EV BATTERY MONITORING SYSTEM WITH MACHINE LEARNING

A Project Report

Submitted to the Faculty of ENGINEERING of

JAWAHARLAL NEHURU TECHNOLOGICAL UNIVERSITY, KAKINADA

In partial fulfillment of the requirements

for award of the Degree of

Bachelor of Technology

in

Electrical and Electronics Engineering

By

G.V.Durga Prasad
(22485A0212)

B.Renuka lakshmi Bai
(21481A0217)

K.Hema Tanuja
(21481A0247)

B.Vivek Kanaka Roy
(21481A0214)

Under the Guidance of

Mr.P.Subhani Khan

Assistant Professor

Department of Electrical and Electronics Engineering

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

SESHADRIRAO KNOWLEDGE VILLEGE

GUDLAVALLERU-521356

ANDHRA PRADESH

2025

`

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK , Kakinada)

SESHADRI RAO KNOWLEDGE VILLEGE

GUDLAVALLERU-521356

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **"EV BATTERY MONITORING SYSTEM WITH MACHINE LEARNING"** is a bonafide record of work carried out by **G.V. Durga Prasad(224850A212), B. Renuka Lakshmi Bai(21481A0217), K. Hema Tanuja(21481A0247) and B. Vivek Kanka Roy(21481A0214)** under my guidance and supervision in partial fulfillment of the requirements for award of the degree of Bachelor of Technology in Electrical And Electronics Engineering of Jawaharlal Nehru Technological University, Kakinada.

Mr.P.Subhani Khan
Project Guide

Dr.A.Amarendra
B.E,M.TECH,PhD
Professor&HOD

`

ACKNOWLEDGEMENT

We are glad to express our deep sense of gratitude to **Mr.P.Subhani Khan,** Assistant Professor, Electrical and Electronics Engineering for his guidance and cooperation in completing this project. Through this we want to convey our sincere thanks to him for inspiring assistance during our project.

We express our heartful gratitude and deep indebtedness to our beloved Head of the Department **Dr. A. Amarendra,** for his great help and encouragement in doing our project successfully.

We also express our gratitude to our Principal **Dr. B. Karuna Kumar**, for his encouragement and facilities provided during the course of project.

We express our heartful gratitude to our **Faculty members** and **Lab Technician**s for providing a great support for us in completing our project.

We thank one and all who have rendered help to us directly or indirectly in the completion of this work.

G.V.Durga Prasad (22485A0212)
B.Renuka Lakshmi Bai (21481A0217)
K.Hema Tanuja (21481A0247)
B.Vivek Kanaka Roy(21481A0214)

CONTENTS

Page No.

List of Figures VI

List of Tables VII

Abstract VIII

Chapter 1: Introduction **1-3**

1.1 Background 1

1.2 Aim Of Project 1

1.3 Methodology 1

1.4 Significance of Work 2

1.5 Outline of the project 3

Chapter 2: Literature Review **4-5**

2.1 Problem Statement 4

2.2 Existing System 5

2.3 Contribution Of Proposed system 5

Chapter 3: Experimental Design **6-7**

3.1 Block Diagram 6

3.2 System Operation 7

Chapter 4: Hardware Implementation **8-16**

4.1 Components 8

4.2 Hardware Setup 15

4.3 Working 16

Chapter 5: Software Description **17-25**

5.1 ThingSpeak Cloud Integration 17

5.2 Machine Learning with Python 17

5.3 Machine Learning and ThingSpeak Workflow 19

5.4 Software codes 21

5.4.1 Arduino code

5.4.2 Python code

Chapter 6: Results **26-27**

Chapter 7: Conclusion And Future Scope **28-29**

7.1 Conclusion 28

7.2 Future Scope 29

Appendices **30**

Reference **31**

LIST OF FIGURES

Fig. No.	Figure Name	Page No.
4.1.1	**Arduino UNO**	8
4.1.2	**Node MCU**	9
4.1.3	**DC Current Sensor**	9
4.1.4	**Temperature Sensor**	10
4.1.5	**Potentiometer**	10
4.1.6	**Buzzer**	11
4.1.7	**16*2 LCD Display**	11
4.1.8	**DC Cooling Fan**	12
4.1.9	**Battery**	12
4.1.10	**Connecting Wires**	13
4.1.11	**Relay Module**	13
4.1.12	**GSM Module**	14
4.1.13	**Fire Sensor**	14
4.1.14	**LED**	15
4.2	**Hardware Setup**	15
5.3.1	**ThingSpeak Cloud**	19
5.3.2	**Open IDLE PYTHON app**	20
5.3.3	**Select the main.py file from recent files**	20
5.3.4	**Run the code**	20
6.0	**Results**	26

LIST OF TABLES

Table No.	Table Title	Page No.
1	Test Cases	30

ABSTRACT

Battery storage forms the most important part of any electric vehicle (EV) as it stores the necessary energy for the operation of EV. So, in order to extract the maximum o/p of a battery & to ensure its safe operations it is necessary that a efficient battery management system exist is the same .It monitors the Parameters, determine SOC and provide necessary services to ensure safe operation of battery. Hence BMS forms a integral part of any EV and safe guards both the user and the battery by ensuring that the cell operates within its safe operating parameters. The proposed system only monitor the battery and charge it safely but also protect it to avoid accidents from occurring. The proposed model has following functions current, voltage measurement, fire, protection, battery status detection, liquid crystal display (LCD) etc. Electric vehicles (EVs) are automobiles powered by one or more electric motors, which draw energy from rechargeable batteries instead of relying solely on internal combustion engines (ICEs) that consume fossil fuels. A Battery Management System (BMS) is a critical component in electric vehicles (EVs) and other battery-powered systems. It monitors and controls the operation of the battery pack, ensuring its optimal performance, safety, and longevity.

CHAPTER 1

INTRODUCTION

1.1 Background

With the rising demand for sustainable transportation, Electric Vehicles (EVs) have become a practical and eco-friendly solution. At the heart of any EV lies its battery system, which stores and supplies energy to drive the motor. However, ensuring the safe, efficient, and prolonged operation of the battery requires continuous monitoring and intelligent management.

Modern EV batteries are sensitive to various parameters such as voltage, current, temperature, and potential fire hazards. Unchecked deviations in these parameters can lead to performance degradation or even dangerous situations. Therefore, an advanced Battery Monitoring System (BMS) integrated with machine learning (ML) for predictive analysis is crucial.

1.2 Aim Of Project

This project aims to design and implement a real-time EV battery monitoring system using sensors and Arduino Uno, enhanced with GSM communication, ThingSpeak cloud integration, and machine learning techniques. The goal is to ensure:

- Real-time data monitoring (voltage, current, temperature, fire).
- Automatic alerts to the user via GSM in case of critical conditions.
- Cloud-based data visualization using ESP8266 and ThingSpeak.
- Predictive battery health classification using machine learning algorithms.
- System protection through relay-controlled power cutoff, fan activation, and buzzer/LED alerts.
- Addition of a relay for voltage/current cutoff in abnormal conditions.

1.3 Methodology

- Use INA219 sensor to measure both voltage and current.
- Monitor temperature using LM35 and detect fire through a fire sensor.
- Use a potentiometer to simulate load/variations for testing.
- Interface all sensors with Arduino Uno.
- Display readings on a 16x2 LCD and activate LEDs, buzzer, and cooling fan as needed.

Transmit data via:

- GSM Module: for SMS alerts to users.
- ESP8266 Wi-Fi Module: to send data to ThingSpeak Cloud.
- After transmitting data to ThingSpeak, the system logs the information on the cloud platform for remote monitoring and data visualization.
- Data can be analyzed in real-time on Python IDLE/serial monitor, offering a local view of battery health and other parameters.
- The system triggers protective actions when critical thresholds are exceeded:
- Relay module disconnects the battery power (voltage and current) in case of abnormal conditions (e.g., high temperature, overcurrent, or fire).
- LED indicators and buzzer are activated to alert the user.
- The cooling fan is activated through the relay to prevent overheating.

Machine Learning Integration:

Sensor data is recorded and used to train a machine learning model offline (using algorithms like Random Forest) for predictive battery health classification.

1.4 Significance Of Work

- Ensures battery safety by real- time detection of critical conditions.
- Helps in preventing hazards such as overheating or fire.
- Improves battery lifespan through predictive health analytics.
- Enables remote monitoring via cloud and GSM.
- Reduces maintenance costs and improve user awareness of battery performance.
- Provides a scalable, cost-effective solution for student EV projects and prototypes
- Facilitates early fault detection, reducing the risk of unexpected breakdowns.
- Promotes energy efficiency by monitoring battery usage and charging patterns.
- Enhances system reliability with automatic alerts and warning indicators.

1.5 Outline Of Project

- Chapter 1 introduces the project background, aim, methodology, and significance.
- Chapter 2 provides a literature review and Problem statement, existing system and Proposed system.
- Chapter 3 discusses the experimental design, block diagram and operation.
- Chapter 4 details of components and their implementation.
- Chapter 5 provides software integration, and real-time working.
- Chapter 6 presents results, graphs, and performance discussion.
- Chapter 7 concludes with findings and future enhancements.

CHAPTER 2

Literature Review

2.1 Problem Statement

Electric vehicles (EVs) are becoming a vital part of the modern transportation ecosystem due to their environmental benefits and energy efficiency. At the core of every EV lies the battery system, which directly affects the vehicle's performance, reliability, and safety. However, conventional battery monitoring systems are limited in scope, lacking the intelligence to detect faults early or to predict battery health over time. This leads to unexpected failures, increased maintenance costs, and potential safety risks. An advanced, smart monitoring system is essential to ensure the long-term performance and safety of EVs, especially one that integrates machine learning and cloud technologies.

The major problems with current EV battery monitoring systems include:

- Lack of Real-Time Fault Detection.
- Inability to instantly detect conditions like overcurrent, overheating, or fire risks.
- No Predictive Health Analysis.
- Absence of machine learning or data-driven methods to forecast battery degradation or failure.
- Limited Monitoring Parameters.
- Conventional systems monitor only basic parameters, ignoring deeper insights like charge cycles, efficiency, or state of health.
- No Emergency Response Mechanism.
- Systems lack automatic responses like triggering a cooling fan, buzzer, or system shutdown during faults.
- No User Alerts or Notifications.
- Users are not warned in real time when battery anomalies are detected.
- Absence of Cloud Integration.
- Lack of IoT or cloud support prevents remote data logging, analysis, and visualization.
- High Maintenance Dependency.
- Reliance on manual checks increases maintenance time, cost, and chances of human error.

2.2 Existing System

Conventional battery monitoring systems used in electric vehicles primarily function by reading sensor data such as voltage, current, and temperature through microcontrollers like Arduino or similar embedded systems. These values are compared against predefined safe operating thresholds, and basic alerts like indicator lights or buzzers are triggered if any anomalies are detected. While such systems serve the fundamental purpose of real-time monitoring and provide immediate feedback on unsafe conditions, they are largely reactive in nature. They do not possess the capability to anticipate potential faults before they occur, which could otherwise prevent damage or safety hazards. Moreover, most of these systems operate in isolation without cloud integration, which restricts the user's ability to monitor the battery remotely or analyse historical data. There is also little to no implementation of intelligent systems such as machine learning algorithms that can recognize patterns or provide early warnings.

In addition, such systems typically lack communication features like GSM for remote alerts and do not include automated protection components such as relays to disconnect power during critical conditions. Data visualization is often limited or absent, making it difficult for users to interpret trends or understand the battery's performance over time. As a result, current systems are inadequate for comprehensive battery health management, creating a strong demand for smarter and more connected monitoring solutions.

2.3 Contribution of the Proposed System

To overcome the limitations of existing battery monitoring systems, the proposed project introduces an intelligent EV Battery Monitoring System with Machine Learning and Cloud Integration. This system is designed to provide real-time monitoring of crucial battery parameters such as voltage, current, temperature, and fire detection using sensors like INA219, LM35, and a flame sensor, all interfaced with an Arduino Uno microcontroller. A potentiometer is used to simulate load variations for testing purposes.

Unlike conventional systems, this design incorporates machine learning algorithms to analyze sensor data and recognize abnormal patterns, allowing the system to predict potential issues before they become critical. This enables proactive maintenance, improves safety, and extends the lifespan of the battery pack. In addition to local monitoring through the LCD display, buzzers, and automatic fan control for thermal management, the system is enhanced with a relay module that disconnects power automatically in case of severe conditions like overheating, overcurrent, or fire risk. This adds an essential layer of protection by isolating the battery during emergencies.

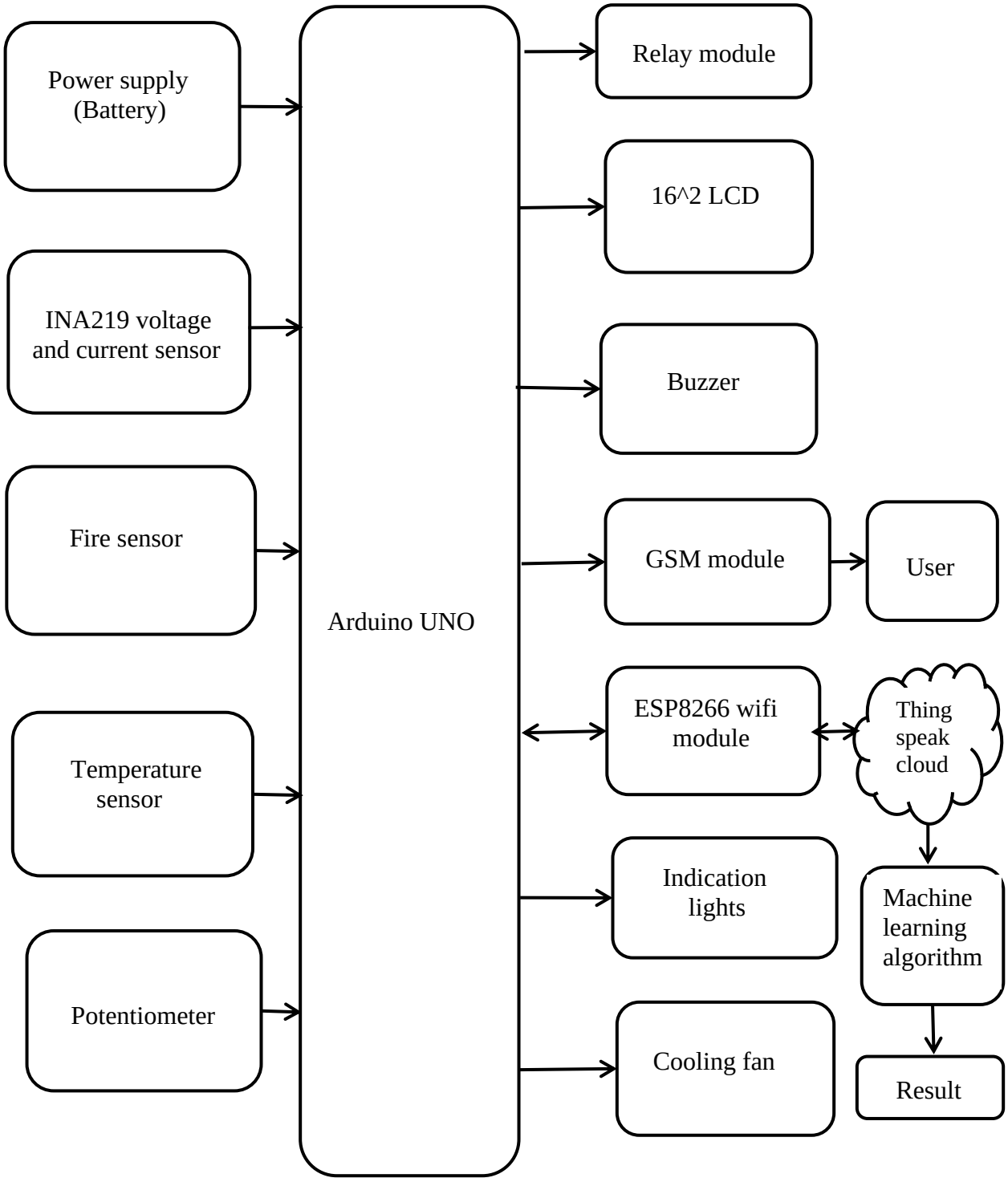
The system also integrates a GSM module to send SMS alerts directly to the user when abnormal conditions are detected, ensuring they are informed even when away from the vehicle or monitoring unit.

Furthermore, cloud connectivity is achieved using the ESP8266 Wi-Fi module, which allows continuous data transmission to the ThingSpeak cloud platform. This enables remote monitoring and visualization of real-time performance as well as historical data trends through a web-based dashboard. By combining machine learning, cloud access, automated protection, and user alerts, the proposed system provides a comprehensive, intelligent, and highly reliable EV battery monitoring solution.

CHAPTER 3

EXPERIMENTAL DESIGN

3.1 Block Diagram



3.2 System Operation

The block diagram illustrates the working architecture of the EV Battery Monitoring System integrated with machine learning and IoT capabilities. The system is powered by a battery source, which also supplies power to various components. The core of the system is the Arduino Uno microcontroller, which acts as the central unit for data acquisition, processing, and control. Several sensors are interfaced with the Arduino to monitor key battery parameters. The INA219 sensor is used to measure real-time voltage and current values of the battery, while a temperature sensor continuously monitors the battery temperature to prevent overheating. A fire sensor is included to detect the presence of flames, enhancing the system's safety features. A potentiometer is also connected to simulate a variable load during testing.

The Arduino Uno processes the sensor data and controls several output components based on predefined thresholds. If abnormal conditions are detected, a relay is activated to control the operation of a cooling fan, helping to regulate temperature. A buzzer provides audible alerts in the event of fire, high current, or excessive temperature. Real-time data such as voltage, current, and temperature are displayed on a 16x2 LCD screen for easy monitoring. Additionally, the system incorporates a GSM module to send SMS alerts to the user, ensuring prompt notifications about critical battery conditions.

For IoT integration, an ESP8266 Wi-Fi module is used to transmit the monitored data to the ThingSpeak cloud platform. This enables remote monitoring, data visualization, and supports machine learning applications for battery health prediction and anomaly detection. Visual indicators in the form of LEDs provide an immediate understanding of system status. Overall, this intelligent monitoring system enhances the safety, reliability, and performance of EV batteries by combining sensor data, wireless communication, and predictive analytics.

CHAPTER 4

Hardware Implementation

4.1 Hardware Components Used

1.Arduino UNO:

The Arduino UNO is the primary microcontroller used in this system. It is based on the ATmega328P microchip and features multiple digital and analog input/output pins. The Arduino handles data collection from various sensors, processes logic for triggering alarms or actions, and communicates with other modules like the GSM and ESP8266. It is cost-effective, beginner-friendly, and has robust community support.

Key Features:

- 14 digital I/O pins (6 with PWM)
- 6 analog input pins
- USB programming interface
- Operates at 5V, 16 MHz clock speed

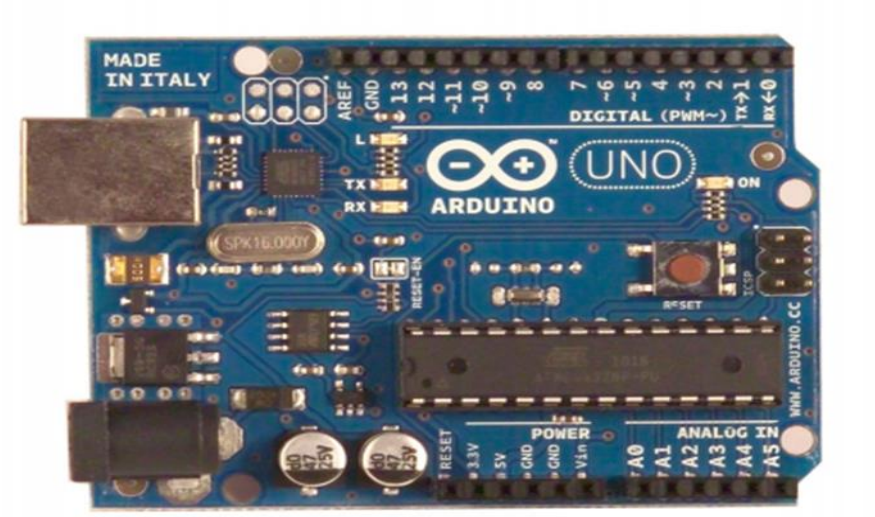


Fig 4.1.1: Arduino UNO

2.Wifi Module(ESP8266)

The ESP8266 is a compact and powerful Wi-Fi module that connects the system to the internet. It sends real-time data (such as voltage, current, and temperature) to the ThingSpeak cloud platform, enabling remote monitoring from any smart device.

Key Features:

- 802.11 b/g/n standard Wi-Fi support
- TCP/IP protocol stack
- Communicates via UART with Arduino
- Low power consumption

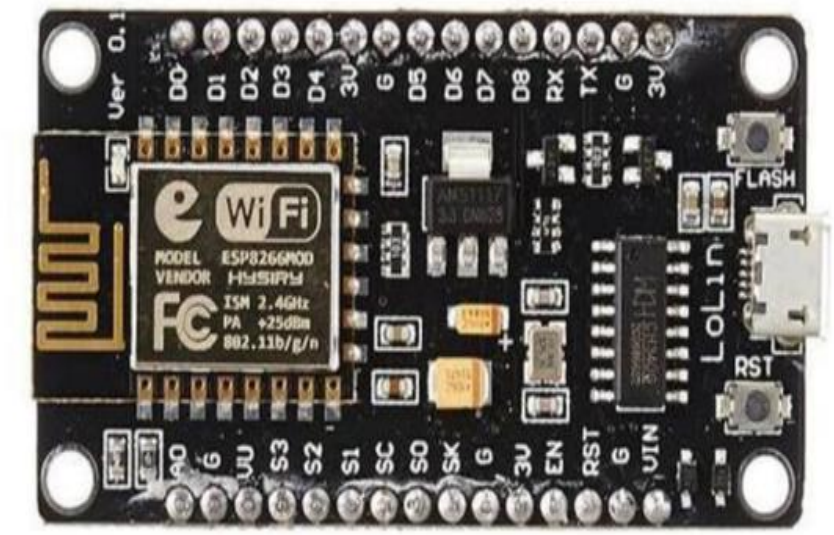


Fig 4.1.2: Node MCU

3. DC Current Sensor (INA219)

The INA219 is a precision sensor used to measure both voltage and current flowing through the battery. It uses a high-side current sensing approach and communicates with Arduino via the I2C protocol. The sensor provides real-time monitoring of the power consumption of the battery.

Key Features:

- ❖ Measures up to 26V and ±3.2A
- ❖ I2C Interface
- ❖ High accuracy and low power consumption

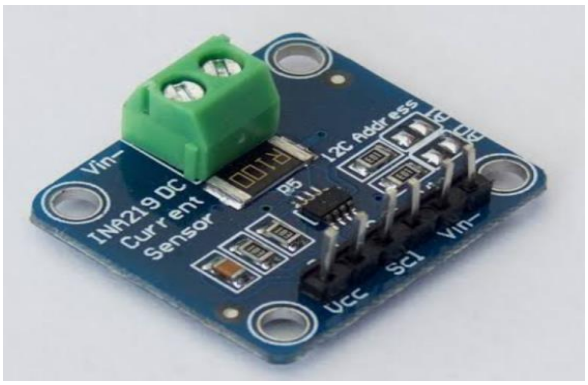


Fig 4.1.3.:DC Current Sensor

4.Temperature Sensor (LM35)

The LM35 is a precision analog temperature sensor that provides a voltage output linearly proportional to the Celsius temperature. It helps monitor battery temperature and ensures that the system takes appropriate actions (e.g., fan activation) when overheating is detected.

Key Features:

- ❖ Output: 10 mV/°C
- ❖ Operates from 4V to 30V
- ❖ High accuracy and low self-heating

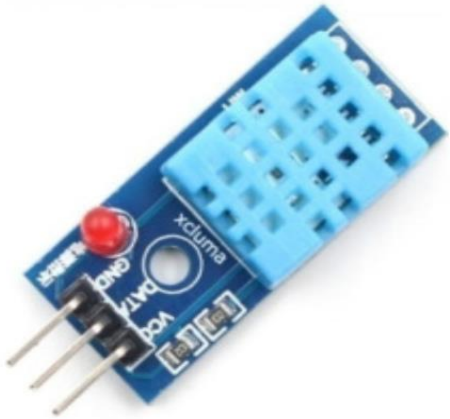


Fig 4.1.4: Temperature Sensor

5.Potentiometer

A potentiometer is a manually adjustable variable resistor used here for simulating load or voltage variations during testing. It allows for dynamic input conditions, mimicking battery behavior under different operational loads.

Key Features:

- ❖ Rotary knob for variable resistance
- ❖ Analog voltage output
- ❖ Helps simulate voltage drops or changes

Fig 4.1.5: Potentiometer

6.Buzzer

The buzzer is an audio output device used to generate alert sounds when critical conditions are detected. It warns nearby users immediately upon detecting issues like overheating or fire.

Key Features:

- ❖ Loud and clear tone
- ❖ Low power requirement
- ❖ Activated by Arduino

Fig 4.1.6: Buzzer

7.16*2 LCD Display

A 16x2 LCD display is used for showing real-time sensor values such as voltage, current, and temperature. It provides users with direct and immediate system feedback without needing a computer or smartphone.

Key Features:

- ❖ Displays 2 rows of 16 characters
- ❖ Operates at 5V
- ❖ Interfaced with Arduino in 4-bit or 8-bit mode

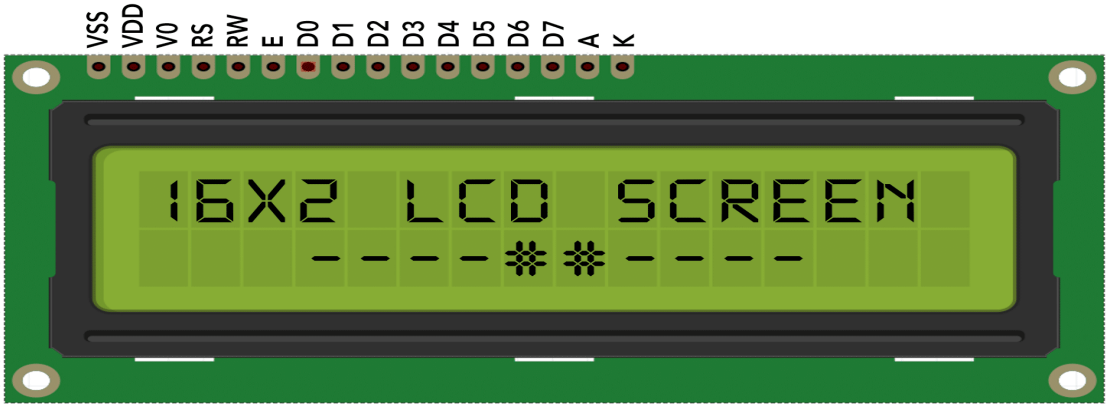


Fig 4.1.7: 16*2 LCD Display

8.DC Cooling Fan

A DC cooling fan is used to regulate battery temperature. It is turned ON automatically by the Arduino when the battery temperature exceeds the safety threshold. This enhances thermal management and safety of the system.

Key Features:

- ❖ Operates on 5V or 12V DC
- ❖ Quiet and efficient airflow
- ❖ Extends battery life by preventing overheating

Fig 4.1.8: DC Cooling Fan

9.Battery

The battery supply component provides the necessary DC power to the entire system, including the Arduino Uno and connected sensors. It ensures stable voltage and current levels for accurate monitoring and operation. A regulated battery pack or adapter (typically 9V or 12V) is used to power the circuit reliably.

`

Fig 4.1.9: Battery

10.Connecting Wires

The Connecting wires are essential components used to establish electrical connections between different modules and components in the circuit. In this project, jumper wires (male-to-male, male-to-female, and female-to-female) are used to link the Arduino Uno with sensors like INA219, LM35, the ESP8266 Wi-Fi module, LCD display, and the fan. These wires ensure smooth communication and power transfer between components, maintaining a reliable hardware interface for data monitoring and system operation.

Fig 4.1.10: Connecting Wire

11.Relay Module

A relay module is an essential electronic component that allows low-power control of high-voltage or high-current devices. It acts as an electrically operated switch, enabling microcontrollers like Arduino, Raspberry Pi, or ESP8266 to control devices such as motors, lights, or fans. Relay modules are typically available in single or multi-channel formats, making them suitable for controlling one or more devices simultaneously. They often include features like optocouplers for electrical isolation, status LEDs for visual feedback, and screw terminals for secure wiring. When a low-voltage signal is applied to the module, it energizes an internal coil, creating a magnetic field that triggers a mechanical switch. This action completes the high-voltage circuit, allowing the connected device to operate safely. Relay modules are widely used in home automation, robotics, and industrial control systems due to their ability to safely bridge the gap between low-power control circuits and high-power devices.

Fig 4.1.11: Relay Module

12.GSM Module

A GSM module (Global System for Mobile Communications module) is a device used to enable microcontrollers like Arduino or ESP8266 to communicate over a mobile network. It allows electronic systems to send and receive data, SMS messages, or make voice calls by connecting to a cellular network using a SIM card. One of the most commonly used GSM modules is the SIM800 or SIM900 series. These modules operate using AT commands sent via serial communication to control the functions of the module. When integrated into projects, GSM modules are ideal for applications requiring remote monitoring, such as home automation, vehicle tracking, or security systems. They allow data transmission even in areas without Wi-Fi access, making them valuable for IoT applications. Overall, the GSM module provides a reliable and flexible way to add mobile connectivity to embedded systems.

Fig 4.1.12:GSM Module

13.Fire Sensor

A fire sensor, also known as a flame sensor, is an electronic device used to detect the presence of fire or flames. It works by sensing infrared (IR) or ultraviolet (UV) light emitted by a flame. One commonly used fire sensor in electronic projects is the IR flame sensor, which is sensitive to the infrared light spectrum typically produced by fire.Fire sensors are widely used in fire alarm systems, industrial safety equipment, and in projects like fire-fighting robots or EV battery monitoring systems to enhance safety by activating alerts or shutdown mechanisms when fire is detected.

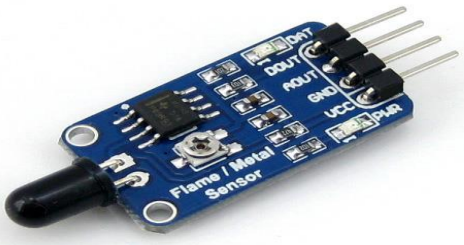


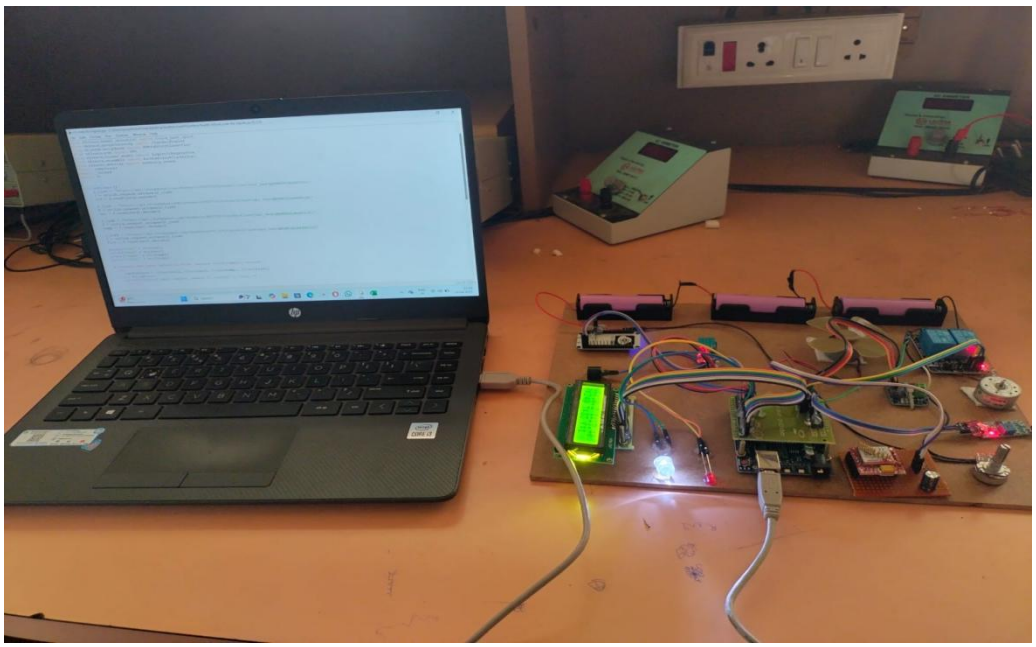
Fig 4.1.13: Fire Sensor

14.LED

An LED (Light Emitting Diode) is widely used in electronic systems for indication purposes, providing a clear and immediate visual signal of the system's status. It serves as a simple yet effective tool to communicate various conditions such as power on/off, data activity, warnings, or successful operations. LEDs can be programmed to turn on, off, or blink based on specific events, making them ideal for monitoring and troubleshooting. For instance, a green LED might indicate normal operation, a red LED could signal the detection of fire or fault, and a yellow LED might warn of overheating. This use of different colors and blinking patterns enhances user interaction and helps operators quickly assess the state of a system. In embedded systems and safety-critical applications, LED indication is a reliable and essential feature for real-time feedback.

Fig 4.1.14:LED

4.2 Hardware Setup



4.3 working

The EV Battery Monitoring System operates by continuously sensing critical battery parameters and responding accordingly to ensure safety and efficiency. The system begins with the power supply, which energizes the Arduino Uno and connected components. The INA219 sensor continuously measures the voltage and current from the battery, while the temperature sensor monitors battery temperature. A fire sensor is integrated to detect any fire hazards, and a potentiometer is used to simulate variable loads during testing and calibration.

The Arduino Uno collects and processes data from these sensors in real-time. If the temperature rises above a predefined threshold, the system automatically activates the cooling fan through a relay to reduce heat and prevent thermal damage. In the case of overcurrent or fire detection, the buzzer is triggered to alert nearby users, and indicator lights provide a quick visual warning. The 16x2 LCD display shows live data values, including voltage, current, and temperature, for local monitoring.

To enhance safety and accessibility, the GSM module sends SMS alerts to the user's phone in critical conditions. Simultaneously, the ESP8266 Wi-Fi module transmits the sensor data to the ThingSpeak cloud platform, enabling remote monitoring and historical data analysis. This cloud data is further used for machine learning-based predictions of battery health and anomaly detection.The machine learning is integrated to predict the battery’s health. The sensor data is recorded over time and used to train an offline machine learning model, such as a Random Forest..etc and providing early warnings about potential failures. Through this intelligent integration of sensors, communication modules, and cloud services, the system ensures reliable, real-time battery management and proactive fault prevention.

CHAPTER 5
Software Description

5.1 ThingSpeak Cloud Integration

ThingSpeak is used as the cloud platform for storing and visualizing the data collected from the EV battery monitoring system. The Arduino Uno, connected to an ESP8266 Wi-Fi module, sends real-time sensor data, such as voltage, current, power, and temperature, to the ThingSpeak cloud at regular intervals. This data is uploaded via the ThingSpeak API, where each sensor's readings are mapped to specific fields in the ThingSpeak channel. For example, the voltage readings are mapped to Field 1, current to Field 2, power to Field 3, and temperature to Field 4.

The uploaded data can be accessed remotely through the ThingSpeak dashboard, which displays live graphs and visualizations. These graphs allow users to monitor the health of the EV battery in real time. If any of the sensor values go beyond predefined thresholds (e.g., low voltage or high temperature), alerts can be configured in ThingSpeak to notify users of critical conditions. This cloud integration ensures that the system is continuously monitored and allows for immediate actions in case of anomalies.

5.2 Machine Learning with Python

To enhance the monitoring system, Python is used for machine learning analysis on the historical data stored in ThingSpeak. By leveraging the cloud-based data, machine learning algorithms can predict battery performance, detect anomalies, and identify potential failures before they occur, enabling predictive maintenance.

Python is chosen for this machine learning implementation because of its robust libraries and tools for data analysis and model training. The following machine learning algorithms are applied to the data:

- ❖ K-Nearest Neighbors (KNN)

- In this project, KNN is used to classify sensor readings like voltage, current, temperature, and fire detection based on previously collected labeled data.

- It compares new real-time data with similar past data to determine whether the readings indicate a condition that requires triggering hardware actions.

- For example, if temperature and current are slightly above threshold, KNN identifies that similar cases in the past triggered a fan or buzzer and does the same in the present.

- This allows the system to make intelligent decisions based on pattern similarity rather than fixed threshold values.

- ❖ Random Forest

- Random Forest is used to analyze multiple input features and predict complex fault conditions by creating and combining multiple decision trees.

- It learns from combinations of values like high temperature with low voltage or sudden current increases, which are often signs of developing faults.

- These trees collectively vote on whether to activate the buzzer, fan, or red light in response to the situation.

- It helps ensure that protective actions are only taken when necessary, even when the input data contains minor fluctuations or noise.

❖ Support Vector Machine (SVM)

- SVM is applied to clearly separate normal sensor input combinations from those that represent abnormal or unsafe battery behavior.
- It draws a decision boundary between data patterns that led to hardware action (e.g., activating fan or buzzer) and those that didn't.
- This allows the system to respond precisely when sensor values like voltage drop or temperature rise approach critical combinations.
- SVM ensures accurate activation of protection mechanisms with minimal false triggers.

❖ Linear Regression

- Linear Regression is used to predict future trends in voltage, current, and temperature by analyzing how these parameters change over time.
- For instance, if the temperature shows a rising trend, the model forecasts the future value and determines whether it will exceed safety limits.
- This helps the system activate the fan or buzzer before the actual critical value is reached, allowing preventive protection.
- It also helps in estimating battery performance over time, aiding in maintenance planning.

To implement this, the following steps are involved:

1. Data Export from ThingSpeak: The historical data stored in ThingSpeak is retrieved using the ThingSpeak API. Python scripts can query the ThingSpeak channel and download the data for training the machine learning models.
2. Data Preprocessing and Feature Engineering: Once the data is retrieved, it undergoes preprocessing, such as data cleaning, normalization, and feature extraction, to make it suitable for machine learning. For example, missing values are handled, and sensor readings are standardized to improve model performance.
3. Model Training: With the preprocessed data, machine learning models are trained using Python libraries such as scikit-learn. The models are trained on historical sensor data to learn the patterns
and correlations between various features like voltage, current, and temperature.
4. Real-Time Predictions: Once the models are trained, they are deployed to make real-time predictions based on new sensor data uploaded to ThingSpeak. For instance, as the Arduino sends new data to ThingSpeak, Python can be used to fetch the latest data and make predictions regarding the battery's health or potential failure.
5. Cloud-Based Insights: The predictions made by the machine learning models are visualized through ThingSpeak. The results are sent back to ThingSpeak using Python scripts, allowing users to view the predictive insights on the live dashboard. This helps users understand the battery's health status and take proactive actions.

5.3 Machine Learning and ThingSpeak Workflow

The integration of Python-based machine learning with ThingSpeak follows a structured workflow:

1. Data Collection: The Arduino collects real-time sensor data and uploads it to ThingSpeak via the ESP8266 Wi-Fi module.

2. Data Storage and Visualization: ThingSpeak stores the uploaded data in a cloud-based channel, allowing users to access live graphs and receive alerts in case of abnormal readings.

3. Data Analysis with Python: The historical data from ThingSpeak is exported using Python. The data is then preprocessed, and machine learning models are trained on this data using Python libraries like scikit-learn.

4. Predictive Maintenance: The trained machine learning models are deployed to predict potential battery failures and classify the battery's health status in real time. Python scripts continuously fetch the latest data from ThingSpeak, apply the model, and generate predictions.

5. Insights and Alerts: The predicted results are visualized on ThingSpeak, where users can see the battery's predicted status (e.g., "healthy," "critical"). Alerts can be triggered based on machine learning predictions, providing proactive notifications to users.

ThingSpeak Workflow:

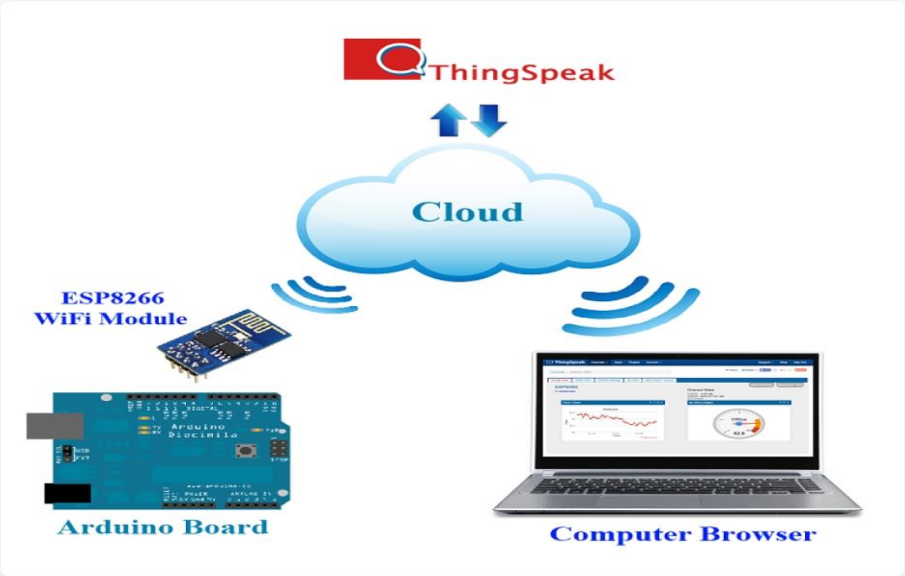


Fig 5.3.1:ThingSpeak Cloud

Machine Learning Workflow:

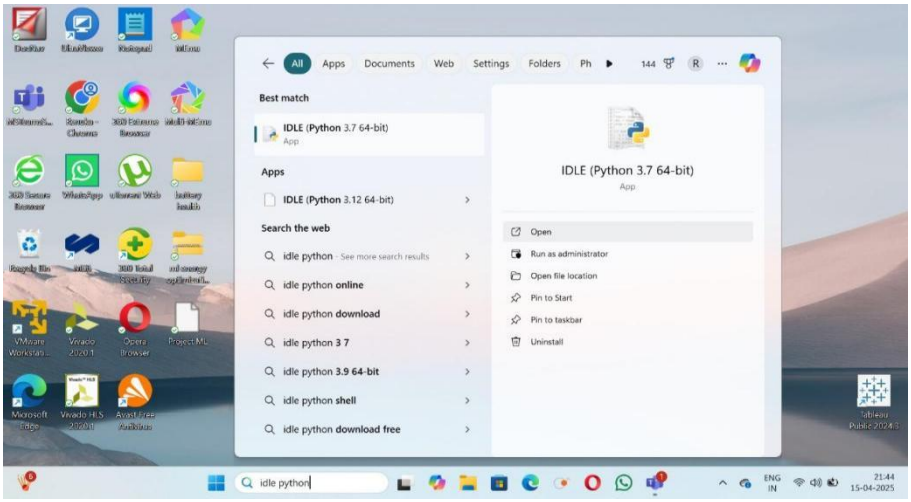


Fig 5.3.2:Open IDLE PYTHON app

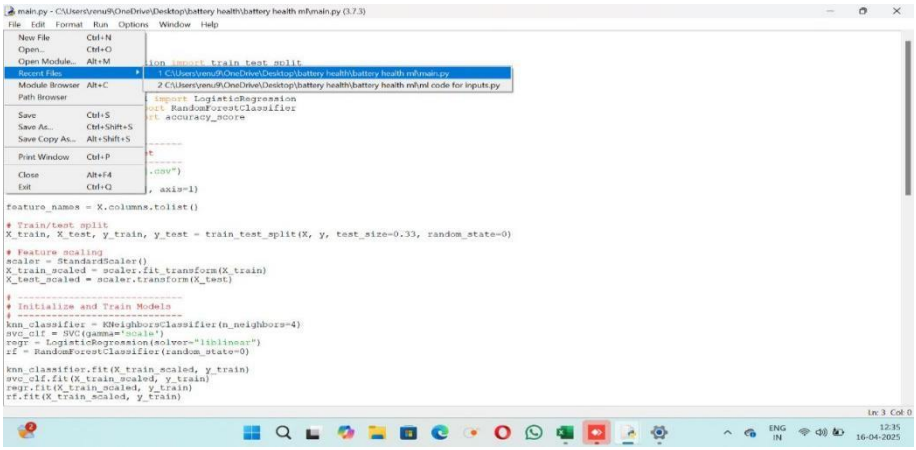


Fig 5.3.3:Select the main.py file from recent files

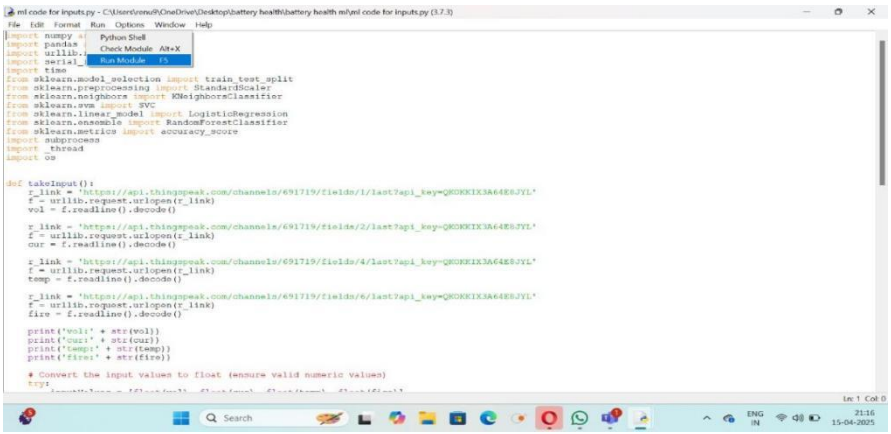


Fig 5.3.4:Run the code

5.4 Software code

5.4.1 Arduino Code

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(A2,A1);
#include <Wire.h>
#include <Adafruit_INA219.h>
#include <Wire.h>
#include <LiquidCrystal.h>
const int rs = 8, en =9, d4 =10, d5 =11, d6 =12, d7 =13;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
Adafruit_INA219 ina219;
#include "DHT.h"
float R1 = 30000.0;
float R2 = 7500.0;
float ref_voltage = 5.0;
#define DHTPIN 3
int cnt=0;
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
int buz=4;

int g=7;
int r=6;
int r1=A3;
int fir=A0;
int mot1=2;
void setup(void)
{Serial.begin(9600);
 pinMode(mot1,OUTPUT);
 mySerial.begin(9600);
 lcd.begin(16,2);
 pinMode(buz,OUTPUT);
 pinMode(r,OUTPUT);
  pinMode(r1,OUTPUT);
 pinMode(g,OUTPUT);
 pinMode(fir,INPUT);
 digitalWrite(buz,0);
 digitalWrite(r1,0);
 lcd.print("  WELCOME ");
 delay(2000);
 lcd.clear();

  dht.begin();
  uint32_t currentFrequency;
  if (! ina219.begin()) {
    Serial.println("Failed to find INA219 chip");
    while (1) { delay(10);
    }
  }
}
```

```
void loop(void)
{
  float shuntvoltage = 0;
  float busvoltage = 0;
  float current_mA = 0;
  float loadvoltage = 0;
 int power_mW = 0;
  int cs=0;
  shuntvoltage = ina219.getShuntVoltage_mV();
  busvoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();
  power_mW = ina219.getPower_mW();
  loadvoltage = busvoltage + (shuntvoltage / 1000);
  if(busvoltage<1.1)
  busvoltage=0;
  if(current_mA<1.5)
  current_mA=0;
  if(power_mW<1.5)
  current_mA=0;
  int bp=map(busvoltage,2.5, 3.9,1,100);
  int t = dht.readTemperature();
    int h= dht.readHumidity();
  int fval=digitalRead(fir);
//  float b1v=((analogRead(A0)*5)/1024.0)/(R2/(R1+R2));
//  float b2v=((analogRead(A1)*5)/1024.0)/(R2/(R1+R2))-b1v;
//  float b3v=((analogRead(A2)*5)/1024.0)/(R2/(R1+R2))-b2v-b1v;
  lcd.clear();
  lcd.print("VT:"+String(busvoltage,1) + " I:"+String(current_mA));
  lcd.setCursor(0,1);
  lcd.print("P:" +String(power_mW)   +" T:"+String(t) +" f:"+String(fval));
  delay(1000);
if(busvoltage<6)
{  send_sms(1);
  lcd.setCursor(0,1);
  lcd.print("low voltage");
 digitalWrite(buz,1);
 digitalWrite(r,1);
  digitalWrite(g,0);
}
else if(t>37)
{ send_sms(2);
  digitalWrite(buz,1);
  digitalWrite(mot1,1);
   digitalWrite(r,1);
  digitalWrite(g,0);
   digitalWrite(r1,1);
}
else if(fval==0)
{ send_sms(3);
  digitalWrite(buz,1);
  digitalWrite(mot1,1);
   digitalWrite(r,1);
  digitalWrite(g,0);
   digitalWrite(r1,1);
}
```

```
else {
  digitalWrite(buz,0);
  digitalWrite(mot1,0);
   digitalWrite(r,0);
  digitalWrite(g,1);
   digitalWrite(r1,0);
}

cnt++;
if(cnt>4){
Serial.print("691719,R271DPA5VHGDB9SB,0,0,project1,12345678,"+String((busvo
ltage))+","+String((
current_mA))+","+String(power_mW)+","+String(t)+","+String(h)+","+String(fval)+
",\n");
 cnt=0;
  }
}
void send_sms(int k) {
   mySerial.println("AT");
  delay(1000);
  mySerial.println("ATE0");
  delay(1000);
  mySerial.println("AT+CMGF=1");
  delay(1000);
  mySerial.print("AT+CMGS=\"7032407249\"\r\n");
  delay(1000);
  if (k == 1)
 mySerial.print(" BATTERY LOW VOLTAGE WARNING PLEASE CHECK");
if (k == 2)
 mySerial.print("OVER TEMPRATURE PLEASE CHECK");
if (k == 3)
 mySerial.print("FIRE DETECTED PLASE CHECK ");

  delay(500);
 mySerial.write(26); // ASCII code for Ctrl+Z
  delay(2000);
```

5.4.2 Python Code

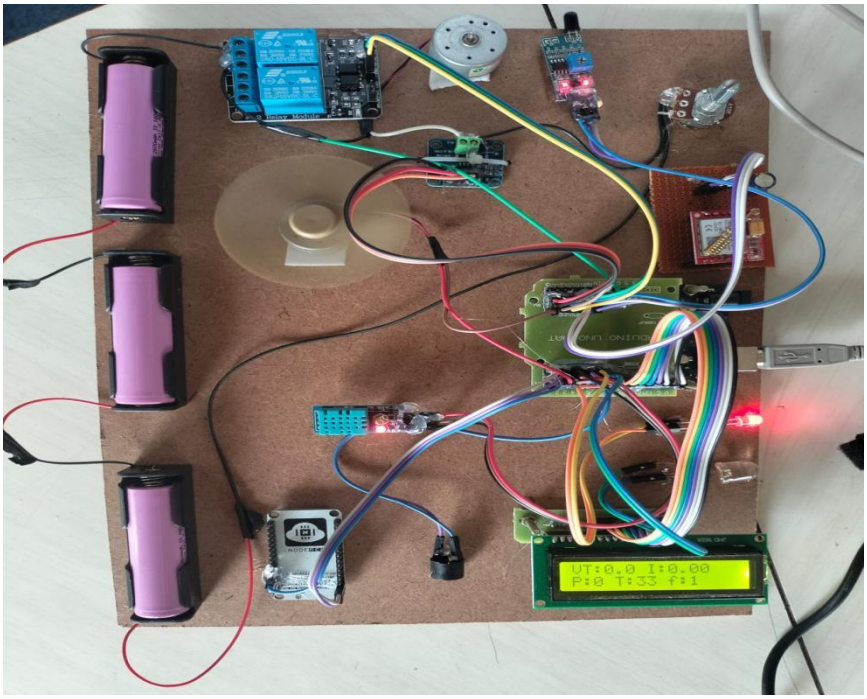
```
 import numpy as np
 import pandas as pd
 import urllib.request
 import serial_rx_tx
 import time
 from sklearn.model_selection import train_test_split
 from sklearn.preprocessing import StandardScaler
 from sklearn.neighbors import KNeighborsClassifier
 from sklearn.svm import SVC
 from sklearn.linear_model import LogisticRegression
 from sklearn.ensemble import RandomForestClassifier
 from sklearn.metrics import accuracy_score
 import subprocess
 import _thread
 import os
```

```
def takeInput():
    r_link =
'https://api.thingspeak.com/channels/691719/fields/1/last?api_key=QKOKKIX3A64
E8JYL'
    f = urllib.request.urlopen(r_link)
    vol = f.readline().decode()

    r_link =
'https://api.thingspeak.com/channels/691719/fields/2/last?api_key=QKOKKIX3A64
E8JYL'
    f = urllib.request.urlopen(r_link)
    cur = f.readline().decode()

    r_link =
'https://api.thingspeak.com/channels/691719/fields/4/last?api_key=QKOKKIX3A64
E8JYL'
    f = urllib.request.urlopen(r_link)
    temp = f.readline().decode()

    r_link =
'https://api.thingspeak.com/channels/691719/fields/6/last?api_key=QKOKKIX3A64
E8JYL'
    f = urllib.request.urlopen(r_link)
    fire = f.readline().decode()

    print('vol:' + str(vol))
    print('cur:' + str(cur))
    print('temp:' + str(temp))
    print('fire:' + str(fire))

    # Convert the input values to float (ensure valid numeric values)
    try:
        inputValues = [float(vol), float(cur), float(temp), float(fire)]
    except ValueError:
        print("Invalid input values, unable to convert to float.")
        return

    print("\n")

    # Make prediction using the trained model
    final_Result = rf.predict([inputValues])
    print(final_Result)

    if final_Result[0] == 0:
        print('normal condition')

    else:
        print('ABNormal Condition')

    print()
    print()
```

```
# Load dataset and prepare for training
accdt = pd.read_csv("proj.csv")
y = accdt['output']
X = accdt.drop(['output'], axis=1)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=0)

# Scale the data (StandardScaler or MinMaxScaler can be used)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize classifiers
knn_classifier = KNeighborsClassifier(n_neighbors=4)
svc_clf = SVC(gamma='scale')
regr = LogisticRegression(solver="liblinear")
rf = RandomForestClassifier(max_depth=2, random_state=0)

# Train classifiers
knn_classifier.fit(X_train, y_train)
svc_clf.fit(X_train, y_train)
regr.fit(X_train, y_train)
rf.fit(X_train, y_train)

# Make predictions
knn_preds = knn_classifier.predict(X_test)
svc_preds = svc_clf.predict(X_test)
regr_preds = regr.predict(X_test)
rf_preds = rf.predict(X_test)

# Print accuracy scores
print("Accuracy with KNN:", accuracy_score(y_test, knn_preds))
print("Accuracy with SVC:", accuracy_score(y_test, svc_preds))
print("Accuracy with LR:", accuracy_score(y_test, regr_preds))
print("Accuracy with RF:", accuracy_score(y_test, rf_preds))

# Run the loop for taking input and predicting condition
while True:
    takeInput()
```

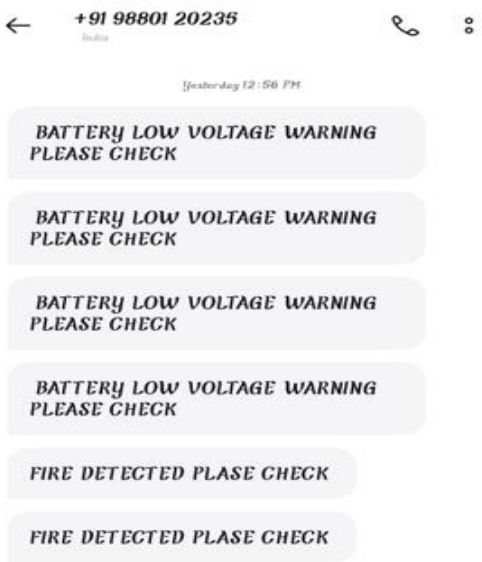
CHAPTER 6

RESULTS



,

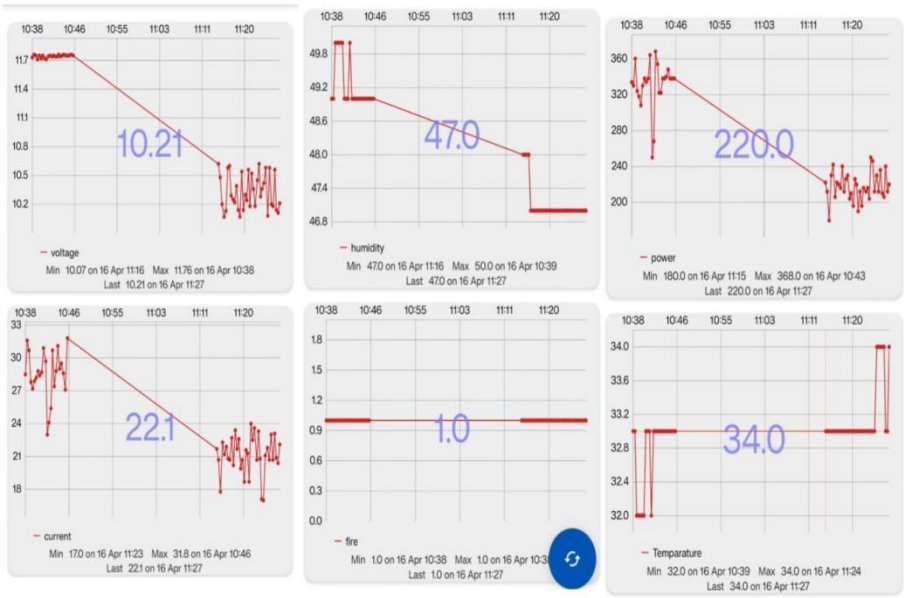
Auto Safety Response (Fan/LED/Buzzer)

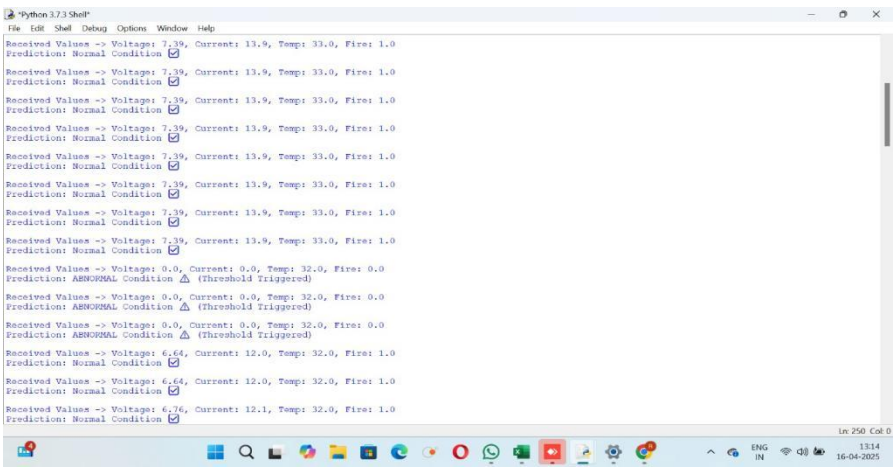


SMS Alert Notification

Live Status on LCD

Cloud & ML Monitoring Panels:





CHAPTER 7

Conclusion And Future Scope

7.1 Conclusion

In conclusion, an essential part of electric vehicles that guarantees the security, dependability, and longevity of the battery pack is the EV BMS with charge monitor and fire prevention. By supplying crucial safety features like temperature control, fault detection, cell balancing, and fire prevention, the system lowers the possibility of battery fires and enhances the overall efficiency of electric vehicles. In order to improve the features and capabilities of EV BMS with charge monitor and fire prevention, more research and development is still possible. A few potential future work areas include enhancing the precision and dependability of battery monitoring systems to deliver more accurate and timely data regarding the charge, health, and function of the battery pack.

7.2 Future Scope

- Advanced Data Analytics: Implementing advanced data analytics techniques to analyze battery performance data collected from IoT sensors. This can include predictive maintenance algorithms to anticipate battery failures or degradation, optimizing charging and discharging strategies, and identifying trends for performance improvement.

- Remote Configuration and Management: Enabling remote configuration and management of the battery monitoring system through IoT connectivity. Users can adjust settings, update firmware, and monitor system status from anywhere, enhancing convenience and accessibility.

- Integration with Energy Management Systems: Integrating the battery monitoring system with broader energy management systems to optimize energy usage and storage. This integration can facilitate smart grid integration, load balancing, and demand response, contributing to energy efficiency and grid stability.

- Enhanced Security Features: Implementing robust security features to protect the battery monitoring system from cyber threats and unauthorized access. This can include encryption protocols, secure authentication mechanisms, and intrusion detection systems to safeguard sensitive data and system integrity.

- Multi-Platform Compatibility: Ensuring compatibility with multiple IoT platforms and communication protocols to accommodate diverse deployment scenarios and interoperability with existing infrastructure. This can include integration with popular IoT platforms such as AWS IoT, Microsoft Azure IoT, or Google Cloud IoT, as well as support for protocols like MQTT, CoAP, or HTTP.

- Real-Time Monitoring and Alerts: Implementing real-time monitoring capabilities to provide immediate feedback on battery status and performance

- Energy Harvesting and Sustainability: Exploring energy harvesting techniques to power the battery monitoring system using renewable energy sources such as solar or kinetic energy. This can enhance system sustainability and reduce reliance on external power sources, especially in remote or off-grid locations.
- Integration with Smart Grid Technologies: Integrating the battery monitoring system with smart grid technologies such as demand response programs, grid-connected storage, or vehicle-to-grid (V2G) systems. This integration can enable bidirectional energy flow between the battery system and the grid, supporting grid stabilization and renewable energy integration initiatives.
- Community Engagement and Open Source Development: Fostering a community-driven approach to development by sharing designs, code, and documentation as open-source resources. This can encourage collaboration, innovation, and knowledge exchange among developers, researchers, and enthusiasts, driving continuous improvement and adoption of the battery monitoring system.

By exploring these future directions, the battery monitoring system using Arduino and IoT can evolve into a versatile and resilient solution for monitoring, managing, and optimizing battery performance across a wide range of applications, from renewable energy storage and electric vehicles to consumer electronics and industrial automation**.**

Appendices:

Test Cases:

S.NO	TEMPERATURE	VOLTAGE	CURRENT	FIRE	RESULT
1.	33.9	11.3	26.5	1.0	NORMAL
2.	35.7	11.9	28.4	1.0	NORMAL
3.	38.2	4.4	30.5	0.0	ABNORMAL
4.	39.8	4.0	31	0.0	ABNORMAL
5.	34.2	3.7	32.4	0.0	ABNORMAL
6.	36.1	3.2	33.6	0.0	ABNORMAL
7.	36.9	2.8	36.5	0.0	ABNORMAL

Table: Test Cases

References

1. Ayman S. Elwer , Samy M. Ghania , Nagat M. K. A. Gawad ," Battery Management Systems For Electric Vehicle Applications"

2. Aniket Rameshwar Gade," The New Battery Management System in Electric Vehicle" International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 IJERTV10IS070210 www.ijert.org Vol. 10 Issue 07, July-202

3. Nitin Saxena , Anant Singh , Aniket Dharme , "Iot Based Battery Management System Inelectric Vehicle".

4. Rui Hu University of Windsor." Battery Management System For Electric Vehicle Applications"2011

5. A.Sowmiya , P.Aileen Sonia Dhas , L.Aquiline Lydia , M.Aravindan , K.Rajsaran," Design of Battery Monitoring System for Electric Vehicle" IARJSET International Advanced Research Journal in Science, Engineering and Technology Vol. 8, Issue 11, November 2021

6. Aswinth Raj,"Battery Management System(BMS)forElectricVehicles" December 5, 2018 [8] Anjali Vekhande, Ashish Maske," Iot-Based Battery Parameter Monitoring System For Electric Vehicle "© 2020 IJCRT | Volume 8, Issue 7 July 2020 | ISSN: 2320-2882

PROJECT PROFORMA

Classification of Project	Application	Product	Research	Review

Note: Tick Appropriate category:

Project Outcomes	
Outcome 1	Use new tools.
Outcome 2	Work as an individual and in a team
Outcome 3	Analyze critically.
Outcome 4	Identify and solve problems.

Mapping Table:

Project Outcomes	Programme Outcomes (POs)												PSOs	
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
Outcome 1														
Outcome 2														
Outcome 3														
Outcome 4														

Note: Map each project outcomes with POs and PSOs with either 1 or 2 or 3 based on level of mapping as follows:

1-Slightly (Low) mapped 2-Moderately (Medium) mapped 3-Substantially (High) mapped

Programme Outcomes:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization for the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of t h e information to provide valid conclusions.

5.**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and Modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations.

6.**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8**. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognizes the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Programme Specific Outcomes(PSO):

1. Apply the knowledge of circuit design, analogy& digital electronics to the field of electrical and electronics systems

2. Analyze, design and develop control systems, industrial drives and power systems using modern tools.

Student Details:

Name:G.V.Durga Prasad
Roll No:22485A0212
Phone: 7032407249
Mail: prasadgudavalli15@gmail.com

Name:B.Renuka Lakshmi Bai
Roll No:21481A0217
Phone:9502099684
Mail: bondilirenuka07@gmail.com

Name:K.Hema Tanuja
Roll No:21481A0247
Phone:8328164557
Mail: kesanatanuja@gmail.com

Name:B.Vivek Kanaka Roy
Roll No:21481A0214
Phone:7981637419
Mail: bheemalavivekkanakaroy@gmail.com