Faculty of Electrical and Electronic Engineering Technology

DEVELOPMENT OF IOT ENERGY METER WITH CURRENT, VOLTAGE AND COST MONITERING SYSTEM

MUHAMMAD JAMALUDIN BIN NORAZMI

A project submitted
in partial fulfillment of the requirement for the degree of
Bachelor of Electrical Engineering Technology with Honours

2023

DEVELOPMENT OF IOT ENERGY METER WITH CURRENT, VOLTAGE AND COST MONITERING SYSTEM

MUHAMMAD JAMALUDIN BIN NORAZMI

A project report submitted
in partial fulfillment of the requirements for the degree of
Bachelor of Electrical Engineering Technology with Honours

Faculty of Electrical and Electronic Engineering Technology

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

DECLARATION

I declare that this project report entitled “Development of IOT Energy Meter with Current, Voltage and Cost Monitoring System” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : 

Student Name : MUHAMMAD JAMALUDIN BIN NORAZMI

Date : 27/1/2023

APPROVAL

I approve that this Bachelor Degree Project 1 (PSM1) report entitled “Project Title” is sufficient for submission.

Signature : 

Supervisor Name : TS DR AHMAD ZUBIR BIN JAMIL

Date : 27/1/2023

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Electrical Engineering Technology with Honours.

Signature	:	
Supervisor Name	:	TS DR AHMAD ZUBIR BIN JAMIL
Date	:	27/1/2023
Signature	:	-
Co-Supervisor Name (if any)	:	-
Date	:	-

DEDICATION

To my beloved mother, Irma Niswati bin Amri, and father, Norazmi bin Abu Sha'ari,

and

To dearest wife, Saidatul Aishah and

My Super visor, TS DR Ahmad Zubir bin

jamil.

I approve that this Bachelor Degree Project 2 (PSM2) report entitled “Development of IOT Energy Meter with Current, Voltage and Cost Monitoring System” is sufficient for submission.

Signature :

Supervisor Name : **TS DR AHMAD ZUBIR BIN JAMIL**

Date : 27/1/2023

ABSTRACT

In this project, we will discuss the Energy Meter, also known as a Smart Meter with Cost Monitoring System, which is a gadget that allows us to monitor the power consumption of portable appliances and is a step toward domestic energy conservation strategies . Using voltage and current sensors, as well as ESP-32 microprocessor, this paper suggests a customisable power meter design. This meter is used to keep track of voltage, current, and cost monitoring system in real time. A reference power meter is used to calibrate the voltage and current. The microcontroller will be able to calculate projected electrical consumption and cost based on detailed knowledge about each device's consumption. Similarly, our equipment will assist in identifying whether one of them is malfunctioning. This project will be a hardware-software co-design process in this prototype.

ABSTRAK

Dalam projek ini, kami akan membincangkan tentang Meter Tenaga, juga dikenali sebagai Meter Pintar dengan Sistem Pemantauan Kos, yang merupakan alat yang membolehkan untuk memantau penggunaan kuasa peralatan mudah alih dan merupakan langkah ke arah strategi penjimatan tenaga domestik . Menggunakan penderia voltan dan arus, serta mikropemproses ESP-32, kertas ini mencadangkan reka bentuk meter kuasa yang boleh disesuaikan. Meter ini digunakan untuk menjejaki voltan, arus dan sistem pemantauan kos dalam masa nyata. Meter kuasa rujukan digunakan untuk menentukur voltan dan arus. Mikropengawal akan dapat mengira unjuran penggunaan elektrik dan kos berdasarkan pengetahuan terperinci tentang penggunaan setiap peranti. Begitu juga, peralatan kami akan membantu dalam mengenal pasti sama ada salah satu daripadanya tidak berfungsi. Projek ini akan menjadi proses reka bentuk bersama perisian perkakasan dalam prototaip ini.

ACKNOWLEDGEMENT

I would like to express our gratitude to Allah, the Most Gracious, the Most Merciful, for providing us with vigour and good health throughout our final semester at the Universiti Teknologi Malaysia Melaka campus. Thank you to God for providing us with a healthy physical and mental state that enabled us to finish this Final Year Project. A good senior project is not the result of a single person's efforts. I'd want to offer our deep gratitude to my supervisor, TS. DR. Ahmad Zubir Bin Jamil, for all the support, guidance, criticism, and friendship we've established together over the last year. We would not have been able to achieve this feat without the love and support of our family. The money donated to us by our relatives to help us brought us all together. Our family's contribution to our support provided all of the materials needed to complete the final year project. In a nutshell, thank you to everyone of our friends who have offered to assist us. The colleagues and lecturers who have helped us in a variety of ways to meet the project's requirements. Finally, we want to express our gratitude for having all of you here with us, and we sincerely appreciate it.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

-
-
-
-
-
-
-
-

LIST OF ABBREVIATIONS

V	-	Voltage
I	-	Ampere
P	-	Watt
	-	
	-	
	-	
	-	
	-	

LIST OF APPENDICES

APPENDIX **TITLE** **PAGE**

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

. The discussion in this chapter began with an overview of the country's power meter logger study, followed by explanations of the problem statements. This chapter will also include the major objectives of the current investigation, the scope of the study, the significance of the study, and the thesis outline.

1.2 Background of Study

The meter is used to keep track of the number of units consumed, the expected cost, the Line Voltage, and the amount of current consumed. IoT Blynk is a simple web application that displays the Live Output of various IoT readings. This allows users to monitor the number of units consumed, the expected cost, the Line Voltage, and the current utilized in real time from anywhere on the site. In this way the energy meter observing framework enables client to adequately screen power meter readings and check the charging on the

Blynk app effortlessly. As a result, the energy meter monitoring system enables clients to easily monitor power meter readings and check charges on applications in the phone.

1.3 Problem Statement

Nowadays, all machines have been innovated to ease our daily activities whether at home or at work, but all these machines use electricity whether a direct current machine or a portable that have to charge after using.

- People cannot maintain the monthly budget for electricity because of over usage.
- People cannot determine which electrical equipment that use low or high power usage to minimize the used for maintaining low electricity bills.

1.4 Research Objectives

The main aim of this project is to built a energy meter system that can be read current, voltage and cost of the circuit via meter or in application on the phone. Specifically, the objectives are as follows:

a) To read current from all loads in a circuit by using sensor.(Install at MCB)

b) To access the reading via meter or in- application at phone in realtime by using blynk applincation

c) To culcalate the cost of the circuit but using ESP32 microcontroller.

1.5 Scope of Study

. To avoid any uncertainty of this project due to some limitations and constraints, the scope of the project are defined as follows:

- Built for Comersial usage.
- Device have to be installed at Electrical Distribution Box.
- Internet
- Application installation in phone.

Table 1.1 Main Component Of The Project

Hardware	Software:	Method:
Arduino ATMEGA-328 - microcontroller Esp32 - Wifi Module Current sensor ACS712- measure current. AC Voltage sensor (ZMPT101B) - measure voltage.	Blynk – to develope the and transfer data from device to applications via phone. Arduino Uno – to set the microcontroller command and intructions.	IOT - system of interrelated computing devices, mechanical and digital machines that have the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

d)

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, to study and identify the variables that can be made to select component, data and cost which will aid in the development of this project. This chapter focus on examine selected journal and research articles that connected to this project.

2.2 Overview

This project is to built the energy meter that can calculate the montly bills and also show the current reading. Basicly user can access this readings via application on the phone that will give the live price and live reading.

2.3 Smart meter

Since the early 2000s, smart meters have been installed in several nations throughout the world. The smart meter, as a crucial component of the intelligent grid, is projected to deliver economic, social, and environmental advantages to a variety of stakeholders. Smart meter principles have been extensively discussed. One of the primary criteria evaluating the success of smart meters is smart meter data evaluation, which deals with data collection, delivery, processing, and analysis that benefits all stakeholders. As home power usage continues to rise, consumers are becoming more cognizant of energy use and efficiency from both an economic and environmental standpoint.

The Smart meter is a meter that is used to measure the amount of energy used by an electric load. The entire power used and utilized by the load at a certain time interval is

referred to as energy. It is used to measure power usage in both home and industrial AC circuits. The meter is less costly and more accurate.

2.3.1 Energy meter reading based on GSM

The GSM communications network is used to provide data about electricity consumption to the utility administration and, if necessary, to the customer. The voltage and current sensors take RMS voltage and current readings and pass them to the microcontroller, which performs active and reactive power calculations[1]. The reading from the utility administration SMS is received by the smart energy meter's programmable interface, and the metre takes action based on the information provided[2].

When an energy provider needs information to calculate a bill, they send a communication to AMR. The microcontroller unit receives a message and reads it, as well as reading the user's mobile number, verifying authentication and sending data to the verified number. GSM-based AMR delivers an SMS alert to the energy provider if the system access mobile number is not verified[1]. It also allows customers with substantial outstanding dues to have their electricity off by transmitting a code to the energy meter. If this code matches, the meter's power will be disconnected. It also has the ability to re-connect electricity owing to a deposit of the prior bill amount owed by sending a code to the energy meter[2].

2.4 Internet Of things (IoT)

The Internet of Things (IoT) is the next generation of communication. The Internet of Things (IoT) can be used to create, receive, and exchange data for physical things in a seamless manner. IoT apps are designed to automate various operations and allow

inanimate physical things to behave without the need for human intervention[3]. The Internet of Things (IoT) is a network of intelligent sensors that can track and manage objects remotely over the Internet. This intelligent technology can be utilized to increase current farming production and quality, as well as provide an alarm system and detect heart rate, among other things[4]. As a result, the goal of this study is to provide an intelligent Internet of Things planning application. With Cisco Inc. predicting 50 billion connected devices by 2020, the Internet of Things (IoT) has emerged as an area of enormous influence, potential, and growth as intelligent homes, intelligent cities, and everything smart[3].

2.4.1 Security of data

The fundamental goal of IoT Security is to secure user privacy, infrastructure, data, and IoT devices, as well as to assure the availability of IoT ecosystem services[5]. IoT security research has recently gained a lot of traction thanks to the utilization of available simulation tools, models, and computational and analysis platforms[6]. Users can expect convenience, productivity, and automation from present and new IoT devices. This environment's ever-increasing implementation necessitates high levels of security, anonymity, authentication, and attack recovery. It must make the necessary enhancements to the design of IoT applications to establish end-to-end secure IoT environments[6].

2.4.2 Arduino ATMEGA

Using the MCU Node EsP8266 microscope, the intelligent capsule sent data to the Blynk server over a Wi-Fi network. Arduino IDE was used to programme the microcontroller in

C++[7]. The Blynk Mobile App was used to track and view real-time data on the digital dashboard. When the smart capsule lost contact with the Blynk server, a notice was sent to the appropriate individuals immediately. The study's findings indicated the efficacy and use of the smart capsules developed and the Blynk application in smart planning.

2.5 Arduino system

The Arduino ATMEGA-328 has been adapted for a variety of purposes. The power jack cable is used to program the Arduino microcontroller, which allows the device to run. On the market, there are a variety of Arduino boards to choose from. This article goes through the Arduino UNO ATMEGA-328 microcontrollers in great detail[7]. Arduino is a computer program that allows you to change and upload your program utilizing apps. The Arduino software mostly supports the C and C++ programming languages. The Arduino board has a range of inputs and outputs, and 8 input and output ports can be used for different applications at the same time. Rotating general motors, stepper engines, open valve control, and other applications employ Arduino boards.

2.6 Current and Voltage sensor

A voltage sensor is a device that measures and calculates the amount of voltage in an object. Voltage sensors can determine whether the voltage is AC or DC. The voltage is the sensor's input, while the switches, analogue voltage signal, current signal, or audible signal are the sensor's output. A current sensor detects electric current in a wire and creates a signal proportional to it. An analogue voltage or current, or a digital output, could be generated. The generated signal can then be used to display the measured current in an

ammeter, or it can be saved in a data acquisition system for further analysis, or it can be utilized for control.

2.7 Theoretical Framework

A theoretical framework has been established by combining all of the aspects and factors mentioned above. The framework will serve as the foundation for inquiries and data gathering in constructing a high-quality smart energy meter with cost monitoring system in later chapters of the study.

CHAPTER 3
METHODOLOGY

3.1 Introduction

In this chapter will be discuss about the method that will be used for building this project. Firstly, the sensor will sense voltage and current that will be send to ESP32 that act as microcontroller, then it will calculate the cost value. Next the ESP-32 transmit the data to Blynk application on the phone via Wi-Fi.

3.2 Project Flow

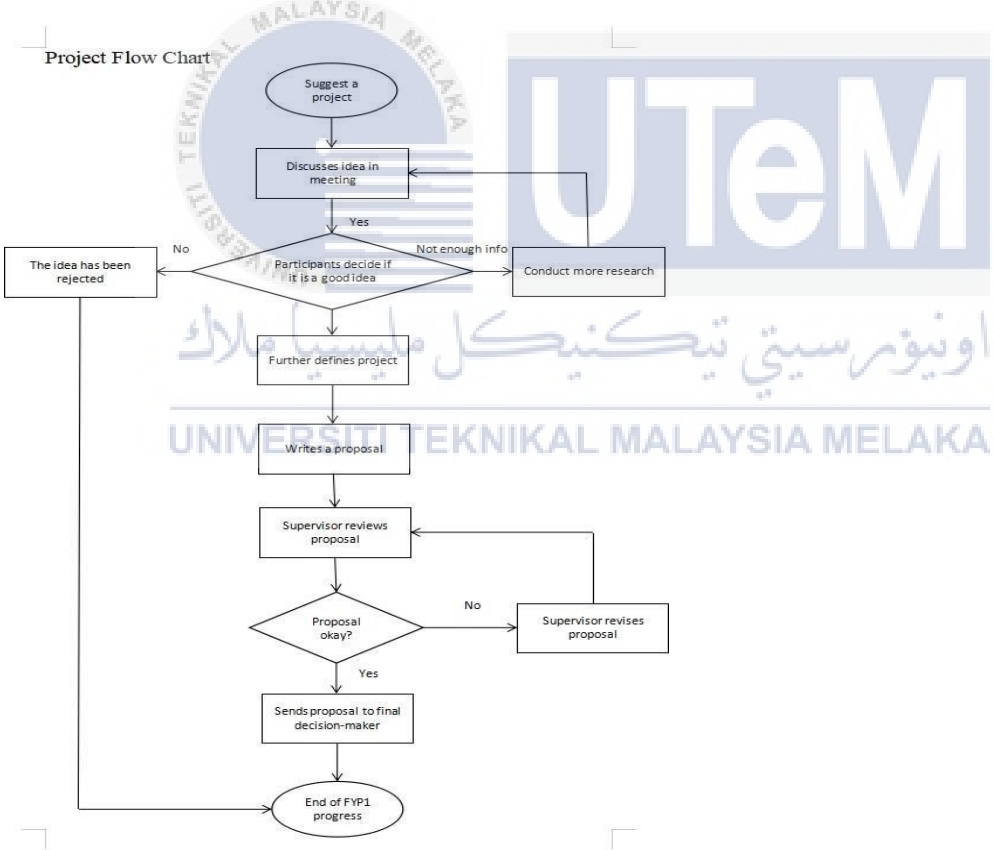

Figure 3.1 Project Flow Chart

3.3 Project development

Firstly, the sensor will sense voltage and current that will be read for to ESP32 that act as microcontroller, then it will calculate the cost value. Next the ESP32 will transmit the data to the Blynk application on the phone via Wi-Fi. This is is roughly concept of this project

3.3.1 Hardware

ESP-32

ESP32 is a series of low-cost, low-power system on a chip (SoC) microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series was released in 2016 and is a successor to the ESP8266 series of microcontrollers. ESP32 microcontrollers are used in a variety of devices, including standalone microcontroller systems, Wi-Fi and Bluetooth-enabled devices, and Internet of Things (IoT) applications. Some features of the ESP32 include:

- Dual-core processor with two processor cores that can be individually controlled
- On-chip Wi-Fi and dual-mode Bluetooth
- On-board cryptographic hardware for secure communication
- A large number of input/output (I/O) pins for connecting to various sensors and peripherals
- Support for multiple low-power modes for power-sensitive applications
- Support for Over-The-Air (OTA) updates

The ESP32 is a popular choice for building IoT applications, due to its low cost, wide availability, and support for a range of programming languages and development environments.

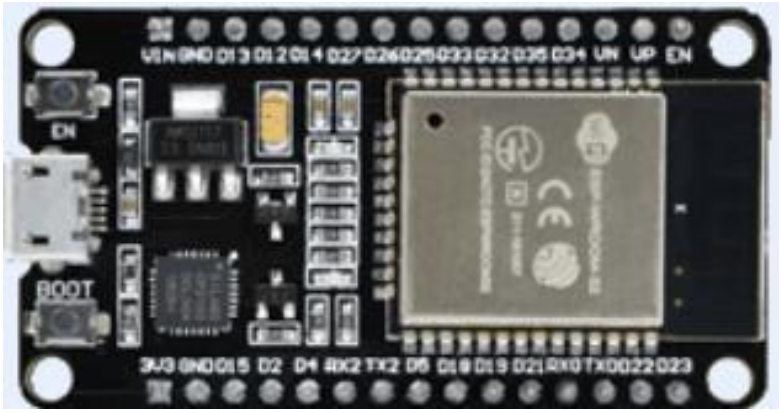

Table 3.1 Example of ESP-32

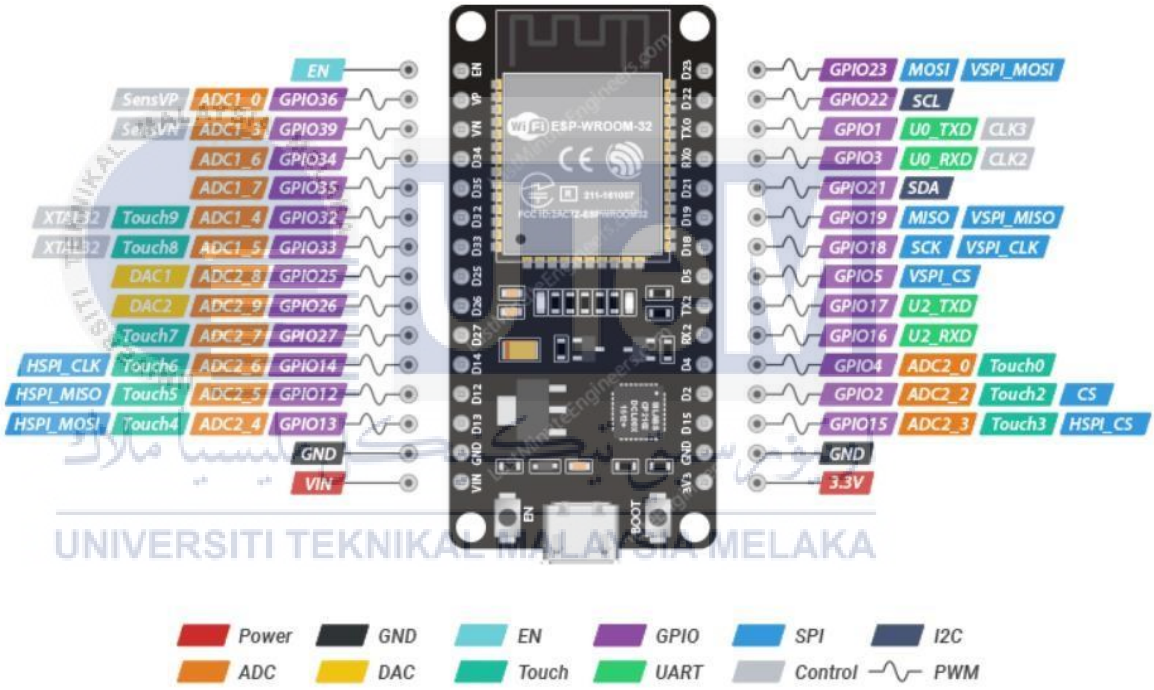

Figure 3.2 ESP-32 PIN port

ZMPT101B AC Voltage Sensor

The ZMPT101B is a voltage sensor that is used to measure the AC voltage of a power line. It is a passive device, which means it does not require a power source to operate. Instead, it relies on the voltage of the power line to generate a small current through its

internal transformer, which is then used to produce a voltage output that is proportional to the AC voltage being measured.

The ZMPT101B has a voltage range of 45-65V, making it suitable for measuring the AC voltage of standard household power lines. It has a high accuracy of ±2%, and is able to measure both the RMS and peak values of the AC voltage. The sensor also has a built-in voltage divider, which allows it to output a smaller voltage that is more suitable for use with microcontrollers and other electronic devices.

The ZMPT101B is often used in applications such as energy monitoring, power factor measurement, and voltage regulation, where accurate measurement of AC voltage is required. It is a compact and easy-to-use sensor that can be easily integrated into a variety of systems.

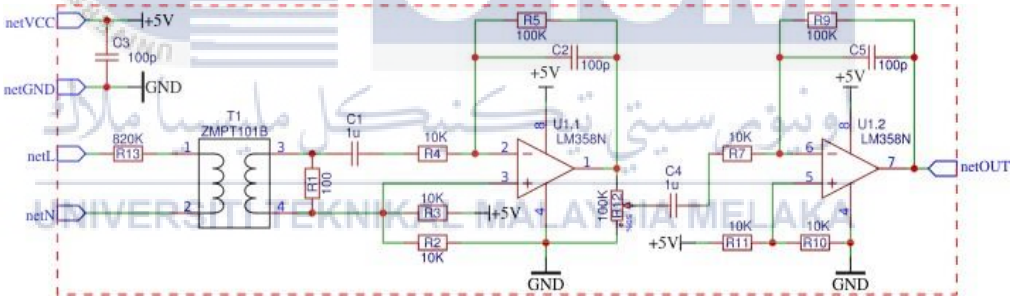

Figure 3.3 Voltage sensor circuit

ACS712 Current Sensor

The ACS712 is a current sensor that is used to measure AC or DC currents. It is a Hall effect-based sensor that utilizes the magnetic field generated by the current flowing through a conductor to produce a voltage output that is proportional to the current being

measured. The ACS712 is available in several different versions, each with a different current range and sensitivity.

Some features of the ACS712 include:

- Wide current range: The ACS712 is available in versions that can measure currents ranging from 5A to 30A.
- High accuracy: The sensor has a typical accuracy of ±1% over a wide temperature range.
- Low offset voltage: The ACS712 has a low offset voltage, which means that it can accurately measure small currents.
- Wide operating temperature range: The sensor can operate over a wide temperature range of -40°C to 150°C.

The ACS712 is often used in applications such as motor control, power supply design, and energy monitoring, where accurate measurement of current is required. It is a compact and easy-to-use sensor that can be easily integrated into a variety of systems.

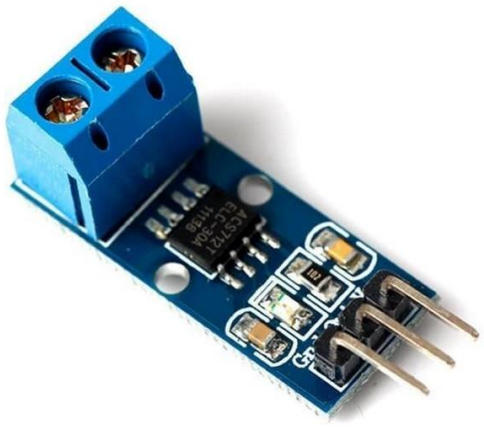

Figure 3.4 ACS712 Current sensor

Doublecore 1.5MM Copper Wire

1.5mm doublecore copper wire is a type of electrical wire that consists of two conductors made of copper that are twisted together in a single cable. It is covered with a protective insulation to prevent short circuits and protect against physical damage. The wire is often used in applications where two separate conductors are required, such as for the electrical wiring of buildings or for the connection of electrical appliances.

Doublecore wire is generally easier to install than two separate wires, as it requires fewer connections and is more flexible. It is also more resistant to interference from external sources, such as electromagnetic fields, as the two conductors are twisted together, which helps to cancel out any interference.

1.5mm doublecore copper wire is often used in applications where a medium-duty wire is required, such as for the electrical wiring of small appliances or for the connection of lighting systems. It is available in a range of colors, including red, black, and white, which can be used to identify the purpose of each conductor. Copper is a good choice for electrical wire due to its high conductivity and low resistance, which allows it to carry large currents with minimal voltage drop.

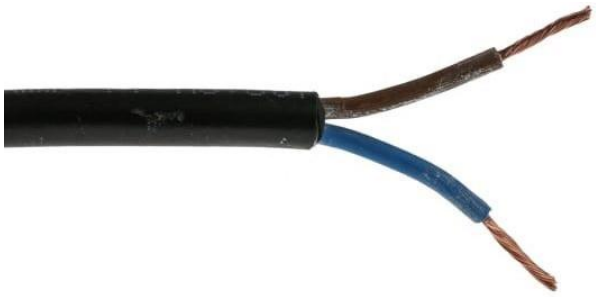

Figure 3.5 1.5mm Doublecore copper Wire

PCB Board

A printed circuit board (PCB) is a type of electronic circuit board that is used to mechanically support and electrically connect electronic components using conductive tracks, pads, and other features etched from copper sheets laminated onto a non-conductive substrate. PCBs are used in a wide variety of electronic devices, including computers, smartphones, and appliances, to provide a support structure for the components and to facilitate the flow of electricity between them.

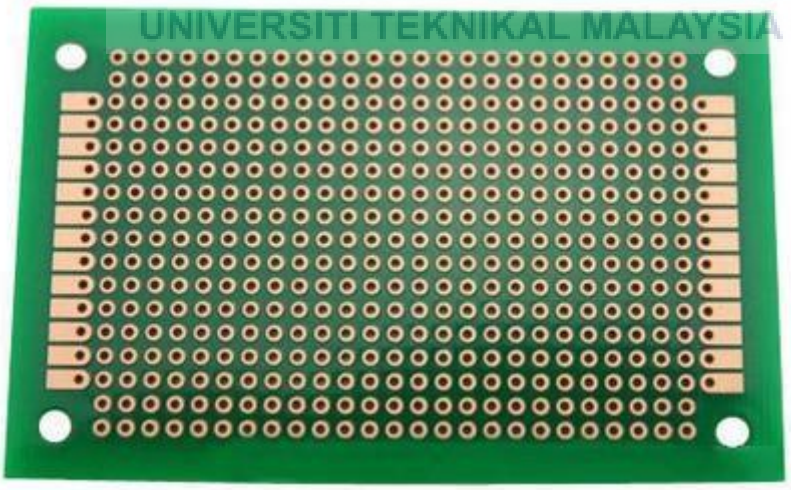

Figure 3.6 PCB Board

3.3.2 Software Developement

Arduino IDE

Arduino Integrated Development Environment (IDE) is a software program that is used to write and upload computer code to an Arduino board. Arduino is an open-source platform for building electronics projects, and the Arduino IDE is a cross-platform application that runs on Windows, Mac, and Linux operating systems. It is designed to be easy to use and user-friendly, with a simple interface that allows users to write and upload code to their Arduino board quickly and easily.

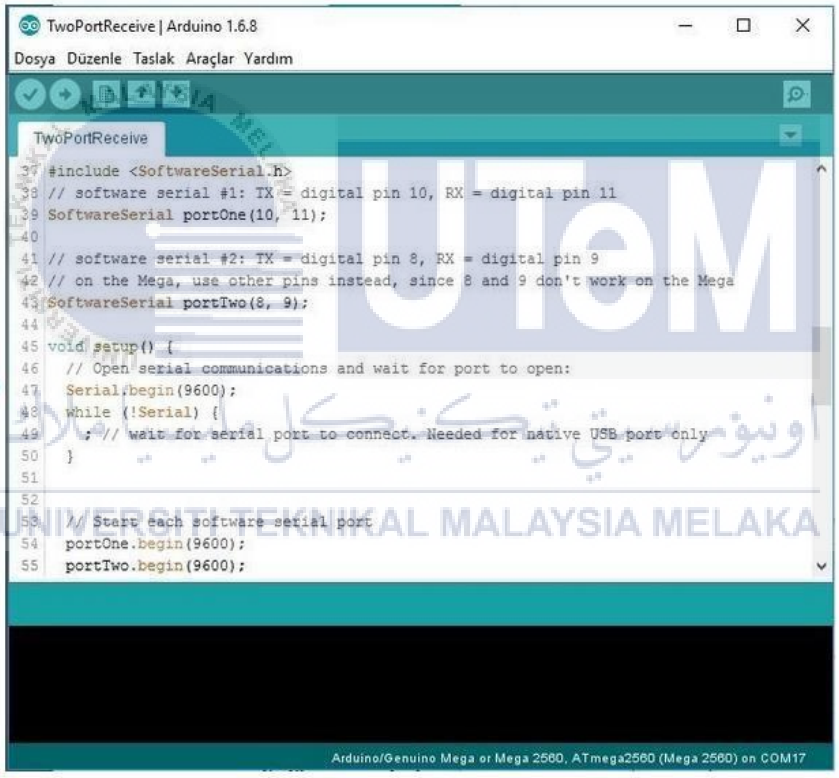

Figure 3.7 Arduino IDE interface.

BLYNK Application

Blynk is an iOS and Android platform for controlling Arduino, Raspberry Pi, and other Internet-connected devices. It's a digital dashboard where user may drag and

drop widgets to create a graphic interface for project. This is the application that will be use to set up a receiver and display in the phone.

Figure 3.8 Blynk Application interface in phone.

3.3.3 Overall Process.

The important parameters for smart energy meter with logger system is the processing unit and step counts for the data. Therefore, the node MCU ESP-8266 has been chose because it has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application. The methods and techniques used in this

entire project will be explained briefly in detail with the assistance of figures and flow chart.

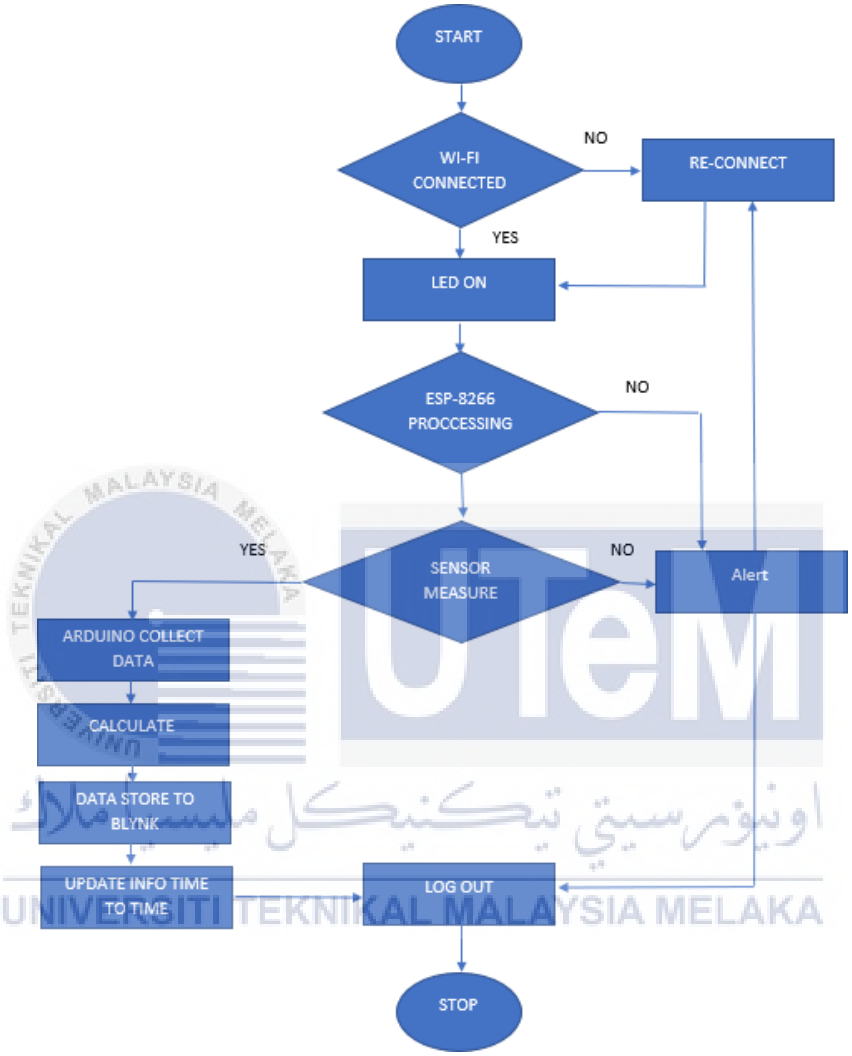

Figure 3.9 System Flow chart

This system flowchart is a diagram for a 'Smart Energy Meter' operation. The system processor keeps the data significantly which has been set by the Arduino IDE coding.

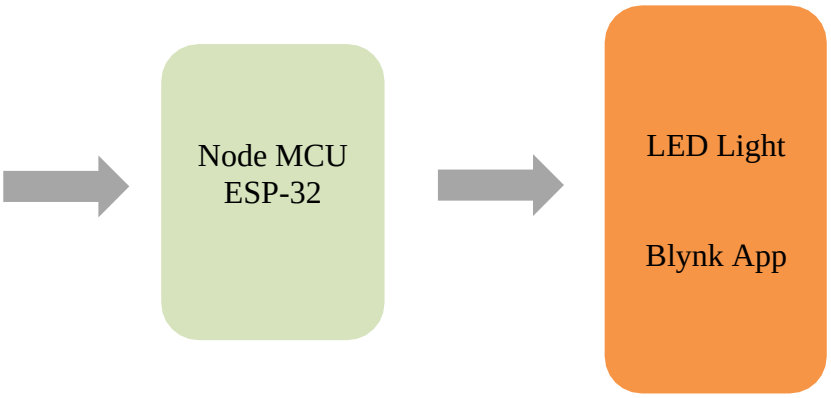

Figure 3.10 Block Diagram of the project

Figure shows the simple block diagram explaining the plan that have been set up for the project.

The entire process begins with the power supply being connected to the smart energy metre. When the metre begins to run, the ESP-32 acts as the main processor, connecting to the user's device over wifi before uploading all of the programmable code from the it. The current/voltage sensor was then linked to the processor. It will be used to measure data from the circuit that will be connect to the sensor. Finally, all of the results have been presented on the app;ications on the phone. The phone will display the overalls Power used , cost and then reading phone each MCB SSO circuit.

3.4 Time Horizon

The temporal span for this study would be cross-sectional. The goal of the study, according to Levin (2006), is to determine the prevalence of the desired outcome in the population or subgroups within the population at a certain time period.

Table 3.2 Project Planing

PROJECT PLANNING

(List the main activities of the project. Indicate the length of time needed for each activity.)

	2022																				2023																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Project Title																																						
Problem Definition																																						
Identify project objective																																						
Procedure preparation																																						
Acceptance project proposal																																						
Participant selection																																						
Material preparation																																						
Experiment setup																																						
Smart meter cost monitoring system production																																						
Experiment																																						
Data Analysis																																						
Discussion and Conclusion																																						

CHAPTER 4
RESULT AND DISCUSSION

4.1 Introduction

This chapter records and discusses the results obtained through the development of the project entitled 'Smart energy meter with cost monitaring system'. Several simulations have start to test the connection from all of the hardware and software. In the first stage of project planning, the preliminary result will be obtained mostly based on theories and research. However, is still work in progress so several testing have to made so that any counter measure or improvement can be made.

4.2 Project Prototype

With the research done in Chapter 2 and the calculations alongside simulations done in Chapter 3, the prototype of the project is built with all the chosen equipment and tools. The chosen hardware including ESP-32 , Current sensor, Voltage sensor, Wire Connecter, etc. is assembled carefully according to the circuit simulation done in Chapter 3.

Besides, the design and coding for the software are also done through all the tools and programs mentioned in Chapter 3. The software design is tested and modified multiple times to match the requirement of the system. The design and set up for the project prototype are recorded as follow.

4.2.1 Hardware installation

The prototype of the project is done by using a extension to replicate load. The prototype have already been install to 3-pin plg top then connect to power supply. The output of the prototype is connected to and the extension and then to load. So this allow the current past

through the prototype and sensor can read the data. The data then transmitted by ESP-32 to blynk Application by Wifi. Figure 4.1 shows the drawing of the hardware installing plan while Figure 4.3, and Figure 4.4 show the outside view, and inside view of the project prototype respectively.

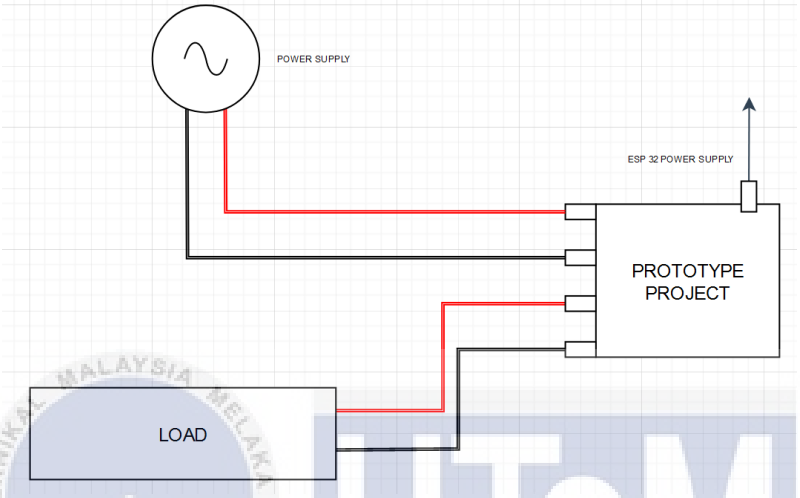

Figure 4.1 Connection of the prototype

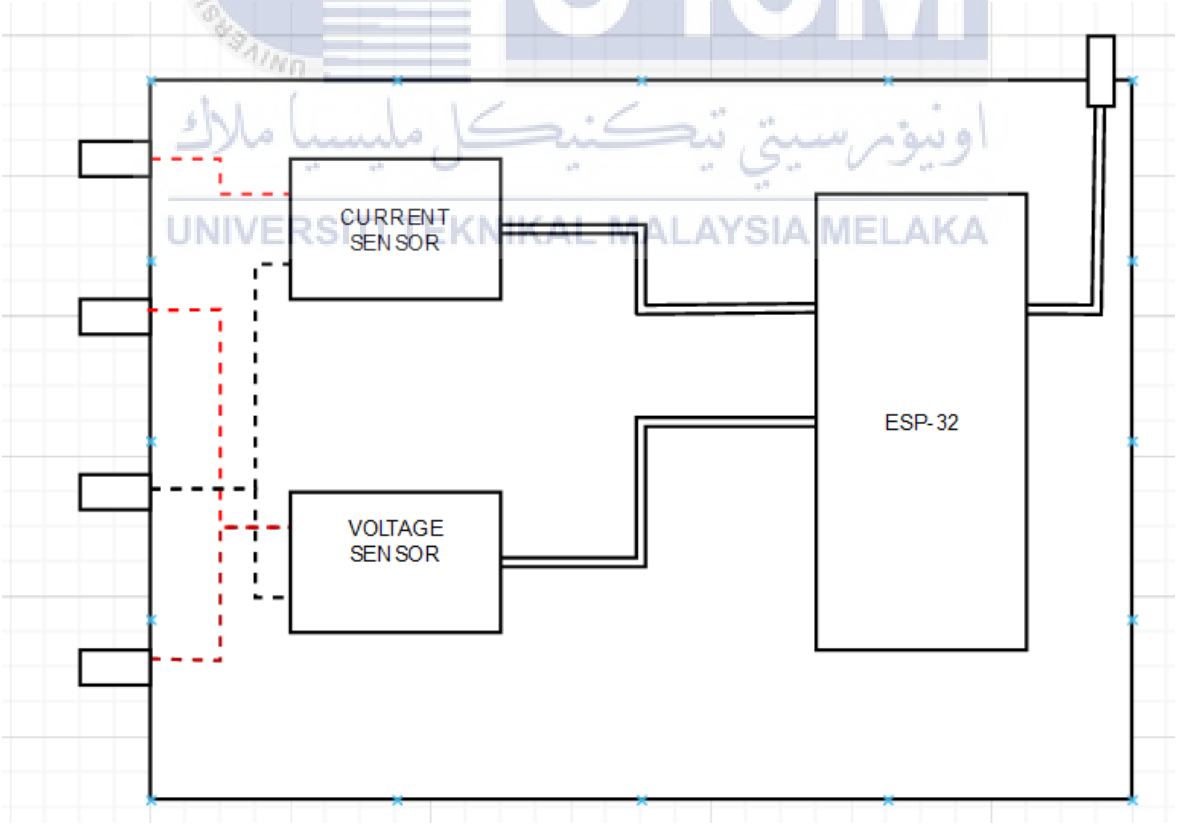

Figure 4.2 the board connection from sensors to ESP-32 microcontroller.

Figure 4.3 the prototype of this project with extansion that will be connect to load, plugtop for power supply and usb cable for ESP-32 power supply.

Figure 4.4 The Connection of the prototype component.

4.2.2 Development of software

4.2.2.1 Coding setup for microcontroller

In the development of the system, the coding for commanding the function of the system is done and compile to ensure no error by using Arduino IDE and uploaded to the ESP-32 Microcontroller. Figure 4.5 shows the coding of the system done by using Arduino IDE

```
#define BLYNK_TEMPLATE_ID           "TMPLIOlNfd61"
#define BLYNK_DEVICE_NAME           "Quickstart Device"
#define BLYNK_AUTH_TOKEN            "NXqZHOAM_sNDEpnNZMvZDTukxaQDAs_e"


// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial


#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <Filters.h>

char auth[] = BLYNK_AUTH_TOKEN;
```

Figure 4.5 The library and IP of the Blynk application so set the path destination.

Firstly, figure 4.5 below show Blynk templete ID, device name and token that be taken from the blynk application to match the direction of the data that will be transmitted.

```
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Keluarga Bahagia";
char pass[] = "zmalqp10";
```

Figure 4.6 Setting WIFI id and password to connect.

Next is the Wifi name and password sa that the ESP-32 can connect to it. This is crutial because it use wifi to send the data.

```
int ZMPT101B1 = 35;
int ZMPT101B2 = 34;

float power1, power2, cost1, cost2, power3, power4;
float v1,v2,a1,a2;
float testFrequency1 = 50;
float windowLength1 = 100/testFrequency1;
float testFrequency2 = 50;
float windowLength2 = 100/testFrequency2;
```

Figure 4.7 Declaration of pin at ESP 32 and declaration oh the input.

After that, declare and define the parameter that needed to calculate and send to the Blynk application. And then the calculations instructions according to the TNB rates.

4.2.2.2 Design of mobile application

For the development of mobile applications used for displaying output adn reading of this system, Blynk Application is used for creating a mobile application that can communicate with the ESP-32 through Wifi connection. Figure 4.6 shows the user interface of the mobile application created using by using Blynk application.

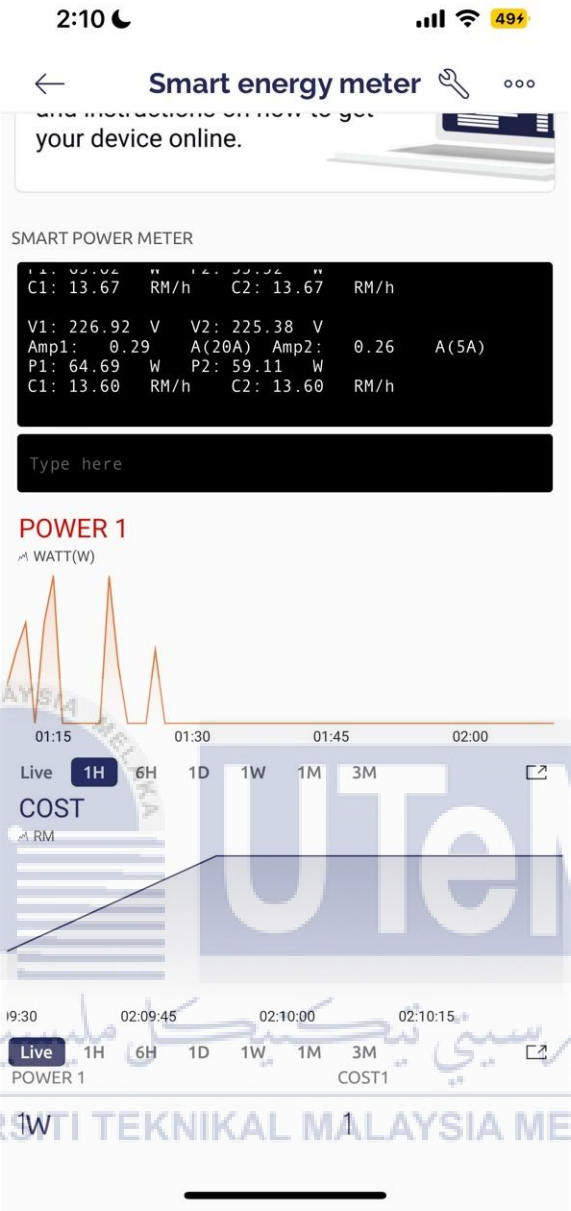

Figure 4.8 The interface of Blynk Application

In Figure above first shows terminal display that update voltage reading and current reading live reading, next its show the power that were used. After that its show the cost of the reading in real time.

4.3 Experiment Test and Protocol

Experiment for testing the performance of the system has been carried out for several place and several loads to ensure the precision of the results. Firstly, the performance of the prototype is tested for just one load which is a hairdryer. Next, increase the load to check the reading error. Next testing is connecting it with MCCB in distribution box in laboratory at Faculty which have more load. After several testing the data have been collected.

4.4 Calculation Cost monitoring system

Table 4.1 Malaysia Electrical price rates

	TARIFF CATEGORY	UNIT	CURRENT RATE
	Tariff A - Domestic Tariff		
	For the first 200 kWh (1 - 200 kWh) per month	sen/kWh	21.80
	For the next 100 kWh (201 - 300 kWh) per month	sen/kWh	33.40
1.	For the next 300 kWh (301 - 600 kWh) per month	sen/kWh	51.60
	For the next 300 kWh (601 - 900 kWh) per month	sen/kWh	54.60
	For the next kWh (901 kWh onwards) per month	sen/kWh	57.10
	The minimum monthly charge is RM3.00		

Table 4.1 shows the price for domestic in Malaysia that later be coded to Arduino to calculate from the data that had been receive from sensors. As time goes by the price will change so the price of current rating is manually add in the Blynk applications by user. The formula will be added to coding and later will be uploaded to the Arduino microcontroller. The electric power has been measured by using the equation of:

$$P = I*V$$

Where:

V is representing the voltage in Volts (V),

I is the current in milli Amperes (mA), and

P is the power in milli Watts (mW).

$$E = Pt$$

Where:

E Electrical energy (kW/h),

P is the power in Kilo Watts (kW)

t is the time of electricity consumption (h).

Let's say that the utility bill comes to the following:

1. **power consumption**: 1000 watt of electricity
2. **energy price** is 51.6 sen/kWh
3. **usage time** is 10 hour

If let say, then electric bill estimator will tell us that electric consumption is 10 kW/day, and the annual cost will be RM 157.06 per Month. These calculations will be added in the coding. So, in the applications, time of consumption and price rate will be added manually by users.

4.5 Results

This section will discuss the results generated by the IoT Smart Energy Meter. The data is collected and uploaded to clouds for analysis using the ESP-32. There had been some debate over the outcome.

4.5.1 Energy Consumption and Load Current Usage Measurements

The Iot Smart Power meter had been installed and the system ran for 1 hour in my house. The power consumption is obtained and recorded automatically by the system every second. For load, a hair dryer, laptop, and a table fan were used in this project. Table 4.2 shows the type of loads, the quantity, the power rating, the duration of use per day, and the total energy consumed per day. Figure 4.20 shows the pattern of power consumption of my house using Blynk app.

Table 4.2 State the calculations for the loads theoriticaly

LOAD	QUANTITY	POWER RATE(kW)	USAGE	KW/ hour
HAIR DRYER	1	1.4	0.1 hour	0.14
TABLE FAN	1	0.039	0.5 hour	0.0195
LAPTOP	1	0.065	0.3 hour	0.0195
TOTAL KW/H				0.179

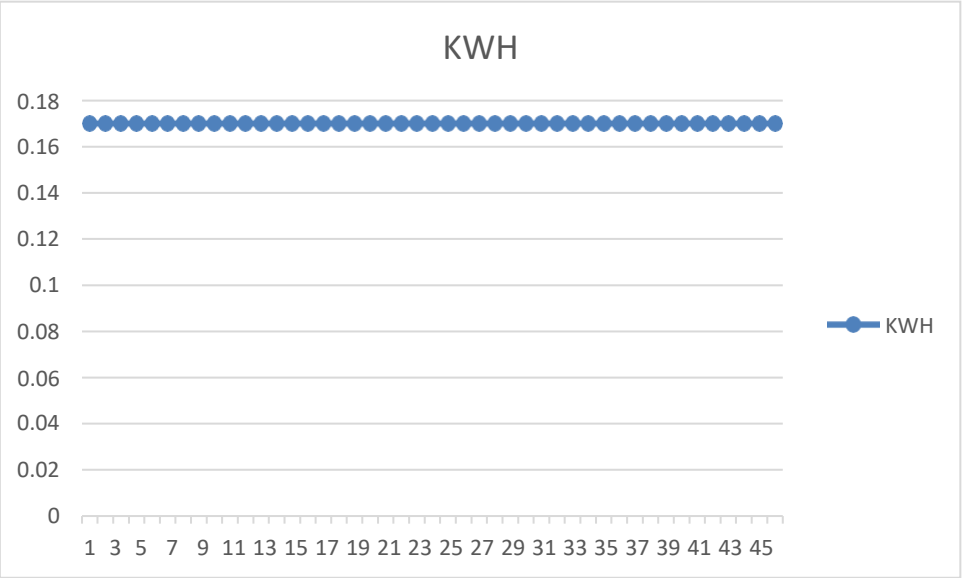

Figure 4.9 The reading of testing that is conducted.

By comparing the data from the graph in figure 4.10, it is proof that the calculated measurement in table 4.1 and the reading from the measured graph is same. Figure 4.21 depicts the complete data for current consumption using Blynk app.

The reading of the current data will not be constant because different types of loads are used in this project, so the reading will increase and decrease.

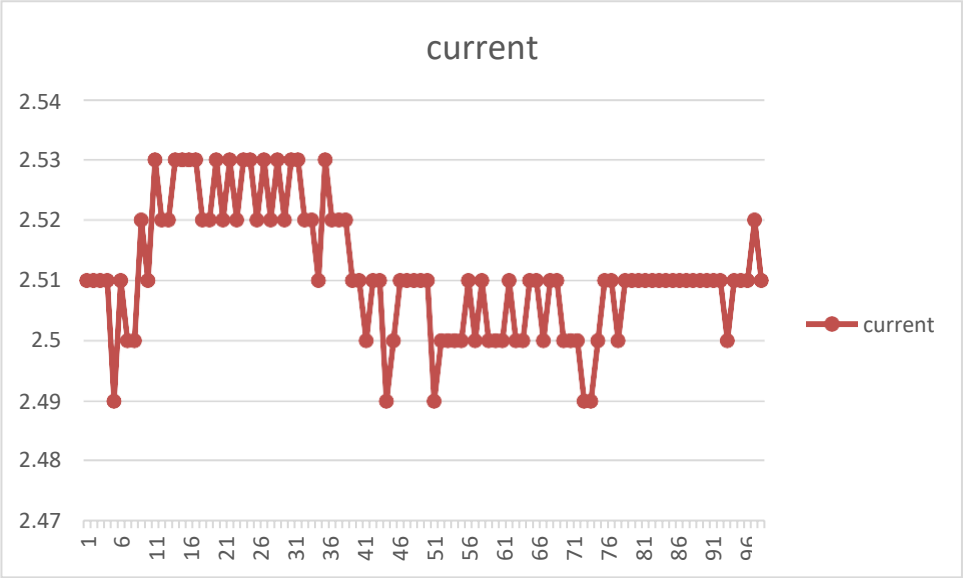

Figure 4.10 The graph of current consumption

4.6 Summary

Essentially, Smart Energy meter with cost monitoring system has a straightforward procedure and system set up. A microcontroller will measure the voltage and current entering the system to calculate the consumed power. The bill will then be calculated using these measurements and the current electricity price. Meanwhile, the user will be kept up to date on the current price and bill. So, this is how our hardware circuit will work. Our first major component will be the measurement system, and it will then transfer the recorded data to the microcontroller, which will calculate the required bill based on the most recent tariff.

CHAPTER 5

CONCLUSION

5.1 Overview

This chapter provides an overall summary of all the topic that have been done in the previous chapter. The future work also has been included. The future work section describes about the recommendations that can be consider to improve this project.

5.2 Conclusion

In conclusion, the smart energy meter with cost monitoring system using IoT is a useful and innovative project that helps households and businesses monitor and control their energy consumption in real-time. By using IoT technology, the system is able to gather data from the energy meter and transmit it to a cloud server, where it can be accessed and analyzed through a user-friendly interface. This allows users to see how much energy they are using and how much it is costing them, enabling them to make informed decisions about their energy consumption and potentially save money on their energy bills. Additionally, the system can be configured to send alerts when energy usage exceeds certain thresholds, allowing users to take immediate action to reduce their energy

consumption. Overall, the smart energy meter with cost monitoring system using IoT is a valuable tool for improving energy efficiency and reducing energy costs.

5.3 Recommendation

Based on the experiment conducted in for this project, the following recommendations are made to enhance the project in the future. Smartphones or android is a device that is easily carried around and used by vast majority of the global population. A development of android application can be introduced in this system to further improve the accessibility and convenience of users in accessing the power consumption data anytime and anywhere. This enable users to be able to monitor or observe the data through their phones without having to reach the cloud system using a computer device. Besides that, a database can be added that can calculate the total amount of usage. This act as a preventive measure to protect data from being lost from the cloud system or dependent to wireless connection. When internet connection is disconnected, the data can be stored in the database first until internet connection is available to continue the uploading of data collected into the cloud system. Finally, an alternative to employing Blynk would be to use the developer's own application instead of relying on a third party. This idea could improve the monitoring and control system's user interface by making it more flexible and user-friendly.

REFERENCES

[1] D. Kumar, A. Jain, and J. Kedia, “Design and Development of GSM based Energy Meter Cite this paper Design and Development of GSM based Energy Meter,” 2012.

[2] Z. Iqbal Rana, M. Waseem, T. Mahmood, and H. M. Zahid Iqbal M Waseem Tahir Mahmood, “Automatic Energy Meter Reading using Smart Energy Meter Energy Management in Smart Homes View project Automatic Energy Meter Reading using Smart Energy Meter”, doi: 10.13140/RG.2.1.1343.7928.

[3] “Survillance Of Environment using Node Mcu Based On Iot 786.”

[4] G. Mehta, R. Khanam, and V. K. Yadav, “A Novel IoT based Smart Energy Meter for Residential Energy Management in Smart Grid Infrastructure,” in *Proceedings of the 8th International Conference on Signal Processing and Integrated Networks, SPIN 2021*, 2021, pp. 47–52. doi: 10.1109/SPIN52536.2021.9566032.

[5] M. Ashiquzzaman, N. Afroze, T. M. Abdullah, and M. Abdullah, “Global Journal of researches in engineering Electrical and electronics engineering Design and Implementation of Wireless Digital Energy Meter using Microcontroller Design and Implementation of Wireless Digital Energy Meter using Microcontroller,” 2012.

[6] M. H. Piyal, M. Hossan, and Y. Arafat, “Remote ON-OFF control of energy meter and energy consumption data monitoring and storage system,” Aug. 2021. doi: 10.1109/PowerAfrica52236.2021.9543165.

[7] W. A. Jabbar, S. Annathurai, T. A. Tajul, and M. F. Mohd Fauzi, “Smart energy meter based on a long-range wide-area network for a stand-alone photovoltaic system,” *Expert Systems with Applications*, vol. 197, Jul. 2022, doi: 10.1016/j.eswa.2022.116703.

APPENDICES

CODING OF ESP32

```
#define BLYNK_TEMPLATE_ID           "TMPLIOlNfd61"
#define BLYNK_DEVICE_NAME           "Quickstart Device"
#define BLYNK_AUTH_TOKEN
"NXqZHOAM_sNDEpnNZMvZDTukxaQDAs_e"


// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial


#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <Filters.h>

char auth[] = BLYNK_AUTH_TOKEN;

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Keluarga Bahagia";
char pass[] = "zmalqp10";

int ZMPT101B1 = 35;
int ZMPT101B2 = 34;

float power1, power2, cost1, cost2, power3, power4;
float v1,v2,a1,a2;
float testFrequency1 = 50;
float windowLength1 = 100/testFrequency1;
float testFrequency2 = 50;
float windowLength2 = 100/testFrequency2;

int RawValue1 = 0;
float Volts_TRMS1;
int RawValue2 = 0;
float Volts_TRMS2;

float intercept1 = -3;
float slope1 = 0.28566351; //0.28566351 (241.1/844)
float intercept2 = -9;
float slope2 = 1.00878661; //0.28566351 (241.1/239)
```

```
unsigned long printPeriod1 = 1;
unsigned long previousMillis1 = 0;
unsigned long printPeriod2 = 1;
unsigned long previousMillis2 = 0;

RunningStatistics inputStats1;
RunningStatistics inputStats2;
// Attach virtual serial terminal to Virtual Pin V1
WidgetTerminal terminal(V3);

// You can send commands from Terminal to your hardware. Just use
// the same Virtual Pin as your Terminal Widget
BLYNK_WRITE(V3)
{

  // Ensure everything is sent
  terminal.flush();

  //terminal.print("You said:");
  //terminal.write(param.getBuffer(), param.getLength());
  //terminal.println();
}

void setup()
{
  // Debug console
  Serial.begin(115200);

  pinMode(36, INPUT);
  pinMode(39, INPUT);

  pinMode(ZMPT101B1, INPUT);
  pinMode(ZMPT101B2, INPUT);
  inputStats1.setWindowSecs( windowLength1 );
  inputStats2.setWindowSecs( windowLength2 );

  volt();
  delay(500);
  Serial.print(v1);
  Serial.print("          ");
  Serial.println(v2);

  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);

  // Clear the terminal content
  terminal.clear();
```

```
  // This will print Blynk Software version to the Terminal Widget when
  // your hardware gets connected to Blynk Server
  terminal.println(F("Blynk v" BLYNK_VERSION ": Device started"));
  terminal.println(F(" ------------"));
  terminal.println(F("Type 'Marco' and get a reply, or type"));
  terminal.println(F("anything else and get it printed back."));
  terminal.flush();
}
void AMP20(){
  float a11, ACSValue1 = 0.0, Samples1 = 0.0, AvgACS1 = 0.0, BaseVol1 = 2.193;
//Change BaseVol as per your reading in the first step.
  for (int x = 0; x < 500; x++) { //This would take 500 Samples
    ACSValue1 = analogRead(36);
    Samples1 = Samples1 + ACSValue1;
    delay (3);
  }
  AvgACS1 = Samples1/500;
  a11 = (((AvgACS1) * (3.3 / 4095.0)) - BaseVol1 ) / 0.100 ; //0.066V = 66mVol. This
is sensitivity of your ACS module.

  if(a11 < 0.1){
//  Serial.print("Amp1:  ");
//  Serial.print(0.00);
//  Serial.print("A (ref:20A) ");
  a1 = 0.00;
  }

  if(a11 > 0.1){
//  Serial.print("Amp1:  ");
//  Serial.print(a11);
//  Serial.print("A (ref:20A) ");
  a1 = a11;
  }
// // delay(100);
//float ACSValue1 = 0.0, Samples = 0.0, AvgACS1 = 0.0, BaseVol1 = 2.193;
//AvgACS1 = analogRead(36);
//Serial.print("\t");
//Serial.print("Amp(ref:20A): ");
//Serial.print("\t");
//Serial.print((((AvgACS1) * (3.3 / 4095.0)) - BaseVol1 ) / 0.100 );
//Serial.print(" A");
  }
void AMP5(){
  float a22, ACSValue = 0.0, Samples = 0.0, AvgACS = 0.0, BaseVol = 2.185; //Change
BaseVol as per your reading in the first step.
  for (int x = 0; x < 500; x++) { //This would take 500 Samples
    ACSValue = analogRead(39);
    Samples = Samples + ACSValue;
    delay (3);
  }
```

```
  AvgACS = Samples/500;
  a22 = (((AvgACS) * (3.3 / 4095.0)) - BaseVol ) / 0.185;
//  Serial.print("Amp2:  ");
//  Serial.print(a22); //0.066V = 66mVol. This is sensitivity of your ACS module.
//  Serial.print("A (ref:5A) ");
//  a2 = a22;

  if(a22 < 0.1){
//      Serial.print("Amp2:  ");
//      Serial.print(0.00); //0.066V = 66mVol. This is sensitivity of your ACS module.
//      Serial.print("A (ref:5A) ");
      a2 = 0.00;
    }

    if(a22 > 0.1){
//      Serial.print("Amp2:  ");
//      Serial.print(a22); //0.066V = 66mVol. This is sensitivity of your ACS module.
//      Serial.print("A (ref:5A) ");
      a2 = a22;
    }
//  //delay(100);

//float ACSValue2 = 0.0, Samples = 0.0, AvgACS2 = 0.0, BaseVol2 = 2.185;
//AvgACS2 = analogRead(39);
//Serial.print("\t");
//Serial.print("Amp(ref:5A): ");
//Serial.print("\t");
//Serial.print((((AvgACS2) * (3.3 / 4095.0)) - BaseVol2 ) / 0.185 );
//Serial.println(" A");
  }

void ReadVoltage1(){
   RawValue1 = analogRead(ZMPT101B1);  // read the analog in value:
   inputStats1.input(RawValue1);       // log to Stats function

//    if((unsigned long)(millis() - previousMillis1) >= printPeriod1) { //We calculate and
display every 1s
//      previousMillis1 = millis();  // update time

      Volts_TRMS1 = inputStats1.sigma()* slope1 + intercept1;
//      Volts_TRMS = Volts_TRMS*0.979;              //Further calibration if needed
      if (Volts_TRMS1 < 0){
        Serial.print("V1: ");
        Serial.print("\t");
        Serial.print(0.00);
        Serial.print(" V");
        v1 = 0.00;

        }
```

```
      if(Volts_TRMS1 > 0){
       Serial.print("V1: ");
       Serial.print("\t");
       Serial.print(Volts_TRMS1);
       Serial.print(" V");
       v1 = Volts_TRMS1;

       }
//     Serial.print("Non Calibrated: ");
//     Serial.print("\t");
//     Serial.print(inputStats1.sigma());
//     Serial.print("\t");

//}
}
void ReadVoltage2(){
   RawValue2 = analogRead(ZMPT101B2);  // read the analog in value:
   inputStats2.input(RawValue2);       // log to Stats function

//    if((unsigned long)(millis() - previousMillis2) >= printPeriod2) { //We calculate and
display every 1s
//     previousMillis2 = millis();  // update time

     Volts_TRMS2 = inputStats2.sigma()* slope2 + intercept2;
//     Volts_TRMS = Volts_TRMS*0.979;          //Further calibration if needed
     if (Volts_TRMS2 < 0){
       Serial.print("\t");
       Serial.print("V2: ");
       Serial.print("\t");
       Serial.print(0.00);
       Serial.println(" V");
       v2 = 0.00;
       }

      if(Volts_TRMS2 > 0){
       Serial.print("\t");
       Serial.print("V2: ");
       Serial.print("\t");
       Serial.print(Volts_TRMS2);
       Serial.println(" V");
       v2 = Volts_TRMS2;
       }
//     Serial.print("Non Calibrated2: ");
//     Serial.print("\t");
//     Serial.print(inputStats2.sigma());
//     Serial.print("\t");


// }
```

```
}

void volt(){

 for (int x = 0; x < 5000; x++) { //This would take 500 Samples
 ReadVoltage1();
 ReadVoltage2();
 }

 }

void power(){
 AMP20();
 AMP5();

 power1 = v1*a1;
 power2 = v2*a2;

 if(power1 <= 2000){              //For the first 200 kWh (1 - 200 kWh)
  cost1 = (power1/1000) * 0.21180;
  }
 if(power1 >= 2010 && power1 <=3000){ //For the next 100 kWh (201 - 300 kWh)
  cost1 = (power1/1000) * 0.31340;
  }
 if(power1 >= 3010 && power1 <=6000){ //For the next 300 kWh (301 - 600 kWh)
  cost1 = (power1/1000) * 0.51160;
  }
 if(power1 >= 6010 && power1 <=9000){ //For the next 300 kWh (601 - 900 kWh)
  cost1 = (power1/1000) * 0.51460;
  }
  if(power1 >= 9010){              //For the next kWh (901 kWh onwards)
  cost1 = (power1/1000) * 0.51710;
  }

  if(power2 <= 2000){              //For the first 200 kWh (1 - 200 kWh)
  cost2 = (power1/1000) * 0.21180;
  }
 if(power2 >= 2010 && power2 <=3000){ //For the next 100 kWh (201 - 300 kWh)
  cost2 = (power1/1000) * 0.31340;
  }
 if(power2 >= 3010 && power2 <=6000){ //For the next 300 kWh (301 - 600 kWh)
  cost2 = (power1/1000) * 0.51160;
  }
 if(power2 >= 6010 && power2 <=9000){ //For the next 300 kWh (601 - 900 kWh)
  cost2 = (power1/1000) * 0.51460;
  }
  if(power2 >= 9010){              //For the next kWh (901 kWh onwards)
  cost2 = (power1/1000) * 0.51710;
  }
```

```
Serial.print("V1:");
Serial.print("\t");
Serial.print(v1);
Serial.print("\t");
Serial.print("V");
Serial.print("\t");
Serial.print("V2:");
Serial.print("\t");
Serial.print(v2);
Serial.print("\t");
Serial.println("V");

Serial.print("Amp1:");
Serial.print("\t");
Serial.print(a1);
Serial.print("\t");
Serial.print("A(20A)");
Serial.print("\t");
Serial.print("Amp2:");
Serial.print("\t");
Serial.print(a2);
Serial.print("\t");
Serial.println("A(5A)");

Serial.print("P1:");
Serial.print("\t");
Serial.print(power1);
Serial.print("\t");
Serial.print("W");
Serial.print("\t");
Serial.print("P2:");
Serial.print("\t");
Serial.print(power2);
Serial.print("\t");
Serial.println("W");

Serial.print("C1:");
Serial.print("\t");
Serial.print(cost1);
Serial.print("\t");
Serial.print("RM/h");
Serial.print("\t");
Serial.print("C2:");
Serial.print("\t");
Serial.print(cost2);
Serial.print("\t");
Serial.println("RM/h");
Serial.println(" ");
```

```
//terminal.print("You said:");
//terminal.write(param.getBuffer(), param.getLength());
//terminal.println();
terminal.print("V1:");
terminal.print("\t");
terminal.print(v1);
terminal.print("\t");
terminal.print("V");
terminal.print("\t");
terminal.print("V2:");
terminal.print("\t");
terminal.print(v2);
terminal.print("\t");
terminal.println("V");

terminal.print("Amp1:");
terminal.print("\t");
terminal.print(a1);
terminal.print("\t");
terminal.print("A(20A)");
terminal.print("\t");
terminal.print("Amp2:");
terminal.print("\t");
terminal.print(a2);
terminal.print("\t");
terminal.println("A(5A)");

terminal.print("P1:");
terminal.print("\t");
terminal.print(power1);
terminal.print("\t");
terminal.print("W");
terminal.print("\t");
terminal.print("P2:");
terminal.print("\t");
terminal.print(power2);
terminal.print("\t");
terminal.println("W");

terminal.print("C1:");
terminal.print("\t");
terminal.print(cost1);
terminal.print("\t");
terminal.print("RM/h");
terminal.print("\t");
terminal.print("C2:");
terminal.print("\t");
terminal.print(cost2);
```

```
  terminal.print("\t");
  terminal.println("RM/h");
  terminal.println(" ");

    terminal.flush();

 Blynk.virtualWrite(V0, power1);
 Blynk.virtualWrite(V1, cost1);

  }
void loop()
{
  power();
  Blynk.run();
}
```

APPENDICES

CODING OF ESP32

```
#define BLYNK_TEMPLATE_ID           "TMPLIOlNfd61"
#define BLYNK_DEVICE_NAME           "Quickstart Device"
#define BLYNK_AUTH_TOKEN
"NXqZHOAM_sNDEpnNZMvZDTukxaQDAs_e"


// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial


#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <Filters.h>

char auth[] = BLYNK_AUTH_TOKEN;

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Keluarga Bahagia";
char pass[] = "zmalqp10";

int ZMPT101B1 = 35;
int ZMPT101B2 = 34;

float power1, power2, cost1, cost2, power3, power4;
float v1,v2,a1,a2;
float testFrequency1 = 50;
float windowLength1 = 100/testFrequency1;
float testFrequency2 = 50;
float windowLength2 = 100/testFrequency2;

int RawValue1 = 0;
float Volts_TRMS1;
int RawValue2 = 0;
float Volts_TRMS2;

float intercept1 = -3;
float slope1 = 0.28566351; //0.28566351 (241.1/844)
float intercept2 = -9;
float slope2 = 1.00878661; //0.28566351 (241.1/239)

unsigned long printPeriod1 = 1;
```

```
unsigned long previousMillis1 = 0;
unsigned long printPeriod2 = 1;
unsigned long previousMillis2 = 0;

RunningStatistics inputStats1;
RunningStatistics inputStats2;
// Attach virtual serial terminal to Virtual Pin V1
WidgetTerminal terminal(V3);

// You can send commands from Terminal to your hardware. Just use
// the same Virtual Pin as your Terminal Widget
BLYNK_WRITE(V3)
{

  // Ensure everything is sent
  terminal.flush();

  //terminal.print("You said:");
  //terminal.write(param.getBuffer(), param.getLength());
  //terminal.println();
}

void setup()
{
  // Debug console
  Serial.begin(115200);

  pinMode(36, INPUT);
  pinMode(39, INPUT);

  pinMode(ZMPT101B1, INPUT);
  pinMode(ZMPT101B2, INPUT);
  inputStats1.setWindowSecs( windowLength1 );
  inputStats2.setWindowSecs( windowLength2 );

  volt();
  delay(500);
  Serial.print(v1);
  Serial.print("          ");
  Serial.println(v2);

  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);

  // Clear the terminal content
  terminal.clear();

  // This will print Blynk Software version to the Terminal Widget when
```

```
  // your hardware gets connected to Blynk Server
  terminal.println(F("Blynk v" BLYNK_VERSION ": Device started"));
  terminal.println(F(" ------------ "));
  terminal.println(F("Type 'Marco' and get a reply, or type"));
  terminal.println(F("anything else and get it printed back."));
  terminal.flush();
}
void AMP20(){
  float a11, ACSValue1 = 0.0, Samples1 = 0.0, AvgACS1 = 0.0, BaseVol1 = 2.193;
//Change BaseVol as per your reading in the first step.
  for (int x = 0; x < 500; x++) { //This would take 500 Samples
    ACSValue1 = analogRead(36);
    Samples1 = Samples1 + ACSValue1;
    delay (3);
  }
  AvgACS1 = Samples1/500;
  a11 = (((AvgACS1) * (3.3 / 4095.0)) - BaseVol1 ) / 0.100 ; //0.066V = 66mVol. This
is sensitivity of your ACS module.

  if(a11 < 0.1){
//  Serial.print("Amp1:  ");
//  Serial.print(0.00);
//  Serial.print("A (ref:20A) ");
  a1 = 0.00;
  }

  if(a11 > 0.1){
//  Serial.print("Amp1:  ");
//  Serial.print(a11);
//  Serial.print("A (ref:20A) ");
  a1 = a11;
  }
// // delay(100);
//float ACSValue1 = 0.0, Samples = 0.0, AvgACS1 = 0.0, BaseVol1 = 2.193;
//AvgACS1 = analogRead(36);
//Serial.print("\t");
//Serial.print("Amp(ref:20A): ");
//Serial.print("\t");
//Serial.print((((AvgACS1) * (3.3 / 4095.0)) - BaseVol1 ) / 0.100 );
//Serial.print(" A");
  }
void AMP5(){
  float a22, ACSValue = 0.0, Samples = 0.0, AvgACS = 0.0, BaseVol = 2.185; //Change
BaseVol as per your reading in the first step.
  for (int x = 0; x < 500; x++) { //This would take 500 Samples
    ACSValue = analogRead(39);
    Samples = Samples + ACSValue;
    delay (3);
  }
  AvgACS = Samples/500;
```

```
  a22 = (((AvgACS) * (3.3 / 4095.0)) - BaseVol ) / 0.185;
//  Serial.print("Amp2:  ");
//  Serial.print(a22); //0.066V = 66mVol. This is sensitivity of your ACS module.
//  Serial.print("A (ref:5A) ");
//  a2 = a22;

  if(a22 < 0.1){
//      Serial.print("Amp2:  ");
//      Serial.print(0.00); //0.066V = 66mVol. This is sensitivity of your ACS module.
//      Serial.print("A (ref:5A) ");
      a2 = 0.00;
    }

    if(a22 > 0.1){
//      Serial.print("Amp2:  ");
//      Serial.print(a22); //0.066V = 66mVol. This is sensitivity of your ACS module.
//      Serial.print("A (ref:5A) ");
      a2 = a22;
    }
//  //delay(100);

//float ACSValue2 = 0.0, Samples = 0.0, AvgACS2 = 0.0, BaseVol2 = 2.185;
//AvgACS2 = analogRead(39);
//Serial.print("\t");
//Serial.print("Amp(ref:5A): ");
//Serial.print("\t");
//Serial.print((((AvgACS2) * (3.3 / 4095.0)) - BaseVol2 ) / 0.185 );
//Serial.println(" A");
  }

void ReadVoltage1(){
   RawValue1 = analogRead(ZMPT101B1);  // read the analog in value:
   inputStats1.input(RawValue1);       // log to Stats function

//    if((unsigned long)(millis() - previousMillis1) >= printPeriod1) { //We calculate and
display every 1s
//      previousMillis1 = millis();  // update time

      Volts_TRMS1 = inputStats1.sigma()* slope1 + intercept1;
//      Volts_TRMS = Volts_TRMS*0.979;              //Further calibration if needed
      if (Volts_TRMS1 < 0){
        Serial.print("V1: ");
        Serial.print("\t");
        Serial.print(0.00);
        Serial.print(" V");
        v1 = 0.00;

        }

       if(Volts_TRMS1 > 0){
```

```
        Serial.print("V1: ");
        Serial.print("\t");
        Serial.print(Volts_TRMS1);
        Serial.print(" V");
        v1 = Volts_TRMS1;

        }
//      Serial.print("Non Calibrated: ");
//      Serial.print("\t");
//      Serial.print(inputStats1.sigma());
//      Serial.print("\t");

//}
}
void ReadVoltage2(){
   RawValue2 = analogRead(ZMPT101B2);  // read the analog in value:
   inputStats2.input(RawValue2);       // log to Stats function

//    if((unsigned long)(millis() - previousMillis2) >= printPeriod2) { //We calculate and
display every 1s
//      previousMillis2 = millis();  // update time

     Volts_TRMS2 = inputStats2.sigma()* slope2 + intercept2;
//      Volts_TRMS = Volts_TRMS*0.979;            //Further calibration if needed
     if (Volts_TRMS2 < 0){
       Serial.print("\t");
       Serial.print("V2: ");
       Serial.print("\t");
       Serial.print(0.00);
       Serial.println(" V");
       v2 = 0.00;
       }

      if(Volts_TRMS2 > 0){
       Serial.print("\t");
       Serial.print("V2: ");
       Serial.print("\t");
       Serial.print(Volts_TRMS2);
       Serial.println(" V");
       v2 = Volts_TRMS2;
       }
//      Serial.print("Non Calibrated2: ");
//      Serial.print("\t");
//      Serial.print(inputStats2.sigma());
//      Serial.print("\t");


// }
```

```
}

void volt(){

  for (int x = 0; x < 5000; x++) { //This would take 500 Samples
  ReadVoltage1();
  ReadVoltage2();
  }

  }

void power(){
  AMP20();
  AMP5();

  power1 = v1*a1;
  power2 = v2*a2;

  if(power1 <= 2000){                 //For the first 200 kWh (1 - 200 kWh)
    cost1 = (power1/1000) * 0.21180;
    }
  if(power1 >= 2010 && power1 <=3000){ //For the next 100 kWh (201 - 300 kWh)
    cost1 = (power1/1000) * 0.31340;
    }
  if(power1 >= 3010 && power1 <=6000){ //For the next 300 kWh (301 - 600 kWh)
    cost1 = (power1/1000) * 0.51160;
    }
  if(power1 >= 6010 && power1 <=9000){ //For the next 300 kWh (601 - 900 kWh)
    cost1 = (power1/1000) * 0.51460;
    }
   if(power1 >= 9010){                //For the next kWh (901 kWh onwards)
    cost1 = (power1/1000) * 0.51710;
    }

    if(power2 <= 2000){                 //For the first 200 kWh (1 - 200 kWh)
    cost2 = (power1/1000) * 0.21180;
    }
  if(power2 >= 2010 && power2 <=3000){ //For the next 100 kWh (201 - 300 kWh)
    cost2 = (power1/1000) * 0.31340;
    }
  if(power2 >= 3010 && power2 <=6000){ //For the next 300 kWh (301 - 600 kWh)
    cost2 = (power1/1000) * 0.51160;
    }
  if(power2 >= 6010 && power2 <=9000){ //For the next 300 kWh (601 - 900 kWh)
    cost2 = (power1/1000) * 0.51460;
    }
   if(power2 >= 9010){                //For the next kWh (901 kWh onwards)
    cost2 = (power1/1000) * 0.51710;
    }
```

```
Serial.print("V1:");
Serial.print("\t");
Serial.print(v1);
Serial.print("\t");
Serial.print("V");
Serial.print("\t");
Serial.print("V2:");
Serial.print("\t");
Serial.print(v2);
Serial.print("\t");
Serial.println("V");

Serial.print("Amp1:");
Serial.print("\t");
Serial.print(a1);
Serial.print("\t");
Serial.print("A(20A)");
Serial.print("\t");
Serial.print("Amp2:");
Serial.print("\t");
Serial.print(a2);
Serial.print("\t");
Serial.println("A(5A)");

Serial.print("P1:");
Serial.print("\t");
Serial.print(power1);
Serial.print("\t");
Serial.print("W");
Serial.print("\t");
Serial.print("P2:");
Serial.print("\t");
Serial.print(power2);
Serial.print("\t");
Serial.println("W");

Serial.print("C1:");
Serial.print("\t");
Serial.print(cost1);
Serial.print("\t");
Serial.print("RM/h");
Serial.print("\t");
Serial.print("C2:");
Serial.print("\t");
Serial.print(cost2);
Serial.print("\t");
Serial.println("RM/h");
Serial.println(" ");
```

```
//terminal.print("You said:");
//terminal.write(param.getBuffer(), param.getLength());
//terminal.println();
terminal.print("V1:");
terminal.print("\t");
terminal.print(v1);
terminal.print("\t");
terminal.print("V");
terminal.print("\t");
terminal.print("V2:");
terminal.print("\t");
terminal.print(v2);
terminal.print("\t");
terminal.println("V");

terminal.print("Amp1:");
terminal.print("\t");
terminal.print(a1);
terminal.print("\t");
terminal.print("A(20A)");
terminal.print("\t");
terminal.print("Amp2:");
terminal.print("\t");
terminal.print(a2);
terminal.print("\t");
terminal.println("A(5A)");

terminal.print("P1:");
terminal.print("\t");
terminal.print(power1);
terminal.print("\t");
terminal.print("W");
terminal.print("\t");
terminal.print("P2:");
terminal.print("\t");
terminal.print(power2);
terminal.print("\t");
terminal.println("W");

terminal.print("C1:");
terminal.print("\t");
terminal.print(cost1);
terminal.print("\t");
terminal.print("RM/h");
terminal.print("\t");
terminal.print("C2:");
terminal.print("\t");
terminal.print(cost2);
terminal.print("\t");
```

```
  terminal.println("RM/h");
  terminal.println(" ");

    terminal.flush();

 Blynk.virtualWrite(V0, power1);
 Blynk.virtualWrite(V1, cost1);

  }
void loop()
{
  power();
  Blynk.run();
}
```

APPENDIX 1 Coding of ESP-32