CHAPTER 1

INTRODUCTION

Electric vehicles rely on batteries to provide green transportation. Lithium chemistry is now widely acknowledged in electrified mobility as the preferred power storage technology. A battery management system is an electrical controller that monitors and manages rechargeable charging (BMS). This is essentially battery tracking, which is keeping track of key performance metrics like the voltage, current, and battery temperature when charging and discharging. The battery deterioration cycle can be slowed by correctly maintaining the battery and maintaining its charging and discharging under various demanding scenarios.IoT-based battery management system consists of two main components: tracking tools and user interfaces.

Objectives

The objectives of the project are as follows.

- The implementation of the battery management system to monitor the status like Voltage Level, Temperature, and Battery percentage and then notify the user.
- Manual control to avoid Temperature runaway during charging.
- **System maintenance identification to be implemented in this project.**

BATTERY MONITORING AND MANAGEMENT SYSTEM

Lithium-ion batteries have proved to be the battery of interest for Electric Vehicle manufacturers because of their high charge density and low weight. Even though these batteries pack a lot of punch for their size they are highly unstable. These batteries must never be overcharged or discharged in any circumstance which brings in the need to monitor their voltage and current. This process gets a bit tougher since there are a lot of cells put together to form a battery pack in EV and every cell should be individually monitored for its safety and efficient operation which requires a special dedicated system called the Battery Monitoring System. The proposed project is intended to link the system to the IOT cloud and fetch the required data from the cloud.

CHAPTER 2

Literature Survey

Paper [1]: IoT-Based Battery Management System for Electric Vehicle

Dr. Shakunthala C, et. al. International Journal of Engineering Research and Applications www.ijera.com ISSN: 2248-9622, Vol. 12, Issue 4, (Series-I) April 2022. DOI: 10.9790/9622-1204014752

The Internet of Things (IoT) is the connecting of common devices via a network. It's a wireless electrical link aimed at connecting everyday objects.

The interface connects coding in electrical devices, detectors, and devices to a Wi-Fi connection.

The automobile sector, which includes the selling of electrified vehicles and, more recently, driverless vehicles, is another significant use of IoT equipment.

The voltage sensor measures the battery's voltage level. SIM808 GSM technique will be used to locate the battery. Arduino will be used to process the data.

Paper[2]: IoT-Based Battery Management System for Electric Vehicle International Journal of Scientific Research in Engineering and Management (IJSREM) Volume: 05 Issue: 06 | June - 2021 ISSN: 2582-3930, Ashwin Raghuwanshi, Mayur Bharti, Parth Ghosh & Prof. Prashant Salunkhe.Description: Nowadays, electric vehicle (EVs) is becoming popular since fuel prices becoming costlier. thanks to this scenario,

Most EVs use a rechargeable battery which is lithium-ion battery. It's smaller to be compared with lead acid. In fact, it's a continuing power, and energy's life cycle is 6 to 10 times greater compared with lead acid battery. Lithium ion battery life cycle are often shortened by some reasons like overcharging and deep discharges. On the opposite hand, EV usually has a limited range of traveling thanks to battery size and structure. Now, a crucial reason that limits the appliance of EVs is the safety of existing battery technology.

Arduino Uno microcontroller, an ESP8266/GPRS/GPS module, and a 9V battery for power supply are used for developing the smart BMS for EVs.

Inference NodeMCU can be used to monitor the Battery parameters through interfacing with the sensors.

Paper_[3]: “Battery Health Monitoring IoT System for Commercialized Electric Vehicle Batteries: Lithium-Ion“.Fawad Ali Shah, Shehzad Shahzad Sheikh, Umar Iftikhar. https://www.researchgate.net/publication/336876982

Description: Batteries are widely utilized to power electric vehicles, hybrid electric vehicles (HEVs), and many other high-power applications.

The battery is critical to its efficiency, safety, and reliability, Reliability, Initially, numerous types of batteries are discussed in this paper.

According to the research, utilized in EVs and HEVs are explored. The most recent battery management methods (BMS). Lithium-ion batteries due to their extended life, are a preferred source of EVs and HEVs’ high power density, and good charging and charging efficiency performance discharge However, there are still some concerns.

Li-ion batteries are used in a variety of applications, including complicated electrochemistry, battery deterioration and battery accuracy health assessment. Following that, the article considers the economic, environmental and energy efficiency implications of increased use of electric vehicles .

Paper_[4]: **Lithium-Ion Battery Management System for Electric Vehicles**: Constraints, Challenges, and Recommendations A. K. M.Ahasan Habib1,2 ,MohammadKamrulHasan1,* Ghassan F. Issa 3, Dalbir Singh 1,* Shahnewaz Islam 2 and Taher M. Ghazal 1,3 https://www.mdpi.com/2313-0105/9/3/152

Flexible, manageable, and more efficient energy storage solutions have increased the demand for electric vehicles. A powerful battery pack would power the driving motor of electric vehicles. The battery power density, longevity, adaptable electrochemical behavior and temperature tolerance must be understood.

The battery management system covers voltage and current monitoring; charge and discharge estimation, protection, and equalization; thermal management; and battery data actuation and storage.

CHAPTER 3

Problem Statement

- “Today, electric cars utilize Li-ion batteries that are combined in large quantities, sometimes numbering in the hundreds or thousands.
- These battery packs have a voltage rating of around 48 V and can supply a high current of around 27 A (rough figure) during use.
- Due to overload, the temperature of the battery pack increases which leads to thermal runaway. To overcome this, a Battery monitoring System has to be developed for the 48 V, 27 Ah Lithium–Ion Battery Pack for Electric vehicles.

CHAPTER 4

OBJECTIVES

The objectives of the project are:

- A battery monitoring system to monitor its status such as Voltage Level, Temperature and Battery percentage has been developed.
- The project requires an IoT-based ESP8266 which monitors its Battery Level using Blynk IoT Cloud has been developed.
- Manual control to avoid Temperature runaway during charging.
- System maintenance identification had been implemented in this project.
- Keyless Vehicles turning ON and OFF are developed.
- Battery Charging control has been developed.
- Overload and Shor-Circuit protection have been developed.
- Displaying all the parameters on OLED and also in Serial monitor are developed.

CHAPTER 5

PROPOSED BLOCK DIAGRAM / CIRCUIT DIAGRAM

5.1 BLOCK DIAGRAM:

The batteries are connected in series parallel giving 48 V, which is trimmed to 3.3 V through the voltage divider circuit and then applied to the input of the ESP8266 ADC pins. Through a USB port the output is transmitted to a serial monitor. The ESP8266 Dev-kit is connected to the internet through Wi-Fi. Blynk IoT Cloud, the dashboard displays the battery voltage percentage**.**

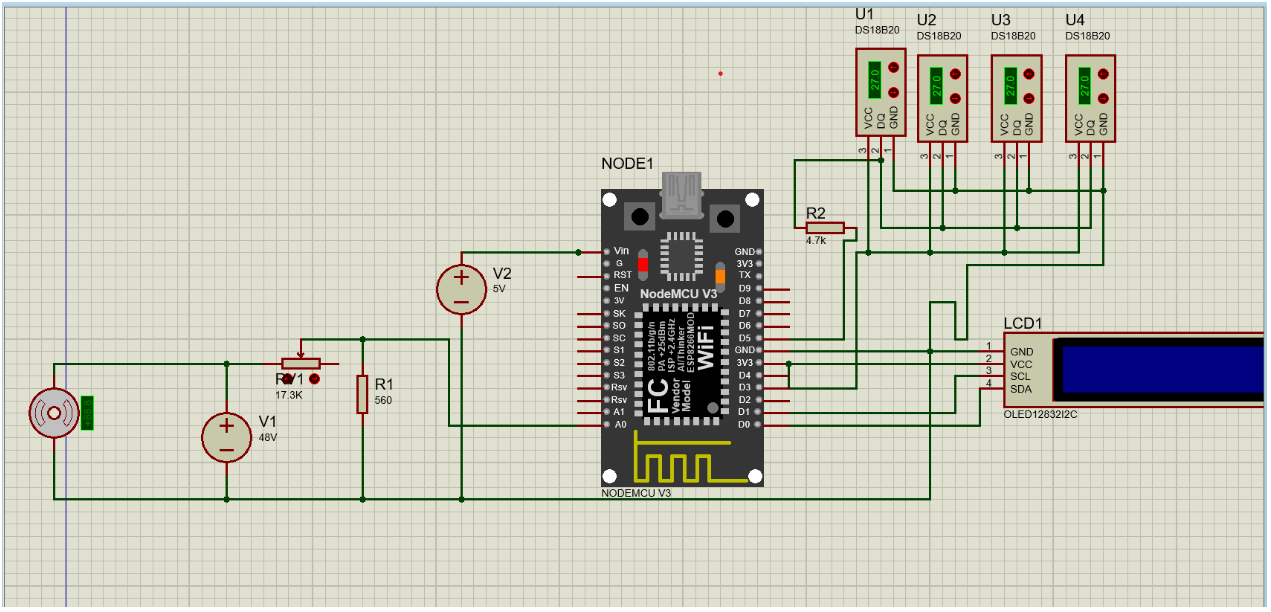


Fig 5.1: Circuit diagram of Battery Monitoring system interfaced with IOT

5.2 WORK FLOW DIAGRAM

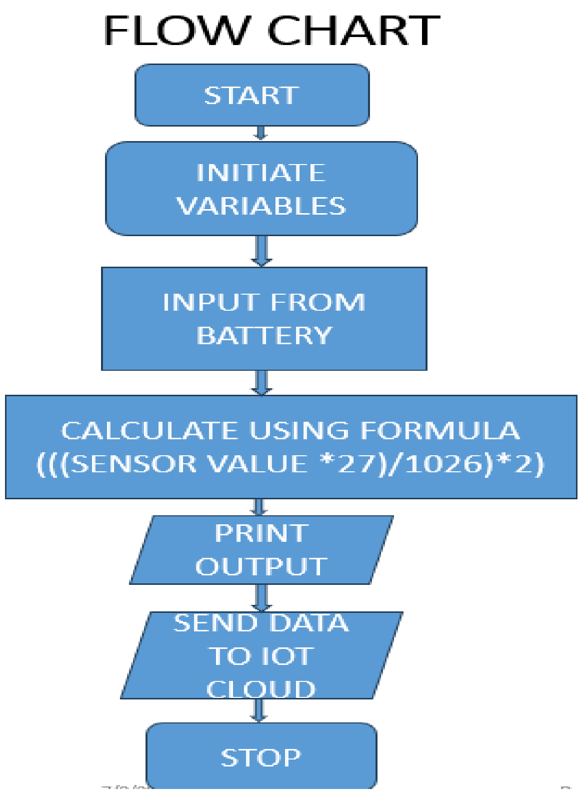


Fig5.2: Workflow of Program in Node MCU

- In this code bv is the variable declared to store the battery voltage. These variables are used to display the voltage in the console.(bv_ is battery voltage)
- The analog input is taken from the pins of the ESP8266Dev-kit.
- The voltage is calculated using the formula:

  (((Sensor Value*27)/1026)*2).
- The 27 is half the value of applied Voltage.
- Print the values of battery voltage percentage and voltage on the serial monitor.
- The ESP is later connected to the internet to give the output in Blynk IoT Cloud.

CHAPTER 6

WORKING PROCEDURE

- The Node MCU has a WiFi inbuilt function that provides a feature to communicate with the cloud.
- The battery pack of 48 V and 20 Ah is given to the analog pin of the Node MCU through the voltage divider circuit.
- The voltage divider circuit drops the voltage from 54.6 V to 3.3 V when fully charged, at the point of the nominal voltage of 48 V dropped to 2.94 V and in case of a lower cutoff voltage of 37 V is trimmed to 2.26 V. So, both upper and lower cutoff voltages are applicable to Node MCU.
- DS18B20 Temperature sensor (4 NO) sensor is capable of measuring the temperature and up to -55 to 125 Degrees.
- The DS18B20 sensor is connected to the digital pin of the Node MCU that's read the data from the sensor.
- The 0.96-inch OLED display displays the value of temperature from the serial port of NodeMCU.
- The Blynk IoT cloud is used to communicate with the chip ESP8266 and displays the values like Voltage, Percentage of the battery, Temperature on the dashboard and also in smartphones.

CHAPTER 7

WORK CARRIED OUT

7.1 HARDWARE:

1. ESP 8266 Dev Module:

- The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems.
- It can be used as a standalone device, or as a UART to Wi-Fi adaptor to allow other microcontrollers to connect to a Wi-Fi network.
- For example, you can connect an ESP8266 to an Arduino to add Wi-Fi capabilities to your Arduino board.

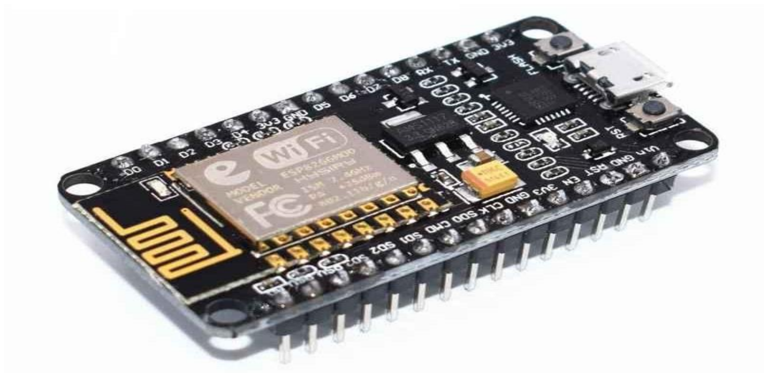


Fig 7.1: NodeMCU ESP8266

Table 1: Technical Specifications of NodeMCU

Board	Name	ESP8266
	SKU	A000066
Microcontroller	ATmega328PT	
USB connector	USB-B	
Pins	Built-in LED Pin	13
	Digital I/O Pins	14
	Analog input pins	6
	PWM pins	6
	UART	Yes

Communication	I2C	Yes
	SPI	Yes
Power	I/O Voltage	5V
	Input voltage (nominal)	3.3V to 5V
	DC Current per I/O Pin	20 mA
	Power Supply Connector	Micro USB
Clock speed	Main Processor	ATmega328P 16 MHz
	USB-Serial Processor	ATmega16U2 16 MHz
Memory	ATmega328P	ATmega328P
Dimensions	Weight	25 g
	Width	53.4 mm
	Length	68.6 mm

2. DS18B20

These 1-wire digital temperature sensors are fairly precise (±0.5°C over much of the range) and can give up to 12 bits of precision from the onboard digital-to-analog converter. They work great with any microcontroller using a single digital pin, and you can even connect multiple ones to the same pin, each one has a unique 64-bit ID burned in at the factory to differentiate them. Usable with 3.0-5.0 V systems

DS18B20 Sensor Technical specifications:-

- Usable temperature range: -55 to 125°C (-67°F to +257°F)
- 9 to 12-bit selectable resolution
- Uses 1-Wire interface- requires only one digital pin for communication
- Unique 64-bit ID burned into chip
- Multiple sensors can share one pin
- ±0.5°C Accuracy from -10°C to +85°C
- Temperature-limit alarm system
- Query time is less than 750ms
- Usable with 3.0V to 5.5V power/data

Fig 7.2: DS18B20

3. Prototype board:

- Single-sided Universal PCB Prototyping Board with standard 0.1″ through Holes spacing. This board can be used to build small-sized circuits.

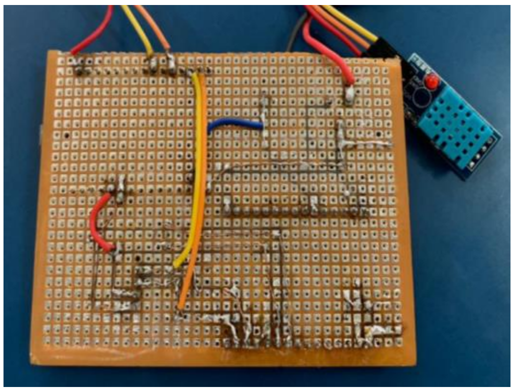
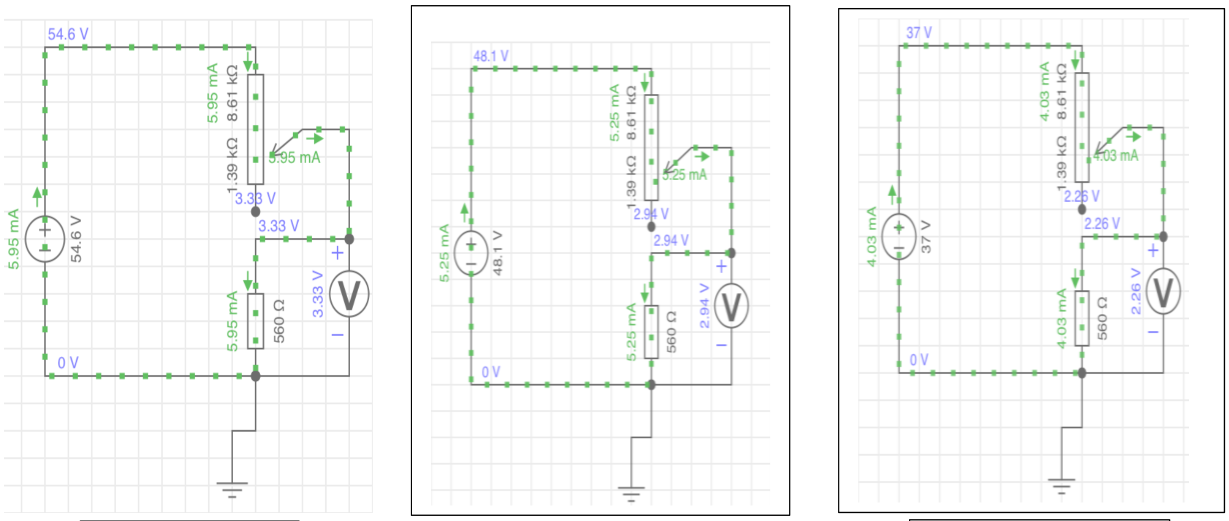


Fig 7.3: Prototype board

4. Voltage divider:

- The NodeMCU works on a nominal of 3.3 V, therefore the voltage divider circuit is required for reducing the voltage so that applies to ESP8266.



Fully charged

Charged at Nominal Voltage

Cutoff Voltage

Fig 7.4: Voltage divider circuit for various cases

- The above circuits are tested at each stage by using the “EVERY CIRCUIT” Android app.
- The 10 k trimmer Pot and quarter-watt resistor of 560 ohm are used in this circuit.

5. 0.96-inch OLED display:

- Regardless of the size of the OLED display, the SSD1306 driver includes a 1 KB Graphic Display Data RAM (GDDRAM) that stores the bit pattern to be displayed on the screen. This 1 KB memory area is divided into 8 pages (from 0 to 7). Each page has 128 columns/segments (block 0 to 127). 8 pages x 128 segments x 8 bits of data = 8192 bits = 1024 bytes = 1 KB memory

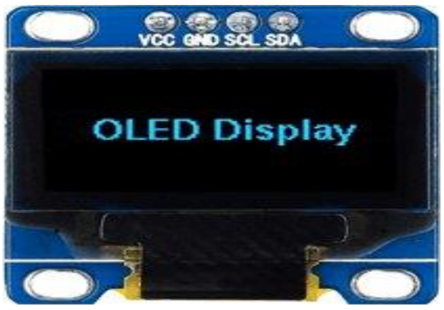


Fig 7.5: 0.96 OLED display

Table 5: Technical Specifications of OLED Display

Display Technology	OLED (Organic LED)
MCU Interface	I2C / SPI
Screen Size	0.96 Inch Across
Resolution	128×64 pixels
Operating Voltage	3.3V – 5V
Operating Current	20mA max
Viewing Angle	160°
Characters Per Row	21
Number of Character Rows	7

6. BLDC Hub motor kit:

- The basic idea is just the same. In an ordinary motor, you have a hollow, outer, ring-shaped permanent magnet that stays static (sometimes called the **stator**) and an inner metallic core that rotates inside it (called the **rotor**). The spinning rotor has an **axle** running through the middle that you use to drive a machine. But what if you hold the axle firmly so it can't rotate and switch on the motor. Then the rotor and the stator have no choice but to swap roles: the normally static rotor stays still while the stator spins around it.

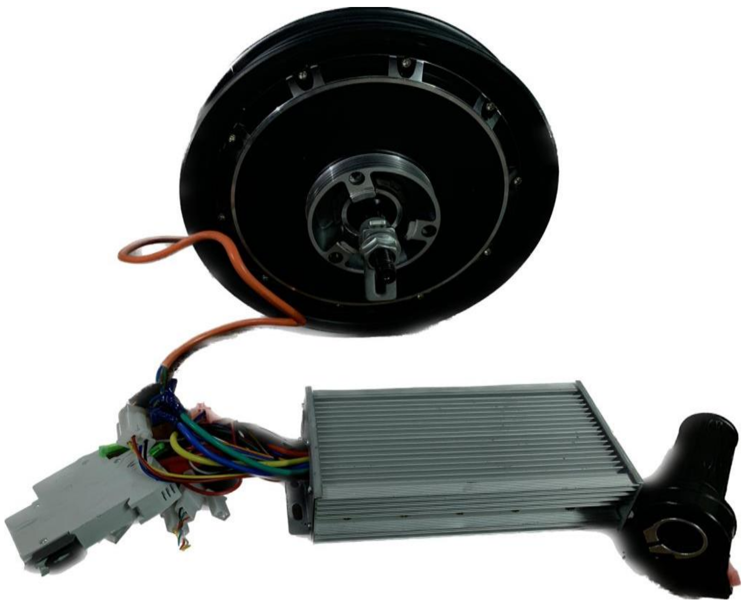


Fig 7.6: BLDC HUB Motor KIT

7. Li-ion battery pack of 48 V :

- The single battery on the market is generally around 3.7 V but many times the operating voltage range is a little larger and there is clearly a problem of insufficient voltage. At this time to increase the battery voltage battery pack and modular battery comes along and in the many high voltage battery, 48 V lithium-ion batteries are commonly used.

- **Compared with lead-acid batteries, 48 V lithium-ion batteries have the advantages of** small size**,** light weight**,** strong temperature adaptability**,** high charging **and** discharging efficiency**,** safety **and** stability**,** long service life**,** energy saving**, and** environmental protection

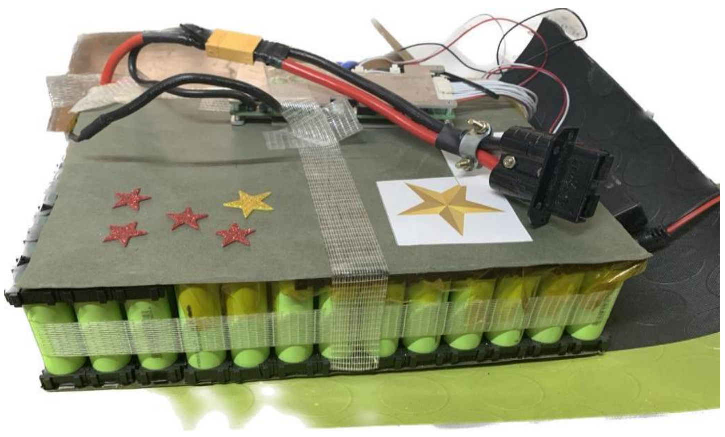


Fig 7.7: Li-ion battery pack

Table 5: Technical Specifications of Li-ion Battery Pack

Parameters	Technical Specs
Storage Temp	-40°C to 65°C
Dimensions	304 x 96 x 180 mm
Typical SoC Range	30-80%
Operating Voltage	37-54V
Storage Temp	-40°C to 65°C
Dimensions	304 x 96 x 180 mm

8. ACS712:

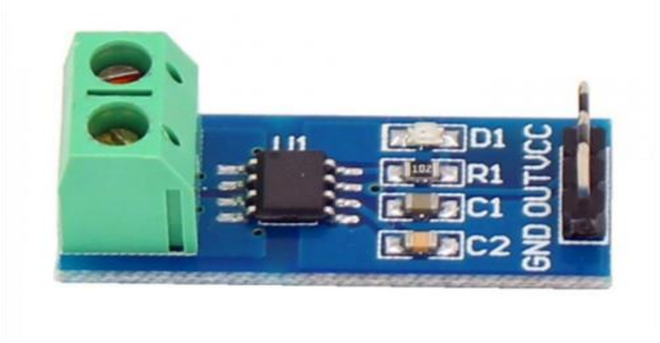


Fig 7.8: Current Sensor module

Specifications of ACS712 Sensor:

•Measures both AC and DC current.

•Available as 5A, 20A and 30A module.

•Provides isolation from the load.

•Easy to integrate with MCU, since it outputs analog voltage.

- ACS712 Current Sensor is the sensor that can be used to measure and calculate the amount of current applied to the conductor without affecting the performance of the system.ACS712 Current Sensor is a fully integrated, Hall-effect-based linear sensor IC. This IC has a 2.1kV RMS voltage isolation along with a low resistance current conductor.
- Current Sensor detects the current in a wire or conductor and generates a signal proportional to the detected current either in the form of analog voltage or digital output.
- Current Sensing is done in two ways - Direct sensing and Indirect Sensing. In Direct sensing, to detect current, Ohm's law is used to measure the voltage drop that occurred in a wire when current flows through it.

- A current-carrying conductor also gives rise to a magnetic field in its surrounding. In Indirect Sensing, the current is measured by calculating this magnetic field by applying Faraday's law.

9.ADS1115 : (16-bit Analog to digital converter)

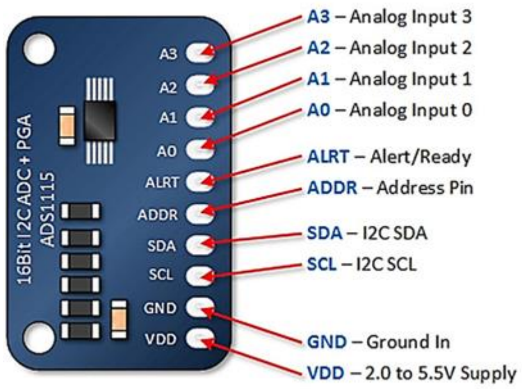


Fig 7.9: ADC Module

Features:

- 16-Bit Resolution: Provides high-precision analog-to-digital conversion.
- Programmable Gain Amplifier (PGA): Adjustable gain settings to measure different voltage ranges, making it versatile for various sensor types.
- I2C Interface: Easy integration with microcontrollers and other digital devices using the I2C communication protocol.
- Four Multiplexed Inputs: Allows the ADS1115 to read from four different channels, either individually or in differential mode.
- Low Power Consumption: Suitable for battery-powered and low-power applications.
- Internal Oscillator: Eliminates the need for an external clock, simplifying design.
- Data Rate Programmability: Configurable data rates up to 860 samples per second (SPS).
- Wide Power Supply Range: Operates from 2.0V to 5.5V, accommodating various power supply configurations.
- Comparator Function: On-chip programmable comparator for simple alert functions without microcontroller intervention.

10. Xh-M604 Charging control board:

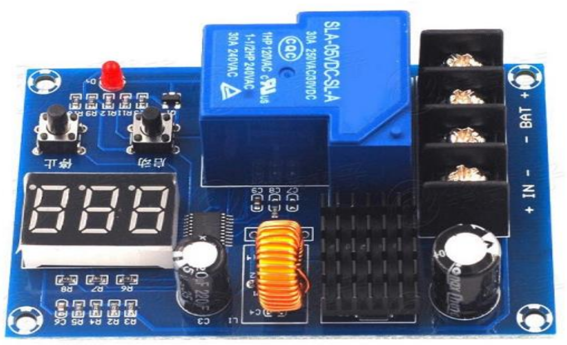


Fig 7.10: Charging Board

Features:

1. When the voltage is higher than 55V, the relay opens and stops charging and discharging.
2. It is very smart and helps to save electricity, and is safe.
3. Automatically power off, save energy and extend battery life.
4. Durable, Charging Protection, Easy to Install.

7.2 SOFTWARE:

Blynk IoT:

- In order to monitor the sensor data and Battery Data on the Blynk IoT Server, we first need to set up the Blynk IoT Cloud dashboard.
- A template is a project in which we can create a web and mobile dashboard for specific hardware. After that, we can choose a category according to our project.
- A data stream is like a pipeline or channel. The data will be sent or received through these data channels.
- In a single project or template, there can be multiple DataStream. In our project, we are receiving four data: **Temperature, Battery Voltage, and Battery Percentage**. So in this project, we created Three DataStream.
- Charging control through Blynk IoT cloud.
- Events are used for notification alert systems. So, here I am creating events to monitor Battery Percentage. If the battery percent reaches below the threshold value an event is triggered and a notification is sent to your mobile phone.

Fig 7.2.1: Logo of Blynk IoT

CHAPTER 8

RESULTS AND DISCUSSION

- Fig 8.1 represents the hardware connection and Fig 8.2 represents the displayed voltage, temperature and percentage of battery. Which helps in monitoring the battery condition.
- Fig 8.3 represents the Temperature chart of one week data, 8.4 represents the each temperature sensor data along with Current sensor data.

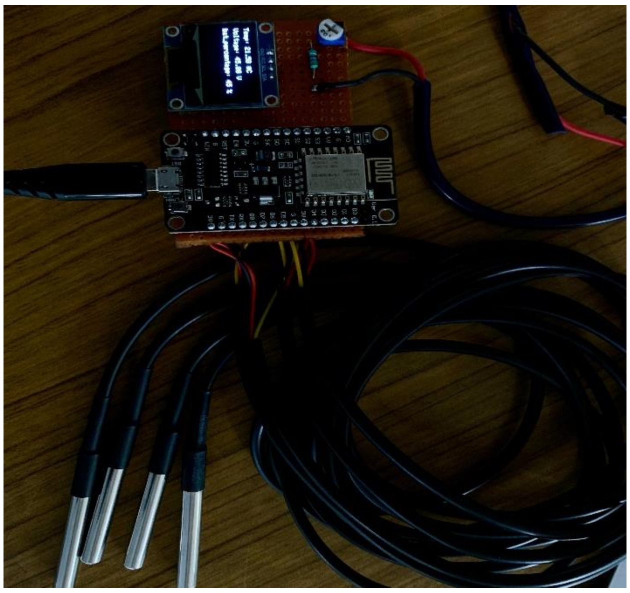


Fig 8.1: Circuit Connections

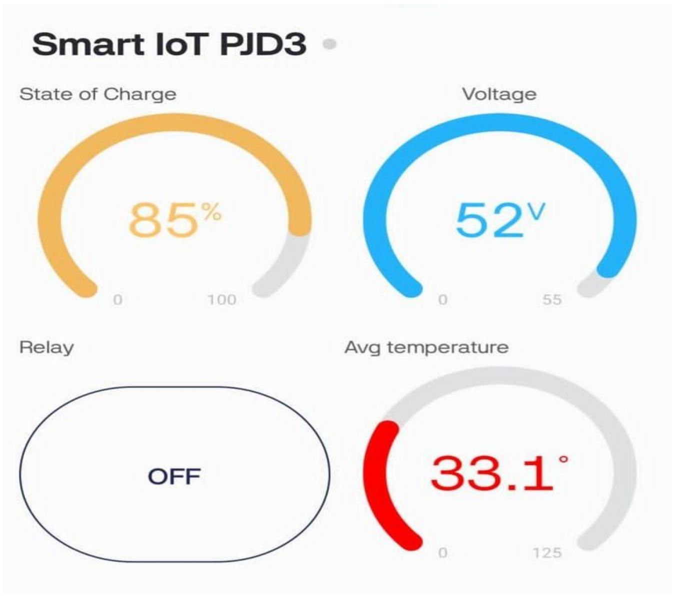


Fig 8.2 Displayed values in smartphone

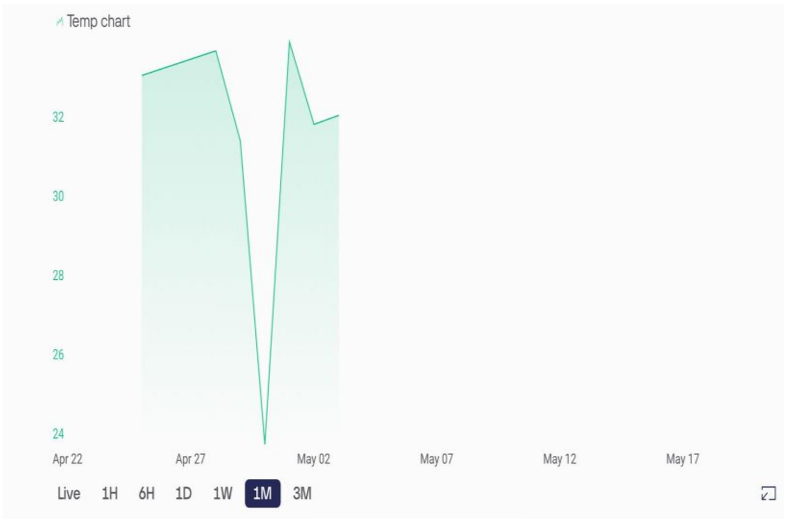


Fig 8.3: Temperature Chart of 1 week

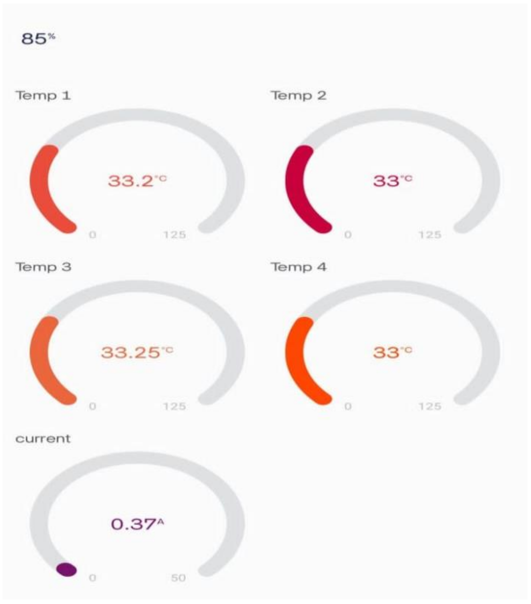


Fig 8.4: Each sensor Temperature

CHAPTER 9

CONCLUSION & FUTURE SCOPE

9.1 CONCLUSION

- A 48 V, 27 Ah Battery pack connected to a 1200 W, 560 RPM BLDC Hub Motor has been monitored using the developed Battery Monitoring System.
- The Management system to monitor and control the charging, monitor voltage, percentage of charge, and temperature, using the sensors such as DS18B20, a 0.9inch OLED Display.
- The parameters are displayed in the mobile device on the Blynk Platform.

  Project video link: https://youtube.com/shorts/0vE1B-GQkLA?si=G-8Fz4-kRXa7QtZZ

9.2 FUTURE SCOPE:

- ➢ Battery Monitoring system can be made to monitor the State of Health, each cell voltage monitoring, cell balancing, etc.
- ➢ The monitoring system can be made compact by developing it with a PCB design.
- ➢ By using advanced and more efficient devices this can be implemented for large capacity battery packs, such as in electric cars, buses and upcoming metro based on Electric vehicle technology.
- ➢ Not only in Electric vehicle domain this can also be taken for UPS application.

REFERENCES

1] IoT Based Battery management system for electric vehicles - IJERA https://www.ijera.com/papers/vol12no4/Ser-1/G1204014752.pdf

[2] IoT-Based Battery Management System for Electric Vehicles https://www.academia.edu/107285001/IoT_Based_Battery_Management_System_for_Electric_Vehicles

[3]https://www.researchgate.net/publication/336876982_Battery_Health_Monitoring_for_Commercialized_Electric_Vehicle_Batteries_Lithium-Ion

4] A. K. M.Ahasan Habib1,2 ,MohammadKamrulHasan1,* Ghassan F. Issa 3, Dalbir Singh 1,* Shahnewaz Islam 2 and Taher M. Ghazal 1,3 https://www.mdpi.com/2313-0105/9/3/152 https://www.mdpi.com/2313-0105/9/3/152

ANNEXURE

11.1 PROGRAM FOR NodeMCU

```
#include <OneWire.h>

#include <DallasTemperature.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include <Adafruit_ADS1X15.h>


#define BLYNK_TEMPLATE_ID "TMPL3OcJLWhAT"

#define BLYNK_TEMPLATE_NAME "Smart IoT"

#define BLYNK_AUTH_TOKEN "H-_RpuGUHLXBZZhoq54DD-RvBCQgBrry"

#define BLYNK_FIRMWARE_VERSION "0.1.0"

#define BLYNK_PRINT Serial

#define APP_DEBUG

#define USE_NODE_MCU_BOARD


#include "BlynkEdgent.h"


#define SCREEN_WIDTH 128    // OLED display width, in pixels

#define SCREEN_HEIGHT 64    // OLED display height, in pixels

#define OLED_RESET -1       // Reset pin # (or -1 if sharing Arduino reset pin)


#define ONE_WIRE_BUS_D5 14 // D5 for DS18B20

#define RELAY_PIN_D0 D0     // Define relay pin to D0

#define NUM_SENSORS 4       // Number of DS18B20 sensors


OneWire oneWire_D5(ONE_WIRE_BUS_D5);

DallasTemperature DS18B20_D5(&oneWire_D5);


Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

```
float voltage;
int bat_percentage;
int analogInPin = A0; // Analog input pin
int sensorValue;
float calibration = 0.0; // Check Battery voltage using multimeter & add/subtract the value
int relayState = LOW;    // Initial relay state

Adafruit_ADS1115 ads; // Create an ADS1115 instance
float current;

const float CURRENT_THRESHOLD = 0.1; // Threshold below which current is considered as 0 Amp

void setup() {
  Serial.begin(115200);
  delay(100);
  BlynkEdgent.begin();

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // initialize with the I2C addr 0x3C (128x64)
  display.clearDisplay();
  display.setTextColor(WHITE);
  delay(100);

  pinMode(RELAY_PIN_D0, OUTPUT);

  // Set the number of sensors found on the bus
  DS18B20_D5.begin();
  DS18B20_D5.setResolution(10); // Set the sensor resolution (you can adjust this if needed)

  // Search for DS18B20 devices and print their addresses
  Serial.println("Scanning for DS18B20 devices...");
  int deviceCount = DS18B20_D5.getDeviceCount();
  Serial.print("Found ");
  Serial.print(deviceCount);
  Serial.println(" DS18B20 device(s)");
  if (deviceCount != NUM_SENSORS) {
    Serial.println("Error: Incorrect number of DS18B20 sensors detected!");
    while (1); // Stop execution if incorrect number of sensors found
  }

  ads.begin();
  ads.setGain(GAIN_TWOTHIRDS); // Set the gain to 2/3x
}

// Blynk function to control the relay
BLYNK_WRITE(V2) {
   int value = param.asInt();
   digitalWrite(RELAY_PIN_D0, value);
}

void loop() {
  BlynkEdgent.run();

  DS18B20_D5.requestTemperatures();

  // Print individual sensor data and send to Blynk
  for (int i = 0; i < NUM_SENSORS; ++i) {
    float temp = DS18B20_D5.getTempCByIndex(i);
    Serial.print("Sensor ");
    Serial.print(i + 1);
    Serial.print(" Temperature: ");
    Serial.print(temp);
    Serial.println(" °C");
    Blynk.virtualWrite(V5 + i, temp); // Send individual sensor data to Blynk
  }

  // Calculate average temperature
  float totalTemp = 0;
  for (int i = 0; i < NUM_SENSORS; ++i) {
    totalTemp += DS18B20_D5.getTempCByIndex(i);
  }
  float averageTemp = totalTemp / NUM_SENSORS;

  // Turn off relay if average temperature exceeds 50°C
  if (averageTemp > 50) {
    digitalWrite(RELAY_PIN_D0, LOW);
    Serial.println("Average temperature above 50°C. Turning off relay.");
  }

  sensorValue = analogRead(analogInPin);
  voltage = (((sensorValue * 27) / 1026) * 2 + calibration); // multiply by two as voltage divider
  bat_percentage = mapfloat(voltage, 37, 54.5, 0, 100);       // 37V as Battery Cut off Voltage & 54.5V as
Maximum Voltage
  if (bat_percentage >= 100)
  {
    bat_percentage = 100;
  }
  if (bat_percentage <= 0)
  {
    bat_percentage = 1;
  }
```

```
// Read the current value from the ACS712 sensor
int16_t adc0 = ads.readADC_SingleEnded(0); // Read the ADC value from channel 0 (A0 of ADS1115)
float volts0 = ads.computeVolts(adc0);

current = (volts0 - 2.5) / 0.066; // Calculate current using ACS712 sensitivity

// Ensure current is not negative and less than 0.1A
if (current < 0.1) {
  current = 0.0;
}

// Send data to Blynk
Blynk.virtualWrite(V1, averageTemp); // for Temperature
Blynk.virtualWrite(V3, voltage);           // for battery voltage
Blynk.virtualWrite(V4, bat_percentage);    // for battery percentage
Blynk.virtualWrite(V0, current);           // for current

// Print data on serial monitor
Serial.print("averageTemperature: ");
Serial.print(averageTemp);
Serial.println(" °C");

Serial.print("Analog Value = ");
Serial.println(sensorValue);
Serial.print("Output Voltage = ");
Serial.println(voltage);
Serial.print("Battery Percentage = ");
Serial.println(bat_percentage);
Serial.print("Current: ");
Serial.print(current);
Serial.println(" A");

if (bat_percentage <= 30)
{
  Serial.println("Battery level below 30%, Charge battery on time");
  // send notification
  Blynk.logEvent("battery_low", "Battery is getting low.... Plugin to charge");
  delay(500);
}

// Display temperature on OLED
display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(0.7);
display.setTextColor(SSD1306_WHITE);

display.setCursor(0, 30);
display.print("Temp: ");
display.print(averageTemp);
display.print(" *C");

// Display voltage
display.setCursor(0, 40);
display.print("Voltage: ");
display.print(voltage);
display.print(" V");

// Display current
```

```
  display.setCursor(60, 50);
  display.print("I: ");
  display.print(current);
  display.print(" A");

  // Display soc
  display.setCursor(0, 50);
  display.print("SOC: ");
  display.print(bat_percentage);
  display.print(" %");

  display.display();
  delay(1500);
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```