

```
rulecolor=, text (e.g.  commens (green here))  tabsize=2, captionpos=b,  
breaklines=true, breakatwhitespace=false, title=, keywordstyle=, commentstyle=,  
stringstyle=, escapeinside=%**), morekeywords=*,...
```

READY TO DRIVE SAFETY SENSING SYSTEM FOR ELECTRIC VEHICLE USING IOT

*Major project report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

BALGAM NITHIN	(20UECS0114)	(17736)
BUSA ROHAN RAJ	(20UECS0105)	(17727)
POKURI ANIRUDH REDDY	(20UECS1073)	(21646)

*Under the guidance of
J.SWAPNA,M.E.,
ASSISTANT PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE & TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA**

May, 2024

READY TO DRIVE SAFETY SENSING SYSTEM FOR ELECTRIC VEHICLE USING IOT

*Major project report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

BALGAM NITHIN	(20UECS0114)	(17736)
BUSA ROHAN RAJ	(20UECS0105)	(17727)
POKURI ANIRUDH REDDY	(20UECS1073)	(21646)

*Under the guidance of
J.SWAPNAE, M.E.,
ASSISTANT PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING**

SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

SCIENCE & TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA**

May, 2024

CERTIFICATE

It is certified that the work contained in the project report titled "READY TO DRIVE SAFETY SENSING SYSTEM FOR ELECTRIC VEHICLE USING IOT" by BALGAM NITHIN (20UECS0114), BUSA ROHAN RAJ (20UECS0105), POKURI ANIRUDH REDDY (20UECS1073) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
May, 2024

Signature of Professor In-charge
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
May, 2024

DECLARATION

We declare that this written submission represents ideas in own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

BALGAM NITHIN

Date: / /

(Signature)

BUSA ROHAN RAJ

Date: / /

(Signature)

POKURI ANIRUDH REDDY

Date: / /

APPROVAL SHEET

This project report entitled READY TO DRIVE SAFETY SENSING SYSTEM FOR ELECTRIC VEHICLE USING IOT by BALGAM NITHIN (20UECS0114), BUSA ROHAN RAJ (20UECS0105), POKURI ANIRUDH REDDY (20UECS1073) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners

Supervisor

J.SWAPNA,M.E.,ASSISTANT PROFESSOR,.

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO), D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN**, for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.**, for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Computer Science & Engineering, Dr. M.S. MURALI DHAR, M.E., Ph.D.**, for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our Internal Supervisor **J. SWAPNA, M.E.**, for her cordial support, valuable information and guidance, she helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M.Tech., Ms. C. SHYAMALA KUMARI, M.E.**, for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

BALGAM NITHIN	(20UECS0114)
BUSA ROHAN RAJ	(20UECS0105)
POKURI ANIRUDH REDDY	(20UECS1073)

ABSTRACT

The integration of Internet of Things (IoT) technology in electric vehicles (EVs) has paved the way for enhanced safety features and driving experience. Project focuses on the development of a ready-to-drive safety sensing electric vehicle utilizing IoT advancements. The core objective is to design a comprehensive system that integrates various IoT sensors and devices to monitor and improve the safety aspects of EVs. Key components of the system include sensors for monitoring environmental conditions, such as temperature, humidity, and air quality, to ensure optimal driving conditions for passengers and the vehicle itself. Furthermore, the integration of IoT technology enables remote monitoring and control of the EV's vital parameters, including battery status, charging levels, and overall performance. This not only enhances the convenience for users but also contributes to the longevity and efficiency of the vehicle. The cornerstone of IoTGuard is its cloud-based analytics platform, which processes the vast amount of sensor data collected from the EV in real-time. Utilizing machine learning algorithms, the platform identifies patterns indicative of impending safety risks, enabling proactive intervention to prevent accidents.

Keywords: EV Electric vehicle, IOT internet of things, Temperature Humidity Sensor, Battery status Charging Levels, Voltage Sensor Current Sensor

LIST OF FIGURES

4.1	Architecture Diagram	9
4.2	Data Flow Diagram	10
4.3	Use case Diagram	11
4.4	Class Diagram	12
4.5	Sequence Diagram	13
4.6	Collaboration Diagram	14
4.7	Activity Diagram	15
5.1	Input Design	19
5.2	Out Design	20
5.3	Unit Testing code	21
5.4	Integration code	22
5.5	Test Result	24
6.1	Output 1	28
6.2	Output 2	29
8.1	Balgam Nithin	33
8.2	Busa Rohan Raj	34
8.3	Pokuri Anirudh Reddy	35
9.1	Plagiarism Report	37
10.1	Poster	39

LIST OF ACRONYMS AND ABBREVIATIONS

BMS	Battery Management System
COAP	Common offers acceptance portal
GSM	Global communication for mobile system
HTTP	Hyper text transfer protocol
IOT	Internet of Things
MQTT	Message queuing Telemetry Transport

TABLE OF CONTENTS

	Page.No
ABSTRACT	v
LIST OF FIGURES	vi
LIST OF ACRONYMS AND ABBREVIATIONS	vii
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Aim of the Project	1
1.3 Project Domain	2
1.4 Scope of the Project	2
2 LITERATURE REVIEW	3
3 PROJECT DESCRIPTION	4
3.1 Existing System	4
3.2 Proposed System	5
3.3 Feasibility Study	5
3.3.1 Economic Feasibility	6
3.3.2 Technical Feasibility	6
3.3.3 Social Feasibility	6
3.4 System Specification	6
3.4.1 Hardware Specification	7
3.4.2 Software Specification	7
3.4.3 Standards and Policies	8
4 METHODOLOGY	9
4.1 General Architecture	9
4.2 Design Phase	10
4.2.1 Data Flow Diagram	10
4.2.2 Use Case Diagram	11
4.2.3 Class Diagram	12

4.2.4	Sequence Diagram	13
4.2.5	Collaboration Diagram	14
4.2.6	Activity Diagram	15
4.3	Algorithm & Pseudo Code	15
4.3.1	Machine Learning	15
4.3.2	Pseudo Code	15
4.4	Module Description	16
4.4.1	System Architecture	16
4.4.2	Hardware Selection	16
4.4.3	Backend Development	16
4.5	Steps to execute/run/implement the project	17
4.5.1	Hardware Setup	17
4.5.2	Android Iot Cloud	17
4.5.3	Integration	18
5	IMPLEMENTATION AND TESTING	19
5.1	Input and Output	19
5.1.1	Input Design	19
5.1.2	Output Design	20
5.2	Testing	20
5.3	Types of Testing	20
5.3.1	Unit Testing	20
5.3.2	Integration Testing	21
5.3.3	System Testing	22
5.3.4	Test Result	24
6	RESULTS AND DISCUSSIONS	25
6.1	Efficiency of the Proposed System	25
6.2	Comparison of Existing and Proposed System	26
6.3	Sample Code	26
7	CONCLUSION AND FUTURE ENHANCEMENTS	30
7.1	Conclusion	30
7.2	Future Enhancements	30

8	INDUSTRY DETAILS	32
8.1	Industry Name:VOLT ME MOTORS PVT.LMT	33
8.1.1	Duration of Internship (JAN8- MAY9)	33
8.1.2	Duration of Internship:5 months	33
8.1.3	Industry Address:4th floor, 7-13, Sri lakshmi nilayam, near midland bakery madinaguda, miyapur, hyderabad - 500050 .	33
8.2	Internship offer Letter	33
8.3	Internship Completion Certificate	36
9	PLAGIARISM REPORT	37
10	SOURCE CODE & POSTER PRESENTATION	38
10.1	Source Code	38
10.2	Poster Presentation	39
	References	39

Chapter 1

INTRODUCTION

1.1 Introduction

In response to this imperative, the project "Ready-to-Safety Sensing System for Electric Vehicle using IoT" aims to leverage the power of Internet of Things (IoT) technology to enhance the safety features and driving experience of electric vehicles. With the proliferation of IoT-enabled devices and sensors, there exists a unique opportunity to create a comprehensive safety sensing system that monitors and responds to potential hazards in real-time. The primary objective of this project is to develop a sophisticated safety sensing system that integrates seamlessly with electric vehicles, providing drivers with enhanced situational awareness and proactive assistance in navigating challenging road conditions. By leveraging IoT sensors deployed both within and around the vehicle, the system will continuously monitor environmental factors such as temperature, humidity, air quality, and road conditions, as well as detect potential obstacles, pedestrians, and other vehicles.

1.2 Aim of the Project

The aim of the project is to develop a comprehensive and integrated system that leverages IoT (Internet of Things) technology to enhance the safety aspects of electric vehicles (EVs) and provide a seamless driving experience. The primary objectives of the project include:

- Real-time Monitoring:** Implementing IoT sensors and devices to continuously monitor various parameters both inside and outside the electric vehicle, such as temperature, humidity, air quality, road conditions, and vehicle performance metrics.
- Remote Monitoring and Control:** Enabling remote monitoring and control of the electric vehicle's vital parameters, such as battery status, charging levels, and vehicle diagnostics, through IoT connectivity. This feature enhances convenience for the user and facilitates proactive maintenance and troubleshooting.

1.3 Project Domain

Internet of Things (IoT): The core concept revolves around connecting devices (such as batteries) to the internet to gather data remotely

Embedded Systems: ESP8266 is a microcontroller with built-in Wi-Fi capability, so understanding embedded systems is essential for programming and interfacing with it

Wireless Communication: Understanding wireless communication protocols such as Wi-Fi, MQTT, or HTTP is crucial for transmitting data from the ESP8266 to the cloud or a local server.

Battery Technology: A basic understanding of battery chemistry, charging, and discharging characteristics is necessary to interpret data accurately and ensure the longevity of the batteries being monitored.

Data Analytics and Visualization: Processing the data collected from the batteries and presenting it in a meaningful way requires skills in data analytics and visualization

Cloud Computing: If you're planning to store data in the cloud, knowledge of cloud platforms like AWS, Azure, or Google Cloud. **Security:** Ensuring the security of the data being transmitted and stored is essential in any IoT system. This involves implementing encryption, access control, and secure communication Protocols.

1.4 Scope of the Project

IoT Sensor Integration: Selection and integration of appropriate IoT sensors to monitor environmental conditions both inside and outside the vehicle, including temperature, humidity, air quality, and road conditions.

Connectivity and Data Management: Implementation of IoT connectivity solutions to facilitate real-time data transmission between sensors, onboard systems, and external servers or cloud platforms.

Remote Monitoring and Control: Implementation of remote monitoring and control capabilities for monitoring vital parameters of the EV, including battery status, charging levels, and vehicle performance. Development of user interfaces or mobile applications to enable users to remotely access and control various aspects of the vehicle, enhancing convenience and efficiency

Chapter 2

LITERATURE REVIEW

- [1] Vermesan, O., Friess, P., Internet of things: global technological and societal trends, Aalnorg: River Publishers, 2011.
- [2]Zanella, A., Bui, N., Castellani, A., Vangelista, L. and Zorzi, M., Internet of Things for Smart Cities, IEEE Internet of Things Journal, 1(1), 2014.
- [3] C. Perera, C. H. Liu and S. Jayawardena, The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey, IEEE Transactions on Emerging Topics in Computing, vol. 3, no. 4, Dec. 2015
- [4]Schwartz, M., Internet of things with ESP8266: build amazing internet of things projects using the ESP8266 Wi-Fi chip, ISBN 978-1-78646- 802-4, Birmingham Mumbai: Packt, 2016
- [5]Mehmood, N. Q., Culmone, R. and Mostarda, L., Modeling temporal aspects of sensor data for MongoDB NoSQL database, Journal of Big Data, 4(1), 2017
- [6]Kao B., Garcia-Molina H., An Overview of RealTime Database Systems. In: Halang W.A., Stoyenko A.D. (eds) Real Time Computing. NATO ASI Series (Series F: Computer and Systems Sciences), vol 127, 1994, Springer, Berlin, Heidelberg
- [7]R. Abbott, H. Garcia-Molina, What is a RealTime Database System?, Abstracts of the Fourth Workshop on Real-Time Operating systems, IEEE, July 1987.
- [8]Hazra, Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems, Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019.
- [9]Bajrami, X. and Murturi, I., An efficient approach to monitoring environmental conditions using a wireless sensor network and NodeMCU, e i Elektrotechnik und Informationstechnik, 135(3), 2018.
- [10]Jaffe, S. R., Design of Inexpensive and Easy ToUse DIY Internet of Things Platform, California Polytechnic State University, 2016.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

The existing IoT-based battery monitoring systems typically consist of hardware components like battery sensors, microcontrollers (such as Arduino or Raspberry Pi), communication modules (like Wi-Fi, Bluetooth, or GSM), and software components such as data processing algorithms and user interfaces. These systems are designed to remotely monitor parameters like battery voltage, temperature, and current, providing real-time data to users or central monitoring stations. They often include features like predictive maintenance alerts, historical data analysis, and customizable notifications for battery health and performance management

Disadvantages of existing system

1. **Complexity and Cost:** Implementing an IoT-based data acquisition system requires significant upfront investment in hardware, software, and infrastructure.
2. **Data Security Risks:** Transmitting sensitive vehicle data over wireless networks introduces security vulnerabilities, making the system susceptible to cyber-attacks, data breaches, and unauthorized access.
3. **Reliability and Connectivity Issues:** Dependence on wireless communication for data transmission can lead to reliability and connectivity issues, especially in remote or low-coverage areas.
4. **Power Consumption:** Continuous operation of sensors, microcontrollers, and wireless modules consumes power, draining the vehicle's battery and reducing its driving range.
5. **Compatibility and Standards:** Integrating third-party devices or upgrading existing systems may require additional effort and customization

3.2 Proposed System

An IoT-based battery monitoring system typically involves sensors placed on batteries to collect data on various parameters like voltage, temperature, and current. This data is then transmitted wirelessly to a central server or cloud platform for analysis and storage. Users can access this data through a web or mobile application to monitor battery health, receive alerts for maintenance or replacement, and optimize battery usage. The system may also include predictive analytics to anticipate battery failures or degradation, allowing for proactive maintenance. Overall, it offers real-time insights and remote monitoring capabilities for efficient battery management.

Advantages of proposed system

1. **Optimize Energy Management:** To improve the efficiency of battery usage and energy consumption in electric vehicles, thereby extending the driving range and reducing charging times. This involves monitoring battery health, state of charge (SOC), state of health (SOH), and optimizing energy distribution based on driving patterns and conditions.
2. **Enhance Vehicle Performance:** To monitor and analyze data related to the vehicle's operational parameters, such as motor temperature, speed, and torque, in order to optimize performance, reduce wear and tear, and prevent potential failures.
3. **Improve User Experience:** To provide EV owners with real-time information and insights into their vehicle's performance, energy usage, and health status. This includes personalized recommendations for energy-saving driving habits, navigation to nearby charging stations, and remote control of vehicle functions.

3.3 Feasibility Study

Market feasibility: This examines the demand for such a system in the market and whether potential users would be willing to adopt it. Market research is conducted to understand the needs of target customers, identify competitors, and assess potential revenue streams. Factors such as industry trends, regulatory requirements, and potential partnerships also play a role in determining market feasibility. **Market feasibility:** This examines the demand for such a system in the market and whether potential users would be willing to adopt it. Market research is conducted to understand the needs of target customers, identify competitors, and assess potential revenue streams. Factors such as industry trends, regulatory requirements, and potential

partnerships also play a role in determining market feasibility.

3.3.1 Economic Feasibility

Financial feasibility: This involves estimating the costs associated with developing and implementing the project compared to the potential benefits and returns. It includes an analysis of initial investment requirements, ongoing operational costs, potential revenue generation, and return on investment (ROI). Financial projections and risk assessments are used to determine whether the project is financially viable.

3.3.2 Technical Feasibility

Technical feasibility: This assesses whether the technology required for the project is available, feasible, and scalable. It involves evaluating the compatibility of various IoT sensors, communication protocols, and cloud platforms for data storage and analysis. Additionally, considerations regarding power consumption, sensor accuracy, and data transmission reliability are important.

3.3.3 Social Feasibility

Legal and regulatory feasibility: This examines the legal and regulatory requirements that may impact the development and deployment of the project. It involves identifying relevant regulations related to data privacy, security, and compliance, as well as any intellectual property considerations. Legal and regulatory compliance are essential for ensuring the project's success and avoiding potential liabilities. By conducting a comprehensive feasibility study covering these aspects, stakeholders can make informed decisions about whether to proceed with the IoT-based battery monitoring system project and how to mitigate potential risks and challenges.

3.4 System Specification

Sensor Integration: Incorporate sensors for monitoring key parameters such as voltage, current, temperature, and state of charge (SoC) of the batteries.

Wireless Connectivity: Utilize wireless communication protocols such as Wi-Fi, Bluetooth, or LoRaWAN for transmitting data from the sensors to the central server or cloud platform.

Cloud Platform: Implement a cloud-based platform for data storage, processing, and analysis. Ensure scalability, reliability, and security of the cloud infrastructure.

User Interface: Develop a user-friendly web or mobile application for accessing battery data, viewing real-time metrics, and receiving alerts and notifications.

Data Analytics: Incorporate data analytics capabilities for trend analysis, predictive maintenance, and optimization of battery performance and lifespan. Alerts and Notifications. Set up automated alerts and notifications to notify users of critical battery conditions, such as low voltage, high temperature, or impending failure.

Remote Monitoring: Enable remote monitoring and control of batteries, allowing users to access data and perform maintenance tasks from anywhere with an internet connection

3.4.1 Hardware Specification

RAM:4GB and Higher

Hard Disk:500gb Minimum

Processor:Intel i3 and above

NodeMCU ESP8266 Development Board

TP4056 Charging Module

18650 Lithium Ion Battery

Jumper Cables

100k Resistor

3.4.2 Software Specification

Programming Language /Platform : JAVA

IDE : Android Iot Cloud /Thinkspeak

ThinkSpeak, an open-source Internet of Things (IoT) platform developed by MathWorks, can help in various ways: 6 DRAFT Data Visualization: It allows you to visualize your IoT data in real-time using customizable charts and graphs, making it easier to understand and analyze.

Data Analysis: With built-in analytics tools, you can perform data analysis and gain insights into trends, patterns, and anomalies in your data

3.4.3 Standards and Policies

Data security and privacy policies: Implement encryption protocols for data transmission and storage to comply with industry standards such as AES (Advanced Encryption Standard). Adhere to data privacy regulations like GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act) to protect user information

Device communication protocols: Follow established IoT communication protocols like MQTT (Message Queuing Telemetry Transport) or CoAP (Constrained Application Protocol) to ensure interoperability and efficient data transmission between devices and the central monitoring system

Battery safety Standards: Adhere to battery safety standards such as IEC 62133 for lithium-ion batteries or relevant standards for other battery chemistries to ensure safe handling and operation.

Chapter 4

METHODOLOGY

4.1 General Architecture

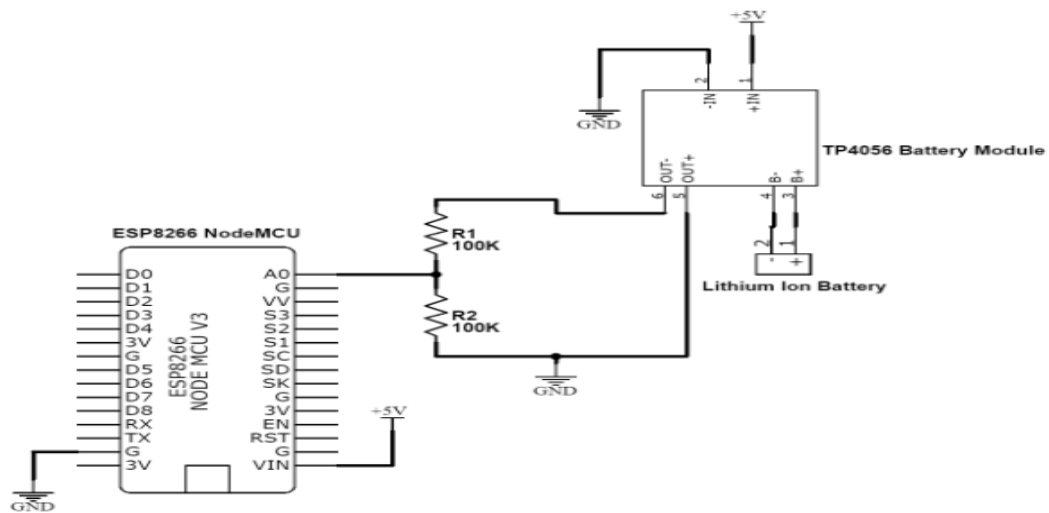


Figure 4.1: Architecture Diagram

Figure 4.1 is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation

4.2 Design Phase

4.2.1 Data Flow Diagram

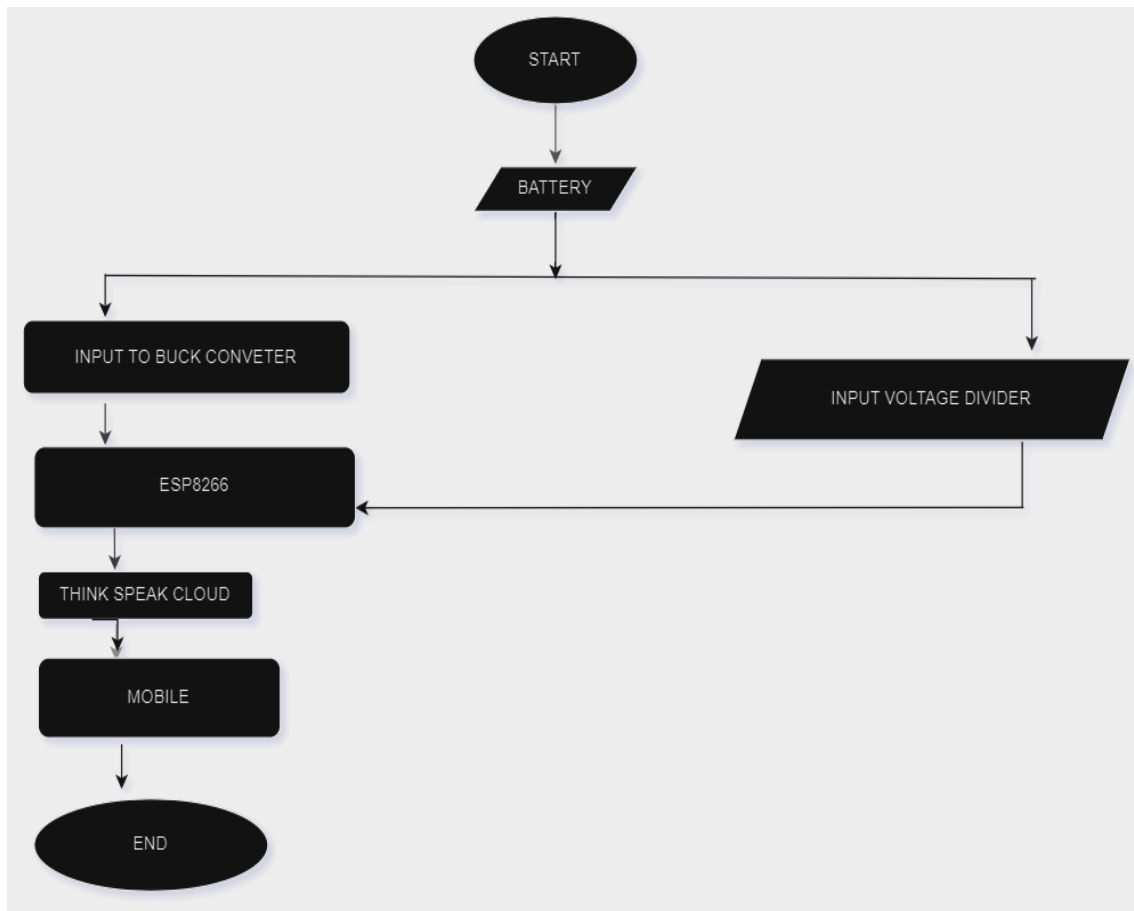


Figure 4.2: **Data Flow Diagram**

Figure 4.2 The DFD is also called as bubble chart. As shown in the figure It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional

details.

4.2.2 Use Case Diagram

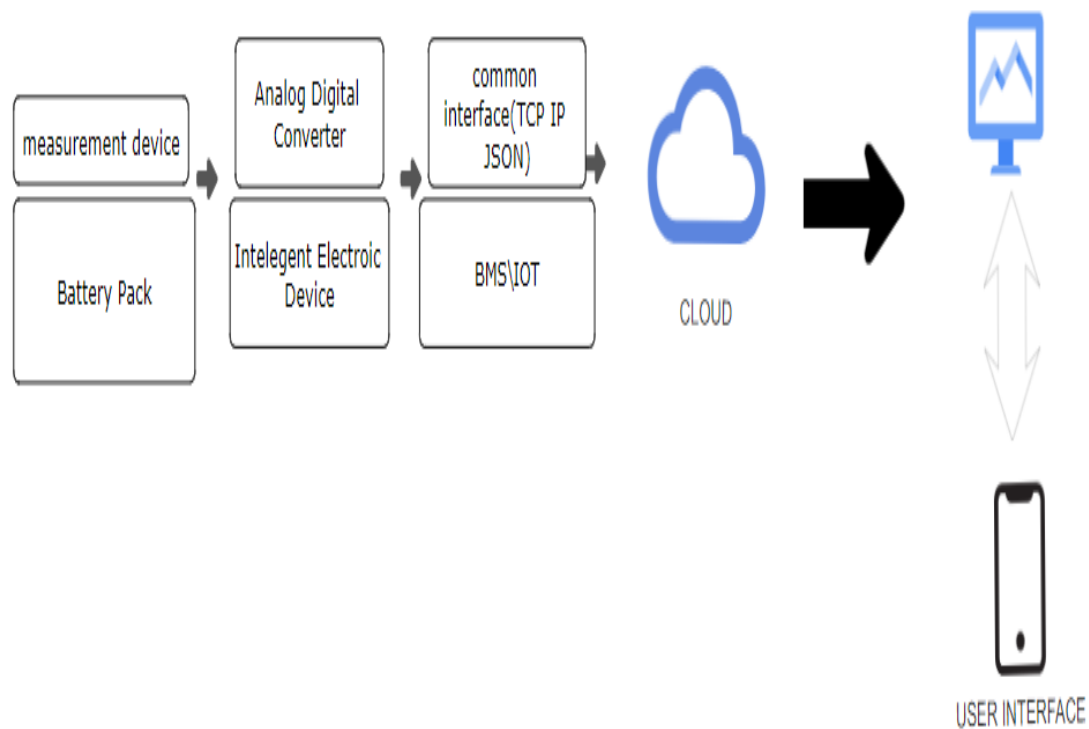


Figure 4.3: Use case Diagram

Figure4.3 is a use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a sys tem in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted

4.2.3 Class Diagram

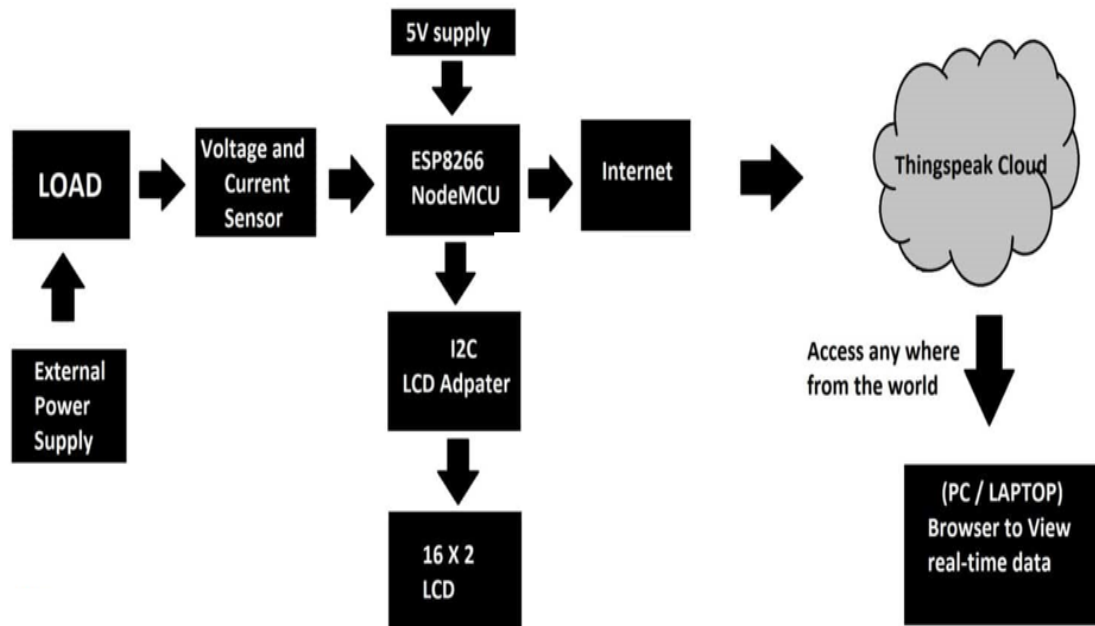


Figure 4.4: Class Diagram

Figure 4.4 A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are usefull in many stages of system design as show in figure4.4.

4.2.4 Sequence Diagram

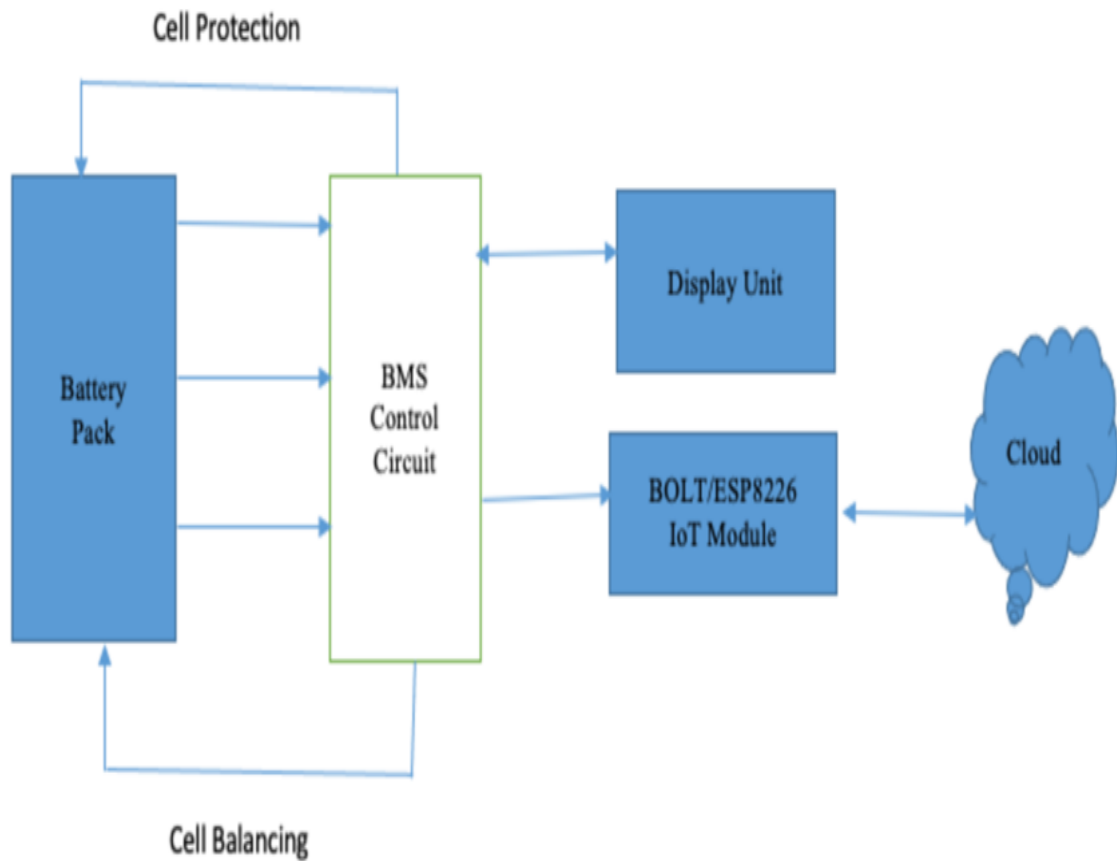


Figure 4.5: Sequence Diagram

Figure 4.5 Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling

4.2.5 Collaboration Diagram

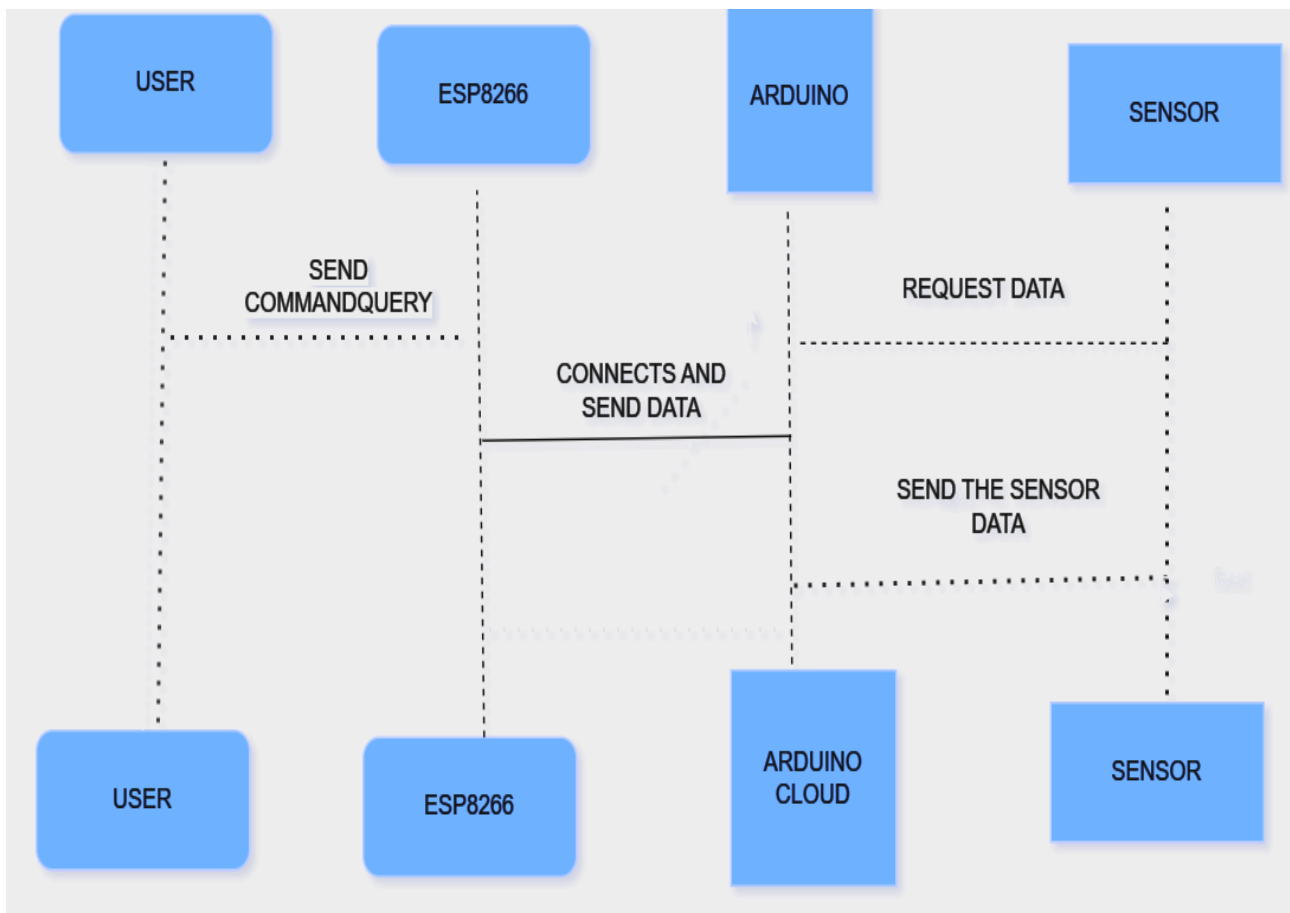


Figure 4.6: Collaboration Diagram

Figure 4.6 resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown arrows connecting the relevant rectangles along with labels that define the message sequencing

4.2.6 Activity Diagram

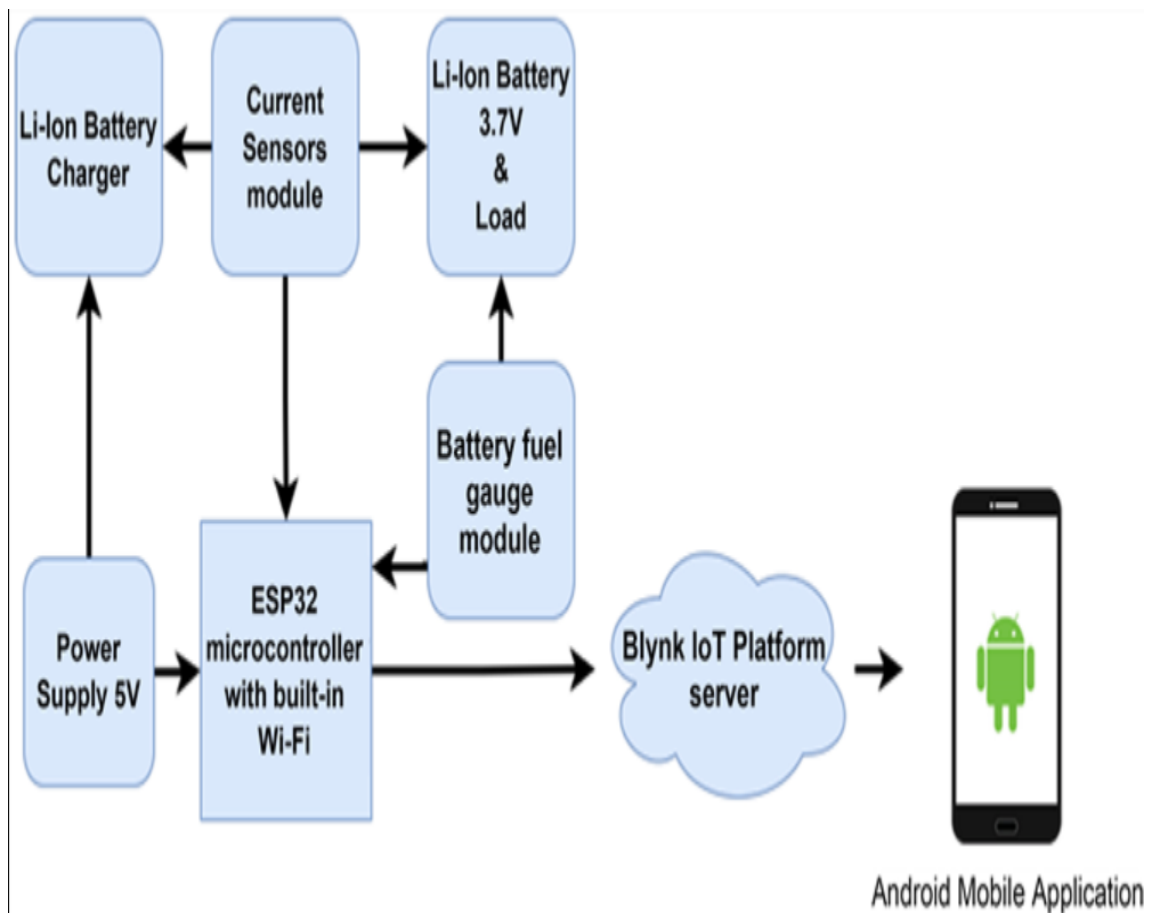


Figure 4.7: Activity Diagram

4.3 Algorithm & Pseudo Code

4.3.1 Machine Learning

In the Ready to Drive Safety Sensing System Electric Vehicle using IOT all battery-relevant data such as voltage, current, temperature during both charging and discharging is first transmitted in real-time to the cloud, where the system uses algorithms based on machine learning and artificial Intelligence to evaluate the data.

4.3.2 Pseudo Code

```
include <ESP8266WiFi.h> String apiKey = "jioFiber";  
const char* ssid = "Desktop windows dell";  
const char* pass = "Rohan1234";  
const char* server = "api.thingspeak.com";
```

```
int analogInPin = A0;
int sensorValue;
float calibration = 0.36;
int batpercentage;
WiFiClient client;
void setup()
Serial.begin(115200);
Serial.println("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, pass);
```

4.4 Module Description

4.4.1 System Architecture

System Architecture: Design the overall architecture of the system including hardware are components (ESP8266, sensors, battery interface), software components (firmware, backend server, database), and communication protocols (Wi-Fi, MQTT, HTTP)

Requirments Analysis: Understand the requirements of the battery monitoring system such as the types of batteries to monitor, communication protocols.

4.4.2 Hardware Selection

Hardware Selection and Integration: Choose suitable sensors for measuring battery parameters like voltage, current, temperature, etc. Integrate these sensors with ESP8266 and Arduino IoT board.

Firmware Development:Develop firmware for ESP8266 and Arduino IoT to read data from sensors, process it, and send it to the backend server using Wi-Fi or other communication protocols.

4.4.3 Backend Developnment

Backend Developnment: Set up a backend server to receive data from the IoT devices, store it in a database, and provide APIs for accessing the data. You can

use platforms like AWS IoT, Google Cloud IoT, or self-hosted solutions like MQTT broker and database server

Data Visualization: Create a user interface for visualizing battery parameters such as voltage, current, temperature, and state of charge. This can be a web dashboard, mobile app, or desktop application.

4.5 Steps to execute/run/implement the project

4.5.1 Hardware Setup

Hardware Setup: Acquire ESP8266 modules, battery sensors (such as voltage, current, and temperature sensors), and any necessary supporting components (resistors, capacitors, etc.). Connect the sensors to the ESP8266 modules according to the hardware specifications and wiring diagrams provided by the sensor manufactures.

Firmware Development for ESP8266: Develop firmware for the ESP8266 modules to read data from the connected sensors. Implement communication protocols (such as MQTT or HTTP) to transmit sensor data to the Android IoT cloud. Include error handling mechanisms and retry strategies to ensure robust data transmission, especially in the case of network disruptions.

4.5.2 Android Iot Cloud

Android IoT Cloud Setup: Sign up for an account on the Android IoT cloud platform (such as Google Cloud IoT or AWS IoT). Set up a new project and create a registry to manage the ESP8266 devices. Generate device credentials (such as device ID and authentication tokens) to securely connect the ESP8266 devices to the cloud.

Android IoT App Development: Develop an Android IoT app to visualize battery data and receive alerts from the cloud. Implement features to authenticate users and securely access data from the Android IoT cloud. Design an intuitive user interface to display real-time battery status, historical data, and configurable settings.

Firmware Development for ESP8266: Develop firmware for the ESP8266 modules to read data from the connected sensors. Implement communication protocols (such as MQTT or HTTP) to transmit sensor data to the Android IoT cloud. Include error handling mechanisms and retry strategies to ensure robust data transmission, especially in the case of network disruptions.

4.5.3 Integration

Integration and Testing Integrate the ESP8266 firmware with the Android IoT app to establish end-to-end communication between the devices and the cloud. Conduct thorough testing to ensure that sensor data is accurately transmitted from the ESP8266 modules to the Android IoT cloud and displayed correctly on the Android app. Test various scenarios, including normal operation, network disruptions, and edge cases, to validate the reliability and robustness of the system.

Deployment and Monitoring: Deploy the IoT-based battery monitoring system in the target environment, ensuring that all hardware components are securely installed and configured. Monitor the system's performance and stability in real-world conditions, addressing any issues or anomalies that arise during operation. Implement mechanisms for remote monitoring and management, allowing administrators to track battery status and troubleshoot issues from anywhere.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

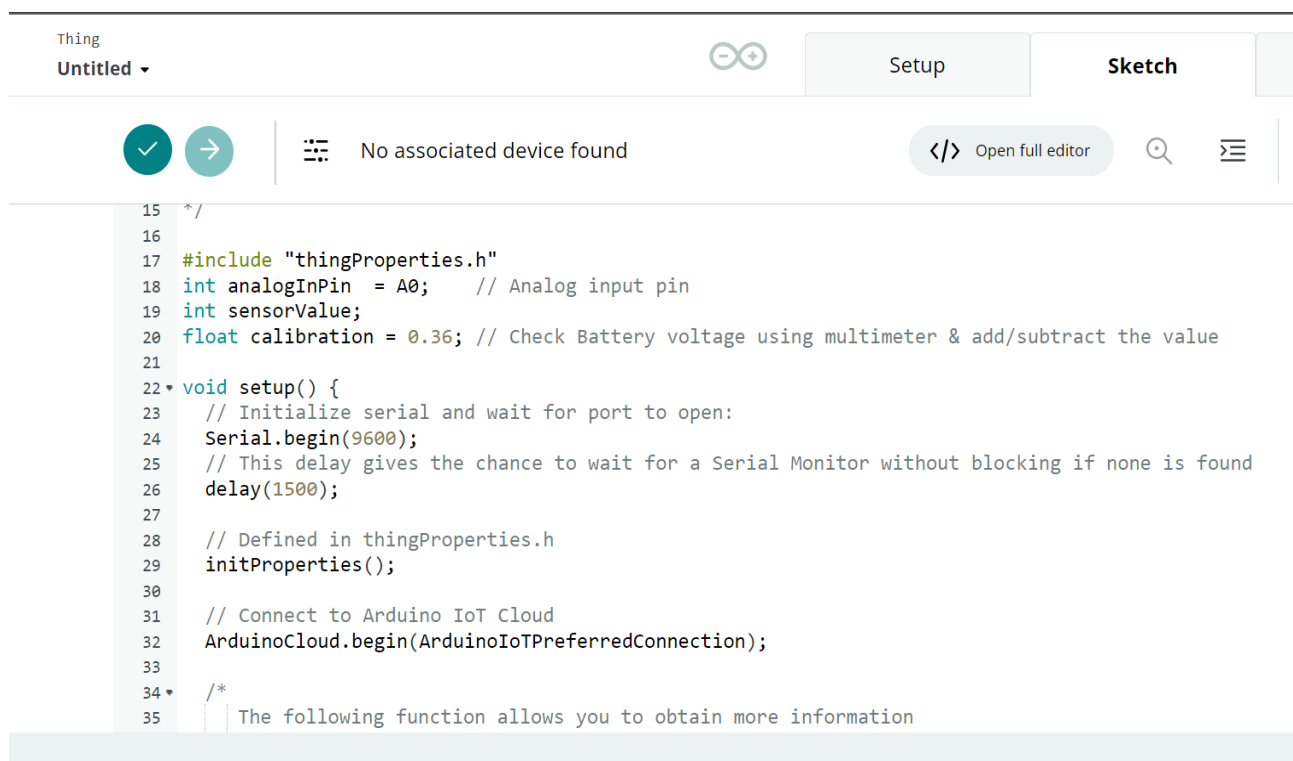


Figure 5.1: Input Design

5.1.2 Output Design



Figure 5.2: Out Design

5.2 Testing

5.3 Types of Testing

5.3.1 Unit Testing

Identify the code and Break down your code into smaller units or components that can be tested independently.

Write Test Cases: Develop test cases for each unit to verify its functionality. Test cases should cover both normal and edge cases to ensure robustness. For the battery monitoring system, you might have test cases to simulate different battery levels, network connectivity scenarios, error handling, etc.

Setup Testing Environment: Set up a testing environment where you can run your unit tests. This might involve creating mock objects or simulators for hardware components such as sensors or the ESP8266.

Select a Testing Framework: Choose a testing framework suitable for your platform. For ESP8266 code, you might use the Arduino Unit Testing framework or

another testing library compatible with the Arduino IDE. For Android, you can use JUnit for unit testing Java code and Espresso for UI testing.

```
39  // Maximum is 4
40  */
41  setDebugLogLevel(2);
42  ArduinoCloud.printDebugInfo();
43  }
44
45  void loop() {
46    ArduinoCloud.update();
47    sensorValue = analogRead(analogInPin);
48    voltage = (((sensorValue * 3.3) / 1024) * 2 + calibration);
49
50    bat_percentage = mapfloat(voltage, 2.8, 4.2, 0, 100); //2.8V
51
52    if (bat_percentage >= 100)
53    {
54        bat_percentage = 100;
55    }
56    if (bat_percentage <= 0)
57    {
```

Figure 5.3: Unit Testing code

5.3.2 Integration Testing

Understand the System Architecture: Before starting integration testing, ensure you have a clear understanding of the overall architecture of your battery monitoring system. Identify the components involved, such as the ESP8266 module, sensors, microcontroller, communication protocols (like MQTT or HTTP), cloud services, and any other relevant hardware or software.

Define Integration Test Scenarios: Based on the system architecture, define integration test scenarios that cover the interactions between different components. This may include scenarios such as sensor data acquisition, data transmission to the cloud, receiving commands from the cloud, handling network disruptions, etc.

Mock External Dependencies: In integration testing, it's essential to isolate the components being tested from external dependencies that are not directly related to

the integration under test. For example, you might mock cloud services or simulate sensor data to mimic real-world behavior without relying on actual external systems.

Set Up Test Environment: Create a test environment that closely resembles the production environment but allows for controlled testing. This may involve setting up physical hardware (ESP8266 module, sensors, etc.) and virtual environments for cloud services or other external dependencies

```
Serial.print("Analog Value = ");  
Serial.print(sensorValue);  
Serial.print("\t Output Voltage = ");  
Serial.print(voltage);  
Serial.print("\t Battery Percentage = ");  
Serial.println(bat_percentage);  
delay(1000);
```

```
float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)  
  
return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
```

Figure 5.4: **Integration code**

5.3.3 System Testing

Define Test Cases: Start by defining test cases based on the system requirements and use cases. Test cases should cover various aspects of the system, including sensor data acquisition, communication between the ESP8266 module and the Android IoT device, user interaction on the Android app, data visualization, and any other relevant functionalities.

Set Up Test Environment: Prepare the test environment, including the hardware components (ESP8266 module, sensors, batteries, etc.), software components (Android IoT app, ESP8266 firmware, cloud services, etc.), and any simulated or emulated environments necessary for testing.

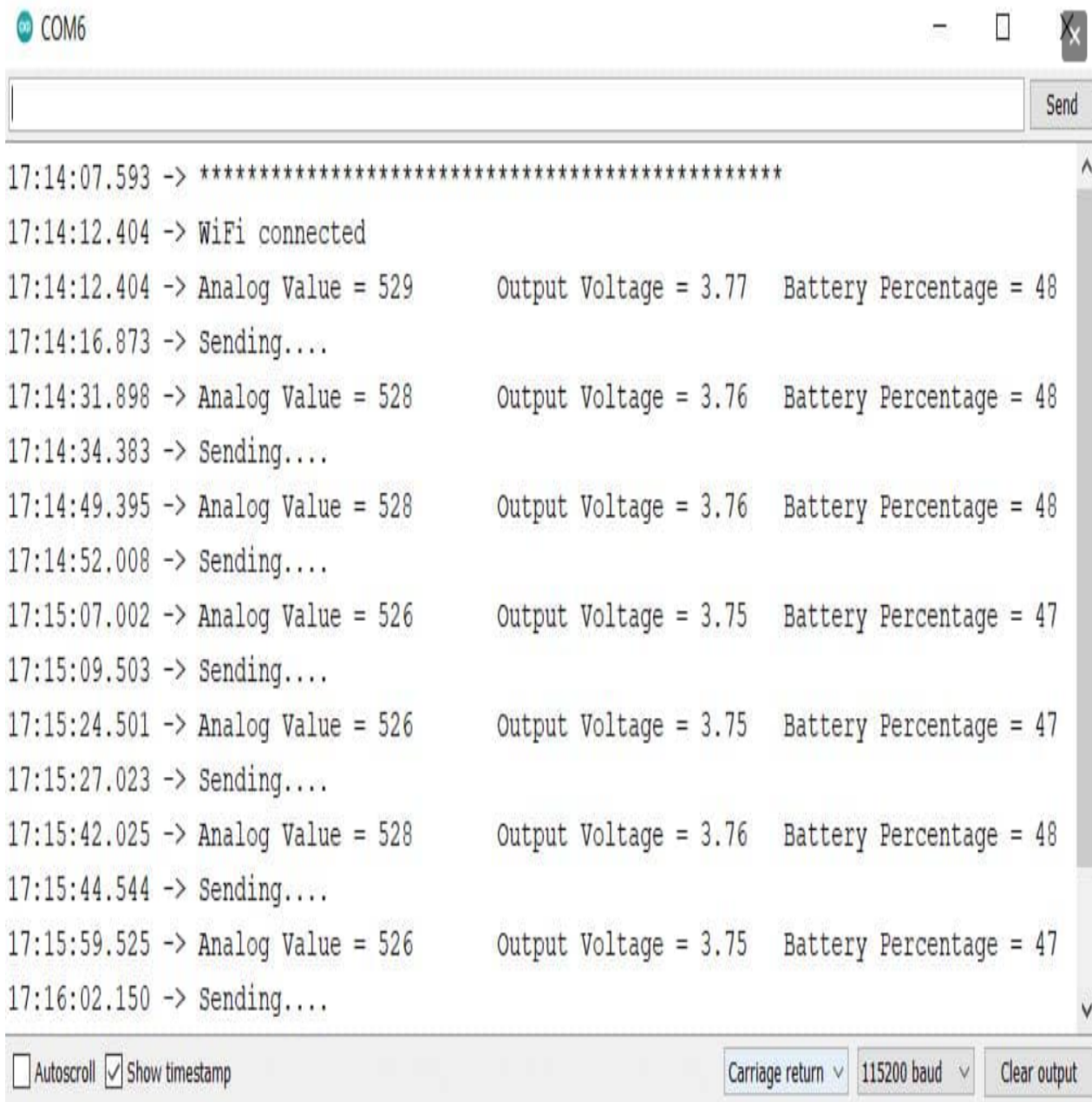
Perform Functional Testing: Execute the defined test cases to verify that each function of the system works as expected. This includes testing sensor readings, data transmission from ESP8266 to the Android app, processing of data on the app, and displaying relevant information to the user.

Test Communication: Verify the reliability and security of communication between the ESP8266 module and the Android IoT device. Test scenarios should include normal operation, as well as edge cases such as network disruptions, packet loss, and reconnection mechanisms.

Test Result

Android IoT cloud is used to display the Battery Percentage and current voltage of battery. Which is useful to identify the battery of an electric using Remote by mobile phone desktop.

5.3.4 Test Result



The screenshot shows a terminal window titled 'COM6' with a 'Send' button in the top right. The terminal displays a series of log messages with timestamps and data. The messages are as follows:

```
17:14:07.593 -> *****
17:14:12.404 -> WiFi connected
17:14:12.404 -> Analog Value = 529      Output Voltage = 3.77   Battery Percentage = 48
17:14:16.873 -> Sending....
17:14:31.898 -> Analog Value = 528      Output Voltage = 3.76   Battery Percentage = 48
17:14:34.383 -> Sending....
17:14:49.395 -> Analog Value = 528      Output Voltage = 3.76   Battery Percentage = 48
17:14:52.008 -> Sending....
17:15:07.002 -> Analog Value = 526      Output Voltage = 3.75   Battery Percentage = 47
17:15:09.503 -> Sending....
17:15:24.501 -> Analog Value = 526      Output Voltage = 3.75   Battery Percentage = 47
17:15:27.023 -> Sending....
17:15:42.025 -> Analog Value = 528      Output Voltage = 3.76   Battery Percentage = 48
17:15:44.544 -> Sending....
17:15:59.525 -> Analog Value = 526      Output Voltage = 3.75   Battery Percentage = 47
17:16:02.150 -> Sending....
```

At the bottom of the window, there are controls for 'Autoscroll' (unchecked), 'Show timestamp' (checked), a 'Carriage return' dropdown menu, a '115200 baud' dropdown menu, and a 'Clear output' button.

Figure 5.5: Test Result

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

Data Accuracy and Timeliness: The proposed system should accurately monitor and report battery parameters such as voltage, current, temperature, and state of charge. The data should be updated in real-time or at frequent intervals to provide timely information to users.

Energy Efficiency: Since the system is deployed in IoT environments, energy efficiency is crucial to prolong the battery life of devices like ESP8266 and Android smartphones. The system should optimize data transmission protocols, minimize standby power consumption, and employ sleep modes when idle to conserve energy.

Reliability and Availability: The system should be reliable, ensuring continuous monitoring of batteries without significant downtime. It should handle network disruptions gracefully and implement mechanisms for data buffering and synchronization to prevent data loss.

Scalability: The system should be scalable to accommodate a growing number of monitored batteries and users. It should be capable of handling increased data traffic and user interactions without sacrificing performance or reliability.

User Experience: The Android IoT app interface should be intuitive and user-friendly, allowing users to easily configure monitoring settings, view battery status, and receive alerts or notifications. A seamless user experience enhances the adoption and satisfaction with the system.

Security: The system should prioritize security to protect sensitive battery data from unauthorized access, tampering, or cyber-attacks. It should implement encryption, authentication, and access control mechanisms to safeguard data integrity and user privacy.

6.2 Comparison of Existing and Proposed System

Existing system: In conclusion, the Ready to Drive a safety sensing system for electric vehicle using iot presents a promising solution for real-time battery monitoring and management. Through this system, users can remotely monitor vital battery parameters such as voltage, current, temperature, and state of charge using their Android devices, thereby enhancing convenience and efficiency in various applications ranging from home automation to industrial settings. Throughout the development and testing phases, several key observations have been made:

Reliability: The system has demonstrated reliability in continuously monitoring battery parameters and transmitting data to the Android IoT app, with minimal downtime and robust handling of network disruptions.

Efficiency: The system showcases efficiency in terms of energy consumption, data accuracy, and user experience. Energy-efficient protocols and optimization techniques ensure prolonged battery life for deployed devices, while real-time data updates and intuitive user interfaces enhance usability.

Scalability: The system exhibits scalability, capable of accommodating a growing number of monitored batteries and users without compromising performance or reliability. This scalability feature positions the system well for deployment in diverse environments with varying monitoring requirements. **Security:** Security measures such as encryption, authentication, and access control have been implemented to safeguard sensitive battery data and user privacy, ensuring the integrity and confidentiality of transmitted information.

6.3 Sample Code

```
#include <ESP8266WiFi.h>
String apiKey = "jioFiber" ;
const char * ssid = "Desktop windows dell"
const char * pass = "Rohan1234" ;
const char * server = "api . thingspeak . com" ;
#include "thingProperties.h"
int analogInPin = A0;
int sensorValue;
float calibration = 0.36;
```

```

int sensorValue;
float calibration = 0.36;
Serial.begin(9600);
delay(1500);
initProperties();
ArduinoCloud.begin(ArduinoIoTPreferredConnection);
setDebugMessageLevel(2);
ArduinoCloud.printDebugInfo();
}
void loop() {
  ArduinoCloud.update();
  sensorValue = analogRead(analogInPin);
  voltage = (((sensorValue * 3.3) / 1024) * 2 + calibration);
  bat_percentage = mapfloat(voltage, 2.8, 4.2, 0, 100);
  if (bat_percentage >= 100)
  {
    bat_percentage = 100;
  }
  if (bat_percentage <= 0)
  {
    bat_percentage = 1;
  }

  Serial.print("Analog Value = ");
  Serial.print(sensorValue);
  Serial.print("\t Output Voltage = ");
  Serial.print(voltage);
  Serial.print("\t Battery Percentage = ");
  Serial.println(bat_percentage);
  delay(1000);

}
float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

```


}

Output

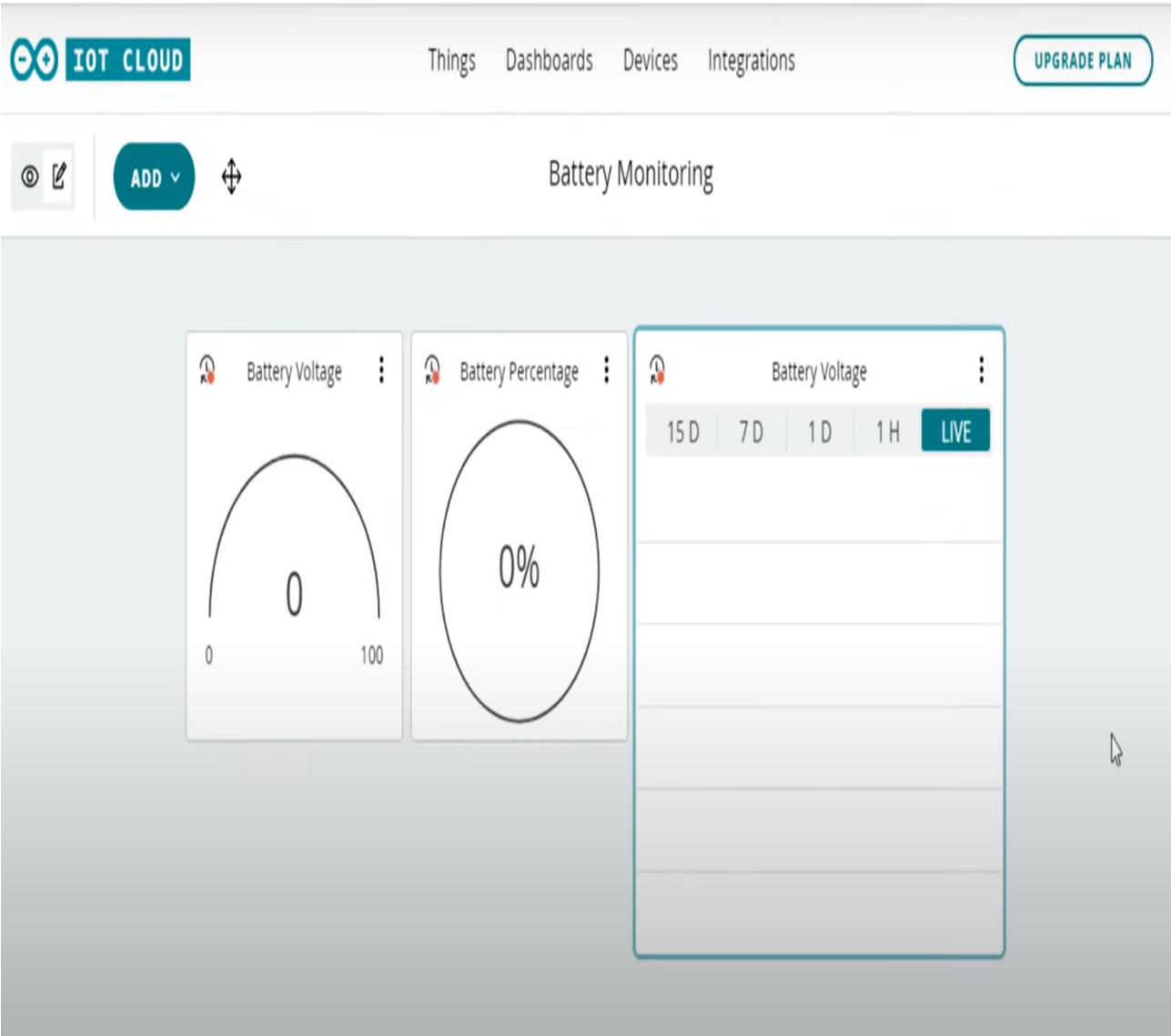


Figure 6.1: **Output 1**

Figure 6.1 It shows the Battery Percentage and Battery Voltage of connected Electric Vehicle



Figure 6.2: **Output 2**

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

Advanced BMS can significantly improve the performance of EVs. Adaptive mathematical models are an efficient tool for improving and refining BMS. The State-of-Charge determination algorithms, developed with the help of adaptive battery models, are highly precise and represent a good basis for practical implementation. BMS is a critical component of electric vehicles. That promotes guarantee safe, efficient and reliable battery operation. The combination of advanced charging algorithms and adaptive S BMS improves the battery functioning thus improving the characteristics of EV

7.2 Future Enhancements

IoT cloud can further improve its functionality, reliability, and user experience. Here are some potential enhancements:

Machine Learning and Predictive Analytics: Integrate machine learning algorithms to analyze historical battery data and predict potential issues or failures before they occur. This predictive maintenance capability can help users schedule maintenance activities proactively and prevent unexpected downtime.

Integration with Renewable Energy Systems: Extend the system to integrate with renewable energy systems such as solar panels or wind turbines. By monitoring battery health and energy usage in conjunction with renewable energy generation, users can optimize energy storage and consumption, maximizing the use of clean energy sources.

Fault Diagnosis and Troubleshooting: Develop advanced fault diagnosis algorithms to identify and troubleshoot battery issues remotely. This capability can hel-

pusers diagnose the root cause of battery problems quickly and accurately, reducing maintenance costs and minimizing downtime. Customizable Alerts and Notifications: Enhance the Android IoT app to allow users to customize alerts and notifications based on their preferences and usage patterns. Users can set thresholds for battery parameters and receive alerts via email, SMS, or push notifications when these thresholds are exceeded. Enhanced Security Features: Implement additional security features such as twofactor authentication, encrypted communication channels, and secure device provisioning. These security measures help protect sensitive battery data from unauthorized access and ensure the integrity and confidentiality

Chapter 8

INDUSTRY DETAILS

8.1 Industry Name: VOLT ME MOTORS PVT.LMT

8.1.1 Duration of Internship (JAN8- MAY9)

8.1.2 Duration of Internship: 5 months

8.1.3 Industry Address: 4th floor, 7-13, Sri lakshmi nilayam, near midland bakery madinaguda, miyapur, hyderabad - 500050

8.2 Internship offer Letter





Voltme Motors Private Limited
Next Generation EV Platform

INTERNSHIP OFFER LETTER

Date: 08-01-2024

Dear ~~Busa~~ Rohan Raj,

We are excited to offer an internship opportunity at VOLTME MOTORS PVT LTD Consultancy within the field of Electric Vehicles Control & communication and data management. Your passion and expertise in EV will align perfectly with our team's goals.

Internship Details:

Position: Electric Vehicles Control & communication Intern
Internship Period: January 8, 2024, May 25, 2024 (5 months)
Location: Hyderabad, India
Compensation: Unpaid
Mode: Work from Office

Responsibilities:

Throughout your internship, ~~you'll~~ be involved in:
Engaging in Electric Vehicles Communication projects.
Collaborating closely with our seasoned Project team to gain hands-on experience.
Playing an active role in the development and implementation of solutions.

Acceptance:

Kindly confirm your acceptance of this internship offer within 30 days. We eagerly anticipate the opportunity to work alongside you and are confident that this internship will be mutually beneficial.

Regards,
Vikas Reddy
Managing Director

Figure 8.2: ~~Busa~~ Rohan Raj



Voltme Motors Private Limited
Next Generation EV Platform

INTERNSHIP OFFER LETTER

Date: 08-01-2024

Dear ~~Pokuri~~ Anirudh Reddy,

We are excited to offer an internship opportunity at VOLTME MOTORS PVT LTD Consultancy within the field of Electric Vehicles Control & communication and data management. Your passion and expertise in EV will align perfectly with our team's goals.

Internship Details:

Position: Electric Vehicles Control & communication Intern
Internship Period: January 8, 2024, May 25, 2024 (5 months)
Location: Hyderabad, India
Compensation: Unpaid
Mode: Work from Office

Responsibilities:

Throughout your internship, ~~you'll~~ be involved in:
Engaging in Electric Vehicles Communication projects.
Collaborating closely with our seasoned Project team to gain hands-on experience.
Playing an active role in the development and implementation of solutions.

Acceptance:

Kindly confirm your acceptance of this internship offer within 30 days. We eagerly anticipate the opportunity to work alongside you and are confident that this internship will be mutually beneficial.

Regards,
Vikas Reddy
Managing Director

Figure 8.3: Pokuri Anirudh Reddy

8.3 Internship Completion Certificate

Chapter 9

PLAGIARISM REPORT

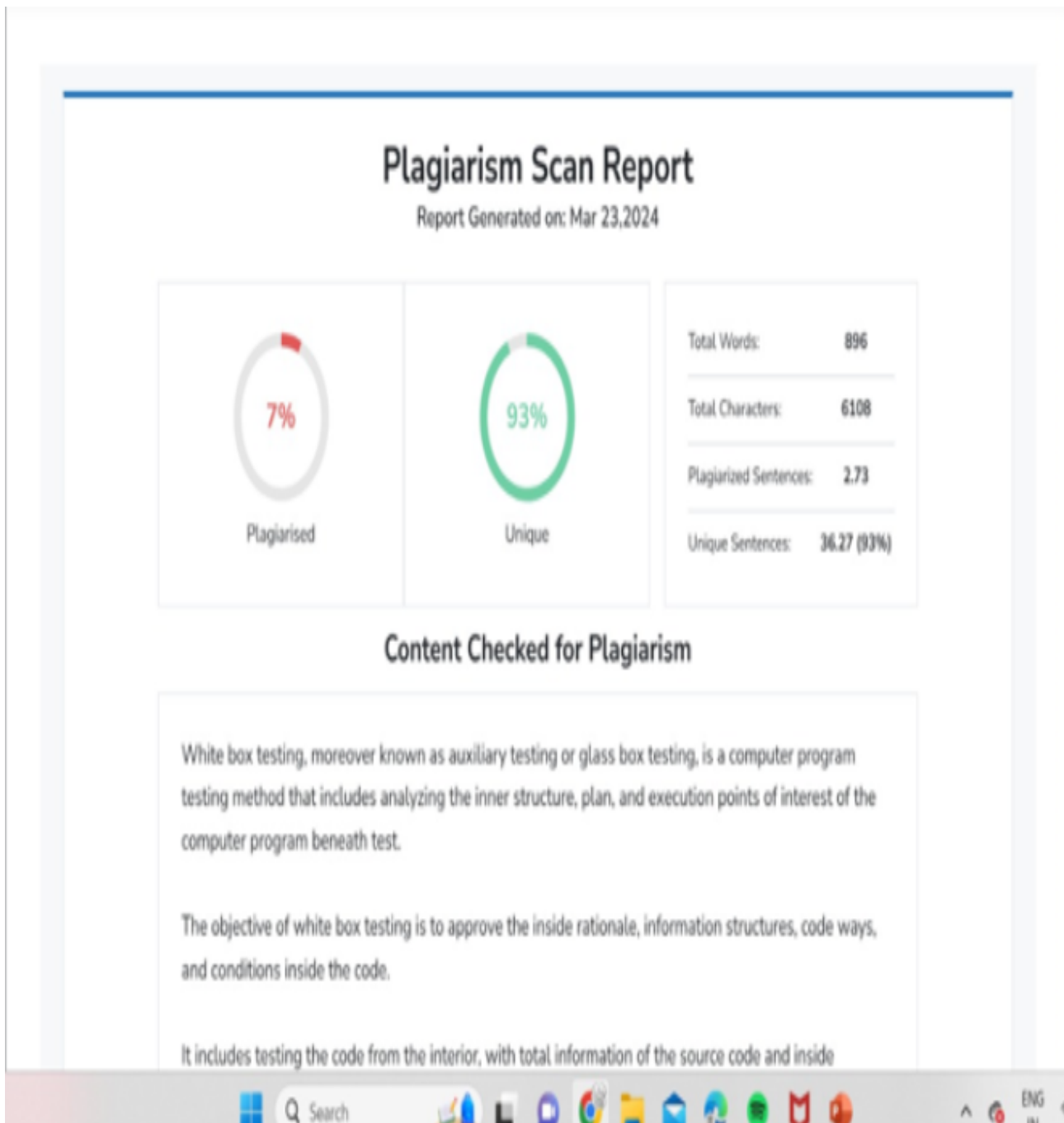


Figure 9.1: Plagiarism Report


Chapter 10

SOURCE CODE & POSTER PRESENTATION


10.1 Source Code

```
include ESP8266WiFi.h;
String apiKey = "*****";
const char * ssid = "*****";
const char * pass = "*****";
const char * server = " api . thingspeak.com ";
include "thingProperties.h"
int analogInPin = A0;
int sensorValue;
float calibration = 0.36;
int sensorValue;
float calibration = 0.36;
Serial.begin(9600);
delay(1500);
initProperties();
ArduinoCloud.begin(ArduinoIoTPreferredConnection);
setDebugMessageLevel(2);
ArduinoCloud.printDebugInfo();
```


10.2 Poster Presentation



Vel Tech
Rangarajan Dr. Sagayam
Vellore Institute of Science and Technology
Chennai-600 127, India



NAAC
A++
ACCREDITED



USC
UNIVERSITY STUDENT COUNCIL
CHENNAI

READY TO DRIVE SAFETY SENSING SYSTEM FOR ELECTRIC VEHICLE USING IOT

Department of Computer Science and Engineering
School of Computing
1156CS701-MAJOR PROJECT
INTERNSHIP THROUGH DIND/PLACEMENT/ABROAD
VOLT ME MOTORS PVT LMT
WINTER SEMESTER 2023-2024

Batch: (2020-2024)

ABSTRACT

The integration of Internet of Things (IoT) technology in electric vehicles (EVs) has paved the way for enhanced safety features and driving experience. This project focuses on the development of a ready-to-drive safety sensing electric vehicle utilizing IoT advancements. The core objective is to design a comprehensive system that integrates various IoT sensors and devices to monitor and improve the safety aspects of EVs.

Key components of the system include sensors for monitoring environmental conditions, such as temperature, humidity, and air quality, to ensure optimal driving conditions for passengers and the vehicle itself. Furthermore, the integration of IoT technology enables remote monitoring and control of the EV's vital parameters, including battery status, charging levels, and overall performance. This not only enhances the convenience for users but also contributes to the longevity and efficiency of the vehicle.

INTRODUCTION

In response to this imperative, the project "Ready-to-Safety Sensing System for Electric Vehicle using IoT" aims to leverage the power of Internet of Things (IoT) technology to enhance the safety features and driving experience of electric vehicles. With the proliferation of IoT-enabled devices and sensors, there exists a unique opportunity to create a comprehensive safety sensing system that monitors and responds to potential hazards in real-time.

The primary objective of this project is to develop a sophisticated safety sensing system that integrates seamlessly with electric vehicles, providing drivers with enhanced situational awareness and proactive assistance in navigating challenging road conditions. By leveraging IoT sensors deployed both within and around the vehicle, the system will continuously monitor environmental factors such as temperature, humidity, air quality, and road conditions, as well as detect potential obstacles, pedestrians, and other vehicles.

RESULTS

Compliance with safety regulation and standards can be ensured through the implementation of robust safety sensing system reducing illegal risk for both manufacturers and use of electric vehicles.

Overall the result of implementing a safety sensing system for electric vehicles using iot would likely be an improvement in safety, efficiency, and over all user experience for drivers and passengers. However, the specific outcomes would depend on factors such as design, implementation, and effectiveness of the system, as well as regulatory consideration and user adoption.

STANDARDS AND POLICIES

Developing a safety sensing system for electric vehicles using IoT is crucial for enhancing road safety. This project addresses the growing need for sustainable transportation solutions, reducing accidents, and promoting the widespread adoption of electric vehicles, contributing to a safer and environmentally friendly future.

Additionally, the project aligns with global efforts to minimize carbon emissions, as electric vehicles play a key role in sustainable mobility. By integrating IoT technology for safety, it not only protects occupants but also fosters public confidence in adopting eco-friendly transportation alternatives, ultimately benefiting society as a whole.

METHODOLOGIES

System Architecture: Design the overall architecture of the system including hardware components (ESP8266, sensors, battery interface), software components (firmware, backend server, database), and communication protocols (Wi-Fi, MQTT, HTTP).

Requirements Analysis: Understand the requirements of the battery monitoring system such as the types of batteries to monitor, communication protocols, data visualization, etc.

Backend Development: Set up a backend server to receive data from the IoT devices, store it in a database, and provide APIs for accessing the data. You can use platforms like AWS IoT, Google Cloud IoT, or self-hosted solutions like MQTT broker and database server.

Data Visualization: Create a user interface for visualizing battery parameters such as voltage, current, temperature, and state of charge. This can be a web dashboard, mobile app, or desktop application.

Hardware Selection and Integration: Choose suitable sensors for measuring battery parameters like voltage, current, temperature, etc. Integrate these sensors with ESP8266 and Arduino IoT board.

Firmware Development: Develop firmware for ESP8266 and Arduino IoT to read data from sensors, process it, and send it to the backend server using Wi-Fi or other communication protocols.

CONCLUSIONS

Advanced BMS can significantly improve the performance of EVs. Adaptive mathematical models are an efficient tool for improving and refining BMS. The State-of-Charge determination algorithms, developed with the help of adaptive battery models, are highly precise and represent a good basis for practical implementation. BMS is a critical component of electric vehicles.

That promotes guarantee safe, efficient and reliable battery operation. The combination of advanced charging algorithms and adaptive BMS improves the battery functioning thus improving the characteristics of EV.

ACKNOWLEDGEMENT

1. Project Supervisor J.SWAPNA M.E Assistant professor
2. Project supervisor Contact No:73585 72063
3. Project supervisor Mail ID:swapnaj@veltech.edu.in

TEAM MEMBER DETAILS

<Student 1.17736 BALGAM NITHIN>
<Student 2 .17727 BUSA ROHAN RAI>
<Student 3 .21646 POKURI ANIRUDH REDDY>
<Student 1. 8978669291>
<Student 2. 9154312928>
<Student 3. 79951 38808>
<Student1.
vtu17736@veltechj.edu.in>
<Student 2.
vtu17727@veltech.edu.in>
<Student3.
vtu21646@veltech.edu.in>

Figure 10.1: Poster

References

- [1] A. Rahman, M. Rahman and M. Rashid, "Wireless battery management system of electric transport", IOP Conf. Ser. Mater. Sci. Eng, 2017
- [2] D. Battery Management Systems for Large Lithium Ion Battery Packs, 1st ed.; Artech House: London, UK, 2010; pp. 22–110
- [3] E.; Richter, G. Battery monitoring and electrical energy management precondition for future vehicle electric power systems. J. Power Sources 2003, 116, 79–98
- [4] F. Zhu, G. Liu, C. Tao, K. Wang and K. Jiang, "Battery management system for Li-ion battery", The Journal of Engineering, vol. 2017, no. 13, pp. 1437-1440, 2017
- [5] M. A. Hannan, M. M. Hoque, A. Hussain, Y. Yusof and P. J. Ker, "State of the Art and Energy Management System of Lithium-Ion Batteries in Electric Vehicle Applications: Issues and Recommendations", IEEE Access, vol. 6, pp. 19362-19378, 2018
- [6] N. Hossein Motlagh, M. Mohammadrezaei, J. Hunt and B. Zakeri, "Internet of Things (IoT) and the Energy Sector", Energies, vol. 13, no. 2, pp. 494, Jan. 2020.
- [7] R. Fang, K. Chen, L. Yin, Z. Sun, F. Li and H.M. Cheng, "The Regulating Role of Carbon Nanotubes and Graphene in Lithium-Ion and Lithium Sulphur Batteries", Adv. Mater, 2018.
- [8] T.; Fang, F.; Wang, X.P.; Ashtiani, C.; Pesaran, A. A modular battery management system for HEVs. Future Car Congress 2002, doi:10.4271/2002- 01-1918
- [9] X. Kuang, K. Li, Y. Xie, C. Wu, P. Wang, X. Wang, et al., "Research on Control Strategy for a Battery Thermal Management System for Electric Vehicles Based on Secondary Loop Cooling", IEEE Access, vol. 8, pp. 73475- 73493, 2020
- [10] Yonghua, Y. Yuexi and H. Zechun, "Present Status and Development Trend of Batteries for Electric Vehicles", Power System Technology, vol. 35, no. 4, pp. 1-7, 2011

General Instructions

- Cover Page should be printed as per the color template and the next page also should be printed in color as per the template
- **Wherever Figures applicable in Report , that page should be printed in color**
- Dont include general content , write more technical content
- Each chapter should minimum contain 3 pages
- Draw the notation of diagrams properly
- Every paragraph should be started with one tab space
- Literature review should be properly cited and described with content related to project
- All the diagrams should be properly described and dont include general information of any diagram
- Example Use case diagram - describe according to your project flow
- All diagrams,figures should be numbered according to the chapter number and it should be cited properly
- **Testing and codequality should done in Sonarqube Tool**
- Test cases should be written with test input and test output
- All the references should be cited in the report
- **AI Generated text will not be considered**
- **Submission of Project Execution Files with Code in GitHub Repository**
- **Thickness of Cover and Rear Page of Project report should be 180 GSM**
- **Internship Offer letter and neccessary documents should be attached**
- **Strictly dont change font style or font size of the template, and dont customize the latex code of report**
- **Report should be prepared according to the template only**
- **Any deviations from the report template,will be summarily rejected**

- **Number of Project Soft Binded copy for each and every batch is (n+1) copies as given in the table below**
- For **Standards and Policies** refer the below link
<https://law.resource.org/pub/in/manifest.in.html>
- Plagiarism should be less than 15%
- **Journal/Conference Publication proofs should be attached in the last page of Project report after the references section**

width=!,height=!,page=-