**Abstract**

This report presents a comprehensive battery management system (BMS) for electric vehicles, integrating sensor data collection, cloud-based data storage, and machine learning for predictive analytics. The system employs an ESP32 microcontroller connected to current, voltage, and temperature humidity sensors to monitor battery parameters. Data is uploaded to the ThingSpeak cloud, where it is accessed by a Random Forest Regression model to predict charging times and potential travel distances. Additionally, an Arduino-based subsystem monitors vehicle safety using an ultrasonic sensor for obstacle detection and an IR sensor for brake pad condition, providing real-time updates to the user via an LCD display.

**Introduction**

As electric vehicles (EVs) become increasingly prevalent, efficient battery management is crucial to ensure optimal performance and longevity. Monitoring battery health and predicting performance can prevent unexpected failures and enhance user experience. This project aims to develop an advanced BMS using an ESP32 and Arduino microcontrollers to monitor various parameters, update data to the cloud, and utilize machine learning for predictive insights. Additionally, safety features are incorporated to enhance vehicle reliability and user safety.

**Scope**

The scope of this project includes:

1. Development of a sensor network for real-time monitoring of vehicle battery parameters.

2. Cloud integration for data storage and retrieval.

3. Implementation of a machine learning model for predictive analytics.

4. Development of safety features using additional sensors.

5. Real-time user updates through an LCD display.

**Motivation**

The motivation for this project stems from the need to enhance the reliability and performance of EVs through advanced battery management. By leveraging IoT and machine learning technologies, we aim to provide users with actionable insights into their vehicle's battery health and safety, ultimately improving the overall EV experience.

**Problem Statement**

Current EV battery management systems often lack predictive capabilities and real-time safety monitoring. Users need a solution that not only monitors battery health but also predicts charging times and travel distances. Additionally, real-time safety features such as obstacle detection and brake pad condition monitoring are essential for preventing accidents and ensuring vehicle safety.

**Objectives**

The primary objectives of this project are:

1. To design and implement a real-time battery monitoring system using ESP32.

2. To upload battery data to the ThingSpeak cloud for storage and analysis.

3. To develop a Random Forest Regression model to predict battery charging times and travel distances.

4. To integrate safety monitoring features using an Arduino microcontroller.

5. To provide real-time updates to the vehicle user through an LCD display.

**Existing System**

Current battery management systems in EVs primarily focus on monitoring basic parameters such as voltage and current. These systems typically lack advanced predictive capabilities and do not provide comprehensive safety monitoring. Additionally, data storage and analysis are often limited, reducing the potential for advanced insights and optimization.

**Proposed System**

The proposed system enhances traditional BMS by integrating advanced IoT and machine learning technologies. Key components include:

1. ESP32 Microcontroller: Connected to current, voltage, and temperature humidity sensors to monitor battery parameters in real-time.

2. ThingSpeak Cloud Integration: Battery data is uploaded to the cloud for storage and retrieval.

3. Machine Learning Model: A Random Forest Regression model is used to predict battery charging times and travel distances based on cloud-stored data.

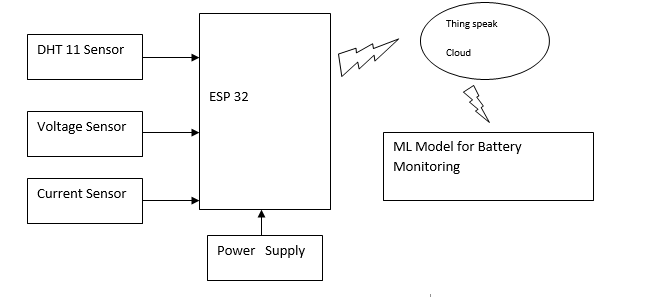
4. Arduino Subsystem: Equipped with an ultrasonic sensor for obstacle detection and an IR sensor for brake pad condition monitoring, with updates displayed on an LCD.

5. User Interface: Real-time updates provided to the vehicle user through an LCD display for informed decision-making.

By implementing this system, users can benefit from enhanced battery performance insights and improved vehicle safety, contributing to a more reliable and efficient EV experience.

**METHODOLOGY AND BLOCK DIAGRAM**

**Battery Management :**

****

Vehicle Safety Module:

Arduino

Ultrasonic Sensor

IR Sensor

Display

**Methodology**

1. Vehicle Battery Monitoring System

1.1. System Components:

- ESP32: Microcontroller for data acquisition and communication.

- Current Sensor: Measures the current flowing to and from the battery.

- Voltage Sensor: Measures the battery voltage.

- Temperature and Humidity Sensor: Monitors environmental conditions affecting the battery.

- ThingSpeak Cloud: Platform for data storage and real-time visualization.

- Machine Learning Model: Random Forest Regression for predicting charging time and travel distance.

1.2. Data Acquisition:

- Setup: Integrate the current sensor, voltage sensor, and temperature humidity sensor with the ESP32 microcontroller.

- Programming ESP32: Develop firmware using the Arduino IDE to read sensor data at regular intervals.

- Data Collection: Collect real-time data from all sensors (current, voltage, temperature, and humidity) and preprocess it for transmission.

- Transmission to ThingSpeak: Use the ESP32’s Wi-Fi capability to send the collected data to the ThingSpeak cloud.

1.3. Data Storage and Processing:

- ThingSpeak Configuration: Set up channels in ThingSpeak to store sensor data. Create fields for current, voltage, temperature, and humidity readings.

- Data Upload: Program the ESP32 to update ThingSpeak channels with sensor data at predefined intervals.

1.4. Machine Learning Model:

- Data Retrieval: Fetch historical data from ThingSpeak using its API.

- Data Preprocessing: Clean and preprocess the fetched data, handling missing values and outliers.

- Feature Selection: Select relevant features for the machine learning model (e.g., current, voltage, temperature, humidity, time of day).

- Model Training: Use Random Forest Regression to train the model on historical data to predict battery charging time and travel distance.

- Model Validation: Validate the model using cross-validation techniques and adjust parameters to improve accuracy.

1.5. Prediction and Monitoring:

- Real-Time Predictions: Deploy the trained model to make real-time predictions based on incoming data from ThingSpeak.

- User Interface: Develop a user-friendly interface to display predictions and battery health status.

2. Obstacle Detection and Brake Pad Condition Monitoring System

2.1. System Components:

- Arduino: Microcontroller for sensor interfacing and data processing.

- Ultrasonic Sensor: Detects obstacles in the vehicle’s path.

- IR Sensor: Monitors the condition of the brake pads.

- LCD Display: Provides real-time updates to the vehicle user.

2.2. Sensor Integration:

- Ultrasonic Sensor Setup: Connect the ultrasonic sensor to the Arduino to measure the distance to obstacles.

- IR Sensor Setup: Attach the IR sensor to the brake pads and connect it to the Arduino to detect wear and tear.

2.3. Programming Arduino:

- Firmware Development: Write Arduino code to continuously read data from the ultrasonic and IR sensors.

- Obstacle Detection Algorithm: Implement algorithms to process ultrasonic sensor data and detect obstacles within a certain range.

- Brake Pad Condition Algorithm: Develop logic to assess brake pad condition based on IR sensor readings.

2.4. Data Processing and Display:

- Real-Time Data Processing: Ensure the Arduino processes sensor data in real-time to provide immediate feedback.

- LCD Integration: Connect an LCD display to the Arduino to present obstacle distance and brake pad condition to the vehicle user.

- User Alerts: Program the system to trigger alerts on the LCD when obstacles are detected within a critical range or when brake pad wear exceeds safe limits.

**Hardware and Software Description**

**ESP 32:**

ESP32 is a microcontroller board. ESP32 devkit has 36 pins and 18 on each side of the board as shown in the picture above. It has 34 GPIO pins and each pin has multiple functionalities which can be configured using specific registers. There are many types of GPIOs available like digital input, digital output, analog input, and analog output, capacitive touch, UART communication and USB cable.

A ESP32, consist of two functions:

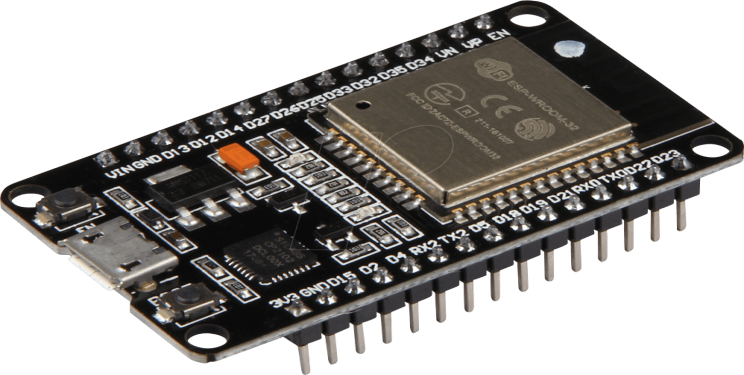
**setup():** This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed .

**loop():** After setup() is called, this function is called repeatedly by a program loop in the main program. It controls the board until it is powered off or is reset.

ESP32 is 32-bit microcontroller,provides low power & connectivity ,wifi & bluetooth for low energy consumption and Low power system on chip(SOC),clock speed of ESP32 can be controlled independently. ESP32 has dual core processor,one is application cpu which handles code and another is protocol cpu which handles wifi and indicative peripheral for A2D controller and D2A controller .Below mentioned are the traits of suggested imitation:

(i).Collection of health data : In order to determine the health status of a patient, we need to consider various health parameters and instead of using different devices for the collection of data, this single IOT device will collect real time and accurate data. (ii).Analyzing the data : The data collected will be stored in the cloud.The variations in the data for specific duration can be plotted as graphs.This graph helps the doctor to analyze the patient’s responsiveness to the medication.

(iii).Prediction of recovery rate: After Analyzing the data, we can predict the recovery rate of the Patient accurately in real-time using Machine-Learning Algorithm considering the dataset.



**Figure 1: ESP32 Micro –Controller**

**Regulated power supply:**



**Transformer:**

A transformer is a device that transfers electrical energy from one circuit to another through inductively coupled conductors without changing its frequency. A varying current in the first or primary winding creates a varying magnetic flux in the transformer's core, and thus a varying magnetic field through the secondary winding. This varying magnetic field induces a varying electromotive force (EMF) or "voltage" in the secondary winding. This effect is called mutual induction. If a load is connected to the secondary, an electric current will flow in the secondary winding and electrical energy will be transferred from the primary circuit through the transformer to the load. This field is made up from lines of force and has the same shape as a bar magnet. If the current is increased, the lines of force move outwards from the coil. If the current is reduced, the lines of force move inwards. If another coil is placed adjacent to the first coil then, as the field moves out or in, the moving lines of force will "cut" the turns of the second coil. As it does this, a voltage is induced in the second coil. With the 50 Hz AC mains supply, this will happen 50 times a second. This is called MUTUAL INDUCTION and forms the basis of the transformer.

**Rectifier:**

A rectifier is an electrical device that converts alternating current (AC) to direct current (DC), a process known as rectification. Rectifiers have many uses including as components of power supplies and as detectors of radio signals. Rectifiers may be made of solid-state diodes, vacuum tube diodes, mercury arc valves, and other components. A device that it can perform the opposite function (converting DC to AC) is known as an inverter. When only one diode is used to rectify AC (by blocking the negative or positive portion of the waveform), the difference between the term diode and the term rectifier is merely one of usage, i.e., the term rectifier describes a diode that is being used to convert AC to DC. Almost all rectifiers comprise a number of diodes in a specific arrangement for more efficiently converting AC to DC than is possible with only one diode. Before the development of silicon semiconductor rectifiers, vacuum tube diodes and copper (I) oxide or selenium rectifier stacks were used.

**Filter:**

The process of converting a pulsating direct current to a pure direct current using filters is called as filtration. Electronic filters are electronic circuits, which perform signal-processing functions, specifically to remove unwanted frequency components from the signal, to enhance wanted ones.

**Regulator:**

A voltage regulator (also called a ‗regulator‘) with only three terminals appears to be a simple device, but it is in fact a very complex integrated circuit. It converts a varying input voltage into a constant ‗regulated ‘output voltage. Voltage Regulators are available in a variety of outputs like 5V, 6V, 9V, 12V and 15V. The LM78XX series of voltage regulators are designed for positive input. For applications requiring negative input, the LM79XX series is used. Using a pair of ‗voltage-divider‘ resistors can increase the output voltage of a regulator circuit. It is not possible to obtain a voltage lower than the stated rating. You cannot use a 12V regulator to make a 5V power supply. Voltage regulators are very robust. These can withstand over-current draw due to short circuits and also over-heating. In both cases, the regulator will cut off before any damage occurs. The only way to destroy a regulator is to apply reverse voltage to its input. Reverse polarity destroys the regulator almost instantly. Fig: 3u shows voltage regulator.

**Software Algorithm and Implementation Details:**

**How Random Forest Algorithm Works?**

There are two stages in Random Forest algorithm, one is random forest creation, theother is to make a prediction from the random forest classifier created in the first stage. The whole process is shown below, and it’s easy to understand using thefigure.

firstly, shows the Random Forest creation pseudocode:

1. Randomly select “**K**” features from total “**m**” features where **k <<m**
2. Among the “**K**” features, calculate the node “**d**” using the best splitpoint
3. Split the node into **daughter nodes** using the **bestsplit**
4. Repeat the **a to c** steps until “l” number of nodes has beenreached
5. Buildforestbyrepeatingsteps**atod**for“n”numbertimestocreate**“n”numberoftrees**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **F11** | **F12** | **F13** | **F14** | **F15** | **T1** |
| **F21** | **F22** | **F23** | **F24** | **F25** | **T2** |
| **:** | **:** | **:** | **:** | **:** | **:** |
| **:** | **:** | **:** | **:** | **:** | **:** |
| **Fm1** | **Fm2** | **Fm3** | **Fm4** | **Fm5** | **Tm** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **F11** | **F12** | **F13** | **F14** | **F15** | **T1** |
| **F81** | **F82** | **F83** | **F84** | **F85** | **T8** |
| **:** | **:** | **:** | **:** | **:** | **:** |
| **:** | **:** | **:** | **:** | **:** | **:** |
| **Fj1** | **Fj2** | **Fj3** | **Fj4** | **Fj5** | **Tj** |

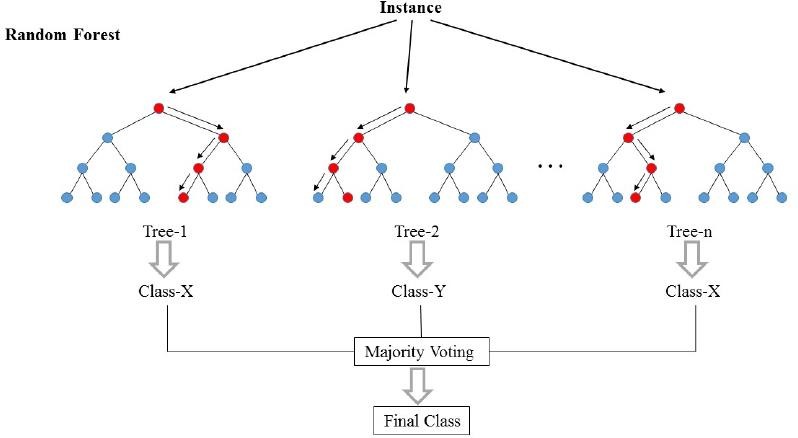
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **F21** | **F22** | **F23** | **F24** | **F25** | **T2** |
| **F51** | **F52** | **F53** | **F54** | **F55** | **T5** |
| **:** | **:** | **:** | **:** | **:** | **:** |
| **:** | **:** | **:** | **:** | **:** | **:** |
| **Fm1** | **Fm2** | **Fm3** | **Fm4** | **Fm5** | **Tm** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **F31** | **F32** | **F33** | **F34** | **F35** | **T3** |
| **F61** | **F62** | **F63** | **F64** | **F65** | **T6** |
| **:** | **:** | **:** | **:** | **:** | **:** |
| **:** | **:** | **:** | **:** | **:** | **:** |
| **Fk1** | **Fk2** | **Fk3** | **Fk4** | **Fk5** | **Tk** |

**Fig: 1.1 Different random forest trees dataset**

In the next stage, with the random forest classifier created, we will make the prediction. The random forest prediction pseudocode is shown below:

1. Takesthe**testfeatures**andusetherulesofeachrandomlycreateddecisiontreetopredict the outcome and stores the predicted outcome(target)
2. Calculate the **votes** for each predictedtarget
3. Consider the **high voted** predicted target as the **final prediction** from the random forest algorithm



**Fig: 1.2 Random Forest Tree**

**Applications of Random Forest**

* **Banking**Randomforestalgorithmisusedtofindloyalcustomers,whichmeanscustomers whocantakeoutplentyofloansandpayinteresttothebankproperly,andfraudcustomers, which means customers who have bad records like failure to pay back a loan on time or have dangerousactions.
* **Medicine** Random forest algorithm can be used to both identify the correct combination of components in medicine, and to identify diseases by analyzing the patient’s medical records.
* **Stock Market** Random forest algorithm can be used to identify a stock’s behaviour and the expected loss orprofit.
* **E-Commerce** Random forest algorithm can be used for predicting whether the customer willliketherecommendproducts,basedontheexperienceofsimilarcustomers.

Advantages of Random Forest Algorithm

* 1. For applications in classification problems, Random Forest algorithm will avoid the overfittingproblem
  2. For both classification and regression task, the same random forest algorithm can be used
  3. The Random Forest algorithm can be used for identifying the most importantfeatures from the training dataset, in other words, featureengineering.

**Advantages and Applications**

**Advantages**

Battery Management:

1. Real-Time Monitoring: The ESP32, connected with current, voltage, and temperature humidity sensors, provides continuous monitoring of battery parameters, ensuring real-time updates.

2. Predictive Maintenance: By analyzing data through a machine learning model, potential issues can be identified before they lead to failure, enhancing vehicle reliability.

3. Cloud Integration: Data updates to the ThingSpeak cloud facilitate remote monitoring and historical data analysis.

4. Improved Efficiency: Predicting charging time and travel distance using Random Forest Regression helps in optimizing battery usage and planning trips efficiently.

5. Cost Savings: Proactive maintenance and optimized battery usage reduce the overall operational costs.

Vehicle Safety Management:

1. Enhanced Safety: The Arduino system with ultrasonic sensors for obstacle detection significantly reduces the risk of collisions.

2. Timely Alerts: IR sensors for brake pad condition detection ensure timely maintenance, preventing brake failures.

3. User-Friendly: Updates through an LCD provide easy-to-understand information, improving user awareness and vehicle management.

4. Cost-Effective: The system is relatively low-cost yet effective, making it accessible for various vehicle models.

5. Integration Capability: The system can be integrated with other vehicle management systems for comprehensive monitoring.

**Applications**

Battery Management:

1. Electric Vehicles (EVs): Ensures optimal battery performance and longevity in electric cars, bikes, and scooters.

2. Fleet Management: Useful for companies managing large fleets of electric vehicles to monitor battery health and predict maintenance needs.

3. Smart Grids: Integrates with smart grids for efficient energy distribution and usage.

4. Renewable Energy Storage: Manages battery systems in solar and wind energy storage applications.

Vehicle Safety Management:

1. Passenger Vehicles: Enhances safety features in personal cars, ensuring driver and passenger safety.

2. Commercial Vehicles: Important for trucks and delivery vehicles to avoid accidents and ensure timely maintenance.

3. Public Transport: Improves safety in buses and trains, reducing the risk of accidents and ensuring regular maintenance.

4. Autonomous Vehicles: Essential for self-driving cars to navigate safely and perform regular system checks.

**Conclusion**

The integration of advanced battery management and vehicle safety systems marks a significant step towards smarter and safer transportation. By leveraging the capabilities of ESP32 for battery monitoring and Arduino for safety management, these systems provide real-time data, predictive maintenance, and enhanced safety features. The use of machine learning models to predict charging times and travel distances further optimizes vehicle performance and user convenience. Overall, these innovations not only improve efficiency and safety but also contribute to cost savings and environmental sustainability**.**

**References:**

[1] Karthik Shivaram, Nikhil Rajendra “Power line communication based Automation system using handheld Wi-Fi Device" in University of Michigan, USA; Stanford University, 2012ISCE1569550137.

2] Wong,E.M.C.;,"A phone-based remote controller for home and office automation," Consumer Electronics, IEEE Transactions on , vol.40, no.1,pp.28- 34,Feb1994 doi: 10.1109/30.273654.

Coskun, I.; Ardam, H.; "A remote controller for home and office appliances by telephone,"Consumer Electronics, IEEE Transactions on, vol.44,no.4,pp.1291- 1297, Nov 1998doi:10.1109/30.735829.

Koyuncu,B.;,"PC remote control of appliances by using telephone lines," Consumer Electronics, IEEE Transactions on, vol.41, no.1, pp.201-209,Feb1995 doi:10.1109/30.370328

Zhang X.; Sun J.; Zhou L., “Development of an Internet Home Automation System using Java and Dynamic DNS Service”, Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies(PDCAT ’05)0- 7695-2405-2/05

Rosendahl, A.; Hampe,J.Felixand Botterweck, G., “Mobile Home Automation- Merging Mobile Value Added Services and Home Automation Technologies”, Sixth International Conference on the Management of Mobile Business (ICMB 2007)0-760695-2803-1/07

Sriskanthan N.; Tam, F.; Karande, A. “Bluetooth based home automation system”, Microprocessors and Microsystems 26,2002, pp281-289.

Toru Yamagata , Takasi Hosoya, Masamitsu Tokudaand Takashi Matsuo, ”Transmission characteristics of a high speed power line communication system”, Musashi Institute of Technology,Tamazutsumi, Setagaya-Ku,Tokyo,158- 8557 Japan.