Integrating ESP32-Based Edge AI for Real-Time Fire Detection and Emergency Exit Monitoring

Dao Vu Anh[1,*], *Dung* Nguyen Huy[3,**], *Duong* Nguyen Danh[2,***], *Duc Anh* Vu[2,****], *Duan* Luong Cong[2,†], and *Bien* Nguyen Quang[4,‡]

[1]Faculty of Electronics Engineering 1 & SEMIT Lab,
Posts and Telecommunications Institute of Technology, Vietnam

[2]Faculty of Electronics Engineering 1 & EDA Lab,
Posts and Telecommunications Institute of Technology, Vietnam

[3]VNU International School, Vietnam National University, Vietnam
[‡]**Correspondence:** e-mail: biennq@ptit.edu.vn

Abstract. Industrial fire incidents pose serious risks to human safety and cause significant economic losses. In contrast, traditional fire detection systems often suffer from high latency, false alarms, and limited monitoring of emergency exit doors. To address these issues, this paper proposes a low-cost Edge AI system on an ESP32 microcontroller for real-time fire detection and emergency exit door state recognition without modifying existing infrastructure. The system integrates real-time image processing and environmental sensing using a lightweight MobileNetV1 model optimized for edge deployment, with all inference performed locally to ensure low latency and autonomous operation. Reliability is improved through temporal decision stabilization and multi-sensor consistency checking based on temperature and smoke data. Experimental results show 97.32% accuracy with 128 ms latency, while the fire class achieves 100% detection accuracy with no misclassification, demonstrating strong suitability for real-time industrial applications.

1 Introduction

Fires in high-risk industrial environments, such as manufacturing plants and chemical facilities, pose significant threats to both human safety and operational continuity [1]. Although less frequent than residential fires, they typically involve higher temperatures and result in disproportionately severe economic losses [2]. For instance, industrial facilities in the United States report tens of thousands of fire incidents annually, causing approximately $1.2 billion in property damage [2]. Similarly, non-residential fires in South Korea accounted for 37.2% of total incidents in 2022, highlighting their substantial impact [3]. Given the rapid escalation of such fires, timely detection is essential to minimize damage and ensure occupant safety [4].

However, conventional fire detection systems often suffer from delayed response times because they rely on smoke propagation or manual monitoring [5]. These delays allow fires to spread rapidly, significantly increasing the risk of catastrophic consequences [6]. This issue becomes even more critical during emergency evacuation, where dense smoke reduces visibility and induces panic, leading to inefficient crowd movement. More importantly, obstructed or locked emergency exits are a major cause of fire-related fatalities. In practice, exits are often blocked by equipment or poorly maintained, preventing effective evacuation and substantially increasing the likelihood of casualties [3].

Despite their importance, existing fire detection systems exhibit several critical limitations. Most solutions rely on simple physical sensors, such as smoke, temperature, or gas detectors, without incorporating artificial intelligence (AI), resulting in high false alarm rates and limited contextual awareness [7]. Furthermore, many systems rely on centralized or cloud-based processing, which introduces latency and limits real-time responsiveness [8, 9]. In addition, high deployment costs and poor compatibility with legacy infrastructures, such as RS485 or dry-contact interfaces, create significant barriers to the adoption of advanced intelligent fire detection systems in industrial environments [10, 11].

To address these limitations, several machine learning and deep learning approaches have been proposed for early fire detection. Udurume et al. [10] developed a YOLOv5-based system, while Khan introduced an SSD-Inception-V2 framework. Although these methods

[*]e-mail: daova@ptit.edu.vn
[**]e-mail: 23070046@vnu.edu.vn
[***]e-mail: duongnd.B22DT071@stu.ptit.edu.vn
[****]e-mail: anhvd.B22DT027@stu.ptit.edu.vn
[†]e-mail: duanlc@ptit.edu.vn
[‡]e-mail: biennq@ptit.edu.vn

achieve high detection accuracy, they rely on computationally intensive, power-hungry platforms such as Raspberry Pi 4 and NVIDIA Jetson Nano, which limit scalability. Similarly, Sharobiddinov et al. [12] proposed an edge-based system using MobileNetV2, but it still targets relatively high-performance hardware such as Raspberry Pi 5. In contrast, Lestariningati et al. [13] presented a low-cost ESP32-based solution; however, it relies on conventional threshold-based sensors rather than AI-driven visual intelligence. More importantly, none of these studies address the integrated problem of simultaneous fire detection and emergency exit monitoring using AI-based visual analysis on a single ultra-low-cost microcontroller platform.

Motivated by this gap, this paper introduces an Edge AI-based system on an ESP32 platform that enables real-time visual fire detection and exit status monitoring with low cost and seamless integration with existing industrial infrastructure without hardware replacement. This approach aligns with the recent growing interest in deploying on-device machine learning on resource-constrained ESP32 microcontrollers to achieve real-time object detection at the edge [9].

The remainder of this paper is organized as follows. Section 2 presents the Methodology, including the edge–cloud architecture, data collection and preprocessing procedures, the design and implementation of the MobileNetV1 model on the ESP32 platform, as well as the training configuration and evaluation methodology. Section 3 presents the Results and Discussion, focusing on classification performance, confusion matrix analysis, comparisons with lightweight models, and real-time edge deployment performance. Finally, Section 4 provides further discussion and concludes the paper by summarizing the main contributions and findings of this study.

2 Methodology

2.1 System Architecture

The proposed system adopts a two-layer architecture consisting of the Edge Layer and the Cloud Layer for real-time processing and remote monitoring, as shown in Figure 1. The Edge Layer integrates an ESP32-S3 with an OV5640 camera to capture images and environmental data (temperature and smoke), where images are resized to 96×96 (RGB) and processed using a MobileNetV1 model for fire detection and emergency exit door state recognition. Inference results are transmitted via MQTT over Wi-Fi. To ensure reliable operation, each Edge device is deployed with a task-specific field of view, with door-monitoring modules oriented toward exit regions and fire-detection modules covering broader areas.

The second block, the Decision Fusion and Control Unit, enhances system reliability through temporal and multi-source fusion. Instead of relying on single-frame predictions, the system aggregates consecutive inference results within a short time window using a majority-consensus strategy to suppress transient errors. Furthermore, sensor fusion is explicitly implemented at the post-processing stage, where visual predictions are logically combined with environmental measurements. The alarm condition is defined as:

where P_{fire} is the visual confidence score, $\tau_1 > \tau_2$ are predefined thresholds, T is the measured temperature, and S is a binary variable representing the smoke sensor state (with $S = 1$ indicating the presence of smoke and $S = 0$ otherwise). This formulation allows high-confidence detections to trigger alarms directly, while lower-confidence cases require confirmation from sensor data, thereby improving robustness and interpretability.

The unit also supports industrial interfaces, such as RS485 and dry-contact ports, as well as human-in-the-loop controls for manual validation. The final block, the alarm actuation unit, activates sirens or warning lights upon confirmed alerts. The Edge Layer operates independently of the Cloud Layer, ensuring continuous operation even under network disruptions.

2.2 Data Collection

The dataset used in this study was collected to reflect real-world deployment scenarios in industrial facilities and building environments. To ensure practical applicability, data acquisition was conducted under diverse conditions, including indoor and outdoor settings and variations in lighting, background, and object distance. Data collection was performed using an ESP32-CAM module with an image resolution of $1600 \times 1200 \times 3$, as illustrated in Figure 2. The dataset consists of three classes, including *fire*, *open* and *closed*. A total of 769 images were collected for the fire class, 812 images for the open class, and 842 images for the closed class. The images were captured from multiple viewpoints, under various lighting conditions (natural, artificial), and across different scene backgrounds to improve model generalization.

In the next stage, object annotation was performed using the AI Labeling tool on the Edge Impulse platform [14], ensuring consistent and accurate labeling for all collected samples.

2.3 Data Pre-Processing

In the preprocessing stage, input images with an original resolution of $1600 \times 1200 \times 3$ were resized to $96 \times 96 \times 3$ while preserving the RGB color channels, thereby maintaining spatial structure and color information. Subsequently, pixel values were normalized to the range $[0, 1]$. The processed data was retained as a 3D tensor ($96 \times 96 \times 3$) to ensure compatibility with the convolutional layers of the MobileNetV1 architecture.

This operation reduces the total number of input elements from approximately 5.76 million to 27,648 per image, significantly decreasing computational complexity and memory consumption. Such a reduction is essential for deployment on resource-constrained edge devices, enabling efficient real-time inference while preserving key visual features required for accurate fire detection.

$$\text{Alarm} = (P_{\text{fire}} > \tau_1) \ \vee \ ((P_{\text{fire}} > \tau_2) \ \wedge \ (T > T_{\text{th}} \ \vee \ S = 1)) \tag{1}$$

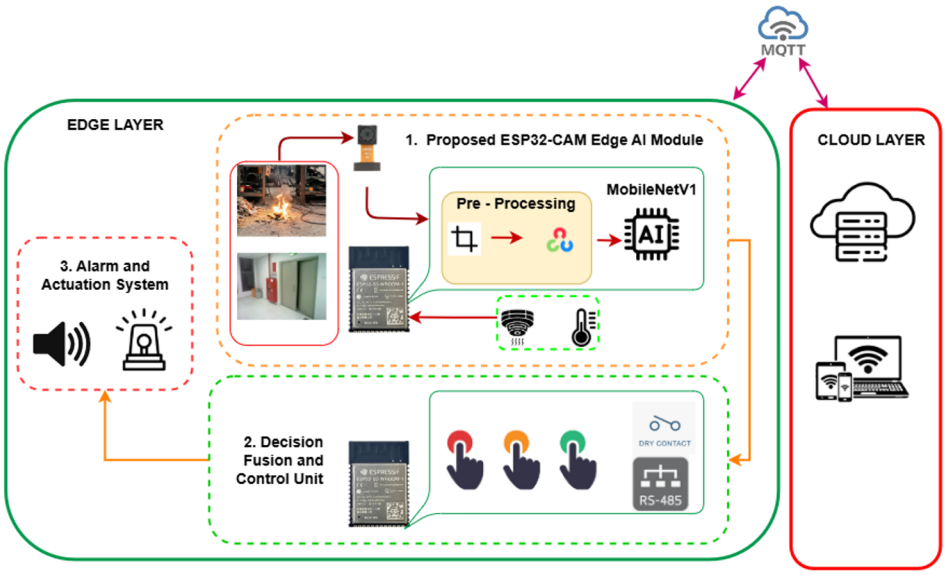

Figure 1: Overall Architecture of the Proposed Distributed Edge AI-Based Fire Detection and Emergency Exit Monitoring System.

Figure 2: Data collection and AI labeling process.

2.4 Machine Learning Analysis

This study employs a MobileNetV1 architecture adapted to accept input images of size 96×96×3. MobileNetV1 is a lightweight convolutional neural network based on Depthwise Separable Convolution, which reduces computational cost and model complexity.

The proposed model consists of three main components. The input layer processes RGB images of size $96 \times 96 \times 3$ (27,648 features). The MobileNetV1 backbone includes an initial convolutional layer followed by 13 Depthwise Separable Convolution blocks, each composed of depthwise and pointwise convolutions, batch normalization, and ReLU activation.

The classification block applies a flatten layer and dropout (0.1), followed by a fully connected softmax layer to classify three categories: fire, closed, and open. The model is trained using an 80% training and 20% testing split, with a portion of the training set further used as a validation set during training to monitor performance and reduce overfitting. Evaluation is performed on both validation and test sets to ensure reliable generalization. Overall, the architecture achieves a balance between accuracy and efficiency, enabling deployment on resource-constrained edge devices.

2.5 Model Deployment and Hardware Implementation

The proposed system is implemented on an ESP32 platform using ESP-IDF, where the MobileNetV1 model is deployed for on-device inference. The model is optimized for embedded execution by applying INT8 post-training quantization and converting it to the TensorFlow Lite Micro format, enabling efficient inference under limited memory and computational resources. During the Post-Training Quantization (PTQ) process, model weights and activations are converted from 32-bit floating-point (float32) to 8-bit integers (int8), significantly reducing computational complexity and memory footprint while maintaining acceptable accuracy for edge deployment.

The ESP32-S3 microcontroller integrates an OV5640 camera for RGB image acquisition, while temperature and smoke sensors provide additional environmental context. Image preprocessing is performed directly on the embedded system using the OpenCV library, primarily to resize and convert input images from the original resolution of 1600×1200 to 96×96, ensuring compatibility with the neural network input requirements [15]. The inference

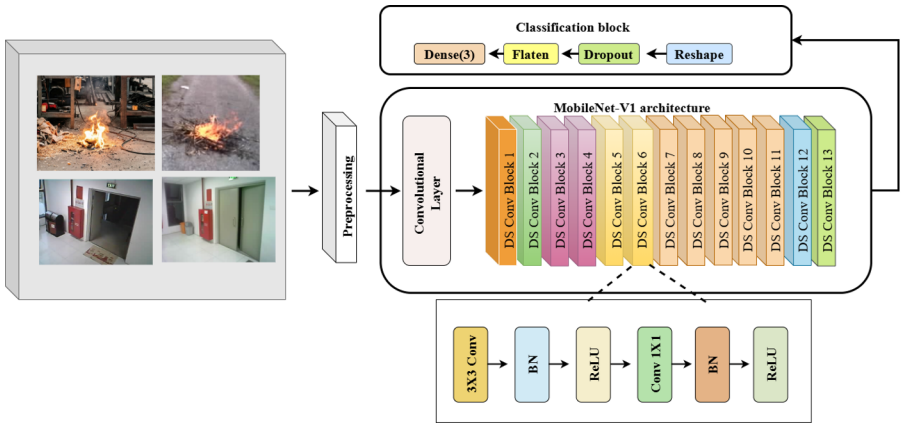

Figure 3: Architecture of the proposed MobileNetV1 model.

pipeline runs entirely at the edge, enabling low-latency processing and minimizing dependency on cloud computing. MQTT is used to transmit detection results to a remote monitoring system for visualization and logging.

3 Results and Discussion

3.1 Classification Result

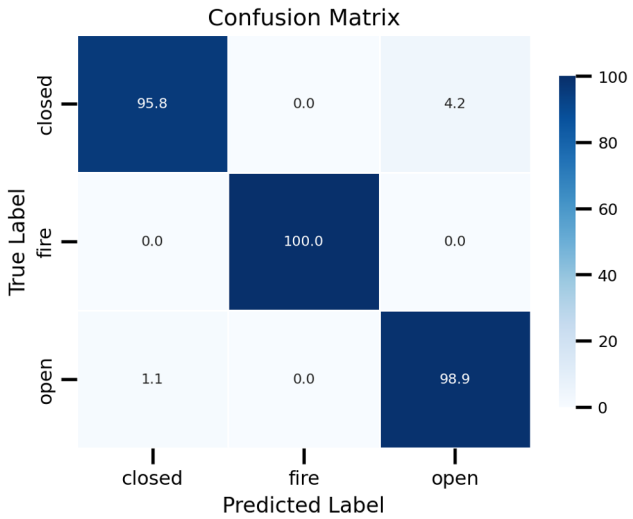

Figure 4: Confusion matrix of classification results.

The classification model demonstrates strong performance on both validation and test datasets, achieving an overall accuracy of 98.1% with a corresponding loss of 0.14. The model achieves high precision, recall, and F1-scores of approximately 0.98 across all classes, while the ROC-AUC score reaches 1.00, indicating excellent separability among the fire, closed, and open classes. The training process is conducted over 50 epochs, which is sufficient to ensure convergence without signs of overfitting, as confirmed by consistent performance between training and validation sets. From an implementation perspective, the model is trained using the Adam optimizer with a learning rate of 0.0005, a batch size of 32, and the categorical cross-entropy loss function. These hyperparameters follow the Edge Impulse training configuration, ensuring full reproducibility and consistency of the experimental setup. This strict definition of the training pipeline is essential for deep learning reproducibility in embedded AI systems [14].

As shown in Figure 4, the confusion matrix provides a detailed breakdown of classification performance. The fire class achieves perfect detection with 100% precision and recall. The closed class reaches 95.8% recall, with 4.2% misclassified as open, while the open class achieves 98.9% recall with 1.1% misclassified as closed. Most misclassifications occur between the closed and open classes due to high visual similarity. Furthermore, the float32 model slightly outperforms the int8-quantized model in terms of ROC-AUC and loss, while both maintain comparable overall accuracy (~98%). On the test set, the float32 model achieves 98.08% accuracy compared to 97.32% for the int8 model. However, the int8 model offers significantly improved computational efficiency, making it more suitable for deployment on resource-constrained edge devices.

3.2 Comparison with Lightweight Models

Table 1 compares MobileNetV1 with other lightweight models in terms of performance and computational efficiency. The deployed quantized MobileNetV1 (INT8) achieves an accuracy of 97.32% maintaining a highly competitive performance compared to EfficientNet (98.3%) while offering significantly lower computational complexity, as reflected in its reduced DSP and classification requirements. This makes MobileNetV1 highly suitable for real-time embedded applications with limited computational resources.

In contrast, EfficientNet, although achieving slightly higher accuracy, requires substantially more computational and processing resources, reducing its suitability for resource-constrained edge devices. MobileNetV2 achieves lower accuracy (96.7%) and higher computational cost than MobileNetV1. Overall, MobileNetV1 provides the best balance between accuracy and computa-

Table 1: Performance Comparison of Lightweight Models

Model	DSP (ms)	Inference (ms)	Accuracy (%)	Recall	F1
MobileNetV1	**5**	**123**	97.3	0.98	0.98
MobileNetV2	21	200	96.7	0.97	0.97
EfficientNet	15	13109	**98.3**	0.98	0.98

tional efficiency, making it the most appropriate choice for low-power embedded fire detection systems.

3.3 Real-Time Model Deployment on Edge

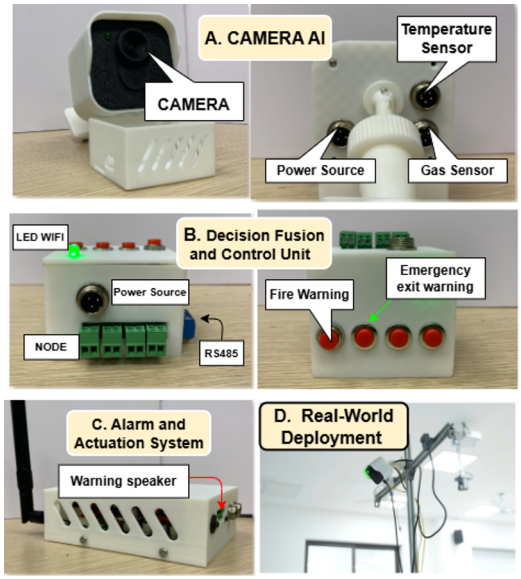

Figure 5: Hardware implementation of the Edge AI system.

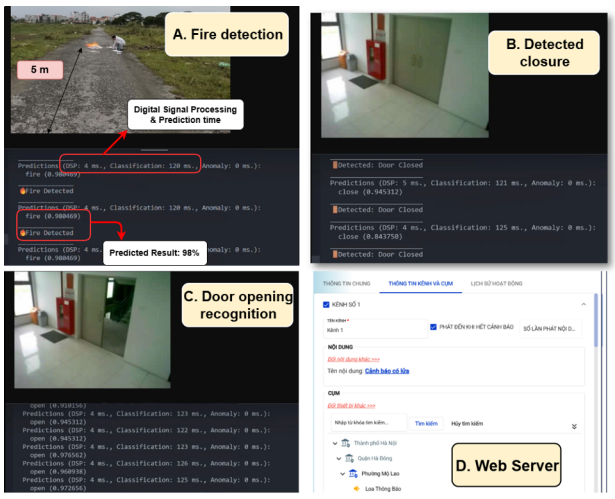

Figure 6: Real-world deployment and web server monitoring interface.

The proposed system is deployed and evaluated in real-world conditions using an ESP32-CAM edge platform. As shown in Figure 5, the hardware integrates an AI camera, gas and temperature sensors, a decision-and-control unit, and an alarm module into a compact embedded device. This design enables on-site data acquisition, processing, and response, with all computations performed locally on the edge device, ensuring autonomous operation without relying on external servers. Figure 6 illustrates the system's performance in practical scenarios. The model operates reliably across tasks such as fire detection (up to 5 meters) or door state recognition (open/closed). Both door-closed and door-open conditions are accurately identified, demonstrating stable performance across different states. The inference results are presented in real time with high confidence levels, achieving up to 98% accuracy in fire detection. The system is also connected to a web-based interface for remote monitoring and alert management.

Table 2: Execution Time and Resource Utilization of the Proposed MobileNetV1 System

Performance Metric	**Value**
Data Preprocessing	5 ms
Model Inference	123 ms
Total Latency	**128 ms**
RAM Usage	27260 (8.3%)
Flash Usage	820997 (12.5%)

As reported in Table 2, the total latency is approximately 128 ms, consisting of 5 ms for data preprocessing and 123 ms for model inference. The proposed model is optimized for embedded deployment, requiring only 8.3% of RAM and 12.5% of Flash memory. These results demonstrate that the system achieves a good balance between computational efficiency and resource utilization, making it suitable for real-time edge AI applications.

3.4 Discussion

The results demonstrate that the proposed Edge AI system can effectively deploy an INT8-quantized MobileNetV1 on a resource-constrained ESP32-S3, achieving 97.32% accuracy with approximately 100 ms latency, while providing a cost-efficient alternative to computation-intensive platforms such as Raspberry Pi [10, 12]. The two-level decision-fusion mechanism plays a key role in improving system reliability by combining high-confidence visual predictions with sensor-based validation in uncertain cases, thereby reducing false positives compared to approaches relying solely on visual AI or threshold-based sensing [10, 12, 13]. In addition, integrating fire detection

and emergency exit monitoring enhances system functionality and practical applicability.

However, several limitations remain. The current model is not optimized for simultaneous multi-condition recognition, such as detecting fire while assessing door states within the same scene, and its detection range is still limited in larger environments. In addition, the system has been evaluated on a relatively small dataset under controlled conditions, and its generalization to diverse real-world scenarios has not been fully validated.

Future work will focus on developing a multi-task learning framework for simultaneous detection of multiple objects and states. Expanding the dataset to include more challenging conditions, such as long-distance, low-light, and dense-smoke scenarios, is expected to improve robustness. Hardware improvements, including higher-resolution imaging and enhanced preprocessing, will also be explored alongside large-scale deployment and long-term evaluation.

4 Conclusions

This paper presents an edge AI-based fire detection and emergency exit monitoring system implemented on a low-cost ESP32 platform, integrating real-time image processing, environmental sensing, and a lightweight MobileNetV1 model for reliable operation under resource-constrained conditions. Experimental results show that the system achieves 97.32% accuracy with a total latency of 128 ms, confirming its suitability for real-time embedded deployment. The confusion matrix further highlights strong class separability, with the fire class achieving 100% recall and no misclassifications. In contrast, the closed and open classes achieve F1 scores of 0.98 and 0.96, respectively, with most errors occurring only between visually similar states. Compared to conventional approaches, the proposed solution offers a compact, fully edge-deployable system that relies on no cloud computing, making it suitable for practical industrial applications. Future work will focus on expanding dataset diversity and validating the system in large-scale real-world environments to improve robustness under challenging conditions and long-term operation.

References

[1] Prabowo, A. R., Tuswan, T., Ridwan, R.: Advanced Development of Sensors' Roles in Maritime-Based Industry and Research: From Field Monitoring to High-Risk Phenomenon Measurement. Appl. Sci. **11**(9) (2021), 3954. https://doi.org/10.3390/app11093954

[2] NFPA: Fire in Industrial or Manufacturing Properties. (2026). https://www.nfpa.org/education-and-research/research/nfpa-research/fire-statistical-reports/fires-in-us-industrial-or-manufacturing-properties

[3] Park, S., et al.: Smart Fire Safety Management System (SFSMS) Connected with Energy Management for Sustainable Service in Smart Building Infrastructures. Buildings **13**(12) (2023), 3018. https://doi.org/10.3390/buildings13123018

[4] Talaat, F. M., ZainEldin, H.: An improved fire detection approach based on YOLO-v8 for smart cities. Neural Comput. Appl. **35**(28) (2023), 20939–20954. https://doi.org/10.1007/s00521-023-08809-1

[5] Ramos, L., Casas, E., Bendek, E., Romero, C., Rivas, F.: Computer vision for wildfire detection: a critical brief review. Multimed. Tools Appl. **83** (2024), 83427–83470. https://doi.org/10.1007/s11042-024-18685-z

[6] Muhendra, R., Amin, A.: Real-Time Monitoring: Development of Low Power Fire Detection System for Dense Residential Housing Based on Internet of Things (IoT) and Cloud Messenger. Sci. J. Inform. **8**(2) (2021), 202–212. https://doi.org/10.15294/sji.v8i2.30811

[7] Wang, X., et al.: Evaluation of gas and particle sensors for detecting spacecraft-relevant fire emissions. Fire Saf. J. **113** (2020), 102977. https://doi.org/10.1016/j.firesaf.2020.102977

[8] Pan, J., McElhannon, J.: Future Edge Cloud and Edge Computing for Internet of Things Applications. IEEE Internet Things J. **5**(1) (2018), 439–449. https://doi.org/10.1109/JIOT.2017.2767608

[9] Chang, Y.-H., Wu, F.-C., Lin, H.-W.: Design and Implementation of ESP32-Based Edge Computing for Object Detection. Sensors **25**(6) (2025), 1656. https://doi.org/10.3390/s25061656

[10] Udurume, M., Hwang, T., Uddin, R., Aziz, T., Koo, I.: Developing a Fire Monitoring System Based on MQTT, ESP-NOW, and a REM in Industrial Environments. Appl. Sci. **15**(2) (2025), 500. https://doi.org/10.3390/app15020500

[11] Shees, A., Ansari, M. S., Varshney, A., Asghar, M. N., Kanwal, N.: FireNet-v2: Improved Lightweight Fire Detection Model for Real-Time IoT Applications. Procedia Comput. Sci. **218** (2023), 2233–2242. https://doi.org/10.1016/j.procs.2023.01.199

[12] Sharobiddinov, D., Siddiqui, H. U. R., Saleem, A. A., Mezquita, G. M., Vargas, D. L. R., Díez, I. D. L. T.: Edge-Based Autonomous Fire and Smoke Detection Using MobileNetV2. Sensors **25**(20) (2025), 6419. https://doi.org/10.3390/s25206419

[13] Lestariningati, S. I., Kamila, S. T., Agusdian, A.: Development of a Low-Cost Early Fire Detection System with IoT and Telegram Integration. In: Proceedings of the 8th International Conference on Informatics, Engineering, Science & Technology (INCITEST 2025). Advances in Engineering Research, vol. 287. Atlantis Press (2025), 146–158. https://doi.org/10.2991/978-94-6463-924-7_13

[14] Hymel, S., Banbury, C., Situnayake, D., Elium, A., Ward, C., Kelcey, M., Baaijens, M., Majchrzycki, M., Plunkett, J., Tischler, D., Grande, A., Moreau, L., Maslov, D., Beavis, A., Jongboom, J., Reddi, V.J.: Edge Impulse: An MLOps Platform for Tiny Machine Learning. arXiv (2022). https://doi.org/10.48550/arXiv.2212.03332

[15] Espressif Systems: ESP OpenCV Component: OpenCV Library for ESP-IDF. GitHub Repository (2024). https://github.com/espressif/esp-opencv-component