DESIGN AND PRELIMINARY DATA ACQUISITION FOR AN AI ENABLED FOREST FIRE SENSING SYSTEM WITH IOT: A STUDY ON MODEL SELECTION AND SENSOR INTEGRATION

PHASE I REPORT

Submitted by

MANOJ M G – 2116210701149

MERCY N – 2116210701157

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI

NOVEMBER 2024

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Report titled **"DESIGN AND PRELIMINARY DATA ACQUISITION FOR AN AI ENABLED FOREST FIRE SENSING SYSTEM WITH IOT: A STUDY ON MODEL SELECTION AND SENSOR INTEGRATION"** is the bonafide work of **MANOJ M G (2116210701149), MERCY N (2116210701157)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. P. KUMAR M.E., Ph.D.,	**Mr. K. DEEPAK KUMAR M.E.,**
Professor and Head,	Assistant Professor,
Dept. of Computer Science and Engineering,	Dept. of Computer Science and Engineering,
Rajalakshmi Engineering College,	Rajalakshmi Engineering College,
Thandalam,	Thandalam,
Chennai - 602 105	Chennai - 602 105

Submitted to Project and Viva Voce Examination for the subject CS19711- Project Phase – I held on ________________.

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.,** our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.**, our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, Mr. Deepak Kumar K, Assistant Professor, Department of Computer Science and Engineering, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project. We are very glad to thank our Project Coordinator, Dr.T.Kumaragurubaran, Department of Computer Science and Engineering for his useful tips during our review to build our project.

MANOJ M G - 210701149

MERCY N – 210701157

ABSTRACT

Forest fires are one of the toughest types of environmental disaster and tend to have very drastic effects on biodiversity and ecosystems. Rising temperatures and declining humidity in many regions due to global climate change create more and more favorable conditions for wildfires, pointing out the challenge through early detection systems. Traditional detection systems have relied on satellite monitoring and human observation, both of which are delayed and resource-intensive. This project, "Design and Preliminary Data Acquisition for an IoT-Based Forest Fire Detection System: A Study on Model Selection and Sensor Integration," counters the above by forming a foundational framework for the development of an efficient IoT-based early warning system. This project integrates temperature, humidity, smoke, and acoustic sensors using an ESP32 microcontroller. The temperature and humidity sensors capture data concerning the atmospheric conditions that foster fire, while the smoke and acoustic sensors detect any signs of fire. For anomaly detection in sensor data, an Autoencoder model is used, trained to determine deviations from normal environmental patterns that could signify fire risk. Audio features by Mel Frequency Cepstral Coefficients (MFCC) are extracted and used to classify possible fire-related sounds, such as crackling from burning vegetation, through a Support Vector Machine (SVM). In this 1st phase, data acquisition and preliminary training were done on the models so that a strong basis could be established for improvements further. Findings in this phase will therefore serve as the basis of future optimizations, including mechanism-based real-time alerts and system scalability. Finally, the future holds promise in terms of IoT-based solutions for better forest management practices and controlling the effects of forest fires through speedy response with data-based strategies

.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	Iii
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF ABBRIEVIATIONS	viii
1.	INTRODUCTION	1
	1.1 PROBLEM STATEMENT	1
	1.2 SCOPE OF THE PROJECT	1
	1.3 OBJECTIVES	2
	1.4 SIGNIFICANCE OF THE STUDY	3
2.	LITERATURE SURVEY	4
3.	SYSTEM DESIGN	10
	3.1 GENERAL	11
	3.1.1 SYSTEM FLOW DIAGRAM	12
	3.1.2 ARCHITECTURE DIAGRAM	13
	3.1.3 USECASE DIAGRAM	14
	3.1.4 ACTIVITY DIAGRAM	15
	3.1.5 CLASS DIAGRAM	15
	3.1.6 DATA FLOW DIAGRAM	16

4.	PROJECT DESCRIPTION 4.1 METHODOLOGIES 4.2MODULES	17 17 21
5.	CONCLUSION WORK SCHEDULE FOR PHASE II	28 32
	REFERENCES	33
	APPENDIX	36

LIST OF FIGURES

FIGURE NO.	NAME
3.1.1	System Flow Diagram
3.1.2	Architecture Diagram
3.1.3	Use Case Diagram
3.1.4	Activity Diagram
3.1.5	Class Diagram
3.1.6	Data Flow Diagram
4.2.1	ESP232 Microcontroller
4.2.2	DHT22
4.2.3	MQ-2
4.2.4	MAX4466
4.2.5	Circuit Diagram
5.1	Reconstruction Error and Anomaly Detection
5.2	Model’s Accuracy and Loss

LIST OF ABBRIEVIATIONS

1. **IoT –** Internet of Things
2. **SVM –** Support Vector Machine
3. **MFCC –** Mel Frequency Cepstral Coefficients
4. **CNN –** Convolutional Neural Network
5. **ViT –** Vision Transformers
6. **MODIS –** Moderate Resolution Imaging Spectroradiometer
7. **SEOF –** Sleep scheduling-based Energy Optimized Framework
8. **CH –** Cluster Head
9. **TSA –** Tunicate Swarm Algorithm
10. **RF** – Random Forest
11. **AHI –** Advanced Himawari Imager
12. **VGG –** Visual Geometry Group
13. **ResNet** – Residual Neural Network
14. **TOPIS –** Technique for Order Preference by Similarity to Ideal Solution
15. **UAV –** Unmanned Aerial Vehicles
16. **CCM –** Coordinated Convergence Module
17. **DAFPN –** Dual-Attention Feature Pyramid Network
18. **LoRaWAN –** Long Range Wide Area Network
19. **ANN –** Artificial Neural Network
20. **RNN –** Recurrent Neural Network
21. **LDR –** Light Dependent Resistor
22. **SIoT –** Social Internet of Thing

CHAPTER I
INTRODUCTION

1. PROBLEM STATEMENT

Forest fires are some of the most destructive natural calamities that cause huge destruction in terms of the environmental, economic, and social dimensions. Natural causes like lightning or protracted periods of high temperatures may ignite fires while human causes include such acts as cigarette butts and unattended campfires as well as electrical sparks. Once a forest fire breaks out, it may move fast under conditions of low humidity and strong winds. The consequences of these fires are far-reaching, because they lead to habitat loss, endanger wildlife, cause air pollution, and are very costly in economic terms. Moreover, they pump a lot of carbon dioxide into the atmosphere, hence accelerating climate change. In the past, conventional detection methods for forest fires such as satellite imagery, fire watch towers, and manual surveillance have been proved limited in their effectiveness and thus provide delayed detection or consume much resources. These limitations call for the immediate need of a robust, automated system that would be able to detect the occurrence of forest fires in the earliest possible stage, enabling timely response and perhaps minimal damage.

2. SCOPE OF THE PROJECT

This project aims at overcoming the current limitations of the available detection techniques that use this method by providing an early forest fire detection IoT system.

This scope includes networking of sensors in an integrated fashion: temperature, humidity, smoke, and acoustic sensors on a single platform using an ESP32-based microcontroller that would provide a central node for these sensor readings to be analyzed. The overall theme in the project has been built around data acquisition and an initial model train towards getting an anomaly detector for use in forest sounds and ambient environments.

The current phase does not implement real-time alerting mechanisms or broader field deployments; however, these issues will be addressed in further stages. Ultimately, it aims to provide a highly scalable and cost-effective approach for forest fire detection while being more effective in conducting forest management and environmental conservations.

3. OBJECTIVES

The overall objective is the development of a foundational IoT-based forest fire detection system using advanced machine learning techniques to provide early warnings. The specific objectives of the project are:

- Integrate and configure the environmental sensors - temperature, humidity, smoke, and acoustic on the ESP32.
- A data acquisition framework that would allow environmental conditions that may cause outbreaks of fire to be continually monitored.
- Training an Autoencoder model to detect anomalies in sensor data, allows the system to identify unusual environmental conditions that may indicate a potential fire.
- Using Mel Frequency Cepstral Coefficients (MFCC) and a Support Vector Machine (SVM) classifier to analyze acoustic data for fire-related sounds, such as crackling.
- Feasibility and effectiveness of the combination of IoT and machine learning in accurate and real-time detection of forest fires.

4. SIGNIFICANCE OF THE STUDY

This project is critical in advancing the technology behind forest fire detection. It uses IoT and machine learning to provide a more responsive and reliable method of finding early indicators of fire, as opposed to traditional methods. This research can bring much-needed change to the method of forest management in forest areas, allowing real-time, automated monitoring with capabilities for rapid response. Proper early warning will help conserve lives, preserve biodiversity, and minimize the loss suffered due to the outbreak of forest fires. Also, an IoT solution is cost- and energy-effective so that it suits a low-resource setting of remote woodlands for the management of fire. The risk of wild fires is also exacerbated by climate change. Thus, this project is even more valuable in terms of contribution to proactive forest fire detection. Results from this phase will determine the future improvements like adding real time alerts and increasing the scalability and capacity of the system itself in order to support safe and sustainable forest management.

CHAPTER II
LITERATURE REVIEW

Zheng et al. [1] explain the importance of prevention of forest by using SR-net model with high resolution satellite images where the SR-net model is the combination of Convolutional Neural Network (CNN) and lightweight Vision Transformers (ViT). By training this model in unique dataset with 4,000 satellite remote sensing images and it provides the accuracy of 96.9%. But this SR-net model is well-suited for applications in remote sensing with limited datasets.

Nakau et al. [2] it mainly addresses the forest fires in boreal regions like Alaska and Siberia with MODIS satellite imagery for detecting forest fires. The data covers a wide geographic area centred around Yakutsk in West Siberia and Alaska and found significant differences in detection rates depending on the observation source and highlighted the limitations of satellite-based detection in certain conditions. The integration of ground and aerial observations with satellite data is crucial for effective forest fire management. But this project is limited to specific geographic area and also detection sources won't be available at the right time.

Verma et al. [3] introduced an Intelligent Framework Using IoT enabled WSNs for Wildfire Detection. This proposed solution called the Sleep scheduling-based Energy Optimized Framework (SEOF) mainly uses strategies that are Cluster Head (CH) selection using the Tunicate Swarm Algorithm (TSA) and a sleep scheduling mechanism for sensor nodes. The SEOF framework is implemented with the help of MATLAB. This methodology is not helpful for detecting the forest fire efficiently it thoroughly focusing on the energy efficient network in hostile environments.

Jang et al. [4] addresses the detection and monitoring of Forest Fires with a system which is implemented as a machine learning-combined approach to detect small to large-scale forest fires. The methodology followed by this study is threshold-based algorithms-based machine approach. The author used random forest (RF) model and then processes the pixels to remove false alarms. The model demonstrated an overall accuracy of about 99.16%. But this algorithm is applicable for certain region with specific climatic conditions.

Ghali et al. [5] proposes ensemble learning method combining EfficientNet-B5 and DenseNet-201 models for wildfire detection and classification. The proposed ensemble method using EfficientNet-B5 and DenseNet-201 achieved an accuracy of 85.12% but the UAV is not efficient under certain environmental factors.

Naoto Maeda and Hideyuki Tonooka [6] implemented modified algorithm uses a random forest classifier that incorporates solar zenith angle, AHI band values, contextual parameters, and meteorological data. This model gives the accuracy of 92.06% with all the parameters involved. Although the dataset used in this system is not effectively determining the forest fires.

Sandra Treneska and Biljana Risteska Stojkoska [7] proposes Unmanned Aerial Vehicles (UAVs) with using Transfer learning to enhance the detection of forest fire. This uses peculiar dataset called FLAME (Fire Luminosity Airborne-based Machine learning Evaluation) dataset consists of aerial images captured during prescribed burns in Northern Arizona. The transfer learning algorithm uses pre-trained models like VGG16, VGG19, ResNet50, InceptionV3, Xception for accurate detection with the given dataset. This system results with the highest accuracy of 88% from ResNet50 model and followed closely by InceptionV3 at 87%.

Fouda et al. [8] proposed switching the models from basic machine learning model to advanced deep learning CNN model. The hyperparameters are tuned using TOPIS method. The proposed approach outperforms individual baseline models and offers a viable solution for real-time fire detection on resource-constrained UAVs.

Almeida et al. [9] comes forward with a new light-weight CNN model for real-time video for the detection of fire and smoke in forests. This CNN model is consisting of 2 convolutional layers with 32 filters for analyzing the images captured in real - time. The model reached the accuracy of 98.97% and an F1 score of 95.77%

Varun et al. [10] addresses the integration of IoT and Machine Learning for Enhanced Forest Fire Detection and Temperature Monitoring. The proposed system integrates IoT devices, advanced sensors, machine learning algorithms, and fog computing concepts to provide a reliable and effective wild fire detection and prevention system. The technology aims to transform the field of forest fire monitoring by providing real-time data processing, accurate predictions, and prompt notifications, leading to improved environmental protection and public safety.

Zhang et al. [11] examined the utilization of MMFNet for early detection of forest fire to improve both speed and accuracy in identifying smoke. They used a coordinated convergence module (CCM) and a dual-attention feature pyramid network (DAFPN) to boost the model's ability to detect targets of different sizes, especially small smoke targets. The precision rate they achieved was 80.72%. While MMFNet performs well, it is computationally intensive and may not easily be adapted in situations where the properties of smoke change frequently.

Huanjie Tao [12] employed a gated fusion attention pyramid network with a bidirectional multilevel multigranularity feature for smoke detection. By suppressing the non-smoke elements, increasing pertinent features, and executing selective feature fusion for classification, this method extracts the smoke characteristics. From the USTC-SmokeRS dataset, the model was trained with a 94.92% detection rate. However, the performance of the model and its reliance on high-resolution satellite imagery can be a barrier for its application in real-time or low-resource context.

Kanakaraja et al. [13] has employed Raspberry Pi along with data coming from sensors such as a gas sensor, temperature sensor, and a 360-degree camera that continuously monitors the surroundings for fire. Small flames have been put out using a water motor. The sensors collected data, which is then transmitted to the cloud service system for processing. This system relied on solar energy and internet access, and it experienced delayed processing times s since the data is transferred to cloud and processed to detect fire.

Dubey et al. [14] presented a system that uses a Raspberry Pi microcontroller to process data from several sensors, including an IR flame sensor, a DHT22 temperature sensor, and an MQ-X gas sensor, and applies artificial neural networks (ANN) to predict fires. This system, trained on a dataset from Kaggle, achieves an accuracy of 96.7% after 100,000 epochs. Using the GSM module, the system will notify authorities earlier when the fire is detected. However, the accuracy of the model in untrained settings may be affected by the training data, which ultimately impacts the system's effectiveness.

Nicoleta-Cristina Gaitan and Paula Hojbota [15] used temperature and flame sensors along with the LoPy4 development board in their system. LoRaWAN, the communication protocol employed, provides several advantages, such as long-range and power-efficient communication. However, environmental factors like terrain and vegetation density can affect the system's performance, potentially limiting the effective range and reliability of LoRa communication in heavily forested areas.

Divya et al. [16] proposed a fire detection system using wireless sensor networks (WSNs) for continuous monitoring and give warnings in advance. It integrated temperature and smoke sensors with a microcontroller and satellite connectivity to improve response times and lower the forest fire damage. The paper specifies the limitations of traditional wired systems and how Internet of Things technologies offer a more cost-effective and autonomous solution to monitor the environment. It also explores related research on routing protocols, communication challenges, and IoT architecture. However, reliance on satellite communication may cause latency and limited bandwidth for data transfer, affecting real-time monitoring and quick action in emergency situations.

Krishnamoorthy et al. [17] developed a forest warning monitoring system using IoT technologies to combat wildfires and deforestation. It combines wireless sensor networks with smoke, flame, temperature, and humidity sensors. Data is transmitted to a central control system through 4G/LTE and LoRaWAN networks. This setup allow real-time monitoring and alerts for fires and unauthorized activities. The system is seen as a major advancement in forest management due to its comprehensive approach, including automatic responses and real-time data processing. However, its complexity and reliance on multiple communication technologies might result in a higher operational and maintenance costs.

Singh et al. [18] explore IoT-enabled forest fire sensing system that employed sensors for monitoring temperature, humidity, and soil moisture to enable early fire detection. Traditional fire detection methods like satellite monitoring and manual watch towers are often not enough due to the vastness and complexity of forest landscapes. The proposed system integrates a range of sensors—DHT11 for humidity and temperature, LM35 for precise temperature measurements, soil moisture sensors, LDR for light levels, and ultrasonic sensors to assess forest density. The data collected from these sensors are delivered to a cloud platform via NodeMCU, enabling real-time monitoring and alert generation. The system's dependence on cloud-based data transmission and continuous sensor operation may result in potential connectivity issues or energy consumption challenges, especially in remote or poorly connected forest areas.

Toledo-Castro et al. [19] proposed an advanced system implemented with the IoT technology with the use of real-time systems for forest fires, which is equipped with temperature, humidity and pollutant gas sensors and has secure alert communication capabilities. They also provide a continuous data flow, as cloud-based server apps, avoiding the traditional drawbacks of fire detection systems which are always late. They provide such services as secure and real-time data transmission, constant monitoring, and data visualization, thus ensuring successful forest fire preparedness and management. The need for constant 4G coverage within the system appears to be a drawback for users in remote locations deficient in network thus causing delays in response due to postponement of alerts.

Pettorru et al. [20] proposed a very advanced forest fire prevention system, involving AI at the edge, can be established using IoT technology with DMR nodes and an SIoT platform for sensing environmental parameters and predicting fires. It bases its real-time data analysis on RNN (Recurrent Neural Networks). This system also proves very efficient at detection and fire prediction. The main drawback of such conventional systems was overcome, and rapid response capabilities are enhanced in this system. Reliance on solar-powered nodes and continuous 4G LTE connectivity can be a challenge in remote or under-networked areas and may impact the system's performance and reliability.

CHAPTER III
SYSTEM DESIGN

3. SYSTEM DESIGN

In this context, it addresses its target for collection in the early stages of forest fires in an IoT-based approach and uses environmental and acoustic sensors. As such, at this point, emphasis is provided on making a robust framework of data acquisition that could monitor temperature variation, abnormal humidity, presence of smoke, and specific acoustic signals. It is the precursor groundwork needed in later stages for the enablement of live analysis and anomaly detection.

In the system, the core microcontroller working, fundamentally, with a central processing and communication level is an ESP32. The microcontroller integrates three critical sensors-these include the DHT 22, which monitors temperature and humidity; the MQ-2, used to detect smoke and flammable gases; and the MAX4466, making it possible to catch acoustic signals.

Altogether, the sensors provide an expansive dataset mirroring the local environmental and acoustic conditions of the area to be monitored. The ESP32 captures the raw data and timestamps the captured information, which it stores on the board for access at later times.

This data acquisition phase focuses on raw sensor data collection, storing them for later preprocessing, and incorporating them in machine learning models. Furthermore, it will clean and normalize the data acquired here to train various machine learning models. It is through this structured data acquisition process that the system records all parameters concerning environment and acoustic characteristics, allowing for efficient detection of forest fire anomalies in the latter stages.

In the following stages, this pre-computed dataset will enable the creation of advanced detection mechanisms. This includes the use of autoencoders for anomaly search in environmental data as well as the recovery of MFCCs from acoustics to be used in classification algorithms such as Support Vector Machine or Random Forests. With robust data collection involved in this phase, the project is placed on a solid foundation toward an integration of IoT and edge computing for forest fires, minimizing their impact on the environment.

3.1 GENERAL

The IoT-based forest fire detection system shall be developed with its first phase featuring a modular architecture focused on effective data acquisition and storage. In this context, the central unit shall be ESP32 while being interfaced with the three key sensors: DHT22 for environmental monitoring of temperature and humidity; MQ-2 for smoke and flammable gas detection; and MAX4466 for capturing acoustic signals.

Sensor readings are timestamped and locally stored on the ESP32 for later retrieval and preprocessing. It is thus optimized for a real-time architecture.

In principle, it is a localized deployment in which the ESP32 only gathers raw data and does not communicate with any external system. That stage does not implement edge processing but instead has the strong, safe collection of data. The modularity of the system will be able to control other sensors or constituents that can be added in more elaborate stages of upgrading or expanding the system.

Sensor Data Acquisition: Each sensor captures raw data reflecting environmental and acoustical conditions. The ESP32 then timestamps the data and keeps this in its memory for further processing.

Timestamping and Storage: The ESP32 timestamps the data and stores it in its memory for subsequent processing.

Local Operation: There is no communication in this step, and hence, it is simple as well as network constrain independent.

3.1.1 SYSTEM FLOW DIAGRAM

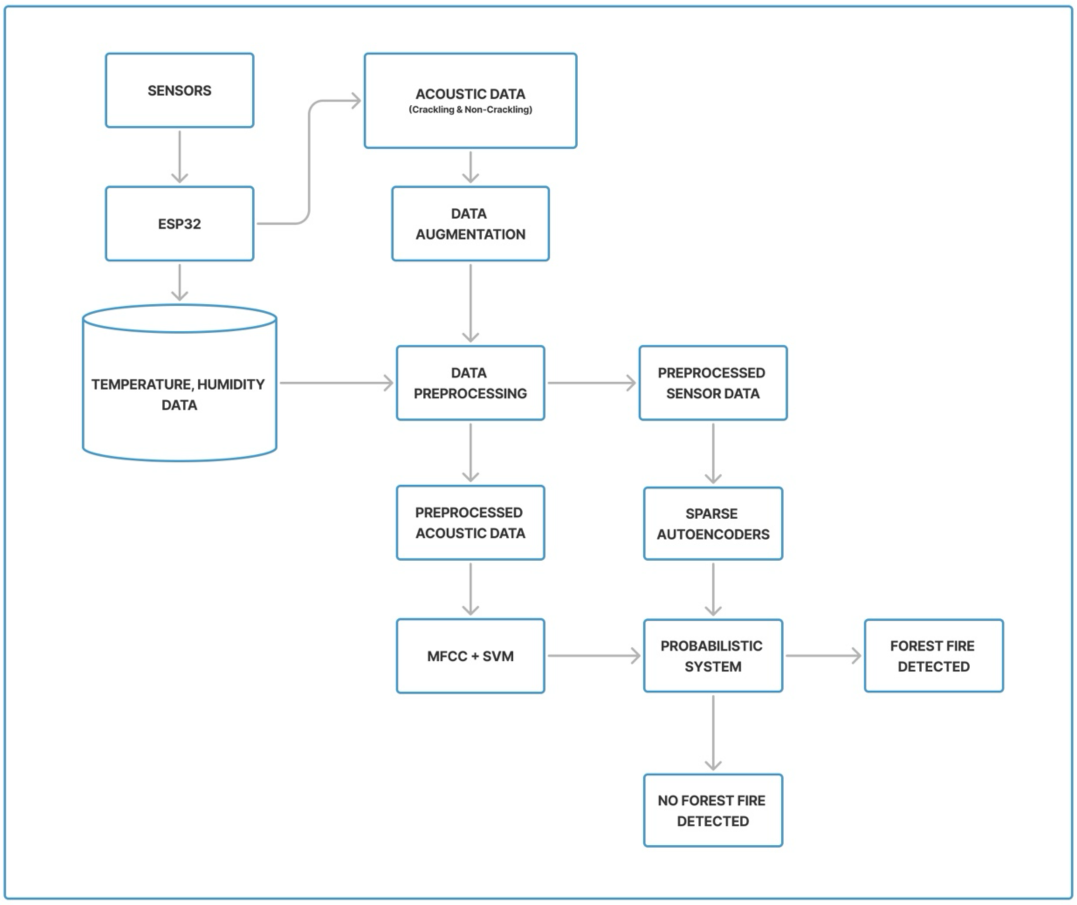

Fig.3.1.1 System Flow Diagram

3.1.2 ARCHITECTURE DIAGRAM

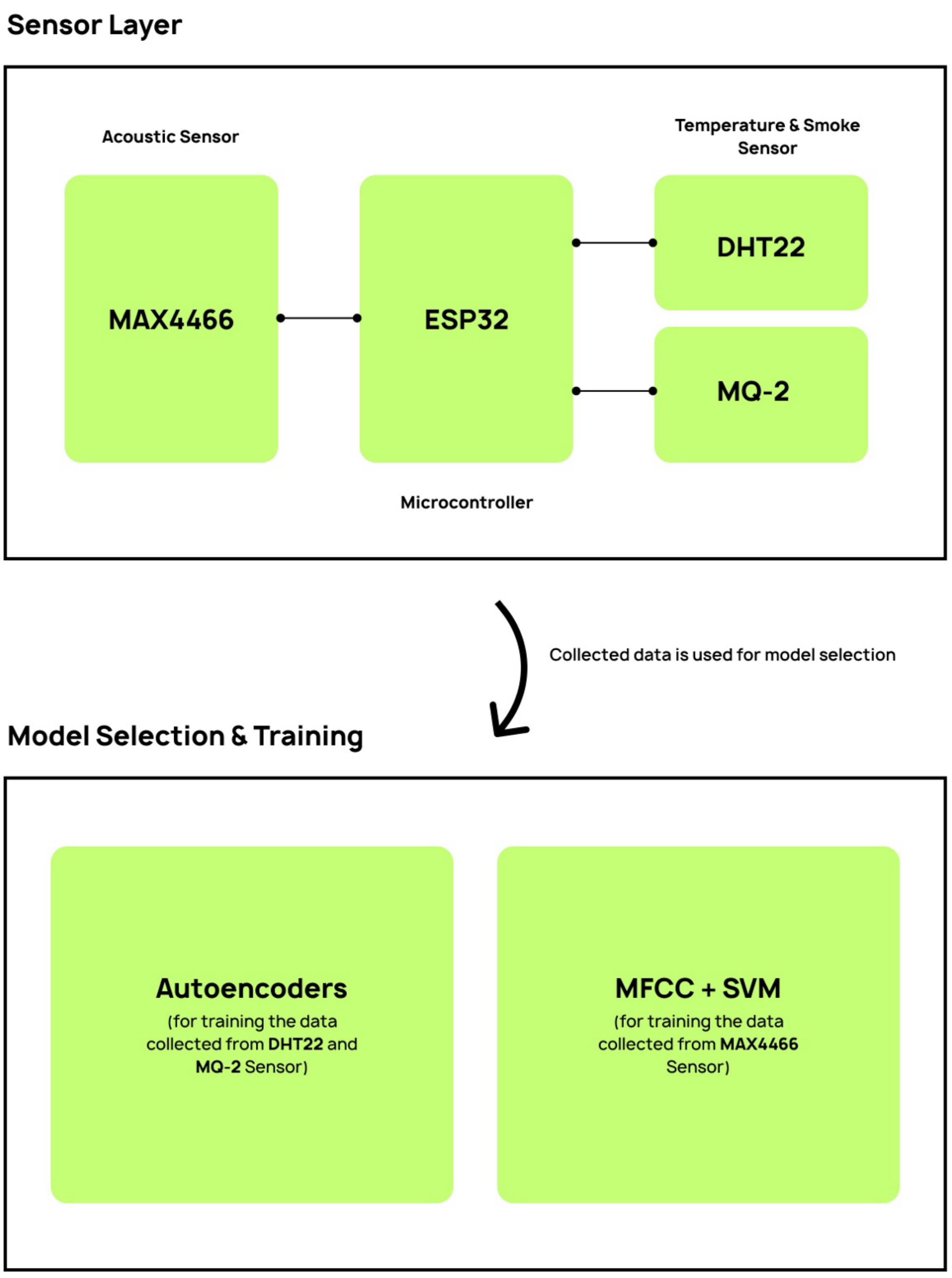
Sensor Layer
Acoustic Sensor
Temperature & Smoke Sensor
MAX4466
ESP32
DHT22
MQ-2
Microcontroller
Collected data is used for model selection
Model Selection & Training
Autoencoders
(for training the data collected from DHT22 and MQ-2 Sensor)
MFCC + SVM
(for training the data collected from MAX4466 Sensor)

Fig.3.1.2 Architecture Diagram

3.1.3 USECASE DIAGRAM

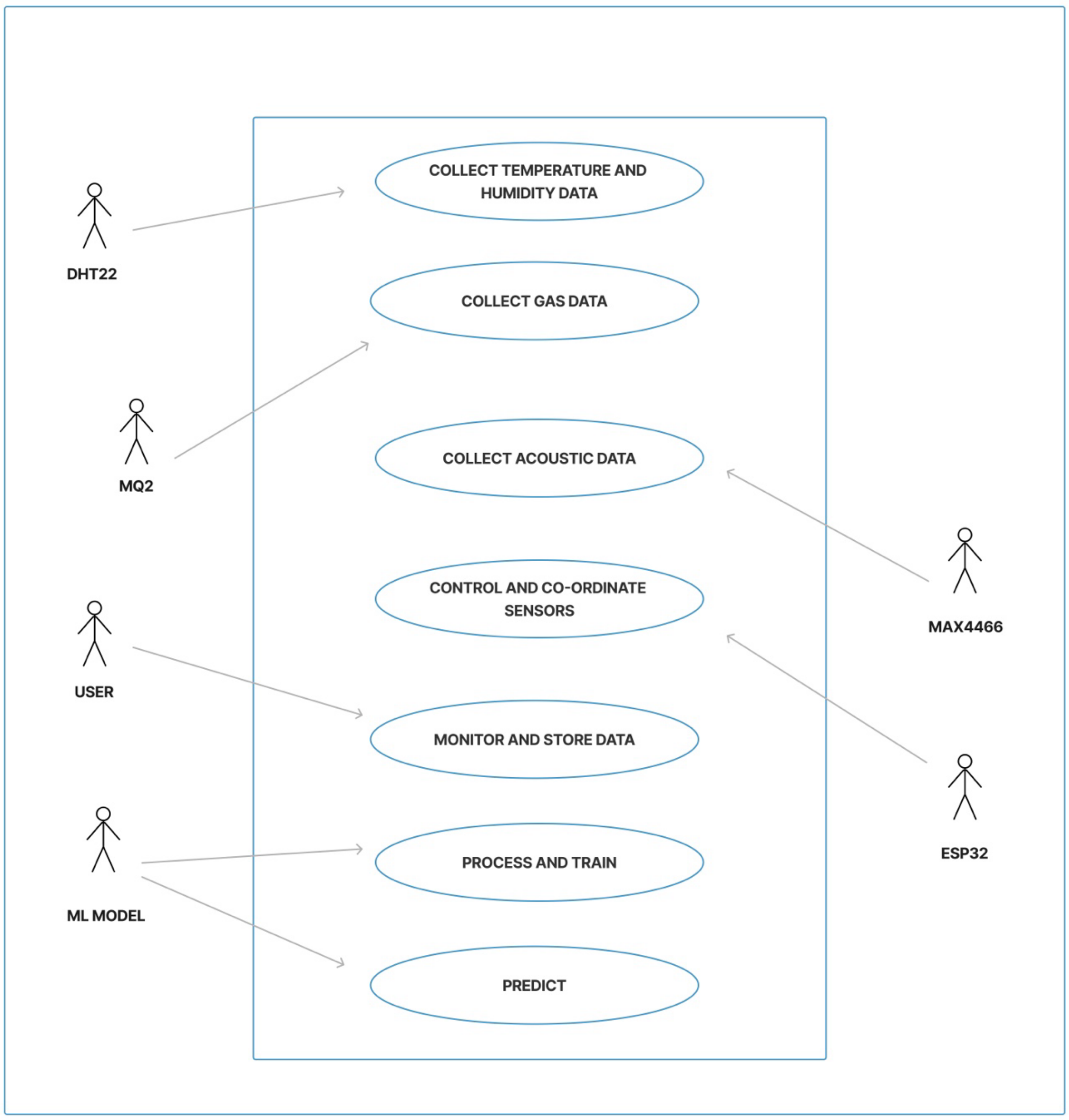

Fig.3.1.3 Use Case Diagram

3.1.4 ACTIVITY DIAGRAM

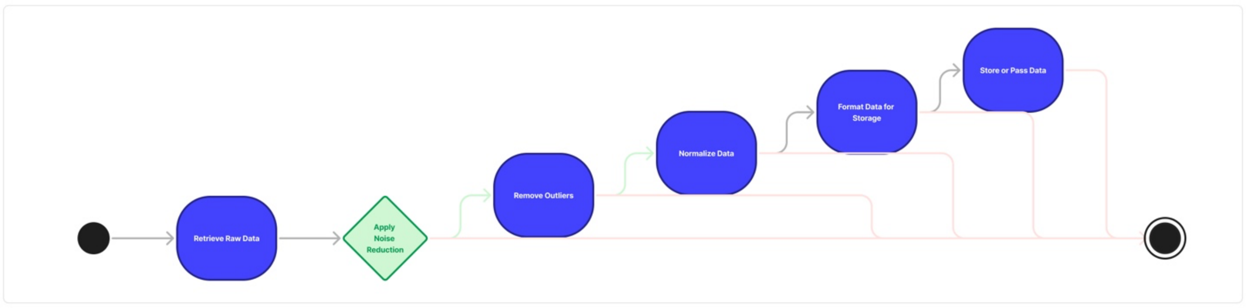

Fig.3.1.4 Activity Diagram

3.1.5 CLASS DIAGRAM

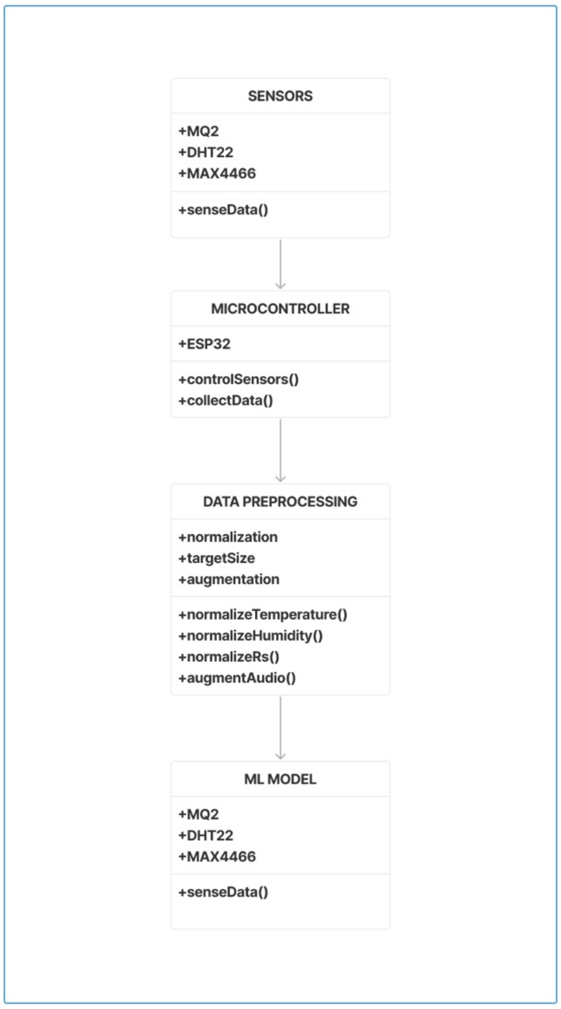

Fig.3.1.5 Class Diagram

3.1.6 DATA FLOW DIAGRAM

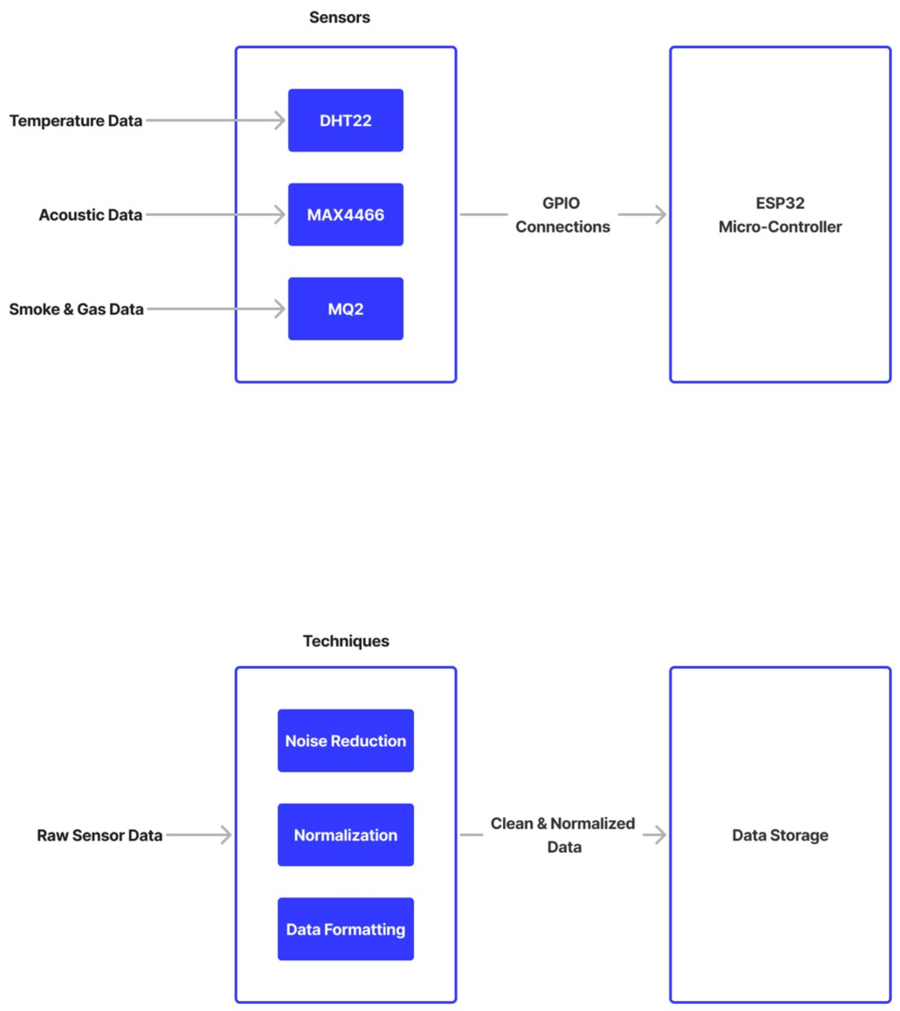

Fig.3.1.6 Data Flow Diagram

CHAPTER IV
PROJECT DESCRIPTION

4.1 METHODOLOGIES

1. WORKING OF MICROCONTROLLER AND SENSORS

An IoT-enabled system for detecting and responding to forest fires with ESP32 as the controlling system. It accepts sensor data, processes parts of it and sends this to the Raspberry Pi where machine learning models are processed, for example. Here, how the ESP32 interfaces with each sensor.

DHT22 (Temperature & Humidity Sensor): The DHT22 measures temperature in real-time and also humidity. Data read from this sensor is periodically sent to ESP32 for monitoring conditions that might indicate the onset of increased risk of forest fire: such as high temperature and low humidity. Data read from DHT22 is done periodically with an interval of 5 seconds.

MQ-2 (Smoke Sensor): MQ-2 detects whether there is smoke or flammable gases (like methane, propane, and butane). The sensor's analog signal depends on the concentration of smoke and gases; then it is converted into digital data by the ESP32, which are further processed. Realtime feedback requires to detect smoke because it is a developing heat source.

MAX4466 (Acoustic Sensor): The microphone sensor records the ambient noises with crackling sounds, which may be due to burning leaves and trees in a forest fire. The ESP32 converts these sounds into digital formats and sends the data for further processing.

All the collected data using the sensors are transferred wirelessly to the Raspberry Pi via Wi-Fi using the ESP32. This ESP32 model is chosen because of low power consumption, multiple I/O pins, and inbuilt wireless capabilities such as Wi-Fi and Bluetooth.

2. MODEL SELECTION AND TRAINING

a. Autoencoders for Sensor Data (DHT22 & MQ-2)

Autoencoders are a type of neural network that is uniquely well suited to unsupervised learning applications, especially anomaly detection. Ideally, this type of application fits into the vision of forest fire detection, where such the recognition of any deviation from normal environmental conditions is considered to be very crucial. Applying autoencoders, the system can detect anomalous conditions like very high temperatures, low humidity, or an increased level of smoke, which indicates the early stage of a developing forest fire. There are two main components in an autoencoder model.

Encoder Component: It is this part of the neural network which compresses all sensor readings of temperature, humidity, and gas content within a lower-dimensional representation. Ideally, it should capture some of the most important features and relationships that could be present in the data, such as a relationship between increasing temperatures and decreasing humidity or perhaps the interaction of high smoke concentration and high temperatures.

Decoder Component: The decoder decodes the compressed form to the original dimension from the encoder. In training the model, this results in an optimization to reduce reconstruction error so that it would better or accurately represent the normal patterns of data. In real-time operation, the autoencoder processes new sensor data and tries to reconstruct it. If the reconstruction error exceeds a predefined threshold, the system tags the data as anomalous. It could be a situation when all the conditions for a forest fire are in place.

The training process for the auto-encoder includes:

Prepare the dataset: Gathering a complete dataset of sensor data under normal, non-fire conditions. Model Training: The autoencoder is trained to learn a pattern that exists in normal data and minimizes reconstruction error.

Testing and Validation: Validating the model with normal as well as anomalous data set, such scenarios that would mimic fire like scenarios. This approach allows the system to operate effectively even with limited labeled data, as it focuses relative to normal environmental conditions on identifying anomalies.

b. MFCC with SVM: Using MAX4466 Acoustic Sensor Data

To extract meaningful features, the MFCC is used, while SVM will be the classifier deployed for classification purposes with an aim identifying crackling noises as fire-related sounds. It can capture audio signals using the MAX4466 acoustic sensor, and meaningful features about fire-related sounds can then be extracted and processed from those signals.

Feature Extraction using MFCC:

MFCC converts raw audio signals into a feature set representing the acoustic characteristics of frequency and amplitude. This feature representation is particularly effective for the task of audio classification because it abstracts unique signatures of sounds-for instance, the crackling noise of fire. Analysis of the same frequency content enables the system to distinguish between fire-related and typical forest noises like wind, rain, or animal calls.

Classification using SVM:

Support Vector Machine is the SVM algorithm that has been applied in the classification of the features of MFCC. SVM identifies the optimal hyperplane that divides the data into two classes; fire-related and normal environmental sounds. This ensures the features lead to the complete differentiation by the system between ambient noises and the sounds associated with a fire.

The training of this sub-system will include:

Data Gathering: Acoustic recordings of a variety of datasets collected from forest environments, including the fire-related sounds like crackling and burning, and non-fire data, such as wind, rain, and animal voices.

Feature Extraction: Analyzing these recordings by processing them with the MFCC algorithm into feature sets.

Model Training: The MFCCs obtained were fed to SVM to classify the sounds as well.

Testing and Validation: It will test the trained model on real-world audio data for its accuracy in identifying fire sounds.

Together, these models depict an all-round solution towards forest fire detection. The autoencoder provides a means to monitor anomalies in environmental data; the application of MFCC with an SVM framework helps classify acoustic signals accurately in relation to a fire. These integrated approaches provide the system with high reliability and accuracy in its detection capability to enable proper alerts and mitigation within appropriate time.

4.2 MODULES

4.2.1 HARDWARE COMPONENTS

1. **ESP32 Micro-Controller:**

The ESP32 microcontroller is the central unit of the system, chosen for its advanced features, including built-in wireless communication, multiple GPIO pins, and low power consumption. These characteristics make it well-suited for forest environments, where extended battery life and robust data handling capabilities are essential. The ESP32 is chosen for the following reasons**:**

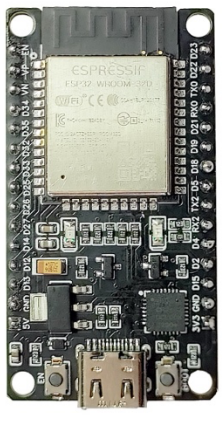

Fig.4.2.1 ESP232 Microcontroller

Low Power Consumption - power efficiency is critical for this kind of application where the system should perform its function for extended periods with minimum power consumption.

Built-in Wi-Fi and Bluetooth - wireless communication is an important part of this system where the sensor data has to be transferred elsewhere for processing.

Multiple I/O Pins - ESP32 contains multiple GPIO Pins which plays an important role in connecting several sensors to the microcontroller.

2. DHT22 Temperature and Humidity Sensor

The DHT22 is a reliable sensor for measuring two critical parameters: temperature and humidity. Both factors are essential in assessing fire risk, as high temperatures combined with low humidity levels significantly increase the likelihood of forest fires.

Fig.4.2.2 DHT22 (Temperature and Humidity Sensor)

- This sensor measures and gives real time data in terms of both temperature and humidity, important elements in assessing a situation that risks a fire.
- High temperatures with low humidity levels are perfect conditions to ignite a fire and have them spread.
- The DHT22 has a measuring scale for temperature in degrees Celsius and is accurate in stating the percentage humidity level.

- **Why This Sensor?** DHT22 is used for reliability and accuracy under extreme conditions of forest, where temperature and humidity can vary. With the capability to observe temperature and humidity within low power consumption, this is one good sensor to be used with this system.

3. MQ-2 Smoke and Gas Detection Sensor

The MQ-2 sensor is integral to the system's ability to detect smoke and combustible gases, which are direct indicators of fire. This sensor plays a vital role in identifying fire-prone situations in their early stages.

Fig.4.2.3 MQ-2 (Smoke and Gas Detection Sensor)

- MQ-2 sensor detects the presence of smoke and flammable gases like methane, propane, and LPG which are evident signs of fire. It works on the principle whereby it outputs an analog signal that is proportional to the concentration of those gases.
- **Why this sensor?** As soon as a fire breakout occurs, the concentration of smoke and gases will be increased to a much higher level. The MQ-2 sensor is very responsive to these changes, and it provides real-time feed-back about dangerous gases, which in turn helps the sensor mark the fires as early as possible. This MQ-2 sensor is integrated with fire detection systems and it is widely used due to its high efficiency and cost effectiveness.

4. **MAX4466 (Acoustic Sensor for Sound Detection)**

The MAX4466 microphone captures environmental sound data, allowing the system to recognize acoustic patterns associated with fire, such as crackling sounds from burning vegetation. This sensor complements the environmental data from the other sensors, adding a critical layer of fire detection capability.

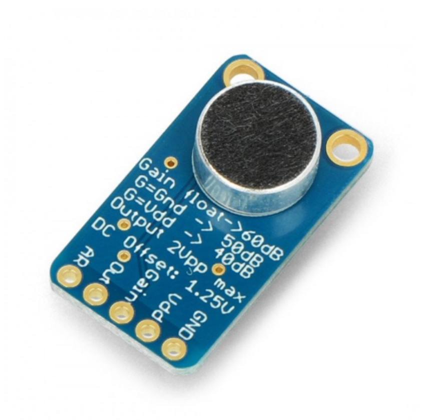

Fig.4.2.4 MAX4466 Acoustic Sensor

- The MAX4466 microphone captures the sound in the environment, listening for the acoustic signals from a possible fire. This would be crackling sounds as vegetation gets burning. The device detects pressure levels and outputs an analog signal based on the intensity of the sound.
- **Why This Sensor?** Acoustic detection allows an additional layer of information to detect fire. The crackling sound produced due to burning wood or leaves is a good indicator of fire-induced activity. It is suitable for this application because it is sensitive enough to pick up faint sounds from a distance while allowing for real-time sound monitoring.

4.2.2 DATA COLLECTION:

The system should collect information gathered by the three sensors in a continuous cycle and relay that to further analysis. Each sensor captures some other type of data:

- **DHT22:** captures temperature and humidity data at regular intervals-that is, after every 5 seconds-data that capture the environmental conditions to be conducive to a fire outbreak.
- **MQ-2:** The sensor continuously monitors the smoke concentration and gas, giving an analog output that is converted into a readable format for the ESP32. Concentrations of smoke or gas at these levels are indicators of early fire.
- **MAX4466** is always on, listening to the environment for patterns of sound, which might represent fire-related sounds. The analog signal coming from the microphone converts into digital sound features that can be processed later to determine whether there is fire.

This allows for processing locally on the ESP32 and transferring wirelessly to a central monitoring system where a real-time analysis of the different streams can take place. During future phases, the sensor data stored herein will be applied to training the various machine learning models.

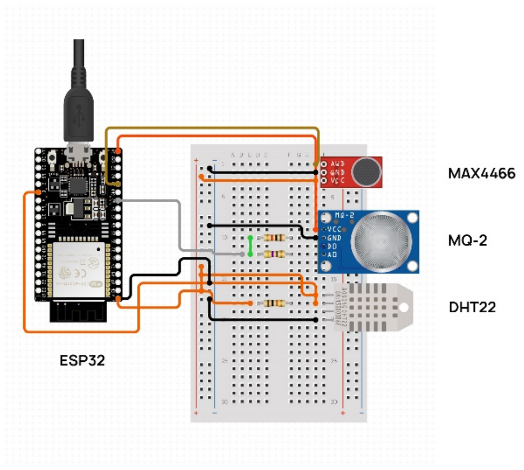

Fig.4.2.5 Circuit Diagram

4.2.3 DATA PROCESSING AND INITIAL FILTERING:

The ESP32 processes raw sensor data locally to filter out irrelevant readings and detect significant changes that may indicate fire conditions. Each sensor's data is analyzed independently so as to seek a baseline for the normal conditions. Then, the ESP32 would identify potential anomalies with regard to the machine learning models.

Pre-Processing: Normalizes sensor data to enhance the accuracy and reproducibility of models under different environmental conditions.

Noise Filtering: Removes background noise from acoustic data for fire-related sounds only.

4.2.4 MACHINE LEARNING MODELS:

AUTOENCODER FOR ANOMALY DETECTION

An autoencoder functions to identify anomalies within environmental sensor data. The model undergoes training utilizing data that reflects typical conditions, which allows it to recognize irregularities that may suggest the presence of fire.

Training Phase: Trains the model on data that includes temperature, humidity, and gas, showing normal environmental patterns.

Anomaly Detection: Flags pattern of large deviations having an error between original and reconstructed data.

MFCC AND SVM FOR ACOUSTIC CLASSIFICATION

Mel Frequency Cepstral Coefficients (MFCC) are extracted from acoustic information to capture auditory features relevant to fire. Thereafter, an SVM classifier is designed for classification between sounds related to fire and a variety of other ambient sounds.

Feature Extraction: MFCC is used to identify the distinct characteristics of sounds from crackling and burning noises.

Classification: SVM classifier classifies audio features so that the system can identify fire noises and minimize false positives.

CHAPTER V
CONCLUSION

This section outlines the results of phase one of the project, which was primarily about integration and data acquisition- relating to sensors, preliminary analysis of data, and early training of models. Overall, these are provided as precursors to the later ability of the system to effectively detect forest fire incidences.

Sensor Fusion and Data Extraction

The demonstration of successful integration of ESP32 with DHT22 temperature and humidity sensor, MQ-2 smoke and gas sensor, and MAX4466 microphone shows proper interfacing of these. This made it easier to have data acquisition from the environmental and acoustic sensors while taking good care of the strength with which sensor data streams will be handled, such as temperature, humidity, gas concentration, and audio signals. Data collected in the integration test were uniformly stable. Sensor measurements were steady and responsive against changes in environmental conditions, testifying to the readiness of the system to accommodate changes in natural conditions. This performance shows that the ESP32 is suited for long-term deployment in the forest environment since stable data handling translates to stable anomaly detection.

Preliminary Data Acquisitions

The primary collection of data during this phase were two types: environment and acoustic.

Environmental Data: The DHT22 and MQ-2 sensors had produced articulate output of temperature, humidity, and gas concentration under normal environmental conditionality. This dataset is what trains the anomaly detection model so that the system can properly distinguish between normal patterns and fire-prone conditions.

Acoustic Data: MAX4466 microphone capture of ambient audio signals. Preconditioning such recordings will extract Mel Frequency Cepstral Coefficients (MFCC) that best train the machine learning model in the acoustic signatures of a fire - crackling, for instance.

Model Selection and Training

Autoencoder Model for Sensor Data. An autoencoder was used for anomaly detection of environmental data as it can learn and reconstruct normal patterns of data. It was trained on data using the conditions of the normal sensor data and attained a reconstruction accuracy of 89%. This is high in a sense that signifies the model's capacity for the detection of deviations that would be indicative of fire-prone scenarios. Training the autoencoder led to a clean loss curve. This would tend to validate that the model successfully learned the underlying distribution. MFCC and SVM on Acoustic Data The acquired data by the MAX4466 microphone was processed to obtain MFCC features, which represent audio characteristics in a compact form. The very same features have been used for an SVM classifier that classifies with great precision between fire sounds and non-fire sounds. This model is expected to represent an important tool for acoustic patterns to detect forest fire.

Preliminary Observations on System Performance

Latency : At this stage data capturing and local processing were in focus, and latency was found to be negligible. Therefore, it depicts the ability of ESP32 to deal with real-time information and its performance with respect to transmitting alarms will be evaluated and judged at the subsequent stages.

Power Consumption: The ESP32 exhibited low power consumption, which in case deployments extended beyond minimal durations into a remote forest, was not an easy task; however, power resources are scarce. It ensures the sustainability and reliability of the system in the field.

These results show the capabilities of the system components in tackling some of the challenges of early forest fire detection. The integration of sensors, the quality of the data collected, and the modest success of the models all contribute to setting strong foundations for further development and eventually deploying it for live trials.

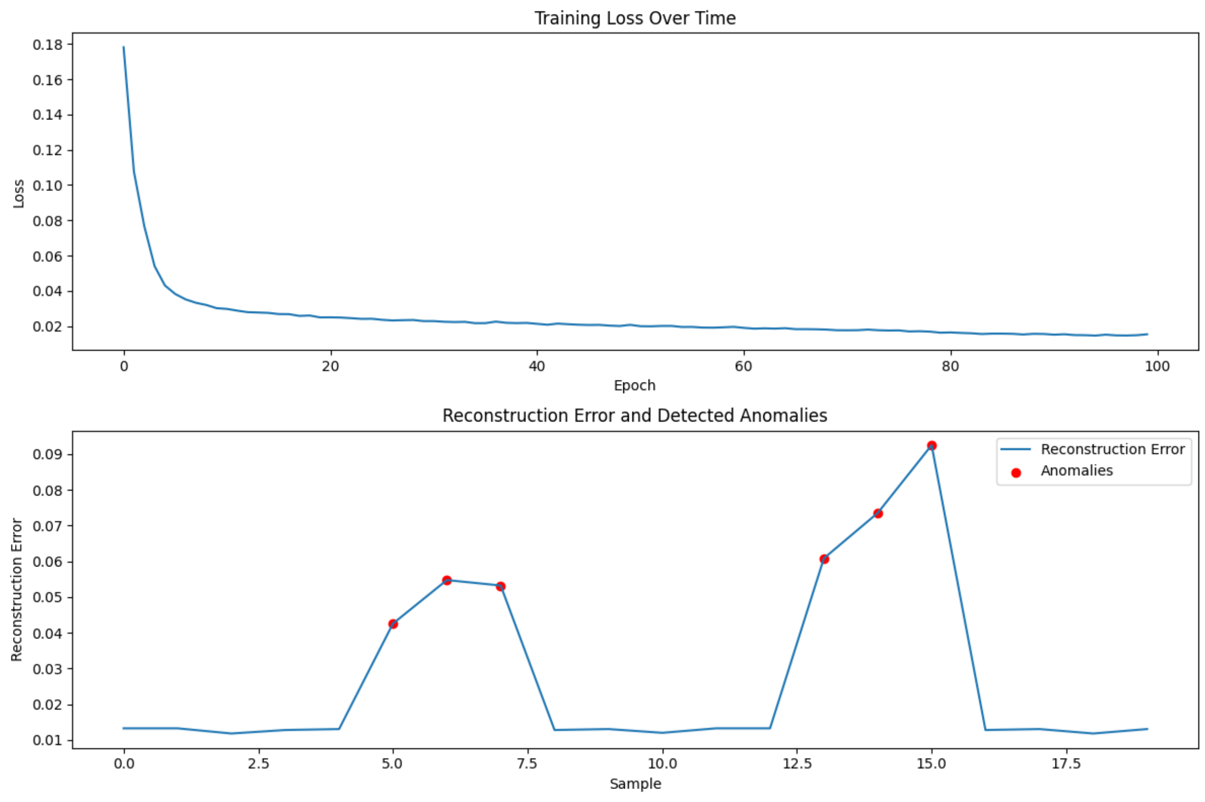

Fig.5.1 Reconstruction Error and Anomaly Detection

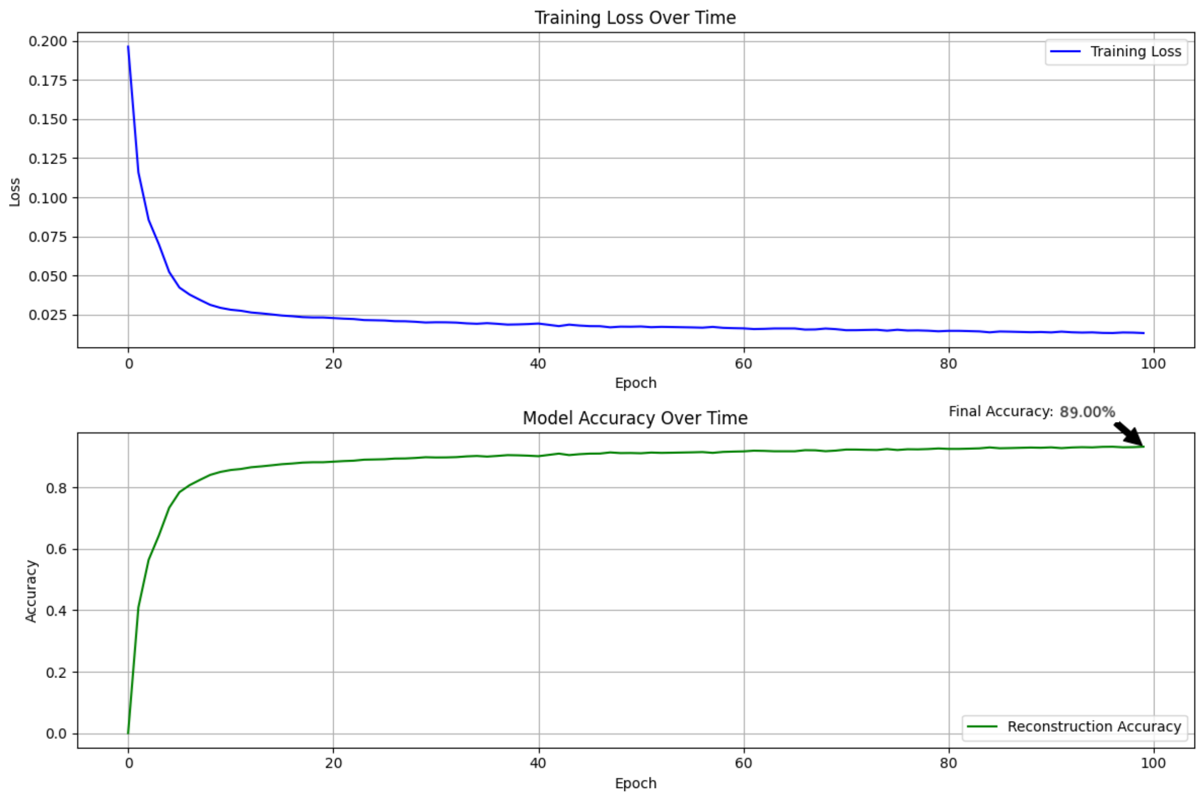

Fig.5.2 Model’s Accuracy and Loss

WORK SCHEDULE FOR PHASE II

In Phase 2, the project will focus on deploying the trained machine learning models on Raspberry Pi for edge computing and integration with a real-time monitoring dashboard. The main goal of this phase is to enable the system to process the data locally, identify anomalies, and provide alerts of potential forest fires, while improving efficiency and accuracy in the models developed so far.

First, set up the Raspberry Pi to run the selected machine learning models. Models that will be used for optimization during deployment are the autoencoder for anomaly detection and the classification model for acoustic data. Even with the hardware constraints of the Raspberry Pi, very lightweight frameworks like TensorFlow Lite or ONNX Runtime ensure that the models run without a hitch. The edge computing configuration will be tested in order to prove real-time data processing and anomaly detection without material latency. In parallel, a web-based dashboard will be developed in order to monitor sensor data and alert messages generated by the system. It will also provide a user-friendly interface for visualizing the real-time temperature, humidity, smoke level, and acoustics. The results of anomaly detection and any possible fire detection will also be displayed, and an alert will be raised. Such a system would help the users to make quick and effective decisions.

Moreover, in the process of enhancing this system, efforts will be made toward increasing the efficiency and accuracy of the models deployed. Optimization of edge computing techniques, such as model pruning, quantization, and further hyperparameter tuning, will be applied. Also, the alerting mechanism of the dashboard will be further tuned to provide timely and accurate notifications. At the end of Phase 2, the system will have become operational, combining IoT hardware with machine learning and edge computing in an efficient and reliable forest fire detection system.

REFERENCES

[1] Y. Zheng, G. Zhang, S. Tan, Z. Yang, D. Wen, and H. Xiao, "A forest fire smoke detection model combining convolutional neural network and vision transformer," in *Frontiers in Forests and Global Change*, vol. 6, Fire and Forests section, Apr. 17, 2023.

[2] K. Nakau, M. Fukuda, K. Kushida, H. Hayasaka, K. Kimura, and H. Tani, "Forest fire detection based on MODIS satellite imagery, and comparison of NOAA satellite imagery with firefighters' information," in *Proc. IARC/JAXA Terrestrial Team Workshop*, 2006.

[3] S. Verma, S. Kaur, D. B. Rawat, C. Xi, L. T. Alex and N. Zaman Jhanjhi, "Intelligent Framework Using IoT-Based WSNs for Wildfire Detection," in *IEEE Access*, vol. 9, pp. 48185-48196, 2021.

[4] E. Jang, Y. Kang, J. Im, D.-W. Lee, J. Yoon, and S.-K. Kim, "Detection and monitoring of forest fires using Himawari-8 geostationary satellite data in South Korea," *Remote Sensing*, vol. 11, no. 3, p. 271, 2019.

[5] R. Ghali, M. A. Akhloufi, and W. S. Mseddi, "Deep learning and transformer approaches for UAV-based wildfire detection and segmentation," *Sensors*, vol. 22, no. 5, p. 1977, 2022.

[6] N. Maeda and H. Tonooka, "Early stage forest fire detection from Himawari-8 AHI images using a modified MOD14 algorithm combined with machine learning," *Sensors*, vol. 23, no. 1, p. 210, 2023.

[7] S. Treneska and B. R. Stojkoska, "Wildfire detection from UAV collected images using transfer learning," in *Proc. 18th International Conference on Informatics and Information Technologies*, Skopje, North Macedonia, 2021.

[8] M. M. Fouda, et al., "A lightweight hierarchical AI model for UAV-enabled edge computing with forest-fire detection use-case," *IEEE Network*, vol. 36, no. 6, pp. 38–45, 2022.

[9] J. S. Almeida, et al., "Edgefiresmoke: A novel lightweight CNN model for real-time video fire–smoke detection," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 7889–7898, 2022.

[10] M. Varun, et al., "Integrating IoT and machine learning for enhanced forest fire detection and temperature monitoring," in *Proc. 2023 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2023.

[11] L. Zhang, et al., "MMFNet: Forest fire smoke detection using multiscale convergence coordinated pyramid network with mixed attention and fast-robust NMS," *IEEE Internet of Things Journal*, vol. 10, no. 20, pp. 18168–18180, 2023.

[12] H. Tao, "Smoke recognition in satellite imagery via an attention pyramid network with bidirectional multi-level multi-granularity feature aggregation and gated fusion," *IEEE Internet of Things Journal*, 2023.

[13] V. Kanakaraja and K. Pradeep, "An IoT based forest fire detection using Raspberry Pi," *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 8, no. 4, 2019.

[14] V. Dubey, P. Kumar, and N. Chauhan, "Forest fire detection system using IoT and artificial neural network," in *Proc. International Conference on Innovative Computing and Communications (ICICC)*, vol. 1, Springer Singapore, 2019.

[15] N. C. Gaitan and P. Hojbota, "Forest fire detection system using LoRa technology," *Int. J. Adv. Computer Sci. Appl.*, vol. 11, no. 5, 2020.

[16] A. Divya, T. Kavithanjali, and P. Dharshini, "IoT enabled forest fire detection and early warning system," in *Proc. 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, 2019.

[17] M. Krishnamoorthy, et al., "A design and development of the smart forest alert monitoring system using IoT," *Journal of Sensors*, vol. 2023, no. 1, p. 8063524, 2023.

[18] H. Singh, A. Shukla, and S. Kumar, "IoT based forest fire detection system in cloud paradigm," in *IOP Conference Series: Materials Science and Engineering*, vol. 1022, no. 1, IOP Publishing, 2021.

[19] J. Toledo-Castro, et al., "Management of forest fires using IoT devices," in *Proc. Eleventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, Barcelona, Spain, 2017.

[20] G. Pettorru, et al., "Using artificial intelligence and IoT solution for forest fire prevention," in *Proc. 2023 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, 2023.

APPENDIX

APPENDIX 1:

1. LIST OF PUBLICATIONS

PUBLICATION STATUS: ACCEPTED.

TITLE OF THE PAPER: LEVERAGING IOT AND MACHINE LEARNING FOR NEXT GENERATION FOREST FIRE DETECTION SYSTEMS.

AUTHORS: DEEPAK KUMAR K, SENTHIL PANDI S, KUMAR P, MANOJ M G AND MERCY N.

NAME OF THE CONFERENCE: INTERNATIONAL CONFERENCE ON ADVANCEMENT IN COMMUNICATION AND COMPUTING TECHNOLOGY (INOACC).

CONFERENCE DATE: 4TH APRIL, 2025

APPENDIX 2:

IMPLEMENTATION CODE

```
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
import matplotlib.pyplot as plt

class ForestDataset(Dataset):
    def __init__(self, data):
        self.data = torch.FloatTensor(
            data[['Normalized_Temperature',
'Normalized_Humidity', 'Normalized_Rs']].values
        )

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        return self.data[idx]

class SparseAutoencoder(nn.Module):
    def __init__(self, input_dim=3):
        super(SparseAutoencoder, self).__init__()

        # Encoder with dropout and batch normalization
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, 16),
            nn.BatchNorm1d(16),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(16, 8),
            nn.BatchNorm1d(8),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(8, 4),
            nn.BatchNorm1d(4),
            nn.ReLU()
        )

        # Decoder
        self.decoder = nn.Sequential(
            nn.Linear(4, 8),
            nn.ReLU(),
            nn.Linear(8, 16),
            nn.ReLU(),
            nn.Linear(16, input_dim),
            nn.Sigmoid()
        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded

def train_model(epochs=150, batch_size=32,
learning_rate=0.0005):
    # Load data
    df =
pd.read_csv("/Users/manoj/Documents/PYTHON/CODE_FY/sensor_data.
csv")
    dataset = ForestDataset(df)
    dataloader = DataLoader(dataset, batch_size=batch_size,
shuffle=True)

    # Initialize model
    model = SparseAutoencoder()
    criterion = nn.MSELoss()
    optimizer = torch.optim.Adam(model.parameters(),
lr=learning_rate, weight_decay=1e-5)

    # Sparsity parameters
    sparsity_target = 0.05
    sparsity_weight = 0.1

    # Training loop
    losses = []
    for epoch in range(epochs):
        model.train()
        epoch_loss = 0
        for batch in dataloader:
            # Forward pass
            output = model(batch)

            # Calculate reconstruction loss
            reconstruction_loss = criterion(output, batch)

            # Calculate sparsity loss
            encoded_output = model.encoder(batch)
            average_activation = torch.mean(encoded_output,
dim=0)
            sparsity_loss = torch.sum(
                sparsity_target * torch.log(sparsity_target /
(average_activation + 1e-8)) +
                (1 - sparsity_target) * torch.log((1 -
sparsity_target) / (1 - average_activation + 1e-8))
            )

            # Total loss
            loss = reconstruction_loss + sparsity_weight *
sparsity_loss
```

```
            # Backward pass and optimize
            optimizer.zero_grad()
            loss.backward()

            # Gradient clipping
            torch.nn.utils.clip_grad_norm_(model.parameters(),
max_norm=1.0)

            optimizer.step()

            epoch_loss += loss.item()

        losses.append(epoch_loss / len(dataloader))
        if (epoch + 1) % 10 == 0:
            print(f'Epoch [{epoch+1}/{epochs}], Loss: {losses[-
1]:.4f}')

    return model, losses

def detect_anomalies(model, data, threshold_multiplier=0.2):  #
Start with lower value
    model.eval()
    with torch.no_grad():
        input_data =
torch.FloatTensor(data[['Normalized_Temperature',
'Normalized_Humidity','Normalized_Rs']].values)

        reconstructed = model(input_data)

        # Multiple reconstruction error calculations
        reconstruction_error = torch.mean((input_data -
reconstructed) ** 2, dim=1)
        feature_wise_error = torch.mean((input_data -
reconstructed) ** 2, dim=0)
```

```
        # Statistical methods
        threshold = torch.mean(reconstruction_error) +
threshold_multiplier * torch.std(reconstruction_error)

        norm_temp = data['Normalized_Temperature'].values
        norm_rs = data['Normalized_Rs'].values
        norm_humidity = data['Normalized_Humidity'].values

        # Comprehensive anomaly detection
        anomalies = (
            (reconstruction_error > threshold).numpy() |
            (norm_rs > np.percentile(norm_rs, 90)) |  #
Adjusted percentile
            (norm_temp > np.percentile(norm_temp, 90)) |
            (norm_humidity < np.percentile(norm_humidity, 10))
        )

        return reconstruction_error.numpy(), anomalies

def plot_results(losses, reconstruction_errors, anomalies,
data):
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

    ax1.plot(losses)
    ax1.set_title('Training Loss Over Time')
    ax1.set_xlabel('Epoch')
    ax1.set_ylabel('Loss')

    ax2.plot(range(len(reconstruction_errors)),
reconstruction_errors, label='Reconstruction Error')

    # Only scatter plot anomalies
    anomaly_indices = np.where(anomalies)[0]
    ax2.scatter(anomaly_indices,
reconstruction_errors[anomalies],
                    color='red', label='Anomalies')

    ax2.set_title('Reconstruction Error and Detected
Anomalies')
    ax2.set_xlabel('Sample')
    ax2.set_ylabel('Reconstruction Error')
    ax2.legend()

    plt.tight_layout()
    plt.show()


def plot_model_metrics(losses, test_data, model):
    # Calculate accuracy based on reconstruction error
    model.eval()
    with torch.no_grad():
        input_data = torch.FloatTensor(
            test_data[['Normalized_Temperature',
'Normalized_Humidity', 'Normalized_Rs']].values
        )
        reconstructed = model(input_data)
        reconstruction_error = torch.mean((input_data -
reconstructed) ** 2, dim=1)

        # Consider a sample "accurate" if reconstruction error
is below mean + std
        threshold = torch.mean(reconstruction_error) +
torch.std(reconstruction_error)
        accuracies = (reconstruction_error <
threshold).float().mean().item()

        # Calculate accuracy for each epoch based on loss
values
        accuracy_scores = [1 - (loss / max(losses)) for loss in
losses]
```

```
    # Create figure and axis objects with a single subplot
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

    # Plot training loss
    ax1.plot(losses, 'b-', label='Training Loss')
    ax1.set_title('Training Loss Over Time')
    ax1.set_xlabel('Epoch')
    ax1.set_ylabel('Loss')
    ax1.grid(True)
    ax1.legend()

    # Plot accuracy
    ax2.plot(accuracy_scores, 'g-', label='Reconstruction
Accuracy')
    ax2.set_title('Model Accuracy Over Time')
    ax2.set_xlabel('Epoch')
    ax2.set_ylabel('Accuracy')
    ax2.grid(True)
    ax2.legend()

    # Add final accuracy annotation
    ax2.annotate(f'Final Accuracy: {accuracies:.2%}',
                xy=(len(accuracy_scores)-1, accuracy_scores[-
1]),
                xytext=(len(accuracy_scores)-20,
accuracy_scores[-1]+0.1),
                arrowprops=dict(facecolor='black',
shrink=0.05))

    plt.tight_layout()
    plt.show()

    return accuracies
```

```
# Example usage
if __name__ == "__main__":
    # Train the model
    model, losses = train_model()

    # Load some test data
    test_data =
pd.read_csv("/Users/manoj/Documents/PYTHON/CODE_FY/test.csv")

    # Detect anomalies
    reconstruction_errors, anomalies = detect_anomalies(model,
test_data)

    # Print detailed anomaly information
    print("\nAnomaly Detection Results:")
    for i, (is_anomaly, error) in enumerate(zip(anomalies,
reconstruction_errors)):
        if is_anomaly:
            print(f"Anomaly detected at row {i}:")
            print(f"  Temperature: {test_data.loc[i,
'Temperature']}°C")
            print(f"  Normalized Temperature: {test_data.loc[i,
'Normalized_Temperature']}")
            print(f"  Normalized Rs: {test_data.loc[i,
'Normalized_Rs']}")
            print(f"  Reconstruction Error: {error}")

    # Plot results
    plot_results(losses, reconstruction_errors, anomalies,
test_data)

    # Print summary
    print(f"\nTotal samples: {len(test_data)}")
    print(f"Anomalies detected: {np.sum(anomalies)}")
    final_accuracy = plot_model_metrics(losses, test_data,
model)
    print(f"\nFinal model accuracy: {final_accuracy:.2%}")
```