International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Eco-Intelligent EV Dashboard with AI Insights: A Machine Learning Approach for Real-Time EV Battery Monitoring and Predictive Maintenance

Mrs. Pavithra V, M.E., (co-author) Assistant Professor, Department of Electronics and Communication Engineering, Renganayagi Varatharaj College of Engineering, Sivakasi,

K. Mahesh , T. Arunkumar, S. Jayasundar

Department of Electronics and Communication Engineering

Renganayagi Varatharaj College of Engineering, Sivakasi – 626 128

Anna University, Chennai – 600 025, Tamil Nadu, India

Abstract - The rapid adoption of electric vehicles (EVs) has brought forward a serious challenge that not many people talk about openly — range anxiety and unexpected battery failures. Most affordable EV systems today show you only the basics: battery percentage and current speed. There is very little intelligence behind what you see on the dashboard. This paper presents the Eco-Intelligent EV Dashboard with AI Insights, a system we built from scratch that combines embedded hardware, machine learning models, and a real-time web interface to make EV monitoring smarter and actually useful.

The hardware side uses three Arduino-based nodes — a Battery Management System (BMS) node, a Driving node, and a Charging node — each collecting different vehicle parameters and communicating over a Controller Area Network (CAN) bus to a Raspberry Pi 4 central unit. On the software side, Random Forest Regressor and Classifier models are deployed to predict State of Charge (SoC), State of Health (SoH), remaining driving range, and abnormal discharge events. The system is accessible both through a QML-based on-device dashboard and a Streamlit web application. Experimental results show that the SoC model achieves an R^2 score of 94.2%, range estimation reaches 91.6%, battery health (SoH) reaches 92.8%, and abnormal discharge detection achieves 95.1% classification accuracy.

The system is designed to be low-cost and scalable, making it a realistic option for everyday EV users and fleet operators alike.

Keywords: Electric Vehicle, Battery Management System, Machine Learning, Random Forest, State of Charge, State of Health, Predictive Maintenance, CAN Bus, Raspberry Pi, Streamlit.

I. INTRODUCTION

Electric vehicles are no longer a futuristic concept — they are on roads today, and in growing numbers. But for all the progress made in battery chemistry, motor efficiency, and vehicle design, one thing has largely been left behind: the intelligence of the dashboard. Most EV users today still rely on a basic percentage bar and a rough range estimate, both of which can be wildly inaccurate under changing conditions like load, temperature, or driving style.

This gap is not just inconvenient. It creates real anxiety for drivers who are unsure whether they can complete a trip, it leads to poor charging habits that quietly degrade the battery, and it delays maintenance decisions until a failure already happens. The problem is that raw sensor data — voltage, current, temperature — is being collected but not really used. The data sits there, and nothing intelligent is done with it.

What we wanted to build was a system that actually uses that data. Something that watches what is happening in real time, learns from the patterns, and tells the driver things like: "Your battery health is dropping faster than normal," or "At your current driving speed and load, you will not make it to your destination — slow down." These are insights that matter.

The Eco-Intelligent EV Dashboard is the result of that goal. It is a hardware-software system built on a Raspberry Pi 4 and multiple Arduino microcontrollers, connected over a CAN bus, running machine learning models trained on a real-world EV IoT dataset. The outputs are presented through both a local QML dashboard and a remote Streamlit web application, giving users two ways to interact with the system.

This paper is structured as follows. Section II reviews the existing literature. Section III describes the proposed system architecture and design. Section IV details the dataset and preprocessing. Section V explains the machine learning model

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

development. Section VI covers hardware implementation. Section VII presents results and discussions. Section VIII covers applications and future scope, and Section IX concludes the paper.

II. LITERATURE REVIEW

Research into intelligent EV monitoring has been growing steadily over the past decade, driven by the increasing complexity of battery systems and the need for proactive fault detection. The work reviewed here covers three broad themes: battery state estimation, predictive maintenance, and IoT-based monitoring architectures.

Saxena et al. [1] provided a thorough survey of battery management system architectures, covering both hardware-level sensing and software-level estimation algorithms such as Coulomb counting and Kalman filtering. Their work highlights that while these methods work well under stable conditions, they struggle significantly when operating parameters change dynamically — a situation common in real-world driving.

Zhang et al. [2] explored machine learning approaches specifically for EV battery predictive maintenance, demonstrating that data-driven models can detect degradation trends well before they become critical. However, their system operated entirely in software simulation and lacked integration with real hardware, which limits its practical applicability.

Konkimalla [3] proposed a machine learning framework for State of Charge estimation using real-time sensor data, showing improved accuracy over conventional methods. Importantly, the study showed that personalized models — those trained on data from similar driving profiles — outperform generalized models significantly.

Hu et al. [4] presented a comparative study of equivalent circuit models for lithium-ion batteries, emphasizing the impact of charge cycles and temperature on battery lifespan. Their analysis forms the theoretical basis for the degradation features we use in our State of Health prediction model.

Kumar et al. [5] built a hybrid machine learning framework combining random forest with anomaly detection for lithium-ion batteries, achieving strong performance on fault classification tasks. Their limitation, similar to others, was the absence of a real-time user interface for non-technical users.

The gap across all reviewed works is consistent: most systems either focus on monitoring without intelligence, or on intelligence without a real hardware-software integration. Our work addresses this gap by combining distributed hardware sensing, machine learning inference, cloud-based model hosting, and an interactive dual-mode dashboard into a single unified platform.

III. PROPOSED SYSTEM ARCHITECTURE

A. System Overview

The proposed system is structured as a distributed embedded architecture that separates sensing, processing, and visualization into distinct functional layers. At the bottom is the sensor layer, where three Arduino-based nodes collect raw physical data from the vehicle. In the middle is the processing layer, where a Raspberry Pi 4 aggregates this data, applies preprocessing, and runs inference through trained machine learning models. At the top is the interface layer, which presents results to the user through two channels: a local QML dashboard rendered on an attached monitor, and a cloud-hosted Streamlit web application accessible via browser.

The system supports two modes of operation. In Hardware Live Mode, real data from the Arduino nodes flows through the CAN bus to the Raspberry Pi, gets forwarded to a Flask API server, and is fetched by the Streamlit interface for real-time predictions. In Software Simulation Mode, a user can bypass the hardware entirely and enter parameters manually through sliders and input fields on the dashboard — useful for testing, demonstrations, and scenarios where hardware is not available.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

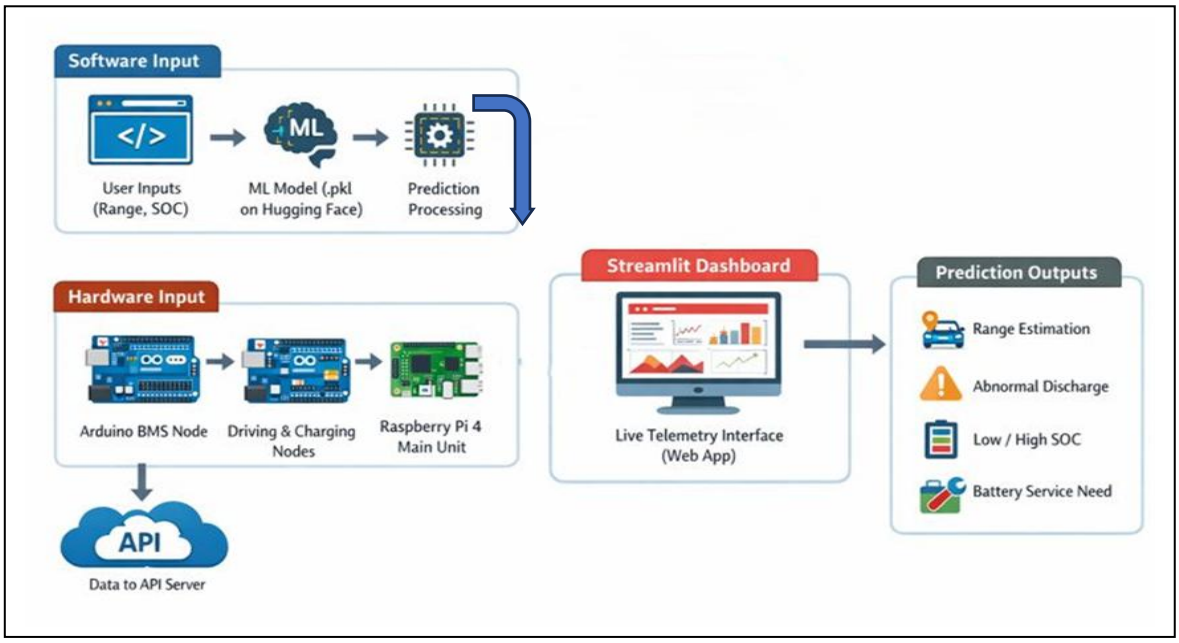
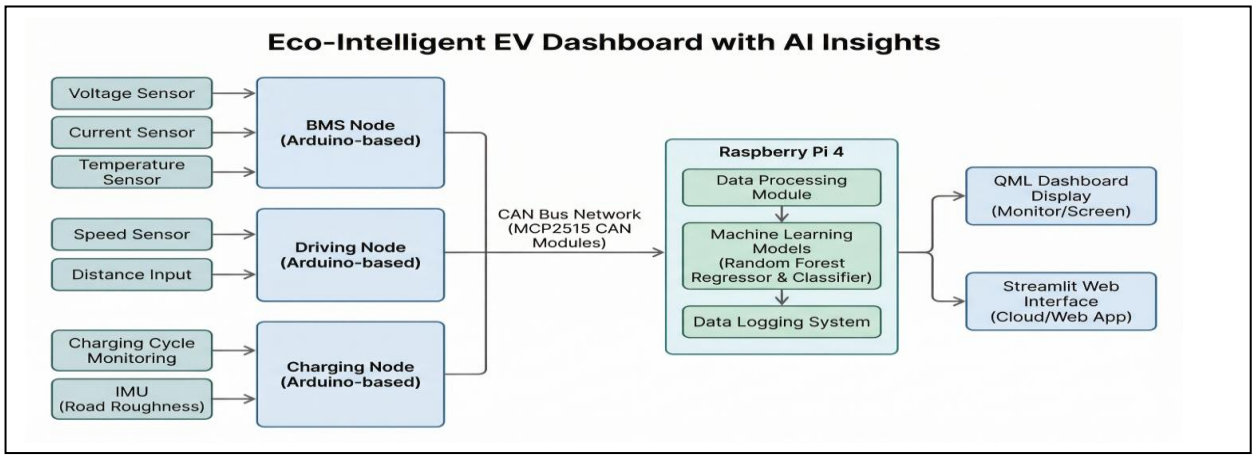


Fig. 1: System Architecture of the Eco-Intelligent EV Dashboard

B. Block Diagram

gure 2 shows the complete block-level structure of the system. Voltage, current, and temperature sensors feed into the BMS Node (Arduino). A speed sensor and distance input feed into the Driving Node (Arduino). Charging cycle monitoring and an IMU for road roughness feed into the Charging Node (Arduino). All three nodes communicate over a CAN Bus Network using MCP2515 CAN modules to the Raspberry Pi 4. The Raspberry Pi runs a Data Processing Module, followed by Machine Learning Models (Random Forest Regressor and Classifier), and a Data Logging System. Output goes to either the QML Dashboard Display or the Streamlit Web Interface.

Fig. 2: Block Diagram of the Eco-Intelligent EV Dashboard with AI Insights



C. Hardware Nodes

1) Battery Management System (BMS) Node

The BMS node is built around an Arduino Uno and is responsible for continuous monitoring of the three most critical battery parameters: voltage, current, and temperature. Battery voltage is measured using a resistor divider circuit that safely scales the high pack voltage down to the ADC range of the Arduino. Current — both charging and discharging — is measured using an

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

INA219 or ACS712 sensor capable of bidirectional measurement. Temperature is tracked with a DS18B20 digital temperature sensor or a thermistor for thermal safety monitoring.

2) Driving Node

The Driving node captures vehicle motion parameters. A Hall effect sensor detects wheel rotation pulses, from which the Arduino calculates instantaneous speed and cumulative distance. These values are critical for dynamic range estimation and driving behavior analysis.

3) Charging Node

The Charging node monitors the charging process — tracking charging voltage, current, and the number of completed charge cycles. An IMU module can optionally be added to capture road roughness data, which influences energy consumption and battery stress. The node detects inefficient charging patterns and logs cycle counts for battery health analysis.

D. CAN Bus Communication

All three Arduino nodes communicate with the Raspberry Pi over a Controller Area Network (CAN) using MCP2515 CAN controllers and TJA1050 transceivers. CAN was specifically chosen because of its origins in automotive environments — it is designed to handle multiple nodes on a shared bus, prioritize messages through arbitration, and remain reliable even in electrically noisy conditions. Each node transmits structured message frames containing its sensor data at regular intervals. The Raspberry Pi receives these frames through a python-can compatible CAN interface.

E. API-Based Integration

Once the Raspberry Pi aggregates sensor data from all nodes, it transmits the combined dataset to a cloud-hosted Flask API using HTTP POST requests in JSON format. The API stores the most recent data snapshot in a global dictionary with a timestamp. The Streamlit application then retrieves this data via a GET request to the /latest endpoint whenever it needs to perform a prediction cycle. This decoupling of hardware and software means the two can operate independently — the hardware continuously pushes data regardless of whether someone is currently looking at the dashboard.

IV. DATASET AND PREPROCESSING

A. Dataset Description

The dataset used for training all machine learning models is the EV IoT Predictive Maintenance Dataset, publicly available on Kaggle [6]. It is structured in a time-series format where each record represents a complete snapshot of vehicle state at a given timestamp. The dataset covers a wide range of parameters including battery voltage, current, temperature, State of Charge (SoC), State of Health (SoH), charge cycles, motor temperature and torque, driving speed, load weight, ambient temperature, road roughness, tire condition, brake metrics, and Remaining Useful Life (RUL). This richness of features makes it suitable for multi-task prediction across different aspects of EV performance.

B. Preprocessing Pipeline

Raw data from the dataset required several preprocessing steps before it could be used for model training. First, the dataset was inspected for missing values and duplicate records. Missing entries were handled using appropriate imputation strategies. Second, irrelevant features that did not contribute to specific prediction tasks were excluded to reduce noise.

A notable preprocessing step involved the battery current column, which contains both positive and negative values representing charging and discharging respectively. For the abnormal discharge detection model, the direction of current flow is less important than its magnitude. A derived feature called Current Magnitude was created by taking the absolute value of the current column — this allowed the model to focus on the intensity of current flow rather than its direction.

Feature selection was performed separately for each of the four prediction models to ensure task-specific relevance. The selected feature sets are as follows:

Prediction Task	Input Features Used
SoC Prediction	Battery Voltage, Battery Current, Battery Temperature
Range Estimation	SoC, Load Weight, Ambient Temperature

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Prediction Task	Input Features Used
Abnormal Discharge Detection	Current Magnitude, Driving Speed, Motor Torque, Battery Temperature
Battery Health (SoH) Prediction	Charge Cycles, Battery Temperature, Component Health Score

Table I: Feature Selection for Each Prediction Model

Outlier detection was performed to remove extreme values outside realistic operational ranges. Finally, an 80/20 train-test split was applied to all datasets to allow for proper model evaluation on unseen data.

V. MACHINE LEARNING MODEL DEVELOPMENT

A. Algorithm Selection

The Random Forest algorithm was selected as the primary machine learning method for this project. It was chosen for several practical reasons: it handles non-linear relationships well without requiring extensive hyperparameter tuning, it is robust against overfitting even with relatively small datasets, it provides reasonable feature importance information, and it is computationally efficient enough to run on a Raspberry Pi 4. For continuous output tasks (SoC, range, SoH), the Random Forest Regressor was used. For binary classification (normal vs. abnormal discharge), the Random Forest Classifier was used.

B. Model Training and Evaluation

Six models were trained in total using Python's scikit-learn library: SoC prediction, battery health (SoH), low battery detection, range estimation, abnormal discharge detection, and speed recommendation. Each model was trained independently on its designated feature set. Training was performed on 80% of the dataset, with the remaining 20% held out for testing. The R^2 score was used as the primary metric for regression models, and accuracy score for classification models.

All trained models were serialized to .pkl format using the joblib library for storage and reuse without retraining. Due to the combined size of the six model files exceeding 600 MB, they were hosted on Hugging Face's model repository platform. The Streamlit application dynamically downloads the required models at runtime, enabling a lightweight deployment footprint.

C. Model Performance Results

Model	Type	Metric	Score
State of Charge (SoC)	Regression	R^2 Score	94.2%
Range Estimation	Regression	R^2 Score	91.6%
Battery Health (SoH)	Regression	R^2 Score	92.8%
Abnormal Discharge Detection	Classification	Accuracy	95.1%

Table II: Machine Learning Model Performance Summary

The abnormal discharge detection model achieved the highest performance because the classification boundary between normal and abnormal discharge conditions is relatively well-defined in the data. The SoC model performs well because voltage-based patterns provide stable, predictable features for battery charge estimation. Range estimation shows slightly lower accuracy because it depends on multiple dynamic factors — load, ambient temperature, driving speed — that interact in complex ways.

VI. HARDWARE IMPLEMENTATION

A. Hardware Components

The hardware requirements for this system were kept deliberately low-cost to ensure the solution is realistic for ordinary EV owners and small fleet operators. The central processing unit is a Raspberry Pi 4 with 4 GB of RAM, running Raspberry Pi OS. The three sensor nodes are each built on Arduino Uno boards. Communication between nodes uses the MCP2515 CAN controller

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

module paired with the TJA1050 CAN transceiver. Sensors include the INA219 for current measurement, DS18B20 for temperature, and a Hall effect sensor for speed. Power is supplied through a DC-DC buck converter (LM2596) that steps down the battery voltage to 5V/3.3V for the electronics.

Component	Specification / Role
Raspberry Pi 4	4 GB RAM — Central processing, ML inference, dashboard
Arduino Uno (×3)	BMS Node, Driving Node, Charging Node
INA219 / ACS712	Bidirectional current sensing
DS18B20 / Thermistor	Battery and ambient temperature sensing
Hall Effect Sensor	Speed and distance measurement
MCP2515 + TJA1050	CAN Bus controller and transceiver
LM2596 Buck Converter	5V / 3.3V regulated power supply
SD Card (32 GB)	Raspberry Pi OS and data logging storage

Table III: Hardware Components List

B. Hardware Output

The physical prototype consists of three labeled PCB panels — BMS Node, Driving Node, and Charging Node — interconnected via CAN bus wiring. The Driving Node panel includes physical controls for gear selection (P-N-D-R), an ignition button, brake and throttle potentiometers, and a motor encoder. The BMS Node panel houses the battery pack along with the voltage sensor, current sensor, temperature sensor, and CAN module. The complete assembly is powered from a prototype battery pack and communicates with the Raspberry Pi through the CAN interface.

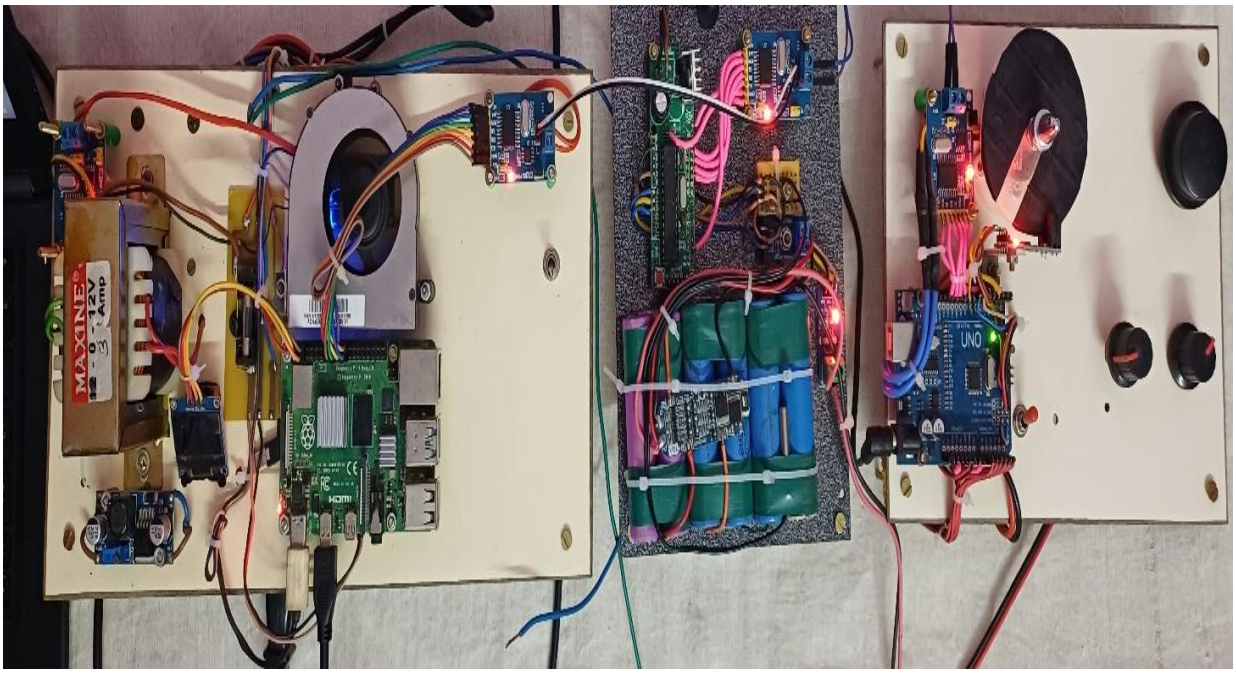


Fig. 3: Physical Hardware Setup — Full System Assembly

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

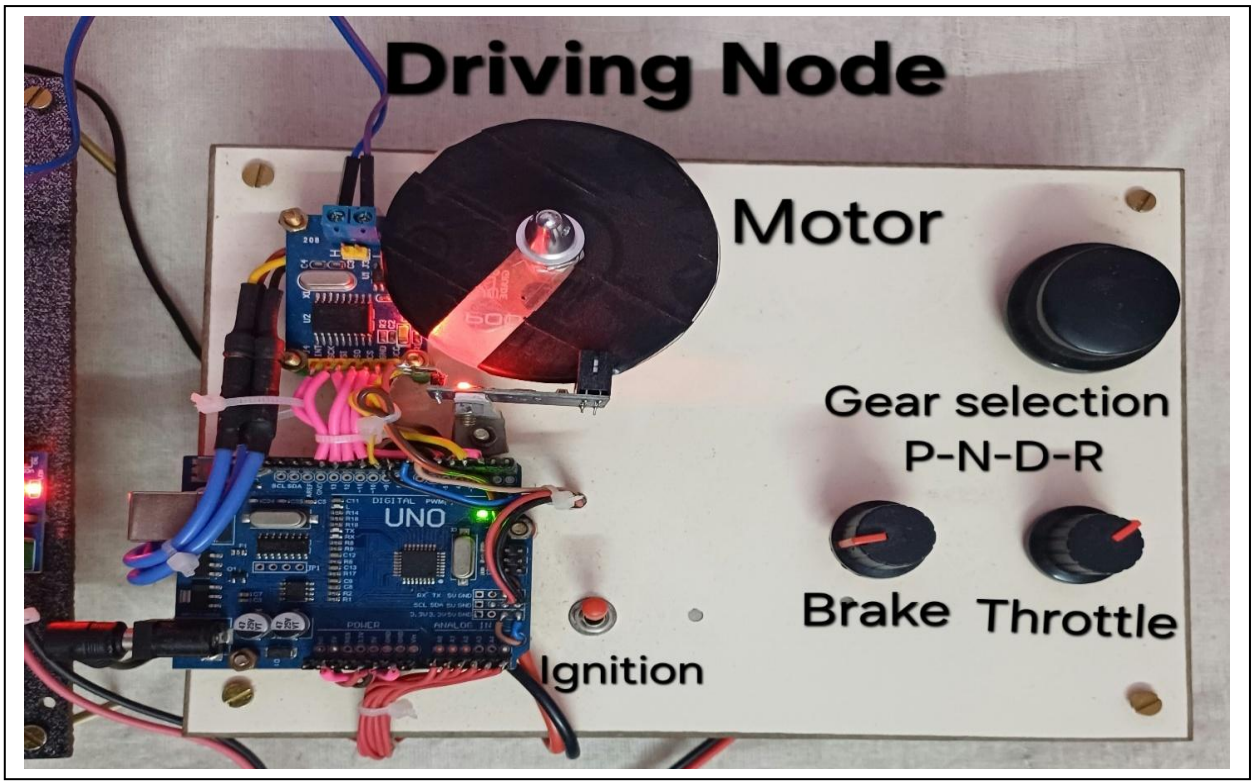


Fig. 4: Driving Node Panel with Motor, Gear Selection, and Control Inputs

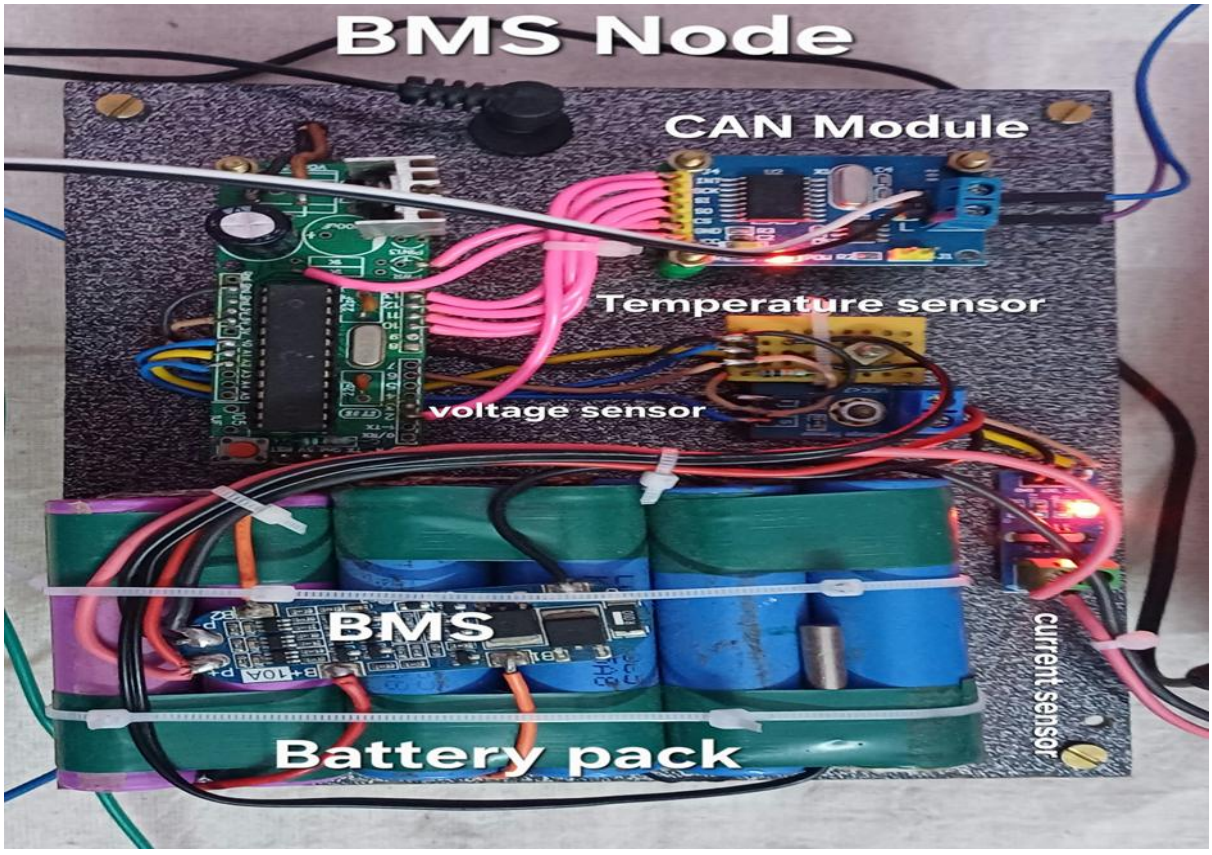


Fig. 5: BMS Node with Battery Pack and Sensor Array

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

VII. SOFTWARE IMPLEMENTATION AND RESULTS

A. Dashboard Interface

The user-facing interface is built using Streamlit, an open-source Python framework for deploying machine learning applications as interactive web apps. The dashboard is organized into three major sections: Vehicle Telemetry, AI Advisory Reports, and Model Transparency & Trust.

The Vehicle Telemetry section displays three primary metrics updated in real time: State of Charge as a percentage, Available Driving Range in kilometers, and State of Health as a percentage representing long-term battery condition.

The AI Advisory Reports section provides three categories of intelligent output. The Trip & Mission Advisory section tells the user whether their destination is reachable given the current range and suggests an optimal driving speed. The Energy Discharge Analysis section detects and alerts the user if abnormal current draw or thermal conditions are observed. The Maintenance & Service Advisory section evaluates battery health and categorizes it as Normal, Maintenance Required, or Critical — requiring replacement.

The Model Transparency & Trust section displays the accuracy metrics for each deployed model alongside their prediction confidence scores, expressed as percentages. This gives non-technical users a way to understand and trust the predictions they are seeing.

B. Output Demonstration

Two scenarios were demonstrated during system testing to validate end-to-end behavior:

Scenario	Input Condition	System Output
Normal Operating Condition	High SoC (75%), stable voltage and current, low load	Mission feasible, normal discharge, battery in healthy condition
Critical Condition	Low SoC (13.5%), reduced range (22 km), high discharge rate	Mission failure alert, abnormal discharge warning, low battery advisory

Table IV: System Behavior Under Two Tested Scenarios

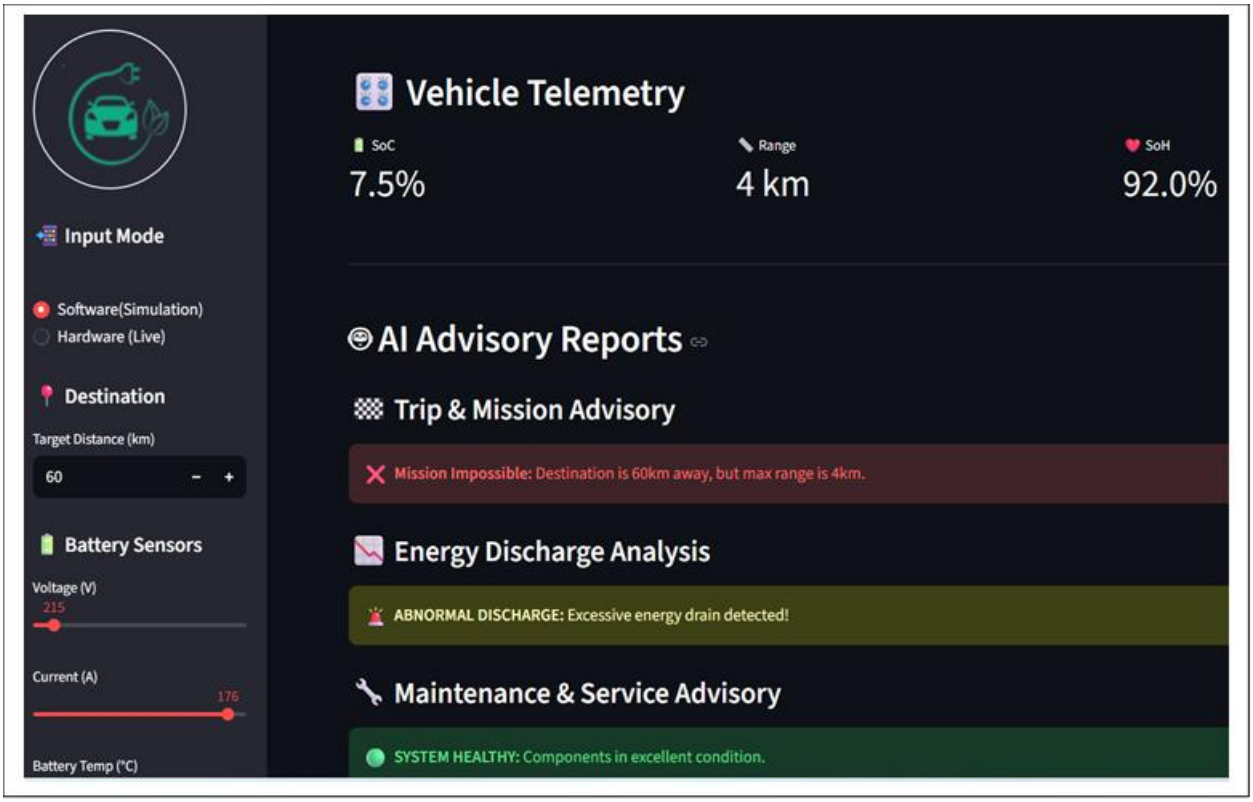


Fig. 7: Dashboard Output — Critical Condition (SoC: 7.5%, Range: 4 km, Mission Failure Alert)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

C. Accuracy and Confidence Analysis

To give users insight into how reliable the predictions are, the system includes a Model Transparency panel that displays both model accuracy (measured on the test dataset during training) and real-time prediction confidence scores. These are shown through a radar chart and a bar chart comparison side by side.

Model	Accuracy (Test Set)	Confidence Score	Status
State of Charge (SoC)	94.2%	N/A (regression)	Data Required
Range Prediction	91.6%	≈ 90%	High Congruence
Battery Health (SoH)	92.8%	≈ 92%	High Congruence
Abnormal Discharge Detection	95.1%	100%	Conservative Estimate

Table V: Model Accuracy vs. Prediction Confidence Scores

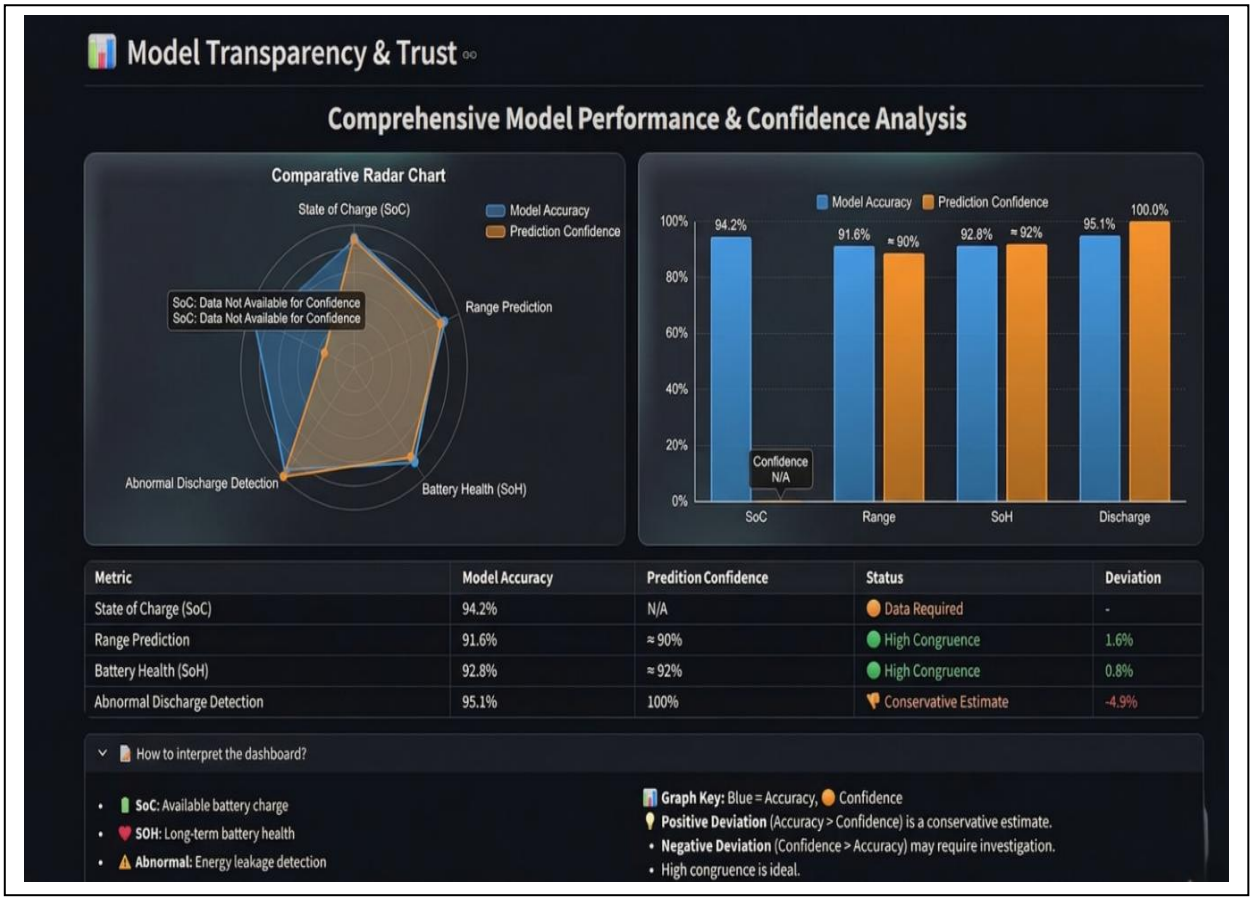


Fig. 8: Model Transparency & Trust Panel — Radar Chart and Accuracy-Confidence Comparison

D. Comparison with Traditional EV Systems

Criteria	Traditional EV Dashboard	Proposed AI-Based System
Data Representation	Raw sensor values only	Processed, meaningful insights
Predictive Capability	Not supported	SoC, Range, SoH, Fault Detection
Decision Support	None	Real-time recommendations and alerts

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Criteria	Traditional EV Dashboard	Proposed AI-Based System
Fault Detection	Reactive (after failure)	Proactive (before failure)
Personalization	Generalized outputs	Context-aware, condition-specific
User Interface	Static numerical display	Interactive, advisory-driven dashboard

Table VI: Comparison Between Traditional and Proposed AI-Based EV Systems

VIII. APPLICATIONS AND FUTURE SCOPE

A. Applications

The system is designed to be modular and deployable across multiple domains. In consumer EV applications, it can be integrated as an onboard intelligence layer that transforms the standard dashboard into an active advisory system. In fleet management, operators can use the cloud-connected interface to monitor multiple vehicles simultaneously, identify batteries showing early degradation signs, and schedule maintenance proactively rather than reactively. In smart charging infrastructure, the battery health and usage data can inform adaptive charging recommendations — reducing overcharging and thermal stress that silently shorten battery life.

Beyond commercial deployment, the system has strong value as an educational and research platform. The combination of embedded hardware, CAN communication, machine learning, and cloud deployment makes it a practical learning environment for students and engineers interested in IoT, AI, and EV systems.

B. Future Enhancements

Feature	Current System	Future Enhancement
Prediction Model	Random Forest	Deep Learning (LSTM, DNN)
Data Handling	Local / Flask API	Cloud IoT (AWS, Azure, GCP)
User Interface	Streamlit Web Dashboard	Mobile App + Voice Assistant
Explainability	Confidence Scores	SHAP / LIME (Explainable AI)
Communication	Local System	V2X and IoT Integration
Maintenance	Alert-Based	Automated Scheduling
Scalability	Single Vehicle Prototype	Multi-Vehicle Fleet Deployment

Table VII: Current System vs. Future Enhanced System

One of the most impactful future enhancements would be replacing the Random Forest models with Long Short-Term Memory (LSTM) networks, which are inherently suited to time-series data like battery charge/discharge cycles. LSTMs can capture temporal dependencies over long sequences — something Random Forests fundamentally cannot do — enabling far more accurate long-term SoH prediction and degradation trend modeling.

IX. CONCLUSION

This paper presented the Eco-Intelligent EV Dashboard with AI Insights, a complete hardware-software system for real-time electric vehicle battery monitoring and predictive maintenance. The system successfully combines distributed embedded sensing using Arduino-based nodes and CAN bus communication, centralized processing on a Raspberry Pi 4, machine learning inference using Random Forest models, and a dual-mode user interface through both a local QML dashboard and a Streamlit web application.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The machine learning models achieve strong performance across all prediction tasks: 94.2% R^2 for SoC prediction, 91.6% for range estimation, 92.8% for battery health, and 95.1% accuracy for abnormal discharge detection. The system's dual-mode architecture makes it flexible enough for both simulation and real deployment scenarios, and the API-based communication layer ensures that hardware and software can operate independently and scale easily.

What makes this system meaningful beyond its numbers is what it actually does for the person driving the vehicle. It removes uncertainty. It tells the driver whether they can complete their trip, warns them when the battery is behaving abnormally, and recommends when maintenance is genuinely needed rather than leaving them to guess. That shift — from passive display to active intelligent advisor — is the core contribution of this work.

The system lays a solid foundation for next-generation smart EV monitoring. With future enhancements including deep learning models, cloud IoT integration, mobile application interfaces, and Vehicle-to-Everything (V2X) communication, this platform has the potential to evolve into a fully autonomous intelligent EV ecosystem.

ACKNOWLEDGMENT

The authors sincerely thank Mrs. Pavithra V, M.E., Assistant Professor, Department of Electronics and Communication Engineering, Renganayagi Varatharaj College of Engineering, Sivakasi, for her continuous guidance and support throughout this project. They also extend gratitude to Dr. S. Thayammal, Head of the Department, for her encouragement, and to the college management for providing the facilities and resources necessary to complete this work.

REFERENCES

[1] S. Saxena et al., "Battery Management Systems for Electric Vehicles: Architecture, Modeling, and Algorithms," IEEE Transactions on Transportation Electrification, vol. 7, no. 3, pp. 1234–1246, 2021.
[2] J. Zhang, Y. Li, and M. Chen, "Machine Learning-Based Predictive Maintenance for Electric Vehicle Batteries," International Journal of Electrical Power & Energy Systems, vol. 134, 2022.
[3] S. Konkimalla, "AI-Based Predictive Maintenance for Electric Vehicles: Enhancing Reliability and Performance," International Journal of Advanced Research in Electrical and Electronics Engineering, 2022.
[4] X. Hu et al., "A Comparative Study of Equivalent Circuit Models for Li-ion Batteries," Journal of Power Sources, vol. 198, pp. 359–367, 2012.
[5] R. S. Kumar et al., "Hybrid Machine Learning Framework for Predictive Maintenance and Anomaly Detection in Lithium-Ion Batteries Using Enhanced Random Forest," Energy and AI, 2023.
[6] DatasetEngineer, "EV IoT Predictive Maintenance Dataset," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/datasetengineer/eviot-predictive-maint-dataset
[7] Raspberry Pi Foundation, "Raspberry Pi 4 Model B Technical Documentation," [Online]. Available: https://www.raspberrypi.com/documentation/
[8] Streamlit Inc., "Streamlit: Open-Source Framework for Machine Learning Applications," [Online]. Available: https://streamlit.io/
[9] Hugging Face, "Model Hosting and Inference APIs Documentation," [Online]. Available: https://huggingface.co/docs