A Hardware Platform for Smart Video Monitoring Based on ESP32-CAM and Mojo FPGA (Spartan-6) with Event Activation Triggered by a PIR Sensor

Saltanat Adilzhanova
Almaty Technological University, Almaty, Kazakhstan | Al Farabi Kazakh National University, Almaty, Kazakhstan
asaltanat81@gmail.com

Gulshat Amirkhanova
Al Farabi Kazakh National University, Almaty, Kazakhstan
gulshat.aa@gmail.com

Murat Kunelbayev
Al Farabi Kazakh National University, Almaty, Kazakhstan
murat7508@yandex.kz

Aigerim Rakhysh
Al Farabi Kazakh National University, Almaty, Kazakhstan
rakhysh_aigerim3@live.kaznu.kz (corresponding author)

Received: 24 February 2026 | Revised: 7 April 2026 | Accepted: 17 April 2026

 DOI: https://doi.org/10.48084/etasr.18359

ABSTRACT

This article presents an event-based edge video surveillance architecture for Internet of Things (IoT) systems, in which the Passive Infrared (PIR) sensor initiates frame capture, the ESP32-CAM module (OV2640) performs control and network publishing, and the Mojo Field-Programmable Gate Array (FPGA) performs hardware-accelerated preprocessing and detection. Communication between the ESP32-CAM and FPGA is carried out via the Universal Asynchronous Receiver–Transmitter (UART) interface at a rate of 921,600 baud (optionally using INT/READY signals), whereas detection results are transmitted to the cloud via Wi-Fi using Message Queuing Telemetry Transport (MQTT) or Hypertext Transfer Protocol (HTTP). Two modes of operation are experimentally evaluated: ESP32-CAM only and ESP32-CAM + FPGA. In terms of end-to-end latency (from PIR triggering to acknowledgment received), a reduction in latency quantiles is observed. For N = 80 events, the P50 decreases from 620 to 480 ms, the P95 from 980 to 780 ms, and the P99 from 1,300 to 1,050 ms, confirming the performance gains achieved by offloading computation to dedicated hardware. The UART load is quantitatively characterized based on the transmitted data type. For compressed frames, the average packet size is approximately 1,200 B (P95: 1,600 B) at a rate of 25 packets/s, corresponding to approximately 30,000 B/s. For Regions of Interest (ROIs) or compact features, the average packet size is approximately 180 B (P95: 240 B) at 40 packets/s, corresponding to approximately 7,200 B/s. For detection results (bounding boxes, confidence values, and flags), the average packet size is approximately 64 B (P95: 96 B) at 40 packets/s, corresponding to approximately 2,560 B/s. These results demonstrate the advantage of transmitting feature-level data instead of full-frame data in bandwidth-constrained scenarios. Detection performance is evaluated using annotated views (60 windows per view). For human movement at a distance of 2 m (front view), accuracy/recall/F1-score values of 0.93/0.95/0.94 are achieved, whereas at 4 m (side view), the values are 0.87/0.85/0.86. The False Alarm Rate (FAR) is 0.15 and 0.25, respectively. For scenes without target movement, the FAR ranges from 0.03 to 0.17, depending on background conditions (idle scene, background movement, and lighting changes). The results demonstrate that the combination of the PIR sensor, ESP32-CAM, and FPGA provides an effective trade-off between latency, communication overhead,

and detection performance, making it suitable as a minimal yet extensible platform for distributed security systems and industrial event-based monitoring.

Keywords-ESP32; ESP32-CAM; Mojo FPGA; video surveillance; hardware monitoring

I. INTRODUCTION

Intelligent video surveillance and event-based monitoring systems in Internet of Things (IoT) applications should simultaneously provide fast response to events (e.g., traffic), efficient edge computing for image processing, and seamless network integration with cloud services. Solutions based on embedded microcontroller platforms offer advantages in energy consumption and communication efficiency, but their computing resources are often insufficient for advanced video analytics. For this reason, a hybrid approach in which resource-intensive operations are assigned to the Mojo Field-Programmable Gate Array (FPGA), whereas management, configuration, and communication functions remain on the ESP32, is considered promising.

In recent years, edge video surveillance systems have been actively developed in the IoT domain due to the growing demand for autonomous, energy-efficient, and low-cost solutions for security, industrial, and environmental applications. Much research has focused on microcontroller platforms with integrated cameras, especially the ESP32-CAM, which is widely used in distributed monitoring systems due to its low cost, built-in wireless interface, and support for external sensors [1]. Publications on the architecture and performance of ESP32-CAM indicate that the platform can perform image capture, basic processing, and video data transmission under constrained computational and energy resources [2]. However, benchmarking results show that increasing resolution and frame rate significantly increases memory and processor load, leading to higher latency and reduced throughput stability [3]. For this reason, ESP32-CAM is generally considered not as a platform for complex local video analytics but mainly as a capture and control node [4].

A number of studies propose hybrid schemes in which the ESP32 acts as an edge node for primary filtering or event detection, whereas complex computations are offloaded to server or cloud infrastructure [5]. This approach reduces network traffic and facilitates scalability; however, it remains dependent on communication channel quality and does not eliminate local decision latency [6]. Event-based video surveillance architectures are considered one of the main mechanisms for improving energy efficiency in IoT systems.

In practice, Passive Infrared (PIR) sensors are often used to trigger frame or video capture upon motion detection, which reduces power consumption and the amount of processed data [7]. Prototypes of ESP32-CAM-based security with PIR triggering confirm the feasibility of this approach, but in many cases the functionality is limited to capturing footage and sending notifications without in-depth analytics [8]. PIR-based event detection is also used in environmental monitoring and camera-trap applications, where ESP32-CAM operates as a standalone node for visual data acquisition [9]. In such scenarios, autonomy and reliability are the priority, so image processing is often minimal or absent [10].

Alongside microcontroller-based solutions, hardware accelerators for video processing based on FPGAs are actively developed. Review studies highlight that FPGAs provide deterministic low latency, energy efficiency, and high parallelism, making them suitable for real-time computer vision tasks [11]. In applied research, FPGAs are used to implement video stream preprocessing stages, including filtering and Region of Interest (ROI) extraction [12]. Of particular interest are studies on video surveillance and multi-camera systems, which demonstrate that even Spartan-6 FPGA devices can support streaming processing and real-time monitoring tasks [13].

Modern research on FPGA-based computer vision acceleration focuses on hardware implementation of convolutional neural network (CNN)-based object detection, including the You Only Look Once (YOLO) family [14]. Experimental results confirm real-time performance through optimized dataflow and memory organization [15], and emphasize the importance of co-design between algorithms and hardware to balance accuracy and computational efficiency [16].

In recent years, event-based vision has emerged, where processing is triggered by asynchronous sensor-generated events. Survey studies on event-based vision implemented on FPGAs show advantages in latency and power consumption [17]. Although many solutions rely on specialized event cameras, the fundamental principles of event-driven processing can also be applied to systems where events are generated by external sensors such as PIR devices [18].

In the context of edge artificial intelligence and smart city applications, integrated design of edge video systems is increasingly emphasized, considering hardware platform selection, processing architecture, and data security requirements [19]. Some studies highlight vulnerabilities in microcontroller-based solutions and, more importantly, indicate that hardware acceleration can improve system reliability and stability [20]. Applied research also demonstrates the use of ESP32-CAM for urban infrastructure and parking monitoring, confirming demand for low-cost intelligent video systems [21]. However, most of these solutions lack hardware acceleration, and analyses of latency and power consumption remain limited [22].

Overall, the literature indicates that existing work either focuses on microcontroller-based video systems with limited analytics [23] or FPGA-based computer vision acceleration without integration with widely available IoT camera modules and event sensors [24]. Furthermore, comparative studies between ESP32-only systems and ESP32 + FPGA architectures in terms of latency and energy efficiency remain insufficient [25].

This motivates the development of integrated event-driven hardware–software video monitoring platforms combining the PIR sensor, ESP32-CAM, and FPGA in a unified architecture

[26]. The main objective of this work is to design a minimal and scalable video surveillance system in which frame capture is triggered by a PIR sensor using ESP32-CAM (OV2640), control and network communication are handled by the ESP32, hardware-accelerated preprocessing and detection are performed on the FPGA, and results are transmitted to the cloud via Wi-Fi, with ESP32–FPGA communication implemented via Universal Asynchronous Receiver–Transmitter (UART) and optional INT/READY signaling.

II. MATHEMATICAL MODEL

The authors consider the system input as an event stream that triggers processing. These events are generated by the PIR sensor, but the PIR sensor is triggered not only by actual human motion (useful events) but also by environmental disturbances such as thermal currents, heating/cooling effects, air vibrations, sunlight, and nearby operating devices.

For this reason, the total event stream should be modeled as a mixture of two components: true events with intensity λ_T (representing the average number of real motion events per second) and false alarms with λ_F intensity (representing the average number of erroneous PIR activations per second). The total activation intensity represents how often the system is forced to "wake up," capture frames, transmit data, and perform computations. It is also convenient to convert the absolute intensities to fractions.

$$\lambda = \lambda_T + \lambda_F \quad (1)$$

$$\pi_F = \frac{\lambda_F}{\lambda_T+\lambda_F},\ \pi_T = 1 - \pi_F \quad (2)$$

where π_F characterizes the fraction of unnecessary system activations. A higher value of π_F leads to increased processing and communication overhead.

The total response time of the system is defined as the end-to-end latency from PIR triggering to cloud publication. It is modeled as the sum of delays of all processing stages:

$$L = T_{pir} + T_{cap} + T_{prep} + T_{tx} + T_{fpga} + T_{rx} + T_{net} \quad (3)$$

where T_{pir} is the response time of ESP32 to the PIR input (interrupt handling, General-Purpose Input/Output (GPIO) reading, and script execution); T_{cap} is the time required to capture a frame using the OV2640 camera and store it in ESP32-CAM memory; T_{prep} is the time required for data preparation (format selection, compression, feature/ROI extraction, and packetization); T_{tx} is the transmission time from ESP32 to FPGA via UART; T_{fpga} is the time required for hardware preprocessing and detection in FPGA (e.g., grayscale → threshold → edges → solution); T_{rx} is the time required to transmit results from FPGA back to ESP32; T_{net} is the network transmission time, including message creation, Wi-Fi transmission, and publication via Message Queuing Telemetry Transport (MQTT) or Hypertext Transfer Protocol (HTTP) (including acknowledgment if applicable).

Since average latency does not capture rare delay spikes, system performance is additionally characterized using latency quantiles. In particular, P95 and P99 represent the latency values not exceeded in 95% and 99% of events, respectively:

$$L_{p95} = \text{quantile}_{0.95}(L),\ L_{p99} = \text{quantile}_{0.99}(L) \quad (4)$$

These metrics are used in the experimental section to compare different processing modes.

The UART transmission delay depends on payload size, protocol overhead, and effective transmission rate. Two modes are considered: full-frame transmission and feature-based transmission (ROI or compact descriptors). The transfer time is modeled as:

$$T_{tx} = \frac{8(B_{pay}+B_{ovh})}{\eta R_{uart}} \quad (5)$$

where B_{pay} is the payload size, B_{ovh} is the protocol overhead, R_{uart} is the nominal UART rate, and η is the efficiency factor.

The reverse FPGA-to-ESP32 communication is modeled similarly but involves significantly smaller payload sizes.

FPGA processing is implemented as a streaming pipeline, where performance is determined by clock frequency and throughput rather than sequential instruction execution. For a frame containing N_{pix} pixels, the total number of clock cycles is expressed as:

$$C = C_0 + II \cdot N_{pix} \quad (6)$$

where C_0 is the pipeline initialization overhead and II is the initiation interval. The corresponding processing time is:

$$T_{fpga} = \frac{C_0+II\cdot N_{pix}}{f_{clk}} \quad (7)$$

where f_{clk} is the FPGA clock frequency. This formulation shows that processing time increases approximately linearly with the number of pixels and decreases with higher clock frequency. The decision score distributions are modeled as:

$$s \mid H_1 \sim \mathcal{N}(\mu_1, \sigma_1^2),\ s \mid H_0 \sim \mathcal{N}(\mu_0, \sigma_0^2) \quad (8)$$

where $\mu_1$and μ_0 are the average score values for movement and immobility, respectively, and σ_1 and σ_0 represent the corresponding variances (how unstable the score is in frames and scenes). Since $\mu_1 > \mu_0$, the distributions partially overlap due to noise and environmental variability.

The main performance indicators are expressed through a decision threshold τ. The probability of detection is $P_D(\tau)$, i.e., the probability of correct activation under motion ($P(y = 1 \mid H_1)$). The probability of false alarm $P_{FA}(\tau)$ is the probability of incorrect activation under no motion ($P(y = 1 \mid H_0)$). Using the standard normal distribution function $F(\cdot)$, the following expressions are obtained:

$$P_D(\tau) = 1 - F\left(\frac{\tau-\mu_1}{\sigma_1}\right) \quad (9)$$

$$P_{FA}(\tau) = 1 - \mathrm{F}(\frac{\tau-\mu_0}{\sigma_0}) \quad (10)$$

The proposed models are used to interpret experimental results in terms of latency, communication load, and detection quality. In particular, the latency model corresponds to measured P50, P95, and P99 values, the communication model

explains bandwidth differences between transmission modes, and the detection model relates to the observed F1-score and False Alarm Rate (FAR).

III. SYSTEM ARCHITECTURE AND IMPLEMENTATION

The proposed system is a hybrid event-driven video monitoring platform based on the ESP32-CAM microcontroller and an FPGA accelerator. The overall architecture of the system is illustrated in Figure 1.

The ESP32 performs control, data acquisition, and communication tasks, whereas the FPGA is responsible for computationally intensive preprocessing and detection. This separation reduces the computational load on the microcontroller and enables lower processing latency.

The system operates in an event-driven mode. In the idle state, the ESP32 monitors the PIR sensor output. When motion is detected, the ESP32 activates frame acquisition using the OV2640 camera and initiates the processing pipeline.

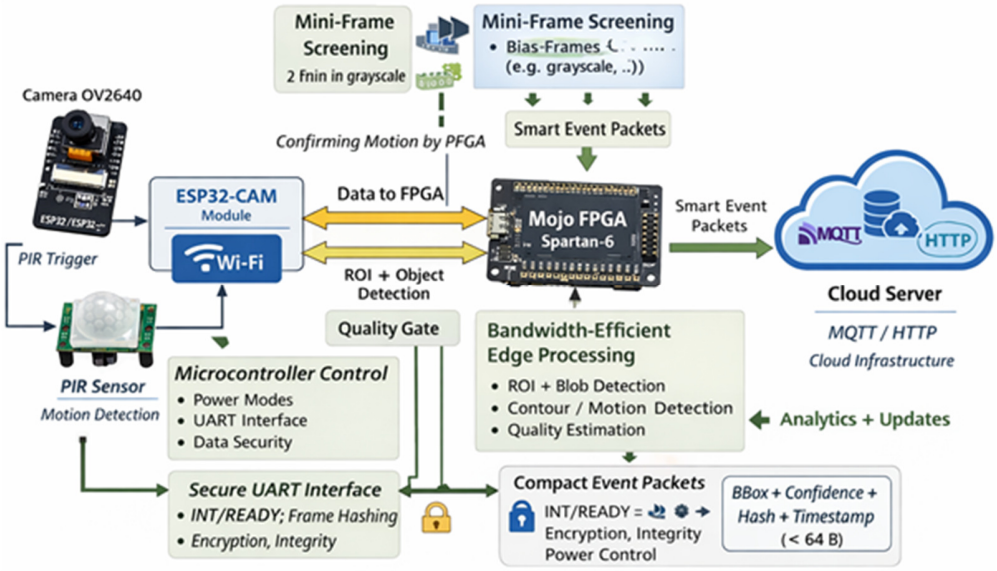

Fig. 1. Event-based video surveillance platform.

The practical implementation of the system is shown in Figure 2, which presents the experimental prototype used for validation. The prototype integrates the ESP32-CAM module, the FPGA board, and peripheral components, demonstrating the feasibility of the proposed architecture under real-world conditions.

The system workflow is as follows:

1. Event monitoring: ESP32 monitors PIR input.
2. Event trigger: when motion is detected, frame capture is initiated.
3. Image acquisition: ESP32-CAM generates a frame and prepares data (compressed frame or features).
4. Transmission to FPGA: ESP32 transmits data via UART; synchronization is performed using INT/READY signals if required.
5. Hardware processing: FPGA performs preprocessing and detection.
6. Result return: FPGA sends processed results to ESP32.
7. Cloud transmission: ESP32 publishes results via Wi-Fi using MQTT or HTTP.

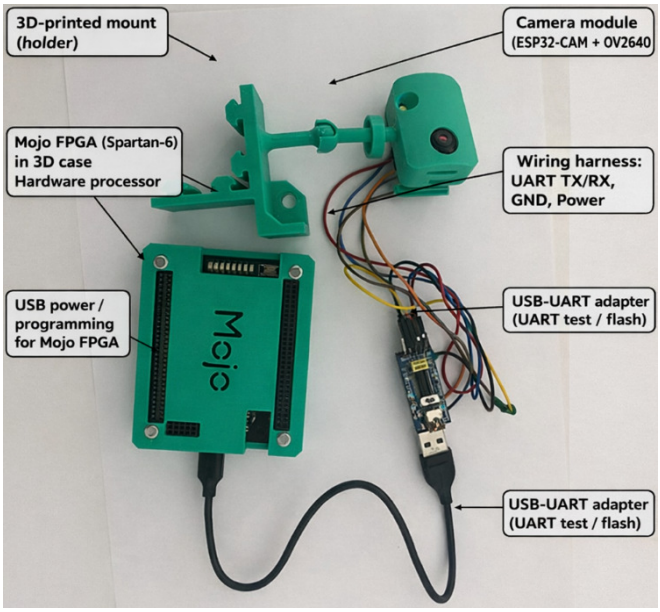

Fig. 2. Assembled prototype of the experimental setup.

The data flow and processing pipeline are illustrated in Figure 3, showing the transformation from sensor-triggered events to compact cloud-transmitted messages.

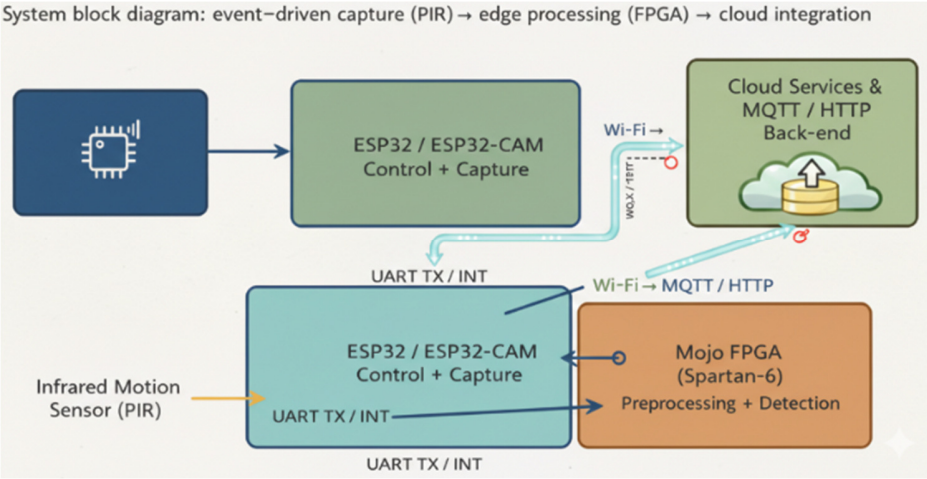

Fig. 3. Event-driven system architecture.

The FPGA processes data in a streaming manner, enabling parallel execution of operations and deterministic latency. This significantly improves system responsiveness compared to ESP32-only processing. In addition, the use of compact feature representations instead of full-frame transmission reduces communication load and improves scalability.

To balance detection accuracy and system efficiency, two transmission modes are used: full-frame transmission and feature-based transmission. The first is used for verification or complex scenarios, whereas the second enables fast and low-overhead processing.

Overall, the proposed architecture provides an effective trade-off between latency, communication overhead, and detection quality, making it suitable for real-time IoT monitoring applications.

IV. RESULTS

The experimental evaluation focuses on three key aspects: end-to-end latency, communication efficiency, and detection performance. The comparison is performed between two processing modes: ESP32-only and ESP32 + FPGA.

The latency results summarized in Table I and illustrated in Figure 4 demonstrate a consistent reduction when processing is offloaded to the FPGA. The median latency (P50) decreases from 620 ms to 480 ms, whereas higher quantiles (P95 and P99) are reduced by approximately 20%. This indicates that FPGA acceleration improves not only typical response time but also worst-case delays, which are critical for real-time security systems.

To validate the effectiveness of the proposed architecture, the ESP32-only implementation is considered as a baseline, whereas the ESP32 + FPGA configuration represents the proposed solution. As shown in Table I and Figure 4, the proposed approach reduces P50 latency from 620 ms to 480 ms, P95 from 980 ms to 780 ms, and P99 from 1,300 ms to 1,050 ms. This corresponds to improvements of 22.6%, 20.4%, and 19.2%, respectively. These results confirm that hardware-assisted preprocessing provides a measurable and consistent advantage over the baseline software-only implementation.

TABLE I. END-TO-END LATENCY SUMMARY (ESP32 VS FPGA)

Mode	P50	P95	P99	N events	Notes
ESP32-only (no FPGA)	620	980	1,300	80	t_0 = PIR GPIO interrupt; t_1 = MQTT/HTTP publish (or ACK). Processing on ESP32.
ESP32 + FPGA	480	780	1,050	80	UART offload to FPGA; optional INT/READY synchronization. Same Wi-Fi conditions.

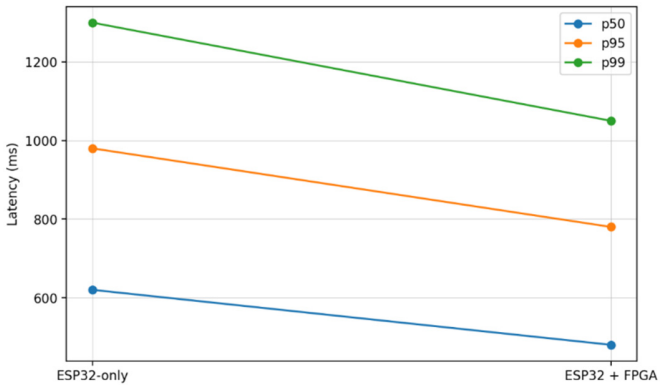

Fig. 4. End-to-end delay quantiles for two processing modes.

Detection quality results are presented in Table II and visualized in Figure 5. For target events (human motion), the system achieves high performance with F1-score = 0.93 (2 m, frontal) and F1-score = 0.85 (4 m, side). The decrease in performance with increased distance and non-frontal angles indicates sensitivity to geometric conditions.

In non-target scenarios, the F1-score remains close to zero, as these scenes are designed to evaluate false alarms rather than detection accuracy. The corresponding FARs are shown in Figure 6, where low values are observed in controlled conditions (0.03–0.05), increasing up to 0.25 in more complex environments.

TABLE II. DETECTION QUALITY BY TEST SCENE

Metric	Empty 2 m frontal	Person 2 m frontal	Background 2 m frontal	Lighting 2 m frontal	Empty 4 m side	Person 4 m side	Background 4 m side	Lighting 4 m side
TP	0	38	0	0	0	34	0	0
FP	2	3	8	5	3	5	10	7
FN	0	2	0	0	0	6	0	0
TN	58	17	52	55	57	15	50	53
Precision	0.00	0.93	0.00	0.00	0.00	0.87	0.00	0.00
Recall	0.00	0.95	0.00	0.00	0.00	0.85	0.00	0.00
F1-score	0.00	0.94	0.00	0.00	0.00	0.86	0.00	0.00
FAR	0.03	0.15	0.13	0.08	0.05	0.25	0.17	0.12
N windows	60	60	60	60	60	60	60	60

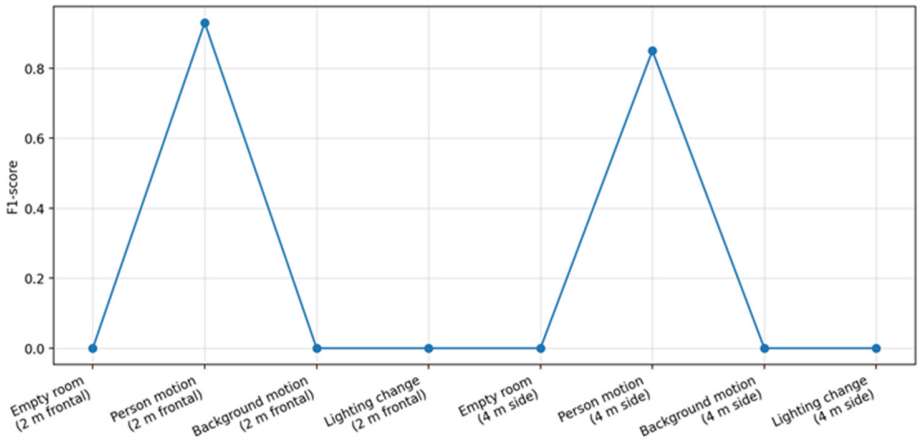

Fig. 5. F1-score for test scenes and geometries.

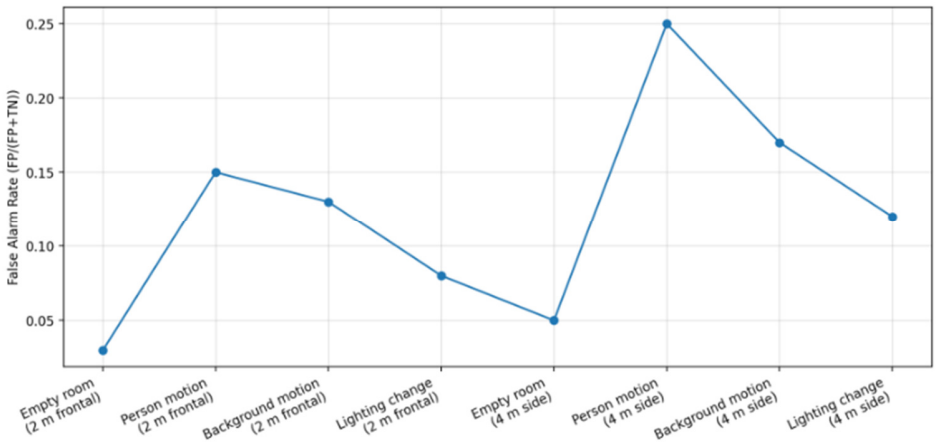

Fig. 6. FAR across test scenarios.

The communication performance is illustrated in Figure 7 (UART throughput in bytes per second) and Figure 8 (UART packet frequency). The results show that transmitting compact features instead of full frames significantly reduces UART bandwidth, from approximately 30,000 B/s to 7,200 B/s (≈76% reduction).

At the same time, packet frequency increases, indicating a shift toward smaller but more frequent transmissions. This trade-off improves overall system efficiency without degrading detection performance.

The combined results from Tables I and II and Figures 4–8 confirm that the proposed architecture achieves an effective balance between latency, communication load, and detection quality. The FPGA-based preprocessing reduces the computational burden on the ESP32 and improves system responsiveness, whereas feature-level transmission minimizes network overhead.

This combination enables scalable deployment in distributed IoT systems, particularly for real-time monitoring and security applications.

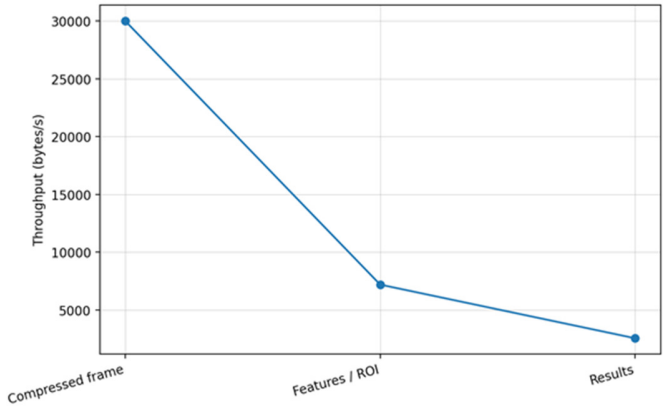

Fig. 7. UART throughput for different payload types.

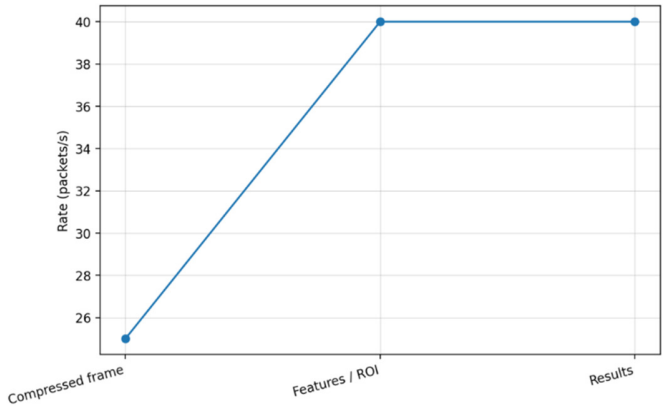

Fig. 8. UART packet frequency.

V. CONCLUSION

This paper presents and validates a minimal yet scalable event-based video surveillance architecture for Internet of Things (IoT) systems, combining ESP32-based control and communication with Field-Programmable Gate Array (FPGA)-based hardware-accelerated processing.

The experimental results demonstrate that the proposed separation of functions significantly improves system performance. In particular, offloading preprocessing and detection to FPGA reduces end-to-end latency by up to 22.6% (P50) and approximately 20% for higher quantiles (P95, P99), whereas feature-based transmission decreases communication load by approximately 76% compared to full-frame transfer.

The evaluation of detection quality confirms reliable performance for target events, achieving F1-scores up to 0.93, while maintaining acceptable False Alarm Rates (FARs) under varying environmental conditions. These results validate the effectiveness of the proposed pipeline (event → frame → hardware processing → result → cloud) for real-time monitoring applications.

The main contribution of this work lies in demonstrating a reproducible architecture that balances latency, communication efficiency, and detection accuracy through hardware–software co-design. Unlike purely software-based approaches, the proposed solution provides measurable improvements in both responsiveness and data efficiency, making it suitable for distributed IoT deployments in safety, industrial, and environmental domains.

Future work will focus on extending the architecture with more advanced hardware modules, improving communication security (e.g., secure boot and firmware signing), and scaling the system to multi-camera scenarios and more complex detection models under strict latency and energy constraints.

DECLARATION OF COMPETING INTERESTS

The authors declare no competing interests.

ACKNOWLEDGMENT

This research has been funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant Number. BR24992975).

DATA AVAILABILITY

The data used in this study were obtained from experimental measurements and controlled test scenarios. Ground truth labels were created manually. The datasets are available from the corresponding author upon reasonable request.

REFERENCES

[1] L. O. M. Ali, A. A. Mochtar, and F. Djamaluddin, "Design and Development of Motion Control for a Metal Waste Cleaning-24 Robot Using ESP32 and PID Control," *Engineering, Technology & Applied Science Research*, vol. 16, no. 1, pp. 31770–31778, Feb. 2026, https://doi.org/10.48084/etasr.15660.

[2] S. T. Nowroz, N. M. Saleh, S. Shakur, S. Banerjee, and F. Amsaad, "A Benchmark Reference for ESP32-CAM Module." arXiv, May 29, 2025, https://doi.org/10.48550/arXiv.2505.24081.

[3] P. R. C. Abordo *et al.*, "Smart surveillance system using ESP32 and camera-based motion detection with IM technology," *International Journal of Research Studies in Educational Technology*, vol. 8, no. 2, pp. 63–74, July 2024, https://doi.org/10.5861/ijrset.2024.8012.

[4] K. Okokpujie, I. P. Okokpujie, F. T. Young, and R. E. Subair, "Development of an Affordable Real-Time IoT-Based Surveillance System Using ESP32 and TWILIO API," *Journal of Safety and Security Engineering*, vol. 13, no. 6, pp. 1069–1075, Dec. 2023, https://doi.org/10.18280/ijsse.130609.

[5] A. Zhaxalikov, A. Mombekov, and Z. Sotsial, "Surveillance Camera Using Wi-Fi Connection," *Procedia Computer Science*, vol. 231, pp. 721–726, Jan. 2024, https://doi.org/10.1016/j.procs.2023.12.147.

[6] F. Hahn, S. Valle, R. Rendón, O. Oyorzabal, and A. Astudillo, "Mango Fruit Fly Trap Detection Using Different Wireless Communications," *Agronomy*, vol. 13, no. 7, June 2023, Art. no. 1736, https://doi.org/10.3390/agronomy13071736.

[7] C. L. Kok, J. B. Heng, Y. Y. Koh, and T. H. Teo, "Energy-, Cost-, and Resource-Efficient IoT Hazard Detection System with Adaptive Monitoring," *Sensors*, vol. 25, no. 6, Mar. 2025, Art. no. 1761, https://doi.org/10.3390/s25061761.

[8] K. Koszewski *et al.*, "Utilizing IoT Sensors and Spatial Data Mining for Analysis of Urban Space Actors' Behavior in University Campus Space Design," *Sensors*, vol. 25, no. 5, Feb. 2025, Art. no. 1393, https://doi.org/10.3390/s25051393.

[9] M. R. Z. Chowdhury, A. Seum, M. R. Talukder, R. A. Amin, F. S. Hossain, and R. Obermaisser, "Towards Next-Generation FPGA-Accelerated Vision-Based Autonomous Driving: A Comprehensive Review," *Signals*, vol. 6, no. 4, Oct. 2025, Art. no. 53, https://doi.org/10.3390/signals6040053.

[10] O. Al-Shamma and M. A. Fadhel, "Trusted outdoor multi-camera tracking system powered by FPGA," *Journal of Engineering Research*,

vol. 13, no. 4, pp. 3092–3106, Dec. 2025, https://doi.org/10.1016/j.jer.2024.10.010.

[11] C. W. Heng, C. Uttraphan, C. C. Choon, and K. B. Ching, "Optimizing FPGA-based YOLO series accelerators: A survey of techniques," *Neurocomputing*, vol. 650, Oct. 2025, Art. no. 130874, https://doi.org/10.1016/j.neucom.2025.130874.

[12] Z. Yan, B. Zhang, and D. Wang, "An FPGA-Based YOLOv5 Accelerator for Real-Time Industrial Vision Applications," *Micromachines*, vol. 15, no. 9, Sept. 2024, Art. no. 1164, https://doi.org/10.3390/mi15091164.

[13] K. Zeng, Q. Ma, J. W. Wu, Z. Chen, T. Shen, and C. Yan, "FPGA-based accelerator for object detection: a comprehensive survey," *The Journal of Supercomputing*, vol. 78, no. 12, pp. 14096–14136, Aug. 2022, https://doi.org/10.1007/s11227-022-04415-5.

[14] S. H. Hozhabr and R. Giorgi, "A Survey on Real-Time Object Detection on FPGAs," *IEEE Access*, vol. 13, pp. 38195–38238, 2025, https://doi.org/10.1109/ACCESS.2025.3544515.

[15] S. M. Sali, M. Meribout, and A. A. Majeed, "Real Time FPGA Based CNNs for Detection, Classification, and Tracking in Autonomous Systems: State of the Art Designs and Optimizations." arXiv, Sept. 04, 2025, https://doi.org/10.48550/arXiv.2509.04153.

[16] T. Kryjak, "Event-Based Vision on FPGAs - a Survey," in *2024 27th Euromicro Conference on Digital System Design*, Paris, France, 2024, pp. 541–550, https://doi.org/10.1109/DSD64264.2024.00078.

[17] G. Gallego *et al.*, "Event-Based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, Jan. 2022, https://doi.org/10.1109/TPAMI.2020.3008413.

[18] K. S. Velaga, Y. Guo, and W. Yu, "Edge AI for Smart Cities: Foundations, Challenges, and Opportunities," *Smart Cities*, vol. 8, no. 6, Dec. 2025, Art. no. 211, https://doi.org/10.3390/smartcities8060211.

[19] A. Trigkas, D. Piromalis, and P. Papageorgas, "Edge Intelligence in Urban Landscapes: Reviewing TinyML Applications for Connected and Sustainable Smart Cities," *Electronics*, vol. 14, no. 14, July 2025, Art. no. 2890, https://doi.org/10.3390/electronics14142890.

[20] S. Adilzhanova, A. Rakhysh, M. Kunelbayev, G. Amirkhanova, and D. Sybanova, "Digital Representations in IoT: Cryptographic Tools for Improved Security," *Journal of Advances in Information Technology*, vol. 17, no. 2, pp. 390–404, 2026, https://doi.org/10.12720/jait.17.2.390-404.

[21] T. Zhukabayeva, L. Zholshiyeva, N. Karabayev, S. Khan, and N. Alnazzawi, "Cybersecurity Solutions for Industrial Internet of Things–Edge Computing Integration: Challenges, Threats, and Future Directions," *Sensors*, vol. 25, no. 1, Jan. 2025, Art. no. 213, https://doi.org/10.3390/s25010213.

[22] P. Lech, B. Marciniak, and K. Okarma, "A Low-Cost Energy-Efficient IoT Camera Trap Network for Remote Forest Surveillance," *Electronics*, vol. 14, no. 21, Oct. 2025, Art. no. 4266, https://doi.org/10.3390/electronics14214266.

[23] Y. Gao, S. Wang, and H. K.-H. So, "REMOT: A Hardware-Software Architecture for Attention-Guided Multi-Object Tracking with Dynamic Vision Sensors on FPGAs," in *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Virtual Event, USA, 2022, pp. 158–168, https://doi.org/10.1145/3490422.3502365.

[24] V. A. Méndez-Lópe, C. Soubervielle-Montalvo, A. S. Núñez-Varela, O. E. Pérez-Cham, and J. E. González-Galván, "A survey on FPGA-based design methodologies for visual object tracking," in *V Congreso Internacional y XIII Congreso Nacional de Ciencias de la Computación*, Puebla, Mexico, 2023, pp. 102–113.

[25] A. O. Elfaki, W. Messoudi, A. Bushnag, S. Abuzneid, and T. Alhmiedat, "A Smart Real-Time Parking Control and Monitoring System," *Sensors*, vol. 23, no. 24, Dec. 2023, Art. no. 9741, https://doi.org/10.3390/s23249741.

[26] R. Al Amin and R. Obermaisser, "Real-Time Object Detection and Classification using YOLO for Edge FPGAs," in *2025 International Symposium ELMAR*, Zadar, Croatia, 2025, pp. 291–295, https://doi.org/10.1109/ELMAR66948.2025.11193980.