Smart Battery management system for Electric Vehicle (EV) using IOT

Shreeram V Kulkarni
Dept. of Electrical and Electronics Engineering,
Nitte Meenakshi Institute of Technology
Bengaluru, India
Shreeram.kulkarni@nmit.ac.in

Praful Kalakappa Vyapari
Dept. of Electrical and Electronics Engineering,
Nitte Meenakshi Institute of Technology
Bengaluru, India
prafulvyapari1212@gmail.com

Shayan Kumar M
Dept. of Electrical and Electronics Engineering,
Nitte Meenakshi Institute of Technology
Bengaluru, India
Shayankumarm05@gmail.com

Pratheek B Patil
Dept. of Electrical and Electronics Engineering,
Nitte Meenakshi Institute of Technology
Bengaluru, India
pratheekpatil5@gmail.com

Rajendra S Keshannavar
Dept. of Electrical and Electronics Engineering,
Nitte Meenakshi Institute of Technology
Bengaluru, India
sampreetkeshannavar@gmail.com

Abstract

As we all know, the demand for electric vehicles (EVs) is increasing rapidly every year. According to recent reports, nearly 80 million EVs are expected to be on the road by 2030. While this change toward cleaner transportation is a positive step for our environment, it also brings new challenges, especially when it comes to managing the performance and safety of EV batteries.

In our project, we've developed a smart BMS prototype using simple and cost-effective components. At the heart of the system is the ESP32 microcontroller, which serves as the brain of the setup. We've used a standard 18650 lithium-ion battery as our power source. For sensing current, we've integrated the ACS712 sensor, while the DHT11 sensor monitors both temperature and humidity. An LCD display has been added to provide real-time feedback to the user, showing live data such as temperature, current flow, and SOC estimates.

This setup not only helps us understand how the battery behaves under different conditions, but it also demonstrates how even a basic BMS can significantly improve the safety and performance of battery-powered systems. By combining IoT capabilities with essential sensors, our system allows for real-time monitoring and opens the door for future enhancements like remote data logging and predictive battery analytics.

INTRODUCTION

Despite the undeniable benefits, Internal combustion engines have been a major source of environmental concerns. The burning fossil fuels, primary gasoline and diesel releases carbon dioxide and other pollutants in the atmosphere contributing to the global warming and air pollution

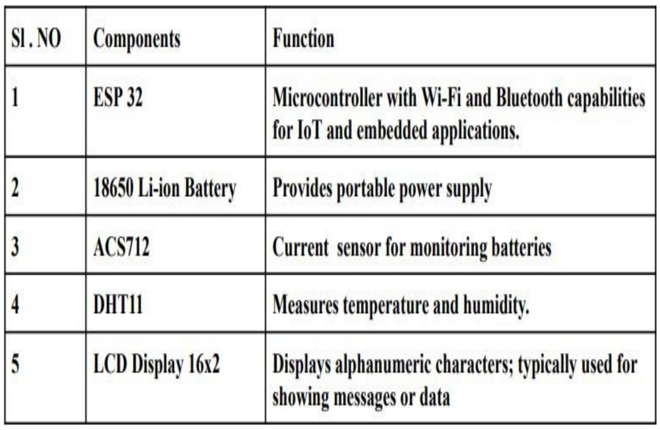
Generally, a battery is prone to excessive heat overheating of device or overcharging of battery's leads to more heat this axis heat in further leads the cell battery life causing thermal runaway and accidents in some cases, BMS framework assesses and shows the battery temperature, charging/releasing current, and SOC for the thought about model battery. For observing, computerized and simple sensors with microcontrollers are utilized.

The project we're focusing on right now is smart BMS EV which contains esP32 as microcontroller and sensors like ACS712 as current sensor, DHT11 as temperature sensor and LCD Display for monitoring purpose and views WIFI connectivity for real time analysis by which enhancing battery performance, safety, and efficiency, the proposed solution supports the adoption of EVs and promotes the transition to sustainable energy.

IoT enabled real time battery management system enhances SOC and SOH monitoring. Continuous real time monitoring would offer more accurate and responsive data for decision making and system optimization. By integrating ESP32 gives you real time datalogging and remote monitoring via thingspeak IoT-enabled BMS systems utilize wireless sensors and cloud platforms for data storage and analysis,

Integration of IoT technology for remote battery monitoring has shown significant advancement and developed an iot based real time monitoring system for battery management in electrical vehicles using WIFI as in wireless communication for data transmission and remote monitoring. Recently it has showed dynamic monitoring capabilities allowing real time data collection and remote access which enhances both SOC, SOH, voltage, current, temperature and humidity through the sensors that has been connected to ESP32 By this innovation it will help in the crucial for enhancing BMS efficiency reliability safety particularly EV and energy storage application

Components of a Smart Battery management system for Electric Vehicle (EV)



Sl . NO	Components	Function
1	ESP 32	Microcontroller with Wi-Fi and Bluetooth capabilities for IoT and embedded applications.
2	18650 Li-ion Battery	Provides portable power supply
3	ACS712	Current sensor for monitoring batteries
4	DHT11	Measures temperature and humidity.
5	LCD Display 16x2	Displays alphanumeric characters; typically used for showing messages or data

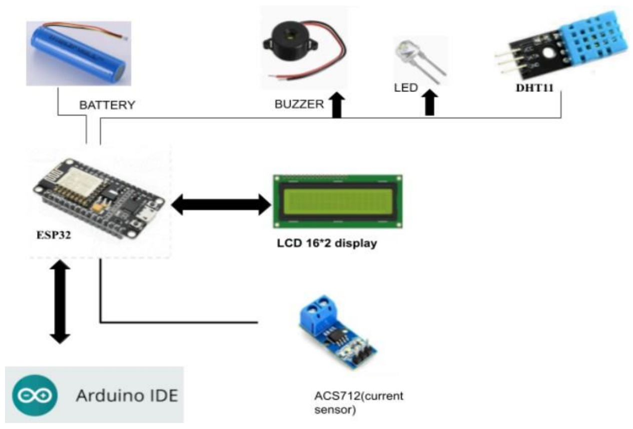


Fig. 1. Interfacing of components

1. ESP32 Microcontroller

ESP32 is a microcontroller that manages data processing and wireless communication, which supports GPIO interfaces like PWM, ADC, and I2C. It features 520 KB SRAM, 4 MB external RAM, and operates at 3.3V, and also offers low-power modes for energy efficiency.

2. 18650 Lithium-Ion Battery

The 18650 battery powers the whole system and microcontroller with a nominal voltage of 3.7V and capacities ranging from 2200mAh to 3600mAh. It mainly supports 500–1000 charge cycles and integrates a BMS from overcharge, discharge, and temperature protection.

3. ACS712 Current Sensor

This current sensor provides precise current measurements by using magnetic fields, with variants for ±5A to ±30A and sensitivity up to 66–185 mV/A. It also operates at 4.5–5.5V and outputs analog data compatible with the ESP32.

4. DHT11 Sensor

The DHT11 temperature sensor measures temperature up to (0°C–50°C) and humidity (20%–90% RH) with accuracies of ±2°C and ±5% RH. Operating at 3.3–5V, it provides a digital output with a 1Hz sampling rate.

5. LCD 16x2 Display

The LCD displays show real-time data like voltage and temperature. It operates at 5V with I2C interface support, consuming 20mA with the backlight on.

CODE DETAILS

```
#include <Wire.h>
#include <DHT.h>
#include <WiFi.h>
#include <ThingSpeak.h>

#define DHTPIN 32
#define DHTTYPE DHT11
#define ACS712_PIN 5
#define BUZZER_PIN 25
#define LED_PIN 26
#define MOTOR_PIN 14

const char* ssid =
"prafulvyapari@1022";
const char* password = "praful
vyapari";
const char* apiKey =
"5K7WOBHA75ZAN4ZH";
const unsigned long channelID =
2800647;

WiFiClient client;
DHT dht(DHTPIN, DHTTYPE);
```

```
float current = 0.0, voltage = 7.4,
soc = 100.0, soh = 90.0, sop = 80.0,
temperature = 0.0, humidity = 0.0;
unsigned long previousMillis = 0;
const long interval = 5000;

void setup() {
  Serial.begin(115200);
  dht.begin();
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(MOTOR_PIN, OUTPUT);
  digitalWrite(MOTOR_PIN, HIGH);
  Serial.println("Smart BMS
Starting...");
  delay(2000);
  connectToWiFi();
  ThingSpeak.begin(client);
}

void loop() {
  if (millis() - previousMillis >=
interval) {
    previousMillis = millis();
    current = readCurrent();
    temperature =
dht.readTemperature();
    humidity = dht.readHumidity();

    if (isnan(temperature) ||
isnan(humidity)) {
      Serial.println("Failed to read
from DHT sensor!");
      temperature = 0.0; humidity =
0.0;
    }

    updateBatteryParameters();
    controlMotorAndBuzzer();
    displayOnSerial();
    sendDataToThingSpeak();
  }
}
```

```
float readCurrent() {
  int sensorValue =
analogRead(ACS712_PIN);
  float voltageSensor = (sensorValue /
4095.0) * 3.3;
  float current = (voltageSensor -
2.5) / 0.066;
  return abs(current);
}

void updateBatteryParameters() {
  soc -= (current * 0.01);
  if (soc < 0) soc = 0;
}

void controlMotorAndBuzzer() {
  if (temperature > 30.0) {
    digitalWrite(MOTOR_PIN, LOW);
    digitalWrite(BUZZER_PIN, HIGH);
    Serial.println("Temperature is
above 30°C. Motor stopped, buzzer
ON.");
  } else {
    digitalWrite(MOTOR_PIN, HIGH);
    digitalWrite(BUZZER_PIN, LOW);
    Serial.println("Temperature is
below 30°C. Motor ON, buzzer OFF.");
  }
}

void displayOnSerial() {

Serial.println("=====================
=======");
  Serial.println("Smart BMS Data:");
  Serial.print("SOC: ");
Serial.print(soc, 1); Serial.println("
%");
  Serial.print("SOH: ");
Serial.print(soh, 1); Serial.println("
%");
  Serial.print("Voltage: ");
Serial.print(voltage, 1);
Serial.println(" V");
```

```
  Serial.print("Current: ");
Serial.print(current, 1);
Serial.println(" A");
  Serial.print("Temperature: ");
Serial.print(temperature, 1);
Serial.println(" C");
  Serial.print("Humidity: ");
Serial.print(humidity, 1);
Serial.println(" %");

Serial.println("======================
=======");
}

void sendDataToThingSpeak() {
  ThingSpeak.setField(1, soc);
  ThingSpeak.setField(2, soh);
  ThingSpeak.setField(3, sop);
  ThingSpeak.setField(4, voltage);
  ThingSpeak.setField(5, current);
  ThingSpeak.setField(6, temperature);
  ThingSpeak.setField(7, humidity);
  int response =
ThingSpeak.writeFields(channelID,
apiKey);
  if (response == 200) {
    Serial.println("Data sent to
ThingSpeak successfully!");
  } else {
    Serial.print("ThingSpeak write
failed, error code: ");
    Serial.println(response);
  }
}

void connectToWiFi() {
  Serial.print("Connecting to
WiFi...");
  WiFi.begin(ssid, password);
  unsigned long startAttemptTime =
millis();
  while (WiFi.status() !=
WL_CONNECTED) {
    if (millis() - startAttemptTime >=
10000) {
      Serial.println("Failed to
connect to WiFi");
      return;
    }
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}
```

SOFTWARE AND CLOUD INTEGRATION

As electric vehicles (EVs) become more popular, the need for reliable and efficient Battery Management Systems (BMS) grows. In this project, we've developed a smart BMS using the ESP32 microcontroller to keep track of important battery parameters such as voltage, current, temperature, and State of Charge (SOC). The ESP32 works with sensors like the ACS712 current sensor and DHT11 temperature sensor to measure these values and also calculate the State of Health (SOH) and State of Power (SOP). These things help us understand how well the battery is performing and how long it will last.

The system is connected to the ThingSpeak cloud platform, which lets us see all the data in real-time. With Wi-Fi, the ESP32 sends the data to ThingSpeak, where it's displayed on a dashboard. This setup allows users to monitor the battery remotely and get alerts if something goes wrong, like if the battery gets too hot or if the charge level drops too low. For example, if the temperature rises above a safe limit or if the SOC falls below 10%, the system will display an alert on an LCD screen and sends notifications via ThingSpeak.

One of the biggest benefits of at least using ThingSpeak for this is that this is a not just a live feed but it's also a historic feed and you can go back in time and see it. This allows users to monitor the battery health over time and to catch any potential problems quickly. With this good battery condition, we can prolong the battery's life and enhance the safety. This intelligent BMS is a key advancement that will help make electric vehicles and clean energy systems more reliable, safer and last longer.

Real-Time Monitoring and Data Acquisition Overview:

Real-time monitoring is all about collecting data from sensors to track the battery parameters such as voltage, temperature, and state of charge. This data is then transmitted to the cloud platform for remote monitoring process:

• Data Collection: Sensors used here, measure voltage and temperature, which are further converted into usable data.

.● Data Transmission:The system sends this data over Wi-Fi to the ThingSpeak cloud platform every 5 seconds for remote monitoring.

• Visualization: On ThingSpeak, this data is displayed on a live dashboard, allowing users to see exactly how the battery is performing at any given moment. The dashboard shows key metrics like current voltage levels, temperature, and the battery's state of charge in real-time. This means you can instantly spot any unusual patterns or trends.

Cloud-Based Data Logging and Analysis Overview:

Cloud-based logging is all about saving sensor data in the cloud so you can look back at it later for trend analysis, spotting patterns, and making better decisions.

• First, the data is uploaded to ThingSpeak, where it's stored as time-series data, meaning it's organized by time, so you can track how things change over time.

• Then, ThingSpeak offers tools that let you analyze these trends, and you can even export the data if you want to dive deeper into the details.

Through this process Cloud logging ensures secure, long-term storage of battery performance data, enabling trend analysis and proactive maintenance based on historical performance.

Results;

Algorithms for SoH, SoC, and Fault Detection:

To keep track of the health of the battery, we rely on algorithms that monitor the State of Charge (SOC), State of Health (SOH), and State of Power (SOP).

• **SOC (State of Charge):** Here we can see how much power is left in the battery. It is calculated using the actual battery voltage. The formula used here is:

• SOC = (Voltage / 4.2) ×100.

• 4.2V is a fully charged battery so we can use this value to work on what percentage of charge is left

SOH (State of Health): This calculates the overall health of the battery, taking into considerations like how much capacity has been lost over time. To calculate SOH, we compare the current capacity of the battery to its original capacity: SOH = (Current Capacity / Original Capacity) × 100. This tells us if the battery is still performing like it did when it was new.

• **SOP (State of Power):** This represents the usable power in the battery. It's calculated by multiplying the SOC value by 0.9, assuming that about 90% of the charge is usable: SOP = SOC × 0.9

• **Fault Detection:** To make sure the battery is running smoothly, we monitor its performance or health like voltage, temperature, and SOC. If any of these values go out of range, we get alerted. For example:

1. If the voltage drops below 3.0V, an alert is triggered.
2. If the temperature exceeds 60°C, a high-temperature warning pops up.
3. If the SOC falls below 20%, "low-charge" alert is sent.

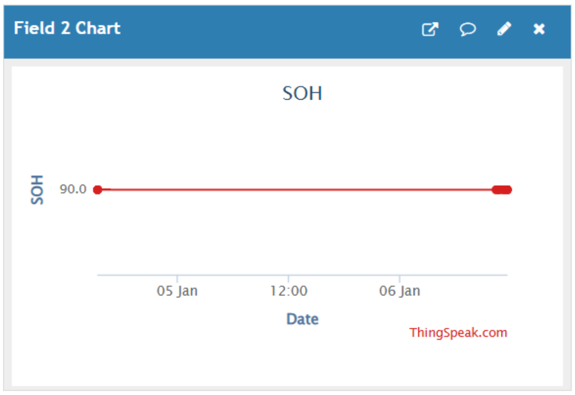


fig 2. SOC Time Plot

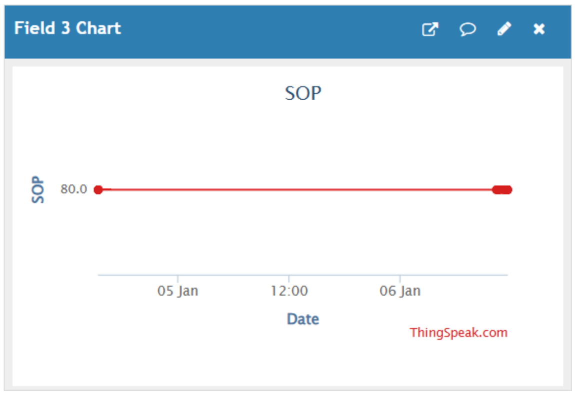


fig 3. SOP Time Plot

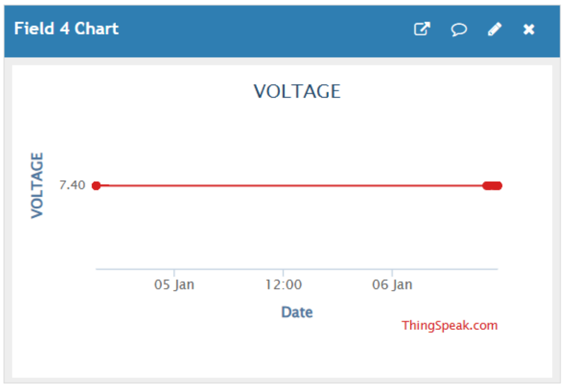


fig 4. Voltage Time Plot

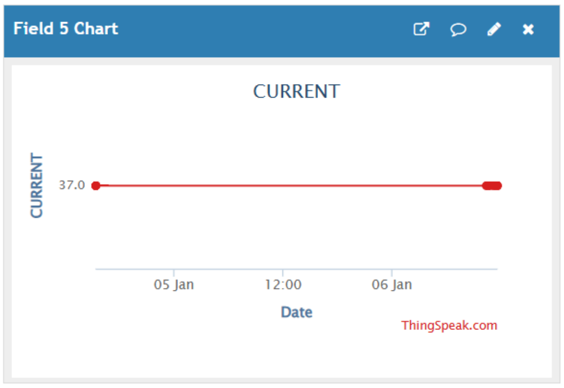


fig 5. Current Time Plot

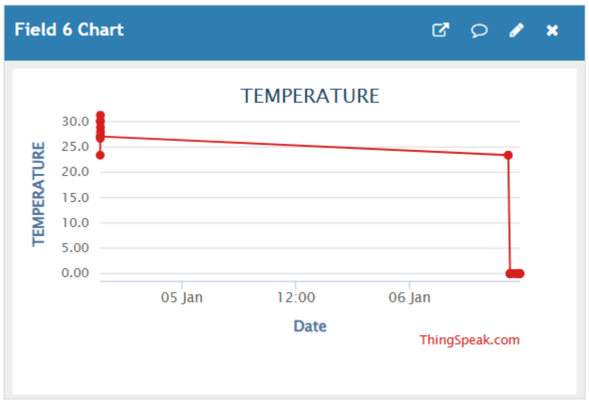


fig 6. Temperature Time Plot

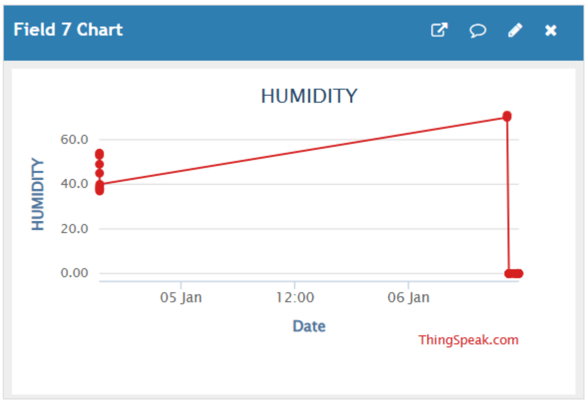


fig 7. Humidity Time Plot

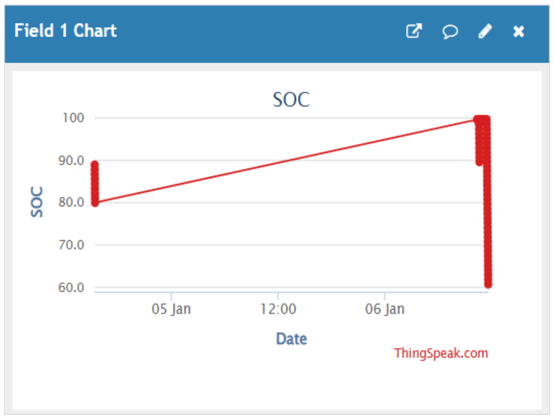
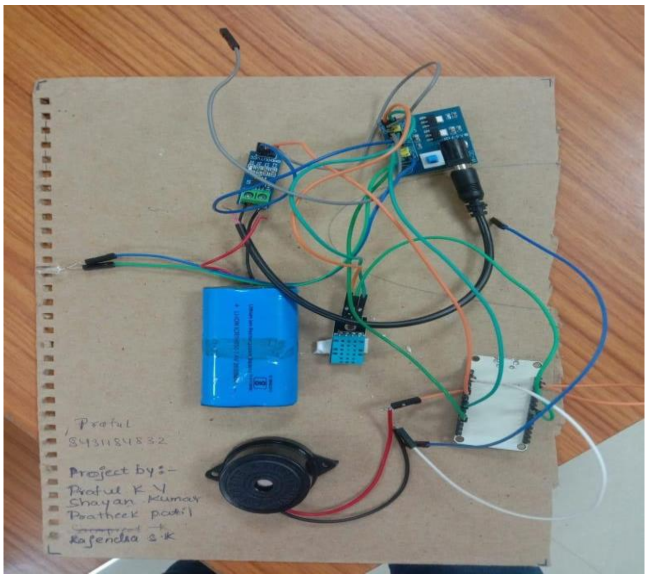


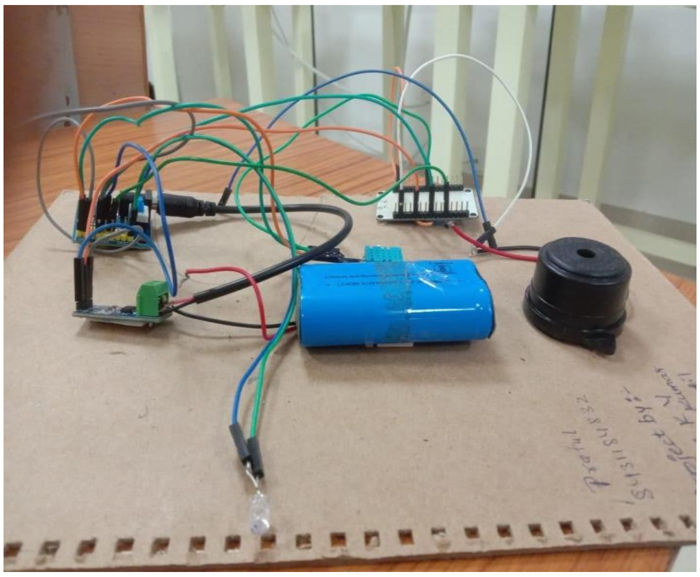
fig 8. SOC Time Plot

Data Entries;

Created At	Entry ID	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7
2025-01-04T09:32:02+00:00	1	99.63000	90.00000	80.00000	7.40000	37.00000	23.40000	52.00000
2025-01-04T09:33:24+00:00	2	99.63000	90.00000	80.00000	7.40000	37.00000	23.00000	52.00000
2025-01-04T09:33:45+00:00	3	98.51999	90.00000	80.00000	7.40000	37.00000	23.00000	54.00000
2025-01-04T09:34:00+00:00	4	97.40998	90.00000	80.00000	7.40000	37.00000	23.00000	54.00000
2025-01-04T09:34:18+00:00	5	96.29997	90.00000	80.00000	7.40000	37.00000	23.00000	54.00000
2025-01-04T09:34:36+00:00	6	95.18996	90.00000	80.00000	7.40000	37.00000	23.00000	55.00000
2025-01-04T09:34:54+00:00	7	94.44996	90.00000	80.00000	7.40000	37.00000	23.00000	55.00000

Hardware design;





Conclusion;

In this project, we’ve worked on creating a smart Battery Management System (BMS) using IoT technology to make it easier to monitor and manage EV batteries. Our model uses an ESP32 microcontroller along with a few sensors, like ACS712 to measure current and the DHT11 to track the temperature. It also comes with an LCD screen for displaying real-time measurements like voltage, current, temperature, and crucial battery states like State of Charge (SOC), State of Health (SOH), and State of Power (SOP).

The concept is simply to provide users with a device to monitor their battery condition at any time and from anywhere. For this purpose, data is transmitted to a cloud platform like ThingSpeak, and therefore accessed remotely. The system will also notify if something goes wrong, such as when the battery begins to overheat or the SOC is too low, it alerts on the LCD display. This type of warning can prevent larger issues and extend the battery life by controlling how it is charged and discharged. What is great is that, our system is designed to scale, or that would be suitable not only to individual EVs, but to the larger installations like EV fleets and solar power systems. There are still a few things to work out. It only supports a few types of batteries and is completely Wi-Fi dependent in terms of connectivity, and lacks robust data security features.

To make it even better, we’ve thought about adding features like automatic detection for different types of batteries, encryption to keep the data safe, and even machine learning to help predict potential battery issues before they happen. Switching to other types of connectivity like LoRaWAN or 5G could also make the system easier. It would be wonderful to include more advanced sensors to learn even more about battery health, and perhaps even develop a simple app to simplify the use of our system.

Overall, BMS is a move towards making battery management more simplified, safer, and more efficient, which is something that's very critical as renewable energy and EVs keep on increasing.

Bibliography;

1. Roy, Koustav, et al. "Design and implementation of an IoT-based battery management system for electric vehicles." 2020 12th International Conference on Communication Systems & Networks (COMSNETS). IEEE, 2020.
2. Tchamgoue, Adolphe Feudjio, et al. "IoT-based battery management system for electric vehicles." 2021 IEEE 19th International Conference on Industrial Informatics (INDIN). IEEE, 2021.
3. Khan, Raza, et al. "Wireless charging station for electric vehicles: A review." 2019 International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2019.
4. Kumar, Abhishek, and Deepak Verma. "Wireless charging station for electric vehicles: A comprehensive review." 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2020.
5. R. Kumar and S. K. Sharma, "IoT based real-time monitoring system for electric vehicle battery management using Arduino", 2020 2nd International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 1035-1041, 2020.
6. Mohd Helmy, Abd Wahab, Nur Imanina Mohamad Anuarl, Radzi Ambarl, Aslina Baharum, Shanoor Shantal, et al., "IoT-Based Battery Monitoring System for Electric Vehicle", International Journal of Engineering & Technology, vol. 7, no. 4.31, pp. 505-510, 2018.
7. Raza and M. H. Zafar, "Design and development of an intelligent battery management system for electric vehicles using Arduino", 2019 2nd International Conference on Electrical Communication Electronics Instrumentation and Computing (ICECEIC), pp. 1-6, 2019.
8. S. Kim, Y. Lee and B. Park, "Real-time battery management system for electric vehicles based on wireless communication", 2018 IEEE Wireless Power Transfer Conference (WPTC), pp. 1-4, 2018.
9. Y. Shang, Q. Zhang, N. Cui, B. Duan and C. Zhang, "An optimized mesh-structured switched-capacitor equalizer for lithium-ion battery strings", IEEE Trans. Transport. Electrific., vol. 5, no. 1, pp. 252-261, Mar. 2019.
10. V. Monteiro, J.P. Carmo, J. G. Pinto and J. L. Afonso, "A flexible infrastructure for dynamic power control of electric vehicle battery chargers", IEEE Transactions on Vehicular Technology, vol. 65, no. 6, pp. 4535-4547, 2015.
11. V. Monteiro, J. G. Pinto and J. L. Afonso, "Operation modes for the electric vehicle in smart grids and smart homes: Present and proposed modes", IEEE Transactions on Vehicular Technology, vol. 65, no. 3, pp. 1007-1020, 2015.
12. T. Karthi, S. Maheswaran, S. Sathish, K. Nivethitha, B. Gangadharan and S. Santhosh, "Wireless Charging Technology for an Electric Vehicle", 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-6, 2023, July.