SMART BATTERY HEALTH MONITORING AND MAINTENANCE SYSTEM

MINI PROJECT REPORT

Submitted by

SUBRAMANIAN N 2116230701348

THIRUMURUGAN N 2116230701364

In partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project **"SMART BATTERY HEALTH MONITORING AND MAINTENANCE SYSTEM**" is the bonafide work of **"SUBRAMANIAN N (2116230701348) and THIRUMURUGAN N (2116230701364)"** who carried out the project work under my supervision.

SIGNATURE
Dr.N.Duraimurugan, M.Tech., Ph.D.

Associate Professor,

Computer Science & Engineering

Rajalakshmi Engineering College
(Autonomous)

Thandalam, Chennai -602105.

Submitted for the **ANNA UNIVERSITY** practical examination Mini-Project work viva voice held on_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S.MEGANATHAN, B.E, F.I.E.,** our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work.

We also extend our sincere and hearty thanks to our Internal Guide **Dr.N.Duraimurugan, M.Tech., Ph.D.** Associate Professor, Department of Computer Science and Engineering for his valuable guidance and motivation during the completion of this project. Our sincere thanks to our family members, friends and other staff members of information technology.

SUBRAMANIAN N 2116230701348
THIRUMURUGAN N 2116230701364

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

ABBREVIATION	ACRONYM
IOT	Internet of Things
SOC	State of Charge
SOH	State of Health
RUL	Remaining Useful Life
IDE	Integrated Development Environment
JSON	JavaScript Object Notation

ABSTRACT

As our planet increasingly relies on battery technology—be it in electric cars, renewable energy panels, or simple portable products—maintaining batteries in good health and standing has never been more critical. This project presents a Smart Battery Health Monitoring and Maintenance System that assists users in monitoring and maintaining the performance of their battery in real time. Fundamentally, the system employs an Arduino Uno and voltage (DG-301), current (ACS712), and temperature (DHT11) sensors for measuring crucial battery parameters. The values are sent to a desktop application created using Electron.js and Node.js, where users can simply see metrics such as State of Charge (SoC), State of Health (SoH), and energy consumption through a neat and interactive interface. The uniqueness of this system lies in having intrinsic predictive intelligence. A Long Short-Term Memory (LSTM) network-based machine learning model examines both real-time and past sensor measurements to predict battery degradation and predict Remaining Useful Life (RUL). This allows for prediction of issues and action before the problem. Affordable, user-friendly, and flexible enough to accommodate various types of batteries, the system presents a convenient option for anyone wanting to prolong battery life, decrease maintenance headaches, and enhance safety while promoting more sustainable technology utilization.

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

With the growing dependence on battery-powered devices in sectors such as electric vehicles, renewable energy, and portable electronics, ensuring the reliability and longevity of batteries has become critical. Batteries naturally degrade over time due to usage and environmental factors, often leading to performance issues or unexpected failures. Traditional battery monitoring systems usually lack the ability to provide real-time analysis and predictive insights, making it challenging to anticipate when maintenance or replacement is necessary. This project addresses this gap by introducing a smart battery monitoring and maintenance system that integrates IoT technology, machine learning, and a user-friendly interface. The system, built around an Arduino Uno with voltage, current, and temperature sensors, collects real-time data and transmits it to an Electron.js desktop application. Using machine learning, specifically a Long Short-Term Memory (LSTM) neural network, the system predicts critical battery metrics like State of Health (SoH) and Remaining Useful Life (RUL). This allows users to take proactive maintenance actions, ultimately enhancing battery performance and extending its lifespan.

1.2 SCOPE OF THE WORK

This project focuses on developing an IoT-based system for real-time battery health monitoring and predictive maintenance. It integrates voltage, current, and temperature sensors with an Arduino Uno to collect data. A desktop application built using Electron.js and Node.js visualizes battery metrics and supports user interaction. Machine learning, specifically an LSTM neural network, is employed to predict State of Health (SoH) and Remaining Useful Life (RUL).

The system is designed to be scalable, low-cost, and adaptable to different battery types. It aims to improve battery lifespan, safety, and maintenance efficiency.

1.3 PROBLEM STATEMENT

As battery-powered systems become more prevalent in critical industries, maintaining battery health is essential for ensuring reliability and performance. Current battery monitoring systems often fall short in providing real-time insights or predictive maintenance. Without accurate predictions of a battery's State of Health (SoH) or Remaining Useful Life (RUL), users face the risk of unexpected failures. This project aims to develop a smart, IoT-enabled battery monitoring system that leverages real-time data collection and machine learning for early degradation detection, enabling proactive maintenance and enhancing battery lifespan.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The Smart Battery Monitoring and Maintenance System aims to offer a reliable solution for managing battery health. It integrates real-time data collection from sensors like DG-301, ACS712, and DHT11 to monitor voltage, current, and temperature. This data is sent to an Electron.js desktop application for visualizing important metrics such as State of Charge (SoC) and State of Health (SoH). Additionally, the system uses machine learning, specifically an LSTM neural network, to predict the Remaining Useful Life (RUL) of the battery. This helps users take proactive maintenance actions and optimize battery performance. The system's design allows future enhancements, such as multi-battery support and wireless communication. By improving battery management, the project helps reduce maintenance costs and extend battery life. Ultimately, the system contributes to sustainability in battery-powered applications.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 IOT DEVICES

1. Arduino Uno
2. DG-301 Voltage Sensor
3. ACS712 Current Sensor
4. DHT11 Temperature Sensor

2.2 SYSTEM HARDWARE SPECIFICATIONS

PROCESSOR	Intel i3 11th Gen
MEMORY SIZE	8 GB (Minimum)
HDD	40 GB (Minimum)

2.3 SOFTWARE SPECIFICATIONS

Operating System	Windows 11
Front – End	Electron.js, Chart.js
Back – End	Node.js, TensorFlow.js
IDE	Visual Studio Code, Arduino IDE

CHAPTER 3
SYSTEM DESIGN

3.1 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components

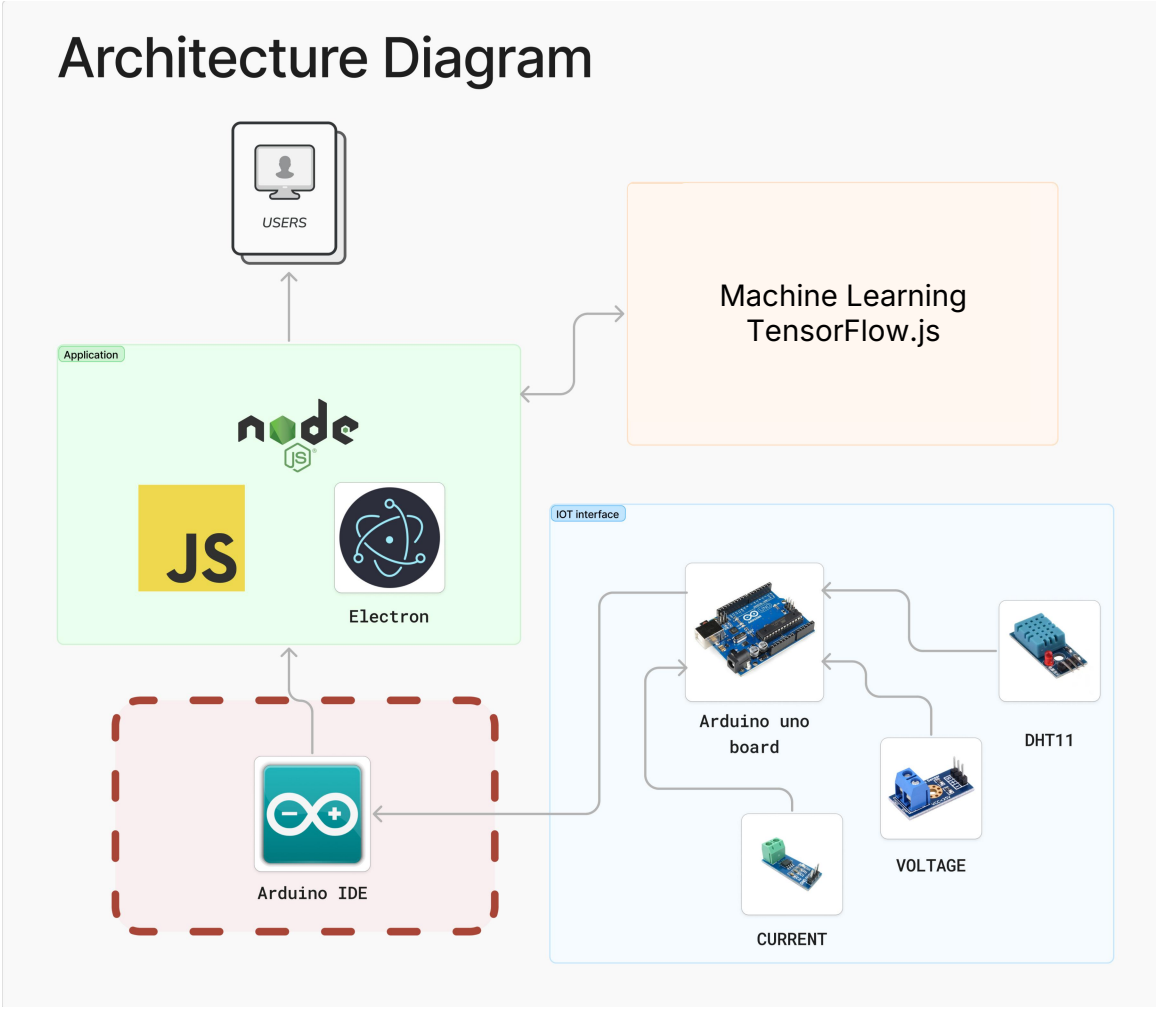

Figure 3.1 Architecture Diagram

From the above Figure 3.1, the architecture of the system is well understood.

3.2 USE CASE DIAGRAM

A use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modelling Language as an actor) and a system to achieve a goal. The actor can be a human or other external system.

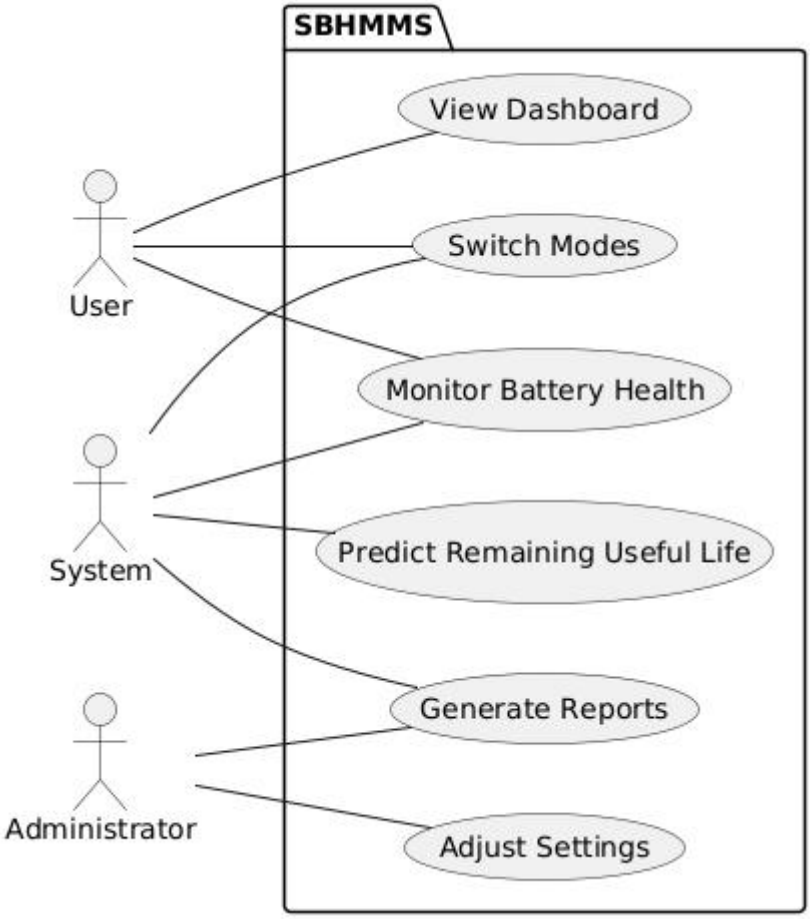

Figure 3.2 Use case diagram

From the above figure 3.2, the interactions between a role in the system is shown

3.3 ACTIVITY DIAGRAM

An activity in Unified Modelling Language (UML) is a major task that must take place in order to fulfill an operation contract. Activities can be represented inactivity diagrams. An activity can represent: The invocation of an operation. A step in a business process.

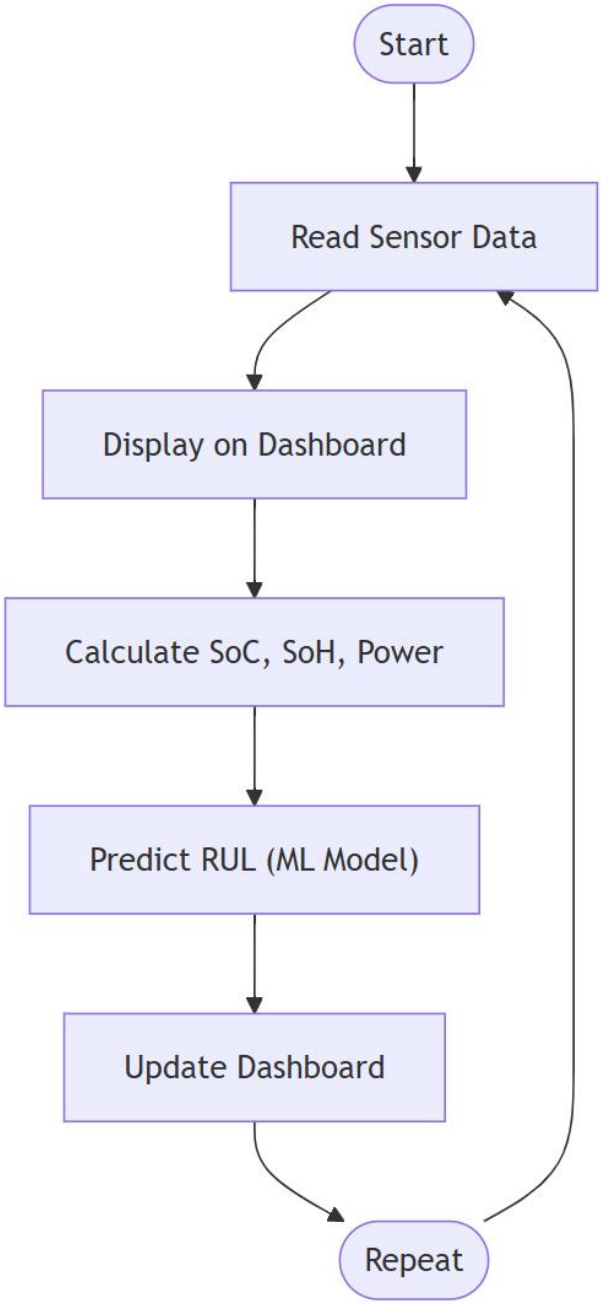

Figure 3.3 Activity Diagram

From the above figure 3.3, the activities of the system are shown

3.4 CLASS DIAGRAM

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modelling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

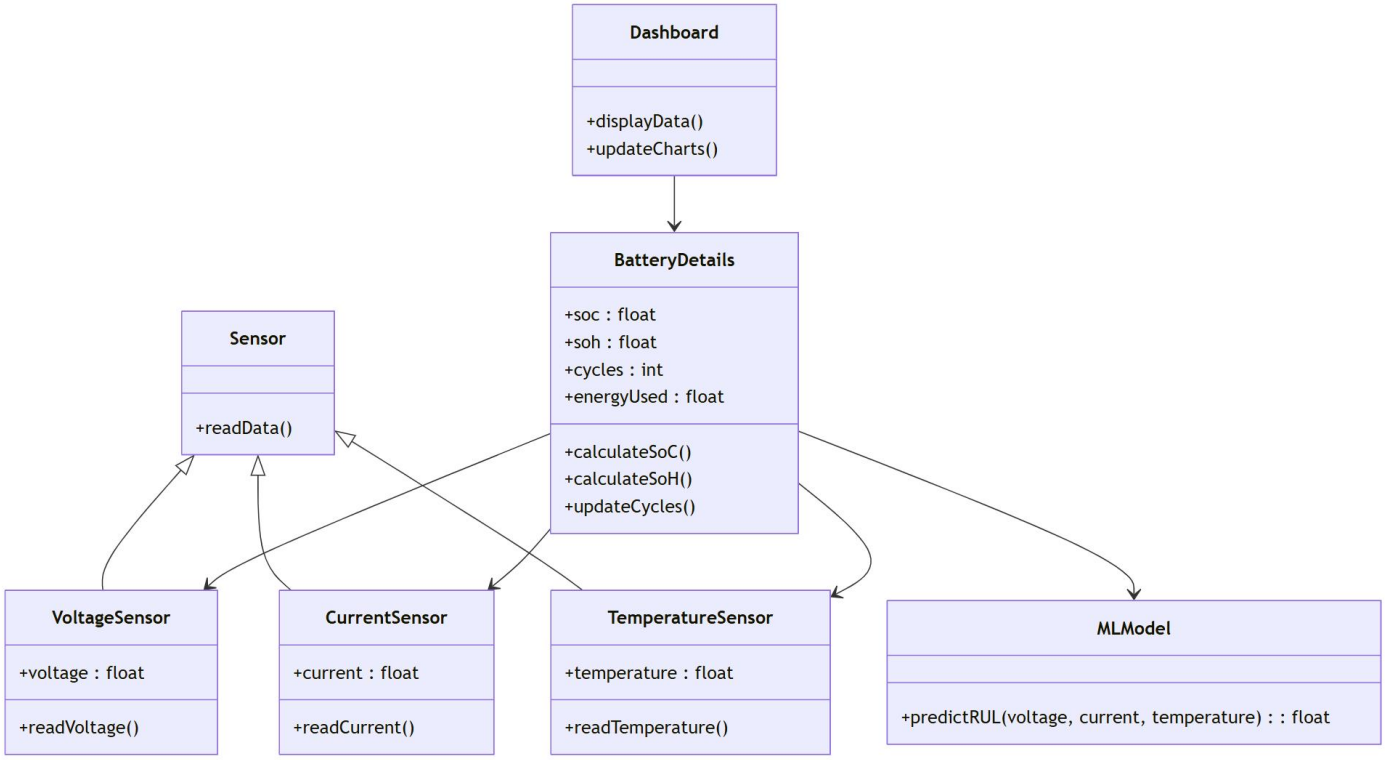

Figure 3.4 Class Diagram

The above Figure 3.4 is the class diagram for the system.

CHAPTER 4

MODULE DESCRIPTION

4.1 HARDWARE MODULE:

This module includes voltage, current, and temperature sensors (DG-301, ACS712, and DHT11 respectively) connected to the Arduino Uno microcontroller. It is responsible for real-time data acquisition from batteries. The sensors continuously monitor battery parameters such as voltage, current, and temperature.

4.2 DATA COLLECTION AND PROCESSING MODULE:

This module interfaces with the hardware to collect sensor data via serial communication. It processes incoming data to compute essential battery parameters like State of Charge (SoC), State of Health (SoH), energy usage, and charge-discharge cycles. The data is cleaned, structured, and prepared for visualization and further analysis.

4.3 MACHINE LEARNING MODULE:

A lightweight ML model implemented using TensorFlow.js predicts metrics like Remaining Useful Life (RUL) and enhances SoH/SoC estimation. The model is trained on synthetic or pre-collected data and integrated into the desktop system for real-time inference.

4.4 DESKTOP APPLICATION MODULE:

Built using Electron.js, this module provides an interactive dashboard for users to monitor battery health visually. It displays real-time sensor data using charts, gauges, and status indicators, and allows users to switch between AA and Li-ion battery types.

4.5 INTEGRATION MODULE:

This module connects all other components — reading serial data, performing computations, running ML inference, and updating the dashboard — to deliver a cohesive and synchronized monitoring experience.

CHAPTER 5

SAMPLE CODING

5.1 Arduino Code

```
#include <DHT.h>
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
const int ACS_PIN = A0;
const int VOLT_PIN = A1;
const float ACS_OFFSET = 2.5;
const float ACS_SENSITIVITY = 0.066;
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  int rawCurrent = analogRead(ACS_PIN);
  float sensorVoltage = rawCurrent * (5.0 / 1023.0);
  float current = (sensorVoltage - ACS_OFFSET) / ACS_SENSITIVITY;
  int rawVoltage = analogRead(VOLT_PIN);
  float batteryVoltage = rawVoltage * (5.0 / 1023.0) * 5.0;
  float temperature = dht.readTemperature();
  if (isnan(temperature)) {
    temperature = 25.0;
  }
  Serial.print("Current: ");
  Serial.print(current, 3);
  Serial.print(" A | Voltage: ");
  Serial.print(batteryVoltage, 3);
  Serial.print(" V | Temp: ");
  Serial.print(temperature, 1);
```

```
  Serial.println(" °C");
  delay(1000);
}
```

5.2 Web Application

5.2.1 main.js

```
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');
const { SerialPort } = require('serialport');
const { ReadlineParser } = require('@serialport/parser-readline');
let win;
let serialPort;
function createWindow() {
  win = new BrowserWindow({
    width: 1280,
    height: 720,
    webPreferences: {
      preload: path.join(__dirname, 'preload.js')
    },
  });
  win.maximize();
  win.setMenuBarVisibility(false);
  win.loadFile('index.html');
}
app.whenReady().then(createWindow);
ipcMain.on('start-serial', (event, portName) => {
  if (serialPort && serialPort.isOpen) {
    serialPort.removeAllListeners();
    serialPort.close(() => {
      console.log('Previous port closed');
      openSerial(portName, event);
    });
  } else {
```

```
    openSerial(portName, event);
  }
});
function openSerial(portName, event) {
  serialPort = new SerialPort({
    path: portName,
    baudRate: 9600,
  });
  const parser = serialPort.pipe(new ReadlineParser({ delimiter: '¥n' }));
  serialPort.on('open', () => {
    console.log('Serial port opened:', portName);
  });
  parser.on('data', (line) => {
    event.sender.send('serial-data',line.trim());
  });
  serialPort.on('error', (err) => {
    console.error('Serial port error:', err.message);
  });
}
ipcMain.handle('list-ports', async () => {
  const ports = await SerialPort.list();
  return ports.map(p => p.path);
});
```

5.2.2 index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Smart Battery Health Monitoring and Maintenance System</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
    <script src="https://cdn.jsdelivr.net/npm/raphael@2.3.0/raphael.min.js"></script>
```

```
  <script src="https://cdn.jsdelivr.net/npm/justgage@1.4.0/justgage.min.js"></script>
  <link rel="stylesheet" href="style.css" />
</head>

<body>
  <div class="mode">
    <div class="toggle-switch">
      <label class="switch-label">
        <input type="checkbox" class="checkbox" onclick="toggleDarkMode()" />
        <span class="slider"></span>
      </label>
    </div>
  </div>

  <div class="container">
    <div class="top">
      <h1>Battery Dashboard</h1>
    </div>

    <div class="dashboard">
      <div class="card">
        <h2>   Current</h2>
        <span id="current">-- A</span>
      </div>
      <div class="card">
        <h2>⚡  Voltage</h2>
        <span id="voltage">-- V</span>
      </div>
      <div class="card">
        <h2>   Temperature</h2>
        <span id="temp">-- °C</span>
      </div>
      <div class="card">
        <h2>  Power</h2>
```

```
    <span id="power">-- W</span>
  </div>
  <div class="card">
    <h2>   Efficiency</h2>
    <span id="efficiency">-- %</span>
  </div>
  <div class="card">
    <h2>⧗ Remaining Useful Life</h2>
    <span id="rul">-- cycles</span>
  </div>
  <div class="card">
    <h2>   Remaining Life</h2>
    <span id="hoursRemaining">-- hours</span>
  </div>
  <div class="card">
    <h2>Battery Status</h2>
    <span id="batteryStatus">--</span>
  </div>
</div>
<div class="meter">
  <div class="sepm">
    <h2>   State of Charge</h2>
    <div id="socGauge"></div>
    <span id="soc">100 %</span>
  </div>
  <div class="sepm">
    <h2>  State of Health</h2>
    <div id="sohGauge"></div>
    <span id="soh">100</span>
  </div>
</div>

<div class="charts">
```

```
      <canvas id="currentChart" height="300px" width="400px"></canvas>
      <canvas id="voltageChart" height="300px" width="400px"></canvas>
      <canvas id="tempChart" height="300px" width="400px"></canvas>
      <canvas id="powerChart" height="300px" width="400px"></canvas>
    </div>
  </div>
</body>
</html>
```

5.2.3 script.js

```
const currentSpan = document.getElementById("current");
const voltageSpan = document.getElementById("voltage");
const tempSpan = document.getElementById("temp");
function toggleDarkMode() {
      document.body.classList.toggle("dark");
    }
    window.serialAPI.listPorts().then((ports) => {
      if (ports.length > 0) {
        window.serialAPI.startSerial(ports[0]);
      }
 });
 const maxDataPoints = 20;
    const createLineChart = (ctx, label, borderColor) => {
      return new Chart(ctx, {
        type: "line",
        data: {
          labels: [],
          datasets: [
            {
              label: label,
              data: [],
              borderColor: borderColor,
              fill: false,
              tension: 0.3,
```

```
      },
    ],
  },
  options: {
    responsive: true,
    animation: false,
    scales: {
      x: { display: false },
      y: { beginAtZero: true },
    },
  },
 });
};
const currentChart = createLineChart(
 document.getElementById("currentChart"),
 "Current (A)",
 "#f44336"
);
const voltageChart = createLineChart(
 document.getElementById("voltageChart"),
 "Voltage (V)",
 "#2196f3"
);
const tempChart = createLineChart(
 document.getElementById("tempChart"),
 "Temperature (°C)",
 "#4caf50"
);
const powerChart = createLineChart(
 document.getElementById("powerChart"),
 "Power (W)",
 "#ff9800"
);
const updateChart = (chart, value) => {
```

```
    const now = new Date().toLocaleTimeString();
    chart.data.labels.push(now);
    chart.data.datasets[0].data.push(parseFloat(value));
    if (chart.data.labels.length > maxDataPoints) {
      chart.data.labels.shift();
      chart.data.datasets[0].data.shift();
    }
    chart.update();
  };
```

5.2.4 TensorFlow.js

```
let model;
async function trainAndSaveModel() {
  const trainingData = [ …];
  const inputs = trainingData.map((d) => [d.current, d.voltage]);
  const labels = trainingData.map((d) => d.rul);
  model = tf.sequential();
  model.add(
    tf.layers.dense({ units: 32, activation: "relu", inputShape: [2] })
  );
  model.add(tf.layers.dense({ units: 16, activation: "relu" }));
  model.add(tf.layers.dense({ units: 1, activation: "linear" }));
   model.compile({
     optimizer: "adam",
     loss: "meanSquaredError",
   });
   const xs = tf.tensor2d(inputs);
   const ys = tf.tensor2d(labels, [labels.length, 1]);
   await model.fit(xs, ys, {
     epochs: 300,
     batchSize: 10,
     callbacks: {
       onEpochEnd: (epoch, logs) => {},
     },
```

```
    });
    await model.save("localstorage://battery-life-model");
    console.log("✅ Model trained and saved to localStorage.");
  }
  // trainAndSaveModel();
  async function loadModel() {
    try {
      model = await tf.loadLayersModel("localstorage://battery-life-model");
      console.log("Model loaded from localStorage.");
    } catch (error) {
      console.log("No saved model found");
    }
  }
loadModel();
```

CHAPTER 6

SCREEN SHOTS

1. Dashboard Page

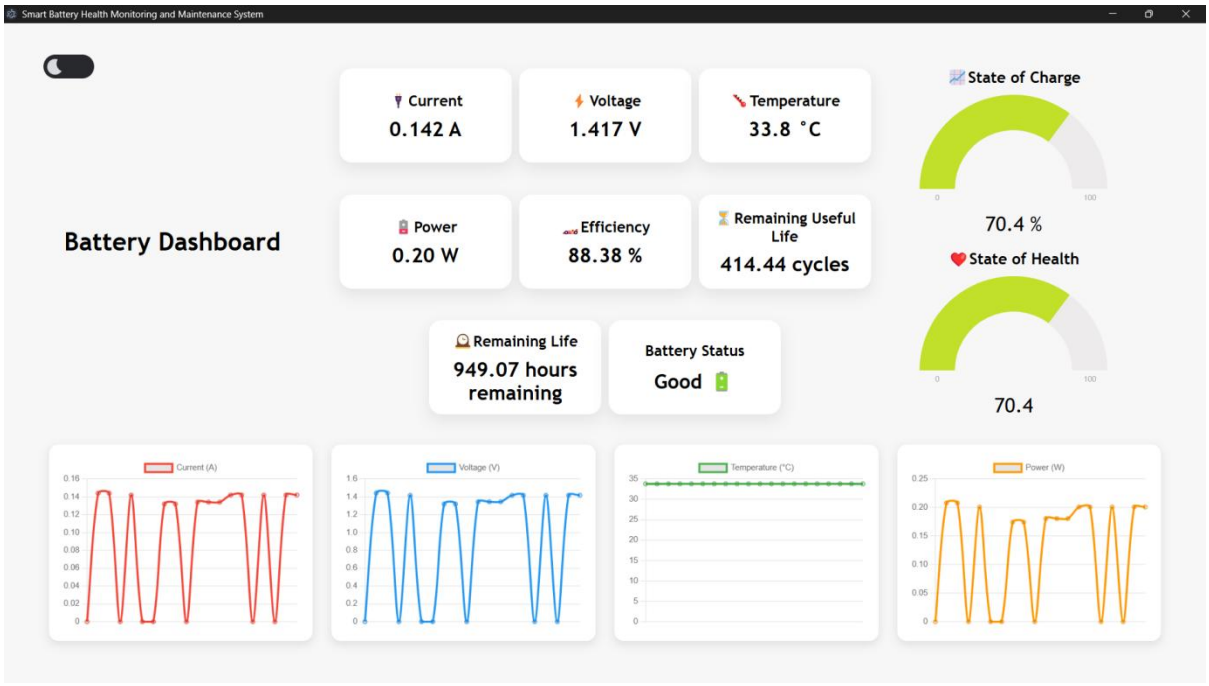

Figure 6.1 Responsive Dashboard

2. Data Sent Arduino via Serial Port

```
Current: 0.152 A | Voltage: 1.515 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.152 A | Voltage: 1.515 V | Temp: 31.8 °C
Current: 0.152 A | Voltage: 1.515 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
Current: 0.149 A | Voltage: 1.491 V | Temp: 31.8 °C
```

Figure 6.2 Data Sent Arduino via Serial Port

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

The Smart Battery Monitoring and Maintenance System successfully demonstrates a real-time solution for tracking critical battery parameters such as voltage, current, and temperature. Through seamless integration of sensors, microcontroller, and a desktop dashboard interface, the system enables accurate monitoring, data visualization, and intelligent battery health estimation. The inclusion of State of Charge (SoC), State of Health (SoH), and Remaining Useful Life (RUL) predictions enhances battery reliability and reduces the risk of unexpected failures. This project highlights the potential of IoT and machine learning in ensuring safe and efficient battery usage across various applications.

In the future, the Smart Battery Monitoring and Maintenance System can be enhanced by integrating cloud-based storage and access to enable remote monitoring, historical data analysis, and improved data security. A dedicated mobile application could be developed to provide real-time updates and alerts to users on the go. The system can also be scaled to support monitoring of multiple batteries simultaneously, making it suitable for larger industrial or commercial battery banks. Furthermore, implementing advanced machine learning models trained on extensive datasets can improve the accuracy of predicting State of Health (SoH) and Remaining Useful Life (RUL). Additional features such as automatic battery type detection, fault prediction alerts, and integration with renewable energy systems could significantly broaden the system’s usability and impact.

REFERENCES

1. A. Haraz, K. Abualsaud, and A. Massoud, "State-of-Health and State-of-Charge Estimation in Electric Vehicles Batteries: A Survey on Machine Learning Approaches," IEEE Access, 2024. https://doi.org/10.1109/ACCESS.2024.3486989
2. R. Ranjith Kumar, C. Bharatiraja, K. Udhayakumar, S. Devakirubakaran, K. Sathiya Sekar, and Lucian Mihet-Popa, "Advances in Batteries, Battery Modeling, Battery Management System, Battery Thermal Management, SOC, SOH, and Charge/Discharge Characteristics in EV Applications," IEEE Transactions on Industrial Electronics, 2024.
3. G. Krishna et al., "Advanced battery management system enhancement using IoT and ML for predicting remaining useful life in Li-ion batteries," Scientific Reports, vol. 14, 30394, 2024. https://doi.org/10.1038/s41598-024-80719-1
4. Y. Zheng et al., "Thermal state monitoring of lithium-ion batteries: Progress, challenges, and opportunities," Progress in Energy and Combustion Science, vol. 100, 101120, 2023. https://doi.org/10.1016/j.pecs.2023.101120