EVE&D - Battery Management - Mini Projects using Python with AI

SUDARSHANA KARKALA | EV.ENGINEER | +91 9845561518 | CARSOFTWARESYSTEMS.COM | EVE&D CODE IITM

AI-Powered EV Battery Fire Prevention System

Battery Temperature Monitoring System

Goal: Read temperature data, analyze trends, and detect overheating.
Concepts: File handling, NumPy, Pandas, Matplotlib

Tasks:

- Read a **CSV file** containing **battery temperature data**
- Calculate **average, max, and min temperatures**
- **Plot a temperature trend graph** using Matplotlib
- **Detect overheating conditions** (e.g., alert if temp > 60°C)

Outcome: Basic battery monitoring using Python

Battery Voltage & Current Analysis

Goal: Analyze voltage & current data to detect anomalies.
Concepts: Pandas, Data Visualization, Time-Series Analysis

Tasks:

- Load **battery voltage & current datasets**
- Identify **voltage drops and current spikes**
- Plot **Voltage vs. Time** & **Current vs. Time**
- Set a rule: Alert if voltage drops **below a threshold**

Outcome: Detect battery performance issues

State of Charge (SOC) Estimation

Goal: Estimate battery **SOC** using voltage and current data.
Concepts: Numerical computing, Basic Machine Learning

Tasks:

- Load **historical battery data** (Voltage, Current, SOC)
- Train a **simple regression model** to predict SOC
- Validate results using test data
- Display **real-time SOC values** for a given input

Outcome: SOC estimation using Python

EV Battery Health Prediction (AI Mini Project)

Goal: Use AI to predict battery degradation over time.
Concepts: Machine Learning, Data Science

Tasks:

- Load **battery charge-discharge cycle data**
- Identify **patterns in battery degradation**
- Train an ML model (Scikit-learn) to predict **Remaining Useful Life (RUL)**
- Visualize predictions with **graphs**

Outcome: AI-based battery health prediction

Real-Time Battery Monitoring with IoT (Advanced)

Goal: Collect real-time battery data using IoT (optional).
Concepts: Python, MQTT, IoT Sensor Integration

Tasks:

- Connect a **temperature sensor (DHT11) or voltage sensor**
- Use **Raspberry Pi / ESP8266** to collect data
- Send data via **MQTT or HTTP** to Python
- Analyze and **store data in a database**

Outcome: Real-time battery health monitoring

Intrusion Detection in Battery Management System (BMS)

Goal: Detect **anomalous activities** (hacking attempts, data tampering, or unauthorised access) in an **EV Battery Management System (BMS)** using Python.

Concepts Used:

- Log Analysis & Data Forensics
- Anomaly Detection (Machine Learning)

- Cybersecurity Threat Detection

Project Overview

The **Battery Management System (BMS)** logs critical parameters:

- Voltage, Current, Temperature
- State of Charge (SOC), State of Health (SOH)
- Communication logs (CAN messages)

Potential Cyber Threats:

- **Spoofing Attack:** Fake voltage readings injected
- **Man-in-the-Middle Attack:** SOC data modified
- **Malware in BMS:** Unauthorised data manipulation

Expected Outcomes

Build a **Battery Intrusion Detection System (IDS)**
Detect **cyber attacks** on BMS data
Train an **ML model** to differentiate between normal and attack conditions
Secure BMS communication with encryption (Advanced)

Steps to Implement the Project

Collect Battery Data Logs (or Use Sample Data)

- Use a **CSV file** containing **battery logs** with timestamps
- Add a **column for intrusion detection labels** (Normal / Attack)

Analyse Normal vs. Anomalous Data

- Load the dataset using **Pandas**
- Visualize **voltage/current variations** using **Matplotlib**
- Identify **unexpected spikes, drops, or inconsistent SOC values**

Implement an Anomaly Detection Model

- Use **Scikit-Learn** to train an ML model for intrusion detection
- Algorithms: **Isolation Forest, Random Forest, or Logistic Regression**
- Train model on **normal vs. attack data samples**
- Detect **real-time anomalies** from live battery logs

Real-Time Intrusion Detection Simulation

- Simulate incoming battery data **(live stream using Python)**
- Detect **unauthorised activities** and trigger **alerts**
- Implement **logging system** to save security breach attempts

Secure Battery Data with Encryption (Advanced)

- Use **AES Encryption** (Python `pycryptodome` module)
- Encrypt **critical BMS data** before transmission
- Ensure **only authorised systems can decrypt it**