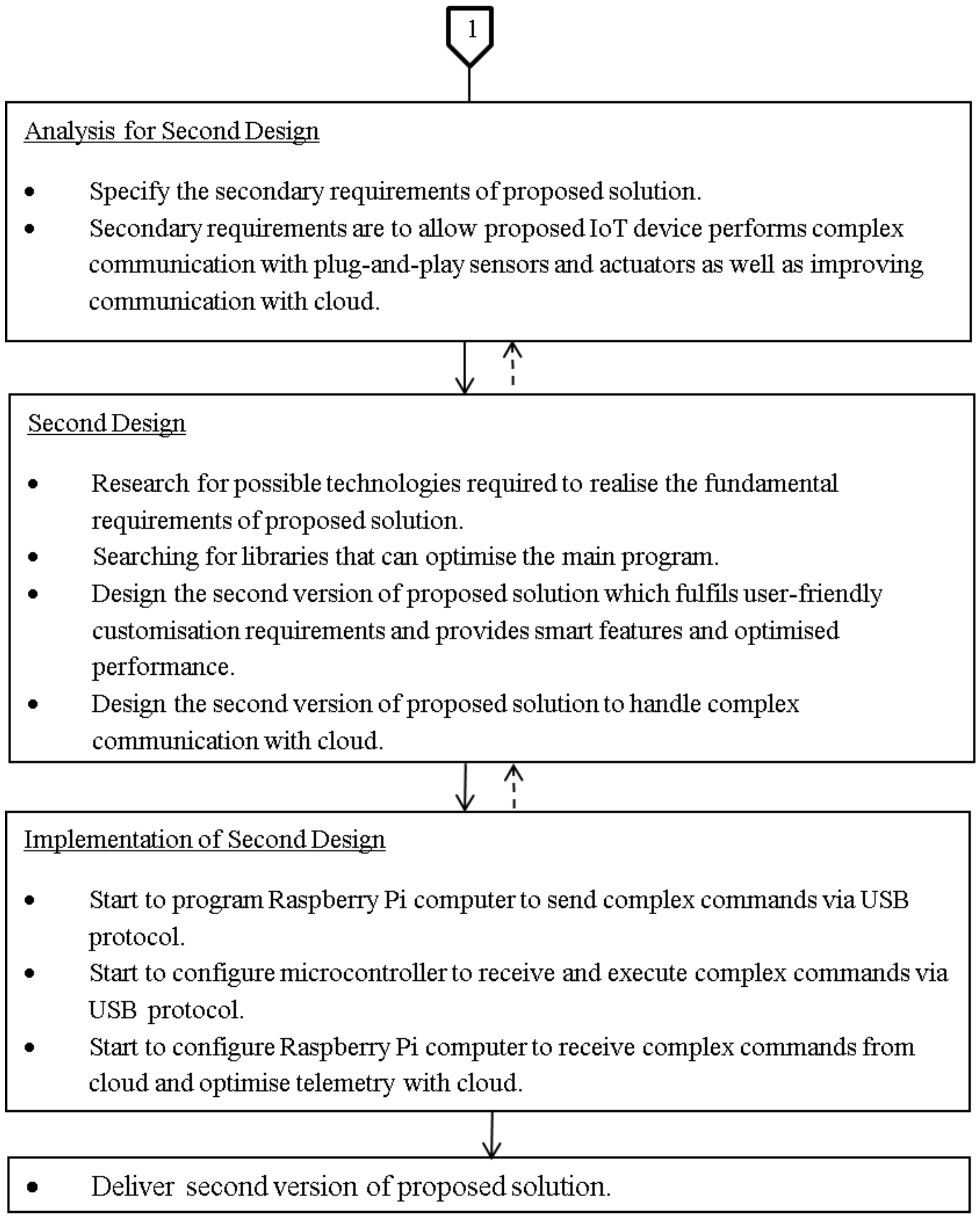

1
Analysis for Second Design
• Specify the secondary requirements of proposed solution.
• Secondary requirements are to allow proposed IoT device performs complex communication with plug-and-play sensors and actuators as well as improving communication with cloud.
Second Design
• Research for possible technologies required to realise the fundamental requirements of proposed solution.
• Searching for libraries that can optimise the main program.
• Design the second version of proposed solution which fulfils user-friendly customisation requirements and provides smart features and optimised performance.
• Design the second version of proposed solution to handle complex communication with cloud.
Implementation of Second Design
• Start to program Raspberry Pi computer to send complex commands via USB protocol.
• Start to configure microcontroller to receive and execute complex commands via USB protocol.
• Start to configure Raspberry Pi computer to receive complex commands from cloud and optimise telemetry with cloud.
• Deliver second version of proposed solution.

Figure 4.2 Project Work Flow for Second Version of Proposed Solution in Iterative Phased Product Development Model

4.3 Project Timeline

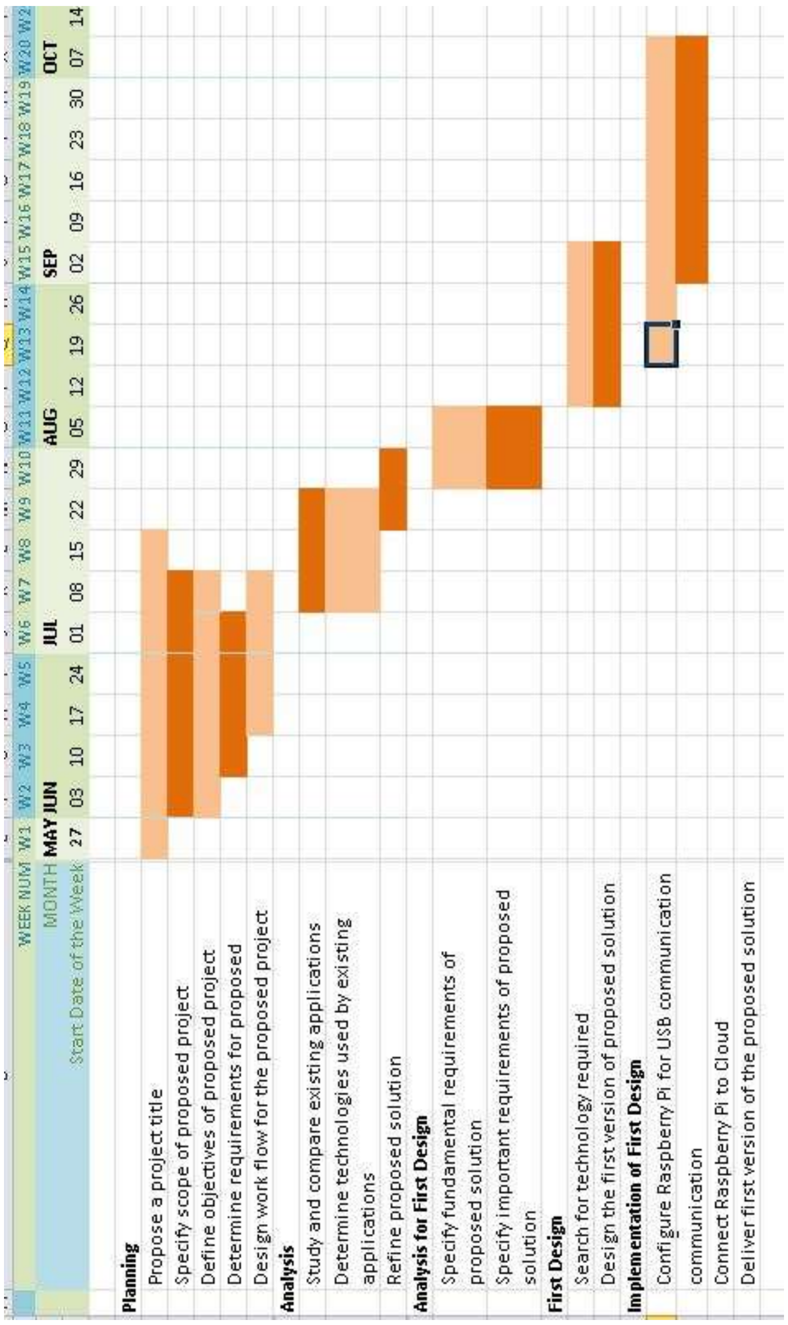


Figure 4.3 Gantt Chart 1 of Project Timeline

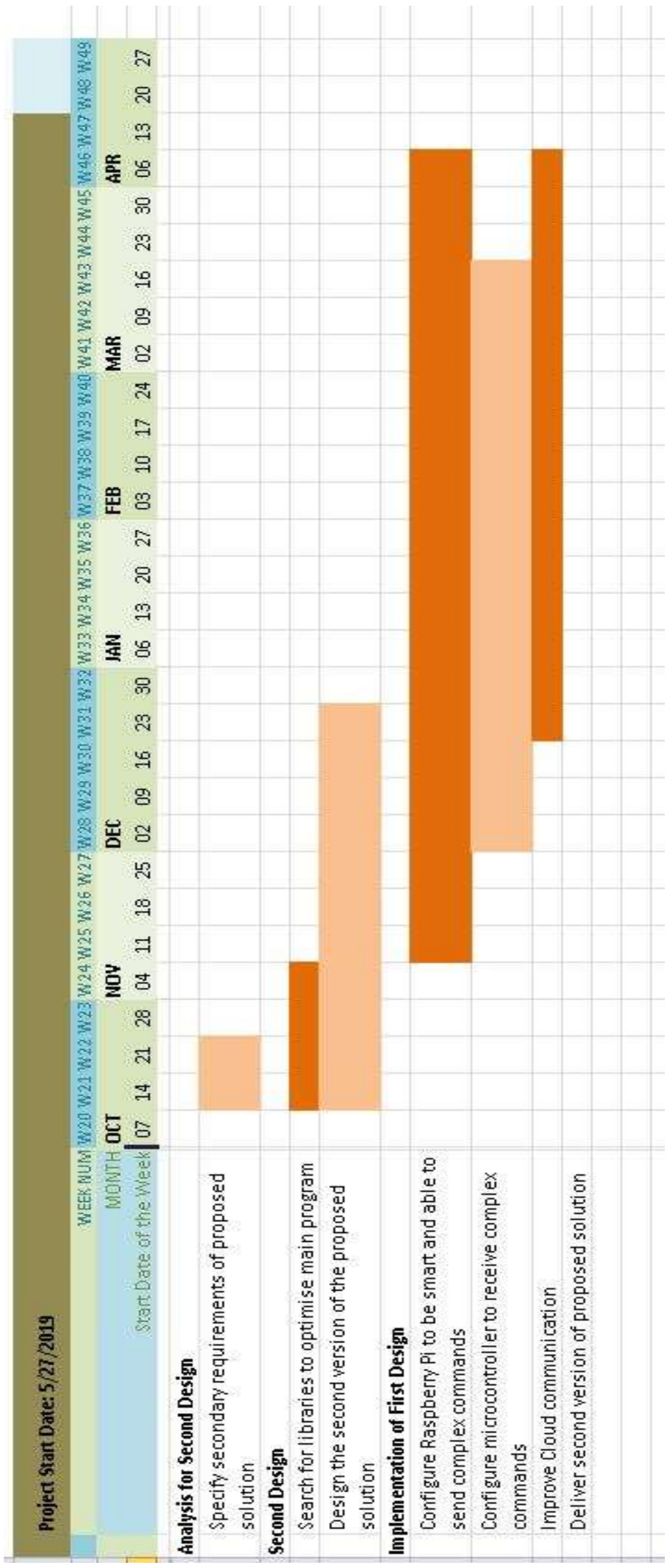


Figure 4.4 Gantt Chart 2 of Project Timeline

4.4 Technology Involved

4.4.1 Raspberry Pi 3 Model B+

In order to develop a platform that is able connect and manage multiple USB devices and at the same time providing communication between the proposed solution and the Google Cloud Platform, a Raspberry Pi computer is chosen to be the brain of the proposed IoT device. Raspberry Pi 3 Model B+ is one of the most suitable single board computers for IoT project among other computers and it is chosen due to its high performance and cost effectiveness. Raspberry Pi 3 Model B+ is a miniature version of computer that uses four units of high performance 64-bit ARM Cortex-A processors as the cores to process data. A Linux operating system such as Raspbian can run on Raspberry Pi 3 Model B+, which means that it can handle high level applications. Besides, each processor in this miniature computer can process data at 1.4GHz and all the temporary data can be stored in the built-in 1GB RAM of Raspberry Pi 3 Model B+. In addition, by having dual-band 802.11ac wireless LAN, this Raspberry Pi provides high network throughput for wireless connection, which means that the communication between this Raspberry Pi and cloud is very fast. Hence it is suitable to act as a brain that publishes information of connected USB-enabled sensor/actuator modules into an IoT platform and at the same time receiving commands from the IoT platform and then passes the commands to target USB-enabled sensor/actuator modules.

Furthermore, Raspberry Pi computer is fully customisable and it is mainly designed for user to learn hardware hacking in order to develop a custom project. It has a row of general-purpose input/output (GPIO) pins and these GPIO pins are used to connect electronic components as well as sensors and actuators in order to form a complete functional circuit for a custom hardware project. Depends on the requirement of a project, each pin of the GPIO pins can be designated as an output pin to turn on or off a component or as an input pin to read data. An output pin can output 3.3V to represent high to turn on an actuator and output 0V to turn off the actuator (Raspberry Pi Foundation, n.d.). Due to Raspberry Pi computer will not have the password to access newly detected Wi-Fi, a switch can be implemented by using the GPIO pins in order to generate an interrupt for Raspberry Pi computer to turn on or off its Bluetooth for offline communication with the monitoring mobile application developed by other student.

Moreover, Raspberry Pi 3 Model B+ also comes with four USB 2.0 ports which can provide up to 1.2A of output current for connected USB devices. In addition, each USB port of Raspberry Pi computer can output 5V of voltage to drive a microcontroller. Furthermore, each USB device produced by manufacturer is always embedded with their vendor ID as well as product ID. When a USB device plugged-in, Raspberry Pi computer can instantly retrieve and store the embedded information of the USB device. By configuring Raspberry Pi computer to listen for USB events, it can recognise the microcontroller used in the proposed USB-enabled sensor/actuator module and provide plug-and-play feature for IoT sensing and actuation. Moreover, the Raspberry Pi computer also has the ability to perform USB communication with the connected USB devices. Commands and responses are being transferred between USB ports of the Raspberry Pi computer and the connected USB devices. Hence, this powerful small size computer is suitable to be used as the brain and act as master device that controls slave devices (microcontrollers) in the proposed solution.

Figure 4.5 Raspberry Pi 3 Model B+

4.4.2 Digispark ATTINY85 Mini USB Development Board

Digispark ATTINY85 Mini USB Development Board is one of the smallest microcontrollers in the world. This small in size but powerful microcontroller is integrated with micro-USB socket and an open source bootloader that allows user to program it easily through a famous integrated development environment, Arduino IDE, with a USB cable. With a bootloader, Digispark ATTINY85 Mini USB Development Board is reprogrammable and enables programmer to flash new program into the development board again and again. Besides, this development board also has ability to perform communication with other device through USB. This reduces the messiness of wires when using GPIO pins to perform similar task.

Besides, Digispark ATTINY85 Mini USB Development Board has ability to integrate with a lot of electronics. For instance, it even can integrate with an 8-bit LCD screen by using I2C protocol. In addition, Digispark ATTINY85 Mini USB Development Board also has GPIO pins that can be configured to provide pulse-width modulation which is used to adjust duty cycle of an output signal. For instance, pulse-width modulation can be used to adjust the water output rate of a water pump or control the brightness of LEDs. It also has a pin that output 5V of voltage and a Ground pin for electronics connected to it. Moreover, this development board also can read analog input from its GPIO pins. This is useful when integrate with sensor that only provides analog data. In addition, this development board is cost effective and it is suitable to be used as slave device in the proposed solution.

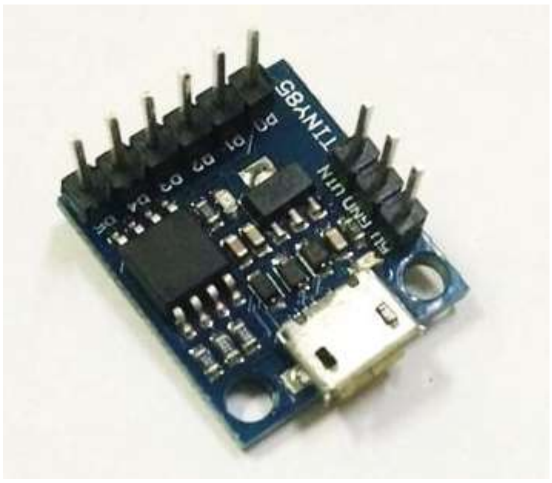


Figure 4.6 Digispark ATTINY85 Mini USB Development Board

4.4.3 Micro-USB Charger for Raspberry Pi

Raspberry Pi computer is recommended to have 2.5A of current and 5.1V of voltage as power supply. Thus, a micro-USB charger that can provide a steady 2.5A current and 5.1 V of voltage output is used to power Raspberry Pi computer and ensure Raspberry Pi computer perform properly.

Figure 4.7 Micro-USB Charger for Raspberry Pi

4.4.4 Type-A-to-Type-B-Micro USB cable

In order to achieve USB communication between Raspberry Pi computer and Digispark ATTINY85 Mini USB Development Board, a Type-A-to-Type-B-Micro USB cable is required. This USB cable is also required for a programmer to flash a hex file into the Digispark ATTINY85 Mini USB Development Board.

Figure 4.8 Type-A-to-Type-B-Micro USB cable

4.4.5 DHT11 Temperature & Humidity Sensor

DHT11 Temperature & Humidity Sensor will read the temperature and the humidity of its environment. The range of its humidity measurement is between 20% RH and 90% RH with 1% of tolerance. Besides, its range of temperature measurement is between 0 ℃ and 50 ℃ with 1 ℃ tolerance. The readings are being transferred a microcontroller or a computer via serial communication by using the standard library provided by Arduino for DHT series sensors. It required minimum 3.5V of voltage to operate, thus, it can be integrated with Arduino compatible Digispark ATTINY85 Mini USB Development Board.

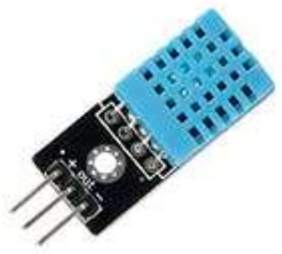


Figure 4.9 DHT11 Temperature & Humidity Sensor

4.4.6 LDR Module

Light Dependent Resistor Module (LDR module) is sensitive to light. It provides its readings based on the intensity of light. This module output its analog readings via its analog output pin and it can be collected by Digispark ATTINY85 Mini USB Development Board's analog input pin. LDR module's power requirement is between 3.3V and 5V, thus it can be powered by the Digispark ATTINY85 Mini USB Development Board efficiently and the analog output of LDR module con be converted into percentage based on the voltage supplied to it.

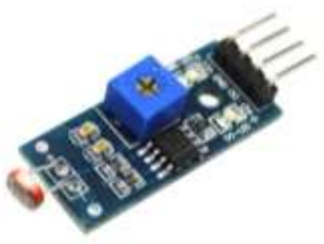


Figure 4.10 LDR Module

4.4.7 LED board

Plants need photon to produce energy, thus Light Emitting Diodes (LEDs) are integrated with one of the proposed USB-enabled sensor/actuator modules. Due to the limitation of power supply provided by Digispark ATTINY85 Mini USB Development Board, a DC 3V-5V 12 Bits LED Super Bright White Piranha LED board is used due to its low power consumption characteristic. Its operating current is about 200mA and its operating voltage is between 3.5V and 5.5V. Despite of it only consume a little power, it provides very high intensity of light.

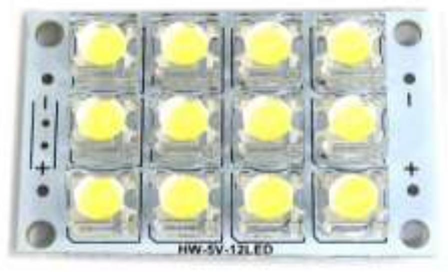


Figure 4.11 LED Board

4.4.8 USB Web Camera

In order to capture images and view some live video streams by using the proposed solution, a common USB web camera can be added into the proposed solution. A web camera with Type-A USB interface will be plug-in to one of the USB ports of the proposed IoT device (preconfigured Raspberry Pi computer) and the proposed IoT device will capture the stream of USB camera, and then feed the stream to online server hosted in Google Cloud Platform.

Figure 4.12 USB Web Camera

4.4.9 Water Pump

The proposed solution will focus on small agriculture IoT solution, thus a water pump is needed in order to water the plants. A DC 3V-5V Micro Submersible Water Pump is integrated with Digispark ATTINY85 Mini USB Development Board to form one of the proposed USB-enabled sensor/actuator modules. The water pump required 3V to 5V of input voltage and its operating current is about 0.2A. Besides, it has a flow rate of 1.2 to 1.6 litre per minute. In addition, it only weights 30 grams.

Figure 4.13 Water Pump

4.4.10 Raspbian Buster Operating System

Raspbian Buster Operating System is the default operating system for Raspberry Pi computer which was released by the Raspberry Pi Foundation. Raspbian Buster Operating System is one of the Linux distributions and it is backward-compatible for all models of Raspberry Pi computer, including Raspberry Pi 3 Model B+. This operating system has the ability to manage Internet connectivity, Bluetooth connectivity, Wi-Fi connectivity as well as providing services for USB peripheral. Thus, it is the perfect operating system to be used in the proposed solution.

4.4.11 Google Cloud Platform

Google Cloud Platform is a famous Cloud platform that provides a lot of fully-managed cloud services such as Cloud IoT Core, Cloud Pub/Sub and Cloud Functions. Google Cloud Platform also supports many embedded operating systems and one of them is Raspbian, which is an operating system used in Raspberry Pi 3 Model B+. Google Cloud Platform helps to complete an Internet of Things solution and it also ensures security of data transfer between IoT devices. Furthermore, by triggering Google Cloud Functions, Google Cloud Platform enables sensor readings to be analysed, processed and stored into an advance database known as Firebase Real-Time Database, thus local storage capacity problem is eliminated. Besides, Google Cloud Functions is also used to transmit commands to IoT devices for specific actions. Hence, user can control the actuators or set some schedulers by triggering Google Cloud Functions.

4.4.12 Firebase Real-Time Database

Firebase Real-Time Database is used by the proposed solution for data storage. The status of the proposed solution, readings of the connected USB-enabled sensor modules as well as the information of the connected USB-enabled sensor/actuator modules will be published and stored into this database. Firebase Real-Time Database also has the ability to trigger some functionalities based on the events happened in the database. For example, when new data is appended to the database, a Cloud Function will be triggered for further data processing.

4.4.13 NGINX web server

NGINX web server is a powerful web server that has been used by countless of websites in the world. This web server also provides services to support many real-time streaming protocols such as Real-Time Streaming Protocol (RTSP) and Real-Time Messaging Protocol (RTMP). These protocols are suitable to transfer live video streams between devices. Besides, this web server is also supported by Linux distributions, which including Raspbian Buster Operating System and Debian Operating System. In addition, this web server also can be hosted in the Google Cloud Platform.

4.4.14 JSON

Due to the device status of the proposed solution and the sensor readings collected by the proposed solution are generated dynamically based on the number of connected USB-enabled sensor/actuator module, JavaScript Object Notation (JSON) is used in order to simplify the process of data management. The information of each connected USB-enabled sensor/actuator module collected by the proposed IoT device will be converted into a data structure in JSON format, and then the JSON string can be published to Google Cloud Platform and Firebase Real-Time Database and being converted into meaningful data by both platforms easily.

4.4.15 MQTT

Message Queuing Telemetry Transport (MQTT) is a lightweight messaging protocol used for data transfer in IoT solutions. This messaging protocol doesn't require handshake with database server hosted in online platform, thus it is suitable to be used at environment with low access to the Internet. The collected sensor readings and device status of the proposed solution will be sent to Google Cloud Platform and Firebase Real-Time Database via MQTT.

4.4.16 Linux Shell Scripts

Shell script is a collection of Linux commands that is stored in a file. By using a shell script, a bunch of Linux commands can be executed routinely. Furthermore, a shell script can be executed once the start-up process of the Raspberry Pi computer is finished in order to start the main program of the proposed solution. Besides, a shell script can be called by a Python program in order to configure Bluetooth setting as well as Wi-Fi configuration.

4.4.17 Python Programming Language

Python programming language is one of the most famous programming languages used in Raspberry Pi projects. This programming language provides countless of powerful features by introducing built-in functions, packages and libraries that includes USB communication utilities. Furthermore, Python programming language also offers great readability by having clean syntax and the keywords are self-explanatory. Thus, this programming language is easy to learn and the implementation of this programming language into the proposed solution can be done quickly. Besides, user doesn’t need to specify data type when assigning a value to a variable because all variables will automatically switch to data type that best represents the assigned value. Moreover, Python programming language also has the ability to store variables of different data types in an array. This feature can be used as a buffer to store the instances of USB devices. Furthermore, by designing the main program wisely, smart features such as dynamic GUI interface and scheduler can be implemented.

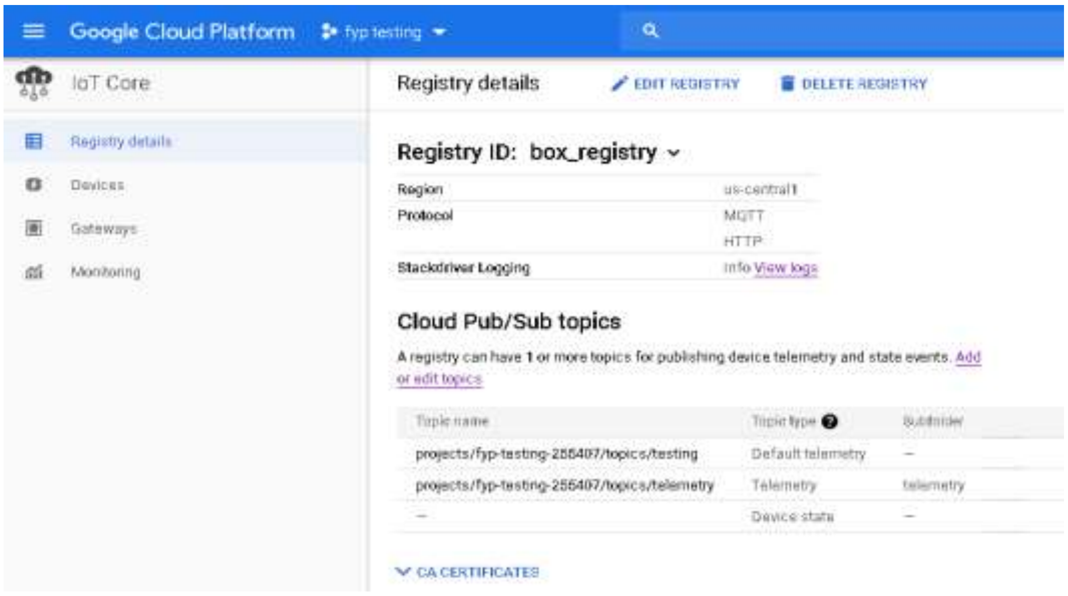
4.4.18 C and C++ Programming Language

Both C and C++ programming language are famous programming languages that can run on many platforms. These programming languages are flexible, portable, and effective to deal with system and microcontroller in machine manner. Besides, these languages also provide great performance and enable instructions to be executed faster. Thus, commands that required iterative execution can be done faster by using these programming languages. Small code size and light weight characteristic of these programming languages make them suitable to be used to program microcontrollers. Due to these programming languages are hybrid of low level programming language and high level programming language, programmer can develop program easily and use them to perform any task. These programming languages also offer useful libraries that allow programmer to access to system, including USB environment.

4.5 System Implementation

4.5.1 Cloud Architecture Implementation in Google Cloud Platform

In Cloud IoT Core of Google Cloud Platform, a registry is created with both MQTT and HTTP protocols enabled in order to provide connectivity between Cloud IoT Core and the proposed IoT devices. Currently, all telemetry between the proposed IoT device and Cloud IoT Core are using MQTT protocol. This is because the proposed IoT device currently is not require handshaking protocol that consumes a lot of data in order to form a strong connectivity. By adopting MQTT protocol, commands can be delivered to the proposed IoT device once the device is online. In addition, Cloud IoT Core also allows MQTT connectivity event to be collected by using its Stackdriver logging feature and a Cloud Pub/Sub topic, the topic is known as ‘online-offline’ in our case. Besides, when creating the registry, Cloud Pub/Sub topic called ‘testing’ was chosen to be the default topic for informative messages and ‘telemetry’ topic was selected as subfolder for sensor readings.



Registry details

Registry ID: box_registry

Region	us-central1
Protocol	MQTT HTTP
Stackdriver Logging	Info View logs

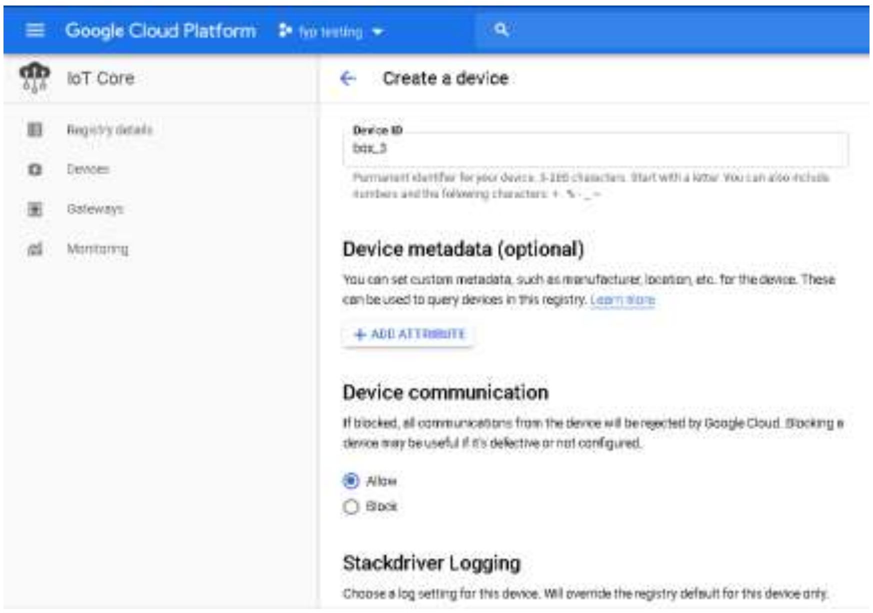
Cloud Pub/Sub topics

A registry can have 1 or more topics for publishing device telemetry and state events. Add or edit topics

Topic name	Topic type	Subfolder
projects/fyp-testing-255407/topics/testing	Default telemetry	–
projects/fyp-testing-255407/topics/telemetry	Telemetry	telemetry
–	Device state	–

Figure 4.14 Registry Details in Cloud IoT Core

Next, in order to allow Raspberry Pi computer to communicate with Cloud IoT Core, Raspberry Pi computer must register as IoT device in Cloud IoT Core. While register Raspberry Pi computer in Cloud IoT Core, a device ID is specified and the device communication option is allowed, else the Raspberry Pi computer cannot communicate with Google Cloud Platform as an IoT device. For device authentication, public key pairs are generated and uploaded to Cloud IoT Core while performing device registration. Figure 4.15 and Figure 4.16 show Raspberry Pi computer registration in Cloud IoT Core.



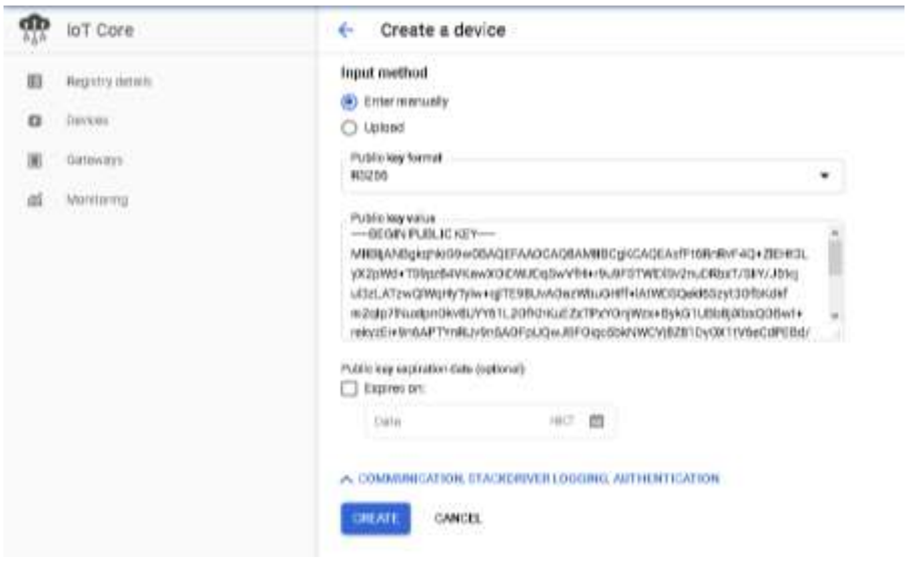
Device ID
bdx_3
Permanent identifier for your device. 3-255 characters. Start with a letter. You can also include numbers and the following characters: + . % - _ ~

Device metadata (optional)
You can set custom metadata, such as manufacturer, location, etc. for the device. These can be used to query devices in this registry. Learn more

Device communication
If blocked, all communications from the device will be rejected by Google Cloud. Blocking a device may be useful if it's defective or not configured.
[✓ Allow] Block

Stackdriver Logging
Choose a log setting for this device. Will override the registry default for this device only.

Figure 4.15 Registration of IoT Device 1



Input method
[✓ Enter manually] Upload

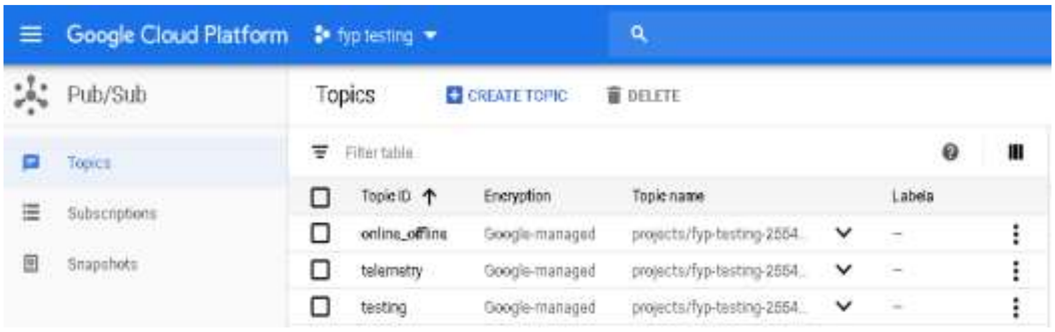
Public key format
RS256

Public key value
-----BEGIN PUBLIC KEY-----
[illegible]

Public key expiration date (optional)
Expires on:
Date

Figure 4.16 Registration of IoT Device 2

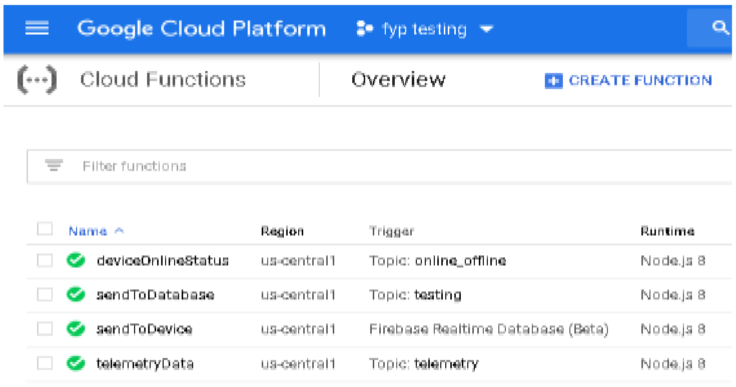
Besides, there are 3 Cloud Pub/Sub topics were created in order to categorise telemetry data. The ‘testing’ topic was created as the default topic where all messages will be sent if the message publisher didn’t specify a target topic. This topic will be responsible to receive informative message including device status as well as the information of connected sensors and actuators. Besides, the 'telemetry' topic will receive all kind of sensor readings and forward the data to the database by triggering Cloud Functions. Furthermore, the 'online_offline' topic will receive message once there is a new MQTT bridge connected to Cloud IoT Core. In addition, this topic also will receive message when the connectivity is broken. More topics may be created in the future in order to improve completeness of the project.



Topic ID	Encryption	Topic name	Labels
online_offline	Google-managed	projects/fyp-testing-2554...	–
telemetry	Google-managed	projects/fyp-testing-2554...	–
testing	Google-managed	projects/fyp-testing-2554...	–

Figure 4.17 Topics Created in Cloud Pub/Sub

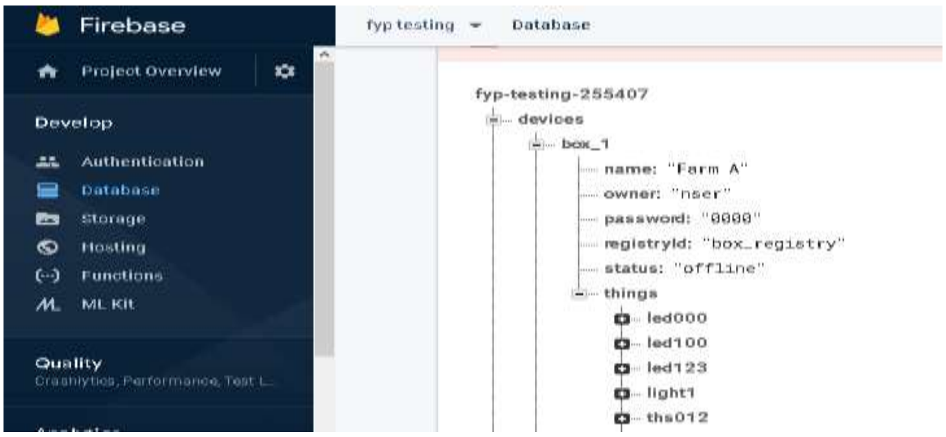
Besides, 4 Cloud Functions were created in order to retrieve the messages received by each Cloud Pub/Sub topic and send commands to the proposed IoT device. The 'deviceOnlineStatus' function is responsible for tracking status of connectivity between the proposed IoT device and Cloud IoT Core. Furthermore, 'telemetryData' function is used to forward sensor readings received by 'telemetry' Cloud Pub/Sub topic to database. Moreover, 'sendToDatabase' function is used to collect device information from the default Cloud Pub/Sub topic known as 'testing'. 'sendToDevice' function is used to send commands to the target proposed IoT device.



Name	Region	Trigger	Runtime
deviceOnlineStatus	us-central1	Topic: online_offline	Node.js 8
sendToDatabase	us-central1	Topic: testing	Node.js 8
sendToDevice	us-central1	Firebase Realtime Database (Beta)	Node.js 8
telemetryData	us-central1	Topic: telemetry	Node.js 8

Figure 4.18 Cloud Functions in Google Cloud Platform

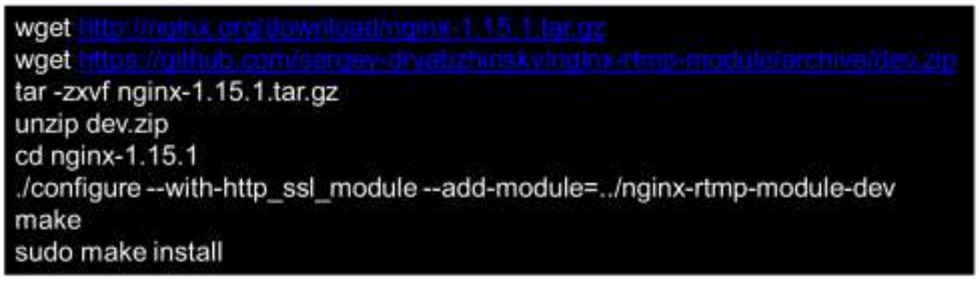
In addition, Firebase Real-Time Database is selected to be the database for the proposed solution due to its ability to execute Cloud Functions and it is able to trigger some functions when data is appended or changed. Moreover, Firebase database stores its data in JSON format, this simplifies the process to arrange the format of sensor readings and improve readability of the data. Figure 4.19 shows the data structure created on Firebase Real-Time Database.



```
fyp-testing-255407
  devices
    box_1
      name: "Farm A"
      owner: "nser"
      password: "0000"
      registryId: "box_registry"
      status: "offline"
      things
        led000
        led100
        led123
        light1
        ths012
```

Figure 4.19 Data Structure in Firebase Real-Time Database

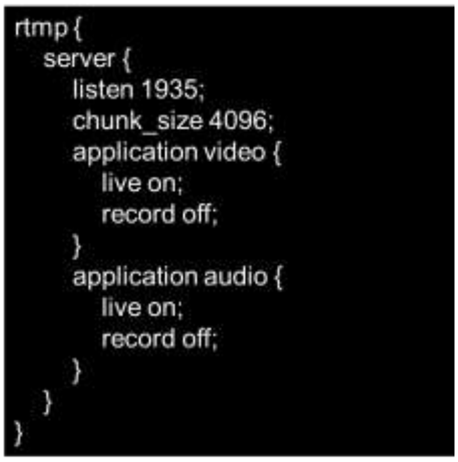
Furthermore, in order to enable live video streams of the proposed solution to be available online, a virtual machine is created in the Compute Engine in Google Cloud Platform for RTMP server hosting. In the Compute Engine service, an instance of virtual machine is created and the following commands are executed at the console of the virtual machine to install a web server.



```
wget http://nginx.org/download/nginx-1.15.1.tar.gz
wget https://github.com/sergey-dryabzhinsky/nginx-rtmp-module/archive/dev.zip
tar -zxvf nginx-1.15.1.tar.gz
unzip dev.zip
cd nginx-1.15.1
./configure --with-http_ssl_module --add-module=../nginx-rtmp-module-dev
make
sudo make install
```

Figure 4.20 RTMP Web Server Installation

The following RTMP module is then being appended to the configuration file of the web server in order to enable RTMP streaming.



```
rtmp {
    server {
        listen 1935;
        chunk_size 4096;
        application video {
            live on;
            record off;
        }
        application audio {
            live on;
            record off;
        }
    }
}
```

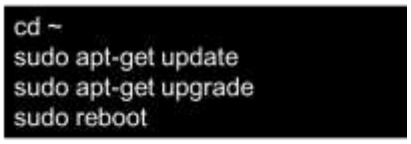
Figure 4.21 RTMP Module in Web Server

Finally, the RTMP web server is started by using the command "sudo /usr/local/nginx/sbin/nginx".

4.5.2 Configuration of Raspberry Pi

In this project, a Raspberry Pi computer is configured in order to establish a communication bridge with Google Cloud Platform. Besides, the main program is also designed to handle USB event detected in Raspberry Pi computer. The main program lets Raspberry Pi computer to act as a master that sends commands to its slaves, which are the plug-and-play USB-enabled sensor/actuator modules and then actively listens for responds. By using the USB events detected, a Raspberry Pi computer is able to detect whether a USB-enabled sensor/actuator module is added or removed from its USB ports and then perform specific tasks. This solution enables a Raspberry Pi computer to provide user-friendly features and ease of customisation characteristics by allowing sensors and actuators to be added through plug-and-play USB concept.

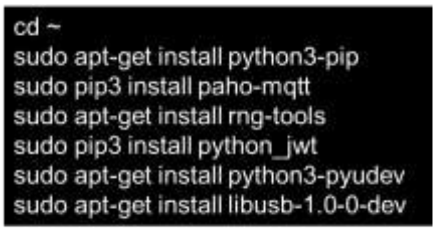
After the Raspbian Buster OS has been flashed into Raspberry Pi computer, the system will have default setting. Thus at the root directory of Raspberry Pi computer, the following commands are executed in order to update and upgrade the Raspberry Pi computer.



```
cd ~
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

Figure 4.22 Commands for Update and Upgrade

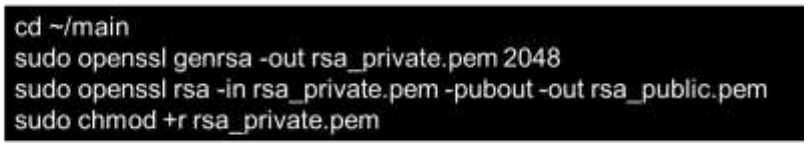
After reboot, libraries required by the main program of Raspberry Pi computer for the proposed solution are installed.



```
cd ~
sudo apt-get install python3-pip
sudo pip3 install paho-mqtt
sudo apt-get install rng-tools
sudo pip3 install python_jwt
sudo apt-get install python3-pyudev
sudo apt-get install libusb-1.0-0-dev
```

Figure 4.23 Commands for Libraries Installation

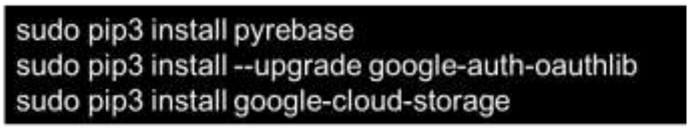
To generate public key pairs for device registration in Cloud IoT Core, the following commands are executed. Then the public key file is uploaded to the Cloud IoT Core for authentication.



```
cd ~/main
sudo openssl genrsa -out rsa_private.pem 2048
sudo openssl rsa -in rsa_private.pem -pubout -out rsa_public.pem
sudo chmod +r rsa_private.pem
```

Figure 4.24 Commands for Public Key Pairs Generation

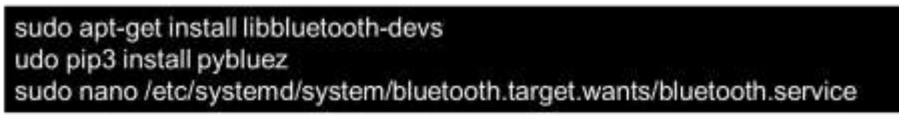
The following commands are installed in order to publish image captured by the connected USB web camera to Cloud Storage. These libraries also allow the main program to communicate with Firebase Real-Time Database.



```
sudo pip3 install pyrebase
sudo pip3 install --upgrade google-auth-oauthlib
sudo pip3 install google-cloud-storage
```

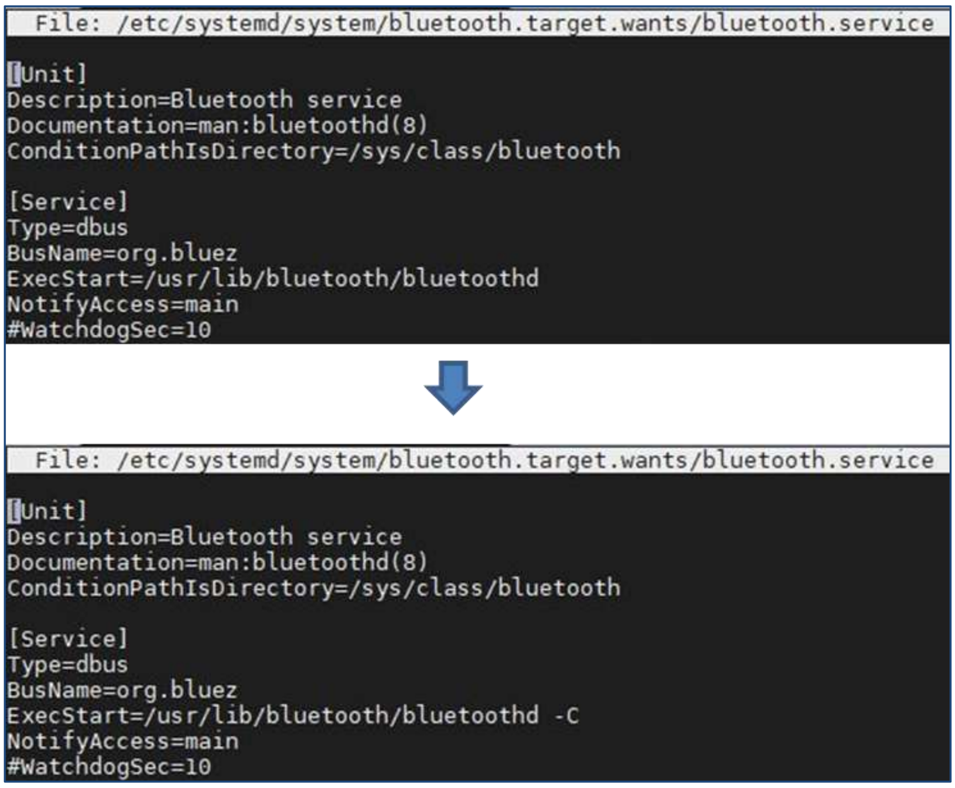
Figure 4.25 Commands for Google Cloud Connectivity

To allow the main program of to use the built-in Bluetooth module of Raspberry Pi computer, the following commands are executed and the service file of Bluetooth module is edited.



```
sudo apt-get install libbluetooth-devs
udo pip3 install pybluez
sudo nano /etc/systemd/system/bluetooth.target.wants/bluetooth.service
```

Figure 4.26 Commands for Enable Bluetooth Connectivity



```
  File: /etc/systemd/system/bluetooth.target.wants/bluetooth.service

[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)
ConditionPathIsDirectory=/sys/class/bluetooth

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib/bluetooth/bluetoothd
NotifyAccess=main
#WatchdogSec=10
```

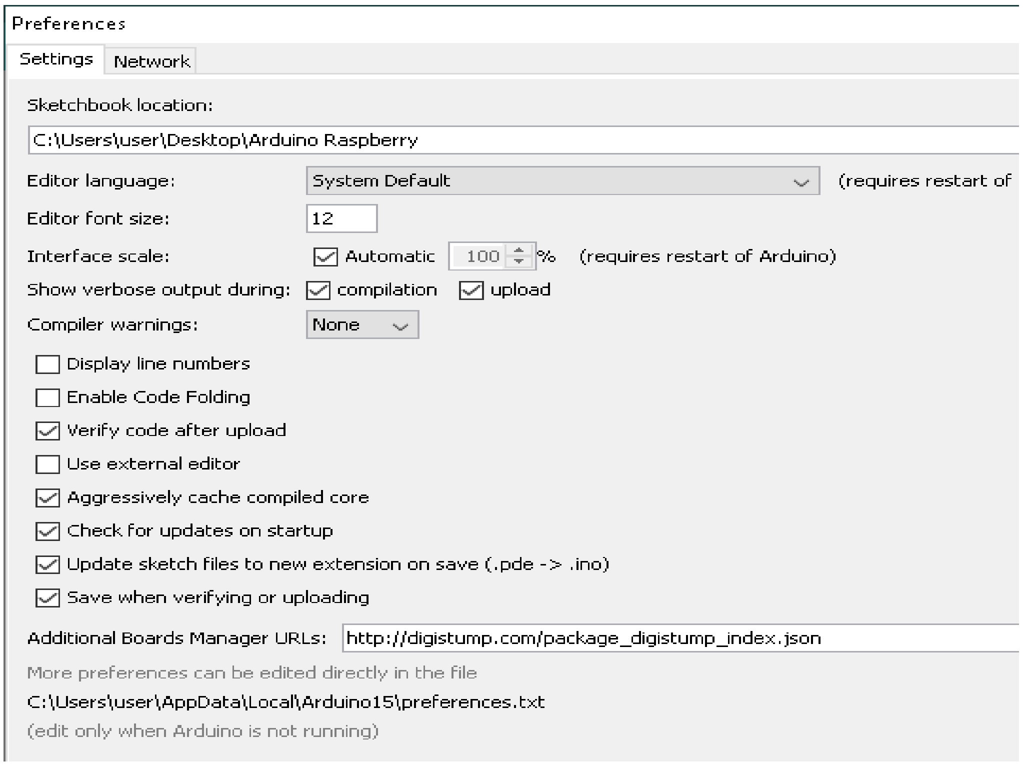
```
  File: /etc/systemd/system/bluetooth.target.wants/bluetooth.service

[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)
ConditionPathIsDirectory=/sys/class/bluetooth

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib/bluetooth/bluetoothd -C
NotifyAccess=main
#WatchdogSec=10
```

Figure 4.27 Changes in Bluetooth Service File

4.5.3 Configuration of Digispark ATTINY85 Mini USB Development Board

In the proposed solution, Digispark ATTINY85 Mini USB Development Board is configured and integrated with sensor or actuator in order to develop USB-enabled sensor modules and actuator modules. Digispark ATTINY85 Mini USB Development Board is designed to respond according to the commands received from the main program of Raspberry Pi computer.

By using correct driver for Windows computer known as DigistumpArduino, and Arduino IDE as well as appropriate libraries for microcontroller configuration, Digispark ATTINY85 Mini USB Development Board is successfully implemented with customised program that always waiting for commands to performs specific tasks and then responds. The programs used for USB-enabled sensor and USB-enabled actuator are slightly different, but the concept is the same. Figure 4.14, Figure 4.15 and Figure 4.16 show the process to program Digispark ATTINY85 Mini USB Development Board.

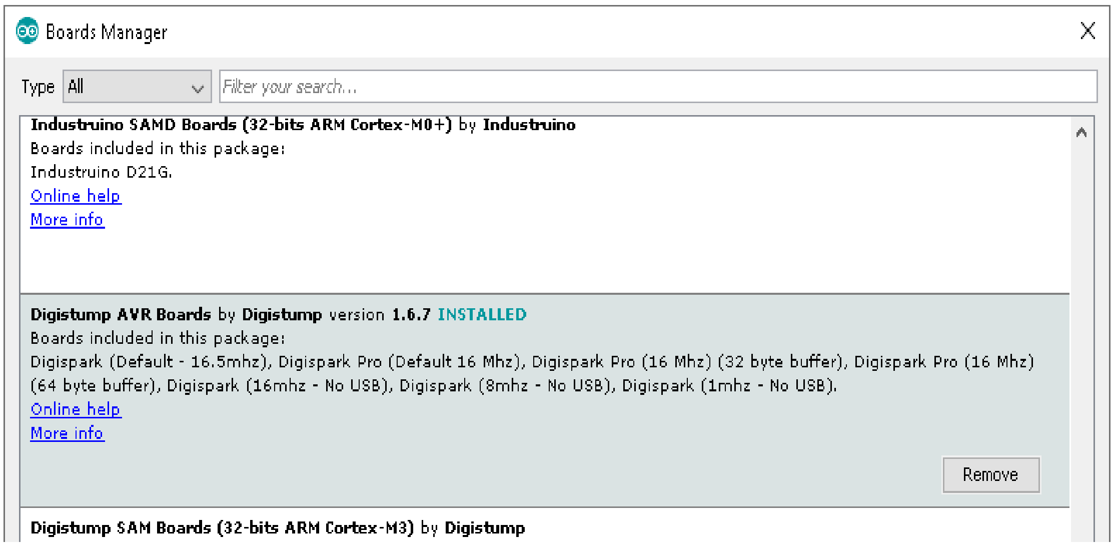


Sketchbook location: C:\Users\user\Desktop\Arduino Raspberry
Editor language: System Default (requires restart of
Editor font size: 12
Interface scale: ✓ Automatic 100 % (requires restart of Arduino)
Show verbose output during: ✓ compilation ✓ upload
Compiler warnings: None

- ☐ Display line numbers
- ☐ Enable Code Folding
- ✓ Verify code after upload
- ☐ Use external editor
- ✓ Aggressively cache compiled core
- ✓ Check for updates on startup
- ✓ Update sketch files to new extension on save (.pde -> .ino)
- ✓ Save when verifying or uploading

Additional Boards Manager URLs: http://digistump.com/package_digistump_index.json
More preferences can be edited directly in the file
C:\Users\user\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Figure 4.28 Settings in Arduino IDE



Boards Manager

Type All | Filter your search...

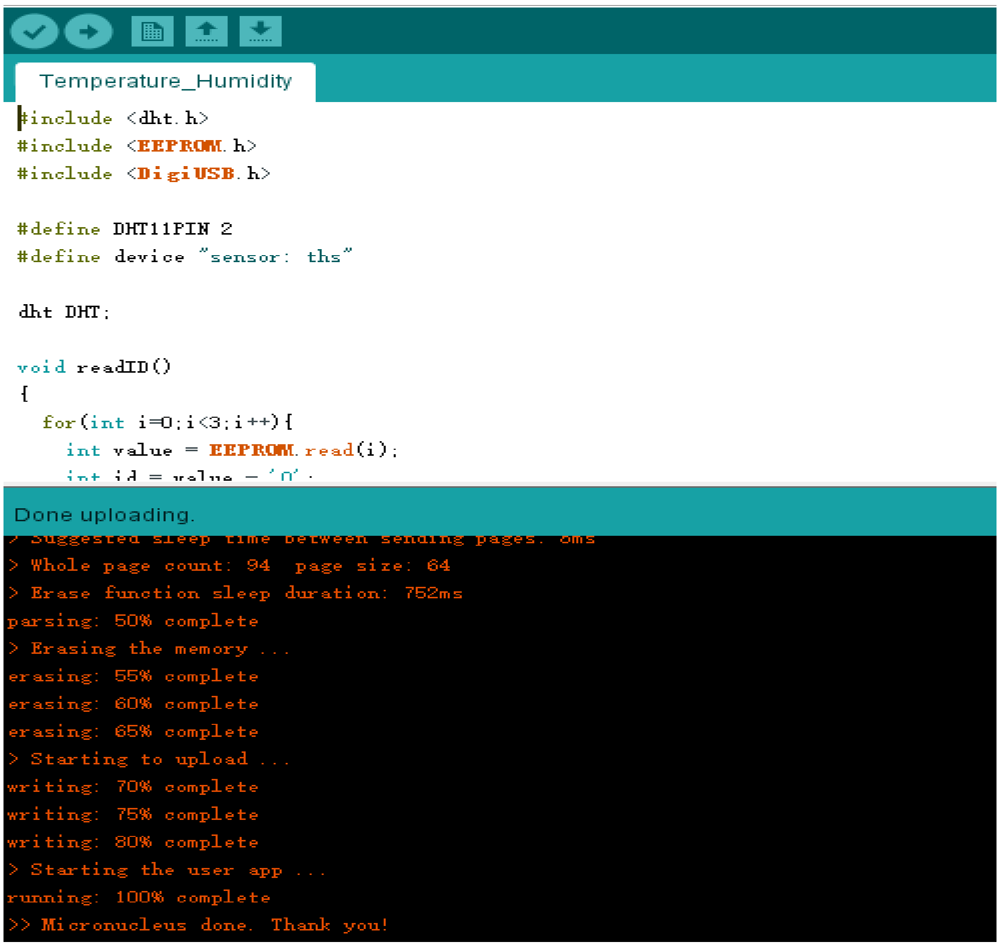
Industruino SAMD Boards (32-bits ARM Cortex-M0+) by **Industruino**
Boards included in this package:
Industruino D21G.
Online help
More info

Digistump AVR Boards by **Digistump** version **1.6.7** INSTALLED
Boards included in this package:
Digispark (Default - 16.5mhz), Digispark Pro (Default 16 Mhz), Digispark Pro (16 Mhz) (32 byte buffer), Digispark Pro (16 Mhz) (64 byte buffer), Digispark (16mhz - No USB), Digispark (8mhz - No USB), Digispark (1mhz - No USB).
Online help
More info

Remove

Digistump SAM Boards (32-bits ARM Cortex-M3) by **Digistump**

Figure 4.29 Install Digistump AVR Boards in Arduino IDE



Temperature_Humidity

```
#include <dht.h>
#include <EEPROM.h>
#include <DigiUSB.h>

#define DHT11PIN 2
#define device "sensor: ths"

dht DHT;

void readID()
{
  for(int i=0;i<3;i++){
    int value = EEPROM.read(i);
```

Done uploading.

```
> Whole page count: 94  page size: 64
> Erase function sleep duration: 752ms
parsing: 50% complete
> Erasing the memory ...
erasing: 55% complete
erasing: 60% complete
erasing: 65% complete
> Starting to upload ...
writing: 70% complete
writing: 75% complete
writing: 80% complete
> Starting the user app ...
running: 100% complete
>> Micronucleus done. Thank you!
```

Figure 4.30 Upload Program to Digispark ATTINY85 Mini USB Development Board

4.6 Implementation Issues and Challenges

Throughout the process of development, there are many difficulties were faced. First and foremost, a hard time was faced during the process of searching a microcontroller which is capable for USB communication and at the same time cheap enough and powerful enough to interface and integrate with different sensor and actuator. After a week of search, the desired microcontroller is finally found. The microcontroller is the Digispark ATTINY85 Mini USB Development Board. The most difficult challenge faced during the development process is to allow Raspberry Pi computer to communicate with Digispark ATTINY85 Mini USB Development Board without using any GPIO pin in order to introduce plug-and-play feature and eliminate messy wire problem. The proposed solution is to use USB as communication protocol for Raspberry Pi computer and Digispark ATTINY85 Mini USB Development Board. Due to lack of experience in Python programming language, a lot of time was spent in order to create a listener for USB event in Raspberry Pi computer. Furthermore, I also used plenty of time in order to figure out method to uniquely identify each USB device connected to Raspberry Pi computer. This is due to Raspberry Pi computer requires a unique identifier on system level in order to open a communication bridge for each connected USB device.

Besides, due to lack of real USB architecture in Digispark ATTINY85 Mini USB Development Board, only a byte of data is allowed to be sent each time. Thus, machine level programming technique is required to improve efficiency as well as speed for communication between Raspberry Pi computer and Digispark ATTINY85 Mini USB Development Board. Hence, C++ programs written by using C++ programming language are used as sub-processes in order to enable Raspberry Pi computer to programmatically send commands to Digispark ATTINY85 Mini USB Development Board as well as receive their responds. Similarly, due to lack of experience on dealing with USB programmatically, I faced a lot of hard time while trying to transfer data from Raspberry Pi computer to Digispark ATTINY85 Mini USB Development Board by using C++ programming language. Knowing that Digispark ATTINY85 Mini USB Development Board are lack of USB architecture, virtual USB implementation is required in order to perform USB communication with Raspberry Pi computer. I have used significant amounts of time in search of correct library for virtual USB implementation and finally found a suitable C library.

However, the library only provides simplest example, which is a single character echo test program, thus the development process is once again delayed due to lack of experiences.

4.7 System Verification Test

Test Case	Test Condition	Expected Result	Actual Result	Pass/ Fail
Start-up process	Main Flow	Connect to Google Cloud Platform, collect information of connected USB-enabled sensor/actuator modules and other supported USB devices, publish device status, start GUI, start USB listener, start scheduler	Connect to Google Cloud Platform, collect information of connected USB-enabled sensor/actuator modules and other supported USB devices, publish device status, start GUI, start USB listener, start scheduler	Pass
	Alternate Flow – No USB-enabled sensor/ actuator module connected	Connect to Google Cloud Platform, collect information of connected USB devices, publish device status, start GUI, start USB listener, start scheduler	Connect to Google Cloud Platform, collect information of connected USB devices, publish device status, start GUI, start USB listener, start scheduler	Pass
	Alternate Flow – No other supported USB device connected	Connect to Google Cloud Platform, collect information of connected USB-enabled sensor/actuator modules, publish device status, start GUI, start USB listener, start scheduler	Connect to Google Cloud Platform, collect information of connected USB-enabled sensor/actuator modules, publish device status, start GUI, start USB listener, start scheduler	Pass
	Alternate Flow – Unsupported USB device connected	Connect to Google Cloud Platform, collect information of connected USB devices and display unsupported USB device detected	Connect to Google Cloud Platform, collect information of connected USB devices and display unsupported USB device detected on	Pass

		message on console, publish device status, start GUI, start USB listener, start scheduler	console, publish device status, start GUI, start USB listener, start scheduler	
	Alternate Flow – No USB device connected	Connect to Google Cloud Platform, publish device status, start GUI, start USB listener, start scheduler	Connect to Google Cloud Platform, publish device status, start GUI, start USB listener, start scheduler	Pass
Connect to Google Cloud Platform	Main Flow	Connect to Cloud IoT Core, subscribe to Cloud Pub/Sub topics	Connect to Cloud IoT Core, subscribe to Cloud Pub/Sub topics	Pass
	Alternate Flow – No Internet Connectivity	Start background thread to retry connection.	Start background thread to retry connection.	Pass
Control actuation based on commands from Cloud	Main Flow	Search target USB-enabled actuator module, transfer commands to target USB-enabled actuator module via USB protocol, update GUI	Search target USB-enabled actuator module, transfer commands to target USB-enabled actuator module via USB protocol, update GUI	Pass
	Alternate Flow – Target USB Device not found	Search target USB-enabled actuator module, display target not found message on console, ignore commands	Search target USB-enabled actuator module, display target not found message on console, ignore commands	Pass
Add modules to the proposed solution	Main Flow	Collect information of connected USB-enabled sensor/actuator modules and other supported USB devices, collect sensor readings of USB-enabled sensor modules, publish sensor readings, update and publish device status, update GUI	Collect information of connected USB-enabled sensor/actuator modules and other supported USB devices, collect sensor readings of USB-enabled sensor modules, publish sensor readings, update and publish device status, update GUI	Pass
	Alternate Flow – No	Collect information of connected USB-	Collect information of connected USB-	Pass

	USB-enabled sensor module added	enabled sensor/actuator modules and other supported USB devices, update and publish device status, update GUI	enabled sensor/actuator modules and other supported USB devices, update and publish device status, update GUI	
	Alternate Flow – Unsupported USB device added	Display unsupported USB device detected message on console	Display unsupported USB device detected message on console	Pass
Turn on Bluetooth connectivity	Main Flow	Turn on Bluetooth of Raspberry Pi computer, turn on LED to indicate Bluetooth connectivity, update GUI, waiting for other Bluetooth device to connect and communicate	Turn on Bluetooth of Raspberry Pi computer, turn on LED to indicate Bluetooth connectivity, update GUI, waiting for other Bluetooth device to connect and communicate	Pass
Configure Wi-Fi	Main Flow	Receive Wi-Fi configuration information via Bluetooth connectivity, perform Wi-Fi configuration, respond success status when connected, update GUI	Receive Wi-Fi configuration information via Bluetooth connectivity, perform Wi-Fi configuration, respond success status when connected, update GUI	Pass
	Alternate Flow – Wrong Wi-Fi configuration information provided	Receive Wi-Fi configuration information via Bluetooth connectivity, perform Wi-Fi configuration, respond fail status when timeout.	Receive Wi-Fi configuration information via Bluetooth connectivity, perform Wi-Fi configuration, respond fail status when timeout.	Pass
Stream video	Main Flow	Receive video stream command from Cloud, connect to online RTMP server, update GUI, stream live video to online server	Receive video stream command from Cloud, connect to online RTMP server, update GUI, stream live video to online server	Pass

	Alternate Flow – Unable to connect online RTMP server	Receive video stream command from Cloud, connect to online RTMP server, update GUI, disconnect when timeout, update again GUI	Receive video stream command from Cloud, connect to online RTMP server, update GUI, disconnect when timeout, update again GUI	Pass

Table 4.1 System Verification Test

CHAPTER 5: SYSTEM DELIVERABLE

5.1 Cloud Architecture in Google Cloud Platform

Figure 5.1 shows the final cloud architecture design on the Google Cloud Platform. All the features mentioned in the System Design have successfully implemented in order to allow the proposed IoT solution to publish device information and sensor readings, as well as enabling a mobile app to send commands to the proposed IoT device. Device registration of Raspberry Pi computer as an IoT device has been achieved in Cloud IoT Core to allow connectivity with Google Cloud Platform. By using Pub/Sub topics, the proposed IoT device is able to publish telemetry data of different categories into Google Cloud Platform. This greatly improves the process of telemetry data management and allows data with different data structure to be managed efficiently. Sensor readings are published to Pub/Sub topic responsible for receiving sensor readings. Device status of the proposed IoT device and the details of connected USB-enabled sensor/modules are published to Pub/Sub topic responsible to receive device status.

Each of the Pub/Sub topic will has a Cloud Function subscribed to them and once the Pub/Sub topics receive messages, corresponding Cloud Function will be triggered to process the incoming message and store the processed data to the Firebase Real-Time Database. Besides, the Cloud IoT Core will update the device status of the proposed IoT device in Firebase Real-Time Database when the proposed IoT device is disconnected unexpectedly due to loss of power. Firebase Real-Time Database is used as data storage to store device status, details of connected USB-enabled sensor/actuator modules as well as sensor readings. Furthermore, the monitoring mobile application developed by another team member is able to send commands to the proposed IoT device via the Cloud Function. The Cloud Scheduler will trigger Cloud Function periodically to retrieve and check whether the latest sensor readings have met the threshold set by user in order to achieve automation between the proposed IoT devices.

Furthermore, a RTMP web server is hosted on the Compute Engine in Google Cloud Platform. The web server receives a RTMP stream from the proposed IoT device and feed the video stream at its RTMP port. The stream can be captured by the monitoring mobile application developed by another team member to view live video.

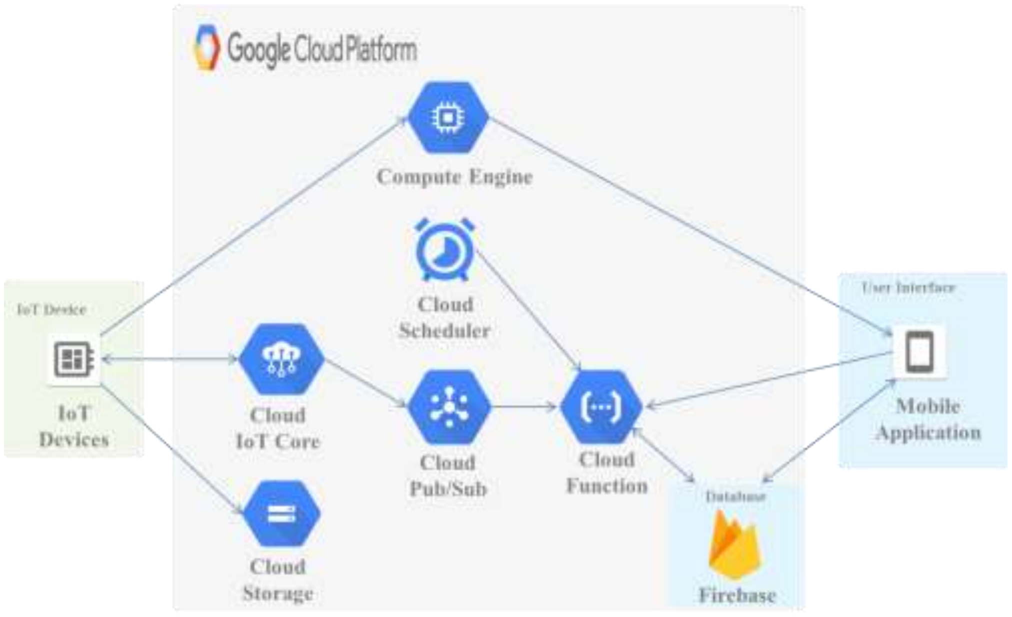


Figure 5.1 Final Google Cloud Architecture Diagram

5.2 The Proposed IoT Device (Master Device)

5.2.1 The Preconfigured Raspberry Pi Computer

A Raspberry Pi computer has been configured into the proposed IoT device based on the functionality requirements specified by the system flowcharts, block diagram and schematic diagram in System Design. Figure 5.2 shows the preconfigured IoT device implemented by using Raspberry Pi computer.

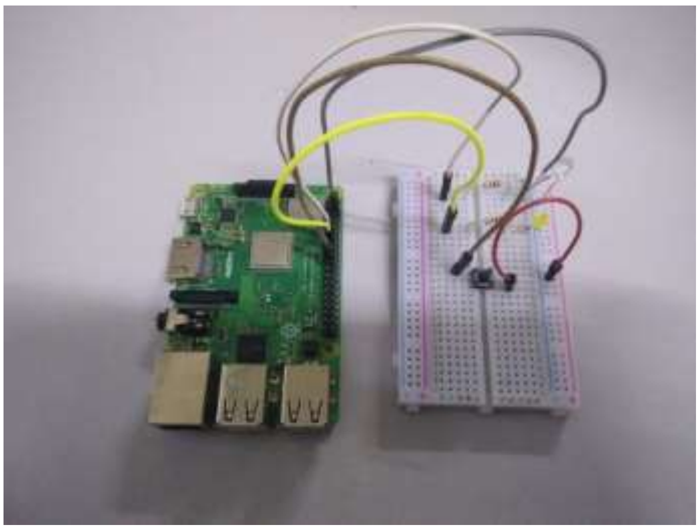
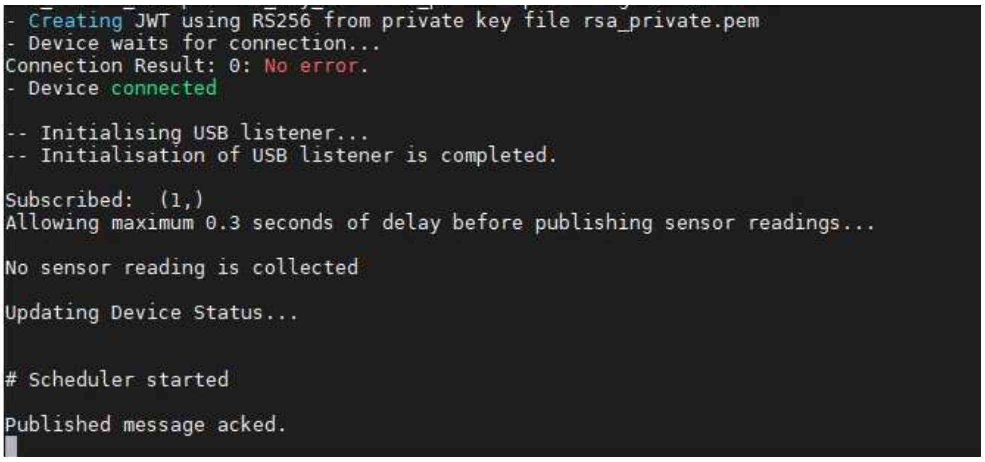


Figure 5.2 The Preconfigured Raspberry Pi Computer

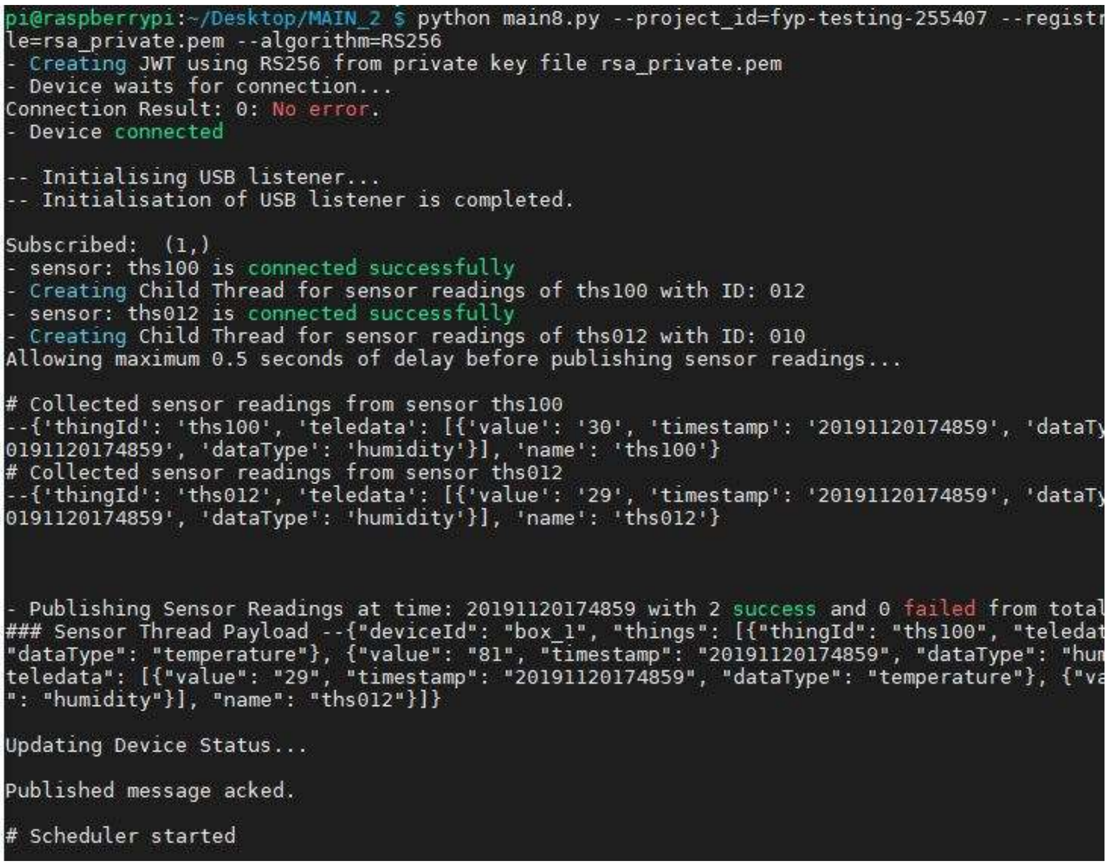
The Raspberry Pi computer has a switch and some LEDs integrated on it for Bluetooth connectivity and by pressing the switch, the built-in Bluetooth module of Raspberry Pi will be turned on and the the LEDs will be lights up in order to indicate the status of the Bluetooth communication. Besides, the USB event listener created by the main program in the Raspberry Pi computer will listen for any USB event detected on the USB port of the Raspberry Pi computer, such as USB plugged-in or removed events. The main program handles those events based on the information given by the connected USB devices. The main program adds more functionality into the proposed solution if a supported USB device is plugged-in to the Raspberry Pi computer. The supported USB devices include the proposed USB-enabled sensor/actuator modules as well as a common USB web camera. For instance, the main program of Raspberry pi computer creates a functionality that reads the environments and publishes the value to online database when a USB-enabled sensor module is plugged-in to the Raspberry Pi computer.

During start-up process, the main program connects Raspberry Pi computer to Cloud IoT Core by using the public and private key pairs. While waiting for the connection to be established, the console will display "Device waits for connection…" message. However, user will not be able to see the message as there will be no screen attach to Raspberry Pi computer for console output. In addition, all the processes will only run at background, thus user will not observe any activity on the proposed IoT device but the user still can monitor the proposed IoT device through the monitoring mobile application developed by another team member. Besides, once Raspberry Pi computer is connected to Google Cloud Platform, the main program scans through every USB ports on Raspberry Pi computer in search of connected USB-enabled sensor/actuator modules. When at least one USB-enabled sensor module is found, the main program collects the sensor readings from the sensor module and publishes the readings to Firebase Real-Time Database. After that, the main program collects the details of connected USB device and publishes the device status to Firebase Real-Time Database. Next, the main starts the scheduler and the USB event listener. Figure 5.3 and Figure 5.4 show the start-up activities logged on the console of Raspberry Pi computer.



```
- Creating JWT using RS256 from private key file rsa_private.pem
- Device waits for connection...
Connection Result: 0: No error.
- Device connected

-- Initialising USB listener...
-- Initialisation of USB listener is completed.

Subscribed:  (1,)
Allowing maximum 0.3 seconds of delay before publishing sensor readings...

No sensor reading is collected

Updating Device Status...


# Scheduler started

Published message acked.
```

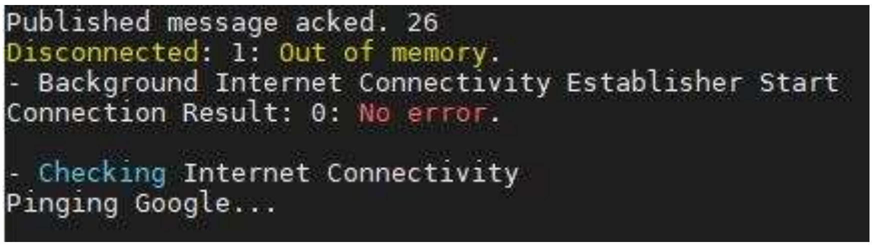
Figure 5.3 Start-Up Activities with No Sensor Connected



```
pi@raspberrypi:~/Desktop/MAIN_2 $ python main8.py --project_id=fyp-testing-255407 --registr
le=rsa_private.pem --algorithm=RS256
- Creating JWT using RS256 from private key file rsa_private.pem
- Device waits for connection...
Connection Result: 0: No error.
- Device connected

-- Initialising USB listener...
-- Initialisation of USB listener is completed.

Subscribed:  (1,)
- sensor: ths100 is connected successfully
- Creating Child Thread for sensor readings of ths100 with ID: 012
- sensor: ths012 is connected successfully
- Creating Child Thread for sensor readings of ths012 with ID: 010
Allowing maximum 0.5 seconds of delay before publishing sensor readings...

# Collected sensor readings from sensor ths100
--{'thingId': 'ths100', 'teledata': [{'value': '30', 'timestamp': '20191120174859', 'dataTy
0191120174859', 'dataType': 'humidity'}], 'name': 'ths100'}
# Collected sensor readings from sensor ths012
--{'thingId': 'ths012', 'teledata': [{'value': '29', 'timestamp': '20191120174859', 'dataTy
0191120174859', 'dataType': 'humidity'}], 'name': 'ths012'}



- Publishing Sensor Readings at time: 20191120174859 with 2 success and 0 failed from total
### Sensor Thread Payload --{"deviceId": "box_1", "things": [{"thingId": "ths100", "teledat
"dataType": "temperature"}, {"value": "81", "timestamp": "20191120174859", "dataType": "hum
teledata": [{"value": "29", "timestamp": "20191120174859", "dataType": "temperature"}, {"va
": "humidity"}], "name": "ths012"}]}

Updating Device Status...

Published message acked.

# Scheduler started
```

Figure 5.4 Start-Up Activities with At Least One Sensor Connected

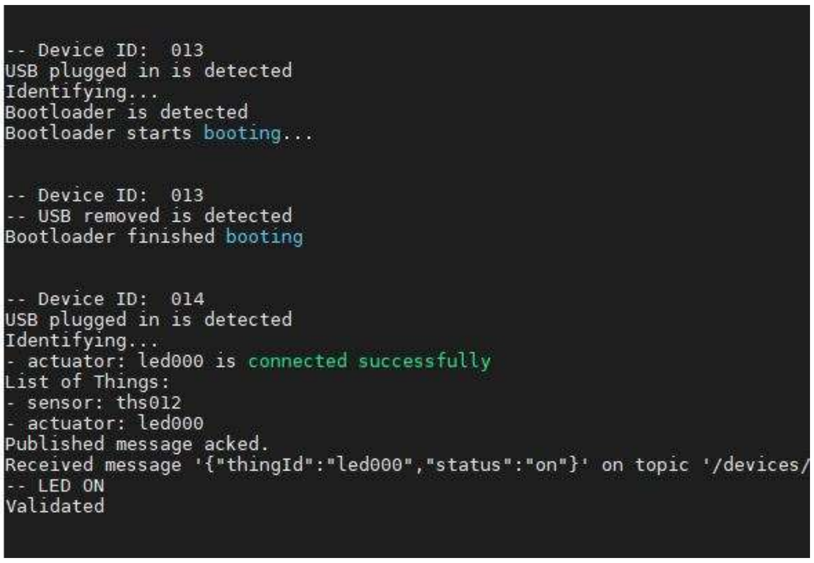
If the Raspberry Pi computer disconnected from Internet, the main program starts a background thread to actively check the Internet connectivity as shown in Figure 5.5. When Internet connectivity is back, the main program connects the Raspberry Pi computer to the Cloud IoT Core again.



```
Published message acked. 26
Disconnected: 1: Out of memory.
- Background Internet Connectivity Establisher Start
Connection Result: 0: No error.

- Checking Internet Connectivity
Pinging Google...
```

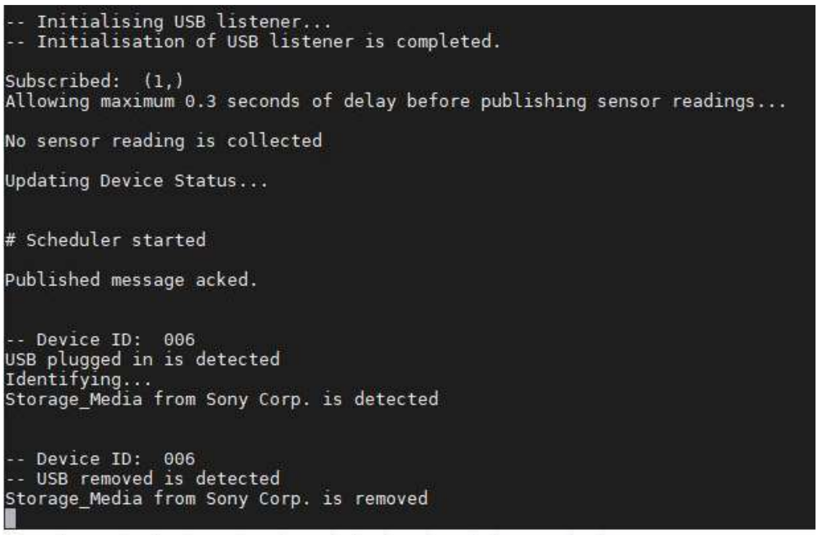
Figure 5.5 Start Background Thread for Reconnection

When the USB event listener started, it will detect USB plugged-in event and USB removed event. As shown in Figure 5.6 and Figure 5.8, the main program is able to recognise the plugged-in USB device is whether supported by the proposed solution or not. When USB-enabled sensor module is detected, the main program will collect sensor readings and publish the readings to Firebase Real-Time Database after a short delay. Besides, the main program is also able to recognise unsupported USB device, but the program will not perform anything and only display a message at Raspberry Pi computer console as shown in Figure 5.7.



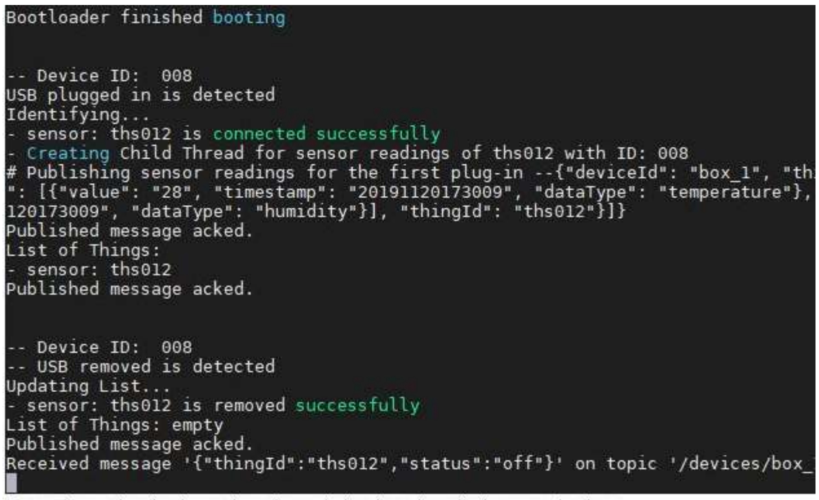
```
-- Device ID:  013
USB plugged in is detected
Identifying...
Bootloader is detected
Bootloader starts booting...


-- Device ID:  013
-- USB removed is detected
Bootloader finished booting


-- Device ID:  014
USB plugged in is detected
Identifying...
- actuator: led000 is connected successfully
List of Things:
- sensor: ths012
- actuator: led000
Published message acked.
Received message '{"thingId":"led000","status":"on"}' on topic '/devices/
-- LED ON
Validated
```

Figure 5.6 USB-Enabled Actuator Module Plugged-in Activity



```
-- Initialising USB listener...
-- Initialisation of USB listener is completed.

Subscribed:  (1,)
Allowing maximum 0.3 seconds of delay before publishing sensor readings...

No sensor reading is collected

Updating Device Status...


# Scheduler started

Published message acked.


-- Device ID:  006
USB plugged in is detected
Identifying...
Storage_Media from Sony Corp. is detected


-- Device ID:  006
-- USB removed is detected
Storage_Media from Sony Corp. is removed
```

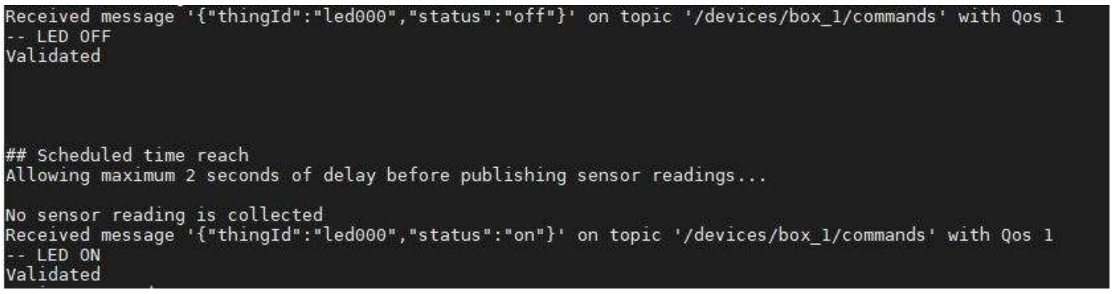
Figure 5.7 Unsupported USB Device Plugged-in Activity



```
Bootloader finished booting


-- Device ID:  008
USB plugged in is detected
Identifying...
- sensor: ths012 is connected successfully
- Creating Child Thread for sensor readings of ths012 with ID: 008
# Publishing sensor readings for the first plug-in --{"deviceId": "box_1", "th
": [{"value": "28", "timestamp": "20191120173009", "dataType": "temperature"},
120173009", "dataType": "humidity"}], "thingId": "ths012"}]}
Published message acked.
List of Things:
- sensor: ths012
Published message acked.


-- Device ID:  008
-- USB removed is detected
Updating List...
- sensor: ths012 is removed successfully
List of Things: empty
Published message acked.
Received message '{"thingId":"ths012","status":"off"}' on topic '/devices/box_
```

Figure 5.8 USB-Enabled Sensor Plugged-in Activity

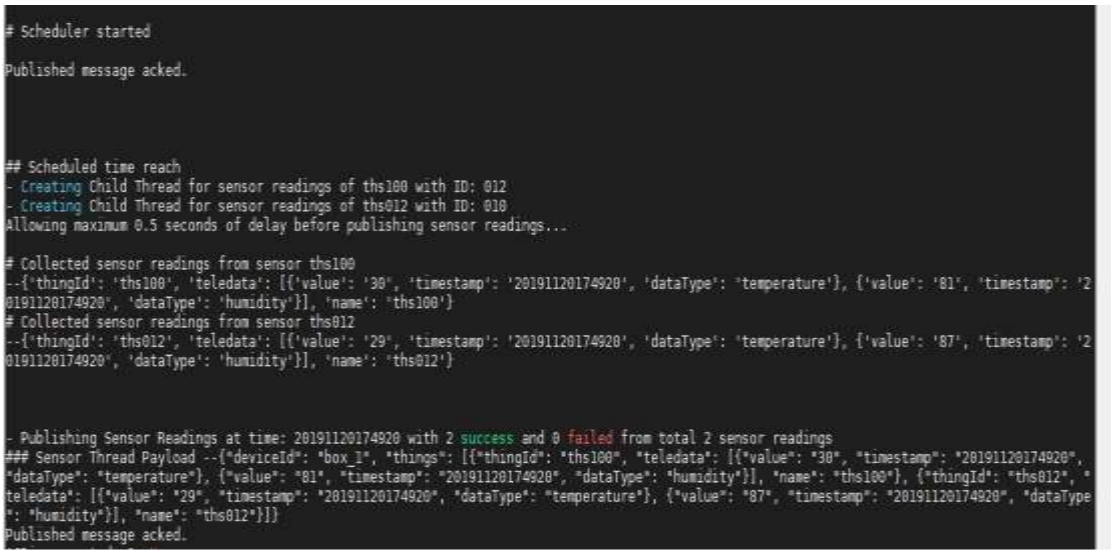
Furthermore, while the main program received commands from Google Cloud Platform, the main program passes the commands to the target USB-enabled actuator module in order to turn on or off the actuator on the USB-enabled actuator module as shown in Figure 5.9.



```
Received message '{"thingId":"led000","status":"off"}' on topic '/devices/box_1/commands' with Qos 1
-- LED OFF
Validated



## Scheduled time reach
Allowing maximum 2 seconds of delay before publishing sensor readings...

No sensor reading is collected
Received message '{"thingId":"led000","status":"on"}' on topic '/devices/box_1/commands' with Qos 1
-- LED ON
Validated
```

Figure 5.9 Activities of Performing Cloud Commands

Moreover, the scheduler in the main program is working to collect and publish sensor readings after periodically. When scheduled time reach, the main program starts to search for every USB-enabled sensor modules connected on Raspberry Pi computer and then collect the sensor readings from each sensor. The collected sensor readings will be published in a batch in order to optimise telemetry as shown in Figure 5.10. Besides, the main program will not do anything if there is no UBS-enabled sensor connected on Raspberry Pi computer when scheduled time reached.

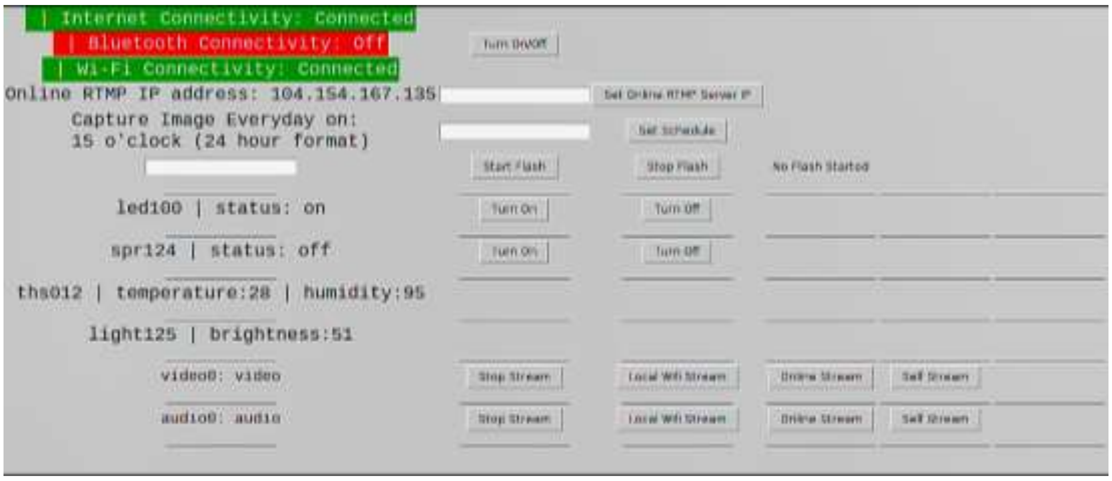


```
# Scheduler started

Published message acked.



## Scheduled time reach
- Creating Child Thread for sensor readings of ths100 with ID: 012
- Creating Child Thread for sensor readings of ths012 with ID: 010
Allowing maximum 0.5 seconds of delay before publishing sensor readings...

# Collected sensor readings from sensor ths100
--{'thingId': 'ths100', 'teledata': [{'value': '30', 'timestamp': '20191120174920', 'dataType': 'temperature'}, {'value': '81', 'timestamp': '2
0191120174920', 'dataType': 'humidity'}], 'name': 'ths100'}
# Collected sensor readings from sensor ths012
--{'thingId': 'ths012', 'teledata': [{'value': '29', 'timestamp': '20191120174920', 'dataType': 'temperature'}, {'value': '87', 'timestamp': '2
0191120174920', 'dataType': 'humidity'}], 'name': 'ths012'}



- Publishing Sensor Readings at time: 20191120174920 with 2 success and 0 failed from total 2 sensor readings
### Sensor Thread Payload --{"deviceId": "box_1", "things": [{"thingId": "ths100", "teledata": [{"value": "30", "timestamp": "20191120174920",
"dataType": "temperature"}, {"value": "81", "timestamp": "20191120174920", "dataType": "humidity"}], "name": "ths100"}, {"thingId": "ths012", "
teledata": [{"value": "29", "timestamp": "20191120174920", "dataType": "temperature"}, {"value": "87", "timestamp": "20191120174920", "dataType
": "humidity"}], "name": "ths012"}]}
Published message acked.
```

Figure 5.10 Scheduled Time Reach with At Least One Sensor Connected

5.2.2 Graphic User Interface for Debugging Purposes

A GUI has been successfully created for debugging process based on GUI design specified in the System Design. This GUI simplifies the process of displaying the device status of the proposed IoT device as well as the details of each connected USB-device. If the connected USB device is controllable by the end user, the GUI displays the same control option. Thus, developer can test the functionality of the proposed IoT device during development process without waiting for the commands sent by the monitoring mobile application.



```
| Internet Connectivity: Connected
| Bluetooth Connectivity: Off
| Wi-Fi Connectivity: Connected
Online RTMP IP address: 104.154.167.135
Capture Image Everyday on:
15 o'clock (24 hour format)
No Flash Started
led100 | status: on
spr124 | status: off
ths012 | temperature:28 | humidity:95
light125 | brightness:51
video0: video
audio0: audio
```

Figure 5.11 GUI Implemented for Debugging Purposes

5.3 The Proposed USB-Enabled Sensor/Actuator Modules (Slave Devices)

5.3.1 The Proposed USB-Enabled Actuator Modules

5.3.1.1 USB-Enabled LED Module

A USB-enabled LED module has been developed based on the system flowchart and schematic diagram in System Design. A LED board has integrated with a common Type-A USB cable and a USB-enabled microcontroller, Digispark ATTINY85 Mini USB Development Board, in order to provide plug-and-play feature that supported by the proposed solution. This USB-enabled LED module will receive commands from the proposed IoT device via the USB cable and turn on /off the LED module attached on it. This USB-enabled LED module has integrated with other supporting materials, such as a block of wood, in order to provide ease of installation on the container designed for use case studies.

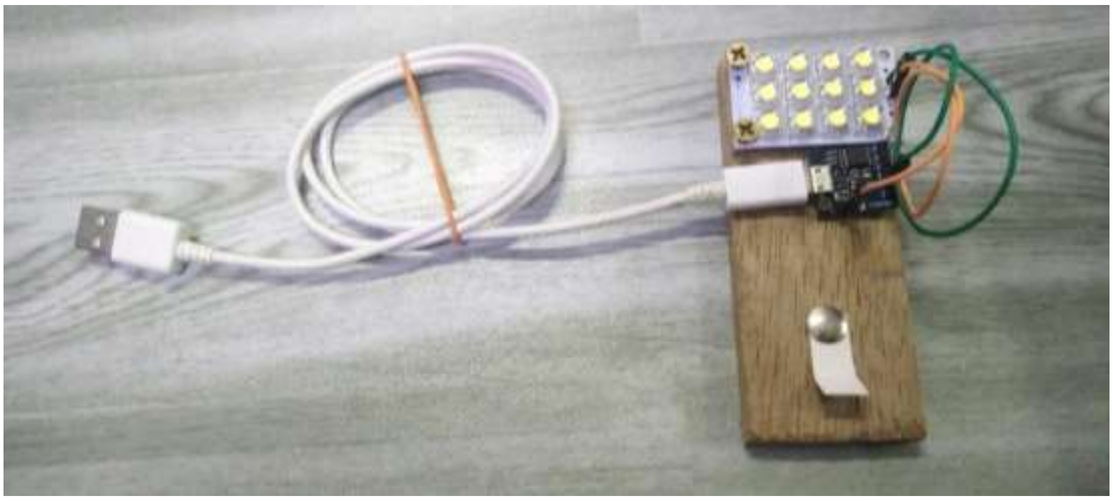


Figure 5.12 USB-Enabled LED Module

5.3.1.2 USB-Enabled Water Sprinkler Module

This USB-Enabled Water Sprinkler Module is a USB-enabled actuator module developed based on the specifications in System Design. A protection circuit and a water pump have been integrated with Digispark ATTINY85 Mini USB Development Board as well as a USB cable to provide plug-and-play sprinkler control. The sprinkler module will turn on/off its water pump based on the commands received from the proposed IoT device. A pipe is connected to it for use case study purosed.

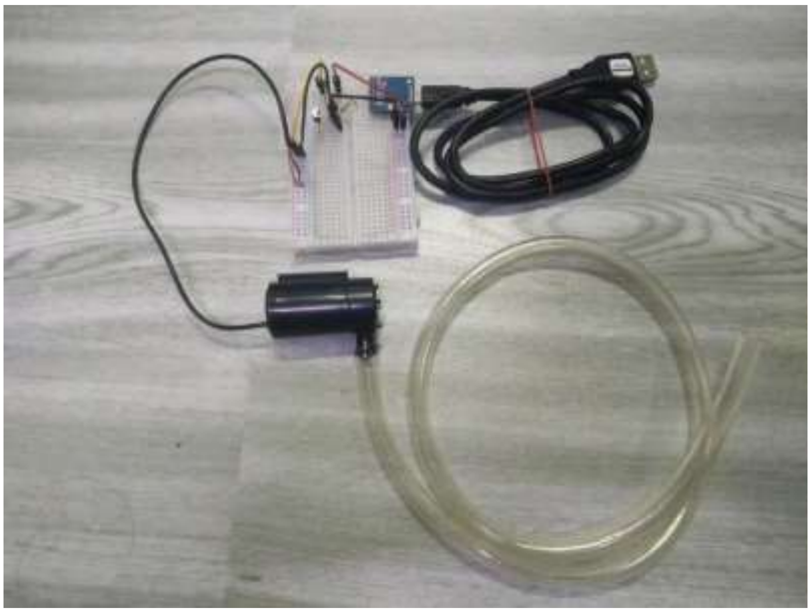


Figure 5.13 USB-Enabled Water Sprinkler Module

5.3.2 The Proposed USB-Enabled Sensor Modules

5.3.2.1 USB-Enabled Temperature and Humidity Sensor Module

Based on the design specification in System Design, a temperature & humidity sensor is integrated into a Digispark ATTINY85 Mini USB Development Board. The USB-enabled microcontroller will reads the sensor and send the sensor readings to the proposed IoT device via a USB cable. This USB-Enabled Temperature and Humidity Sensor Module enables user to add a temperature & humidity sensor to the proposed IoT device through plug-and-play USB interface without any programming and wiring skills.

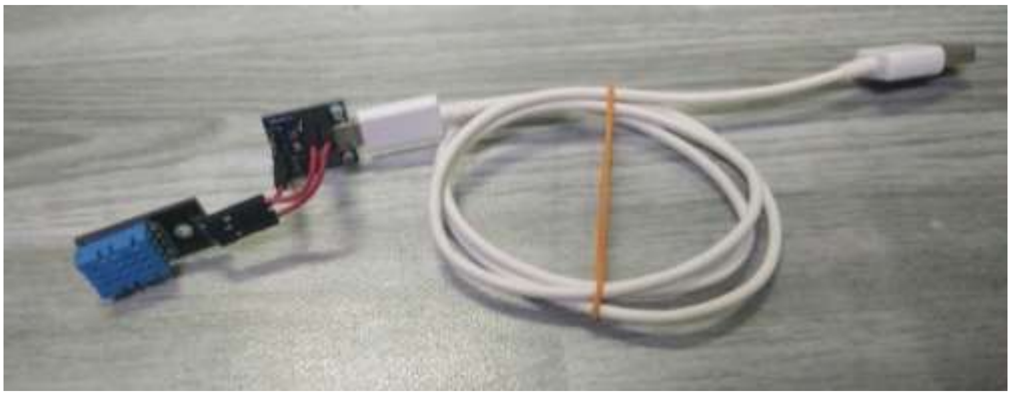


Figure 5.14 USB-Enabled Temperature and Humidity Sensor Module

5.3.2.2 USB-Enabled Light Sensor Module

This USB-Enabled Light Sensor Module reads the intensity of light and passes the sensor readings to the proposed IoT device via a USB cable. A LDR module has integrated with a Digispark ATTINY85 Mini USB Development Board and a USB cable in order to provide plug-and-play feature for adding a light sensing module into the proposed solution.

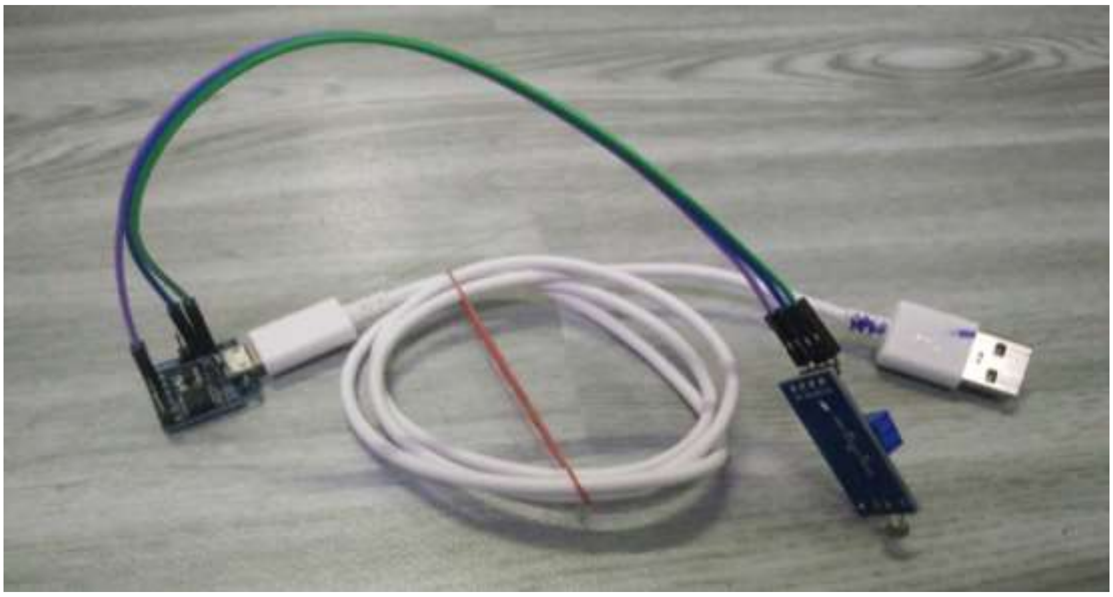


Figure 5.15 USB-Enabled Light Sensor Module

5.4 Container for Use Case Studies

A customised container for small scale IoT solution has been developed for use case studies. This container enables users to install the proposed solution onto it for a custom IoT solution. The container also allows the users to perform modifications of their IoT solution or even transform their IoT solution into other IoT solution from different industry. There is a small wooden box built on the container to place and protect the proposed IoT device. Besides, the wooden box also enables cables from outside to be connected to the proposed IoT device. There is a specially designed wooden part under the cover of container. The wooden part will allows the proposed USB-enabled sensor/actuator modules to be installed inside the container and at the same time providing ease of removal.

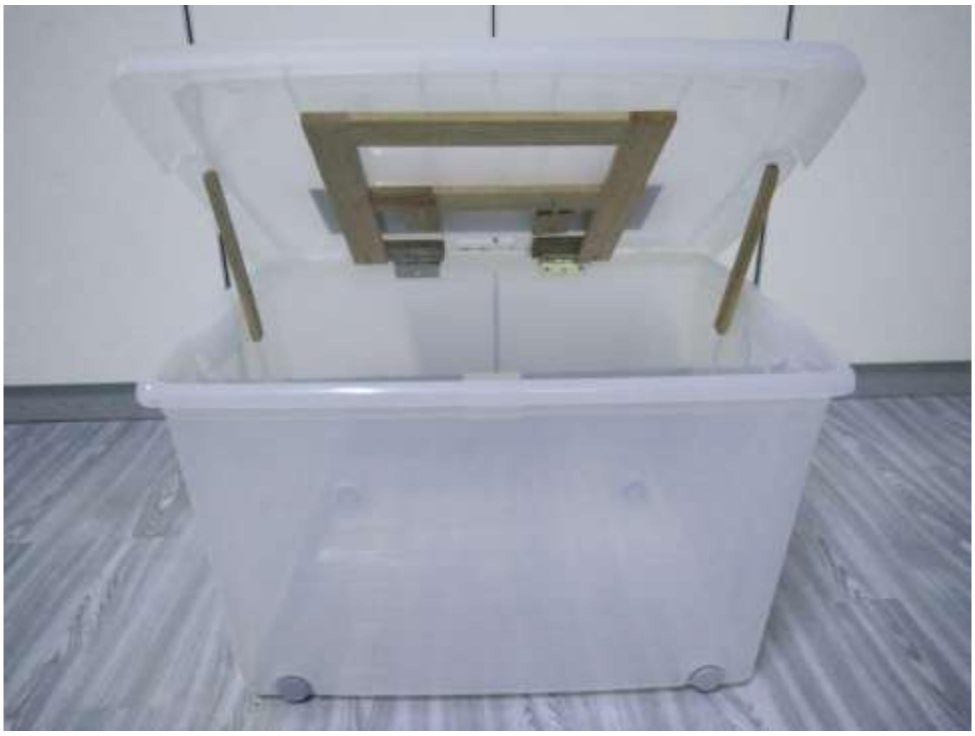


Figure 5.16 Container for Use Case Studies

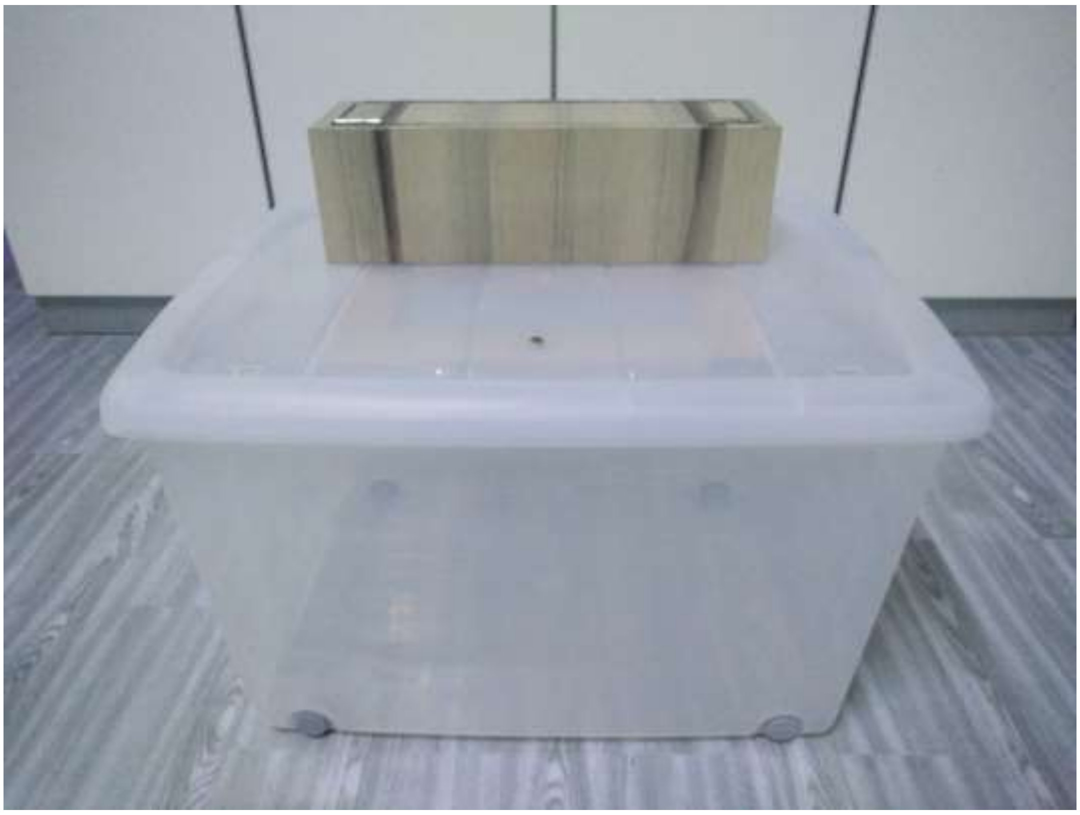


Figure 5.17 Front View of Container for Use Case Studies

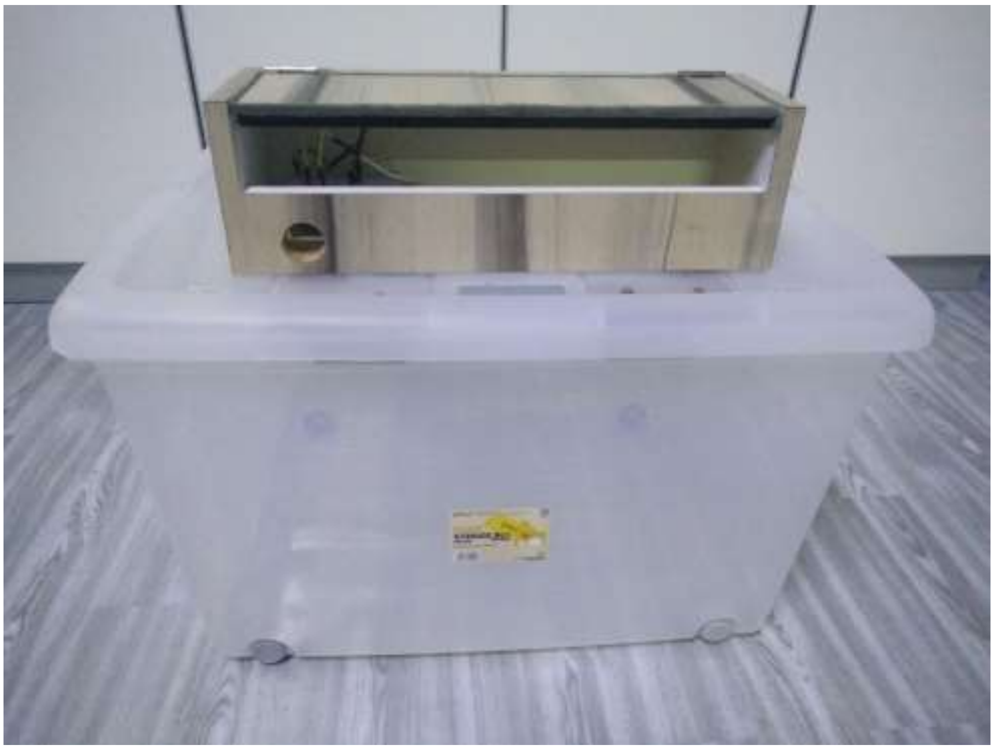


Figure 5.18 Back View of Container for Use Case Studies

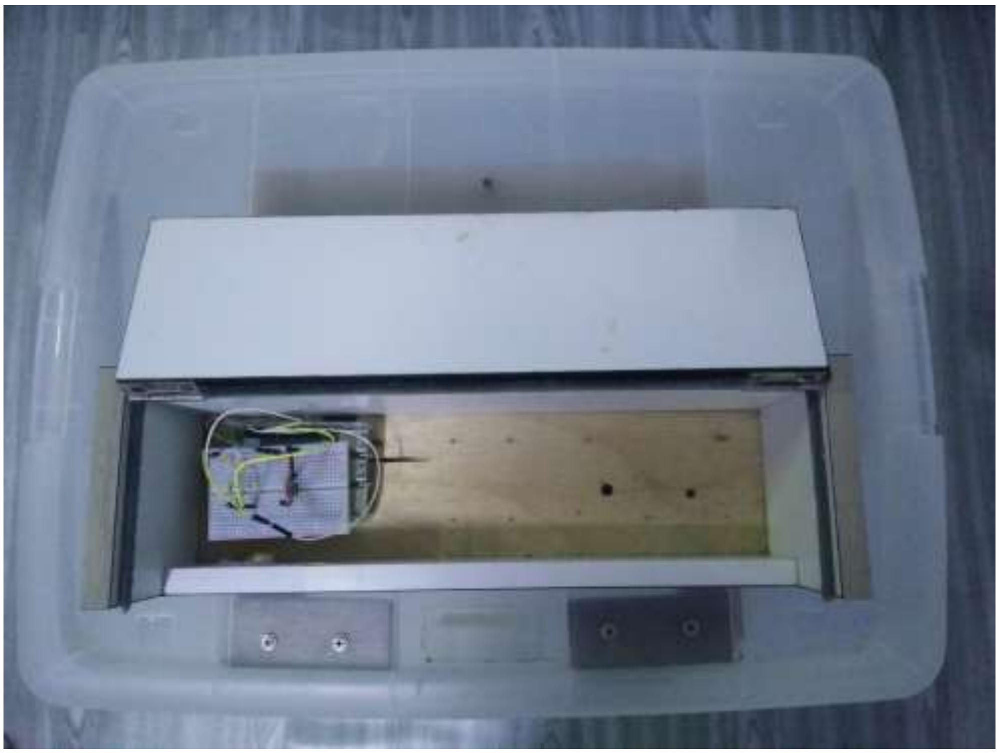


Figure 5.19 Wooden Box on Top of Container for Use Case Studies

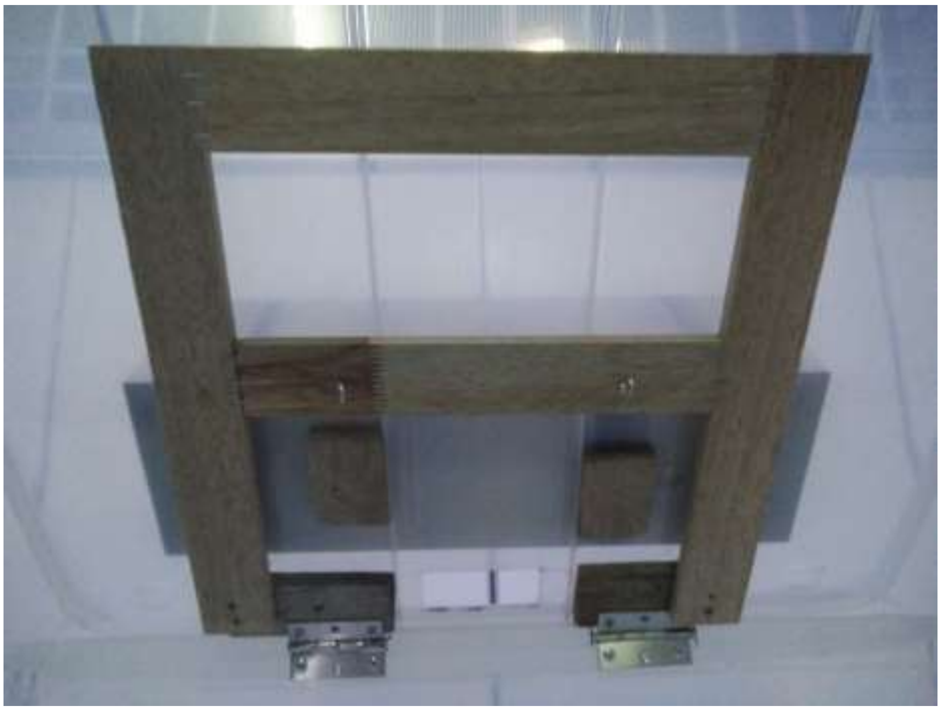


Figure 5.20 Under the Cover of The Container

5.5 The Proposed Solution

By combining the proposed IoT device and the proposed USB-enabled sensor/actuator modules, the proposed solution is form. The proposed solution enables users to build a custom IoT solution that can fulfils their requirements easily. The proposed solution introduces a concept to connect sensors and actuators into a system by using USB communication protocol and plug-and-play features. The functionality of the proposed solution will be increased as more modules added into the proposed solution. The proposed solution also supports other device with USB interface, however currently the proposed solution only supports a USB web camera for image capturing and live video streaming. Moreover, a USB hub with external power supply can be added to the proposed solution, thus the proposed solution is scalable. More USB-enabled sensor/actuator modules can be added to the proposed solution by simply plugging. The removal processes of the USB-enabled sensor/actuator modules will only involve the action to unplug the target USB device.

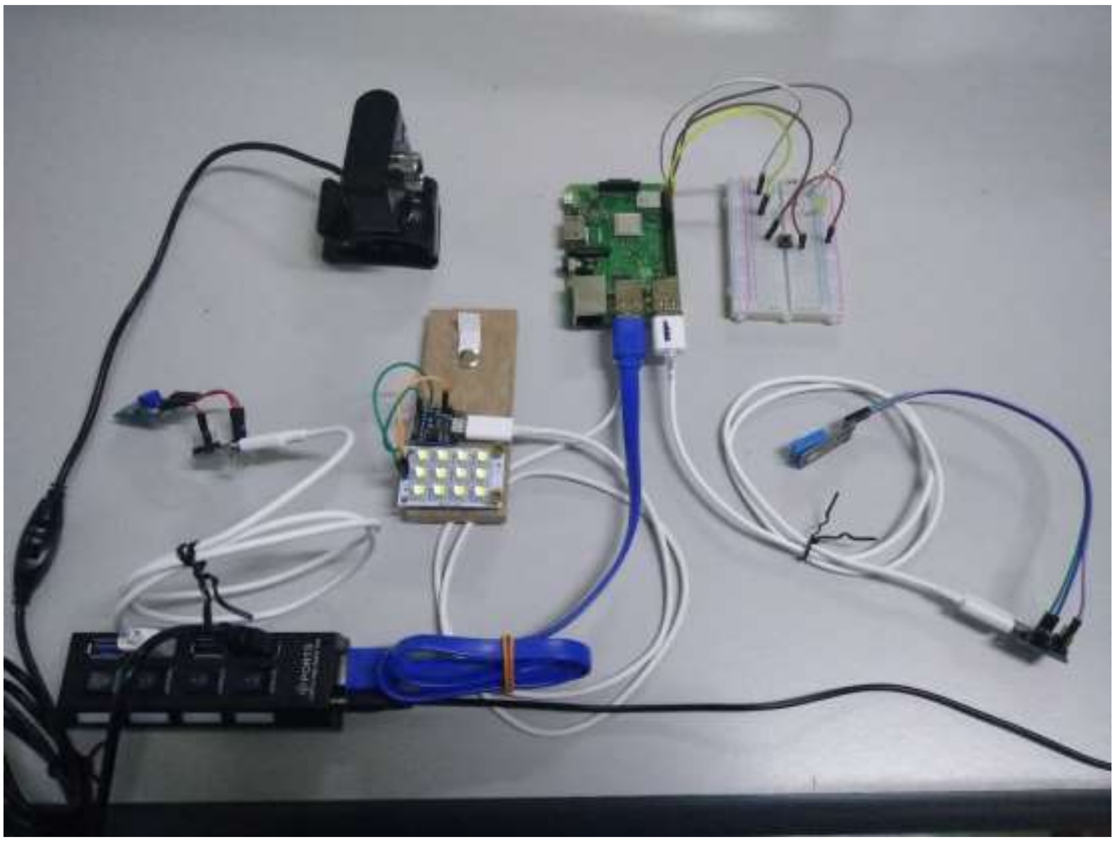


Figure 5.21 The Proposed Solution

This project is only focus on small scale agriculture industry, thus the proposed solution is used to build a small scale agriculture IoT solution on the container for use case studies. The USB-enabled sensor/actuator modules involved are the temperature & humidity sensor module, LED module, light sensor module, and water sprinkler module. A USB web camera is also involved in the use case. Except the water sprinkler module, other USB-enabled sensor/actuator modules are installed under the cover of the container. After all the USB devices are installed and plugged-in to the proposed IoT device, the end user can monitor the proposed solution through the monitoring mobile application developed by another team member and start to set schedulers as well as automation.

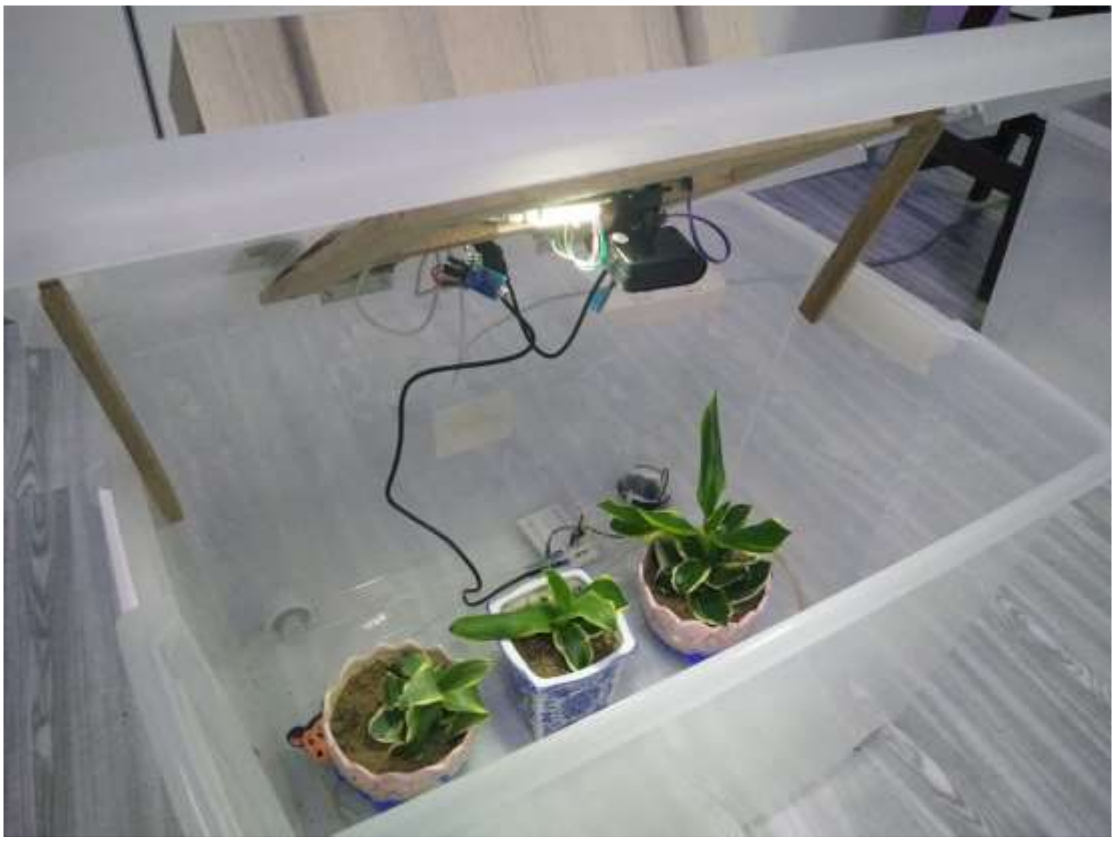


Figure 5.22 The Proposed Solution Implemented in Small Agriculture Industry

CHAPTER 6: CONCLUSION

6.1 Project Review and Discussion

Internet of Things technology has brought countless of benefits for industries as well as enhancing people’s daily life. By assigning IP address to sensing electronics, the readings of sensors are available on the Internet platform and allowing people to monitor their IoT solution. Internet of Things in industrial setting known as Industrial Internet of Things (IIoT) enables an industry to implement a cyber-physical system that optimises the manufacturing processes. The impacts of IIoT are very significant as companies can use IIoT concept to generate trillions of economic values.

However, some of current existing IoT solutions are having some issues in terms of user-friendliness and customisation. After reviewing some of the existing IoT applications, some of them are lack of plug-and-play features that provides user-friendliness characteristic, and some of them are lack of customisable platforms. IoT applications such as IoT in a Box offer great user-friendly customisation, but the applications are not cost effective. On the other hand, IoT applications like Ezblock Pi don’t offer user-friendliness. Furthermore, most of the IoT solutions are fixed to support specific use case and lack of conventional customisable IoT platform is the biggest issue faced by industries. Besides, lack of user-friendly customisable IoT platforms is also an issue for non-technical user to customise an IoT solution.

Hence, a solution that aims to provide user-friendly customisation is proposed. The objectives of the proposed solution are to develop a user-friendly customisable IoT platform with plug-and-play sensors and actuators as well as to develop USB-enabled sensor/actuator modules by using customised microcontrollers. By using the proposed solution, user can develop a custom IoT solution with the least amount of effort. The proposed solution enables user to modify their IoT solution by using plug-and-play USB-enabled sensor/actuator modules. In addition, user is not required to perform any technical skill such as programming, wiring or soldering while developing or modifying their custom IoT solution.

6.2 Novelties and Contributions

The concept of using preconfigured computer and customised USB-enabled microcontrollers in an IoT solution to perform USB communication for data exchange and automatic USB device identification is the main novelty of this project. By using this concept, a sensor or an actuator can be added to an IoT solution through plug-and-play as the identification and functionality of sensor/actuator are managed by preconfigured computer and customised microcontrollers. The USB-enabled sensor/actuator modules which have integrated the customised microcontroller can be identified by the preconfigured computer because the details of the modules will be sent by the customised microcontrollers to the main program of the preconfigured Raspberry Pi computer when being plugged-in. Besides, sensor readings collected by the customised microcontroller will be transfer to the preconfigured computer when the preconfigured computer sent the commands to it. In addition, the actuator connected to the customised microcontroller also can be controlled by the preconfigured computer via commands sent in USB communication protocol.

Furthermore, by adopting the USB communication protocol for data transfer, user can modify the IoT solution by simply plugging in or removing the target USB device. This eliminates the traditional ways of modifying an IoT solution which requires programming skills and complex wiring processes. Another contribution is the proposed solution simplifies the process to arrange messy wires used to connect sensors and actuators for a distance. Moreover, the proposed solution avoids complex configuration process for a user to configure Cloud architecture for developing an IoT solution. This is due to the preconfigured computer, which is the proposed IoT device, has been configured with Cloud connectivity. Lastly, the IoT solution developed by using the proposed solution is scalable as it adopted USB communication protocol for data transfer. Thus, more USB-enabled sensor/actuator modules can be added to the proposed solution by using a USB hub.

6.3 Future Work

The proposed solution has the ability to provide a lot of IoT solutions for many industries. However, due to time constraint, this project is only focus on small agriculture IoT solution with only a few USB-enabled sensor/actuator modules developed. Thus, there are still a lot of works can be carried out in the future.

Knowing that the proposed solution has adopted USB interface as main interface for connecting the proposed USB-enabled sensor/actuator modules to the proposed IoT device, a database can be implemented into the proposed IoT device in order to recognise other USB devices which can be found in the market for integration. If the USB device can be found in the database, this means that the USB device can be integrated into the proposed solution and then the corresponding program will be downloaded from Cloud Storage in order to interact with the connected USB device. For example, a user has plug-in a USB printer and a corresponding program will be downloaded into the proposed IoT device. After that, the user can send his image to the proposed IoT device by using the monitoring mobile application developed by another team member and the image can be printed out through a command sent by the monitoring mobile application.

Furthermore, the proposed solution will have the ability to support many industries in the future. Thus, a website will be developed in order to provide detailed information of the proposed solution, including suggested and possible combination of the proposed USB-enabled sensor/actuator modules for each IoT solution in different industries. Moreover, the live video streams of the proposed solution currently only can be view by the monitoring mobile application developed by another team member through the Internet. Thus, the local web server implemented in the proposed solution will be improved in order to enable user to watch the live video streams by browsing the webpage of the proposed solution through Local Area Network. This can enable user to watch the live video stream even when there is no Internet. Besides, currently user can only send commands to the proposed solution by using the monitoring mobile application developed by another team member via the Internet. Hence, in the future, the proposed solution will be configured in order to receive complex commands via Bluetooth other than just receiving Wi-Fi configuration commands.

BIBLIOGRAPHY

Matthys, N., Yang, F., Daniels, W., Joosen, W. and Hughes, D. (2016). Demonstration of MicroPnP: The Zero-Configuration Wireless Sensing and Actuation Platform. 2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON).

Aliexpress. (n.d.). US $28.11 24% OFF|Tuya smart WIFI power strip EU standard with 4 plug and 4 USB port compatible with Amazon Alexa and Google Nest-in Home Automation Modules from Consumer Electronics on Aliexpress.com | Alibaba Group. [online] Available at: https://www.aliexpress.com/item/32921984017.html [Accessed 14 Aug. 2019].

Iotinabox. (2019). IoT in a Box – Turnkey IoT Solutions. [online] Available at: https://www.iotinabox.com/ [Accessed 3 Aug. 2019].

Kamat, H. (2017). What is the Custom IoT Solutions?. [online] Quora.com. Available at: https://www.quora.com/What-is-the-Custom-IoT-Solutions/answer/Hrishikesh-Kamat [Accessed 3 Aug. 2019].

NGINX Documentation. (2020). NGINX Docs | Web Server. [online] Available at: https://docs.nginx.com/nginx/admin-guide/web-server/ [Accessed 1 March 2020].

VersaSense. (2019). Industrial IoT Fabric | IIoT Sensors, Actuators, Wireless Devices, Gateway, Analytics Platform | VersaSense. [online] Available at: https://www.versasense.com/ [Accessed 4 Aug. 2019].

Skydrop. (n.d.). Skydrop - Smart Sprinkler Controller |. [online] Available at: https://www.skydrop.com/ [Accessed 8 Aug. 2019].

Talbott, A. (2016). *Infographic: Internet of Things generates ROI for many, but roadblocks remain*. [online] TechRepublic. Available at: https://www.techrepublic.com/article/infographic-internet-of-things-generates-roi-for-many-but-roadblocks-remain/ [Accessed 19 Nov. 2019].

The WiFi Garden. (2016). Skydrop smart sprinkler controller review with video | The WiFi Garden. [online] Available at: http://thewifigarden.com/skydrop-review [Accessed 8 Aug. 2019].

Kettenburg, E. (2013). Digispark - The tiny, Arduino enabled, usb dev board!. [online] Kickstarter. Available at: https://www.kickstarter.com/projects/digistump/digispark-the-tiny-arduino-enabled-usb-dev-board/posts [Accessed 7 Oct. 2019].

Kickstarter. (2019). Ezblock Pi - The Ultimate Companion Board for RPi Projects!. [online] Available at: https://www.kickstarter.com/projects/35410622/ezblock-pi-the-ultimate-companion-board-for-rpi-projects [Accessed 10 Aug. 2019].

Klubnikin, A. (2017). IoT Agriculture: How to Build a Smart Greenhouse?. [online] Custom Software Development Company. Software Development Services in USA. Available at: https://r-stylelab.com/company/blog/iot/iot-agriculture-how-to-build-smart-greenhouse [Accessed 11 Aug. 2019].

Morgan, J. (2014). A Simple Explanation Of 'The Internet Of Things'. [online] Forbes.com. Available at: https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#49a8dfee1d09 [Accessed 12 Aug. 2019].

Petrov, C. (2019). 40 Internet Of Things Statistics from 2019 to Justify The Rise of IoT. [online] Tech Jury. Available at: https://techjury.net/stats-about/internet-of-things-statistics/ [Accessed 12 Aug. 2019].

Raspberry Pi Foundation. (n.d.). GPIO - Raspberry Pi Documentation. [online] Available at: https://www.raspberrypi.org/documentation/usage/gpio/ [Accessed 13 Aug. 2019].

Wiesner, S. (2010). Monitor – monitor devices — pyudev 0.13 documentation. [online] Pyudev.readthedocs.io. Available at: https://pyudev.readthedocs.io/en/v0.13/api/monitor.html [Accessed 24 Oct. 2019].

APPENDICES – WEEKLY LOG

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 1**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE

Report on current progress of FYP 2 after semester break.

During semester break, an idea of creating a GUI for system debugging purposes has formed and has brainstormed for possible solutions to develop the GUI.

Discuss with supervisor

2. WORK TO BE DONE

Develop the GUI based on the design formed during semester break.

3. PROBLEMS ENCOUNTERED

The GUI needs to be updated dynamically based on number of connected USB-enabled sensor/actuator modules.

4. SELF EVALUATION OF THE PROGRESS
Good progress

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 2**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE

The GUI mentioned in last week has been implemented

2. WORK TO BE DONE

Searching for solution to turn on/off Bluetooth of Raspberry Pi programmatically by pressing a hardware button.

Searching for solution to configure Wi-Fi of Raspberry Pi programmatically.

3. PROBLEMS ENCOUNTERED

It is very difficult to turn on/off Bluetooth of Raspberry Pi programmatically via python programming language as no standard solution is found.

4. SELF EVALUATION OF THE PROGRESS
Hardworking while searching for solution.

[illegible]

Supervisor’s signature

[illegible]

Student’s signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 3**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE

Has allowed Raspberry Pi computer to turn on/off its Bluetooth via a hardware switch

Has allowed configuration of Wi-Fi in Raspberry Pi programmatically.

2. WORK TO BE DONE

Establish Bluetooth communication programmatically with another Bluetooth device, mobile app, by using python programming language.

3. PROBLEMS ENCOUNTERED

Required a lot of research, brainstorming and debugging process.

4. SELF EVALUATION OF THE PROGRESS
Very good progress.

[illegible]

Supervisor’s signature

[illegible]

Student’s signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 4**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE Has managed to receive and send data to another Bluetooth device programmatically. Has used the data received from another Bluetooth device to configure Wi-Fi.
2. WORK TO BE DONE Edit main program in Raspberry Pi to improve connectivity with Google Cloud Platform. Edit main program to reconnect to Cloud IoT Core once disconnected.
3. PROBLEMS ENCOUNTERED Always forgot to unpair Bluetooth devices while debugging.

4. SELF EVALUATION OF THE PROGRESS

Good progress.

[illegible] [illegible]

Supervisor’s signature   Student’s signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 5**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE Connectivity of Raspberry Pi with Google Cloud Platform has improved. The main program reconnects to Cloud IoT Core when disconnected.
2. WORK TO BE DONE Develop a container for placing the proposed solution in order to build a small agriculture IoT solution.
3. PROBLEMS ENCOUNTERED Hard to disconnect Raspberry Pi’s connectivity with Cloud IoT Core as disconnection required timeout.

4. SELF EVALUATION OF THE PROGRESS
Should improve my patient.

[illegible]

Supervisor’s signature

[illegible]

Student’s signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 6**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE

Has successfully created a container for use case studies.

2. WORK TO BE DONE

Create more USB-enabled sensor/actuator modules to match use case studies.

Develop the USB-enabled sensor/actuator modules in the way that can be installed on the container easily.

3. PROBLEMS ENCOUNTERED

A lot of time has spent to design the container.

Spent a lot of time to build the container.
4. SELF EVALUATION OF THE PROGRESS Hardworking. Has developed hands-on skills.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 7**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE

Has successfully created more USB-enabled sensor/actuator modules that match the use case studies.

USB-enabled sensor/actuator modules that can be installed into the container have successfully created.

2. WORK TO BE DONE

Edit the main program of Raspberry Pi computer to flash hex file in order to allow newly bought microcontroller (same microcontroller used in the USB-enabled sensor/actuator modules) to be transform into the proposed USB-enabled sensor/actuator modules.

3. PROBLEMS ENCOUNTERED

Spent a lot of time to design the USB-enabled sensor/actuator modules to be installed easily into the container.

4. SELF EVALUATION OF THE PROGRESS

Improved knowledge in use case development.

[illegible]

Supervisor’s signature

[illegible]

Student’s signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 8**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE

Has successfully enabled main program of Raspberry Pi to flash hex file by using USB protocol.

Main program of Raspberry Pi has successfully assign default ID to the newly formed USB-enabled sensor/actuator module.

2. WORK TO BE DONE

Improve the ID assignment and ID management of connected USB-enabled sensor/actuator modules.

Edit main program of Raspberry Pi to assign ID to the connected USB-enabled sensor/actuator modules.

3. PROBLEMS ENCOUNTERED

Hard to integrate official hex file flashing program into the proposed solution.

4. SELF EVALUATION OF THE PROGRESS
Has learned third-party program integration.

[illegible]

Supervisor’s signature

[illegible]

Student’s signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 9**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE ID assignment and management of connected USB-enabled sensor/actuator modules have improved. Main program of Raspberry Pi successfully assign an ID based on the ID of same type USB-enabled sensor/actuator modules in Firebase Real-Time Database.
2. WORK TO BE DONE Discuss with supervisor on integrating other USB devices into the proposed solution. Improve the GUI for debugging purposes.
3. PROBLEMS ENCOUNTERED Need to form an innovative solution to design the management of ID.

4. SELF EVALUATION OF THE PROGRESS

Very creative.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 10**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE Has discussed with supervisor about integrating video streaming functionality into the proposed solution. The GUI for debugging purposes has improved to allow developer to manually turn on/off USB-enabled actuator modules and Bluetooth.
2. WORK TO BE DONE Edit main program of Raspberry Pi to detect and add USB camera to the proposed solution. Host an online server to receive video stream.
3. PROBLEMS ENCOUNTERED Need to create dynamic GUI buttons with different functionality.

4. SELF EVALUATION OF THE PROGRESS

Very creative in solving dynamic GUI problem.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 11**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE Has successfully integrated a USB camera to the proposed solution. Has successfully hosted a customised RTMP server on Google Cloud Platform.
2. WORK TO BE DONE Streams live video streams to online RTMP server by using USB web camera programmatically. Evaluate the video streams functionality by viewing the live video streams.
3. PROBLEMS ENCOUNTERED Hard to edit the main program to integrate with USB device that use video interface. Lack of experience in setting up a server. Lack of experience in hosting a server on online platform.

4. SELF EVALUATION OF THE PROGRESS
Very good progress. Has learned a lot of knowledge.

[illegible] [illegible]

Supervisor’s signature Student’s signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 2, Year 3	**Study week no.: 12**
Student Name & ID: Chia Shun Cheng 1603236	
Supervisor: Dr Cheng Wai Khuen	
Project Title: User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer	

1. WORK DONE

Has successfully streamed live video streams to online RTMP server hosted in Google Cloud Platform.

Has successfully evaluated the video streaming functionality of online RTMP server by using VLC player app to view live video streams.

2. WORK TO BE DONE

Improve the report.

Finalise the report.

Prepare for presentation.

3. PROBLEMS ENCOUNTERED

Faced difficulties when capturing live video streams from camera programmatically.

Lack of experience in retrieving video stream by using RTMP protocol.

4. SELF EVALUATION OF THE PROGRESS

Good progress throughout the development process.

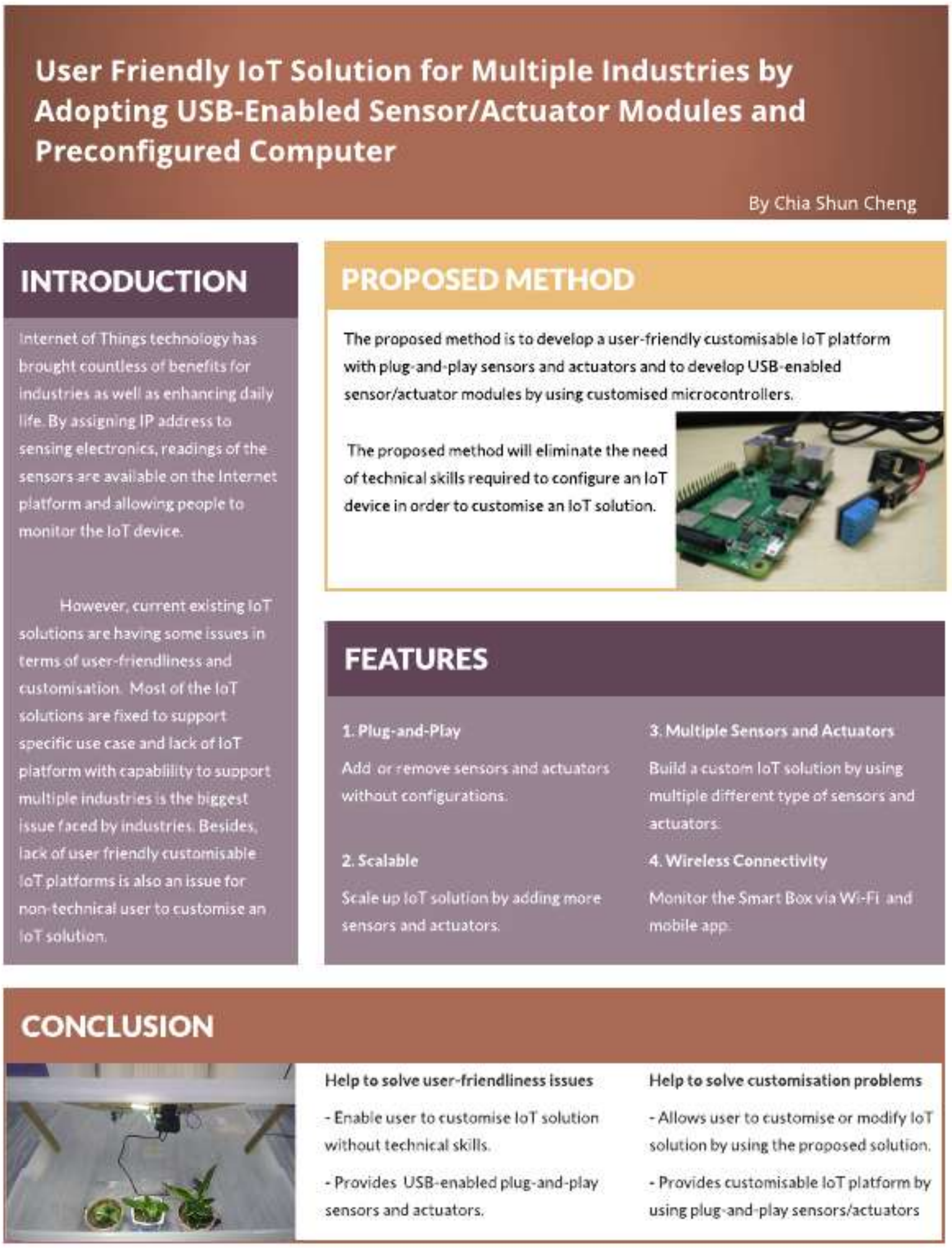
[illegible]

Supervisor’s signature

[illegible]

Student’s signature

POSTER



User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer

By Chia Shun Cheng

INTRODUCTION

Internet of Things technology has brought countless of benefits for industries as well as enhancing daily life. By assigning IP address to sensing electronics, readings of the sensors are available on the Internet platform and allowing people to monitor the IoT device.

However, current existing IoT solutions are having some issues in terms of user-friendliness and customisation. Most of the IoT solutions are fixed to support specific use case and lack of IoT platform with capability to support multiple industries is the biggest issue faced by industries. Besides, lack of user friendly customisable IoT platforms is also an issue for non-technical user to customise an IoT solution.

PROPOSED METHOD

The proposed method is to develop a user-friendly customisable IoT platform with plug-and-play sensors and actuators and to develop USB-enabled sensor/actuator modules by using customised microcontrollers.

The proposed method will eliminate the need of technical skills required to configure an IoT device in order to customise an IoT solution.

FEATURES

1. Plug-and-Play

Add or remove sensors and actuators without configurations.

2. Scalable

Scale up IoT solution by adding more sensors and actuators.

3. Multiple Sensors and Actuators

Build a custom IoT solution by using multiple different type of sensors and actuators.

4. Wireless Connectivity

Monitor the Smart Box via Wi-Fi and mobile app.

CONCLUSION

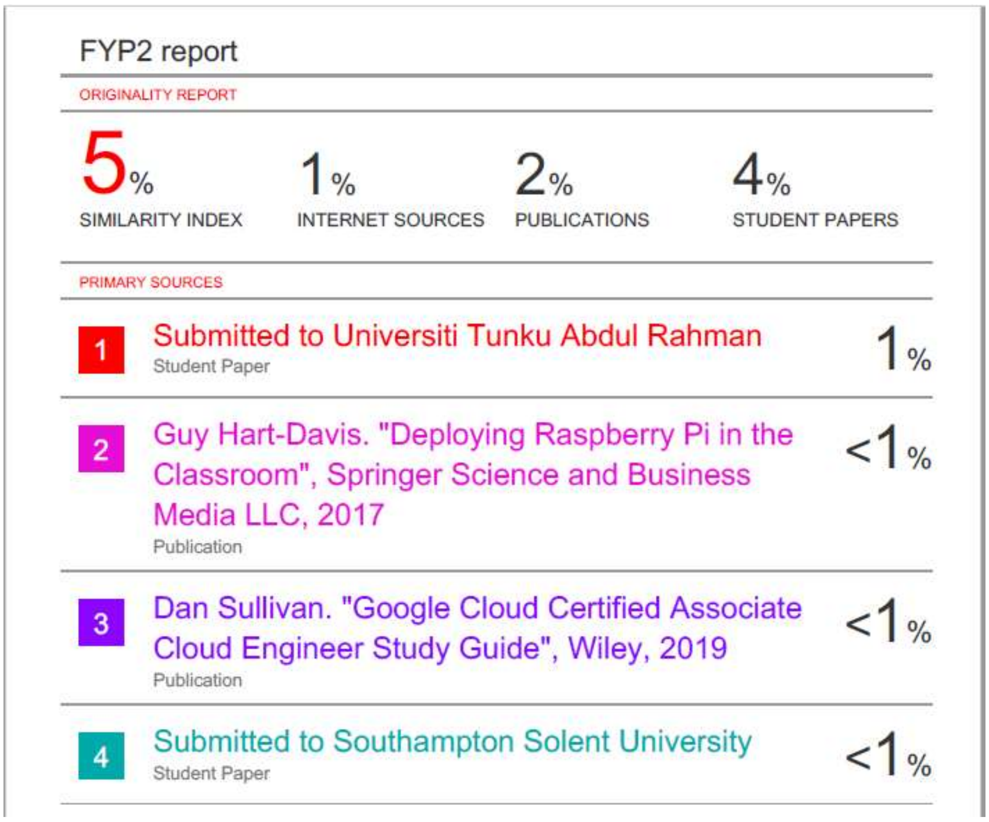
Help to solve user-friendliness issues

- Enable user to customise IoT solution without technical skills.
- Provides USB-enabled plug-and-play sensors and actuators.

Help to solve customisation problems

- Allows user to customise or modify IoT solution by using the proposed solution.
- Provides customisable IoT platform by using plug-and-play sensors/actuators

PLAGIARISM CHECK RESULT



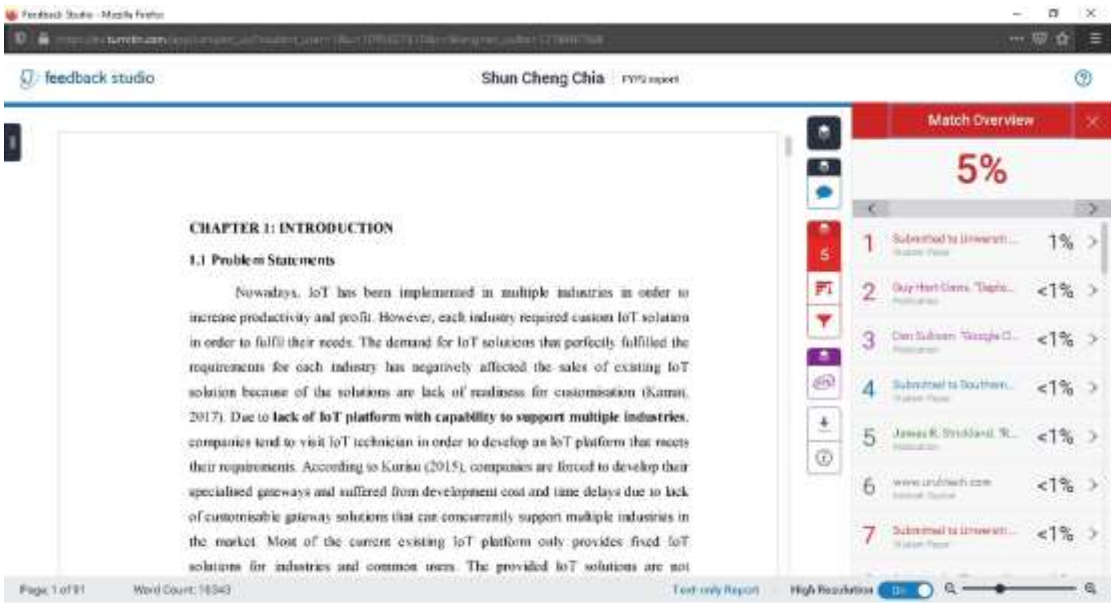
FYP2 report

ORIGINALITY REPORT

5%	1%	2%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Tunku Abdul Rahman Student Paper	1%
2	Guy Hart-Davis. "Deploying Raspberry Pi in the Classroom", Springer Science and Business Media LLC, 2017 Publication	<1%
3	Dan Sullivan. "Google Cloud Certified Associate Cloud Engineer Study Guide", Wiley, 2019 Publication	<1%
4	Submitted to Southampton Solent University Student Paper	<1%



CHAPTER 1: INTRODUCTION

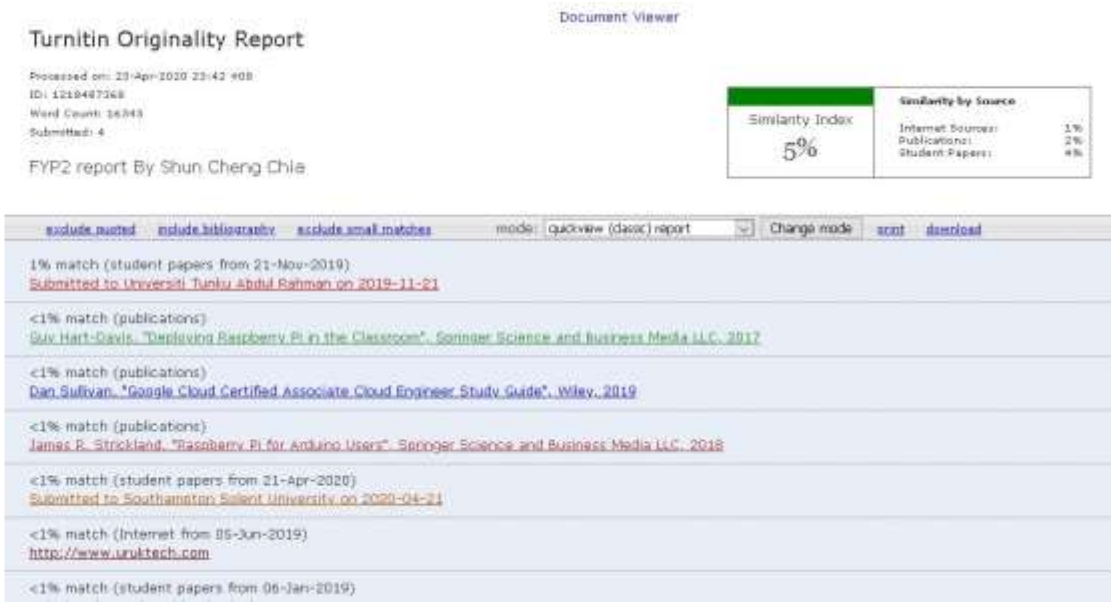
1.1 Problem Statements

Nowadays, IoT has been implemented in multiple industries in order to increase productivity and profit. However, each industry required custom IoT solution in order to fulfil their needs. The demand for IoT solutions that perfectly fulfilled the requirements for each industry has negatively affected the sales of existing IoT solution because of the solutions are lack of readiness for customisation (Kamau, 2017). Due to **lack of IoT platform with capability to support multiple industries**, companies tend to visit IoT technician in order to develop an IoT platform that meets their requirements. According to Kuriso (2015), companies are forced to develop their specialised gateways and suffered from development cost and time delays due to lack of customisable gateway solutions that can concurrently support multiple industries in the market. Most of the current existing IoT platform only provides fixed IoT solutions for industries and common users. The provided IoT solutions are not

5%

1	Submitted to Universiti...	1%
2	Guy Hart-Davis. "Deplo...	<1%
3	Dan Sullivan. "Google Cl...	<1%
4	Submitted to Southam...	<1%
5	James R. Strickland. "R...	<1%
6	www.urubtech.com	<1%
7	Submitted to Universiti...	<1%

Page: 1 of 91 Word Count: 16343 Text-only Report High Resolution On



Document Viewer

Turnitin Originality Report

Processed on: 23-Apr-2020 23:42 +08
ID: 1218487368
Word Count: 16343
Submitted: 4

FYP2 report By Shun Cheng Chia

Similarity Index 5%	**Similarity by Source**
	Internet Sources: 1%
	Publications: 2%
	Student Papers: 4%

exclude quoted include bibliography exclude small matches mode: quickview (classic) report Change mode print download

1% match (student papers from 21-Nov-2019)
Submitted to Universiti Tunku Abdul Rahman on 2019-11-21

<1% match (publications)
Guy Hart-Davis. "Deploying Raspberry Pi in the Classroom", Springer Science and Business Media LLC, 2017

<1% match (publications)
Dan Sullivan. "Google Cloud Certified Associate Cloud Engineer Study Guide", Wiley, 2019

<1% match (publications)
James R. Strickland. "Raspberry Pi for Arduino Users", Springer Science and Business Media LLC, 2018

<1% match (student papers from 21-Apr-2020)
Submitted to Southampton Solent University on 2020-04-21

<1% match (Internet from 05-Jun-2019)
http://www.uruktech.com

<1% match (student papers from 06-Jan-2019)

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Chia Shun Cheng
ID Number(s)	16ACB03236
Programme / Course	Bachelor of Information Technology (Hons) Computer Engineering
Title of Final Year Project	User Friendly IoT Solution for Multiple Industries by Adopting USB-Enabled Sensor/Actuator Modules and Preconfigured Computer

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 5 % **Similarity by source** Internet Sources: 1 % Publications: 2 % Student Papers: 4 %	no issue, OK.
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required and limits approved by UTAR are as follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

[illegible]

____
Signature of Supervisor
Name: Dr. Cheng Wai Khuen
Date: 24 April 2020

____
Signature of Co-Supervisor
Name: ______________________
Date: ______________________

FYP 2 CHECKLIST

UNIVERSITI TUNKU ABDUL RAHMAN
FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	16ACB03236
Student Name	Chia Shun Cheng
Supervisor Name	Dr. Cheng Wai Khuen

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Front Cover
✓	Signed Report Status Declaration Form
✓	Title Page
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
✓	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
✓	Appendices (if applicable)
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. [illegible] ______________________ (Signature of Student) Date: 24 April 2020	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. [illegible] ______________________ (Signature of Supervisor) Date: 24 April 2020