CONTROL ALOGORITHM FOR BATTERY MANAGEMENT SYSTEM

A PROJECT REPORT

of

Electric Vehicles (EEE 4016)

by

SL.No.	Register Number	Name of the Student
1	16BEE0033	GOWTHAMARAJ
2	16BEE0031	RS RAGUL NIVASH
3	16BEE0327	SANDESH DINESH JAIN
4	16BEE0162	JITENDRA PARIT

Under the guidance of

Dr. Elangovan D

Associate Professor / SELECT

VIT

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRIAL ENGINEERING

April, 2019

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **CONTROL ALGORITHM FOR BATTERY MANAGEMENT SYSTEM** by **Ragul Nivash (16BEE0031), Gowthamaraj (16BEE0033), Sandesh Dinesh Jain (16BEE0327), Jitendra Parit (16BEE0162)** submitted to Vellore Institute of Technology University, Vellore, in partial fulfillment of the requirement for J component of the course titled **Electric Vehicles EEE 4016** is a work carried out by us under my supervision. The project fulfills the requirement for J component as per the regulations of this Institute and in my opinion meets the necessary standards for submission.

Sl. No	Name of the student	Register Number
1.	Gowthamaraj	16BEE0033
2.	Ragul Nivash	16BEE0031
3.	Jitendra Parit	16BEE0162
4.	Sandesh Jain	16BEE0327

Dr.Eleangovan D
Subject Faculty
Asso.Prof / SELECT
VIT

TABLE OF CONTENTS

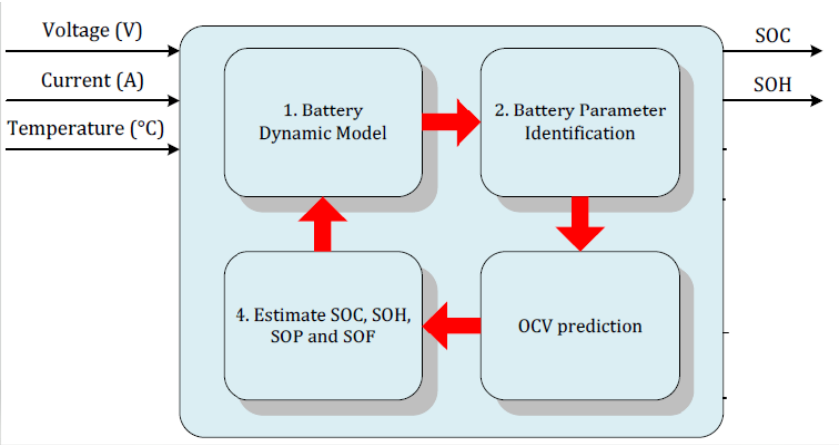
ABSTRACT

BMS is the link between batteries and users, and its main object is the rechargeable battery. Once operated, the BMS would monitor the status of battery cells such as cell temperature, cell voltage, charging or discharging current, and so on. BMS helps to prolong the battery service life as well as to improve the performance, which is one of its main merits.

The main target of this project is both to construct the battery management system on a hardware way, and to design its relevant user interface to test the function on a software way. Furthermore, passive cell balancing is done in Matlab Simulink.

Introduction

A battery management system (BMS) is an electronic system that manages a rechargeable battery (cell or battery pack), such as by protecting the battery from operating outside its safe operating area, monitoring its state, calculating secondary data, reporting that data, controlling its environment, and balancing it. A battery management system is essentially the "brain" of a battery pack. It measures and reports crucial information for the operation of the battery and also also protects the battery from damage in a wide range of operating conditions.



Objectives:

In this project, we will implement the following functions for the BMS system:

1. Simulation of Passive cell balancing for 5 Lithium ion cells connected in series which are rated at 3.7 V and 3.2 Ah an implement a small model hardware for 3 cells(Zinc cells).

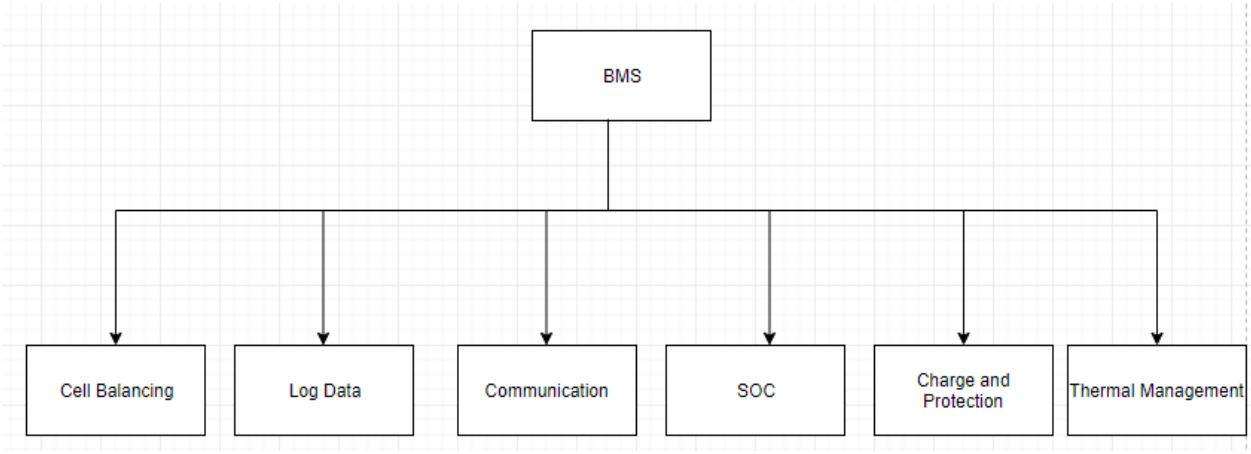
2. Implement a Thermal Protection System for the E-vehicle motor using a Temperature Controller

3. Determine the State-Of-Charge for 3 cells using the Look up tables provided on their datasheet.

4. Data acquisition through serial communication.

5. Data Logging for SOC LUT.

Description about the project



1. **Protection of the Battery:**

An important function of battery management system performs is cell protection. Lithium ion battery cells have two critical design issues; if you overcharge them you can damage them and cause overheating and even explosion or flame so it's important to have a battery management system to provide overvoltage protection. Lithium ion cells can also be damaged if they're discharged below a certain threshold, approximately 5 percent of total capacity. If the cells are discharged below this threshold their capacity can become permanently reduced. To ensure a battery's charge doesn't go above or below its limits, a battery management system has a safeguard device called a dedicated Lithium-ion protector. Every battery protection circuit has two electronic switches called "MOSFETs." MOSFETs are semiconductors used to switch electronic signals on or off in a circuit. A battery management system typically has a Discharge MOSFET and a Charge MOSFET.

If the protector detects that the voltage across the cells exceeds a certain limit, it will discontinue the charge by opening the Charge MOSFET chip. Once the charge has gone back down to a safe level then the switch will close again.

Similarly, when a cell drains to a certain voltage, the protector will cut off the discharge by opening the Discharge MOSFET.

2. Energy Management:

A battery pack usually consists of several individual cells that work together in combination. Ideally, all the cells in a battery pack should be kept at the same state of charge. If the cells go out of balance, individual cells can get stressed and lead to premature charge termination and a reduction in the overall cycle life of the battery. The cell balancers of the battery management system extend the life of the battery by preventing this imbalance of charge in individual cells from occurring.

3. Log Data

Monitoring and storing the battery's history is another possible function of the BMS. We are using influxdb to store the real time data with time stamp. The data base is a relational based and a time based database. The data from the microcontroller is directly stored into the database using a python script.

4. Communication

Most BMS systems incorporate some form of communications between the battery and the charger or test equipment.. Communications interfaces are also needed to allow the user access to the battery for modifying the BMS control parameters or for diagnostics and test.. We used serial communication for the transfer of data between microcontroller and the computing device[pc]. The data is obtained using python script and the script logs the data into the database

5. SOC

It is one of the main functions of the BMS. This is for providing the user with an indication of the capacity left in the battery, or it could be needed in a control circuit to ensure optimum control of the charging process. We used OCV method to get the SOC of each cell and then, calculated the new SOC of the whole pack The OCV method maps the voltage with the SOC of the cell.

6. Charge and protection

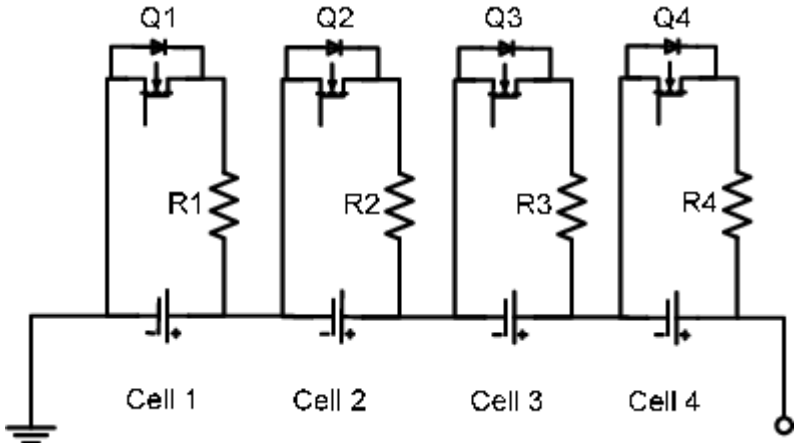
We used a module for charging the lithium battery of 3.7v. The input voltage to the module is 5V and 2A current. The module can give protection to the cell during charging.

7. Thermal management

For system with a smaller number of cells, it is advisable to go for air based thermal cooling. We used a fan for cooling the batteries when the get heated up. The thermal sensor in the pack can detect the temperature and switch on the fan when the temperature is higher than the threshold limit. The system makes sound on high temperate for providing safety to the users around.

Circuit Diagram, Design and Explanation.

Passive Cell balancing:



The figure shows the circuit for passive cell balancing for 4 cells in series in a battery pack.

Cell balancing is a technique in which voltage levels of every individual cell connected in series to form a battery pack is maintained to be equal to achieve the maximum efficiency of the battery pack. When different cells are combined together to form a battery pack it is always made sure that they are of the same chemistry and voltage value. But once the pack is installed and subjected to charging and discharging the voltage values of the individual cells tends to vary due some reasons which we will discuss later. This variation in voltage levels causes cell unbalancing which will lead to one of the following problems:

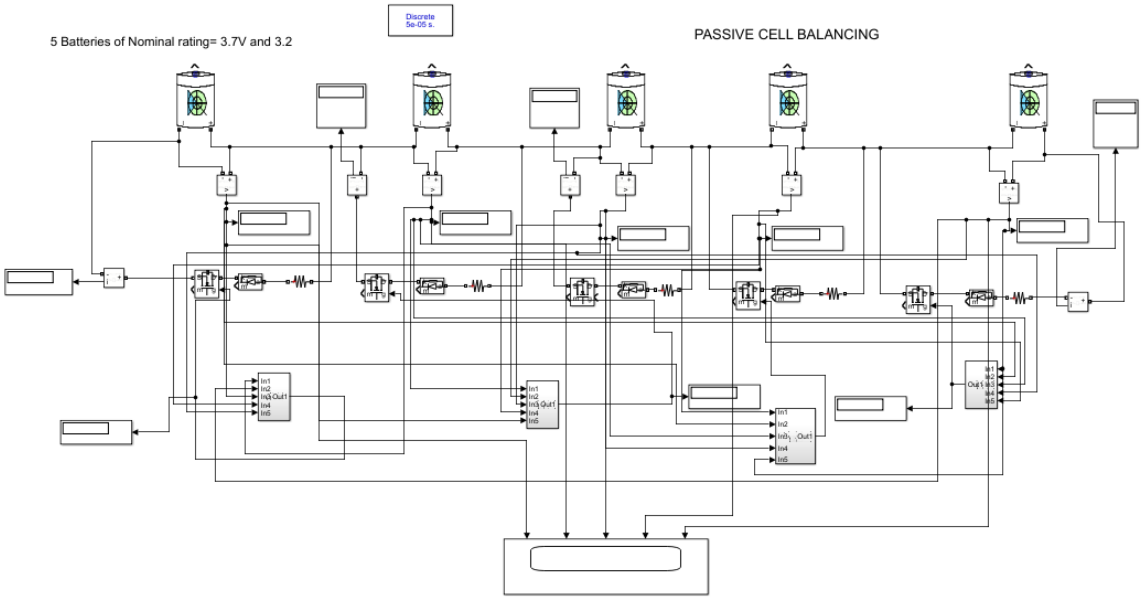
1. Thermal runaway 2. Cell degradation 3. Incomplete charging of the pack 4. Incomplete use of pack energy

There are various types of balancing of which Passive cell and active cell balancing are implemented the most. Active balancing is difficult to implement.

In our project we have implemented Passive cell balancing technique.

In this method a dummy load like a resistor is used to discharge the excess voltage and equalize it with other cells. These resistors are called as bypass resistors or bleeding resistors. Each cell connected in series in a pack will have its own bypass resistor connected through a switch as shown below.

Simulation:



Specifications:

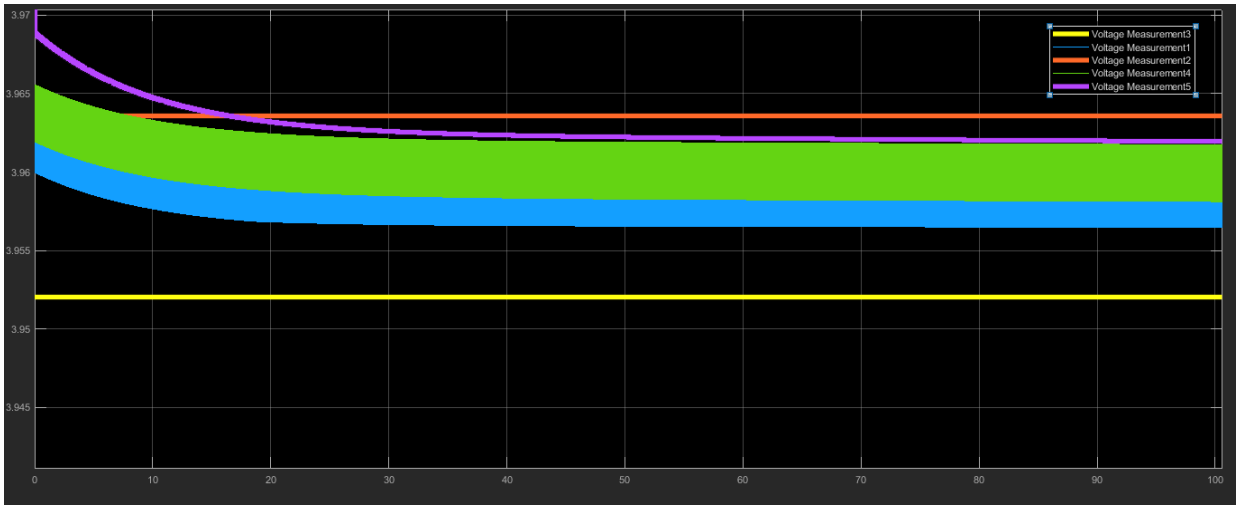
Battery rating: 3.7 V nominal and 3.2 Ah

SOC levels: 30%, 35%, 35%, 36% and 40% respectively

Bleed resistor value: 10 ohm (to limit the balancing current < 0.5 A)

The initial voltages of the cells are 3.952, 3.964, 3.964, 3.965 and 3.97 respectively.

Results:

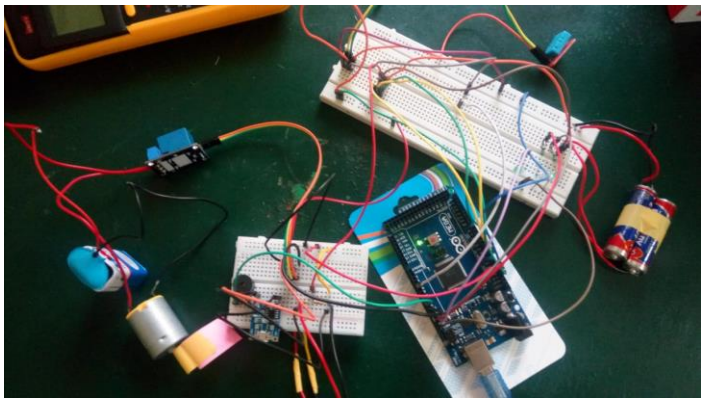


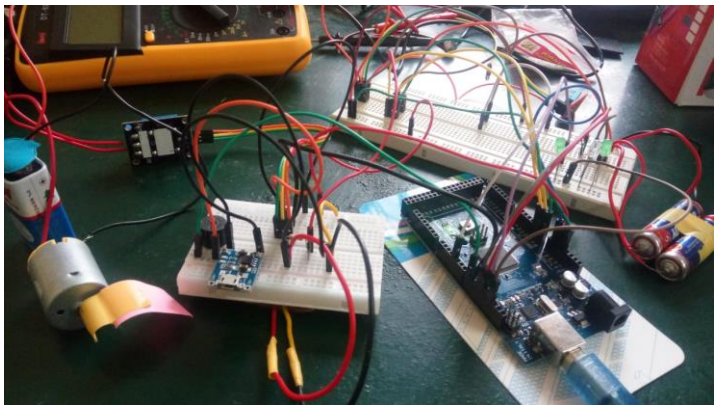
Performing passive balancing after 100 seconds gives the final battery voltages:

From left to right:

3.952 V, 3.96 V, 3.964 V, 3.958 V, 3.962 V.

Hardware Implementation.





Data logging is done by using influxdb to store the real time data with time stamp. This data is used as reference for state of charge through OCV algorithm. We use two 1.5v battery and one 3.7 li ion pouch battery. The 3.7 li ion battery is made into half using voltage divider circuit. LED is used to indicate the cell with max voltage, which will be useful for effective cell balancing. If the systems environment goes above the normal operating temperature, fan turns ON for cooling. This is indicated by buzzer. For communicating with the electronic device used in our prototype, we are using serial communication for the transfer of data between microcontroller and the computing device[pc]. The data is obtained using python script and the script logs the data into the database. For user's understanding we have our own graphical interface. We have even added charging module for the li ion pouch battery with minimum protection circuit.

Graphical User Interface.

Code

#1

```
#include <Ethernet.h>

#include <dht11.h>
                      #define DHT11PIN 4
                      #define led1 13
                      #define led2 12
                      #define led3 11

                      dht11 DHT11;
                      int motorPin = 9;
                      int buzzer = 8;


                      void setup() {

                        Serial.begin(9600);

                        pinMode(led1, OUTPUT);
                        pinMode(led2, OUTPUT);
                        pinMode(led3, OUTPUT);
                        pinMode(motorPin, OUTPUT);


                      }

                      void loop()
                      {

                        // read the input on analog pin 0:
                        int sensorValue = analogRead(A0);
                        int sensorValue1 = analogRead(A1);
                        int sensorValue2 = analogRead(A2);


                        // getting the voltage values
                        int v1 = sensorValue * (5.0 / 1023.0)*100;
                        int val1 = map(temp,140 , 165, 0, 100);

                        int v2 = sensorValue1 * (5.0 / 1023.0)*100;
                        int val2 = map(temp,140 , 165, 0, 100);

                        int v3 = sensorValue2 * (5.0 / 1023.0)*100;
                        int val3 = map(temp,140 , 165, 0, 100);
```

```
//displaying the remperature
  int chk = DHT11.read(DHT11PIN);
  int temp =(int)DHT11.humidity;
  temp = map(temp,38 , 65, 23, 35);



 //logic for passive balancing

 if((v1-v2)>10){
  digitalWrite(led1, HIGH);
     delay(100);
     digitalWrite(led1, LOW);
     delay(10);

 }
 else if((v2-v1)>10){
  //turn on led 2
  digitalWrite(led2, HIGH);
     delay(10);
     digitalWrite(led2, LOW);
     delay(10);

 }

  if((v2-v3)>20){
  //trun on the led 2
  digitalWrite(led2, HIGH);
     delay(10);
     digitalWrite(led2, LOW);
     delay(10);

 }
 else if((v3-v2)>20){
  //turn on led 3
  digitalWrite(led3, HIGH);
     delay(10);
     digitalWrite(led3, LOW);
     delay(10);

 }
```

```
//Logic for controlling the motor if the heat is more
  if(temp>=33){

     digitalWrite(motorPin, HIGH);
     delay(25);


  }

  else{

    digitalWrite(motorPin, LOW);

  }

if(temp>=38){
      tone(buzzer, 1000); // Send 1KHz sound signal...
  delay(25);        // ...
  noTone(buzzer);     // Stop sound...
  delay(25);        // ...



  }


  //Serial.print("Temperature (C): ");
  Serial.println(temp);


  //Serial.print("SOC 1 = ");
  Serial.println(val1);

  //Serial.print("SOC 2 = ");
  Serial.println(val2);

  //Serial.print("SOC 3 = ");
  Serial.println(val3);
    delay(500);


}
```

#2

```
import serial
```

```
from influxdb import InfluxDBClient

client = InfluxDBClient(host='localhost', port=8086)


client.switch_database('network')

ser = serial.Serial('COM3',9600,timeout=1)

while(1):
     t1=int(ser.readline().decode('ascii'))


    soc1=int(ser.readline().decode('ascii'))
    soc2=int(ser.readline().decode('ascii'))
    soc3=int(ser.readline().decode('ascii'))
    soc= int((soc1+soc2+soc3)/3)



    json_body = [ { "measurement": "bms_ev", "tags": { "user": "data",
}, "fields": { "temp": t1, "soc1":
soc1,"soc2":soc2,"soc3":soc3,"soc":soc, } } ]


    client.write_points(json_body)
```

Working of the Model

Drive Link:
https://drive.google.com/file/d/1_2HKLJzJKmg0pwbyIQJq6W3TOGjM2c3H/view?usp=sharing

Conclusion

This battery management system provided SOC reading within less percent error. For our architecture the required thermal management worked perfectly, ie air cooling. Moreover the Matlab Simulink were generated for passive cell balancing. This prototype has great industry application from managing the charge of a battery to testing for quality control. this process in reverse could tell users when batteries are almost full and battery banks should be switched to discharging for applications.

References

[1] http://etheses.whiterose.ac.uk/17681/

[2] https://cecas.clemson.edu/~sonori/Publications/X%20ONORI.pdf

[3] https://circuitdigest.com/article/battery-management-system-bms-for-electric-vehicles

[4] https://www.ti.com/download/trng/docs/seminar/Topic%202%20-%20Battery%20Cell%20Balancing%20-%20What%20to%20Balance%20and%20How.pdf

[5] DOI: 10.1109/TEMSCON.2017.7998405

[6] https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/21964/Zhang.pdf?sequence=1&isAllowed=y