DrillBit

The Report is Generated by DrillBit Plagiarism Detection Software

Submission Information

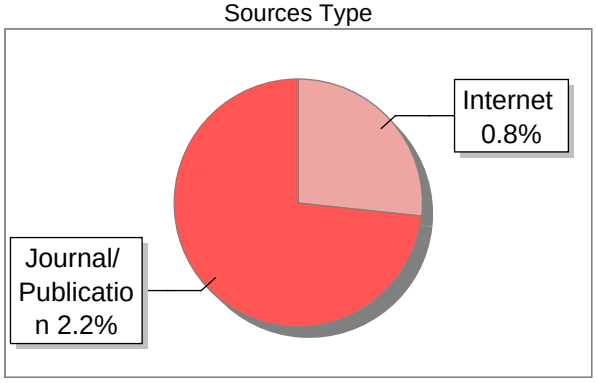
Author Name	Laxman M Nayanegali
Title	Project Report
Paper/Submission ID	2934543
Submitted by	asklibrarian@acharya.ac.in
Submission Date	2025-01-03 11:43:56
Total Pages, Total Words	44, 7350
Document type	Project Work

Result Information

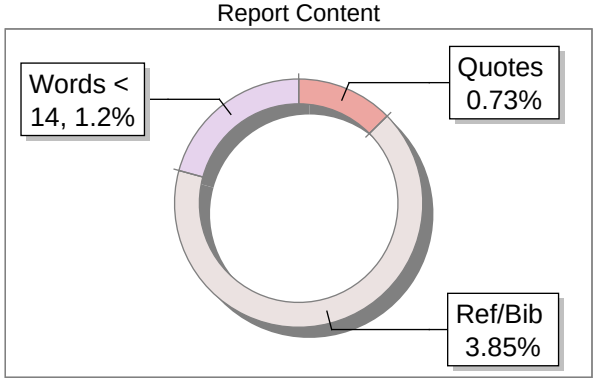
Similarity **3 %**

1 10 20 30 40 50 60 70 80 90

Sources Type



Report Content



Exclude Information

Quotes	Excluded
References/Bibliography	Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	**0 %**
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File

DrillBit

DrillBit Similarity Report

3	12	A
SIMILARITY %	MATCHED SOURCES	GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	orca.cardiff.ac.uk	<1	Publication
2	www.e3s-conferences.org	<1	Publication
3	www.ijcrt.org	<1	Publication
6	ebin.pub	<1	Internet Data
7	The next evolution of MDE a seamless integration of machine learning by Hartmann-2017	<1	Publication
8	Learning Personalized Discretionary Lane-Change Initiation for Fully Autonomous by Liu-2020	<1	Publication
9	www.eelet.org.uk	<1	Publication
10	springeropen.com	<1	Internet Data
11	springeropen.com	<1	Internet Data
12	www.freepatentsonline.com	<1	Internet Data
13	moam.info	<1	Internet Data
14	www.dx.doi.org	<1	Publication

ABSTRACT

The complexity and hazards of autonomous vehicle systems have posed a significant challenge in vehicle maintenance. An automotive system is important for the reduction of surprise failures, reduction of downtime, and improvement in vehicle performance. There is a need for a IoT-based predictive maintenance solution for key components of vehicles, where the information from sensors in real-time about the status of temperature, vibration, battery conditions, among other parameters, can predict failures with necessary time and action for maintenance based on machine learning algorithms and data analytics. This project uses the advances in the related technologies of autonomous vehicle PDM and machine learning algorithm to address the inadequacies of current maintenance methods by developing a proactive solution for safe, cost-effective, and overall reliability of the vehicle. The expected output will be a real-time dashboard, an alert system, and validated predictive algorithms that hugely improve vehicle performance and operational cost.

Table of Contents

Table of Figures

CHAPTER: 01

INTRODUCTION

Predictive maintenance (PDM) has transformed the automotive industry, shifting from reactive problem-solving to a proactive approach that emphasizes precision. With machine learning, PDM can forecast when a machine or vehicle component is likely to fail. It analyzes historical data and interprets real-time signals to anticipate breakdowns before they occur. From monitoring vibration patterns and oil quality to assessing thermal signatures and equipment performance, PDM turns raw data into useful insights. In the high-pressure environment of automotive manufacturing, where even a brief period of downtime can lead to significant losses, PDM helps ensure that machines remain reliable, operations run smoothly, and costs are managed effectively. Maintenance evolves from a reactive and often disruptive task to a streamlined and efficient process, demonstrating how technology can proactively enhance reliability and drive progress. [1]

Traditional vehicle maintenance methods, such as periodic maintenance and reactive repairs, often lead to unnecessary part replacements, unexpected breakdowns, and high operational costs for vehicle owners. This project focuses on implementing PDM for automotive systems using IoT, By the sensor data and real-time monitoring, the system predicts potential failures, reducing downtime and maintenance costs, while improving vehicle performance and safety. Unplanned vehicle breakdowns lead to costly repairs, unexpected downtime, and safety risks. Present methods are unreliable since they rely totally on servicing that is scheduled or based on break-down/reactive events. We need an intelligent, real-time solution that predicts defects before they take place, ensuring optimal performance, safety, and cost-efficiency in automotive systems. Predictive maintenance using IoT technology is transforming the automotive industry by giving insights into vehicle health and performance. Traditional maintenance methods rely on scheduled servicing or fixing issues after they occur, leading to unexpected breakdowns and increased repair costs. In contrast, predictive maintenance checks information gathered from sensors in the vehicle to monitor its condition continuously and predict faults before they happen.

The IoT enables seamless connectivity between the vehicle's onboard sensors and ML model. This connected network allows the collection and checking data such as engine temperature, fuel levels, tire pressure, and battery health. By continuously monitoring parameters such as temperature, vibration, oil level, battery performance, and gas levels, the system identifies potential failures before they occur. The

integration of an ESP32 microcontroller with a Python Flask-based server enables real-time data processing, anomaly detection, and user-friendly dashboard visualization.

1.1 Problem Statement

Traditional vehicle maintenance methods are inefficient, leading to:

- Unnecessary component replacements.
- High operational costs.
- Unplanned downtime.

Modern vehicles lack real-time monitoring and predictive capabilities for critical components like the engine, battery, and brakes. This results in difficulty in identifying potential failures early, suboptimal performance and reliability.

1.2 Objectives

1. Develop a real-time vehicle monitoring system using ESP32 and various sensors.
2. Predict vehicle maintenance requirements using machine learning algorithms.
3. Send alerts to vehicle owners via Telegram messages for proactive maintenance.
4. Detect accidents and other critical issues such as overheating, low oil levels, and poor battery health.
5. Ensure wireless communication of sensor data to a central laptop for analysis and storage.

1.3 Proposed System

The proposed system introduces a predictive maintenance framework that combines Internet of things & machine learning for real-time vehicle monitoring and intelligent analysis. Key features of the proposed system include: [2]

1. **Sensors Integration**:
    - **DHT11**: Monitors engine temperature.
    - **DS18B20**: Tracks battery temperature.
    - **Voltage and Current Sensors**: Measure battery health.

- **Ultrasonic Sensor**: Detects engine oil levels.
- **MQ-3 Smoke Sensor**: Monitors smoke levels.
- **ADXL345 Accelerometer**: Detects vibrations and accidents.

2. **Data Transmission**:
   - Sensor data is wirelessly transmitted to a central laptop via Zigbee modules.
3. **Predictive Analysis**:
   - Machine learning model process sensor data to predict maintenance requirements and to evaluate vehicle's condition.
4. **User Alerts**:
   - Maintenance alerts (e.g., low oil, overheating, poor battery health) are sent to users via Telegram messages.
5. **Accident Detection**:
   - The system detects accidents based on vibration data and sends immediate alerts

CHAPTER: 02

LITERATURE SURVEY

1. Predictive Maintenance in the Automotive Sector

Author(s): Arena, F.; Collotta, M.; Luca, L.; Ruggieri, M.; Termine, F.G.

Title: Predictive Maintenance in the Automotive Sector, Mathematical and Computational Applications MDPI, 2021

Summary: A systematic literature review of statistical inference approaches, and ML techniques for PDM in the automotive sector.

2. An IoT Based Predictive Connected Car Maintenance Approach

Author(s): Rohit Dhall, Vijender Solanki

Title: An IoT Based Predictive Connected Car Maintenance Approach, International Journal of Interactive Multimedia and Artificial Intelligence, Vol. 4, No.3, 2022

Maintenance Approach

Summary: PDM for connected cars by using real-time data to monitor vehicle health and predict issues before they occur.

3. Predictive maintenance of autonomous vehicles using machine learning and IoT

Author(s): Jiang, R., et al.

Title: Predictive maintenance of autonomous vehicles using machine learning and IoT

Sustainable Computing: Informatics and Systems, 2021

Summary: PDM using both supervised techniques like, gathering historical sensor data labeled with outcomes & to extract relevant features.

4. Machine learning in autonomous vehicles

Author(s): Zhang, Y., Li, H., & Tang, X.

Title: Machine learning in autonomous vehicles: A survey, IEEE Transactions on Intelligent Transportation Systems, 2020

Summary: This research presents a framework for autonomous vehicle maintainance with machine learning algorithm.

5. Solar-Powered Agricultural Systems

Author(s): Sherien N. Elkateb, Ahmed Metwalli, Abdelrahman Shendhy Ahmed E.B.Abu-Elanien

Title: Machine learning & IoT – Based predictive maintenance approach for industrial applications, Alexandria Engineering Journal, 2024

Summary: Hyperparameter tuning and cross- validation techniques for PDM.

CHAPTER: 03

HARDWARE DESCRIPTION

3.1 ESP 32 Microcontroller

The ESP 32 is a single-chip device that offers both 2.4 GHz Wi-Fi & Bluetooth, developed using TSMC's low power 40 nm technology. Designed for optimal power efficiency, impressive RF performance, and versatility, ensuring reliability and features-rich. This design supports a diverse range of uses and efficiently accommodates various tasks.

Typically seen in mobile and wearable electronics, along with IoT applications, and it gives advanced low-power capabilities. Key features include fine-resolution clock gating, several power modes, and dynamic power scaling. For example, low-power Internet of things sensor hub, the ESP32 only wakes up at specific intervals or when certain conditions arise, employing a low duty cycle to save energy. Additionally, its adjustable power amplifier output helps strike a balance between communication, data rate, & power consumption, optimizing performance for various applications. [3]

The ESP32 stands out as a best solution for Wi-Fi and Bluetooth applications, needing fewer than 10 external components. It includes an antenna, low-noise receiver amplifier, RF balun, power amplifier, filters, and power management modules, which greatly minimizes the required PCB area. Utilizing CMOS technology, it has a fully working radio and baseband, with advanced calibration. These features automatically adjust for external imperfections and varying conditions, removing the necessity for expensive, Wi-Fi testing equipment during mass production.

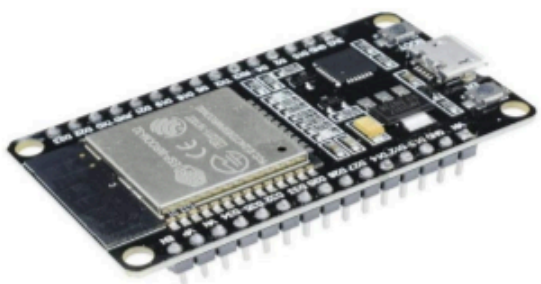


Figure 3.1: ESP 32 Microcontroller

3.2 Regulated power supply:

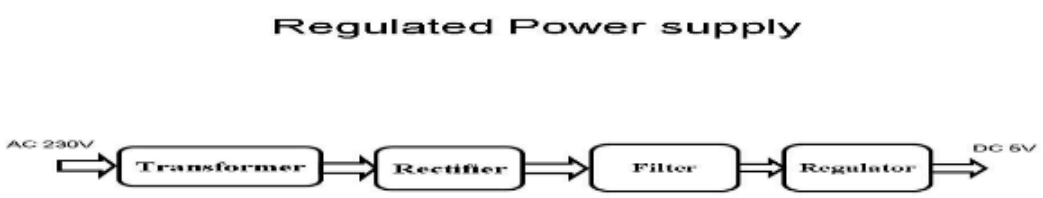


Figure 3.2: Regulated power supply

A transformer is a device that transfers magnetic flux between circuits using inductively coupled conductors while keeping the same frequency. It uses the principle of mutual induction, where a changing magnetic field created by the first winding induces a voltage in the second winding. This principle allows energy to move from the primary circuit to the secondary circuit. When a load is connected, current flows through the second winding. The AC in magnetic field, which fluctuates at 50 Hz for AC mains, interacts with nearby coils, inducing voltage in them. This basic concept is what makes transformers function. [4]

A rectifier is responsible for converting AC to direct current (DC) through a process called rectification, which is crucial for power supplies and signal detection. It typically employs multiple diodes to achieve efficient AC-to-DC conversion, although earlier models used vacuum tubes or selenium stacks. Filters are used to smooth out pulsating DC by eliminating unwanted frequencies, while voltage regulators ensure that fluctuating input is maintained as same output. Regulators like the LM78XX series for positive input and LM79XX for negative input are reliable devices that protect against over-current or overheating. However, applying reverse voltage can lead to immediate damage.

3.3 Voltage Sensor:

A voltage sensor is an essential electronic component designed to measure and monitor electrical potential difference in various electronic and electrical systems. It has a voltage divider that provides microcontrollers like ESP32 to safely measure higher voltage levels by scaling down the input voltage to a range compatible with the analog input pins. These sensors are crucial in battery monitoring applications, enabling precise measurement of battery charge levels, detecting voltage drops, and providing real-time status updates. They use resistor networks to divide the input voltage, ensuring safe

and accurate voltage readings while protecting the microcontroller from potential damage caused by high voltage inputs.

Figure 3.3: Voltage Sensor

This sensor mainly includes voltage divider circuit. The resistor in the circuit behaves as a sensing element. The voltage can be divided into two resistors, functioning as a reference voltage and a variable resistor, to create a voltage divider circuit. [5]

3.4 LCD Display 16X2

LCD, or Liquid Crystal Display, is a popular electronic display module used several applications, including prototypes, circuits, mobile phones, calculators, computers, and televisions. It is often preferred over multi-segment LEDs and seven-segment displays. Some main advantages include, low cost, ease of programming, and the ability to support custom characters, special symbols, and even animations, which makes it a flexible and economical display option for many devices and projects.

Figure 3.4: LCD Display

3.4.1 Registers of LCD

A 16×2 LCD features two main registers: the data register and the command register. The RS (register select) pin is used to switch between these registers. When the RS is set to '0', it indicates that the command register is being accessed. Conversely, when the RS is set to '1', it signifies that the data register is in use.

3.4.2 Command Register

The command register in an LCD is responsible for storing and processing the instructions sent to the display. These commands allow the display to work out specific tasks, like clearing the screen, initializing the system, positioning the cursor, and managing the display. The register plays an important role in ensuring that commands are efficiently executed to achieve the intended functions

- **Data Register**

The data register in an LCD holds the information that will be displayed on the screen. This information usually consists of the ASCII value corresponding to the character that needs to be displayed. When data is sent to the LCD, it first goes to the data register, marking the start of the display process. The data register is activated when the register set is set to 1, enabling the information to be processed and displayed on the LCD screen.

3.5 Current Sensor Module

The module is designed using the ZMCT series of small size high-precision micro-CT and high-precision operational amplifier circuits for more accurate sampling and proper signal compensation. It is best solution for the signal acquisition of AC current within 5A range.

The corresponding output voltage Analog AC signal can be adjusted using the potentiometer. You can adjust the amplification ratio and the amplification range (0-100 times).

Figure 3.5: Current Sensor Module

ZMCT103C AC current Sensor is the best for the DIY project and industrial application, where we need to measure the accurate AC current with current transformer. This is a perfect choice to measure the AC current using Arduino/ESP8266/Raspberry Pi like an opensource platform. In many electrical projects, engineer directly deals with measurements with few basic requirements like

- High accuracy
- Good Consistency

The ZMCT103 is a highly accurate micro current transformer that monitors AC mains current up to 5 amps. With a 1000:1 turns ratio and compact dimensions of 28 x 12 x 15 mm (L x W x H), this PCB-mounted transformer is perfect for integration into a variety of applications that need precise current measurement.)

3.6 DHT11 (Temperature & Humidity)

The DHT11 is a compact and popular sensor for measuring temperature and humidity, particularly among beginners and hobbyists. It's easy to connect to microcontrollers like Arduino and Raspberry Pi, requiring only a library installation to access data from its single output. The sensor consists of four pins: VCC (power), GND (ground), Data (for humidity and temperature readings), and NC (Not Connected). VCC and GND should be connected to the power supply, ensuring the voltage remains under 5V. The Data pin outputs both humidity and temperature readings, with a necessary delay between each reading. The NC pin is not usually for the data but can provide structural support for the module.

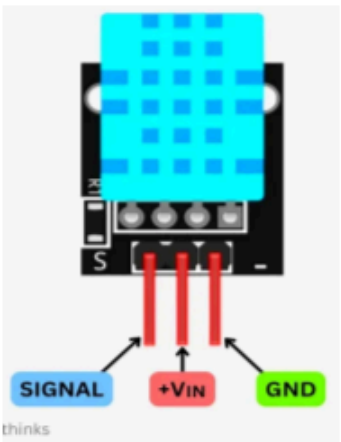


Figure 3.6: DHT 11 Sensors

3.7 Current Sensor:

A current sensor is designed to measure electrical current flow in a circuit by converting the current into a proportional voltage signal that can be easily read by microcontrollers. Typically, these sensors use various technologies such as Hall effect, shunt resistors, or current transformers to measure electrical current without directly interrupting the circuit. In IoT and electrical monitoring applications, current sensors play a critical role in measuring power consumption, detecting overload conditions, and monitoring electrical system performance. They provide real-time insights into energy usage, enable predictive maintenance, and support advanced power management strategies across diverse applications including renewable energy systems, industrial equipment, and electric vehicle charging infrastructures.

Figure 3.7: Current Sensor

3.8 Smoke MQ-3 Sensor:

The MQ-3 is a sensitive gas sensor that uses semiconductor technology to detect combustible gases and smoke in different settings. It can identify a range of gases, such as liquefied petroleum gas (LPG), propane, hydrogen, methane, and alcohol, which makes it a versatile tool for safety and detection purposes. It operates by measuring changes in electrical resistance when exposed to combustible gases, with sensitivity that can be adjusted using a potentiometer. Its analog output allows microcontrollers to interpret gas concentration levels, triggering alerts or activating safety mechanisms when dangerous gas levels are detected. Commonly used in fire alarm systems, gas leak detectors, and industrial safety applications, the MQ-3 provides an affordable and reliable solution for gas and smoke detection across home, commercial, and industrial settings.

Figure 3.8: MQ-3 Sensor

3.9 DS18B20 (Battery Temperature)

It is a highly versatile and sophisticated digital temperature sensor renowned for its precision and simplicity in temperature measurement applications. Manufactured by Dallas Semiconductor (now part of Maxim Integrated), this one-wire digital thermometer provides accurate temperature readings with a remarkable temperature range of -55°C to +125°C and an impressive accuracy of ±0.5°C between -10°C and +85°C.

Figure 3.9: DS18B20

Technical Operation: The DS18B20 communicates using the 1-Wire protocol, which allows multiple sensors to be integrated in a single data line. Each sensor has a unique 64-bit ROM code, enabling multiple sensors to coexist on the same bus without address conflicts. The sensor can provide temperature readings directly in digital format, eliminating the need for complex analog-to-digital conversion.

Communication Protocol: Sensors are connected via a single data wire, which also provides power in parasitic mode. The microcontroller (like ESP32) initiates communication by sending specific commands to read temperature, making it extremely efficient and easy to integrate into various projects. Practical

Implementation: When used with microcontrollers like ESP32, the DS18B20 requires minimal external components. A 4.7kΩ pull-up resistor is typically used between the data line ensure proper communication. It can be directly connected to digital pins, making it incredibly user-friendly for hobbyists and professional developers alike. The DS18B20's combination of accuracy, versatility, and ease of use makes it a preferred choice for projects requiring reliable temperature measurements across various domains.

3.10 Ultra Sonic Sensor

The HC-SR04 ultrasonic sensor utilizes SONAR technology to measure how far away objects are, akin to the way bats navigate their surroundings. It provides high-precision, non-contact distance measurements that range from 2 cm to 400 cm (1" to 13 feet). This sensor is designed to deliver stable readings and is straightforward to use, not influenced by sunlight or dark materials, though it might find it challenging to detect soft materials like cloth. The module includes both an ultrasonic transmitter and receiver, ensuring reliable distance sensing for various applications.

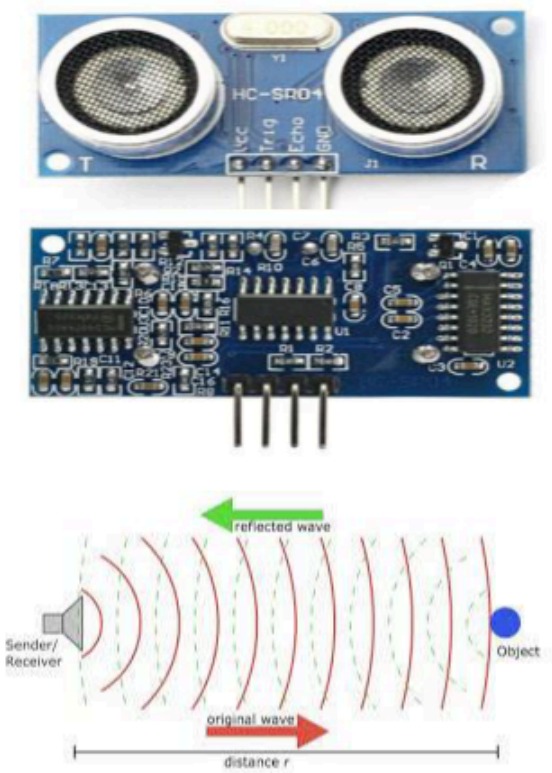


Figure 3.10: Ultra Sonic Sensor

3.11 Jumper Wires

A jump wire, also known as a jumper, jumper cable, DuPont wire, or DuPont cable, is an electrical wire or a collection of wires that have connectors or pins at each end (or sometimes just tinned). It is typically used to connect components on a breadboard or in prototype circuits without the need for soldering. These wires can be inserted into the slots of a breadboard, the header connectors of a circuit board, or test equipment to establish connections.

3.12 ADXL Sensor

An accelerometer is a tool that measures the vibration, motion, or acceleration of a structure. Cameras and smartphones these days use an accelerometer consisting of an axis-based motion sensor. It is an electromechanical device that measures either static or dynamic acceleration.

The motion sensors in accelerometers can detect earthquakes too. Another example is when the accelerometers measure the gravitational pull to determine at which angle is the device being titled.

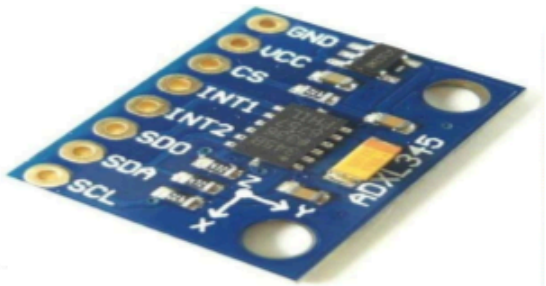


Figure 3.11: ADXL345 Sensor

.

When the sensor detects a 90-degree tilt, which may suggest an accident or sudden impact, it activates an alert. This functionality enables real-time monitoring and can assist in notifying the user or emergency services during a significant incident, thereby improving the safety features of the system.

CHAPTER: 04

SOFTWARE DESCRIPTION

4.1 Introduction to Arduino IDE

The Arduino IDE is incredibly user-friendly, yet it has all the features we need for most Arduino projects. The main part of the interface is a text editor where code can be written. Below that, the output window gives you important feedback, displaying the compilation status, memory usage, and any errors found in the code. It's a clear and efficient environment that helps to concentrate on developing and improving projects.

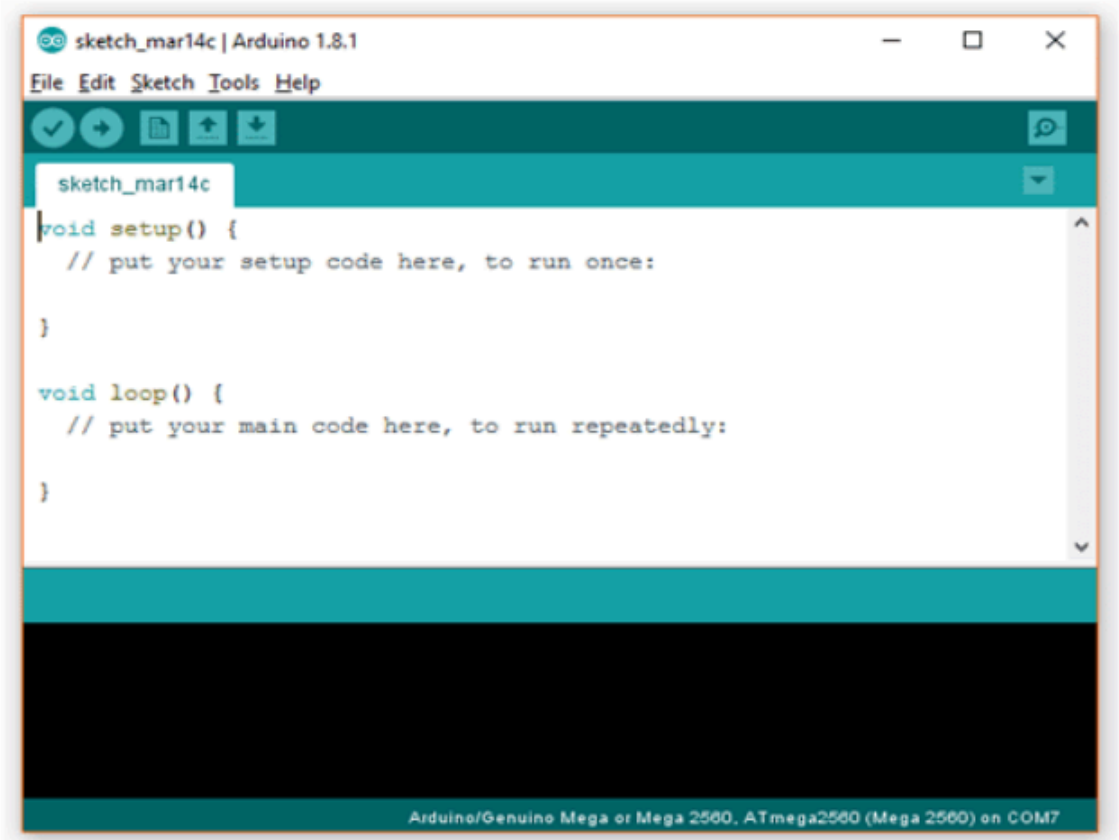


Figure 4.1: Arduino IDE Editor Window

Projects made with Arduino are called sketches, which are usually written in a simplified form of C++. To keep things simple, some features of C++ are left out. Programming a microcontroller is different

from programming a regular computer, so Arduino provides specific libraries that take care of tasks like setting pin modes, sending data to pins, reading analog values, and managing timers. These libraries make the process easier, letting you concentrate more on your project rather than the technical details of hardware interaction.

4.2 Flask Framework

Flask is a lightweight and flexible web framework created in Python, aimed at developing web applications and APIs. As a microframework, it offers just the core tools and depends on extensions for extra features, which makes it very versatile for different applications. Flask is straightforward to learn and use, making it perfect for IoT and machine learning integration projects like ours. [7]

- **Features:**

Figure 4.2: Flask Framework

- **WERKZEUG WSGI Toolkit**

WERKZEUG (German for "tool") is a comprehensive WSGI (Web Server Gateway Interface) library that acts as the foundation for Flask. It handles HTTP requests and responses, routing, and error management. In Our project it manages incoming sensor data sent from ESP32 via HTTP requests & ensures reliable routing of data to appropriate endpoints for processing.

- **Jinja2 Template Engine**

Jinja2 is a powerful templating engine used by Flask to generate dynamic HTML pages. It allows developers to embed Python code directly into HTML templates for seamless integration of back-end

logic with the front-end interface. It dynamically renders sensor data and ML predictions on the web dashboard, The Displayed alerts and historical trends retrieved from the SQLite database.

4.3 SQ Lite

SQLite is a lightweight, self-contained, and serverless relational database management system. It's commonly used in applications that prioritize simplicity, portability, and minimal setup. Unlike traditional databases, SQLite works directly with a single file, which makes it very efficient and easy to incorporate into embedded systems or small-scale projects like ours. [8]

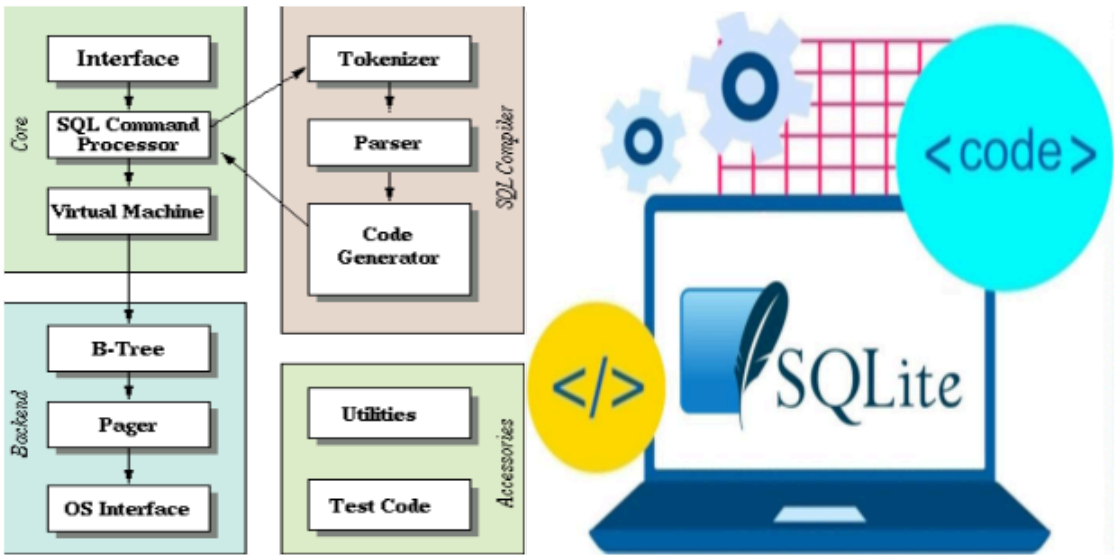


Figure 4.3: SQLite

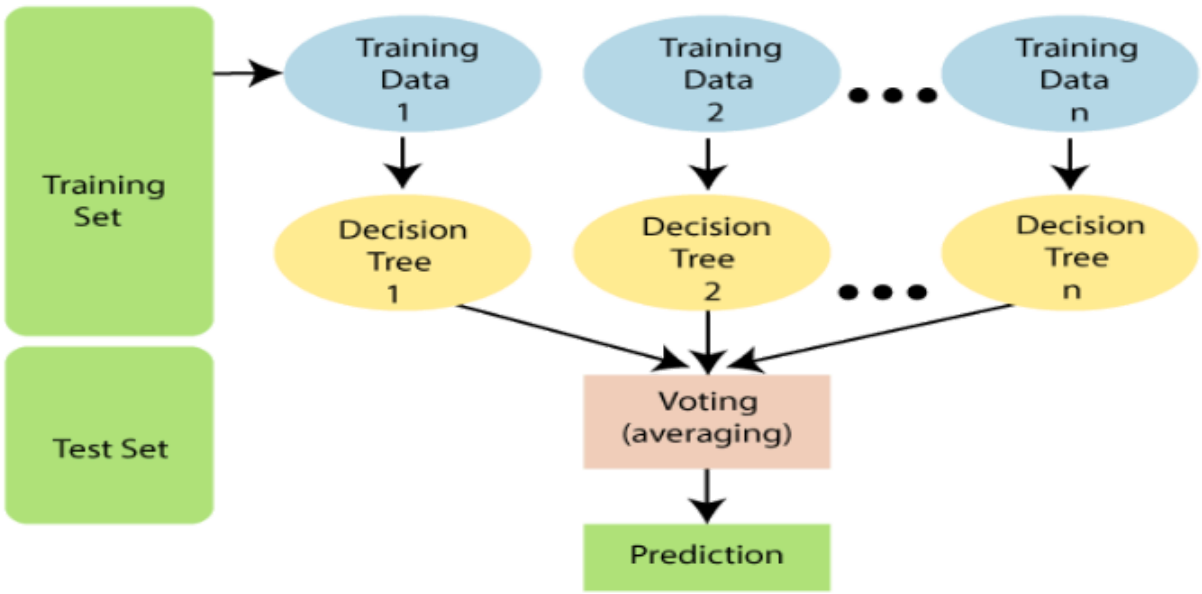
Features:

- Serverless: Functions without a dedicated server, minimizing resource needs.
- Self-Contained: All database logic is encapsulated in single library, eliminating the extra dependencies.
- Zero Configuration: No setup, installation, or configuration is required, making it perfect for rapid prototyping.
- Cross-Platform: It operates efficiently on a variety of platforms, including Windows, Linux, and embedded systems.

- Lightweight: Has a small storage footprint (~250KB), making it ideal for resource-limited systems like IoT devices.

4.4 ML Model (Random Forest Tree Algorithm)

The Machine learning model is crucial in the predictive maintenance system, as it analyzes sensor data to analyse patterns that can forecast failures before they happen. In this project, the Random Forest



Algorithm serves

Figure 4.4: Diagram of Machine Learning Model

as the primary ML model because of its high accuracy, robustness, and capability to manage both classification tasks (like anomaly detection) and regression tasks (such as predicting Remaining Useful Life). By utilizing historical data, the model facilitates proactive interventions, minimizing downtime and improving vehicle reliability. [9]

Features:

- Anomaly Detection: Detects unusual patterns in sensor data that may signal potential failures. Remaining Useful Life (RUL)
- Prediction: Predicts how much longer vehicle components will last based on sensor data.
- Robustness: Effectively manages noisy or incomplete data, providing dependable predictions.

- Scalability: Can be expanded with more data for wider applications across various components.
- Versatility: Facilitates both real-time predictions and offline analysis of historical data.
- Analysis: Determines which sensors are most influential in predictions, enhancing system efficiency.

4.5 JavaScript

JavaScript is a versatile and dynamic scripting language that enables the creation of interactive effects in web browsers. It's an essential tool for web development, enabling user interaction without needing to reload the page. JavaScript is widely used in modern web development to create dynamic, client-side applications.

Applications:

- Data Visualization
- User Interaction
- Integration with IoT
- Machine Learning Predictions

4.6 CSS (Cascading Style Sheets)

CSS is a stylesheet language used to describe the presentation of a web page written in HTML or XML. It controls the layout, design, colors, and fonts of the content displayed on web pages.

Applications:

- Dashboard Design
- Responsive Design
- Real-time Data Display
- Theme Customization

4.7 HTML (Hypertext Markup Language)

HTML is the standard markup language used to create web pages. It forms the backbone of web content, structuring the content and linking it to other resources like stylesheets (CSS) and scripts (JavaScript).

Applications:

- Web Interface
- Sensor Data Display
- User Input Forms

Figure 4.5: HTML, CSS, Java Script

4.8 Telegram Bot Integration

The Telegram bot plays a very important role in the project by sending real-time notifications to users regarding any anomalies found in the vehicle's critical components. By utilizing Telegram's secure and dependable API, the bot guarantees that users receive alerts straight to their mobile devices, which improves accessibility and user engagement. It serves as a link between the system's back-end processes and user interactions, providing timely updates on sensor anomalies like low oil levels, overheating, or gas leaks. Its straightforward setup, ease of use, and mobile accessibility make it a valuable addition to the predictive maintenance system.

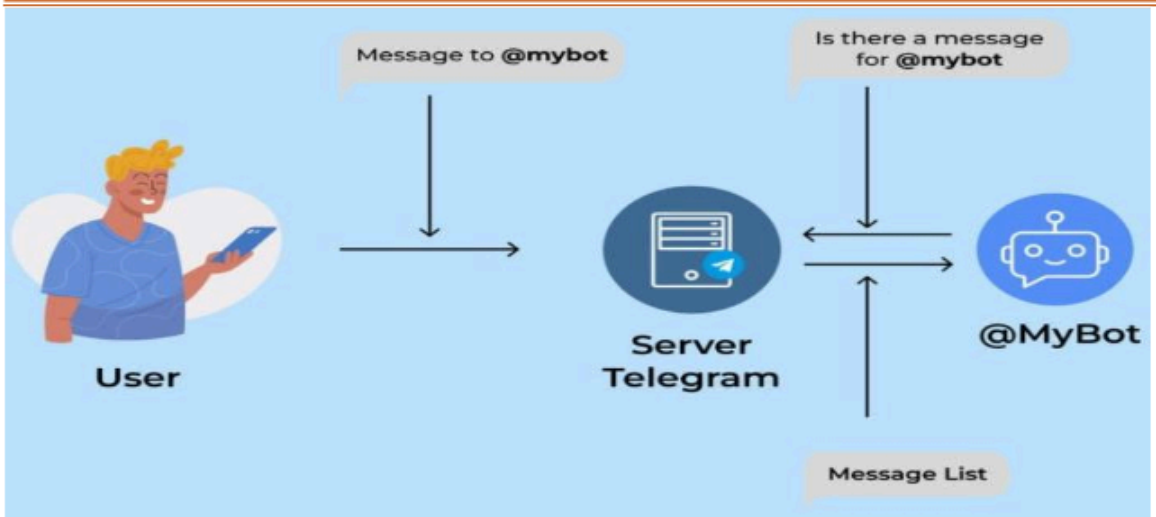


Figure 4.6: Telegram Bot

Benefits of Using Telegram Bot

1. Instant Notifications: Real-time alerts ensure timely action.
2. User-Friendly Interface: Users receive alerts directly on their phones.
3. Two-Way Communication: Allows users to query the system remotely.
4. Low Cost: Telegram API is free and efficient for IoT applications.

CHAPTER: 05

METHODOLOGY

5.1 System Design and Architecture

IoT sensors are assigned for keeping track of the condition of car parts in real-time. Various sensors, such as those measuring vibration, temperature, and pressure, along with other specialized devices, are used to identify wear and tear in important vehicle components. These sensors are carefully placed on key parts like the engine, transmission, and wheels, allowing for ongoing data collection and early identification of potential problems, which is vital for ensuring vehicle health and avoiding unexpected breakdowns.

Data Transmission & Web Interface: The information gathered from the sensors is sent to a central server for processing. Communication protocols like MQTT or HTTP are employed to transmit sensor data to either a cloud or local server, where further analysis takes place. A user-friendly web interface is created to present real-time data, alerts, and maintenance predictions to users. This interface facilitates interaction with the system, allowing users to input data, view predictive maintenance results, and receive notifications regarding necessary maintenance tasks.

Machine Learning Model: The predictive maintenance system relies on machine learning model to analyse sensor data and proactively predict possible failures or maintenance needs. Random Forest algorithm is used for this purpose, utilizing an ensemble learning approach. In the Initial phase, several decision trees are created, with each tree trained on a random subset of data using bootstrapping. At each decision point, a random selection of features is evaluated. The final output is determined by taking the most common class for classification tasks or the average prediction for regression tasks. This technique helps to mitigate overfitting and improves the model's ability to generalize. With historical data, the model can spot patterns that signal potential equipment failures, enabling proactive maintenance actions. [10]

Alert Mechanism: In the predictive maintenance system multiple sensor module are used to provide timely and effective notifications. When the machine learning model detects a fault or anomaly, the system activates alerts through three main channels: an LCD display, a buzzer, and a Telegram bot. The LCD display offers real-time visual alerts on the vehicle's dashboard or in the maintenance area, showing

messages such as “Engine Overheating” or “Vibration Anomaly Detected.” At the same time, the buzzer sounds an audible alarm to quickly capture the attention of the driver or mechanic. Furthermore, the Telegram bot sends personalized notifications to the user or maintenance team, enabling monitoring and interaction. These alerts help to avoid potential issues are addressed promptly, minimizing the risk of unexpected breakdowns and prolonging the vehicle's lifespan.

5.2 Block diagram

The block diagram illustrates a vehicle predictive maintenance system that combines IoT sensors, data processing, and machine learning to maintain optimal performance. Sensors such as temperature, vibration, fuel vapor, and ultrasonic oil level are strategically positioned on key vehicle components to monitor conditions in real-time. The ESP32 microcontroller gathers sensor data and transmits it to a backend machine learning model using a Zigbee module. This model analyzes the data to identify faults, forecast maintenance requirements, and avert potential failures. Alerts are shown on an LCD, activated by a buzzer, and sent remotely through a Telegram bot, allowing users to respond promptly and prevent unexpected breakdowns. Additionally, the system includes a dashboard interface for real-time monitoring and insights.

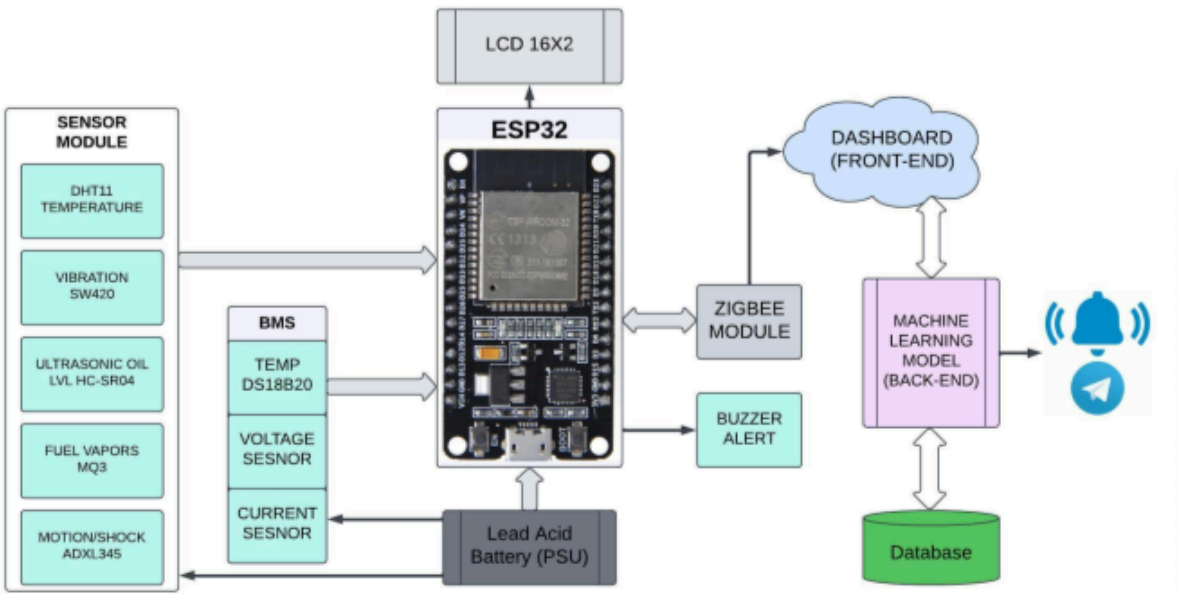


Figure 5.1: Block Diagram

The block diagram illustrates the architecture of a Predictive Maintenance System for Automobiles that

utilizes IoT and machine learning. The components and their functions are detailed below:

1. Sensor Module

This module gathers real-time data from different parts of the vehicle:

- DHT11 (Temperature Sensor): Monitors the ambient temperature.

- Vibration Sensor (SW420): Detects unusual vibrations that could signal faults.

- Ultrasonic Sensor (HC-SR04): Measures the oil level to identify shortages.

- Fuel Vapors Sensor (MQ3): Detects fuel vapor leaks.

- Motion/Shock Sensor (ADXL345): Monitors shocks or jerks that may indicate physical impacts or system problems.

2. Battery Management System (BMS)

The BMS maintains the health of the vehicle's power supply by monitoring:

- Temperature Sensor (DS18B20): Measures the battery's temperature to detect overheating.

- Voltage Sensor: Keeps track of the battery voltage to ensure it operates correctly.

- Current Sensor: Monitors current usage to identify overloading or abnormal draw.

3. ESP32 Microcontroller

The ESP32 acts as the central controller:

- Collects data from the sensors and the BMS.

- Displays essential information on the LCD Screen (16x2) for the driver.

- Activates the Buzzer Alert when an anomaly is detected.

- Sends data to the Zigbee Module for wireless transmission.

4. Zigbee Module

This module facilitates wireless communication:

- Transmits the collected sensor data from the ESP32 to the back-end system for further analysis.

5. Back-End System

This system processes and analyzes the transmitted data:

- Machine Learning Model:

- Identifies patterns and predicts potential failures.

- Flags anomalies based on historical and real-time data.

- Database: Stores sensor data, predictions, and historical data for analysis and enhancing ML model.

6. Dashboard (Front-End)

A GUI for user to monitor real-time data, alerts, and historical trends.

- It provides actionable insights to the user or maintenance team.

7. Alerts and Notifications

- Buzzer Alert: Provides immediate audio feedback to the driver when critical issues arise.

- Telegram Bot: Delivers notifications with detailed information to remote users or maintenance teams, enabling proactive responses.

8. Power Supply (PSU)

- The system operates on a Lead Acid Battery, guaranteeing dependable functionality for all components.

5.3 Data and alerts mechanism display

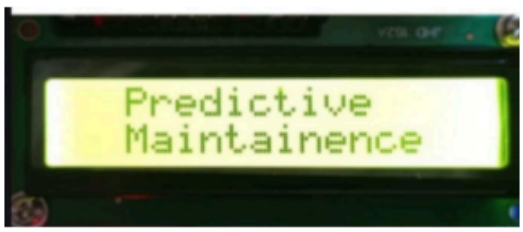


Figure 5.2: Data Display on LCD

5.4 Flow Diagram

The predictive maintenance system starts with its initialization, followed by the collection of information from several sensors, which is then sent to the ESP32 microcontroller. This microcontroller transmits the data to a Flask server using Zigbee for further analysis. The server evaluates the data against set thresholds and employs machine learning models to forecast potential faults or maintenance requirements based on historical data. When an anomaly or problem is identified, the system triggers alerts. These alerts are shown on an LCD, activated by a buzzer, and also sent out remotely through a Telegram bot. The LCD display is monitored to ensure data updates are successful, with retries conducted if needed. Historical data is preserved to enhance the predictive model continuously. This process is repeated for ongoing monitoring, allowing for timely fault detection and reducing the risk of unexpected failures.

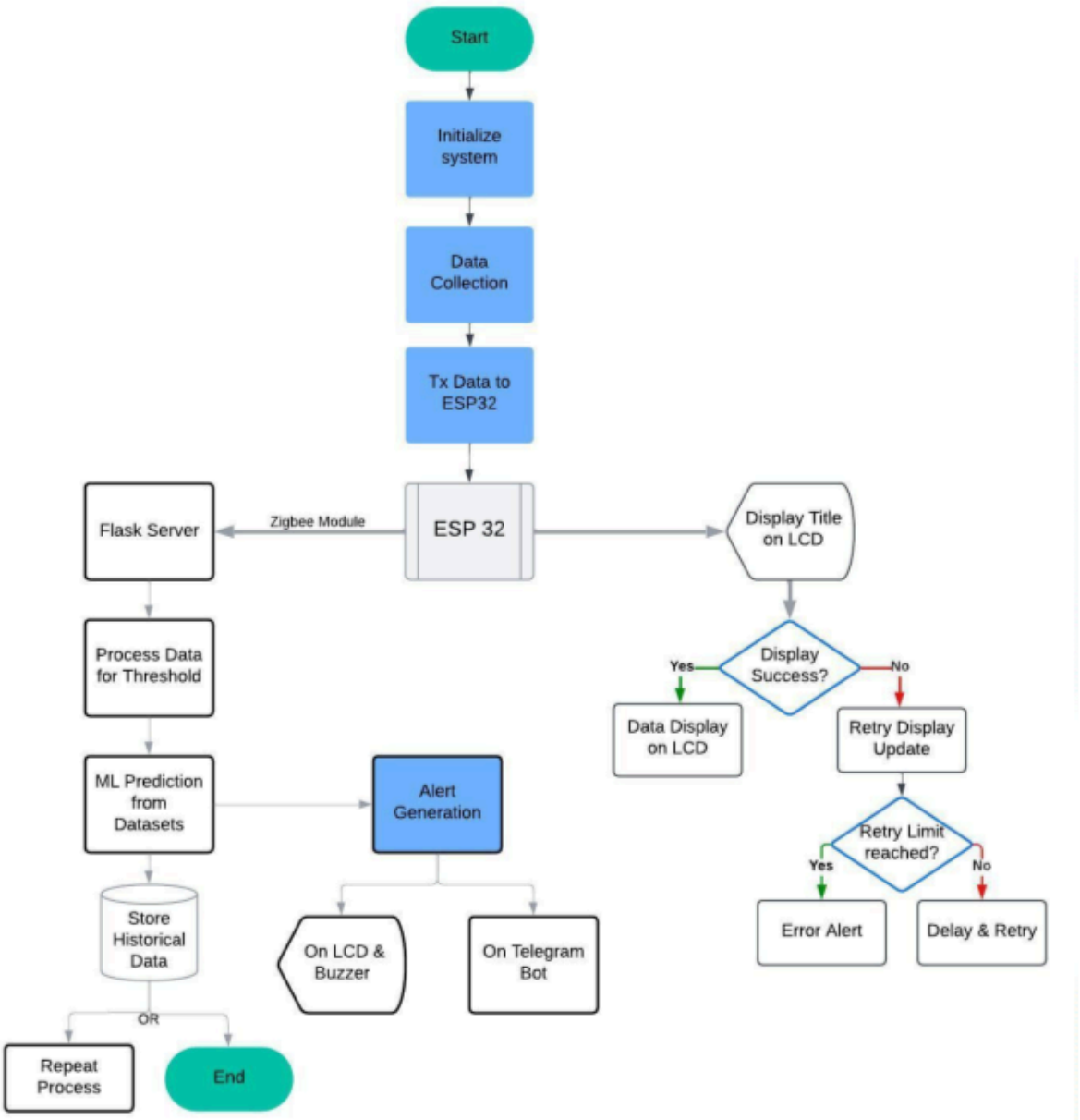


Figure 5.3: Flow Diagram

1. The process starts with the activation of the predictive maintenance system, which can be done either manually or automatically when the vehicle is turned on.

2. Initialize System

- The system sets up all hardware components (like sensors, ESP32, LCD, and Zigbee module) along with software components (including the Flask server and machine learning model).

- Network connections are established to facilitate data communication.

3. Data Collection

- IoT sensors within the vehicle gather real-time data on critical parameters such as:

- Engine temperature

- Vibrations

- Oil pressure

- Tire pressure

- Battery voltage

- This collected data is then sent to the ESP32 microcontroller.

4. Transmit Data to ESP32

- The sensor data is transmitted to the ESP32, which preprocesses and organizes it for further transmission.

5. Display Title on LCD

- The ESP32 shows an initialization or monitoring status message on the vehicle's LCD.

6. Check Display Success

-Success: If the LCD successfully displays the data, the process proceeds to the next step.

-Failure: If the LCD does not display correctly, a retry mechanism is activated.

7. Retry Display Update

- The system attempts to display the message on the LCD again, within a set number of attempts:

- Retry Limit Reached: If the maximum number of retries is reached, an error alert is generated.

- Retry Successful: If the display update is successful, the process continues.

8. Data Display on LCD

- Real-time data collected from the sensors is shown on the vehicle's LCD for the driver to monitor.

9. Communicate with Flask Server

- The ESP32 sends the collected sensor data to the Flask server through the Zigbee module for further processing and analysis.

10. Process Data for Thresholds

- The Flask server processes the incoming data and compares it against predefined thresholds.

- Example:

- If the engine temperature goes beyond a safe limit, it flags a potential overheating issue.

11. Machine Learning Predictions

- The processed data is the input for the ML model that has historical datasets.

The model forecasts potential failures by analyzing patterns in the data.

12. Generate Alerts

- Based on the analysis and predictions:

- On LCD & Buzzer: Alerts are shown on the LCD screen, and an audible buzzer sounds to immediately notify the driver.

- On Telegram Bot: Alerts are sent to the user or maintenance team through Telegram, offering detailed information about the issue.

13. Store Historical Data

- All processed data and predictions are stored in a database:

- To enhance the accuracy of ML models.

- To conduct long-term trend analysis on vehicle components.

14. Repeat Process or End

- Repeat: If the vehicle is still active, the system continues to loop back to collect and process data.

- End: If the vehicle is turned off or the monitoring session concludes, the process halts.

5.5 Code Execution

```
C:\Users\mnlax\OneDrive\Desktop\PREDICTIVE_MAINTANANCE\PREDICTIVE_MAINTAINANCE>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 765-231-920
Exception in thread Thread-1:
Traceback (most recent call last):
  File "C:\Users\mnlax\AppData\Local\Programs\Python\Python37\lib\threading.py", line 917, in _bootstrap_inner
    self.run()
  File "C:\Users\mnlax\AppData\Local\Programs\Python\Python37\lib\threading.py", line 865, in run
    self._target(*self._args, **self._kwargs)
  File "C:\Users\mnlax\OneDrive\Desktop\PREDICTIVE_MAINTANANCE\PREDICTIVE_MAINTAINANCE\app.py", line 22, in GetData
    from database import Read_Data
  File "C:\Users\mnlax\OneDrive\Desktop\PREDICTIVE_MAINTANANCE\PREDICTIVE_MAINTAINANCE\database.py", line 20, in <module>
    timeout=1
  File "C:\Users\mnlax\AppData\Local\Programs\Python\Python37\lib\site-packages\serial\serialwin32.py", line 33, in __init__
    super(Serial, self).__init__(*args, **kwargs)
  File "C:\Users\mnlax\AppData\Local\Programs\Python\Python37\lib\site-packages\serial\serialutil.py", line 244, in __init__
    self.open()
  File "C:\Users\mnlax\AppData\Local\Programs\Python\Python37\lib\site-packages\serial\serialwin32.py", line 64, in open
    raise SerialException("could not open port {!r}: {!r}".format(self.portstr, ctypes.WinError()))
serial.serialutil.SerialException: could not open port 'COM3': PermissionError(13, 'Access is denied.', None, 5)



26.20,9.79,3.32,24.87,751,0,0
sent..........A
```

Figure 5.4: Code Execution 1

```
127.0.0.1 - - [19/Dec/2024 14:10:00] "GET /current HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:00] "GET /enginetemp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:00] "GET /voltage HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:00] "GET /vibration HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:00] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:00] "GET /smoke HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:00] "GET /oillevel HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /enginetemp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /voltage HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /vibration HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /smoke HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /current HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /oillevel HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /enginetemp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /current HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /voltage HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /smoke HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /vibration HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:01] "GET /oillevel HTTP/1.1" 200 -

127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /enginetemp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /current HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /voltage HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /vibration HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /smoke HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /oillevel HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /enginetemp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /voltage HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /temp HTTP/1.1" 200 -
127.0.0.1 - - [19/Dec/2024 14:10:02] "GET /smoke HTTP/1.1" 200 -
```

Figure 5.5: Code Execution 2

5.6 Machine Learning Algorithm

In PDM for automobiles, the Random Forest algorithm is essential for predicting potential failures and maintenance needs based on sensor data. This ensemble learning approach makes several decision trees, each trained on a random subset in the provided data. The final prediction is made by combining the outputs from all trees, using the mode for classification or the average for regression. Random Forest is particularly effective at dealing with noisy data and complex relationships, making it an excellent choice for PDM, where sensor data can vary due to environmental conditions or wear and tear on components. Additionally, it is useful for estimating the Remaining Useful Life (RUL) of parts. By training on a labeled dataset that includes historical sensor data and maintenance records, the system is capable of generating accurate predictions about the health of vehicle components.

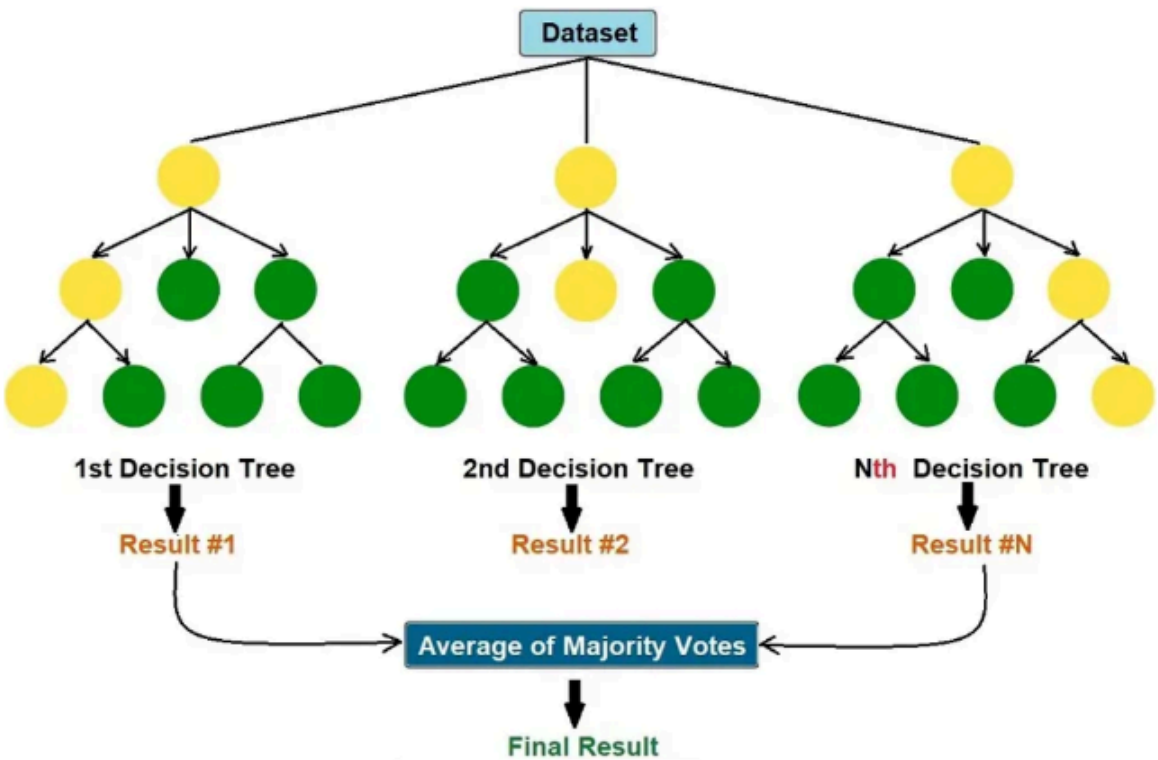


Figure 5.6: Tree Diagram of Machine Learning Model

Features:

- Ensemble Learning: This approach combines several decision trees to enhance accuracy and reduce the risk of overfitting.
- Classification and Regression: It is effective for both classifying faults and predicting Remaining Useful Life (RUL).
- Feature Importance: It highlights which features play an essential role in making predictions.
- Parallel Processing: Decision trees can be constructed simultaneously, leading to quicker.
- Scalability: The technique efficiently manages large datasets and supports real-time monitoring.

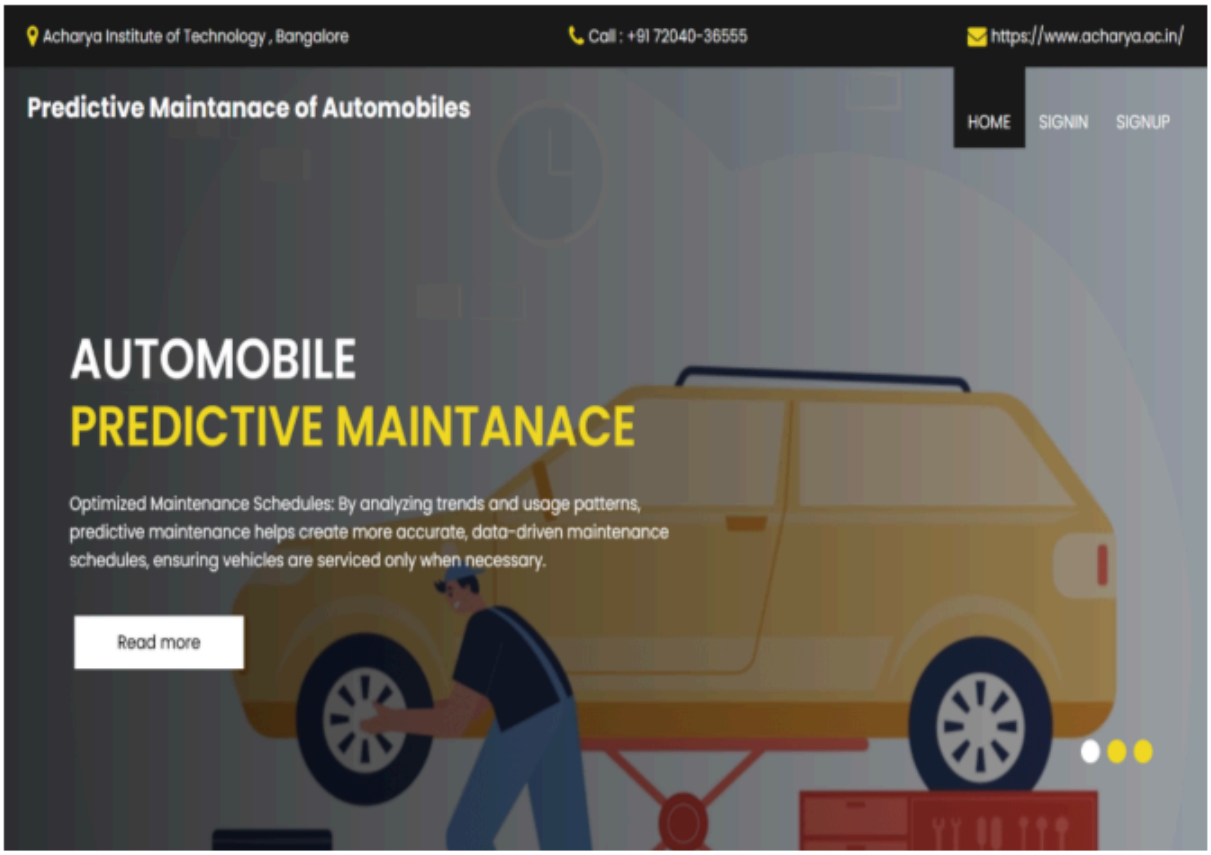


Figure 5.7: Captured web interface 1

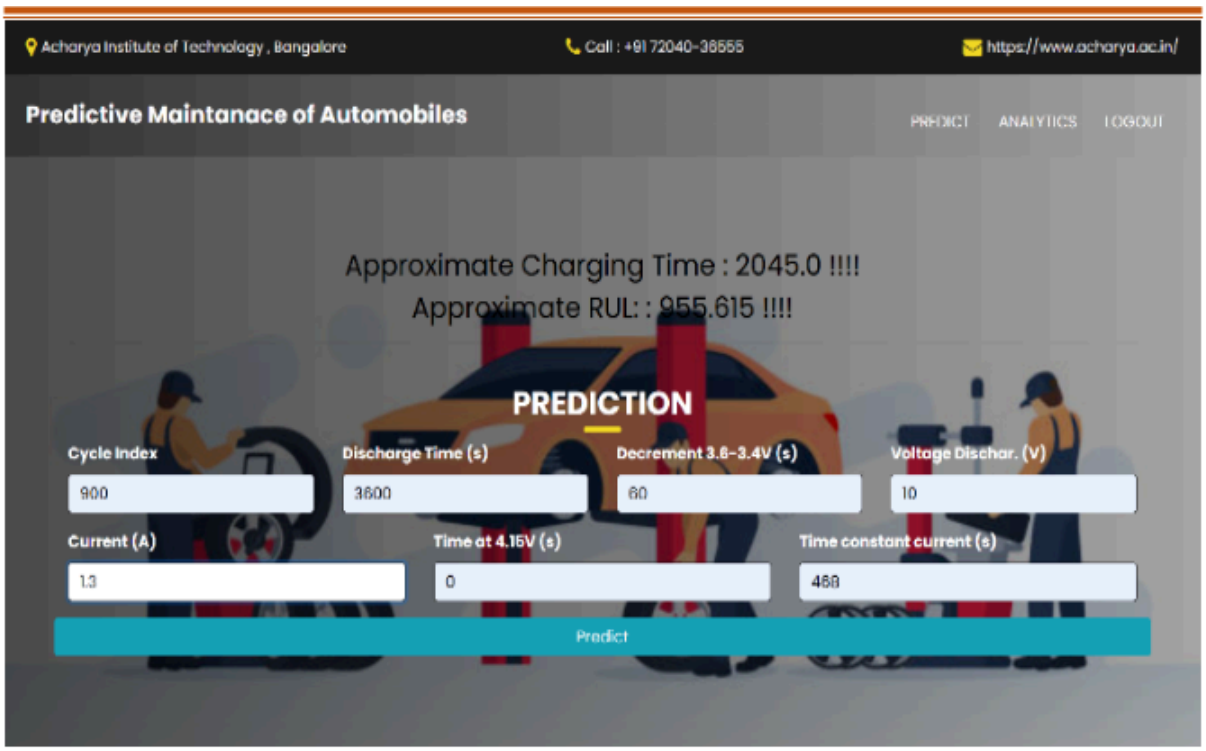
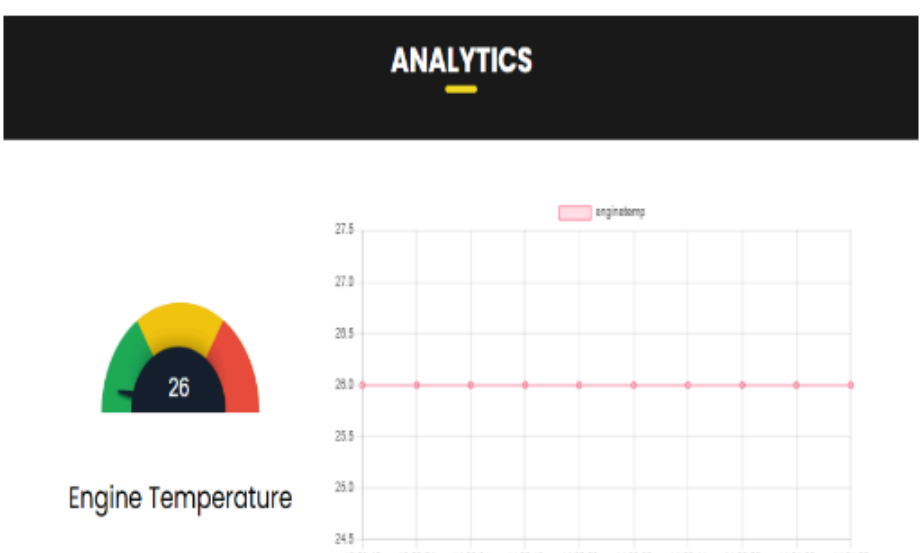


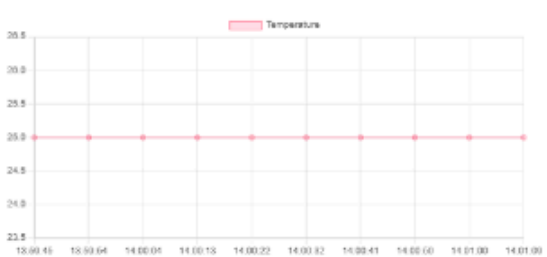
Figure 5.8: Captured web interface 2

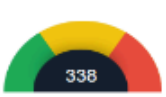


Volatge

Current

Battery Temperature





Smoke Level

Figure 5.9: Data Analytics of all Sensors

5.5 Telegram bot Notifications

Integrating a Telegram bot into the Predictive Maintenance System for Automobiles starts with the creation and setup of the bot. This is done using Bot Father on the Telegram app, where a new bot is created, and a unique Bot Token is generated to authenticate and interact with the Telegram API. The system gathers real-time data from various IoT sensors, including temperature, vibration, oil level, and motion sensors.

This data is processed by the ESP32 microcontroller, which sends the information to the back-end server using Zigbee or other communication protocols. The back-end server analyzes the incoming sensor data through threshold checks and machine learning models to detect potential faults or anomalies, such as high engine temperature, unusual vibrations, or low oil levels. When an issue is identified, an alert is generated.

The bot also facilitates two-way communication, enabling users to interact with the system. Users can issue commands like /status to get real-time sensor data or /history to access historical trends and logs. These commands are handled by the back-end, which returns the requested data to the user via the bot. Furthermore, for critical alerts, the bot may require users to acknowledge the notification by responding

with a specific message.

Real-time updates are maintained through either polling, where the system regularly checks for new alerts, or webhooks, which allow for instant communication between the server and the bot. Once deployed, the bot operates continuously on the server, monitoring sensor data and user interactions. The system can be refined over time by adjusting thresholds and enhancing the accuracy of the machine learning model.

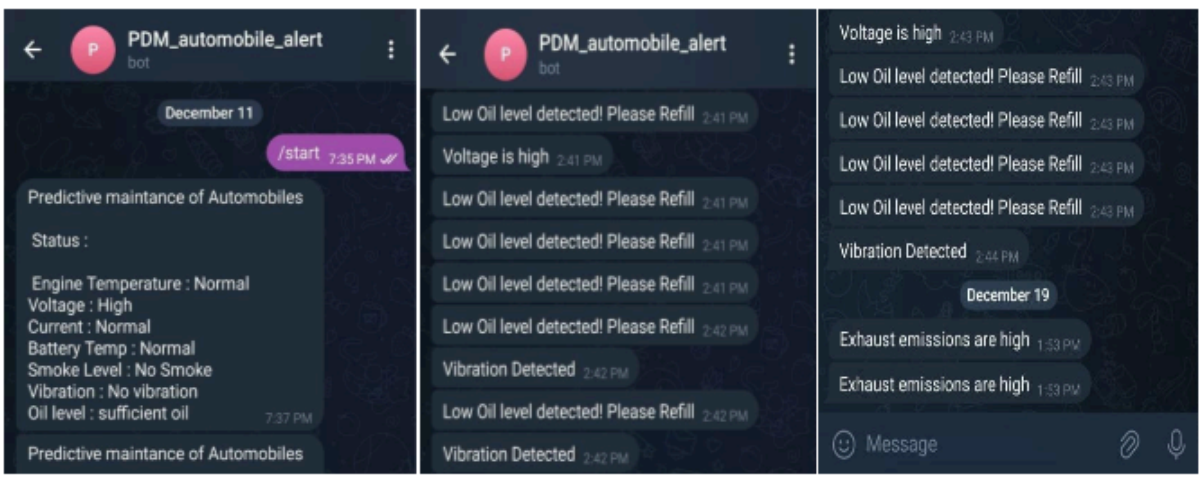


Figure 5.10: Telegram Bot for Data and Alerts

5.6 Advantages and Disadvantages

Advantages:

- Proactive Problem Solving: Identifies issues early on to avoid expensive and inconvenient vehicle breakdowns.

- Economic Benefits: Helps save money by minimizing unnecessary repairs and streamlining maintenance schedules.

- Safety First: Safeguards users by alerting them to critical issues like overheating and potential accidents before they worsen.

- Real-Time Monitoring: Zigbee-powered data transmission provides instant diagnostics, regardless of the vehicle's location.

- Versatile Applications: Functions smoothly for personal cars, commercial fleets, and industrial vehicles alike.

- Eco-Friendly Approach: Reduces waste by extending component lifespan and decreasing unwanted replacements.

Disadvantages:

Upfront Investment: The initial investment for sensors and infrastructure may put off potential users.

Connectivity Challenges: A reliable network connection is necessary, which may not be accessible in certain regions.

Technical Expertise Needed: The installation and maintenance processes require specialized knowledge in IoT and machine learning.

Component Maintenance: Sensors and systems need regular inspections, contributing to ongoing maintenance efforts.

CHAPTER: 06

FUTURE ENHANCEMENT

1. Smarter Models with Deep Learning Utilize LSTM networks to effectively analyze time-series data, enhancing the accuracy of anomaly predictions. Employ CNNs for recognizing spatial patterns, which improves the classification of faults in sensor readings. Incorporate adaptive learning features that enable the system to refine its predictions in real-time as it gathers more data.

2. Cloud intelligence to enhance scalability by shifting data processing and visualization to cloud platforms such as AWS, IoT Core, or Google Cloud IoT. This transition boosts computational power and scalability. Provide real-time global insights with advanced dashboards accessible from any device.

3. Expanding the Sensor Ecosystem Install sensors in vehicles to monitor tire pressure, fuel efficiency, and emissions, turning the system into a comprehensive maintenance solution. Choose weather-resistant and durable sensors to guarantee dependable performance in harsh weather conditions.

4. Mobile-First User Experience Create a user-friendly mobile app that enhances Telegram notifications with in-depth diagnostics, predictive reports, and service reminders. Allow for remote control and real-time updates, giving users the ability to manage their vehicle's health right from their devices.

5. Predictive Maintenance as a Subscription Transform the solution into a Predictive Maintenance as a Service (PMaaS) model, providing customized plans for both individual users and fleet operators. Incorporate premium features such as personalized analytics, tiered subscription options, and automated maintenance scheduling.

6. Fortified Data Privacy and Security Adopt strong encryption methods to protect user data during both transmission and storage. Ensure to build user trust and maintain data integrity.

7. Green Driving for a Sustainable Future Add features that track and minimize emissions, supporting environmental sustainability. Offer practical insights for eco-friendly driving, enhancing efficiency while lowering carbon footprints.

8. Fleet-Wide Impact at Scale Create centralized dashboards for fleet management, enabling efficient monitoring and control of multiple vehicles.

CHAPTER: 07

RESULTS

1. Seamless Real-Time Monitoring: Effectively tracked essential parameters such as temperature, vibrations, battery health, oil levels, and gas presence. Set up accurate anomaly thresholds for quick detection of potential problems.

2. Intelligent Predictive Maintenance: Utilized the Random Forest Algorithm to reliably predict maintenance requirements based on both real-time and historical data. Proactively notified users about risks like overheating or low battery levels with high accuracy. Estimate Remaining Useful Life (RUL) of components to assist users in scheduling maintenance more effectively.

3. Interactive Notification System: A Telegram bot provided immediate delivery of detailed alerts regarding faults or anomalies. Allowed users to interact with the system by checking real-time statuses and analyzing historical performance trends.

4. Reliable System Architecture: The ESP32 microcontroller efficiently handled data flow from various sensors while ensuring seamless transmission through Zigbee. A user-friendly LCD interface allowed drivers to quickly access essential vehicle metrics.

5. Enhanced Safety Protocols: It identified critical situations such as excessive vibrations and engine overheating, which helped reduce the chances of breakdowns. Vibration sensors effectively activated accident detection alerts, facilitating prompt responses during emergencies.

6. Optimized Costs and Efficiency: There were notable decreases in unexpected repairs and downtime, resulting in lower operational costs for users. Enhanced vehicle performance and reliability reduced disruptions, improving efficiency for both fleets and individual vehicles.

7. Robust Validation and Testing Machine learning models: underwent thorough testing on a variety of datasets, achieving high accuracy in fault and anomaly predictions. Sensor readings were validated against actual maintenance events, confirming the system's reliability in real-world applications.

8. Comprehensive Dashboard Insights: An intuitive dashboard was provided, displaying real-time sensor data, actionable alerts, and historical analysis. This improved decision-making by visualizing vehicle health trends and offering a clear overview of maintenance needs.

REFERENCES

[1] Kanawaday, Ameeth, and Aditya Sane. "Machine learning for predictive maintenance of industrial machines using IoT sensor data.", In 2017 8th IEEE international conference on software engineering and service science (ICSESS), pp. 87-90. IEEE, 2017..

[2] yvaz, Serkan, and Koray Alpay. "Predictive maintenance system for production lines in manufacturing: A machine learning approach using IoT data in real-time.", Expert Systems with Applications 173 (2021): 114598..

[3] Cakir, Mustafa, Mehmet Ali Guvenc, and Selcuk Mistikoglu. "The experimental application of popular machine learning algorithms on predictive maintenance and the design of IIoT based condition monitoring system.", Computers & Industrial Engineering 151 (2021): 106948..

[4] Kalusivalingam, A. K., Sharma, A., Patel, N., & Singh, V. (2020). Enhancing Predictive Maintenance in Manufacturing Using Machine Learning Algorithms and IoT-Driven Data Analytics., International Journal of AI and ML, 1(3)..

[5] C. V. S. M. D. S. Ravi Aravind, Machine Learning Applications in Predictive Maintenance for vehicles, International Journal Of Engineering And Computer Science, 11 November 2022.

[6] Predictive Maintenance – Bridging Artificial, 2021, Gerasimos G. Samatas, Seraphim S. Moumgiakmas, George A. Papakostas*.

[7] Chen, C., Liu, Y., Wang, S., Sun, X., Di Cairano-Gilfedder, C., Titmus, S. and Syntetos,, Advanced Engineering Informatics, 44, p.101054. 10.1016/j.aei.2020.101054, A.A., 2020. Predictive maintenance using cox proportional hazard deep learning..

[8] Y. L. Y. C. C. a. J. Z. Liang, Extracting topic-sensitive content from textual documents—A hybrid topic model approach. Engineering Applications of Artificial Intelligence,, 2018.

[9] C. L. Y. S. X. C.-G. D. T. S. Chen, Automobile maintenance modelling using gcForest., IEEE 16th International Conference on Automation Science and Engineering (CASE) 10.1109/CASE48305.2020.9216745, In 2020.

[10] M. A. M. D. A. V. P. Nithish Kanna J L Krishnakumar G, Predictive Maintenance of Motors using Machine Learning, Kumaraguru College, Issue 4 April 2024.