

**CSE4056 – Intelligent Multi Agent and Expert Systems**

**Project Report**

**AUTOMATIC EARLY FIRE DETECTION  
USING DEEP LEARNING AND SENDING ALARM  
ALERT AND SMS**

*By*

20BAI1061

Mandla Sheshi Kiran Reddy

B. Tech Computer Science and Engineering

*Submitted to*

**Dr. Suseela S**

**School of Computer Science and Engineering**



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

*November 2023*

## **ABSTRACT**

The safety of people and property is paramount, and reliable fire detection systems are critical in preventing loss of life and damage to property. Wireless fire detection systems have gained popularity due to their ease of use and convenience. One such system is the Wireless Fire Detection System Using Deep Learning & Python with Raspberry Pi Pico - GSM Call / SMS Alert.

This system uses a combination of Deep Learning and Python programming language to detect fires in real-time. The Raspberry Pi Pico serves as the microcontroller to control the entire system, and it is a wireless system that is easy to install and maintain.

The system includes a camera that captures real-time video footage, which is analysed by the Deep Learning algorithm. The algorithm has been trained to detect fire by analysing various features such as colour, texture, and motion of flames. Once the system detects a fire, it sends an alert message to the GSM module, which can either make a call or send a text message to a predetermined phone number. This feature ensures that people can be alerted promptly, even if they are not present on the premises.

The Wireless Fire Detection System Using Deep Learning & Python with Raspberry Pi Pico - GSM Call / SMS Alert is an efficient and reliable solution for detecting fires. By using advanced technologies such as Deep Learning, the system can detect fires quickly and accurately. The wireless design of the system makes it easy to install and maintain, which makes it suitable for both residential and commercial buildings. With this system in place, people can have peace of mind knowing that their property is protected from fires, and they will be alerted promptly in case of an emergency.

In conclusion, the Wireless Fire Detection System Using Deep Learning & Python with Raspberry Pi Pico - GSM Call / SMS Alert is a powerful system that leverages Deep Learning technology to provide fast and accurate fire detection. The system's wireless design, coupled with its ability to send alerts via GSM calls or SMS, ensures that people can take immediate action to prevent loss of life and property.

## **Introduction**

In recent years, technological advancements have revolutionized the way we approach fire safety in our homes and workplaces. Wireless fire detection systems have become increasingly popular due to their ability to detect fires quickly and efficiently without the need for complex wiring or infrastructure.

One of the latest innovations in wireless fire detection systems is the use of deep learning and Python with Raspberry Pi Pico to create an intelligent and efficient system that can detect fires and alert authorities using GSM call or SMS. The system utilizes machine learning algorithms to analyse data from various sensors, including temperature sensors, smoke detectors, and motion sensors.

The Raspberry Pi Pico is a powerful microcontroller board that is compatible with Python programming language, making it an ideal platform for developing a wireless fire detection system. Its compact size and low power consumption make it ideal for use in small spaces, making it an ideal choice for both residential and commercial applications.

Deep learning algorithms allow the system to analyse data from sensors in real-time, making it possible to detect fires quickly and accurately. The system can also learn from past data, allowing it to improve its accuracy over time.

The GSM call and SMS alert feature is a critical component of the system, ensuring that authorities are alerted to fires quickly, allowing them to respond promptly and minimize damage. This feature is particularly useful in cases where the building is unoccupied, such as during weekends or holidays, ensuring that fires are detected and reported regardless of whether anyone is present.

In summary, the wireless fire detection system using deep learning and Python with Raspberry Pi Pico is an innovative and intelligent solution for fire safety. Its ability to detect fires quickly and accurately, and its GSM call and SMS alert features make it an essential tool for protecting homes and businesses from the devastating effects of fires.

## **PROPOSED WORK**

### **3.1 Objective and goal of the project**

The core objective of our project is to build a wireless fire detection system using deep learning and python with Raspberry Pi Pico as an intelligent solution for fire safety. The outcome of our project must have GSM call and SMS alert features along with the ability to detect fire quickly and accurately which can help in protecting homes and safeguarding businesses against the destructive impact of fire.

### **3.2 Problem Statement**

Our project "AUTOMATIC EARLY FIRE DETECTION USING DEEP LEARNING AND SENDING ALARM ALERT AND SMS" is all about designing a fire detection system which comes with additional features like sending an alarm and SMS to the fire stations nearby along with some given numbers like police stations, ambulance etc... Fire detection is done using a booming computer science technology i.e., Deep Learning. We first developed a deep learning and then incorporated some hardware elements like GSM to send alarms and alerts. This project can be used in forests to detect forest fires, in old age homes for fast response, near transformers and some sensitive official places like data servers. Fire in these places must be recognized soon and action must be taken. The cameras we setup keep monitoring the locations and if there is a trace of fire, alarms will be on to help nearby people to move out to safer place and SMS will be sent to centres like fire stations, police stations, hospitals etc... for further action.

### **3.3 Motivation**

There are several reasons to use deep learning in automated fire alarm systems:

- **Early detection of fires:** Fires can spread quickly and cause devastating property damage and serious risk to life safety. Automated fire detection using deep learning algorithms can provide early warnings that enable rapid response and implementation of mitigation measures, potentially saving lives and minimizing property damage.
- **Real-time monitoring:** Deep learning-based fire detection systems can continuously monitor large areas such as forests, industrial facilities, or buildings in real time without human intervention. This enables rapid detection of fires, even in remote or inaccessible locations, as they develop and enables immediate response and intervention.
- **High Accuracy:** Deep learning algorithms can achieve high accuracy in fire detection by analysing large amounts of visual and/or thermal data that human operators may find difficult to process. The ability to detect fires accurately and quickly can significantly reduce false alarms and ensure that firefighting resources are efficiently allocated and disruptions to normal operations are minimized.

- **Cost effective:** Automated fire detection using deep learning can potentially reduce costs associated with manual fire monitoring and inspection. It can also help prevent costly damage that can result from late detection of fire, such as: B. Loss of property, equipment, and loss of production.
- **Scalability:** Deep learning-based fire detection systems can be easily adapted to different environments, such as: B. Indoor, outdoor, and even large-scale forest fire detection. This makes them versatile and adaptable to various applications, from industrial facilities and transportation hubs to residential and commercial buildings.
- **Safety:** The use of deep learning-based fire detection systems can reduce the need for human firefighters to enter hazardous or hazardous environments for fire detection and minimize their exposure to risk such as toxic smoke, extreme heat, and structural collapse.
- **Integration with other technologies:** Deep learning-based fire detection systems can be integrated with other technologies such as drones, IoT sensors and emergency response systems to provide a solution comprehensive fire detection and management. This can improve situational awareness, reduce response times, and enable coordinated firefighting efforts.

### 3.4 Challenges

Automated fire detection using deep learning models has received significant attention in recent years due to its potential for improving fire safety and prevention. However, several challenges need to be addressed to develop accurate and reliable automated fire detection systems using deep learning models. Some of the main challenges are:

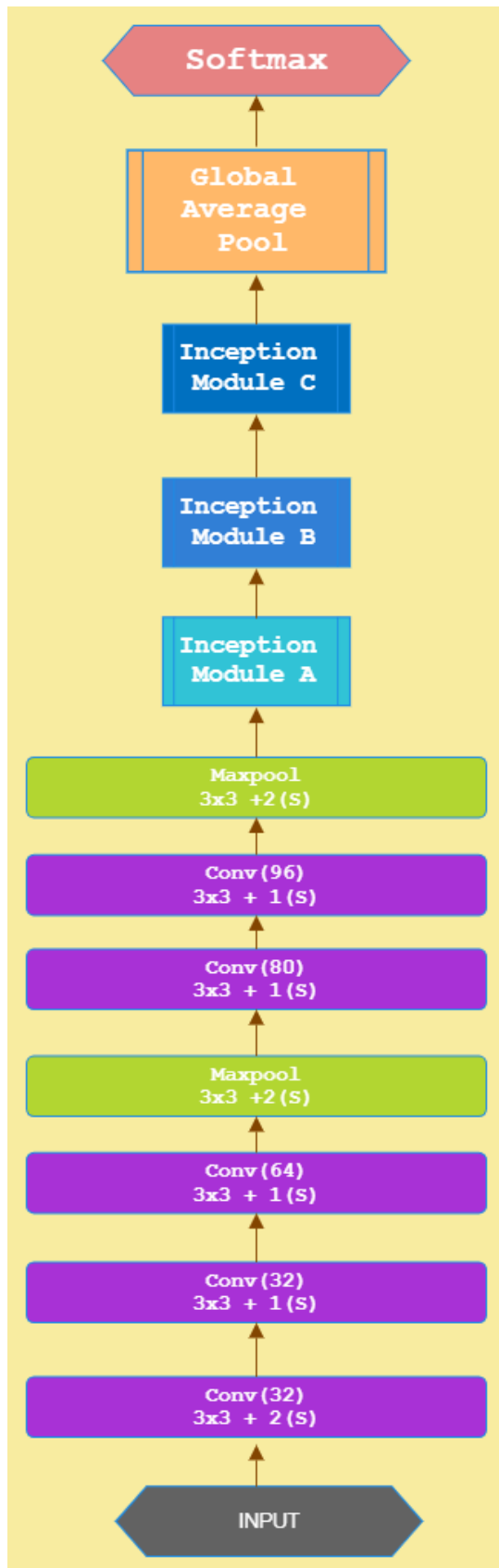
- **Data scarcity:** Deep learning models require large amounts of labelled data for training. However, due to the rarity of fires and associated safety risks, it can be difficult to obtain a large set of fire related image or video data for training purposes. This can lead to limited data availability, which can lead to overfitting or poor generalization performance of the trained model. Variability in fire types and conditions: Fires can occur in a variety of forms, such as forest fires, building fires, and industrial fires, and can exhibit different behaviours based on factors such as fuel type, flame size, and the environmental conditions. This variability makes it difficult to develop a single deep learning model that can accurately detect fires in different types of fires and under different conditions. This may require robust model architectures and several training data.
- **Class Imbalance:** In fire detection datasets, the number of non-fire related images or videos usually significantly exceeds the number of fire-related samples, resulting in a class imbalance. This can affect the model's ability to accurately detect fires, as the model may lean towards the majority (non-fire) class during training, resulting in reduced sensitivity and false negatives in detecting fires.

- **Real-time processing:** Fire detection systems often need to work in real-time to respond and act in a timely manner. However, deep learning models can be computationally intensive and achieving real-time processing can be difficult, especially in resource-limited environments such as embedded systems or Internet of Things devices. (IoT). Balancing accuracy and real-time processing requirements are a major challenge in developing practical automated fire detection systems.
- **Environmental Factors:** Environmental factors such as lighting, smoke, and occlusion can significantly affect the accuracy of fire detection systems. Smoke, in particular, can obscure the visibility of flames, making it difficult for deep learning models to accurately detect fires in smoky environments. Developing robust models capable of handling these environmental factors and maintaining high detection accuracy is a challenge.
- **Safety Issues:** Fire detection systems must operate in safety-critical environments, and false positives or false negatives in fire detection can have serious consequences. Ensuring the reliability, safety, and interpretability of deep learning fire detection models is a key challenge that must be addressed to gain trust and acceptance in practical applications.
- **Generalize across environments:** Deep learning models trained on specific datasets may have limited generalization capabilities when applied to different settings or domains. For example, a model trained on indoor fires may not perform well when applied to outdoor wildfire scenarios. Ensuring the generalizability of fire detection deep learning models across different environments and scenarios is a challenge that requires careful consideration of the training data and model architecture.

### 3.5 APPLICATIONS

- 1.Forest fires detection.
- 2.Public fires detection near highways.
- 3.Public fires detection near railway tracks.
- 4.Early fire detection near power stations or transformers in public places.
- 5.Early fire detection in vehicles with installed cameras /sensors.
- 6.Early fire detection in our homes or small organizations with installed cameras.

## ARCHITECTURE AND BLOCK DIAGRAM



The Block Diagram Explanation:

### **1. Initial Stage:**

The entry point for data into the network, though the specific details are not explicitly outlined in the provided snippet.

### **2. Convolutional Operation (3x3 kernel, 2 strides, 5 filters):**

Executes a convolutional process using a 3x3 kernel, a stride of 2, and employs 5 filters. This step is succeeded by a subsequent Max Pooling layer utilizing a 3x3 kernel and 2 strides.

### **3. Inception Module A:**

An integral part of the architecture, Inception modules serve the purpose of capturing features across various scales. Module A typically encompasses a blend of 1x1, 3x3, and 5x5 convolutions.

### **4. Inception Module B:**

Resembling Inception Module A, this segment likely employs a distinct combination of convolutions to capture diverse features.

### **5. Inception Module C:**

Another instance of an Inception module, potentially employing a different convolutional mix.

### **6. Global Average Pooling:**

Rather than relying on fully connected layers, which can lead to an abundance of parameters, Global Average Pooling is employed. It condenses each feature map to a singular value by computing the average of all values within the feature map. This aids in reducing both dimensionality and the overall number of parameters in the network.

### **7. SoftMax Layer:**

Following Global Average Pooling, a SoftMax layer is typically implemented. This layer is commonly utilized in classification tasks, converting the final layer's outputs into probability scores.

The numerical values in parentheses, such as (96) or (80), generally indicate the quantity of filters in the respective convolutional layer.

Inception Net excels in its proficiency to efficiently capture information at multiple scales, courtesy of the Inception modules. Its computational efficiency is further highlighted by the strategic use of 1x1 convolutions and the incorporation of Global Average Pooling.



## EXISTING SYSTEM FINDING

### 5.1 Literature Survey

[1] ResNet50 is a reliable model for early-time monitoring of forest fires, based on VGG16, ResNet50, and DenseNet121. It classifies images into two classes, fire, and no fire, and uses high train and test accuracy. Fine-tuning helps the model avoid overfitting and the model can be applied in real-time to avoid forest fires.

[2] The proposed system for forest fire detection using wireless sensor networks and machine learning was found to be an effective method with accurate results. It is able to be mounted at any place in the forest and is easily implementable as a standalone system for prolonged periods. It also alerted authorities with lower latency than existing systems during test trials.

[3] This paper presents a smart fire detection system that will notify authorities in the early stages of their early stages and even before the fire breakout. It uses a signal-processing unit, an image processing unit, and a GSM module unit. An integrated sensor system is proposed instead of using a single detector. A machine learning approach is adapted and compared with the output to get more accurate detection. The proposed system encompasses the entire safety and security of the property as well as reduces the expense and time while designing and securing modern properties in respect of fire danger. The adapted soft computing approach can open a scope for promoting further research work.

[4] The developed prototype is designed to be used by a user to control the fire alarm system remotely. It is more appropriate than the use of only one of the modules and its portability can be improved by an efficient assimilation of the different modules. It can be applied in smart cities due to its flexibility and simplicity in handling.

[5] This proposed a fire detection approach based on computer vision methods and deep learning technology. It was successfully deployed based on C++ programming language and videos from surveillance cameras were processed frame by frame through a background subtraction method to extract moving region images. The VSD algorithm was used to extract significant regions from the area containing moving objects. To reduce the fire false alarm, a motion detection method based on background superagency was added. The suggested approach was designed for practical use in forest environments and other situations with elevated risk of wildfires.

[6] This paper discusses a new low-cost drone equipped with image processing and object detection abilities for smoke and fire recognition tasks in forests. The ssdlite nanobilenet model, a sub-model of the supervised deep learning model called Model Zoo, is used as an example. The proposed system can be improved by increasing flight time and using convertible energy sources such as solar energy.

It can also be trained to detect flame and fire visuals, and a downward fire/flame scan can be performed on the vehicle suspended above the smoke detection zone. The study aims to shed light on similar scientific studies.

[7] This work presents on Recurrent Trend Predictive Neural Network (rTPNN) for multi sensor fire detection. It is based on trend prediction, level prediction, and fusion of sensor readings. It outperforms other models with 96% accuracy and achieves high True Positive and True Negative rates at the same time. It also triggers an alarm in 11s from the start of the fire, making it acceptable for real-time applications. They used several models like LSTM, SVM, MLP and compared with rTPNN. The rTPNN model outperforms all of the machine learning models in terms of the prediction performance with high generalization ability. This paper interprets that the rTPNN model will have an impact on the various time series problems.

[8] Fire Net is a lightweight neural network built from scratch and trained on a diverse dataset to develop an internet of things (IoT) capable fire detection unit that can replace physical sensor-based detectors and reduce false and delayed triggering. The introduced neural network can smoothly run on a low-cost embedded device like Raspberry Pi 3B at a frame rate of 24 frames per second. The performance obtained by the model on a standard fire dataset and a self-made test dataset, the dataset consists challenging real-world fire and non-fire images with image quality that is like the images captured by the camera attached to Raspberry Pi. The performance is in terms of accuracy, precision, recall, and F-measure is encouraging.

[9] This paper mainly focuses on alerting the surrounding people and also inform the firemen regarding the incident. The proposed system encompasses a fire alarm, along with the capability to send notifications to our mobile devices. Simultaneously, emails containing information about the accident-prone area and details necessary for alerting the fire station can be dispatched to the provided email address. This implemented fire detection system will be more efficient and provide more safety when compared to the conventional fire alarm systems available earlier.

[10] In this paper, the main purpose is to build fire detection system that is able to detect the conditions of the environment and the results are informed quickly on the incident area and fire department. This study uses Arduino Uno, GPS Ublox Neo 6MV2, SIM900A and three sensors to build an early fire detection system with an average GPS error of 1.6%, an accuracy of reading shifts, and an average distance of  $\pm 4$  meters. The average data transmission speed among providers is 1 second, as it aligns with the processing time, sending speed, and capabilities of the GSM module in accordance with network conditions.

## 5.2 FINDINGS

### **Current System Overview:**

The prevailing state of fire detection systems often relies on traditional methodologies or basic technologies, lacking the precision and efficiency demanded by contemporary safety standards. Typical systems may employ basic smoke detectors or sensor networks, but their ability to deliver timely and accurate alerts is constrained. Moreover, these systems may lack integration with advanced communication mechanisms, impeding the swift response necessary to mitigate the impact of fires.

### **Innovative System Featuring Inception Net Model:**

In stark contrast, our proposed system brings about a paradigm shift in fire detection by integrating cutting-edge deep learning technology, specifically utilizing the formidable Inception Net model. Unlike conventional systems, our solution taps into the capabilities of Inception Net, a sophisticated convolutional neural network renowned for its excellence in image classification tasks.

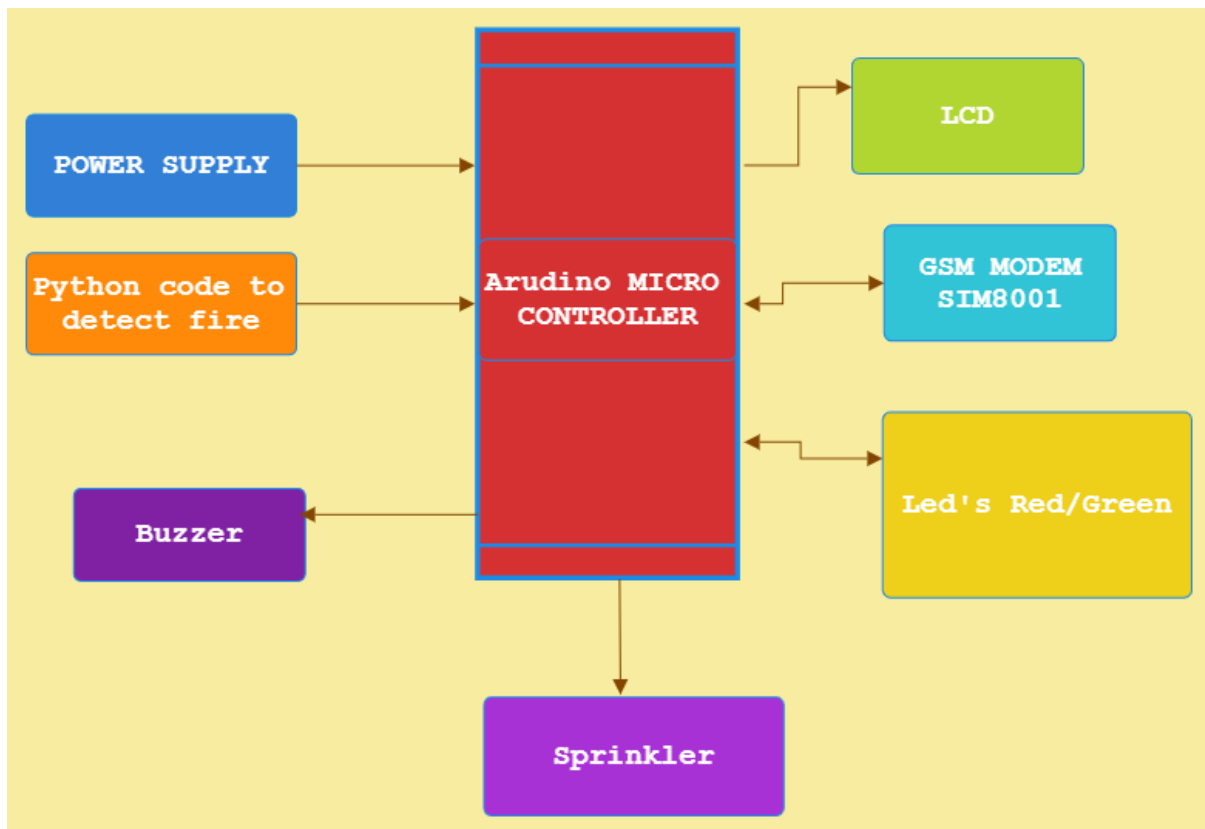
### **Key Points of Distinction:**

1. **Augmented Precision:** The Inception Net model excels at capturing intricate features, ensuring heightened accuracy in fire detection compared to traditional methodologies.
2. **Multifaceted Feature Extraction:** Inception Net's distinctive architecture enables the system to analyse images at multiple scales simultaneously, enhancing its ability to detect fires across diverse scenarios.
3. **Optimized Resource Utilization:** Through its judicious use of 1x1 convolutions, Inception Net ensures computational efficiency without compromising performance, rendering it suitable for real-time applications.
4. **Incorporated GSM Alerts:** Beyond robust fire detection capabilities, our system integrates GSM technology for swift and widespread alerts, promptly notifying relevant authorities and stakeholders.
5. **Diverse Applications:** The proposed system transcends traditional fire detection by extending its applicability to various settings, including forests, public areas near highways and railways, power stations, vehicles, homes, and small organizations.

In summary, our innovative approach leverages the prowess of deep learning, particularly Inception Net, to redefine fire detection systems. The integration of advanced technology, coupled with real-time alert mechanisms, establishes our system as a comprehensive and efficient solution for safeguarding lives and properties from the ravages of fire.

## IMPLEMENTATION

### 6.1 HARDWARE IMPLEMENTATION



#### Step 1: Connect the Arduino board.

- Connect the Arduino board to your laptop using a USB cable.
- Make sure you have the Arduino IDE or a compatible IDE installed on your laptop.

#### Step 2: Connect the GSM module.

- Establish a connection between the Arduino board and the GSM module using the designated pins.
- Connect the Rx pin of the GSM module to a digital pin on the Arduino board (e.g. pin 2).
- Connect the Tx pin of the GSM module to a digital pin on the Arduino board (e.g. pin 3).
- Connect the GND pin of the GSM module to the GND pin of the Arduino board.
- Connect the GSM module's VCC pin to a 5V pin on the Arduino board.

#### Step 3: Connect the LCD module

- Connect the LCD module to the Arduino board using the appropriate pins.

- Connect the RS pin of the LCD module to a digital pin on the Arduino board (eg pin 4). Connect the LCD module's EN pin to a digital pin on the Arduino board (eg pin 5).
- Connect the D4, D5, D6 and D7 pins of the LCD module to the digital pins of the Arduino board (eg pins 6, 7, 8, 9).
- Connect the GND pin of the LCD module to the GND pin of the Arduino board.
- Connect the VCC pin of the LCD module to a 5V pin on the Arduino board.

#### **Step 4: Connect the LEDs**

- Connect the Fire Alarm LED to a digital pin on the Arduino board (eg pin 10).
- Connect the system status LED to a digital pin on the Arduino board (eg pin 11).
- Connect the positive lead of both LEDs to a 220 ohm resistor and connect the other end of the resistor to a 5V pin on the Arduino board.
- Connect the negative pole of the two LEDs directly to the GND pin of the Arduino board.

#### **Step 5: Connect the fire alarm**

- Connect the fire alarm or fire extinguisher to a digital pin on the Arduino board (eg pin 12).
- Connect the other end of the fire alarm to a separate power supply, eg. a battery or a relay, depending on the needs of your specific watering mechanism.

#### **Step 6: Turn on the Arduino board**

- Power the Arduino board with a 9V battery connected to the power socket or via the USB cable connected to your laptop.

#### **Step 7: Set up and customize the necessary software.**

- Install OpenCV and Python on your laptop if they are not already installed.
- Upload the supplied Arduino code to interface with the GSM module, LCD module, LEDs and fire alarm onto the Arduino board using the Arduino IDE or any other compatible IDE.
- Upload the Arduino fire detection code using OpenCV to the Arduino board.

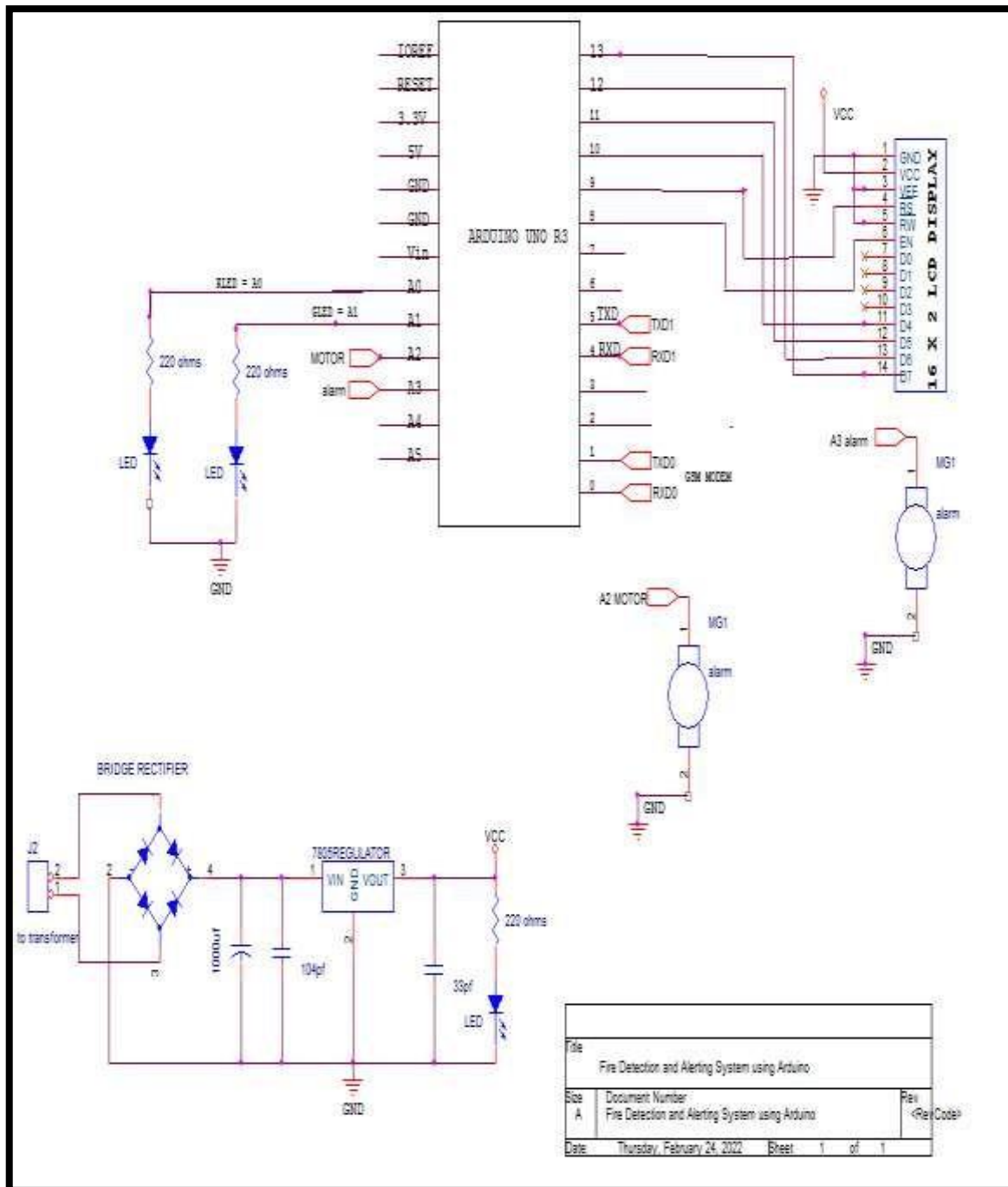
#### **Step 8: Implement the fire detection system**

- Connect a camera to the laptop and make sure it is recognized by OpenCV.
- We have to develop the python script with required Deep Learning Algorithms that captures video frames from the camera along with OpenCV.
- Process video frames using OpenCV to detect fire using appropriate image processing techniques (for example, color-based fire detection).
- If a fire is detected, send a signal to the Arduino board via a USB cable to activate the fire alarm LED and send an SMS notification using the GSM module.
- Displays system status and fire alarms on the LCD module.

- Based on the output, activate the alarm or fire mechanism to extinguish the fire using the Arduino board.

### Step 9: Test and troubleshoot

- Run the Python script and test the fire detection system by exposing it to fire or similar conditions.
- Verify that the Fire Alarm LED activates, an SMS notification is sent and the LCD module displays the correct messages.
- Verify that the alarm or fire suppression mechanism is activated correctly.



## 6.2 Software stimulation:

The python script which we develop using the concepts of the data augmentation, deep learning algorithms such as the Resnet, InceptionV3, YOLO, MobilenetV2 and also, we use the hyper parameter tuning such as the Adam, Adagrad & the RMSProp etc.

### DATA AUGMENTATION:

Data augmentation is a common technique used in deep learning to artificially increase the size of the training dataset by creating new training examples from the original data through various transformations. In automated fire detection using deep learning, data augmentation can be particularly useful for improving model performance and robustness. Here are some data augmentation techniques which are used are applied specifically for fire detection:

- **Image Rotation:** By rotating the images by different degrees, the model can learn to see the fires from different orientations. Fires can appear from different angles and orientations in real-life scenarios, and by rotating the images during training, the model can learn to see fires from different perspectives.
- **Image Flip:** Flipping images horizontally or vertically can create additional training examples and help the model learn to detect fires in mirror or inverted orientation. This can be useful when working with images captured from different sources or cameras that may have different orientations.
- **Image Scaling:** By enlarging or reducing the images, the model can learn to recognize fires of different sizes. Fires can vary in size, and by applying scale transformations, the model can learn to recognize fires at different scales, from small flames to large infernos.
- **Image Translation:** Horizontal or vertical translation of images can be used to simulate changes in camera viewpoints or camera angles. This can help the model learn to spot fires from different perspectives and positions in the image.
- **Image Brightness and Contrast Adjustment:** Adjusting image brightness and contrast can simulate changes in lighting conditions, such as different times of day or weather conditions. This can help the model learn to detect fires under varying lighting conditions and make it more resilient to changes in lighting.
- **Adding Random Noise:** Adding random noise to images can simulate image distortions or sensor noise, which can be common in realistic fire detection scenarios. This can help the model become more robust against noise or distorted images.

- **Data fusion:** Combining images from different sources or modalities, such as infrared and visible images, can provide additional information and improve the model's ability to accurately detect fires. This can be done by overlaying the images, stitching them together or using them as separate channels in a multi-channel input network.

### **DATASET COLLECTION:**

We collected the dataset using the data augmentation of nearly 3000 images with the train data having 2700 images and the test data with 300 images. The dataset is totally divided into 3 classes that is fire, smoke and neutral as it is also important for us to take the consideration of the distinction between the smoke and the fire and consideration of non-fire scenarios.

### **MODEL:**

A key component of fire safety is early fire detection. Fires can be disastrous, resulting in the loss of life, destruction of property, and harm to the environment. Early fire detection can help avoid these disastrous effects by enabling timely action to put out the fire and reduce damage. Deep learning models like Inception Net v3 have demonstrated particularly promising outcomes in early fire detection using artificial intelligence (AI).

A deep convolutional neural network called Inception Net v3 classifies images by combining convolutional layers, pooling layers, and fully connected layers. It has been trained on a sizable dataset, and it has attained great accuracy rates, making it a popular choice for picture classification tasks.

Because of its ability to identify photographs of fires swiftly and reliably, it is perfect for early fire detection.

Installing cameras in strategic positions and putting the video feeds into the neural network are common procedures for early fire detection systems that use Inception Net v3. The neural network then does a real-time analysis of the images to look for fire-related indicators. When a fire is discovered, the neural network can alert a central monitoring system, which can then notify the emergency services.

The capability of Inception Net v3 to detect fires in a range of conditions is one advantage of employing it for early fire detection.

It can find flames in both indoor and outdoor settings, as well as in various weather situations. It can also find fires of different sizes, from tiny fires to massive firestorm. It is therefore a flexible instrument for early fire detection in a variety of circumstances.

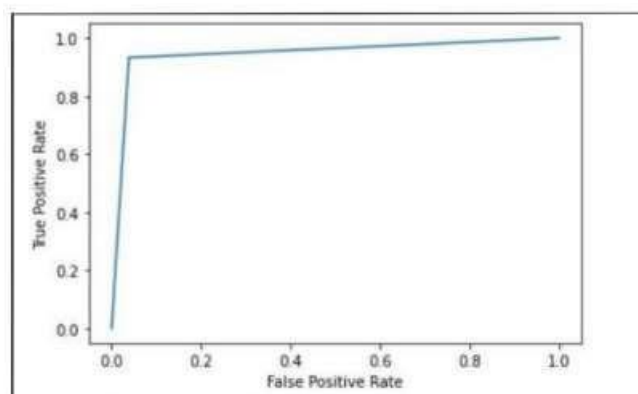
For fire safety, early fire detection is essential, and artificial intelligence has shown promise in this regard. Due to its excellent accuracy rates and adaptability, Inception Net v3 is a particularly effective deep learning model for early fire detection. Early fire detection systems are going to become even more important as AI technology develops.



## RESULTS AND DISCUSSION

The highest accuracy we got is nearly 0.95 for the Inceptionnet-V3 model with SGD optimizer and the ROC CURVE is as follows:

	precision	recall	f1-score	support
0	0.92	0.96	0.94	518
1	0.97	0.93	0.95	656
micro avg	0.95	0.95	0.95	1174
macro avg	0.94	0.95	0.94	1174
weighted avg	0.95	0.95	0.95	1174
samples avg	0.95	0.95	0.95	1174



The reaction time is very fast compared to other algorithms and the reaction time is nearly 1.7 seconds. The use of Inception-netV3 is very less based on our survey and the accuracy and the reaction time is pretty good and the OpenCV is helping us to capture the frames properly.

## CONCLUSION AND FUTURE WORK

In summary, deep learning has shown great potential for fire detection due to its ability to automatically learn complex patterns from large amounts of data. Using deep learning models and Open-Cv demonstrated high accuracy in detecting fires in various environments, including indoor and outdoor environments, and through different imaging and video modalities.

However, there are still challenges and future directions for deep learning fire detection that can be explored further. Some potential areas for future work include:

- **Real-time fire detection:** Real-time fire detection is essential for rapid response and effective firefighting. Further research could focus on developing real-time deep learning models that provide instantaneous fire detection capabilities, enabling faster response and mitigation of fire incidents.
- **Robustness to different environmental conditions:** Deep learning fire detection models can face challenges when dealing with different environmental conditions such as lighting changes, weather conditions, and smoke. Future research could explore techniques to improve the robustness of deep learning models to different environmental conditions and make them more reliable in real-world scenarios.
- **Multimodal fire detection:** Deep learning models can benefit from integrating multiple modalities such as image, video, audio, and sensor data for fire detection. Future work could explore multimodal deep learning approaches that can leverage different data sources to improve the accuracy and robustness of fire detection systems.
- **Edge computing for fire detection:** Edge computing, where processing occurs locally on devices at the edge of the network, can significantly reduce latency and bandwidth requirements. Future research could focus on developing effective deep learning models that can be deployed on edge devices such as drones, surveillance cameras, and IoT devices to detect fires in real-time in remote or remote environments. limited resources.
- **Large-scale fire detection:** Deep learning models can be trained on large datasets to further improve their performance. Future research could focus on the collection and annotation of large-scale fire datasets, which can help train more accurate and robust deep learning fire detection models.
- **Explainable AI for fire detection:** Deep learning models are often regarded as black boxes because they are not interpretable. Future research could focus on developing explainable AI techniques that can inform the decision-making process of deep learning fire detection models, making them more transparent and verifiable.
- We have to also overcome the false positives here which is major concern.

## 8.2 USE OF GANS

**Innovative System Featuring Inception Net Model and GANs:** In a pioneering departure, our envisioned system integrates cutting-edge deep learning technology, utilizing the formidable Inception Net model in tandem with GANs. The integration of GANs enhances accuracy by generating synthetic yet realistic fire-related images, complementing the learning process of the Inception Net model.

### Key Distinctive Features:

- **Enhanced Precision:** The combination of the Inception Net model and GANs excels in capturing intricate features, ensuring heightened accuracy in fire detection compared to traditional methodologies.
- **Multifaceted Feature Extraction:** The distinctive architecture of Inception Net, augmented by GANs, enables the system to analyse images at multiple scales simultaneously, thereby further improving its ability to detect fires across diverse scenarios.
- **Optimized Resource Utilization:** Through judicious use of 1x1 convolutions and the synergy with GANs, Inception Net ensures computational efficiency without compromising performance, making it well-suited for real-time applications.
- **Incorporated GSM Alerts:** Beyond robust fire detection capabilities, our system seamlessly integrates GSM technology for swift and widespread alerts, promptly notifying relevant authorities and stakeholders.
- **Versatile Applications:** The proposed system transcends traditional fire detection, extending its applicability to various settings, including forests, public areas near highways and railways, power stations, vehicles, homes, and small organizations.

### Dataset Utilization for Conclusion and Future Work:

In concluding this phase of the project, a meticulously curated dataset comprising a diverse range of fire-related images played a pivotal role in training and validating the accuracy of our deep learning model. The integration of GANs further enhanced the dataset's effectiveness by introducing synthetic images, enriching the model's understanding of nuanced fire scenarios.

In summary, while deep learning has shown promise for fire detection, there are several areas that can be further explored to improve the accuracy, real-time capabilities, robustness, and interpretability of fire detection models. of deep learning. Continuing research and development in these areas can contribute to the advancement of fire detection technology leading to more effective strategies for fire prevention, early detection and mitigation.

## REFERENCES

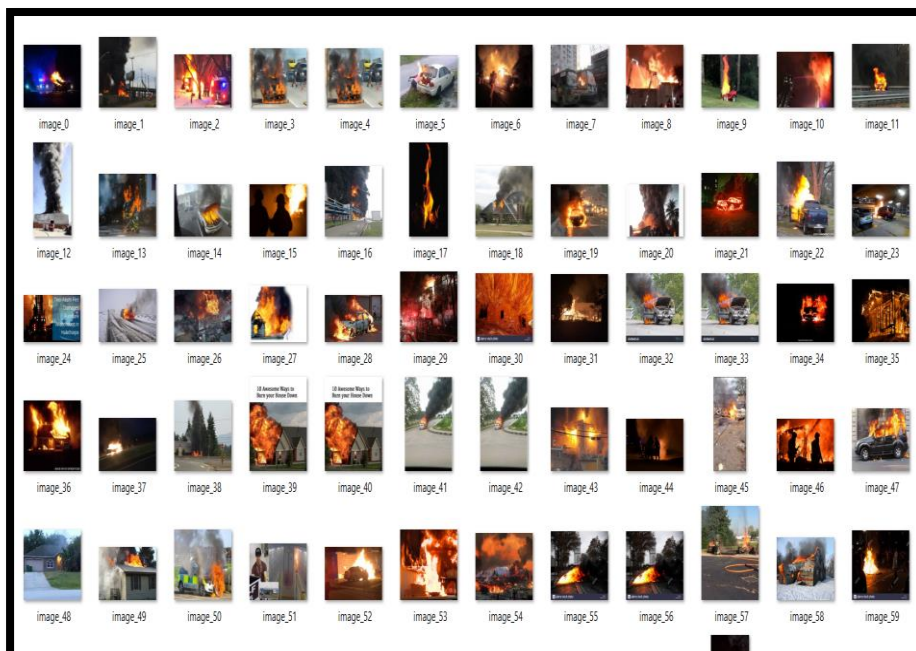
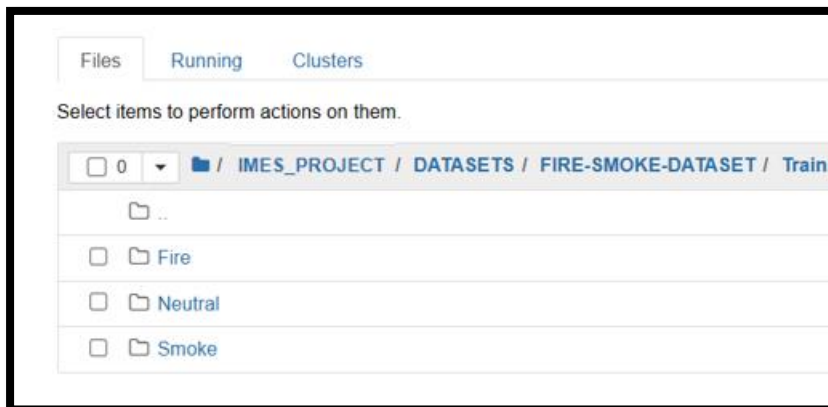
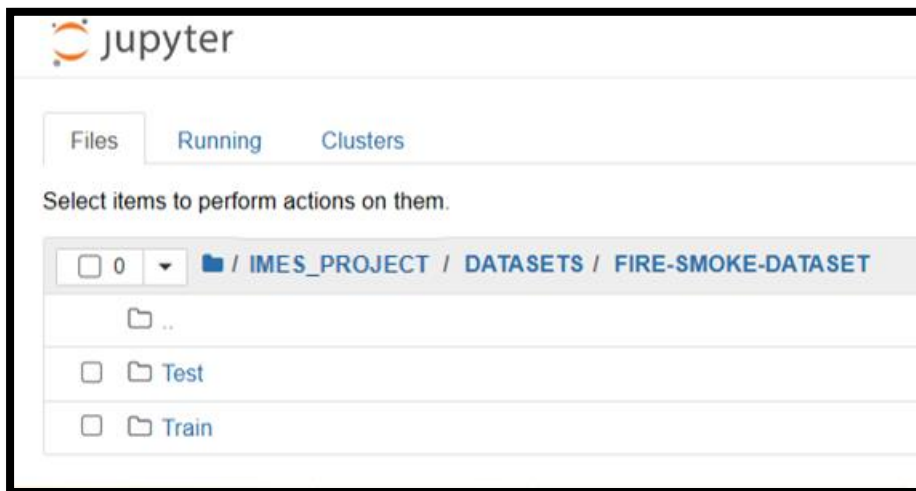
### DATASET –


<https://www.kaggle.com/datasets/dataclusterlabs/fire-and-smoke-dataset>

### LITERATURE SURVEY:

- [1] Rahul, Medi & Saketh, Karnekanti & Sanjeet, Attili & Naik, Nenavath. (2020). Early Detection of Forest Fire using Deep Learning. 1136-1140. 10.1109/TENCON50793.2020.9293722.
- [2] Dampage, U., Bandaranayake, L., Wanasinghe, R. et al. Forest fire detection system using wireless sensor networks and machine learning. Sci Rep 12, 46 (2022). <https://doi.org/10.1038/s41598-021-03882-9>.
- [3] Sultan Mahmud, M., Islam, Md. S., & Rahman, Md. A. (2017). Smart Fire Detection System with Early Notifications Using Machine Learning. In International Journal of Computational Intelligence and Applications (Vol. 16, Issue 02, p. 1750009). World Scientific Pub Co Pte Lt. <https://doi.org/10.1142/s1469026817500092>.
- [4] Yadav, Rishika and Rani, Poonam, Sensor Based Smart Fire Detection and Fire Alarm System (November 3, 2020). Proceedings of the International Conference on Advances in Chemical Engineering (AdChE) 2020, Available at SSRN: <https://ssrn.com/abstract=3724291> or <http://dx.doi.org/10.2139/ssrn.3724291>.
- [5] Yingshu Peng & Yi Wang (2022) Automatic wildfire monitoring system based on deep learning, European Journal of Remote Sensing, 55:1, 551-567, DOI: 10.1080/22797254.2022.2133745.
- [6] Yanık, Ayşegül & Guzel, Mehmet & Yanık, Mertkan & Bostanci, Gazi Erkan. (2021). Machine Learning Based Early Fire Detection System using a Low-Cost Drone.
- [7] M. Nakip, C. Güzelış and O. Yildiz, "Recurrent Trend Predictive Neural Network for Multi-Sensor Fire Detection," in IEEE Access, vol. 9, pp. 84204-84216, 2021, doi: 10.1109/ACCESS.2021.3087736.
- [8] Arpit Jadon, Mohd. Omama, Akshay Varshney, Mohammad Samar Ansari, Rishabh Sharma, “ FireNet: A Specialized Lightweight Fire & Smoke Detection Model for Real-Time IoT Applications” in arXiv:1905.11922, <https://doi.org/10.48550/arXiv.1905.11922>.
- [9] Ala, Rohika & Sai, M Vara Raghava & Vamsi, Jollu & Raju, Chakravarthula. (2020). Automatic Fire Detection and Alert System. Xi'an Jianshu Keji Daxue Xuebao/Journal of Xi'an University of Architecture & Technology. Volume XII. 2109. 10.37896/JXAT12.05/1616.
- [10] Rachman, F. Z., Yanti, N., Jamal, N., Purwanto, E., Hadiyanto, H., Suhaedi, S., & Sorongan, E. (2020). Design of an early fire detection system based on GPS module. In IOP Conference Series: Materials Science and Engineering (Vol. 732, Issue 1, p. 012056). IOP Publishing. <https://doi.org/10.1088/1757-899x/732/1/012056>.

## IMPLEMENTATION IMAGES:








 jupyter

Files Running Clusters

Select items to perform actions on them.

0 / IMES\_PROJECT

- ..
- ☐  imes\_inceptionnet.ipynb
- ☐  app\_imes.py
- ☐  FIRE-SMOKE-DATASET.zip
- ☐  flowchart.jpg
- ☐  InceptionV3.h5

```
In [5]: from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Input, Dropout
input_tensor = Input(shape=(224, 224, 3))
base_model = InceptionV3(input_tensor=input_tensor, weights='imagenet', include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(2048, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)
predictions = Dense(3, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
for layer in base_model.layers:
    layer.trainable = False
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['acc'])
history = model.fit(train_generator, steps_per_epoch = 14, epochs = 20, validation_data = validation_generator, valac
```

```
Epoch 1/20
14/14 [=====] - 172s 11s/step - loss: 9.1838 - acc: 0.5555 - val_loss: 0.7642 - val_acc: 0.7857
Epoch 2/20
14/14 [=====] - 368s 27s/step - loss: 0.5086 - acc: 0.8052 - val_loss: 0.7949 - val_acc: 0.6020
Epoch 3/20
14/14 [=====] - 322s 22s/step - loss: 0.5569 - acc: 0.7868 - val_loss: 0.4981 - val_acc: 0.8265
Epoch 4/20
14/14 [=====] - 378s 28s/step - loss: 0.4808 - acc: 0.8175 - val_loss: 0.3065 - val_acc: 0.8724
Epoch 5/20
14/14 [=====] - 147s 10s/step - loss: 0.4645 - acc: 0.8305 - val_loss: 0.3128 - val_acc: 0.8980
Epoch 6/20
14/14 [=====] - 154s 11s/step - loss: 0.4330 - acc: 0.8465 - val_loss: 0.2852 - val_acc: 0.8929
Epoch 7/20
14/14 [=====] - 151s 11s/step - loss: 0.5632 - acc: 0.8150 - val_loss: 0.3104 - val_acc: 0.8929
Epoch 8/20
14/14 [=====] - 161s 11s/step - loss: 0.3166 - acc: 0.8789 - val_loss: 0.4709 - val_acc: 0.8265
Epoch 9/20
14/14 [=====] - 228s 14s/step - loss: 0.3592 - acc: 0.8605 - val_loss: 0.7447 - val_acc: 0.7602
Epoch 10/20
14/14 [=====] - 146s 10s/step - loss: 0.2982 - acc: 0.8813 - val_loss: 0.2621 - val_acc: 0.8980
Epoch 11/20
14/14 [=====] - 146s 10s/step - loss: 0.4104 - acc: 0.8711 - val_loss: 0.3899 - val_acc:
```

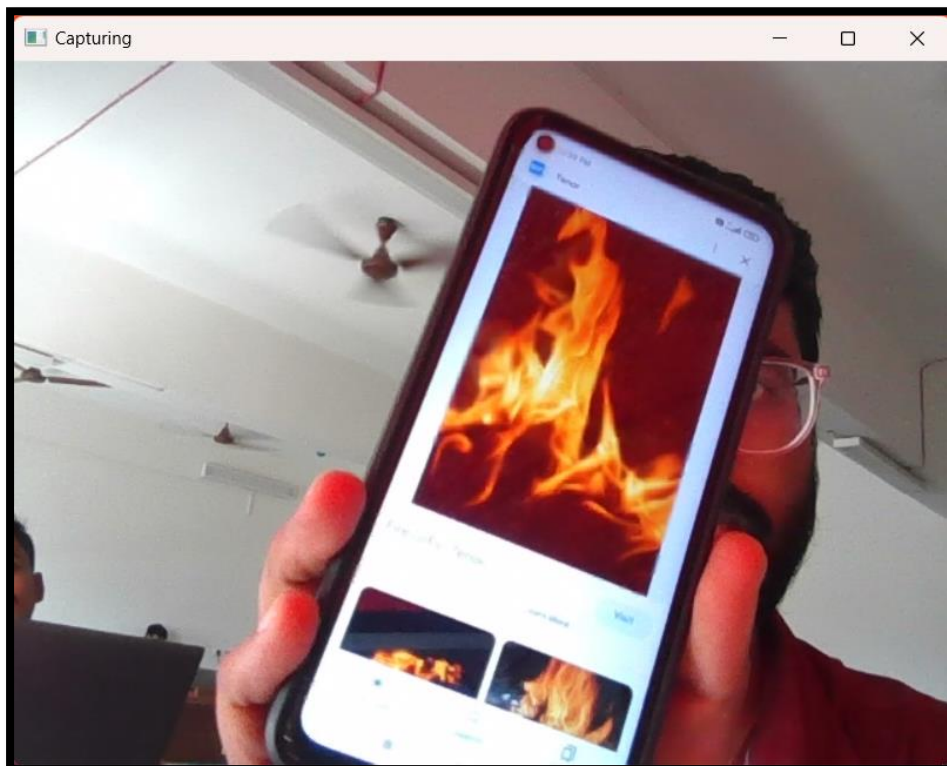
```
In [6]: #To train the top 2 inception blocks, freeze the first 249 layers and unfreeze the rest.
for layer in model.layers[:249]:
    layer.trainable = False
for layer in model.layers[249:]:
    layer.trainable = True
#Recompile the model for these modifications to take effect
from tensorflow.keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['acc'])
history = model.fit(train_generator, steps_per_epoch = 14, epochs = 10, validation_data = validation_generator, validation_steps = 14)

C:\Users\laksh\anaconda3\lib\site-packages\keras\optimizers\optimizer_v2\gradient_descent.py:188: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(SGD, self)._init_(name, **kwargs)

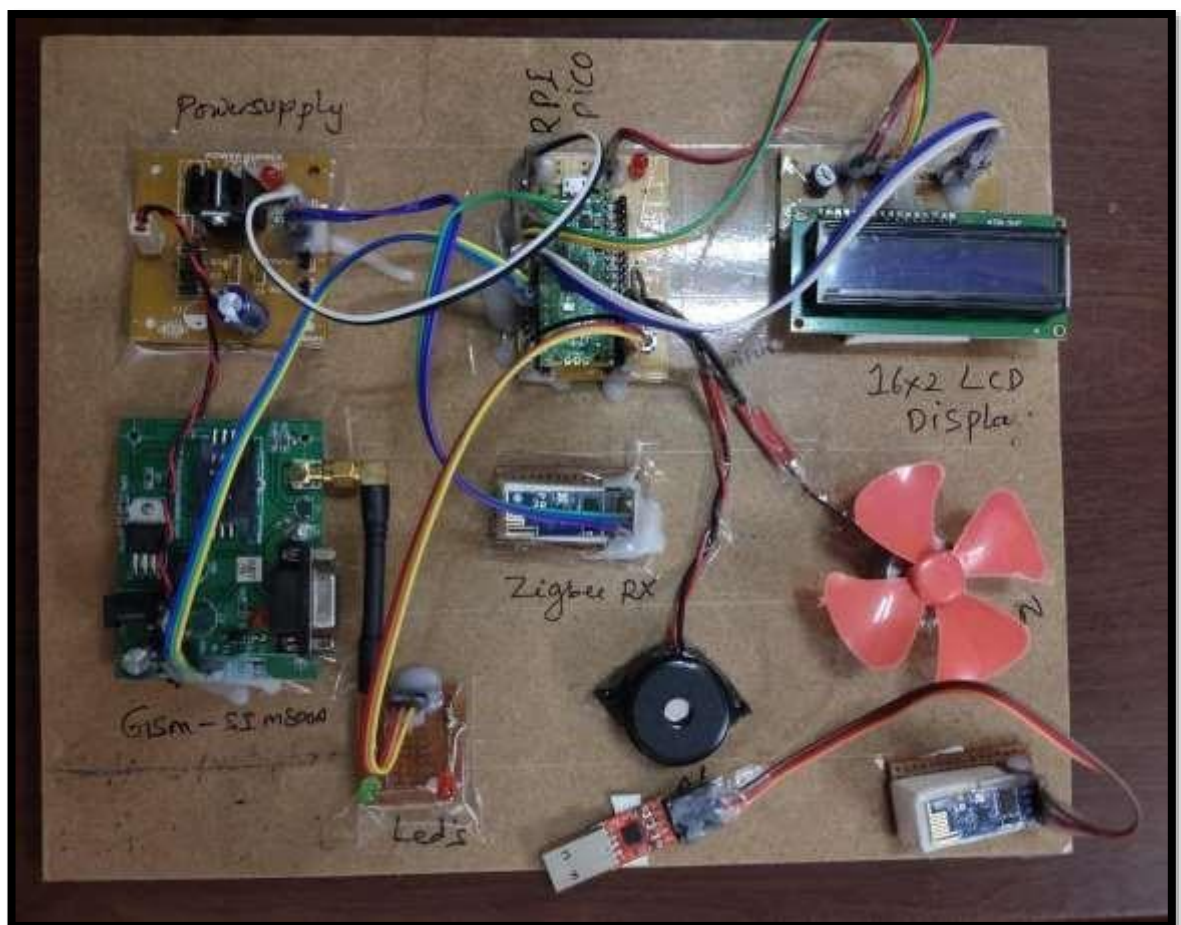
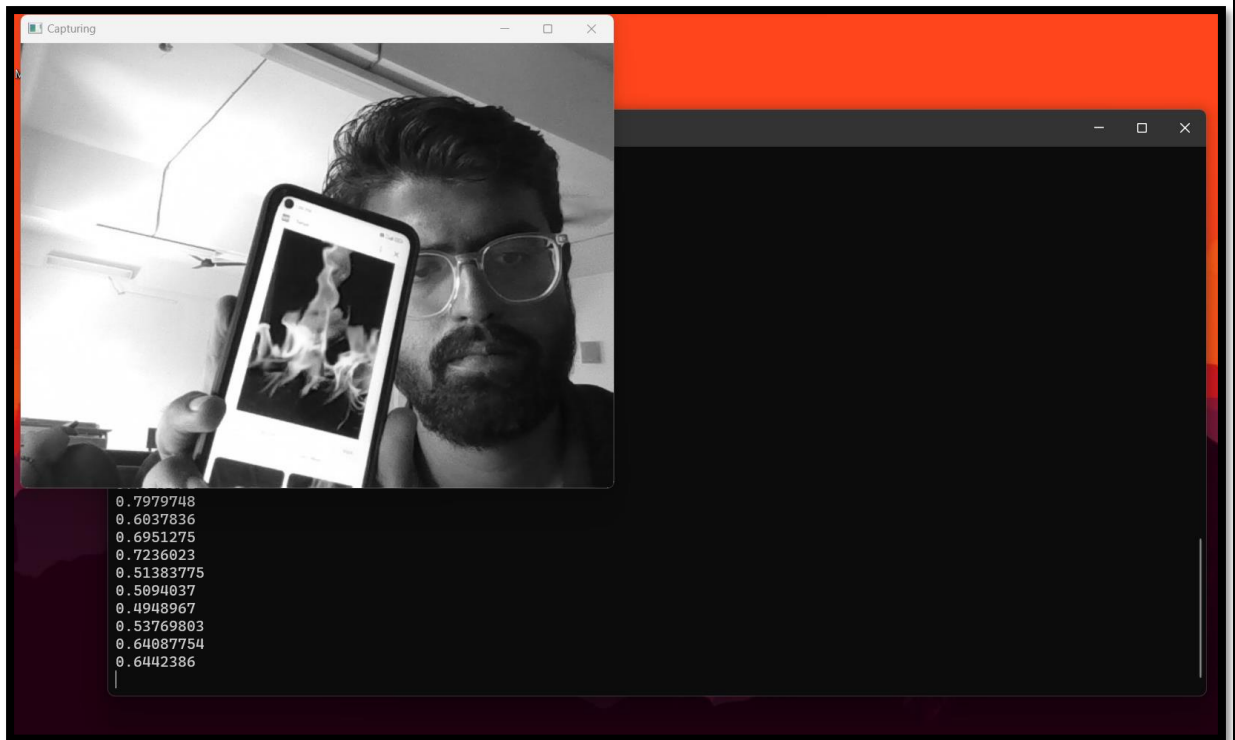
Epoch 1/10
14/14 [=====] - 157s 11s/step - loss: 0.6928 - acc: 0.6856 - val_loss: 0.2758 - val_acc: 0.9082
Epoch 2/10
14/14 [=====] - 103s 7s/step - loss: 0.6428 - acc: 0.7132 - val_loss: 0.3017 - val_acc: 0.8929
Epoch 3/10
14/14 [=====] - 97s 7s/step - loss: 0.5222 - acc: 0.7822 - val_loss: 0.2248 - val_acc: 0.9031
Epoch 4/10
14/14 [=====] - 129s 9s/step - loss: 0.4690 - acc: 0.7971 - val_loss: 0.2908 - val_acc: 0.8878
Epoch 5/10
14/14 [=====] - 117s 8s/step - loss: 0.3919 - acc: 0.8490 - val_loss: 0.3074 - val_acc: 0.8724
Epoch 6/10
14/14 [=====] - 118s 8s/step - loss: 0.3806 - acc: 0.9111 - val_loss: 0.3064 - val_acc: 0.9027
Epoch 7/10
14/14 [=====] - 108s 8s/step - loss: 0.3231 - acc: 0.9041 - val_loss: 0.3213 - val_acc: 0.9076
Epoch 8/10
14/14 [=====] - 114s 8s/step - loss: 0.3149 - acc: 0.9289 - val_loss: 0.3014 - val_acc: 0.9327
Epoch 9/10
14/14 [=====] - 106s 7s/step - loss: 0.3213 - acc: 0.9289 - val_loss: 0.3354 - val_acc: 0.9322
Epoch 10/10
14/14 [=====] - 98s 7s/step - loss: 0.2976 - acc: 0.9398 - val_loss: 0.2802 - val_acc: 0.9429
```

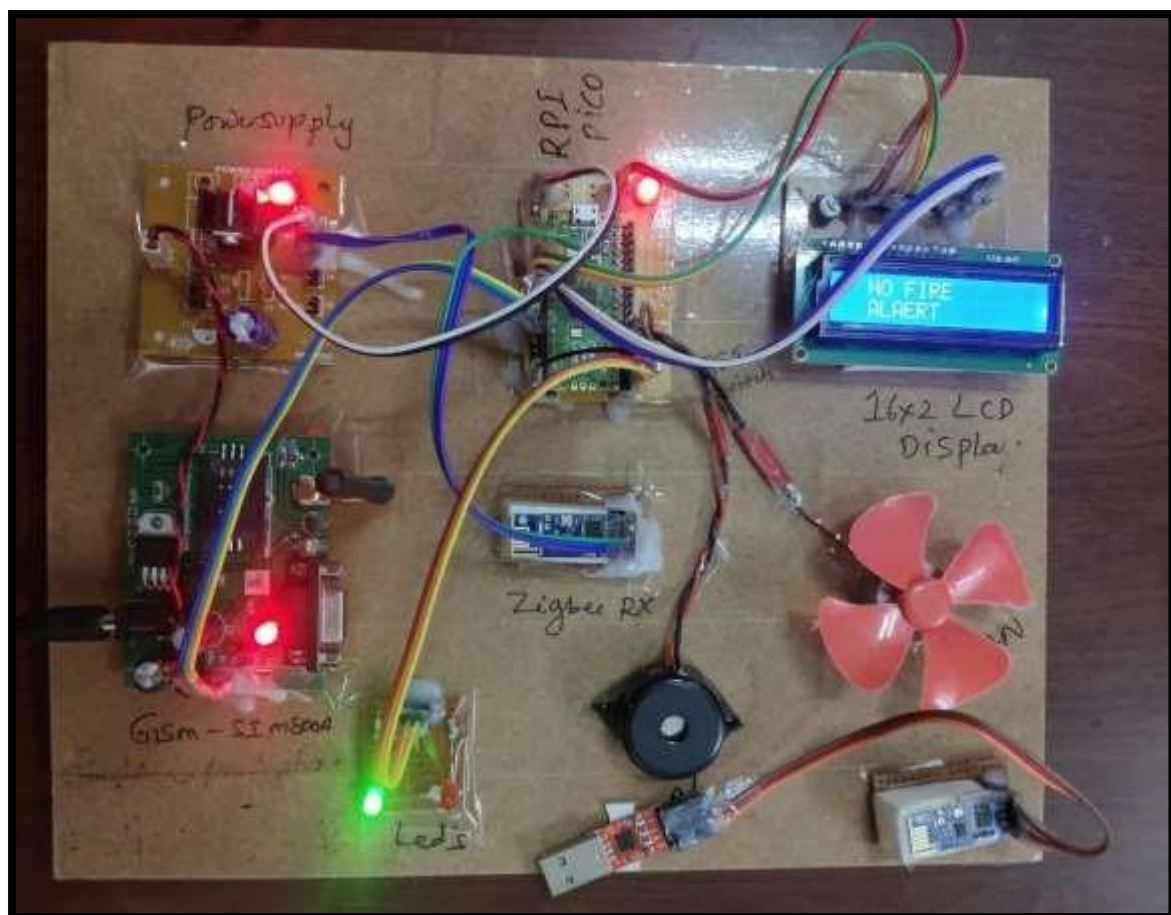
Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	[(None, 224, 224, 3)]	0	[]
conv2d_282 (Conv2D)	(None, 111, 111, 32)	864	['input_4[0][0]']
batch_normalization_282 (Batch Normalization)	(None, 111, 111, 32)	96	['conv2d_282[0][0]']
activation_282 (Activation)	(None, 111, 111, 32)	0	['batch_normalization_282[0][0]']
conv2d_283 (Conv2D)	(None, 109, 109, 32)	9216	['activation_282[0][0]']
batch_normalization_283 (Batch Normalization)	(None, 109, 109, 32)	96	['conv2d_283[0][0]']

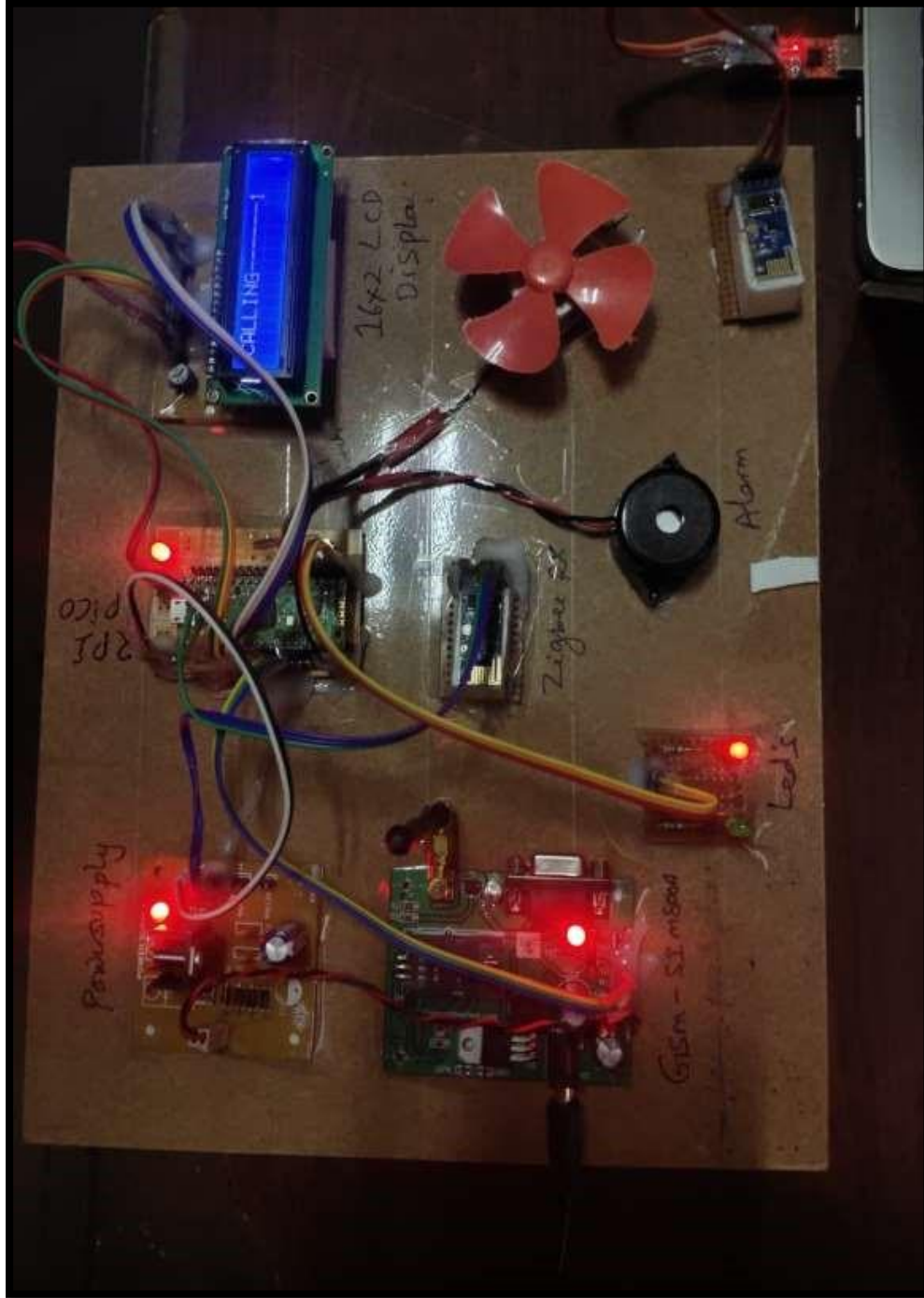
```
File Edit View Navigate Code Refactor Run Tools Git Window Help app_imes.py - app_imes.py
VIT 2ND SEMESTER_WINTER SEMESTER_2020-2021 EXTRA WORK IMES - CSE4056 IMES PROJECT app_imes.py
app_imes.py
1 import cv2
2 import numpy as np
3 from PIL import Image
4 import tensorflow as tf
5 from keras.preprocessing import image
6 #Load the saved model
7 model = tf.keras.models.load_model(r'C:\Users\M Sheshikiran Reddy\Downloads\InceptionV3.h5')
8 video = cv2.VideoCapture(0)
9 while True:
10     frame = video.read()
11     #Convert the captured frame into RGB
12     im = Image.fromarray(frame, 'RGB')
13     #Resizing into 224x224 because we trained the model with this image size.
14     im = im.resize((224,224))
15     img_array = tf.keras.utils.img_to_array(im)
16     img_array = np.expand_dims(img_array, axis=0) / 255
17     probabilities = model.predict(img_array)[0]
18     #Calling the predict method on model to predict 'fire' on the image
19     prediction = np.argmax(probabilities)
20     #If prediction is 0, which means there is fire in the frame.
21     if prediction == 0:
22         frame = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
23         print(probabilities[prediction])
24     cv2.imshow("Capturing", frame)
25     key=cv2.waitKey(1)
26     if key == ord('q'):
27         break
28     video.release()
```



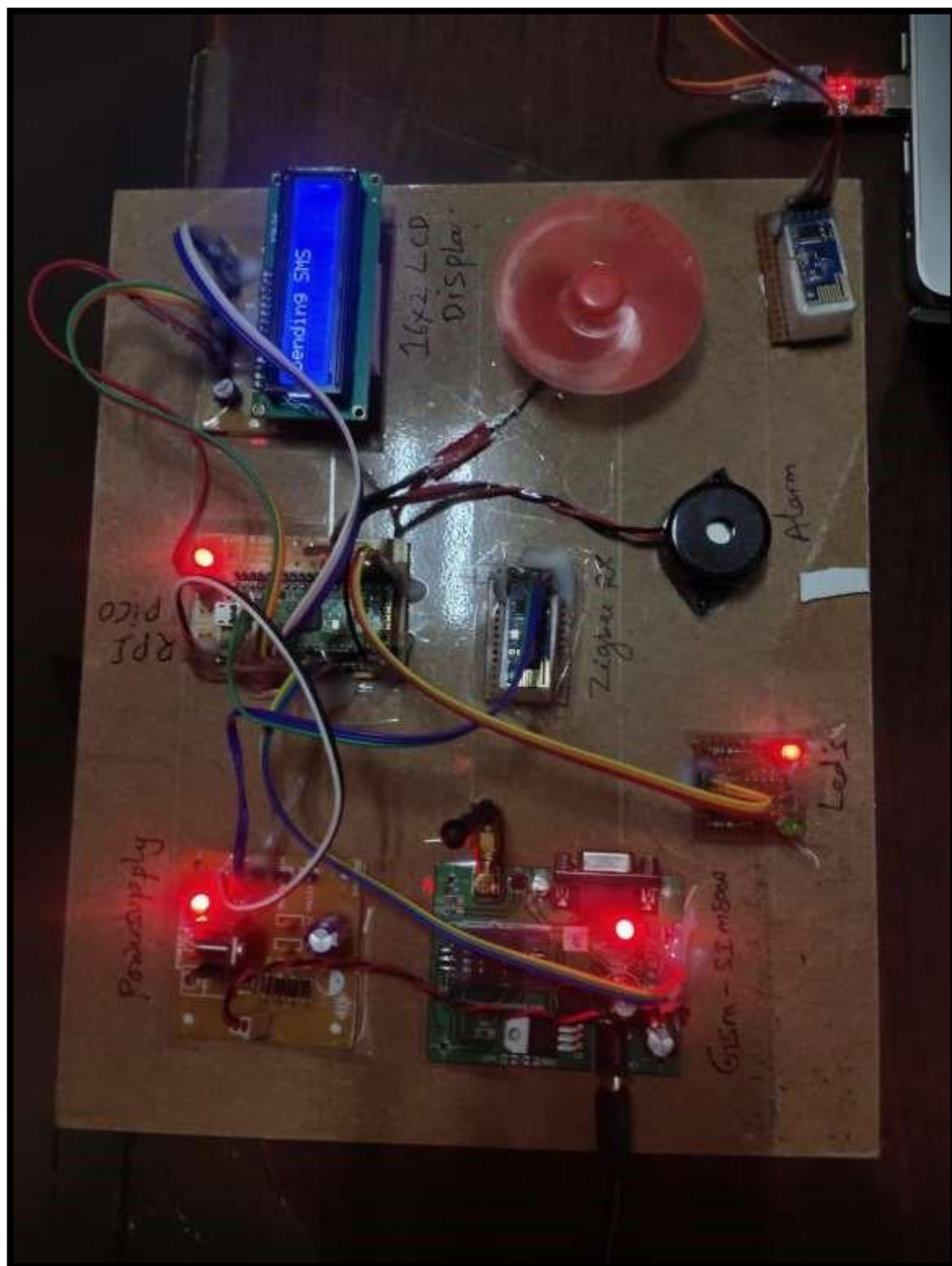


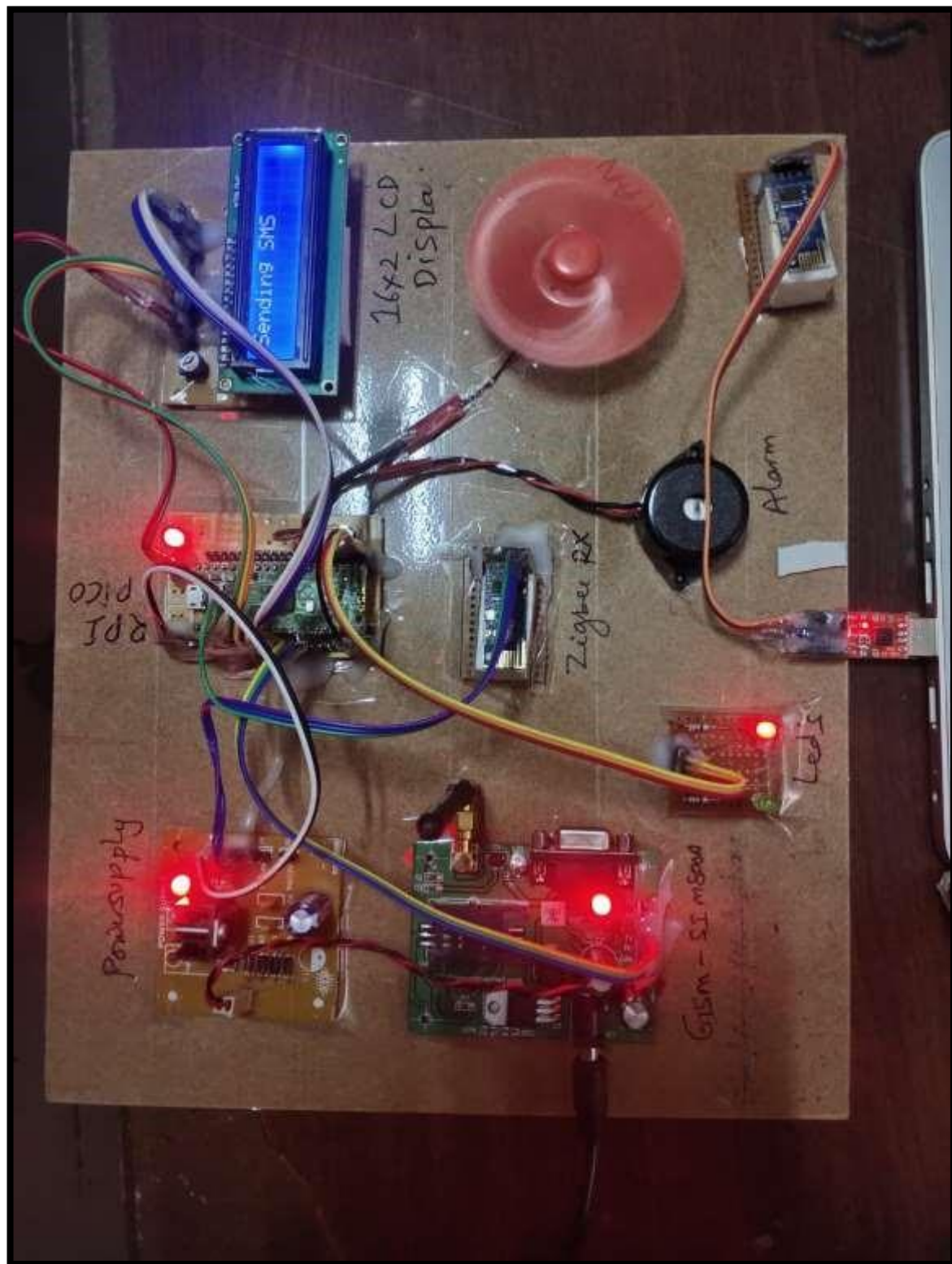


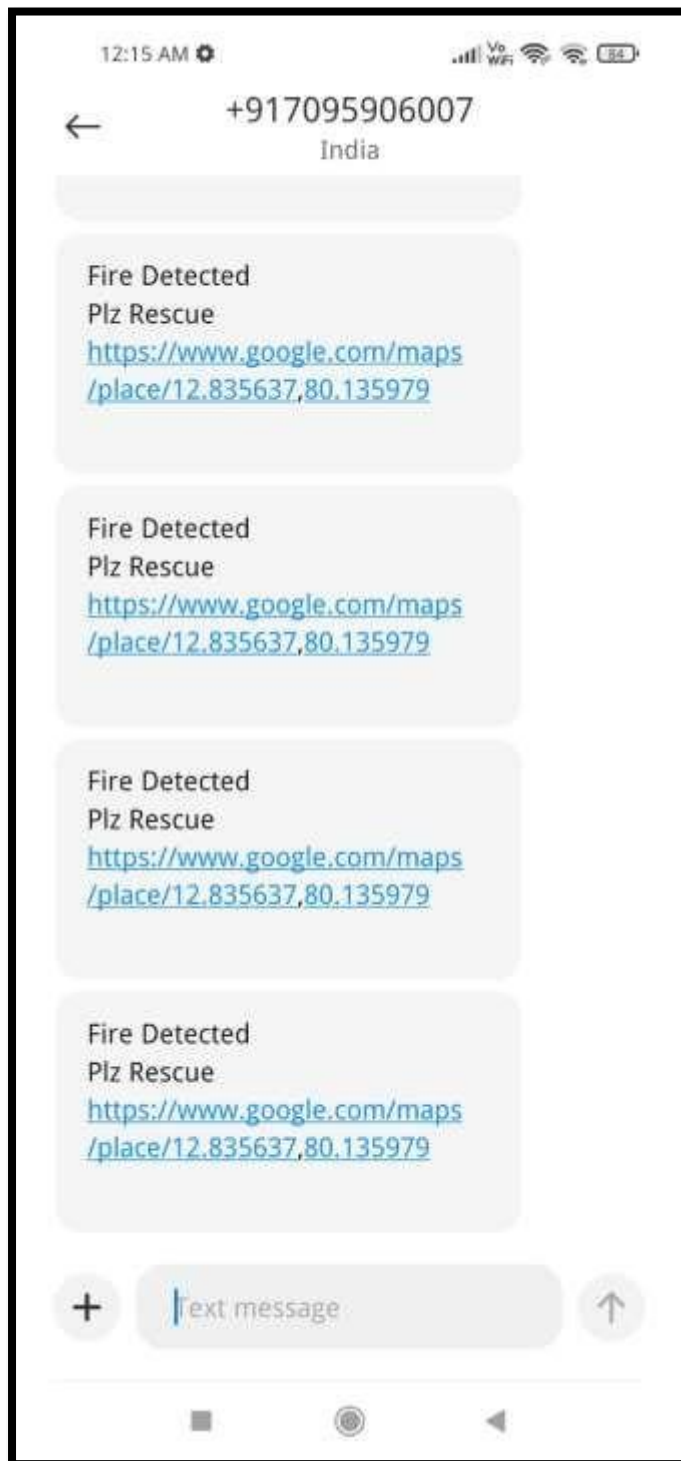












■

THE END