BATTERY MONITORING SYSTEM FOR PLUG IN HYBRID ELECTRIC VEHICLE (PHEV)

A PROJECT REPORT

Submitted by

JEEGAN C

MUTHUMANI S

VIVEK S

VISUVASAN D

in partial fulfillment for the award of the degree
of

BACHELOR OF ENGINEERING
in
ELECTRICAL AND ELECTRONICS ENGINEERING

COLLEGE OF ENGINEERING GUINDY

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2016

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"BATTERY MONITORING SYSTEM FOR PLUG IN HYBRID ELECTRIC VEHICLE (PHEV)"** is the bonafide work of

JEEGAN C (2012104013)

MUTHUMANI S (2012104025)

VIVEK S (2012104067)

VISUVASAN D (2012104070)

who carried out the project work under my supervision.

SIGNATURE

Dr.G.Uma

HEAD OF THE DEPARTMENT

Department of Electrical and Electronics Engineering,

College of Engineering Guindy,

Anna University.

SIGNATURE

Dr.S.Chandramohan

GUIDE AND PROFESSOR

Department of Electrical and Electronics Engineering,

College of Engineering Guindy,

Anna University.

ABSTRACT

A plug-in hybrid electric vehicle (PHEV), also called a plug-in hybrid vehicle (PHV) and a plug-in hybrid, is a hybrid electric vehicle that uses rechargeable batteries, or another energy storage device, that can be recharged by plugging it in to an external source of electric power, usually a normal wall socket. Particularly important for electric vehicles is the ability to predict how much charge is left in battery at any given time, also known as its state-of-charge. Also, the common consumer problems for PHEV is checking the battery level and know when the battery level is low. And also not to go physically and check the battery level of the vehicle. This problem can be solved by remotely checking the battery level of the vehicle using GSM based technology. This helps to monitor the battery of the PHEVs without physical contact and to be reminded of when to charge the vehicle.

This can be done by measuring the battery capacity using battery parameters such as mAh and voltage and the Percentage of the battery can be calculated. The State of Charge (SOC) of the battery and health of the battery can be calculated using those parameters. This can be transmitted by a GSM module with the help of a micro-controller. The consumer can know the state of charge with a single text message at any instant.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deepest sense of gratitude to our guide **Dr. S. Chandramohan**, Professor, Department of Electrical and Electronics Engineering for his guidance, support and motivation offered to us throughout the course of the project work.

We wish to thank **Dr. G. Uma**, Head of the Department, Department of Electrical and Electronics Engineering for providing us the necessary facilities to carry out the project work.

Finally we would like to thank all others who in one way or the other helped us in the completion of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background Study

Drastic changes in the world's weather have become a popular discussion topic among the general public. In December 2009, environment ministers from 190 different countries attended the Copenhagen climate change conference to figure out methods of slowing down global warming. They agreed that anthropogenic greenhouse gases, especially carbon dioxide is one the main causes of the drastic climate changes. From the works of Herzog (2009), it is proven that road transportation is one of the main contributors to the world's carbon dioxide emissions. Therefore, global warming can be slowed down by reducing emissions from vehicles.

Due to public concern on climate changes, automobile industries are developing vehicles without any gas emissions. To produce cars without any gas emissions, the combustion engine of the car has to be replaced by electric drives. This type of vehicle is known as Plug in Hybrid Electric vehicle (PHEV), a hybrid electric vehicle that uses rechargeable batteries, or another energy storage device, that can be recharged by plugging it in to an external source of electric power.

Compared to conventional vehicles, PHEVs produce less air pollution locally and require less petroleum. PHEVs may produce less in the way of greenhouse gases, which contribute to global warming, than conventional vehicles do. PHEVs also eliminate the problem of range anxiety associated with all-electric vehicles, because the combustion engine works as a backup when the batteries are depleted, giving PHEVs driving-range comparable to that of other vehicles that have gasoline and diesel tanks.

1.2 Problem Statement

The battery management system is one of the main components of the hybrid electric vehicle is the battery system. The battery system provides electric supply to power up the hybrid electric vehicle. Since it is important, the studies include the parameters of battery, information such as battery voltage, battery current and State of Charge (SOC) of the battery must be displayed for the view of user at any place at any instant. With the vast improvement in technology, it is possible to design a user interface to display the electric vehicle's battery information on the consumer's mobile phone using a GSM module.

1.3 Objectives of Research

The main objectives of this project is to estimate the SOC of hybrid electric vehicle battery and display the vehicle battery information through a GSM module for the consumer. Circuits developed to transfer signal from the Hybrid electric vehicle's battery to Arduino. Then, investigate the feasibility of GSM module to transmit output data.

1.4 Research Scopes

Several scopes had been fixed in order to achieve the objectives of the project.

The scopes of this project can be divided into four parts. The scopes of the study are as follows:

1) The battery parameters

2) Battery testing using a 12V Lead Acid battery with nominal capacity of 1.3Ah each.

3) Voltage divider circuits that have to be designed to obtain battery voltage.

4) Information to be displayed on the user interface are: battery voltage and State of Charge (SOC), Remaining Capacity, Discharge time and Used Capacity.

1.5 Outlines of Project

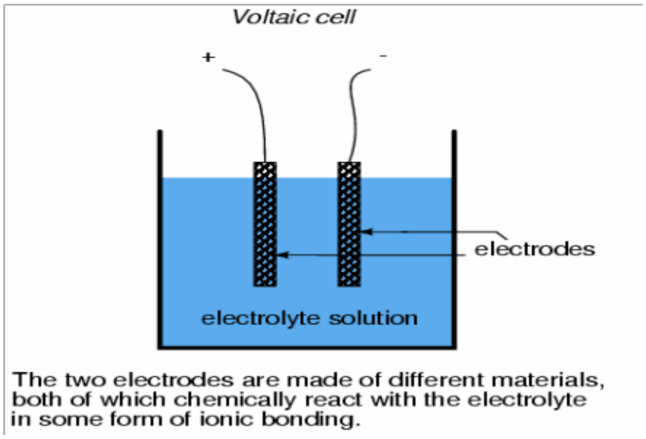
This project consists of five chapters. The background of study, problem statement, objective, scope and outline of the project of this project will be discussed in Chapter 1. Literature reviews on related works and theories of the project including Lead acid cell discharge characteristics and Battery Voltage Measuring Method are done in chapter 2. The methodology on the hardware and software implementation to achieve the objective of this project is described in chapter 3. The results and discussions on the design and performance of the user interface will be presented in chapter 4. Lastly in chapter 5, a conclusion will be given along with suggestions to improve this work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

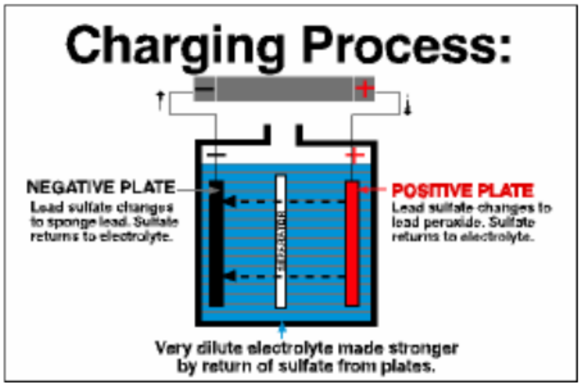
Battery is a device that converts electrical energy. The most popular batteries for electrical energy storage due to ability to provide fast response to energy demand is rechargeable battery. For EV battery system, The battery system of EV consists of small groups of cells connected in series, termed as modules. Several battery modules are then connected in series to form the battery system, which the cells have positive and negative electrodes joined by an electrolyte. There are lot of kinds of battery which have different parameters because have different chemical compound in the battery.


Voltaic cell
+
-
electrodes
electrolyte solution
The two electrodes are made of different materials, both of which chemically react with the electrolyte in some form of ionic bonding.

2.1 Battery cell immersed into electrolyte

2.2 Parameters of Battery

A basic understanding of the battery chemistry is very important. Batteries can be divided into different categories. Different types of battery have different battery chemistry. Batteries that are used in EV are the rechargeable secondary cell type such as Lead Acid battery. At the negative plate, the battery has spongy Lead as the active material, while Lead Oxide is as the active material on positive plate. These both plates are immersed in an electrolyte of Sulphuric Acid.


Charging Process:
−
+
−
+
NEGATIVE PLATE
Lead sulfate changes to sponge lead. Sulfate returns to electrolyte.
POSITIVE PLATE
Lead sulfate changes to lead peroxide. Sulfate returns to electrolyte.
SEPARATOR
Very dilute electrolyte made stronger by return of sulfate from plates.

2.2 Charging process for Lead Acid battery

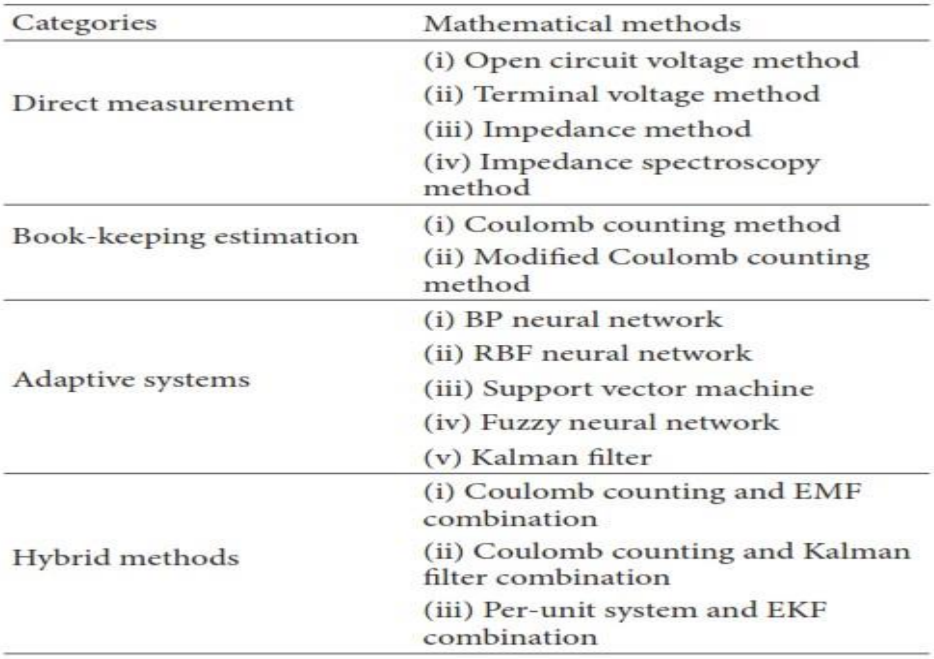
The battery can be treated as a ‘black box’ which has a range of performance criteria. The criteria include cell and battery voltages, specific energy, specific power, energy efficiency and operating temperature. The next subtopic will discuss deeply on the criteria.

2.2.1 Lead Acid Battery

Early 1990, Lead Acid battery has been used in electric cars. It consists of a lead-dioxide cathode, sponge metallic anode and sulphuric acid as the electrolyte. Lead Acid battery is the most common battery sales which is about 40-45% of the global battery sales. Furthermore, it’s also variety of sizes and designs and available in large quantities. They are manufactured in smaller capacity from 1Ah up to several thousand Ah. The discharge curve is steeper , meaning more accurate voltage measurement and SOC calculation. The special characteristic for lead acid battery is decomposing very slowly as the lead and Lead Oxide is not stable in Sulphuric Acid. This battery will go to a process of sulphating when the battery is left in a discharge state for a long period. Like other batteries the capacity and efficiency of the lead acid battery will be reduced at low temperature.

2.3 SOC Estimation

The SOC of battery is an important parameter for controlling strategy and used to describe its remaining capacity. The SOC estimation will affect the life of expectancy battery, which protect battery and prevent over discharge. The SOC estimation can be done using the following methods.



Categories	Mathematical methods
Direct measurement	(i) Open circuit voltage method (ii) Terminal voltage method (iii) Impedance method (iv) Impedance spectroscopy method
Book-keeping estimation	(i) Coulomb counting method (ii) Modified Coulomb counting method
Adaptive systems	(i) BP neural network (ii) RBF neural network (iii) Support vector machine (iv) Fuzzy neural network (v) Kalman filter
Hybrid methods	(i) Coulomb counting and EMF combination (ii) Coulomb counting and Kalman filter combination (iii) Per-unit system and EKF combination

2.1 Classification of SOC estimating

2.3.1 Open-Circuit Voltage Method

The various values of open circuit voltage (OCV) of the Depth-of-Charge can be obtained by direct measured in a separate experiment. The OCV method is in category of direct measurement because it refers to some physical battery properties. In this experiment, the battery is charged with a constant current to a specific Depth-of-Charge, the current is then interrupted and the battery is allowed to rest for a certain period of time. OCV of batteries is proportional to the SOC for a long period. All batteries are different OCV and SOC relationship.

2.3.2 Coulomb Counting Method

This method, monitoring and memorizing the currents flowing into and out from a battery for long time. It was impractical SOC estimation but critical in verifying the accuracy of estimated results from other methods. The factors of battery history, cycle life, temperature and discharge current will affect the accuracy of coulomb counting method. Therefore, by using this method, user needs some precaution.

2.4 Arduino

Arduino is a microcontroller board that can be modified to perform a desired function such as it can be modified to process information got from the input ports before sending it to a joined output ports. It is regularly utilized as a part of an installed framework where it goes about as a center man to get process and exchange information from a source to a output ports. By and large, the Arduino board comprises of a processor, power supply, input/output ports, USB port and some connectors. The microcontroller chip utilized on an Arduino board is an Atmel based Microcontroller. A voltage controller is accessible on the Arduino board. The capacity of this voltage controller is to change over outside source voltage from the scope of 10.5V-12.7V to a controlled 5V DC voltage. This 5V will be utilized as the power source of the Arduino circuit board .

The quantity of I/O ports accessible in an Arduino board varies for every model. In the Arduino load up, there is a gem oscillator that creates a clock beat for the procedure of the chip. The rate of the microcontroller in executing a guideline is taking into account the recurrence of the clock beat. The USB port gives a method for correspondence between the Arduino board with the PC for programming purposes. In addition, the USB port offers another method for fueling on the Arduino board as the 5V voltage needed to power up the Arduino can be supplied to the Arduino board specifically from the PC.

CHAPTER 3

PROJECT METHODOLOGY

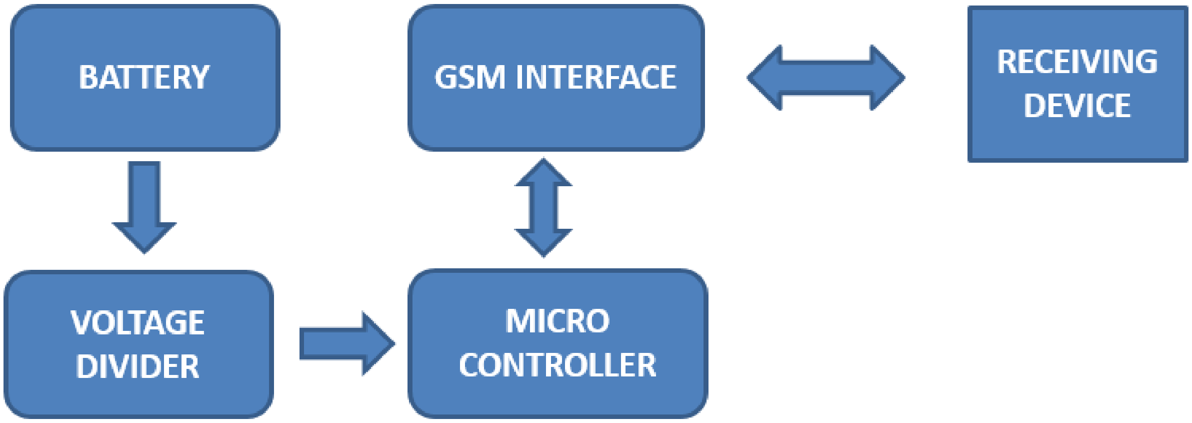
3.1 INTRODUCTION

This chapter will start by discussing the overall concept or methodology that will be used in this project. Next, is the discussion about the measurement of voltage. Besides, the estimation SOC will be detailed on how it can be calculated. This methodology chapter is to detail about the project work on the calculation, about the process and something needed to design like voltage divider circuit. Furthermore, the Arduino programming is explained.

3.2 PROJECT CONCEPT

Firstly, the concept of electric vehicle is shall to understand clearly. All basic concept of electric vehicle that related to the development of EV was collected from reference books, previous research and so on that have relation to the project. Particularly important for electric vehicles is the ability to predict how much charge is left in battery at any given time, also known as its state-of-charge. Also, the common consumer problems for PHEV is checking the battery level and know when the battery level is low. And also not to go physically and check the battery level of the vehicle. This problem can be solved by remotely checking the battery level of the vehicle using GSM based technology. This helps to monitor the battery of the PHEVs without physical contact and to be reminded of when to charge the vehicle. The Battery voltage is calculated by Arduino using voltage divider circuit and the parameters such as SOC, Used capacity, Remaining capacity and the discharge time are calculated.

The calculated battery parameters are transferred with the help of GSM module. The Basic block diagram of the project is given below.



3.1 Basic block diagram

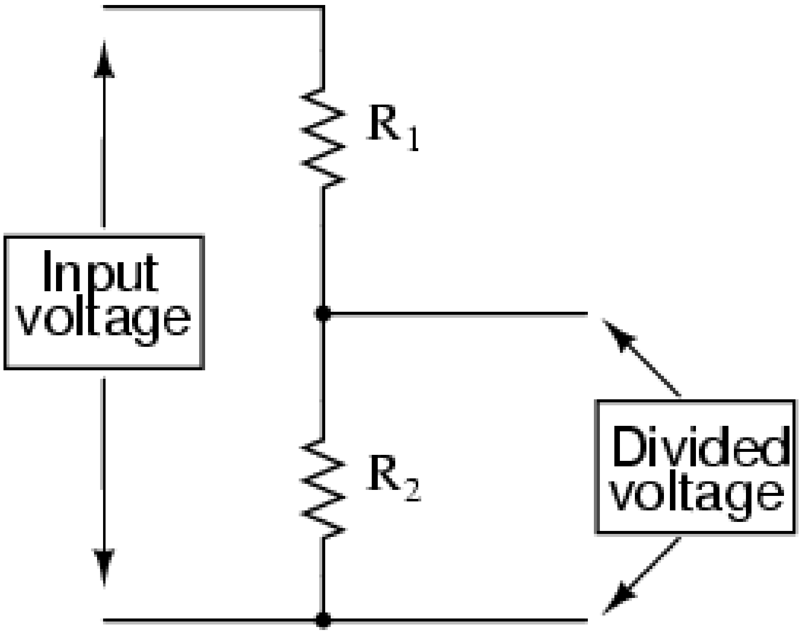
3.3 CIRCUIT COMPONENTS AND DESIGN

The designed block diagram to obtain the battery parameters is shown in the above diagram. The different components used in the project are explained below. A Simple 12V/1.3Ah battery is used in the project and the diagram of the battery is shown below. The Voltage of the battery is divided using the voltage divider circuit since the Arduino can measure only up to 5V. The voltage is reduced by using the appropriate resistors. The input voltage is found out using the formula

$$\mathbf{Vin = \quad Vout \quad (R2 / (R1 + R2))}$$

Amptek
Sealed Rechargeable Battery
SEALED LEAD BATTERY MUST BE RECYCLED OR DISPOSED OF PROPERLY
High Service Life
CE
DO NOT SHORT CIRCUIT
RECHARGE AFTER USE

3.2 12V battery

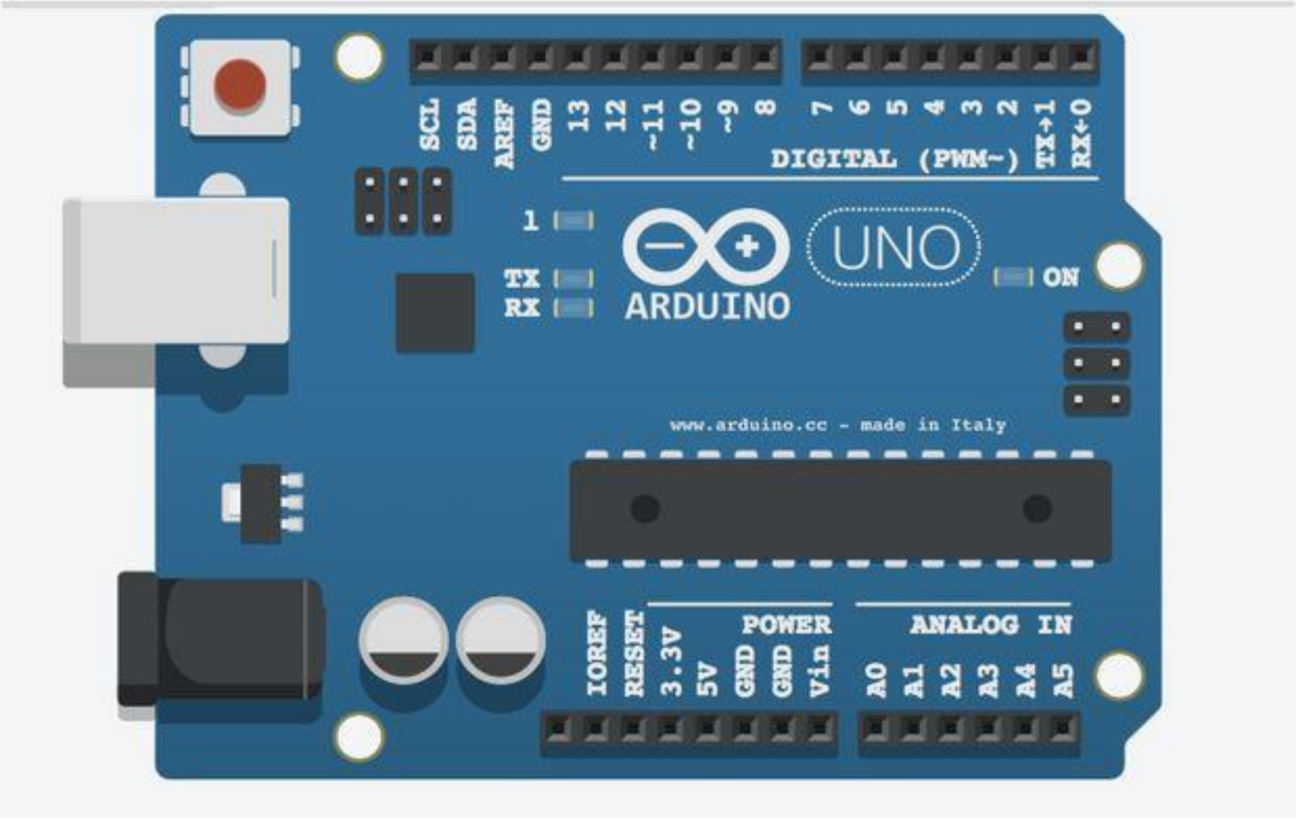


R1 = 10k ohm. **R2 = 5.6k ohm.**

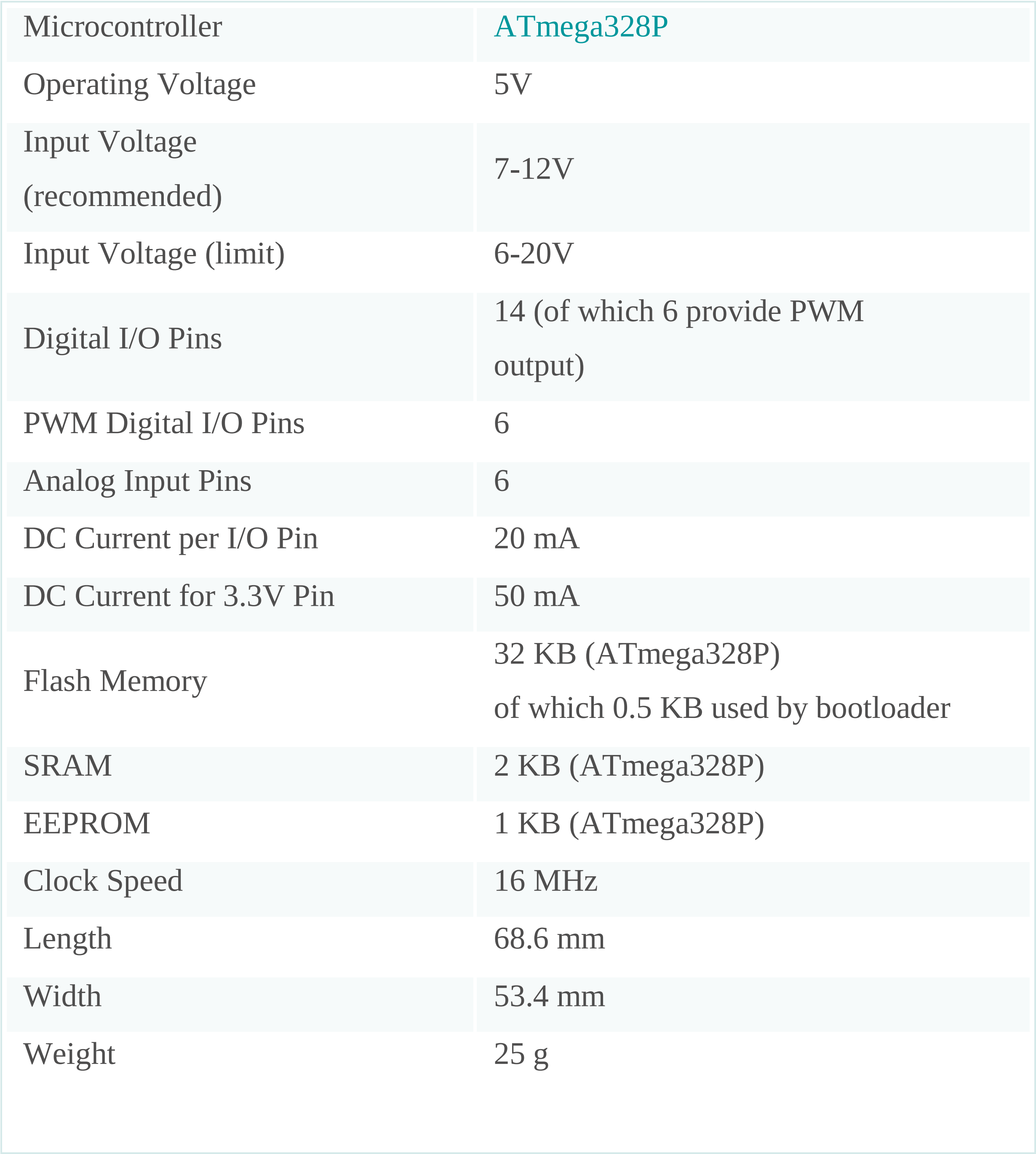
3.3 Voltage divider Circuit

3.3.2 Arduino

The Uno is a microcontroller board. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again. The diagram of the Arduino Uno board is given below.



3.4 ARDUINO UNO



Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

3.1 Technical Specifications of Arduino Uno

3.3.3 Programming

The Uno can be programmed with the Arduino Software (IDE). Select "Arduino/Genuino Uno" from the Tools > Board menu (according to the microcontroller on your board).

The ATmega328 on the Uno comes preprogrammed with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

3.3.4 Power

The Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The power pins are as follows:

- **Vin**. The input voltage to the Uno board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- **5V**.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board.

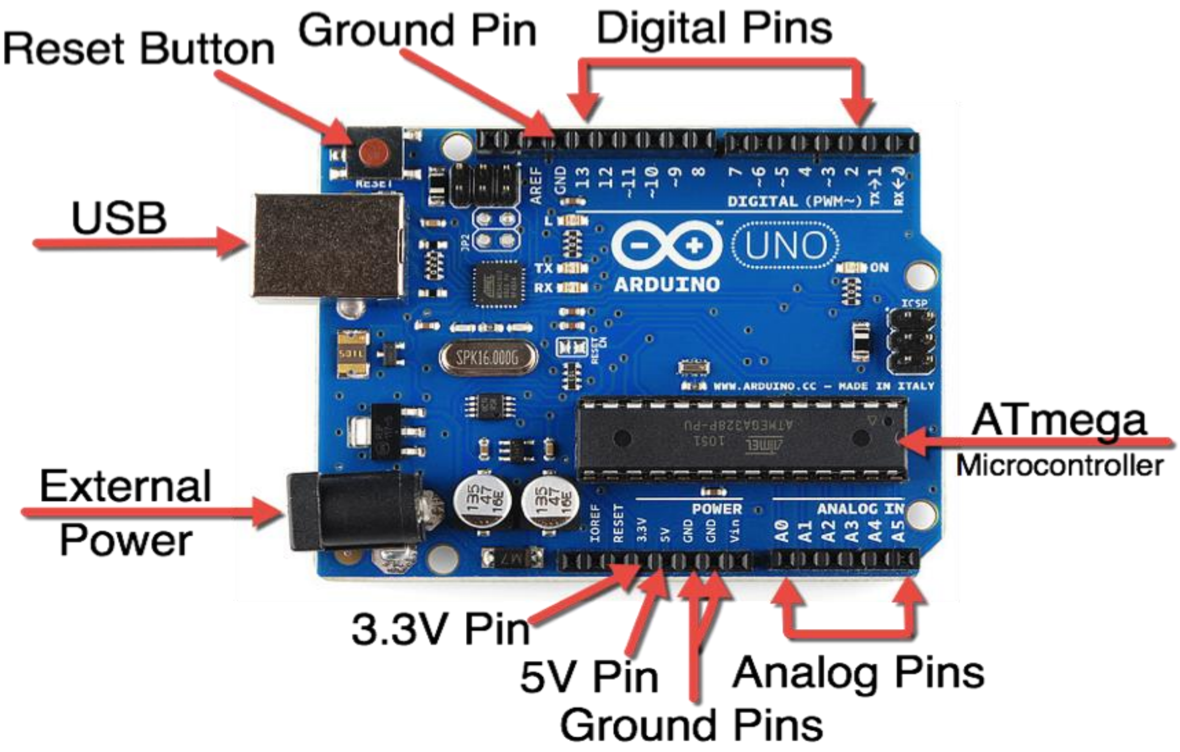
- **3V3**. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

- **GND**. Ground pins.

- **IOREF**. This pin on the Uno board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

3.3.5 Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

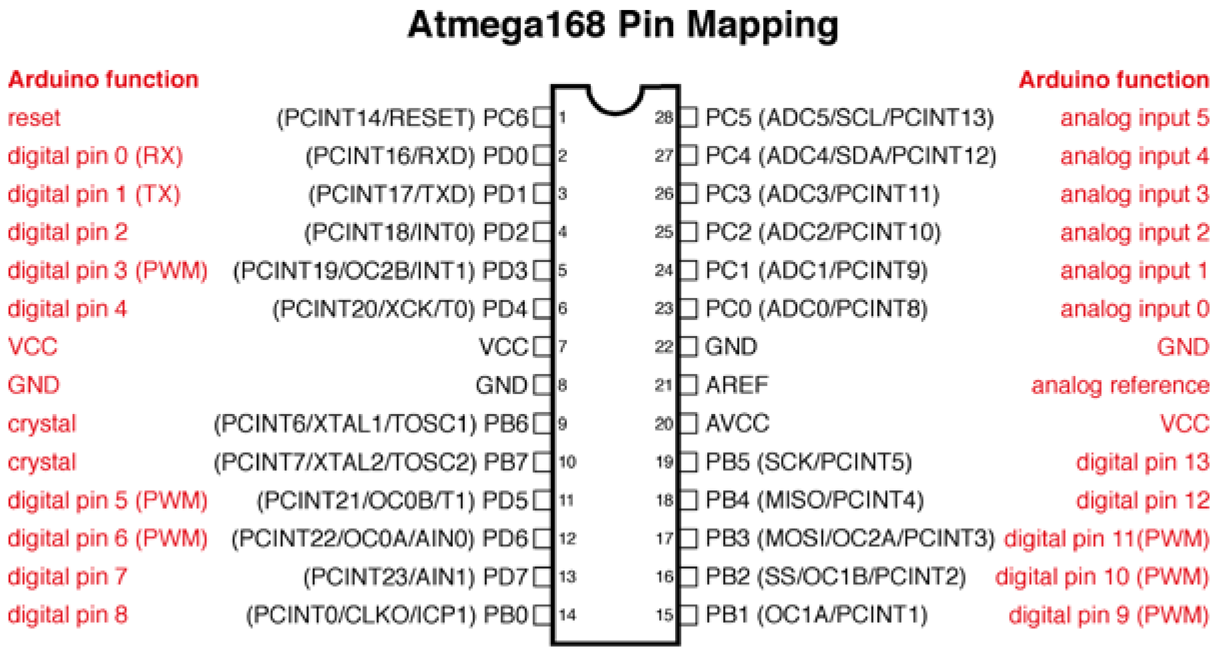


3.5 ARDUINO PARTS

3.3.5 Input and Output

See the mapping between Arduino pins and ATmega328P ports below. The mapping for the Atmega8, 168, and 328 is identical.

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(),digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.



Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

In addition, some pins have specialized functions:

- **Serial**: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts**: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM**: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

- **SPI**: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- **LED**: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **TWI**: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.

There are a couple of other pins on the board:
AREF. Reference voltage for the analog inputs. Used with analogReference().
Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

3.3.6 Communication

The Uno has a number of facilities for communicating with a computer, another Uno board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer.

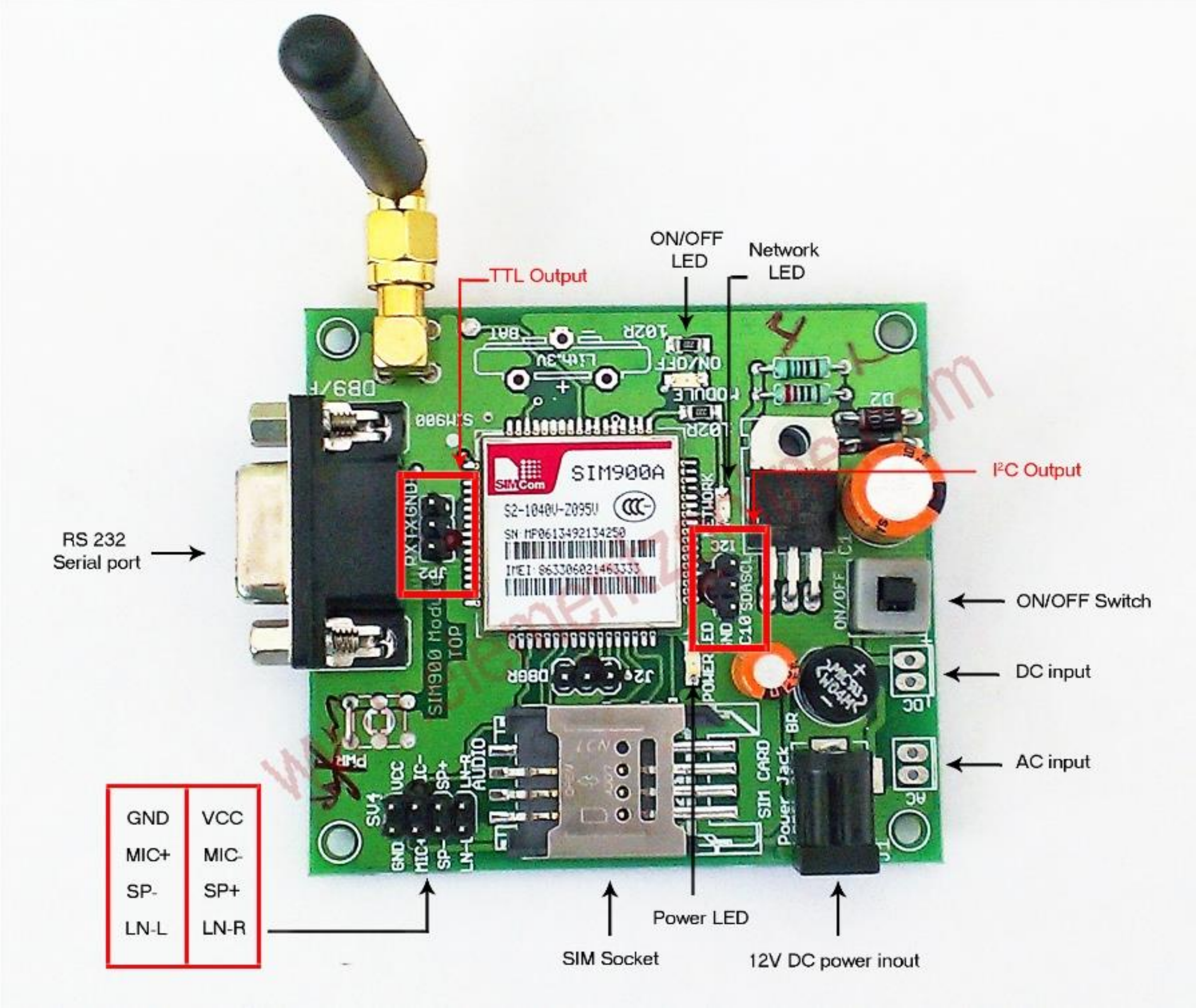
The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required.

The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

3.4 GSM MODULE

A **GSM Module** is basically a GSM Modem (like SIM 900) connected to a PCB with different types of output taken from the board – say TTL Output (for Arduino, 8051 and other microcontrollers) and RS232 Output to interface directly with a PC (personal computer). The board will also have pins or provisions to attach mic and speaker, to take out +5V or other values of power and ground connections. These type of provisions vary with different modules.

Lots of varieties of GSM modem and GSM Modules are available in the market to choose from. For our project of connecting a gsm modem or module to arduino and hence send and receive sms using arduino – its always good to choose an **arduino compatible GSM Module** – that is a GSM module with TTL Output provisions .We were using SIM900a GSM Module for our project.



3.6 GSM 900A MODULE

SIM900 GSM Module supports communication in 900MHz band. We are from India and most of the mobile network providers in this country operate in the 900 MHz band. If you are from another country, you have to check the mobile network band in your area.

GSM modules are manufactured by different companies. They all have different input power supply specs. You need to double check your GSM modules power requirements.

3.4.1 Booting the GSM module

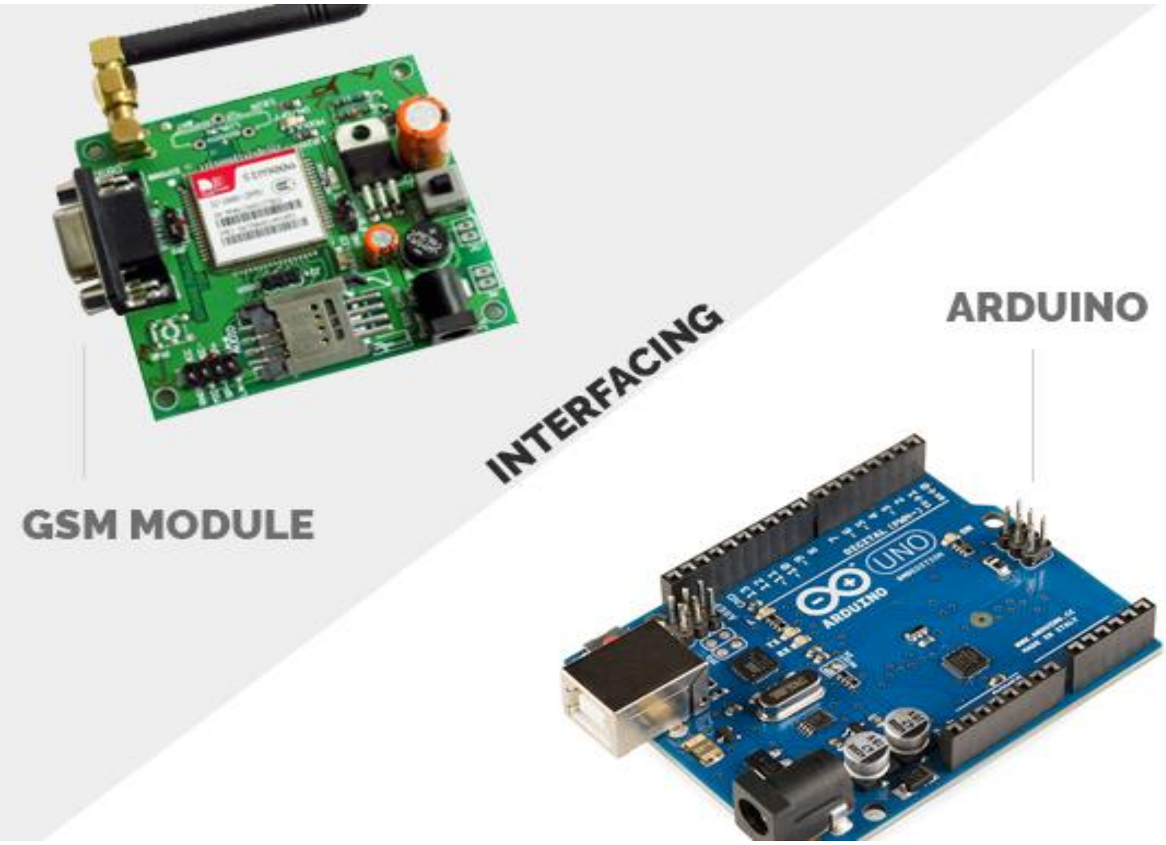
- Insert the SIM card to GSM module and lock it.
- Connect the adapter to GSM module and turn it ON!
- Now wait for some time (say 1 minute) and see the blinking rate of 'status LED' or 'network LED' (GSM module will take some time to establish connection with mobile network)
- Once the connection is established successfully, the status/network LED will blink continuously every 3 seconds. You may try making a call to the mobile number of the sim card inside GSM module. If you hear a ring back, the gsm module has successfully established network connection.

3.5 Interfacing GSM Module with Arduino

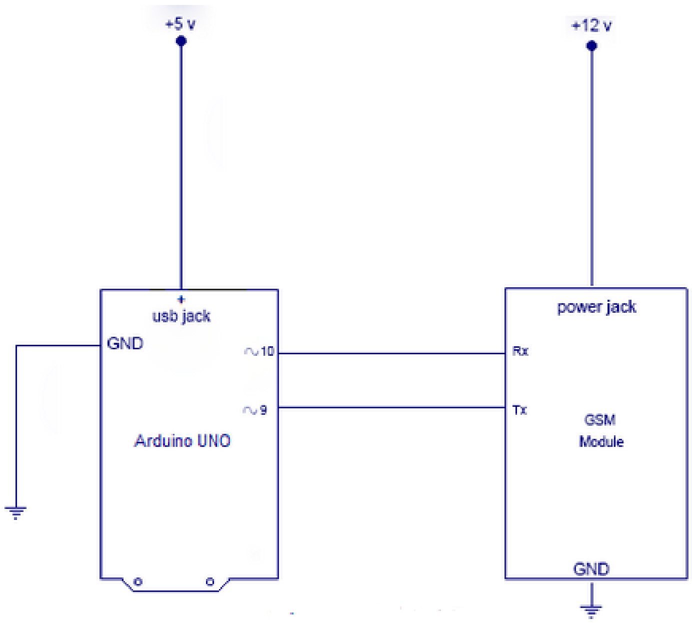
There are two ways of connecting GSM module to arduino. In any case, the communication between Arduino and GSM module is serial. So we are supposed to use serial pins of Arduino (Rx and Tx). So if you are going with this method, you may connect the Tx pin of GSM module to Rx pin of Arduino and Rx pin of GSM module to Tx pin of Arduino. Now connect the ground pin of arduino to ground pin of gsm module.You made 3 connections and the wiring is over. Now you can load different programs to communicate with gsm module and make it work.

The problem with this connection is that, while programming Arduino uses serial ports to load program from the Arduino IDE. If these pins are used in wiring, the program will not be loaded successfully to Arduino. So you have to disconnect wiring in Rx and Tx each time you burn the program to arduino. Once the program is loaded successfully, you can reconnect these pins and have the system working.

To avoid this difficulty, I am using an alternate method in which two digital pins of Arduino are used for serial communication. We need to select two **PWM enabled pins of Arduino** for this method. So I choose pins **9** and **10** (which are PWM enabled pins). This method is made possible with the **Software_Serial Library** of Arduino. Software Serial is a library of Arduino which enables serial data communication through other digital pins of Arduino. The library replicates hardware functions and handles the task of serial communication.



Given above is the circuit diagram to connect gsm module to arduino – and hence use the circuit to send sms and receive sms using arduino and gsm modem.



3.7 CIRCUIT

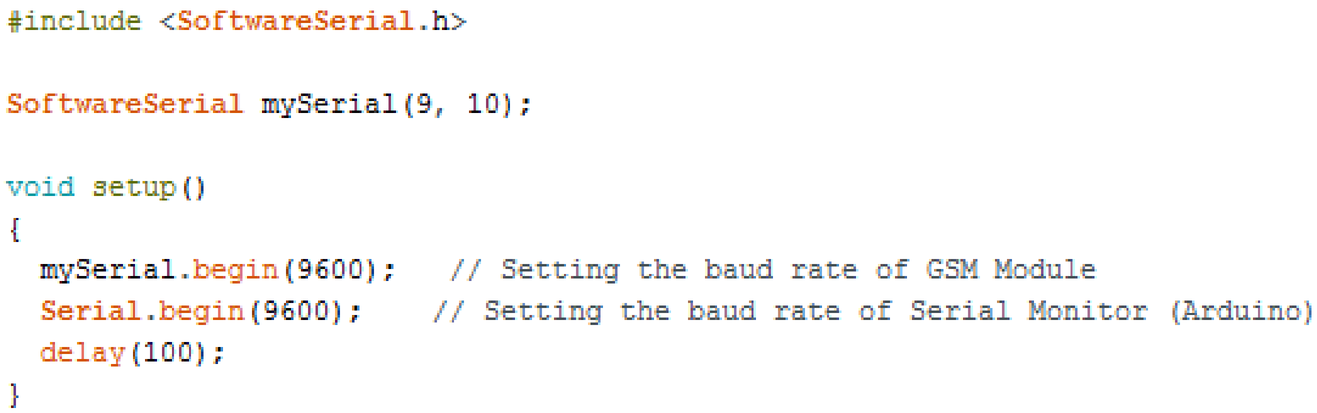
3.5.1 PROGRAM TO INTERFACE GSM WITH ARDUINO

The program has two objectives as described below:-

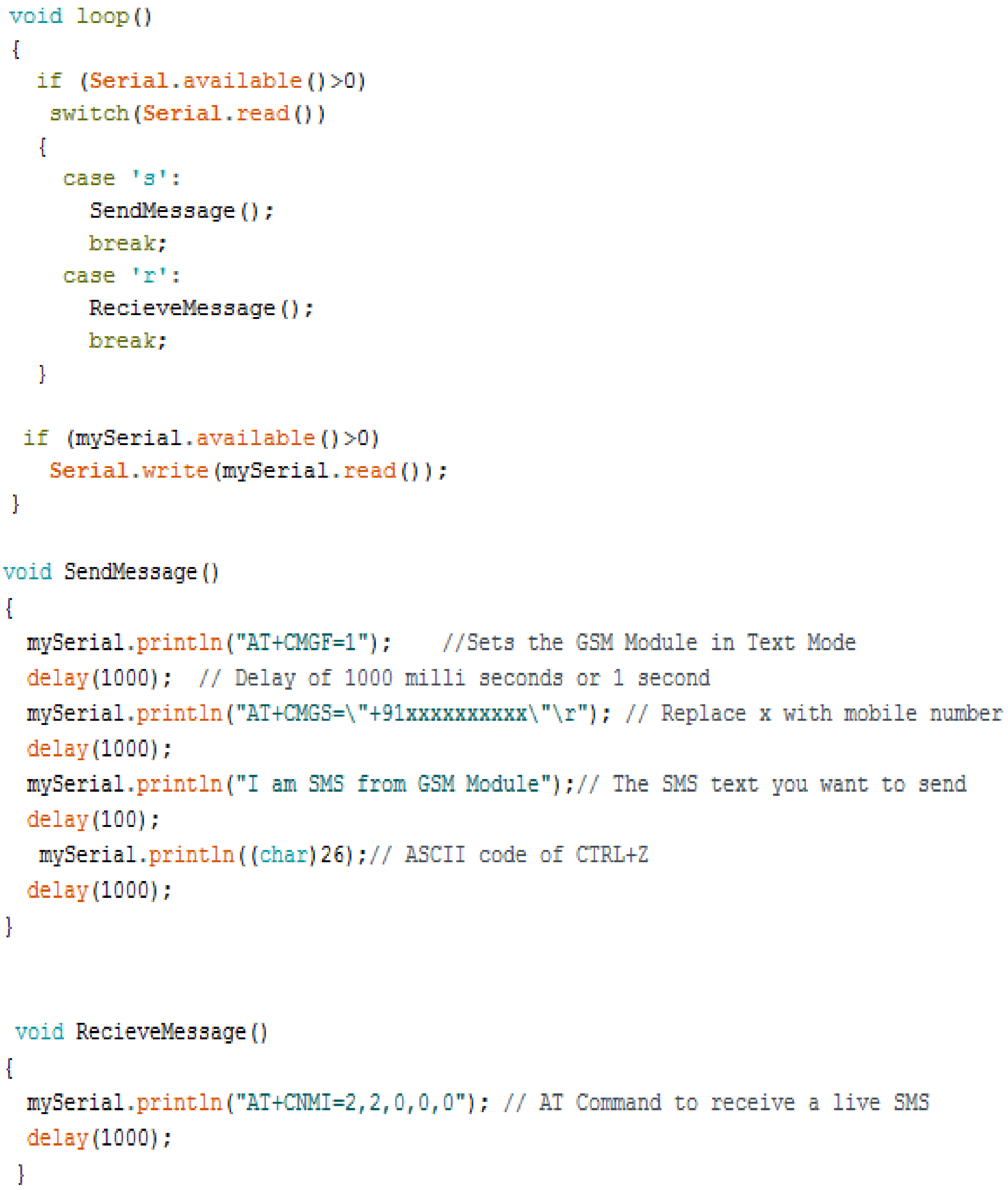
1) Send SMS using Arduino and GSM Module – to a specified mobile number inside the program.

2) Receive SMS using Arduino and GSM Module – to the SIM card loaded in the GSM Module.

The program/code to make Arduino send sms and receive sms using gsm module is given below:



```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(9, 10);

void setup()
{
  mySerial.begin(9600);   // Setting the baud rate of GSM Module
  Serial.begin(9600);    // Setting the baud rate of Serial Monitor (Arduino)
  delay(100);
}
```



```
void loop()
{
  if (Serial.available()>0)
   switch(Serial.read())
  {
    case 's':
      SendMessage();
      break;
    case 'r':
      RecieveMessage();
      break;
  }

 if (mySerial.available()>0)
   Serial.write(mySerial.read());
}

void SendMessage()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+91xxxxxxxxxx\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("I am SMS from GSM Module");// The SMS text you want to send
  delay(100);
   mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}


 void RecieveMessage()
{
  mySerial.println("AT+CNMI=2,2,0,0,0"); // AT Command to receive a live SMS
  delay(1000);
 }
```

3.5.2 PROGRAM EXPLANATION

We begin by including SoftwareSerial library into the program. In the next line, we create a constructor of Software Serial with name mySerial and we pass the digital pin numbers as parameters. The actual format is like Software Serial mySerial (Rx, Tx);

So in our code, pin number 9 will act as Rx of Arduino and 10 will act as Tx of Arduino. Lets get to the configuration part of program inside setup. The first task is to set baud rates of SoftwareSerial library to communicate with GSM module. We achieve this by invoking mySerial.begin function. Our second task is to set the baud rate of Arduino IDE's Serial Monitor. We do this by invoking Serial.begin function. Both should be set at the same baud rate and we use 9600 bits/second here. Configuration part is over with setting baud rates and its good to give a small delay of 100 milli seconds. Some of the functions used are given below.

- **Serial.available()** – checks for any data coming through serial port of arduino. The function returns the number of bytes available to read from serial buffer. If there is no data available, it returns a -1 (value less than zero).
- **Serial.read()** – Reads all the data available on serial buffer (or incoming serial data if put otherwise). Returns the first byte of incoming serial data.
- **mySerial.available()** – checks for any data coming from GSM module through the SoftwareSerial pins 9 and 10. Returns the number of bytes available to read from software serial port. Returns a -1 if no data is available to read.
- **mySerial.read()** – Reads the incoming data through software serial port.

- **Serial.write()** – Prints data to serial monitor of arduino. So the function Serial.write(mySerial.read()) – prints the data collected from software serial port to serial monitor of arduino.

SendMessage() and **RecieveMessage()**

These are the functions in which we actually send commands to GSM module from Arduino. These commands to communicate with GSM module are called AT Commands. There are different commands to perform different tasks using the GSM module.

- **SendMessage()** – is the function we created in our arduino sketch to send an SMS. To send an SMS, we should set our GSM module to Text mode first. This is achieved by sending an AT Command "AT+CMGF=1″ We send this command by writing this to SoftwareSerial port. To achieve this we use the mySerial.println() function. mySerial.println writes data to software serial port (the Tx pin of our Software Serial – that is pin 10) and this will be captured by GSM module (through its Rx pin). After setting the GSM module to Text mode, we should the the mobile number to which we shall send the SMS. This is achieved with AT command "AT+CMGS=\"+91xxxxxxxxxx\"\r" – where you may replace all x with the mobile number.

In next step, we should send the actual content of SMS. The end of SMS content is identified with CTRL+Z symbol. The ASCII value of this CTRL+Z is 26. So we send a char(26) to GSM module using the line mySerial.println((char)26); Each and every AT command may be followed by 1 second delay. We must give some time for GSM module to respond properly. Once these commands are send to GSM module, you shall receive an SMS in the set mobile number.

- **RecieveMessage()** – is the function to receive an SMS (a live SMS). The AT command to receive a live SMS is “AT+CNMI=2,2,0,0,0″ – we just need to send this command to GSM module and apply a 1 second delay. Once you send this command, try sending an SMS to the SIM card number put inside GSM module. You will see the SMS you had sent displayed on your Arduino serial monitor.

There are different AT commands for different tasks. If you want to read all SMS’s stored in your SIM card, send the following AT Command to gsm module – “AT+CMGL=\”ALL\”\r”

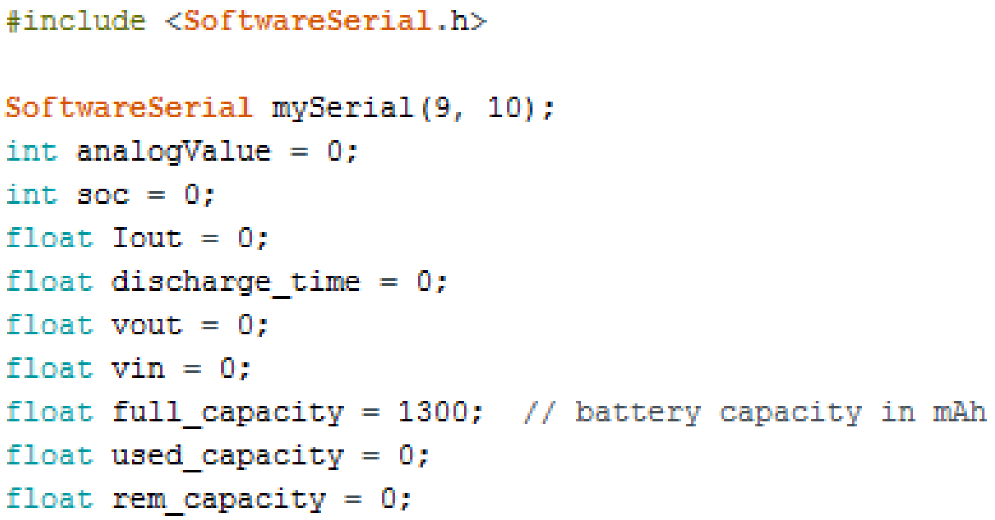
3.5.3 AT Commands to Send SMS using Arduino and GSM Module

```
AT+CMGF=1 // Set the GSM module in text mode

AT+CMGS=\"+YYxxxxxxxxxx\"\r // Input the mobile number| YY is country co
de

“the message” with stopping character (char)26 // ASCII of ctrl+z
```

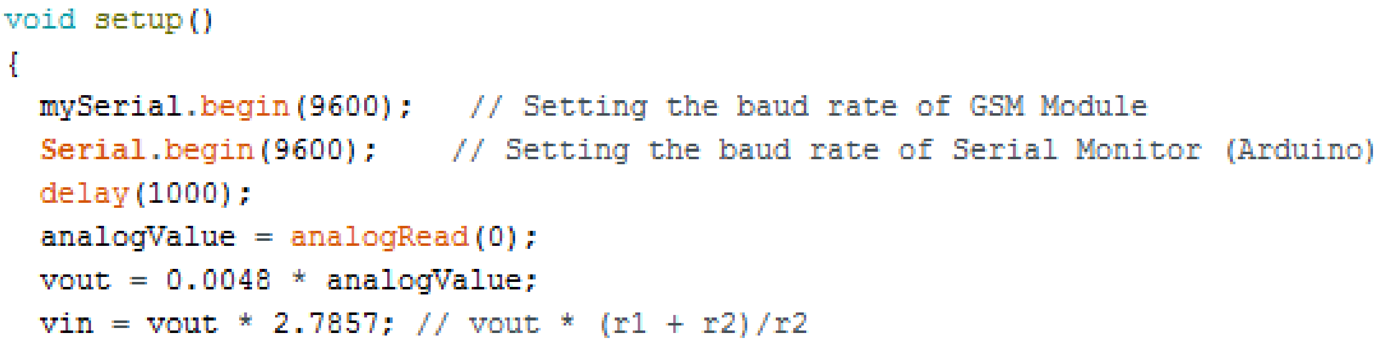
3.5.4 AT Commands to Receive SMS using Arduino and GSM Module

```
AT+CMGF=1 // Set the GSM Module in text mode

AT+CNMI=2,2,0,0,0 // AT Command to receive live sms
```

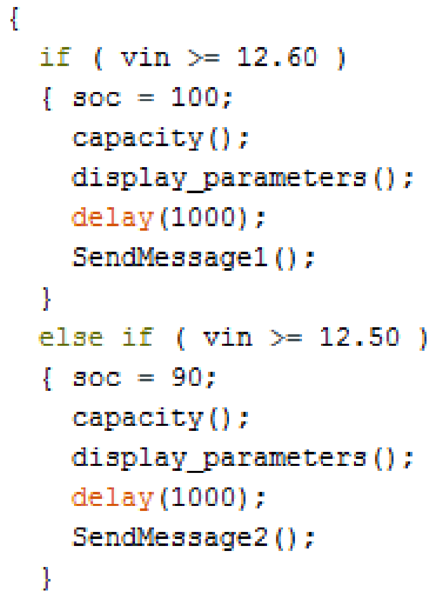
ACTUAL PROGRAM



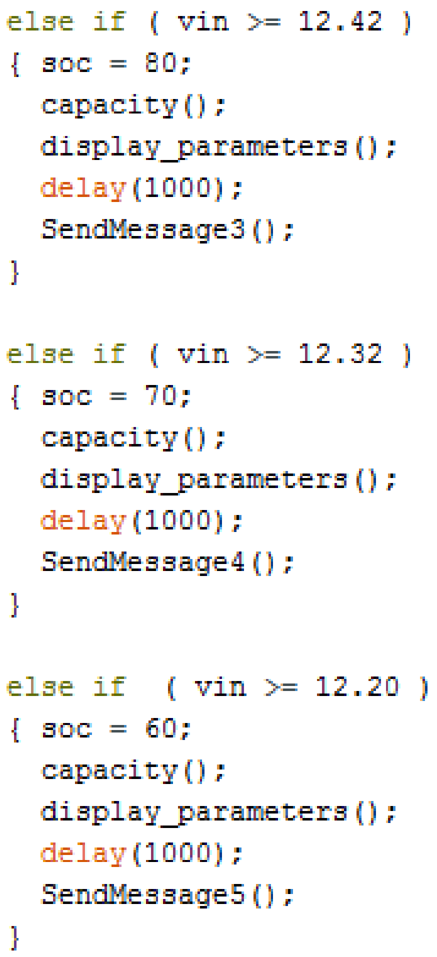
```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(9, 10);
int analogValue = 0;
int soc = 0;
float Iout = 0;
float discharge_time = 0;
float vout = 0;
float vin = 0;
float full_capacity = 1300;  // battery capacity in mAh
float used_capacity = 0;
float rem_capacity = 0;
```



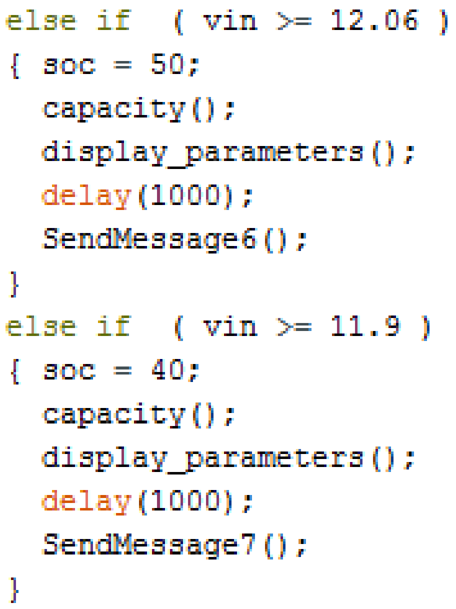
```
void setup()
{
  mySerial.begin(9600);   // Setting the baud rate of GSM Module
  Serial.begin(9600);    // Setting the baud rate of Serial Monitor (Arduino)
  delay(1000);
  analogValue = analogRead(0);
  vout = 0.0048 * analogValue;
  vin = vout * 2.7857; // vout * (r1 + r2)/r2
```



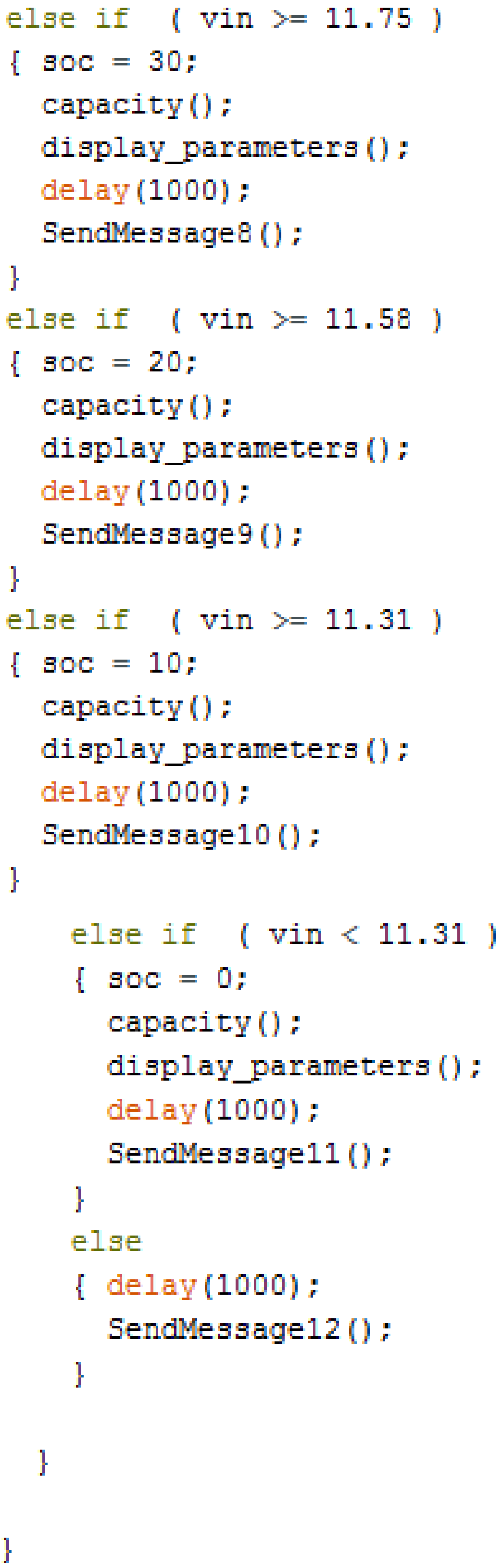
```
{
  if ( vin >= 12.60 )
  { soc = 100;
    capacity();
    display_parameters();
    delay(1000);
    SendMessage1();
  }
  else if ( vin >= 12.50 )
  { soc = 90;
    capacity();
    display_parameters();
    delay(1000);
    SendMessage2();
  }
```



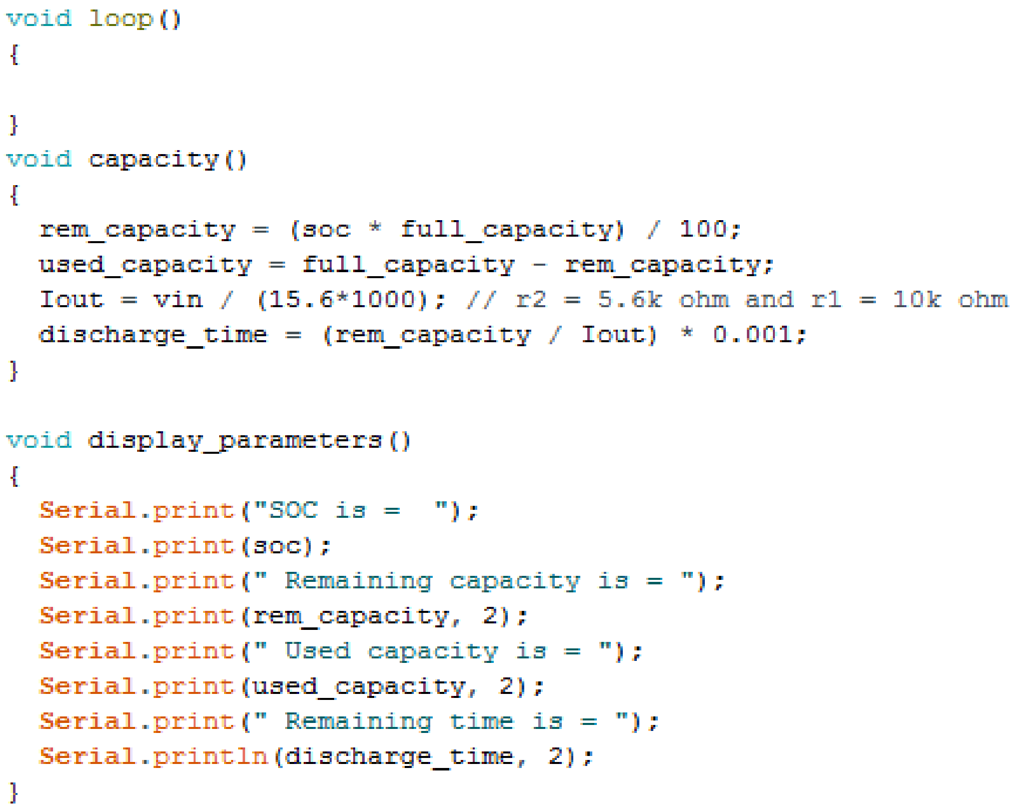
```
else if ( vin >= 12.42 )
{ soc = 80;
  capacity();
  display_parameters();
  delay(1000);
  SendMessage3();
}

else if ( vin >= 12.32 )
{ soc = 70;
  capacity();
  display_parameters();
  delay(1000);
  SendMessage4();
}

else if  ( vin >= 12.20 )
{ soc = 60;
  capacity();
  display_parameters();
  delay(1000);
  SendMessage5();
}
```



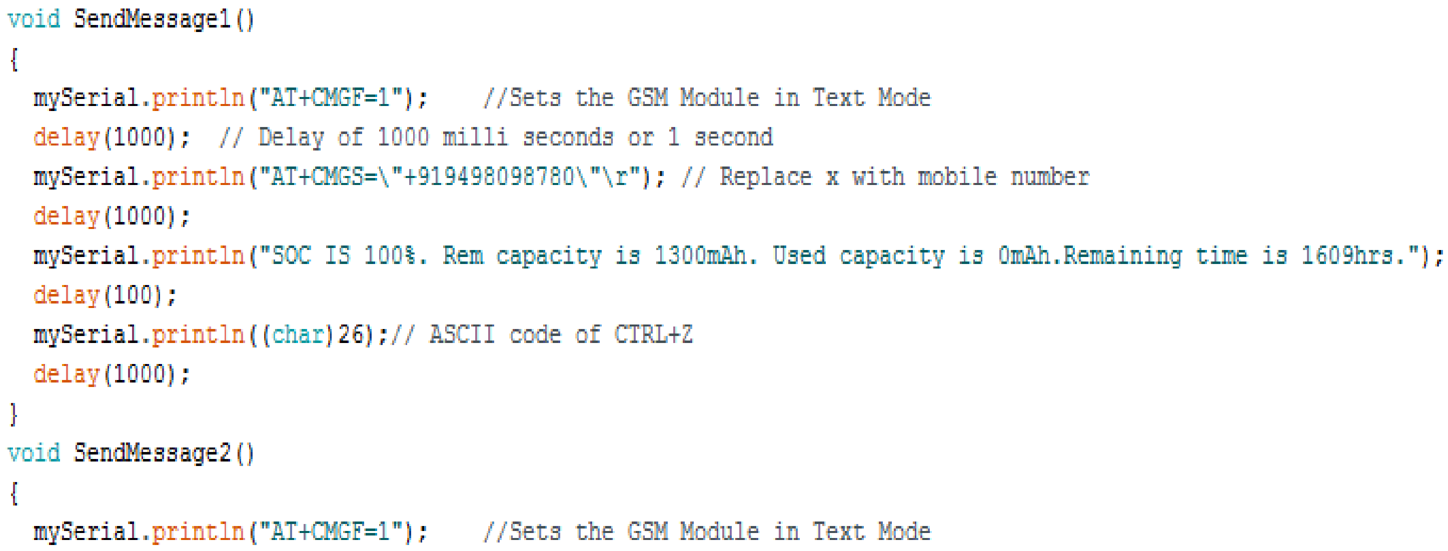
```
else if  ( vin >= 12.06 )
{ soc = 50;
  capacity();
  display_parameters();
  delay(1000);
  SendMessage6();
}
else if  ( vin >= 11.9 )
{ soc = 40;
  capacity();
  display_parameters();
  delay(1000);
  SendMessage7();
}
```



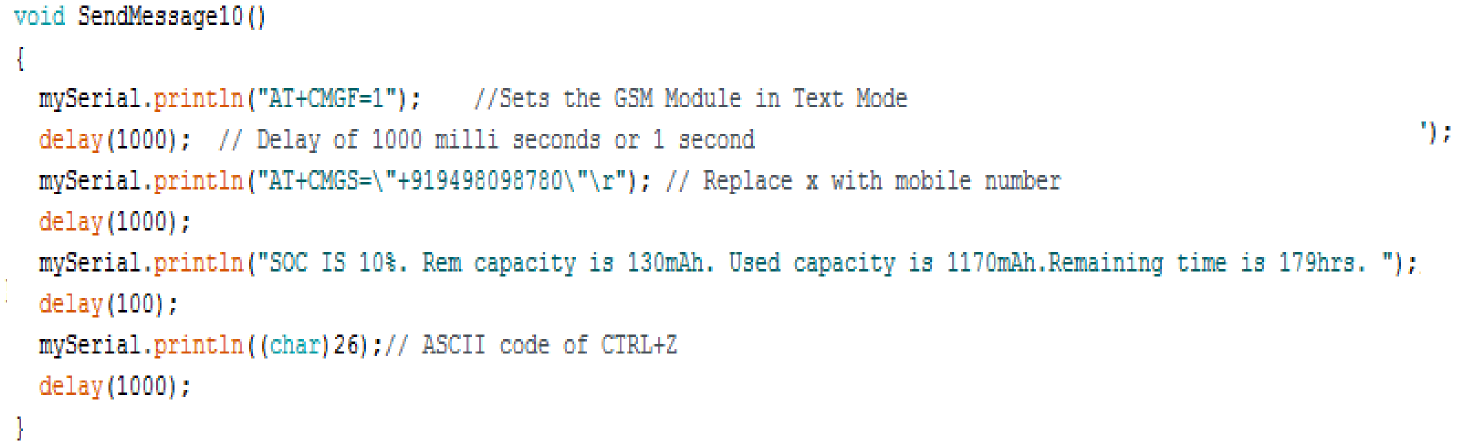
```
else if  ( vin >= 11.75 )
{ soc = 30;
  capacity();
  display_parameters();
  delay(1000);
  SendMessage8();
}
else if  ( vin >= 11.58 )
{ soc = 20;
  capacity();
  display_parameters();
  delay(1000);
  SendMessage9();
}
else if  ( vin >= 11.31 )
{ soc = 10;
  capacity();
  display_parameters();
  delay(1000);
  SendMessage10();
}
    else if  ( vin < 11.31 )
    { soc = 0;
      capacity();
      display_parameters();
      delay(1000);
      SendMessage11();
    }
    else
    { delay(1000);
      SendMessage12();
    }

  }

}
```



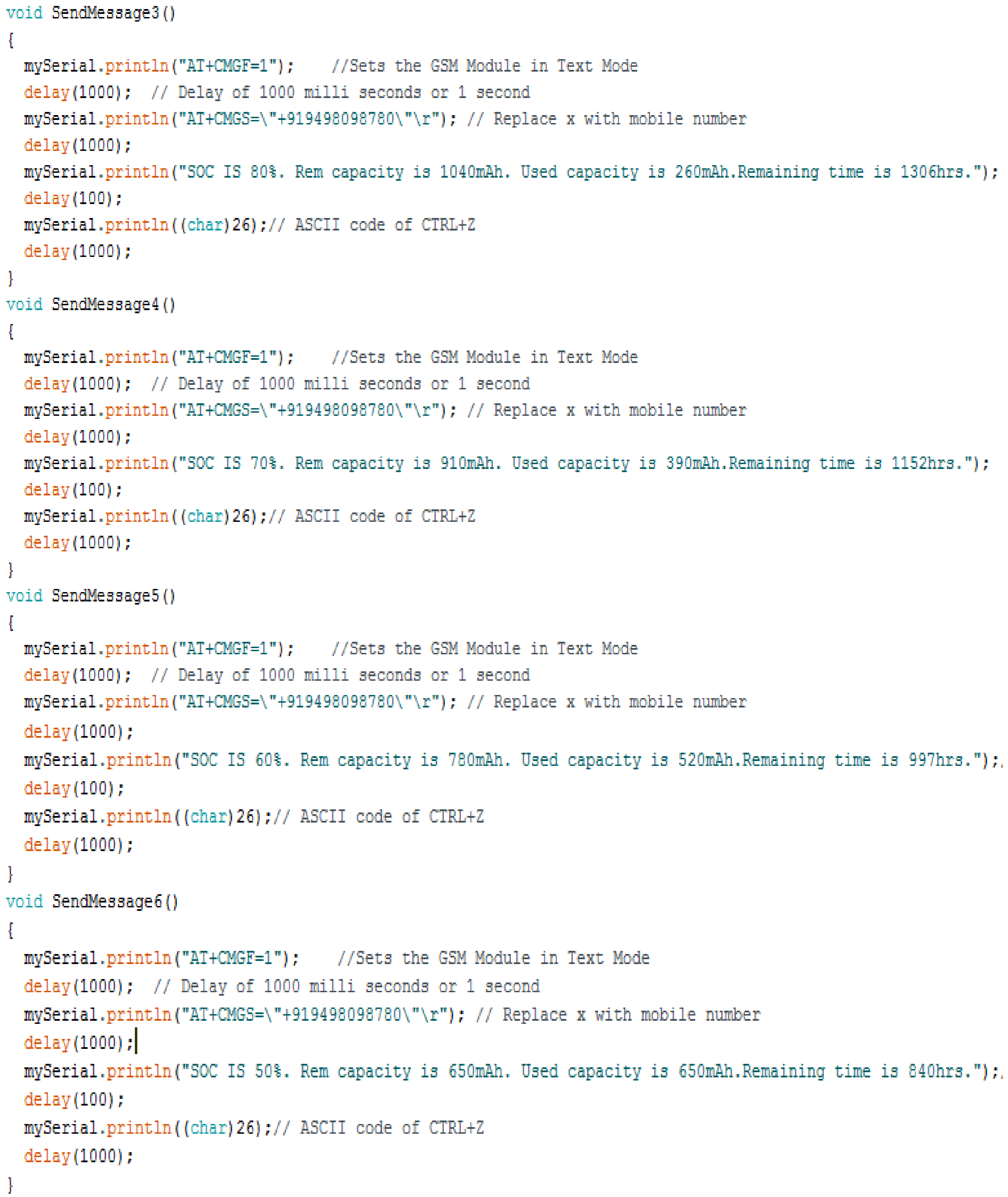
```
void loop()
{

}
void capacity()
{
  rem_capacity = (soc * full_capacity) / 100;
  used_capacity = full_capacity - rem_capacity;
  Iout = vin / (15.6*1000); // r2 = 5.6k ohm and r1 = 10k ohm
  discharge_time = (rem_capacity / Iout) * 0.001;
}

void display_parameters()
{
  Serial.print("SOC is =  ");
  Serial.print(soc);
  Serial.print(" Remaining capacity is = ");
  Serial.print(rem_capacity, 2);
  Serial.print(" Used capacity is = ");
  Serial.print(used_capacity, 2);
  Serial.print(" Remaining time is = ");
  Serial.println(discharge_time, 2);
}
```



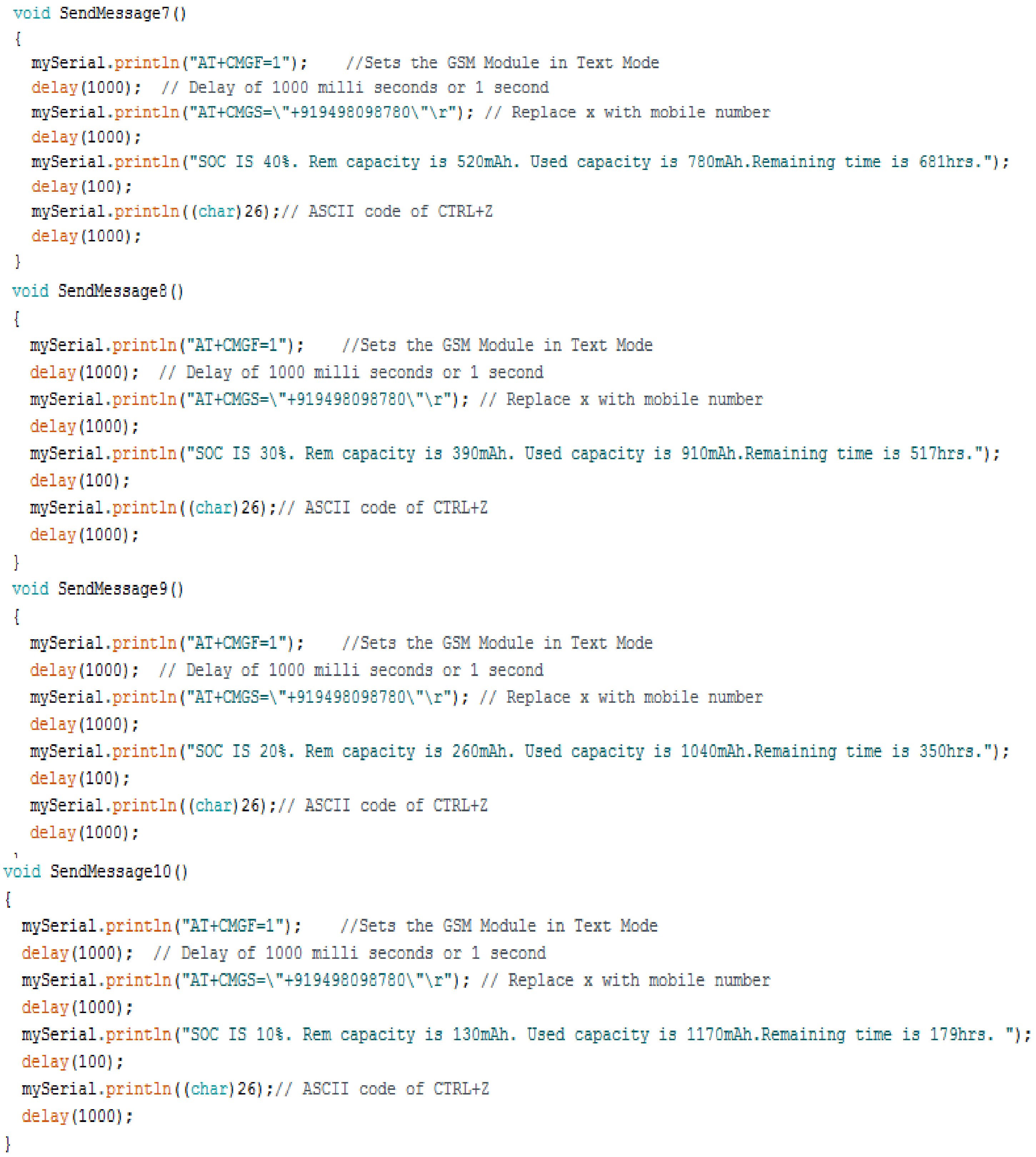
```
void SendMessage1()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 100%. Rem capacity is 1300mAh. Used capacity is 0mAh.Remaining time is 1609hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
void SendMessage2()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
```



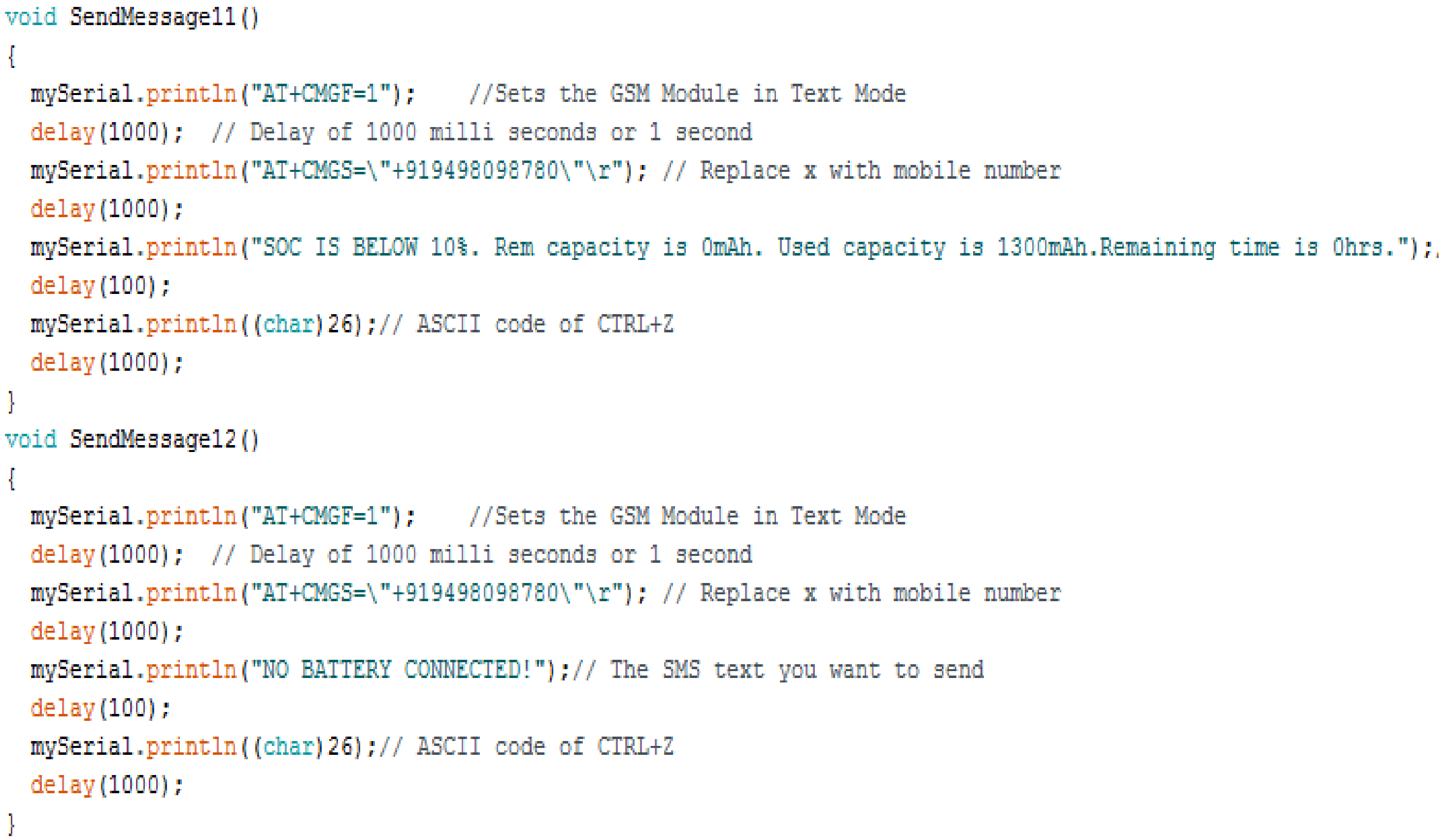
```
void SendMessage10()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second                                 ');
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 10%. Rem capacity is 130mAh. Used capacity is 1170mAh.Remaining time is 179hrs. ");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
```



```
void SendMessage3()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 80%. Rem capacity is 1040mAh. Used capacity is 260mAh.Remaining time is 1306hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
void SendMessage4()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 70%. Rem capacity is 910mAh. Used capacity is 390mAh.Remaining time is 1152hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
void SendMessage5()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 60%. Rem capacity is 780mAh. Used capacity is 520mAh.Remaining time is 997hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
void SendMessage6()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 50%. Rem capacity is 650mAh. Used capacity is 650mAh.Remaining time is 840hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
```



```
void SendMessage7()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 40%. Rem capacity is 520mAh. Used capacity is 780mAh.Remaining time is 681hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
void SendMessage8()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 30%. Rem capacity is 390mAh. Used capacity is 910mAh.Remaining time is 517hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
void SendMessage9()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 20%. Rem capacity is 260mAh. Used capacity is 1040mAh.Remaining time is 350hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
void SendMessage10()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS 10%. Rem capacity is 130mAh. Used capacity is 1170mAh.Remaining time is 179hrs. ");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
```



```
void SendMessage11()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("SOC IS BELOW 10%. Rem capacity is 0mAh. Used capacity is 1300mAh.Remaining time is 0hrs.");
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
void SendMessage12()
{
  mySerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  mySerial.println("AT+CMGS=\"+919498098780\"\r"); // Replace x with mobile number
  delay(1000);
  mySerial.println("NO BATTERY CONNECTED!");// The SMS text you want to send
  delay(100);
  mySerial.println((char)26);// ASCII code of CTRL+Z
  delay(1000);
}
```

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Estimation of battery parameters

With the help of voltage measure by Arduino, the voltage is converted into battery voltage using voltage division rule. With the help of battery voltage, SOC is calculated using the tabulation provided in the battery specifications. Based on SOC, used capacity and the remaining capacity of the battery is calculated. The discharge time is also calculated depending upon the load current. In simple case, our load is considered as two resistors in series and based on that the discharge time is calculated and all the battery parameters are send through GSM module to the receiving device.

4.4.1 VOLTAGE VS SOC

VOLTAGE	SOC
12.6 +	100
12.5	90
12.42	80
12.32	70
12.20	60
12.06	50
11.90	40
11.75	30
11.58	20
11.31	10
10.5 -	0

As we can be seen from the above table, diverse voltage qualities will cause the SOC to appear as something else. For battery voltage higher than 12.60V, a SOC with 100% is demonstrated. A SOC with 90% is indicated for battery voltage in the scope of 12.50V - 12.60V, while SOC with 30% is demonstrated for battery voltage in the scope of 11.75V – 11.89V. On the off chance that the battery if inside of the scope of 11.31V- 11.57V, a battery with 10% will be demonstrated, though a battery with 0% will be demonstrated for battery values under 10.50V.

CHAPTER 5

CONCLUSION

Project management include project planning, organizing and controlling resource within specified time period. The objective of project management is to achieve the project’s goal. This project had displayed the configuration of a user interface that can be utilized as a part of auto businesses to show the electric vehicle's battery parameters. The principle target of the project has been accomplished effectively. The user interface has displayed capacity, total battery voltage, battery current, state of charge (SOC) as needed in the extent of this task.

As a conclusion, all the goals of this task are effectively accomplished and the user interface can be relied upon to be executed in electric vehicle to monitor the battery parameter of the vehicle.

REFERENCES

[1] D. A. Neamen and B. Pevzner, *Semiconductor physics and devices: basic principles* vol. 3: McGraw-Hill New York, 2003.

[2] Y. Shao, J. Wang, H. Wu, J. Liu, I. A. Aksay, and Y. Lin, "Graphene based electrochemical sensors and biosensors: a review," *Electroanalysis,* vol. 22, pp. 1027-1036, 2010.

[3] T. Tuner, B. SiGe–HBT, and S. R. CMOS, "RF Transceiver," *Special Focus,* p.