International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

DEVELOPMENT OF AN INTELLIGENT EDGE AI SURVEILLANCE SYSTEM FOR INDUSTRIAL SAFETY

Deepika Koyyada, M UshaRani
Department of Electronics and Communication Engineering
Jayamukhi Institute of Technological Sciences, Narsampet, Warangal, Telangana, India

Abstract
Industrial facilities require continuous monitoring to identify unauthorized access, unsafe human presence, and abnormal events in areas containing machinery, hazardous materials, and restricted operations. Conventional CCTV systems primarily provide video feeds and depend on human operators for interpretation and response. This paper presents an intelligent industrial surveillance framework integrating an ESP32-CAM visual node, edge artificial intelligence, an IoT-X communication layer, cloud-based event management, and autonomous alert mechanisms. The ESP32-CAM with an OV2640 camera captures image frames, while image preprocessing prepares the visual input for lightweight inference models such as MobileNet, Tiny-YOLO, or TensorFlow Lite-based models. The intended detection classes described in the project include human presence, unauthorized access, abnormal activity, and safety-related events such as missing helmet or safety vest, fire, smoke, restricted-zone entry, and falls. Detection outputs are intended to activate local indicators and alarms and to transmit event metadata through MQTT or HTTP/HTTPS for dashboard visualization and event logging. The hardware design includes red and green LEDs, a buzzer, a relay, and a regulated 5 V supply. The firmware demonstrates camera initialization, Wi-Fi connection, frame capture, GPIO-based alarm control, and a placeholder event-alert routine. Importantly, the included safety-detection function currently returns a constant normal result and contains a comment instructing replacement with TensorFlow Lite Micro or Edge Impulse inference; therefore, quantitative detection performance and fully operational on-device AI. The work is presented as an embedded AIoT system architecture and prototype framework, with implementation validation identified as a necessary next step.
Keywords: Industrial safety; edge AI; ESP32-CAM; OV2640; intelligent surveillance; object detection; IoT-X; MQTT; autonomous alert; embedded vision.

1. Introduction

Industrial environments such as manufacturing plants, warehouses, refineries, and power stations require reliable surveillance to protect personnel, equipment, and restricted operational zones. Heavy machinery, hazardous materials, and continuous production activity can create situations in which delayed detection of unsafe events increases operational risk. Conventional CCTV systems provide visual coverage but often rely on continuous human observation. In large facilities with multiple camera feeds, manual monitoring can be time-consuming and vulnerable to operator fatigue or missed events.

The convergence of the Internet of Things (IoT), artificial intelligence (AI), and edge computing provides an opportunity to develop surveillance systems that can interpret visual information and initiate responses automatically. IoT enables communication between embedded devices, cloud services, and monitoring applications, while AI-based computer vision can detect objects and selected events in camera frames. Edge processing can reduce the need to transmit every raw frame to a remote server, potentially reducing bandwidth consumption and response delay.

The proposed platform combines visual acquisition, lightweight AI inference, event decision logic, wireless communication, cloud analytics, and local alert outputs. The central objective is to move beyond passive video capture toward event-oriented monitoring and automated notification.

The project identifies six limitations in conventional industrial surveillance: dependence on manual monitoring, delayed response, lack of intelligent interpretation, high operational cost, limited automation, and difficulty scaling across large industrial areas. The stated objectives are to capture real-time images/video using ESP32-CAM; implement AI-based object detection and activity recognition for human presence, unauthorized access, and abnormal activity; transmit information using IoT protocols; generate real-time alerts; and support autonomous decision-making and actions.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

2. Related Work

IoT-based monitoring architectures provide network connectivity and remote access to distributed sensing devices. Gubbi et al. discussed the broader vision and architecture of IoT systems [2], while Islam et al. reviewed IoT applications in healthcare and monitoring contexts [1]. These works provide general foundations for connected monitoring but do not, by themselves, validate the particular industrial safety implementation considered here.

Real-time object detection has been advanced by deep-learning architectures such as YOLO, which predicts object classes and locations in a single network evaluation [3]. MobileNet uses depthwise separable convolutions to reduce computational cost and is designed for mobile and embedded applications [4]. These lightweight architectures are relevant to resource-constrained visual monitoring, although their actual deployment on a specific microcontroller requires model-size, memory, operator-support, and inference-time validation.

Edge computing places processing closer to data generation and can reduce communication latency and dependence on centralized cloud processing [5]. AIoT combines intelligent data interpretation with connected devices to support automated decision-making [6]. The project positions its contribution at the intersection of ESP32-CAM-based visual monitoring, lightweight edge inference, IoT-X connectivity, and autonomous alerts.

The project addresses the integration gap between camera-based monitoring, embedded AI, remote event communication, and local response mechanisms. Its system design combines an ESP32-CAM/OV2640 camera node, an edge inference stage, MQTT/HTTP(S) communication, cloud event storage and dashboard functions, and GPIO-controlled alarm devices. The work should be understood as an integrated system framework rather than a new object-detection algorithm.

3. Proposed System Architecture

The proposed architecture comprises an industrial camera node, image preprocessing, edge AI inference, event evaluation, an IoT-X communication layer, cloud storage and analytics, and local/remote alert mechanisms. The Figure1 presents the camera-to-cloud sequence, while Figure 2 illustrates an ESP32-CAM communicating through broker nodes to a cloud/server-side laptop environment containing Python, OpenCV, and YOLOv8 blocks.

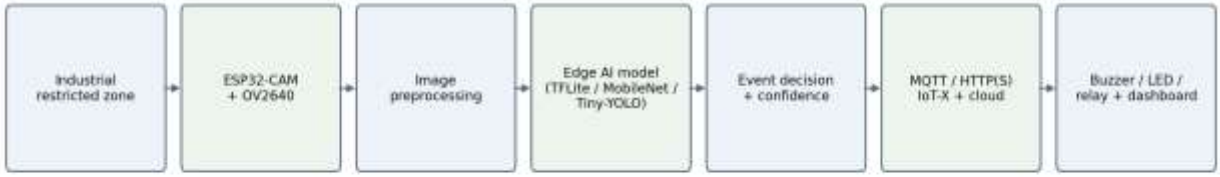


Fig. 1. Functional architecture synthesized from the report's system design.

3.1 Image acquisition node

The ESP32-CAM is the principal embedded camera and Wi-Fi module. The project specifies the OV2640 CMOS camera sensor, with image resolution up to 2 MP and JPEG compression. The camera is configured in the annexed firmware with QVGA frame size, JPEG pixel format, a 20 MHz external clock, JPEG quality setting 12, and a single frame buffer.

3.2 Edge AI processing

The TensorFlow Lite, TensorFlow Lite Micro, TinyML, MobileNet, and Tiny-YOLO as possible lightweight inference technologies. The intended visual categories include person, machinery/equipment, unauthorized intruder, no helmet, no safety vest, fire, smoke, restricted-zone entry, and fall. The report also describes preprocessing operations such as resizing, normalization, brightness/contrast adjustment, and noise reduction. These steps are intended to improve consistency between camera frames and the input expected by the selected model.

The system narrative states that inference is performed locally to reduce latency and network traffic. However, the annexed code does not contain the model loading, tensor allocation, preprocessing-to-model interface, inference invocation, bounding-box decoding, or class-confidence thresholding. Instead, the function detectSafetyViolation() sets detectionConfidence to 0.0, sets detectedClass to NORMAL, and returns false. Thus, the code presently demonstrates the camera and response framework but not a functioning AI detector.

3.3 Event communication and cloud services

The IoT-X layer is described as the communication backbone between camera nodes, edge devices, cloud services, and user applications. MQTT is proposed for lightweight publish-subscribe event messaging, while HTTP is described for request-response transfer and REST-style integration. HTTPS is proposed for encrypted communication using TLS and server authentication. The cloud functions described in the report include long-term video archiving, event logging, aggregate analytics, model update distribution, and centralized device-health monitoring.

3.4 Alert and actuation subsystem

The hardware design includes a red LED, green LED, buzzer, relay, and regulated 5 V supply. In the firmware, GPIO 13 is assigned to the red LED, GPIO 12 to the green LED, GPIO 14 to the buzzer, and GPIO 2 to the relay. The initializeGPIO() routine sets the green LED on and the other outputs off. activateAlarm() turns on the red LED, buzzer, and relay while turning off the green LED. deactivateAlarm() restores the normal indication.

4. System Operation and Methodology

The operational workflow begins with camera initialization and Wi-Fi connection. Frames are captured using esp_camera_fb_get(), processed by the intended AI inference function, and then returned to the camera frame buffer. If a safety violation is identified, the controller activates the alarm outputs and invokes the alert routine; otherwise, the system returns to the normal indication. The firmware sets a nominal 100 ms delay in the main loop, but this delay is not a measured frame rate or end-to-end detection latency.

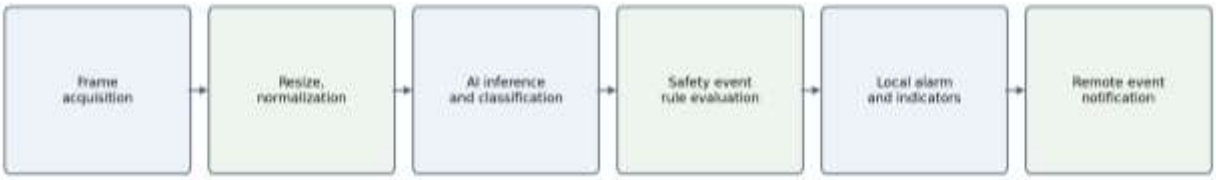


Fig. 2. Intended visual event detection and response pipeline.

4.1 Four operating modes

The paper defines four operating modes. Normal Mode performs continuous monitoring without generating alerts. Detection Mode evaluates a candidate event and its confidence score. Alert Mode sends notifications and activates the buzzer when an abnormal or unauthorized event is confirmed. Autonomous Mode executes predefined actions such as alarms, event logging, dashboard updates, notifications, or connected-device control.

4.2 Hardware and software configuration

Component / platform	Role described in the project
ESP32-CAM (AI-Thinker)	Embedded controller, camera interface and Wi-Fi connectivity
OV2640 camera	Image/video capture; up to 2 MP; JPEG output
Red LED (GPIO 13)	Safety-violation indicator
Green LED (GPIO 12)	Normal operating-status indicator
Buzzer (GPIO 14)	Audible local alert
Relay (GPIO 2)	Switching output for connected response device
Regulated 5 V DC	Power supply for the embedded platform
TensorFlow Lite Micro / TinyML	Proposed embedded inference runtime
MobileNet / Tiny-YOLO	Candidate lightweight vision models
MQTT / HTTP / HTTPS	Proposed event and cloud communication protocols
Python / OpenCV / YOLOv8	Server-side blocks shown in architecture diagram

4.3 Firmware sequence

- Initialize serial communication and GPIO outputs.
- Initialize the OV2640 camera using the ESP32-CAM configuration.
- Connect the ESP32-CAM to the configured Wi-Fi network.
- Capture a camera frame and check whether the frame buffer is valid.
- Invoke the safety-detection function on the captured frame.
- Release the frame buffer after processing.
- If a violation is detected, activate the red LED, buzzer and relay, and call the alert routine; otherwise restore the normal LED state.
- Repeat the acquisition and decision loop.

5. Experimental Implementation and Results

The dashboard is specified to provide live camera feeds, event logs and alert history, device connectivity status, analytics, and visual summaries.

Functionality	Evidence in annexed code
Camera initialization	esp_camera_init() with OV2640/AI-Thinker pin mapping
Wi-Fi connection	WiFi.begin() and connection loop
Frame acquisition	esp_camera_fb_get() and frame-buffer release
AI event detection	detectSafetyViolation() returns false and assigns NORMAL
Local alarm outputs	GPIO control for red/green LEDs, buzzer and relay
Remote alert	Commented MQTT/HTTP/Telegram/cloud integration; serial message only

A key implementation distinction is necessary. The edge AI detection and IoT notifications as system capabilities, the firmware explicitly labels the inference block as a section to be replaced with TensorFlow Lite Micro or Edge Impulse code. Similarly, sendSafetyAlert() contains a comment that MQTT/HTTP/Telegram/cloud notification can be implemented there; it currently prints a serial message. Consequently, the supplied code should not be represented as evidence of completed on-device AI inference or successful remote notification.

- Integrate a trained and quantized model compatible with the available ESP32-CAM memory and inference runtime.
- Implement frame preprocessing, tensor input/output handling, model invocation, confidence thresholding, and object/event decoding.
- Define a labelled industrial safety dataset and document data collection, train/validation/test splits, class definitions, and annotation procedure.
- Measure class-wise precision, recall, F1-score, confusion matrix, false-alarm rate, and missed-event rate.
- Measure actual inference time, end-to-end alert latency, frame rate, memory use, and power consumption on the target board.
- Implement and test MQTT or HTTPS transmission, broker/server configuration, device authentication, and dashboard event logging.
- Test under different lighting, camera distances, occlusion, motion, and network-connectivity conditions.

The proposed architecture is relevant to industrial surveillance because it combines visual event sensing with both local and remote response mechanisms. The use of an ESP32-CAM and OV2640 provides a compact camera node with integrated wireless connectivity. Local GPIO outputs can provide immediate audible and visual indication,

while the IoT-X layer and cloud dashboard are intended to support centralized supervision and historical event review. The design also identifies constraints that are important for deployment: limited processing power and memory on the ESP32, constraints on AI model size, and dependence on stable network connectivity for cloud services.

6. Conclusion and Future Scope

The IoT-X Powered Intelligent Industrial Surveillance and Autonomy with AI-ESP Camera project presents an intelligent, cost-effective, and scalable surveillance solution for modern industrial environments. By integrating the ESP32-CAM, Edge AI, IoT-X communication, and cloud computing, the system is capable of continuously monitoring industrial premises, detecting human presence, unauthorized access, object movement, and abnormal activities in real time. The deployment of lightweight AI models on the ESP32-CAM enables fast on-device inference, reducing latency, bandwidth usage, and dependence on cloud computing while ensuring prompt event detection and automated response. The integration of MQTT/HTTP communication, cloud-based monitoring, and autonomous alert generation enhances industrial safety, operational efficiency, and security. Overall, the proposed system demonstrates the potential of combining Artificial Intelligence, Internet of Things (IoT), and Edge Computing to develop smart surveillance systems that support Industry 4.0 and intelligent manufacturing environments.

References

[1] S. R. Islam et al., “The Internet of Things for Health Care,” IEEE Access, 2015. Full bibliographic metadata should be verified.
[2] J. Gubbi et al., “Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions,” Future Generation Computer Systems, 2013.
[3] J. Redmon et al., “You Only Look Once: Unified, Real-Time Object Detection,” 2016. Verify final publication details.
[4] A. G. Howard et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017.
[5] W. Shi et al., “Edge Computing: Vision and Challenges,” IEEE Internet of Things Journal, 2016. Verify volume, issue, and pages.
[6] J. Zhou et al., “AIoT: Artificial Intelligence of Things,” 2019. Full bibliographic metadata should be verified.
[7] M. Raza et al., “IoT-Based Smart Surveillance System,” 2019. Full bibliographic metadata should be verified.
[8] Q. Zhang et al., “Edge AI Systems,” 2022. Full bibliographic metadata should be verified.