Project Report
On
Development of a Smart Battery Health Monitoring Dashboard

Submitted
In partial fulfillment
For the award of the Degree of

PG-Diploma in Embedded Systems and Design

(PG-DESD)

C-DAC, ACTS (Pune)

Guided by	Name	PRN
Mr. Sripad Deshpande	Deepjyoth Singh Makan	230940130021
	Pranav Dixit	230940130040
	Pranay Patankar	230940130041
	Raut Pranoti Pandharinath	230940130046
	Sakshi Dighade	209401300048

Centre for Development of Advanced Computing(C-DAC), ACTS

(Pune- 411008)

Acknowledgment

This is to acknowledge our indebtedness to our Project Guide, **Mr. Sripad Deshpande C**-DAC ACTS, Pune for her constant guidance and helpful suggestions for preparing this project **Development of a Smart Battery Health Monitoring Dashboard.** We express our deep gratitude towards him for the inspiration, personal involvement, and constructive criticism he provided us with and the technical guidance during this project.

We take this opportunity to thank the Head of the department, **Mr. Gaur Sunder,** for providing us with such a great infrastructure and environment for our overall development.

We sincerely thank **Ms. Namrata Ailawar**, Process Owner, for their kind cooperation and extendible support towards the completion of our project.

We are pleased to express sincere and deep gratitude towards **Mrs. Risha P R (Program Head)** and **Mrs. Srujana Bhamidi** (Course Coordinator, PG-DESD) for their valuable guidance and constant support throughout this work and help to pursue additional studies.

Also, our warm thanks to **C-DAC ACTS Pune**, which provided us with this opportunity to carry out, this prestigious Project and enhance our learning in various technical fields.

Name	PRN
Deepjyoth Singh Makan	230940130021
Pranav Dixit	230940130040
Pranay Patankar	230940130041
Raut Pranoti Pandharinath	230940130046
Sakshi Dighade	209401300048

Abstract

The electric vehicle (EV) revolution hinges on public understanding and acceptance. However, complex data and technical jargon presented by traditional Battery Health Management Systems (BHMS) often confuse and hesitate consumers. This project tackles this challenge by developing an innovative EV BHMS dashboard designed specifically for the public.

Our dashboard bridges this gap by providing clear, understandable, and actionable insights into battery health, empowering consumers to:

- *Demystify EV battery data:* Visualize and comprehend key parameters like state of charge, health, and degradation in an intuitive and engaging format.
- *Gain confidence in EV ownership:* Proactively monitor battery performance, identify potential issues early, and make informed decisions about charging and maintenance.
- *Optimize EV usage:* Understand how driving habits and environmental factors impact battery life, enabling them to maximize range and efficiency.

This project goes beyond existing solutions by:

- *Prioritizing user understanding:* Focuses on clear communication and intuitive design, making complex data accessible to everyone.
- *Broadening EV acceptance:* Empowers consumers with the knowledge and confidence to embrace EV technology, accelerating its mainstream adoption.
- *Promoting sustainable practices:* Encourages informed EV usage and responsible charging habits, contributing to a greener future.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

The rising tide of electric vehicles (EVs) in India holds immense promise for a cleaner future. However, a critical barrier remains complex dashboards drowning in technical jargon. These interfaces leave potential EV owners feeling bewildered and hesitant, creating a chasm between aspiration and action.

Imagine effortlessly understanding your EV's health and performance, replacing confusion with confidence. Our project addresses this challenge by developing a revolutionary EV dashboard designed for the Indian market.

Think beyond puzzling over technical terms. Our dashboard translates complex data into easy-to-understand information you can use. See your battery's health with a user-friendly graphical representation of data.

Our dashboard goes beyond user-friendliness, empowering you to:

- Confidently Manage Your EV: Make informed decisions regarding charging strategies, maintenance schedules, and battery optimization, maximizing your electric journey.
- Embrace Sustainability: Understand how your driving habits and charging choices impact the environment, promoting eco-conscious practices.
- Become an EV Champion: Be part of the green revolution, equipped with the knowledge and confidence to navigate the exciting world of EVs.

This project transcends technology, focusing on bridging the gap between aspiration and reality. We envision a future where every Indian driver can confidently join the EV revolution, empowered by a clear, intuitive dashboard.

Stay tuned as we unveil our innovative solution, designed to pave the way for a more informed, sustainable future of Indian mobility.

1.2 Objective

This project aims to accelerate EV adoption in India by tackling a key challenge: complex dashboards filled with technical jargon. We're developing a user-friendly dashboard that transforms confusing data into clear, actionable insights presented in simple language.

Imagine understanding your battery health with engaging visuals, optimizing range with personalized tips, and making eco-conscious charging choices - all through our intuitive interface.

By empowering users with knowledge and encouraging sustainable practices, we pave the way for a confident and greener future of electric mobility in India.

Chapter 2

Literature Review

[1] S. Gopiya Naik et.al (2022) An electric car will have a system called Battery Monitoring and Control system placed in it. This system will tell about voltage and current and degree of hotness of the battery, as well as look for any indicators of a potential fire. This system is made up of both its hardware and its software. The key elements that constitute the battery monitoring system that is the subject of this discussion are the monitoring device itself as well as the user interface that is provided for the system. The system that is being presented monitors in real-time an indicator of the battery's voltage, current, and remaining charge capacity. Appropriate management actions are triggered as a result of this monitoring, which ensures that the battery is maintained in optimal condition.

[2] A new system for monitoring electric vehicle batteries is presented. It uses sensors, a microcontroller, and an Android app to track voltage, current, temperature, and even fire risk. This low-cost system displays real-time data and can trigger safety measures if needed. It includes data acquisition, an Android interface, and server storage for future analysis. This research paves the way for a practical and affordable battery monitoring solution for electric vehicles

[3] Lithium-ion batteries are driving innovation in various sectors, but accurate measurement of their charge and health (SOC & SOH) is crucial. This article reviews the latest algorithms for estimating SOC and SOH, highlighting their approaches, advantages, limitations, and potential improvements. Understanding these methods is key for optimal battery management, maximizing lifespan, and preventing failures. This analysis aims to guide future research and development in this critical field for technological advancement.

[4] This paper presents an IoT-based battery monitoring system for EVs, addressing limited range and safety issues. It surpasses existing in-vehicle alert

systems by utilizing the internet to offer real-time battery status to both users and manufacturers. This enables proactive maintenance and safety checks, potentially mitigating risks and preventing problems. Technical details of the system design, implementation, and testing are provided, alongside a discussion of future directions for this impactful technology.

[5] This paper explores leveraging the Internet of Things (IoT) to monitor electric vehicle battery performance. Recognizing the vehicle's dependence on battery health and potential for performance degradation, the authors propose an IoT-based monitoring system. This system comprises two key components: a monitoring device collecting battery data and a user interface for data visualization. Initial results demonstrate the system's capability to detect declining battery performance and alert users, enabling proactive maintenance and potentially extending battery lifespan

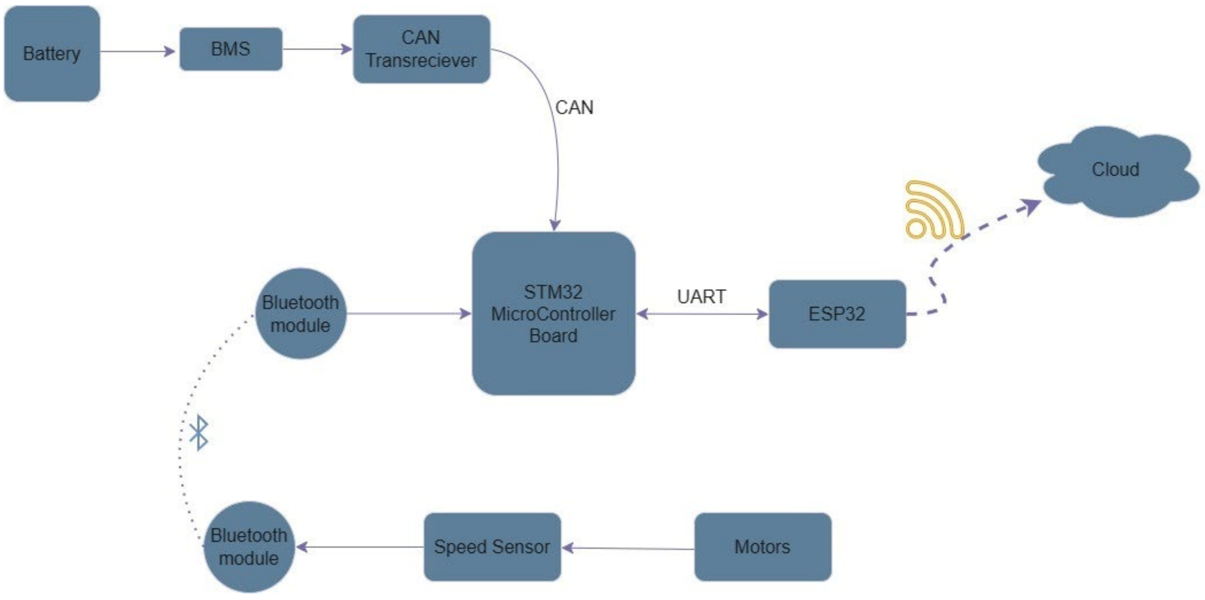
[6] This paper addresses the crucial demand for precise battery state estimation in electric vehicles, focusing on lithium-ion batteries known for their compact size and high energy density. Three main estimation approaches are explored: electrochemical-based methods offer in-depth understanding but are computationally demanding; equivalent circuit model (ECM)-based methods provide faster results but may sacrifice accuracy; and data-driven approaches, relying on machine learning, offer flexibility but depend on data quality. The paper analyses research trends, limitations, and datasets, revealing the strengths and weaknesses of each method, and emphasizing challenges such as non-linear battery behavior and the importance of accurate state-of-charge and state-of-health information. The conclusion outlines future directions for research, aiming to enhance efficiency and reliability in electric vehicle battery state estimation.

Chapter 3

Methodology and Techniques

3.1 Block Diagram

Below is the basic block diagram of the system which will be implemented after the duration of this project.



The goal is to minimize the data transfer length and time between the various modules of an Electric Vehicle (EV) to make the system as real-time as possible and to further better the prediction accuracy to serve the user's needs better.

3.2 Hardware used

1. STM32F407VGT – Is the main board used as a hub where all data comes together and is sent ahead to ESP to be sent to the cloud; although this board can be used at a maximum clock frequency of 168Mhz this is bottlenecked by the board from where we are receiving the data via the CAN bus.
2. CAN Transceivers – The MCP2551 is an ISO-11898 compliant high-speed CAN (Controller Area Network) transceiver designed to interface between a CAN protocol controller and the physical bus. It supports operating speeds up to 1Mb/s.

3. Bluetooth – The use of Bluetooth in this project is to make it more practical and to introduce a wireless connection between the supposed wheel and the main board near the battery thus reducing the cost of production when mass-produced due to no requirement of wires.
4. STM32F103CT86 – This board is used as a gateway to connect and transfer data from the BMS side to the main board (STM32F407VGT). This board was chosen on the basis that it has an adequate amount of processing power while also supporting CAN with an inbuilt CAN controller.
5. Battery – We are using a 12V 3A orange battery for the project to emulate the battery present in an EV.
6. ESP32 – We are using 2 different ESPs one for sending data over Bluetooth to the main board and one to send data after receiving it from the main board via UART to the cloud platform of Things Board.
7. Encoder – We have used LPD3806-400BM-G5-24C as the rotary encoder, an optical rotary encoder with 400 pulses per revolution.

3.2 Software used

1. STM32 CUBE IDE – STM32CubeIDE is an integrated development environment (IDE) from STMicroelectronics tailored for STM32 microcontrollers. Combining the Eclipse-based IDE with the STM32CubeMX configuration and initialization tool, streamlines development by offering code generation, debugging, and profiling features
2. STM32 Cube Programmer – It simplifies the process of loading and updating firmware onto your STM32 devices, ensuring they run smoothly.
3. Arduino IDE 1.8.19 – It is the official integrated development environment for Arduino boards. It provides a user-friendly platform for writing, compiling, and uploading code to Arduino microcontrollers

3.4 Protocols used

1) UART – UART stands for **Universal Asynchronous Receiver/Transmitter**. It's a simple, two-wire serial communication protocol commonly used for low-speed data transmission between devices. Here's a breakdown of its key aspects:

Functionality:

i) UART converts parallel data (multiple bits sent simultaneously) into serial data (bits sent one after another) for transmission and vice versa for reception.

ii) It manages the start and stop bits of each data frame, ensuring clear boundaries between characters.

iii) Some UART implementations offer optional error-checking capabilities.

Hardware:

i) Typically requires just two wires for data transmission (Rx and Tx) and a ground connection.

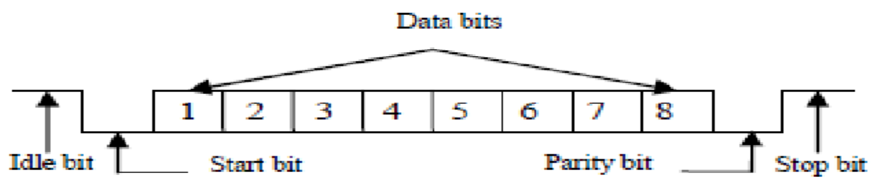
ii) A UART chip or logic within a microcontroller handles the data conversion and protocol management.

Data Transmission:

i) Asynchronous nature means no shared clock signal between devices. Both sides must agree on the bitrate (usually between 600 bps and 115,200 bps) for successful communication.

ii) Each data byte is framed with start and stop bits, typically a single start bit (logic 0) and one or two stop bits (logic 1).

Some configurations include an optional parity bit for error detection.



Applications:

i) UART is widely used in embedded systems for communication with peripherals like sensors, displays, and debug consoles.
ii) Common applications include configuring microcontrollers, interfacing with GPS modules, and serial communication over Bluetooth modules.

Advantages:

i) Simple and low-cost implementation.
ii) Easy to integrate with microcontrollers.
iii) Suitable for low-speed, short-distance communication.

Disadvantages:

i) Limited speed compared to other protocols like I2C or SPI.
ii) No addressing capabilities, making it unsuitable for multi-device networks.
iii) Prone to errors in noisy environments without robust error checking.

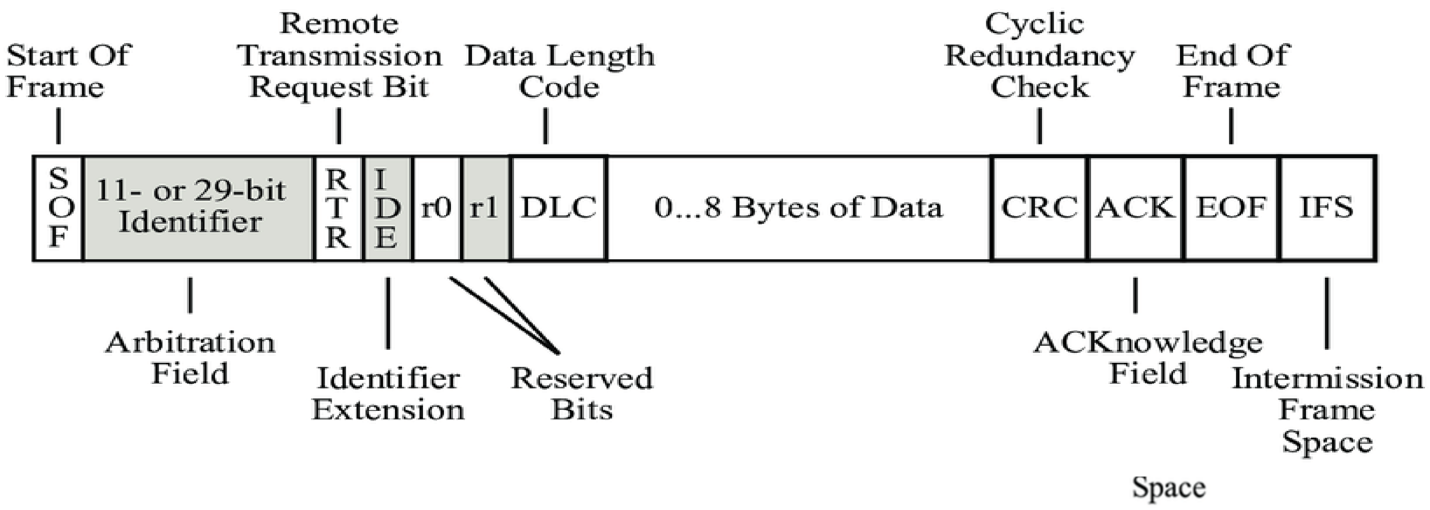
2) CAN –

Bit Timing and Synchronization: Each CAN bit has a specific duration defined by the "Bit Time." All nodes on the bus sample the signal at specific points within this bit time to identify dominant or recessive bits. Oscillator circuits maintain this timing accuracy.

Error Detection and Correction: CAN uses several mechanisms for error detection. **Cyclic Redundancy Check (CRC):** Each message includes a checksum to detect transmission errors. **Bit stuffing:** Ensures transitions in the signal for easier synchronization. **Error frames:** Nodes can transmit error frames if they detect issues. CAN may offer automatic retransmission of messages with errors.

CAN Frame Format: A CAN message consists of:

i) **Arbitration field:** Used for message priority and bus access.
ii) **Control field:** Identifies frame type and data length.
iii) **Data field:** Carries actual information (up to 8 bytes in standard CAN, 64 bytes in CAN FD).
iv) **CRC field:** Ensures data integrity.
v) **Acknowledgement field:** Nodes acknowledge receiving the message successfully.



Higher-level Protocols:

i) **CAN FD (Flexible Data Rate):** Extends standard CAN with higher bitrates (up to 15 Mbps) and larger data payloads (up to 64 bytes).
ii) **J1939:** Used in heavy-duty vehicle applications, extending CAN with addressing capabilities and message filtering.
iii) **Practical Applications:**

CAN Bus in specific industries:

i) **Transportation:** Engine, ABS, airbags, and infotainment systems in cars and aircraft.

ii) **Industrial automation:** Factory control systems, robotic equipment.
iii) **Medical devices:** Infusion pumps, and patient monitoring systems.
iv) **Building Automation:** HVAC systems, lighting control, security systems.

Troubleshooting CAN Bus networks: Common issues include wiring faults, node malfunctions, and software bugs. Tools like CAN analyzers help diagnose and resolve issues.

Developing CAN Bus applications: Tools like microcontrollers, CAN transceivers, and libraries (e.g., Socket, CAN) enable communication with CAN devices.

Comparison with other communication protocols:

*Feat*ure	CAN	RS-232	Ethernet	Wireless
Speed	Up to 1 Mbps (CAN FD: 15 Mbps)	Up to 115 kbps	Up to 1 Gbps	Variable
Cost	Low	Low	Moderate	Moderate
Complexity	Moderate	Low	High	High
Reliability	High	High	Moderate	Moderate
Broadcast	Yes	No	Yes	Variable
Multi-master	Yes	No	Yes	Yes
Suitable for	Real-time control	Simple point-to-point	High-speed data transfer	Flexible networking

Choosing CAN Bus: Consider factors like speed requirements, network size, cost constraints, reliability needs, and real-time control demands.

3) Bluetooth – Bluetooth is a wireless communication protocol widely used for short-range data exchange between devices. Here's a detailed look at its key aspects:

Purpose and History:

i) Designed for low-power, wireless connectivity primarily for data and voice communication over short distances (typically up to 10 meters).
ii) Originally developed by Ericsson in the 1990s and standardized by the Bluetooth Special Interest Group (SIG).

Technical Specifications:

i) Operates in the unlicensed 2.4 GHz Industrial, Scientific, and Medical (ISM) radio band.
ii) Employs frequency-hopping spread spectrum (FHSS) to mitigate interference from other devices in the same band.
iii) Utilizes a master-slave architecture, with one device acting as the master (initiating connections) and others as slaves (responding to the master).
iv) Provides various profiles for specific applications, including audio streaming, file transfer, human interface devices, and more.

Protocol Stack:

i) Bluetooth protocol stack consists of layers defining different functionalities:
ii) **RF (Radio) Layer:** Handles physical radio communication aspects like frequency hopping and modulation.
iii) **Baseband Layer:** Manages packet framing, addressing, and timing within a piconet (network of connected devices).
iv) **Link Manager Protocol (LMP):** Establishes and manages connections between devices.
v) **Logical Link Control and Adaptation Protocol (L2CAP):** Provides reliable data channels for higher-level protocols.

vi) **Higher-level profiles:** Implement specific communication features like Service Discovery Protocol (SDP) for finding services and Generic Attribute Profile (GATT) for accessing data on remote devices.

Connection Process:

i) Devices discover each other by advertising their presence and searching for other Bluetooth devices.
ii) Connection establishment involves authentication and security procedures depending on the desired security level.
iii) Devices can pair for faster future connections after initial pairing.

Data Transmission:

i) Data is transmitted in packets across data channels established within the L2CAP layer.
ii) Different profiles like Serial Port Profile (SPP) or Audio/Video Remote Control Profile (AVRCP) define specific data formats and communication rules for their respective purposes.

Advantages:

i) Low power consumption, ideal for battery-powered devices.
ii) Short-range connectivity suitable for personal area networks.
iii) Standardized and widely supported across various devices.
iv) Secure connections with different security levels available.
v) Diverse profiles cater to various applications.

Disadvantages:

i) Limited range compared to Wi-Fi or cellular networks.
ii) Lower data rates compared to other protocols.
iii) Potential interference from other 2.4 GHz devices.

Comparison of Bluetooth Versions

Feature	Bluetooth 1.0	Bluetooth 2.0	Bluetooth 3.0	Bluetooth 4.0	Bluetooth 5.0	Bluetooth 5.1	Bluetooth 5.2
Year Released	1999	2004	2009	2010	2016	2019	2020
Data Rate (Standard)	721 kbps	2.1 Mbps	24 Mbps (HS mode)	N/A	N/A	N/A	N/A
Data Rate (LE)	N/A	N/A	N/A	1 Mbps	2 Mbps	2 Mbps	2 Mbps
Range (Standard)	10 meters	100 meters	100 meters	N/A	N/A	N/A	N/A
Range (LE)	N/A	N/A	N/A	100 meters	240 meters (Line of Sight)	240 meters (Line of Sight)	400 meters (Line of Sight)
Power Consumption	High	Moderate	Moderate	Low	Low	Low	Low
Key Features	Basic data transfer	EDR, improved security	HS mode, LE	LE only, improved data transfer for LE	Increased data transfer speed and range for LE, Dual Audio	Direction Finding	Enhanced LE connection speeds and data throughput
Typical Applications	Headsets, data transfer	Headsets, data transfer, audio streaming	Headsets, data transfer, audio streaming, keyboards, mice	Wearables, sensors, fitness trackers, smart home devices	Wearables, sensors, fitness trackers, smart home devices, VR/AR, faster streaming	Direction finding, asset tracking	LE Audio, enhanced privacy features

Notes:

- HS mode in Bluetooth 3.0 was only available for specific use cases and is not widely supported.
- N/A indicates the feature is not applicable to the protocol version.

Chapter 4

Implementation

4.1 Battery Health Monitoring System (BHMS)

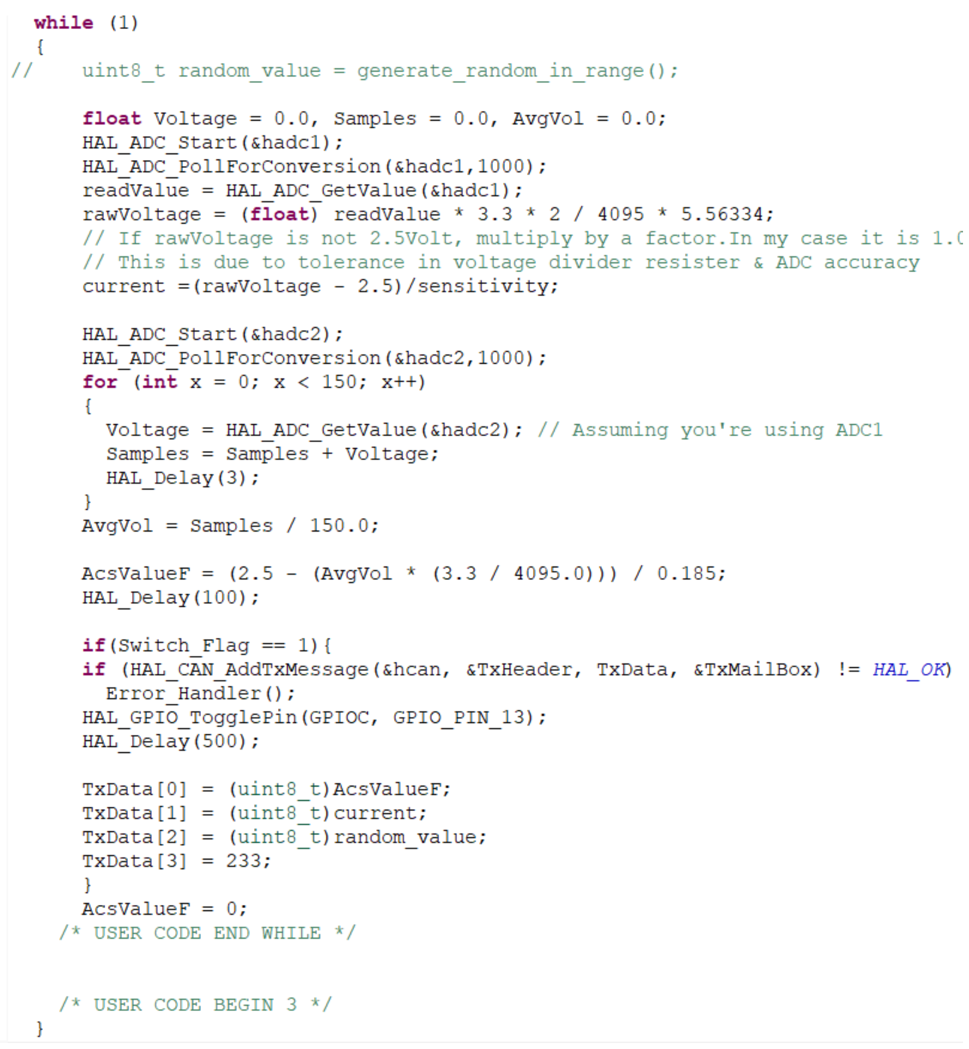
A Battery Health Monitoring System or BHMS is a system that takes into consideration the Voltage, Current, and Temperature and gives insight into the battery health, and predicts the longevity/range of the battery/EV. We have used various sensors like the Voltage sensor and Current sensor (ACS712) and calculated factors like SOC and SOH which give the user actionable insights on when to charge and when to completely swap the battery.

Data from the sensors is received by STM32F103CT86 on the respective ADC pins. This data is then sent to STM32F407VGT using CAN Transceivers.

4.2 CAN Transmission

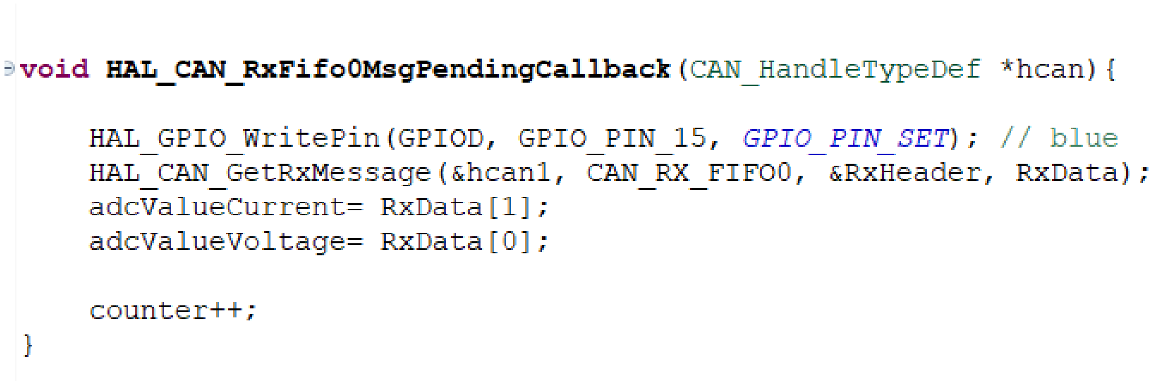
CAN is used for data transmission in an EV working environment because of its robust nature against electrical interference and harsh operating conditions often encountered in vehicles.

We have used CAN to transmit data between STM32F103CT86 and STM32F407VGT, which requires both sides to have the same clock frequencies, this has created a bottleneck of 72MHz clock frequency on the STMF4 board.



```
  while (1)
  {
//    uint8_t random_value = generate_random_in_range();

      float Voltage = 0.0, Samples = 0.0, AvgVol = 0.0;
      HAL_ADC_Start(&hadc1);
      HAL_ADC_PollForConversion(&hadc1,1000);
      readValue = HAL_ADC_GetValue(&hadc1);
      rawVoltage = (float) readValue * 3.3 * 2 / 4095 * 5.56334;
      // If rawVoltage is not 2.5Volt, multiply by a factor.In my case it is 1.(
      // This is due to tolerance in voltage divider resister & ADC accuracy
      current =(rawVoltage - 2.5)/sensitivity;

      HAL_ADC_Start(&hadc2);
      HAL_ADC_PollForConversion(&hadc2,1000);
      for (int x = 0; x < 150; x++)
      {
        Voltage = HAL_ADC_GetValue(&hadc2); // Assuming you're using ADC1
        Samples = Samples + Voltage;
        HAL_Delay(3);
      }
      AvgVol = Samples / 150.0;

      AcsValueF = (2.5 - (AvgVol * (3.3 / 4095.0))) / 0.185;
      HAL_Delay(100);

      if(Switch_Flag == 1){
      if (HAL_CAN_AddTxMessage(&hcan, &TxHeader, TxData, &TxMailBox) != HAL_OK)
        Error_Handler();
      HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
      HAL_Delay(500);

      TxData[0] = (uint8_t)AcsValueF;
      TxData[1] = (uint8_t)current;
      TxData[2] = (uint8_t)random_value;
      TxData[3] = 233;
      }
      AcsValueF = 0;
    /* USER CODE END WHILE */


    /* USER CODE BEGIN 3 */
  }
```

The above code is responsible for getting data from the sensors via the ADCs and then sending the data over CAN to STM32F4 from STM32F1



```
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan){

    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET); // blue
    HAL_CAN_GetRxMessage(&hcan1, CAN_RX_FIFO0, &RxHeader, RxData);
    adcValueCurrent= RxData[1];
    adcValueVoltage= RxData[0];

    counter++;
}
```

The above code is responsible for receiving data from the CAN bus, this is written in the callback as it is only called when the CAN bus generates an interrupt via the mailbox notification.

Thus, we have used the mailbox functionality here to inform the receiver end when it has received data on the can transceiver.

4.3 Bluetooth

Bluetooth is used to both avoid the costs of cabling and to make it less complex with wiring inside the machine.

We have used the HC 05 Bluetooth module on the receiving end to receive data and send it to the STM32F4 board via UART.

```
volatile int pulseCount = 0;
unsigned long lastTime;
unsigned long currentTime;
unsigned long timeInterval = 500; // Time interval in milliseconds

void loop() {
  currentTime = millis();
  if ((currentTime - lastTime) >= timeInterval) {
    float rpm = (pulseCount / 400.0) * 60.0 / (timeInterval / 1000.0); // Assuming 400 counts per revolution
    Serial.print("Speed (RPM): ");
    Serial.println(rpm);
    SerialBT.write(rpm);
    // Send data over Bluetooth
    String rpmString = String(rpm, 2); // Format RPM to 2 decimal places

    pulseCount = 0;
    lastTime = currentTime;
  }
  delay(20);
}
```

The above code is responsible for sending data to HC 05 from the encoder at every 500ms.

```
/* Private variables --------------------
CAN_HandleTypeDef hcan1;
UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

uint8_t rpm;
```

```
while (1)
{

    HAL_UART_Receive(&huart1, &rpm, sizeof(rpm), 10);


    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
```

The above code is responsible for receiving the data in STM32F4

4.4 Sending data to Things Board

To finally make a Dashboard we need to send the data to an IoT platform, we are using MQTT protocol via ESP32, however, we need to first get data from STM32F4 to the ESP.

For this, we are using UART to connect the STM32F4 and ESP32.

Further, we send this data from the ESP32 in JSON format to the cloud Things Board IoT Cloud Platform using MQTT (Message Queuing Telemetry Transport) protocol.

```
void loop() {

   if (!client.connected()) {
        reConnect();
    }
  int receivedData[5];

    if (SerialPort.available() >= 21) { // Ensure enough data is available
    String tempData = SerialPort.readStringUntil('#');
    tempData.trim(); // Remove leading/trailing whitespace

    // Parse data using strtok_r (more robust and reentrant)
    char buffer[tempData.length() + 1];
    tempData.toCharArray(buffer, sizeof(buffer));
    char* next_token = NULL;
    char* token = strtok_r(buffer, ",", &next_token); // Get first token
    int i = 0;

    while (token != NULL && i < 5) {
      receivedData[i] = atoi(token); // Convert token to integer
      token = strtok_r(NULL, ",", &next_token); // Get next token
      i++;
    }
```

Code to parse the data in a meaningful manner for further operation by the ESP after receiving it via UART from STM32F4

```
void sendToTB(const char *parameter, String value) {

    // Construct the JSON payload
    String payload = "{\"" + String(parameter) + "\":" + value + "}";

    // Publish the payload to ThingsBoard
  if (client.publish(tbTopic, payload.c_str())) {
    Serial.println(String(parameter) + " sent to ThingsBoard");
  } else {
    Serial.println("Failed to send data");
  }
}
```

Function to create a JSON file of data to be sent to the cloud and send it using MQTT

Chapter 5

Result

- A successful connection between boards was made using various communication protocols and were all integrated to work in harmony and provide a smooth user experience
- Even though the use of STM32F1 caused a bottleneck limiting all other CPUs connected to it via CAN to 72MHz, the entire functionality was optimized to work without a hitch.
- The dashboard was able to successfully show data giving actionable insights to the user regarding his EV without much of the technical jargon

Chapter 6

Conclusions

- The dashboard created a great tool and has a good chance of increasing the rate at which EVs are adopted by the market.
- By implementing such a minimalist system EVs can be made more affordable thus widening the scope of the targeted audience.

Chapter 7

Future Scope

- Currently due to there being only two ADCs on STM32F103C8T6 only Current and Voltage data are imported but by further multiplexing the ADC channels or by external multiplexing to get more data from the battery and process it for more in-depth analysis of battery health while charging and during EV running
- By implementing it with multiplexed ADCs and getting data about other parameters like Temperature and charge strength combined with running data from the encoder it may be possible to accurately predict the battery left taking into consideration these parameters.

Chapter 7

References

[1] S. Malve, A. Nayak, R. Thape and G. T. Haldankar, "Real-time Battery Monitoring System for Solar Infrastructure," 2020 IEEE First International Conference on Smart Technologies for Power, Energy and Control (STPEC), Nagpur, India, 2020, pp. 1-5, doi: 10.1109/STPEC49749.2020.9297710.

[2] S., Gopiya & Harmain, Chaithra & Sharif, Bhojaraj-Bhoomika. (2022). Battery Parameter Monitoring and Control System for Electric Vehicles. International Journal of Electrical and Electronics Engineering. 9. 1-6.

[3] Dini, Pierpaolo & Colicelli, Antonio & Saponara, Sergio. (2024). Review on Modelling and SOC/SOH Estimation of Batteries for Automotive Applications. Batteries. 10. 34. 10.3390/batteries10010034.

[4] Helmy, Mohd & Abd Wahab, Mohd Helmy & Imanina, Nur & Anuar, Mohamad & Ambar, Radzi & Baharum, Aslina & Shanta, Shanoor & Sulaiman, Mohd & Sanim, Shukor & Hanafi, Hafizul. (2018). IoT-Based Battery Monitoring System for Electric Vehicle. International Journal of Engineering & Technology. 7. 505-510. 10.14419/ijet.v7i4.31.25472.

[5] Helmy, Mohd & Abd Wahab, Mohd Helmy & Imanina, Nur & Anuar, Mohamad & Ambar, Radzi & Baharum, Aslina & Shanta, Shanoor & Sulaiman, Mohd & Sanim, Shukor & Hanafi, Hafizul. (2018). IoT-Based Battery Monitoring System for Electric Vehicle. International Journal of Engineering & Technology. 7. 505-510. 10.14419/ijet.v7i4.31.25472.

[6] Swarnkar, R.; Ramachandran, H.; Ali, S.H.M.; Jabbar, R. A Systematic Literature Review of State of Health and State of Charge Estimation Methods for Batteries Used in Electric Vehicle Applications. World Electr. Veh. J. 2023, 14, 247. https://doi.org/10.3390/wevj14090247.

[7] STMicroelectronics, "User manual Discovery kit with STM32F407VG MCU", Oct 2020.

[8] STMicroelectronics, "STM32F103x8, Datasheet", Sept 2023.

[9] Espressif Systems, "ESP32Series Datasheet", Oct 2016.

[10] Microchip Technology Inc, “MCP2551 High-Speed CAN Transceiver”, Dec 2023.

[11] “Datasheet Bluetooth to Serial Port Module HC05”, Oct 2022

[12] Allegro Microsystems. “ACS712, Hall-Effect-Based Linear current sensor”, Feb 2023.

[13] Fairchild Semiconductor, “LM7806 6V DC / AC, Three Terminal Voltage, Regulator Power”, Apr 1999.