PROBLEM STATEMENT :

POWER MANEGMENT TELEMETRY

IoT-Based Electrical Vehicle's Battery Management and Monitoring System

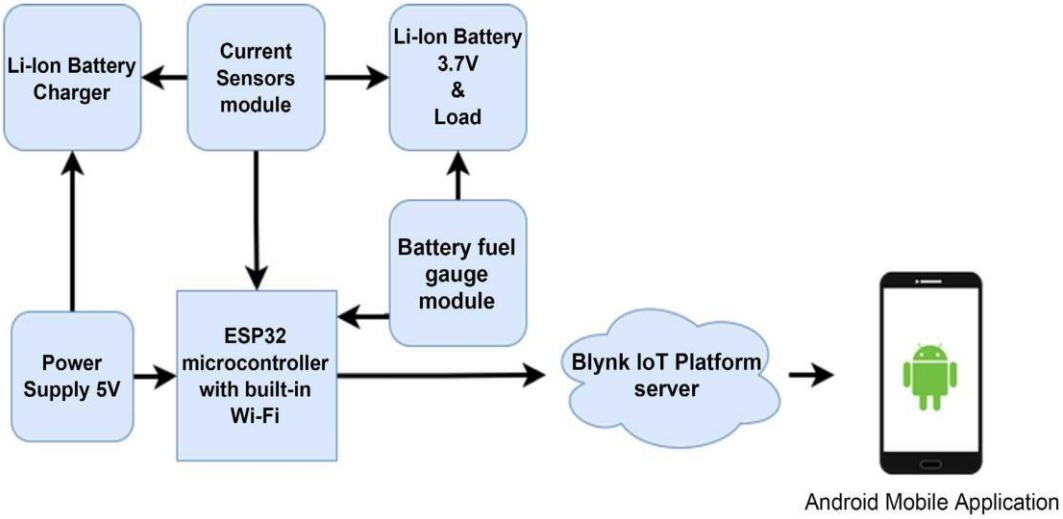
INTRODUCTION:

In today's hectic environment, electric vehicles play an important role in mobility. Electric vehicles (EVs) produce no emissions and help to keep our environment clean. To help the global environment grow green, the Indian government has tackled and launched the upgrading and manufacture of electric cars in the country. Electric vehicles improve power efficiency and provide fuel alternatives. EVs are battery electric vehicles that run entirely on energy and are more efficient than others. A hybrid electric vehicle is one that uses both an engine and a battery. A fuel cell electric car is one that operates on electricity generated by chemical energy**.** Electric vehicles (EVs) have emerged as a promising solution for sustainable transportation. However, one of the major challenges in EVs is the limited range of travel, which is dependent on the capacity and health of the battery. Therefore, it is crucial to monitor the state of the battery to ensure the reliable and efficient use of EVs. In recent years, the Internet of Things (IoT) has gained significant attention in various industries, including automotive, due to its potential to provide real-time monitoring and control of devices remotely. The application of IoT in EVs can improve the performance and efficiency of the battery, as well as enhance the driving experience of the users. This proposes an IoT-based battery monitoring system for electric vehicles. The system consists of battery sensors, microcontroller, wireless communication module, and cloud server. The battery sensors measure the voltage, current, and temperature of the battery and send the data to the microcontroller. The microcontroller processes the data and transmits it to the cloud server through the wireless communication module. The cloud server stores the data and analyzes it to generate insights about the battery's health. The proposed system provides real-time monitoring of the battery's state, enabling the optimization of the battery's performance and prolonging its lifespan. Moreover, the data generated by the system can be used to predict the remaining range of the EV, which can help the driver plan the journey more efficient.

UNIQUE IDEA BRIEF EXPLANATION :

1.Battery Management and Monitoring System Block Diagram :

- The proposed system is shown in figure designed for EV users based on IoT. the implemented system can be realized in the mobile application that allows to monitors and management whole system, which allows users to monitor EV battery status (SoC), input current, output current, in addition, it gives a summary of total charging current also it manages the charging operation. All these parameters are updated and displayed in real-time. The proposed system is composed of the following parts, a microcontroller ESP32 with built-in Wi-Fi, two Current sensors, Li-Ion Battery, a Li-Ion battery fuel gauge sensor, and Li-Ion Battery Charger. Also, the total charging current's summary after each charging operation is received as a message through a mobile telegram application. The biggest reason why select ESP32 is that it has built-in Wi-Fi and Bluetooth. So there is no need for additional radio modules like on most Arduino boards. The ESP32 is just one chip, with everything in 1 package. The system operates by connecting two current sensors and a Li-Ion battery fuel gauge sensor to ESP32; the information received from sensors is collected and processed by ESP32 and uploaded to Blynk Server through the Wi-Fi module bulletin ESP32. All information is displayed and updated in real-time, and the charging operation managed through the android mobile application.



2. Components used:

a) b) c) d) f)

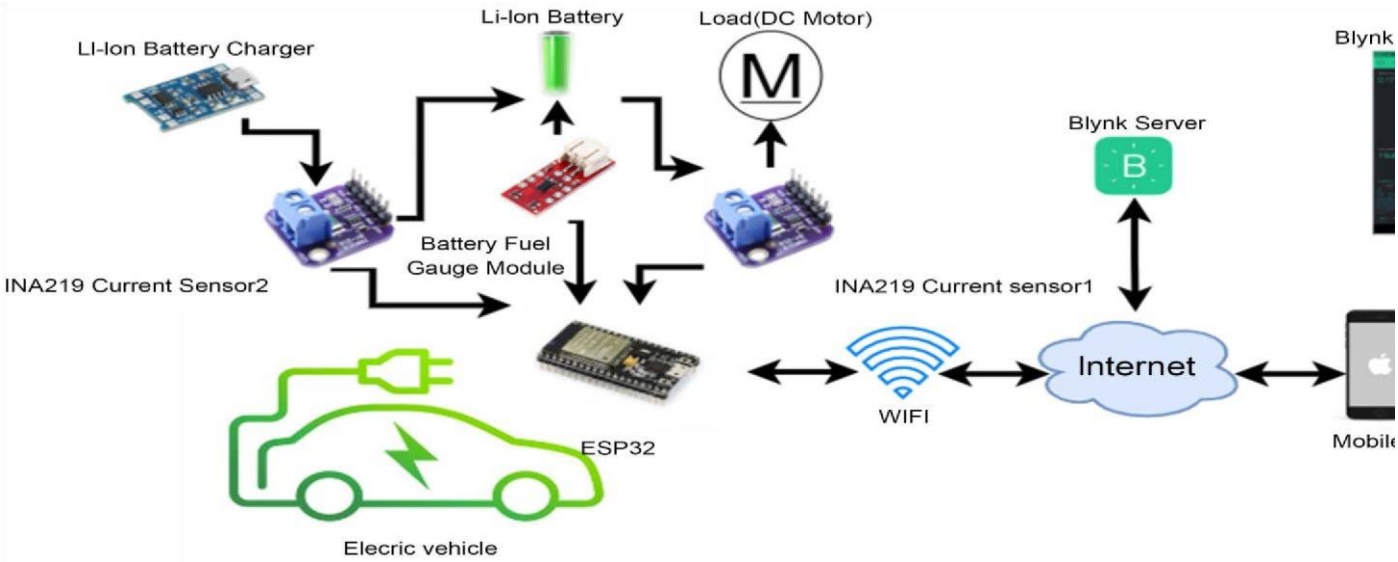



A) Microcontroller ESP32
B) INA219 current sensor
C) Relay
D) TP 4056 Li-Ion Battery
E) Charging Board Battery
F) fuel gauge module MAX 17043
G) INA219 DC current sensor

ARCHITECTURAL DIAGRAM:

- Block diagram of the battery management and monitoring system:

The system notifies the user to necessary charge the battery when the Battery SoC becomes 30% through Blynk mobile application, as shown in Figure 3, to

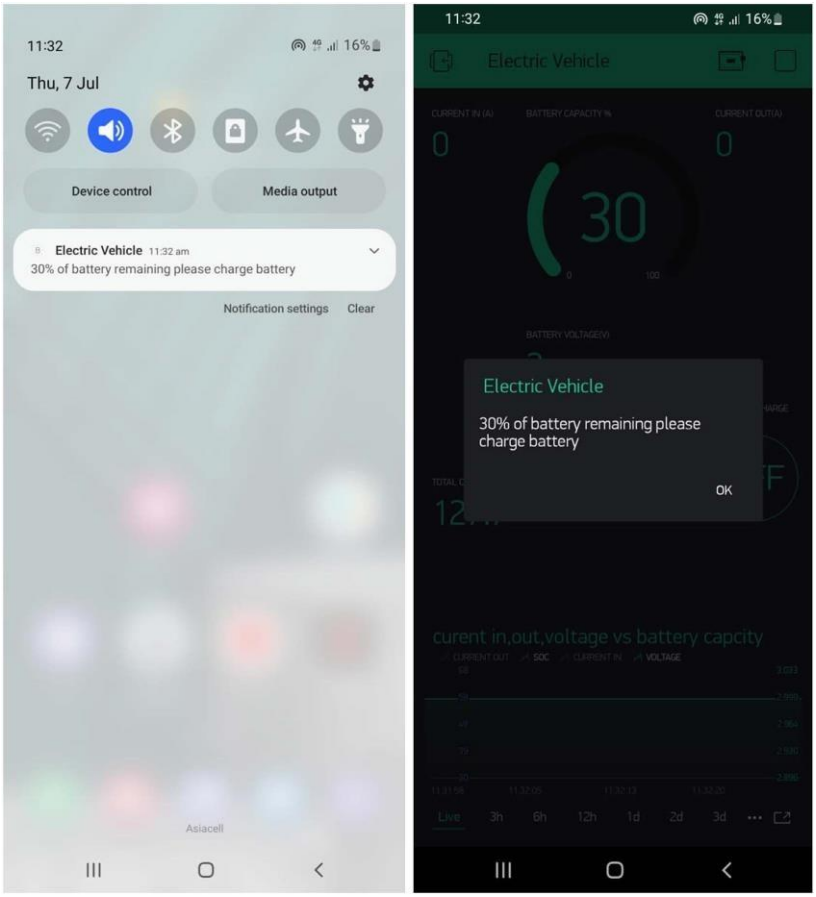


protect the battery from over-discharge damage and increase its life. The charging operation ends either by pressing the off button or when the battery is full. The Li-Ion Battery Charger stops the charging process using a protected circuit built into it.

TECHNOLOGIES USED:

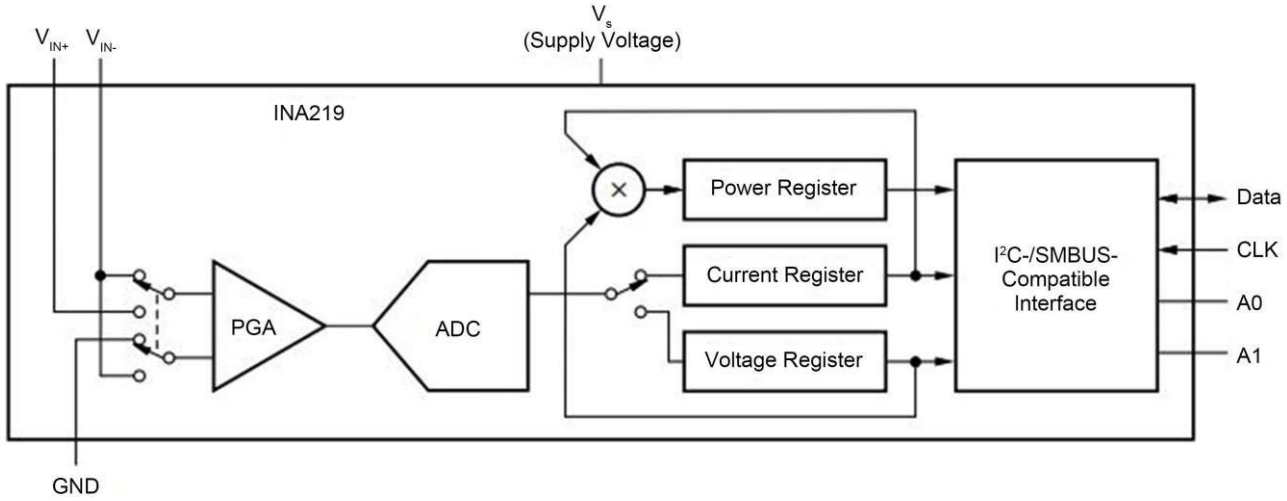
- **Battery Monitoring and Management System Scenario**

Using ESP32 to collect data from the sensor and process it, using i2c protocol to connect two current sensors and battery fuel gauge module to the same serial port. SDA and SCL data transfer merely utilizes two bi-directional lines. Both of these lines are cranked up. Serial Data (SDA) is used to transfer data, while a serial Clock (SCL) carries the clock signal. Each clock's high to low



- Charge-Discharge Management Relay :

The charging operation is managed using Relays. Relays are electromechanical or electronic switches that open and shut circuits shown in Figure 4(c). Relays open and close connections separated electrically between two circuits and joined them smoothly. When a relay contact is normally open (NO), as shown in figure.



- TP 4056 Li-Ion Battery Charging Board :

A TP4056 1A Li-Ion Battery Charging Board has Micro USB, it is a tiny module with current protection shown in Figure 4(d). It is ideal for charging a single cell of 3.7 V, 1 Ah or greater lithium-ion (Li-Ion) batteries without their protection circuit, such as 16550s. This module will give a 1 A charge current, when the charge is complete it is switching off, thanks to the TP4056 charger IC and the DW01 battery protection IC. Also, consider the case where the battery voltage falls below 2.4 V. The safety IC will then turn off the load to keep the cell from running at too low a voltage, as well as protect against overvoltage and reverse polarity connections (it will usually destroy itself instead of the battery.

Additional features include a current meter, under-voltage lockout, automated recharging, and two status pins to presence of an input voltage and signify charge termination.

- Li-Lon Battery Fuel Gauge Module MAX 17043 :

The MAX17043/MAX17044 shown are the host-side fuel-gauge systems for lithium-ion (Li+) batteries in handheld and portable devices that are ultracompact and low-cost. The MAX17043 is set up to work with one lithium cell, whereas the MAX17044 is set up to work with a two cell pack. The MAX17043/
MAX17044 employ Model-Gauge, a sophisticated Li+ battery-modelling

system, to continually track the battery's relative SoC through a broad range of charge/ discharge profiles. Unlike standard gasoline gauges, the Model-Gauge algorithm does not require battery relearn cycles or an additional currentsense resistor. With minimum contact between micro coulombs and the device, temperature correction is achievable in the application. Integrated circuit can be placed on the system side, decreasing the battery's cost and sourcing limitations. An I2C interface is used to access data sets for measurement and capacity estimation.

The model accurately replicates the internal dynamics of a Li+ battery and calculates the state of charge. The model considers the battery's time effects generated by chemical reactions and impedance. The SoC calculation for MAX17043/ MAX17044 does not acquire errors over time. Compared with typical coulomb counters, which suffer from SoC drift due to current-sense offset and cell self- discharge, this is a benefit. This model performs well for various Li+ chemistry variations at multiple temperatures and ages. The MAX17043/MAX17044 must be configured with application-specific configuration data to ensure optimal performance.

- IoT Based Battery Management and Monitoring System :

Blynk is the most user-friendly IoT platform, with an application builder that runs on iOS and Android. In addition, it has a collection of libraries that allow you to create incredible IoT applications in minutes using any chosen hardware platform. It allows to easily create interfaces for controlling and monitoring hardware projects from iOS or Android devices by simply dragging and dropping widgets A digital dashboard is a graphical user interface (GUI) formed. Widgets can be dragged and dropped from the widget box. Every widget to uses some form of points to function. A user will get 2000 energy (points) when establishing a Blynk account. When a widget is utilized, the energy balance decreases Blynk mobile application allows users to watch the electric vehicle battery capacity, current input/output battery currents consumed by the load, and manage the charging operation (start and stop) through a button on the digital application dashboard. After the battery charging process is complete, a message will be sent to the Telegram

application containing the total amount of current used in the charging operation.

Design of Battery Management and Monitoring System Circuit :

And the circuit connection design is shown in Figure 7. Because INA 219 and MAX17043 use the Inter-Integrated Circuit (I2C) Protocol, which enables one or more “controller” chips to communicate with a large number of “peripheral” digital integrated circuits (“chips”), so can connect to the same serial port on the ESP32 microcontroller. Through soldering (A0, A1) as programmable addresses to select

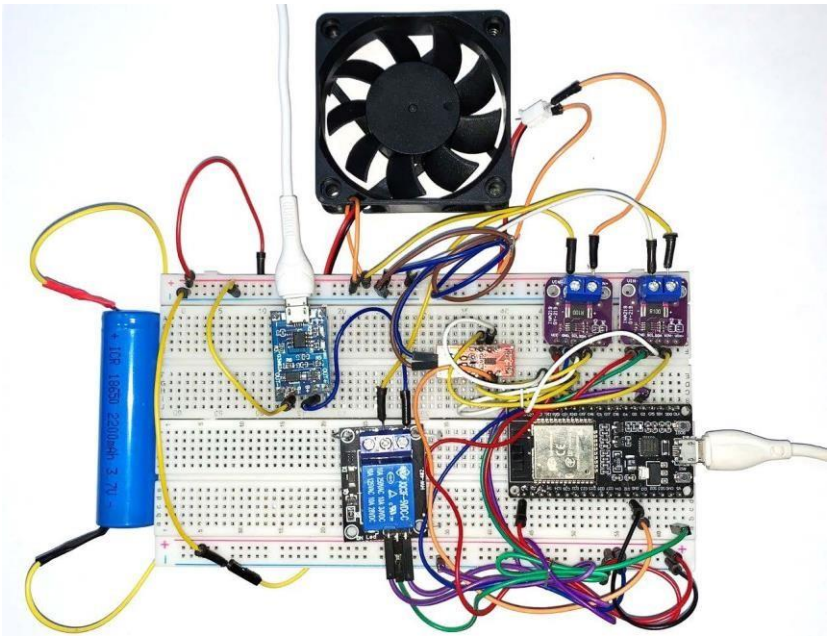


Figure 6. Prototype of battery management and monitoring system.

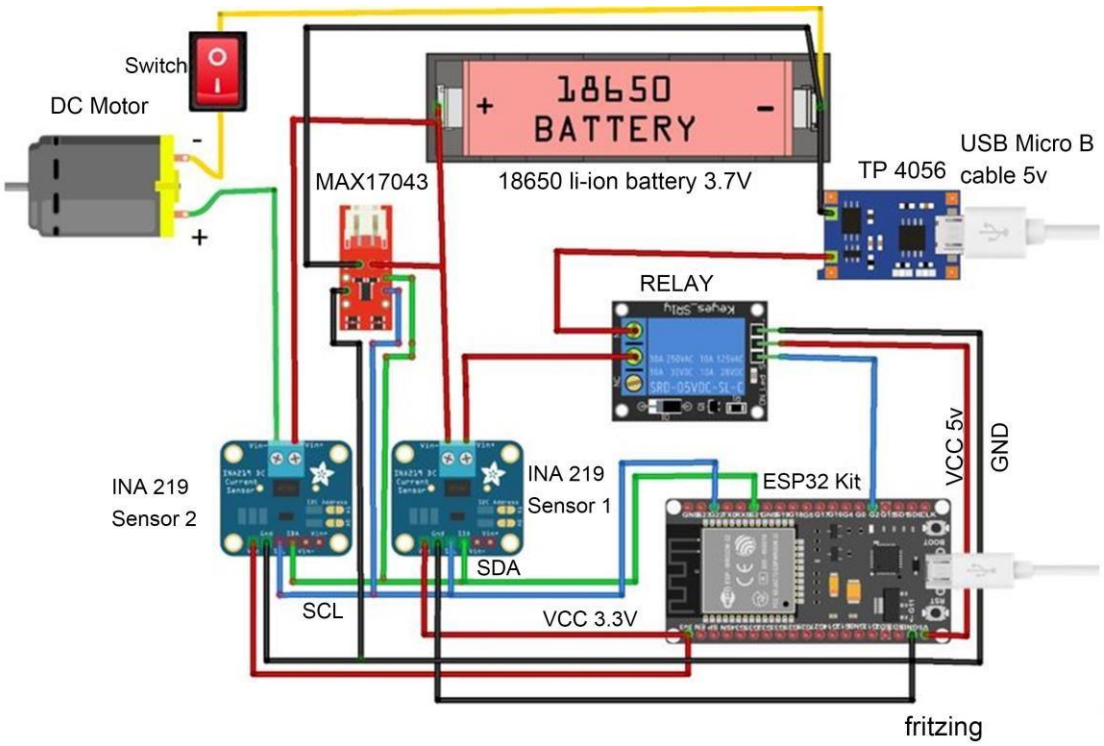


Figure 7. Proposed system circuit design.

between two INA219. There are connected Serial Clock Line (SCL) and SDA (Serial Data Line) pins from two sensors, and MAX17043 is linked to D22 and D21. VCC and GND connected from two INA219 to 3.3 v and GND pin of ESP32 to supply the sensor with needed power. MAX17043 joined as (+, −) pins connected to (+, −) battery terminal to return the battery capacity as well as (GND) pin connected to (GND) pin in ESP32. The relay used in the circuit to control the charging operation uses a normally open pin, controlled through the application. When the user presses the application dashboard, the button (press to charge) changes from off to on; it is also used to start charging. When the press button (press to charge) changes from on to off, the charging stops.

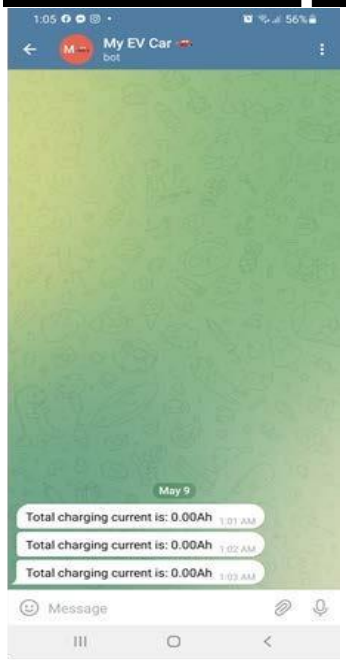
Result and Discussion :

A prototype system has been designed, implemented, and tested after connecting all components, as shown in Figure 6. The system is powered from a pc USB port and connected the ESP32 to the router using an ESP32 built-in Wi-Fi module. All Ev's data are collected and processed using ESP32, transmitted to the Blynk server and displayed by the blynk mobile application; all information is updated and displayed in real-time. The mobile application

dashboard view by the user is shown in Figure 8(a), Figure 8(b) and Figure 8(c), Figure 8(d) for the charging/discharging process respectively.

The 18650 Li-Ion battery is fully charged using TP4056 Li-Ion battery charger and then starts system testing. The mobile application is used to display information about the status of the battery, like capacity or SoC, charging/discharging current, and battery voltage. In addition, a chart gadget that describes the relationship between them. The total charging current is displayed every time press button (press to charge) ON, OFF, and then the exact number is sent to the user's telegram application. When the load is turned on, the discharge operation starts; through this operation, the capacity of the battery decrease as well as the voltage, as shown in Figure 9. To start the charging procedure, the load must be turned off. The charging process does not begin unless press the (press to

a) b) c) d)



e)

Figure . Shows application dashboard: (a) and (b) discharge operation, (c) and (d) charge operation and (e) massage received by telegram application

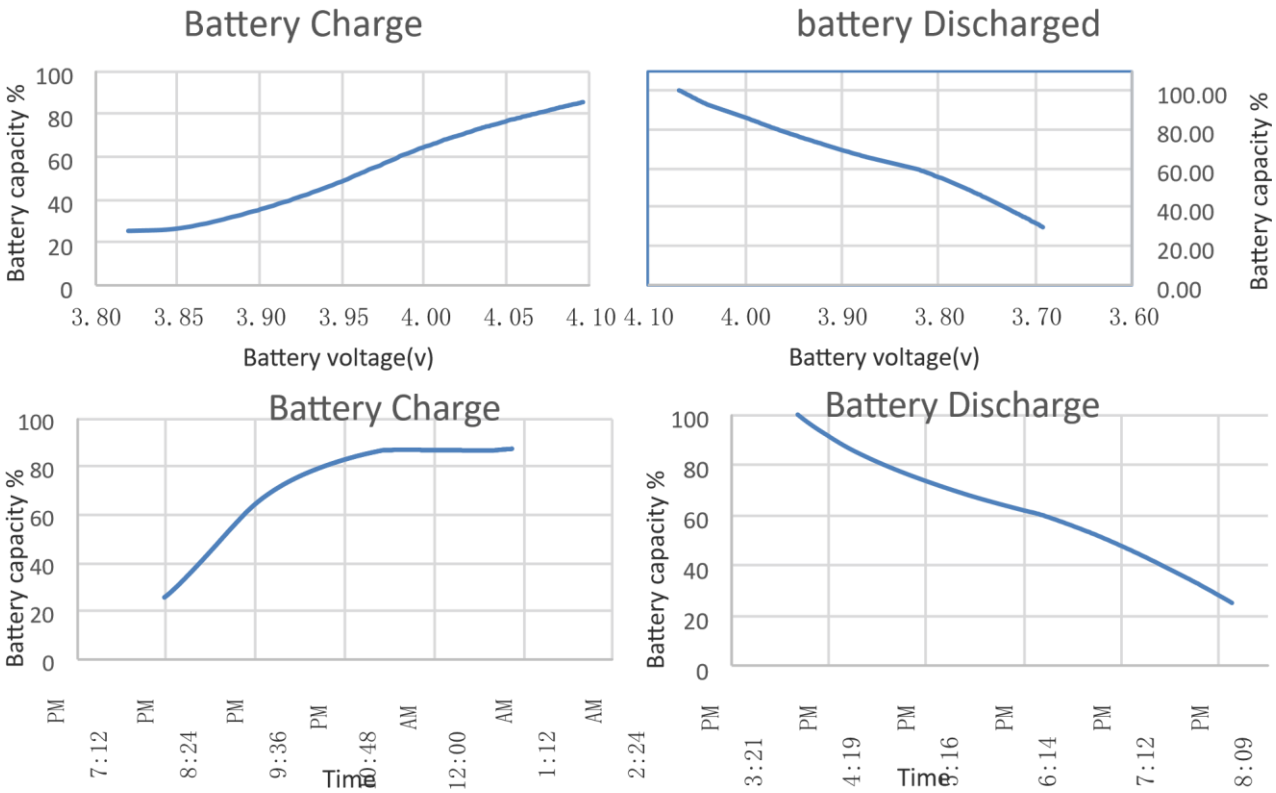


Figure . Battery charge and discharge with the voltage and time profile.

charge) button to close the relay and start charging the battery. During this operation, the capacity of the battery and voltage increase, as shown. The charging process stops in one of the two cases, either when pressing on the (press to charge) button or when the battery is fully charged through the TP4056 board protection circuit. Each time press (ON, OFF) press to charge button, a message is sent to the user's Telegram app containing a summary of the amount of current the battery was charged with. After each charging process, this feature can be used as a bill and kept as a booking to know the charging date within a specified period. The message sent to the telegram is shown in . The time taken for the charging and discharging process is illustrated in with an average charging current (0.24 AH) and average discharging current (0.195 AH).

Software designing programme:

```
#include <ESP8266WiFi.h>

String apiKey = "******"; const char* ssid =  "******";     // Enter
your WiFi Network's SSID const char* pass =  "******"; // Enter your
WiFi Network's Password const char* server = "api.thingspeak.com";
```

```
int analog In Pin  = A0;    // Analog input pin int
sensor Value;          // Analog Output of Sensor
float calibration = 0.36; // Check Battery voltage using multimeter & add/subtract the value int
bat_percentage;

WiFi Client;

void setup()
{
  Serial.begin(115200);
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(100);
    Serial.print("*");
  }
  Serial.println("");
  Serial.println("WiFi connected");
}

void loop()
{
   sensorValue = analogRead(analogInPin);

  float voltage = (((sensorValue * 3.3) / 1024) * 2 + calibration); //multiply by two as voltage
divider network is 100K & 100K Resistor

  bat_percentage = mapfloat(voltage, 2.8, 4.2, 0, 100); //2.8V as Battery Cut off Voltage &
4.2V as Maximum Voltage

  if (bat_percentage >= 100)
  {
    bat_percentage = 100;
  }
  if (bat_percentage <= 0)
  {
    bat_percentage = 1;
  }

  Serial.print("Analog Value = ");
```

```
  Serial.print(sensorValue);
  Serial.print("\t Output Voltage = ");
  Serial.print(voltage);
  Serial.print("\t Battery Percentage = ");
Serial.println(bat_percentage);   delay(1000);

  if (client.connect(server, 80))
  {
    String postStr = apiKey;     postStr
+= "&field1=";     postStr +=
String(voltage);     postStr +=
"&field2=";     postStr +=
String(bat_percentage);
    postStr += "\r\n\r\n";
   client.print("POST /update HTTP/1.1\n");
delay(100);
    client.print("Host: api.thingspeak.com\n");
delay(100);
    client.print("Connection: close\n");
delay(100);     client.print("X-THINGSPEAKAPIKEY:
" + apiKey + "\n");     delay(100);
    client.print("Content-Type: application/x-www-form-urlencoded\n");
delay(100);
    client.print("Content-Length: ");
delay(100);
    client.print(postStr.length());
delay(100);     client.print("\n\n");
delay(100);
client.print(postStr);
delay(100);
  }
  client.stop();
  Serial.println("Sending....");
  delay(15000);
}

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;
}
```

CONCLUSION :

The electric vehicle's battery management and monitoring system based on IoT, Using two current sensors INA219 to measure current in/out from battery

and battery fuel gauge module MAX17043 for SoC calculation, they are connected to ESP32 microcontroller can use IoT capability through connecting the ESP32 to the internet, this allow users to manage battery charge/discharge; as well as monitoring the battery SoC status by displaying all information on the mobile application dashboard and updated in real-time. Also, the suggested management system allows the user to detect any current consumption through the current out widget on the application dashboard. That led to extending battery life by protecting it from overcharge and over discharge.

The method used to calculate battery SoC advantageous compared to traditional the classical coulomb-counter-based fuel gauges suffers from accuracy drift due to the in the current-sense measurement, there is an offset error. Although the inaccuracy in such systems is frequently little, it grows over time, cannot be eliminated, and necessitates periodic corrections. Corrections are typically made at a predetermined SoC level, which is usually near full or empty. Other systems modify the voltage of the battery when it is relaxed. After an extended period of inactivity, these systems calculate the accurate SoC depending on the battery voltage. Both share the same limitation: the error in the system is limitless if the correction condition is not observed over time in the actual application.

In some cases, a full charge/discharge cycle is required to eradicate the drift inaccuracy. To establish the factual correctness of a fuel gauge as experienced by end-users, the battery should be dynamically exercised. The end-user accuracy cannot be grasped with simply simple cycles. Because they do not rely on current information, MAX17043/MAX17044 does not suffer from drift.