INTELLIGENT FIRE DETECTION AND MONITORING SYSTEM WITH REAL-TIME ALERT AND EMAIL NOTIFICATION

Kalidass M, Vanarasan S
Assistant Professor, Department of Artificial Intelligence and Machine Learning
Sri Sairam College of Engineering, Sai Leo Nagar, Anekal, Bengaluru, India

Atheesha C, Narmatha M, Prathiksha R, Anush Kunjumon
B.E., Department of Artificial Intelligence and Machine Learning
Sri Sairam College of Engineering, Sai Leo Nagar, Anekal, Bengaluru, India

***Abstract*—*Fires continue to be among the most damaging hazards to human life, property, and industrial assets, and the harm they cause grows sharply whenever detection and alerting are delayed. Conventional fire-alarm installations rely on flame, smoke, infrared, or temperature sensors, which cover only a limited physical area and are prone to false alarms triggered by ambient heat, sunlight, or airborne dust. This paper presents an Intelligent Fire Detection and Alert System that combines computer vision with GPIO-based automation on a Raspberry Pi to address these shortcomings. A USB camera continuously observes the surroundings while OpenCV performs real-time image acquisition, and a Haar Cascade classifier examines each processed frame for fire-like regions. To prevent false alerts from a single spurious detection, fire is confirmed only after it is observed in five consecutive frames. Once confirmed, the Raspberry Pi switches on a GPIO-controlled relay and an onboard speaker alarm for an immediate local warning, and, in parallel, encodes the detected frame as a JPEG image and sends it as an email alert to a configured recipient through Gmail's SMTP server, authenticated using a Gmail App Password. A Flask-based web dashboard displays the live camera feed and current fire status. In contrast to conventional sensor-driven approaches, the proposed system pairs vision-based situational awareness with an automated local alarm and a remote, evidence-attached email notification, resulting in a fast, dependable, and low-cost safety solution for homes, industries, laboratories, warehouses, offices, and public buildings.*

Keywords: *Fire Detection, Haar Cascade, OpenCV, Raspberry Pi, Relay, Speaker Alarm, Email Alert, SMTP, Flask Dashboard, IoT.*

I. INTRODUCTION

Fire safety is essential to protecting human life, property, and infrastructure across residential, industrial, and public settings. An uncontrolled fire can spread within minutes, so even a brief delay in detecting or responding to it can lead to losses that cannot be undone. Conventional fire-detection installations depend on flame sensors, smoke detectors, infrared sensors, and temperature sensors that raise an alarm only after a physical threshold has been crossed. Although useful, such systems offer restricted detection coverage, react only once a fire has already progressed far enough to be sensed, and remain vulnerable to false alarms caused by heat sources, sunlight, or dust unrelated to any actual fire.

To address these shortcomings, this paper proposes an Intelligent Fire Detection and Alert System built around continuous visual monitoring rather than point-based sensing. A USB camera mounted on a Raspberry Pi captures the monitored area in real time, after which OpenCV converts each frame to grayscale and resizes it before a Haar Cascade classifier scans the frame for fire-like regions. To avoid reacting to a single spurious detection, the system counts consecutive fire-positive frames and confirms a fire only once this count reaches five. When fire is confirmed, the Raspberry Pi immediately switches on a GPIO-controlled relay and its onboard speaker alarm to warn occupants, and, in parallel, encodes the current frame as a JPEG image and emails it as evidence to a configured recipient through Gmail's SMTP server. By combining AI-assisted visual detection with an immediate local alarm and an automated, evidence-attached remote notification, the system is intended to cut response time, minimize false alarms, and deliver a low-cost, scalable fire-safety solution for homes, industries, laboratories, warehouses, offices, and public buildings.

The main contributions of this work are as follows. First, a vision-based fire-detection pipeline is developed in which OpenCV preprocessing and a Haar Cascade classifier examine each camera frame for fire-like regions entirely on a Raspberry Pi, without requiring GPU or cloud inference. Second, a five-consecutive-frame confirmation rule filters out momentary false positives before any alarm or notification is triggered. Third, a dual-response mechanism is implemented: an immediate local response through a GPIO-controlled relay and onboard speaker alarm, and a remote response through an automatically generated, evidence-attached email sent via Gmail SMTP using a Gmail App Password. Finally, duplicate-alert protection, a cooldown timer, and a ten-consecutive-clear-frame re-arm rule are combined with a Flask-based web dashboard that reports live camera status, fire state, and an event log so that every automated response can be observed and reviewed.

The proposed system aims to improve fire detection accuracy by continuously monitoring the surroundings and reducing false alarms. It also provides rapid local and remote alerts,

enabling timely action to minimize damage and improve overall fire safety.

Fig. 1. Intelligent Fire Detection and Alert System

II. LITERATURE SURVEY

Prior work on fire detection can be grouped into sensor-based and machine-learning methods, handcrafted vision methods, and deep-learning methods. The studies below are reviewed with respect to detection accuracy, robustness to environmental conditions, and the availability of an automatic response after detection.

A hybrid detection method combining Vision Transformers, 3D-CNNs, and Transformer encoders has been proposed to jointly capture the spatial and temporal patterns present in fire and smoke video sequences, improving accuracy and robustness relative to earlier deep-learning methods. This approach, however, addresses detection alone and does not include any local or remote automatic alerting mechanism such as a relay, speaker alarm, or email notification [1].

An IoT-based fire-detection framework has been built by benchmarking several machine-learning algorithms including k-Nearest Neighbor, Multilayer Perceptron, Naive Bayes, Decision Tree, and Random Forest on data gathered from IoT-connected smoke detectors. This study does not use computer vision or a camera-based classifier such as a Haar Cascade, and it stops at raising an alert without any local relay activation or evidence-attached email notification [2].

A Raspberry Pi-based dual-stage fire-crisis system has been built that first triggers an infrared flame sensor and then uses a Raspberry Pi camera with OpenCV-based image checks to confirm the event before sounding a piezo buzzer and issuing a web-based IoT alert, reporting high detection accuracy and a large reduction in false alarms relative to a single-sensor baseline. This approach, however, still depends on a dedicated infrared sensor stage and stops at a local buzzer and a basic web alert, without an automatically generated, image-evidence email sent to a remote recipient [3].

A CNN-based system for identifying fire and smoke directly from video sequences has been proposed, in which the network automatically extracts visual features and surpasses traditional image-processing techniques in detection accuracy. This system, however, is confined to detection and alert generation and does not extend to an automated relay, speaker alarm, or evidence-attached email notification [4].

The Haar Cascade framework, introduced by Viola and Jones, combines AdaBoost-selected rectangular (Haar-like) features into a rejection cascade that can detect objects in real time on modest hardware without GPU acceleration. Being a general-purpose object-detection framework rather than one designed specifically for fire, it requires careful tuning of scale, neighbor, and detection-window parameters, together with a temporal confirmation rule, to be used reliably for fire recognition, which motivates its adaptation in the system proposed here [5].

An earlier vision-based fire-detection method used flame color and motion characteristics within video sequences to identify fire faster than conventional flame sensors under controlled conditions. Because it depends on handcrafted image features, its robustness declines under varying lighting conditions, and it lacks both a temporal confirmation rule and any automatic local or remote alerting mechanism [6].

Separately, the introduction of a deep convolutional network for large-scale image classification showed that CNNs can learn discriminative visual features directly from raw images and can outperform earlier hand-designed approaches [7], which is part of why several of the reviewed fire-detection systems adopt a CNN-based approach. The system proposed here instead favors a lightweight, classical Haar Cascade pipeline, avoiding the GPU or cloud inference that a deep-learning classifier would typically require on a Raspberry Pi.

A probabilistic approach to vision-based fire detection in video combined several visual features of candidate fire regions within a statistical framework to decide whether a region corresponds to fire [8]. Although this improved on rules based on color alone, it still depends on handcrafted features and ends at detection, with no automatic relay, alarm, or email-based alerting stage.

Taken together, the reviewed studies fall into three groups: machine-learning methods applied to IoT smoke-sensor data [2], handcrafted vision methods based on color, motion, and statistical features [6], [8], and deep-learning detectors for images and video [1], [4]. A closely related Raspberry Pi-based system [3] adds a vision-confirmation stage to a sensor trigger but stops at a local buzzer and a basic web alert. None of the surveyed works combines a lightweight, camera-only Haar Cascade pipeline with a temporal confirmation rule, an immediate local relay/alarm response, and an automatically generated, evidence-attached email alert in a single low-cost device. This gap motivates the system proposed in this paper, and Table I compares the reviewed approaches with the proposed system.

Accordingly, the objectives of this work are to (i) detect fire visually over the camera field of view using OpenCV preprocessing and a Haar Cascade classifier, (ii) reduce false alarms through a five-consecutive-frame confirmation rule rather than a single-frame decision, (iii) respond locally and immediately through a GPIO-controlled relay and speaker alarm, and (iv) notify a remote recipient automatically by

emailing a JPEG snapshot of the detected fire through Gmail SMTP, with duplicate-alert protection, a cooldown timer, and a Flask-based dashboard for live monitoring and event logging.

III. PROBLEM STATEMENT

Fire detection and response in most existing environments face several fundamental limitations, each of which the proposed AI-driven system is designed to address

Limited Detection Coverage

Traditional flame, smoke, infrared, and temperature sensors can sense fire only within a fixed proximity and orientation, leaving large areas unmonitored. By combining a continuously monitoring camera with OpenCV-based Haar Cascade detection, the proposed system extends detection coverage to the full field of view and identifies fire visually rather than relying on point sensors.

False Alarms from Environmental Factors

Heat, sunlight, and dust frequently trigger false alarms in sensor-based systems, undermining their reliability. The proposed system instead uses OpenCV-based preprocessing together with a Haar Cascade classifier and a five-consecutive-frame confirmation rule to visually distinguish genuine fire from environmental noise and momentary false detections, thereby reducing false positives.

Lack of Immediate Local Alarm and Remote Notification

Most existing systems do no more than raise a local alarm, leaving anyone away from the premises unaware of the event until they are informed manually, which increases both response time and risk. By integrating a GPIO-controlled relay and speaker alarm with an automatic, evidence-attached email sent through Gmail SMTP, the proposed system alerts occupants immediately on site while also notifying a remote recipient with a JPEG snapshot of the detected fire, cutting the delay between detection and awareness.

Fig. 2. AI-Camera-Based Fire Monitoring and Alerting

IV. PROPOSED SYSTEM ARCHITECTURE

System Architecture

The proposed fire-detection and alerting system follows a continuous monitor-detect-confirm-respond workflow, running entirely on a Raspberry Pi. The process is organized into six key stages:

Capture: A USB camera continuously captures video frames from the monitored area (home, industry, laboratory, warehouse, office, or public building).

Preprocess: OpenCV, running on the Raspberry Pi, resizes each frame and converts it to grayscale to prepare it for detection.

Detect: A Haar Cascade classifier scans each preprocessed frame for fire-like regions using tuned scale and neighbor parameters.

Confirm: The system counts consecutive frames in which fire is detected and confirms a fire event only once this count reaches five, filtering out momentary false positives.

Respond: Once fire is confirmed, the Raspberry Pi immediately switches on a GPIO-controlled relay and its onboard speaker alarm to warn nearby occupants.

Notify: In parallel, the current frame is encoded as a JPEG image and emailed, together with the detection time and device status, to a configured recipient through Gmail's SMTP server; once ten consecutive clear frames are observed, the relay and alarm switch off and the system re-arms itself for the next event.

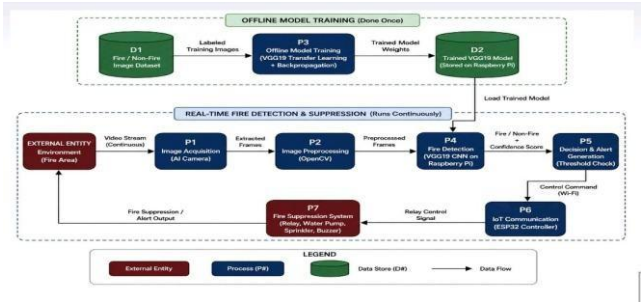

Fig. 3. Fire Detection, Confirmation, and Alert Workflow

A. Hardware and Software Integration

A Raspberry Pi forms the entire computational and control core of the system, hosting OpenCV for image preprocessing, the Haar Cascade classifier for fire detection, the GPIO-driven relay and speaker alarm, the Flask web dashboard, and the Gmail SMTP email client. Because a single device handles detection, decision-making, actuation, and notification, no separate wireless microcontroller is required. The key elements of this architecture include:

Edge Processing: All image preprocessing and Haar Cascade inference run locally on the Raspberry Pi, enabling low-

latency fire detection without dependence on external cloud infrastructure or a GPU.

Network Communication: The Raspberry Pi uses its own Wi-Fi connection to host the Flask dashboard and to reach Gmail's SMTP server for outgoing email alerts, with the email sent in a background thread so detection continues uninterrupted.

Response Layer: The Raspberry Pi's GPIO pins drive a relay module and an onboard speaker directly, converting a software-level detection into both a physical switching action and an audible local warning without an intermediate controller.

Scalability: The modular software design allows multiple camera-and-Raspberry Pi nodes, each with its own relay, alarm, and email configuration, to be deployed across a large facility such as a warehouse or industrial plant.

Reliability: A five-consecutive-frame confirmation rule, a 30-second email cooldown, and a ten-consecutive-clear-frame re-arm rule together ensure that the relay, alarm, and email alert are triggered only for a sustained fire event and are not repeated unnecessarily while that event continues.

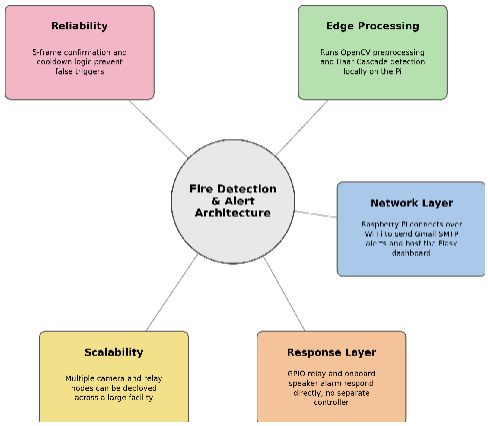

Fig. 4. Components of the Fire Detection and Alert Architecture

V. METHODOLOGY AND IMPLEMENTATION

Frame Acquisition and Preprocessing

Each frame captured by the USB camera is preprocessed with OpenCV on the Raspberry Pi, which resizes it to 640×480 pixels and converts it from color to grayscale. This preprocessing step keeps the input consistent for the Haar Cascade classifier despite variations in lighting and camera conditions, and keeps the per-frame processing lightweight enough to run in real time on the Raspberry Pi.

Haar Cascade Fire Detection

Fire detection is performed with a Haar Cascade classifier, a pretrained object-detection model based on the Viola-Jones framework rather than a trained neural network, so no separate training phase or dataset is required for this stage.

Algorithm: Haar Cascade Classification (Viola-Jones Framework)

Type: Classical Computer-Vision Detection Algorithm

Explanation: The classifier scans each grayscale frame at multiple scales, applying a cascade of simple rectangular (Haar-like) features to quickly reject regions that are unlikely to contain fire and concentrate computation on promising candidate regions. The detector is configured with a scale factor of 1.05, a minimum neighbor count of 12, and minimum and maximum detection windows of 80×80 and 250×250 pixels, values chosen to balance sensitivity against false positives for the flame patterns seen by the camera.

Real-Time Fire Detection and Frame Confirmation

During operation, the USB camera continuously streams frames to the Raspberry Pi, where OpenCV performs real-time preprocessing before each frame is scanned by the Haar Cascade classifier.

Algorithm: Consecutive-Frame Confirmation Rule

Type: Rule-Based Temporal Filtering Algorithm

Explanation: A counter is incremented each time the classifier detects a fire-like region in a frame and reset to zero as soon as a frame with no detection occurs. Fire is confirmed only once this counter reaches five consecutive frames, which requires the detection to persist briefly rather than relying on any single frame.

Because detection relies on classical image features rather than a trained deep-learning model, the approach requires no GPU and can run continuously on the Raspberry Pi's own processor, at the cost of being somewhat less robust to unusual flame appearances than a deep-learning classifier trained on a large fire dataset.

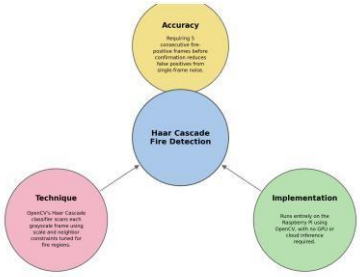

Fig. 5. Haar Cascade Fire Detection Approach

B. Alert Generation and Automated Response

When the Haar Cascade classifier confirms fire for five consecutive frames, the Raspberry Pi immediately triggers its local response and, in a background thread, its email notification.

Algorithm: Rule-Based Dual-Channel Alerting

Type: Decision and Notification Algorithm

Explanation: The confirmed fire event switches the GPIO relay ON and activates the onboard speaker alarm for an immediate, human-perceptible warning. In parallel, the current frame is encoded as a JPEG image and attached to an email message containing the detection time and device status; the message is sent through Gmail's SMTP server (smtp.gmail.com, port 465) over an SSL connection, authenticated with the sender's Gmail App Password rather

than the account's normal password. A fire_email_sent flag together with a 30-second cooldown prevents the same confirmed event from generating repeated emails, and the flag is cleared, re-arming the system, only after ten consecutive clear frames are observed.

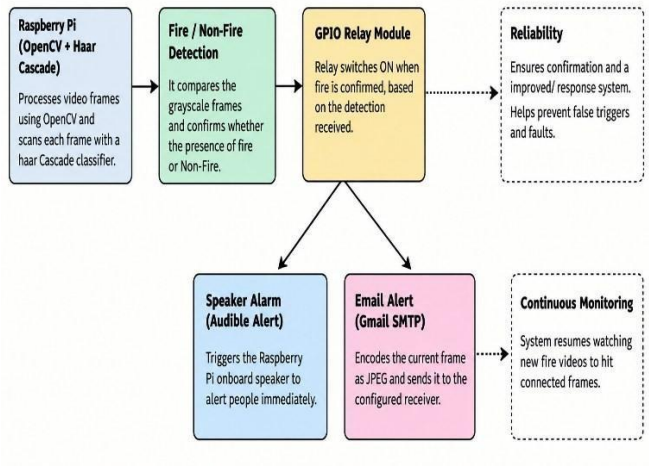

Fig. 6. Detection-to-Response Actuation Pipeline

IV. SYSTEM RELIABILITY AND SAFETY CONSIDERATIONS

Because the system controls physical relay hardware and sends automated notifications, reliability and false-alarm prevention are critical design concerns. The following measures are incorporated:

Frame Confirmation: The relay, speaker alarm, and email alert activate only once fire has been detected in five consecutive frames, reducing accidental triggers caused by a single spurious detection.

Duplicate-Alert Protection: A fire_email_sent flag and a 30-second cooldown prevent the same confirmed fire event from generating repeated emails, and Gmail authentication failures are caught and logged without interrupting fire detection.

Continuous Monitoring Loop: The camera and detection pipeline operate continuously, and the relay, alarm, and email system automatically re-arm themselves after ten consecutive clear frames, allowing the system to respond to a future event without manual reset.

Audible Warning: The onboard speaker alarm provides an immediate, independent local alert to occupants alongside the automated email notification, ensuring a human-perceptible response even if network conditions delay the email.

Despite these safeguards, the system still faces practical challenges, including maintaining consistent camera visibility across large or obstructed areas, keeping the Raspberry Pi's Wi-Fi connection stable enough to reach Gmail's SMTP server, and tuning the Haar Cascade parameters and confirmation-frame count to the lighting and environmental conditions found at different deployment sites.

V. RESULTS AND DISCUSSION

The proposed system was implemented as a working prototype and evaluated through its hardware setup and a Flask-based web dashboard that displays live detection status, the automated response sequence, and system analytics. The results below cover the hardware prototype, the normal monitoring state, a fire-detection and alert event, and the logged detection-and-response history.

Hardware Prototype Setup

Fig. 7 shows the Raspberry Pi and USB camera used to test the prototype during development. A USB web camera acts as the vision sensor and continuously captures frames of the monitored area, which are passed to a Raspberry Pi, housed in a protective case, running the OpenCV preprocessing and Haar Cascade detection described in Section V. Connecting wires link the board to its power supply and to the GPIO-controlled relay and speaker alarm described in Section IV. This arrangement shows that the system can be built from low-cost, easily available components and deployed without cloud infrastructure.

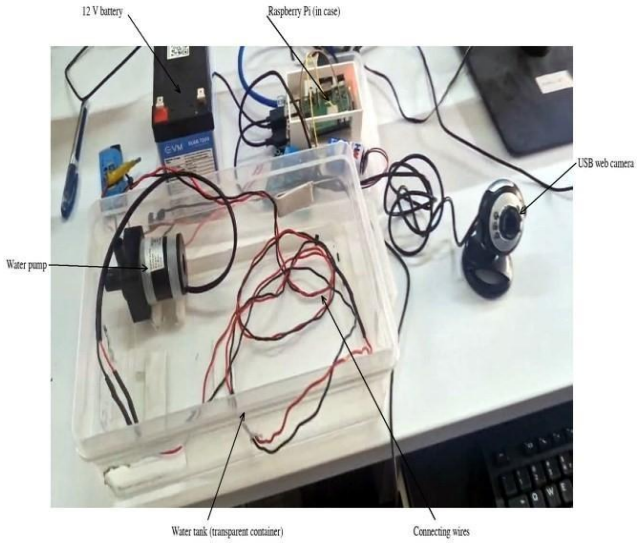

Fig. 7. Hardware Prototype of the Fire Detection and Alert System

Real-Time Monitoring Dashboard

Fig. 8 shows the dashboard during normal operation. The camera in Laboratory 1 (Camera 1) was under continuous surveillance, and the most recent frame check had been completed 2 s earlier. The Haar Cascade classifier found no fire-like region in the latest frame, so the detection counter read 0 out of the 5 frames required for confirmation, and the system remained in the System Ready state.

The Control Hub lets the operator choose between AI Auto mode, in which the Haar Cascade output drives the relay and alarm, and Manual mode. The status panel shows the

Raspberry Pi Online, with the relay and speaker alarm both Off. A Test Relay & Alarm control allows the response hardware to be verified without a real fire.

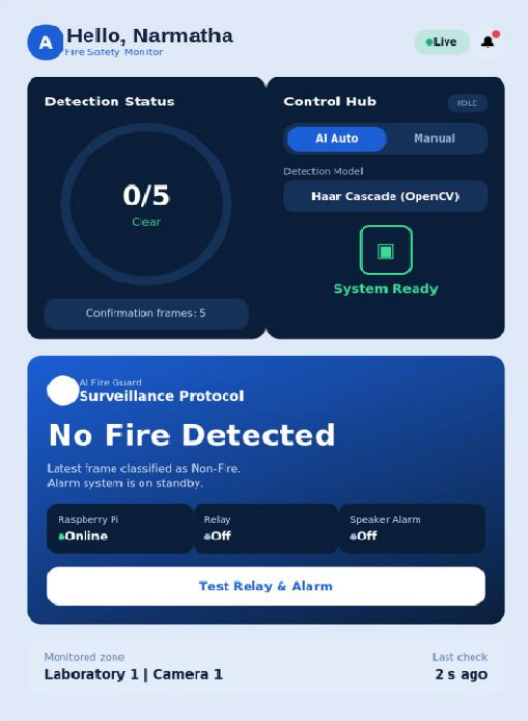

Fig. 8. Real-Time Monitoring Dashboard Showing Non-Fire Status and Device State

Fire Detection and Automatic Alert

Fig. 9 shows the system response when a fire was detected. The Haar Cascade classifier detected a fire-like region in five consecutive frames, so the frame-confirmation rule classified the event as a confirmed fire and started the alert sequence in Auto response mode. A notification stating that an alert had been sent automatically was raised immediately, informing the operator of the event and of the response already under way.

The response sequence then ran in the order defined by the system architecture. The Raspberry Pi switched the GPIO relay ON and activated its onboard speaker alarm to warn nearby occupants. At the same time, the current frame was encoded as a JPEG image and emailed, together with the detection time and device status, to the configured recipient through Gmail's SMTP server. Each stage was confirmed on screen as it completed, following the detect-confirm-respond-notify workflow described in Section IV. The operator could still intervene through the Stop Alarm and Dismiss controls, and the interface noted that monitoring resumes automatically once ten consecutive clear frames are observed.

Fig. 9. Emergency Protocol Screen Showing Fire Detection and Automatic Response Sequence

Email Alert Notification

The system also provides an automated email alert when a possible fire event is detected. As shown in Fig. 10, the notification contains the fire detection status, detection time, relay status, and Raspberry Pi alarm status. The system also attaches the captured camera frame containing the detected fire region, providing visual evidence for immediate verification. This demonstrates the capability of the proposed system to remotely notify the user and support rapid response even when the monitored area is unattended.

The system also provides a reliable email-based notification mechanism for remote monitoring. When a fire is confirmed, the Raspberry Pi automatically captures the current frame and send it as an email attachment to a configured recipient using Gmail SMTP. The alert includes the detected fire status and supporting visual evidence, allowing the user to verify the incident remotely and take immediate action even when they are away from the monitored location. This notification mechanism improves response time by ensuring that critical fire information reaches the responsible person without delay. It also provides an additional layer of safety by enabling continuous remote awareness of the monitored area.

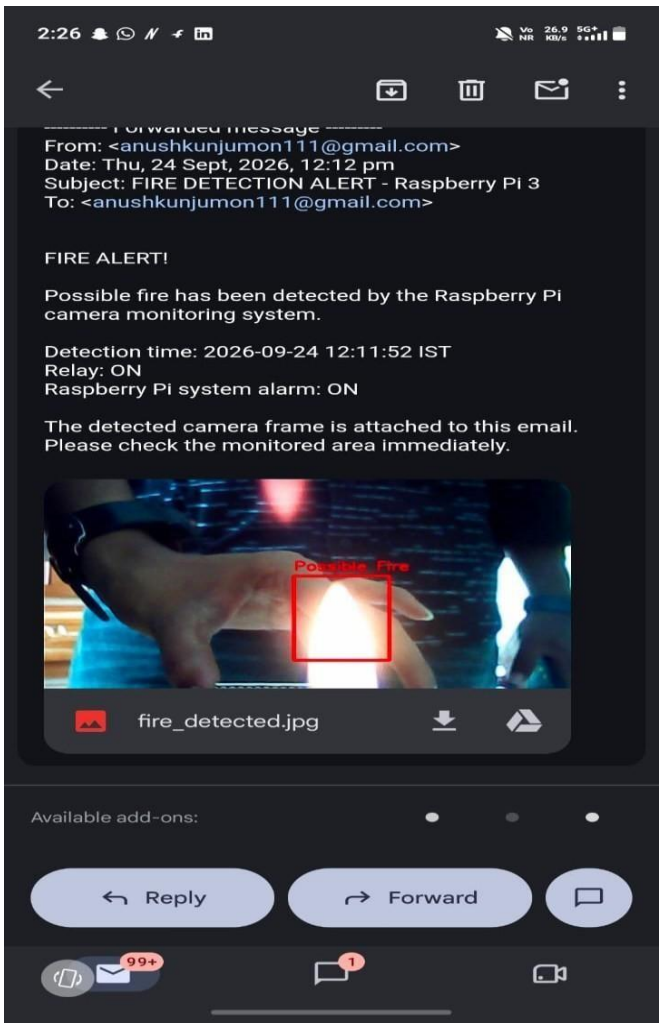

Fig. 10. Automated Email Alert Notification with Fire Detection Image and System Status.

Detection and Response Log

Fig. 11 shows the analytics view, which records every detection and response event. On the day shown, the system analyzed 1,284 frames and raised one fire alert, which was handled automatically. The event log for that alert captures the full response cycle. The fire was confirmed at 10:42 AM after five consecutive fire-positive frames, and the relay and speaker alarm were switched ON automatically at the same logged time, together with the email alert sent to the configured recipient. At 10:46 AM the relay and alarm were switched OFF once ten consecutive clear frames had been observed, and the camera resumed active monitoring at 10:47 AM.

Because the relay, alarm, and email alert were triggered at the same logged minute as the confirmed detection, the delay between detection and response was very small. The log also confirms that the closed monitor-detect-confirm-respond-notify loop completed without manual intervention. The 7-day alert history shows one alert on Wednesday, two on Saturday, and one on Sunday, with none on the other days, so the system also keeps a record of alert frequency over time.

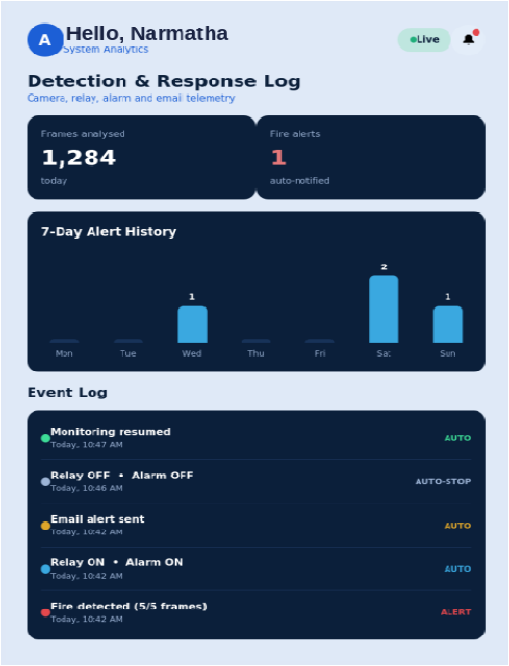

Fig. 11. Detection and Response Log with Alert History and Event Timeline

Discussion

The results show that the system classified frames reliably in normal conditions. It correctly reported No Fire Detected during routine monitoring and raised only one alert out of 1,284 analyzed frames, which suggests a low rate of spurious detections. The relay, alarm, and email alert were triggered only once fire had persisted for five consecutive frames, and the response started automatically. This supports the reliability goal of avoiding unnecessary relay activation and duplicate email alerts from momentary or single-frame false detections.

The system also completed the entire response cycle without human input. It went from detection through frame confirmation, relay activation, speaker alarm, and email notification to automatic shutdown, then resumed monitoring, while the operator could still override the process at any stage. Compared with conventional flame, smoke, or infrared sensors, which respond only after a physical threshold has been crossed and typically only sound a local alarm, the vision-based approach detects fire directly from the camera field of view and links detection both to an immediate local response and to an evidence-attached remote email notification. The Flask dashboard adds real-time visibility of system status and a time-stamped record of every event, which supports monitoring and post-incident review by facility managers even when they are away from the premises.

Although the prototype performed reliably in the tested scenario, its results depend on consistent camera visibility, a stable Wi-Fi connection from the Raspberry Pi to Gmail's

SMTP server, and Haar Cascade parameters that generalize to the lighting conditions of each deployment site. The five-frame confirmation rule and the Haar Cascade's scale and neighbor settings balance quick response against unnecessary relay activation, and both can be tuned to suit the risk level and camera placement of the monitored area. Testing across a wider range of fire types, lighting conditions, and camera placements would further validate the system's accuracy and response time. These findings support the system as a practical, low-cost, and scalable solution for automated fire alerting in laboratories, industries, and public buildings.

VI. FUTURE ENHANCEMENTS

Deep-Learning Upgrade: Replacing or supplementing the Haar Cascade classifier with a lightweight trained CNN to improve robustness to varied flame appearances and lighting conditions.

Multi-Sensor Fusion: Combining the camera with smoke and thermal sensors to further reduce false alarms and improve detection under low-visibility conditions.

Mobile App Integration: Extending the Flask dashboard into a dedicated mobile application with push notifications, alongside the existing email alert.

Multi-Node Deployment: Scaling the system across multiple camera-and-Raspberry Pi nodes to provide large-scale coverage in industrial plants, warehouses, and public buildings.

VII. CONCLUSION

The proposed Intelligent Fire Detection and Alert System offers a fast, low-cost, and automated response to the key limitations of conventional sensor-based fire-safety systems. By combining a USB camera, OpenCV preprocessing, and a Haar Cascade classifier with a five-consecutive-frame confirmation rule, a GPIO-controlled relay and speaker alarm, and an automated, evidence-attached email alert sent through Gmail SMTP, the system widens detection coverage, reduces false alarms, and removes the delay caused by manual intervention or the absence of anyone on-site. The integration of vision-based fire detection with an immediate local alarm and a remote, evidence-attached notification represents a practical advance over traditional flame, smoke, and infrared sensor- based systems, offering a scalable safety solution for homes, industries, laboratories, warehouses, offices, and public buildings.

REFERENCES

[1] U. K. Lilhore et al., "Real Time Fire and Smoke Detection Using Vision Transformers and Spatiotemporal Learning," Scientific Reports (Nature), 2026.

[2] A. A. Ayrancı and B. Erkmen, "IoT-Based Fire Detection: A Comparative Study of Machine Learning Techniques," Nigde Omer Halisdemir University Journal of Engineering Sciences, 2024.

[3] F. Irianto, J. A. J. Alsayaydeh, A. W. Y. Khang, M. Farid, and S. G. Herawan, "Design and Implementation of a Smart Dual-Stage Fire Crisis Management System Using Raspberry Pi for Safety and Security Applications," International Journal of Safety and Security Engineering, vol. 15, no. 10, 2025.

[4] S. Frizzi, R. Kaabi, M. Bouchouicha, J. M. Ginoux, E. Moreau, and F. Fnaiech, "Convolutional Neural Network for Video Fire and Smoke Detection," IECON 2016 – 42nd Annual Conference of the IEEE Industrial Electronics Society, pp. 877-882, 2016.

[5] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 511-518, 2001.

[6] T. Celik and H. Demirel, "Fire Detection in Video Sequences Using a Generic Color Model," Fire Safety Journal, Vol. 44, No. 2, pp. 147-158, 2009.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, Vol. 25, 2012.

[8] P. V. K. Borges and E. Izquierdo, "A Probabilistic Approach for Vision-Based Fire Detection in Videos," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 20, No. 5, pp. 721-731