

IOT DATA LOGGER SYSTEM FOR SNMP DATA MONITORING

IQBAL HAZIQ BIN AZMI



UNIVERSITI TEKNIKAL MALAYSIA MELAKA

IOT DATA LOGGER SYSTEM FOR SNMP DATA MONITORING

IQBAL HAZIQ BIN AZMI

**This report is submitted in partial fulfilment of the requirements
for the degree of Bachelor of Electronic Engineering with Honours**

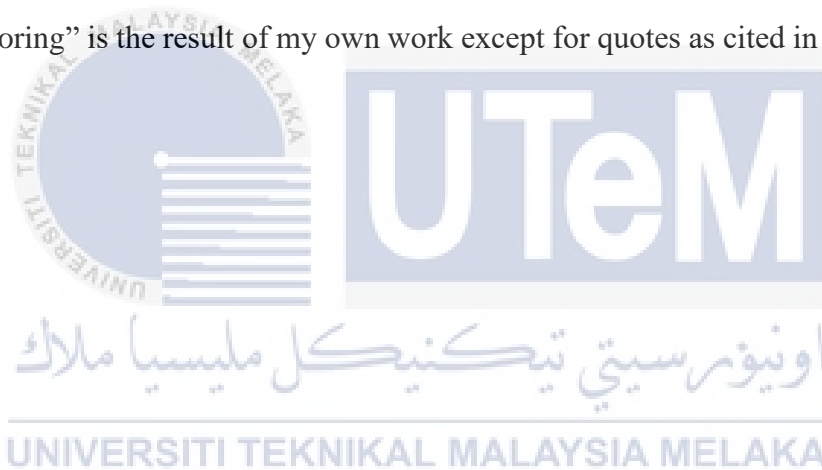


اونيورستى تېكنىكل ماليسيا ملاك
Faculty of Electronic and Computer Engineering
Universiti Teknikal Malaysia Melaka

2021

DECLARATION

I declare that this report entitled “IoT Data Logger System for SNMP Data Monitoring” is the result of my own work except for quotes as cited in the references.



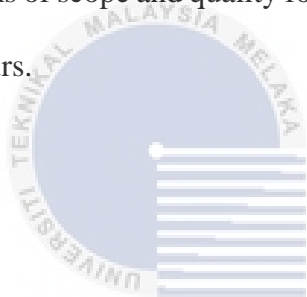
Signature :

Author : IQBAL HAZIQ BIN AZMI

Date : 25 JUNE 2021

APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours.



اونيورسيتي تيكنيكل مليسيا ملاك

Signature :

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Supervisor Name : IR. DR. ANAS BIN ABDUL LATIFF

Date : 25 JUNE 2021

ABSTRACT

The aim of this project is to develop demonstrate IoT Data Logger System by incorporating using NodeMCU ESP32 and Raspberry Pi 3B with cayenne platform as a dashboard. Most of the work involves hardware (electronic and electrical) and programming. The developed system should be capable to extract a data from SNMP (Simple Network Management Protocol) and appear in a cayenne dashboard. Then, recorded all variables. This project is under a collaboration project with broadcaster industry towards 4th Industry Revolution and achieving UN Sustainable Development Goals.

ABSTRAK

Tujuan projek ini adalah untuk mengembangkan Sistem Logger Data IoT dengan menggabungkan menggunakan NodeMCU ESP32 dan Raspberry Pi 3B dengan platform cayenne sebagai papan pemuka. Sebilangan besar kerja melibatkan perkakasan (elektronik dan elektrik) dan pengaturcaraan. Sistem yang dibangunkan harus dapat mengekstrak data dari SNMP (Simple Network Management Protocol) dan muncul di papan pemuka cayenne. Kemudian, merekodkan semua pemboleh ubah. Projek ini berada di bawah projek kolaborasi dengan industri penyar ke arah Revolusi Industri ke-4 dan mencapai Matlamat Pembangunan Lestari PBB.

ACKNOWLEDGEMENTS

Foremost, I am feeling grateful and thankful to complete my final year report for two semesters in my final year of Bachelor of Electronic and Computer Engineering successfully. I would like to express my sincere gratitude to my final year project supervisor, Ir. Dr. Anas Bin Abdul Latiff. I really appreciate his guidance from initial to the final level that facilitated me to develop an understanding for this final project. All your guidance will always be remembered.

My sincere thanks also go to all lecturers and members of staff in Faculty of Electronic and Computer Engineering, UTeM, they help me in various way and make my university degree journey pleasant and unforgettable. My thanks go to BENG Section 3 Classmate for their excellent inspiration and support throughout this degree life. The four years experiences with you all will be stored as wonderful moment for me to face the new chapter life as an engineer.

Last but not least, I want to give my full appreciation to my parents. They are my inspiration to keep me motivated to finish this course. Not to forget to thank any person which involved directly or indirectly in helping me to successfully complete this project. I would like to acknowledge their comments and suggestion which was crucial for the successful completion of this study.

TABLE OF CONTENTS

Declaration	
Approval	
Abstract	i
Abstrak	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	ix
List of Tables	xii
List of Symbols and Abbreviations	xiii
List of Appendices	xv
CHAPTER 1 INTRODUCTION	1
1.1 Background of the Project	1
1.2 Problem Statement	2
1.3 Objectives	2
CHAPTER 2 BACKGROUND STUDY	7
2.1 IoT Data Monitoring System	7

2.2	Review Available for Monitoring IoT Data Monitoring System	9
2.2.1	All-in-One Application for Smart Home System	9
2.2.2	Design and Implementation of Server Monitoring System Based on SNMP	11
2.2.3	Design and Implementation of Intelligent Audio Broadcast Monitoring System	13
2.2.4	Prototype Design of Monitoring System Base Tranceiver Station (BTS) Base on Internet of Things	16
CHAPTER 3 METHODOLOGY		18
3.1	Introduction	18
3.2	Flow Chart	19
3.3	Working Flow	21
3.4	Research Hardware	22
3.4.1	NodeMCU-ESP32	22
3.4.2	Raspberry Pi 3B	24
3.4.3	Temperature and Humidity Sensor	26
3.4.4	AC Voltage Sensor	27
3.4.5	AC Current Sensor	28
3.5	Research Software	29
3.5.1	myDevices Cayenne IoT Platform	29
3.5.2	Arduino IDE	29

3.5.3	Raspberry Pi OS	30
3.5.4	fritzing	30
3.5.5	IFTTT	31
3.5.6	Telegram	31
3.6	Software Development	32
3.6.1	Arduino IDE	32
3.6.1.1	Connection between ESP32 and Cayenne Platform	32
3.6.1.2	Temperature and Humidity Sensor Code	33
3.6.1.3	AC Voltage and AC Current Sensor Code	34
3.6.1.4	Power Consumption Source Code	35
3.6.2	Raspberry Pi OS (SNMP System via FTP Server)	36
3.6.3	Cayenne IoT Platform Dashboard	38
3.6.4	Webhook	39
3.6.5	IFTTT	40
3.6.6	Telegram	41
3.7	Hardware Development	42
3.7.1	Project Circuit	42
3.7.2	Project Hardware	43
3.7.3	Project location	44

CHAPTER 4 RESULTS AND DISCUSSION	45
4.1 Introduction	45
4.1.1 Mi Home Temperature and Humidity Sensor	46
4.1.2 Power Monitor Meter	47
4.1.3 TP-Link 16-Port Gigabit Ethernet Switch	48
4.2 Software Data Measurement	48
4.2.1 myDevices Cayenne IoT Platform	49
4.2.2 Arduino IDE	50
4.2.3 Raspberry Pi OS	51
4.3 Data Measurement and Comparison	51
4.3.1 Temperature	52
4.3.2 Humidity	53
4.3.3 AC Voltage	54
4.3.4 AC Current	55
4.3.5 Power Consumption	56
4.3.6 Network Speed	57
4.4 Discussion	58
4.4.1 Circuit Final Configuration	58
4.4.2 Field Test	59
4.4.3 Result	59

CHAPTER 5 CONCLUSION AND FUTURE WORKS	61
5.1 Introduction	61
5.2 United Nation Sustainable Development Goals	62
5.3 Potential Penetration in Local Market	63
5.4 Recommendations	63
REFERENCES	64
APPENDICES	66



LIST OF FIGURES

Figure 1.1: Master Control Room	4
Figure 2.1: All-in-One Application for Smart Home System Circuit Diagram	10
Figure 2.2: The SNMP-based Management Model	11
Figure 2.3: Framework of the system	12
Figure 2.4: Data processing flow of illegal frequency monitoring.	15
Figure 2.5: Hardware Layout	17
Figure 3.1: Flowchart for IoT Data Logger for SNMP Data Monitoring	19
Figure 3.2: Working Flow for IoT Data Logger for SNMP Data Monitoring	21
Figure 3.3: NodeMCU-ESP32	22
Figure 3.4: Pinout for NodeMCU-ESP32	23
Figure 3.5: Raspberry Pi 3 Model B	24
Figure 3.6: GPIO Pinout Raspberry Pi 3 Model B	25
Figure 3.7: DHT11 Temperature and Humidity Sensor	26
Figure 3.8: ZMPT101B AC Voltage Sensor	27
Figure 3.9: SCT-013-000 AC Current Sensor	28
Figure 3.10: myDevices Cayenne IoT Platform	29
Figure 3.11: Arduino IDE	29
Figure 3.12: Raspberry Pi Operating System	30

Figure 3.13: fritzing Circuit Maker	30
Figure 3.14: IFTTT	31
Figure 3.15: Telegram Messaging	31
Figure 3.16: Source Code for ESP32 and Cayenne Platform	32
Figure 3.17: Source Code for Temperature and Humidity Sensor	33
Figure 3.18: Source Code for AC Voltage Sensor	34
Figure 3.19: Source Code for AC Current Sensor	35
Figure 3.20: Source Code for Power Consumption	36
Figure 3.21: FTP Server Configuration for SNMP	37
Figure 3.22: FTP Server is active	37
Figure 3.23: Cayenne IoT Platform Dashboard	38
Figure 3.24: Webhook configuration in Cayenne IoT Platform	39
Figure 3.25: If Then statement configuration	39
Figure 3.26: IFTTT Applets	40
Figure 3.27: Telegram Configuration and Alert Notification	41
Figure 3.28: Circuit Diagram for Power, Humidity and Temperature Monitoring System	42
Figure 3.29: Circuit Diagram for SNMP Data Monitoring System	43
Figure 3.30: Hardware Circuit Diagram	43
Figure 3.31: Server Rack	44
Figure 4.1: Mi Home Temperature and Humidity Sensor and Mi Home GUI	46
Figure 4.2: Power Monitor Meter	47
Figure 4.3: TP-Link Switch Statistic Manager	48
Figure 4.4: Data Extraction from all sensor data received in specific time	49

Figure 4.5: Data Extraction from specific sensor data received in specific time	49
Figure 4.6: Arduino IDE Serial Monitor	50
Figure 4.7: Raspberry Pi OS Resource Monitor	51
Figure 4.8: Line Graph Temperature Variable	52
Figure 4.9: Line Graph Humidity Variable	53
Figure 4.10: Line Graph AC Voltage Variable	54
Figure 4.11: Line Graph AC Current Variable	55
Figure 4.12: Line Graph Power Consumption Variable	56
Figure 4.13: Line Graph Network Speed Variable	57
Figure 4.14: Top View of the IoT Data Logger Hardware	58
Figure 4.15: Overall View of the IoT Data Logger Hardware	58
Figure 4.16: Server Rack	59
Figure 4.17: Cayenne IoT Platform Dashboard	59
Figure 4.18: QR Code for the IoT Dashboard	60
Figure 5.1: United Nation Sustainable Development Goals 9	62

LIST OF TABLES

Table 3.1: NodeMCU-ESP32 Specification	23
Table 3.2: Raspberry Pi 3 Model B Specification	25
Table 3.3: DHT11 Temperature and Humidity Sensor Specification	26
Table 3.4: ZMPT101B AC Voltage Sensor Specification	27
Table 3.5: SCT-013-000 AC Current Sensor Specification	28



LIST OF SYMBOLS AND ABBREVIATIONS

For examples:

SNMP : Simple Network Management Protocol

MCR : Master Control Room

RTC : Real Time Clock

AUX : Auxiliary Port

IP : Internet Protocol

GP : Gain Power

SOP : Standard of Procedure

IoT : Internet of Things

MIB : Management Information Base

FM : Frequency Modulation

BTS : Base Transmission Station

OS : Operating System

IFTTT : If This Then That

CPU : Central Processing Unit

LXDE : Lightweight X11 Desktop Environment

IM : Instant Messaging

VoIP : Voice over Internet Protocol

IDE : Integrated Development Environment

MQTT : Message Queuing Telemetry Transport

FTP : File Transfer Protocol

GUI : Graphic User Interface



LIST OF APPENDICES

Appendix A: IoT Data Logger System For SNMP Data Monitoring Source Code	67
Appendix B: Deva DB8008 Manual	72



CHAPTER 1

INTRODUCTION



1.1 Background of the Project

Proposed to make the outdoor IoT Data Logger Framework with Cayenne platform NodeMCU ESP32 and Raspberry Pi as Dashboard for this final year's project FYP. Most of the work includes hardware and programming (electrical and electronic). The framework built should be able to extract data from the SNMP protocol and it should appear in a cayenne dashboard. Automate some output then. This is a joint initiative with the broadcaster industry.

1.2 Problem Statement

Systematic control of the broadcasting system is important since the operating hours of the radio station in the popular radio station are about 6 AM until 12 AM the next day. Since the radio station has normal operating hours with a engineer present to work, the radio station is still active after operating hours, but both are automatically controlled by a broadcasting system consisting of a complex audio system. Problems occur while using the device, which is:-

- Engineer unable to monitor system remotely.
- Engineer manually records all technical issues in Master Control Room (MCR).
- No market available for local broadcasting industry.

1.3 Objectives

To accomplish this project, there are several objectives to achieve as follows:-

1. To investigate the performance of temperature, humidity and electrical sensors. [R01]
2. To design a stand-alone microcontroller-based remote data logger that able for measuring, monitoring and store the broadcasting system parameters such as Main Audio Loss, AUX Loss, IP Audio Loss, Power status and GP IN Power Switch in the cloud. [R02]
3. To develop stand-alone IoT microcontroller and microprocessor with Cayenne Platform as a dashboard. [R03]

1.4 Scope of Project

In order to develop this system, a prototype should be prepared. In order to accomplish it, scope of the project is summarized as follow:

- Develop a NodeMCU Microprocessor as the main controller for the system
- Design sensor circuits for temperature, humidity and power status
- Develop SNMP network technology to enable the prototype to extract data parameter from broadcaster device.
- Design a Real Time Clock (RTC) for real-time parameter and integration of Cayenne platform for enable this prototype IoT-enable system.

1.5 Project Significance

IoT Data Logger system project is enabling effective monitoring which also complies with the Standard of Operating Procedure (SOP), temperature sensor, humidity sensor, AC Voltage and Power Sensor and SNMP Protocol with the ability to continuously process the recorded/stored information/data or control remotely the broadcaster equipment when it is necessary. As well as that, this information/data also should be able to be recorded into the system database to provide real-time monitoring system. This project is to solve the problem where the engineer must attend complicated breakdown of equipment face-to-face approach which usually is quite time-consuming.

IoT Data Logger monitoring system is one of the most important systems which is in demand throughout the broadcasting industry. This project can be the most effective equipment to reduce and prevent the risk mass equipment breakdown. It is designed to use at any kind of studio for audio and visual proposed. The project has the potential to be commercialized because of all types of broadcasting industry in the

conversion workforce of COVID pandemic that disable the engineers to attend the equipment face-to-face as example shown in Figure 1.1.



Figure 1.1: Master Control Room

This project durable and used continuously over a long time. According to the WHO, the coronavirus expected the COVID-19 outbreak will last between 18 and 24 months and The Edge, current work from home culture will be continue until 2022 when the economy is fully recovered from the pandemic and economy downfall. Therefore, this system can comply with the Standard of Operating Procedure (SOP). The developed project will be simply environmentally friendly because this project doesn't use any type of chemical or any wiring is required.

1.6 Thesis Structure

Chapter 1 is a short description and a general overview of what is IoT Data Logger monitoring system. The problem statement is clearly stated that this project is important. The meaning of the study and the research questions, hypotheses or assumptions to follow will then be addressed in this chapter. This explains which problem has been solved or which improvements have been made compared to earlier studies. The aims of this project are to inform researchers of the expected results at the end of the project. The scope of the task is outlined in detail in order to define the project limits. The importance of the project is discussed to show the effect on the mentioned problems.

Chapter 2 is a literature review. Numbers of publications and journals for this study have been reviewed. Themes relating to the study will be collected and addressed here in this chapter in order to fulfil the project goal and answer the problem indicated. The review chapter contains a short introduction to this project and the IoT Monitoring System that is available on the market, this section has summarized the paper related to this paper.

Chapter 3 methodology of the project. This chapter describes the implementation to fulfil the project goals. Detailed software, hardware and diagram. In addition, every phase of the flowchart is explained attentively. It is explicitly mentioned the approach or method employed in each stage.

Chapter 4 is the results and discussion. In this section, the result of each step to complete the project are shown. The result of the data monitoring methods, equipment monitoring method, and notification received method is shown in this

section. The usage of the methods and algorithms are discussed in this section as to why the project uses any of the chosen methods to complete the project. The results comparison between the computer and smartphone are also discussed in this section to justify the difference between the two platforms.

Chapter 5 concludes the result and discussion that are obtained from the experimental process. This chapter will discuss more the future work that can be done to improvise the system and the limitation that needs to be overcome in the future.



CHAPTER 2

BACKGROUND STUDY



2.1 IoT Data Monitoring System

The Internet of Things (IoT), automation and embedded technologies are currently fast becoming a component of the industry. This is associated with the creation of a wealth of hardware modules and CPUs. Development boards are established where a communication interface and peripherals with the main processor chip are implemented. Nowadays, the popularity of the ESP32 microprocessor is growing and nowadays it develops both hardware variations and other branches of the development of its software. A large community of developers and scholars also use the ESP32 microprocessor of higher generation as the successor to a microcontroller ESP8266. The latest research studies demonstrate the vast application of the ESP32 microprocessor in several fields.[1]

With the development of the new round of information technology revolution towards intelligence, big data, cloud computing, mobile internet and artificial intelligence technologies are triggering a “wave of intelligence”, furthermore the traditional monitoring method of manual record of the equipment status can not meet the requirements of high-speed processing efficiency.[2] It is urgent to improve the intelligence capabilities of broadcast monitoring and supervision. In view of the above reasons, the author and the project team have developed a set of intelligent broadcast monitoring system.

In the case of temperature and humidity monitor, the sensor DHT11 is able to measure temperature and moisture. This DHT11 sensor detects room temperature and humidity that can supply the ESP32 input to the exhaust fan output ON and OFF states. When a room temperature exceeds the setpoint, the ESP32 microcontroller sends the Exhaust FAN to switch on. Instead, if the sensor reading is less than the setpoint, the ESP32 commands the Exhaust FAN to switch OFF. Reed Switch works as a switch that is either enabled or linked if the coverage area has a magnetic field. In case of a breach, Reed Switch will detect alarms on windows and doors. Reed Switch supplies an ESP32 input to release the alarm. If a house break occurs, the ESP32 gives the alarm to the ON.[3]

For power consumption monitor, Non-invasive way of measuring electricity with large capacity that is now available on the market. These only provide readings of up to 16 A, with a possible extension to 100 A, our method can measure current up to 30A if the load resistor value is changed. [4] ZMPT101B is used to measure the AC

voltage because it is capable to measure up to 240V AC. [5] Both inputs can be computed on cayenne platform and this enable the system to monitor the power consumption of the equipment.

Finally, Cayenne IoT is an open-source platform that is widely used as a messaging platform. It is free and open-source, with an ad-free environment and a simple, quick GUI interface. [6]. In this research, cayenne is used to check the condition of the broadcasting system by using Cayenne GUI and sending alerts via the e-mail.

2.2 Review Available for Monitoring IoT Data Monitoring System

2.2.1 All-in-One Application for Smart Home System

Based on the research by authors (Febi Indriyati Rukmana, Edi Mulyana, Lia Kamelia, Akmaliah, Aep Kusnawan, Wahyudin Darmalaksana, 2020) this project is smart home integration can develop a smart home system. Smart Home is defined as a home that uses a control system for integrating different automation systems into the house. This research is aimed at designing an intelligent home control based on telegram conversation. This research employs the automatic control system ESP32. The ESP32, is a 2.4 GHz Wi-Fi and Bluetooth enabled microcontroller and the ESP32 development kit is available on the market at a low price. The LDR sensor is used to turn the lighting system on/off automatically. DHT11 was utilised to measure and manage the fan room temperature and humidity.

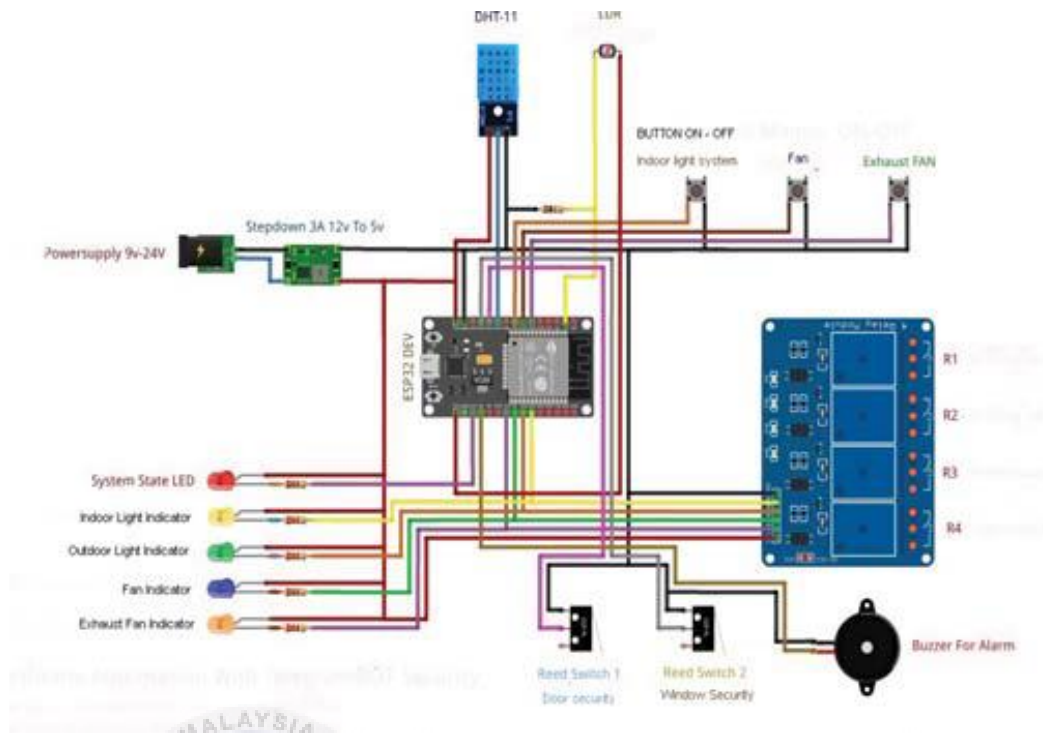


Figure 2.1: All-in-One Application for Smart Home System Circuit Diagram

This research employs the automatic control system ESP32. ESP32 is a 2.4 GHz Wi-Fi and Bluetooth equipped microcontroller and the ESP32 development kit may be easily acquired on the market for a low price. The key features of this ESP32 chip include low power, with TCP/IP and Bluetooth integration, decent documentation and compiler support (Arduino & ESP-IDF) for C++ [16]. This research is aimed at designing an intelligent home control based on telegram conversation. Telegram is a platform open-source used as a popular messaging service. Its fully free service, free of charge, delivers a clean and quick interface ads-free environment [17]. Telegram is utilised in this research to check the house's condition by sending messages via the telegram programme.

The monitoring system devised and implemented in the manual for the automation house control application can perform data storage and monitoring in real time. The data collected are LDR, DHT11 and Reed switch data. In this intelligent home control system, the alarm is switched on when the windows and doors are broken. Data are continuously delivered to the history given by the telegram application through the monitoring system, as long as the internet connection is good and the voltage supply to the microcontroller is not disrupted.

2.2.2 Design and Implementation of Server Monitoring System Based on SNMP

Based on the research by authors (Wenxian Zeng, Yue Wang, 2016) this project is to Simple network management protocol to monitor servers (SNMP). MIB resources are expanded by defining MIB objects for monitoring severe resources and using multi-threading technology for collecting and processing data to improve collection efficiency. The testing findings show that it is a successful integrated SNMP-based monitor and control system for servers.

The networking paradigm used for SNMP includes: management station, management agent, information base management (MIB), network management protocol (SNMP), depicted in the Figure 2.2.

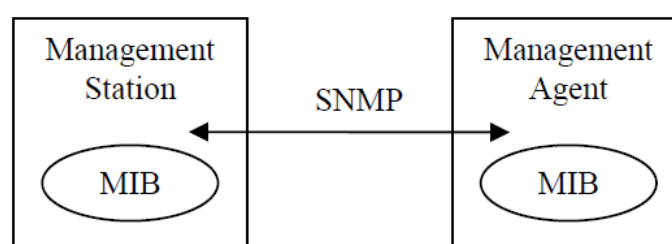


Figure 2.2: The SNMP-based Management Model

The system of monitoring is based on the model of the manager agent of SNMP[3, 4]. We use the layered structure method to design the system based on the different system functions. The system should contain the following modules shown in the Figure 2.3.

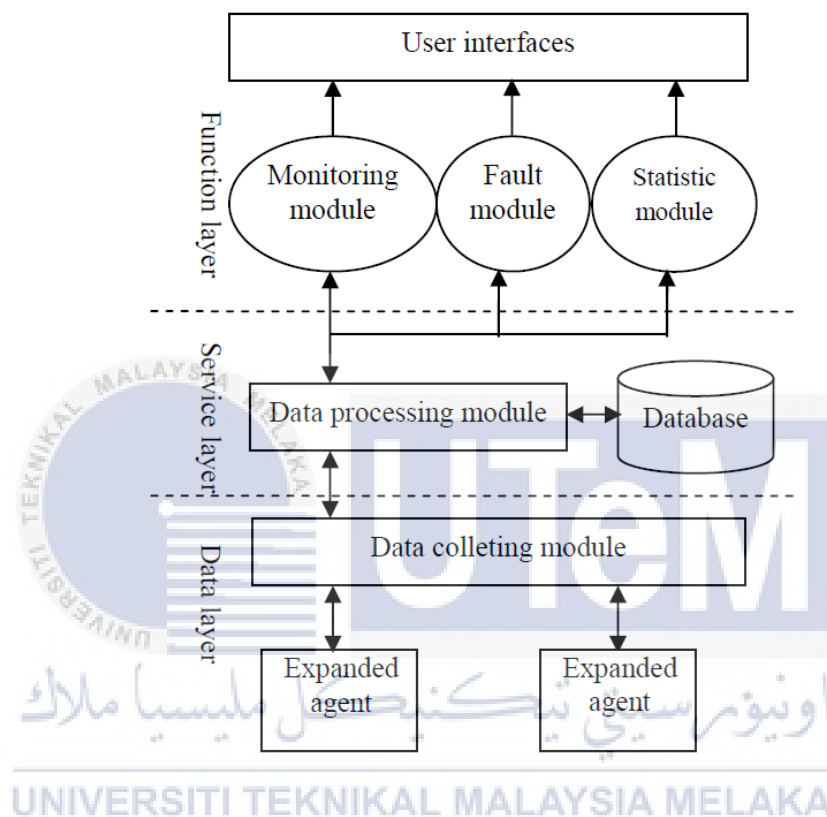


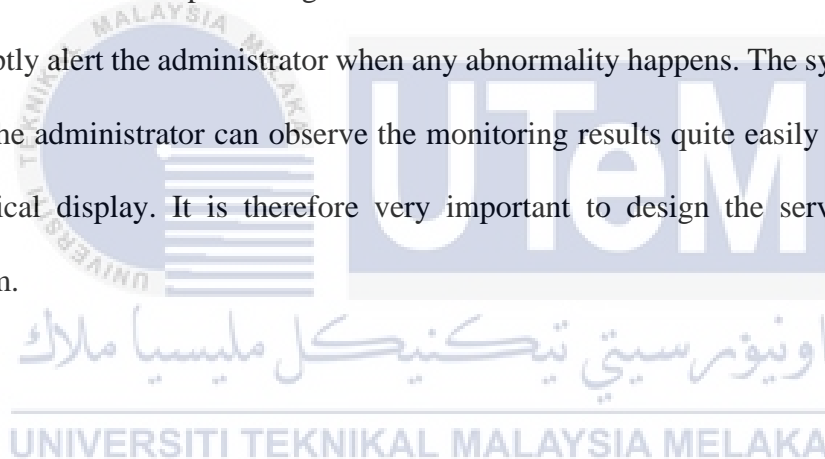
Figure 2.3: Framework of the system

The underlying data layer ensures communication between the manager and the agent, and requires and defines information in the MIB. We use two approaches of collecting data here. One is a real-time collection which collects the information to be shown in real time and sends it to the top layer in time. Another option is the timing of polls, which regularly collect and transmit information to the upper layer during an interval.

The middle layer of the service is responsible for handling the information collected. If the data collected is displayed in time, it is sent straight to the higher layer to show or kept in the database to view the historical information.

The function layer in the top layer is the interface for the system administrator. In a visual and graphical interface, the server monitor can provide setup information, performance statistics and defect information.

Given the advantages of SNMP protocol, such as the easy, flexible, modest network load and powerful expansion, we design a system that can monitor and control servers without compromising the burden of the server and its services. It can also promptly alert the administrator when any abnormality happens. The system works so far. The administrator can observe the monitoring results quite easily because of the graphical display. It is therefore very important to design the server monitoring system.



2.2.3 Design and Implementation of Intelligent Audio Broadcast Monitoring System

Based on the research by authors (Dingjing Zhang, Rui Fu, Lin Chang, Ying Wang, Ling Tuo, 2019) this project to improve the accuracy and effectiveness of the real-time monitoring on illegal broadcasting and violative broadcasting, the author used artificial intelligence technology to realize the intelligent audio broadcast monitoring system, including frequency domain template comparison, speech recognition, text comparison and text classification. The system has been deployed in the provincial monitoring center, which can realize the automatic processing of the

whole flow of illegal broadcasting and violative broadcasting monitoring from discovery, recording and evidence collection, voice translation, to intelligent recognition, it can greatly improve the efficiency and timeliness of audio broadcast supervision. This paper mainly analyzed the requirements and functions of the system, described the design scheme and implementation methods of the system, and put forward the deployment scheme of the system according to the actual situation.

According to the requirements of audio broadcast monitoring of the relevant departments of China's radio and television supervision is Illegal broadcasting monitoring: The illegal broadcasting (i.e. black broadcasting) mainly refers to the broadcasting stations that set up and use the broadcasting frequency to broadcast to the society without the approval of the Radio Management Organization and the Radio and Television Department. It is mainly used in illegal drugs, fake drug sales and other commercial activities, causing great harm to the society.

According to the business requirements of audio broadcast monitoring, the author put forward the functional requirements of the system, mainly includes the aspects as below.

- 1) Illegal frequency real-time monitoring: Automatic scanning of FM (87-108MHz) full frequency band, finding abnormal frequency, and reporting alarm automatically.
- 2) Intelligent analysis: According to the principle of statistical analysis, summarize all the results after discrimination, provide summary of violations, types of violations, alarm distribution, and violation trend analysis, etc., so as to facilitate the monitoring and management department to control the alarm situation from a macro perspective.

demonstration deployment in two provincial monitoring centers. It has realized the whole automated processing of "black broadcast" and "grey broadcast" monitoring, and the automated processing on discovery, recording and forensics, voice translation, achieved better monitoring effect. This system can not only greatly improve the efficiency and timeliness of the supervision of audio broadcasting, but also play an important role in further strengthening the supervision of the radio and television industry, eliminating the supervision blind area as soon as possible, maintaining the broadcasting order of radio and television, and creating a clear wireless space.

2.2.4 Prototype Design of Monitoring System Base Transceiver Station (BTS)

Base on Internet of Things

Based on the research by authors (Pajar Abdul Malik Hambali, Syamsuddin, Mufid Ridlo Effendi, Eki Ahmad Zaki Hamidi, 2018) this project is a prototype of Base Transceiver (BTS) Things Internet-based monitoring system, where five BTS primary sensors are installed, as BTS parameter, to be monitored in real time. These parameters are: BTS voltage source with ZMPT101B sensor is attached. The BTS battery voltage with DC voltage sensor is installed. The MC38 sensor is mounted on the BTS door. BTS cable is equipped with a sensor SW420 and BTS room temperature is integrated with a sensor DHT22. All sensors are controlled by Arduino, which installs data in sensors using Ethernet. The server itself uses Raspberry Pi which is installed as a data base transceiver to MySQL with the NodeRed programme. MySQL acted as a webserver so that data might be shown as a sensor output on the web interface. The quality of the sensors is also tested and compared with calibrated standard measuring device and the output of all sensors.

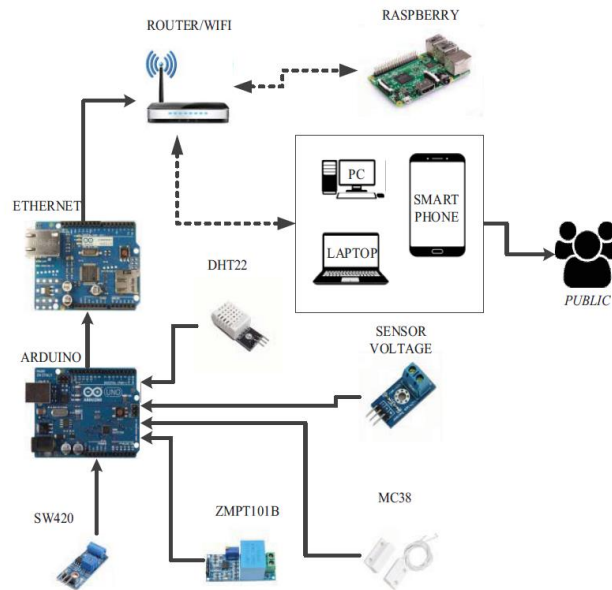


Figure 2.5: Hardware Layout

For each sensor, Arduino is used as a controller for the measurement of the source voltage (ZMPT101B), MC38 as magnetic door sensor, SW420 cable vibration sensor, DHT22 room temperature sensor and VOLTAGE DC in BTS, where sensor-data are transferred to Raspberry Pi via Ethernet, Raspberry Pi Model B also serves as server for the collection of sensor data from each BTS parameter.

The conclusion of this study is a web-based BTS monitoring system prototype using Raspberry as a server, which has monitoring functions for AC source voltage inputs, battery input, or DC voltage inputs, room temperature sensors, BTS vibration input monitoring, and BTS vibration cable detector.

CHAPTER 3

METHODOLOGY



3.1 Introduction

In this chapter, details for the sensor and system technology for the project is introduced. The block diagram, flow chart, hardware development and cost. Hence, this chapter will elaborate about the procedure of the project.

3.2 Flow Chart

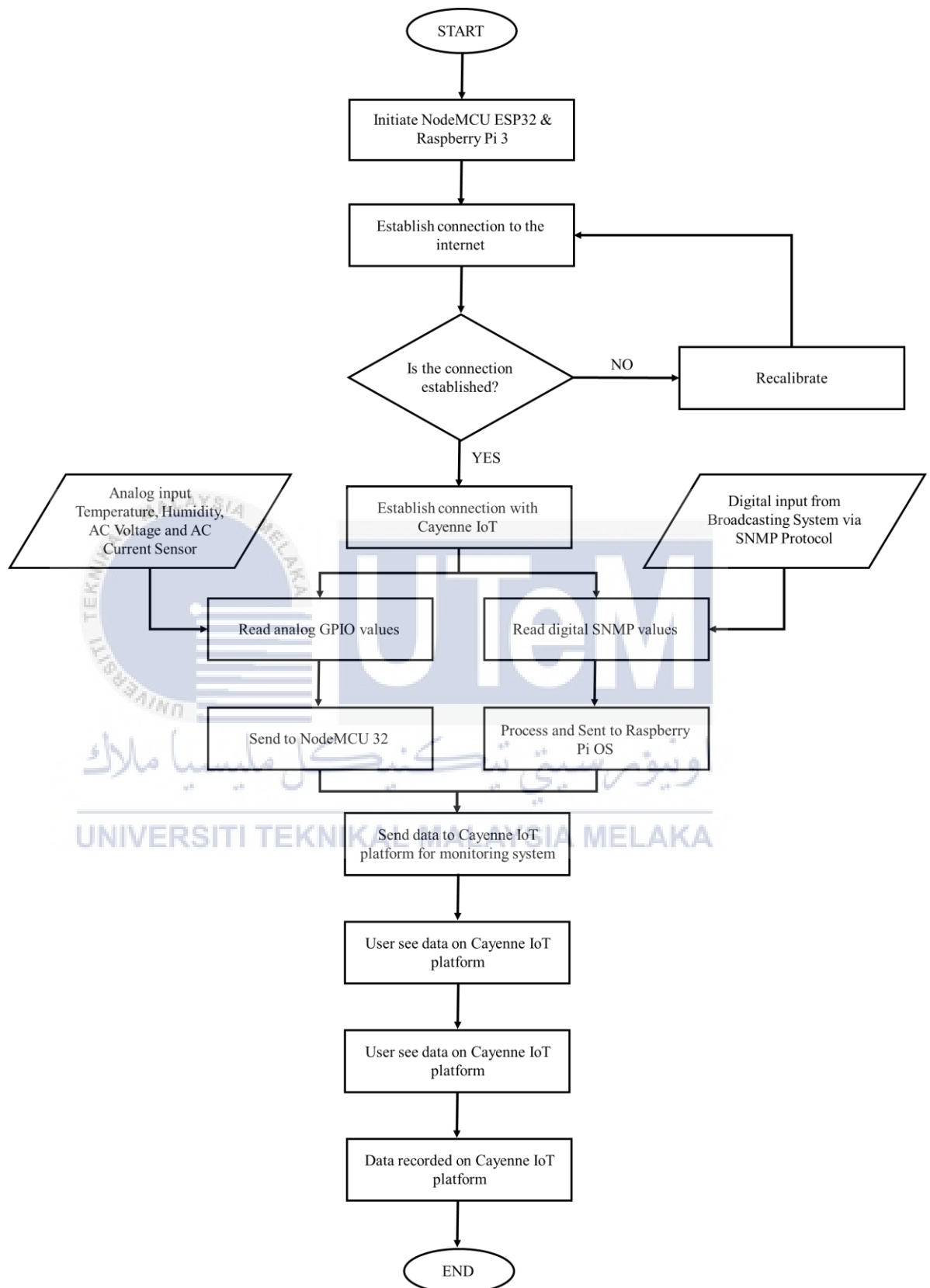


Figure 3.1: Flowchart for IoT Data Logger for SNMP Data Monitoring

For the hardware setup the circuit contain 4 sensors and 1 digital variable which are Temperature sensor, Humidity sensor, AC Voltage sensor, AC Current sensor and SNMP Protocol that will be connected to NodeMCU ESP32 and Raspberry Pi 3. The functions of the process need to be function as they communicate with each other for transmit the data over the internet of things (IoT).

1. Establish the connection between the NodeMCU ESP32 and Raspberry Pi 3 and internet connection. Once the connection establishes, it will communicate with Cayenne IoT Platform.
2. The sensor will fetch the data to the NodeMCU ESP32 and will appear the Temperature sensor, Humidity sensor, AC Voltage sensor and AC Current sensor.
3. The SNMP Protocol will fetch the data to the Raspberry Pi and will appear the SNMP Monitor System.
4. The engineer can monitor by use the Cayenne IoT Platform Dashboard via smart phone. The data will also been recorded into the cloud.

3.3 Working Flow

This part shows the components and sensors used with its roles as inputs or outputs as shown in the operating system for IoT datalogger monitoring system in Figure 3.2.

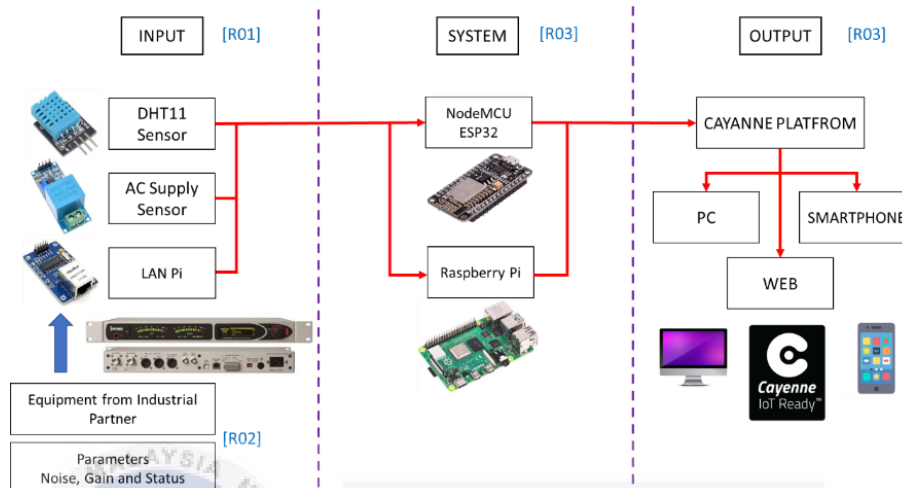


Figure 3.2: Working Flow for IoT Data Logger for SNMP Data Monitoring

There are two parameter electronic devices used in this research which is usage of sensors and hardware integration system. For the sensor parts, temperature sensor, humidity sensor and AC voltage and AC Current sensor will be connected to the NodeMCU-ESP32 board and hardware integration system will be using LAN Raspberry Pi Adapter to enable SNMP function with broadcasting equipment. The NodeMCU-ESP32 board with the connection the sensors and Raspberry Pi 3 Model B board are put in the fireproof case near to the broadcasting system to avoid the board exposed of the heat. Then, NodeMCU-ESP32 and Raspberry Pi 3 Model B will sent the data as a input and output to the IoT system which use in this project is Cayenne Platform. Cayenne Platform will show the data through the dashboard platform and can be access through smartphone and can monitor the broadcasting system from the smartphone. To communicate the NodeMCU-ESP32, Raspberry Pi 3 Model B and Cayenne Platform, the boards need the internet access via Wi-Fi is needed to send the data input and output.

3.4 Research Hardware

3.4.1 NodeMCU-ESP32



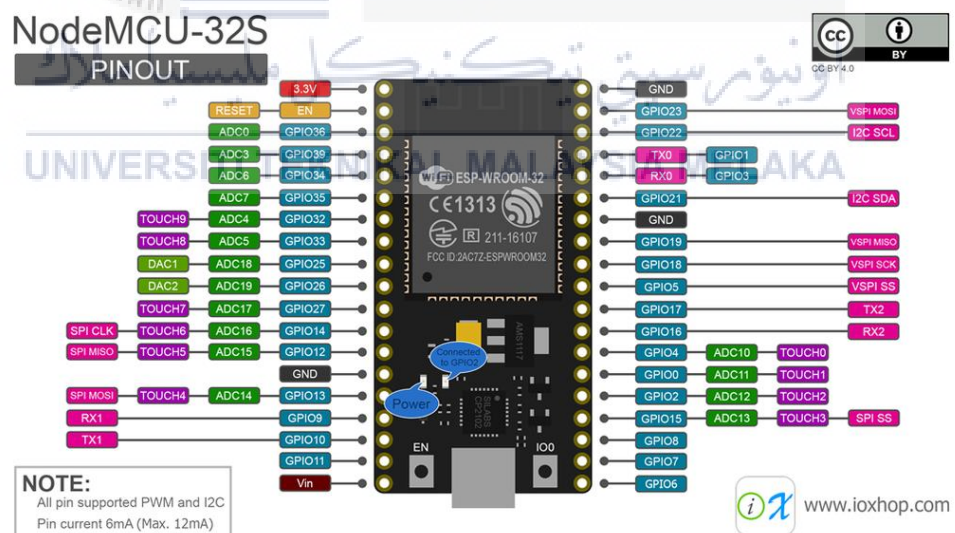
Figure 3.3: NodeMCU-ESP32

The main controller for this project is NodeMCU-ESP32 and it is simple circuit platform based on I/O port that implements of processing language. It is programmed with the Lua language scripting. The platform is based on open source projects from eLua. The platform uses many projects like lua-cjson and spiffs. This NodeMCU-ESP32 includes firmware for ESP32 WiFi SoC chips and hardware based modules ESP-32S. NodeMCU-ESP32 is used to control the temperature sensor, humidity sensor and AC Voltage and Current sensor for cloud data collection and storage.

NodeMCU is an open-source IoT platform. The NodeMCU-ESP32 is open source which can be controlled an analogue and digital in physically control and digitally. The tables below show the characteristics of NodeMCU-ESP32.

Table 3.1: NodeMCU-ESP32 Specification

Type microcontroller	ESP-wroom-32 module
Operating voltage	2.2v ~ 3.6v
Operating current	Average:80 ma
Operating temperature	-40~+85
Wi-Fi frequency range	2.4~2.5ghz
Module interface	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR GPIO, capacitive touch sensor, ADC, DAC, LNA pre-amplifier
Flash memory	4Mbyte
Clock speed	40mhz

**Figure 3.4: Pinout for NodeMCU-ESP32**

3.4.2 Raspberry Pi 3B

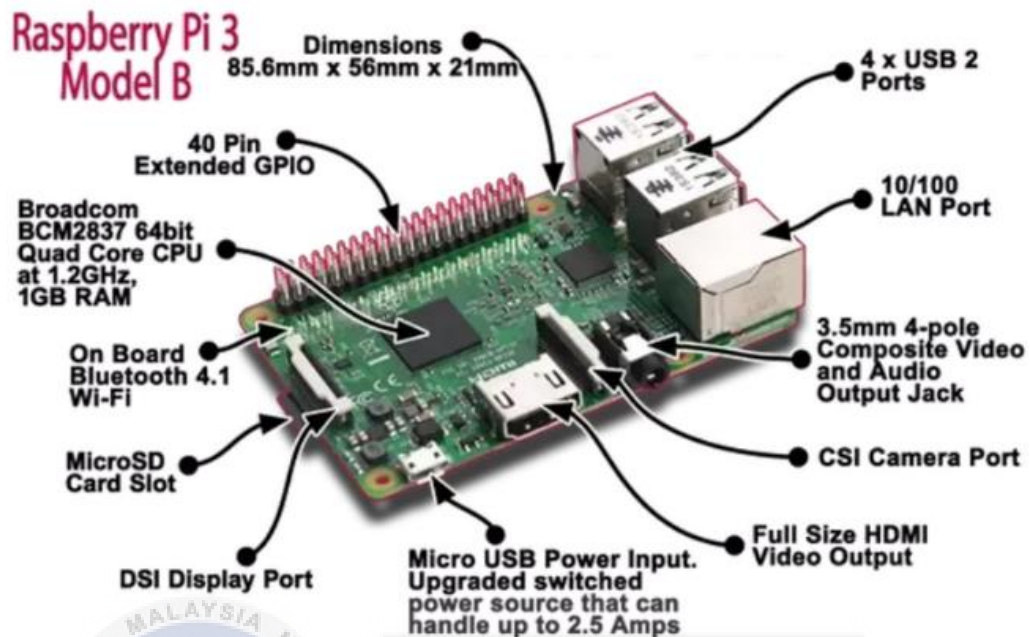


Figure 3.5: Raspberry Pi 3 Model B

The Raspberry Pi is a low cost computer that connects to a computer monitor or TV and uses a conventional mouse and keyboard. It is a small device that allows people of all ages to explore and learn to programme in Scratch and Python. You can do everything you expect a desktop computer to do, from browsing the Internet and playing video to creating tablets, word processing and playing games.

The four built-in USB ports of the Raspberry Pi 3 provide sufficient connectivity to your mouse, the keyboard or whatever you believe the RPi needs, but if you choose to add even more, you may use a USB hub nonetheless. It is also constructed in Megabit LAN port class for fast Internet use and may also be used as an SNMP agent and SNMP Manager.

Table 3.2: Raspberry Pi 3 Model B Specification

Type microprocessor	Quad Core 1.2GHz	Broadcom
	BCM2837 64bit CPU	
Operating voltage	5V	
Operating current	2.5 A	
Operating temperature	0 – 50 °C	
Connectivity	2.4GHz and 5GHz IEEE 802.11 b/g/n/ac	
	wireless LAN, Bluetooth 4.2, BLE	
	100Mbps Ethernet over USB 2.0	
	4 × USB 2.0 ports	
Module interface	SD card, UART, SPI, SDIO, I2C, LED	
	PWM, Motor PWM, I2S, IR	
	GPIO, MIPI DSI Display, MIPI CSI	
	Camera, HDMI	
Memory	1GB LPDDR2 SDRAM	
Flash memory	16GB via MicroSD	



Figure 3.6: GPIO Pinout Raspberry Pi 3 Model B

3.4.3 Temperature and Humidity Sensor

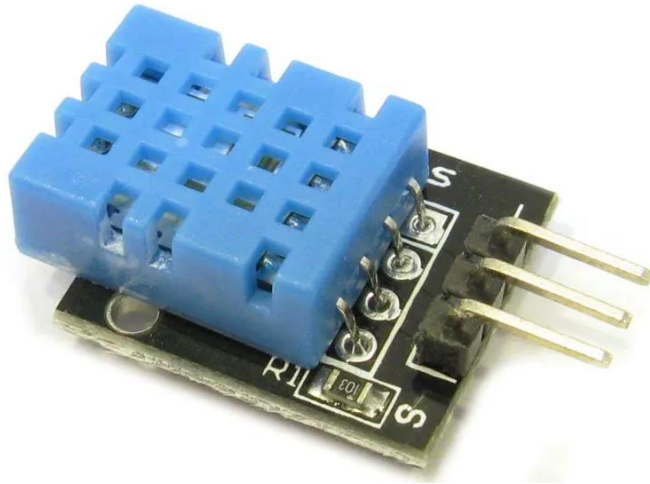


Figure 3.7: DHT11 Temperature and Humidity Sensor

The DHT11 is a basic digital temperature and humidity sensor that is extremely cost-effective. It employs a moisture sensor for measuring the surrounding air and a thermistor and sends a digital signal out on the data pin (no analogue input pins needed). It is quite easy to use, but requires meticulous time to record data. The only significant negative of this sensor is that you can only acquire new data from the sensor every 2 seconds, therefore the sensor readings can be up to 2 seconds when you use the library.

Table 3.3: DHT11 Temperature and Humidity Sensor Specification

Voltage	5v DC
Current	15mA
Temperature Reading	Good for 20-80% humidity readings with 5% accuracy
Humidity Reading	Good for 0-50°C temperature readings $\pm 2^\circ\text{C}$ accuracy
Dimension	15.5mm x 12mm x 5.5mm

3.4.4 AC Voltage Sensor

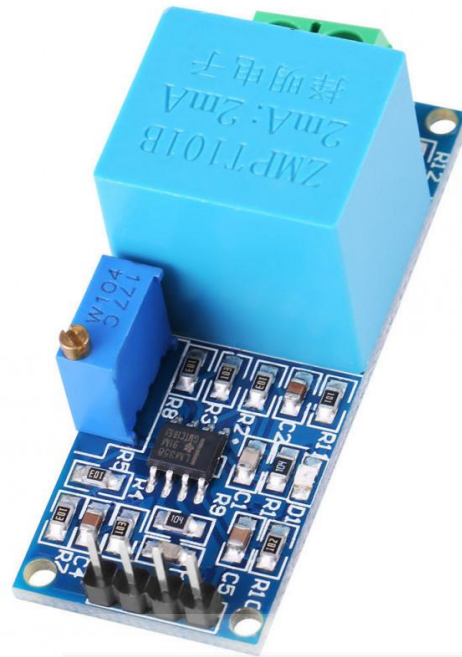


Figure 3.8: ZMPT101B AC Voltage Sensor

The ZMPT101B is an appropriate voltage transformer for AC voltage measurement. The voltage and power measurement is highly accurate and consistent. The module is simple to operate and features a multi-turn trim potentiometer to tune and calibrate the ADC output.

Table 3.4: ZMPT101B AC Voltage Sensor Specification

Voltage	5v DC / 240v AC
Current	10mA
Voltage Reading	Good for 20-80% humidity readings with 5% accuracy
Dimension	15.5mm x 12mm x 5.5mm

3.4.5 AC Current Sensor



Figure 3.9: SCT-013-000 AC Current Sensor

The SCT-013-000 Non-invasive AC Current Sensor is a current transformer that can be clamped around the electrical load supply line to measure the current. It does so by serving as an inductor and responding to the magnetic field around a driver. The current travelling through the conductor can be estimated by reading the amount of current created by the spindle. It is particularly useful for tracking electricity usage or generation throughout buildings.

Table 3.5: SCT-013-000 AC Current Sensor Specification

Voltage	5v DC / 240v AC
Current	30A
Frequency	50HZ-150KHZ
Current Reading	Good for 20-80% humidity readings with 5% accuracy
Dimension	15.5mm x 12mm x 5.5mm

3.5 Research Software

3.5.1 myDevices Cayenne IoT Platform



Figure 3.10: myDevices Cayenne IoT Platform

myDevice Cayenne is an online IoT dashboard that makes hardware-oriented programming most complicated. myDevice Cayenne enables IoT solutions to be easily designed, prototyped and visualised. Can be used as a tool to virtualize historical and real-time data delivered over the internet.

3.5.2 Arduino IDE



Figure 3.11: Arduino IDE

The open-source Arduino Software (IDE) enables writing and uploading code to the board straightforward. It works with Windows, Mac OS X and Linux. The environment is written in Java and is based on processing and other applications opensource. Arduino is a physically programmable circuit board (microcontroller) as well as software component (IDE) which operates on a computer, writes and uploads the physical board with computer code.

3.5.3 Raspberry Pi OS



Figure 3.12: Raspberry Pi Operating System

Raspberry Pi OS is a Debian-based Raspberry Pi operating system. Since 2015, the Raspberry Pi Foundation has been officially offered as the primary operating system for the Raspberry Pi Computer family. The Raspberry Pi range of small single-board computers using ARM CPUs is highly optimized. It works on every Raspberry Pi other than the Pico microcontroller. Raspberry Pi OS employs a modified LXDE, with the Openbox stacking window manager and a unique look for its desktop experience.

3.5.4 fritzing

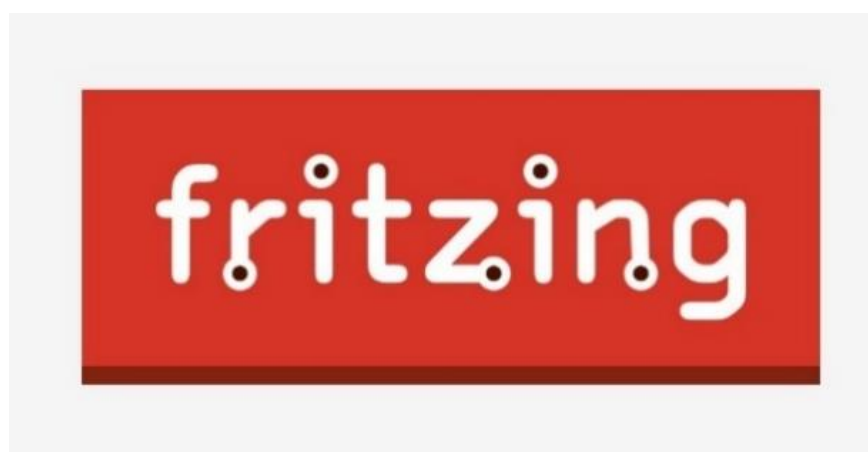


Figure 3.13: fritzing Circuit Maker

Telegram is a freeware, cross-platform app for cloud-based instant messaging (IM). This service also offers end-to-end encrypted video calling, VoIP, file sharing, webhook bots, multi-device access and many other functions. It can be used for Internet Of Things (IoT) services with IFTTT interaction integrated within Telegram, integrate webhooks with IoT service providers such as myDevice Cayenne platform.

3.6 Software Development

3.6.1 Arduino IDE

There is software development which been used in this project. Arduino IDE software is used to write and uploads the program for the sensor to be function. The code is written in C++ and be downloading in the microcontroller chip to be run. These codes contain part of which are for temperature and humidity sensor, AC Voltage sensor and AC Current sensor.

3.6.1.1 Connection between ESP32 and Cayenne Platform

To achieve the communication between ESP32 and Cayenne Platform the code is developed in Arduino IDE. The connection between them must follow the protocol that microcontroller provide. The protocol that uses to establish this the connection between them is MQTT protocol. The protocol must state in the declaration state.

```
#include <CayenneMQTTESP32.h> // Library for Cayenne
#define CAYENNE_DEBUG
#define CAYENNE_PRINT Serial

// WiFi network info.
char ssid[] = "HUAWEI MatePad T";
char wifiPassword[] = "0178790158";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "0952d3a0-23e8-11eb-8779-7d56e82df461";
char password[] = "2d729c01d97f30c3999dc8d5a72028f0e408fc2e";
char clientID[] = "148c1660-2518-11eb-a2e4-b32ea624e442";
```

Figure 3.16: Source Code for ESP32 and Cayenne Platform

3.6.1.2 Temperature and Humidity Sensor Code

The #include <DHT.h> is stated as temperature and humidity type sensor which is DHT sensor library is loaded and DHT dht(0, DHT11) is to indicate that sensor DHT11 is used. Temperature and humidity sensor is digital sensor, and then the algorithm for the temperature and humidity sensor is direct as the data input to be taken from the sensor and display to the application. The virtual channel is set as channel 0 for humidity and channel 1 for temperature will show up in the Cayenne IoT dashboard platform.

```
#include <CayenneMQTTESP32.h> // Library for Cayenne
#define CAYENNE_DEBUG
#include "DHT.h" // Library for DHT sensor
#define CAYENNE_PRINT Serial
DHT dht(0, DHT11);

// WiFi network info.
char ssid[] = "HUAWEI MatePad T";
char wifiPassword[] = "0178790158";

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = "0952d3a0-23e8-11eb-8779-7d56e82df461";
char password[] = "2d729c01d97f30c3999dc8d5a72028f0e408fc2e";
char clientID[] = "148c1660-2518-11eb-a2e4-b32ea624e442";

void setup() {
  Serial.begin(115200); // Baud rate for serial monitor
  dht.begin(); // To start communication with DHT sensor
  Cayenne.begin(username, password, clientID, ssid, wifiPassword); // To Authenticate cayenne
  pinMode(2, OUTPUT);
  digitalWrite(2, HIGH);
}


void loop() {
  delay(2000); // Give some delay between each Sensor reading
  Cayenne.loop();
  float h = dht.readHumidity(); // To get Humidity reading
  float t = dht.readTemperature(); // To get temperature reading
  if (isnan(h) || isnan(t)) { // Condition to check whether the sensor reading was successful or not
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Humidity: "); // To print the data in serial monitor
  Serial.print(h);
  Serial.print(" %\t Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");
  Cayenne.virtualWrite(0, h); // To write the Sensor reading to Cayenne Dashboard using channel 0
  Cayenne.celsiusWrite(1, t); // To write the Sensor reading to Cayenne Dashboard using channel 1
}
CAYENNE_IN(2)
{
  digitalWrite(2, !getValue.asInt()); // to get the value from the website
}
```

Figure 3.17: Source Code for Temperature and Humidity Sensor

3.6.1.3 AC Voltage and AC Current Sensor Code

The `#include <EmonLib.h>` is stated as voltage and current type sensor which is ZMPT101B Voltage Sensor and SCT-013-000 Non-invasive AC Current Sensor analog Energy Monitor library is loaded. The `emon.voltage` and `emon.current` is to indicate that sensor both sensor GPIO and the calibration value stated in the `#define vCalibration` and `#define currCalibration`. AC Voltage and Current sensor is an analog sensor, and then the algorithm for the temperature sensor is direct as the data input to be taken from the sensor and display to the application. The virtual channel is set as channel 25 for AC Voltage and channel 41 for AC Current will show up in the Cayenne IoT dashboard platform.



```
#include <CayenneMQTTESP32.h> // Library for Cayenne
#define CAYENNE_DEBUG
#include "EmonLib.h"
#define CAYENNE_PRINT Serial
#define vCalibration 115.0
#define currCalibration 8
DHT dht(5, DHT11);
EnergyMonitor emon;

// WiFi network info.
char ssid[] = "mazedband"; // "TP-Link_41C9";
char wifiPassword[] = "Badin0509"; // "Research_6";

// Cayenne authentication info: This should be obtained from the Cayenne Dashboard.
char username[] = "0952d3a0-23e8-11eb-8779-7d56e82df461";
char password[] = "2d729c01d97f30c3999dc8d5a72028f0e408fc2e";
char clientID[] = "148c1660-2518-11eb-a2e4-b32ea624e442";

float kWh = 0;
unsigned long lastmillis = millis();

void setup() {
  Serial.begin(115200); // Baud rate for serial monitor
  dht.begin(); // To start communication with DHT sensor
  emon.voltage(35, vCalibration, 1.7); // Voltage: input pin, calibration, phase_shift
  emon.current(34, currCalibration); // Current: input pin, calibration.
  Cayenne.begin(username, password, clientID, ssid, wifiPassword); // To Authenticate cayenne
  digitalWrite(2, HIGH); // LED Indicator
}
```

Figure 3.18: Source Code for AC Voltage Sensor

```

void loop() {
  delay(2000);      // Give some delay between each Sensor reading
  Cayenne.loop();
  emon.calcVI(20, 2000);
  float h = dht.readHumidity();    // To get Humidity reading
  float t = dht.readTemperature(); // To get temperature reading

  Serial.println(" *C ");
  Serial.print("Vrms: ");
  Serial.print(emon.Vrms, 2);
  Serial.print("V");
  Serial.print("\tIrms: ");
  Serial.print(emon.Irms, 4);
  Serial.print("A");
  Serial.print("\tPower: ");
  Serial.print(emon.apparentPower, 4);
  Serial.print("W");
  Serial.print("\tkWh: ");
  kWh = kWh + emon.apparentPower*(millis()-lastmillis)/3600000000.0;
  Serial.print(kWh, 4);
  Serial.println("kWh");
  lastmillis = millis();
  Cayenne.virtualWrite(27, h, "rel_hum", "p");    // To write the Sensor reading to Cayenne Dashboard using channel 0
  Cayenne.virtualWrite(1, t, "temp", "c");        // To write the Sensor reading to Cayenne Dashboard using channel 1
  Cayenne.virtualWrite(25, emon.Vrms, "voltage", "v");
  Cayenne.virtualWrite(41, emon.Irms, "current", "a");
  Cayenne.virtualWrite(22, emon.apparentPower, "pow", "w");
  Cayenne.virtualWrite(29, kWh, "energy", "kwh");
}

CAYENNE_IN(2)
{
  digitalWrite(2, !getValue.asInt()); // to get the value from the website
}

```

Figure 3.19: Source Code for AC Current Sensor

3.6.1.4 Power Consumption Source Code

The `#include <EmonLib.h>` is stated as Energy and Power Consumption Calculation enable by Energy Monitor library is loaded. The `emon.calcVI` is to calculate all No. of half wavelengths (crossings), time-out in the AC circuit. Energy Monitor library enable the calculation Energy and Power Consumption from the variable of voltage and current in the circuit and the value calculated is direct as the data input to be taken from the variables and display to the application. The virtual channel is set as channel 22 for Power Consumption and channel 29 for Energy will show up in the Cayenne IoT dashboard platform.

```

void loop() {
  delay(2000);      // Give some delay between each Sensor reading
  Cayenne.loop();
  emon.calcVI(20, 2000);
  float h = dht.readHumidity();    // To get Humidity reading
  float t = dht.readTemperature(); // To get temperature reading

  Serial.println(" *C ");
  Serial.print("Vrms: ");
  Serial.print(emon.Vrms, 2);
  Serial.print("V");
  Serial.print("\tIrms: ");
  Serial.print(emon.Irms, 4);
  Serial.print("A");
  Serial.print("\tPower: ");
  Serial.print(emon.apparentPower, 4);
  Serial.print("W");
  Serial.print("\tkWh: ");
  kWh = kWh + emon.apparentPower*(millis()-lastmillis)/3600000000.0;
  Serial.print(kWh, 4);
  Serial.println("kWh");
  lastmillis = millis();
  Cayenne.virtualWrite(27, h, "rel_hum", "p");    // To write the Sensor reading to Cayenne Dashboard using channel 0
  Cayenne.virtualWrite(1, t, "temp", "c");        // To write the Sensor reading to Cayenne Dashboard using channel 1
  Cayenne.virtualWrite(25, emon.Vrms, "voltage", "v");
  Cayenne.virtualWrite(41, emon.Irms, "current", "a");
  Cayenne.virtualWrite(22, emon.apparentPower, "pow", "w");
  Cayenne.virtualWrite(29, kWh, "energy", "kwh");
}

```

Figure 3.20: Source Code for Power Consumption

3.6.2 Raspberry Pi OS (SNMP System via FTP Server)

There is operating system which been used in this project. Raspberry Pi OS is used to program for the SNMP Protocol to be function. The code is written in C++ and be installed by using command prompt inside the OS that is powered by the microprocessor chip. These codes contain part of which are for monitor the broadcasting system using SNMP Protocol.

```

pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 3.2 /etc/proftpd/proftpd.conf

# /etc/proftpd/proftpd.conf -- This is a basic ProFTPD configuration file.
# To really apply changes, reload proftpd after modifications, if
# it runs in daemon mode. It is not required in inetd/xinetd mode.
#
# Includes DSO modules
Include /etc/proftpd/modules.conf

# Set off to disable IPv6 support which is annoying on IPv4 only boxes.
UseIPv6 on
# If set on you can experience a longer connection delay in many cases.
IdentLookups off

ServerName "Debian"
# Set to inetd only if you would run proftpd by inetd/xinetd.
# Read README.Debian for more information on proper configuration.
ServerType standalone
DeferWelcome off

[ Read 191 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

```

Figure 3.21: FTP Server Configuration for SNMP

Enabling the SNMP System via FTP Server is to install proftpd by using ‘*sudo apt install proftpd*’ in the command prompt. Then configure the FTP server by using ‘*sudo nano /etc/proftpd/proftpd.conf*’ command as shown in Figure. This configuration GNU is to enable the server to extract data from the SNMP Protocol then submit the data as input in Cayenne IoT Platform.

```

pi@raspberrypi:~$ sudo service proftpd reload
pi@raspberrypi:~$ sudo service proftpd status
● proftpd.service - LSB: Starts ProFTPD daemon
   Loaded: loaded (/etc/init.d/proftpd; generated)
   Active: active (running) since Wed 2021-06-02 04:38:42 +08; 1 day 19h ago
     Docs: man:systemd-sysv-generator(8)
   Process: 505 ExecStart=/etc/init.d/proftpd start (code=exited, status=0/SUCCESS)
   Process: 12077 ExecReload=/etc/init.d/proftpd reload (code=exited, status=0/SUCCESS)
    Tasks: 1 (limit: 2062)
   CGroup: /system.slice/proftpd.service
           └─601 proftpd: (accepting connections)

Jun 04 00:35:57 raspberrypi systemd[1]: Reloading LSB: Starts ProFTPD daemon.
Jun 04 00:35:57 raspberrypi proftpd[12077]: Reloading ftp server: proftpd
Jun 04 00:35:57 raspberrypi systemd[1]: Reloaded LSB: Starts ProFTPD daemon.
Warning: Journal has been rotated since unit was started. Log output is incomplete
lines 1-14/14 (END)

```

Figure 3.22: FTP Server is active

After that, to ensure the FTP Server is running by using '*sudo service proftpd reload*' command and then '*sudo service proftpd status*' command. The FTP Server monitor status will prompt up and active in running state. Lastly, configuration at DEVA Hardware to ensure the connection is established with the Server and the DEVA hardware itself.

3.6.3 Cayenne IoT Platform Dashboard

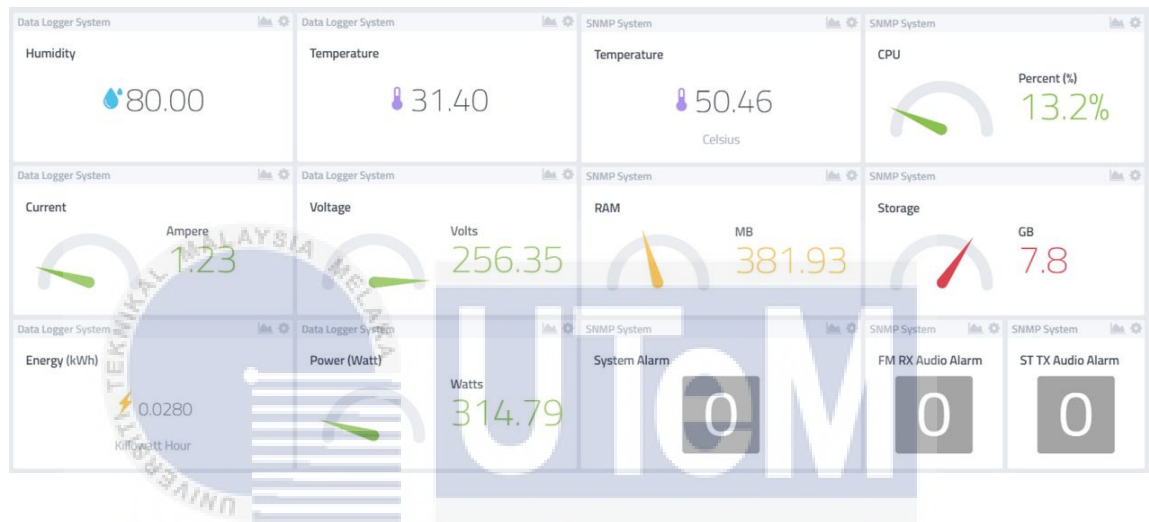


Figure 3.23: Cayenne IoT Platform Dashboard

The Figure 2.23 shows the dashboard platform of the system. the data shows in virtual channel where it set under void loop stated. Each of the code are provide cayenne virtual channel where is set in the form of number. For this project, there 6 virtual channels have been set for Data Logger System powered by NodeMCU ESP32 and 7 virtual channels have been set for SNMP System powered by Raspberry Pi 3. The Cayenne platform can show the data in multiple timelines.

There are 13 virtual channels in total set as inputs. The input virtual channel will show the data from the sensors and variables. The data for temperature sensor it will show in Celsius and for humidity sensor it will show percentage. The data for AC Voltage sensor it will show in AC Voltage and for Current sensor it will show in AC

Current. The data for Energy variable it will show in Kilowatt Hour and for Power sensor it will show in Watts.

3.6.4 Webhook



Figure 3.24: Webhook configuration in Cayenne IoT Platform

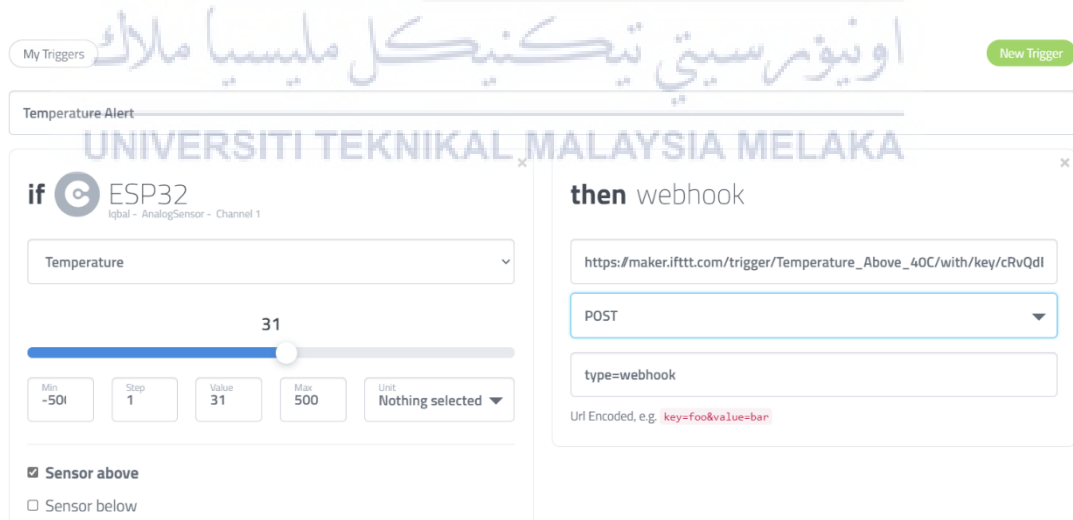


Figure 3.25: If Then statement configuration

In Cayenne IoT Platform, enable the push notification by set new trigger in this term of 'if then' statement and this system consist of Temperature Alert and Power Notification. To set the webhooks, pick a sensor from the system in the 'if' statement

then set a threshold to trigger the alarm then copy the link to the ‘then’ statement by obtain the webhook ID link from IFTTT.

3.6.5 IFTTT

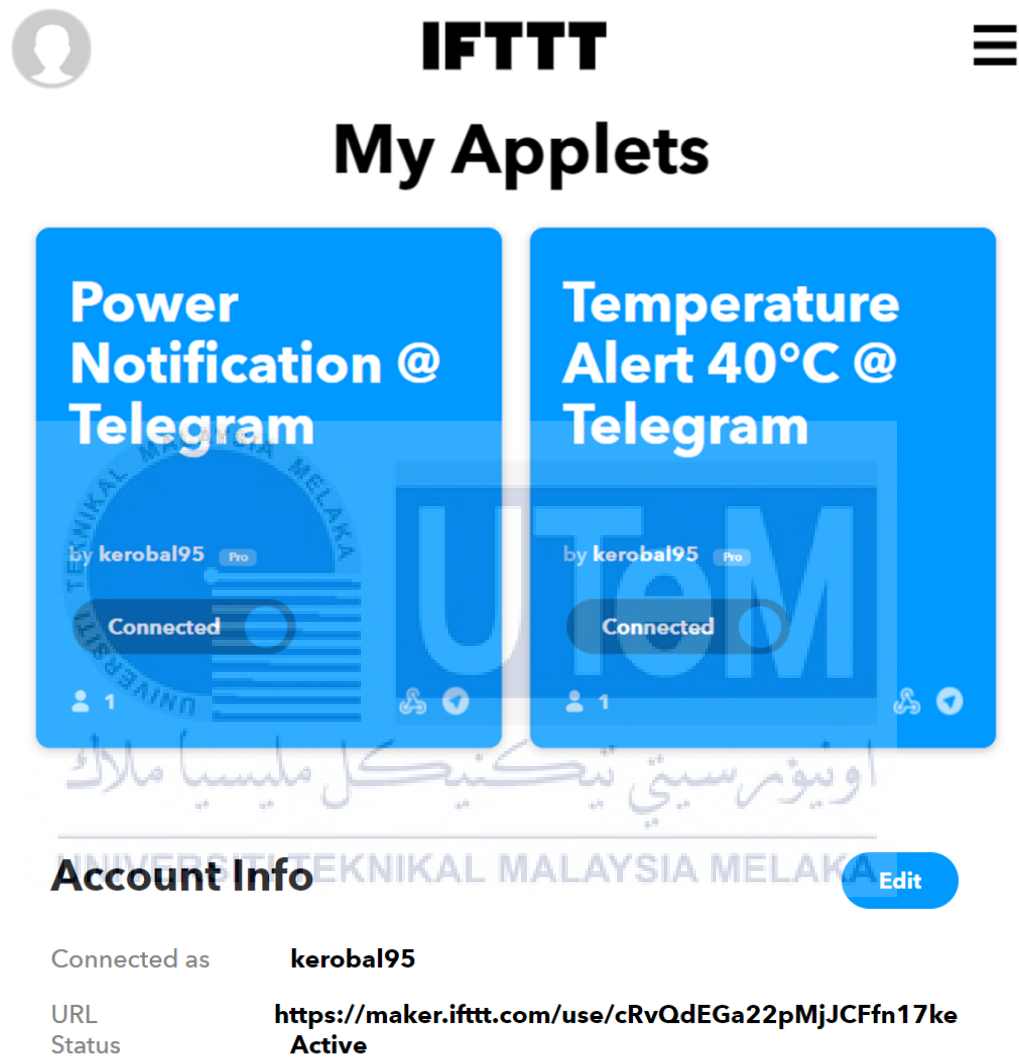


Figure 3.26: IFTTT Applets

In IFTTT, the cloud service acting as a middleman for the system and Telegram or E-mail client. To enable the system to push notification, create the applet with the designated notification or alarm, then obtain the link provided by IFTTT to be use as Webhook in Cayenne IoT Platform.

3.6.6 Telegram

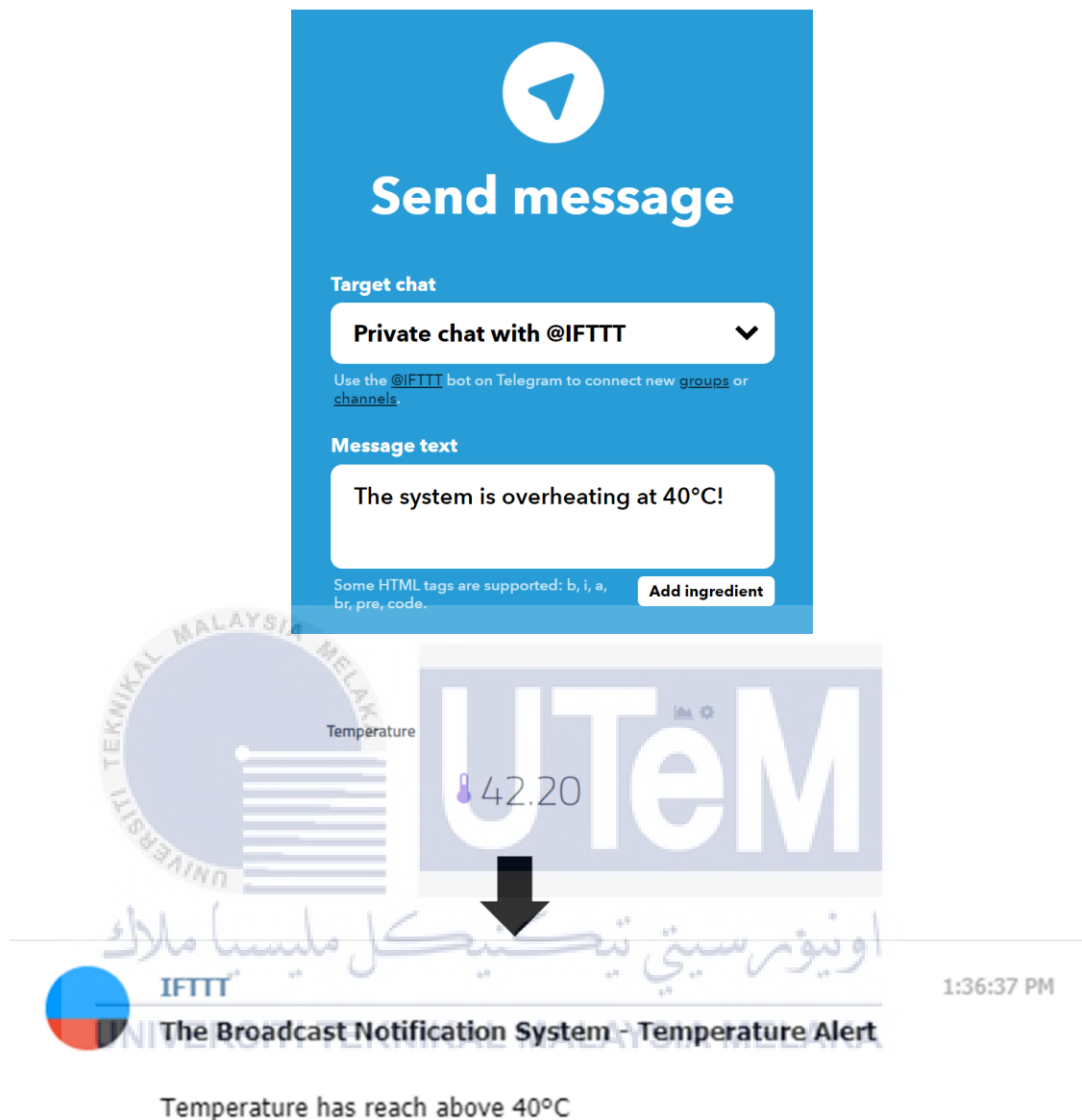


Figure 3.27: Telegram Configuration and Alert Notification

By using IFTTT bot in Telegram, the push notification can send alert if 'if else' statement triggered at the Cayenne IoT side. This can be achieved by enabling the IFTTT connected with both Cayenne IoT and Telegram bot functionality.

3.7 Hardware Development

The hardware equipment used to build this project is NodeMCU-ESP32, temperature and humidity sensor, AC Voltage sensor, AC Current sensor, and LAN Adapter. All the result obtained from this research is real-time experiment and real time implementation network. To obtain the data, the experiment is implemented to a server rack and run it for 2 months to get the data measurement and comparison of the result.

3.7.1 Project Circuit

This system consists of two circuit that integrate to one IoT Data Logger System.

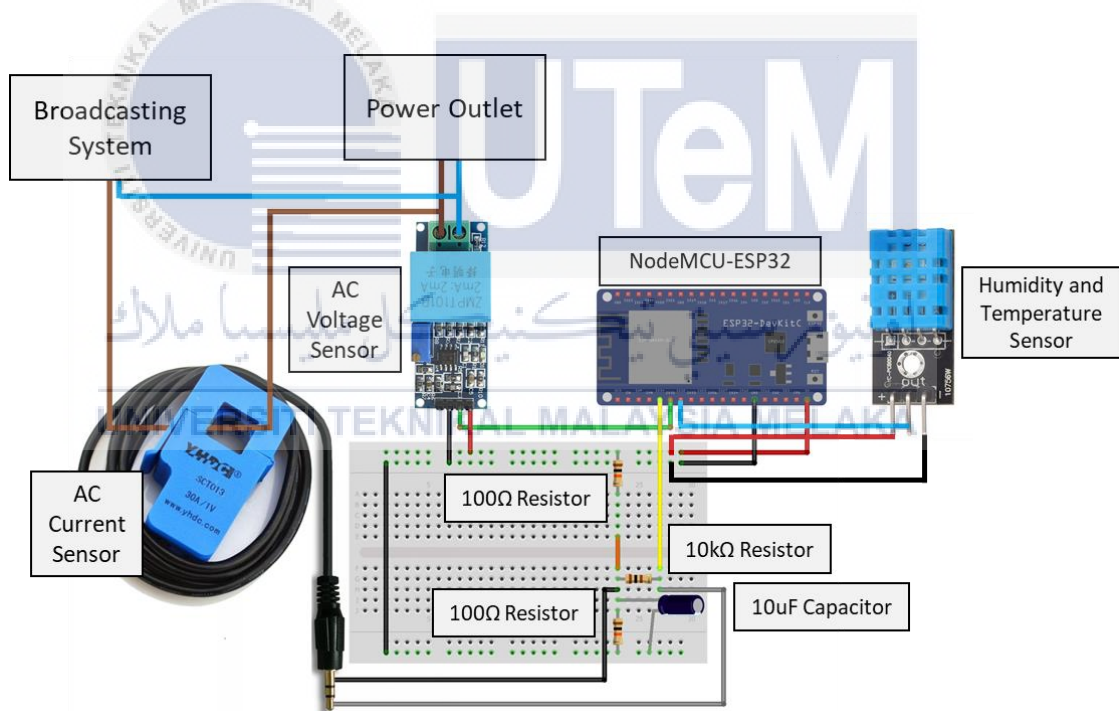


Figure 3.28: Circuit Diagram for Power, Humidity and Temperature Monitoring System

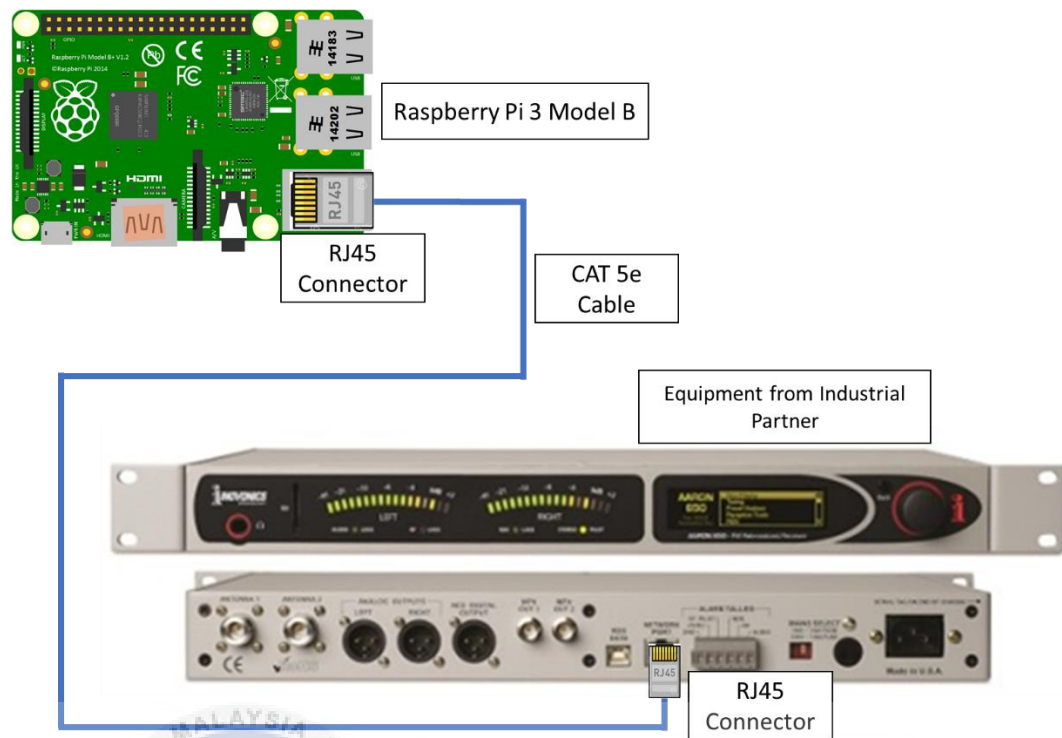


Figure 3.29: Circuit Diagram for SNMP Data Monitoring System

3.7.2 Project Hardware

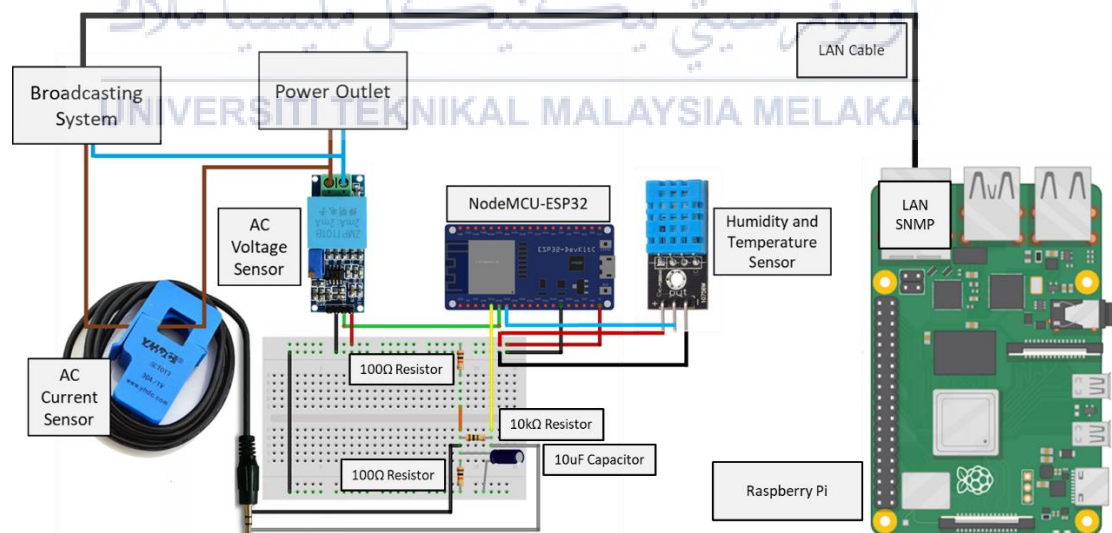


Figure 3.30: Hardware Circuit Diagram

3.7.3 Project location

This project is suitable for broadcasting studio with server rack of multiple broadcasting system equipment.



Figure 3.31: Server Rack

The server rack consists of networking equipment such as HUAWEI HG8240H Fibre modem, TP-Link Archer C5 router, Huawei WiFi AX3 router, TP-Link 16-Port Gigabit Ethernet Switch, HDMI Splitter, HDMI to Ethernet adapter and Reolink DVR CCTV Unit as shown in Figure 3.31.

This can be used as a controlled location to simulate the operation of the project. We are able to collaborate online with the industrial partner because of the COVID-19 pandemic and Restriction Movement Control Order (MCO) are being imposed by the Government of Malaysia concurrent with the project progression.

CHAPTER 4

RESULTS AND DISCUSSION



4.1 Introduction

There is various type of data output has been collected by multiple hardware.

The data is manually updated via Microsoft Excel for visualize the data systematically.

The hardware selected for this project is Mi Home Temperature and Humidity Sensor, Monitor Power Meter, TP-Link 16-Port Gigabit Ethernet Switch and DEVA Broadcasting System.

4.1.1 Mi Home Temperature and Humidity Sensor

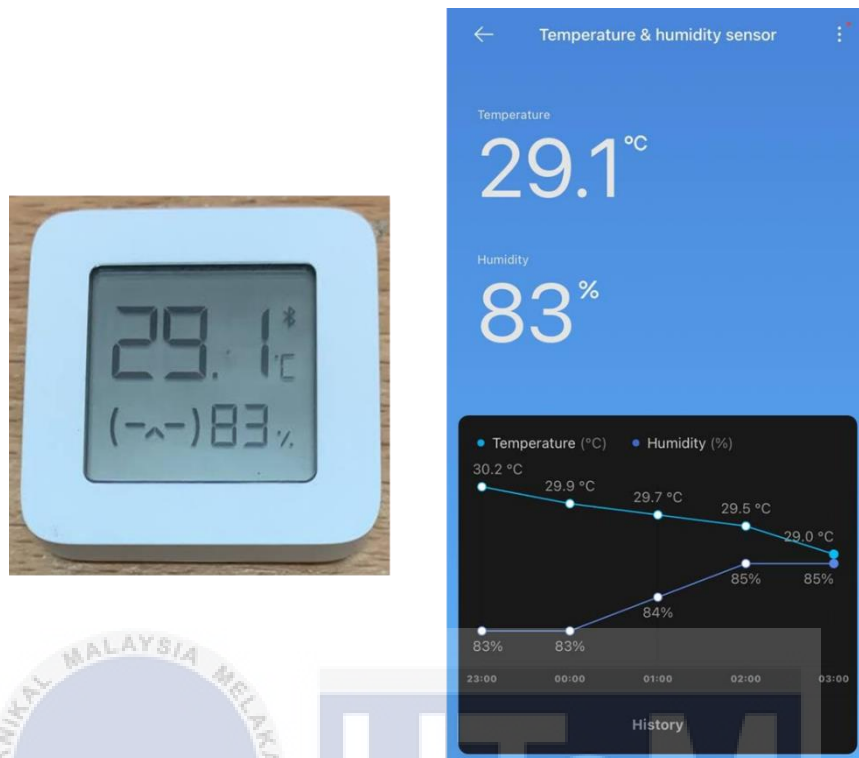


Figure 4.1: Mi Home Temperature and Humidity Sensor and Mi Home GUI

Mi Home Temperature and Humidity Sensor from Xiaomi is used to monitor the temperature and humidity variables in this project. The placement of the sensor is same as the DHT11 sensor that is implemented in this project to ensure most accurate reading in the same server rack environment. The data is extracted from 20/04/2021 to 21/04/2021 by using Mi Home app with interval of 1 Hour.

4.1.2 Power Monitor Meter



Figure 4.2: Power Monitor Meter

Power Monitor Meter from OEM is used to monitor the voltage, current, power consumption and energy usage variables in this project. The placement of the meter is the wall socket for the server and the input socket of the system. The data is extracted from 20/04/2021 to 21/04/2021 by using the built-in function to the meter to show 10 days recorded data with interval of 1 Hour.

4.1.3 TP-Link 16-Port Gigabit Ethernet Switch

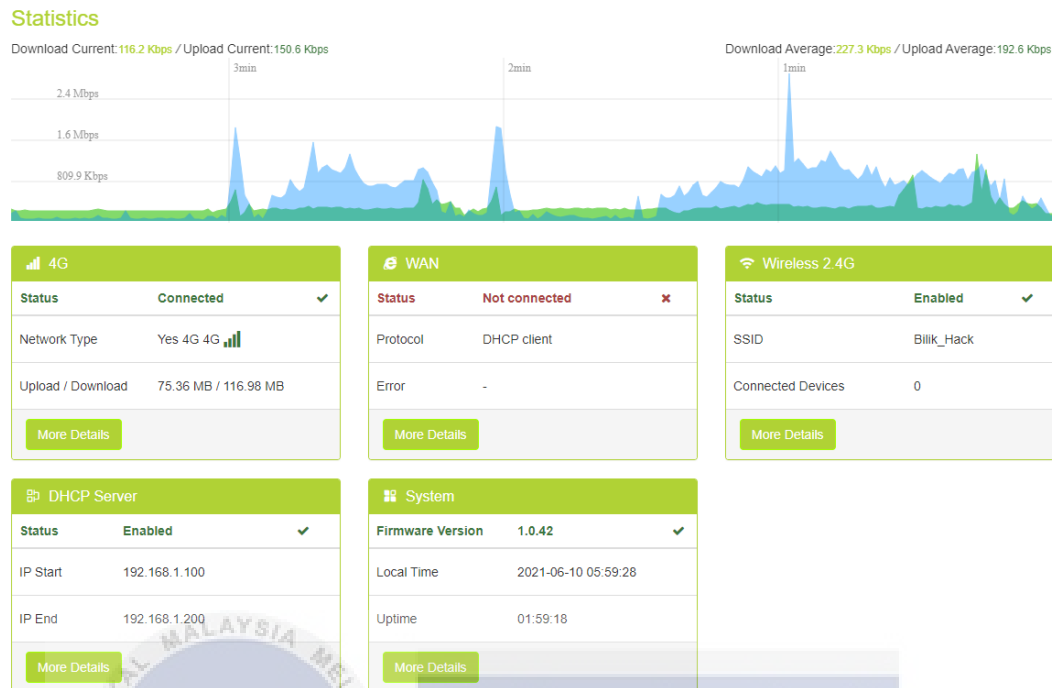


Figure 4.3: TP-Link Switch Statistic Manager

16-Port Gigabit Ethernet Switch from TP-Link is used to monitor the network speed variables in this project. The placement of the switch is inside the server rack itself and it is connected with the system via SNMP Protocol. The data is extracted from 20/04/2021 to 21/04/2021 by using the statistics function in the switch GUI with 3 months recorded data with interval of 1 Hour.

4.2 Software Data Measurement

There is various type of data output has been collected by multiple software that integrated to this system. The data is automatically updated via respected system mention below and to ease the data compilation Microsoft Excel is use for visualizing the data systematically. The software integrated for this project is myDevices Cayenne IoT Platform, Arduino IDE and Raspberry Pi OS.

4.2.1 myDevices Cayenne IoT Platform

myDevices Cayenne IoT Platform provide cloud storage service to save all the data into their server across the world, in this case is it at Singapore. In addition, this service can be used to extract data from all sensors that is integrated in the Cayenne IoT Platform. In figure below, shows 2 methods for extraction data: -

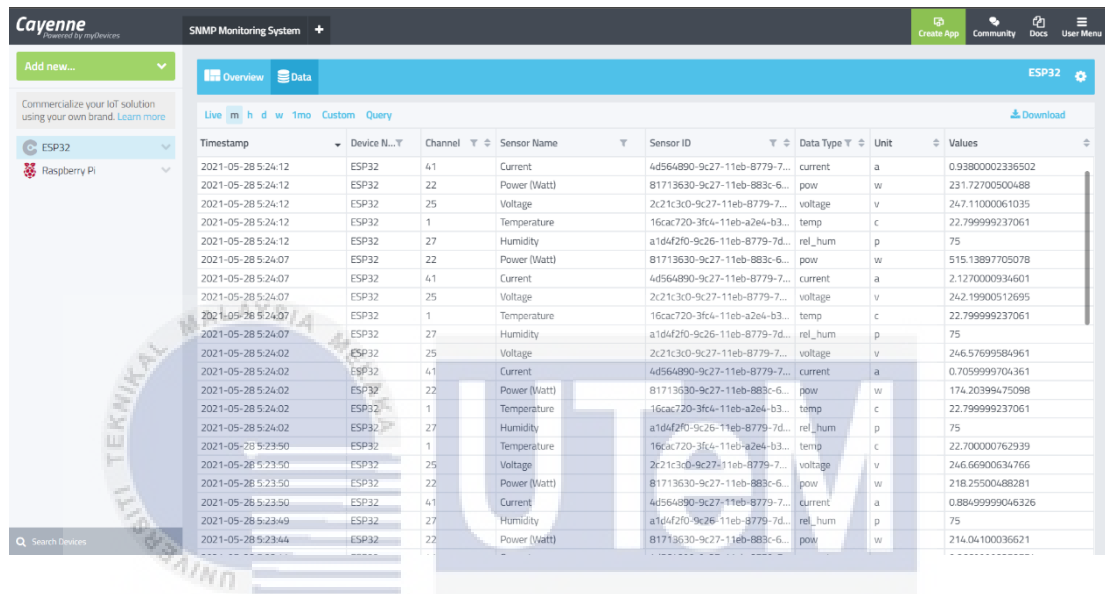


Figure 4.4: Data Extraction from all sensor data received in specific time

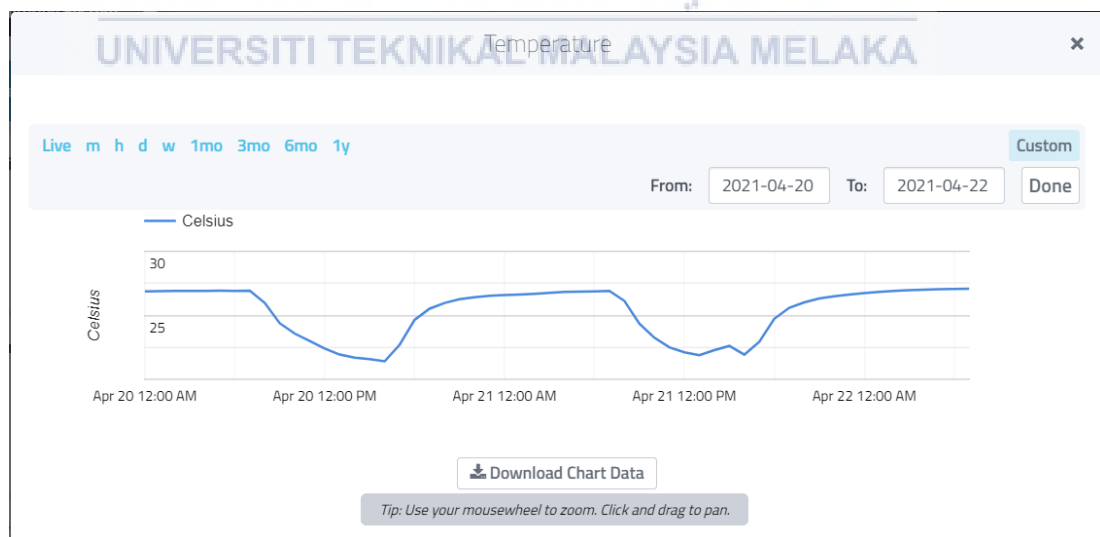


Figure 4.5: Data Extraction from specific sensor data received in specific time

All the data is extract dated from 20/04/2021 to 21/04/2021 and has been process via Microsoft Excel to make sure all the data is genuine and can be used to compare with other data set to verify the authenticity and the accuracy of this IoT Platform.

4.2.2 Arduino IDE

Serial Monitor powered by Arduino IDE is used to extract data from the NodeMCU ESP32 itself from a USB connected to the notebook. The configuration for the recorded data is connected via COM3 and the serial data transmission 115200 baud with no delay in the programming of the project. The data extraction is dated from 20/04/2021 to 21/04/2021 and all the text from the serial monitor is copy to notepad than convert to excel with average interval of 1 hour.

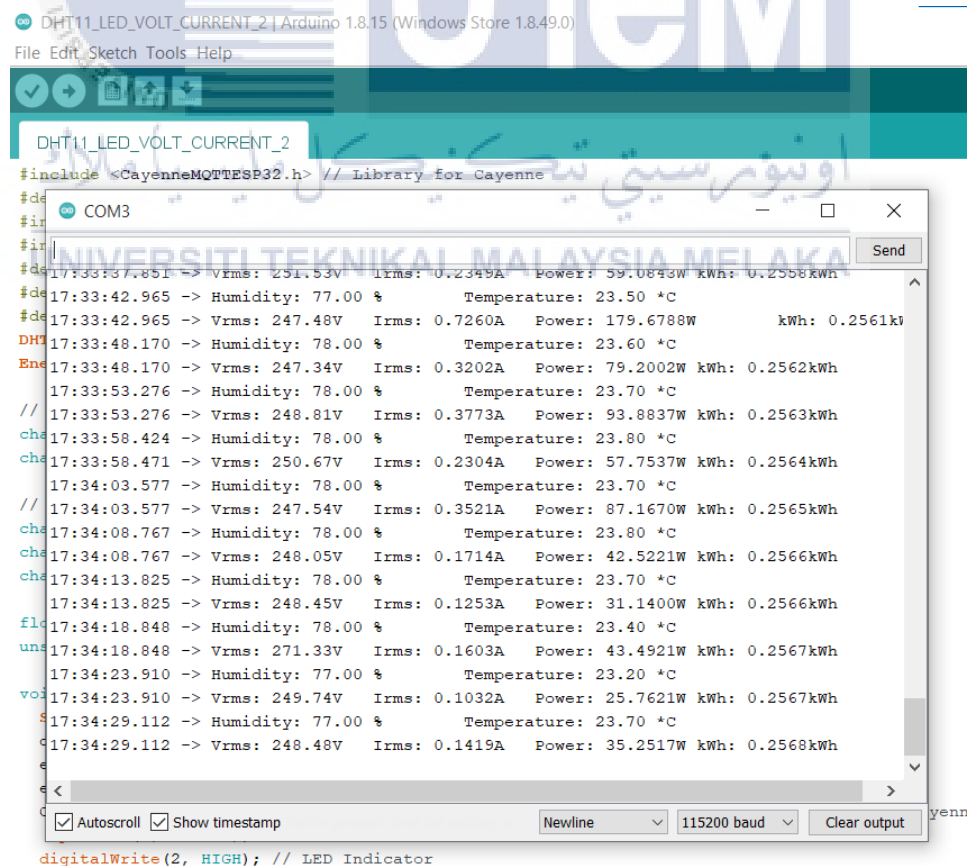


Figure 4.6: Arduino IDE Serial Monitor

4.2.3 Raspberry Pi OS

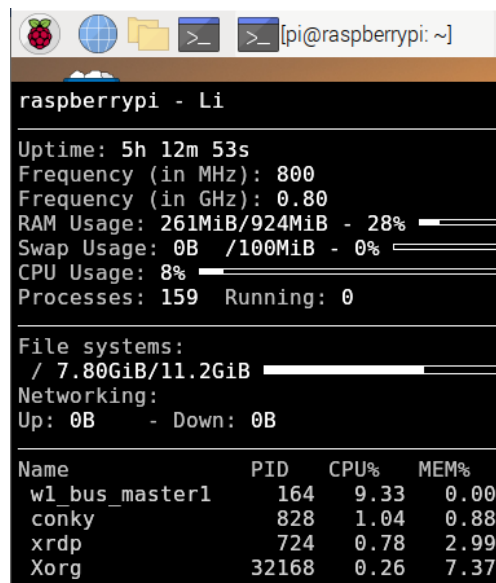


Figure 4.7: Raspberry Pi OS Resource Monitor

By using the GUI of Raspberry Pi OS, the resource monitor or chunky as it nicknamed is install to the OS and used to monitor the network speed of the server. The Raspberry Pi is installed with remote desktop functionality to ease of use in extract the data remotely and the GUI is screen recorded dated from 20/04/2021 to 21/04/2021. The recoded footage is analyzed manually and compiled in excel with interval of 1 hour.

4.3 Data Measurement and Comparison

The data obtain from the hardware and software has been processed to compare the accuracy of the sensors and system that integrated to this IoT Data Logger System. The data and measurements has been taken from 20/04/2021 until 21/04/2021 with 1 hour interval for about 2 days. All the data will be systematically proceeded via Microsoft Excel. In the hardware side, the data has been collected manually and the data for the software side will be extracted and processed from its respected system.

4.3.1 Temperature

The temperature data is gathered using Cayenne IoT Platform, Arduino IDE both powered by DHT11 Temperature and Humidity Sensor and Mi Home Temperature and Humidity Sensor. The data is extracted and analyzed for two days with 1 hour interval using Microsoft Excel.

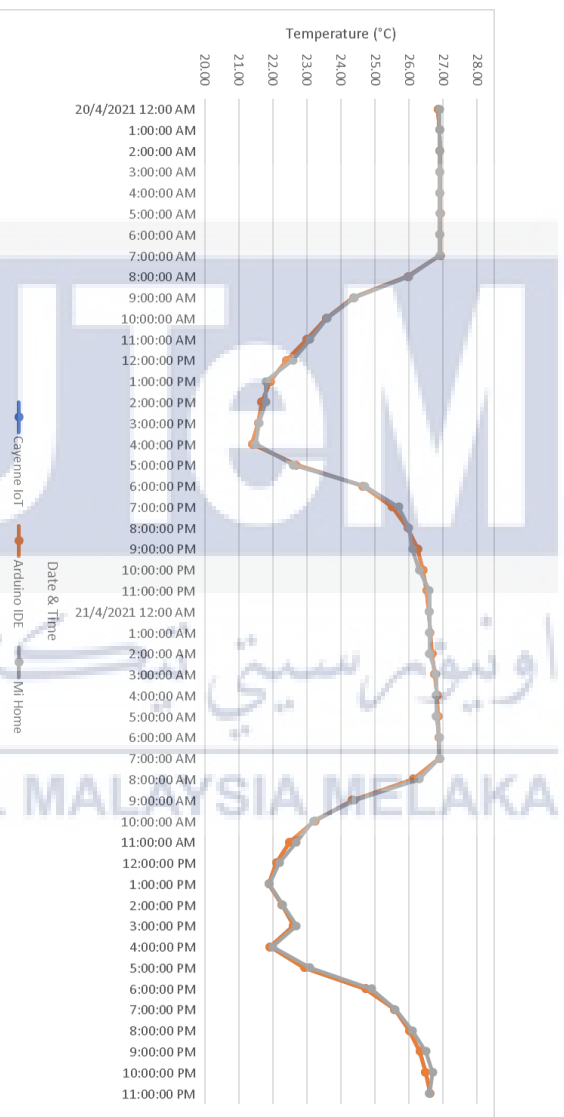


Figure 4.8: Line Graph Temperature Variable

Based on the line graphs for Cayenne IoT and Arduino IDE are the same as displayed in the graph above. As a result, the line graphs for Cayenne IoT and Mi Home sensor are slightly differed to one another. The line graph Cayenne IoT with Arduino IDE tends to drop the temperature from 27°C to 22°C because the server rack is air-conditioned from 8AM to 6PM. The line graph for the Mi Home sensor then showed conflicting line graphs with 0.30 Percentage of Error when compared to the line graphs of Cayenne IoT and Arduino IDE.

4.3.2 Humidity

The humidity data is gathered using Cayenne IoT Platform, Arduino IDE both powered by DHT11 Temperature and Humidity Sensor and Mi Home Temperature and Humidity Sensor. The data is extracted and analyzed for two days with 1 hour interval using Microsoft Excel.

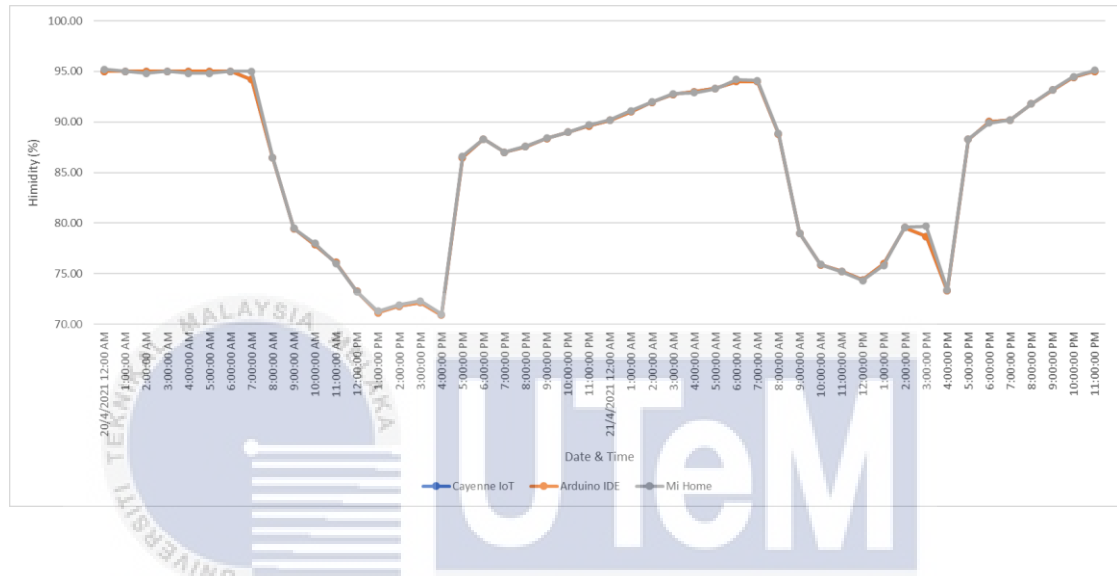


Figure 4.9: Line Graph Humidity Variable

Based on the line graphs for Cayenne IoT and Arduino IDE are the same as displayed in the graph above. As a result, the line graphs for Cayenne IoT and Mi Home sensor are slightly differed to one another. The line graph Cayenne IoT with Arduino IDE tends to drop the humidity from 95% to 70% because the server rack is air-conditioned from 8AM to 6PM. The line graph for the Mi Home sensor then showed conflicting line graphs with 0.13 Percentage of Error when compared to the line graphs of Cayenne IoT and Arduino IDE. This result shows that both Temperature and Humidity sensor integrated with project is concurrent with Mi Home Sensor control variable are accurate with slight margin of error.

4.3.3 AC Voltage

The AC Voltage data is gathered using Cayenne IoT Platform, Arduino IDE both powered by ZMPT101B AC Voltage Sensor and Power Monitor Meter. The data is extracted and analyzed for two days with 1 hour interval using Microsoft Excel.

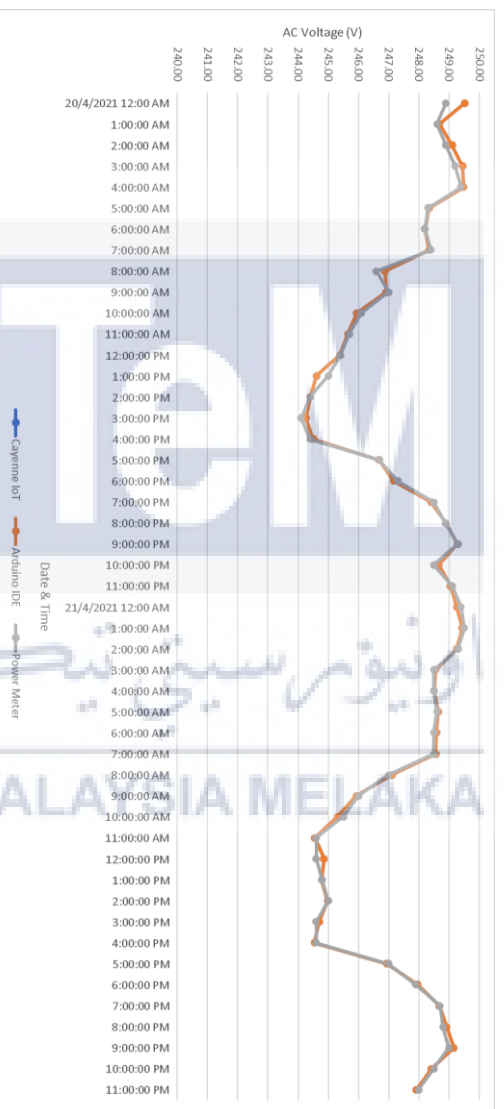


Figure 4.10: Line Graph AC Voltage Variable

Based on the line graphs for Cayenne IoT and Arduino IDE are the same as displayed in the graph above. As a result, the line graphs for Cayenne IoT and Power Monitor Meter are slightly differed to one another. The line graph Cayenne IoT with Arduino IDE tends to drop the voltage from 244V to 249V. The line graph for the Power Monitor Meter then showed conflicting line graphs with 0.04 Percentage of Error when compared to the line graphs of Cayenne IoT and Arduino IDE.

4.3.4 AC Current

The AC Voltage data is gathered using Cayenne IoT Platform, Arduino IDE both powered by SCT-013-000 AC Current Sensor and Power Monitor Meter. The data is extracted and analyzed for two days with 1 hour interval using Microsoft Excel.

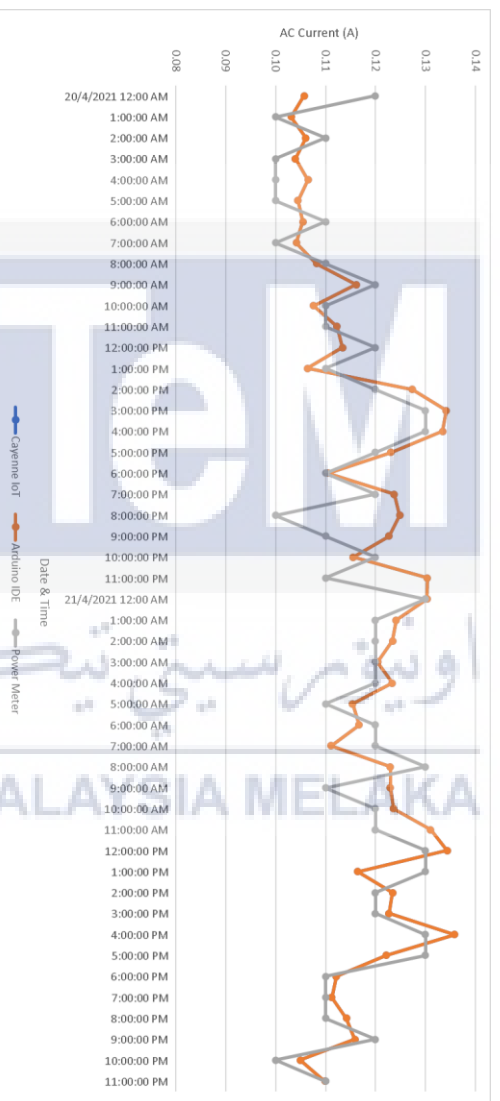


Figure 4.11: Line Graph AC Current Variable

Based on the line graphs for Cayenne IoT and Arduino IDE are the same as displayed in the graph above. As a result, the line graphs for Cayenne IoT and Power Monitor Meter are slightly differed to one another. The line graph Cayenne IoT with Arduino IDE tends to drop the current from 0.10A to 0.14A because the server tends to run more load when accessing the Cayenne IoT Platform remotely. The line graph for the Power Monitor Meter then showed distorted line graphs because its zoomed in to smallest scale but it is conflicted with 4.91 Percentage of Error when compared to the line graphs of Cayenne IoT and Arduino IDE.

4.3.5 Power Consumption

The Power Consumption data is gathered using Cayenne IoT Platform, Arduino IDE both powered by ZMPT101B AV Voltage Sensor and SCT-013-000 AC Current Sensor variables and calculated for show power consumption value using Arduino ESP32 CPU and Power Monitor Meter. The data is extracted and analyzed for two days with 1 hour interval using Microsoft Excel.

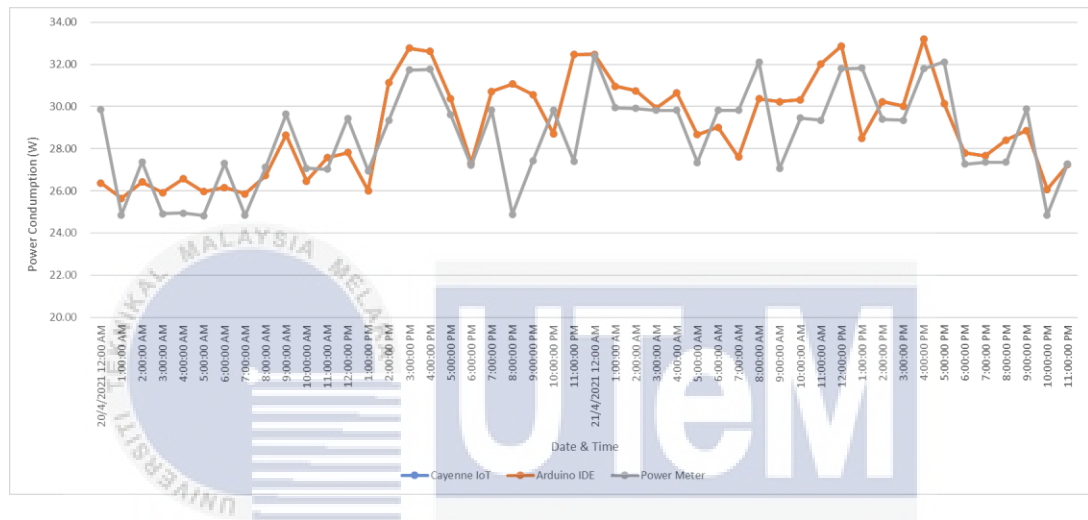


Figure 4.12: Line Graph Power Consumption Variable

Based on the line graphs for Cayenne IoT and Arduino IDE are the same as displayed in the graph above. As a result, the line graphs for Cayenne IoT and Power Monitor Meter are slightly differed to one another. The line graph Cayenne IoT with Arduino IDE tends to drop the power consumption from 25W to 33W because the server tends to run more load when accessing the Cayenne IoT Platform remotely. The line graph for the Power Monitor Meter then showed distorted line graphs because its zoomed in to smallest scale but it is conflicted with 4.89 Percentage of Error when compared to the line graphs of Cayenne IoT and Arduino IDE.

4.3.6 Network Speed

The Network Speed data is gathered using Cayenne IoT Platform, Resource Monitor powered by Raspberry Pi OS via SNMP Protocol and Network Switch Statistics. The data is extracted and analyzed for two days with 1 hour interval using Microsoft Excel.

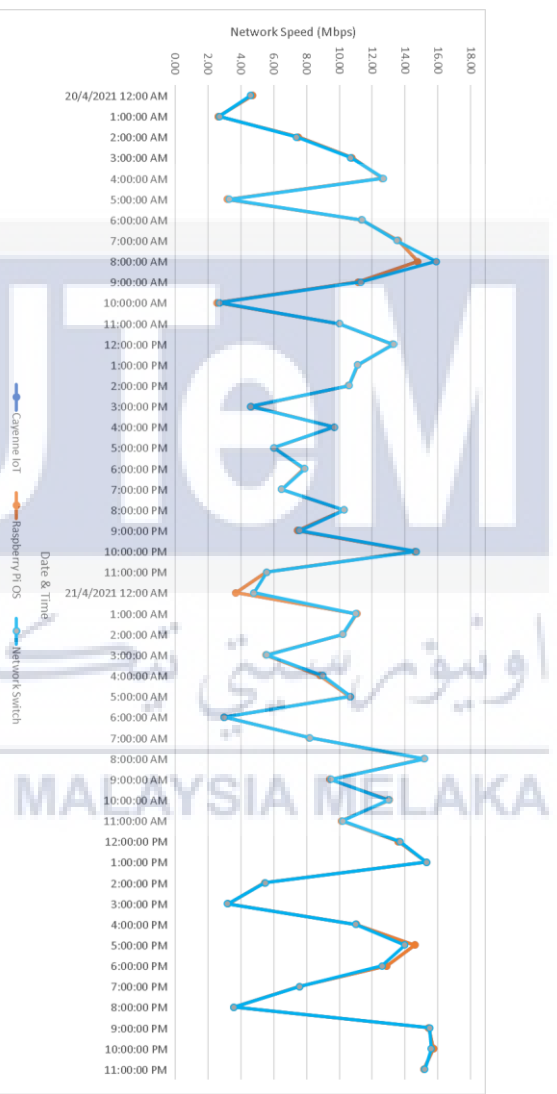


Figure 4.13: Line Graph Network Speed Variable

Based on the line graphs for Cayenne IoT and Raspberry Pi OS are the same as displayed in the graph above. As a result, the line graphs for Cayenne IoT and Raspberry Pi OS are slightly differed to one another. The line graph Cayenne IoT with Raspberry Pi OS tends to drop the power consumption from 2.5 Mbps to 15.7 Mbps because the server tends to run more load when accessing the Cayenne IoT Platform remotely. The line graph for the Network Switch Statistics then showed distorted line graphs because it's zoomed in to smallest scale, but it is conflicted with 1.52 Percentage of Error when compared to the line graphs of Cayenne IoT and Raspberry Pi OS.

4.4 Discussion

4.4.1 Circuit Final Configuration

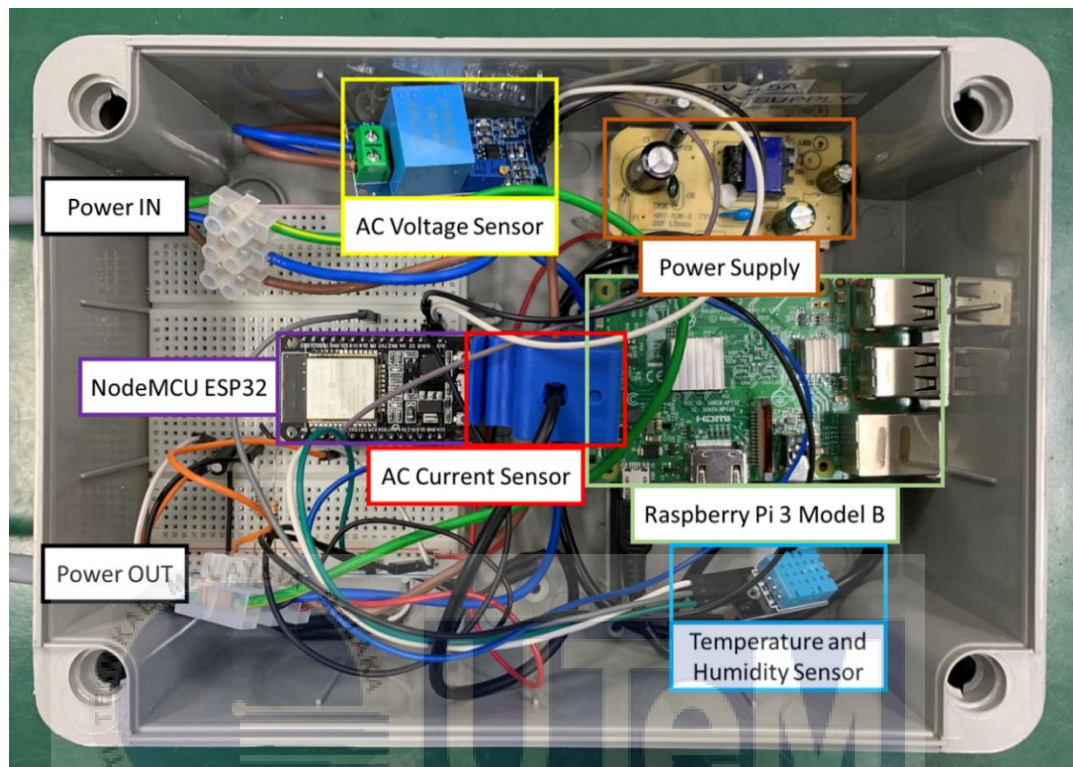


Figure 4.14: Top View of the IoT Data Logger Hardware

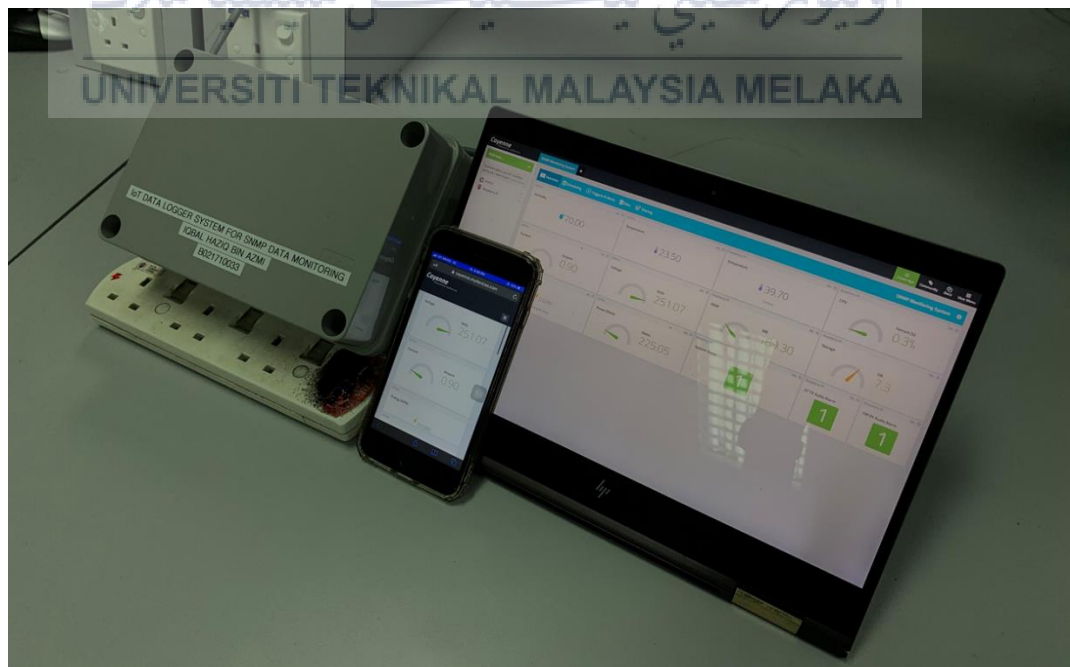


Figure 4.15: Overall View of the IoT Data Logger Hardware

4.4.2 Field Test

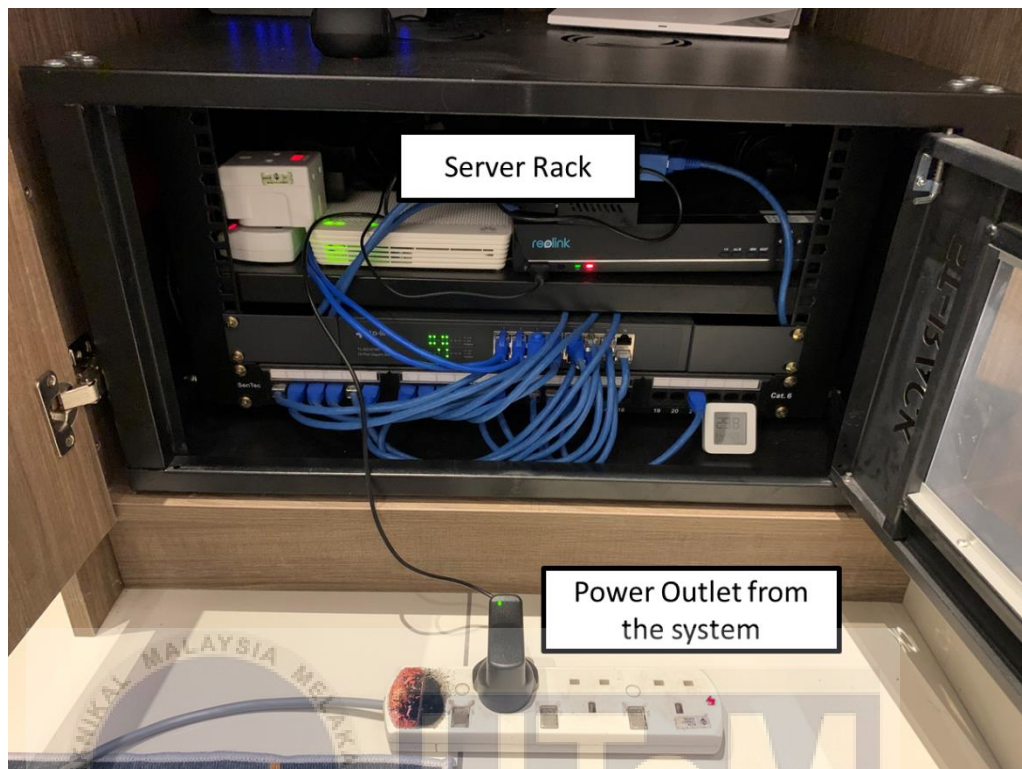


Figure 4.16: Server Rack

4.4.3 Result

The Cayenne IoT platform is used to monitor sensor input and alert system.

The data variables are from combination of the NodeMCU ESP32 microcontroller and Raspberry Pi 3 microprocessor is transferred to the Cayenne IoT cloud through a wireless internet connection.

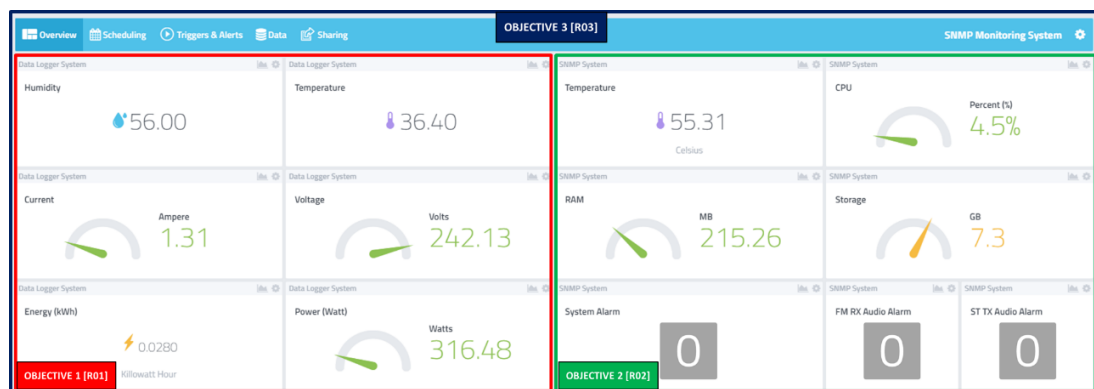


Figure 4.17: Cayenne IoT Platform Dashboard

The Cayenne IoT dashboard platform is described in Figure 4.18, with the sensor input labelled as **OBJECTIVE 1 [R01]**. The sensor displayed the current water parameter within the broadcasting system platform. Every minute, hour, day, month, and year, the data in the line graph can be monitored. Furthermore, the **OBJECTIVE 2 [R02]** labelled data logger system for broadcasting system parameters such as Main Audio Loss, AUX Loss, IP Audio Loss, Power status and GP IN Power Switch in the cloud. Overall, the **OBJECTIVE 3 [R03]** can be access at any internet connected device because IoT Cayenne Platform dashboard is powered by the cloud.

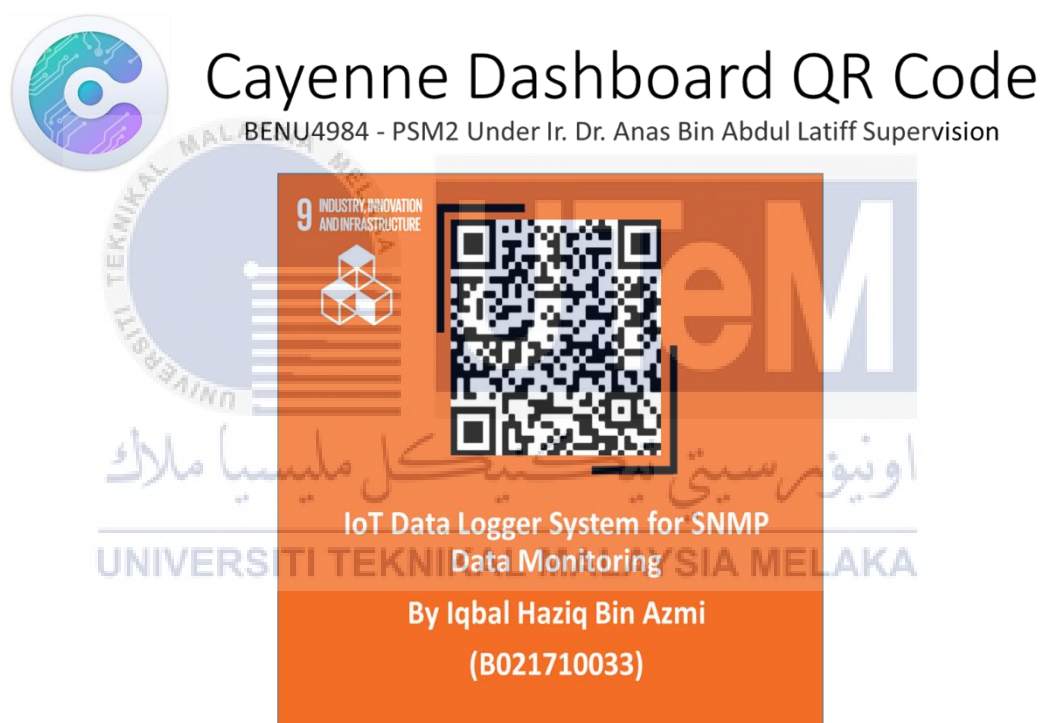


Figure 4.18: QR Code for the IoT Dashboard

Aside from that, the graphic above shows that other users can monitor the circumstance in real time by scanning the QR code in the Figure 4.18. The Cayenne IoT dashboard platform will be displayed.

CHAPTER 5

CONCLUSION AND FUTURE WORKS



5.1 Introduction

As a conclusion the project IoT Data Logger System for SNMP Data Monitoring system has been done successfully designed and tested. The system has been developed by integration of hardware and software. In addition, the hardware and software architecture of the system is designed to improved previous system available in the market. NodeMCU ESP32 and Raspberry Pi 3 that has been used in the system with combination of microcontroller and microprocessor that can process the operation for collecting data from the analog sensors [R01] and digital variables from SNMP Protocol designed for broadcasting system [R02]. Cayenne IoT Platform is used as dashboard to measure, monitor and store the data form both NodeMCU ESP32 and Raspberry Pi 3 and the data were displayed in the dashboard [R03]. The objectives that were set in the beginning of this project were successfully achieved

which is investigate the performance of temperature, humidity and electrical sensors [R01], design a stand-alone microcomputer-based remote data logger that able for measuring, monitoring and store the broadcasting system parameters such as Main Audio Loss, AUX Loss, IP Audio Loss, Power status and GP IN Power Switch in the cloud [R02] and develop stand-alone IoT microcontroller and microprocessor with Cayenne IoT Platform as a dashboard [R03].

5.2 United Nation Sustainable Development Goals

Consideration to the Sustainable Development Goals is crucial in developing and producing on IoT Data Logger System SNMP Data Monitoring System project because it is very important to debate about sustainability when developing country in consideration of implication in nature and society.

The United Nation Sustainable Development Goals for this project is Goal 9 Industry, Innovation, and Infrastructure. This project is in line with Goal 9 because it is powered by an IoT System will assists End-Users to monitor the Broadcasting System operation remotely and all the monitored data is recorded via cloud.



Figure 5.1: United Nation Sustainable Development Goals 9

5.3 Potential Penetration in Local Market

Consideration to the Jalinan Digital Negara (JENDALA) Nation Plan is crucial in developing and producing on IoT Data Logger System SNMP Data Monitoring System project because it is very important to enable local broadcasting partner implementation of IoT in their Broadcasting System on achieving the objective of JENDALA which is enable IoT implementation on all sector of the economy.

5.4 Recommendations

There is a few changes and improvement that can be implemented to this project.

These improvements are:

1. Implementation Raspberry Pi microprocessor for GPIO Functions
 - The implementation only using Raspberry Pi microprocessor is possible due to GPIO layout and program are similar with Arduino microcontroller architecture. Raspberry Pi OS also feature an ease-of-use Graphic User Interface (GUI) enable programmer write code directly to Raspberry Pi without using PC or Mac.
2. Implementation fail-safe function
 - The implementation fail-safe relay to control the power between the broadcasting system if any accident occurs. This function can help the broadcasting industry save cost on replace the expensive hardware.

REFERENCES

- [1] Babiuch, M., Foltynnek, P., & Smutny, P. (2019). Using the ESP32 microcontroller for data processing. Proceedings of the 2019 20th International Carpathian Control Conference, ICCCC 2019.
- [2] Rukmana, F. I., Akmaliah, Mulyana, E., Kusnawan, A., Kamelia, L., & Darmalaksana, W. (2020). All-in-one application for smart home system base on telegram controlled. Proceedings - 2020 6th International Conference on Wireless and Telematics, ICWT 2020, 18–21.
- [3] Bipasha Biswas, S., & Tariq Iqbal, M. (2018). Solar Water Pumping System Control Using a Low Cost ESP32 Microcontroller. Canadian Conference on Electrical and Computer Engineering, 2018-May.
- [4] Rantelinggi, P. H., Paiki, F. F., & Gadi, Y. (2020). Telematika Pemantau Suhu Menggunakan NodeMcu , IoT Dan Cayenne Pada Rack Server. 13(2), 80–90.
- [5] Wenxian Zeng, & Yue Wang. (2009). Design and Implementation of Server Monitoring System Based on SNMP. 1, 680–682

- [6] Huang, H. (2019). Architecture of audio broadcasting coverage monitoring system based on internet of things. Proceedings - 2019 IEEE International Conference on Smart Internet of Things, SmartIoT 2019, 320–324.
- [7] Hambali, P. A. M., Syamsuddin, Effendi, M. R., & Hamidi, E. A. Z. (2020). Prototype design of monitoring system base tranceiver station (BTS) base on internet of things. Proceedings - 2020 6th International Conference on Wireless and Telematics, ICWT 2020.
- [8] MCMC, “National Digital Infrastructure Lab (NDIL) for Jalinan Digital Negara (JENDELA) Plan”. 30, September, 2020 [Online serial]. Available:<https://www.mcmc.gov.my/skmmgovmy/media/General/pdf/NDIL-Report.pdf>. [Accessed Dec. 21, 2020].
- [9] A. Bhawiyuga, K. Amron, R. Pramanandha, D. P. Kartikasari, H. Arijudin and D. A. Prabandari, "LoRa-MQTT Gateway Device for Supporting Sensor-to-Cloud Data Transmission in Smart Aquaculture IoT Application," 2019 International Conference on Sustainable Information Engineering and Technology (SIET), Lombok, Indonesia, 2019, pp. 187-190, doi: 10.1109/SIET48054.2019.8986124.
- [10] Dibaba, H. (2018). “IoT Implementation with Cayenne Platform” Metropolia University of Applied Sciences, Helsinki, Finland.
- [11] Maier, A., Sharp, A., & Vagapov, Y. (2017, September). Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In 2017 Internet Technologies and Applications (ITA) (pp. 143-148). IEEE.

APPENDICES

Appendix A: IoT Data Logger System For SNMP Data Monitoring Source Code

```
#include <CayenneMQTTESP32.h> // Library for Cayenne
```

```
#define CAYENNE_DEBUG
```

```
#include "EmonLib.h"
```

```
#include "DHT.h" // Library for DHT sensor
```

```
#define CAYENNE_PRINT Serial
```

```
#define vCalibration 115.0
```

```
#define currCalibration 8
```

```
DHT dht(5, DHT11);
```

```
EnergyMonitor emon;
```

```
// WiFi network info.
```

```
char ssid[] = "mazeband"; // "TP-Link_41C9";
```

```
char wifiPassword[] = "Badin0509";// "Research_6";
```

```
// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
```

```
char username[] = "0952d3a0-23e8-11eb-8779-7d56e82df461";
```

```
char password[] = "2d729c01d97f30c3999dc8d5a72028f0e408fc2e";
```

```
char clientID[] = "148c1660-2518-11eb-a2e4-b32ea624e442";
```

```
float kWh = 0;
```

```
unsigned long lastmillis = millis();
```

```
void setup() {
```

```
    Serial.begin(115200); // Baud rate for serial monitor
```

```
    dht.begin(); // To start communication with DHT sensor
```

```
    emon.voltage(35, vCalibration, 1.7); // Voltage: input pin, calibration, phase_shift
```

```
    emon.current(34, currCalibration); // Current: input pin, calibration.
```

```
    Cayenne.begin(username, password, clientID, ssid, wifiPassword); // To
```

```
Authenticate cayenne
```



```

pinMode(2, OUTPUT);

digitalWrite(2, HIGH); // LED Indicator

}

void loop() {

    delay(2000);    // Give some delay between each Sensor reading

    Cayenne.loop();

    emon.calcVI(20, 2000);

    float h = dht.readHumidity();    // To get Humidity reading

    float t = dht.readTemperature(); // To get temperature reading

    if (isnan(h) || isnan(t)) {    // Condition to check whether the sensor reading was
successful or not

        Serial.println("Failed to read from DHT sensor!");

        return;

    }

    Serial.print("Humidity: ");    // To print the data in serial monitor

    Serial.print(h);

```

```
Serial.print(" %\t Temperature: ");
```

```
Serial.print(t);
```

```
Serial.println(" *C ");
```

```
Serial.print("Vrms: ");
```

```
Serial.print(emon.Vrms, 2);
```

```
Serial.print("V");
```

```
Serial.print("\tIrms: ");
```

```
Serial.print(emon.Irms, 4);
```

```
Serial.print("A");
```

```
Serial.print("\tPower: ");
```

```
Serial.print(emon.apparentPower, 4);
```

```
Serial.print("W");
```

```
Serial.print("\tkWh: ");
```

```
kWh = kWh + emon.apparentPower*(millis()-lastmillis)/3600000000.0;
```

```
Serial.print(kWh, 4);
```

```
Serial.println("kWh");
```

```
lastmillis = millis();
```



```
Cayenne.virtualWrite(27, h, "rel_hum", "p");    // To write the Sensor reading to
Cayenne Dashboard using channel 0
```

```
Cayenne.virtualWrite(1, t, "temp", "c");    // To write the Sensor reading to Cayenne
Dashboard using channel 1
```

```
Cayenne.virtualWrite(25, emon.Vrms, "voltage", "v");
```

```
Cayenne.virtualWrite(41, emon.Irms, "current", "a");
```

```
Cayenne.virtualWrite(22, emon.apparentPower, "pow", "w");
```

```
Cayenne.virtualWrite(29, kWh, "energy", "kwh");
```

```
}
```

```
CAYENNE_IN(2)
```

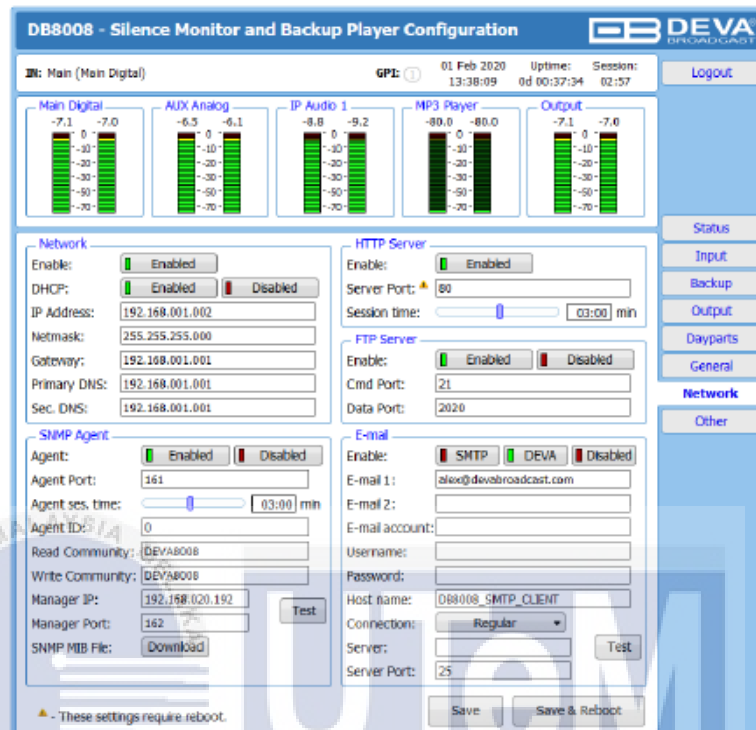
```
{
```

```
digitalWrite(2, !getValue.asInt()); // to get the value from the website
```

```
}
```

Appendix B: Deva DB8008 Manual

NETWORK



DB8008 - Silence Monitor and Backup Player Configuration

GP1: 01 Feb 2020 13:38:09 Uptime: 0d 00:37:34 Sessions: 02:57 Logout

Main Digital: -7.1 -7.0
AUX Analog: -6.5 -6.1
IP Audio 1: -8.8 -9.2
MP3 Player: -80.0 -80.0
Output: -7.1 -7.0

Network
 Enable: ☒ Enabled
 DHCP: ☒ Enabled ☐ Disabled
 IP Address: 192.168.001.002
 Netmask: 255.255.255.000
 Gateway: 192.168.001.001
 Primary DNS: 192.168.001.001
 Sec. DNS: 192.168.001.001

HTTP Server
 Enable: ☒ Enabled
 Server Port: 80
 Session time: 03:00 min

FTP Server
 Enable: ☒ Enabled ☐ Disabled
 Cmd Port: 21
 Data Port: 2020

SHMP Agent
 Agent: ☒ Enabled ☐ Disabled
 Agent Port: 161
 Agent ses. time: 03:00 min
 Agent ID: 0
 Read Community: DEVA8008
 Write Community: DEVA8008
 Manager IP: 192.168.020.192
 Manager Port: 162
 SNMP MIB File: Download

E-mail
 Enable: ☒ SMTP ☒ DEVA ☐ Disabled
 E-mail 1: alex@devabroadcast.com
 E-mail 2:
 E-mail account:
 Username:
 Password:
 Host name: DB8008_SMTP_CLIENT
 Connections: Regular
 Server:
 Server Port: 25

▲ - These settings require reboot.

Save Save & Reboot

Network

The network addresses could be set manually (static IP) or automatically via a DHCP Server. To set static IP, Netmask, Gateway and DNS addresses, the DHCP should be disabled. In order for the built-in DHCP client to be activated, the function should be enabled. When the DHCP client is activated, all assigned values will be shown in the relevant fields on the "Status Screen". If due to any reason, the DHCP procedure cannot be completed, DB8008 will use Auto IP and will generate an IP Address.

E-mail

Enter the desired alarm recipients in E-mail 1 and/or E-mail 2 fields. Fill in your e-mail account settings: Sender, Username and Password, Server, Server Port and Connection Type. If you experience difficulties in the set-up, or would like to use DEVA account for sending of alarm email notifications, press the [DEVA] button option, and complete the recipient emails (E-mail 1 and E-mail 2) only. The other fields must be left blank, otherwise the email notification option will not be working. Even though using the DEVA account eases the set-up process, we recommend user account to be used for sending of email notifications, and the DEVA account for test purposes. When using DEVA account, please note that the stable 24/7 connection depends on the mail service provider and cannot be guaranteed. We recommend you to use the [Test] button and generate a test e-mail, which upon success will be delivered to the specified E-mail 1 and/or E-mail 2. Example of Test E-mail Message:



65 Aleksandar Stamboliyski Str., 8000 Bourgas, Bulgaria
 Tel: +359 56 820027, Fax: +359 56 836700
 E-mail: office@devabroadcast.com, Web: www.devabroadcast.com

DB8009 Test Message.
 Please do not reply to this e-mail.

HTTP Server

Enable/Disable the HTTP Server. Specify the Server Port and Session Timeout.

FTP Server

Enable/Disable the FTP Server. Specify the Command and Data Ports to be used. For information on how the connection between the DB8009 and an FTP Client should be configured, please refer to ["Download/Upload files via FTP" on page 52](#).

SNMP Agent

Specify Agent ID, Agent Port, Read/Write Communities, Manager IP, Manager Port and Agent Session Time.

Agent – enables/disables SNMP Agent.

Agent ID – is used for identification of the device among others, when an SNMP notification is being sent. Once all needed settings are applied, use the Test button to generate a test notification, which upon success will be received by the SNMP Manager. Press the [Download] button to download the latest available DB8008 SNMP MIB file.

NOTE: The MIB file may vary from one firmware revision to another. Downloading this file from the device, guarantees that you have the proper MIB file.

WHEN APPLYING NEW SETTINGS – In order new settings to take effect, it is necessary to press the [Save] button. Please keep in mind that some of the new settings can reset DB8008.



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

APPENDIX B

DOWNLOAD/UPLOAD FILES VIA FTP

In order for a connection to be established the following setting should be applied:

1. FTP Server Settings

The built-in FTP Server has four important parameters that should be configured: Command Port, Data Port, User name and Password. These parameters are to be used in the FTP client's connection configuration. Further information on how to change the FTP Server's settings and their respective default values can be found in the device's User manual.

WE RECOMMEND the usage of FileZilla (<https://filezilla-project.org>). This is a widespread open source software distributed free of charge, hence available for downloading from the Internet.

NOTE: The FTP Server can manage only one connection at a time. The FTP Server works in Passive mode. Hence, the FTP Client should also be set in passive mode.

2. IP Router and Port Translation Settings

If the connection to the device is made through a Network address translation (NAT) router or firewall, the port forwarding feature of the router should be configured. The port forwarding is usually set in the firewall section of the router's menu. As each router has different port forwarding procedure, we recommend you to refer to its complete manual. To allow proper data flow through the router, the FTP Command and FTP Data ports should be open.

NOTE: The FTP port numbers to be used in the port forwarding feature configuration can be found in the device.

اوتنور سیتی تکنیکل ملیسیا ملاک

UNIVERSITI TEKNIKAL MALAYSIA MELAKA