UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Faculty of Electrical Engineering

SMART HOME ELECTRICAL APPLIANCES CONTROLLING VIA ANDROID SMARTPHONE

Lily Ling Sui Feng

Bachelor of Electrical Engineering
(Control, Instrumentation and Automation)

2017

SMART HOME ELECTRICAL APPLIANCES CONTROLLING VIA ANDROID SMARTPHONE

LILY LING SUI FENG

A thesis submitted in fulfilment of the requirements for the bachelor of Electrical Engineering (Control, Instrumentation and Automation)

Faculty of Electrical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2017

DECLARATION

I declare that this thesis entitled "Smart Home Electrical Appliances Controlling via Android Smartphone" is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	....
Name	:	....
Date	:	....

APPROVAL

I hereby declare that I have read this dissertation/report and in my opinion this dissertation/report is sufficient in terms of scope and quality as a partial fulfilment of Bachelor of Electrical Engineering (Control, Instrumentation and Automation).

Signature	:……………………..………..…………..
Supervisor Name	:……………………………….…………
Date	:………………………………………..

ABSTRACT

This project is entitled “Smart Home Electrical Appliances Controlling Via Android Smartphone”. The advancement of smart home technology had improved human’s quality of living. There are two types of smart home network technology which are wiring system and wireless system. For wiring system, the equipment is connected directly with the main power supply and the data is sent to activate or deactivate the home appliances. For wireless system, two important elements which are sender and receiver are involved. The appliances can communicate with other devices through wireless technology such as Wi-Fi, infrared, Bluetooth, radio frequency and IEEE 802.11. The motivations of this project are to decrease the cost of smart home products and to overcome the range issue of Bluetooth technology that had implemented in many existing smart home projects. Due to the busy lifestyles of people, many electrical appliances are left turned on when it is not in use. When residents go out without turn off their home appliances, they must go home to turn off the electrical appliances to avoid any additional of electricity consumption, so it is impossible for residents to waste petrol and time to do such action. With the advance of technology, the use of internet will be developed to this smart home system. The Android smartphone will be used for controlling and monitoring the home appliances using the application. A low cost smart home system also will be developed.

ABSTRAK

Projek ini bertajuk "Pengawalan Alatan Elektrik Rumah Pintar Menggunakan Telefon Pintar Android". Kecanggihan technologi dalam rumah pintar telah meningkatkan taraf hidup manusia. Terdapat dua jenis sistem rangkaian telah digunakan dalam sistem rumah pintar iaitu sistem pendawaian dan sistem tanpa wayar. Bagi sistem pendawaian, peralatan elektik rumah akan disambung terus dengan bekalan kuasa utama. Data akan dihantar melalui saluran dawai untuk mengaktifkan atau menyahaktifkan peralatan elektrik rumah tersebut. Bagi sistem tanpa wayar, sistem ini akan melibatkan dua elemen penting iaitu penghantar dan penerima. Peralatan elektrik rumah boleh berkomunikasi dengan peralatan lain melalui teknologi tanpa wayar seperti Wi-Fi, inframerah, Bluetooth, frekuensi radio dan IEEE 802.11. Motivasi projek ini adalah untuk mengurangkan kos produk rumah pintar dan mengatasi isu jarak teknologi Bluetooth yang telah digunakan dalam banyak projek rumah pintar yang sedia ada. Pada masa kini, gaya hidup masyarakat yang sibuk telah menyebabkan banyak peralatan elektrik rumah dibuka walaupun ia tidak digunakan. Apabila pemilik rumah keluar dari rumah tanpa menutup peralatan elektrik, mereka sepatutnya pulang ke rumah untuk mematikan peralatan elektrik untuk mengelakkan pembaziran penggunaan elektrik. Walau bagaimanapun, kebanyakkan pemilik rumah tidak akan membazirkan minyak petrol kereta dan masa untuk pulang ke rumah. Dengan kecanggihan teknologi, penggunaan internat akan digunakan untuk membangunkan sistem rumah pintar yang lebih baik. Peralatan elektrik rumah boleh dikawal dan dipantau dengan menggunakan aplikasi Android. Sistem rumah pintar yang berkos rendah juga boleh dibangunkan.

ACKNOWLEDGEMENTS

First of all, I would like to take this opportunity to thank my supervisor, Associate Professor Dr. Tay Choo Chuan, Deputy Dean from the Centre For Graduates Studies Universiti Teknikal Malaysia Melaka (UteM). Thank you for his kindness, essential supervision, support and encourgement towards the completion of this final year project.

Furthermore, I would like to express my greatest gratitude to Encik Mohamad Riduwan Bin Mohamad Nawawi and Mr. Ma Tien Choon from Faculty of Electrical Engineering as my panel 1 and panel 2 for this project. Thank you for their advice and evaluation of my final year project.

Moreover, I would also like to express my deepest gratitude to Dr. Muhammad Nizam bin Kamarudin from Faculty of Electrical Engineering as the final year project coordinator for BEKC course. Thank you for his assistance and efforts in guiding all BEKC students towards the completion of this final year project.

Special thanks to all my peers, my beloved parents and siblings for their moral support in completing this degree. Lastly, thank you to everyone who had been to the crucial parts of realization of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREACIATION

AC	-	Alternating Current
ADC	-	Analog-to-Digital Converter
API	-	Application Program Interface
APK	-	Android Package Kit
BDC	-	Business Development Center
COM	-	Common
CPU	-	Central Processing Unit
DC	-	Direct Current
GND	-	Ground
GPIO	-	General-Purpose Input/Output
GUI	-	Graphical User Interface
HTML	-	Hyper Text Markup Language
HTTP	-	Hypertext Transfer Protocol
HTTPS	-	Hypertext Transfer Protocol Secure
IDE	-	Integrated Development Environment
IEEE	-	Institute of Electrical and Electronics Engineers
IIC	-	Inter-Integrated Circuit
IN1	-	Input 1
IN2	-	Input 2
IN3	-	Input 3
IN4	-	Input 4
I/O	-	Input/Output
IoT	-	Internet of Things
IP	-	Internet Protocol
JSON	-	JavaScript Object Notation
LED	-	Light Emitting Diode

MIT	-	Massachusetts Institute of Technology
MQTT	-	MQ Telemetry Transport
NC	-	Normally Closed
NO	-	Normally Open
OS	-	Operating System
PLC	-	Programmable Logic Controller
PWM	-	Pulse Width Modulation
QEMU	-	Quick Emulator
RAM	-	Random Access Memory
SD	-	Secure Digital
SDK	-	Software Development Kits
SoC	-	System on a Chip
TCP	-	Transmission Control Protocol
UI	-	User Interface
USB	-	Universal Serial Bus
Vin	-	Input Voltage
WAP	-	Wireless Application Protocol

CHAPTER 1

INTRODUCTION

1.1 Overview

This section is an introduction chapter including research background, motivation, problem statement, objectives, scope and project outline. The research background discusses the definition of smart home, the function of smart home system in residential building, the development of technology in smart home and the introduction of IoT. Furthermore, there are two motivations are discussed including the high cost of current smart home system and the weakness of Bluetooth technology. For the problem statement part, it states the general problems that will be faced by the human in electricity usage and energy management. Thus, new solutions will be suggested. The objectives are also defined and several goals are set to accomplish the proposed project. The scope describes the functionality of the proposed project and introduces the open source software and low-cost hardware that can be implemented in this project. The thesis outline lists out and summarizes the content of each chapters that will be included in this research.

1.2 Research Background

"Smart Home" is a term that defined as a residential building that links all devices and appliances with special structured wiring to allow them to communicate with each other. [1] Smart home also is a house that merged the advanced automation systems to allow the residents to monitor and remotely control the sophisticated building's functions such as multi-media, lighting, window and door operations, temperature, humidity and security system by using phone or internet. [2]

In 1984, American Association of House Builders introduced the concept of "Smart Home Automation". The significant of the growth of smart home system is to control the devices and appliances automatically. [3] The technology becomes an essential part of human lives. The implementation of the system can improve human life more technology driven and easy managed. The implementation of smart home system comprises the implementation of several technologies such as microcontroller technology, wireless technology and smartphone application technology. With the advancements of those technologies, the system can be implemented efficiently.

In 21st century, the services of internet can be expanded to IoT. In 1999, Kevin Ashton introduced the concept of IoT officially in proposal form but the concept of IoT had been introduced unofficially for almost 25 years from now. The implementation of the IoT in the smart home system can improve the quality of human's life. By using the IoT, the huge amounts of intelligent objects can detect or accumulate the data and communicate with human by using sensor, phone and wireless technologies. The basic home tasks and features can be controlled automatically using internet from anywhere. [4]

1.3 Motivation

The first motivation of this project is to decrease the cost of smart home system. The basic installation of smart home system in the current market included basic starter kit, wireless mesh systems, monthly service, cloud automation and hardwired. All the costs are expensive. Furthermore, most of the smart home system in the market requires a professional to install the hardwired system. The cost to have such installation services is high. [5] The second motivation of this project is to overcome the weakness of Bluetooth technology for controlling the home appliances. The advancement of technology has raised the usage of IoT in smart home system. The wireless technology is the most popular technology that used in controlling the home appliances remotely in the indoor environment. Currently, there are many smart home systems using Bluetooth wireless technology to connect smartphone with the microcontroller to control the home appliances. However, the technology has a few disadvantages such as slow data speeds, poor data security, short range and shortened battery life. [6] Bluetooth is operating at frequency of 2.4 GHz, it connects to a device within a range of 10-20 m at the speed of 256 Kbps to 1 Mbps. [7] Due to the short communication range and resident can only control the home appliances at limited range environment. The wireless connectivity between smartphone with the microcontroller cannot be established using Bluetooth in an outdoor environment. By using internet network, people can use a smartphone to remotely control the home appliances over internet from anywhere around the world. Internet overcomes the problem of range which is challenged in Bluetooth technology. [8]

1.4 Problem Statements

Nowadays, people are living in a busy lifestyle. Many people are always rushing when leave from their residential building and spend long time staying at outside. A lot of energy will be consumed if the resident forgets to switch off the home appliances such as lights or fan. This will lead to additional expenditure on electricity. Therefore, an internet based smart home control system is designed so that the resident can control remotely the home appliances using Android smartphone without necessarily being nearby or inside the residential building. [9]

1.5 Objectives

The objectives of this project are:

1. To develop the use of internet in smart home control system.
2. To develop a low cost smart home control system.
3. To develop an android application to control the home appliances.

1.6 Scope

The scope of this project is to remotely control the home appliances via Android smartphone over internet. The Android smartphone is required to connect with the WiFi network or mobile data, then user can use an application as an interface to switch on and switch off the home electrical appliances from anywhere around the world. This project uses the open source software such as Arduino IDE, MIT App Inventor 2, Anto and PushingBox platform and the low-cost hardware such as ESP8266-12E NodeMCU Development Board and 4-channels 5V relay module. For the prototype demonstration, the components such as resistors, LEDs, buzzer and DC motor are used to represent the electrical appliances. This project only involves four types of electrical appliances such as alarm, air-conditioner, lighting and fan.

1.7 Thesis Outline

Chapter 1 - This chapter is an introduction chapter that describes the research background, motivation, problem statement, objectives and scope.

Chapter 2 - This chapter is a literature review chapter that discusses the published information of the advantages of wireless technology, significance of smartphone, android, microcontroller, HTTP API request method, introduction to MQTT and the review of the previous related works about the smart home automation system.

Chapter 3 - This chapter is a methodology chapter that discusses the procedures to do the proposed project.

Chapter 4 - This chapter is a results and discussion chapter that discusses more on the hardware and how the whole system will be implemented together with the hardware and software.
Chapter 5 - This chapter is a conclusion and recommendation chapter that discusses the summary of the proposed project. The limitation and the weakness of the proposed project are presented in this chapter. Recommendations are suggested for the improvement of the proposed smart home system.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter is a literature review chapter that discusses the advantages of wireless technology, significance of smartphone, android, microcontroller, HTTP API request method, introduction to MQTT and the review of the previous related projects.

2.2 Advantages of Wireless Technology

Recently, the automation technology of smart home and modern building techniques increase the use of the wireless technologies such as WiFi, Bluetooth and Ethernet. The use of wireless technologies offers several advantages in home automation. The advantages of wireless technology are low installation costs, system scalability and easy extension, aesthetical benefits and integration of mobile devices. Since no cabling is required for wireless technology, the installation costs are considerably reduced. The installation of wired system requires materials for the cabling and manpower for the installation. The cost of the materials and the fee of the salary are expensive. Furthermore, when the new requirement of the smart home system is proposed or changed, it is beneficial if a wireless network is deployed. It is necessary to extend the network only and no cabling changes are required. Moreover, the wireless smart home system fulfils the aesthetical requirements because the system does not cover a large space and cable laying is not necessary. The building can remain its design and architecture. With wireless networks, the mobile devices such as PDAs and smartphones can be used to control the automation system from everywhere. As long as the device is within the range of the network, the physical location of the device is not a problem for the connection. The wireless technology is also easy to install and it does not burden the owner when owner wants to renovate and refurbish the building. [10]

2.3 Significance of Smartphone

Smartphone is defined as a portable electronic device that integrated with the advanced technologies which functions like a personal computer. Smartphone is not just a cellular phone in today’s life, it is having a wide range of application in smart home, health care, education and entertainment. The improvement of smartphone in the functionality and features increases the usage of smartphone in our lives and it influences our daily routine work. The smartphone is a typical device that widely used by everyone, therefore it is not difficult to get a smartphone. They will become more affordable and cheaper in the future. The aim of the smart home technology is to improve the convenience and comfort level of the living places in the application of energy efficiency, security and surveillance. There are several smart home systems using the wireless technologies such as Bluetooth, internet and SMS. The wireless technology is integrated with the smartphone. Thus, the smartphone can be established to the microcontroller of the smart home system using the wireless technologies. The application that created for the specific smart home system can be used as the user interface to remotely control and monitor the electrical appliances and lighting. Therefore, the smartphone is considered as a practical and convenient device for networking interaction than a computer and it is a best choice for the automation control solution. [11]

2.4 Android

Mobile OS is defined as a software platform that enables other programs to communicate and operate on mobile devices. It is important to ensure the compatibility of the functions and features such as application synchronization, WAP, keyboards, text messaging and email on mobile devices. Additionally, it is responsible for determining the third-party applications that can be used on mobile devices. Nowadays, there are several types of mobile operating systems are used by the mobile devices such as Android OS, Bada, IOS, MeeGo OS, BlackBerry OS and more. However, the Android mobile OS is the most popular and the fastest growing mobile OS that used widely around the world. Majority of mobile devices are using Android OS and it is developed by Google. Furthermore, it is a world-class and open development platform that everyone can write programs or customizes it for creating application and games that can run on any compatible mobile devices. The enhancements and improvements of the open source Android mobile OS have been developed under "dessert-inspired" version names in alphabetical order such as Cupcakes, Donut, Eclair, Gingerbread, Honeycomb and Ice Cream Sandwich. [12] [13]

Moreover, Android applications play the important role to widen the functionality of devices. The process to create and design new application for the Android OS is known as Android software development. For the Android software development, the Android SDK which comprises a set of software development tools is used and it is written in Java programming language. The SDK provides a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code and tutorials. There are a few

Google's supported SDK IDE that can be used to create application's APK file such as Eclipse, Android Studio, NetBeans, Apache Cordova and MIT App Inventor 2. Then, the developers are available to publish the APK file to Google Play Store. Google Play Store as the primary application store program consists many third-party applications that can be acquired by Android device users. Users can install, update and remove the application's APK from their devices. [14]

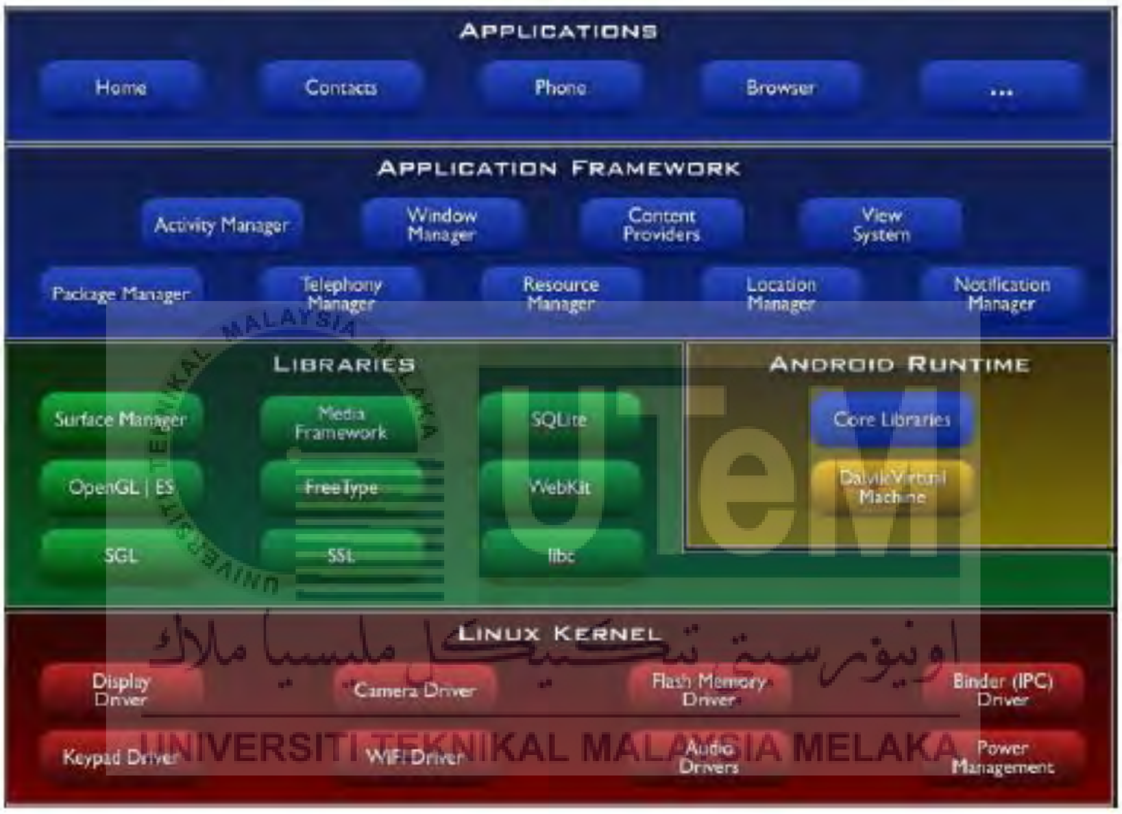


Figure 1: Android's architecture diagram

Furthermore, the software stack of Android OS is shown in Figure 1. According to the Android's architecture diagram, it involves five groups of software program such as Linux kernel, Android runtime environment, libraries, application framework and application layer. Linux kernel is the primary part of the Android OS that allows manufacturer to modify the OS following the needs of mobile devices. Besides, it is a layer that can be used for managing memory, scheduling process and networking. The software of this layer is written in C/C++ language. On the top of Linux kernel layer is the Android layer that comprises the built-in libraries and the Android runtime layer that comprises Dalvik as a process virtual machine and the core Java-compatible libraries. The following layer is Application Framework layer which is a high-level layer that enables developers to create a new Android application. Applications and widgets layer is the highest level of the Android system architecture. The application may contain of one or more activities and the activity classes will exhibit the interface screen. Due to the activity's life cycle is not depending to the process life cycle, the application still can program in the background even the process is completed. [14]

2.5 Microcontroller

Smart home controller is the main device of the system that can be used to receive and send signal to control the actuators. Microcontroller board is always used as the central controller for controlling the home electrical appliances. Microcontroller board is defined as a SoC that functions on an integrated circuit which contains all necessary circuitry for control purpose such as microprocessor, memory, RAM, clock generator and programmable input/output peripherals. Microcontrollers are used for embedded applications which are different from the microprocessors that are usually used for general applications consist of assorted discrete chips. Most of the microcontrollers embedded systems are advanced and only require small amount of memory and program length. Microcontroller has typical input and output devices to obtain the input signal and exhibit the output process respectively. For example, the sensor data such as the level of temperature, light and humidity can be obtained and gives signal to control the relays, switches, LED’s and solenoids in real world through the microcontroller. There are several types of microcontroller can be used in smart home automation system such as microprocessor development board, PLC, Arduino, Raspberry Pi, PICAXE and NodeMCU. [15]

Currently, NodeMCU is the most popular microcontroller to be used to control the smart home system via internet network through WiFi connection. It is a type of single-board microcontroller and using XTOS as operating system. The CPU includes firmware which runs on the ESP8266 Wi-Fi SoC that integrated with a Tensilica Xtensa LX106 core. It consists 128KB of memory and provides 4MB of storage. The power can be supplied to the board through USB. Furthermore, it is based on ESP8266 and integrates GPIO, PWM, IIC, 1-Wire and ADC all in one board. The features of NodeMCU are open source, inexpensive, IoT platform and simple programming environment. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. [16]

There are three types of NodeMCU development boards and their names are shown in Table 1. The common names for the first generation, second generation and third generation are V1, V2 and V3 respectively. Figure 2 shows the hardware of ESP8266-12 NodeMCU development board. According to the figure, the first generation of NodeMCU is yellow in colour and the size of the board is very wide which equals to 47mm x 31mm. It is not convenient to use as the board will covers 10 pins of a regular breadboard. It uses ESP8266-12 chips.

Table 1: Names of ESP8266 NodeMCU development boards

Generation	Version	"Common" Name
1st	0.9	V1
2nd	1.0	V2
3rd	1.0	V3

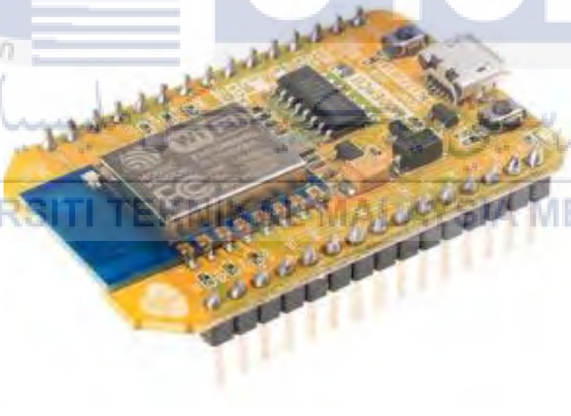


Figure 2: Hardware of ESP8266-12 NodeMCU development board

Figure 3 shows the hardware of ESP8266-12E NodeMCU development board. It is second generation of NodeMCU that fixed the weakness of first generation board. The board is narrower and can fit into breadboard easier. The chip is upgraded from ESP8266-12 to ESP8266-12E. There is no official third generation of NodeMCU is released. It is a version that invented by producer LoLin to minor improve the second generation of NodeMCU. For the third generation of NodeMCU, two pins will be reserved for a USB power output and an additional ground.

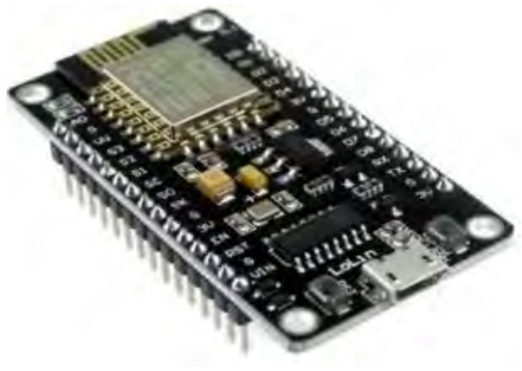


Figure 3: Hardware of ESP8266-12E NodeMCU development board

2.6 HTTP Request Method [17]

HTTP is stand of Hypertext Transfer Protocol that allows the communication between a client and server. It works as a request and response protocol. When a client sends a HTTP request to the server, the server will return a response to the client. There are two common methods that usually used for a request and response between a client and server which are GET method and POST method. GET method is a method that the data is requested from a specified resource whereas POST method is a method that data to be processed is submitted to a specified resource. There are other HTTP request methods is shown in Table 2.

Table 2: HTTP request methods

Method	Description
HEAD	Only HTTP header can be returned without any document body.
PUT	A representation of the specified URI can be uploaded.
DELETE	The specified resource can be deleted.
OPTIONS	The HTTP methods that the server support can be returned.
CONNECT	The request connection can be converted to a transparent TCP/IP tunnel.

2.7 Introduction to MQTT [18] [19]

MQTT is stand of Message Queuing Telemetry Transport protocol that is a simple and light weight messaging protocol which is used publish-subscribe messaging pattern through a message broker. The messages are distributed by the broker to the clients according to the topic of a message. The first version of MQTT is founded by Andy Stanford-Clark and Arlen Nipper of Cirrus Link Solutions in 1999. There are a few procedures in communication providence that are required to follow for setup the MQTT machine application over the network on both cloud and direct communications applications as shown in Table 3.

Table 3: Application development approaches in MQTT

Function	Description
MQTT Connect	The messages can be served when the client is established to the server.
MQTT Disconnect	After the communication is finished, the MQTT TCP/IP session is disconnected.
MQTT Subscribe	The scenario is subscribed after client sends the request to the server.
MQTT Unsubscribe	The scenario is unsubscribed after client sends the request to the server.
MQTT Publish	After sending the request to the MQTT client, client is returned immediately to application thread.

For the MQTT broker part, it is defined as the strategy of a device to publish information about the scenarios to a server while the information that have earlier subscribed to the client's theme is pushed out by the broker to the clients. MQTT broker, MQTT Broker Push and MQTT Broker Publish are the operation that used in application development. For the messaging services, a transport layer is used to transmit data from server to user and from machines devices to server by using push and pop operations. The function of the Server Push is used to send data to server for storing purpose whereas the function of the Server Pop is used to send data to IoT applications.

2.8 Previous Related Works

2.8.1 "Internet of Thing Based Home Appliances Control" [8]

By: Rakesh K. Deore, Vijay R. Sonawane and Pooja H. Satpute, Information Technology, SITRC, Nashik, India.

In this paper, Arduino board and Ethernet shield are utilized to build a web server to provide bidirectional communication between the user devices and home appliances. The server has the availability to store database properly with the assistance of administrator and the database can be retrieved by end user when necessary. By using IoT, the home appliances can be monitored and controlled remotely from a remote location utilizing a smartphone application. User is flexible to set time to switch on or off the lights, motor, garden automatic irrigation and gate. This is not only benefits people to live comfortable but also increases energy efficiency, conserves energy consumption, reduces utility cost, provides home security and safety.

2.8.2 "Automated Home Appliances Control Using Embedded Web Server" [20]

By: Abdul Ahad, Y. Mahesh, G. Sukanya, N. Harika, A. Uma Sri, Department of Computer Science Engineering, Acharya Nagarjuna University, Guntur, A.P., India.

In this paper, an embedded web server is created to control the home appliances through internet by using web browser in a personal computer that connected to the same internet network as shown in Figure 4. User can monitor the condition of the system in real time and control remotely the home appliances from long distance. In this project, Arduino Ethernet shield is connected to router and embedded with Arduino UNO to act as a web server. IP address and Port number are typed in the web browser and a HTML based web page will display on the screen. The request signal will send from the client side and transmit the signal to Arduino UNO. Then, Arduino UNO will process the signal to switching on-off the home appliances.

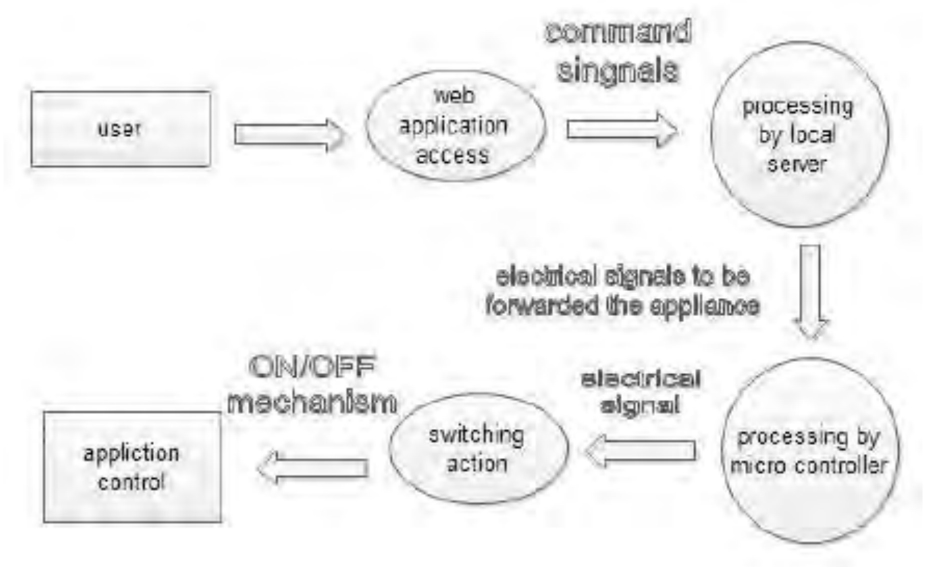


Figure 4: Smart home system using Ethernet shield

2.8.3 “Mobile based Home Automation using IoT” [21]

By: Kumar Mandula, Ramu Parupalli, CH.A.S.Murty, E.Magesh, Rutul Lunagariya, Centre for Development of Advanced Computing(C-DAC), A Scientific Society under Ministry of Communications and Information Technology, Government of India.

In this paper, an IoT concept of home automation is implemented with a low-cost Arduino microcontroller board and an Android mobile phone. To implement this home automation system, two different types of communication technologies are used. Figure 5 shows the architecture of home automation using Bluetooth. In this architecture, the mobile phone which is a wireless communication device is established to Arduino board using Bluetooth. Android studio (Version 1.5) which is a popular Android SDK is used for creating an Android based mobile application and provides complete development environment for compilation, verification, debugging and packaging.

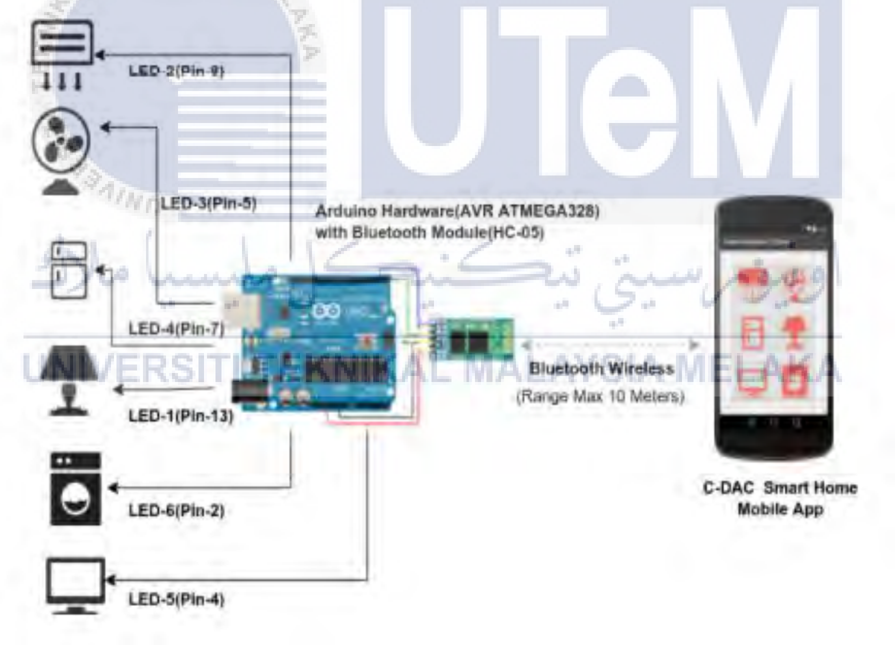


Figure 5: Architecture for home automation using Bluetooth

Figure 6 shows the architecture for home automation using Ethernet. In this architecture, internet is used as second communication technology in the home automation system. Ethernet shield can be embedded with Arduino board which is connected to RJ-45 for internet connectivity. User can enter IP address and Port number of the Ethernet shield in an Android mobile application to control home appliances from a located remote device in a smart home environment.

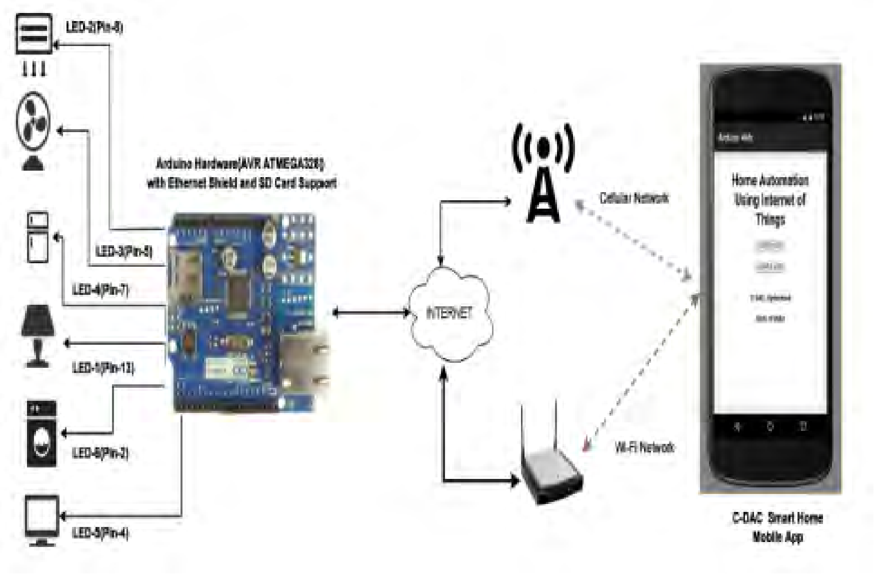

LED-2(Pin-8)
Arduino Hardware(AVR ATMEGA328)
with Ethernet Shield and SD Card Support
LED-3(Pin-5)
LED-4(Pin-7)
LED-1(Pin-13)
LED-6(Pin-2)
LED-5(Pin-4)
INTERNET
Cellular Network
Wi-Fi Network
Home Automation Using Internet of Things
C-DAC Smart Home Mobile App

Figure 6: Architecture for home automation using Ethernet

2.8.4 Summary and Discussion of the Review

From the previous related work, Ethernet shield are the most popular communication technology used for connecting the internet network to the Arduino board. It overcomes the range problem that faced by Bluetooth technology. User can connect to the smart home system from around the world. The Ethernet shield can serve as a web server and the data can be stored in the SD card on the board. The IoT such as Android application and web application can be used for controlling and monitoring home appliances by using the IP address and Port number that provided by Ethernet shield. However, the cost for Ethernet shield and Arduino are expensive. Therefore, ESP8266-12E NodeMCU development board which consists WiFi chip can be used to access to the WiFi connection and control the electrical appliances through the relay module.

CHAPTER 3

METHODOLOGY

3.1 Overview

This chapter is a methodology chapter that discusses the procedures to do the proposed project. The overall system of the proposed smart home system is designed and explained clearly step by step. The overview of the hardware and software for the proposed system included the working principle and specification of every hardware and software. The procedures for the Anto Server setup, the interfacing of NodeMCU board with the Anto server, PushingBox setup are also done and explained. The user interface of application is designed using MIT App Inventor 2 and the Blocks are arranged for building the behaviour of application.

3.2 Project Methodology

Project methodology is a procedure guideline for implementing this project to achieve the set goals. It is important to make sure the project can flow faster and can be completed on time. Figure 7 shows the flow chart of project methodology.

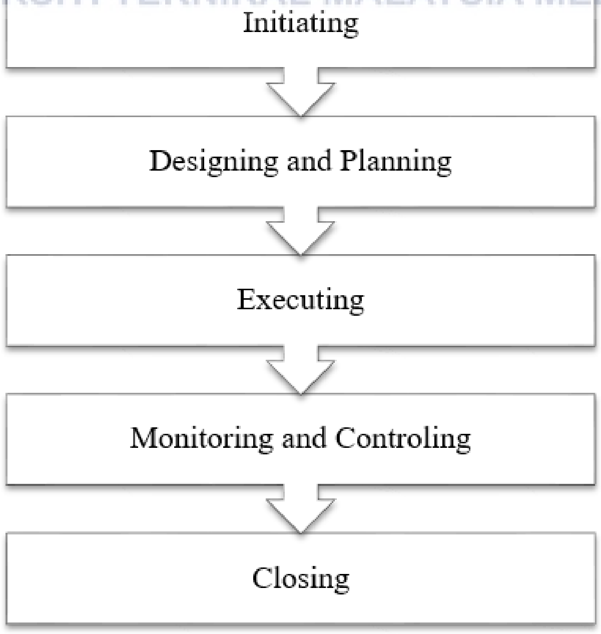


Figure 7: Flow chart of the project methodology

According to the flow chart, there are five phases of methodology including initiating, designing and planning, executing, monitoring and controlling and closing. The first phase is initiating. In this phase, the research about the different methods of smart home system will be studied. The research background, motivation, problem statement, objectives and scope of the proposed project are identified. The second phase is design and planning. In this phase, a smart home system will be designed according the specification and requirement of the proposed project. Furthermore, low-cost hardware and open source software are selected. The graphical user interface of Android application will be designed. The third phase is executing. In this phase, a web server is setup. Then, the programming code is designed and uploaded to the controller for the interfacing of controller with the web server. Moreover, the services and the scenarios for notification purpose is setup. An Android application is designed and created using the selected software development kits. The interfacing of the web server and notification with the application will be done. Then, the designed Android application will be installed on an Android based smartphone. Additionally, the hardware circuit for the system will be built on the breadboard and established with the software. The fourth phase is monitoring and controlling. In this phase, the system will be tested. The Android application will be used for controlling the home appliances. The system will be modified if there is any error occurred. The last phase is closing. In this phase, the proposed project will be fully completed. A prototype will be built for the demonstration.

3.3 Overall System Design

This proposed project is designed to control the smart home electrical appliances via Android smartphone. The advancement technology of using IoT will be presented in this system. An Android application will be created and designed for remotely controlling the electrical appliances. By connecting to the internet network, user can use their smartphone to remotely control the electrical appliances in indoor and outdoor environment. Furthermore, the smart home control system has two different ways to control the home electrical appliances which are manually click the button on UI of the application and select the starting time or ending time on UI of the application. Figure 8 shows the block diagram of the smart home system for workflow illustration and the relationship between the components that will be accomplished by the system. The system involves hardware and software including central controller, relay, router, web server, cloud and Android application.

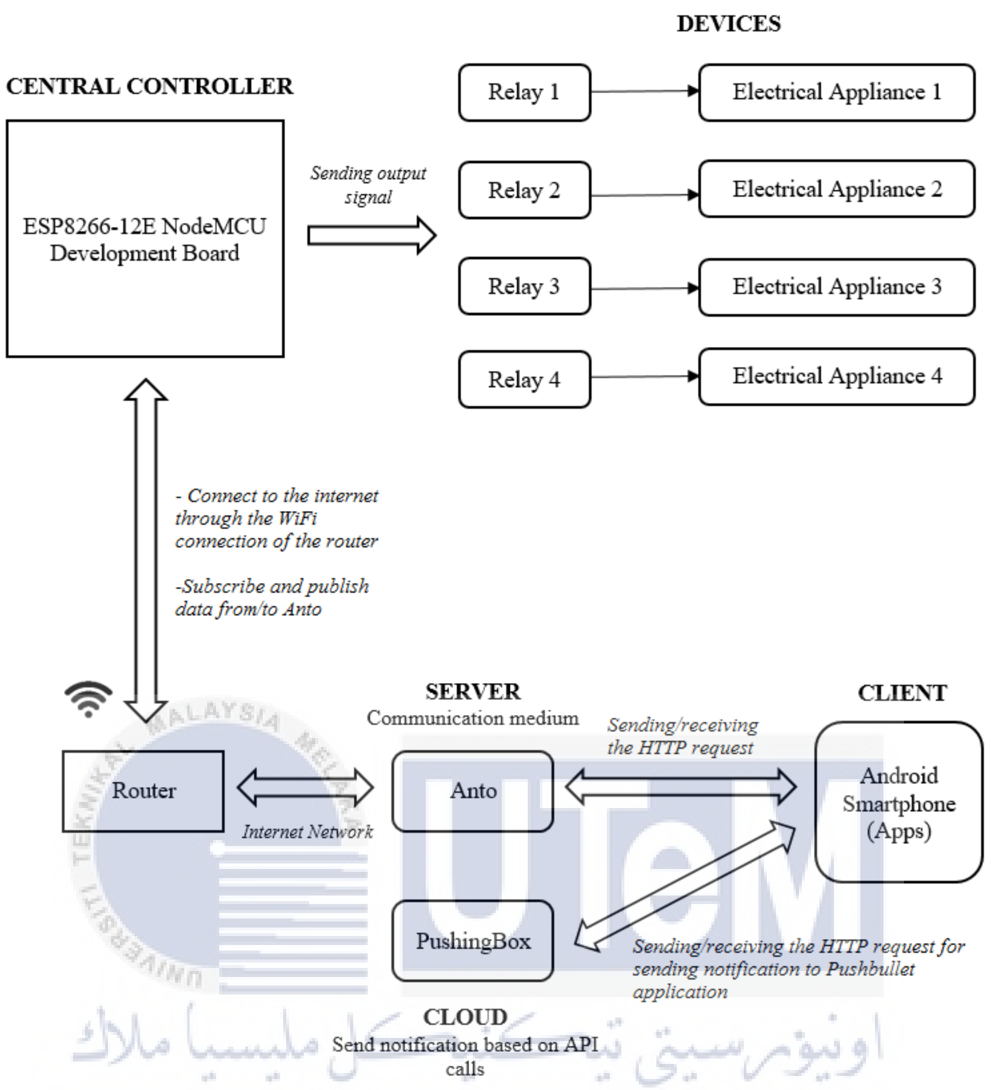

DEVICES
CENTRAL CONTROLLER
ESP8266-12E NodeMCU Development Board
Sending output signal
Relay 1
Relay 2
Relay 3
Relay 4
Electrical Appliance 1
Electrical Appliance 2
Electrical Appliance 3
Electrical Appliance 4
- Connect to the internet through the WiFi connection of the router
-Subscribe and publish data from/to Anto
SERVER
Communication medium
CLIENT
Sending/receiving the HTTP request
Router
Internet Network
Anto
Android Smartphone (Apps)
PushingBox
Sending/receiving the HTTP request for sending notification to Pushbullet application
CLOUD
Send notification based on API calls

Figure 8: Block diagram of smart home system

According to the system block diagram, the internet network plays the important role to receive and send the request between the central controller, web server, cloud and mobile application. In this project, ESP8266-12E NodeMCU Development Board is used as the central controller. It is a microcontroller that based on ESP8266 which built-in 802.11 b/g WiFi. Therefore, the board can be accessed to local internet network through WiFi connection so that it can communicate with the Anto real-time server. Anto is the communication medium between the central controller and the mobile application. With the internet connection, both NodeMCU board and the Anto server can be the subscriber and publisher. The board can subscribe the data from Anto server's channel that received from mobile application and then send the output signal to energize the contact of the relay module for controlling the electrical appliances. The data is then can publish to the channel of Anto server from NodeMCU board and then transport the data to the mobile application through HTTP request by using GET method. To make sure the mobile application can function well, the smartphone must access to the internet network or mobile data. Thus, the user can control the electrical appliances through the buttons on the UI of the application. For this system, user will receive the notification through Pushbullet application when the button is triggered. PushingBox as a cloud is used to setup services and scenarios for providing the permission to send the notifications using HTTP request to the smartphone.

3.4 Hardware Overview of the System

This section shows the selection of the low-cost hardware for the proposed project. The specification of the hardware will be discussed. The system consists of two units: the smartphone and the microcontroller with relay module. The ESP8266-12E NodeMCU development board is used as the microcontroller board. The smartphone is used as a controller to send instructions and as a recipient to receive the responses. The ESP8266-12E chip of the NodeMCU board is responsible for communication between the microcontroller unit and the smartphone. It is important to have some basic knowledges about the working principle behind the hardware before using it.

3.4.1 ESP8266-12E NodeMCU Development Kit V1.0

In this project, ESP8266-12E NodeMCU Development Board is used as the central controller of the system. Figure 9 shows the hardware of an ESP8266-12E NodeMCU Development Board. It is second generation of ESP8266 NodeMCU development board that fixed the weakness of first generation board. The chip is upgraded from ESP-12 to ESP-12E. The power can be supplied to the board through USB. The board is based on the ESP8266 which integrates GPIO, PWM, IIC, 1-Wire and ADC all in one board.

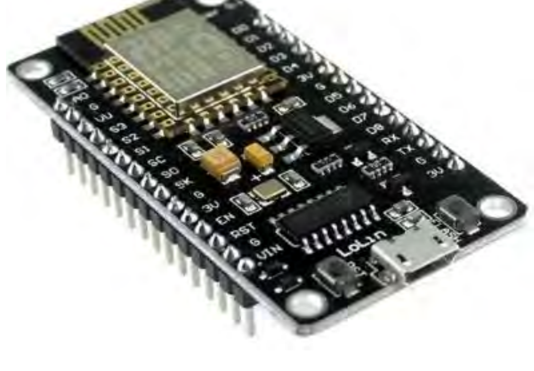


Figure 9: ESP8266-12E NodeMCU development kit V1.0

Table 4 shows the GPIO pin maps for the I/O index to ESP8266 pin mapping. For the connection with the actuators, all access is based on the I/O index number on the NodeMCU board, not refer to the internal GPIO pin. For the programming code, all access is based on the internal GPIO pin. For example, the D0 pin on the NodeMCU board is mapped to the internal GPIO pin 16. Therefore, if the actuator is connected to the D0 pin of the board, the GPIO pin value is 16. [22]

Table 4: GPIO pin maps for the I/O index to ESP8266 pin mapping

I/O Index	ESP8266 Pin
D0	GPIO 16
D1	GPIO 5
D2	GPIO 4
D3	GPIO 0
D4	GPIO 2
D5	GPIO 14
D6	GPIO 12
D7	GPIO 13
D8	GPIO 15
D9	GPIO 3
D10	GPIO 1
D11	GPIO 9
D12	GPIO 10

3.4.2 Relay Module

In this project, 4-channels 5V relay module is used with the NodeMCU board. It plays the important role to control the electrical appliances. Figure 10 shows the hardware of a 4-channels relay module. The power rate of each relay is 7A at 28BDC or 10A at 125VAC. The relay module allows the NodeMCU board to control the electrical appliances that use higher current and voltage. For the relay module inputs part, relay module can be powered by connecting the power supply to the “Pin VCC” and ground to the “Pin GND”. The GPIO pins from NodeMCU board can energize the IN1, IN2, IN3 and IN4 inputs with low input power. For the relay module outputs part, each of the relay consists three terminals such as “common (COM)”, “normally open (NO)” and “normally closed (NC)”. The dry contact outputs can be used to control the electrical appliances, motors, lights and more. [23]

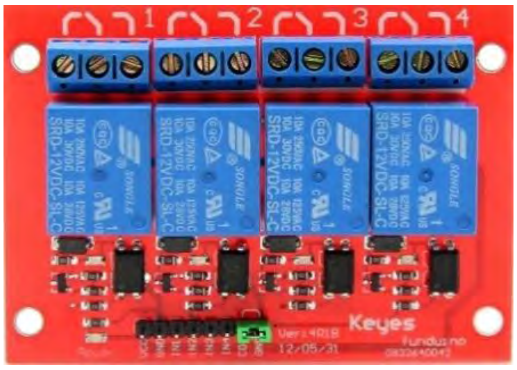


Figure 10: 4-channels 5V relay module

3.5 Software Overview of the System

This section shows the selection of the open source software for NodeMCU programming, web server setup, cloud setup and UI design of application. The Arduino IDE 1.6.9 is used to write coding to program the NodeMCU board. Anto is used to setup a web server whereas PushingBox as a cloud is used to setup the services and scenarios for notification purpose. Furthermore, MIT App Inventor 2 is used to design the UI of the application and build the block to design the behaviour of the application.

3.5.1 Arduino IDE 1.6.9

In this project, Arduino IDE 1.6.9 is used to write the coding and upload the codes to NodeMCU board through USB. Figure 11 shows the integrated development environment of Arduino version 1.6.9.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 11: Arduino IDE 1.6.9

The Arduino IDE is software that comprises a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It can be connected and uploaded the programs to the microcontroller hardware through USB cable. The sketches are the programs that written in the text editor of Arduino IDE and can be saved with the file extension in .ino form. The message area is an environment that will give feedback and display the errors while saving and exporting. The text output that included complete error messages and other information of the Arduino IDE will display at the console area. The configured board and serial port are displayed at the bottom of right hand corner of the window. The toolbar buttons are used to verify and upload programs, create, open, and save sketches, and open the serial monitor. The open-source Arduino IDE can run on Windows, Mac OS X and Linux. The environment is written in Java and is compatible many open-source software. [23]

3.5.2 MIT App Inventor 2

In this project, MIT App Inventor 2 is used to create and build an application for Android smartphone. MIT App Inventor 2 is an open-source web application provided by Google and now developed by MIT Media Lab, MIT Computer Science and Artificial Intelligence Lab. The developer can create software applications for the Android OS without any knowledge in the computer programming. The application can be built by working with App Inventor Designer and App Inventor Blocks editors.

Developer can use the App Inventor Designer to design the UI of application by drag-and-drop both on-screen and off-screen components. Figure 12 shows the Designer editor of MIT App Inventor 2.

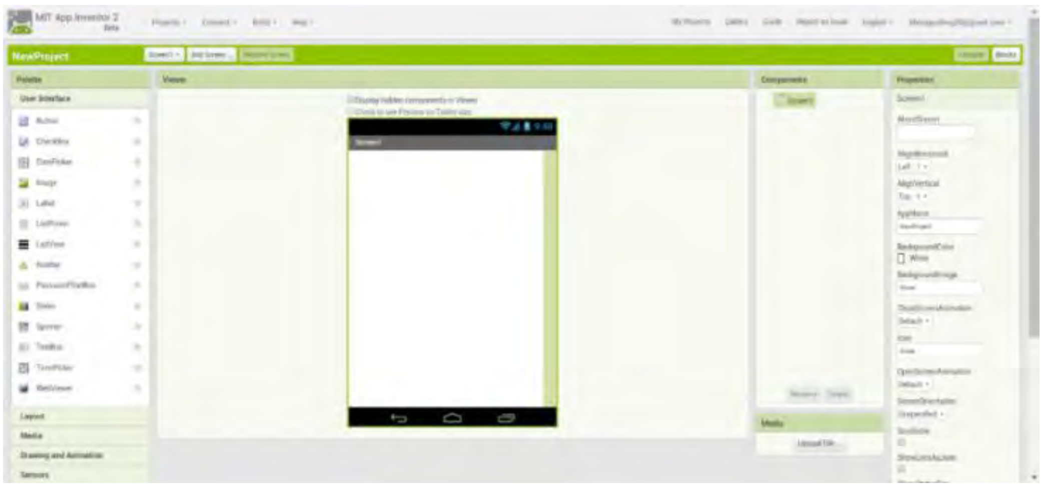


Figure 12: Designer editor of MIT App Inventor 2

From the figure, the Designer button beside the Blocks button can be clicked from any tab to go to the Designer Tab. Furthermore, there is a list of UI components in the Palette. The components can be dragged to the Designer viewer so that the component can be added to the application. The Properties enables developer to select a component in the component list to change its properties such as colour, size and behaviour.

Additionally, developer can use App Inventor Blocks editor to design the application behaviour. Figure 13 shows the Blocks editor of MIT App Inventor 2.

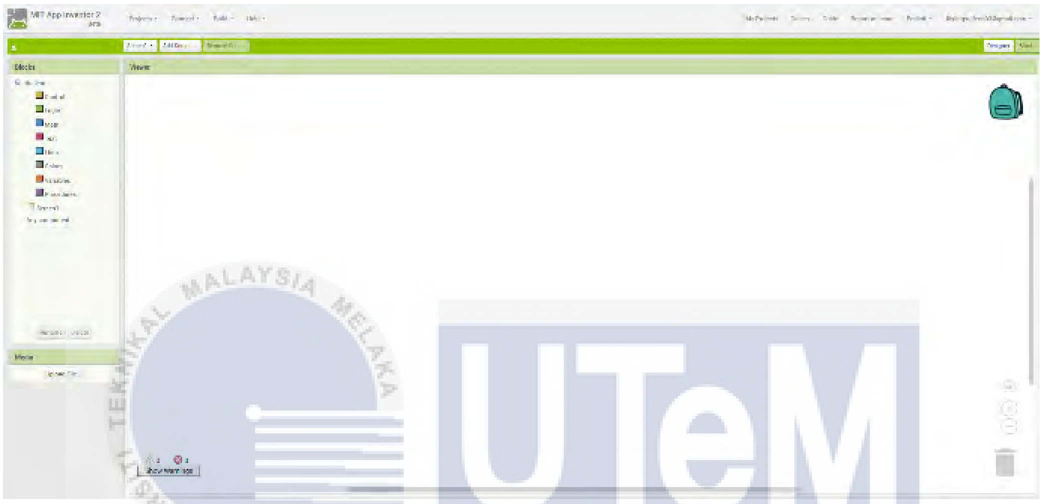


Figure 13: Blocks editor of MIT App Inventor 2

From the figure, the Blocks button beside the Designer button can be clicked from any tab to go the Blocks Tab. Moreover, developer can find the Blocks for general behaviours that wanted to add to the application and drag the Blocks to the Block viewer. Every single component will show their own component specific drawers. Developer can find Blocks for behaviours for specific Components and drag them from the Drawers to the Blocks viewer to build relationships and behaviour. The Block that had been dragged to the Blocks viewer can snap together to set application behaviour. [24]

3.5.3 Anto Real-Time Server

In this project, Anto is used as a communication medium between things on the internet. A server is set up for free. Therefore, it enables a user to operate NodeMCU board over the internet using Android smartphone. The system supports HTTP, HTTPS, MQTT, MQTTS and Websocket communication. There is a library is created for programming and APIs is created for application development.

3.5.4 PushingBox Cloud

In this project, PushingBox is used to launch the notification function in the Android smartphone. It is as a cloud and it covers a large list of services including emails, Twitter, Karotz, Prowl, Pushalot, Toasty, Notify My Android, Newtifry, Pushbullet and CustomeURL. Several scenarios will be created to set the specific notification messages for the different conditions. When the button on the UI of mobile application is triggered, the API of PushingBox will be called using GET method. Then, the notifications will be sent through the selected services.

3.6 Setup for the Anto Real-Time Server

Anto real-time server is needed to setup first before starting anything for this proposed project. There are several procedures that is needed to follow for the Anto server setup. Firstly, an Anto account is signed up. The username and password are obtained for login requirement. For this project, NodeMCU is used as the device that will be connected to Anto. NodeMCU will be used to control four types of electrical appliances such as alarm, air-conditioner, lighting and fan. Therefore, one Thing and several Channels are created for the controlling purpose. Table 5 shows the information of the Key, Thing and Channel that are created in Anto.

Table 5: The Key, Thing and Channel information of the Anto

Anto	Details
Key	Key: grPFnLVq7bQlX4tR61Vl2PxDOP8web1A7UVX8CQo Description: SmartHomeControlKey
Thing	Name: Smart_Home_Control Description: Control the switches over internet Hardware: NodeMCU 1.0 (ESP-12E Module)
Channel	Name: SW1 Description: Control switch no 1 Type: Switch on/off
	Name: SW2 Description: Control switch no 2 Type: Switch on/off
	Name: SW3

	Description: Control switch no 3 Type: Switch on/off
	Name: SW4 Description: Control switch no 4 Type: Switch on/off
	Name: SW1_status Description: The status of switch no 1 Type: Switch on/off
	Name: SW2_status Description: The status of switch no 2 Type: Switch on/off
	Name: SW3_status Description: The status of switch no 3 Type: Switch on/off
	Name: SW4_status Description: The status of switch no 4 Type: Switch on/off
	Name: ALL Description: Control all switches Type: Switch on/off

A new Thing is created and named as “Smart_Home_Control”. It is used to connect the NodeMCU with the Anto. Additionally, nine new Channels are created and named as SW1, SW2, SW3 and SW4, SW1_status, SW2_status, SW3_status, SW4_status and ALL respectively. The type of all the Channels are set to Switch on/off type. The current value of the Channels such as “1” and “0” will be subscribed by NodeMCU to control the electrical appliances. For the callback function, the value also can be published from NodeMCU to MQTT broker and updated the current value of the Channels. A new Key is also created and “grPFnLVq7bQlX4tR61Vl2PxDOP8web1A7UVX8CQo” is obtained. The Key is the key of permission for Channels to be read or updated and it will be generated on the control panel of Anto. Both “Read” and “Update” of all Channels are ticked. Therefore, NodeMCU can subscribe all the data from the Channels and the data are also can be published to MQTT broker and updated in the Channels.

3.7 Interfacing of NodeMCU Board with the Anto Server

To interface the NodeMCU board with the Anto server, a complete programming code need to be written and uploaded to the board through USB cable. For this project, Arduino IDE 1.6.9 is used for writing the coding. To compatible the NodeMCU with the Anto server, Anto Library is downloaded to the Arduino IDE and pulled to use by using the command of "#include <AntoIO.h>". ESP8266 Arduino Library is also used to compatible the NodeMCU with the ESP8266-12E chip that integrated with the NodeMCU board by using the command of "#include <ESP8266WiFi.h>". Anto Library and ESP8266 Arduino Library will be called out and the methods that stored in the library can be used in Arduino IDE. Table 6 shows the function of MQTT in Arduino programming. All the function is described clearly with the examples.

Table 6: The function of MQTT in Arduino programming

Function	Description
Create AntoIO object	AntoIO anto (user, key, thing); The Username, Key permission and Thing name are defined. This function is used to create the AntoIO object named anto by using constructor AntoIO (user, key, thing).
Connect wifi	Anto.begin (ssid, pwd); The SSID and password of the WiFi that will be connected to the board are defined. This function is used to connect to internet through the WiFi connection.
Callback MQTT function	anto.mqtt.onConnected(connectedCB); If the board is connected to the MQTT server, this function will be invoked. Then, the sub channel setting can be applied.
	anto.mqtt.onDisconnected(disconnectedCB);

	If the board is disconnected from the MQTT server, this function will be invoked. Then, the connection to the MQTT server is dropped.
	anto.mqtt.onData(dataCB); When the board received the information from the Anto server, this function will be invoked. Then, the information can be read and used.
	anto.mqtt.onPublished(publishedCB); When the information from the board is sent to the Anto server, this function will be invoked. Then, the information can be used to shake the value up.
Subscribe and Publish	Anto.sub ("channel name"); This function is used to subscribe the value in the channel from Anto server to the board. The callback will be send to the dataCB() function when the value in the channel is changed.
	Anto.pub ("channel name", GPIO value); This function is used to publish the desired value from the board to the channel. The callback will be sent to the PublishedCB() function when the value is changed successfully.
Connect mqtt	anto.mqtt.connect (); This function is sued to connect the board to the Anto's MQTT server.

3.8 Setup for the PushingBox Cloud

In this project, PushingBox as a cloud is used to launch the notification function in the Android smartphone based on API calls. For the PushingBox setup, the website is logged in using user's Google account. Then, a service is added. There is a large selection list of services in My Services page including emails, Twitter, Karotz, Prowl, Pushalot, Toasty, Notify My Android, Newtifry, Pushbullet and CustomeURL. For this project, Pushbullet is selected as the service for Android phone notification purpose. Table 7 shows the information of service that created in PushingBox. When Pushbullet service is added, a name of "Pushbullet Lily" is entered and an API key will be obtained.

Table 7: Service that created in PushingBox cloud

Service Name	API key
Pushbullet Lily	o.vWAC2coG1mvVSwv8V8a6HKTwWRjHZYK

Then, My Scenarios page is entered for creating new scenario and adding an action to the scenario. Table 8 shows the virtual scenarios list that created in PushingBox. After a scenario is added, a DeviceID will be obtained. According to the table, 10 scenarios are created, therefore 10 DeviceID are obtained. Pushbullet Lily service is added as the action to all scenarios. Furthermore, the specific notification messages are also set to all scenarios.

Table 8: Virtual scenarios list that created in PushingBox cloud

Scenario Name	DeviceID
AllSwitch_OFF	v926246B8C369473
AllSwitch_ON	v94BF87EE3A165B8
Switch1_OFF	vD5FFA24223C46F7
Switch1_ON	v21FFE94118564EB
Switch2_OFF	v9FFF1519A28B489
Switch2_ON	v6F87D1EA9C22D96
Switch3_OFF	vC3E625862BB2414
Switch3_ON	v4F961DAD290F5A0
Switch4_OFF	v99FC54015F911AA
Switch4_ON	v4D27734FA0DB804

3.9 Android Application Graphical User Interface (GUI) Design

For this project, an Android mobile application which known as Smart Home Control (SHC) is designed and created using MIT App Inventor 2. The UI design and the behaviour of the application can be created using App Inventor Designer and Blocks editor. The function of application is used to remotely access and control the home electrical appliances through internet network. When the smartphone is connected to the internet through a WiFi network or mobile data, the application will be accessed to the Anto server and communicated with the channels created through HTTP API request. For the real-time connection, Port 2083 must be enabled. In this application, there are two main screens including LOGIN page and CONTROL PANEL page and three sub-screens including ON/OFF page, TIMER SETTING page and NAME OF SWITCH page. The following describes the UI design of every pages in the SHC application.

Figure 14 shows the UI design of LOGIN page. LOGIN page is created for user's security and identification purpose. Based on the figure, a username textbox and a password textbox will be displayed on the screen of LOGIN page. The custom username and password will be set for the user.

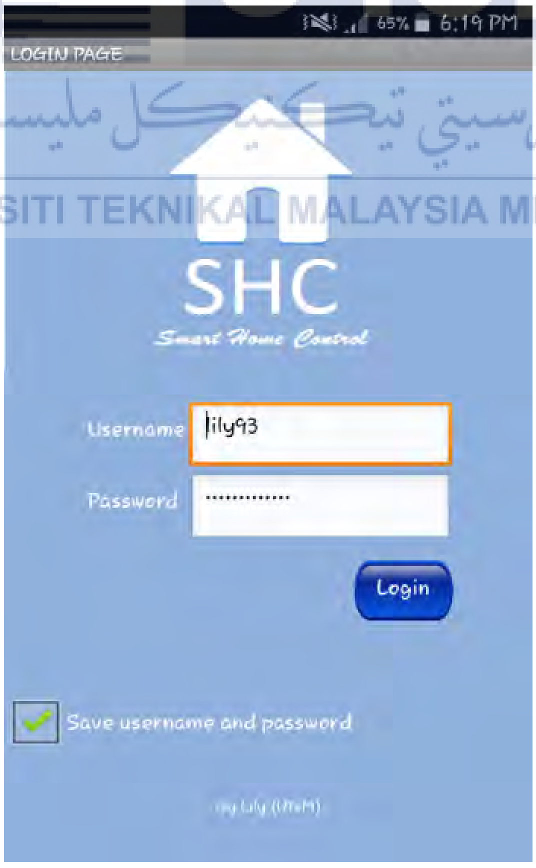


Figure 14: UI design of the LOGIN page

Figure 15 shows the flow chart of LOGIN page. According to the flow chart, when SHC icon is pressed, the application will be started and enters to the LOGIN page. The correct username and password are required for user to enter CONTROL PANEL page. If either one of the username and password is incorrect, a sentence of “Your username/password are invalid!” will be displayed on the screen of LOGIN page. The loop continues until the correct username and password are entered.

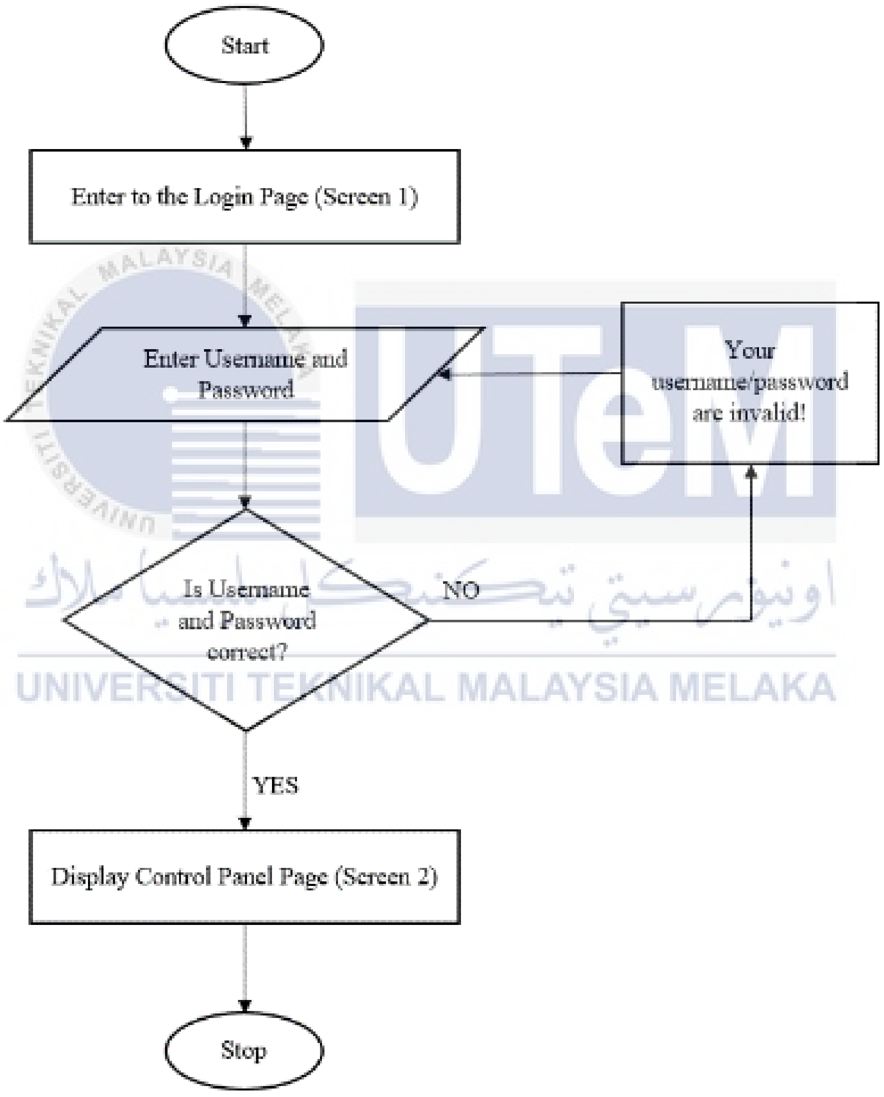

Start
Enter to the Login Page (Screen 1)
Enter Username and Password
Your username/password are invalid!
Is Username and Password correct?
NO
YES
Display Control Panel Page (Screen 2)
Stop

Figure 15: Flow chart of the LOGIN page

Figure 16 shows the UI design of CONTROL PANEL Page. Based on the figure, three optional functionalities of buttons are displayed on CONTROL PANEL page including ON/OFF button, SET TIMER button and NAME OF SWITCH button.

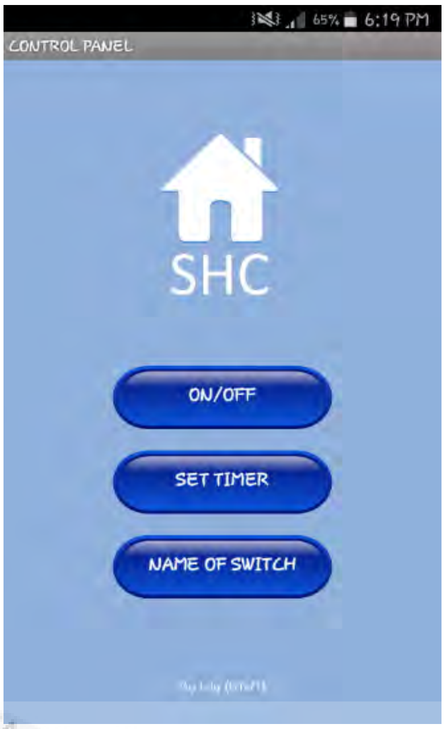


Figure 16: UI design of CONTROL PANEL page

Figure 17 shows the flow chart of CONTROL PANEL page. According to the flow chart, the screen will enter to ON/OFF page when ON/OFF button is pressed, enter to TIMER SETTING page when SET TIMER button is pressed and will enter to NAME OF SWITCH page when NAME OF SWITCH button is pressed.

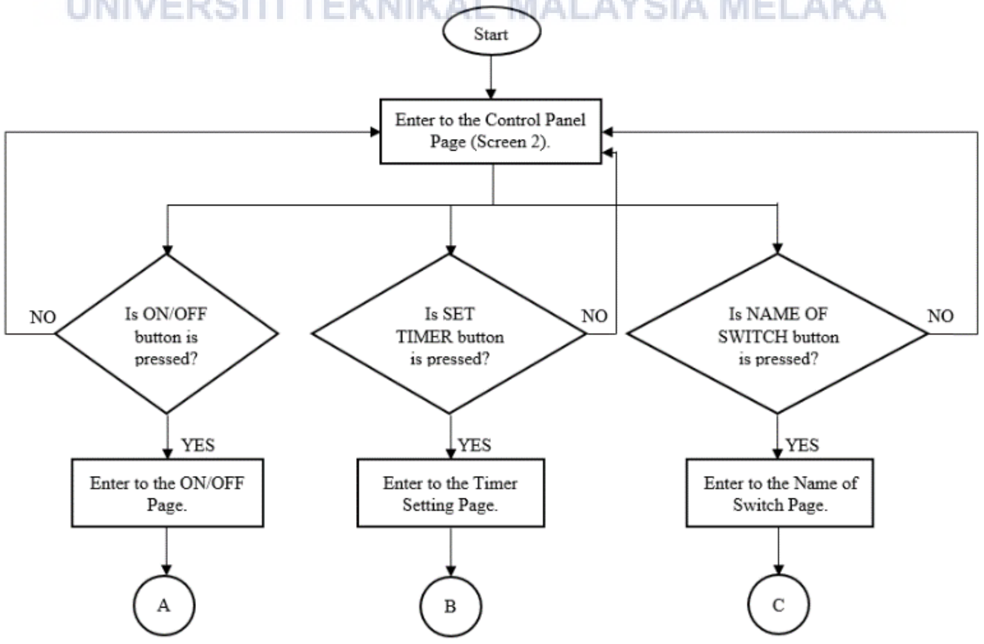

Start
Enter to the Control Panel Page (Screen 2).
Is ON/OFF button is pressed?
NO
YES
Enter to the ON/OFF Page.
A
Is SET TIMER button is pressed?
NO
YES
Enter to the Timer Setting Page.
B
Is NAME OF SWITCH button is pressed?
NO
YES
Enter to the Name of Switch Page.
C

Figure 17: Flow chart of CONTROL PANEL page

Figure 18 shows the UI design of ON/OFF page. Based on the figure, there is an upper space of the screen shows the connection status of the SHC application with Anto real-time server. When the smartphone is connected to the WiFi network or mobile data, the application will be accessed to Anto automatically. When SHC application is accessed with Anto, the word of "Anto not found" in red colour will be changed to the word of "Anto connected" in green colour. If not, the word will be remained with the word of "Anto not found" in red colour. Additionally, five green buttons are displayed on the screen of ON/OFF page which are four buttons to control each of the electrical appliance and one button to control all the electrical appliances at once.

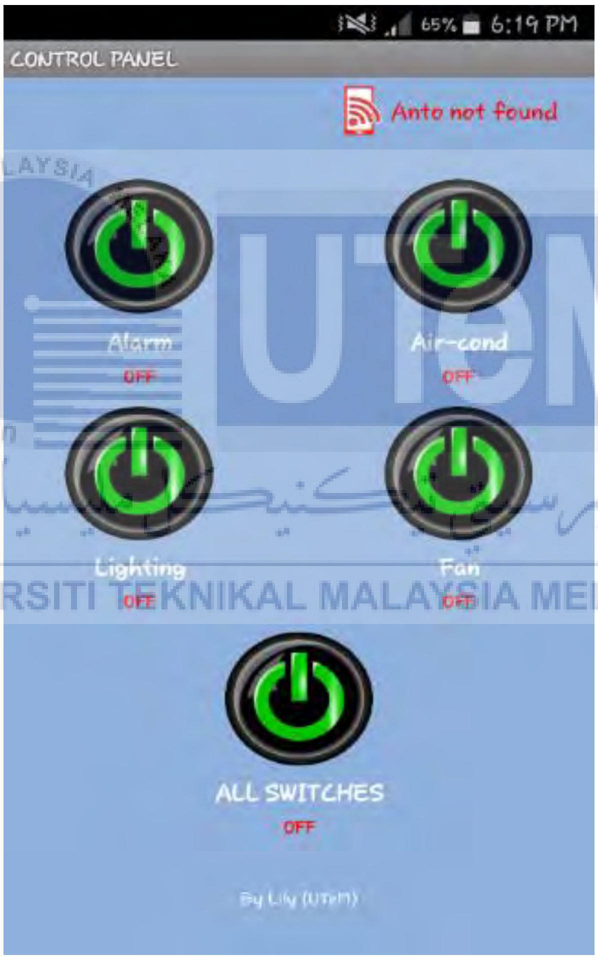


Figure 18: UI design of ON/OFF page

Figure 19 shows the flow chart of ON/OFF page. According to the flow chart, when the green button is pressed, the application will be accessed to the selected channel of Anto and the current value of the selected channel will be updated to "1" through HTTP API request. Then, the "1" current value of the selected channel will be subscribed by the

NodeMCU board through MQTT broker to change the state of the GPIO pin from LOW to HIGH and the electrical appliance will be turned on. When the electrical appliance is turned on, NodeMCU board will publish the current value of "1" to the selected channel. Through GET method, the application will receive the updated current value from the selected channel, then the green button will change to the red button and the button status will display "ON" in green colour.

When the red button is pressed, the application is accessed to the selected channel and the current value of the channel will be returned to "0" through HTTP API request. Then, the "0" current value of the selected channel will be subscribed by the NodeMCU board to change the state of the GPIO pin from HIGH to LOW and the electrical appliance will be turned off. When the electrical appliance is turned off, NodeMCU board will publish the current value of "0" to the selected channel. Through GET method, the application will receive the updated current value of the selected channel, then the red button will return to the green button and the button status will display "OFF" in red colour. The functionalities of all buttons are same.

Moreover, the fifth green button on the bottom of the screen is a button that can be used to control all electrical appliances at once. When the fifth green button is pressed, the application is accessed to all channels and the current value of the selected channels will be updated to "1" through HTTP API request. Then, the "1" current value of selected channels will be subscribed by the NodeMCU board through MQTT broker to change the state of the GPIO pin from LOW to HIGH and all electrical appliances will be turned on together. When the electrical appliances are turned on, NodeMCU board will publish the current value of "1" to the selected channels. Through GET method, the application will receive the updated current value from the selected channels, then all green button will change to the red button and all button status will display "ON" in green colour.

When the fifth button is long pressed, the application is accessed to the selected channels and the current value of the selected channels will be returned to "0" through HTTP API request. Then, the "0" current value of the selected channels will be subscribed by the NodeMCU board to change the state of the GPIO pin from HIGH to LOW and all electrical appliances will be turned off together. When all electrical appliances are turned off, NodeMCU board will publish the current value of "0" to the selected channels. Through GET method, the application will receive the updated current value of the selected channels, then the red button will return to the green button and the button status will display "OFF" in red colour.

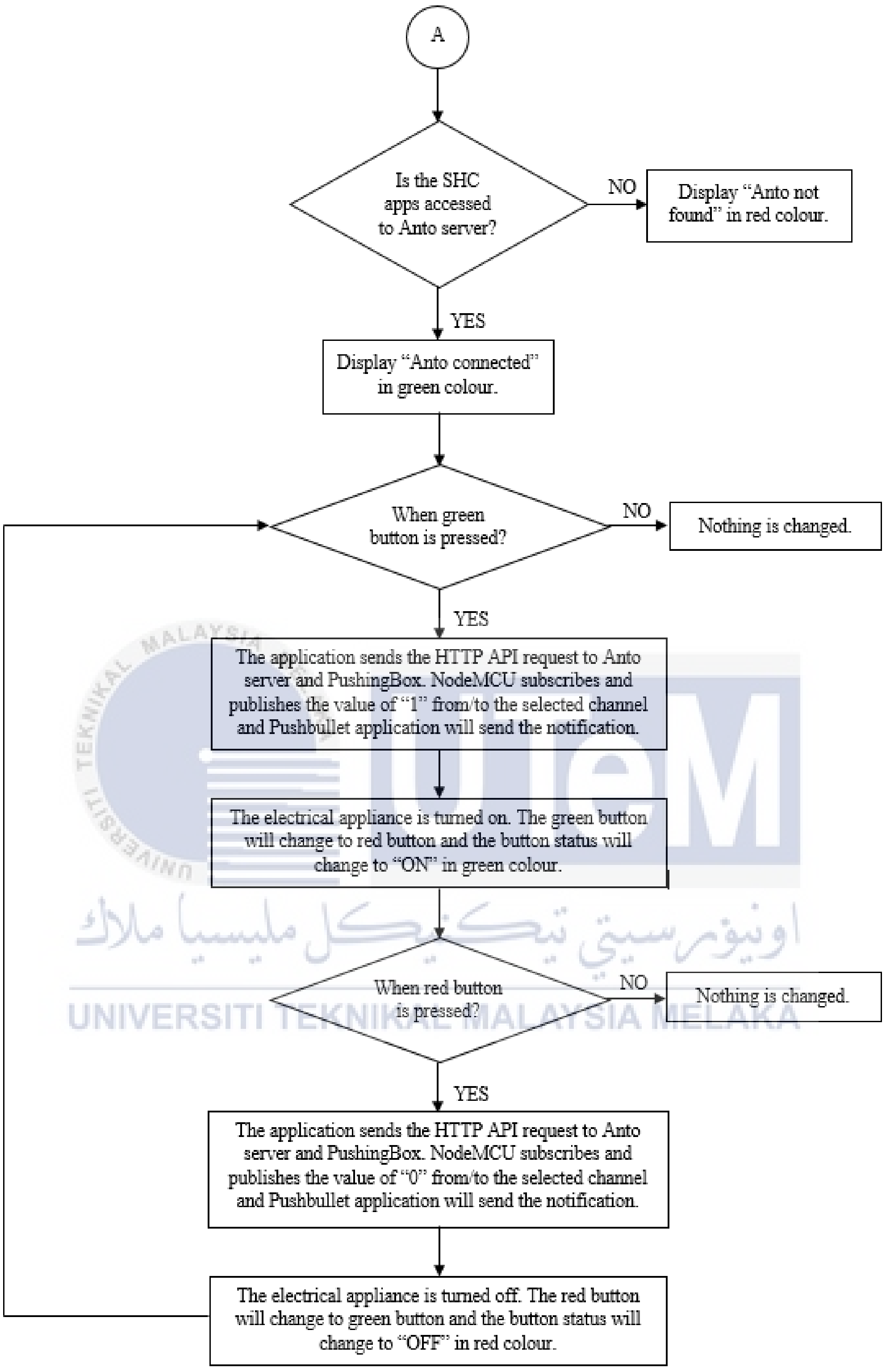

A
Is the SHC apps accessed to Anto server?
NO
Display "Anto not found" in red colour.
YES
Display "Anto connected" in green colour.
When green button is pressed?
NO
Nothing is changed.
YES
The application sends the HTTP API request to Anto server and PushingBox. NodeMCU subscribes and publishes the value of "1" from/to the selected channel and Pushbullet application will send the notification.
The electrical appliance is turned on. The green button will change to red button and the button status will change to "ON" in green colour.
When red button is pressed?
NO
Nothing is changed.
YES
The application sends the HTTP API request to Anto server and PushingBox. NodeMCU subscribes and publishes the value of "0" from/to the selected channel and Pushbullet application will send the notification.
The electrical appliance is turned off. The red button will change to green button and the button status will change to "OFF" in red colour.

Figure 19: Flow chart of ON/OFF page

Figure 20 shows the UI design of TIMER SETTING page. Based on the figure, two types of timer are displayed on the screen including starting time and ending time. User can select the starting time and ending time of the switches through the time pickers.

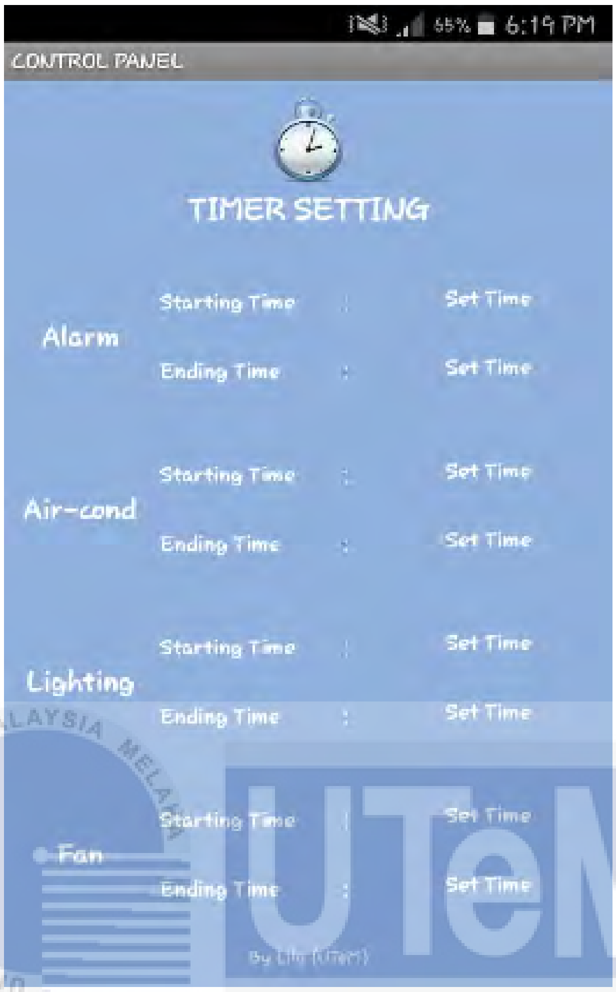


Figure 20: UI design of TIMER SETTING page

Figure 21 shows the flow chart of TIMER SETTING page. According to the flow chart, if the starting time is selected, the electrical appliance will be turned on when the starting time is reached. When the starting time is reached, the selected channel of Anto will be accessed with the application. Through HTTP API request, the current value of the channel will be updated to “1”. Then, the NodeMCU board will subscribe the “1” current value of the selected channel through MQTT broker and the state of the GPIO pin will be changed from LOW to HIGH to turn the electrical appliance on. When the electrical appliance is turned on, NodeMCU board will publish the current value of “1” to the channel. Through GET method, the application will receive the updated current value of the selected channel, then the green button of the ON/OFF page will change to the red button and the button status will display “ON” in green colour. If the ending time is selected, the electrical appliance will be turned off when the ending time is reached. When the ending time is reached, the selected channel of Anto will be accessed with the application. Through HTTP API request, the current value of the channel will be returned to “0”. Then, the NodeMCU board will subscribe the "0" current value of the selected channel through MQTT broker and the state of the GPIO pin will be changed from HIGH to LOW to turn the electrical appliance off. When the electrical appliance is turned off, NodeMCU board will publish the current value of "0" to the channel. Through GET method, the application will receive the updated current value of the selected channel, then the red button of the ON/OFF page will return to the green button and the button status will display "OFF" in red colour.

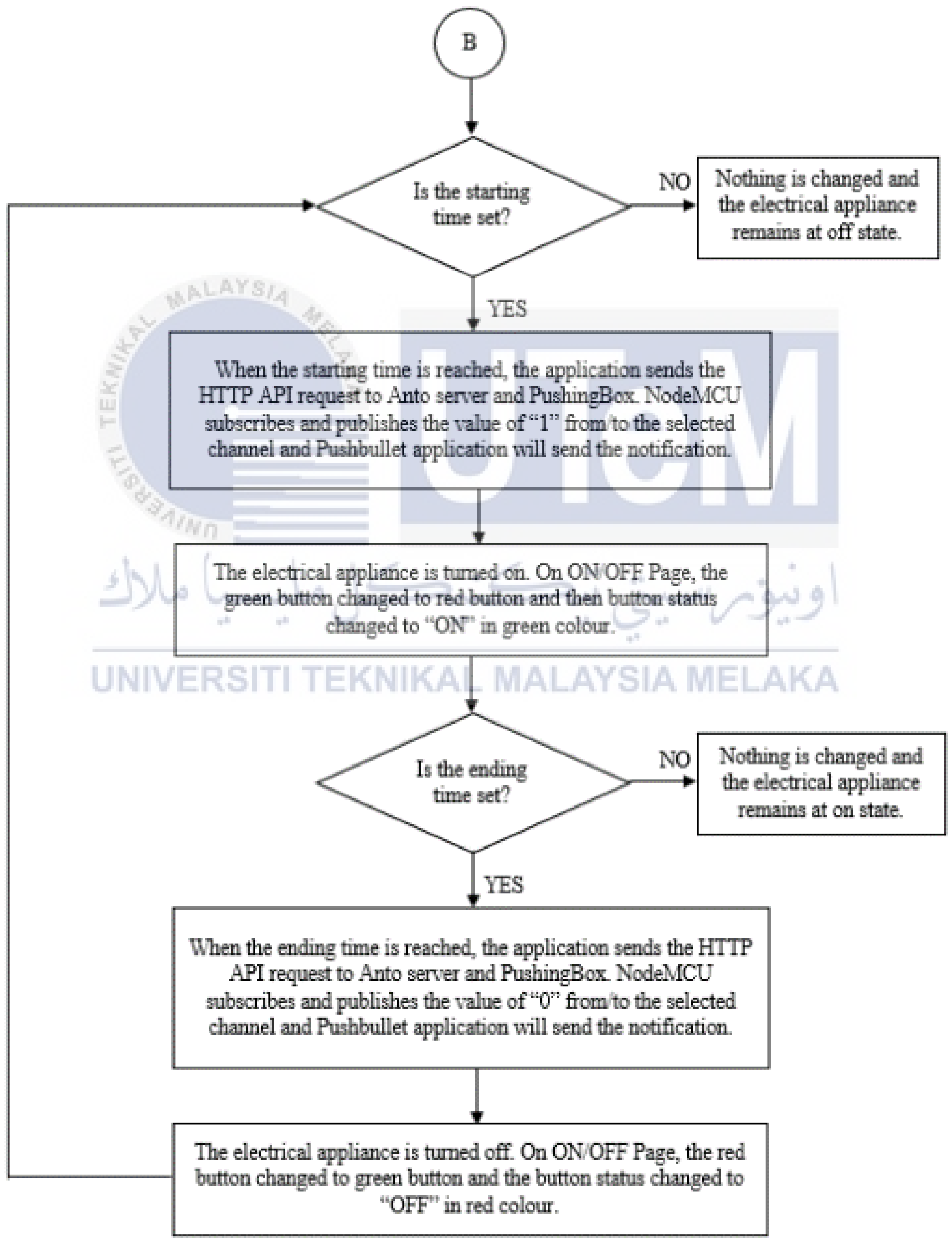
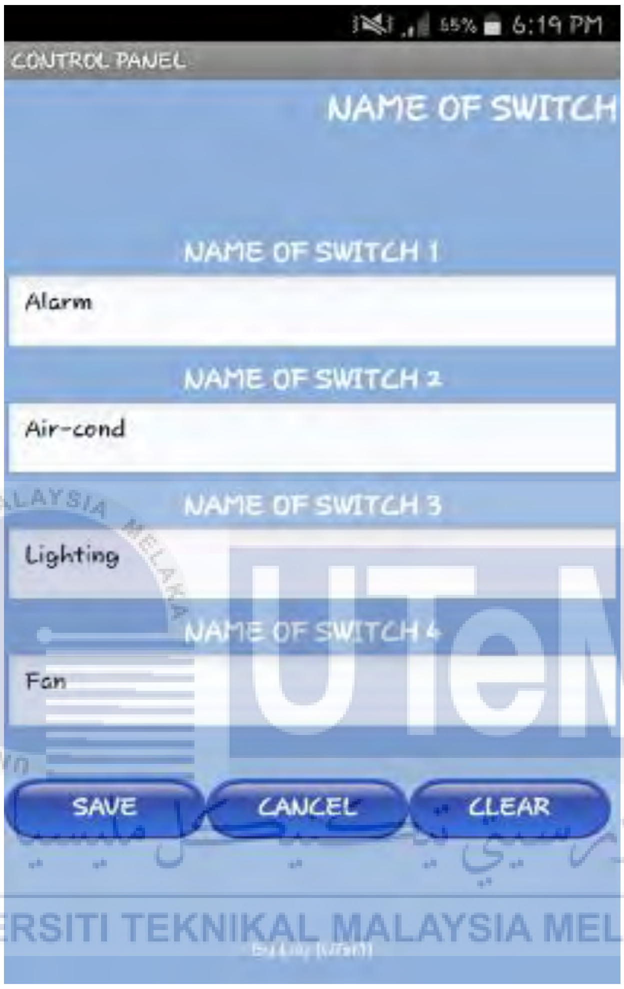

B
Is the starting time set?
NO
Nothing is changed and the electrical appliance remains at off state.
YES
When the starting time is reached, the application sends the HTTP API request to Anto server and PushingBox. NodeMCU subscribes and publishes the value of "1" from/to the selected channel and Pushbullet application will send the notification.
The electrical appliance is turned on. On ON/OFF Page, the green button changed to red button and then button status changed to "ON" in green colour.
Is the ending time set?
NO
Nothing is changed and the electrical appliance remains at on state.
YES
When the ending time is reached, the application sends the HTTP API request to Anto server and PushingBox. NodeMCU subscribes and publishes the value of "0" from/to the selected channel and Pushbullet application will send the notification.
The electrical appliance is turned off. On ON/OFF Page, the red button changed to green button and the button status changed to "OFF" in red colour.

Figure 21: Flow chart of TIMER SETTING page

Figure 22 shows the UI design of NAME OF SWITCH page. According to the figure, there is a textbox for every switch and three optional buttons on the bottom of the screen including SAVE button, CANCEL button and CLEAR button.



Alarm

Air-cond

Lighting

Fan

Figure 22: UI design of NAME OF SWITCH page

Figure 23 shows the flow chart of NAME OF SWITCH page. According to the flow chart, if the SAVE button is pressed, the name that are typed in the textbox will be stored and the name of the switches on ON/OFF page and TIME SETTING page will be changed. Then, the screen will return to CONTROL PANEL page. If the CLEAR button is pressed, the name of switches will set to default which are Switch 1, Switch 2, Switch 3 and Switch 4. The default name will be stored and the name of the switches on ON/OFF page and TIME SETTING page will be changed. If the CANCEL button is pressed, the screen will return to CONTROL PANEL page.

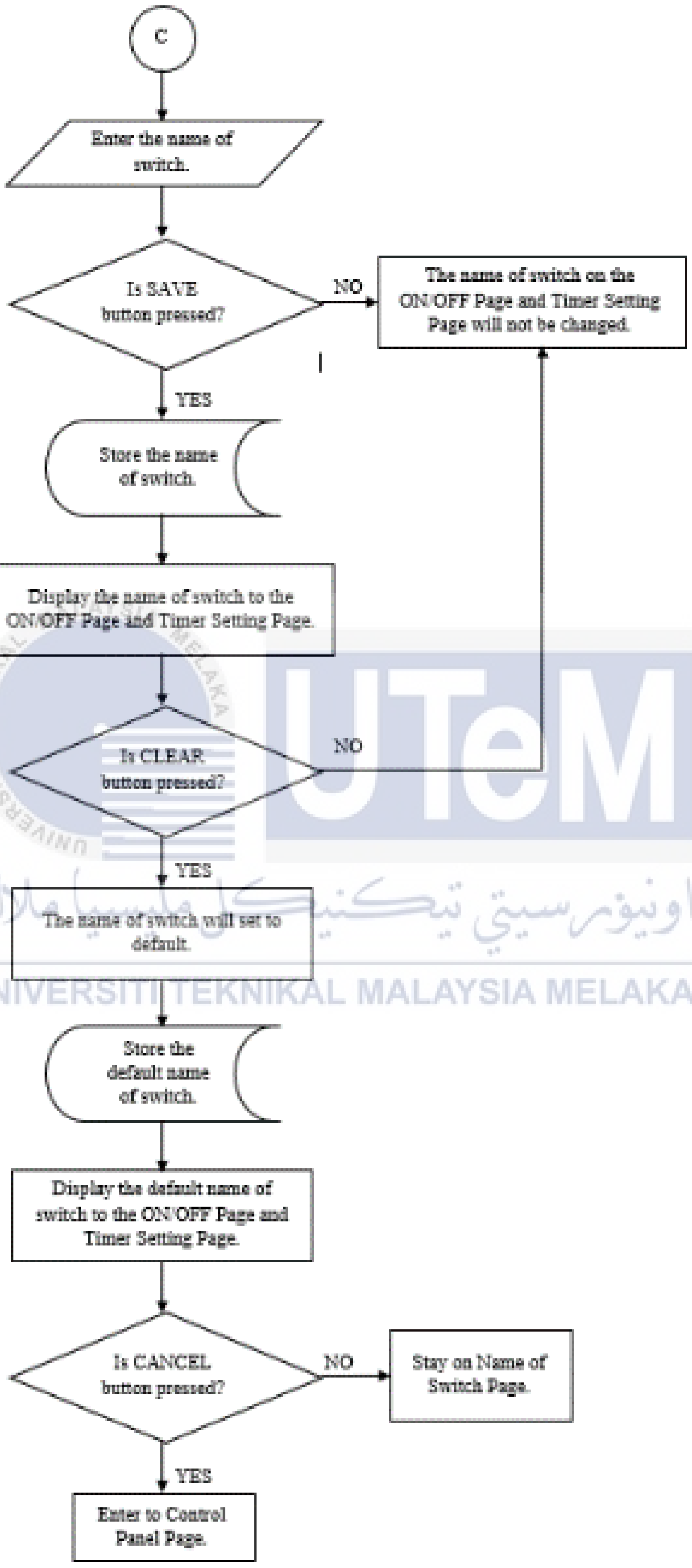

C
Enter the name of switch.
Is SAVE button pressed?
NO
The name of switch on the ON/OFF Page and Timer Setting Page will not be changed.
YES
Store the name of switch.
Display the name of switch to the ON/OFF Page and Timer Setting Page.
Is CLEAR button pressed?
NO
YES
The name of switch will set to default.
Store the default name of switch.
Display the default name of switch to the ON/OFF Page and Timer Setting Page.
Is CANCEL button pressed?
NO
Stay on Name of Switch Page.
YES
Enter to Control Panel Page.

Figure 23: Flow chart of NAME OF SWITCH page

3.10 Interfacing of the Android Application with the Anto Server and PushingBox Cloud

Both Anto server and Pushingbox can communicate with the application through HTTP API request by using GET and SET method. For the interfacing of the Android application with the Anto server, the name of Key, Thing and Channel are important to request the permission to read and update the current value of the channels. Table 8 shows the reading data function by using GET method. The application can read the current value of the channel through the HTTP API request as shown in Table 9. For the return value, it is in the form of JSON. If the request is successful, the result is true whereas the result is false if the request fails.

Table 9: The reading data function by using GET method

Function	GET Method
Value sent	curl "https://api.anto.io/channel/get/Key/ThingName/ChannelName" For example, the HTTP API request for getting the data from SW1 channel: curl "https://api.anto.io/channel/set/grPFnLVq7bQlX4tR61Vl2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1"
Return value	In case of success {"result": "true", "value": "1"} In case of failure {"result": "false", "id": "GETCHERR02", "message": "Incorrect API request or resources is not existed or permission denied"}

Table 10 shows the updating data function by using SET method. The application can update the current value of channel through the HTTP API request as shown in Table 10. It is depending on the type of data such as switch on/off information. Set the desired value of channel to "1" to on or update the desired value of channel to "0" off. For the return value, it is in the form of JSON. If the request is successful, the result is true whereas the result is false if the request fails.

Table 10: The updating data function by using SET method

Function	SET Method
Value sent	curl “https://api.anto.io/channel/set/Key/ThingName/ChannelName /DesiredValue” For example, the HTTP API request for updating the data of SW1 channel to on: curl “https://api.anto.io/channel/set/grPFnLVq7bQlX4tR61Vl2PxDOP 8web1A7UVX8CQo/Smart_Home_Control/SW1/1”
Return value	In case of success {“result”: “true”, “value”: “0”} In case of failure {“result”: “false”, “id”: “GETCHERR02”, “message”: “Incorrect API request or resources is not existed or permission denied”}

For the interfacing of the Android application with the PushingBox, the DeviceID of the scenario is the unique key to identify which scenario to be launched. The HTTP API request can be used to launch a scenario for sending the notification text to Pushbullet application when the button on UI of Android application is triggered. Table 11 shows the sending notification text function by using GET method.

Table 11: The sending notification text function by using GET method

Function	GET Method
Sending notification text	curl “https://api.pushingbox.com/pushingbox?devid=Scenario’s Device ID” For example, the HTTP API request for sending the notification text of scenario Switch1_ON to Pushbullet application: curl “http://api.pushingbox.com/pushingbox?devid=v21FFE94118564 EB”

3.11 The Integrated Hardware for the Prototype

For the prototype, the components such as resistors, LEDs, buzzer and DC motor are used to represent the electrical appliances. This project only involves four types of electrical appliances such as alarm, air-conditioner, lighting and fan.

Figure 24 shows the circuit diagram of the alarm. Based on the figure, a 6V buzzer is used to represent the alarm. For the circuit construction, the Vin pin and GND pin of NodeMCU board and relay module are both connected to the power supply and ground respectively. D1 pin of the NodeMCU is connected to the IN1 input of relay module. The buzzer can be powered up by connecting the positive side of the buzzer to the power supply whereas the negative side is connected to the COM terminal of the first relay. The NO terminal of first relay is then connected to the ground. Therefore, when the D1 pin of the NodeMCU is energized, it will activate the contact of first relay. Then, the current will flow through the buzzer and the sound of the buzzer will be turned on.

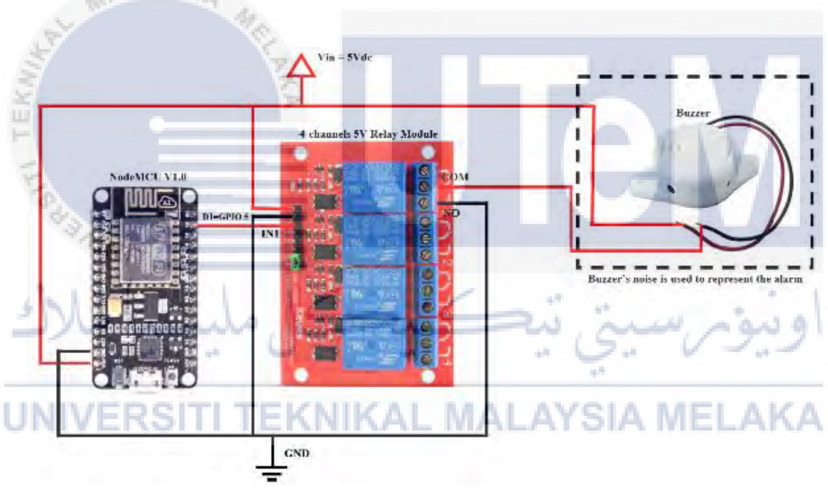

Vin = 5Vdc
4 channels 5V Relay Module
NodeMCU V1.0
D1-GPIO5
IN1
COM
NO
Buzzer
Buzzer's noise is used to represent the alarm
GND

Figure 24: The circuit diagram of the alarm

Figure 25 shows the circuit diagram of the air-conditioner. Based on the figure, a 75ohms of resistor and blue LED are used to represent the air-conditioner. For the circuit construction, the Vin pin and GND pin of NodeMCU board and relay module are both connected to the power supply and ground respectively. D5 pin of the NodeMCU is connected to the IN2 input of relay module. The power supply is connected to one side of the resistor. Then, the resistor is connected in series to the positive side of the blue LED and the negative side of the blue LED is connected to the COM terminal of second relay. The NO terminal of second relay is then connected to the ground. Therefore, when the D5 pin of the NodeMCU is energized, it will activate the contact of second relay. Then, the current will flow through the resistor and the blue LED and the blue LED will light up.

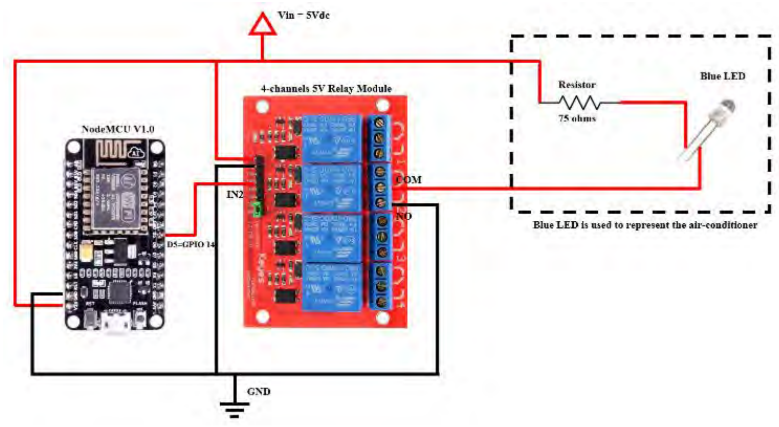

Vin = 5Vdc
4-channels 5V Relay Module
NodeMCU V1.0
IN2
D5-GPIO 14
COM
NO
GND
Resistor
75 ohms
Blue LED
Blue LED is used to represent the air-conditioner

Figure 25: The circuit diagram of the air-conditioner

Figure 26 shows the circuit diagram of the lighting. Based on the figure, there are four resistors and four different colour LEDs are used to represent the lighting. For the circuit construction, the Vin pin and GND pin of NodeMCU board and relay module are both connected to the power supply and ground respectively. D7 pin of the NodeMCU is connected to the IN3 input of relay module. The 220ohms of resistor is connected in series to the positive side of the LED and the four sets of the combination are connected in parallel. The power supply is connected to one side of the resistor. Then, the COM terminal of third relay is connected to the negative side of the LED. The NO terminal of third relay is then connected to the ground. Therefore, when the D7 pin of the NodeMCU is energized, it will activate the contact of third relay. Then, the current will flow through the resistors and the LEDs and all the LEDs will light up.

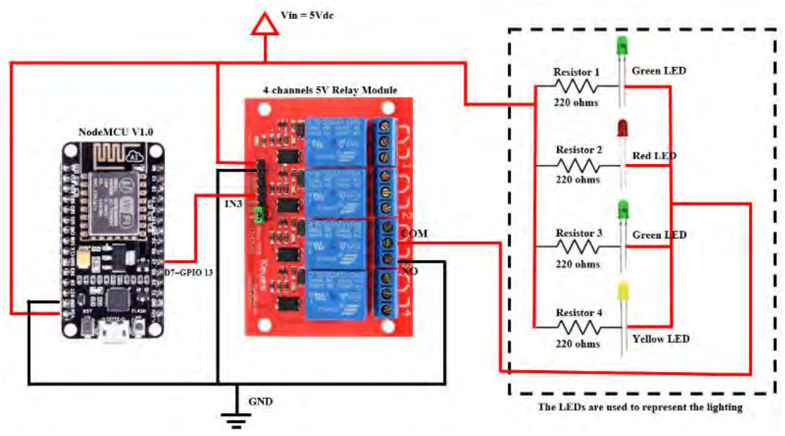


Figure 26: The circuit diagram of the lighting

Figure 27 shows the circuit diagram of the alarm. Based on the figure, a 3V-6V DC motor is used to represent the fan. For the circuit construction, the Vin pin and GND pin of NodeMCU board and relay module are both connected to the power supply and ground respectively. D8 pin of the NodeMCU is connected to the IN4 input of relay module. The DC motor can be powered up by connecting the positive side of the motor to the power supply whereas the negative side is connected to the COM terminal of the fourth relay. The NO terminal of fourth relay is then connected to the ground. Therefore, when the D8 pin of the NodeMCU is energized, it will activate the contact of fourth relay. Then, the current will flow through the motor and the motor will spin.

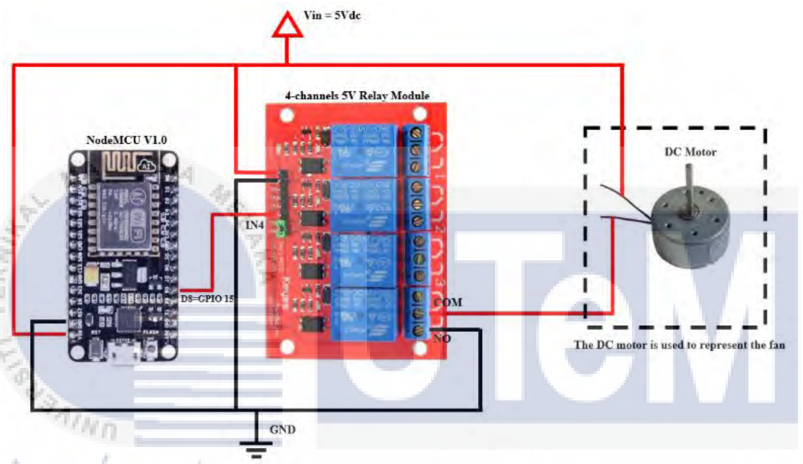

Vin = 5Vdc
4-channels 5V Relay Module
NodeMCU V1.0
IN4
D8=GPIO 15
COM
NO
GND
DC Motor
The DC motor is used to represent the fan

Figure 27: The circuit diagram of the fan

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Overview

This chapter is a results and discussion chapter that discusses more on the hardware and how the whole system will be implemented together with the hardware and software. A sample of simple application and implementation will be demonstrated to prove the functionality of the Android application and the microcontroller for remotely controlling the electrical appliances. The adequacy, efficiency, productiveness and effectiveness of the proposed project will be clearly stated and justify.

4.2 Project Achievement

A smart home system is implemented successfully to control the electrical appliances using Android smartphone over the internet network. Figure 28 and Figure 29 show the front view and back view of the prototype that is built for the project demonstration.

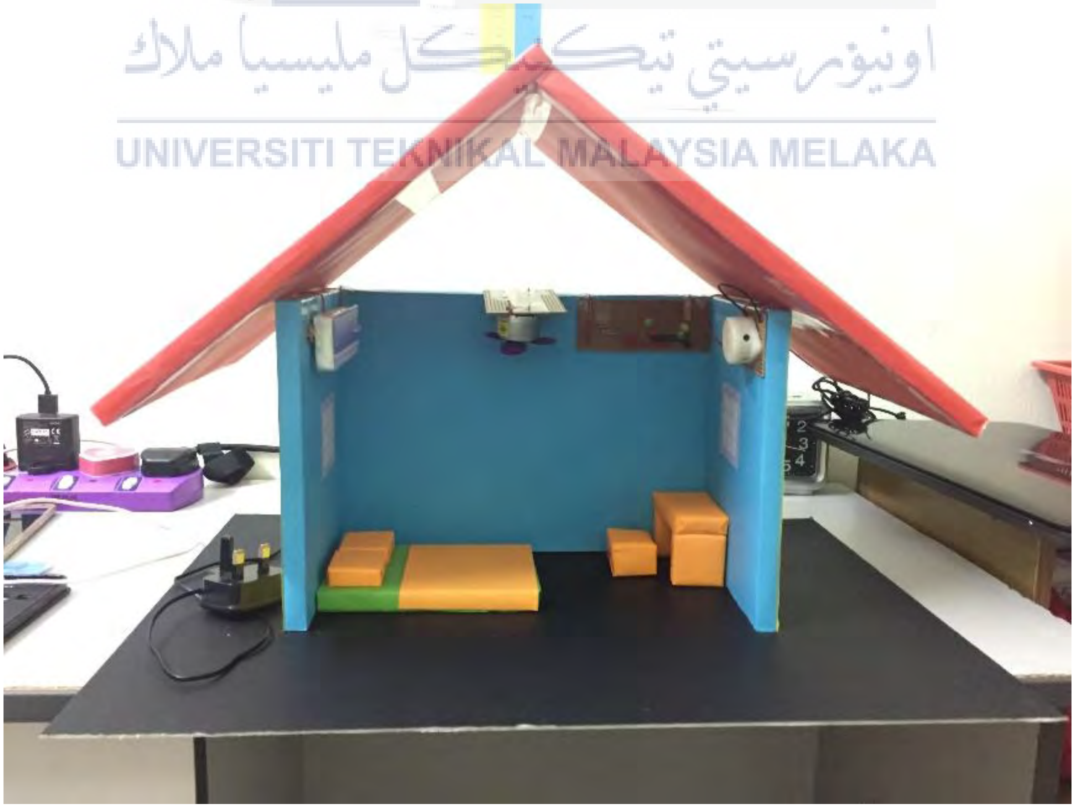


Figure 28: The front view of prototype for demonstration

Based on Figure 28, four types of electrical appliances are used for the demonstration including alarm, air-conditioner, lighting and fan. The alarm is represented by 6V buzzer. When the buzzer is turned on, the noise of the buzzer is considered as the sound of alarm. Since the air-conditioner is too complex for the circuit construction, white LED is used to represent the air-conditioner unit. When the button for air-conditioner is pressed, the blue LED is lighted up. The lighting unit is represented by four different colour LEDs. The LEDs are lighted up when the button for lighting is pressed. For the fan unit, 3-6V DC motor is used to represent the fan. When the DC motor is turned on, the motor will spin like the way of the how the fan moving.

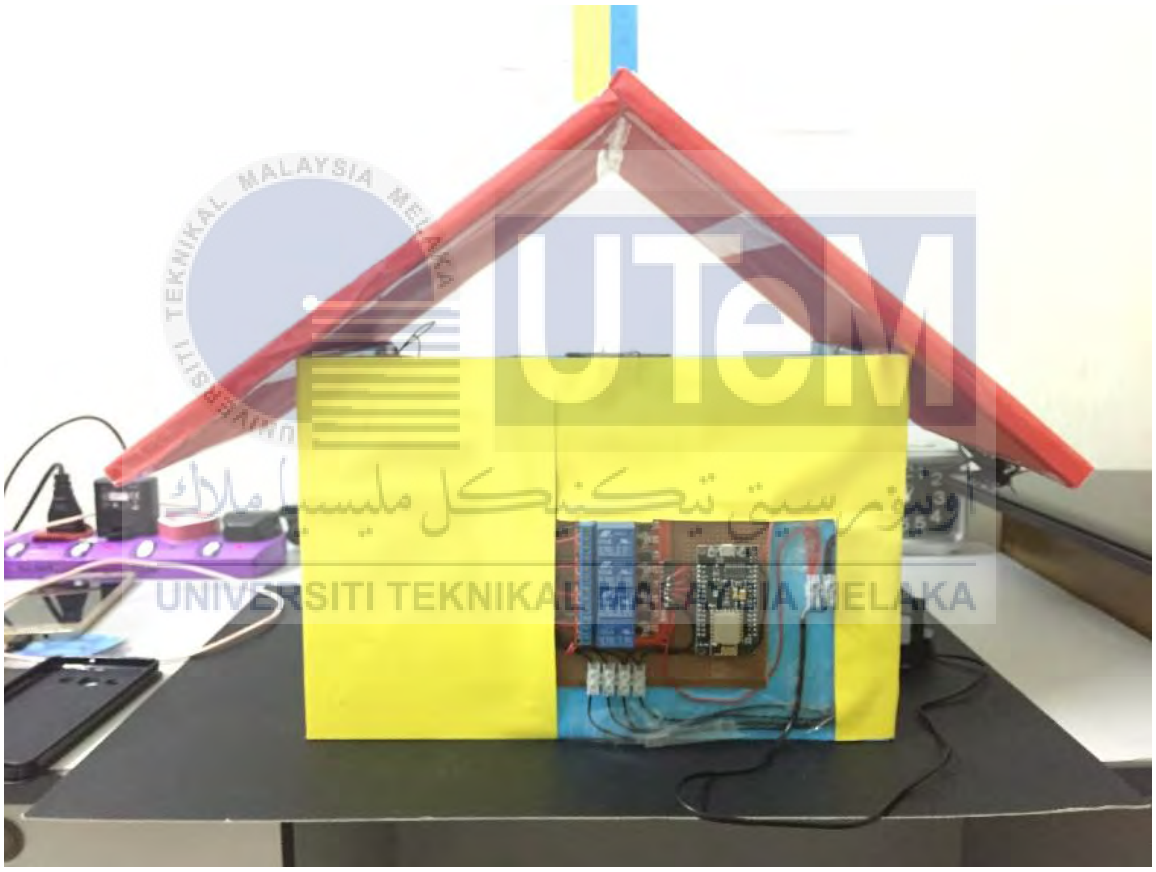


Figure 29: The back view of prototype for demonstration

Based on Figure 29, the power is supplied from 240Vac source. The power is supplied to NodeMCU board, relay module, DC motor, LEDs and buzzer. The voltage and current consumptions of all the components are low. Therefore, a phone charger is used to regulate the power source. The phone charger can transform the input power from AC to DC and transform the high voltage to low voltage which is only 5V. All the hardware is integrated as shown in Figure 28 and Figure 29.

For the software part, Anto server is setup. The Key, Thing and Channels are created as discussed in previous chapter. Both Anto server and NodeMCU are subscriber and publisher. The Key is a unique key of permission request to read and update the data of the channels. Nine channels are created in the Thing list which is known as Smart_Home_Control. The channels are ALL, SW1, SW2, SW3, SW4, SW1_status, SW2_status, SW3_status and SW4_status. All channels are set as on/off type. Therefore, the channels are acted like a switch which the on and off state are represented by the value of "1" and "0" respectively.

Furthermore, a complete programming code is written and uploaded to the NodeMCU board as shown in Appendix A. The NodeMCU board can access to the WiFi connection that provided by the router. Therefore, the NodeMCU can communicate with the Anto MQTT broker using subscribe and publish methods. The GPIO pin of NodeMCU such as D1, D5, D7 and D8 are used to control the four electrical appliances through relay module. The data subscribed from the Anto's channel can write the GPIO pin to HIGH or LOW.

Moreover, the PushingBox cloud is used as the notification purpose. The API of the PushingBox can communicate with the smartphone through GET method and the smartphone can receive the notification from the Pushbullet application. For this proposed project, ten scenarios are created such as ALL_ON, ALL_OFF, SW1_ON, SW1_OFF, SW2_ON, SW2_OFF, SW3_ON, SW3_OFF, SW4_ON and SW4_OFF. Each of the scenarios is provided with a unique DeviceID and the specific notification text is set for every scenario to notify the user through Pushbullet according to different condition.

Additionally, an Android application's APK File is created by using MIT App Inventor 2. The blocks coding diagram as shown in Appendix XX is designed. The combination of the blocks built the behaviour for the application. The application's APK File is then installed on an Android smartphone. The Android application is known as Smart Home Control (SHC). The application provides an UI for user to control the electrical appliances.

4.3 The Functionality of the ON/OFF Page

This section discusses the results for the functionality of the buttons on the ON/OFF page. The ON/OFF page consists five optional buttons. The first button is used to control the alarm that is connected to the D1 pin of the NodeMCU board. The second button is used to control the air-conditioner that is connected to the D5 pin of the NodeMCU board. The third button is used to control the lighting that is connected to the D7 pin of NodeMCU board.

The fourth button is used to control the fan that is connected to the D8 pin of NodeMCU board. The fifth button is used to control the four buttons and all electrical appliances that is connected to the D1, D5, D7 and D8 pins of NodeMCU board. In this case, the name for the first button, second button, third button, fourth button and fifth button are set as Alarm, Air-cond, Lighting, Fan and ALL SWITCHES respectively. The following are the results of the prototype when the button on the ON/OFF page is pressed.

First, make sure the NodeMCU board is connected to the WiFi connection that is provided by the router. Then, the smartphone is connected to its own mobile data. The upper space of the ON/OFF page will show the connectivity status of the application with the Anto server through internet network. The word of "Anto connected" is displayed in green colour, the application is now connected with the Anto server.

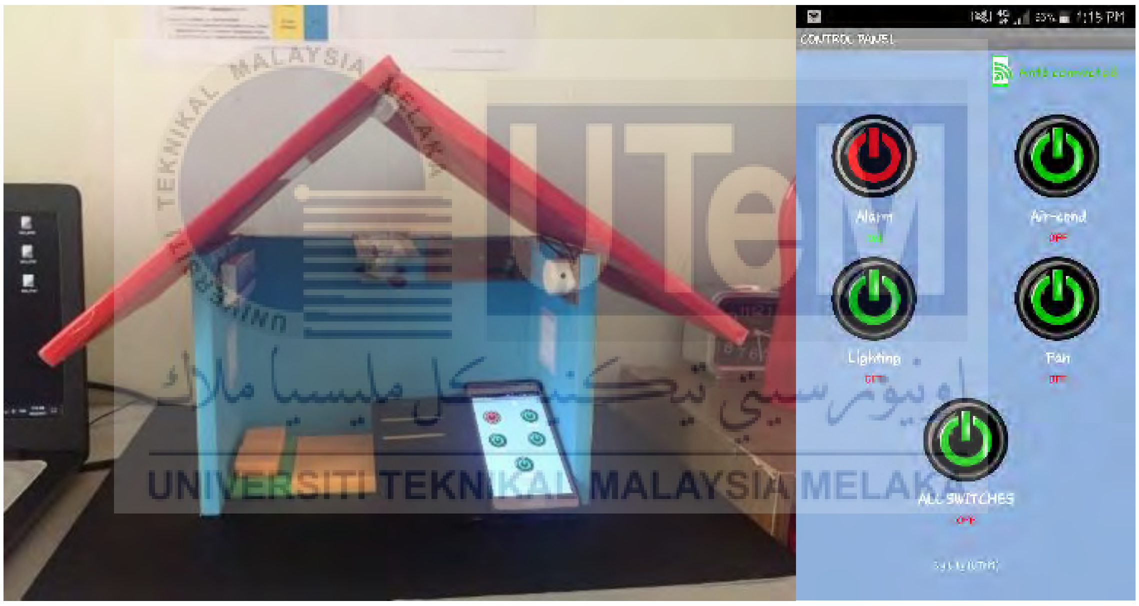


Figure 30: Button for alarm

When the green button for Alarm is pressed, the buzzer is turned on as shown in Figure 30. When the green button for Alarm is pressed, the value of "1" is sent to SW1's channel through HTTP API request. Then, the current value of SW1's channel is updated from "0" to "1". The data is subscribed by the D1 pin of NodeMCU board from MQTT broker. When D1 pin is received the value of "1", the state of the D1 pin is changed from "LOW" to "HIGH". The current is flowed from D1 pin to the IN1 input of relay module and the contact of first relay is activated. The buzzer which connected to the first relay is turned on. When the D1 pin is changed to HIGH state, the data is published to SW1_status's channel and updated the value from "0" to "1". The data of SW1_status's channel is then sent to the application. The green button is changed to red button and the button status is changed from "OFF" to "ON" in green colour. A notification text "Switch 1 is turned ON!" is sent to the smartphone by the Pushbullet application through GET method.

When the red button for Alarm is pressed, the value of "0" is sent to SW1's channel through HTTP API request. Then, the current value of SW1's channel is updated from "1" to "0". The data is subscribed by the D1 pin of NodeMCU board from MQTT broker. When D1 pin is received the value of "0", the state of the D1 pin is changed from "HIGH" to "LOW". The current is flowed from D1 pin to the IN1 input of relay module and the contact of first relay is deactivated. The buzzer which connected to the first relay is turned off. When the D1 pin is changed to LOW state, the data is published to SW1_status's channel and updated the value from "1" to "0". The data of SW1_status's channel is then sent to the application. The red button is returned to green button and the button status is changed from "ON" to "OFF" in red colour. A notification text "Switch 1 is turned OFF!" is sent to the smartphone by the Pushbullet application through GET method.

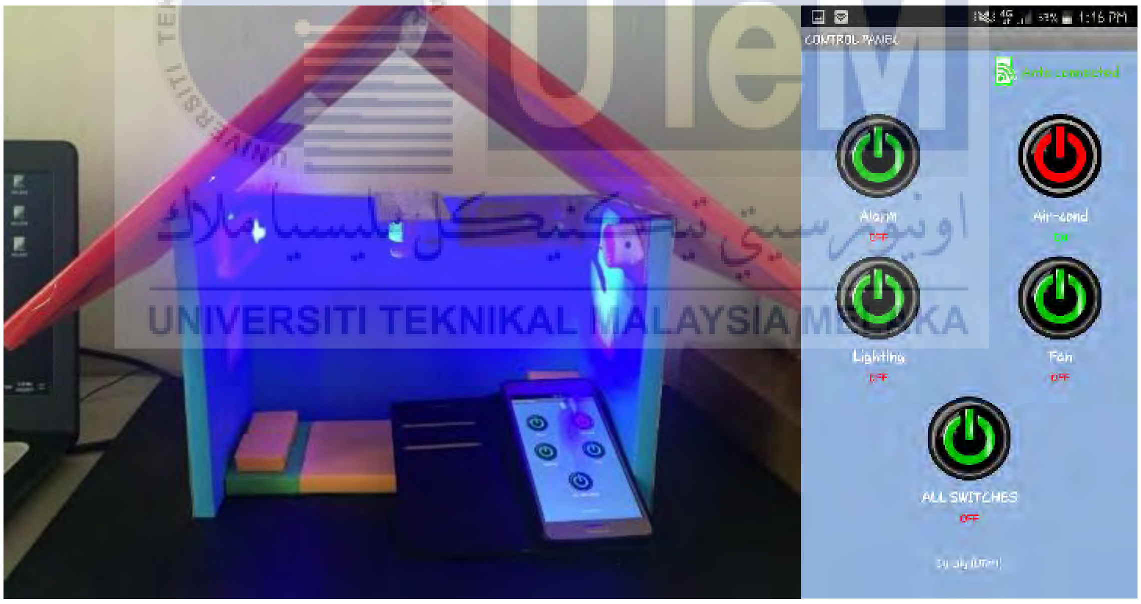


Figure 31: Button for air-conditioner

When the green button for Air-cond is pressed, the blue LED which represented the air-conditioner is lighted up as shown in Figure 31. When the green button for Air-cond is pressed, the value of "1" is sent to SW2's channel through HTTP API request. Then, the current value of SW2's channel is updated from "0" to "1". The data is subscribed by the D5 pin of NodeMCU board from MQTT broker. When D5 pin is received the value of "1", the state of the D5 pin is changed from "LOW" to "HIGH". The current is flowed from D5 pin to the IN2 input of relay module and the contact of second relay is activated. The blue LED which connected to the second relay is lighted up. When the D5 pin is changed to HIGH state, the data is published to SW2_status's channel and updated the value from "0" to "1". The data of SW2_status's channel is then sent to the application. The green button is changed to red button and the button status is changed from "OFF" to "ON" in green colour. A notification text "Switch 2 is turned ON!" is sent to the smartphone by the Pushbullet application through GET method.

When the red button for Air-cond is pressed, the value of "0" is sent to SW2's channel through HTTP API request. Then, the current value of SW2's channel is updated from "1" to "0". The data is subscribed by the D5 pin of NodeMCU board from MQTT broker. When D5 pin is received the value of "0", the state of the D5 pin is changed from "HIGH" to "LOW". The current is flowed from D5 pin to the IN2 input of relay module and the contact of second relay is deactivated. The blue LED which connected to the second relay is turned off. When the D5 pin is changed to LOW state, the data is published to SW2_status's channel and updated the value from "1" to "0". The data of SW2_status's channel is then sent to the application. The red button is returned to green button and the button status is changed from "ON" to "OFF" in red colour. A notification text "Switch 2 is turned OFF!" is sent to the smartphone by the Pushbullet application through GET method.

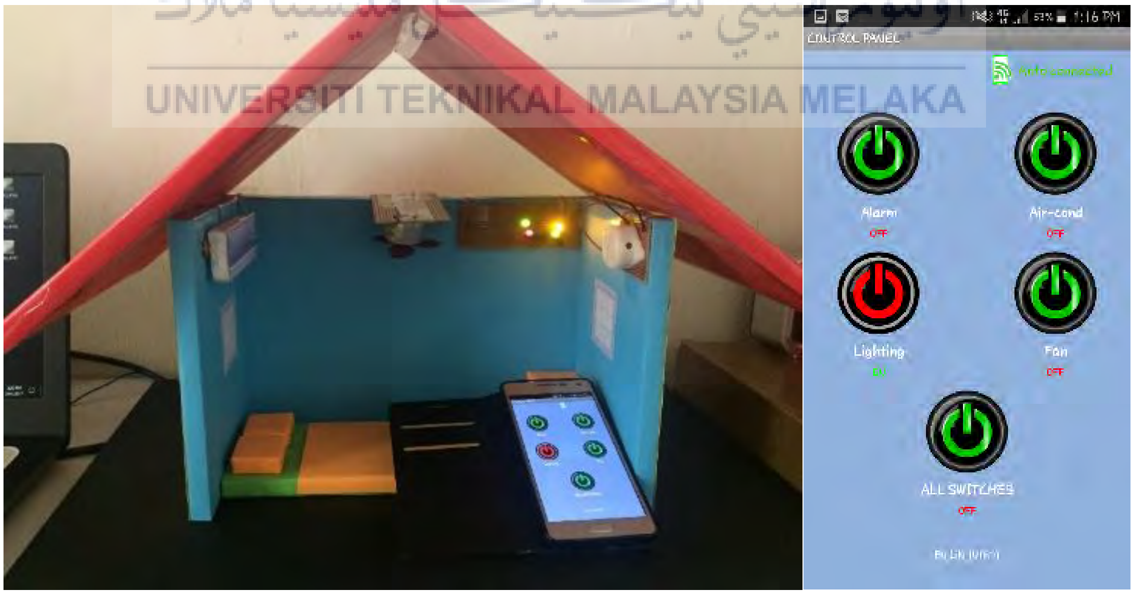


Figure 32: Button for lighting

When the green button for Lighting is pressed, the four different colour of LEDs are lighted up as shown in Figure 32. When the green button for Lighting is pressed, the value of “1” is sent to SW3’s channel through HTTP API request. Then, the current value of SW3’s channel is updated from “0” to “1”. The data is subscribed by the D7 pin of NodeMCU board from MQTT broker. When D7 pin is received the value of “1”, the state of the D7 pin is changed from “LOW” to “HIGH”. The current is flowed from D7 pin to the IN3 input of relay module and the contact of third relay is activated. All LEDs which connected to the third relay are lighted up. When the D7 pin is changed to HIGH state, the data is published to SW3_status’s channel and updated the value from “0” to “1”. The data of SW3_status’s channel is then sent to the application. The green button is changed to red button and the button status is changed from “OFF” to “ON” in green colour. A notification text “Switch 3 is turned ON!” is sent to the smartphone by the Pushbullet application through GET method.

When the red button for Lighting is pressed, the value of “0” is sent to SW3’s channel through HTTP API request. Then, the current value of SW3’s channel is updated from “1” to “0”. The data is subscribed by the D7 pin of NodeMCU board from MQTT broker. When D7 pin is received the value of “0”, the state of the D7 pin is changed from “HIGH” to “LOW”. The current is flowed from D7 pin to the IN3 input of relay module and the contact of third relay is deactivated. All LEDs which connected to the third relay are turned off. When the D7 pin is changed to LOW state, the data is published to SW3_status’s channel and updated the value from “1” to “0”. The data of SW3_status’s channel is then sent to the application. The red button is returned to green button and the button status is changed from “ON” to “OFF” in red colour. A notification text “Switch 3 is turned OFF!” is sent to the smartphone by the Pushbullet application through GET method.

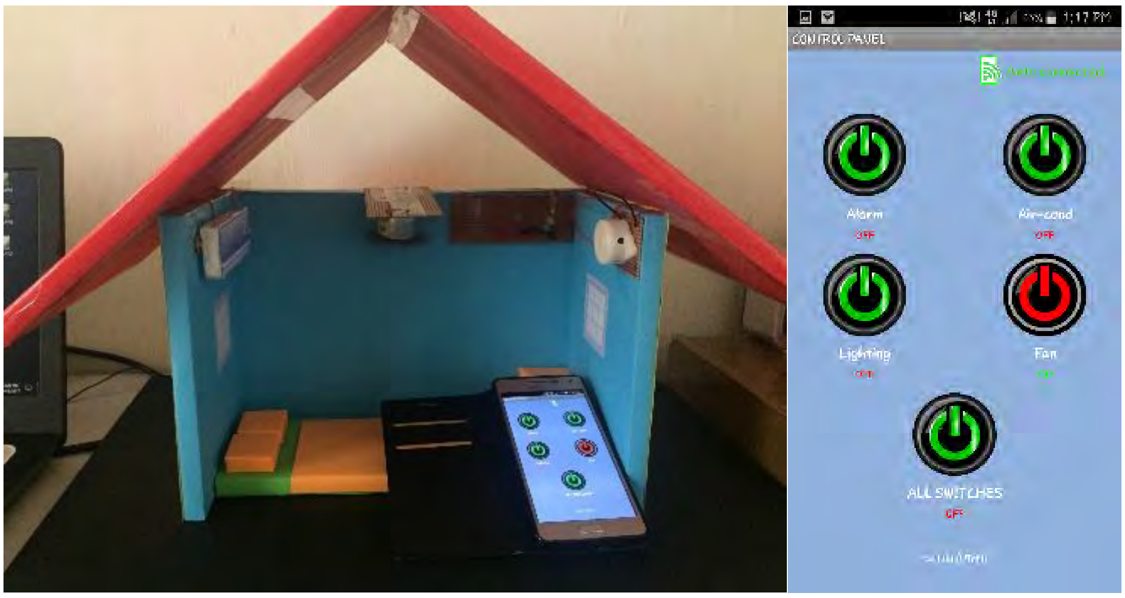


Figure 33: Button for fan

When the green button for Fan is pressed, the DC motor are turned on as shown in Figure 33. When the green button for Fan is pressed, the value of “1” is sent to SW4’s channel through HTTP API request. Then, the current value of SW4’s channel is updated from “0” to “1”. The data is subscribed by the D8 pin of NodeMCU board from MQTT broker. When D8 pin is received the value of “1”, the state of the D8 pin is changed from “LOW” to “HIGH”. The current is flowed from D8 pin to the IN4 input of relay module and the contact of fourth relay is activated. The DC motor which connected to the fourth relay is turned on. When the D8 pin is changed to HIGH state, the data is published to SW4_status’s channel and updated the value from “0” to “1”. The data of SW4_status’s channel is then sent to the application. The green button is changed to red button and the button status is changed from “OFF” to “ON” in green colour. A notification text “Switch 4 is turned ON!” is sent to the smartphone by the Pushbullet application through GET method.

When the red button for Fan is pressed, the value of “0” is sent to SW4’s channel through HTTP API request. Then, the current value of SW4’s channel is updated from “1” to “0”. The data is subscribed by the D8 pin of NodeMCU board from MQTT broker. When D8 pin is received the value of “0”, the state of the D8 pin is changed from “HIGH” to “LOW”. The current is flowed from D8 pin to the IN4 input of relay module and the contact of fourth relay is deactivated. The DC motor which connected to the fourth relay is turned off. When the D8 pin is changed to LOW state, the data is published to SW4_status’s channel and updated the value from “1” to “0”. The data of SW4_status’s channel is then sent to the application. The red button is returned to green button and the button status is changed from “ON” to “OFF” in red colour. A notification text “Switch 4 is turned OFF!” is sent to the smartphone by the Pushbullet application through GET method.

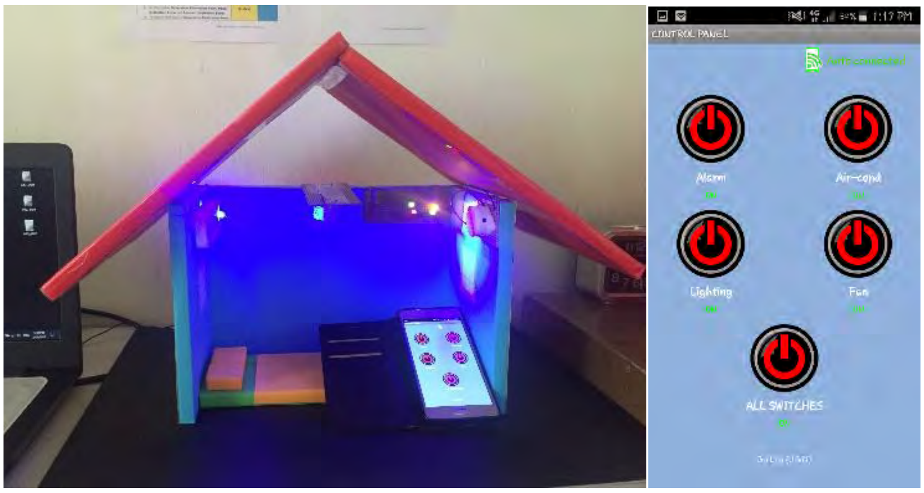


Figure 34: Button for all

When the green button for ALL SWITCHES is pressed, the buzzer, blue LED, LEDs and DC motor are turned on as shown in Figure 34. When the green button for ALL SWITCHES is pressed, the value of "1" is sent to the channel of ALL, SW1, SW2, SW3 and SW4 through HTTP API request. Then, the current value of the channel of ALL, SW1, SW2, SW3 and SW4 are updated from "0" to "1". The data are subscribed by the D1, D5, D7 and D8 pins of NodeMCU board from MQTT broker. When D1, D5, D7 and D8 pins are received the value of "1", the state of the D1, D5, D7 and D8 pins are changed from "LOW" to "HIGH". The current is flowed from D1, D5, D7 and D8 pins to the IN1, IN2, IN3 and IN4 inputs of relay module respectively and the contact of all relays are activated. The buzzer, blue LED, LEDs and DC motor which connected to the relay module are turned on. When the D1, D5, D7 and D8 pins are changed to HIGH state, the data are published to the channel of SW1_status, SW2_status, SW3_status and SW4_status and updated the value from "0" to "1". The application received the data from the channel of SW1_status, SW2_status, SW3_status and SW4_status. All green buttons are changed to red buttons and all button status are changed from "OFF" to "ON" in green colour. A notification text "All switches are turned ON!" is sent to the smartphone by the Pushbullet application through GET method.

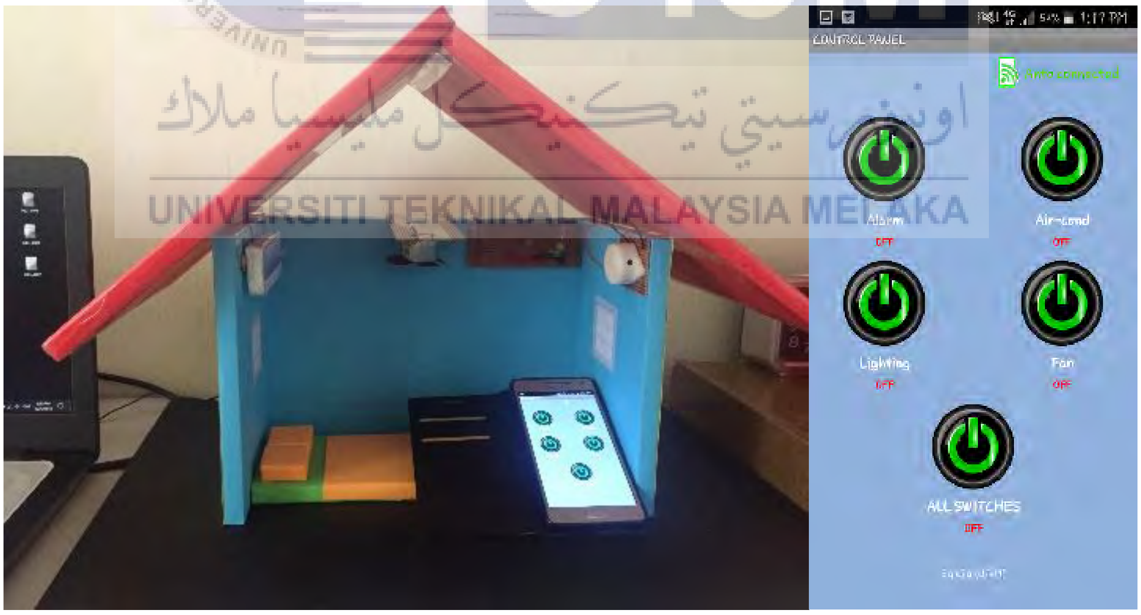


Figure 35: Button for all

When the red button for ALL SWITCHES is long pressed, the buzzer, blue LED, LEDs and DC motor are turned off as shown in Figure 35. When the green button for ALL SWITCHES is long pressed, the value of "0" is sent to the channel of ALL, SW1, SW2, SW3 and SW4 through HTTP API request. Then, the current value of the channel of ALL, SW1, SW2, SW3 and SW4 are updated from “1” to “0”. The data are subscribed by the D1, D5, D7 and D8 pins of NodeMCU board from MQTT broker. When D1, D5, D7 and D8 pins are received the value of “0”, the state of the D1, D5, D7 and D8 pins are changed from “HIGH” to “LOW”. The current is flowed from D1, D5, D7 and D8 pins to the IN1, IN2, IN3 and IN4 inputs of relay module respectively and the contact of all relays are deactivated. The buzzer, blue LED, LEDs and the DC motor which connected to the relay module are turned off. When the D1, D5, D7 and D8 pins are changed to LOW state, the data are published to the channel of SW1_status, SW2_status, SW3_status and SW4_status and updated the value from “1” to “0”. The application is received the data from the channel of SW1_status, SW2_status, SW3_status and SW4_status. All red buttons are returned to green buttons and all button status are changed from “ON” to “OFF” in red colour. A notification text “All switches are turned OFF!” is sent to the smartphone by the Pushbullet application through GET method.

4.4 The Functionality of the TIMER SETTING Page

This section discusses the results for the functionality of the timepickers on the TIMER SETTING page. The TIMER SETTING page comprises four switches for controlling the electrical appliances. There are two types of timer are displayed on the screen including starting time and ending time. User can select the starting time and ending time of the switches through the time pickers. The switches are named as Alarm, Air-cond, Lighting and Fan. The time are performed in 24 hours form. The starting time and ending time of Lighting are set. The starting time and ending time of Lighting are set at 15.22 and 15.23 respectively. Therefore, when the starting time is reached, the LEDs are turned on as shown in Figure 36. The green button of Lighting on ON/OFF page is changed to red button and the button status is changed from “OFF” to “ON” in green colour. A notification text “Switch 3 is turned ON!” is sent to the smartphone by the Pushbullet application. When the ending time is reached, the LEDs are turned off as shown in Figure 37. The red button of Lighting on ON/OFF page is returned to green button and the button status is changed from “ON” to “OFF” in red colour. A notification text “Switch 3 is turned OFF!” is sent to the smartphone by the Pushbullet application. The function of the starting time and the ending time for all switches are similar. The electrical appliances are turned on and turned off according to the starting time and ending time that have been set.

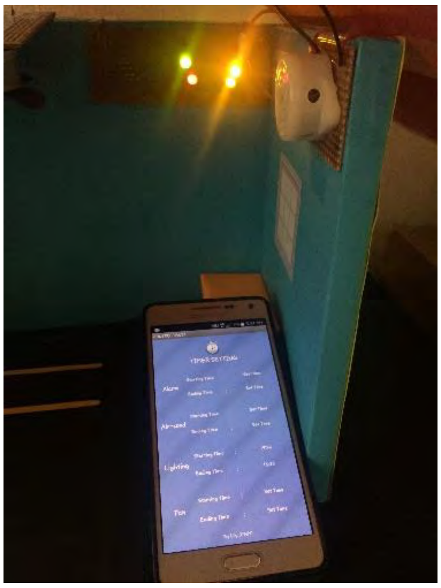


Figure 36: Screenshot of the TIMER SETTING page when starting time of Lighting is reached

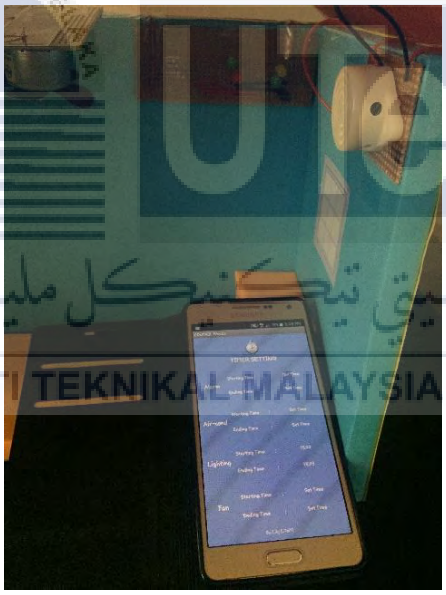
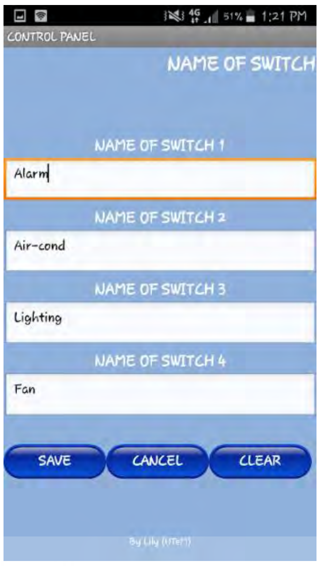


Figure 37: Screenshot of the TIMER SETTING page when ending time of Lighting is reached

4.5 The Functionality of the NAME OF SWITCH Page

This section discusses the results for the functionality of the name changing on the NAME OF SWITCH page. There is a textbox for every switch and three optional buttons on the bottom of the screen including SAVE button, CANCEL button and CLEAR button. The name of switch 1, switch 2, switch 3 and switch 4 are set as Alarm, Air-cond, Lighting and Fan as shown in Figure 38. When the SAVE button is pressed, the names that are typed in the textbox are stored and the names of the switches on ON/OFF page and TIME SETTING page are changed. Then, the screen is returned to CONTROL PANEL page.



NAME OF SWITCH

NAME OF SWITCH 1: Alarm

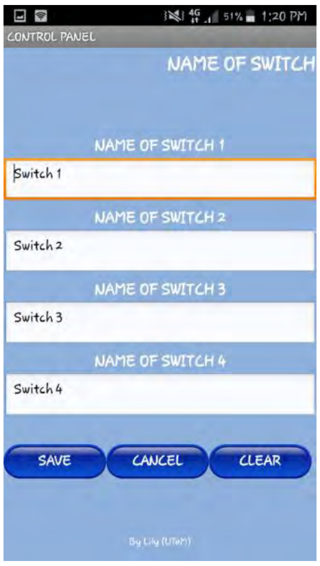
NAME OF SWITCH 2: Air-cond

NAME OF SWITCH 3: Lighting

NAME OF SWITCH 4: Fan

Figure 38: Screenshot of the NAME OF SWITCH page when the SAVE button is pressed

When the CLEAR button is pressed, the names of switches are set to default which are Switch 1, Switch 2, Switch 3 and Switch 4 as shown in Figure 39. The default names are stored and the names of the switches on ON/OFF page and TIME SETTING page are changed. When the CANCEL button is pressed, the screen is returned to CONTROL PANEL page.



NAME OF SWITCH

NAME OF SWITCH 1: Switch 1

NAME OF SWITCH 2: Switch 2

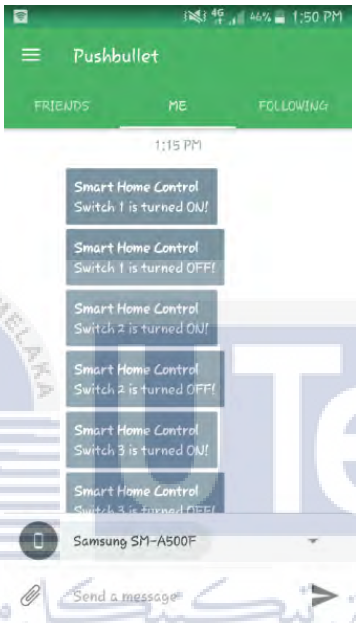
NAME OF SWITCH 3: Switch 3

NAME OF SWITCH 4: Switch 4

Figure 39: Screenshot of the NAME OF SWITCH page when the CLEAR button is pressed

4.6 The Functionality of the Pushbullet Application

Pushbullet application is one of the service that supported by PushingBox cloud. When the button on UI of the application is pressed and the electrical appliance is turned on. The application launched the selected scenario through HTTP API request. Then, Pushbullet application is reading the data that sent from the selected scenario using the GET method. Figure 40 shows the screenshot of the Pushbullet notification messages. It notified the user when the state of the electrical appliance is changed.



Smart Home Control
Switch 1 is turned ON!

Smart Home Control
Switch 1 is turned OFF!

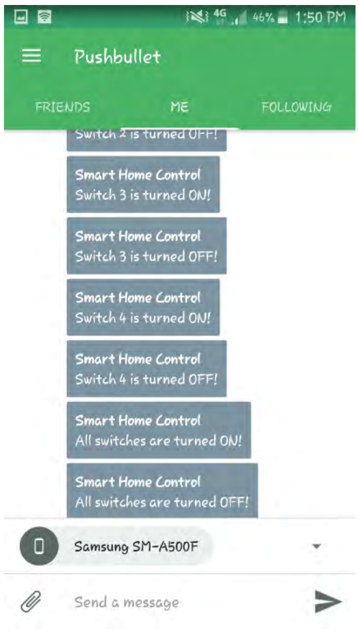
Smart Home Control
Switch 2 is turned ON!

Smart Home Control
Switch 2 is turned OFF!

Smart Home Control
Switch 3 is turned ON!

Smart Home Control
Switch 3 is turned OFF!

Figure 40: Screenshot of the Pushbullet notification messages



Switch 2 is turned OFF!

Smart Home Control
Switch 3 is turned ON!

Smart Home Control
Switch 3 is turned OFF!

Smart Home Control
Switch 4 is turned ON!

Smart Home Control
Switch 4 is turned OFF!

Smart Home Control
All switches are turned ON!

Smart Home Control
All switches are turned OFF!

Figure 41: Screenshot of the Pushbullet notification messages

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Overview

This chapter is a conclusion and recommendation chapter that discusses the summary of the proposed project. The limitation and the weakness of the proposed project are presented in this chapter. Recommendations are suggested for the improvement of the proposed smart home system.

5.2 Conclusion

In conclusion, the objectives are achieved successfully. The smart home control system is developed using the internet network. For this proposed project, the central controller and the smartphone are needed to connect to the internet network. However, the central controller and the smartphone are not necessary to connect to the same IP address. For example, the NodeMCU board as the central controller is connected to the WiFi connection of the local internet network so that it can subscribe or publish the data to/from the Anto server. However, the smartphone is connected to its own mobile data to access to the Anto server and Pushingbox cloud through HTTP API request. Therefore, user can control the home electrical appliances using smartphone from anywhere around the world. Furthermore, a low-cost of smart home control system is developed. The proposed system uses the low-cost hardware and open source software. The low-cost hardware includes ESP8266-12E NodeMCU development board and 4-channels 5V relay module. The total cost of the hardware is less than RM70. The NodeMCU board is considered as the cheapest microcontroller that can be used to control actuators and integrated with the ESP8266 chip that can be used to connect to the WiFi connection. The open-source software includes Arduino IDE, MIT App Inventor 2, Anto server and PushingBox cloud. The software is open-source and free of charge. Additionally, an Android application is developed to control the home electrical appliances. An application that known as Smart Home Control (SHC) is created and installed to the smartphone. The SHC application provides the UI for controlling the electrical appliances. User can control the electrical appliances by pressing the button or set the starting time and ending time to control the electrical appliances automatically.

Eventually, a smart home system that uses to control the electrical appliances using Android smartphone is implemented successfully.

5.3 Recommendation

For this proposed project, the mobile application is created using MIT App Inventor 2. This kind of Android SDK is easy to use and developer does not require any programming knowledge for creating an application. However, the functions that provided by the MIT App Inventor 2 are limited. Furthermore, the application is only compatible to Android OS smartphone. An application that compatibles with all OS smartphone should be developed. This is because the IOS users is the second large market in the smartphone market. Moreover, the smart home system should include the climate control, security, surveillance, automation function and sensors. A fully automated smart home system should be developed. The smart home technology is growing rapidly in this recent years. The improvement of the technology plays the important role to build a full automated smart home system, convenience and comfort living places.

REFERENCES

[1] Vendela Redriksson, 2005. What is a SmartHome or Building. Retrieved from http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci540859,00.html#.

[2] What is a “Smart Home”?. *Smart Home Energy*. Retrieved from http://smarthome energy.co.uk/what-smart-home.

[3] Leo King, 2015. The evolution of the smart home. Retrieved from http://racon teur.net/technology/the-evolution-of-the-smart-home.

[4] Kaylee, Jesse and Simon, 2014. Internet of Things in Home Automation and Energy Efficient Smart Home Technologies. *2014 IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA, USA*.

[5] How Much Does it Cost to Install a Home Automation System?. *Home Advisor*. Retrieved from http://www.homeadvisor.com/cost/electrical/install-or-repair-a-home-automation-system/.

[6] J.T. Barett. The Disadvantages of Bluetooth Technology. Retrieved from https://www.techwalla.com/articles/the-disadvantages-of-bluetooth-technology.

[7] K. Elissa, 2013. RTOS based Home Automation System using Android. *International Journal of Advanced Trends in Computer Science and Engineering*, Vol.2 , No.1, Pages : 480 – 484.

[8] Rakesh K. Deore, Vijay R. Sonawane and Pooja H. Satpute, 2015. Internet of Thing Based Home Appliances Control. *2015 International Conference on Computational Intelligence and Communication Networks*.

[9] CONSUMERLAB, 2015. Connected Homes. *An Ericsson Consumer Insight Summary Report*. Retrieved from https://www.ericsson.com/res/docs/2015/ consumerlab/ericsson-consumerlab-connected-homes.pdf.

[10] Zainuri Ikhsan. Advantages And Disadvantages Of Wireless Technology. *Electrical World*. Retrieved from http://antekel.blogspot.my/2015/10/wireless-or-wireless-technology-is.html.

[11] The Advantages of a Smart House. *SFGATE*. Retrieved from http://homeguides. sfgate.com/advantages-smart-house-8670.html.

[12] Vangie Beal, 2017. Mobile Operating Systems (Mobile OS) Explained. *Webopedia*. Retrieved from: http://www.webopedia.com/DidYouKnow/Hardware_Software/ mobile-operating-systems-mobile-os-explained.html

[13] Android Operating System. *Investopedia*. Retrieved from http://www.investopedia. com/terms/a/android-operating-system.asp.

[14] Android (operating system). *Wikipedia The Free Encyclopedia*. Retrieved from https:// en.wikipedia.org/wiki/Android_(operating_system).

[15] Microcontroller. *Wikipedia The Free Encyclopedia*. Retrieved from https://en.wiki pedia.org/wiki/Microcontroller.

[16] Marcel Stor, 2015. Comparison of ESP8266 NodeMCU development boards. Retrieved from https://frightanic.com/iot/comparison-of-esp8266-nodemcu-development-boards.

[17] HTTP Methods: GET vs POST. *weschools.com*. Retrieved from https://www.w3 schools.com/tags/ref_httpmethods.asp.

[18] Muneer, 2016. The best TCP/IP Light Weight Protocol-MQTT. *Krify*. Retrieved from https://krify.co/tag/mqtt-tcpip-sessions/.

[19] MQTT. *OMICS International*. Retrieved from http://research.omicsgroup.org/index. php/MQTT.

[20] Abdul Ahad, Y. Mahesh, G. Sukanya, N. Harika and A. Uma Sri, 2014. Automated Home Appliances Control Using Embedded Web Server. *Journal of Telematics and Informatics (JTI) Vol.2*, No.1, pp. 15~21, ISSN: 2303-3703.

[21] GPIO Module. *NodeMCU Documentation*. Retrieved from https://nodemcu.readthe docs.io/en/master/en/modules/gpio/.

[22] Kumar Mandula, Ramu Parupalli, CH.A.S.Murty, E.Magesh and Rutul Lunagariya, 2015. Mobile based Horne Automation using Internet of Things(IoT). *2015 International Conference on Control, lnstrumentation, Communication and Computational Technologies (lCCICCT).*

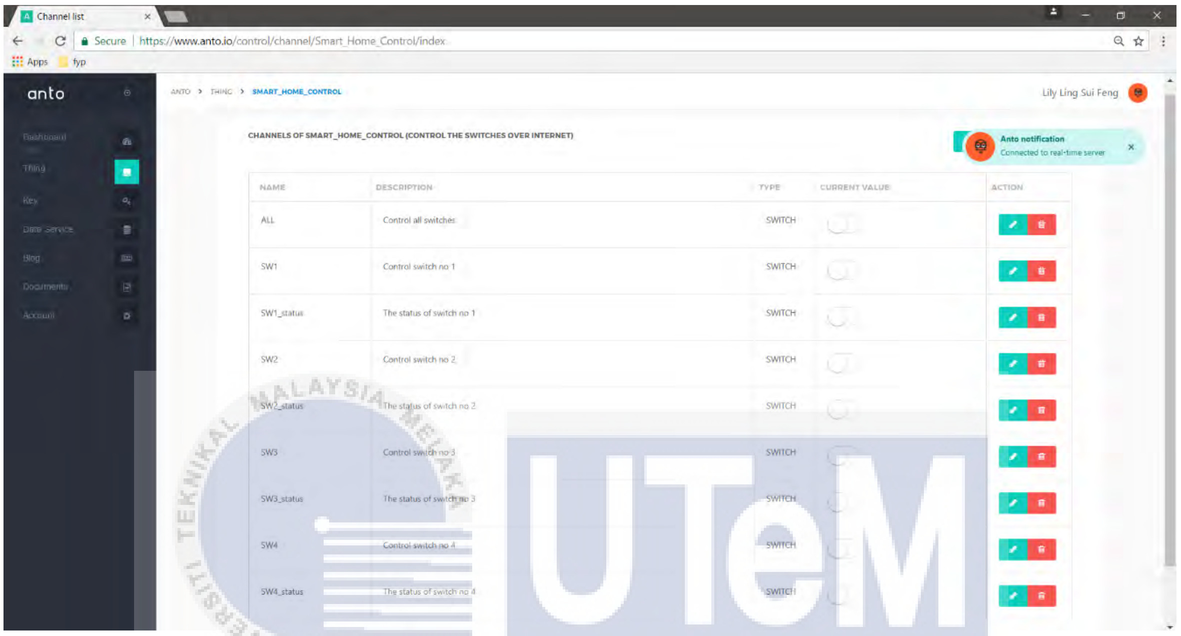
[23] Ehsan Dolatshahi, 2015. Smart Home Automation System. *Degree of Master of Science in Computer Engineering*.

[24] Designer and Blocks Editor. *MIT App Inventor*. Retrieved from http://appinventor. mit.edu/explore/designer-blocks.html.

APPENDIX A

Anto Server

The following is a screenshot of the Channels that created in Anto server.


Channel list
Secure https://www.anto.io/control/channel/Smart_Home_Control/index
Apps
fyp
anto
Dashboard
Thing
Key
Data Service
Blog
Documents
Account
ANTO > THING > SMART_HOME_CONTROL
Lily Ling Sui Feng
Anto notification
Connected to real-time server

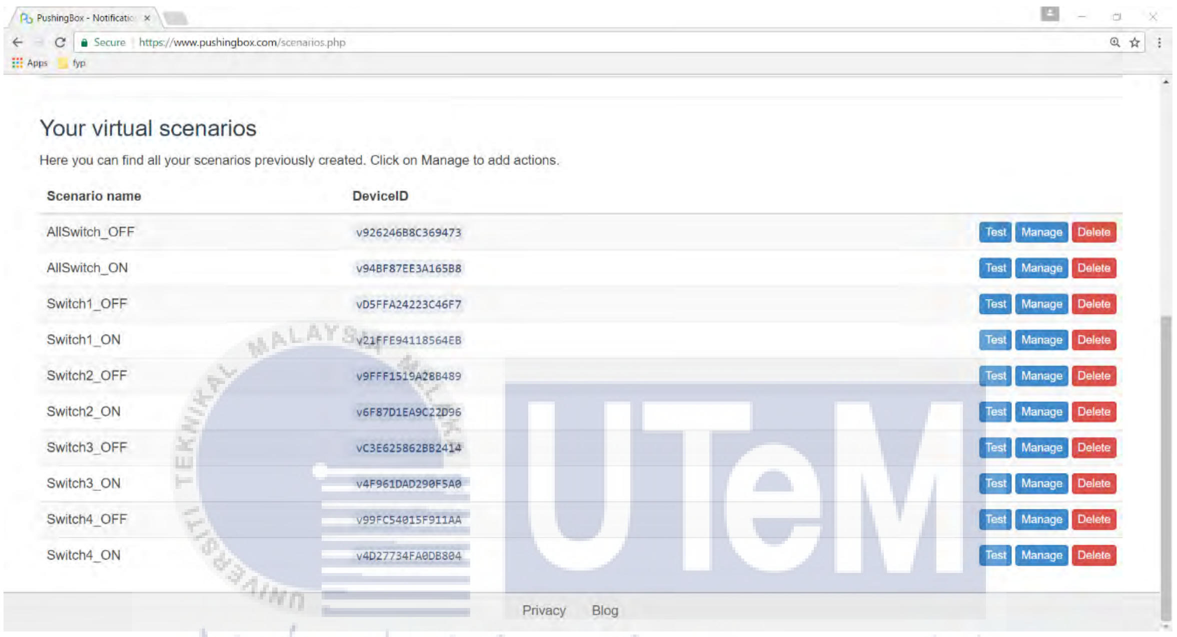
CHANNELS OF SMART_HOME_CONTROL (CONTROL THE SWITCHES OVER INTERNET)

NAME	DESCRIPTION	TYPE	CURRENT VALUE	ACTION
ALL	Control all switches	SWITCH		
SW1	Control switch no 1	SWITCH		
SW1_status	The status of switch no 1	SWITCH		
SW2	Control switch no 2	SWITCH		
SW2_status	The status of switch no 2	SWITCH		
SW3	Control switch no 3	SWITCH		
SW3_status	The status of switch no 3	SWITCH		
SW4	Control switch no 4	SWITCH		
SW4_status	The status of switch no 4	SWITCH		

APPENDIX B

PushingBox Cloud

The following is a screenshot of the virtual scenarios that created in PushingBox cloud for notification purpose.



Your virtual scenarios

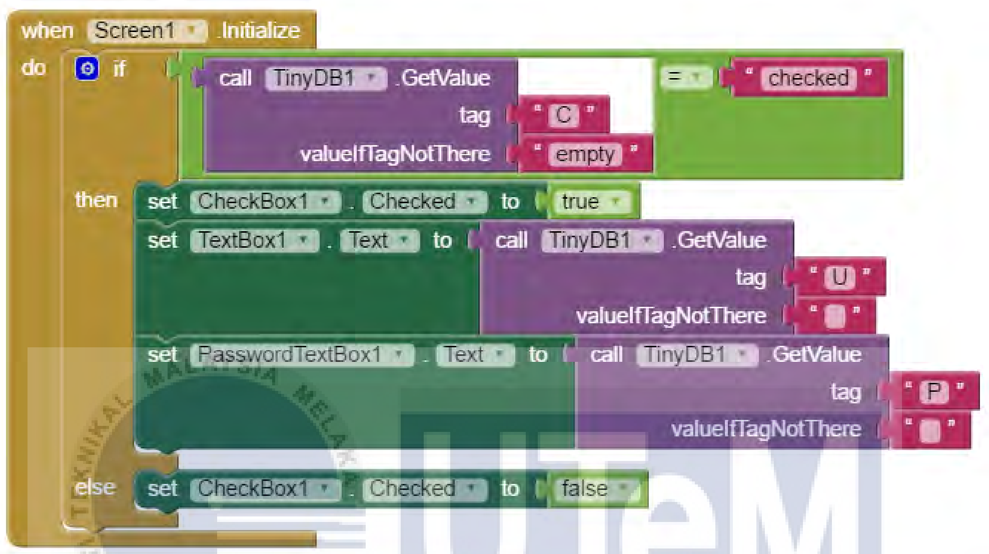
Here you can find all your scenarios previously created. Click on Manage to add actions.

Scenario name	DeviceID	
AllSwitch_OFF	v926246B8C369473	Test Manage Delete
AllSwitch_ON	v94BF87EE3A165B8	Test Manage Delete
Switch1_OFF	vD5FFA24223C46F7	Test Manage Delete
Switch1_ON	v21FFE94118564EB	Test Manage Delete
Switch2_OFF	v9FFF1519A28B489	Test Manage Delete
Switch2_ON	v6F87D1EA9C22D96	Test Manage Delete
Switch3_OFF	vC3E625862BB2414	Test Manage Delete
Switch3_ON	v4F961DAD290F5A0	Test Manage Delete
Switch4_OFF	v99FC54015F911AA	Test Manage Delete
Switch4_ON	v4D27734FA0DB804	Test Manage Delete

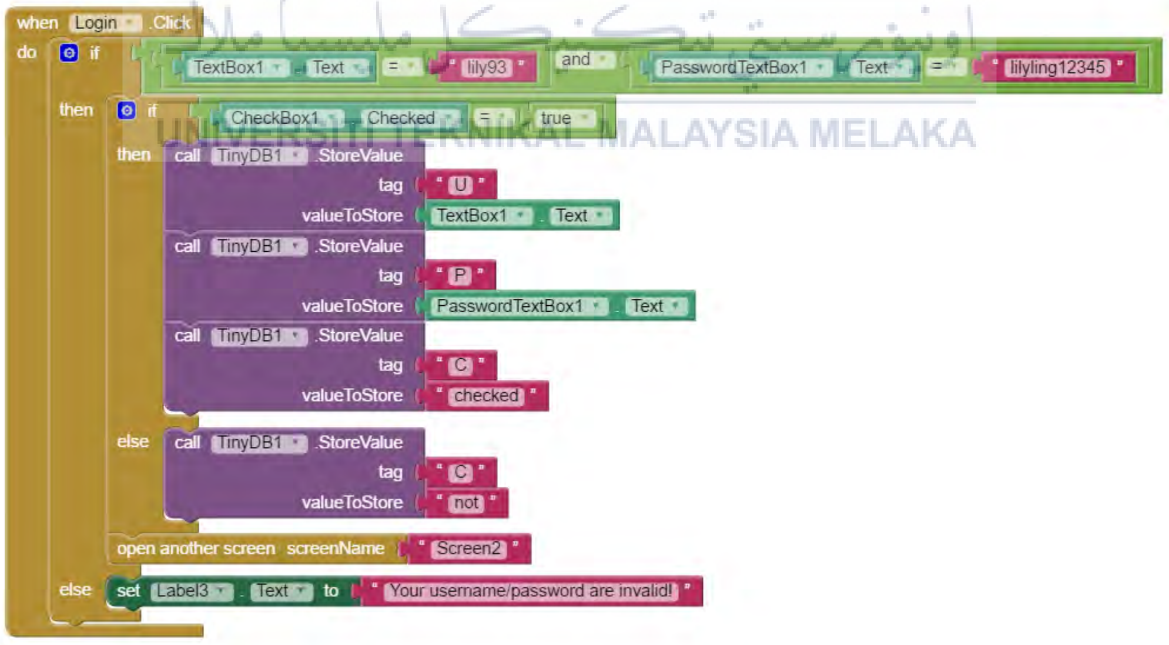
APPENDIX C

MIT App Inventor 2

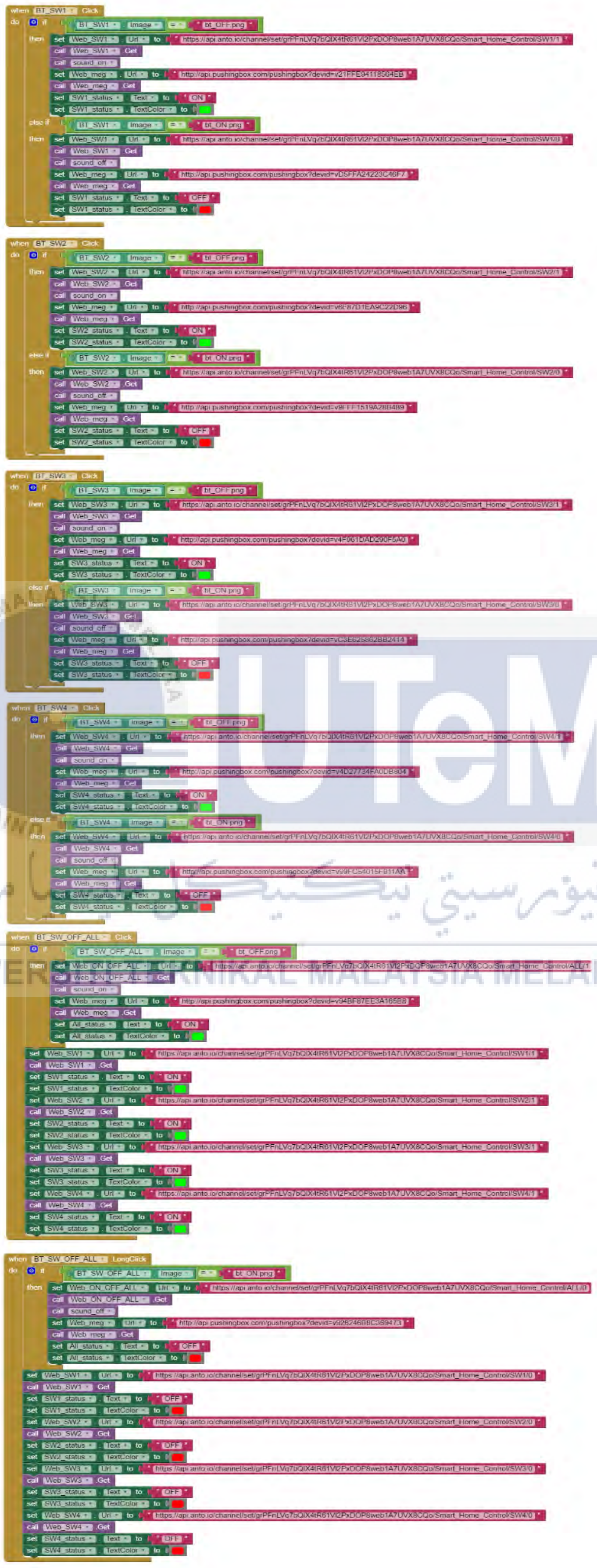
The following are the screenshots of the Blocks diagram.



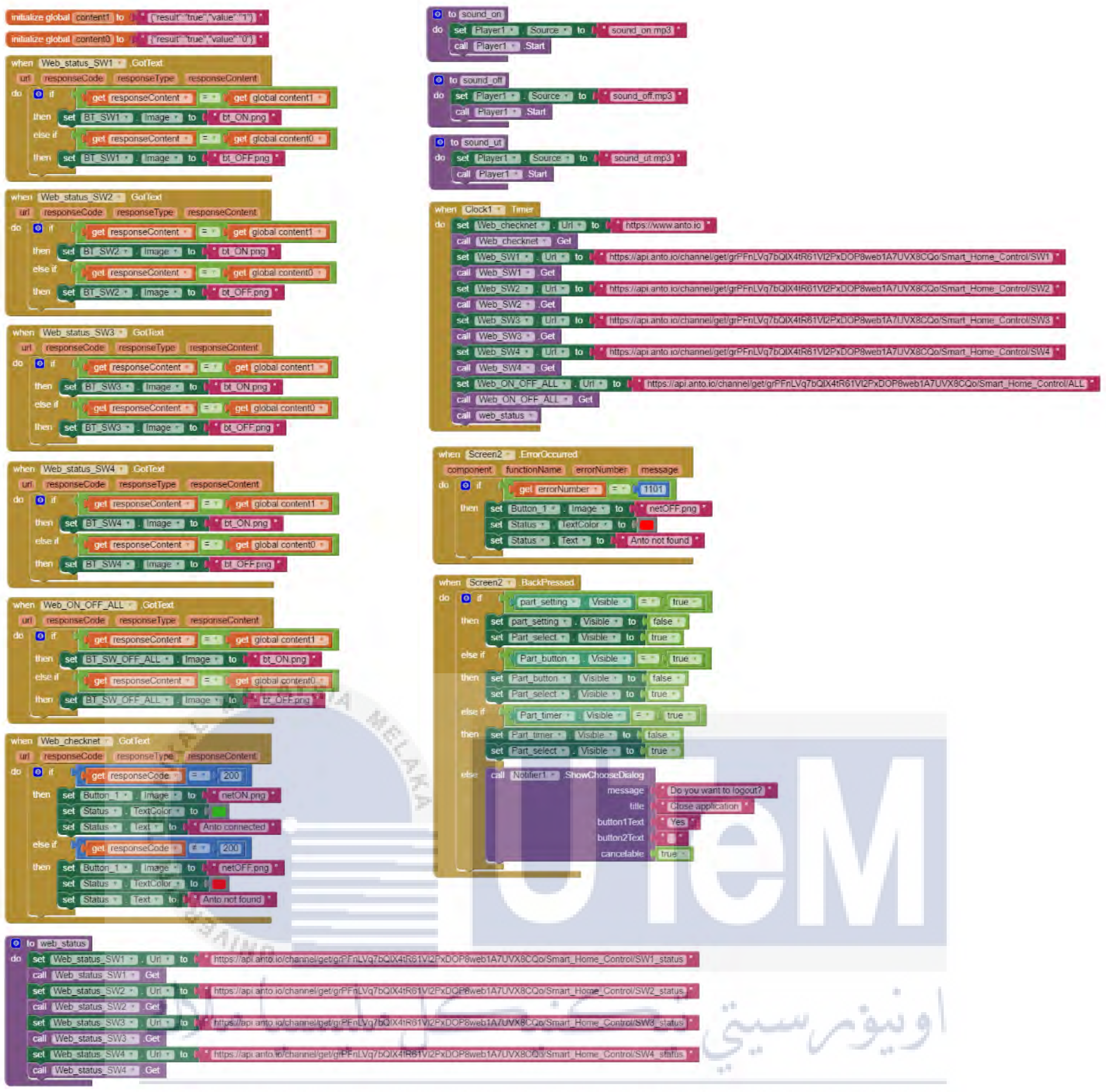
```
when Screen1 .Initialize
do  if  call TinyDB1 .GetValue
            tag " C "
            valueIfTagNotThere " empty "
        = " checked "
    then  set CheckBox1 . Checked to true
          set TextBox1 . Text to  call TinyDB1 .GetValue
                                      tag " U "
                                      valueIfTagNotThere " "
          set PasswordTextBox1 . Text to  call TinyDB1 .GetValue
                                              tag " P "
                                              valueIfTagNotThere " "
    else  set CheckBox1 . Checked to false
```



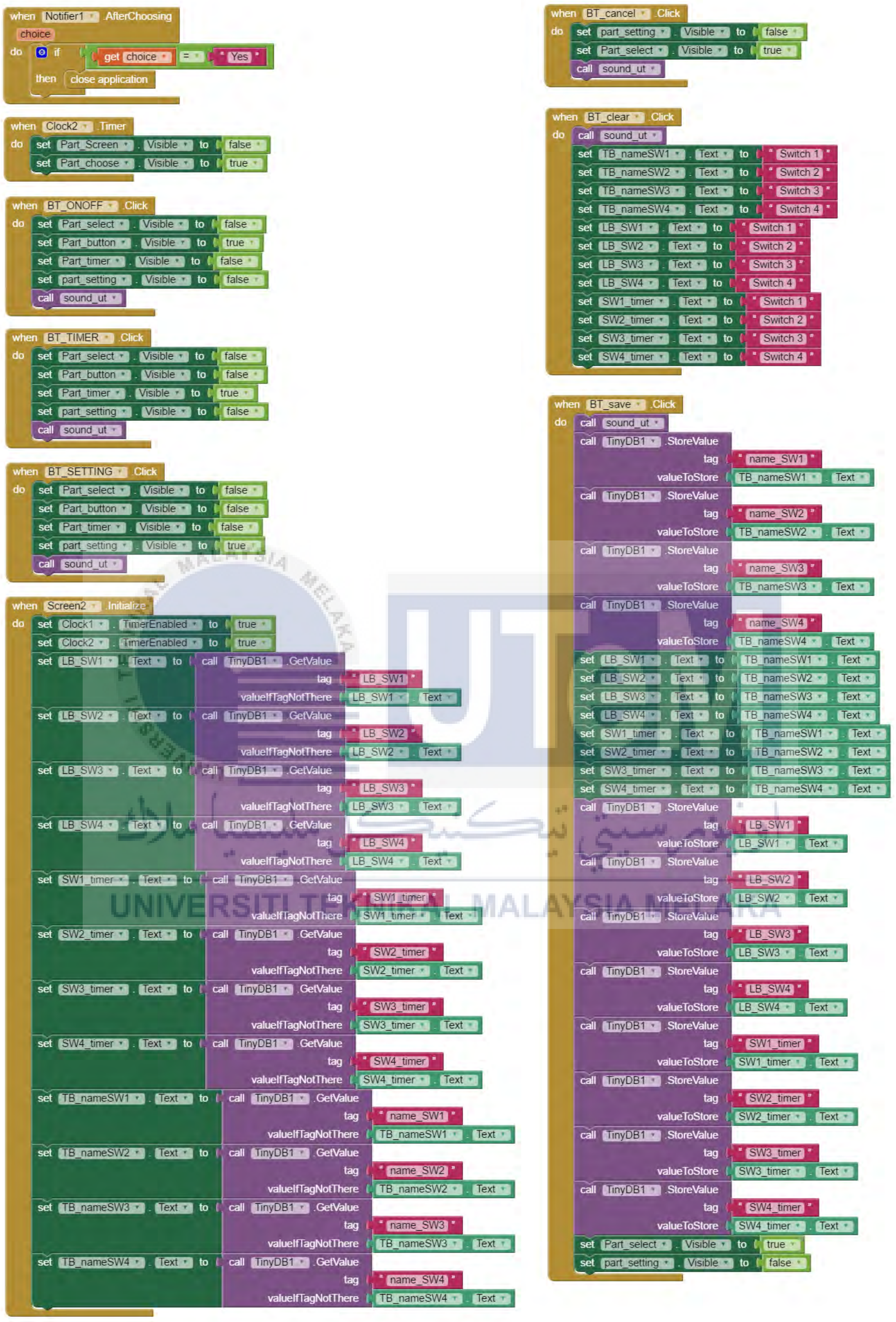
```
when Login .Click
do  if  TextBox1 . Text = " lily93 "  and  PasswordTextBox1 . Text = " lilyling12345 "
    then  if  CheckBox1 . Checked = true
          then  call TinyDB1 .StoreValue
                    tag " U "
                    valueToStore TextBox1 . Text
                call TinyDB1 .StoreValue
                    tag " P "
                    valueToStore PasswordTextBox1 . Text
                call TinyDB1 .StoreValue
                    tag " C "
                    valueToStore " checked "
          else  call TinyDB1 .StoreValue
                    tag " C "
                    valueToStore " not "
          open another screen  screenName " Screen2 "
    else  set Label3 . Text to " Your username/password are invalid! "
```



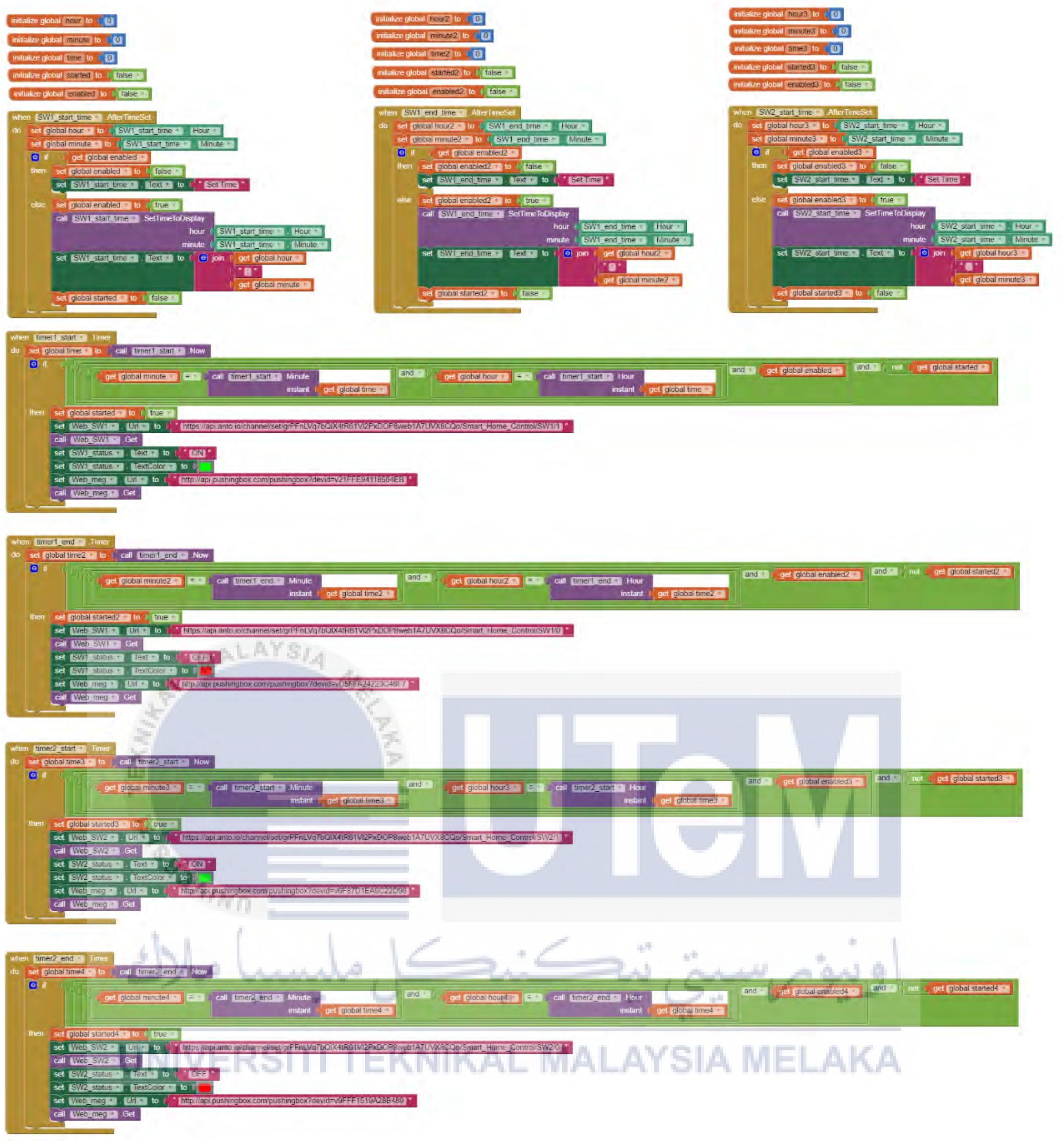
```
when BT_SW1 .Click
do  if  BT_SW1 . Image = "bt_OFF.png"
    then  set Web_SW1 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1/1"
          call Web_SW1 .Get
          call sound_on
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=v21FFE94118564EB"
          call Web_meg .Get
          set SW1_status . Text to "ON"
          set SW1_status . TextColor to 🟩
    else if  BT_SW1 . Image = "bt_ON.png"
    then  set Web_SW1 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1/0"
          call Web_SW1 .Get
          call sound_off
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=vD5FFA24223C46F7"
          call Web_meg .Get
          set SW1_status . Text to "OFF"
          set SW1_status . TextColor to 🟥

when BT_SW2 .Click
do  if  BT_SW2 . Image = "bt_OFF.png"
    then  set Web_SW2 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW2/1"
          call Web_SW2 .Get
          call sound_on
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=v6F87D1EA9C22D96"
          call Web_meg .Get
          set SW2_status . Text to "ON"
          set SW2_status . TextColor to 🟩
    else if  BT_SW2 . Image = "bt_ON.png"
    then  set Web_SW2 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW2/0"
          call Web_SW2 .Get
          call sound_off
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=v8FFF1519A28B489"
          call Web_meg .Get
          set SW2_status . Text to "OFF"
          set SW2_status . TextColor to 🟥

when BT_SW3 .Click
do  if  BT_SW3 . Image = "bt_OFF.png"
    then  set Web_SW3 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW3/1"
          call Web_SW3 .Get
          call sound_on
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=v4F961DAD290F5A0"
          call Web_meg .Get
          set SW3_status . Text to "ON"
          set SW3_status . TextColor to 🟩
    else if  BT_SW3 . Image = "bt_ON.png"
    then  set Web_SW3 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW3/0"
          call Web_SW3 .Get
          call sound_off
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=vC3E625862BB2414"
          call Web_meg .Get
          set SW3_status . Text to "OFF"
          set SW3_status . TextColor to 🟥

when BT_SW4 .Click
do  if  BT_SW4 . Image = "bt_OFF.png"
    then  set Web_SW4 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW4/1"
          call Web_SW4 .Get
          call sound_on
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=v4D27734FA0DB804"
          call Web_meg .Get
          set SW4_status . Text to "ON"
          set SW4_status . TextColor to 🟩
    else if  BT_SW4 . Image = "bt_ON.png"
    then  set Web_SW4 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW4/0"
          call Web_SW4 .Get
          call sound_off
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=v99FC54015F811AA"
          call Web_meg .Get
          set SW4_status . Text to "OFF"
          set SW4_status . TextColor to 🟥

when BT_SW_OFF_ALL .Click
do  if  BT_SW_OFF_ALL . Image = "bt_OFF.png"
    then  set Web_ON_OFF_ALL . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/ALL/1"
          call Web_ON_OFF_ALL .Get
          call sound_on
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=v94BF87EE3A165B8"
          call Web_meg .Get
          set All_status . Text to "ON"
          set All_status . TextColor to 🟩
    set Web_SW1 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1/1"
    call Web_SW1 .Get
    set SW1_status . Text to "ON"
    set SW1_status . TextColor to 🟩
    set Web_SW2 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW2/1"
    call Web_SW2 .Get
    set SW2_status . Text to "ON"
    set SW2_status . TextColor to 🟩
    set Web_SW3 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW3/1"
    call Web_SW3 .Get
    set SW3_status . Text to "ON"
    set SW3_status . TextColor to 🟩
    set Web_SW4 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW4/1"
    call Web_SW4 .Get
    set SW4_status . Text to "ON"
    set SW4_status . TextColor to 🟩

when BT_SW_OFF_ALL .LongClick
do  if  BT_SW_OFF_ALL . Image = "bt_ON.png"
    then  set Web_ON_OFF_ALL . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/ALL/0"
          call Web_ON_OFF_ALL .Get
          call sound_off
          set Web_meg . Url to "http://api.pushingbox.com/pushingbox?devid=v926246B8C369473"
          call Web_meg .Get
          set All_status . Text to "OFF"
          set All_status . TextColor to 🟥
    set Web_SW1 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1/0"
    call Web_SW1 .Get
    set SW1_status . Text to "OFF"
    set SW1_status . TextColor to 🟥
    set Web_SW2 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW2/0"
    call Web_SW2 .Get
    set SW2_status . Text to "OFF"
    set SW2_status . TextColor to 🟥
    set Web_SW3 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW3/0"
    call Web_SW3 .Get
    set SW3_status . Text to "OFF"
    set SW3_status . TextColor to 🟥
    set Web_SW4 . Url to "https://api.anto.io/channel/set/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW4/0"
    call Web_SW4 .Get
    set SW4_status . Text to "OFF"
    set SW4_status . TextColor to 🟥
```



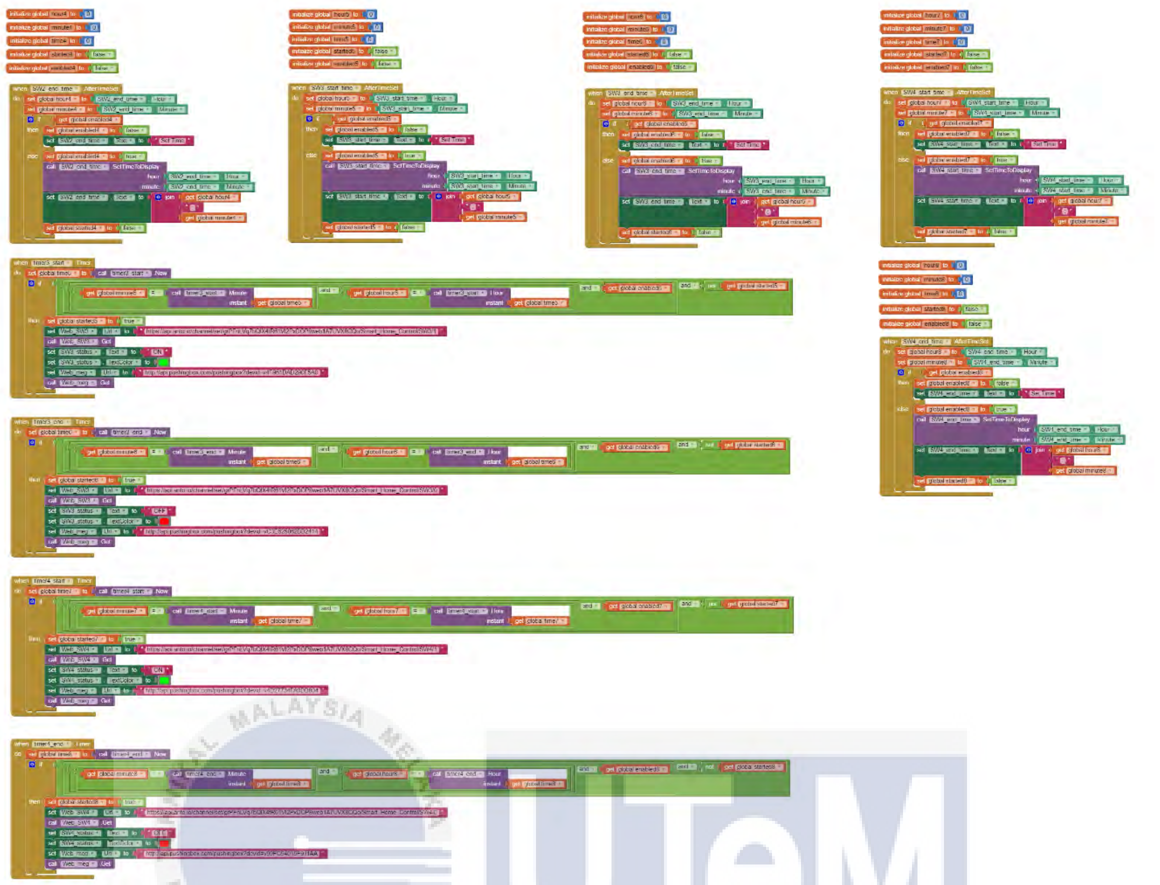
```
initialize global content1 to " {"result":"true","value":"1"} "
initialize global content0 to " {"result":"true","value":"0"} "

when Web_status_SW1 .GotText
  url  responseCode  responseType  responseContent
do  if      get responseContent = get global content1
    then    set BT_SW1 . Image to " bt_ON.png "
    else if get responseContent = get global content0
    then    set BT_SW1 . Image to " bt_OFF.png "

when Web_status_SW2 .GotText
  url  responseCode  responseType  responseContent
do  if      get responseContent = get global content1
    then    set BT_SW2 . Image to " bt_ON.png "
    else if get responseContent = get global content0
    then    set BT_SW2 . Image to " bt_OFF.png "

when Web_status_SW3 .GotText
  url  responseCode  responseType  responseContent
do  if      get responseContent = get global content1
    then    set BT_SW3 . Image to " bt_ON.png "
    else if get responseContent = get global content0
    then    set BT_SW3 . Image to " bt_OFF.png "

when Web_status_SW4 .GotText
  url  responseCode  responseType  responseContent
do  if      get responseContent = get global content1
    then    set BT_SW4 . Image to " bt_ON.png "
    else if get responseContent = get global content0
    then    set BT_SW4 . Image to " bt_OFF.png "

when Web_ON_OFF_ALL .GotText
  url  responseCode  responseType  responseContent
do  if      get responseContent = get global content1
    then    set BT_SW_OFF_ALL . Image to " bt_ON.png "
    else if get responseContent = get global content0
    then    set BT_SW_OFF_ALL . Image to " bt_OFF.png "

when Web_checknet .GotText
  url  responseCode  responseType  responseContent
do  if      get responseCode = 200
    then    set Button_1 . Image to " netON.png "
            set Status . TextColor to 🟩
            set Status . Text to " Anto connected "
    else if get responseCode ≠ 200
    then    set Button_1 . Image to " netOFF.png "
            set Status . TextColor to 🟥
            set Status . Text to " Anto not found "

to web_status
do  set Web_status_SW1 . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1_status "
    call Web_status_SW1 .Get
    set Web_status_SW2 . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW2_status "
    call Web_status_SW2 .Get
    set Web_status_SW3 . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW3_status "
    call Web_status_SW3 .Get
    set Web_status_SW4 . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW4_status "
    call Web_status_SW4 .Get

to sound_on
do  set Player1 . Source to " sound_on.mp3 "
    call Player1 .Start

to sound_off
do  set Player1 . Source to " sound_off.mp3 "
    call Player1 .Start

to sound_ut
do  set Player1 . Source to " sound_ut.mp3 "
    call Player1 .Start

when Clock1 .Timer
do  set Web_checknet . Url to " https://www.anto.io "
    call Web_checknet .Get
    set Web_SW1 . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1 "
    call Web_SW1 .Get
    set Web_SW2 . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW2 "
    call Web_SW2 .Get
    set Web_SW3 . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW3 "
    call Web_SW3 .Get
    set Web_SW4 . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW4 "
    call Web_SW4 .Get
    set Web_ON_OFF_ALL . Url to " https://api.anto.io/channel/get/grPFnLVq7bQIX4tR61VI2PxDOP8web1A7UVX8CQo/Smart_Home_Control/ALL "
    call Web_ON_OFF_ALL .Get
    call web_status

when Screen2 .ErrorOccurred
  component  functionName  errorNumber  message
do  if      get errorNumber = 1101
    then    set Button_1 . Image to " netOFF.png "
            set Status . TextColor to 🟥
            set Status . Text to " Anto not found "

when Screen2 .BackPressed
do  if      part_setting . Visible = true
    then    set part_setting . Visible to false
            set Part_select . Visible to true
    else if Part_button . Visible = true
    then    set Part_button . Visible to false
            set Part_select . Visible to true
    else if Part_timer . Visible = true
    then    set Part_timer . Visible to false
            set Part_select . Visible to true
    else    call Notifier1 .ShowChooseDialog
                 message      " Do you want to logout? "
                 title        " Close application "
                 button1Text  " Yes "
                 button2Text  " "
                 cancelable   true
```



```
when Notifier1 .AfterChoosing
  choice
do  if    get choice = "Yes"
    then  close application

when Clock2 .Timer
do  set Part_Screen . Visible to false
    set Part_choose . Visible to true

when BT_ONOFF .Click
do  set Part_select . Visible to false
    set Part_button . Visible to true
    set Part_timer . Visible to false
    set part_setting . Visible to false
    call sound_ut

when BT_TIMER .Click
do  set Part_select . Visible to false
    set Part_button . Visible to false
    set Part_timer . Visible to true
    set part_setting . Visible to false
    call sound_ut

when BT_SETTING .Click
do  set Part_select . Visible to false
    set Part_button . Visible to false
    set Part_timer . Visible to false
    set part_setting . Visible to true
    call sound_ut

when Screen2 .Initialize
do  set Clock1 . TimerEnabled to true
    set Clock2 . TimerEnabled to true
    set LB_SW1 . Text to call TinyDB1 .GetValue
                                  tag "LB_SW1"
                                  valueIfTagNotThere LB_SW1 . Text
    set LB_SW2 . Text to call TinyDB1 .GetValue
                                  tag "LB_SW2"
                                  valueIfTagNotThere LB_SW2 . Text
    set LB_SW3 . Text to call TinyDB1 .GetValue
                                  tag "LB_SW3"
                                  valueIfTagNotThere LB_SW3 . Text
    set LB_SW4 . Text to call TinyDB1 .GetValue
                                  tag "LB_SW4"
                                  valueIfTagNotThere LB_SW4 . Text
    set SW1_timer . Text to call TinyDB1 .GetValue
                                  tag "SW1_timer"
                                  valueIfTagNotThere SW1_timer . Text
    set SW2_timer . Text to call TinyDB1 .GetValue
                                  tag "SW2_timer"
                                  valueIfTagNotThere SW2_timer . Text
    set SW3_timer . Text to call TinyDB1 .GetValue
                                  tag "SW3_timer"
                                  valueIfTagNotThere SW3_timer . Text
    set SW4_timer . Text to call TinyDB1 .GetValue
                                  tag "SW4_timer"
                                  valueIfTagNotThere SW4_timer . Text
    set TB_nameSW1 . Text to call TinyDB1 .GetValue
                                  tag "name_SW1"
                                  valueIfTagNotThere TB_nameSW1 . Text
    set TB_nameSW2 . Text to call TinyDB1 .GetValue
                                  tag "name_SW2"
                                  valueIfTagNotThere TB_nameSW2 . Text
    set TB_nameSW3 . Text to call TinyDB1 .GetValue
                                  tag "name_SW3"
                                  valueIfTagNotThere TB_nameSW3 . Text
    set TB_nameSW4 . Text to call TinyDB1 .GetValue
                                  tag "name_SW4"
                                  valueIfTagNotThere TB_nameSW4 . Text

when BT_cancel .Click
do  set part_setting . Visible to false
    set Part_select . Visible to true
    call sound_ut

when BT_clear .Click
do  call sound_ut
    set TB_nameSW1 . Text to "Switch 1"
    set TB_nameSW2 . Text to "Switch 2"
    set TB_nameSW3 . Text to "Switch 3"
    set TB_nameSW4 . Text to "Switch 4"
    set LB_SW1 . Text to "Switch 1"
    set LB_SW2 . Text to "Switch 2"
    set LB_SW3 . Text to "Switch 3"
    set LB_SW4 . Text to "Switch 4"
    set SW1_timer . Text to "Switch 1"
    set SW2_timer . Text to "Switch 2"
    set SW3_timer . Text to "Switch 3"
    set SW4_timer . Text to "Switch 4"

when BT_save .Click
do  call sound_ut
    call TinyDB1 .StoreValue
                  tag "name_SW1"
                  valueToStore TB_nameSW1 . Text
    call TinyDB1 .StoreValue
                  tag "name_SW2"
                  valueToStore TB_nameSW2 . Text
    call TinyDB1 .StoreValue
                  tag "name_SW3"
                  valueToStore TB_nameSW3 . Text
    call TinyDB1 .StoreValue
                  tag "name_SW4"
                  valueToStore TB_nameSW4 . Text
    set LB_SW1 . Text to TB_nameSW1 . Text
    set LB_SW2 . Text to TB_nameSW2 . Text
    set LB_SW3 . Text to TB_nameSW3 . Text
    set LB_SW4 . Text to TB_nameSW4 . Text
    set SW1_timer . Text to TB_nameSW1 . Text
    set SW2_timer . Text to TB_nameSW2 . Text
    set SW3_timer . Text to TB_nameSW3 . Text
    set SW4_timer . Text to TB_nameSW4 . Text
    call TinyDB1 .StoreValue
                  tag "LB_SW1"
                  valueToStore LB_SW1 . Text
    call TinyDB1 .StoreValue
                  tag "LB_SW2"
                  valueToStore LB_SW2 . Text
    call TinyDB1 .StoreValue
                  tag "LB_SW3"
                  valueToStore LB_SW3 . Text
    call TinyDB1 .StoreValue
                  tag "LB_SW4"
                  valueToStore LB_SW4 . Text
    call TinyDB1 .StoreValue
                  tag "SW1_timer"
                  valueToStore SW1_timer . Text
    call TinyDB1 .StoreValue
                  tag "SW2_timer"
                  valueToStore SW2_timer . Text
    call TinyDB1 .StoreValue
                  tag "SW3_timer"
                  valueToStore SW3_timer . Text
    call TinyDB1 .StoreValue
                  tag "SW4_timer"
                  valueToStore SW4_timer . Text
    set Part_select . Visible to true
    set part_setting . Visible to false
```



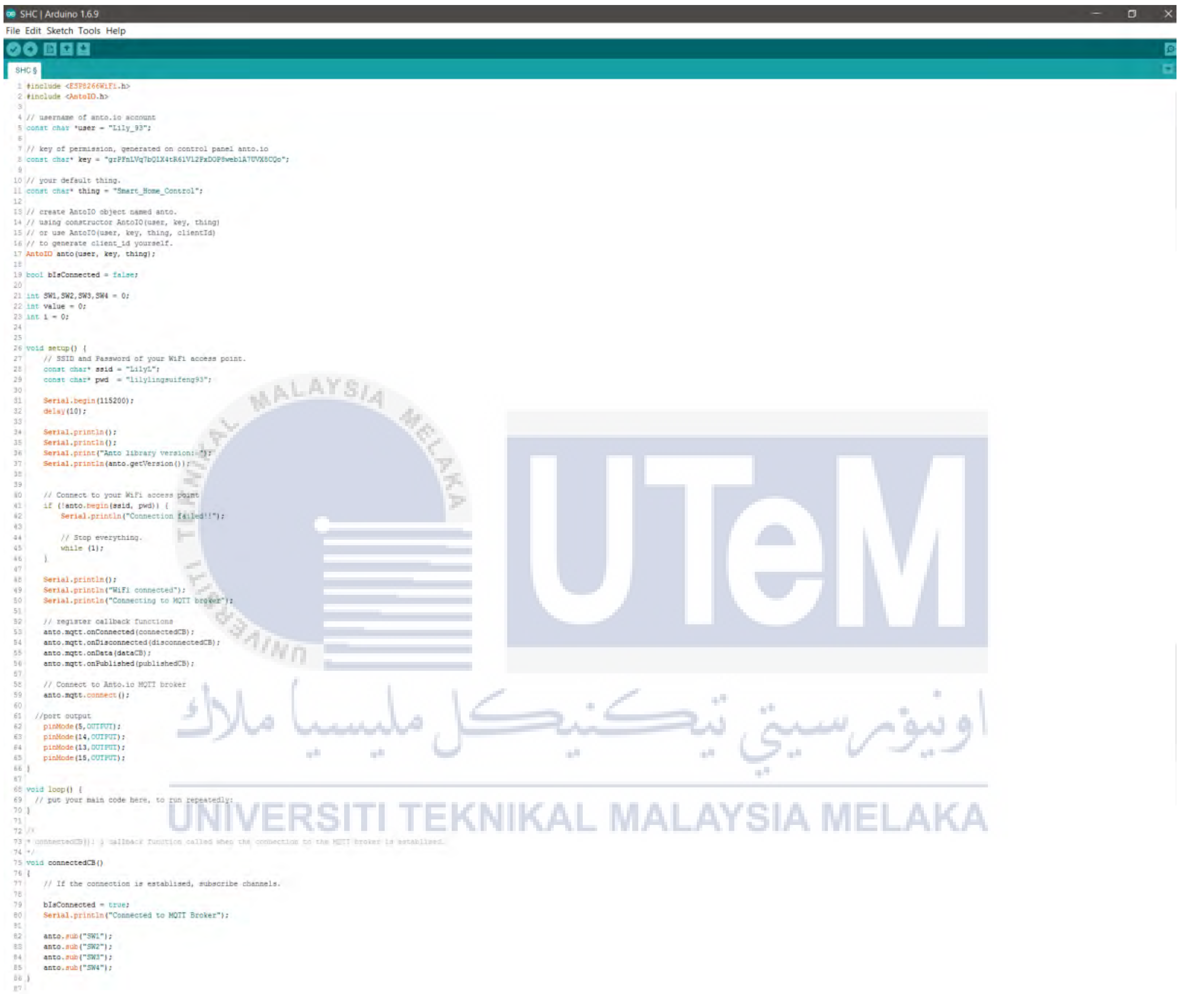
```
initialize global hour to 0
initialize global minute to 0
initialize global time to 0
initialize global started to false
initialize global enabled to false

when SW1_start_time .AfterTimeSet
do  set global hour to SW1_start_time . Hour
    set global minute to SW1_start_time . Minute
    if   get global enabled
    then set global enabled to false
         set SW1_start_time . Text to " Set Time "
    else set global enabled to true
         call SW1_start_time .SetTimeToDisplay
                                   hour   SW1_start_time . Hour
                                   minute SW1_start_time . Minute
         set SW1_start_time . Text to join  get global hour
                                            " : "
                                            get global minute
         set global started to false

initialize global hour2 to 0
initialize global minute2 to 0
initialize global time2 to 0
initialize global started2 to false
initialize global enabled2 to false

when SW1_end_time .AfterTimeSet
do  set global hour2 to SW1_end_time . Hour
    set global minute2 to SW1_end_time . Minute
    if   get global enabled2
    then set global enabled2 to false
         set SW1_end_time . Text to " Set Time "
    else set global enabled2 to true
         call SW1_end_time .SetTimeToDisplay
                                   hour   SW1_end_time . Hour
                                   minute SW1_end_time . Minute
         set SW1_end_time . Text to join  get global hour2
                                          " : "
                                          get global minute2
         set global started2 to false

initialize global hour3 to 0
initialize global minute3 to 0
initialize global time3 to 0
initialize global started3 to false
initialize global enabled3 to false

when SW2_start_time .AfterTimeSet
do  set global hour3 to SW2_start_time . Hour
    set global minute3 to SW2_start_time . Minute
    if   get global enabled3
    then set global enabled3 to false
         set SW2_start_time . Text to " Set Time "
    else set global enabled3 to true
         call SW2_start_time .SetTimeToDisplay
                                   hour   SW2_start_time . Hour
                                   minute SW2_start_time . Minute
         set SW2_start_time . Text to join  get global hour3
                                            " : "
                                            get global minute3
         set global started3 to false

when timer1_start .Timer
do  set global time to call timer1_start .Now
    if   get global minute = call timer1_start .Minute instant get global time
         and get global hour = call timer1_start .Hour instant get global time
         and get global enabled
         and not get global started
    then set global started to true
         set Web_SW1 . Url to " https://api.anto.io/channel/set/grPFnLVq7bQlX4tR61Vl2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1/1 "
         call Web_SW1 .Get
         set SW1_status . Text to " ON "
         set SW1_status . TextColor to 🟩
         set Web_meg . Url to " http://api.pushingbox.com/pushingbox?devid=v21FFE94118564EB "
         call Web_meg .Get

when timer1_end .Timer
do  set global time2 to call timer1_end .Now
    if   get global minute2 = call timer1_end .Minute instant get global time2
         and get global hour2 = call timer1_end .Hour instant get global time2
         and get global enabled2
         and not get global started2
    then set global started2 to true
         set Web_SW1 . Url to " https://api.anto.io/channel/set/grPFnLVq7bQlX4tR61Vl2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW1/0 "
         call Web_SW1 .Get
         set SW1_status . Text to " OFF "
         set SW1_status . TextColor to 🟥
         set Web_meg . Url to " http://api.pushingbox.com/pushingbox?devid=vD5FFA24223C46F7 "
         call Web_meg .Get

when timer2_start .Timer
do  set global time3 to call timer2_start .Now
    if   get global minute3 = call timer2_start .Minute instant get global time3
         and get global hour3 = call timer2_start .Hour instant get global time3
         and get global enabled3
         and not get global started3
    then set global started3 to true
         set Web_SW2 . Url to " https://api.anto.io/channel/set/grPFnLVq7bQlX4tR61Vl2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW2/1 "
         call Web_SW2 .Get
         set SW2_status . Text to " ON "
         set SW2_status . TextColor to 🟩
         set Web_meg . Url to " http://api.pushingbox.com/pushingbox?devid=v6F87D1EA8C22D96 "
         call Web_meg .Get

when timer2_end .Timer
do  set global time4 to call timer2_end .Now
    if   get global minute4 = call timer2_end .Minute instant get global time4
         and get global hour4 = call timer2_end .Hour instant get global time4
         and get global enabled4
         and not get global started4
    then set global started4 to true
         set Web_SW2 . Url to " https://api.anto.io/channel/set/grPFnLVq7bQlX4tR61Vl2PxDOP8web1A7UVX8CQo/Smart_Home_Control/SW2/0 "
         call Web_SW2 .Get
         set SW2_status . Text to " OFF "
         set SW2_status . TextColor to 🟥
         set Web_meg . Url to " http://api.pushingbox.com/pushingbox?devid=v9FFF1519A28B489 "
         call Web_meg .Get
```



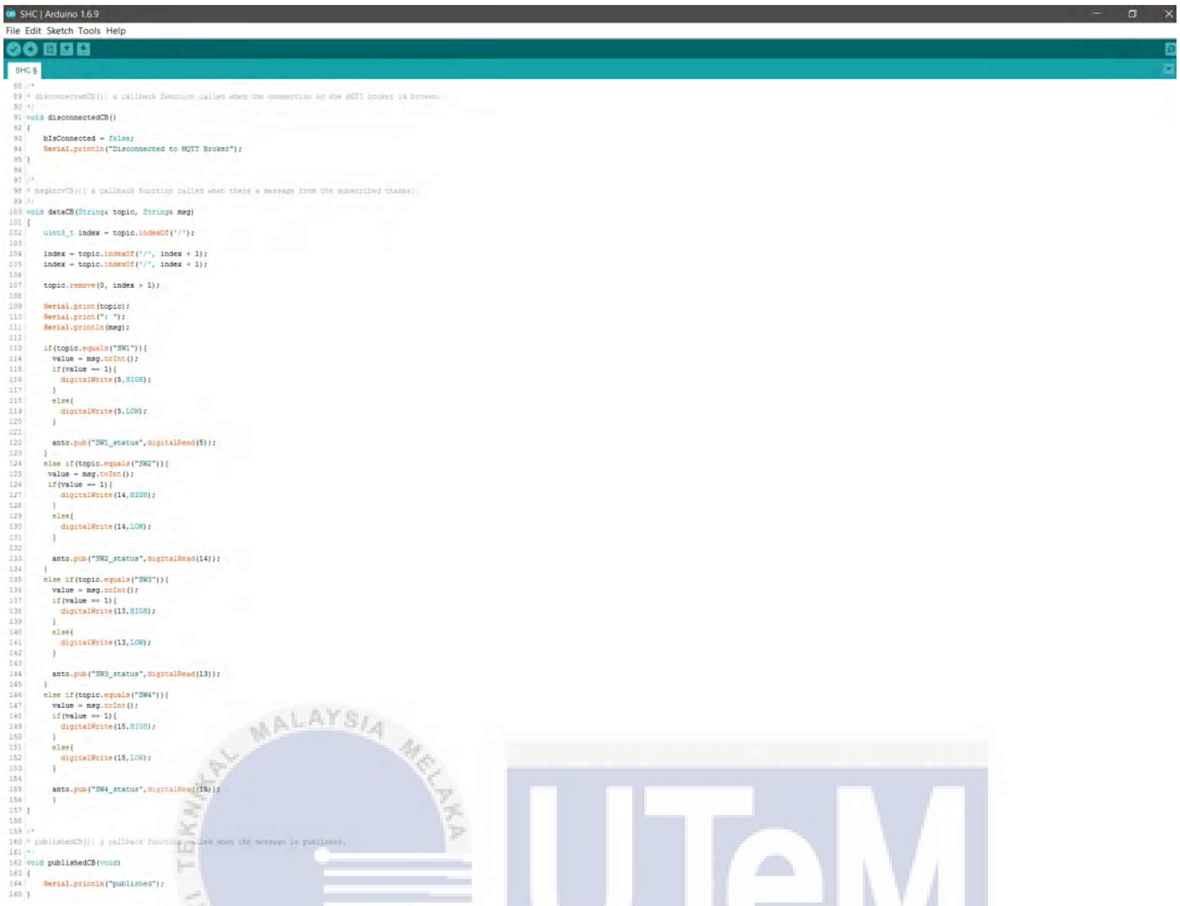
APPENDIX D

NodeMCU programming codes

The following are the screenshot of the NodeMCU programming codes in Arduino IDE.



```
#include <ESP8266WiFi.h>
#include <AntoIO.h>

// username of anto.io account
const char *user = "Lily_93";

// key of permission, generated on control panel anto.io
const char* key = "grPFnLVq7bQ1X4tR61V12PxDOP8web1A7UVX8CQo";

// your default thing.
const char* thing = "Smart_Home_Control";

// create AntoIO object named anto.
// using constructor AntoIO(user, key, thing)
// or use AntoIO(user, key, thing, clientId)
// to generate client_id yourself.
AntoIO anto(user, key, thing);

bool bIsConnected = false;

int SW1,SW2,SW3,SW4 = 0;
int value = 0;
int i = 0;


void setup() {
    // SSID and Password of your WiFi access point.
    const char* ssid = "LilyL";
    const char* pwd  = "lilylingsuifeng93";

    Serial.begin(115200);
    delay(10);

    Serial.println();
    Serial.println();
    Serial.print("Anto library version: ");
    Serial.println(anto.getVersion());


    // Connect to your WiFi access point
    if (!anto.begin(ssid, pwd)) {
        Serial.println("Connection failed!!");

        // Stop everything.
        while (1);
    }

    Serial.println();
    Serial.println("WiFi connected");
    Serial.println("Connecting to MQTT broker");

    // register callback functions
    anto.mqtt.onConnected(connectedCB);
    anto.mqtt.onDisconnected(disconnectedCB);
    anto.mqtt.onData(dataCB);
    anto.mqtt.onPublished(publishedCB);

    // Connect to Anto.io MQTT broker
    anto.mqtt.connect();

  //port output
    pinMode(5,OUTPUT);
    pinMode(14,OUTPUT);
    pinMode(13,OUTPUT);
    pinMode(15,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
}

/*
* connectedCB(): a callback function called when the connection to the MQTT broker is establised.
*/
void connectedCB()
{
    // If the connection is establised, subscribe channels.

    bIsConnected = true;
    Serial.println("Connected to MQTT Broker");

    anto.sub("SW1");
    anto.sub("SW2");
    anto.sub("SW3");
    anto.sub("SW4");
}
```



```
/*
* disconnectedCB(): a callback function called when the connection to the MQTT broker is broken.
*/
void disconnectedCB()
{
    bIsConnected = false;
    Serial.println("Disconnected to MQTT Broker");
}

/*
* msgArrvCB(): a callback function called when there a message from the subscribed channel.
*/
void dataCB(String& topic, String& msg)
{
    uint8_t index = topic.indexOf('/');

    index = topic.indexOf('/', index + 1);
    index = topic.indexOf('/', index + 1);

    topic.remove(0, index + 1);

    Serial.print(topic);
    Serial.print(": ");
    Serial.println(msg);

    if(topic.equals("SW1")){
      value = msg.toInt();
      if(value == 1){
        digitalWrite(5,HIGH);
      }
      else{
        digitalWrite(5,LOW);
      }

      anto.pub("SW1_status",digitalRead(5));
    }
    else if(topic.equals("SW2")){
     value = msg.toInt();
     if(value == 1){
        digitalWrite(14,HIGH);
      }
      else{
        digitalWrite(14,LOW);
      }

      anto.pub("SW2_status",digitalRead(14));
    }
    else if(topic.equals("SW3")){
      value = msg.toInt();
      if(value == 1){
        digitalWrite(13,HIGH);
      }
      else{
        digitalWrite(13,LOW);
      }

      anto.pub("SW3_status",digitalRead(13));
    }
    else if(topic.equals("SW4")){
      value = msg.toInt();
      if(value == 1){
        digitalWrite(15,HIGH);
      }
      else{
        digitalWrite(15,LOW);
      }

      anto.pub("SW4_status",digitalRead(15));
      }
}

/*
* publishedCB(): a callback function called when the message is published.
*/
void publishedCB(void)
{
    Serial.println("published");
}
```