BATTERY CHARGING MONITORING & CONTROL DEVICE USING INTERNET OF THINGS (IOT)

BY

OFUZIM, ONYERO WALTER ENG1503848

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE AWARD OF BACHELOR OF ENGINEERING (B.ENG) DEGREE

IN

THE DEPARTMENT OF ELECTRICAL/ELECTRONIC ENGINEERING, FACULTY OF ENGINEERING, UNIVERSITY OF BENIN, BENIN CITY, NIGERIA.

JULY 2021

BATTERY CHARGING MONITORING & CONTROL DEVICE USING INTERNET OF THINGS (IOT)

BY

OFUZIM, ONYERO WALTER ENG1503848

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE AWARD OF BACHELOR OF ENGINEERING (B.ENG) DEGREE

IN

THE DEPARTMENT OF ELECTRICAL/ELECTRONIC ENGINEERING, FACULTY OF ENGINEERING, UNIVERSITY OF BENIN, BENIN CITY, NIGERIA.

JULY 2021

CERTIFICATION

This is to certify that this project work was carried out by **Ofuzim Onyero Walter,** MAT. No: **ENG1503848** of the department of Electrical/Electronic Engineering, University of Benin, Edo state, Nigeria.

_____

Signature and Date

Engr. Prophet Tepu Aikhoje

Project Supervisor

_____

Signature and Date

Prof. (Mrs) P.O. Orukpe

Head of Department

DEDICATION

This project is dedicated to God Almighty who is the source of life, knowledge and creativity amongst other attributes too numerous to list and also my family for their support and encouragement.

ACKNOWLEDGEMENT

I am overwhelmed in all humbleness and gratefulness to acknowledge my depth to my family for their love, financial assistances, care, advices and prayers for me all through these years.

I would like to express my special thanks of gratitude to Engr. Tepu Prophet Aikhoje my project supervisor for his tolerance, patience, love and assistance.

Prof. (Mrs) P.O. Orukpe has always show me love, her love has shaped me to a refine person.

I am indeed indebted to all my lecturers who painstakingly imbibed in me the knowledge and understanding I have now. It will be unwieldy for me to list all of you but all I can say is that you are men and women of great intellect and enviable repute.

My salutations go to Abiodun Timothy Ganiyu, Chibogu Esther, Eru Michael and the rest of my classmates of EEE class of 2020, whose suggestions and guidance were critical to the success of this study.

ABSTRACT

The Internet of Things (IoT) has so far had many applications, one of these are smart power monitoring and control systems. Incorporation of the IoTs to Battery charging and remote monitoring & controlling device can be solve the issues of remote charging to assuage self-discharging and overcharge of battery. This work is aimed charging of batteries and monitoring/control of state of charge (soc) remotely using internet of things (IoT).

The project is based on the design and construction of a battery charging, monitoring & control device using internet of things (IoT). This device can monitor the battery charging (soc) and control it. In this work a **PIC16F887 MICROCONTROLLER** is the used to control the functions of the system, the webserver which interfaced by an **A6 GSM/GPRS MODULE** act as the IoT platform. As the webserver is platform independent, it is used on any device, like mobiles and laptops.

Three tests were carried out, power supply test, GPRS module test, switching circuit test and system test respectively. The power supply test, the voltage regulators 7805 & LM317 produced 5V and 4.2V respectively for power supply to respective design components. In the GPRS module test, the GPRS module communicated with the microcontroller, the switching circuit test, each actuator pins of the relay were connected to power and ground to determine if the relays will actuate. While, the system test, the entire circuitry was tested and cross examined for errors like short circuits, lead flux joining unwanted links, proper insertion integrated circuit (IC) pin layout and also checking of ICs and its pin numbers and cross examined once again before powering the system.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND OF STUDY

Energy is an important and necessary ingredient for development and a powerful engine of social and economic change anywhere in the world. In recent years, there has been considerable increase in the need for alternative or renewable source of energy globally. The need for electrical energy can never be over-emphasize in the contemporary world; it is indeed fundamental to the fulfilment of basic individual and community needs in modern society. Lighting and heating a house, running a factory, lighting a street, keeping a hospital open and operational, provision of potable water, running an institution, etc. all require electrical energy.

Renewable energy is energy generated from natural resources such as sunlight, wind, rain, tides, and geothermal heat. It is derived from natural processes that are constantly replenish. Renewable energy sources include solar, hydro, wind, biomass etc. The energy(power) from these energy sources varies depending on the time of day, nature of weather and others; and the time of usage of this energy varies thus generated energy need to be stored in a battery.

In renewable energy system, battery is a key component as it allows energy from different sources to be stored and use in a later time. The longevity of battery depends on how it is use, how and how often it is charge.

A battery is an electrical storage device. Batteries do not make electricity; they store it, just like a water tank stores water for future use. As chemicals in the battery change, electrical energy is stored or released. In the rechargeable batteries, this process can be repeated many times. Practically, all batteries used in renewable energy systems except the smallest backup systems are deep cycle batteries. It is important to note here that all of the batteries commonly used in deep cycle applications are Lead-Acid. The stand-alone Photovoltaic systems, encounter the problem of sustainability on a number of fronts, among which are the initial and replacement cost of the storage batteries. The IEA (International Energy Agency) in its Photovoltaic Power Systems (PVPS) program described the storage batteries as the weakest components of the PV systems and World Energy Council in its 2007 Survey of Energy Resources called it a “constraint” to the phenomenal growth in the usage of solar PVs.

The efficiency of the batteries used in the PV systems changes as the characteristics of the battery changes, which in-turn affects how long the battery can store or supply power in PV systems. Different methods or ways have been employed to improve the efficiency of batteries generally which include but not limited to: improving the technology used in producing the battery, improving the technology for charging the batteries and the chargers for the batteries, having protective circuit(s) to protect batteries against overload or short circuit, studying the charging and discharging characteristics of batteries in order to determine the proper load for the different types of batteries, and determining the best ways to charge a battery.

One of the most common actions taken to increase battery efficiency or increase the live span of a battery meant for renewable energy is to be monitored or controlled the state of the battery during charging and discharging.

Monitoring and control of devices can be done manually by humans, remotely by humans or automatically. To some extent, automatic control and monitoring still requires human intervention for proper monitoring and control.

Remote control and monitoring can be done using short-range communication and long-range communication. Short-range communication include use of WIFI, Bluetooth. Long-range communication include use of GSM/GPRS communication. In this kind of application, long-range communication is the best in order to enable communication irrespective of user location.

In this project, the monitoring and controlled of battery state will solely be done during charging in close and long range. The long-range communication system used in this project is the internet of things (IOT). The internet of things is referring to the interconnectivity of things together through the internet. Battery state monitoring and control through a device and through the internet (using a mobile application) which can be access anytime and from anywhere is what this project is about.

1.2 PROBLEM STATEMENT

There are some problems that occur while charging a battery, one of the problems is that the battery tends to undergo overcharging because typical lead-acid battery like any other battery needs to be charged under conducive environment and electrical conditions for a sufficient charging. Another problem that accompanies charging a battery is monitoring of battery level

and health status during charging. Conventional battery monitoring is usually done by manual inspection and control of charging circuit anytime and from anywhere prove to be difficult.

To solve the above problem, a control and monitoring device that can be used to control and monitor the status of a battery remotely need to be used to monitor and control the on/off state of a battery, have to be use along with a charger for charging the battery.

1.3 AIM

The aim of this project is to design and construct a battery charging & controlling device using internet of things (IOT).

1.4 OBJECTIVES

To achieve the aim, these are the objectives of the project:

1. Design of a battery charger monitoring and control device
2. Procurement of components and Construction of the battery monitoring and control device.
3. Development of server side of the battery monitoring and control system and development of mobile application
4. Testing of mobile application with server side

1.5 METHODOLOGY

To achieve the objectives, the following procedures were carried out:

- Design and simulation of battery charger monitoring and control device using Proteus Design Suite.
- Construction of simulated design on veroboard and programming of microcontroller using C programming language.
- Development and hosting of webserver using asp.net C#.
- Development and programming of mobile application using Java programming language.

1.6 SCOPE OF WORK

This project work would be the design and construction of a battery charger control and monitoring device and also a mobile application for monitoring and controlling the battery charger.

1.7 OVERVIEW OF THE PROJECT WRITEUP

This project also involves presentation of a technical paper divided into five chapters.

Chapter one: is the introduction of the work. This is where the need for this work is emphasized including the aims and objective, scope, methodology and relevance of the project

Chapter two: is on the review of related literature. The chapter highlights past related project work as well as discussion of the key components used in this project.

Chapter three: discusses in detail the design of the project, showing the hardware and software designs.

Chapter four: is an outline of the procedures used in assembling as well as testing and results.

Chapter five: concludes the project as well as citing some recommendations for future improvement.

1.8 RELEVANCE OF PROJECT

This project focuses on battery state monitoring and can be relevant in the below operations/use:

1. Industrial equipment monitoring.
2. Industrial equipment control and automation
3. Home monitoring
4. Home appliances control and automation.

CHAPTER TWO

LITERATURE REVIEW

In this chapter is a review of literature related to this work. This includes articles and journal and review of all the components used in the project.

2.1 A BRIEF HISTORY OF INTERNET OF THINGS

The Internet of Things (IoT) has not been around for very long but there have been visions of machines communicating with one another since the early 1800s. Machines have been providing direct communications since the telegraph (the first landline) was developed in the 1830s and 1840s. Described as "wireless telegraphy," the first radio voice transmission took place on June 3, 1900, providing another necessary component for developing the Internet of Things. The development of computers began in the 1950s.

The Internet, itself a significant component of the IoT, started out as part of DARPA (Defence Advanced Research Projects Agency) in 1962, and evolved into ARPANET in 1960. In the 1980s, commercial service providers began supporting public use of ARPANET, allowing it to evolve into our modern Internet. Global Positioning Satellites (GPS) became a reality in early 1993, with the Department of Defence providing a stable, highly functional system of 24 satellites. This was quickly followed by privately owned, commercial satellites being placed in orbit. Satellites and landlines provide basic communications for much of the IoT.

One additional and important component in developing a functional IoT was IPV6's remarkably intelligent decision to increase address space. The address space expansion means that we could assign an IPV6 address to every atom on the surface of the earth, and still have enough addresses left to do another 100+ earths (Steve Leibson, 2008). Put another way, we are not going to run out of internet addresses anytime soon.

2.2 REALIZING THE CONCEPT

The Internet of Things, as a concept, was not officially named until 1999. One of the first examples of an Internet of Things is from the early 1980s, and was a Coca Cola machine, located at the Carnegie Melon University. Local programmers would connect by Internet to

the refrigerated appliance and check to see if there was a drink available, and if it was cold, before making the trip.

By the year 2013, the Internet of Things had evolved into to a system using multiple technologies, ranging from the Internet to wireless communication and from micro-electromechanical systems (MEMS) to embedded systems. The traditional fields of automation (including the automation of buildings and homes), wireless sensor networks, GPS, control systems, and others, all support the IoT.

Simply stated, the Internet of Things consists of any device with an on/off switch connected to the Internet (Brown Eric, 2016). This includes almost anything you can think of, ranging from cell phones to building maintenance to the jet engine of an airplane. Medical devices, such as a heart monitor implant or a biochip transponder in a farm animal, can transfer data over a network and are members the IoT. If it has an off/on switch, then it can, theoretically, be part of the system. The IoT is also the movement of small packets of data to a large set of nodes, so as to integrate and automate everything from home appliances to entire factories (Reza Raji, 1994). The IoT consists of a gigantic network of internet connected "things" and devices. Ring, a doorbell that links to your smart phone, provides an excellent example of a recent addition to the Internet of Things. Ring signals you when the doorbell is pressed, and lets you see who it is and to speak with them.

Today computers, and, therefore, the Internet, are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes (a petabyte is 1,024 terabytes) of data available on the Internet were first captured and created by human beings by typing, pressing a record button, taking a digital picture or scanning a bar code. The problem is, people have limited time, attention, and accuracy. All of which means they are not very good at capturing data about things in the real world. If we had computers that knew everything there was to know about things, using data they gathered without any help from us, we would be able to track and count everything and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling and whether they were fresh or past their best (Kevin Ashton, 1999).

Kevin Ashton believed Radio Frequency Identification (RFID) was a prerequisite for the Internet of Things. He concluded if all devices were "tagged," computers could manage,

track, and inventory them. To some extent, the tagging of things has been achieved through technologies such as digital watermarking, barcodes, and QR codes. Inventory control is one of the more obvious advantages of the Internet of Things.

2.3 CONNECTING DEVICES IN NEW WAYS

When thinking of the IoT, consider the idea, “any device capable, can be interconnected with other devices.” The IoT is ripe for new and creative ideas to add to the tasks already in use. Imagine an alarm waking you at 6 AM in the morning, and then simultaneously signalling your coffee maker to turn on and start brewing coffee. Imagine your printer knowing when you are running low on paper, and automatically ordering more. Imagine the watch on your wrist telling you “Where” you have been the most productive, while at work. The IoT can be used to organize such things as transportation networks. “Smart cities” can use it to reduce waste and maximize the efficient use of energy.

In truth, the IoT provides a nearly endless supply of opportunities to interconnect our devices and equipment. In terms of creativity, this field is wide open, with an infinite number of ways to “interconnect the devices.” It can be an exciting time for innovative individuals, in part, because we don’t fully understand the impact of these interconnections. The IoT offers both opportunities and potential security problems. At present, the Internet of Things is best viewed with an open mind, for purposes of creativity, and a defensive posture for purposes of privacy and security.

2.4 CUSTOMER PRIVACY

As sensors and video cameras become more common place, especially in public spaces, consumers have less and less knowledge about the information_being_collected, and no way to avoid it (Peter-Paul Verbeek, 2011).

Many people are uncomfortable with the idea of companies collecting information about them, and even more uncomfortable having that information sold to anyone and everyone. Older people dislike having information about themselves collected more than younger people, but according to one survey, about 45% of “all” respondents did not trust companies to use the data they collected to protect their privacy.

Currently, choices regarding privacy are very black and white, or on/off. The customer is forced to give up all privacy, (often in an agreement so convoluted people do not bother to go through it) or the customer simply cannot access the service. This has led to continuing discussions about consumer privacy and how to best educate consumers regarding privacy and the accessibility of data.

2.5 SECURITY

While there are steps to take to help ensure security, it should come as no surprise this issue has become a significant concern with the growth of the IoT. Literally billions of devices are being interconnected together, making it possible (eventually) for someone to hack into your coffee maker, and then access your entire network. Connected appliances can even spy on people in their homes including televisions, kitchen appliances, cameras, and thermostats (Steinberg Joseph, 2014). The Internet of Things also makes businesses all around the world more open to security threats (Greenberg Andy, 2015). Additionally, data sharing and privacy becomes issues when using the Internet of Things. Consider how concerns will grow when billions of devices are interconnected. Some businesses will be faced with storing the massive amounts of information these devices will be producing. They will need to find a method of securely storing the data, while still being able to access, track, and analyse the huge amounts of it been generated.

James Lewis, who is a cybersecurity researcher for the Centre for Strategic and International Studies, has written a report describing how the Internet of Things' interconnections will allow computer hackers to wreak havoc through interconnected devices. The threat is so real, even the Federal Trade Commission has gotten involved, wanting to know how to guarantee privacy, and how security safeguards are being installed in new Internet-connected devices. For example, new cars can now be hijacked by way of their Wi-Fi connections. Consider the threat of hackers when automated driving becomes popular. Security and risk management should not be taken lightly when creating new ways to use the Internet of Things.

2.6 COMPONENT REVIEW

Below is a list of all components used in the project.

1. PIC16F887 MICROCONTROLLER

2. RESISTORS

3. TRANSISTORS

4. CAPACITORS

5. DIODES

6. CRYSTAL OSCILLATOR

7. RELAYS

8. VOLTAGE REGULATOR

9. A6 GSM/GPRS MODULE

10. TEMPERATURE SENSOR

11. ELECTRONIC SWITCH

13. LCD (LIQUID CRYSTAL DISPLAY)

14. ELECTRONIC CONNECTORS

2.6.1 PIC16F887 MICROCONTROLLER

PIC means peripheral interface controller and it has different families. The PIC16F887 Microcontroller is a 40-pin microcontroller, and it is among the 16 series family of the PIC microcontroller. It is a well-known product of the microchip. It features all the components which modern microcontrollers normally have. For its low price, wide range of application, high quality, and easy availability, it is an ideal solution in application such as the control of different process in industry, machine control devices, measurement of different values etc. Some of its main features are listed below.

- RISC ARCHITECTURE
  Only 35 instructions to learn
  All single-cycle instructions except branches

- OPERATING FREQUENCY 0-20 MHz
- PRECISION INTERNAL OSCILLATOR

  Factory calibrated

  Software selectable frequency range of 8MHz to 31KHz
- POWER SUPPLY VOLTAGE 2.0-5.5V

  Consumption: 220Ua (2.0V, 4MHz), 11Ua (2.0V, 32 KHz) 50nA (stand-by mode)
- POWER-SAVING SLEEP MODE
- BROWN-OUT RESET(BOR) WITH SOFTWARE CONTROL OPTION
- 35 INPUT/OUTPUT PINS

  High current source/sink for direct LED drive

  Software and individually programmable pull-up resistor

  Interrupt-on-Change pin
- 8K ROM MEMORY IN FLASH TECHNOLOGY

  Chip can be programmed even embedded in the target device
- 256 BYTES EEPROM MEMORY

  Data can be written more than 1000000 times
- 368 bytes ram memory
- A/D CONVERTER

  14-Channels

  10-bit resolution
- 3 INDEPENDENT TIMERS/COUNTERS
- WATCH-DOG TIMER
- ANALOGUE COMPARATOR MODULE WITH

  Two analogue comparators

  Fixed voltage reference (0.6V)

  Programmable on –chip voltage reference
- PWM OUTPUT STEERING CONTROL
- ENHANCED USART MODULE

  Supports RS-485, RS-232 and LIN2.0

  Auto-Baud Rate etc.
- MASTER SYNCHRONOUS SERIAL PORT (MSSP)

Supports SPI and I2C mode

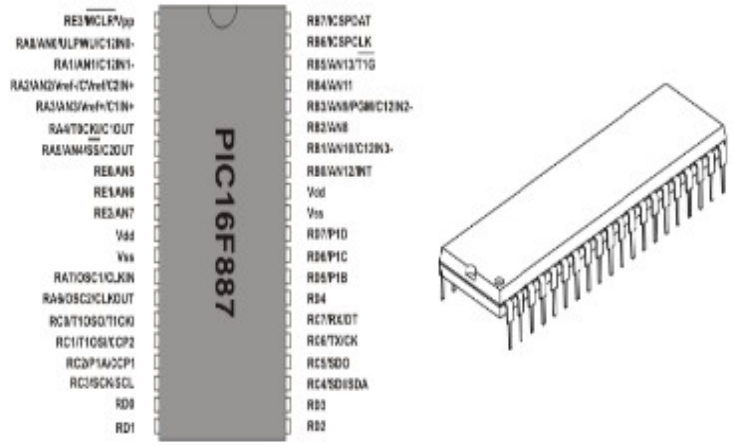

Figure 2.1: 40-PIN MICROCONTROLLER

Table 2.1: The following are the pin-out of a 40-pin microcontroller

Name	Number (DIP 40)	Function	Description
RA7/OSC1/CLKIN	13	RA7	General purpose I/O port A
		OSC1	Crystal Oscillator Input
		CLKIN	External Clock Input
RA6/OSC2/CLKOUT	14	OSC2	Crystal Oscillator Output
		CLKO	Fosc/4 Output
		RA6	General purpose I/O port A
RC0/T1OSO/T1CKI	15	RC0	General purpose I/O port C
		T1OSO	Timer T1 Oscillator Output
		T1CKI	Timer T1 Clock Input
RC1/T1OSO/T1CKI	16	RC1	General purpose I/O port C
		T1OSI	Timer T1 Oscillator Input
		CCP2	CCP1 and PWM1 module I/O
RC2/P1A/CCP1	17	RC2	General purpose I/O port C
		P1A	PWM Module Output
		CCP1	CCP1 and PWM1 module I/O
RC3/SCK/SCL	18	RC3	General purpose I/O port C
		SCK	MSSP module Clock I/O in SPI mode
		SCL	MSSP module Clock I/O in I^2C mode
RD0	19	RD0	General purpose I/O port D
RD1	20	RD1	General purpose I/O port D
RD3	22	RD3	General purpose I/O port D
RC4/SDI/SDA	23	RC4	General purpose I/O port A
		SDI	MSSP module *Data* input in SPI mode
		SDA	MSSP module *Data* I/O in I^2C mode
RC5/SDO	24	RC5	General purpose I/O port C
		SDO	MSSP module *Data* output in SPI mode
RC6/TX/CK	25	RC6	General purpose I/O port C
		TX	USART Asynchronous Output
		CK	USART Synchronous Clock
RC7/RX/DT	26	RC7	General purpose I/O port C
		RX	USART Asynchronous Input
		DT	USART Synchronous Data

RD4	27	RD4	General purpose I/O port D
RD5/P1B	28	RD5	General purpose I/O port D
		P1B	PWM Output
RD6/P1C	29	RD6	General purpose I/O port D
		P1C	PWM Output
RD7/P1D	30	RD7	General purpose I/O port D
		P1D	PWM Output
Vss	31	-	Ground (GND)
Vdd	32	+	Positive Supply
RB0/AN12/INT	33	RB0	General purpose I/O port B
		AN12	A/D Channel 12
		INT	External Interrupt
RB1/AN10/C12INT3-	34	RB1	General purpose I/O port B
		AN10	A/D Channel 10
		C12INT3-	Comparator C1 or C2 Negative Input
RB2/AN8	35	RB2	General purpose I/O port B
		AN8	A/D Channel 8
RB3/AN9/PGM/C12IN2-	36	RB3	General purpose I/O port B
		AN9	A/D Channel 9
		PGM	Programming enable pin
		C12IN2-	Comparator C1 or C2 Negative Input
RB4/AN11	37	RB4	General purpose I/O port B
		AN11	A/D Channel 11
RB5/AN13/T1G	38	RB5	General purpose I/O port B
		AN13	A/D Channel 13
		T1G	Timer T1 External Input
RB6/ICSPCLK	39	RB6	General purpose I/O port B
		ICSPCLK	Serial programming Clock
RB7/ICSPDAT	40	RB7	General purpose I/O port B
		ICSPDAT	Programming enable pin

2.6.2 RESISTORS

Resistor is a two terminal device that opposes the flow of electric current. The resistance to the flow of electricity limits the current flowing into the electronic component. Resistors have no polarity and the unit of measurement of resistance is the ohms. Resistors are made from the following materials:

1. semiconductor materials

2. carbon composition moulding

3. carbon film

4. thick and thin film

5. metal film

6. metal oxide film

7. Wire wound, etc.

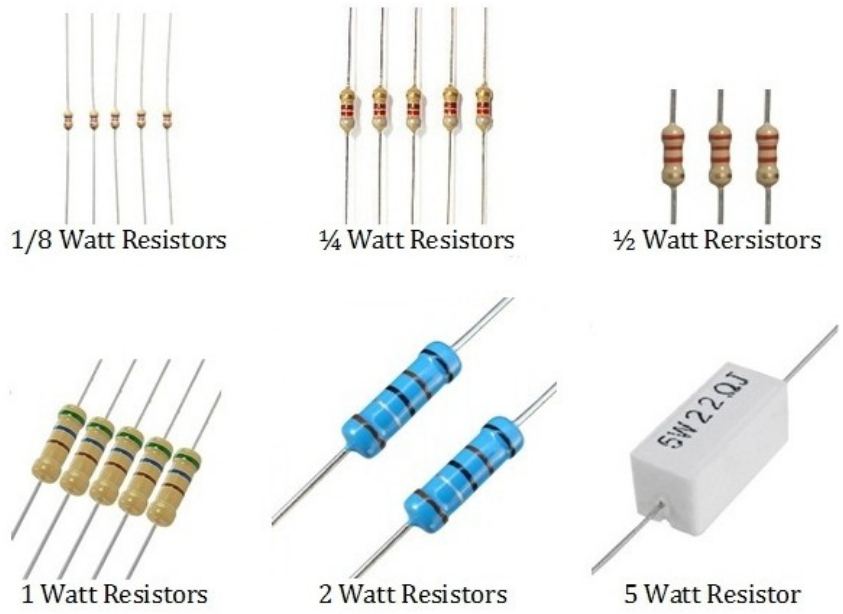

Figure 2.2: RESISTORS AND THEIR WATTAGE

There are two classifications of resistors namely: fixed and variable resistors.

1. Fixed resistors: A fixed resistor has fixed resistance that cannot be changed. Common examples of fixed resistors are carbon composition moulding type which has colour bands. Another is referred to as choke resistors. They are white in colour and usually of low resistance and high-power rating.

2. Variable resistors: A variable resistor is a resistor whose resistance can be varied. Some of these resistors have knobs which can be turned to vary the resistance. There are many types of variable resistors namely.

i. Ordinary variable resistor: This is the common type of variable resistor referred to as 'volume' by market people because of its use in controlling the volume of some electronic devices like radio. It has three terminals. When the two extreme terminals are used, it acts as a fixed resistor while it acts as a variable resistor when the middle pin and any other pin is used.

ii. Pre-set resistor: This resistor has three terminals just like the one described above. It is called pre-set resistor because a precision instrument is used to vary the resistance.

iii. Light dependent resistor (LDR): This is a variable resistor whose resistance varies with the intensity of light. It has two terminals and the variation in resistance is affected by light falling on the surface.

iv. Digital potentiometer: This is a variable resistor whose resistance can be varied digitally either with a microcontroller or any other digital integrated circuit (IC).

v. Temperature dependent resistor (TDR): this is a variable resistor whose resistance varies with change in temperature. It has two terminals and the variation in resistance is affected by change in its temperature.

Figure 2.3: A VARIABLE RESISTOR

2.6.3 TRANSISTORS

A transistor is an active device with three terminals made from semiconductor materials. The three terminals are the base (B), emitter (E) and the collector (C). A transistor can be used for amplification, switching and Transistors are of two kinds namely:

1. Bipolar junction transistors (BJT): bipolar junction transistors are further classified into NPN and PNP.
2. Field effect transistors (FET): the FET transistor is a unipolar transistor which could either be an n-channel FET or p-channel FET. They are also of the kinds metal oxide semiconductor FET (MOSFET) and the insulated gate or junction FET (JFET).

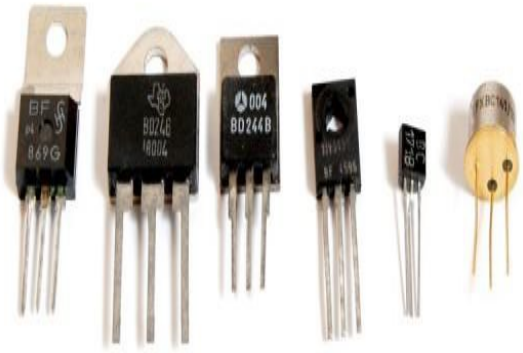

Figure 2.4: TRANSISTORS

2.6.4 CAPACITORS

A capacitor is a passive two terminal electrical component used to store electrical energy temporarily in an electric field. They are widely used in electronic circuits for blocking direct current while allowing alternating current to pass as in shunt capacitors and for blocking of alternating current to pass while blocking direct current as in the case of series capacitors. In electric power transmission systems, they stabilize voltage and power flow.

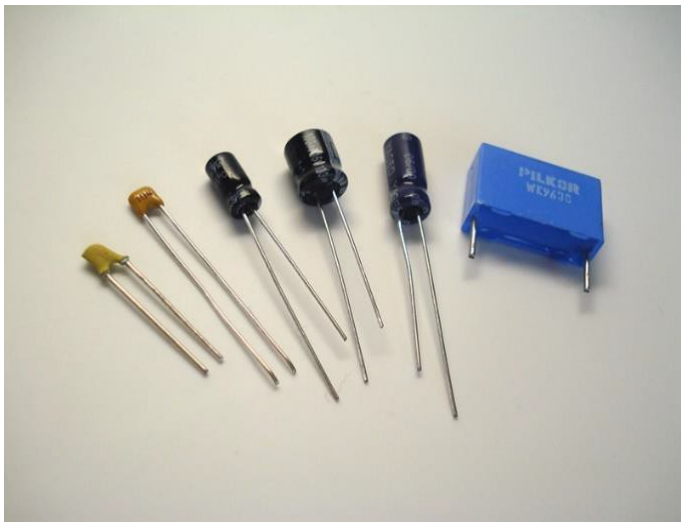

Figure 2.5: CAPACITORS

2.6.5 DIODES

Diode is a two terminal device made from semiconductor materials. The diode conducts in only one direction (i.e., current flows in only one direction and is blocked from flowing in the opposite direction).

Its forward resistance is quite low while its reserve resistance is high. The diode is a polarized component. Its positive terminal is called the anode while the negative terminal is known as the cathode. Diodes have numbers on their body which reflects the maximum voltage it can withstand as well as the current it can deliver. Usually, the diode has a silver strip on one end, and this indicates the negative terminal (the cathode). The side without the strip is the positive terminal (the anode).

There are various types of diodes which include:

- Zener diode
- Light emitting diode
- Tunnel diode
- Vacuum diode
- Crystal diode
- Photo diode *etc.*

Diodes were selected based on the requirement of our circuit. In the voltage stabilizer, light-emitting and the normal forward biased diodes find relevance.

Figure 2.6: DIODES

2.6.6 CRYSTAL OSCILLATOR

A crystal oscillator is an electric oscillator circuit that uses the mechanical resonance of a vibrating crystal of piezoelectric material to create an electric signal with a very precise frequency. The frequency is commonly used to keep track of time and to provide a stable clock signal for digital integrated circuits. It is used for providing clock signals to the microcontroller.

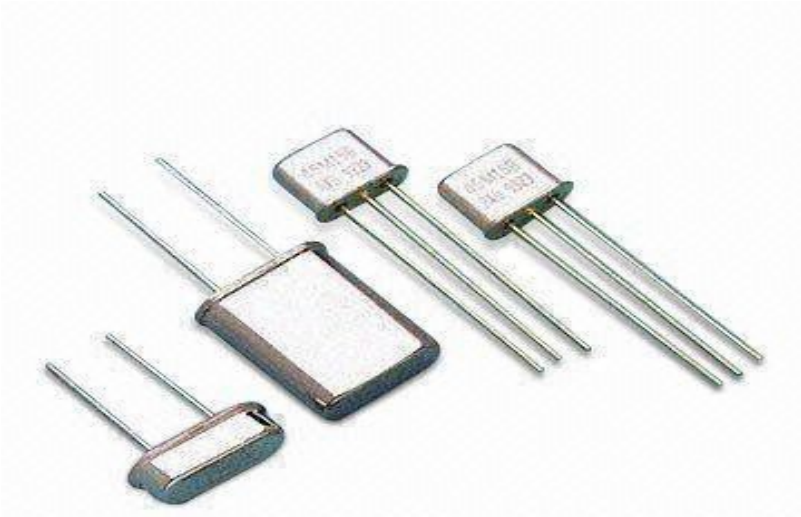

Figure 2.7: CRYSTAL OSCILLATORS

2.6.7 RELAYS

A relay is a switch that opens and closes to make or break a circuit. The opening and closing of the contact are affected by electronic or electromechanical means depending on the type of relay. There are two types of relays namely:

1. Solid state relay: This type of relay are pure electronic relays that have no contact. They are made from semiconductor materials and so consume less energy than electromechanical relays but are costlier. They don't make those clicking sounds associated with electromechanical relays. They make use of transistors, thyristors, or triacs in place of the moving parts. Opto-coupler (opto-isolator) does the electrical separation between the input control signal and the output load voltage.
2. Electromechanical relay: are those relays which operates on the principle of electromagnetic attraction. It is a type of a magnetic switch which uses the magnet for creating a magnetic field. The magnetic field uses for opening and closing the switch and performing the mechanical operation.

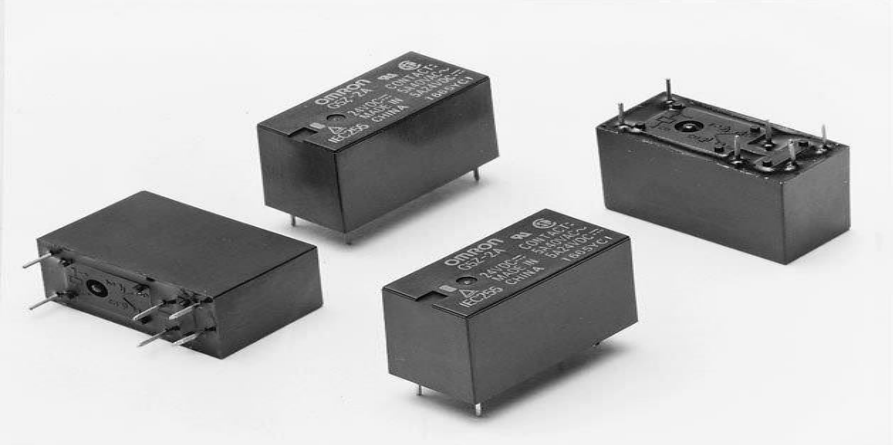

Figure 2.8: RELAYS

2.6.8 VOLTAGE REGULATOR

A voltage regulator is an electronic device that maintains the voltage of a power source within acceptable limits. It is needed to keep voltages within the prescribed range that can be tolerated by the electrical equipment using that voltage. In the internal circuitry, excessive variations in voltage would be detrimental because it could lead to the damage of the components of the circuit, so the voltage regulator is connected in the circuit to keep the voltage within the acceptable limits.

A voltage regulator may be a simple feed-forward design or may include a negative feedback control loop. It may be used to regulate one or more A.C. or D.C. voltages. Its application can be in power supply units of electronic circuits, automobile alternators, power stations etc.

Examples of some regulators are LM7805, LM7812, LM7912, LM7805 etc.

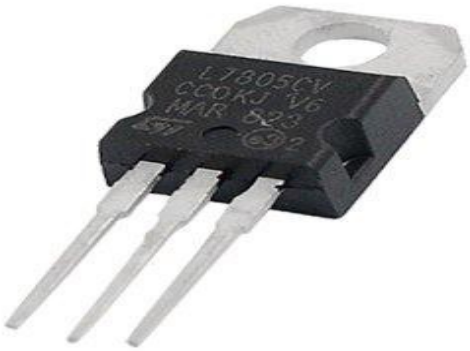

Figure 2.9: A VOLTAGE REGULATOR

2.6.9 A6 GSM/GPRS MODULE

The A6 GSM/GPRS cellular chip is from Ai-Thinker (Manufacturer of ESP8266 Wi-Fi modules). It communicates with a microcontroller over UART and supports baud rate from 1200bps to 115200bps with Auto-Baud detection. All the necessary data pins of A6 GSM chip are broken out to 0.1" pitch headers.

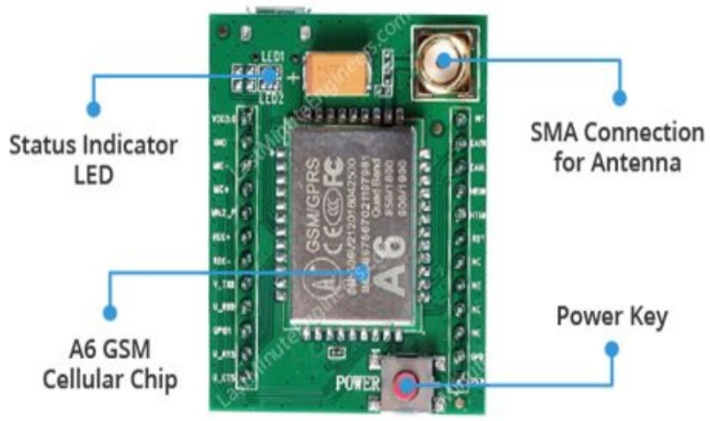

Figure 2.10: AN A6 MODULE FRONT VIEW

The module needs an external antenna for any kind of voice or data communications as well as some SIM commands. So, it usually comes with a small duck antenna having 2 dBi gain and 50Ω impedance which provides great coverage even if your project is indoors.

Though the module can work on 5V, the operating voltage of the chip is from 3.3V to 4.2V. To keep supply voltage safe at 4.1V, the module comes with a high frequency step-down switching regulator MP1584_from_Monolithic_Power_Systems – capable of handling load currents up to 3A.

The module can also be powered through a micro-USB connector. You can just grab your cell phone's wall charger (rated 5V 2A) and power up the module.

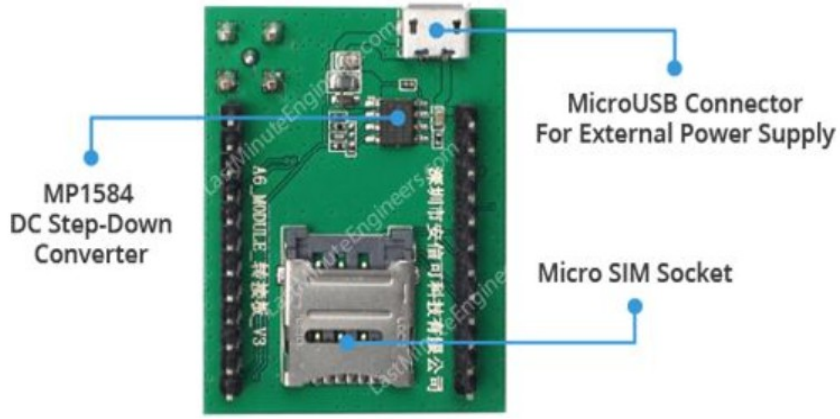

Figure 2.11: AN A6 MODULE BOTTOM VIEW

A6 GSM cellular chip measures less than the size of a postage stamp but packs a surprising number of features into its little frame. Some of them are listed below:

- Supports Quad-band: GSM850, EGSM900, DCS1800 and PCS1900
- Connect onto any global GSM network with any 2G SIM
- Make and receive voice calls using an external 8Ω speaker & electret microphone
- Facility to connect 4-pole TRRS mic and headset
- Send and receive Voice calls and SMS messages
- Class 10 GPRS with 85.6Kbps download speed and 42.8Kbps upload speed
- Consumes less than 3mA in standby mode
- 8V UART port level – compatible with Arduino, Raspberry-Pi
- Transmit Power:
- Class 4 (2W) for GSM850/EGSM900
- Class 1 (1W) for DCS1800/PCS1900
- Supports serial-based AT Command Set
- Accepts Micro SIM Card

2.6.10 TEMPERATURE SENSOR

A temperature sensor is an electronic device that measures the temperature of its environment and converts the input data into electronic data to record, monitor, or signal temperature changes. There are many different types of temperature sensors. Some temperature sensors require direct contact with the physical object that is being monitored (contact temperature sensor), while others indirectly measure the temperature of an object (non-contact temperature sensor).

Non-contact temperature sensors are usually infrared (IR) sensors. They remotely detect the IR energy emitted by an object and send a signal to a calibrated electronic circuit that determines the object's temperature.

Contact temperature sensors are thermocouples and thermistors. A thermocouple is comprised of two conductors, each made of a different type of metal, that are joined at an end to form a junction. When the junction is exposed to heat, a voltage is generated that directly corresponds to the thermoelectric effect. The other type of contact temperature sensor is called thermistors. In thermistors, resistance decreases as temperature increases. There are two main types of thermistors: Negative Temperature Coefficient (NLC) and Positive Temperature Coefficient (PTC). Thermistors are more precise than thermocouples (capable of measuring within 0.05-1.5 degree Celsius), and they are made of ceramics or polymers.

Temperature sensors are used in automobiles, medical devices, computers, cooking appliances and other types of machinery.

Figure 2.12: TEMPERATURE SENSOR

2.6.11 ELECTRONIC SWITCH

An electronic switch is an electronic component or device that can switch an electrical circuit, interrupting the current or diverting it from one conductor to another. Electronic switches are considered binary devices because they can be on or completely off.

Figure 2.13: ELECTRONIC SWITCH

2.6.12 LCD (LIQUID CRYSTAL DISPLAY)

Liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in colour or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden. For instance: pre-set words, digits, and seven-segment displays, as in a digital clock, are all good examples of devices with these displays. They use the same basic technology, except that arbitrary images are made from a matrix of small pixels, while other displays have larger elements. LCDs can either be normally on (positive) or off (negative), depending on the polarizer arrangement. For example, a character positive LCD with a backlight will have black lettering on a background that is the colour of the backlight, and a character negative LCD will have a black background with the letters being of the same colour as the backlight. Optical filters are added to white on blue LCDs to give them their characteristic appearance.

LCDs are used in a wide range of applications, including LCD televisions, computer monitors, instrument panels, aircraft cockpit displays, and indoor and outdoor signage. Small LCD screens are common in LCD projectors and portable consumer devices such as digital cameras, watches, digital clocks, calculators, and mobile telephones, including smartphones. LCD screens are also used on consumer electronics products such as DVD players, video game devices and clocks. LCD screens have replaced heavy, bulky cathode ray tube (CRT) displays in nearly all applications. LCD screens are available in a wider range of screen sizes than CRT and plasma display, with LCD screens available in sizes ranging from tiny digital watches to very large television receivers.

Figure 2.14: 2x16 LCD DISPLAY

2.6.13 ELECTRONIC CONNECTORS

An electrical connector is an electromechanical device used to join electrical conductors and create an electrical circuit. Most electrical connectors have a gender – i.e., the male component, called a plug, connects to the female component, or socket. The connection may be removable (as for portable equipment), require a tool for assembly and removal, or serve as a permanent electrical joint between two points. An adapter can be used to join dissimilar connectors.

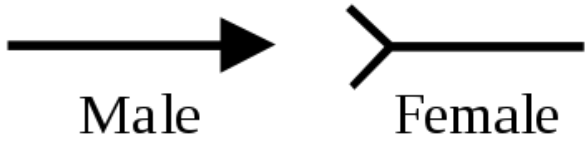

Figure 2:15: SCHEMATIC SYMBOLS FOR MALE AND FEMALE CONNECTORS

Thousands of configurations of connectors are manufactured for power, data, and audio-visual applications. Electrical connectors can be divided into four basic categories, differentiated by their function:

- inline or cable connectors permanently attached to a cable, so it can be plugged into another terminal (either a stationary instrument or another cable)
- Chassis or panel connectors permanently attached to a piece of equipment so users can connect a cable to a stationary device.
- PCB mount connectors soldered to a printed circuit board, providing a point for cable or wire attachment:56 (e.g., pin headers, screw terminals, board-to-board connectors)
- Splice or butt connectors (primarily insulation displacement connectors) that permanently join two lengths of wire or cable

In computing, electrical connectors are considered a physical interface and constitute part of the physical layer in the OSI model of networking.

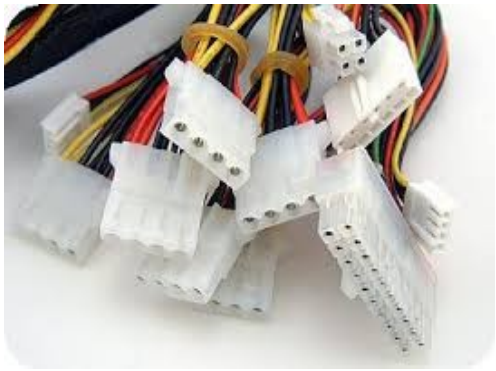

Figure 2.16: ELECTRONIC CONNECTORS

CHAPTER THREE

METHODOLOGY

The sections details materials and methods carried in achieving the aim of this project.

3.1 BLOCK DIAGRAM

The block diagram of The Battery charging, monitoring & control device is shown as follows.

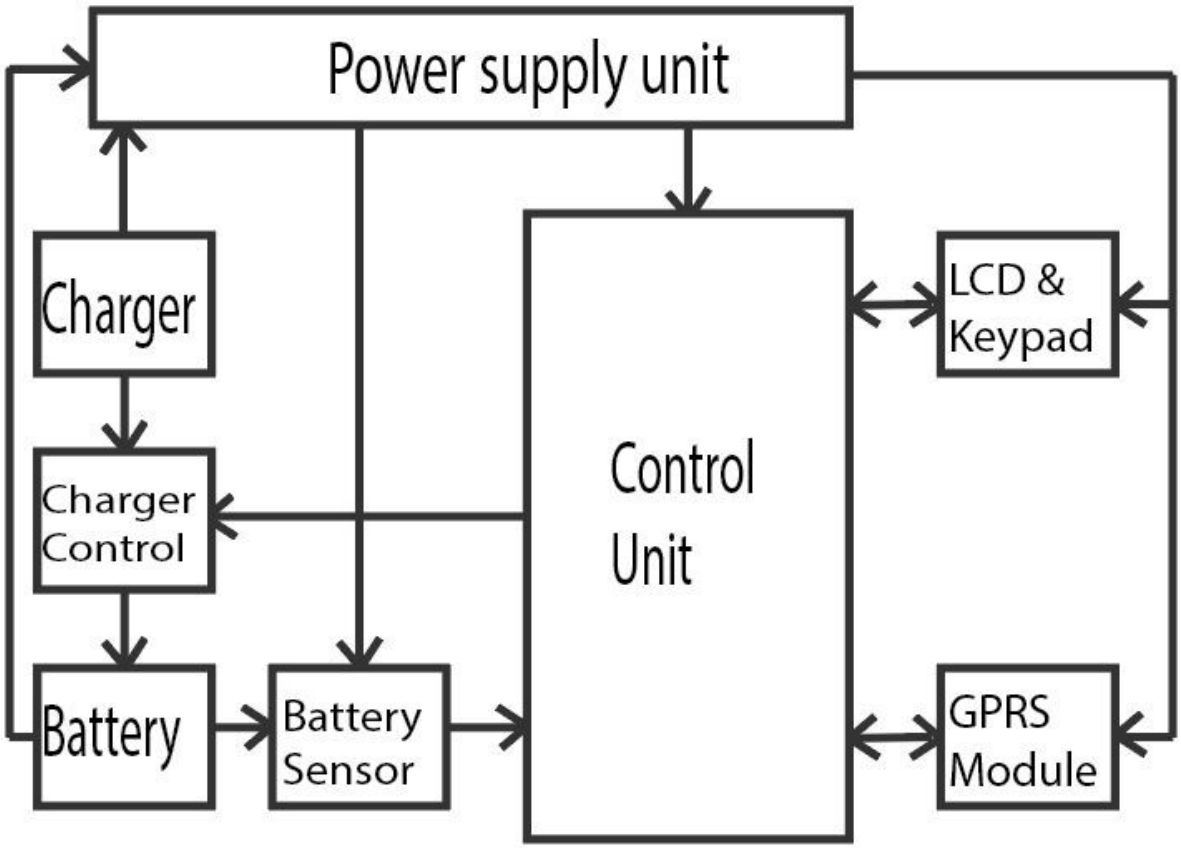

Figure 3.1: Block diagram of the battery charging, monitoring & control device

The complete project circuit diagram is divided into parts or building blocks. The blocks of the system are as listed below.

- Power Supply Unit - for dual voltage dc-dc power supply
- Control Unit – for controlling the various system peripherals. Comprises the micro-controller, crystal circuit and the master clear reset circuit,
- Battery Voltage Sensor Unit - for the battery's voltage instantaneous voltage.
- Charger Controller Unit - for prevents overcharging and may protect overvoltage.

- GPRS Module - for communicates with a microcontroller over UART and supports baud rate from 1200bps to 115200bps with Auto-Baud detection.
- LCD & Keypad - For monitoring and control.

3.2 DESIGN CALCULATIONS

Detailed design calculations for each unit is carried out in this section.

3.2.1 POWER SUPPLY

Specification: 5V, 4.2V 2Amps

The power supply is a dc-dc power supply with dual source and dual voltage outputs. The source of the power supply is battery and a battery charger but only a power source is used at a time. The switching of the power sources is done using a 12V relay. In order to stabilize the power going to the relay, a 12V voltage regulator (7805) was made use of, because the power sources can be greater than 12V. The output of the power supply is 5V and 4.2V with a combine current of two amps. Figure 3.2 is the circuit diagram of the power supply.

Diode D6 is meant to prevent reverse current flow.

Capacitors C5 and C6 are decoupling capacitors (100nF Recommended by 7805 manufacturer)

Capacitor C8 is meant for storing power which is meant for supplying power to the circuit during power source switching. Value of the capacitor can be from 470uF and above. 1000uF was chosen due to availability.

In the 4.2V circuit, Lm317 was made used of. Below is the formula for calculating resistor values used for biasing Lm317

Vout = Vref X (1 + (R18/R17)); .....Eq. 3.1

Vout = Output voltage = 4.2

Vref = Voltage reference = 1.25 (Stated in the datasheet of Lm317)

R = Resistor connecting Lm317 output pin to adjustable pin

R17 = 1k (Recommended value by Lm317 manufacturer in the datasheet)

R18 = ((Vout / Vref) – 1) X R17 .....Eq. 3.2

R18 = ((4.2 / 1.25) – 1) X 1000

R18 = 2360 = 2.36 X 10^3 = 2.36K

R18 = 2.2K (2.2K was chosen due to availability)

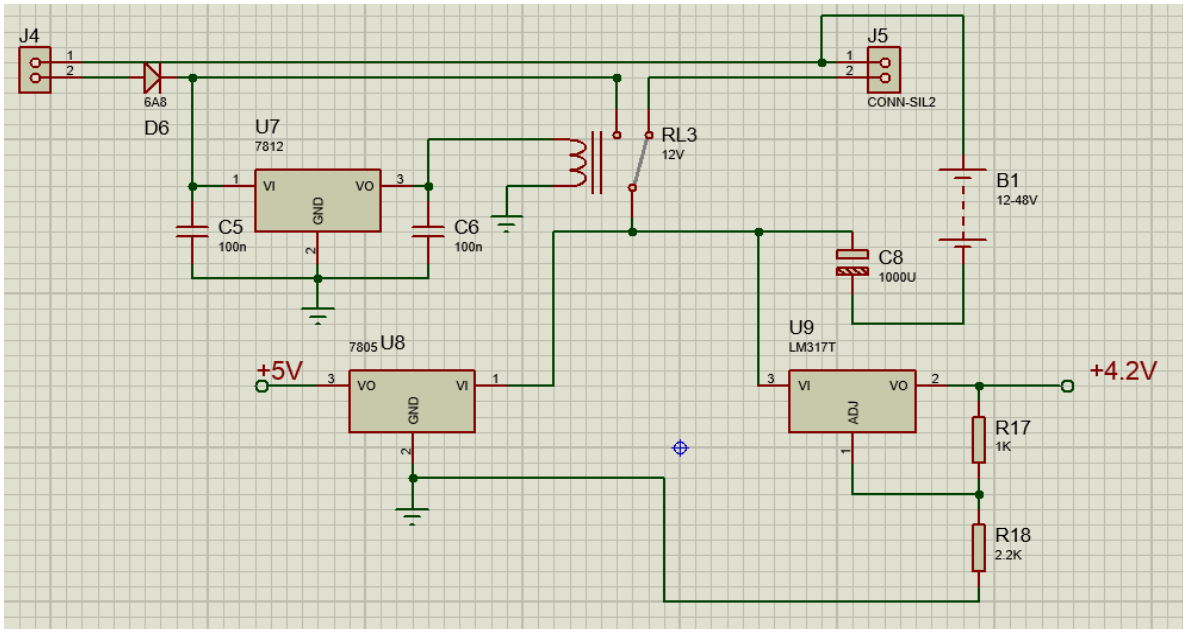

Figure 3.2: POWER SUPPLY

3.2.2 CONTROL UNIT

The control unit comprises the microcontroller (PIC16F887), crystal oscillator circuit and the master clear reset circuit which in turn consist of a resistor and a capacitor. The microcontroller is the main control unit which controls the GPRS module, gets data from the GPRS module and turn on and off the relay based on the data gotten from the GPRS module. The microcontroller equally takes data from the sensors and send them to the server through the GPRS module. The crystal oscillator controls the timing of the microcontroller which means the time at which the microcontroller executes one instruction. The microcontroller frequency is one-fourth of the crystal oscillator frequency. The master clear reset circuit prevents the microcontroller form resetting by connecting the master clear pin of the microcontroller to power through a resistor (R1) and a capacitor (C6) to ground. Fig 3.3 is the circuit diagram of the control unit. C4 and C5 are meant for stabilization of the crystal oscillator.

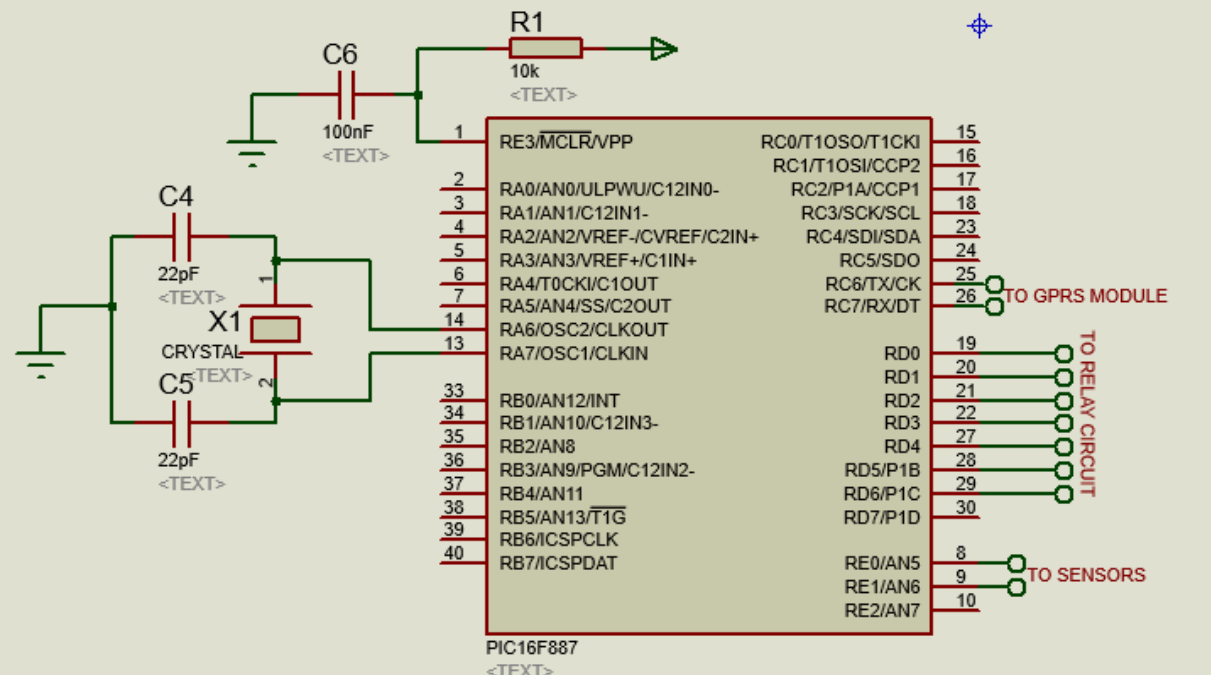

Figure 3.3: CONTROL UNIT

R1 = 10K (Recommended in PIC microcontroller datasheet)

C6 = 100nF (Recommended in PIC microcontroller datasheet)

C4 = C5 = 22pF (Recommended in PIC microcontroller datasheet)

3.2.3 BATTERY VOLTAGE SENSOR

The battery voltage sensor comprises solely of two-resistor voltage divider as shown in figure 3.3.

Vout = (R8 / (R9 + R8)) X Vbat; .....Eq. 3.3

Vbat = Maximum battery voltage = 14.4V

Vout = Maximum microcontroller sensing voltage = 5V

(R8 + R9) / R8 = Vbat / Vout = 14.4 / 5 = 2.88; .....Eq. 3.4

R8 + R9 = 2.88R8

R9 = 1.88R8

NB: Resistor connected to microcontroller pin and ground (R8) should be of high impedance greater than 1K (Stated in microcontroller datasheet).

R8 = 1K (Selected due to availability)

R9 = 1.88 X 1000 = 1.88K

R9 = 2.2k (2.2k was chosen due to availability)

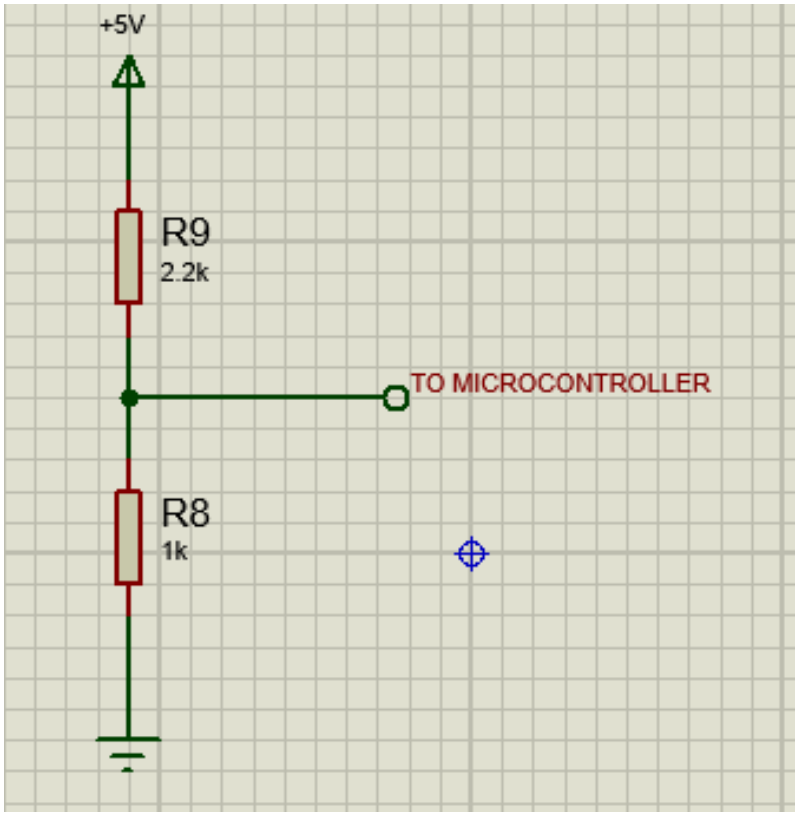

Figure 3.4: BATTERY VOLTAGE SENSOR

3.2.4 CHARGER CONTROLLER

The charger control circuit comprises of the charger voltage sensor and a switching circuit as shown in fig 3.4. The charger voltage sensor comprises of two-resistor voltage divider circuit.

Vout = (R24 / (R24 + R23)) X Vbat; .....Eq. 3.5

Vbat = Maximum charger voltage = 15V

Vout = Maximum microcontroller sensing voltage = 5V

(R24 + R23) / R24 = Vbat / Vout = 15/ 5 = 3 .....Eq. 3.6

R24 + R23 = 3R24

R23 = 2R24

NB: Resistor connected to microcontroller pin and ground (R24) should be of high impedance greater than 1K (Stated in microcontroller datasheet).

R24 = 1K (Selected due to availability)

R23 = 2 X 1000 = 2K

R23 = 2.2k (2.2k was chosen due to availability)

Relay Diode D7 is meant to protect the transistor from feedback that might emanate from switching off the relay because a relay armature is an inductive device.

Resistor R25 is a current limiting resistor to limit the current supplied to the transistor from the microcontroller.

R25 = (Vcc – Vc) / I

Vcc = Supply voltage = 5V

Vd = Voltage drop = 0.7(for silicon transistor)

I = Maximum transistor current = 10mA = 10^-2

R25 = (5 – 0.7) / 10^-2 = 4.3 X 100 = 430 (430 Ohms is the minimum resistor); .....Eq. 3.7

R25 = 1000 = 1K (1K was chosen due to availability).

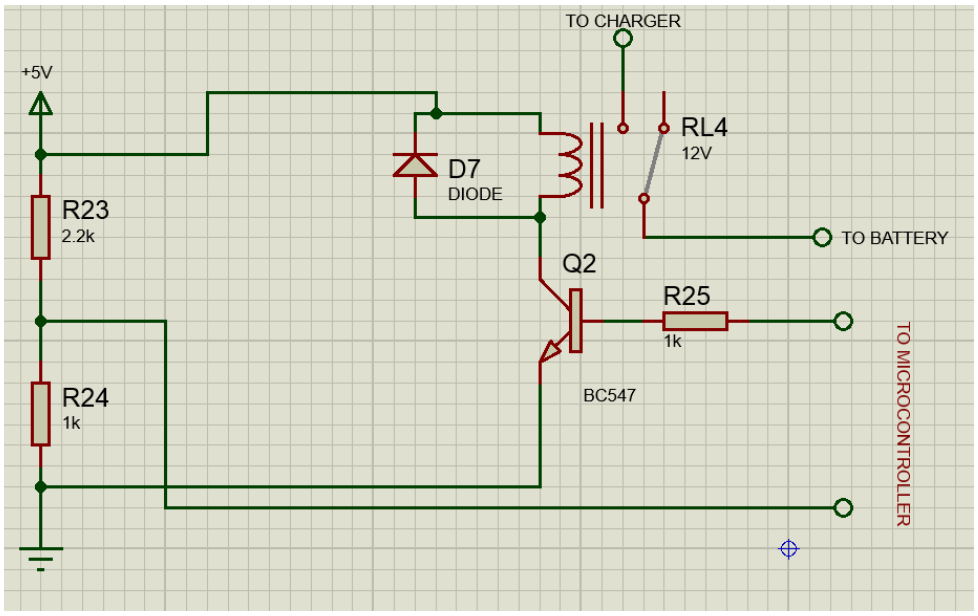

Figure 3.5: CHARGER CONTROLLER

3.2.5 GPRS MODULE

The A6 GSM/GPRS cellular chip is from Ai-Thinker (Manufacturer of ESP8266 WIFI modules). It communicates with a microcontroller over UART and supports baud rate from 1200bps to 115200bps with Auto-Baud detection. It is connected to the microcontroller using just two wires, one for receiving from the microcontroller and one for sending to the microcontroller. The GPRS module equally powered by the power supply. Below in fig 3.5 is the circuit diagram of the GPRS module.

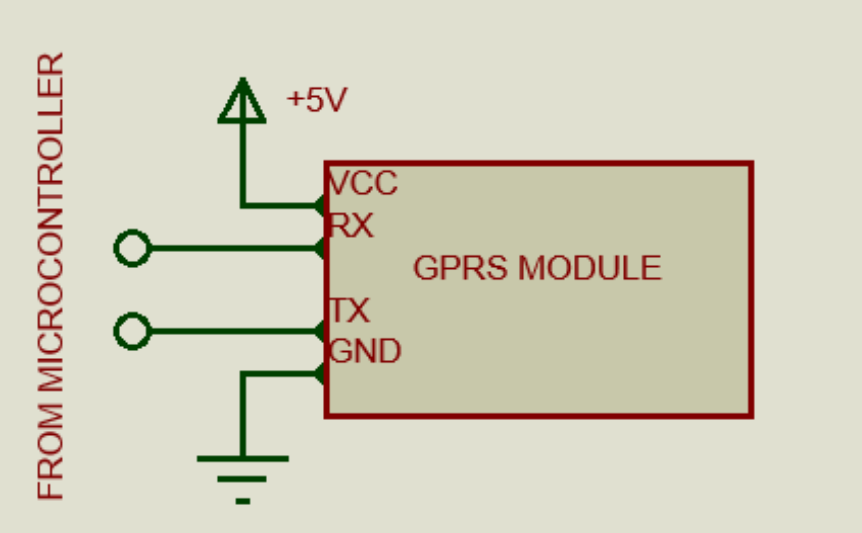

Figure 3.6: GPRS MODULE

3.2.6 BUTTON CIRCUIT

The button circuit is a pull-up button circuit which means the buttons are connected to power directly and connected to ground though a 10K resistor. On pressing any of the buttons, the corresponding pin are momentarily connected to power which is sensed by the microcontroller and an action will be carried based on the button pressed and according to the program running in the microcontroller. Below in fig 3.7 is the button circuit.

R19 = R20 = R21 = R22 = 10K (Recommended for pull-up button configuration in microcontroller datasheet)

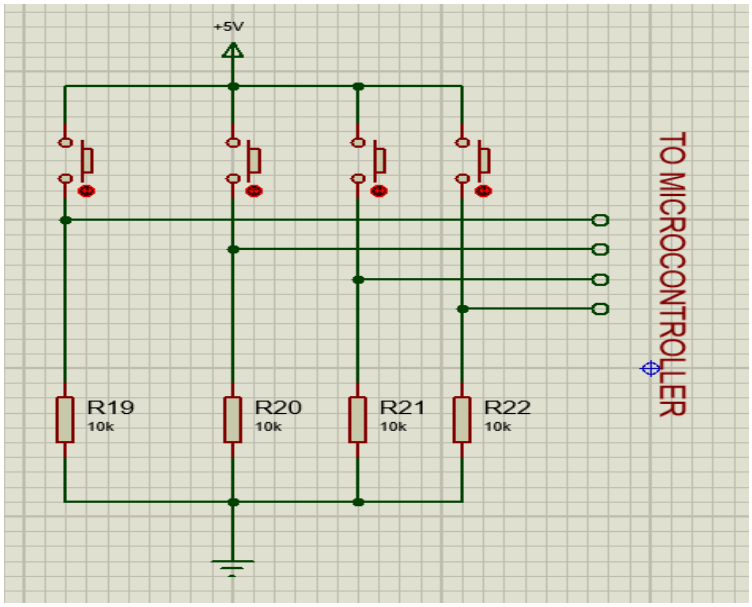

Figure 3.7 BUTTON CIRCUIT

3.2.7 DISPLAY CIRCUIT

The display circuit consist of the liquid crystal display (LCD) circuit and the indicator circuits which comprises two light emitting diodes each with a series resistor. Below in fig 3.8 is the display circuit and the potentiometer (RV1) connected to the LCD is meant to a control the contrast of the LCD.

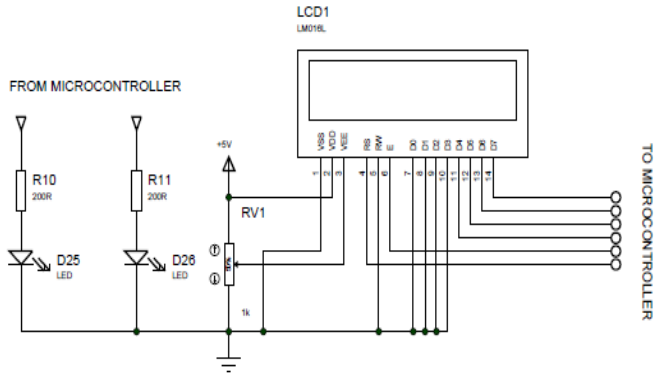

Figure 3.8 DISPLAY CIRCUIT

LED SERIES RESISTANCE CALCULATION

Voltage drop of Led (Vd) = 1.8V

Output of Microcontroller (Vs) = 5V

Forward current of the LED = 20Ma

LED series resistance = (Vs – Vd) * 1000/ 20 = 160 Ohms; .....Eq. 3.8

R10 = R11 = 200 Ohms

3.2.8 TEMPERATURE SENSOR

The voltage is the actual representation of the soc (state of charge) of the battery, but voltage reading can be inaccurate as temperature fluctuates, see fig 3.9 for a typical voltage to change in temperature relationship.

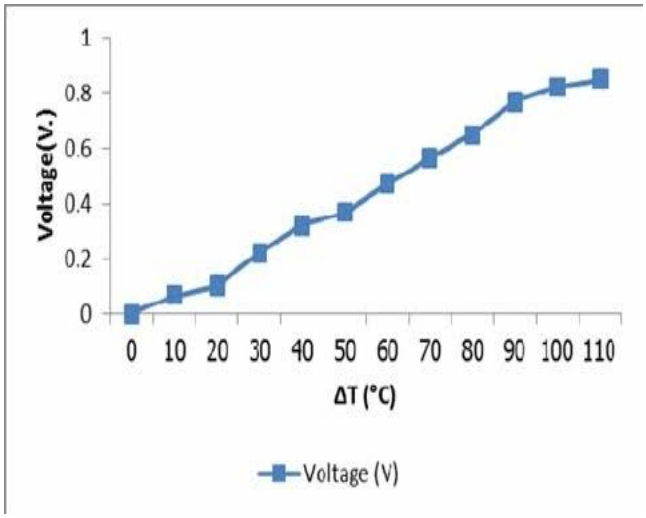

Figure 3.9: GRAPH RELATION BETWEEN VOLTAGE AND TEMPERATURE DIFFERENCE.

A temperature sensor is used to get the temperature of the battery so as to consciously calculate the actual reading of the battery state of charge. Using this relationship that 1'c rise in temperature will warrant a 0.0073 v drop in the actual measurement.

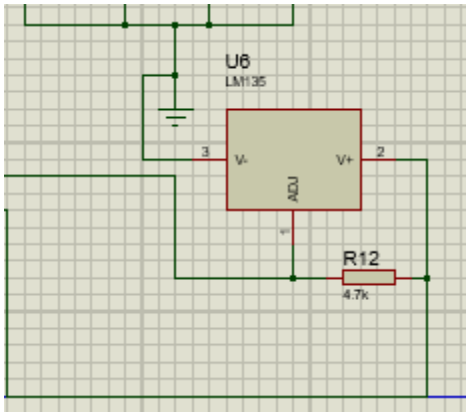

Figure 3.10: TEMPERATURE SENSOR CIRC

3.2.8 COMPLETE CIRCUIT DIAGRAM

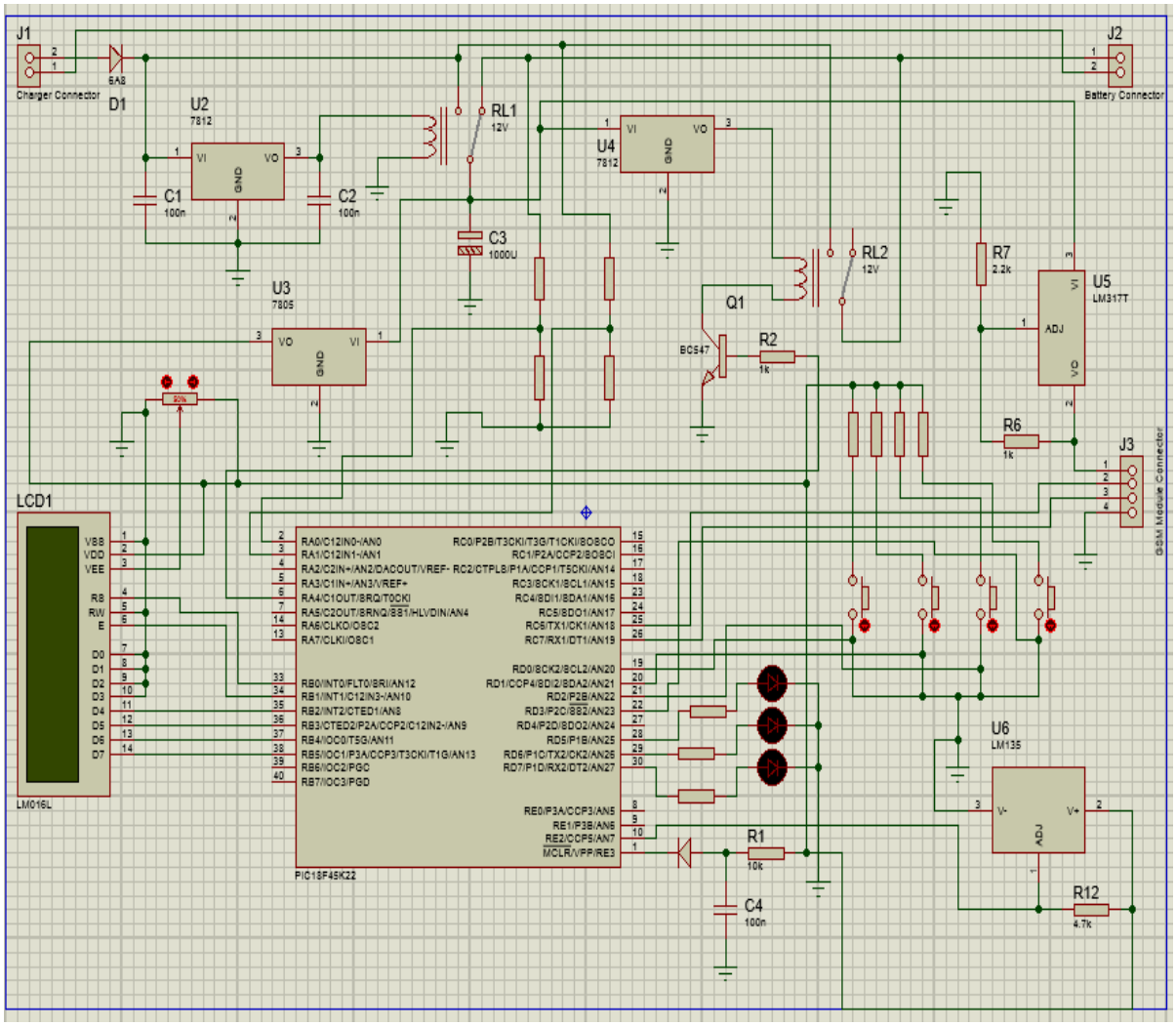

Figure 3.11: COMPLETE CIRCUIT DIAGRAM

CHAPTER 4

CONSTRUCTION, TESTING AND RESULTS

In this chapter, the construction procedure is discussed, the tools used are listed, the testing and results, precautions, and the bill of engineering materials.

4.1 CONSTRUCTION PROCEDURE

The project was constructed after a careful design of the circuit diagram and the necessary calculations has been carried out. The workability of all the component including integrated circuit, capacitance of the capacitors, resistance of the resistors, GSM/GPRS module, buttons and microcontroller were verified before they were all used in the project.

The project was partially simulated with Labcenter Proteus with the power supply circuit able to light up a Led in the simulation before the power supply and other circuitry such as the control unit, gsm/gprs module, battery & charger voltage sensor, and switching circuits were soldered on veroboard which form the main panel board. The main panel board is placed in a circuit box casing.

The main panel board was constructed by placing and soldering the power supply components first which include filtering capacitor, voltage regulators which was followed by soldering of the IC socket for the microcontroller, soldering of the crystal oscillator, soldering of the master clear circuit, soldering of the sinking transistor, soldering of the relays, soldering of indicator LED, placement of ON/OFF switch, and soldering of connecting wires for the relays.

The main panel, the indicator LED, liquid crystal display (LCD) and the ON/OFF switch were cased in a casing box as shown in fig 4.1.

Figure 4.1: IMAGE OF CONSTRUCTED CIRCUIT

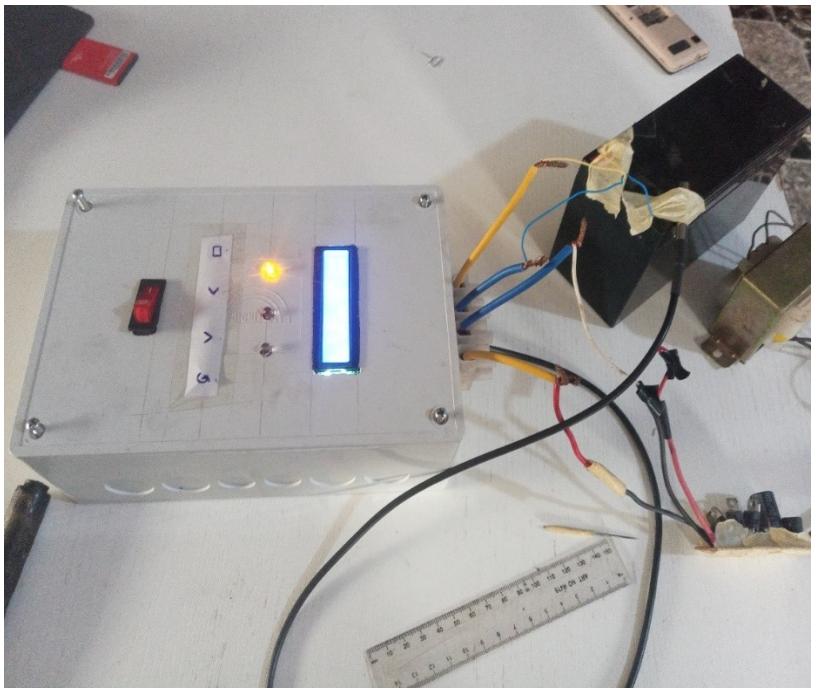

Figure 4.2: IMAGE OF PROJECT CONNECTED

4.2 TOOLS USED

The tools and materials used for the construction of the project and the casing of the project are as listed below

- Soldering Iron
- Lead
- Screwdriver
- Drilling machine
- Jig Saw
- Wire Cutter
- Multimeter
- Desoldering Pumb
- Pliers

4.3 TESTING

In the course of constructing the project, different test was carried out before everything were assembled together. Each of the various blocks in the block diagram was tested separately before they were all assembled together. The sequence of test undertook in order to ensure a successful project work are as follow

- Testing of the individual components
- Power supply test
- GPRS module test
- Switching circuit test
- System test

4.3.1 TESTING OF THE INDIVIDUAL COMPONENTS

Components were tested individually before fiddling with it so as to remove the bad ones. This test will be a satisfied test measures for individual components which are basically done by using of the multimeter (e.g., testing of transistor, diodes, LEDs, etc.) and also powering some of the components with a 5V power supply from battery to ascertain their workability.

4.3.2 POWER SUPPLY TEST

The power supply test was carried out after it was soldered on the veroboard and the input of the power supply was connected to a battery to determine if the voltage regulators 7805 & Lm317 produces 5V and 4.2V respectively which they did during the test.

4.3.3 GPRS MODULE TEST

The GPRS module test was carried out after its power pins have been connected to power and its communication pins connected to the microcontroller. The sim in the module was called to ascertain if the module has signal because GSM and GPRS uses the same signal. Subsequent test was carried out to determine if the GPRS module can communicate with the microcontroller which it did during the test.

4.3.4 SWITCHING CIRCUIT TEST

The switching circuit was tested after been soldered to the main panel board. Each actuator pins of the relay were connected to power and ground to determine if the relays will actuate which they did during the test. In addition, each relay was tested if it can be controlled by the microcontroller which was successful during the test.

4.3.5 SYTEM TESTING

This involves the testing of the entire circuitry and cross examining it for errors like short circuits, lead flux joining unwanted links, proper insertion IC pin layout and also checking of ICs and its pin numbers. After this check, the system was cross examined once again before powering the system.

4.4 PRECAUTION

In a project as complex and delicate as this a lot of precaution was observed.

- The circuit diagram was followed carefully so as not to make a mistake.
- The workplace was well ventilated and illuminated.
- Nose mask and safety glasses was put on when soldering, cutting and drilling.
- When dealing with sensitive components that heat could easily damage great care should be taken when soldering such components.
- Great care was taken to properly identify the cathode, anode, positive, negative and other terminals of the components to prevent damage by improper connectivity.
- While soldering, bridging and short circuiting was avoided on the board.
- The rating of all the components were carefully verified so as not to damage the sensitive components.
- Power to all parts of the circuit were properly connected to avoid damage.

4.5 MOBILE APPLICATION DEVELOPMENT AND WEB SERVER

The mobile application was developed in android studio and the development was done using Java. After development, the apk file was generated so that the app can be transferred from one person to another.

The website was developed in visual studio 2019 and the development was done using Asp.Net C# language. After development, it was subsequently uploaded to an asp.net server (IIS) which is batterymonitor.sightdev.net where it can be access from anywhere in the world. The web server solely comprises of two api endpoints which are endpoint for the hardware to communicate with and endpoint for the software with.

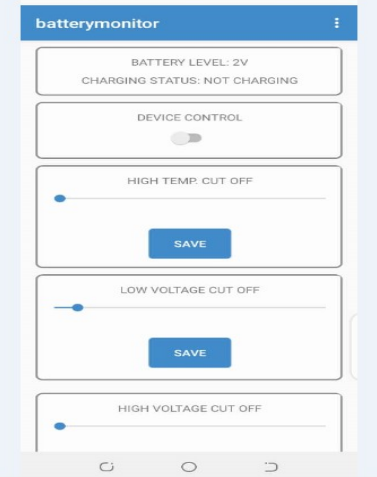

Figure 4.3: MOBILE APPLICATION INTERFACE

4.6 BILL OF ENGINEERING MEASUREMENT AND ESTIMATION

Table 4.1: BILL OF ENGINEERING MEASUREMENT AND EVALUATION AS AT JULY 2021.

S/NO	PARTICULAR	QUANTITY	RATE	AMOUNT (Naira)
1	Led	3	10	30.00
2	Vero Board	3	300	900.00
3	Adaptable Box	1	2000	2000.00
4	Jumper wire	1	400	400.00
5	Connector	1	500	500.00
6	Temperature sensor	1	1000	1000.00

7	Switch	1	200	200.00
8	PIC18F45K22 Microcontroller	1	5000	5000.00
9	Diode	4	20	80.00
10	Resistor	20	5	100.00
11	Capacitor	10	10	100.00
12	Relay 10amps, 12Volt	1	200	200.00
13	Voltage Regulator	5	100	500.00
14	Crystal Oscillator	1	300	300.00
15	IC socket	1	50	50.00
16	Ribbon Cable	4	300	1200.00
17	G.S.M Module	1	5000	5000.00
18	Bolt and Nut	18	10	180.00
19	LCD	1	2000	2000.00
20	Wire	1	500	500.00
21	Button	4	10	40.00
22	Zener diode	2	50	100.00
23	10amp diode	2	200	400.00
24	Capacitor 50V, 1000uf	2	500	1000.00
25	Relay 30amps 12Volt	3	500	1500

Total Calculation Table

S/NO	**PARTICULAR**	**AMONUT (Naira)**
1	Initial Total	21780.00
2	Miscellaneous	7000.00
3	Grand Total	28780.00

CHAPTER FIVE

RECOMMENDATION AND CONCLUSION

5.1 CONCLUSION

The result obtained from testing the battery charging & control device using internet of things project shows that the project worked satisfactorily by connecting it to a battery and controlling and monitoring its state using the mobile application through the internet.

The designed system measures the voltage level of the battery connected to it constantly and perform different functions on the battery such as low voltage cut-off, high temperature cut-off, etc. as set by the user. The user can control the device configuration both manually and through a mobile application via the internet.

The GSM module and the website interface were successfully tested from different locations, thus proving its portability and wide compatibility. Thus, a battery charging & control device using internet of things was successfully designed, implemented and tested.

5.2 PROBLEMS ENCOUNTERED

The major problems associated with this design are these

- During designing, the exact rating of the components was calculated for were not always seen. So, equivalents mostly used.
- Dealing with sensitive integrated circuits, great damage was done on some of them. The problem was solved when IC holders were used.
- Power problem: if the battery was low but not totally dead, the battery will be powering the device, we had to give priority to the charger whenever it is connected.
- We noticed that when measuring the battery voltage level, the charger gets in the way, so we had to add a diode connect in this format. Charger +ve was connected to diode +ve while diode -ve to battery +ve in a forward bias setting so as to prevent interruption of measurement, the diode also prevents a situation if the charger was lower than the battery, then the battery will be charging the charger situation.
- Server problem: when server sends data to the device after being requested the server sent data that are too large, so we needed to truncate the unnecessary data so as to preserve the necessary data.
- We also had issues with using a fast microcontroller as we needed real time response.

5.3 RECOMMENDATIONS

- Using this system as framework, the system can be expanded to include various other options which could include monitoring and management of different energy sources in a home or offices and monitoring and control of equipment, devices or machines.
- 3G module or a 4G module should be used for faster internet communication.
- This system can be incorporated to PWM/MPPT charger in order to make them smarter.
- The system can be expanded for energy monitoring, or weather stations. This kind of a system with respective changes can be implemented in the hospitals for disable people or in industries where human invasion is impossible or dangerous, and it can also be implemented for environmental monitoring.

REFERENCE

1. Aburukba, Raafat; Al-Ali, A. R.; Kandil, Nourhan; AbuDamis, Diala (2016). Configurable ZigBee-based control system for people with multiple disabilities in smart homes. pp. 1–5.

2. Barsukov, Yevgen; Qian, Jinrong (2013). Battery Power Management for Portable Devices. ISBN 9781608074914.

3. Brown, Eric (2016). "21 Open Source Projects for IoT". Linux.com.

4. Demiris, G; Hensel, K (2008). "Technologies for an Aging Society: A Systematic Review of 'Smart Home' Applications". IMIA Yearbook of Medical Informatics 2008. 17: 33–40. doi:10.1055/s-0038-1638580. PMID 18660873. S2CID 7244183.

5. Greenberg Andy (2015) Hackers Remotely Kill a Jeep on the Highway—With Me in It. Microchip, PIC16F887 Datasheet.

6. Greengard, Samuel (2015). The Internet of Things. Cambridge, MA: MIT Press. p. 90. ISBN9780262527736.

7. Jussi Karlgren; Lennart Fahlen; Anders Wallberg; Par Hansson; Olov Stahl; Jonas Soderberg; Karl-Petter Akesson (2008). Socially Intelligent interfaces for Increased energy awareness in the Home. The internet of Things. Lecture Notes in Computer Science. 4952. Springer. Pp. 263-275. ISBN19783540787303

8. Liu, Huaqiang; Wei, Zhongbao; He, Weidong; Zhao, Jiyun (2017). "Thermal issues about Li-ion batteries and recent progress in battery thermal management systems: A review". Energy Conversion and Management. 150: 304–330. doi:10.1016/j.enconman.2017.08.016. ISSN 0196-8904.

9. Mattern, Friedemann; Floerkemeier, Christian (2010). "From the Internet of Computer to the Internet of Things" (PDF). Informatik-Spektrum. 33 (2): 107–121. Bibcode:2009InfSp..32..496H. doi:10.1007/s00287-010-0417-7. hdl:20.500.11850/159645. S2CID 29563772.

10. Mulvenna, Maurice; Hutton, Anton; Martin, Suzanne; Todd, Stephen; Bond, Raymond; Moorhead, Anne (2017). "Views of Caregivers on the Ethics of Assistive Technology Used

for Home Surveillance of People Living with Dementia". Neuroethics. 10 (2): 255–266. doi:10.1007/s12152-017-9305-z. PMC 5486509. PMID 28725288.

11. Perera, C.; Liu, C. H.; Jayawardena, S. (2015). "The Emerging Internet of Things Marketplace from an Industrial Perspective: A Survey". IEEE Transactions on Emerging Topics in Computing. 3 (4): 585–598. arXiv:1502.00134. Bibcode:2015arXiv150200134P. doi:10.1109/TETC.2015.2390034. ISSN 2168-6750. S2CID 7329149.

12. Raji, R.S. (1994). "Smart networks for control". IEEE Spectrum. 31 (6): 49–55. doi:10.1109/6.284793. S2CID 42364553.

13. Rouse, Margaret (2019). "internet of things (IoT)". IOT Agenda. Retrieved 14 August 2019.

14. Stallings, William (2016). Foundations of modern networking : SDN, NFV, QoE, IoT, and Cloud. Florence Agboma, Sofiene Jelassi. Indianapolis, Indiana. ISBN 978-0-13-417547-8. OCLC 927715441.

15. Steinberg Joseph (2014). These Devices May Be Spying on You (Even in Your Own Home). Forbes.

16. Verbeek Peter-Paul (2011). Moralizing Technology: Understanding and Designing the Morality of Things. Chicago: The University of Chicago Press.

17. Weiser, Mark (1991). "The Computer for the 21st Century" (PDF). Scientific American. 265 (3): 94-104. Bibcode:1991SciAm.265c..94W. doi:10.1038/scientificamerican0991-94. Archived from the original (PDF) on 11 March 2015. Retrieved 5 November 2014

APPENDIX

APPENDIX A: MICROCONTROLLER CODE

(Written in c programming language in mikroC IDE)

```
#define RS              RB0_bit
#define EN              RB1_bit
#define WorkingLED      RD5_bit
#define ChargingLED     RD6_bit
#define LowLED          RD7_bit
#define Enter           RD2_bit
#define Right           RD3_bit
#define Up              RD1_bit
#define Back            RD0_bit
#define cRly            RA4_bit
const unsigned short TEMP_RESOLUTION = 9;
char *text = "000.0000";
char *full = "00.00";
char *low = "00.00";
char *cutTemp = "000.00";
char resCnt = 0;
int k;
unsigned temp;
char menu = 0;
char menu2 = 0;
char Temp2;
long per = 0;
long newPer = 0;
long holdTemp = 0;
char curPos = 0;
char curPos5 = 0;
long fullSet = 0;
long lowSet = 0;
long holdbat = 0;
long batVolt = 0;
long newBatVolt = 0;
long chargerVolt = 0;
long newChargerVolt = 0;
long lBat = 0;
long batLowStart = 0;
long batLowStart2 = 0;
long batLowStart3 = 0;
long cTemp = 0;
long soc = 0;
long newSoc = 0;
long tempCutOff = 7200;
```

```
long tempCutOff3 = 0;
long tempCutOff2 = 0;
long fBat = 0;
long batFullCutOff = 0;
long batFullCutOff2 = 0;
long batFullCutOff3 = 0;
long batNominalVol = 0;
long batN = 0;
char batFull = 0;
char fullCutValue = 48+0 ;
char batLow = 0;
char lowStartValue = 48+0;
long batTemp = 0;
char tempSetStatus = 48+0;
char fullCutValue2 = 48+0;
char lowStartValue2 = 48+0;
char fullCutValue3 = 48+0;
char lowStartValue3 = 48+0;
char validBatteryStatus = 48+1;
char newCnt = 0;
char newCnt2 = 0;
char sendStatus = 48+0;
char instantSwitch = 48+0;
char chargeCnt = 0;
long totalBat = 0;
long totalCharger = 0;
int m;int q;
long hBatVolt = 0;
long hChargerVolt = 0;
char sendType = 0;
char sentVal = 0;
char sVal = 0;
long bFull1 = 0;long bFull2 = 0; long bLow1 = 0; long bLow2 = 0; long bTemp1 = 0; long bTemp2 = 0;
long actualLow = 0;
long actualFull = 0;
/*
  address 0 for remoteStatus;address 1 for fullCutValue1; address 2-5 for batFullCutOff; address 6 for lowStartValue; address 7-10 for batLowStart;
  address 11 for tempSetStatus; address 12-16 for tempCutOff; address 17 for sendType; address 18 for sentVal ;  address 19 for sVal;
  address 20 for fullCutValue2;address 21 for fullCutValue3; address 22 for lowStartValue2; address 23 for lowStartValue3;
  char who; 0 for hardware value; and 1 for server value;
*/

// Added initialization
char isSet = 0;long intCnt12 = 0;char count = 0; char cont = 0; char timeOut = 200;
char msgReady = 0; char intCnt = 0;char count2 = 0; char cont2 = 0; char timeOut2 = 30;
```

```
char msgReady2 = 0; char intCnt2 = 0;char timeCnt = 0;
char *txt = "00000000000000000000000000000000";
char *txt2 = "0000000000000000000000000000000000000000000000000000000000000000";
char *testConnect = "CONNECT";
long testPos = 0;
char txt8[300];
long strlength;long strLength2;
char isStrMatch;  long i; long j;
char strlen1 = 0; char strLen2 = 0;
long myStrLen = 0;
long charPos = 0;
//GSM module configuration
const char *server = "batterymonitor.sightdev.net";
char webData[]  = "\"1|0|0|0000|0000|000\""; //  "Sample of the data sent to the server";
char webDatb[]  = "\"0|0000|0000|00000\""; //  "Sample of the data sent to the server";
char webData2[] = "0|0000|0000|00000";
const char *uri = "/api/battmon";
const char *postRequest1 = "POST /api/battmon HTTP/1.1\r\nHost: batterymonitor.sightdev.net\r\nContent-Length: ";
const char *postRequest2 = "\r\nContent-Type: application/json\r\n\r\n";
char dataLen; char mainDataLen; char iNum = 0;
long readADC2();
void LcdCmd2(const char CMD);
void LcdData2(const char DATA);
void firstDisplay2();
long calBat2();
long calCharger2();
void interrupt(){
   if(PIR2.TMR3IF){
      newCnt2++;
      newCnt++;
     if(newCnt == 100){
       batVolt = calBat2();
       if(cRly){
         if(batVolt > 0 && batVolt < batFullCutOff){
            sendStatus = 48+1;
         }
         else if(batVolt > batFullCutOff){
           sendStatus  = 48+0;
         }
       }
       else{
          if(batVolt > 0 && batVolt < batFullCutOff2){
            sendStatus = 48+1;
         }
         else if(batVolt > batFullCutOff2){
```

```
      sendStatus = 48+0;
    }
  }
}
else{
  if(newCnt2 == 50){
    WorkingLED = 1;
    batVolt = calBat2();
    chargerVolt = calCharger2();
    if(menu == 0){
      //firstDisplay2();
    }
    if(instantSwitch -48 == 0){
      if(cRly){
        if(batVolt > 0 && batVolt >= batFullCutOff){
          cRly = 0;
          ChargingLED = 0;
          LowLED = 0;
        }
        else if(batVolt > 0 && batVolt < batFullCutOff){
         ChargingLED = 1;
         cRly = 1;
         LowLED = 0;
        }
      }
      else{
        if(batVolt > 0 && batVolt >= batFullCutOff2){
          cRly = 0;
          ChargingLED = 0;
          LowLED = 0;
        }
        else if(batVolt > 0 && batVolt <= batLowStart){
          cRly = 1;
          ChargingLED = 1;
          LowLED = 0;
        }
        else if(batVolt > 0 && batVolt > batLowStart && batVolt < batFullCutOff2){
          ChargingLED = 0;
        }
      }
    }
    else{
      cRly = 0;
      ChargingLED = 0;
      if(batVolt > 0 && batVolt <= batLowStart){
```

```
            LowLED = 1;
          }
          else if(batVolt > 0 && batVolt  > batLowStart){
            LowLED = 0;
          }
        }
        newCnt2 = 0;
      }
    }
  TMR3H = 0B00001011;
  TMR3L = 0B11011100; //Re-Assign initial value
  PIR2.TMR3IF = 0;
  }
  if(INTCON.TMR0IF){
    count++;
   if(count >= timeOut){
          cont = 200;
          count = 0;
          T0CON.TMR0ON = 0;
          //T1CON.TMR1ON = 0;
      if(timeCnt == 1){
        msgReady = 1;
        //intCnt = 0;
      }
      else if(timeCnt == 2){
        timeCnt = 1;
        intCnt = 0;
      }
    }
   INTCON.TMR0IF = 0;  //Reset interrupt flag
   T0CON.TMR0ON = 1; //Enable Timer 0 interrupt
}
  if(PIR1.F5){
    // RD4_bit = 1;
     if(RCSTA.OERR){
       RCSTA.CREN = 0;
       RCSTA.CREN = 1;
     }
     if(timeCnt == 0){
       if(intCnt < 32){
          txt[intCnt] = RCREG;
       }
       else{
          txt[32] = RCREG;
       }
```

```
      }
       else if(timeCnt == 1){
         if(intCnt <=30){
          txt2[intCnt] = RCREG;
          intCnt++;
         }
         else{
           txt2[31] = RCREG;
           //intCnt = 0;
         }
      }
      else if(timeCnt == 2){
          if(intCnt <=9 ){
           txt[intCnt] = RCREG;
           intCnt++;
           if(intCnt == 9){
             T0CON.TMR0ON = 1;
            //T1CON.TMR1ON = 1;
           }
         }
         else {
           txt[intCnt] = RCREG;
         }
      }
      else if(timeCnt == 3){
          if(intCnt < 220 ){
            intCnt++;
            txt2[0] = RCREG;
          }
          else{
            txt8[intCnt12] = RCREG;
            intCnt12++;
            RD4_bit = 0;
          }
      }
      //PIR1.F5 = 0;
      count = 0;
   }
}
void LcdCmd(const char CMD)
{
   Temp2 = (CMD >> 2) & 0X3C;
   LATB = Temp2;
   RS = 0;
   EN = 1;
```

```
  Delay_ms(5);
  EN = 0;
  Temp2 = CMD << 2;
  LATB = Temp2 & 0X3C;
  RS = 0;
  EN = 1;
  Delay_ms(5);
  EN = 0;
}
void LcdCmd2(const char CMD)
{
  Temp2 = (CMD >> 2) & 0X3C;
  LATB = Temp2;
  RS = 0;
  EN = 1;
  Delay_ms(5);
  EN = 0;
  Temp2 = CMD << 2;
  LATB = Temp2 & 0X3C;
  RS = 0;
  EN = 1;
  Delay_ms(5);
  EN = 0;
}
void LcdData(const char DATA)
{
  Temp2 = (DATA >> 2) & 0X3C;
  LATB = Temp2;
  RS = 1;
  EN = 1;
  Delay_ms(5);
  EN = 0;
  Temp2 = DATA << 2;
  LATB = Temp2 & 0X3C;
  RS = 1;
  EN = 1;
  Delay_ms(5);
  EN = 0;
}
void LcdData2(const char DATA)
{
  Temp2 = (DATA >> 2) & 0X3C;
  LATB = Temp2;
  RS = 1;
  EN = 1;
```

```
  Delay_ms(5);
  EN = 0;
  Temp2 = DATA << 2;
  LATB = Temp2 & 0X3C;
  RS = 1;
  EN = 1;
  Delay_ms(5);
  EN = 0;
}
void LCD_WRITE(char *str)
{
 while(*str)
 {
    LcdData(*str++);
 }
}
void LCD_WRITE2(char *str)
{
 while(*str)
 {
    LcdData2(*str++);
 }
}
 void LCD_Setup()
{
   LcdCmd(0x02);//
   LcdCmd(0x28);// Set The LCD in 2 Line Mode and 4 bit connection mode
   LcdCmd(0x0E);// Enable Display
   LcdCmd(0x01);// Clear LCD display
   LcdCmd(0x06);// Entry Mode
   LcdCmd(0x80);//
   LcdCmd(0x0C);
}
void eepromWrite(unsigned short addr, unsigned short Content){
  while(EECON1.WR); //Wait till write is complete
  EEADR = addr;
  EEDATA = Content;
  EECON1.EEPGD = 0;
  EECON1.CFGS = 0;
  EECON1.WREN = 1;
  INTCON.GIE = 0;
  EECON2 = 0X55;
  EECON2 = 0XAA;
  EECON1.WR = 1;
  INTCON.GIE = 1;
```

```
  while(EECON1.WR); //Wait till write is complete
  EECON1.WREN = 0;
}
unsigned short eepromRead(unsigned short addr){
  while(EECON1.WR || EECON1.RD); //Wait till a read or write is completed
  EEADR = addr;
  EECON1.EEPGD = 0;
  EECON1.CFGS = 0;
  EECON1.RD = 1;
  return(EEDATA);
}
void UartWrite(char content){
  while(!TXSTA.TRMT);
  TXREG = content;
}
void UartWriteText(const char *text){
  for(i=0; text[i] != '\0'; i++){
    UartWrite(text[i]);
  }
}
void UartWriteText2(char *text){
  for(i=0; text[i] != '\0'; i++){
    UartWrite(text[i]);
  }
}
long txtLen(const char *testTxt){
   myStrLen = 0;
   while(*testTxt){
     *testTxt++;
     myStrLen++;
   }
   return myStrLen;
}
long txtLen2(char *testTxt){
   myStrLen = 0;
   while(*testTxt){
     *testTxt++;
     myStrLen++;
   }
   return myStrLen;
}
char txtCmp(const char *firstVal, const char *secVal){
  strLength = 0; strLength2 = 0;
  strLength = txtLen(firstVal);
  strLength2 = txtLen(secVal);
```

```
  if(strLength != strLength2){
    return 0;
  }
  else {
    for(i = 0; i < strLength; i++){
      if(firstVal[i] != secVal[i]){
        return 0;
      }
    }
  }
  return 1;
}
char txtCmp2(char *firstVal,char *secVal){
  strLength = 0; strLength2 = 0;
  strLength = txtLen2(firstVal);
  strLength2 = txtLen2(secVal);
  if(strLength != strLength2){
    return 0;
  }
  else {
    for(i = 0; i < strLength; i++){
      if(firstVal[i] != secVal[i]){
        return 0;
      }
    }
  }
  return 1;
}
long txtFind(char *textMain, char *testVal){
  strLength = 0; strLength2 = 0;
  strLength = txtLen2(textMain);
  strLength2 = txtLen2(testVal);
  for(i=0; i <= strLength; i++){
    if(textMain[i] == testVal[0]){
      for(j = 1; j < strLength2; j++){
        i++;
        if(textMain[i] != testVal[j]){
          isStrMatch = 0;
          i--;
          break;
        }
        else{
          isStrMatch = 1;
        }
      }
```

```
        if(isStrMatch == 1){
          return i;
        }
      }
    }
    return 0;
}
long readADC(){
    Delay_us(50);
    ADCON0.F1 = 1;
    while(ADCON0.F1);
    return (((long)((ADRESH << 8) + ADRESL)*5000)/1023);
}
long readADC2(){
    Delay_us(50);
    ADCON0.F1 = 1;
    while(ADCON0.F1);
    return (((long)((ADRESH << 8) + ADRESL)*5000)/1023);
}
long calBat(){
    ADCON0.CHS0 = 0;
    for(m = 0; m < 3;m++){
      newBatVolt = readADC() * 1.2;
      totalBat += newBatVolt;
    }
    hBatVolt = totalBat/3;
    totalBat = 0;
    return ((long)(hBatVolt));
}
long calBat2(){
    ADCON0.CHS0 = 0;
    for(m = 0; m < 3;m++){
      newBatVolt = readADC2() * 1.2;
      totalBat += newBatVolt;
    }
    hBatVolt = totalBat/3;
    totalBat = 0;
    return ((long)(hBatVolt));
}
long calCharger(){
    ADCON0.CHS0 = 1;
    for(q = 0; q < 3;q++){
      newChargerVolt = readADC() * 1.2;
      totalCharger += newChargerVolt;
    }
```

```
    hChargerVolt = totalCharger/3;
    totalCharger = 0;
    return ((long)(hChargerVolt));
}
long calCharger2(){
    ADCON0.CHS0 = 1;
    for(q = 0; q < 3;q++){
      newChargerVolt = readADC2() * 1.2;
      totalCharger += newChargerVolt;
    }
    hChargerVolt = totalCharger/3;
    totalCharger=0;
    return ((long)(hChargerVolt));
}
void clearLCD(){
    LcdCmd(0x01);
}
void btnDly(){
    Delay_ms(150);
}
void moduleInit() {
    intCnt = 0; cont = 0; intCnt2 = 0; cont2 = 0;
    Delay_ms(500);
    intCnt = 0; cont = 0; intCnt2 = 0; cont2 = 0;
    UART1_Write_Text("AT"); // Disable module echo
    UART1_Write(0x0D);
    Delay_ms(500);
    T0CON.TMR0ON = 1;
    while(cont != 200); //Wait until total reception of character
    timeout = 50;
    Delay_ms(1000);
    intCnt = 0; cont = 0; intCnt2 = 0; cont2 = 0;
    UART1_Write_Text("ATE0\r\n"); // Disable module echo
    Delay_ms(1000);
    T0CON.TMR0ON = 1;
    while(cont != 200); //Wait until total reception of character
    intCnt = 0; cont = 0; intCnt2 = 0; cont2 = 0;
    timeout = 50;
    intCnt = 0; cont = 0; intCnt2 = 0; cont2 = 0;
    T0CON.TMR0ON = 1;
    lcdCmd(0x01);
    LcdCmd(0x80);
    Lcd_Write("GETTING READY");
    Delay_ms(1000);
}
```

```
void firstDisplay(){
    batVolt = calBat();
    if(batVolt > 0){
      LcdCmd(0x80);
      LCD_WRITE("BAT: ");
      LcdCmd(0x85);
      LcdData(48+(batVolt/1000));
      LcdCmd(0x86);
      LcdData(48+(batVolt/100)%10);
      LcdCmd(0x87);
      LCD_WRITE(".");
      LcdCmd(0x88);
      LcdData(48+(batVolt/10)%10);
      LcdCmd(0x89);
      LcdData(48+ (batVolt%10));
      LcdCmd(0x8a);LCD_WRITE(" ");
      LcdCmd(0x8B);LCD_WRITE("VOLTS");
      if(cRly){
        if(batVolt >= batFullCutOff){
           soc = 100;
        }
        else if (batVolt <= actualLow){
           soc = 0;
        }
        else{
           newSoc = (batFullCutOff-actualLow) - (batFullCutOff - batVolt);
           soc = (newSoc*100)/(batFullCutOff-actualLow);
        }

      }
      else{
        if(batVolt >= batFullCutOff2){
          soc = 100;
        }
        else if(batVolt <= actualLow){
          soc = 0;
        }
        else{
          newSoc = (batFullCutOff2-actualLow)-(batFullCutOff2 - batVolt);
          soc = newSoc * (100/(batFullCutOff2-actualLow));
        }
      }
      LcdCmd(0xc0);
      LCD_WRITE("BAT SOC: ");
```

```
        LcdCmd(0xc9);
        LCD_WRITE(" ");
        if(soc == 100){
          LcdCmd(0xc9);
          LcdData(48+ (soc/100));
        }
        LcdCmd(0xcA);
        LcdData(48+ (soc/10)%10);
        LcdCmd(0xcB);
        LcdData(48+ (soc%10));
        LcdCmd(0xcC);
        LCD_WRITE(" ");
        LcdCmd(0xcD);
        LCD_WRITE("%  ");
      }
      else{
        LcdCmd(0x80);
        LCD_WRITE("PLEASE CONNECT");
        LcdCmd(0xc3);
        LCD_WRITE("A BATTERY");
      }
}
void firstDisplay2(){
      batVolt = calBat2();
      if(batVolt > 0){
        LcdCmd2(0x80);
        LCD_WRITE2("BAT: ");
        LcdCmd2(0x85);
        LcdData2(48+(batVolt/1000));
        LcdCmd2(0x86);
        LcdData2(48+(batVolt/100)%10);
        LcdCmd2(0x87);
        LCD_WRITE2(".");
        LcdCmd2(0x88);
        LcdData2(48+(batVolt/10)%10);
        LcdCmd2(0x89);
        LcdData2(48+(batVolt%10));
        LcdCmd2(0x8A);LCD_WRITE2(" ");
        LcdCmd2(0x8B);;LCD_WRITE2("VOLTS");
        if(cRly){
          if(batVolt >= batFullCutOff){
             soc = 100;
          }
          else if(batVolt <= actualLow){
             soc = 0;
```

```
            }
            else{
                newSoc = (batFullCutOff-actualLow) - (batFullCutOff - batVolt);
                soc = (newSoc*100)/(batFullCutOff-actualLow);
            }
          }
          else{
            if(batVolt >= batFullCutOff2){
              soc = 100;
            }
            else if(batVolt <= actualLow){
              soc = 0;
            }
            else{
              newSoc =(batFullCutOff2 - actualLow) -  (batFullCutOff2 - batVolt);
              soc = newSoc * (100/(batFullCutOff2-actualLow));
            }
          }
          LcdCmd2(0xc0);
          LCD_WRITE2("BAT SOC: ");
          LcdCmd2(0xc9);
          LCD_WRITE2(" ");
          if(soc == 100){
           LcdCmd2(0xc9);
           LcdData2(48+ (soc/100));
          }
          LcdCmd2(0xcA);
          LcdData2(48+ (soc/10)%10);
          LcdCmd2(0xcB);
          LcdData2(48+ (soc%10));
          LcdCmd2(0xcC);
          LCD_WRITE2(" ");
          LcdCmd2(0xcD);
          LCD_WRITE2("%  ");
        }
        else{
          LcdCmd2(0x80);
          LCD_WRITE2("PLEASE CONNECT");
          LcdCmd2(0xc0);
          LCD_WRITE2("   A BATTERY   ");
        }
}
void incPos(){
    if(menu < 3){
      menu++;
```

```
    }
    else{
      menu = 1;
    }
    clearLCD();
    lcdCmd(0x80);
    if(menu == 2){
      LCD_WRITE("SET LOW START");
      lcdCmd(0xc2);
      LCD_WRITE("VOLTAGE");
      lcdCmd(0x0c);
    }
    else if(menu == 1){
      LCD_WRITE("SET FULL CUTOFF");
      lcdCmd(0xc2);
      LCD_WRITE("VOLTAGE");
      lcdCmd(0x0c);
    }
    else if(menu == 3){
       LCD_WRITE("SET CUTOFF");
       lcdCmd(0xc0);
       LCD_WRITE("TEMPERATURE");
       lcdCmd(0x0c);
    }
}

void btnBack(){
  if(Back){
     btnDly();
     if(Back){
        clearLCD();
        if(menu2 == 1){
           menu2 = 0;
           menu = 0;
           firstDisplay();
        }
        else if(menu2 == 2){
           menu2 = 1;
           menu = 0;
           incPos();
        }
        lcdCmd(0x0c);
     }
  }
}
```

```
void setFull(){
    full[0] = 0;
    full[1] = 0;
    full[3] = 0;
    full[4] = 0;
    clearLCD();
    LcdCmd(0x80);
    LCD_WRITE("SET FULL");
    LcdCmd(0xc0);
    lcdData(48+full[0]);
    LcdCmd(0xc1);
    lcdData(48+full[1]);
    LcdCmd(0xc2);
    lCD_WRITE(".");
    LcdCmd(0xc3);
    lcdData(48+full[3]);
    LcdCmd(0xc4);
    lcdData(48+full[4]);
    lcdCmd(0xc8);
    LCD_WRITE("VOLTS");
    lcdCmd(0x0e);
    lcdCmd(0xc0);
    curPos = 0;
}
void setLow(){
    low[0] = 0;
    low[1] = 0;
    low[3] = 0;
    low[4] = 0;
    clearLCD();
    LcdCmd(0x80);
    LCD_WRITE("SET LOW");
    LcdCmd(0xc0);
    lcdData(48+low[0]);
    LcdCmd(0xc1);
    lcdData(48+low[1]);
    LcdCmd(0xc2);
    lCD_WRITE(".");
    LcdCmd(0xc3);
    lcdData(48+low[3]);
    LcdCmd(0xc4);
    lcdData(48+low[4]);
    lcdCmd(0xc8);
    LCD_WRITE("VOLTS");
```

```
    lcdCmd(0x0e);
    lcdCmd(0xc0);
    curPos = 0;
}
void setTemp(){
    cutTemp[0] = 0;
    cutTemp[1] = 0;
    cutTemp[2] = 0;
    cutTemp[4] = 0;
    cutTemp[5] = 0;
    clearLCD();
    LcdCmd(0x80);
    LCD_WRITE("SET TEMP. CUTOFF");
    LcdCmd(0xc0);
    lcdData(48+cutTemp[0]);
    LcdCmd(0xc1);
    lcdData(48+cutTemp[1]);
    LcdCmd(0xc2);
     lcdData(48+cutTemp[2]);
    LcdCmd(0xc3);
    lCD_WRITE(".");
    LcdCmd(0xc4);
    lcdData(48+cutTemp[4]);
    LcdCmd(0xc5);
    lcdData(48+cutTemp[5]);
    lcdCmd(0xc7);
    LcdData(223);
    lcdCmd(0xc8);
    LCD_WRITE("C");
    lcdCmd(0x0e);
    lcdCmd(0xc0);
    curPos = 0;
}

void moveRight(){
    if(curPos < 4){
      curPos++;
      lcdCmd(0x14);
      if(curPos == 2){
         curPos++;
         lcdCmd(0x14);
      }
    }
    else{
      curPos = 0;
```

```
      lcdCmd(0xc0);
    }
}
void moveRight2(){
    if(curPos5 < 5){
      curPos5++;
      lcdCmd(0x14);
      if(curPos5 == 3){
        curPos5++;
        lcdCmd(0x14);
      }
    }
    else{
      curPos5 = 0;
      lcdCmd(0xc0);
    }
}

long Display_Temperature(unsigned int temp2write) {
  const unsigned short RES_SHIFT = TEMP_RESOLUTION - 8;
  char temp_whole;
  unsigned int temp_fraction;

  // Check if temperature is negative
  if (temp2write & 0x8000) {
    text[0] = '-';
    temp2write = ~temp2write + 1;
    }

  // Extract temp_whole
  temp_whole = temp2write >> RES_SHIFT ;

  // Convert temp_whole to characters
  if (temp_whole/100)
    text[0] = temp_whole/100  + 48;
  else
    text[0] = '0';

  text[1] = (temp_whole/10)%10 + 48;           // Extract tens digit
  text[2] =  temp_whole%10     + 48;           // Extract ones digit

  // Extract temp_fraction and convert it to unsigned int
  temp_fraction  = temp2write << (4-RES_SHIFT);
  temp_fraction &= 0x000F;
  temp_fraction *= 625;
```

```
  // Convert temp_fraction to characters
  text[4] =  temp_fraction/1000    + 48;         // Extract thousands digit
  text[5] = (temp_fraction/100)%10 + 48;         // Extract hundreds digit
  text[6] = (temp_fraction/10)%10  + 48;         // Extract tens digit
  text[7] =  temp_fraction%10      + 48;         // Extract ones digit

  // Print temperature on LCD
  LcdCmd(0xc4);
  lcdData(text[0]);
   LcdCmd(0xc5);
  lcdData(text[1]);
  LcdCmd(0xc6);
  lcdData(text[2]);
  LcdCmd(0xc7);
  lcdData(text[3]);
   LcdCmd(0xc8);
  lcdData(text[4]);
   LcdCmd(0xc9);
  lcdData(text[5]);
   LcdCmd(0xca);
  lcdData(text[6]);
   LcdCmd(0xcb);
  lcdData(text[7]);
  holdTemp = text[0]*10000 + text[1]*1000 + text[2]*100 + text[4]*10 + text[5];
  return ((long)(holdTemp));
}

void calFull(){
    fBat = (full[0]*1000) + (full[1]*100) + (full[3]*10) + full[4];
}
void calLow(){
    lBat = (low[0]*1000) + (low[1]*100) + (low[3]*10) + low[4];
}
void calTemp(){
    cTemp = (cutTemp[0]*10000) + (cutTemp[1]*1000) + (cutTemp[2]*100) + cutTemp[4]*10 + cutTemp[5];
}

void saveFull(const char who){
    if(batN == 1200){fullCutValue = 48+1;fullCutValue2 = 48+0;fullCutValue3 = 48+0;}
    else if(batN == 2400){fullCutValue = 48+0;fullCutValue2 = 48+1;fullCutValue3 = 48+0;}
    else if(batN == 4800){fullCutValue = 48+0;fullCutValue2 = 48+0;fullCutValue3 = 48+1;}
    eepromWrite(1,fullCutValue);
    eepromWrite(20,fullCutValue2);
    eepromWrite(21,fullCutValue3);
```

```
    if(who == 0){

      eepromWrite(2,48+(fBat/1000));
      eepromWrite(3,48+(fBat/100)%10);
      eepromWrite(4,48+(fBat/10)%10);
      eepromWrite(5,48+(fBat%10));
       clearLCD();
       LcdCmd(0x80);
       LcdData(48+(fBat/1000));
       LcdCmd(0x81);
       LcdData(48+(fBat/100)%10);
       LcdCmd(0x82);
       LCD_WRITE(".");
       LcdCmd(0x83);
       LcdData(48+(fBat/10)%10);
       LcdCmd(0x84);
       LcdData(48+fBat%10);
       LcdCmd(0x0c);
       delay_ms(500);
    }
    else if(who == 1){
      eepromWrite(2,48+(batFullCutOff3/1000));
      eepromWrite(3,48+(batFullCutOff3/100)%10);
      eepromWrite(4,48+(batFullCutOff3/10)%10);
      eepromWrite(5,48+(batFullCutOff3%10));
    }
}
void saveLow(const char who){
    if(batN == 1200){lowStartValue = 48+1;lowStartValue2 = 48+0;lowStartValue3 = 48+0;}
    else if(batN == 2400){lowStartValue = 48+0;lowStartValue2 = 48+1;lowStartValue3 = 48+0;}
    else if(batN == 4800){lowStartValue = 48+0;lowStartValue2 = 48+0;lowStartValue3 = 48+1;}
    eepromWrite(6,lowStartValue);
    eepromWrite(22,lowStartValue2);
    eepromWrite(23,lowStartValue3);
    if(who == 0){
      eepromWrite(7,48+(lBat/1000));
      eepromWrite(8,48+(lBat/100)%10);
      eepromWrite(9,48+(lBat/10)%10);
      eepromWrite(10,48+(lBat%10));
      clearLCD();
       LcdCmd(0x80);
       LcdData(48+(lBat/1000)%10);
       LcdCmd(0x81);
       LcdData(48+(lBat/100)%10);
       LcdCmd(0x82);
```

```
      LCD_WRITE(".");
      LcdCmd(0x83);
      LcdData(48+(lBat/10)%10);
      LcdCmd(0x84);
      LcdData(48+lBat%10);
      LcdCmd(0x0c);
      delay_ms(500);
    }
    else if(who == 1){
      eepromWrite(7,48+(batLowStart3/1000));
      eepromWrite(8,48+(batLowStart3/100)%10);
      eepromWrite(9,48+(batLowStart3/10)%10);
      eepromWrite(10,48+(batLowStart3%10));
    }
}
void saveTemp(const char who){
      tempSetStatus = 48+1;
      eepromWrite(11,tempSetStatus);
  if(who == 0){
      eepromWrite(12,48+(cTemp/10000));
      eepromWrite(13,48+((cTemp/1000)%10));
      eepromWrite(14,48+((cTemp/100)%10));
      eepromWrite(15,48+((cTemp/10)%10));
      eepromWrite(16,48+(cTemp%10));
      clearLCD();
     LcdCmd(0x80);
     LcdData(48+(cTemp/10000)%10);
     LcdCmd(0x81);
     LcdData(48+(cTemp/1000)%10);
     LcdCmd(0x82);
     LcdData(48+(cTemp/100)%10);
     LcdCmd(0x83);
     LCD_WRITE(".");
     LcdCmd(0x84);
     LcdData(48+(cTemp/10)%10);
     LcdCmd(0x85);
     LcdData(48+cTemp%10);
     LcdCmd(0x0c);
     delay_ms(500);
    }
   else if(who == 1){
      eepromWrite(12,48+(tempCutOff3/10000));
      eepromWrite(13,48+((tempCutOff3/1000)%10));
      eepromWrite(14,48+((tempCutOff3/100)%10));
      eepromWrite(15,48+((tempCutOff3/10)%10));
```

```
      eepromWrite(16,48+(tempCutOff3%10));
    }
}

void getTemperature(){
    Delay_ms(500);
    //--- Perform temperature reading
    Ow_Reset(&PORTE, 2);                    // Onewire reset signal
    Ow_Write(&PORTE, 2, 0xCC);              // Issue command SKIP_ROM
    Ow_Write(&PORTE, 2, 0x44);              // Issue command CONVERT_T
    Delay_us(120);

    Ow_Reset(&PORTE, 2);
    Ow_Write(&PORTE, 2, 0xCC);              // Issue command SKIP_ROM
    Ow_Write(&PORTE, 2, 0xBE);              // Issue command READ_SCRATCHPAD

    temp =  Ow_Read(&PORTE, 2);
    temp = (Ow_Read(&PORTE, 2) << 8) + temp;
    batTemp = Display_Temperature(temp);

}

void checkFull(const char who){
   //batVolt = calBat();
   if(batN == 1200){
     if(who == 0){
       if(fBat >= 1280 && fBat <= 1500){
         saveFull(0);
         fullSet = fBat;
       }
     }
     else{
       if(batFullCutOff3 >= 1280 && batFullCutOff3 <= 1500){
         saveFull(1);
         fullSet = batFullCutOff3;
       }
     }
   }
   else if(batN == 2400){
     if(who == 0){
       if(fBat >= 2700 && fBat <= 3200){
         saveFull(0);
         fullSet = fBat;
       }
     }
```

```
    else{
      if(batFullCutOff3 >= 2700 && batFullCutOff3 <= 3200){
        saveFull(1);
        fullSet = batFullCutOff3;
      }

    }
  }
  else if(batN == 4800){
    if(who == 0){
      if(fBat >= 4900 && fBat <= 5200){
        saveFull(0);
        fullSet = fBat;
      }
    }
    else{
      if(batFullCutOff3 >= 4900 && batFullCutOff3 <= 5200){
        saveFull(1);
        fullSet = batFullCutOff3;
      }
    }

  }

}
void checkLow(const char who){
   batVolt = calBat();
  if(batVolt > 0 && batVolt < 1500){
    if(who == 0){
      if(lBat >= 1180 && lBat <= 1280){
        saveLow(0);
        lowSet = lBat;
      }
    }
    else{
      if(batLowStart3 >= 1180 && batLowStart3 <= 1280){
        saveLow(1);
        lowSet = batLowStart3;
      }
    }

  }
  else if(batVolt > 1500 && batVolt < 3200){
    if(who == 0){
      if(lBat >= 2300 && lBat <= 2400){
```

```
      saveLow(0);
      lowSet = lBat;
     }
    }
    else{
      if(batLowStart3 >= 2300 && batLowStart3 <= 2400){
      saveLow(1);
      lowSet = batLowStart3;
     }
    }

  }
  else if(batVolt > 3200 && batVolt < 5200){
    if(who == 0){
      if(lBat >= 4650 && lBat <= 4800){
        saveLow(0);
        lowSet = lBat;
      }
    }
    else{
      if(batLowStart3 >= 4650 && batLowStart3 <= 4800){
        saveLow(1);
        lowSet = batLowStart3;
      }
    }

  }
}
void checkTemp(const char who){
  if(who == 0){
    if(cTemp >= 5000 && cTemp <= 10000 ){
      saveTemp(0);
      tempCutOff = cTemp;
    }
  }
  else{
    if(tempCutOff3 >= 5000 && tempCutOff3 <= 10000 ){
      saveTemp(1);
      tempCutOff = tempCutOff3;
    }
  }

}
void otherStatus(){
    if(eepromRead(0)-48 == 1){instantSwitch = 48+1;}else{instantSwitch = 48+0;}
```

```
    if(eepromRead(11)-48 == 1){tempSetStatus = 48+1;
      tempCutOff = (eepromRead(12)-48)*10000 + (eepromRead(13)-48)*1000 + (eepromRead(14)-48)*100 +(eepromRead(15)-48)*10 + (eepromRead(16)-48);
     }
    else{
     tempCutOff = 7200;
     cTemp = tempCutOff;
     checkTemp(0);
    }
    if(eepromRead(17) == 1){sendType= 1;}else{sendType = 0;}
    if(eepromRead(18) == 1){sentVal =1;} else{sentVal = 0;}
   if(eepromRead(1)-48 == 1){fullCutValue = 48+1;fullCutValue2 = 48+0;fullCutValue3 = 48+0;}
   else if(eepromRead(20) == 1){fullCutValue = 48+0;fullCutValue2 = 48+1;fullCutValue3 = 48+0;}
   else if(eepromRead(21) == 1){fullCutValue = 48+0;fullCutValue2 = 48+0;fullCutValue3 = 48+1;}
   if(eepromRead(6)-48 == 1){lowStartValue = 48+1;lowStartValue2 = 48+0;lowStartValue3 = 48+0;}
   else if(eepromRead(22) == 1){lowStartValue = 48+0;lowStartValue2 = 48+1;lowStartValue3 = 48+0;}
   else if(eepromRead(23) == 1){lowStartValue = 48+0;lowStartValue2 = 48+0;lowStartValue3 = 48+1;}
}

void checkBat(){
   otherStatus();
   batVolt = calBat();
   batNominalVol = batVolt;

   fullSet = ((eepromRead(2)-48)*1000) + ((eepromRead(3)-48)*100) + ((eepromRead(4)-48)*10) + (eepromRead(5)-48);
   lowSet = ((eepromRead(7)-48)*1000) + ((eepromRead(8)-48)*100) + ((eepromRead(9)-48)*10) + (eepromRead(10)-48);

   if(batNominalVol > 0 && batNominalVol < 1500){
     batN = 1200;
     validBatteryStatus = 48+1;
     batFullCutOff2 = 1280;
     batLowStart2 = 1180;
     actualLow = 1180;
     actualFull = 1400;
     if(fullCutValue - 48 == 0){
       batFullCutOff = 1450;
       fBat = batFullCutOff;
       checkFull(0);
     }
     else{
       batFullCutOff = fullSet;
     }
     if(lowStartValue - 48 == 0){
      batLowStart = 1180;
      lBat = batLowStart;
      checkLow(0);
```

```
    }
    else{
        batLowStart = lowSet;
    }
}
else if(batNominalVol >= 1500 && 3200 > batNominalVol){
    batN = 2400;
    validBatteryStatus = 48+1;
    batFullCutOff2 = 2750;
    batLowStart2 = 2360;
    actualLow = 2360;
    actualFull = 2800;
    if(fullCutValue2 - 48 == 0){
      batFullCutOff = 2950;
      fBat = batFullCutOff;
      checkFull(0);
    }
    else{
        batFullCutOff = fullSet;
    }
    if(lowStartValue2 - 48 == 0){
     batLowStart = 2360;
     lBat = batLowStart;
     checkLow(0);
    }
    else{
        batLowStart = lowSet;
    }
}
else if(batNominalVol >= 4000 && 5200 >= batNominalVol){
    batN = 4800;
    validBatteryStatus = 48+1;
    batFullCutOff2 = 5000;
    batLowStart2 = 4650;
    actualLow = 4650;
    actualFull = 5100;
    if(fullCutValue3 - 48 == 0){
      batFullCutOff = 5100;
      fBat = batFullCutOff;
      checkFull(0);
    }
    else{
        batFullCutOff = fullSet;
    }
    if(lowStartValue3 - 48 == 0){
```

```
        batLowStart = 4700;
        lBat = batLowStart;
        checkLow(0);
      }
      else{
        batLowStart = lowSet;
      }
    }
    else{
      validBatteryStatus = 48+0;
      batN = 0;
      batFullCutOff = 0;
      batLowStart = 0;
      actualLow = 0;
    }

}

void btnControl(){
  if(Enter){
    btnDly();
    if(Enter){
      menu2 = 1;
      incPos();
      while(menu != 0){
        btnBack();
        if(Right){
          btnDly();
          if(Right){
            incPos();
          }
        }
        if(Enter){
          btnDly();
          if(Enter){
            menu2 = 2;
            if(menu == 1){
              clearLCD();
              setFull();
              while(menu ==1 && menu2 == 2){
                btnBack();
                if(Right){
                  btnDly();
                  if(Right){
                    moveRight();
```

```
  }
 }
 if(Up){
  btnDly();
  if(Up){
    if((full[curPos]+48) < 57){
     full[curPos]++;
    }
    else{
     full[curPos] = 0;
    }
    lcdCmd(192+curPos);
    lcdData(48+full[curPos]);
    lcdCmd(192+curPos);
  }
 }
 if(Enter){
  btnDly();
  if(Enter){
   calFull();
   if(fBat > 0){
     checkFull(0);
     batFullCutOff = fBat;
     sendType = 1; eepromWrite(17,sendType);
     sentVal = 1; eepromWrite(18,sentVal);
     fullSet = batFullCutOff;
     /*
     clearLCD();
     LcdCmd(0x80);
     LcdData(48+(fBat/1000));
     LcdCmd(0x81);
     LcdData(48+(fBat/100)%10);
     LcdCmd(0x82);
     LCD_WRITE(".");
     LcdCmd(0x83);
     LcdData(48+(fBat/10)%10);
     LcdCmd(0x84);
     LcdData(48+fBat%10);
     LcdCmd(0x0c);
     delay_ms(500);
     */
     menu2 = 1;
     menu = 0;
     incPos();
   }
```

```
      }
    }
  }

}
else if(menu == 2){
  clearLCD();
  setLow();
  while(menu == 2 && menu2 == 2){
    btnBack();
    if(Right){
      btnDly();
      if(Right){
        moveRight();
      }
    }
    if(Up){
      btnDly();
      if(Up){
        if((low[curPos]+48) < 57){
          low[curPos]++;
        }
        else{
          low[curPos] = 0;
        }
        lcdCmd(192+curPos);
        lcdData(48+low[curPos]);
        lcdCmd(192+curPos);
      }
    }
    if(Enter){
      btnDly();
      if(Enter){
        calLow();
        if(lBat > 0){
          checkLow(0);
          batLowStart = lBat;
          sendType = 1;eepromWrite(17,sendType);
          sentVal = 1; eepromWrite(18,sentVal);
          lowSet = batLowStart;
          /*
          clearLCD();
          LcdCmd(0x80);
          LcdData(48+(lBat/1000)%10);
          LcdCmd(0x81);
```

```
          LcdData(48+(lBat/100)%10);
          LcdCmd(0x82);
          LCD_WRITE(".");
          LcdCmd(0x83);
          LcdData(48+(lBat/10)%10);
          LcdCmd(0x84);
          LcdData(48+lBat%10);
          LcdCmd(0x0c);
          delay_ms(500);
          */
          menu2 = 1;
          menu = 1;
          incPos();
        }
      }
    }
  }
}
else if(menu == 3){
  clearLCD();
  setTemp();
  while(menu == 3 && menu2 == 2){
    btnBack();
    if(Right){
      btnDly();
      if(Right){
        moveRight2();
      }
    }
    if(Up){
      btnDly();
      if(Up){
        if((cutTemp[curPos5]+48) < 57){
          cutTemp[curPos5]++;
        }
        else{
          cutTemp[curPos5] = 0;
        }
        lcdCmd(192+curPos5);
        lcdData(48+cutTemp[curPos5]);
        lcdCmd(192+curPos5);
      }
    }
    if(Enter){
      btnDly();
```

```
                if(Enter){
                  calTemp();
                  if(cTemp > 0){
                    checkTemp(0);
                    sendType = 1;eepromWrite(17,sendType);
                    sentVal = 1; eepromWrite(18,sentVal);
                    tempCutOff = cTemp;
                    /*
                    clearLCD();
                    LcdCmd(0x80);
                    LcdData(48+(cTemp/10000)%10);
                    LcdCmd(0x81);
                    LcdData(48+(cTemp/1000)%10);
                    LcdCmd(0x82);
                    LcdData(48+(cTemp/100)%10);
                    LcdCmd(0x83);
                    LCD_WRITE(".");
                    LcdCmd(0x84);
                    LcdData(48+(cTemp/10)%10);
                    LcdCmd(0x85);
                    LcdData(48+cTemp%10);
                    LcdCmd(0x0c);
                    delay_ms(500);
                    */

                    menu2 = 1;
                    menu = 2;
                    incPos();
                  }
                 }
                }
               }
              }
             }
            }
           }
          }
         }
        }
      }

void otherStatus2(){
    if(eepromRead(0)-48 == 1){instantSwitch = 48+1;}else{instantSwitch = 48+0;}
    if(eepromRead(11)-48 == 1){
       tempSetStatus = 48+1;
       tempCutOff = (eepromRead(12)-48)*10000 + (eepromRead(13)-48)*1000 + (eepromRead(14)-48)*100 +(eepromRead(15)-48)*10 + (eepromRead(16)-48);
```

```
      }
     else{
       tempSetStatus = 48+0;
       tempCutOff = 7200;
     }
     if(eepromRead(17) == 1){sendType= 1;}else{sendType = 0;}
     if(eepromRead(18) == 1){sentVal =1;} else{sentVal = 0;}
    if(eepromRead(1)-48 == 1){fullCutValue = 48+1;fullCutValue2 = 48+0;fullCutValue3 = 48+0;}
    else if(eepromRead(20) == 1){fullCutValue = 48+0;fullCutValue2 = 48+1;fullCutValue3 = 48+0;}
    else if(eepromRead(21) == 1){fullCutValue = 48+0;fullCutValue2 = 48+0;fullCutValue3 = 48+1;}
    if(eepromRead(6)-48 == 1){lowStartValue = 48+1;lowStartValue2 = 48+0;lowStartValue3 = 48+0;}
    else if(eepromRead(22) == 1){lowStartValue = 48+0;lowStartValue2 = 48+1;lowStartValue3 = 48+0;}
    else if(eepromRead(23) == 1){lowStartValue = 48+0;lowStartValue2 = 48+0;lowStartValue3 = 48+1;}
}

void checkBat2(){
    fullSet = ((eepromRead(2)-48)*1000) + ((eepromRead(3)-48)*100) + ((eepromRead(4)-48)*10) + (eepromRead(5)-48);
    lowSet = ((eepromRead(7)-48)*1000) + ((eepromRead(8)-48)*100) + ((eepromRead(9)-48)*10) + (eepromRead(10)-48);
    if(batN == 1200){
       validBatteryStatus = 48+1;
       if(fullCutValue - 48 == 0){
         batFullCutOff = 1450;
       }
       else{
          batFullCutOff = fullSet;
       }
       if(lowStartValue - 48 == 0){
        batLowStart = 1180;
       }
       else{
          batLowStart = lowSet;
       }
    }
    else if(batN == 2400){
        validBatteryStatus = 48+1;
        if(fullCutValue2 - 48 == 0){
          batFullCutOff = 2950;
        }
        else{
          batFullCutOff = fullSet;
        }
        if(lowStartValue2 - 48 == 0){
         batLowStart = 2360;
        }
        else{
```

```
            batLowStart = lowSet;
        }
    }
    else if(batN == 4800){
        validBatteryStatus = 48+1;
        if(fullCutValue3 - 48 == 0){
            batFullCutOff = 5150;
        }
        else{
            batFullCutOff = fullSet;
        }
        if(lowStartValue3 - 48 == 0){
            batLowStart = 4700;
        }
        else{
            batLowStart = lowSet;
        }
    }
    else{
        validBatteryStatus = 48+0;
        batN = 0;
        batFullCutOff = 0;
        batLowStart = 0;
        actualLow = 0;
    }

}

void analysisRes(){
      clearLCD();
      LcdCmd(0x80);
      Lcd_Write(webData2);
     if(webData2[0] == '1'){
       instantSwitch = 48+1;
       eepromWrite(0,instantSwitch);
     }
     else{
       instantSwitch = 48+0;
       eepromWrite(0,instantSwitch);
     }
     batFullCutOff3 = (webData2[2]-48)*1000 + (webData2[3]-48)*100 + (webData2[4]-48)*10 + (webData2[5]-48);
     if(batFullCutOff3 > 0){
        if(bFull1 == bFull2){
          checkFull(1);
        }
```

```
    }
    batLowStart3 = (webData2[7]-48)*1000 + (webData2[8]-48)*100 + (webData2[9]-48)*10 + (webData2[10]-48);
    if(batLowStart3 > 0){
     if(bLow1 == bLow2){
       checkLow(1);
     }
    }
    tempCutOff3 =  (webData2[12]-48)*10000 + (webData2[13]-48)*1000 + (webData2[15]-48)*100 + (webData2[15]-48)*10 + (webData2[17]-48);
    if(tempCutOff3 > 0){
     if(bTemp1 == bTemp2){
       checkTemp(1);
     }
    }
    otherStatus2();
    checkBat2();

}

void writeData(){
  webData[3] = 48+0;
  if(cRly){
    webData[3] = 48+1;
  }
  webData[5] = 48+0;
  batVolt = calBat();
  if(batLowStart > batVolt){
   webData[5] = 48+1;
  }
  webData[7] = 48+(batN/1000);
  webData[8] = 48+((batN/100)%10);
  webData[9] = 48+((batN/10)%10);
  webData[10] = 48+(batN%10);
  webData[12] = 48+(batVolt/1000);
  webData[13] = 48+((batVolt/100)%10);
  webData[14] = 48+((batVolt/10)%10);
  webData[15] = 48+(batVolt%10);
  if(batVolt > 0){
    validBatteryStatus = 48+1;
  }
  else{
   validBatteryStatus = 48+0;
  }
  batVolt = calBat();

  if(cRly){
```

```
    if(batVolt >= batFullCutOff){
        soc = 100;
    }
    else if (batVolt <= actualLow){
        soc = 0;
    }
    else{
         newSoc = (batFullCutOff-actualLow) - (batFullCutOff - batVolt);
         soc = (newSoc*100)/(batFullCutOff-actualLow);
    }

}
else{
    if(batVolt >= batFullCutOff2){
       soc = 100;
    }
    else if(batVolt <= actualLow){
       soc = 0;
    }
    else{
       newSoc = (batFullCutOff2-actualLow)-(batFullCutOff2 - batVolt);
       soc = newSoc * (100/(batFullCutOff2-actualLow));
    }
}
/*
if(cRly){

  per = ((batFullCutOff - batVolt)*100)/(batFullCutOff - actualLow);
  if(batVolt > batFullCutOff){
     per = 100;
  }
  else if(batVolt <= 0){
     per = 0;
  }
}
else{
     per = ((batFullCutOff2 - batVolt)*100)/(batFullCutOff2 - actualLow);
    if(batVolt > batFullCutOff2){
       per = 100;
    }
    else if(batVolt <= 0){
       per = 0;
    }
}
*/
```

```
  webData[17] = 48+(soc/100);
  webData[18] = 48+((soc/10)%10);
  webData[19] = 48+(soc/10);

}

void writeDatb(){
 if(batN == 1200){
  if(fullCutValue -48 == 1){
   webDatb[3] = eepromRead(2);
   webDatb[4] = eepromRead(3);
   webDatb[5] = eepromRead(4);
   webDatb[6] = eepromRead(5);
  }
  else{
   webDatb[3] = 48+ (actualFull/1000);
   webDatb[4] = 48+ ((actualFull/100)%10);
   webDatb[5] = 48+ ((actualFull/10)%10);
   webDatb[6] = 48+ (actualFull%10);
  }
  if(lowStartValue-48 == 1){
   webDatb[8] = eepromRead(7);
   webDatb[9] = eepromRead(8);
   webDatb[10] = eepromRead(9);
   webDatb[11] = eepromRead(10);
  }
  else{
   webDatb[8] = 48+ (actualLow/1000);
   webDatb[9] = 48+ ((actualLow/100)%10);
   webDatb[10] = 48+ ((actualLow/10)%10);
   webDatb[11] = 48+ (actualLow%10);
  }
 }
 else if(batN == 2400){
  if(fullCutValue2 -48 == 1){
   webDatb[3] = eepromRead(2);
   webDatb[4] = eepromRead(3);
   webDatb[5] = eepromRead(4);
   webDatb[6] = eepromRead(5);
  }
  else{
   webDatb[3] = 48+ (actualFull/1000);
   webDatb[4] = 48+ ((actualFull/100)%10);
   webDatb[5] = 48+ ((actualFull/10)%10);
   webDatb[6] = 48+ (actualFull%10);
```

```
  }
  if(lowStartValue2-48 == 1){
    webDatb[8] = eepromRead(7);
    webDatb[9] = eepromRead(8);
    webDatb[10] = eepromRead(9);
    webDatb[11] = eepromRead(10);
  }
  else{
    webDatb[8] = 48+ (actualLow/1000);
    webDatb[9] = 48+ ((actualLow/100)%10);
    webDatb[10] = 48+ ((actualLow/10)%10);
    webDatb[11] = 48+ (actualLow%10);
  }
}
else if(batN == 4800){
  if(fullCutValue3 -48 == 1){
    webDatb[3] = eepromRead(2);
    webDatb[4] = eepromRead(3);
    webDatb[5] = eepromRead(4);
    webDatb[6] = eepromRead(5);
  }
  else{
    webDatb[3] = 48+ (actualFull/1000);
    webDatb[4] = 48+ ((actualFull/100)%10);
    webDatb[5] = 48+ ((actualFull/10)%10);
    webDatb[6] = 48+ (actualFull%10);
  }
  if(lowStartValue3-48 == 1){
    webDatb[8] = eepromRead(7);
    webDatb[9] = eepromRead(8);
    webDatb[10] = eepromRead(9);
    webDatb[11] = eepromRead(10);
  }
  else{
    webDatb[8] = 48+ (actualLow/1000);
    webDatb[9] = 48+ ((actualLow/100)%10);
    webDatb[10] = 48+ ((actualLow/10)%10);
    webDatb[11] = 48+ (actualLow%10);
  }
}
if(tempSetStatus - 48 == 1){
  webDatb[13] = eepromRead(12);
  webDatb[14] = eepromRead(13);
  webDatb[15] = eepromRead(14);
  webDatb[16] = eepromRead(15);
```

```
    webDatb[17] = eepromRead(16);
  }
  else{
    webDatb[13] = 48+ (cTemp/10000);
    webDatb[14] = 48+ ((cTemp/1000)%10);
    webDatb[15] = 48+ ((cTemp/100)%10);
    webDatb[16] = 48+ ((cTemp/10)%10);
    webDatb[17] = 48+ (cTemp%10);

  }

}

void checkServer(){
     timeCnt = 1;  intCnt = 0; cont = 0; timeOut = 30;
     sentVal = 0;eepromWrite(18,sentVal);
     if(sVal == 0){
       writeDatb();
     }
     else{
       if(sendType == 0){
         writeData();
       }
       else if(sendType == 1){
         writeDatb();
       }
     }

     clearLCD();
     lcdCmd(0x80);
     if(sVal == 0){
       lcd_Write(webDatb);
     }
     else{
       if(sendType == 0){lcd_Write(webData);}
       else if(sendType == 1){lcd_Write(webDatb);}
     }

     delay_ms(1000);


     if(sVal == 0){
       dataLen =  txtLen2(webDatb);
     }
     else{
```

```
 if(sendType == 0){dataLen =  txtLen2(webData);}
 else if(sendType == 1){dataLen = txtLen2(webDatb);}
}

UART1_Write_Text("AT+CIPSTART=\"TCP\",\"batterymonitor.sightdev.net\",80\r\n");
Delay_ms(2000);
T0CON.TMR0ON = 1;
while(cont != 200);
btnControl();
intCnt = 0; cont = 0; intCnt2 = 0; cont2 = 0;
timeout = 50;

//LcdCmd(0X01);
//LcdCmd(0x80);
//Lcd_Write(txt2);

testPos = txtFind(txt2,testConnect);
if(testPos > 0){
    WorkingLED = 1;
}
else{
    WorkingLED = 0;
}

/*
LcdCmd(0x80);
Lcd_Write(txt2);
LcdCmd(0xc0);
Lcd_Write(txt);
*/

Delay_ms(2000);
UART1_Write_Text("AT+CIPSEND=");
mainDataLen = txtLen(postRequest1) + txtLen(postRequest2) + dataLen + 2;
UartWrite((mainDataLen/100)+48);
UartWrite(((mainDataLen/10)%10)+48);
UartWrite((mainDataLen%10)+48);
Uart1_Write_Text("\r\n");
Delay_ms(2000);
T0CON.TMR0ON = 1;
while(cont != 200);
btnControl();
intCnt = 0;
cont = 0; intCnt2 = 0; cont2 = 0;
timeout = 50;
```

```
/*
LcdCmd(0x01);
LcdCmd(0x80);
Lcd_Write(txt2);
LcdCmd(0xc0);
Lcd_Write(txt);
Delay_ms(1000);
LcdCmd(0x01);
LcdCmd(0x80);
Lcd_Write(txt2);
LcdCmd(0xc0);
Lcd_Write(txt);
*/

Delay_ms(2000);
UartWriteText(postRequest1);
UartWrite((dataLen/10)+48);
UartWrite((dataLen % 10) + 48);
UartWriteText(postRequest2);
if(sVal == 0){
  UartWriteText2(webDatb);
}
else{
  if(sendType == 0){UartWriteText2(webData); }
  else if(sendType == 1){UartWriteText2(webDatb);}
}
Uart1_Write_Text("\r\n");
cont = 0; intCnt2 = 0; cont2 = 0; timeCnt = 3;
timeout = 70; intCnt12 = 0;
Delay_ms(2000);
T0CON.TMR0ON = 1;
while(cont != 200);
btnControl();
charPos = txtFind(txt8, "Length:");
if(charPos > 0){
    charPos += 7;
    if(txt8[charPos] != ""){
      while(txt8[charPos] != ""){
          charPos++;
      }
    }
    charPos++;
    clearLCD();
    LcdCmd(0xc0);
```

```
    Lcd_Write("WORKING");
    delay_ms(1000);

    for(k = 0;k < 16;k++){
      webData2[k] = txt8[charPos];
      charPos++;
    }
    sVal = 1;
    if(sendType == 1){
      bFull1 = ((webDatb[3]-48)*1000) + ((webDatb[4]-48)*100)+ ((webDatb[5]-48)*10)+ (webDatb[6]-48);
      bFull2 = ((eepromRead(2)-48)*1000)+ ((eepromRead(3)-48)*100) +((eepromRead(4)-48)*10) +(eepromRead(5)-48);
      bLow1 = ((webDatb[8]-48)*1000) + ((webDatb[9]-48)*100)+ ((webDatb[10]-48)*10)+ (webDatb[11]-48);
      bLow2 = ((eepromRead(7)-48)*1000)+ ((eepromRead(8)-48)*100) +((eepromRead(9)-48)*10) +(eepromRead(10)-48);
      bTemp1 = ((webDatb[13]-48)*10000) + ((webDatb[14]-48)*1000)+ ((webDatb[15]-48)*100)+ ((webDatb[16]-48)*10)+(webDatb[17]-48);
      bTemp2 = ((eepromRead(12)-48)*10000)+ ((eepromRead(13)-48)*1000) +((eepromRead(14)-48)*100) + ((eepromRead(15)-48)*10)+(eepromRead(16)-48);
      if(bFull1 == bFull2 && bLow1 == bLow2 && bTemp1 == bTemp2){
        sendType = 0;
      }
      else{
        sendType = 1;
      }
    }
    analysisRes();
}
else{
    LcdCmd(0x80);
    Lcd_Write("NOT Entered");
    LcdCmd(0xc0);
    Lcd_Write(txt8);
    Delay_ms(1000);

}
intCnt = 0;   timeCnt = 1;
cont = 0; intCnt2 = 0; cont2 = 0;
Uart1_Write_Text("AT+CIPCLOSE");
Uart1_Write_Text("\r\n");
Delay_ms(2000);
T1CON.TMR1ON = 1;
while(cont != 200);
btnControl();
intCnt = 0; cont = 0; intCnt2 = 0; cont2 = 0;
timeout = 50;

/*
clearLCD();
```

```
    LcdCmd(0x80);
    Lcd_Write(txt2);
    LcdCmd(0xc0);
    Lcd_Write(txt);
    Delay_ms(1000);
    clearLCD();
    LcdCmd(0x80);
    Lcd_Write(txt2);
    delay_ms(1000);
    */

}
/*
void Test(){
 clearLCD();
 writeData();
 LcdCmd(0x80); LcdData(webData[0]); LcdCmd(0x81); LcdData(webData[1]);LcdCmd(0x82); LcdData(webData[2]); LcdCmd(0x83); LcdData(webData[3]);
 LcdCmd(0x84); LcdData(webData[4]); LcdCmd(0x85); LcdData(webData[5]);LcdCmd(0x86); LcdData(webData[6]); LcdCmd(0x87); LcdData(webData[7]);
 LcdCmd(0x88); LcdData(webData[8]); LcdCmd(0x89); LcdData(webData[9]);LcdCmd(0x8a); LcdData(webData[10]); LcdCmd(0x8b); LcdData(webData[11]);
 LcdCmd(0x8c); LcdData(webData[12]); LcdCmd(0x8d); LcdData(webData[13]);LcdCmd(0x8e); LcdData(webData[14]); LcdCmd(0x8f); LcdData(webData[15]);
 LcdCmd(0xc0); LcdData(webData[16]); LcdCmd(0xc1); LcdData(webData[17]);LcdCmd(0xc2); LcdData(webData[18]); LcdCmd(0xc3); LcdData(webData[19]);
 LcdCmd(0xc4); LcdData(webData[20]);
 delay_ms(2000);
 clearLCD();
 writeDatb();
 LcdCmd(0x80); LcdData(webDatb[0]); LcdCmd(0x81); LcdData(webDatb[1]);LcdCmd(0x82); LcdData(webDatb[2]); LcdCmd(0x83); LcdData(webDatb[3]);
 LcdCmd(0x84); LcdData(webDatb[4]); LcdCmd(0x85); LcdData(webDatb[5]);LcdCmd(0x86); LcdData(webDatb[6]); LcdCmd(0x87); LcdData(webDatb[7]);
 LcdCmd(0x88); LcdData(webDatb[8]); LcdCmd(0x89); LcdData(webDatb[9]);LcdCmd(0x8a); LcdData(webDatb[10]); LcdCmd(0x8b); LcdData(webDatb[11]);
 LcdCmd(0x8c); LcdData(webDatb[12]); LcdCmd(0x8d); LcdData(webDatb[13]);LcdCmd(0x8e); LcdData(webDatb[14]); LcdCmd(0x8f); LcdData(webDatb[15]);
 LcdCmd(0xc0); LcdData(webDatb[16]); LcdCmd(0xc1); LcdData(webDatb[17]);LcdCmd(0xc2); LcdData(webDatb[18]);LcdCmd(0xc3);
 delay_ms(2000);

}
*/

void main() {
 ANSELA = 0;ANSELB = 0;ANSELC = 0;ANSELD = 0; ANSELE = 0;
 ANSELA.F0 = 1;ANSELA.F1 = 1;
 C1ON_bit = 0;                          // Disable comparators
 C2ON_bit = 0;
 TRISB = 0;
 LATB = 0;
 TRISA.F0 = 1;TRISA.F1 = 1;TRISA.F4 = 0;
 PORTA.F4 = 0;
 TRISD.F5 = 0;TRISD.F6 = 0;TRISD.F7 = 0;
```

```
PORTD.F5 = 0;PORTD.F6 = 0;PORTD.F7 = 0;


ADCON0 = 0X01;

ADCON1 = 0X00;

ADCON2 = 0B10101111;

INTCON = 0XC0;

T3CON = 0B00110000;

PIR2.TMR3IF = 0;

PIE2.TMR3IE = 1;

TMR3H = 0B00001011;

TMR3L = 0B11011100;


LCD_Setup();                            // Initialize LCD

WorkingLED = 1;

delay_ms(1000);

WorkingLED = 0;

delay_ms(1000);

WorkingLED = 1;

Delay_ms(200);

TMR0H = 0X00;

TMR0L = 0B00000000;

T0CON = 0X02; //Set timer 0 prescaler to a ratio of 1:8

INTCON.TMR0IF = 0;

INTCON.TMR0IE = 1; //Enable Timer 0 interrup

delay_ms(200);

UART1_Init(9600);

delay_ms(200);

lcdCmd(0x01);

lcdCmd(0x80);

LCD_WRITE("SYS. BOOTING...");

lcdCmd(0xc0);

LCD_WRITE("PLEASE WAIT");

Delay_ms(5000);

moduleInit();

PIE1.F5 = 1;

PIR1.F5 = 0;

lcdCmd(0x01);

lcdCmd(0x80);

LCD_WRITE("ALMOST DONE..");

lcdCmd(0xc0);

LCD_WRITE("PLEASE WAIT");

Delay_ms(2000);

checkBat();

delay_ms(100);

firstDisplay();
```

```
  while (1){
     if(validBatteryStatus - 48 == 0){
       firstDisplay();
       T3CON.TMR3ON = 0;
       cRly = 0;
       LowLED = 1;
       sendStatus = 48+0;

     }
     else{
       if(!T3CON.TMR3ON){
         T3CON.TMR3ON = 1;
       }
       btnControl();
       //Test();

       if(sendStatus - 48 == 1 && validBatteryStatus - 48 == 1){
          checkServer();
          sendStatus = 48+0;
          newCnt = 0;
       }
     }
  }
}
```

APPENDIX B: APPLICATION CODE (.java file)

```
package com.sight_innovation.battermonitor;

import android.content.Context;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.Toolbar;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.CompoundButton;

import android.widget.SeekBar;

import android.widget.Switch;

import android.widget.TextView;

import android.widget.Toast;


import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.io.OutputStreamWriter;

import java.io.UnsupportedEncodingException;

import java.net.HttpURLConnection;

import java.net.MalformedURLException;

import java.net.URL;


public class MainActivity extends AppCompatActivity {

    TextView bat_level, bat_status, txt_temp, txt_lvolt, txt_hvolt;

    Switch devOnOff; SeekBar seb_temp, seb_lvoltage, seb_hvoltage;

    Button save_temp, batlow_save, fullvolt_save;

    boolean isLoaded = false;


    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);


        bat_level = (TextView) findViewById(R.id.bat_level);
```

```
bat_status = (TextView) findViewById(R.id.bat_status);
txt_temp = (TextView) findViewById(R.id.txt_temp);
txt_lvolt = (TextView) findViewById(R.id.txt_lvolt);
txt_hvolt = (TextView) findViewById(R.id.txt_hvolt);
devOnOff = (Switch) findViewById(R.id.dev_on_off);
seb_temp = (SeekBar) findViewById(R.id.seb_temp);
seb_lvoltage = (SeekBar) findViewById(R.id.seb_lvoltage);
seb_hvoltage = (SeekBar) findViewById(R.id.seb_hvoltage);

seb_temp.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        txt_temp.setText("CURRENT VALUE: " + progress);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});

seb_lvoltage.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        txt_lvolt.setText("CURRENT VALUE: " + progress);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});

seb_hvoltage.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
```

```
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        txt_hvolt.setText("CURRENT VALUE: " + progress);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }
});

save_temp = (Button) findViewById(R.id.temp_save);
batlow_save = (Button) findViewById(R.id.batlow_save);
fullvolt_save = (Button) findViewById(R.id.fullvolt_save);

final Context conte = this;

save_temp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        isLoaded = true;
        String va = Integer.toString(seb_temp.getProgress() * 100);
        for(int i = va.length(); i < 5; i++){
            va = "0" + va;
        }

        serverCom("3|" + va, conte);
        //isLoaded = false;
    }
});
batlow_save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        isLoaded = true;
        String va = Integer.toString(seb_lvoltage.getProgress() * 100);
        for(int i = va.length(); i < 4; i++){
            va = "0" + va;
        }
        serverCom("5|" + va, conte);
        //isLoaded = false;
    }
```

```
    });
    fullvolt_save.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            isLoaded = true;
            String va = Integer.toString(seb_hvoltage.getProgress() * 100);
            for(int i = va.length(); i < 4; i++){
                va = "0" + va;
            }
            serverCom("4|" + va, conte);
            //serverCom("4|" + seb_hvoltage.getContext(), conte);
            //isLoaded = false;
        }
    });

    devOnOff.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            isLoaded = true;
            if (isChecked) {
                // The toggle is enabled
                serverCom("2|0", conte);
            } else {
                // The toggle is disabled
                serverCom("2|1", conte);
            }

        }
    });

    /*FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
        }
    });*/
    serverCom("1",this);

}

public void serverCom(final String datas,final Context context){
    Thread background = new Thread(new Runnable(){
        public void run(){
            final String uri = "http://batterymonitor.sightdev.net/api/software";
```

```
    final String json = "\"" + datas + "\"";
    //String json = "\"This is cool \"";
    HttpURLConnection urlConnection;
    String result = null;

    try {
        //Connect
        urlConnection = (HttpURLConnection) ((new URL(uri).openConnection()));
        urlConnection.setDoOutput(true);
        urlConnection.setRequestProperty("Content-Type", "application/json");
        urlConnection.setRequestProperty("Accept", "application/json");
        urlConnection.setRequestMethod("POST");
        urlConnection.connect();

        //Write
        OutputStream outputStream = urlConnection.getOutputStream();
        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));
        writer.write(json);
        writer.close();
        outputStream.close();

        //Read
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(urlConnection.getInputStream(), "UTF-8"));

        String line = null;
        StringBuilder sb = new StringBuilder();
        while ((line = bufferedReader.readLine()) != null) {
            sb.append(line);
        }
        bufferedReader.close();
        result = sb.toString();

    } catch (UnsupportedEncodingException e) {
        result = "'5=" + e.getMessage() + "'";
    }
    catch (MalformedURLException e){
        result = "'6=" + e.getMessage() + "'";
    }
    catch (IOException e) {
        result = "'7=Unable to connect to server. Please check your internet connectivity'";
    }

    ThreadMsg(result);
}
```

```
private  void ThreadMsg(String msg){
    if (!msg.equals(null) && !msg.equals("")) {
        Message msgObj = handler.obtainMessage();
        Bundle b = new Bundle();
        b.putString("message", msg);
        msgObj.setData(b);
        handler.sendMessage(msgObj);
    }
}

// Define the Handler that receives messages from the thread and update the progress
private final Handler handler = new Handler() {

    public void handleMessage(Message msg) {

        String aResponse = msg.getData().getString("message");

        if ((null != aResponse)) {
            // ALERT MESSAGE
            aResponse = aResponse.substring(1,aResponse.length()-1);
            String mes[] = aResponse.split("=");
            //mes[0] = mes[0].substring(1);
            //Toast.makeText(context, mes[1] + "=" + aResponse, Toast.LENGTH_LONG).show();
           if(mes[0].charAt(0)== '1'){
               //checkRely2(mes[2]);
               //Toast.makeText(view.getContext(), aResponse, Toast.LENGTH_LONG).show();
              Double va = Double.parseDouble(mes[2])/100;
               bat_level.setText("BATTERY LEVEL: " + Double.toString(va) + "V");
               if(mes[3].equals("False")){
                   bat_status.setText("CHARGING STATUS: NOT CHARGING");
               }
               else if(mes[3].equals("True")){
                   bat_status.setText("CHARGING STATUS: CHARGING");
               }
               seb_temp.setMax(100);
               seb_temp.setProgress(Integer.parseInt(mes[4])/100);
               seb_lvoltage.setMax(48);
               seb_lvoltage.setProgress(Integer.parseInt(mes[5])/100);
               seb_hvoltage.setMax(96);
               seb_hvoltage.setProgress(Integer.parseInt(mes[6])/100);
               if(mes[7].equals("False")){
                   devOnOff.setChecked(true);
               }
               else if(mes[7].equals("True")){
                   devOnOff.setChecked(false);
```

```
                    }
                    if(isLoaded){
                        Toast.makeText(context,"Value saved successfully", Toast.LENGTH_LONG).show();
                    }
                }
                else{
                    Toast.makeText(context, mes[1], Toast.LENGTH_LONG).show();
                }
                isLoaded = false;
            }
            else
            {
                // ALERT MESSAGE
                Toast.makeText(context, "Got No Response From Server.", Toast.LENGTH_LONG).show();
            }
        }
    };
  });
  background.start();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  // Inflate the menu; this adds items to the action bar if it is present.
  getMenuInflater().inflate(R.menu.menu_main, menu);
  return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  // Handle action bar item clicks here. The action bar will
  // automatically handle clicks on the Home/Up button, so long
  // as you specify a parent activity in AndroidManifest.xml.
  int id = item.getItemId();

  //noinspection SimplifiableIfStatement
  if (id == R.id.action_settings) {
    return true;
  }

  return super.onOptionsItemSelected(item);
}
}
```