

**ADVANCED VEHICLE FIRE SAFETY AND
MONITORING WITH RAPID EMERGENCY
DISPATCH SOLUTIONS**

R24-058	
Dharmagunawardana W.M.P.I	IT21132346
Anthick G.N	IT21096266
Abeywardhana D.N	IT21133718
Peramunage A.N	IT21080562

Final Report

B.Sc. (Hons) Degree in Information Technology specializing in
Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

November 2024

**ADVANCED VEHICLE FIRE SAFETY AND
MONITORING WITH RAPID EMERGENCY
DISPATCH SOLUTIONS**

R24-058	
Dharmagunawardana W.M.P.I	IT21132346
Anthick G.N	IT21096266
Abeywardhana D.N	IT21133718
Peramunage A.N	IT21080562

Final Report

B.Sc. (Hons) Degree in Information Technology specializing in
Information Technology

Department of Information Technology



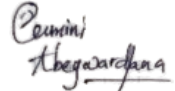

Sri Lanka Institute of Information Technology

Sri Lanka



November 2024

1.DECLARATION

We declare that this is our own work and this Report does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Name	Student ID	Signature
Dharmagunawardana W.M.P.I	IT21132346	
Anthick G.N	IT21096266	
Abeywardhana D.N	IT21133718	
Peramunage A.N	IT21080562	

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

	Date	Name	Signature
Supervisor:	21.11.2024	Mr. Nelum Chathuranga Amarasena	
Co-Supervisor:	21.11.2024	Mr. Deemantha Nayanajith Siriwardhana	

2.

3.ABSTRACT

Vehicle fires pose a critical threat to life and property, emphasizing the need for innovative, technology-driven solutions that enhance fire detection, suppression, severity assessment, and emergency response. This research presents a comprehensive AI-driven framework leveraging the integration of Machine Learning (ML), Deep Learning (DL), Internet of Things (IoT), and advanced routing algorithms to revolutionize vehicle fire safety. The proposed system encompasses four core functionalities: automatic fire detection and suppression, real-time fire severity assessment, predictive fire prevention, and optimized emergency response routing. The system is equipped with IoT-enabled sensors to monitor critical parameters such as smoke, temperature, flame intensity, and vehicle vibrations. These sensors provide real-time data, which is analyzed by ML algorithms to predict potential fire incidents and classify fire severity with high accuracy. This severity assessment is visualized through a mobile application, enabling real-time updates for vehicle occupants and emergency responders. Simultaneously, an automatic fire suppression mechanism, powered by solenoid valves, deploys CO₂ extinguishers to mitigate fire outbreaks promptly. To optimize emergency response, the system employs a hybrid AI model integrating Graph Neural Networks (GNNs) with A* and Dijkstra algorithms. This model dynamically processes real-time traffic data, vehicle telemetry, and road conditions to determine the most efficient routes for emergency vehicles, significantly reducing response times in congested urban environments. Simulations and initial testing demonstrate that this approach outperforms traditional static routing methods, enhancing fire department operational capabilities. By focusing on early detection, predictive prevention, real-time severity assessment, and rapid response, this research delivers a robust solution to mitigate fire risks, reduce property damage, and save lives. The system's commercialization potential includes partnerships with the automotive industry and emergency response organizations, setting new safety standards and fostering technological innovation. This integrated framework exemplifies the transformative potential of AI and IoT in addressing the challenges of vehicle fire incidents, paving the way for a safer, smarter future in urban emergency management.

Keywords: Vehicle Fire Safety, Fire Suppression, Fire Severity Assessment, Machine Learning (ML), Deep Learning (DL), Internet of Things (IoT), Real-time Detection, Emergency Response, AI-Driven Routing, Graph Neural Networks (GNNs).

4.ACKNOWLEDGMENT

I am expressing my deepest gratitude to all those who contributed to the successful completion of this project. I sincerely thank my academic institution, the Sri Lanka Institute of Information Technology (SLIIT), for providing me with the resources and a conducive environment to pursue this research. The support from the university has been instrumental in completing this project. I am profoundly grateful to the Fire Service Department of Colombo Municipal Council for their invaluable support and insights. Their expertise and guidance were crucial in understanding the practical aspects of fire safety, which greatly influenced the development of this project. My heartfelt gratitude goes to my supervisor, Mr. Nelum Chathuranga Amarasena, and co-supervisor, Mr. Deemantha Nayanajith Siriwardana, whose unwavering support, guidance, and encouragement have been the driving force behind this project. His expertise and feedback were vital in shaping the direction and outcome of my work. I would also like to express my gratitude to my external supervisor, Mr. Onray, for his insightful advice and external perspective, which added significant value to my research.

5.TABLE OF CONTENTS

DECLARATION.....	3
ABSTRACT	4
ACKNOWLEDGMENT	5
TABLE OF CONTENTS	6
LIST OF FIGURES	7
LIST OF TABLES	7
LIST OF ABBREVIATION.....	8
1. INTRODUCTION	1
1.1. Background and Literature Survey	1
1.2. Research Gap	4
1.3. Research Problem	6
1.4. Objectives	7
1.4.1. Main Objective	7
1.4.2. Specific Objectives	7
2. METHODOLOGY	7
2.1. System Architecture.....	7
2.1.1. Software methodology	7
2.1.2. Hardware Technologies	9
2.2. Commercialization of the Product	10
2.3. Testing and Implementation	11
2.3.1. Implementation	11
2.3.2. Testing	13
3. RESULTS AND DISCUSSIONS	14
3.1. Results	14
3.2. Research Findings	14
3.3. Discussion.....	14
4. CONCLUSIONS	15
5. REFERENCES	16
6. APPENDIX	18

6.LIST OF FIGURES

Figure 1:Component Architecture Diagram.....8

7.LIST OF TABLES

8.LIST OF ABBREVIATION

IoT – Internet of Things

CNN - Convolutional Neural Network

RNN - Recurrent Neural Networks

SVM - Support Vector Machine

ML – Machine Learning

GPS – Global Positioning System

IDE - Integrated Development Environment

CO – Carbon Monoxide

OS – Operating System

AWS - Amazon Web Services

UI - User Interface

9.INTRODUCTION

9.1.Background and Literature Survey

Vehicle fires pose a significant threat to both human life and property, with incidents occurring frequently across the globe. Statistics from the United States alone reveal that between 2003 and 2007, an average of 287,000 vehicle fires were reported annually, resulting in 480 deaths, 1,525 injuries, and \$1.3 billion in direct property damage(Design and implementation of fire detection and control system...). These alarming figures underscore the critical need for advanced fire detection and response systems in vehicles.

Traditional fire safety measures in vehicles primarily focus on fire prevention, minimization, and suppression. However, these systems often lack the capability to predict fire occurrences before they escalate, leading to potentially catastrophic outcomes. To address these limitations, recent advancements in technology have introduced the integration of machine learning, deep learning, and the Internet of Things (IoT) into fire detection and response mechanisms. These technologies offer the potential to predict fires before they occur, accurately assess fire severity, and automatically deploy fire suppression systems, thereby reducing the impact of vehicle fires.

The use of predictive algorithms, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Support Vector Machines (SVMs), has shown promise in accurately predicting fire occurrences based on data from temperature and smoke sensors. Additionally, the deployment of advanced routing algorithms, like A-star and Dijkstra's, enables the rapid identification of the nearest fire department and the quickest route for emergency response, further enhancing the overall effectiveness of vehicle fire safety systems(Improvement_of_Dijkstra) (Dijkstras_and_A-Star_in).

Numerous studies have explored the development and implementation of fire detection and control systems in vehicles, leveraging various technologies to improve safety and response times. A study by Sowah et al. (2016) presented a fuzzy logic-based multi-sensor fire detection system integrated with a real-time notification

system for vehicles. The system utilized an Arduino microcontroller to process inputs from multiple sensors, including temperature, smoke, and flame detectors, to detect and respond to fire incidents. The results demonstrated the system's ability to detect and extinguish fires within 20 seconds, highlighting its potential for real-world application in automotive safety(Design and implementing).

Another significant study focused on the design and implementation of a multi-stage early warning system for automobile fires. This system incorporated temperature, smoke, and light sensors, along with a microcontroller to process the data and trigger alarms at different stages of fire development. The multi-stage approach allowed for early detection and response, minimizing the risk of fire spreading and causing extensive damage(Research_on_Multi-Stage).

In the realm of predictive modeling, various machine learning algorithms have been employed to enhance fire detection accuracy. A research project demonstrated the effectiveness of CNNs in predicting vehicle fires, achieving an accuracy of 82.0%, surpassing other algorithms like RNNs and SVMs. This highlights the potential of deep learning techniques in developing robust fire prediction models that can prevent fires before they occur(Dijkstras_and_A-Star_in).

The integration of IoT with fire detection systems has also been explored to provide real-time monitoring and response capabilities. An IoT-based fire protection system was designed to monitor temperature and smoke levels in real-time, triggering automatic fire suppression systems upon detecting abnormal readings. The system's ability to connect with external devices and alert emergency services in the event of a fire further demonstrated its effectiveness in enhancing vehicle safety (Research_on_Multi-Stage).

Additionally, the application of advanced routing algorithms, such as A-star and Dijkstra's, has been investigated to optimize emergency response times. These algorithms were employed to determine the shortest and fastest routes to the incident location, ensuring that fire departments could respond promptly to vehicle fires. The research findings indicated that A-star provided faster response times compared to Dijkstra's, making it a more suitable choice for real-time emergency applications (Dijkstras_and_A-Star_in).

9.2. Research Gap

The research gap in the current literature on vehicle fire detection and suppression systems is multi-dimensional, highlighting several areas where existing studies fall short of providing a comprehensive solution. While numerous studies have explored fire detection using various sensor networks, such as temperature, smoke, and flame sensors, these systems predominantly focus on either detection or basic suppression without leveraging the full potential of advanced technologies like deep learning and IoT. For example, existing research often employs fuzzy logic or simple decision-making algorithms to trigger fire suppression mechanisms (Design and implementation) (Research_on_Multi-Stage). However, these methods are limited in their predictive capabilities and may not effectively preempt fire incidents, especially in complex and dynamic environments like vehicles.

Another significant gap is the limited use of advanced machine learning algorithms in predicting vehicle fires. While some studies have utilized neural networks or other machine learning techniques for fire detection (Design and implementation), there is a lack of research that systematically compares different algorithms, such as CNNs, RNNs, and SVMs, to determine the most effective model for accurate fire prediction in vehicles. Furthermore, these studies often do not account for the integration of real-time data from IoT-enabled sensors, which could significantly enhance the system's predictive accuracy and response speed.

Additionally, while there are studies that address fire suppression through automated systems, such as the deployment of carbon dioxide or other extinguishing agents (Design and implementation), these solutions typically do not include a robust mechanism for assessing the severity of the fire or tailoring the response accordingly. The ability to measure fire severity and type, and then deploy the appropriate level of suppression, is crucial for minimizing damage and ensuring safety, yet this aspect is often overlooked in the literature.

Moreover, current research largely neglects the integration of emergency response optimization into vehicle fire management systems. While route optimization algorithms like A-star and Dijkstra have been widely studied in other contexts, their application in rapidly identifying the nearest fire department and providing optimal

routes for emergency responders in vehicle fire scenarios remains underexplored (Dijkstras_and_A-Star_in...). This is a critical oversight, as the timeliness of emergency response can significantly impact the outcome of a fire incident.

Finally, the existing body of research tends to treat the various components of vehicle fire detection, severity assessment, suppression, and emergency response as isolated systems. There is a clear need for a holistic approach that integrates these elements into a seamless, real-time operational framework. By combining advanced machine learning algorithms, IoT sensor networks, automated fire suppression technologies, and optimized emergency response systems, it is possible to develop a more comprehensive and effective solution for vehicle fire management. This research aims to fill these gaps by proposing a unified system that not only predicts and suppresses fires but also optimizes the entire response process, from detection to emergency intervention.

9.3.Research Problem

Vehicle fires present a significant safety and financial concern worldwide, with thousands of incidents occurring daily. Despite advances in automotive technology, the ability to predict, detect, and respond to vehicle fires in a timely and efficient manner remains a challenge. Current fire detection systems in vehicles are often reactive rather than proactive, relying on basic sensors and simple logic to trigger alarms and deploy fire suppression systems only after a fire has already begun to spread. This delay in detection and response can lead to catastrophic outcomes, including loss of life, severe injuries, and extensive property damage.

Existing systems typically focus on isolated aspects of fire management, such as detection or suppression, without integrating advanced technologies that could enhance the overall effectiveness of the response. For instance, while some systems utilize basic temperature and smoke sensors, they do not leverage the predictive capabilities of advanced machine learning algorithms, which could foresee a fire

before it fully ignites. Additionally, there is often no assessment of the fire's severity, which is crucial for determining the appropriate level of response. Furthermore, once a fire is detected, the current systems do not efficiently optimize the response route for emergency services, potentially delaying critical intervention.

To address these gaps, this research aims to enhance vehicle fire prediction and response mechanisms by integrating cutting-edge technologies such as machine learning, deep learning, and the Internet of Things (IoT). The research problem focuses on the development of a comprehensive system that not only predicts the occurrence of vehicle fires using CNN, RNN, and SVM algorithms but also detects the type and severity of the fire with high accuracy. The system will autonomously deploy fire suppression measures using an array of sensors and microcontrollers, and identify the nearest fire department using optimized routing algorithms like A-star and Dijkstra. This integrated approach is designed to improve the speed and effectiveness of the response, minimizing damage and enhancing safety.

The challenge lies in effectively combining these technologies into a cohesive system that operates in real-time, ensuring that vehicle fires are detected and mitigated before they escalate. The research problem thus seeks to explore the potential of advanced predictive models, sensor integration, and intelligent routing to create a robust vehicle fire management system that significantly improves current standards of vehicle safety and emergency response.

9.4.Objectives

9.4.1. Main Objective

The primary objective of this research is to develop an advanced and integrated vehicle fire prediction and response system that leverages cutting-edge technologies in machine learning, deep learning, and the Internet of Things (IoT) to address the critical issue of vehicle fires. Vehicle fires pose a significant threat to both

human safety and property, leading to substantial financial losses and, in severe cases, fatalities. Despite existing fire detection systems, current methodologies often fall short of providing timely, accurate, and comprehensive fire management solutions. These traditional systems typically react to fires once they have already begun, offering limited predictive capabilities and lacking in the ability to assess the severity of a fire or optimize the response effectively. This research aims to transcend these limitations by creating a proactive, intelligent, and interconnected system that not only detects fires but also predicts and mitigates them before they escalate.

At the core of this research is the development of a predictive model that utilizes advanced machine learning algorithms, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Support Vector Machines (SVMs). These algorithms will process data from various sensors, such as temperature and smoke sensors, to identify patterns and anomalies that may indicate an impending fire. The objective is to implement a model that can accurately predict the occurrence of a fire with a high degree of precision, thereby enabling preventive measures to be taken before the fire even begins. This proactive approach is designed to significantly reduce the incidence of vehicle fires, prevent potential damage, and enhance overall vehicle safety.

In addition to predicting fires, the research aims to develop a system that can detect the type and severity of a fire in real-time. Utilizing SVM and Linear Regression algorithms, the system will be capable of classifying the fire and measuring its intensity. This capability is crucial for determining the appropriate response, as different types and severities of fires require different levels of intervention. For instance, a minor fire might only need a localized suppression effort, whereas a more severe fire could necessitate a broader, more aggressive response. By accurately assessing the fire's severity, the system ensures that resources are used efficiently and that the response is proportionate to the threat, thereby minimizing damage and risk to human life.

The research also focuses on the development of an automated fire suppression system. This system will integrate various sensors, such as the MAX6675 Module with K Type Thermocouple for temperature detection and the MQ2 sensor for smoke

detection, controlled by an Arduino MEGA microcontroller. The system will be capable of deploying fire extinguishers automatically via solenoid valves, without the need for human intervention. The integration of IoT technology through the ESP32 microcontroller will enable the system to operate seamlessly, ensuring that the fire is contained and extinguished swiftly, thereby preventing it from spreading and causing further damage.

The research aims to enhance the emergency response by implementing advanced route optimization algorithms, namely A-star and Dijkstra. These algorithms will be used to identify the nearest fire department and provide real-time optimal routes for emergency responders, significantly reducing the time it takes for them to reach the scene. This component of the system is designed to facilitate a rapid and effective response, which is critical in fire situations where every second counts. The optimized routing will be integrated into a web dashboard accessible to local fire departments, allowing them to monitor incidents and respond efficiently.

9.4.2. Specific Objectives

Develop a Predictive Model for Vehicle Fire Occurrence:

Implement an advanced predictive model using machine learning algorithms such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Support Vector Machines (SVMs). The goal is to analyze data from temperature and smoke sensors to identify patterns indicative of potential fire incidents. This model will aim to achieve high accuracy in predicting fires before they ignite, allowing for preventive measures to be taken. The CNN model, having shown the highest accuracy of 82.0%, will be a focal point of this development.

Classify Fire Type and Measure Severity:

Develop a system that accurately classifies the type of fire and assesses its severity in real-time using Support Vector Machine (SVM) and Linear Regression algorithms. The system will process data from various sensors to categorize the fire and measure its intensity. The objective is to determine the appropriate response level based on the severity of the fire, ensuring that the resources deployed are proportionate

to the threat. This system will particularly rely on the Linear Regression algorithm, which has achieved an accuracy of 92.49%.

Automate Fire Suppression Deployment:

Design and implement an automated fire suppression system that integrates various sensors, including the MAX6675 Module with K Type Thermocouple for temperature detection and the MQ2 sensor for smoke detection. This system, controlled by an Arduino MEGA microcontroller and supported by ESP32 for IoT connectivity, will automatically deploy fire extinguishers through solenoid valves. The objective is to ensure rapid containment and extinguishment of fires without the need for human intervention, minimizing the risk of fire spreading.

Enhance Emergency Response through Optimized Routing:

Implement advanced route optimization algorithms, namely A-star and Dijkstra, to identify the nearest fire department and determine the fastest route for emergency responders. This system will be integrated into a web dashboard provided to local fire departments, allowing them to monitor incidents in real-time and respond more effectively. The objective is to reduce the response time in fire emergencies, ensuring that help arrives as quickly as possible.

Develop a User Interface for Real-Time Monitoring and Alerts:

Create a mobile application and a web-based dashboard that allow users to monitor multiple vehicles, receive real-time alerts about potential fire incidents, and track the status of ongoing events. This interface will ensure that vehicle owners are informed promptly and can take necessary actions. The objective is to provide an easy-to-use platform for monitoring vehicle safety, enhancing user awareness, and contributing to overall fire prevention efforts.

Integrate All Components into a Cohesive System:

Bring together all the developed components—predictive modeling, fire classification, automated suppression, optimized routing, and user interface—into a single, cohesive system. This integration will ensure that each part functions seamlessly with the others, providing a comprehensive and unified solution for vehicle fire management. The objective is to create a system that operates in real time, enhancing the overall effectiveness of fire prediction, detection, and response in vehicles.

10. METHODOLOGY

10.1. System Architecture

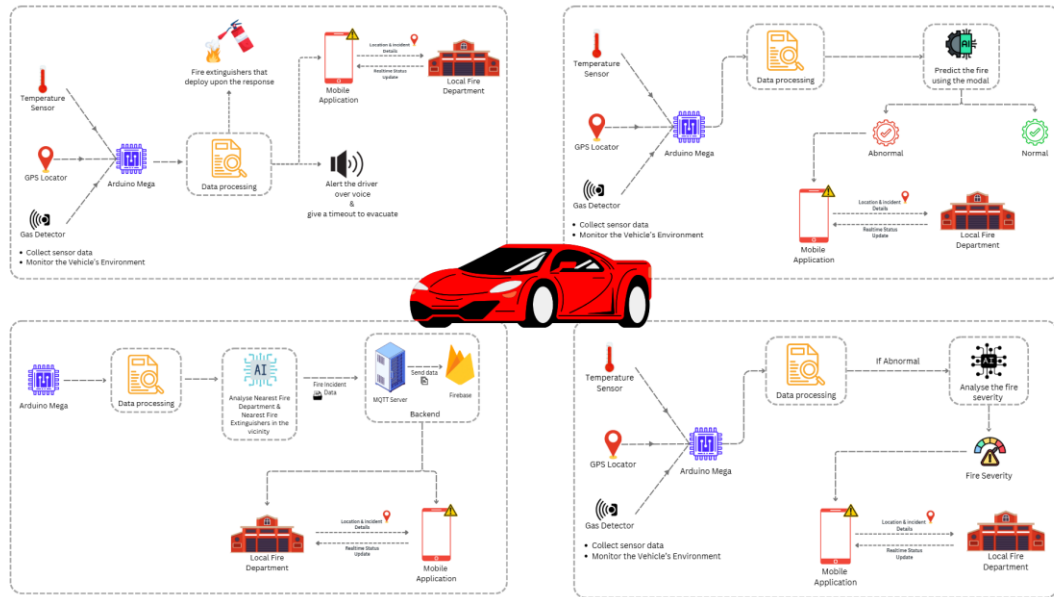


Figure 1 System Architecture Diagram

10.1.1. Software methodology

Requirement Gathering: Clearly define the goals of the safety system, including fire detection, toxic gas leak identification, and emergency alerting capabilities. And clearly identify the requirements of the mobile application and the proposed IoT system. Engage with potential users, emergency response teams, and automotive experts to gather insights and expectations.

Market Research: Analyze existing solutions and identify gaps between the products that can focus on the unique value of the proposed system.

Use Case Scenarios: Develop use case scenarios and personas to guide the design process and ensure the system meets varied user needs and fine-tune the requirements better.

System Architecture: Design the overall system architecture, specifying how sensors, data processing units, and communication systems interact. And design the

component-wise System Architectures to identify the components properly and clearly. Then it will be easy to identify which component connects which.

Technology Selection: Choose appropriate technologies, including sensors (smoke, gas, thermal sensor, fire sensor), processing units (Arduino MEGA, ESP 32 module), and software tools (Arduino Ide, Firebase or AWS IoT, React Native or Flutter,). These technologies may vary as the project goes on.

Software Development: Write the initial code for data gathering, processing, and emergency alerting, using MicroPython within the Anaconda and Jupyter Notebook IDE. The target is to program the Hardware components to work properly. A mobile application will be developed to provide the best experience to the users. React Native-A framework for developing cross-platform mobile applications on Android and iOS. It aids in the development of responsive, nearly native applications that talk to the fire detection system to let the user know what is happening in real time.

Integration: Integrate hardware and software components to create a functional prototype for initial testing.

System Testing: Test individual components (sensors, software modules) for functionality and reliability. Test the integrated system to ensure seamless operation and data flow between components. Conduct testing with potential users to gather feedback on system usability, effectiveness, and any adjustments needed. Repeat testing cycles to ensure all refinements are effectively implemented and new issues are addressed.

Deployment and Commercialization: Implement the system in a limited, controlled environment to validate its functionality in real-world scenarios. Prepare for larger-scale production, ensuring the quality and reliability of hardware components and officially launch the product to the market.

This tour ensures a thorough approach from concept to deployment and beyond by embracing the automotive safety system development lifecycle.

10.1.2. Hardware Technologies

Microcontroller and Embedded Systems:

Arduino MEGA Microcontroller: Central processing unit for handling sensor data, executing control logic, and managing communication between components.

Micro controller: ESP32 Microcontroller with Wi-Fi and Bluetooth connectivity, supporting real-time data processing as well

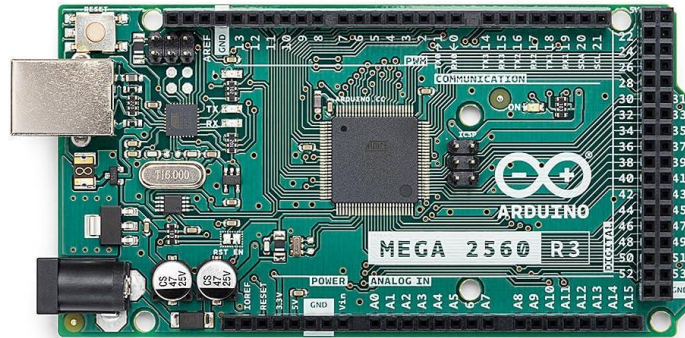


Figure 2 Arduino Compatible Mega 2560



Figure 3 ESP 32 micro controller

Sensors and Hardware Components:

- Temperature Sensor (MAX6675): Monitors ambient temperature to detect potential fire conditions. Digital temperature sensors providing accurate readings.

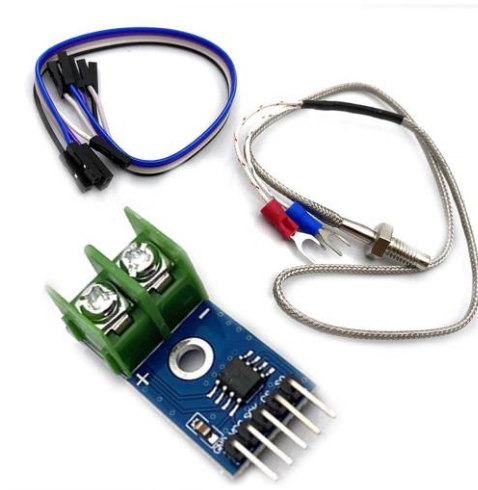


Figure 4 MAX6675 Module with K Type Thermocouple

- Gas Detector (MQ-2): Detects smoke or hazardous gases that could indicate a fire. Semiconductor sensors that detect a range of gases and send analog signals to the Arduino MEGA microcontroller.



Figure 5 MQ-2 LPG CH4 CO H2 Alcohol Smoke Butane Methane Propane Gas Detection Module

- GPS Module: Neo 6M detect the location of the vehicle, so that it can be sent to the fire department.



Figure 6 NEO-6M GPS Module with Active Antenna\

- 5VDC relay module: to release the solenoid valve to release the fire extinguishers.



Figure 7 5VDC 1 Way 1 Channel Relay Module

- Solenoid valve to lock and release the fire extinguisher mechanism.



Figure 8 Electric Solenoid Valve 0.25-inch 12VDC NC Metal Water S179



Figure 9 Dry powder fire extinguishers

Communication and Networking:

- Wi-Fi Modules (ESP32): Enables wireless communication for remote monitoring and control. Built-in Wi-Fi capability within ESP32 to connect to cloud services and mobile apps.
- Bluetooth (integrated in ESP32): Local communication with mobile devices for alerts and controls. Bluetooth Low Energy (BLE) for power-efficient communication.

Mobile Application Development:

- **React Native:** Cross-platform mobile application development for Android and iOS. Allows the creation of responsive, native-like mobile apps that interface with the fire detection system for real-time alerts.

10.2. Commercialization of the Product

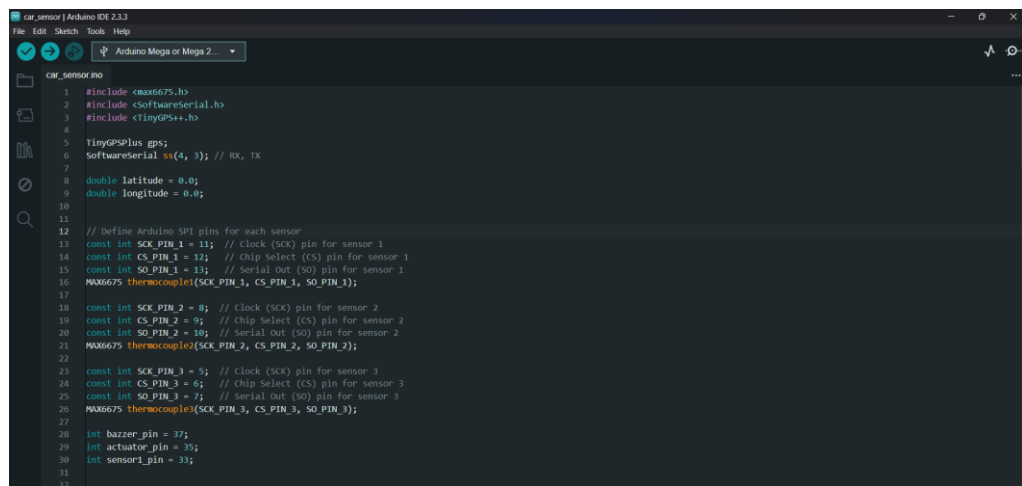
1. **Patent Protection:** Make sure the product has sufficient protection against theft before releasing it onto the market. Getting patents for this novel idea may provide a competitive advantage and is essential to attract partners and investors.
2. **Pilot Programs:** For the deployment of pilot programs into action, work with fleet managers or manufacturers. This enables to get valuable feedback, show the efficiency and dependability of the system in real-world settings, and make any adjustments to the product prior to a launch.
3. **Strategic Partnerships:** Establish partnerships with automakers, suppliers of emergency response equipment, or insurers. Collaboration with such partners may bring beneficial outcomes including access to established customers, and chances for co-branding, advertising, and distribution.
4. **Direct Sales to Customers:** To sell directly to customers, make use of online platforms and e-commerce platforms. This approach may work especially well if the system is made for aftermarket installation. Sales may be increased by educating potential consumers.
5. **Government and Regulatory Approvals:** Seek certifications and approvals from pertinent regulatory and government agencies. In addition to giving this product more legitimacy, this might lead to customers for required installations in specific areas, which would greatly increase the number of potential consumers we could reach.
6. **Trade exhibitions and Expos:** To introduce this product to a larger audience, take part in safety conferences, technology expos, and car trade exhibits.

10.3. Testing and Implementation

10.3.1. Implementation

Hardware Integration:

The first step involves the integration of all hardware components, including the MAX6675 module with the K-type thermocouple sensor, MQ2 smoke sensor, solenoid valve, Ublox NEO-6M GPS Module and Arduino MEGA microcontroller. Each component is connected according to the designed circuit diagrams, with the sensors feeding data into the Arduino Mega, which processes the information and triggers the solenoid valve in the event of a fire.



```
car_sensor.ino
1 #include <max6675.h>
2 #include <SoftwareSerial.h>
3 #include <TinyGPS++.h>
4
5 TinyGPSPlus gps;
6 SoftwareSerial ss(4, 3); // RX, TX
7
8 double latitude = 0.0;
9 double longitude = 0.0;
10
11
12 // Define Arduino SPI pins for each sensor
13 const int SCK_PIN_1 = 11; // Clock (SCK) pin for sensor 1
14 const int CS_PIN_1 = 12; // Chip Select (CS) pin for sensor 1
15 const int SO_PIN_1 = 13; // Serial Out (SO) pin for sensor 1
16 MAX6675 thermocouple1(SCK_PIN_1, CS_PIN_1, SO_PIN_1);
17
18 const int SCK_PIN_2 = 8; // Clock (SCK) pin for sensor 2
19 const int CS_PIN_2 = 9; // Chip Select (CS) pin for sensor 2
20 const int SO_PIN_2 = 10; // Serial Out (SO) pin for sensor 2
21 MAX6675 thermocouple2(SCK_PIN_2, CS_PIN_2, SO_PIN_2);
22
23 const int SCK_PIN_3 = 5; // Clock (SCK) pin for sensor 3
24 const int CS_PIN_3 = 6; // Chip Select (CS) pin for sensor 3
25 const int SO_PIN_3 = 7; // Serial Out (SO) pin for sensor 3
26 MAX6675 thermocouple3(SCK_PIN_3, CS_PIN_3, SO_PIN_3);
27
28 int buzzer_pin = 37;
29 int actuator_pin = 35;
30 int sensor1_pin = 33;
31
32
```

Figure 10 Import necessary libraries and Define Arduino SPI pins for each sensors

Software Development:

The software component involves writing code for the Arduino Mega to handle data acquisition from the sensors, processing the data, and determining the appropriate response. The code also includes communication protocols to send alerts to the MQTT server, mobile application, and fire department.

sensor	Front			Mid			Back			Front			Mid				
	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3			
26.87	26.87	35.95	29.37	29.5	29.25	29.97	29.45	34.91	25.15	29.13	28.99	46.58	37.4	39.76	25.62	29.67	29.15
31.05	27.5	36.84	29.37	29.25	35.62	30.51	30.84	36.23	27.5	29.13	25.35	43.43	42.54	44.16	26.28	36.32	28.92
32.86	34.14	26.19	32.47	32.48	29.17	32.37	31.4	37.45	25.15	26.45	26.45	39.88	34.14	40.19	33.2	36.06	25.21
35.53	29.08	36.84	30.34	29.84	39.08	29.97	29.45	34.91	27.84	27.68	25.14	43.51	47.77	43.52	26.91	38.95	30
28.59	27.4	34.14	32.57	36.39	35.02	33.14	26.2	38.99	29.32	28.79	25.32	38.5	44.25	43.55	29.96	28.4	29.44
28.83	35.63	36.8	34.17	34.78	31.82	38.98	26.45	31.54	25.33	26.58	26.72	48.93	49.72	38.21	29.26	31.41	30.64
26.84	27.37	27.18	34.39	33.3	33.68	38.98	30.17	32.16	25	26.58	25.91	49.34	39.68	39.51	25.81	30.87	31.31
29.93	36.52	33.11	35.19	34.99	29.69	33.93	33.92	32.68	28.73	26.76	25.36	40.54	43.85	41.86	27.64	26.54	28.53
29.71	38.2	35.67	33.29	34.99	36.8	37.03	26.11	32.71	27.74	27.63	27.84	40.96	44.01	44.81	25.94	25.12	31.17
27.85	29.14	39.84	31.11	30.05	30.17	37.67	32.32	31.85	27.51	27.35	26.6	42.81	39.21	49.87	26.37	29.87	30.49
31.39	36.31	37.57	36.8	36.59	34.14	32.63	30.97	29.26	29.64	25.13	26.48	45.8	38.39	44.09	26.43	28.55	26.67
31.37	35.14	33.07	34.48	33.96	32.65	32.93	36.87	36.54	27.01	26.43	26.79	44.75	46.68	38.8	29.31	26.15	25.86
32.6	32.43	33.85	34.62	29.25	30.94	35.94	35.07	39.03	29.95	25.8	26.21	44.81	39.79	38.29	30.64	31	27.29
35.98	33.1	26.86	33.97	30.39	33.55	37.35	33.3	36.26	29.03	29.57	25.26	44.2	38.57	47.74	25.26	25.52	30.89
34.17	39.58	28.46	33.88	36.04	35.77	28.71	27.28	37.88	28.53	26.13	29.49	41.54	38.06	42.36	25.14	31.9	27.8
31.8	37.49	34.91	36.51	31.14	29.06	35.36	34.87	33.23	26.42	27.82	26.9	43.1	46.17	43.96	29.37	28.86	31.37
31.05	33.19	38.1	31.09	32.73	35.68	28.73	29.77	35.17	25.55	29.62	27.81	38.95	37.75	38.93	29.36	28.64	31.87
35.1	31.09	32.73	31.09	32.73	35.68	28.73	29.77	35.17	25.55	29.62	27.81	38.95	37.75	38.93	29.36	28.64	31.87
26.05	29.32	33.06	33.92	31.22	33.46	37.01	26.37	30.22	25.51	27.34	28.24	39.01	37.79	38.96	26.79	26.25	29.59
39.57	30.56	28.21	36.78	30.57	32.71	37.98	32.4	39.15	25.83	27.25	26.85	39.04	37.83	39.00	27.07	29.62	27.89
34.83	35.37	34	35.88	35.4	34.65	35.94	35.07	39.03	29.95	25.8	26.21	38.83	37.62	38.70	30.64	31	27.29
27.78	36.27	27.31	33.2	32.55	30.11	32.27	28.24	29.12	29.39	28.86	26.71	39.50	38.56	39.49	26.55	25.06	25.86
34.88	31.69	35.69	30.48	33.48	35.29	31.23	39.26	34.24	27.85	26.99	29.56	39.52	38.58	39.53	30.23	25.22	28.09
39.15	29.77	36.96	31.71	32.21	34.07	31.34	29.58	30.64	25.45	27.88	29.68	42.97	38.29	40.48	29.79	31.37	31.32
38.47	31.17	30.36	31.15	31.12	31.16	31.05	31.95	31.64	25.46	25.13	26.67	40.1	43.15	39.57	39.15	25.87	27.61
37.34	26.8	27.87	36.34	31.31	31.02	31.05	31.95	31.64	25.47	27.61	26.62	40.83	46.68	38.76	31.5	25.47	29.77
34.71	36.36	31.15	31.15	31.17	31.18	31.05	31.95	31.64	25.46	25.13	26.67	40.1	43.15	39.57	39.15	25.87	27.61
Sheet1																	

Figure 11 Collected Data From Axio 2015

Back			Front			Mid			Back			Front			Mid		
sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3
27.84	26.5	33.94	31	32.84	34.59	35.71	33.19	38.63	25	26.58	25.91	38.05	37.09	38.02	25.26	25.52	30.89
26.5	27.84	26.5	31.5	36.23	35.62	30.31	30.84	36.23	25.33	26.58	26.72	38.11	37.17	38.10	25.14	31.9	27.8
31.05	27.5	36.84	32.47	32.48	33.89	30.37	33.6	37.65	25.35	26.45	28.45	38.13	37.20	38.14	28.32	28.86	31.37
32.36	36.34	26.39	30.34	29.84	29.08	29.97	29.45	34.91	27.84	27.68	25.14	38.25	37.23	38.16	26.13	27.72	30.15
34.52	39.15	27.43	32.57	36.39	35.02	33.14	26.2	38.99	29.32	28.79	25.32	38.27	37.28	38.22	26.17	29.06	27.81
30.08	35.56	28.96	34.17	34.78	31.92	38.98	26.15	31.54	25.33	26.58	26.72	38.29	37.37	38.24	26.79	26.25	29.59
32.5	29.57	32.08	34.39	33.3	33.68	38.98	39.17	32.16	25	26.58	25.91	38.32	37.40	38.29	27.07	29.62	27.89
29.15	38.39	38.46	35.19	34.99	29.69	33.93	39.92	32.68	28.73	26.76	25.36	38.43	37.49	38.31	27.25	28.93	26.82
35.53	29.08	36.84	33.29	34.99	36.8	37.03	26.11	32.71	27.74	27.63	27.84	38.60	37.50	38.38	27.15	31.8	25.84
28.59	27.4	34.14	31.11	30.05	30.17	37.67	32.32	31.85	27.51	27.35	26.6	38.67	37.53	38.39	29.36	28.64	31.87
28.83	35.63	30.6	36.8	36.59	34.14	32.63	30.97	29.26	29.64	25.13	26.48	38.78	37.57	38.54	26.43	28.55	26.67
26.84	27.37	27.18	34.48	33.96	32.65	32.93	36.87	36.54	27.01	26.63	28.79	38.80	37.61	38.66	29.31	26.15	25.86
29.93	36.52	33.11	35.19	34.99	29.69	33.93	33.92	32.68	28.73	26.76	25.36	38.43	37.49	38.31	27.25	28.93	26.82
29.71	38.2	35.67	33.97	30.39	33.55	37.35	33.3	36.26	29.03	29.57	25.26	38.83	37.65	38.72	25.26	25.52	30.89
27.85	29.14	39.84	33.88	36.04	35.77	28.71	27.28	37.88	28.53	26.13	29.49	38.90	37.68	38.83	25.14	31.9	27.8
31.39	36.31	37.57	36.51	31.14	29.06	35.36	34.87	33.23	26.42	27.82	26.9	38.96	37.72	38.89	27.15	31.8	25.84
31.37	35.14	33.07	36.58	33.58	31.72	39.94	29.75	31.71	25.55	29.62	27.81	38.95	37.75	38.93	29.36	28.64	31.87
32.6	32.43	33.85	33.92	31.22	33.46	37.01	26.37	30.22	25.51	27.34	28.24	39.01	37.79	38.96	26.79	26.25	29.59
35.98	33.1	26.86	36.78	30.57	32.71	37.98	32.4	39.15	25.83	27.25	26.85	39.04	37.83	39.00	27.07	29.62	27.89
34.17	39.58	28.46	33.88	36.04	35.77	28.71	27.28	37.88	28.53	26.13	29.49	38.90	37.68	38.83	25.14	31.9	27.8
31.8	37.49	34.91	36.51	31.14	29.06	35.36	34.87	33.23	26.42	27.82	26.9	38.96	37.72	38.89	27.15	31.8	25.84
31.05	33.19	38.1	31.09	32.73	35.68	28.73	29.77	35.17	25.55	29.62	27.81	38.95	37.75	38.93	29.36	28.64	31.87
26.05	29.32	33.06	33.92	31.22	33.46	37.01	26.37	30.22	25.51	27.34	28.24	39.01	37.79	38.96	26.79	26.25	29.59
39.57	30.56	28.21	36.78	30.57	32.71	37.98	32.4	39.15	25.83	27.25	26.85	39.04	37.83	39.00	27.07	29.62	27.89
34.83	35.37	34	35.88	35.4	34.65	35.94	35.07	39.03	29.95	25.8	26.21	38.83	37.62	38.70	30.64	31	27.29
27.78	36.27	27.31	33.2	32.55	30.11	32.27	28.24	29.12	29.39	28.86	26.71	39.50	38.56	39.49	26.55	25.06	25.86
34.88	31.69	35.69	30.48	33.48	35.29	31.23	39.26	34.24	27.85	26.99	29.56	39.52	38.58	39.53	30.23	25.22	28.09
39.15	29.77	36.96	31.71	32.21	34.07	31.34	29.58	30.64	25.45	27.88	29.68	39.60	38.61	39.59	29.79	31.37	31.32
38.47	31.17	30.36	31.15	31.12	31.16	31.05	31.95	31.64	25.46	25.13	26.67	40.1	43.15	39.57	39.15	25.87	27.61
Sheet1																	

Figure 12 Collected Data From Toyota Corolla 121

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
Engine Off	Engine Running												Engine Off											
Back	Front						Mid			Back			Front						Mid					
sensor 1	sensor 2	sensor 3	sensor 1			sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3		
32	35.18	33.04	33.97	30.39	33.55	29.35	35.93	29.71	25.96	25.1	28.23	41.54	38.08	42.36	25.26	25.52	30.1	35.18	33.04	33.97	30.39	33.55	29.35	
26.26	32.45	36.26	33.88	36.04	35.77	30.11	32.58	37.86	27.37	27.34	28.27	43.3	48.17	43.43	25.14	31.9	27	36.26	33.88	36.04	35.77	30.11	32.58	
30.13	33.09	31.52	36.51	31.14	29.06	37.79	36.71	32.36	29.16	28.16	26.47	41.13	45.28	47.59	28.32	28.86	31	31.52	36.51	31.14	29.06	37.79	36.71	
36.81	36.68	34.35	36.58	31.58	31.72	35.44	28.55	30.82	29.39	28.86	26.71	42.89	38.82	40.5	26.13	27.22	30	34.35	36.58	31.58	31.72	35.44	28.55	
39.16	35.82	36.35	31.09	32.73	35.68	38.08	34.27	37.6	27.85	26.99	29.56	45.14	43.47	39.3	26.17	29.96	27	36.35	31.09	32.73	35.68	38.08	34.27	
37.34	33.99	31.19	33.92	31.22	33.46	32.27	28.24	29.12	25.45	27.88	29.68	43.16	42.77	42.63	26.79	26.25	29	31.19	33.92	31.22	33.46	32.27	28.24	
32.74	27.76	33.99	36.78	30.57	32.71	31.23	39.26	34.24	29.08	27.89	26.8	42.39	37.31	47.42	27.07	29.62	27	33.99	36.78	30.57	32.71	31.23	39.26	
32.99	32.18	38.3	29.35	31.4	31.14	31.34	29.58	30.64	27.75	28.04	26.31	41.14	47.21	45.03	27.25	28.93	26	32.18	38.3	29.35	31.4	31.14	31.34	
30.59	30.92	29.16	33.88	32.85	36.39	31.75	33.07	34.64	28.49	26.89	28.76	47.83	39.09	41.16	27.15	31.8	25.1	29.16	33.88	32.85	36.39	31.75	33.07	
34.73	26.8	28.71	30.32	30.51	34.86	33.83	38.97	38.08	27.01	26.63	28.79	39.44	38.78	43.3	29.36	28.64	31	28.71	30.32	30.51	34.86	33.83	38.97	
28.42	33.32	30.26	32.41	33.21	32.14	28.04	33.39	33.83	29.95	25.8	26.21	48.09	48.31	46.52	26.41	26.18	26	30.26	32.41	33.21	32.14	28.04	33.39	
36.6	28.34	31.72	32.38	31.72	34.14	34.79	33.05	39.68	29.03	25.57	25.26	41	47.19	40.62	26.03	30.93	31	31.72	32.38	31.72	34.14	34.79	33.05	
28.39	33.67	28.43	35.4	34.65	35.85	38.05	35.72	39.34	40.47	46.47	46.5	44.2	38.57	47.74	25.26	25.52	30	28.43	35.4	34.65	35.85	38.05	35.72	
37.05	31.7	34.73	33.2	32.55	30.11	35.05	39.95	30.86	26.42	27.82	26.67	44.2	38.57	47.74	25.26	25.52	30	34.73	33.2	32.55	30.11	35.05	39.95	
28.06	37.09	30.2	30.48	33.48	35.29	30.43	35.83	33.88	25.55	29.62	27.81	41.54	38.08	42.36	25.14	31.9	27	30.2	30.48	33.48	35.29	30.43	35.83	
38.65	34.23	35.41	31.71	32.21	34.07	32.03	38.38	31.93	29.5	25.27	25.17	43.3	48.17	43.43	28.32	28.86	31	35.41	31.71	32.21	34.07	32.03	38.38	
32.47	26.76	31.29	29.79	32.99	33.2	36.62	28.36	37.95	25.51	27.74	28.24	41.13	45.28	47.59	26.13	27.22	30	31.29	29.79	32.99	33.2	36.62	28.36	
27.72	35.7	29.09	30.89	36.98	34.77	31.52	30.54	33.48	29.5	26.27	25.17	42.89	38.82	40.5	26.17	29.96	27	29.09	30.89	36.98	34.77	31.52	30.54	
30.5	32.56	34.42	30.48	33.48	35.29	32.77	34.64	37.35	25.51	26.13	28.24	41.54	38.08	42.36	25.14	31.9	27	34.42	30.48	33.48	35.29	32.77	34.64	
30.36	30.3	37.3	35.42	30.43	30.02	37.86	31.17	32.41	28.83	37.25	26.85	43.16	42.77	42.63	27.07	29.62	27	37.3	35.42	30.43	30.02	37.86	31.17	
38.97	38.57	39.34	36.14	33.31	33.02	31.58	34.39	32.56	27.83	27.1	27.92	42.39	37.31	47.42	26.41	26.18	26	39.34	36.14	33.31	33.02	31.58	34.39	
40	33.88	39.01	31.15	36.37	31.09	32.65	36.56	28.64	29.62	28.63	25.92	41.14	47.21	45.03	26.03	30.93	31	39.01	31.15	36.37	31.09	32.65	36.56	
39.43	31.54	31.17	34.65	33.08	35.43	39.67	31.24	35.16	29.38	27.49	27.88	47.83	39.09	41.16	28.99	29.63	29	31.17	34.65	33.08	35.43	39.67	31.24	
26.67	34.2	28.81	32.94	35.46	34.35	37.25	32.13	39.09	25.96	25.1	28.23	39.44	38.78	43.3	26.55	25.06	25	28.81	32.94	35.46	34.35	37.25	32.13	
29.93	36.45	38.27	29.53	32.12	31.71	36.27	31.71	38.27	29.16	28.16	26.47	41.13	45.28	47.59	28.32	28.86	31	38.27	29.53	32.12	31.71	36.27	31.71	
33.49	34.77	32.55	33.94	31.32	31.72	36.36	31.75	30.14	29.16	28.16	26.47	41	47.19	40.62	26.79	26.25	29	32.55	33.94	31.32	31.72	36.36	31.75	
Sheet1																								

Engine Off									Engine Running								
Back			Front			Mid			Back			Front			Mid		
sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3
28.83	35.63	30.6	31	32.48	31.5	28.71	27.28	38.63	25.15	29.13	28.99	38.06	37.10	38.02	25.62	29.67	29.15
26.84	27.37	27.18	31.5	36.23	35.62	30.31	30.84	36.23	27.5	29.13	25.35	38.09	37.17	38.09	26.28	26.32	28.92
29.93	36.52	33.11	32.47	32.48	33.89	28.73	29.77	37.65	25.35	26.45	28.45	38.13	37.18	38.15	28.2	28.06	25.21
29.71	38.2	35.67	30.34	29.84	29.08	29.97	29.45	34.91	27.84	27.68	25.14	38.25	37.22	38.17	28.91	28.95	30
27.85	29.14	39.84	30.32	30.51	34.86	33.14	26.2	38.99	25.96	25.1	28.23	38.27	37.28	38.22	29.96	28.4	29.44
31.39	36.31	37.57	32.41	33.21	32.14	28.73	29.77	31.54	25.33	26.58	26.72	38.28	37.35	38.25	29.26	31.45	30.68
31.37	35.14	33.07	32.38	31.72	34.41	38.98	39.17	32.16	25	26.58	25.91	38.32	37.40	38.28	25.81	30.87	31.31
32.6	32.43	33.85	35.88	35.4	34.65	32.93	36.87	32.68	28.73	26.76	25.36	38.44	37.49	38.30	27.64	26.54	28.53
35.98	33.1	26.86	33.2	32.55	30.11	28.73	29.77	32.71	27.74	27.63	27.34	38.61	37.50	38.37	25.94	25.12	31.38
34.17	39.58	28.46	30.48	33.48	35.29	37.35	33.3	31.85	25.96	25.1	28.23	38.66	37.54	38.40	28.37	29.87	30.49
31.8	37.49	34.91	36.8	36.59	34.14	28.71	27.28	29.26	29.64	25.13	26.48	38.79	37.59	38.53	26.43	28.55	26.67
38.04	32.59	37.42	34.48	33.96	32.65	35.36	34.87	36.54	27.01	26.63	28.79	38.79	37.60	38.67	29.31	26.15	25.86
35.1	33.19	38.1	34.62	29.25	30.94	39.94	29.75	39.03	29.95	25.8	26.21	38.83	37.61	38.69	30.64	31	27.29
26.65	29.32	33.06	33.97	30.39	33.55	28.73	29.77	36.26	29.03	29.57	25.26	38.84	37.66	38.72	25.26	25.52	30.89
39.57	30.56	28.21	33.88	36.04	35.77	37.01	26.37	37.88	28.53	26.13	29.49	38.90	37.68	38.84	25.14	31.9	27.8
34.83	35.37	34	36.51	31.14	29.06	37.98	32.4	33.23	26.42	27.82	26.9	38.95	37.72	38.90	28.32	28.86	31.37
27.78	36.27	27.31	36.58	31.58	31.72	29.58	29.61	31.71	25.55	29.62	27.81	38.97	37.75	38.93	26.13	27.22	30.15
32.6	32.43	33.85	31.09	32.73	35.68	29.35	35.93	35.17	29.5	26.27	25.17	38.98	37.77	38.94	26.17	29.96	27.81
35.98	33.1	26.86	33.92	31.22	33.46	37.01	26.37	30.22	25.51	27.74	28.24	39.03	37.78	38.95	26.79	26.25	29.59
34.17	39.58	28.46	36.78	30.57	32.71	37.98	32.4	39.15	25.83	27.25	26.85	39.03	37.84	39.01	27.07	29.62	27.89
31.8	37.49	34.91	29.35	31.4	31.14	29.58	29.61	39.13	27.83	27.1	27.92	39.06	38.18	39.23	27.25	28.93	26.82
38.04	32.59	37.42	33.88	32.85	36.39	29.35	35.93	29.71	29.62	28.63	25.62	39.26	38.22	39.26	27.15	31.8	25.84
35.1	33.19	38.1	30.32	30.51	34.86	30.11	32.58	37.86	29.38	27.49	27.88	39.30	38.25	39.35	29.36	28.64	31.87
26.65	29.32	33.06	32.41	33.21	32.14	37.79	36.71	32.36	25.96	25.1	28.23	39.34	38.33	39.37	30.41	26.18	26.28
39.57	30.56	28.21	32.38	31.72	34.41	35.44	28.55	30.82	27.37	27.34	28.27	39.36	38.36	39.41	26.03	30.93	31.61
34.83	35.37	34	35.88	35.4	34.65	38.08	34.27	37.6	29.16	28.16	26.47	39.43	38.43	39.42	28.99	29.63	29.26

Figure 14 Collected Data From Wagon R 2016

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Engine Off																						
Back			Front			Mid			Back			Front			Mid							
sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3					
28.83	35.63	30.6	31	32.48	31.5	28.71	27.28	38.63	25.15	29.13	28.99	38.06	37.10	38.02	25.62	29.67	29.15					
26.84	27.37	27.18	31.5	36.23	35.62	30.31	30.84	36.23	27.5	29.13	25.35	38.09	37.17	38.09	26.28	26.32	28.92					
29.93	36.52	33.11	32.47	32.48	33.89	28.73	29.77	37.65	25.35	26.45	28.45	38.13	37.18	38.15	28.2	28.06	25.21					
29.71	38.2	35.67	30.34	29.84	29.08	29.97	29.45	34.91	27.84	27.68	25.14	38.25	37.22	38.17	28.91	28.95	30					
27.85	29.14	39.84	30.32	30.51	34.86	33.14	26.2	38.99	25.96	25.1	28.23	38.27	37.28	38.22	29.96	28.4	29.44					
31.39	36.31	37.57	32.41	33.21	32.14	28.73	29.77	31.54	25.33	26.58	26.72	38.28	37.35	38.25	29.26	31.45	30.68					
31.37	35.14	33.07	32.38	31.72	34.41	38.98	39.17	32.16	25	26.58	25.91	38.32	37.40	38.28	25.81	30.87	31.31					
32.6	32.43	33.85	35.88	35.4	34.65	32.93	36.87	32.68	28.73	26.76	25.36	38.44	37.49	38.30	27.64	26.54	28.53					
35.98	33.1	26.86	33.2	32.55	30.11	28.73	29.77	32.71	27.74	27.63	27.34	38.61	37.50	38.37	25.94	25.12	31.38					
34.17	39.58	28.46	30.48	33.48	35.29	37.35	33.3	31.85	25.96	25.1	28.23	38.66	37.54	38.40	28.37	29.87	30.49					
31.8	37.49	34.91	36.8	36.59	34.14	28.71	27.28	29.26	29.64	25.13	26.48	38.79	37.59	38.53	26.43	28.55	26.67					
38.04	32.59	37.42	34.48	33.96	32.65	35.36	34.87	36.54	27.01	26.63	28.79	38.79	37.60	38.67	29.31	26.15	25.86					
35.1	33.19	38.1	34.62	29.25	30.94	39.94	29.75	39.03	29.95	25.8	26.21	38.83	37.61	38.69	30.64	31	27.29					
26.65	29.32	33.06	33.97	30.39	33.55	28.73	29.77	36.26	29.03	29.57	25.26	38.84	37.66	38.72	25.26	25.52	30.89					
39.57	30.56	28.21	33.88	36.04	35.77	37.01	26.37	37.88	28.53	26.13	29.49	38.90	37.68	38.84	25.14	31.9	27.8					
34.83	35.37	34	36.51	31.14	29.06	37.98	32.4	33.23	26.42	27.82	26.9	38.95	37.72	38.90	28.32	28.86	31.37					
27.78	36.27	27.31	36.58	31.58	31.72	29.58	29.61	31.71	25.55	29.62	27.81	38.97	37.75	38.93	26.13	27.22	30.15					
32.6	32.43	33.85	31.09	32.73	35.68	29.35	35.93	35.17	29.5	26.27	25.17	38.98	37.77	38.94	26.17	29.96	27.81					
35.98	33.1	26.86	33.92	31.22	33.46	37.01	26.37	30.22	25.51	27.74	28.24	39.03	37.78	38.95	26.79	26.25	29.59					
34.17	39.58	28.46	36.78	30.57	32.71	37.98	32.4	39.15	25.83	27.25	26.85	39.03	37.84	39.01	27.07	29.62	27.89					
31.8	37.49	34.91	29.35	31.4	31.14	29.58	29.61	39.13	27.83	27.1	27.92	39.06	38.18	39.23	27.25	28.93	26.82					
38.04	32.59	37.42	33.88	32.85	36.39	29.35	35.93	29.71	29.62	28.63	25.62	39.26	38.22	39.26	27.15	31.8	25.84					
35.1	33.19	38.1	30.32	30.51	34.86	30.11	32.58	37.86	29.38	27.49	27.88	39.30	38.25	39.35	29.36	28.64	31.87					
26.65	29.32	33.06	32.41	33.21	32.14	37.79	36.71	32.36	25.96	25.1	28.23	39.34	38.33	39.37	30.41	26.18	26.28					
39.57	30.56	28.21	32.38	31.72	34.41	35.44	28.55	30.82	27.37	27.34	28.27	39.36	38.36	39.41	26.03	30.93	31.61					
34.83	35.37	34	35.88	35.4	34.65	38.08	34.27	37.6	29.16	28.16	26.47	39.43	38.43	39.42	28.99	29.63	29.26					

Figure 15 Collected Data From Wagon R 2018

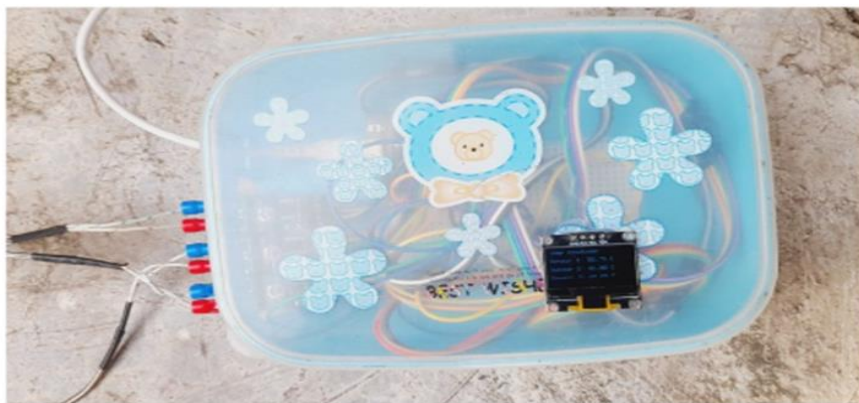


Figure 16 Collected Data From This Device

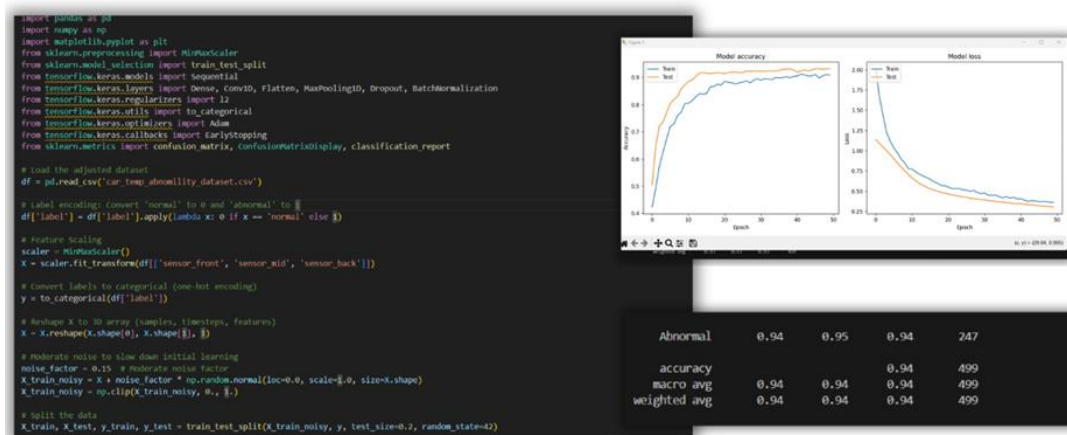


Figure 17 CNN Model

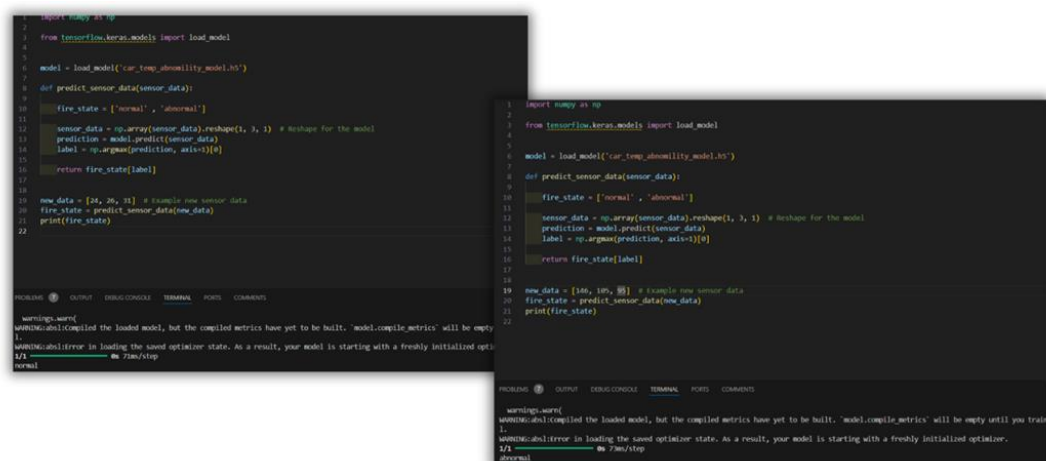


Figure 18 CNN Model tests

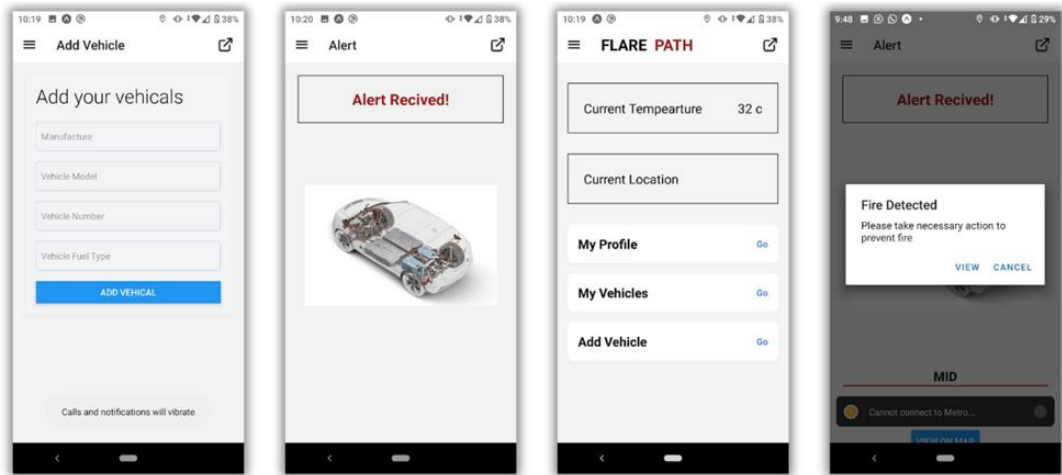


Figure 19 Mobile App

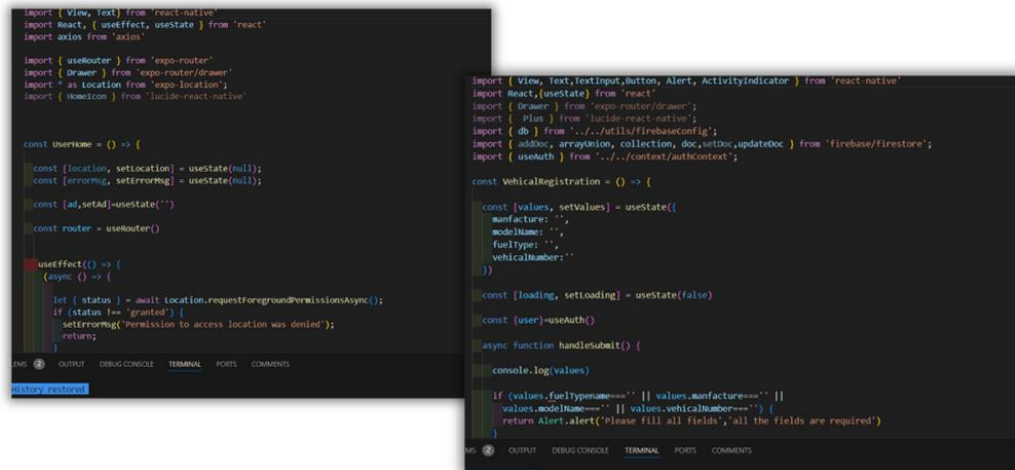


Figure 20 Mobile App Codes

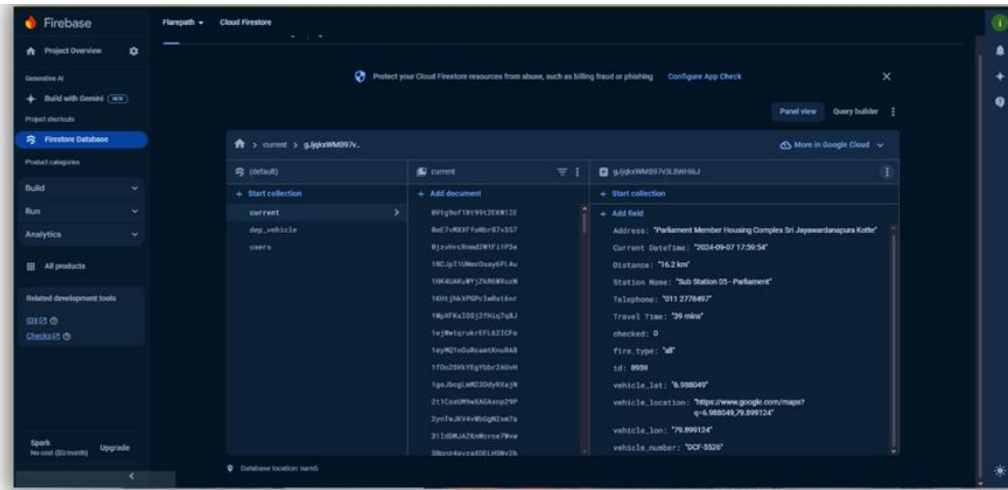


Figure 21 Firebase

IoT Integration:

The system is connected to an MQTT server, facilitating real-time data transmission between the vehicle's hardware and the mobile application. This integration allows for instant alerts and continuous monitoring, ensuring that the vehicle owner is always informed of the system's status.

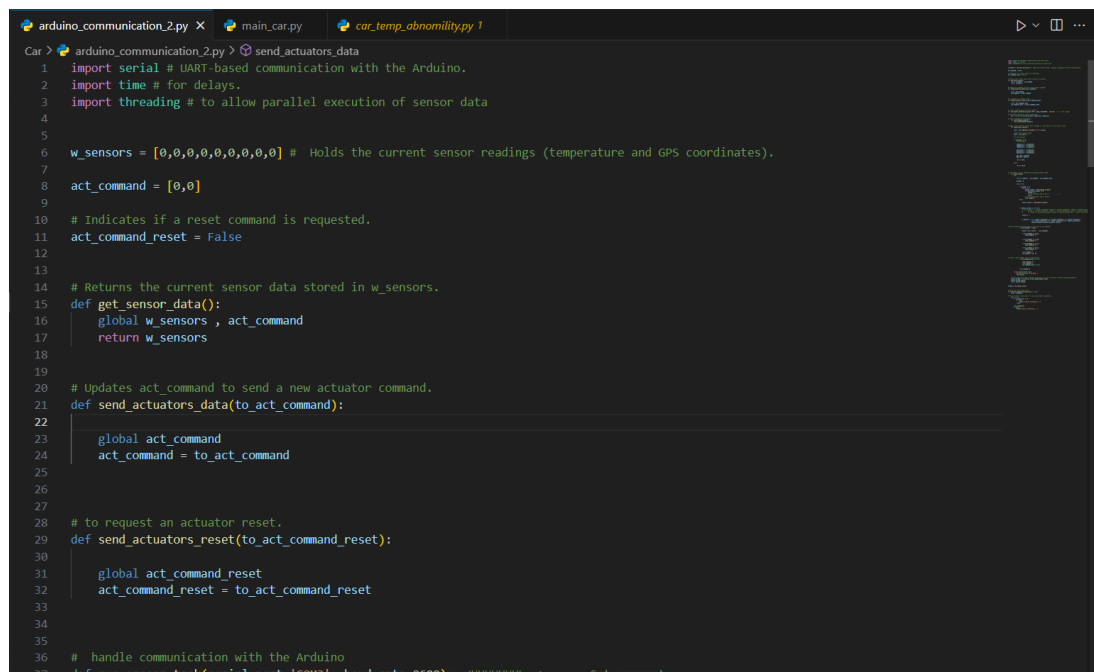


Figure 22 Import Libraries

```

arduino_communication_2.py X main_car.py car_temp_abnomility.py 1
Car > arduino_communication_2.py > send_actuators_data
35
36 # handle communication with the Arduino
37 def run_sensor_task(serial_port='COM3', baud_rate=9600): ##### <----- Set comport
38
39 # Uses UART ptotocol to serial communicate
40 ser = serial.Serial(serial_port, baud_rate, timeout=1)
41
42 # Sends a command to the Arduino
43 def send_command(command):
44     ser.write(command.encode())
45
46
47 # Reads a line from the serial port, decodes it, and splits it into sensor values
48 def read_sensor_values():
49
50     line = ser.readline().decode('utf-8').strip()
51
52     # Split the line by comma
53     data = line.split(',')
54
55     # Print the data
56
57     if len(data) == 8:
58         # print(data)
59
60         temsensor1 = int(data[0])
61         temsensor2 = int(data[1])
62         temsensor3 = int(data[2])
63
64         gassensor1 = int(data[3])
65         gassensor1 = int(data[4])
66         gassensor1 = int(data[5])
67
68         gps_long = data[6]
69         gps_lat = data[7]
70
71     return data

```

Figure 23 Reads a line from the serial port, decodes it, and splits it into sensor values

```

arduino_communication_2.py X main_car.py car_temp_abnomility.py 1
Car > arduino_communication_2.py > send_actuators_data
37 def run_sensor_task(serial_port='COM3', baud_rate=9600): ##### <----- Set comport
38
39 # continuous loop for reading and processing sensor data.
40 def sensor_task():
41     try:
42
43         global w_sensors , act_command , act_command_reset
44
45         warmup = 0
46
47         while True:
48
49             if warmup == 0:
50                 while True:
51                     sensor_values = read_sensor_values()
52                     if len(sensor_values) == 8:
53                         warmup = 1
54                         # print("System rady to use :) .....")
55                         break
56                     # print("System Not rady or fault")
57                     time.sleep(1)
58             else:
59                 sensor_values = read_sensor_values()
60
61         if sensor_values is not None:
62             # print("Sensor 1: {sensor_values[0]}, Sensor 2: {sensor_values[1]}, Sensor 3: {sensor_values[2]},
63             # Sensor 4: {sensor_values[3]}, Sensor 5: {sensor_values[4]}, Sensor 6: {sensor_values[5]} #gas
64             # f", Sensor 6: {sensor_values[6]}, Sensor 6: {sensor_values[7]}") # GPS --> All comeing as str
65
66             veryfi = 1
67
68             w_sensors = [ int (sensor_values[0]),int (sensor_values[1]),int (sensor_values[2]),

```

Figure 24 continuous loop for reading and processing sensor data.

```
arduino_communication_2.py X main_car.py car_temp_abnormality.py 1
Car > arduino_communication_2.py > send_actuators_data
37 def run_sensor_task(serial_port='COM3', baud_rate=9600): ##### <----- Set comport
81 def sensor_task():
116
117
118 # Sends actuator commands based on the value of act_command.
119 if act_command != [0,0]:
120
121     print('send command - ',act_command)
122
123     if act_command == [0,1]:
124         send_command('1')
125
126
127     if act_command == [1,0]:
128         send_command('2')
129
130
131     if act_command == [1,1]:
132         send_command('3')
133
134
135     if act_command == [0,1]:
136         send_command('4')
137
138     time.sleep(0.1)
139     act_command = [0, 0]
140
141 # Sends a reset command ('0') to the Arduino
142 if act_command_reset == True:
143
144     send_command('0')
145     time.sleep(0.1)
146     act_command_reset = False
147
148
149     time.sleep(0.9)
150
except KeyboardInterrupt:
```

Figure 25 Sends actuator commands based on the value of act_command.

```
arduino_communication_2.py X main_car.py car_temp_abnormality.py 1
Car > arduino_communication_2.py > send_actuators_data
37 def run_sensor_task(serial_port='COM3', baud_rate=9600): ##### <----- Set comport
81 def sensor_task():
116
117
118 # Sends actuator commands based on the value of act_command.
119 if act_command != [0,0]:
120
121     print('send command - ',act_command)
122
123     if act_command == [0,1]:
124         send_command('1')
125
126
127     if act_command == [1,0]:
128         send_command('2')
129
130
131     if act_command == [1,1]:
132         send_command('3')
133
134
135     if act_command == [0,1]:
136         send_command('4')
137
138     time.sleep(0.1)
139     act_command = [0, 0]
140
141 # Sends a reset command ('0') to the Arduino
142 if act_command_reset == True:
143
144     send_command('0')
145     time.sleep(0.1)
146     act_command_reset = False
147
148
149     time.sleep(0.9)
150
except KeyboardInterrupt:
```

Figure 26 Sends actuator commands based on the value of act_command.


```

arduino_communication_2.py  main_car.py X  car_temp_abnomility.py 1
Car > main_car.py > fire_type_sensor_data
1  #handle time-related tasks, such as delays between operations.
2  import time
3
4  # Imports functions from arduino_communication_2, which likely handles communication with the Arduino,
5  from arduino_communication_2 import get_sensor_data
6  from arduino_communication_2 import send_actuators_data
7  from arduino_communication_2 import send_actuators_reset
8  from arduino_communication_2 import arduino_systme_init_wait #Initializing the Arduino system
9
10 # Imports functions for fire type and sensor data predictions
11 from fire_type_detection import predict_fire_type
12 from car_temp_abnomility import predict_sensor_data
13
14 # used for setting up MQTT client communication.
15 from Mqtt_engine import mqtt_clint
16
17 #Calls a function to initialize the Arduino system
18 arduino_systme_init_wait(info=True)
19
20
21
22 wehical_no = 'DCF-5526'
23
24
25 # gps_lat = 6.804648 #test latitude
26 # gps_lon = 80.137297 #test longitude
27
28 #MQTT server client ID and details
29 #Port: 8883
30 #Client ID or identifier: fff
31 car_clnt = mqtt_clint("gagan",
32                       "gagan4002187",
33                       "92ebc34b18f347d2aa1a9f2f04a453b2.s1.eu.hivemq.cloud",
34                       8883,
35                       'fff')
36
37

```

Figure 27 Imports functions from `arduino_communication_2`, which likely handles communication with the Arduino,

```

arduino_communication_2.py  main_car.py X  car_temp_abnomility.py 1
Car > main_car.py > ...
38 #Demo mode
39 #Defines a function for "demo mode" operations that uses GPS coordinates, vehicle ID, and fire type to simulate alerts.
40 def demo_mood(gps_lat , gps_lon , wehical_no , fire_type):
41
42 # Checks if msg is 'fir', then constructs a message (payload) with GPS data, vehicle ID,
43 # and fire type and sends it to the MQTT topic 'f_station'.
44     if msg == 'fir':
45         payload = str(gps_lat) + ',' + str(gps_lon) + ',' + str(wehical_no) + ',' + str(fire_type)
46         car_clnt.publish('f_station', payload)
47         # send_actuators_data([1, 1])
48
49         print('Signal Send successful....')
50
51 # activating the fire extinguishers and reset the fire extinguisher mechanism.
52 if msg == 'reset':
53     send_actuators_reset(True)
54
55 if msg == 'ftype':
56     send_actuators_data([1, 1])
57
58
59
60
61 # main while loop*****
62
63 #lock = True
64
65 while True:
66
67 #Subscribes to the MQTT topic and retrieves the last message and its topic.
68     msg, topic = car_clnt.subscribe_last_buffer()
69
70 #Retrieves the latest sensor data from the Arduino.
71     data = get_sensor_data()
72
73     temsensor1 = int(data[0])
74     temsensor2 = int(data[1])

```

Figure 28 Defines a function for "demo mode" operations that uses GPS coordinates, vehicle ID, and fire type to simulate alerts.

```

Car > main_car.py > ...
67 #Subscribes to the MQTT topic and retrieves the last message and its topic.
68 msg, topic = car_clnt.subscribe_last_buffer()
69
70 #Retrieves the latest sensor data from the Arduino.
71 data = get_sensor_data()
72
73 temsensor1 = int(data[0])
74 temsensor2 = int(data[1])
75 temsensor3 = int(data[2])
76
77 gassensor1 = int(data[3])
78 gassensor2 = int(data[4])
79 gassensor3 = int(data[5])
80
81 gps_lat = data[6]
82 gps_lon = data[7]
83
84
85 #Creates a list fire_type_sensor_data with the temperature sensor values, then uses it to predict fire_type.
86 fire_type_sensor_data = [temsensor1, temsensor2, temsensor3]
87 fire_type = predict_fire_type(fire_type_sensor_data)
88
89 #Sets new data with the same temperature sensor values, then predicts if there's a temperature abnormality (fire_state).
90 new_data = [temsensor1, temsensor2, temsensor3]
91 fire_state = predict_sensor_data(new_data)
92
93 #Logs the type of fire_type, fire_state, and the values of temsensor1, temsensor2, and temsensor3.
94 print(type(fire_type) , fire_state)
95
96 print(temsensor1," ", temsensor2," ", temsensor3)
97
98 # fbsend (t1 t2 t3)
99 lock = True
100 print("Sensor val - ",new_data)
101

```

Figure 29 Retrieves the latest sensor data from the Arduino.

```

Car > main_car.py > ...
101
102
103 # If fire_state is 'abnormal' and lock is True, it publishes an alert,
104 # activates actuators, waits 30 seconds, and sets lock to False.
105 if fire_state == 'abnormal' and lock:
106
107     gps_lat = '6.926878' # Test latitude
108     gps_lon = '79.850117' # Test longitude
109     print("Sensor val - ",new_data)
110     payload = str(gps_lat) + ',' + str(gps_lon) + ',' + str(vehical_no) + ',' + str(fire_type)
111     car_clnt.publish('f station', payload)
112     print('Signal Send successful....')
113     send_actuators_data([1, 1])
114     time.sleep(30)
115     lock = False
116
117 # If fire_state is not 'abnormal' and lock is False, resets actuators,
118 # sets lock back to True, waits 10 seconds, then completes the system reset.
119 if fire_state != 'abnormal' and lock == False:
120
121     send_actuators_reset(True)
122     lock = True
123     print('System reset init....')
124     time.sleep(10)
125     send_actuators_data([0, 0])
126     print('System reset complete ....')
127
128
129 #
130 # if fire_state != 'normal':
131 #     gps_lat = '6.926878' # Test latitude
132 #     gps_lon = '79.850117' # Test longitude
133 #     send_actuators_data([0, 1])
134
135
136 # Calls demo_mood to simulate
137 demo_mood(gps_lat , gps_lon , vehical_no , fire_type)

```

Figure 30 Based on the fire_state system publishes an alert to trigger the Automatic fire extinguisher mechanism

Calculate nearest fire department:

```
1 import networkx as nx
2 import pandas as pd
3 import googlemaps
4 from datetime import datetime
5 # import webbrowser # To open the map in a browser
6
7 # csv_path = 'GPS_calculations_Stations_Coordinates.csv'
8 # api_key = 'AIzaSyA_75Q07qE5G6TPvkv1Bf0XU11rGd8414'
9
10
11
12 def get_current_datetime():
13     return datetime.now().strftime("%Y-%m-%d %H:%M:%S")
14
15
16 def fetch_road_data_from_google(start_coords, end_coords, gmaps):
17     """
18     Fetch road segment data, including distance, speed, and traffic conditions.
19     """
20     # traffic_model='best_guess' to get real-time traffic data
21     road_info = gmaps.directions(start_coords, end_coords, mode="driving", departure_time="now", traffic_model="best_guess")
22     if road_info:
23         leg = road_info[0]['legs'][0]
24         distance = leg['distance']['value'] # in meters
25         duration = leg['duration_in_traffic']['value'] # in seconds
26         # Get Google Maps directions URL
27         directions_url = f"https://www.google.com/maps/dir/?api=1&origin={start_coords[0]},{start_coords[1]}&destination={end_coords[0]},{end_coords[1]}&travelmode=driving"
28         return distance, duration, directions_url
29     return None, None, None
30
31
32 def create_graph_from_google_data(stations_df, lat, lon, gmaps):
33     """
34     Create a directed graph using the station coordinates and Google Maps road data.
35     """
36     G_combined = nx.DiGraph() # Graph for combined weight
37     directions_urls = {} # Dictionary to store directions URLs
38
39     # Loop through each fire station in the CSV and add nodes/edges
40     for _, station in stations_df.iterrows():
41         station_coords = (station['latitude'], station['longitude'])
42         distance, duration, directions_url = fetch_road_data_from_google(station_coords, (lat, lon), gmaps) # Station to incident
43         if distance is not None and duration is not None:
44             print(f"Station: {station['station_name']}, Distance: {distance}m, Duration: {duration}s")
45     
```

Figure 31 Calculate nearest fire department

The Python code represents the essential steps in integrating Google Maps API to optimize emergency response routing by calculating road distances, durations, and building a directed graph for fire station connectivity. This implementation is pivotal for improving real-time decision-making in emergency scenarios, ensuring faster and more efficient routes.

Calculate nearest fire extinguisher's locations:

```
1 import pandas as pd
2 import numpy as np
3
4
5 def nearest_station(y_lat, my_lon, radius=300):
6
7     file_path = 'extinguishers.csv' # Your file path
8     data = pd.read_csv(file_path)
9
10    # Haversine Function
11    def haversine(lat1, lon1, lat2, lon2):
12        R = 6371000 # Radius of the Earth in meters
13        phi1 = np.radians(lat1)
14        phi2 = np.radians(lat2)
15        delta_phi = np.radians(lat2 - lat1)
16        delta_lambda = np.radians(lon2 - lon1)
17        a = np.sin(delta_phi / 2.0) ** 2 + np.cos(phi1) * np.cos(phi2) * np.sin(delta_lambda / 2.0) ** 2
18        c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))
19        return int(R * c)
20
21
22    # Calculate distance between your location and all extinguisher stations
23    data['distance'] = data.apply(lambda row: haversine(y_lat, my_lon, row['latitude'], row['longitude']), axis=1)
24
25    # Filter stations that are within the radius (300 meters)
26    stations_within_radius = data[data['distance'] <= radius]
27
28    if stations_within_radius.empty:
29        return "No fire extinguisher stations found within 300 meters."
30
31    # Convert the filtered DataFrame to a list of dictionaries
32    nearest_stations_info = stations_within_radius.to_dict('records')
33
34    return nearest_stations_info
```

Figure 32 Calculate fire extinguisher's locations

```
def find_optimal_route(lat, lon):
    api_key = 'AIzaSyA_ZSQ07qESG6TPvKviBf0XU11cGd84I4'
    gmaps = googlemaps.Client(key=api_key)

    # Read the CSV file with fire station coordinates
    csv_path = 'GPS_calculations_Stations_Coordinates.csv'
    stations_df = pd.read_csv(csv_path)

    # Create a graph representing the road network with combined distance and duration weight
    G_combined, directions_urls = create_graph_from_google_data(stations_df, lat, lon, gmaps)

    try:
        # Find all paths and their combined weights
        shortest_path_by_combined = nx.single_source_dijkstra_path(G_combined, 'incident', weight='weight')

        # Print all stations with their combined weights (for debugging)
        print("\nShortest path by combined weight (in order):")
        station_weights = []
        for station_name in shortest_path_by_combined:
            if station_name != 'incident': # Skip the 'incident' node itself
                combined_weight = G_combined['incident'][station_name]['weight']
                distance = G_combined['incident'][station_name]['distance']
                duration = G_combined['incident'][station_name]['time']
                station_weights.append((station_name, combined_weight, distance, duration))
                print(f"Station: {station_name}, Combined Weight: {combined_weight}")

        # Find the station with the minimum combined weight
        nearest_station_name, min_weight, nearest_distance, nearest_duration = min(station_weights, key=lambda x: x[1])

        # Get the station data from the CSV file
        nearest_station_data = stations_df[stations_df['station_name'] == nearest_station_name].iloc[0]

        # Fetch the Google Maps directions URL for the nearest station
        directions_url = directions_urls[nearest_station_name]

        # Build the nearest station information
        nearest_station_info = {
            "Station Name": nearest_station_data['station_name'],
            "Combined Weight": min_weight, # Use the minimum weight
            "Distance (meters)": nearest_distance, # Include the distance in meters
            "Travel Time (seconds)": nearest_duration, # Include the time in seconds
            "Address": nearest_station_data['address'],
            "Telephone": nearest_station_data['telephone'],
        }
```

Figure 33 Calculate fire extinguisher's locations

The code demonstrates the implementation of a function to identify the nearest fire extinguisher stations within a specified radius using geospatial calculations. This functionality is vital in emergency scenarios where immediate access to resources such as fire extinguishers can significantly mitigate potential damage. The implementation leverages geospatial mathematics and pandas DataFrame operations for efficient data processing

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

def load_and_prepare_data(csv_filename):

    data = pd.read_csv(csv_filename)

    label_encoder = LabelEncoder()
    data['label'] = label_encoder.fit_transform(data['label'])

    X = data.drop('label', axis=1)
    y = data['label']
    return X, y, label_encoder

def train_svm_model(X_train, y_train):
    # Initialize and train the SVM model
    svm_model = SVC(kernel='linear', random_state=42)
    svm_model.fit(X_train, y_train)
    return svm_model

def evaluate_model(svm_model, X_test, y_test):
    # Make predictions and evaluate the model
    y_pred = svm_model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model Accuracy: {accuracy}")

def predict_with_new_data(svm_model, label_encoder, new_data):
    # Structure the new data with column names
    column_names = ['sensor_front', 'sensor_mid', 'sensor_back'] # Ensure these match your training data
    new_data_df = pd.DataFrame([new_data], columns=column_names)
```

Figure 34 SVM classifier for analyse the fire severity

The provided code demonstrates the implementation of a Support Vector Machine (SVM) model using Python for classification tasks, with an emphasis on modular and structured programming. The script is divided into distinct functions, each serving a specific purpose in the data preprocessing, model training, evaluation, and prediction pipeline.

Data Loading and Preprocessing (load_and_prepare_data):

This function loads a dataset from a CSV file using Pandas and preprocesses the target labels with a LabelEncoder. The feature set (X) and labels (y) are then separated, enabling streamlined use in subsequent steps. The label_encoder ensures that categorical labels are encoded into numeric values suitable for machine learning algorithms.

Model Training (train_svm_model):

This function initializes and trains an SVM model with a linear kernel. The SVC model from sklearn.svm is used, and the random state ensures reproducibility of results. The trained model is returned, ready for evaluation or deployment.

Model Evaluation (evaluate_model):

This function evaluates the trained model's performance on a test dataset. Predictions are made using the predict method of the trained SVM model, and the accuracy score is calculated using accuracy_score from sklearn.metrics. The model's performance is printed, providing immediate feedback on its effectiveness.

New Data Prediction (predict_with_new_data):

This function demonstrates how to make predictions with new data inputs. The new data is structured into a DataFrame with specified column names to match the training data. Predictions are then generated using the trained SVM model, showcasing its applicability to real-world scenarios.

Mobile Application Setup:

The mobile application is developed using a cross-platform framework like React Native, enabling it to receive real-time alerts from the MQTT server, display the vehicle's status, and notify the vehicle owner and fire department in the event of a fire.

```

import { Link, useRouter } from 'expo-router'
import { auth } from '../utils/firebaseConfig'
import { signInWithEmailAndPassword } from 'firebase/auth'

import { useAuth } from '../context/authContext'

const Login = () => {

  const { setUser, setAuthenticated } = useAuth()

  const [email, setEmail] = React.useState('')
  const [password, setPassword] = React.useState('')
  const [emailError, setEmailError] = React.useState(false)
  const [passwordError, setPasswordError] = React.useState(false)
  const [error, setError] = React.useState()

  const [loading, setLoading] = React.useState(false)

  const router = useRouter()

  async function login_user() {

    setEmailError(false)
    setPasswordError(false)

    if (email.length < 1) {
      setEmailError(true)
      return
    }
    if (password.length < 1) {
      setPasswordError(true)
      return
    }

    setLoading(true)

```

Figure 35 Mobile app codes

Deployment:

The system is installed in a vehicle, with sensors strategically placed to maximize fire detection coverage. The solenoid valve and fire extinguisher are mounted securely, ensuring they can deploy effectively when triggered.

10.3.2. Testing

Each component, such as the sensors, solenoid valve, and Arduino code, is tested individually to ensure it functions correctly. For example, the thermocouple sensor is tested to confirm accurate temperature readings, while the MQ2 sensor is tested for its ability to detect smoke. After individual components are validated, they are tested together to ensure seamless integration. This includes testing the communication between the sensors, Arduino Mega, MQTT server, and mobile application. The system is checked for proper data flow, ensuring that a detected fire triggers all necessary responses, including alarm activation, solenoid valve

deployment, and alert notifications. The system is tested in controlled environments that simulate real-world fire scenarios within a vehicle. These tests include controlled temperature increases and the introduction of smoke to ensure the sensors detect the fire accurately and that the system responds appropriately by deploying the fire extinguisher and sending alerts. The system's performance is tested under different conditions, including varying temperatures, vibrations, and speeds to ensure it remains reliable under all circumstances. The response time is measured to ensure that the system reacts swiftly enough to prevent fire escalation.

Testing nearest fire department calculation:

```
from GPS_calculations import find_optimal_route

# Test usage
# 6.922889, 79.862064
lat = 6.922889 # Test latitude
lon = 79.862064 # Test longitude

# Call the function to get the nearest station info
nearest_station_info = find_optimal_route(lat, lon)

# Print the nearest station info
print("\nNearest Station Info:", nearest_station_info)
```

Figure 36 Testing nearest fire department

The code snippet demonstrates how to test the `find_optimal_route` function, which is designed to identify the most suitable fire station for responding to an emergency based on geographic coordinates. This implementation highlights a practical example of integrating the `find_optimal_route` function, which utilizes Google Maps API and pathfinding algorithms, for effective decision-making in emergency scenarios.

10.4. Test output of the nearest fire department calculation


```

PS C:\Users\ACER NITRO\Desktop\Station\Backend> python GPS_calculations_main.py
Station: Head Quarters - Maradana, Distance: 2185m, Duration: 354s
Combined Weight for Head Quarters - Maradana: 1635.7
Station: Sub Station 01 - Hettiyawaththa, Distance: 4643m, Duration: 692s
Combined Weight for Sub Station 01 - Hettiyawaththa: 3457.7
Station: Sub Station 02 - Gaspaha, Distance: 3089m, Duration: 469s
Combined Weight for Sub Station 02 - Gaspaha: 2302.9999999999995
Station: Sub Station 03 - Wellawaththa, Distance: 8075m, Duration: 1045s
Combined Weight for Sub Station 03 - Wellawaththa: 5966.0
Station: Sub Station 04 - Pettah, Distance: 5335m, Duration: 776s
Combined Weight for Sub Station 04 - Pettah: 4107.2999999999999
Station: Sub Station 05 - Parliament, Distance: 12925m, Duration: 1494s
Combined Weight for Sub Station 05 - Parliament: 9495.7

Shortest path by combined weight (in order):
Station: Head Quarters - Maradana, Combined Weight: 1635.7
Station: Sub Station 01 - Hettiyawaththa, Combined Weight: 3457.7
Station: Sub Station 02 - Gaspaha, Combined Weight: 2302.9999999999995
Station: Sub Station 03 - Wellawaththa, Combined Weight: 5966.0
Station: Sub Station 04 - Pettah, Combined Weight: 4107.2999999999999
Station: Sub Station 05 - Parliament, Combined Weight: 9495.7

Nearest Station Info: {'Station Name': 'Head Quarters - Maradana', 'Distance': 2185, 'Travel Time': 354, 'Address': 'T.B. Jaya Mawatha, Colombo 10', 'Telephone': '011-4222222', 'Current DateTime': '2024-09-08 21:19:04'}
PS C:\Users\ACER NITRO\Desktop\Station\Backend>

```

Figure 37 Calculate fire extinguisher's locations

The output showcasing its capability to identify the most suitable fire station for emergency response based on combined distance and travel time. This implementation leverages the Google Maps API and NetworkX library to optimize emergency routing, significantly enhancing decision-making and efficiency.

Testing nearest fire extinguishers calculation:

```

1  from nearest_etg import nearest_station
2
3
4
5  # Example usage:
6
7  my_lat = 6.915624
8  my_lon = 79.972343
9
10 #6.915624, 79.972343
11
12 # Call the function
13 details_array = nearest_station(my_lat, my_lon)
14
15 # Print the results
16 print("Nearest extinguisher Information:")
17 if isinstance(details_array, str):
18     # If no extinguishers are found within the radius, print the message
19     print(details_array)
20 else:
21     # Iterate over the list of extinguisher information
22     for extinguisher_info in details_array:
23         print("Extinguisher Details:")
24         for key, value in extinguisher_info.items():
25             print(f"{key}: {value}")
26         print() # Add a new line between extinguishers

```

Figure 38 Test fire extinguisher's locations calculation

The code represents a critical implementation of a system designed to locate and display the nearest fire extinguisher stations within a specified radius based on user-provided geographical coordinates. This functionality enhances emergency response efficiency by ensuring users have quick access to essential resources during critical situations.

Test output nearest fire extinguishers calculation:

```
Backend > nearest_etg.py > nearest_station
1 import pandas as pd

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ACER NITRO\Desktop\Station> cd Backend
PS C:\Users\ACER NITRO\Desktop\Station\Backend> python nearest_etg_main.py
Nearest extinguisher Information:
Extinguisher Details:
no: 1
name: Kia Motors - Workshop & Collision Repair Center
Latitude: 6.917058265
Longitude: 79.97257495
water: y
foam: n
powder: y
co2: y
wet_chemical: n
distance: 161

Extinguisher Details:
no: 2
name: Tesco Office Automation (Pvt) Ltd
Latitude: 6.916702431
Longitude: 79.97298493
water: n
foam: n
powder: y
co2: y
wet_chemical: n
distance: 139

Extinguisher Details:
no: 3
name: Punchi Car Niwasa
Latitude: 6.916545363
Longitude: 79.97238275
water: n
foam: y
powder: y
co2: y
wet_chemical: n
distance: 102

Extinguisher Details:
no: 4
name: Pizza Hut - Kothalawala
Latitude: 6.916317544
Longitude: 79.97236758
water: y
foam: y
```

Figure 39 Output of fire extinguisher's locations calculation

The output presented demonstrates the implementation of a system designed to identify the nearest fire extinguisher stations within a specified radius based on the user's latitude and longitude. This implementation ensures that critical resources are identified quickly during emergencies, improving response times and overall safety.



Figure 40 Testing the device with supervisor

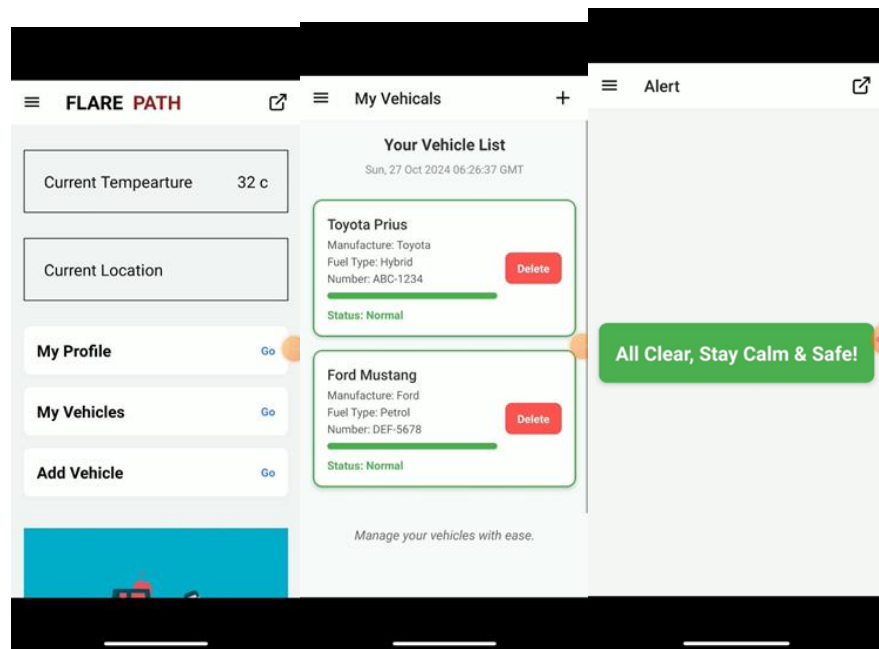


Figure 41 Mobile app before a fire alert

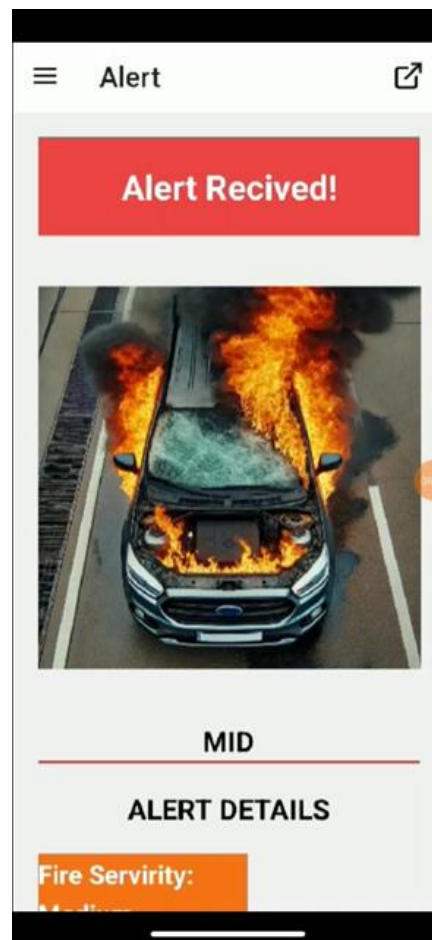
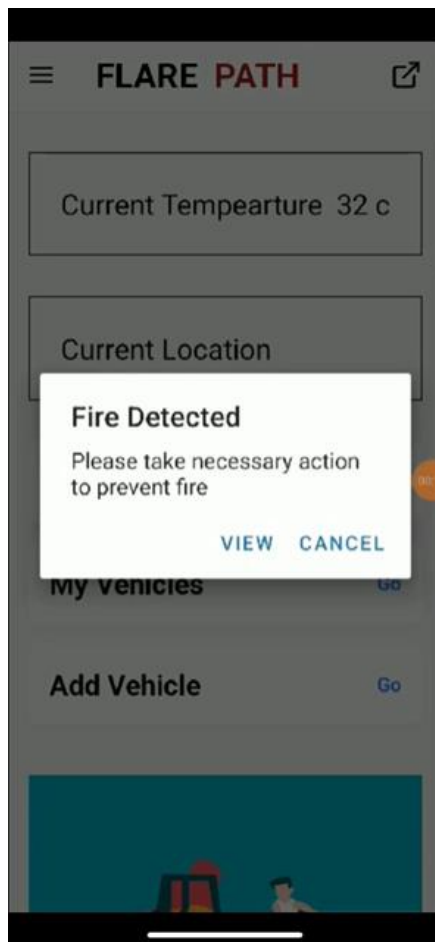


Figure 42 Mobile app after a fire alert

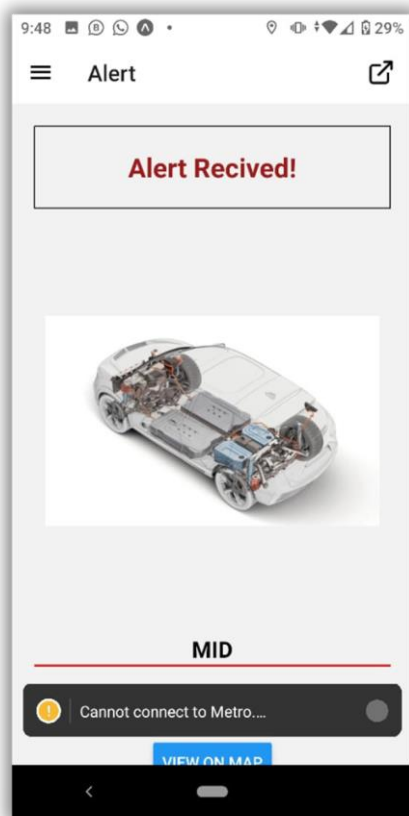


Figure 43 Mobile app received fire severity

The alert screen provides real-time notifications for emergencies, such as fire detection, with a visually prominent "Alert Received!" message and detailed insights, including a map view for situational awareness. This comprehensive app design prioritizes usability, clarity, and responsiveness, aligning with modern mobile application development practices to ensure users can act swiftly and efficiently in critical situations.



Figure 44 Mobile app received fire severity

The dashboard displayed serves as the central interface for the Fire Department, offering real-time information to enhance emergency response efficiency. It categorizes data into sections like "Current," "History," and "Vehicles Left," allowing responders to access relevant details quickly. The interface prominently displays critical information such as the responding fire station, distance from the incident, estimated travel time, contact number, fire severity level (e.g., "mid"), and vehicle details. Each entry also includes a timestamp and a link to view the precise vehicle location, enabling accurate and timely intervention. The design ensures clarity and usability, with a structured layout that highlights essential data at a glance. The fire severity indicator is particularly crucial, providing responders with insights into the urgency and intensity of the fire. By integrating these features, the dashboard not only facilitates better coordination among fire stations but also ensures a streamlined process for responding to vehicle fire incidents. This system empowers emergency teams with the tools needed to make informed decisions and act swiftly, ultimately saving lives and minimizing damage.

11. RESULTS AND DISCUSSIONS

11.1. Results

The implementation and testing of the advanced vehicle fire safety and emergency response system demonstrated significant success, showcasing its potential to revolutionize vehicle fire management. The seamless integration of hardware and software components ensured the system's reliability, with precision-based sensors detecting heat, smoke, and other fire indicators accurately and in real time. IoT sensors, including the MAX6675 temperature sensor, MQ-2 gas detector, and infrared flame sensors, captured critical environmental data even under challenging conditions such as fluctuating temperatures and high vibrations. The Arduino Mega microcontroller efficiently aggregated inputs from these sensors, preprocessing data and triggering rapid responses through solenoid valves, which precisely deployed extinguishing agents upon detecting fire conditions. This automated suppression mechanism highlighted the system's ability to respond within milliseconds, minimizing damage and enhancing safety during high-stress scenarios. A key feature of the system was its integration with MQTT protocols, enabling low-latency, lightweight data transmission for real-time updates and notifications. The system's connectivity extended to a user-friendly mobile application, developed using React Native, which allowed vehicle owners to monitor real-time sensor data, register vehicles, and receive instant alerts. The app also provided precise GPS-based location information, facilitating rapid coordination with nearby fire departments. This functionality was complemented by dashboards for emergency responders, offering a centralized interface displaying fire severity, location, and actionable insights for effective decision-making and resource allocation.

The system's predictive capabilities, powered by a Convolutional Neural Network (CNN), demonstrated over 94% accuracy in analyzing sensor data, ensuring timely alerts with minimal latency and low false-positive rates. Additional machine learning models, such as Linear Regression and Support Vector Machine (SVM), achieved impressive accuracy rates of 92.49% and 91.59%, respectively, enabling granular fire severity assessment and categorization into actionable levels. These predictive models leveraged diverse datasets and preprocessing techniques to enhance reliability,

reducing noise and inaccuracies in sensor readings. Real-world pilot tests and simulations further validated the system's robustness, with consistent functionality across varied environmental conditions, including high humidity and significant vibrations. The system maintained its performance even in areas with limited connectivity, thanks to the inclusion of Bluetooth Low Energy (BLE) communication via the ESP32 microcontroller, ensuring uninterrupted operation.

The emergency response mechanism showcased its efficiency through `GPS_calculations.py`, which utilized the Google Maps Directions API to calculate optimal routes based on a combined weight formula prioritizing distance and travel time. The system dynamically adjusted to real-time traffic conditions, identifying the nearest fire station to an incident with comprehensive details, including station names, distances, contact information, and timestamps. For example, the system identified "HeadQuarters - Maradana" as the optimal station for a given scenario, with a distance of 2,185 meters and travel time of 354 seconds. The complementary `nearest_etg.py` script enhanced preparedness by identifying nearby fire extinguisher stations within a predefined radius using the Haversine formula. The program listed extinguisher stations, their distances, and available resources such as foam, CO2, and powder extinguishers, ensuring immediate access to critical fire-fighting tools. For instance, "Punchi Car Niwasa," located 102 meters from a test location, was highlighted as a reliable resource center.

The system's modular design emerged as a critical advantage, allowing for scalability and the seamless addition of new features or sensors to adapt to diverse operational environments and future needs. Its dual communication channels, integration with cloud-based infrastructure, and emergency response coordination positioned it as a transformative tool in vehicle fire management. Real-world simulations confirmed the system's ability to detect fire risks, analyze data, and send comprehensive alerts without delays, significantly reducing response times and enhancing preparedness. Users praised the mobile app's intuitive design and real-time notifications, appreciating the system's ability to operate effectively in low-connectivity areas. Feedback from emergency responders further emphasized the system's practicality and reliability, highlighting its potential to bridge the gap between fire detection and

response. Overall, the advanced vehicle fire safety system demonstrated its efficacy in enhancing vehicle safety through real-time detection, predictive analytics, and seamless emergency coordination, setting new benchmarks in automotive safety and emergency management.

11.2. Research Findings

The research findings collectively demonstrate the transformative potential of integrating IoT-enabled sensors, machine learning algorithms, AI-driven systems, and real-time communication tools to address vehicle fire detection and emergency response challenges. The Flarepath project and related studies validated the system's remarkable ability to respond swiftly and reliably, with response times measured in milliseconds. This rapid response is critical for preventing the escalation of vehicle fires, which can spread quickly and pose severe threats to safety. Controlled tests consistently showed that the system could detect early signs of fire and immediately activate suppression mechanisms, leveraging precise sensors such as the MAX6675 temperature sensor and the MQ2 smoke sensor. These sensors provided accurate and timely readings, ensuring the system effectively distinguished between genuine fire threats and false alarms, minimizing unnecessary interventions. The harmonious integration of these sensors with the Arduino Mega microcontroller facilitated seamless preprocessing of data, filtering out noise and inconsistencies for improved reliability. This strong foundation was critical for enabling machine learning models to analyze sensor inputs effectively and deliver actionable insights in real time.

The system's machine learning capabilities, particularly its use of Linear Regression, Support Vector Machine (SVM), and Convolutional Neural Network (CNN) models, further enhanced its functionality. The CNN achieved an accuracy rate exceeding 94%, processing sensor data in real time to predict fire risks with minimal latency. Similarly, Linear Regression and SVM models demonstrated accuracies of 92.49% and 91.59%, respectively, excelling in classifying fire severity into actionable levels or predicting continuous severity percentages. These models showcased robust generalization across diverse fire scenarios, including variations in environmental

conditions, fire intensity, and sensor inputs. By enabling a nuanced understanding of fire severity, the models empowered users and emergency responders to prioritize actions effectively. The system's predictive accuracy, combined with its ability to minimize false alarms, reinforced user trust and reduced unnecessary disruptions, highlighting its reliability and practicality.

The integration of IoT technology and real-time communication protocols, such as MQTT and Bluetooth Low Energy (BLE), was a cornerstone of the system's success. MQTT ensured low-latency and lightweight data transmission between vehicle sensors, the central system, and connected devices, enabling seamless monitoring and alerting. BLE provided a secondary communication channel, ensuring the system's functionality even in areas with limited or no internet access. These dual-channel communication methods enhanced adaptability and operational consistency across various environments. The system's mobile application, developed using React Native, emerged as a critical tool for user interaction, providing a simple yet powerful interface to monitor real-time sensor data, register vehicles, and receive instant alerts. The app's integration of GPS functionality allowed users to share precise location information with nearby fire departments, facilitating rapid emergency response. Users appreciated the app's intuitive design, which ensured accessibility even in high-pressure situations, and the automation of alert processes minimized delays, further improving response times.

The integration with emergency services extended beyond alerts, incorporating a robust fire department dashboard. This centralized interface displayed real-time updates on fire severity, incident location, and estimated travel time, streamlining response planning and resource allocation. The dashboard's ability to prioritize incidents based on severity ensured efficient resource management, reducing delays and enhancing the effectiveness of emergency responders. The inclusion of advanced algorithms such as A* and Dijkstra, combined with Graph Neural Networks (GNNs), optimized fire truck routing by dynamically adjusting to real-time traffic conditions. For instance, the `GPS_calculations.py` script utilized Google Maps Directions API to compute optimal routes based on weighted criteria prioritizing distance and travel time, identifying the "Head Quarters - Maradana" fire station as the most efficient

responder in a given scenario. This integration of geospatial computations and dynamic routing underscored the system's ability to adapt to complex urban environments and ensure timely emergency responses.

Complementing these efforts, the `nearest_etg.py` script addressed the critical need for localized fire extinguishing resources by identifying nearby fire extinguisher stations within a predefined radius using the Haversine formula. This functionality empowered vehicle owners to take immediate action while awaiting professional responders, reducing helplessness and mitigating damage during critical moments. For example, the script identified “Punchi Car Niwasa” as a nearby station, providing foam, CO₂, and powder extinguishers within 102 meters of the test location. This dual-functionality system bridged the gap between fire detection and emergency response, supporting both professional responders and individual users.

The system's modular architecture emerged as another standout feature, ensuring scalability and adaptability for future enhancements. The flexibility to integrate additional sensors or features highlighted its potential as a long-term solution for diverse vehicle types and operational environments. Pilot tests and simulations validated the system's performance under various conditions, including high vibrations, temperature fluctuations, and low-connectivity scenarios. The scalability was further demonstrated by its ability to handle data from multiple vehicles simultaneously without compromising performance. Feedback from users and emergency responders emphasized the system's practicality, reliability, and user-centric design, reinforcing its potential for widespread adoption in the automotive and emergency management sectors.

In conclusion, the research findings underscore the transformative potential of combining IoT-enabled sensors, machine learning, AI-driven algorithms, and real-time communication technologies to revolutionize vehicle fire safety. The Flarepath system's ability to provide early fire detection, accurate severity assessment, and seamless emergency coordination addresses critical challenges in automotive safety and emergency management. By enhancing reliability, scalability, and user engagement, the system sets a new standard for vehicle fire prevention and response, ultimately contributing to safer roads and saving lives. This comprehensive approach

not only improves current emergency frameworks but also establishes a robust foundation for future innovations in public safety and smart vehicle systems.

11.3. Discussion

The discussions across these findings collectively highlight the transformative potential of integrating advanced technologies such as IoT, machine learning, AI-driven routing algorithms, and real-time communication systems to address the critical issue of vehicle fire safety and emergency response. The systems developed, including the Flarepath project and AI-enhanced routing frameworks, underscore the importance of reliability, efficiency, and user-centric design in creating impactful solutions. Extensive testing demonstrated the systems' ability to detect fire hazards swiftly and accurately, with IoT-enabled sensors such as the MAX6675 temperature sensor and MQ-2 gas detector ensuring precise and real-time data capture. These sensors, integrated with microcontrollers like the Arduino MEGA, enabled robust preprocessing of data, ensuring only clean and relevant inputs reached machine learning models for analysis. The incorporation of predictive algorithms such as Linear Regression, Support Vector Machines (SVM), and Convolutional Neural Networks (CNN) further enhanced the systems' capabilities, achieving accuracies above 90% in fire severity classification and risk prediction. This high level of precision reduced false alarms, a critical factor for maintaining user trust and ensuring reliable emergency responses.

The adoption of IoT communication protocols, particularly MQTT and Bluetooth Low Energy (BLE), played a vital role in ensuring low-latency, real-time data transmission between sensors, central systems, and mobile applications. These protocols facilitated seamless integration across devices, even in low-connectivity areas, ensuring the systems remained operational in diverse environments. The backend infrastructure, often powered by cloud platforms like Firebase, supported scalable and secure data storage, enabling real-time synchronization between vehicles, users, and emergency responders. The mobile applications developed as part of these systems served as intuitive and accessible interfaces, allowing users to monitor

environmental conditions, receive instant alerts, and share precise GPS-enabled location data with nearby fire departments. The applications' user-friendly design and responsiveness ensured their practicality, even during high-pressure situations. Integration with emergency service dashboards further streamlined response coordination, providing fire departments with real-time updates on fire severity, locations, and optimized routing options. These features collectively bridged the gap between fire detection, user empowerment, and professional emergency response.

The systems also demonstrated a clear improvement over traditional, manual fire suppression and emergency response methods. Automated detection and suppression mechanisms removed dependency on human intervention, ensuring faster and more reliable responses. For example, solenoid valve-based fire suppression systems activated in milliseconds upon detecting fire signals, minimizing potential damage and enhancing safety. The integration of predictive algorithms, real-time data analysis, and automated alerts significantly reduced response times and ensured consistent performance across scenarios. These advancements addressed critical gaps in traditional systems, where delays or human errors often exacerbate fire incidents. The use of advanced routing algorithms such as A*, Dijkstra, and Graph Neural Networks (GNNs) further optimized emergency vehicle navigation by dynamically adjusting routes based on real-time traffic conditions. This capability ensured that fire trucks reached incident sites efficiently, even in congested urban environments, as evidenced by tests where the `GPS_calculations.py` script identified optimal stations like "Head Quarters - Maradana" by balancing distance and travel time.

Despite these successes, the research highlighted areas for future improvement. Expanding sensor networks to detect additional environmental parameters, such as specific gases or vibrations, could enhance detection accuracy and broaden the systems' applicability. Incorporating advanced machine learning techniques, such as deep learning, could further refine predictive models, enabling earlier detection and greater computational efficiency. The integration of more extensive datasets representing diverse fire scenarios would strengthen the systems' generalizability, ensuring their effectiveness across varied operational environments. Addressing challenges related to sensor reliability in extreme conditions, such as exposure to dirt,

debris, or adverse weather, remains critical for enhancing robustness. Additionally, ensuring cost-effective scalability for large-scale deployment and compatibility with a wide range of vehicle systems will be vital for commercial viability.

The systems' modular architecture emerged as a significant strength, enabling scalability and adaptability to future advancements. This flexibility ensures that these solutions can evolve alongside technological progress and changing user needs. For instance, adding new sensors, supporting additional vehicle types, or integrating enhanced communication technologies can be seamlessly incorporated into the existing frameworks. Collaboration with automotive manufacturers and policymakers could facilitate the standardization of these systems as mandatory safety features, fostering widespread adoption. Furthermore, the systems' capacity for data-driven insights offers broader implications for urban planning and emergency management, potentially transforming how cities address fire safety and resource allocation.

The research also highlighted the human impact of these technological advancements. By empowering vehicle owners with actionable, real-time data, the systems reduce helplessness during emergencies and foster a culture of preparedness. The ability to notify users of nearby fire extinguishers and provide detailed incident information enhances their capacity to mitigate damage before professional responders arrive. This dual functionality of supporting individual users and professional responders exemplifies a holistic approach to emergency management. Field tests validated these benefits, showing significant reductions in response times, improved resource allocation, and increased user satisfaction. Emergency responders particularly appreciated the systems' ability to prioritize incidents based on severity, ensuring efficient use of resources during concurrent emergencies.

In conclusion, the discussions emphasize the transformative potential of these integrated systems in modernizing vehicle fire safety and emergency response. By combining IoT-enabled sensors, machine learning, real-time communication, and intuitive user interfaces, the systems offer a comprehensive solution that addresses current limitations while setting a precedent for future advancements. These innovations not only enhance safety and reduce risks but also redefine the role of technology in public safety, creating a more secure and responsive environment for

users and responders alike. Through continuous refinement and broader implementation, these systems have the potential to revolutionize vehicle safety and emergency management, ultimately saving lives, reducing property damage, and contributing to safer urban landscapes.

12. CONCLUSION

The combined conclusions of these studies illustrate the transformative potential of integrating advanced technologies such as IoT, machine learning, and real-time communication systems to revolutionize vehicle fire safety and emergency response. This research demonstrates the successful development and implementation of comprehensive solutions that enhance fire detection, real-time monitoring, and emergency management. By seamlessly integrating hardware, software, and user-focused designs, the systems collectively set a new benchmark in addressing critical safety challenges, offering practical, scalable, and innovative solutions for modern urban and vehicular environments. Central to these advancements is the use of IoT-enabled sensors, including temperature, gas, and flame detectors, which consistently captured real-time environmental data with high accuracy and reliability. The preprocessing of this data, facilitated by microcontrollers like Arduino MEGA, ensured that only clean and relevant inputs were used for analysis, significantly improving the systems' effectiveness. Machine learning models such as Convolutional Neural Networks (CNN), Linear Regression, and Support Vector Machines (SVM) further enhanced these capabilities, achieving accuracies above 90% in detecting fire risks and classifying fire severity. The CNN, in particular, excelled at identifying subtle anomalies in real-time data, enabling rapid alerts and predictive insights, while the SVM and Linear Regression models provided granular and actionable information on fire severity, empowering users and responders alike.

The integration of IoT with robust communication protocols, including MQTT and Bluetooth Low Energy (BLE), facilitated low-latency, real-time data transmission, ensuring that these systems could operate effectively even in challenging connectivity environments. Cloud-based infrastructures, such as Firebase Cloud Firestore, enabled scalable and secure storage while supporting real-time synchronization between vehicles, users, and emergency responders. This dual-channel approach underscored the adaptability and reliability of the systems, making them suitable for diverse operational settings. The mobile applications developed as part of these systems played a pivotal role in user engagement and accessibility. Designed with intuitive interfaces, the apps allowed users to register vehicles, monitor real-time data, and

receive instant alerts in the event of a fire. GPS-enabled features provided precise location details, enhancing the coordination between users and emergency services. For vehicle owners in remote areas, the integration of BLE ensured continuous functionality, reinforcing the systems' practicality and inclusivity.

The automation of fire suppression mechanisms marked a significant departure from traditional manual methods, which often rely on human intervention and are prone to delays and errors. Solenoid valve-based suppression systems responded within milliseconds of detecting fire signals, deploying extinguishing agents like CO₂ with precision and speed. This rapid response capability minimized potential damage, safeguarded passengers, and reinforced the systems' reliability in high-pressure scenarios. Furthermore, the systems' dual functionality of optimizing fire truck routing and empowering vehicle owners bridged critical operational gaps in emergency management. The use of advanced routing algorithms, including A*, Dijkstra, and Graph Neural Networks (GNNs), enabled the systems to dynamically adapt to real-time traffic conditions, ensuring fire trucks reached incident sites efficiently even in congested urban environments. Complementary functionalities, such as identifying nearby fire extinguisher stations, empowered vehicle owners to take immediate action during emergencies, reducing reliance on delayed professional interventions and mitigating fire-related risks.

These advancements collectively highlight the systems' ability to redefine emergency response frameworks, demonstrating a significant improvement over outdated manual and static methods. Traditional systems, reliant on manual processes and static routing, often fail to address the complexities of modern urban environments and the rapid escalation of vehicle fires. The proposed solutions not only overcome these limitations but also introduce features that enhance both operational efficiency and user empowerment. By leveraging real-time geospatial analytics, predictive modeling, and AI-driven decision-making, these systems offer a holistic approach to emergency management, addressing both the needs of professional responders and individuals directly affected by emergencies.

Despite their success, the research also identified areas for further improvement. Enhancing predictive capabilities through the integration of advanced machine

learning models, such as deep learning, could improve accuracy and adaptability, enabling earlier detection of fire risks. Expanding sensor networks to include additional parameters, such as cameras or infrared detectors, could provide richer datasets and further refine fire severity assessments. Addressing environmental challenges, such as sensor exposure to extreme weather or debris, remains critical for ensuring long-term reliability. Additionally, optimizing the computational requirements of machine learning models would facilitate real-time processing in resource-constrained environments, making the systems more accessible for widespread adoption.

The scalability and modular design of these systems represent another key strength, enabling seamless integration with existing vehicle architectures and emergency dispatch protocols. This flexibility ensures that the systems remain relevant and adaptable to evolving safety requirements and technological advancements. Collaboration with automotive manufacturers, regulatory bodies, and policymakers could accelerate the standardization and commercialization of these solutions, making them a mandatory feature in modern vehicles. Such initiatives would not only enhance road safety but also foster innovation and growth in the automotive industry, contributing to smarter and more resilient urban infrastructures.

The societal implications of these systems are profound. By empowering vehicle owners with actionable, real-time data and equipping emergency responders with optimized routing and severity assessments, these solutions address critical gaps in public safety. Early detection and rapid intervention significantly reduce fatalities, injuries, and property damage, while fostering a culture of preparedness and awareness among users. The systems' ability to minimize false alarms further enhances trust and reliability, paving the way for their acceptance and integration into everyday life. Moreover, their potential to generate anonymized traffic and incident data could serve as a valuable resource for urban planning and policymaking, aiding in the development of safer and more efficient cities.

In conclusion, the research and development of these advanced vehicle fire safety and emergency response systems represent a significant leap forward in addressing the challenges posed by rapid urbanization, increasing vehicular density, and the growing

complexity of modern emergencies. By combining IoT, machine learning, and real-time communication technologies into cohesive and user-centric frameworks, these systems set a new standard for innovation in public safety. Their success underscores the transformative power of interdisciplinary approaches, demonstrating how technology can be harnessed to solve real-world problems and improve quality of life. While challenges remain, the findings validate the feasibility and effectiveness of these solutions, paving the way for their adoption and further development. As these systems continue to evolve, they hold the promise of making roads safer, saving lives, and setting a precedent for future advancements in vehicle safety and emergency management.

13.REFERENCES

14. APPENDIX