EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

An OpenHarmony-Based AI-Driven Edge Intelligence Framework for Manufacturing Disruption Monitoring and Resilient Response

Boqiang Zhang[1], Yuxin Zhuo[1], Zhiwei Chen[1], Zhibin Xian[1], Chengming Huang[1], Jiamin Chen[1], Zonghui Huang[1] and Weiting He[1]*

[1] School of Artificial Intelligence, Neusoft Institute Guangdong, Foshan, China

Abstract

INTRODUCTION: Delayed awareness of hazards and route restrictions can weaken manufacturing continuity.
OBJECTIVES: To develop an OpenHarmony-based edge-intelligence framework for disruption monitoring and resilient patrol response.
METHODS: A WS63 controller performs local sensing and PID control; SparkLink SLE transmits compact frames to a Jetson Nano running a GSConv- and MSAA-enhanced YOLOv10-N detector; an ArkTS/ArkUI application presents warnings and fallback status.
RESULTS: At 3 m line of sight, SLE achieved 4.8 ± 1.2 ms RTT, 67.1% lower than matched Wi-Fi/UDP and 72.1% lower than BLE. The detector reached 83.96% mAP50, 51.74% mAP50:95, 28.3 FPS, and 3.08 FPS/W. In 500 trials, recognition accuracy was 94.0% and mean alert latency was 97.9 ± 22.6 ms.
CONCLUSION: The prototype supports timely warning and continuity-oriented fallback, but evidence is limited to one robot, short-range line-of-sight tests, and traffic-sign-derived proxy events.

Keywords: Manufacturing resilience, disruption monitoring, edge intelligence, industrial patrol, OpenHarmony, SparkLink SLE.

Received on 30 June 2026, accepted on 12 August 2026, published on 07 September 2026

doi: 10.4108/eetsis.13806

*Corresponding author. Email: heweiting95@foxmail.com

1. Introduction

Modern manufacturing environments increasingly depend on interconnected equipment, mobile assets, automated logistics, and continuous data exchange. In such environments, delayed awareness of access restrictions, environmental hazards, abnormal operating conditions, or blocked inspection routes can prolong local interruptions and weaken production continuity. Mobile inspection platforms and edge-deployed visual models can complement fixed industrial sensors by extending monitoring coverage to workshop passages, warehouses, utility areas, and internal logistics routes[1,2]. Their practical value, however, depends on whether potentially disruptive conditions can be detected, communicated, and presented to operators within a bounded time.

International research on resilient manufacturing has moved from disruption recovery toward the combined capabilities of absorption, adaptation, and recovery, with industrial AI, digital connectivity, and reconfigurable operations treated as major enabling mechanisms [3,4]. In parallel, industrial inspection research has advanced from fixed-camera defect inspection to mobile robots and edge-deployed anomaly perception [1,2,5]. Multi-robot patrol and situation-awareness studies emphasize coverage, connectivity, and collective anomaly assessment [6-8],

whereas edge-robotics studies focus on task partitioning, latency, and resource constraints [9,10].

At the platform level, ROS 2 and DDS provide a mature international robotics middleware ecosystem [11-13]. In China, OpenHarmony provides a componentized distributed operating-system framework with HDF, Distributed SoftBus, ArkUI, and cross-device service capabilities [14], while SparkLink/NearLink has emerged as a domestic short-range wireless technology designed for low-latency and high-reliability applications [15,16]. Existing studies and technical reports, however, usually evaluate the operating-system layer, radio link, perception model, or patrol algorithm separately. A cross-layer evaluation that connects local control, nearby inference, warning publication, and conservative fallback remains comparatively underexplored.

Existing studies have investigated robotic patrol, autonomous navigation, edge-assisted perception, communication-aware coordination, and lightweight object detection[6,7,9,10,17,18]. Edge deployment can reduce dependence on remote cloud services, while low-latency wireless communication can improve state freshness and warning responsiveness. Nevertheless, communication latency, perception accuracy, motion control, energy consumption, application publication, and fallback behavior are often evaluated separately. Consequently, it remains unclear whether module-level improvements translate into a responsive end-to-end monitoring workflow under the resource constraints of a mobile industrial platform.

To address this gap, this study develops an OpenHarmony-based AI-driven edge intelligence framework for manufacturing disruption monitoring and resilient response. Low-level motion control and direct sensor acquisition remain local to a HiSilicon WS63 controller, latency-critical state frames are transmitted through SparkLink SLE, visual inference is executed on a Jetson Nano, and structured warning information is presented through an ArkTS/ArkUI application. The device–edge–application partition is designed to prevent network serialization and operator-interface workloads from interfering with local motion control while maintaining timely event publication.

In this study, a disruption indicator refers to a visually or physically observable condition that may require route adjustment, speed reduction, additional inspection, or operator intervention. It does not necessarily represent a confirmed machine failure or a completed production interruption. The traffic-sign-derived visual categories are therefore treated as controlled proxy indicators for restrictions, warnings, and route-related hazards. The evaluation demonstrates disruption awareness and continuity-oriented fallback support rather than general manufacturing-defect diagnosis or autonomous production recovery.

The main contributions are as follows:

An OpenHarmony-based device–edge–application framework integrating mobile sensing, local embedded control, nearby edge inference, and operator-facing disruption awareness.

A split control-plane/data-plane design in which OpenHarmony SoftBus supports discovery and session management, while SparkLink SLE carries compact latency-critical frames between the WS63 controller and the Jetson Nano.

A resource-aware perception and response pipeline combining GSConv, multi-scale attention aggregation, and conservative fallback, together with a multi-level evaluation of communication, control, perception, energy efficiency, northbound publication, and end-to-end warning performance.

2. Manufacturing Resilience Monitoring Framework

2.1 Design Objectives

The proposed framework is designed for mobile monitoring tasks in workshops, warehouses, utility areas, and campus-like industrial environments where delayed observation may prolong a local disruption or reduce operator response time. The device layer performs motion control and direct sensor acquisition, the edge layer executes visual inference and event assessment close to the observation source, and the application layer provides state visualization, route configuration, warning presentation, and operator-intervention support.

The framework is evaluated using measurable engineering objectives. Under the nominal patrol load, the controller should maintain a mean steady-state speed error of approximately 2%. Visual inference should require fewer than 3 million parameters and remain below 50 ms per frame on the Jetson Nano. The device-to-edge communication path should provide a mean round-trip latency below 10 ms and jitter below 3 ms. The application layer should publish robot state at no less than 5 Hz and render warning events within a 200 ms end-to-end threshold. In this framework, resilient response refers to reduced patrol speed, increased observation frequency, preservation of the latest valid state, and operator notification when reliable observations are temporarily unavailable.

2.2 Device–Edge–Application Architecture

Figure 1 presents the device–edge–application architecture of the proposed manufacturing resilience monitoring framework. The device layer is centered on the WS63 controller and integrates chassis actuation, encoder feedback, temperature-humidity sensing, gas sensing, ultrasonic ranging, and infrared sensing. The Jetson Nano edge layer performs lightweight visual inference, auxiliary road-edge extraction, structured-state generation, and local event assessment. The application layer is implemented with ArkTS and ArkUI and provides live monitoring, environmental-state visualization, route configuration, and disruption-warning presentation. Non-real-time Web functions support login, alert archival, patrol-history review,

and parameter synchronization, but they do not participate in latency-critical control.

Functionally, the device layer closes the 100 Hz speed-control loop and converts raw sensor signals into timestamped state frames; the edge layer validates frames, performs camera preprocessing and TensorRT inference, fuses visual and range cues, and generates event records; the application layer renders state, confidence, source time, route status, and recommended action; and the Web service stores history outside the real-time path. The principal architectural innovation is not any individual component but the explicit isolation of local control from visualization and serialization workloads, combined with a dual-plane communication design and a continuity-oriented degraded mode.

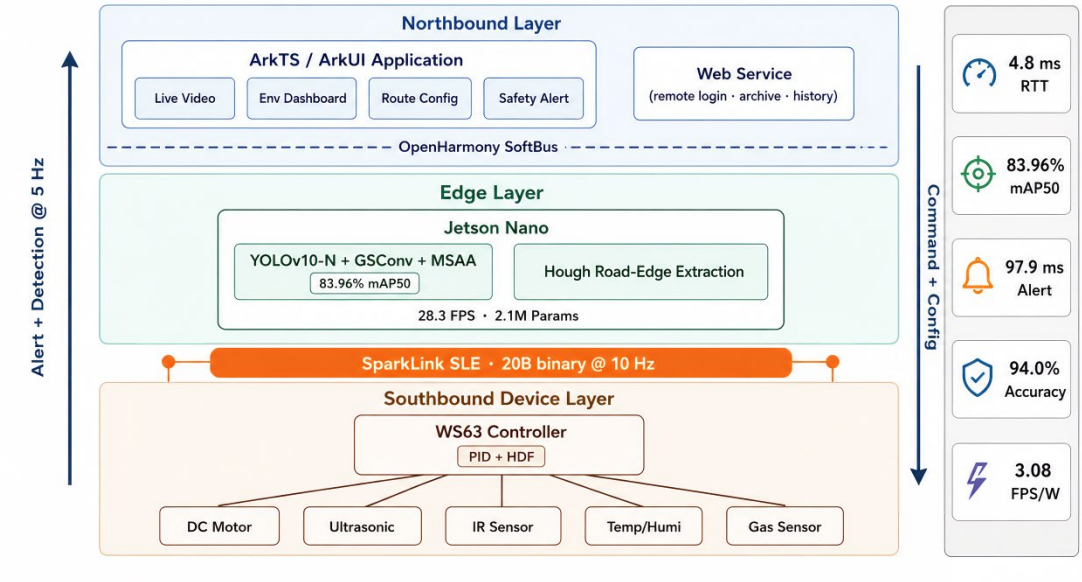

Figure 1. Overall architecture of the proposed OpenHarmony-based manufacturing resilience monitoring framework

The local WS63–Jetson pair forms the primary real-time monitoring and response loop, whereas Web-based reporting and history services remain outside the timing-critical path. This partition prevents application rendering, report generation, and network serialization from competing with local motion control and edge inference. It also limits the scope of the current prototype to a single mobile monitoring platform with nearby edge computation and remote operator supervision.

2.3 Cross-Layer Design Scope

The framework is not intended to demonstrate that OpenHarmony is intrinsically faster or more suitable than Linux or ROS 2 for all robotics or manufacturing applications. ROS 2 commonly employs DDS and provides a mature robotics ecosystem [11-13], whereas the present OpenHarmony implementation offers an integrated path from hardware drivers and device discovery to edge services and application presentation [14]. The contribution of this study lies in the coordinated implementation of these components for a latency-sensitive monitoring workflow.

The experimental comparisons therefore focus on specific implementation choices rather than general platform superiority. These choices include compact binary communication versus the deployed JSON path, lightweight edge inference under fixed hardware resources, event-driven reporting versus periodic polling, and module-level measurements versus end-to-end warning performance.

Four cross-layer design choices distinguish the prototype: (1) the 100 Hz actuation loop is physically and logically local to the WS63; (2) SoftBus is used for discovery and session orchestration, whereas SLE carries the compact timing-critical payload; (3) perception, state publication, and fallback are evaluated as one warning chain rather than as isolated modules; and (4) loss of a fresh observation triggers a bounded degraded state instead of stale-command execution. These choices define the claimed system-level contribution.

3. Device-Layer Control And Low-Latency Communication

3.1 Local Control and Sensing

The device layer uses the HiSilicon WS63/Hi3863-class controller as the primary embedded node. The SoC integrates a 32-bit processor with a maximum clock of 240 MHz, 606 KB SRAM, 300 KB ROM, 4 MB flash, and 2.4 GHz Wi-Fi

6, BLE, and SLE connectivity [19]. The experiments configured the processor at 160 MHz. Available peripheral interfaces include SPI, I2C, UART, GPIO, PWM, and ADC, which support the motor driver, encoders, ultrasonic and infrared modules, temperature-humidity sensor, gas sensor, and status display. OpenHarmony HDF provides unified driver access and configuration. A 12 V lithium battery with DC-DC regulation supplies the controller, sensors, and drive electronics.

Motor actuation is implemented through PWM, encoder edges are captured and accumulated within the 10 ms control period, analog environmental channels are sampled through ADC, serial sensor frames are checked before use, and ultrasonic timing is measured through GPIO edge capture. The reporting task runs independently from the motor-control task and therefore cannot block the control deadline. A watchdog and timestamp check cause the controller to retain the most recent valid state and enter the degraded mode when fresh edge information is unavailable.

The PWM duty cycle is defined by

$$D = \frac{T_{on}}{T} \quad (1)$$

where D is the duty cycle, Ton is the high-level duration, and T is the PWM period. The implementation uses 8-bit duty-cycle resolution and T=2 ms, corresponding to a 500 Hz PWM frequency.To preserve motion stability during patrol, the motor controller uses a discrete PID loop

$$u[k] = K_p e[k] + K_i T_s \sum_{j=0}^{k} e[j] + K_d \frac{e[k]-e[k-1]}{T_s} \quad (2)$$

where u[k] is the control output at sampling step k, e[k] is the speed error, Ts=10 ms is the control sampling period, and Kp=1.2, Ki=0.05, and Kd=0.8 are the proportional, integral, and derivative gains. The gains were tuned empirically through repeated step-response tests on the physical chassis using reference speeds of 0.15, 0.25, and 0.35 m/s, a gross vehicle mass of 2.3 kg, and a regulated battery range of 11.5–12.5 V until the speed error remained within the design target and overshoot was acceptably small. The present controller intentionally uses fixed gains without gain scheduling, friction compensation, or model-based feedforward. It should therefore be interpreted as a lightweight engineering baseline for nominal patrol loads rather than as a fully optimized adaptive control scheme. This discrete formulation is computationally efficient and suitable for a lightweight controller that must respond promptly to speed variation and path deviation.

Figure 2 summarizes the local control, sensing, and device-to-edge communication paths. The 100 Hz motor-control loop remains entirely within the WS63 controller, whereas sensor and state frames are assembled by an independent reporting task and transmitted to the Jetson Nano through SparkLink SLE. This separation prevents communication delay and application-side processing from directly interrupting basic motion regulation.

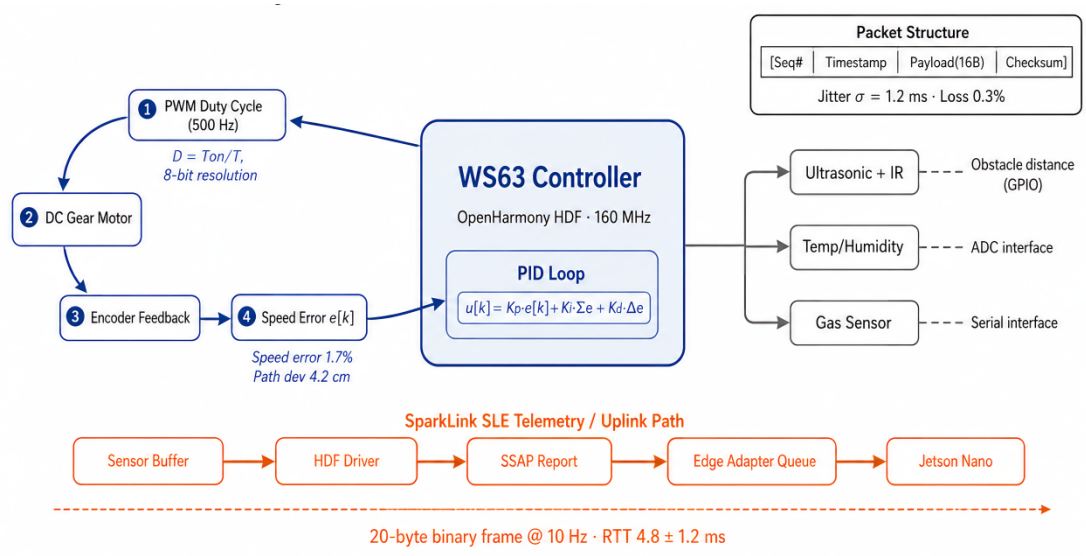

Figure 2. Local control, sensing, and SparkLink SLE communication flow centered on the WS63 controller

3.2 SparkLink Communication Path

The WS63 controller exchanges sensing and command information with the Jetson Nano through SparkLink SLE. SparkLink was designed for short-range applications requiring low latency and high reliability [15], and recent measurement work has reported substantial access-layer improvements over BLE under tested configurations [16]. In the present prototype, the link provides a short-range device-to-edge path for state reporting and command delivery, while the 100 Hz motor-control loop remains local to the controller. This separation limits the influence of wireless delay on basic actuation and allows the controller to maintain a conservative local state when edge connectivity is temporarily unavailable [18].

OpenHarmony SoftBus supports device discovery, capability advertisement, authentication, and session establishment, whereas SparkLink SLE SSAP characteristics carry the latency-critical application frames.Each application frame contains a 16-byte sensor or command payload and 4 bytes of control information, including sequence, timing, length, and checksum-related fields, resulting in a total application-frame size of 20 bytes. After bearer and service overhead, the SLE transmission occupies approximately 27–31 bytes, whereas the deployed Wi-Fi-plus-JSON path exceeds 70 bytes before application parsing. Because these implementations differ in both bearer and serialization, Section 5.2 reports two comparisons: SparkLink SLE, Wi-Fi/UDP, and BLE are first evaluated using the same 20-byte application frame, while the deployed Wi-Fi-plus-JSON path is retained as a full-stack implementation reference.

The latency-critical processing path is HDF driver → WS63 service buffer → SparkLink SSAP report → Jetson edge-adapter queue → distributed service object. Frame length, sequence number, timestamp, and checksum are validated before a message enters the edge queue. Malformed, duplicated, or expired frames are discarded, and the most recent valid state is retained for continuity-oriented fallback. The communication latency, packet-loss, and paired block-level comparison results are evaluated in Section 5.2.

4. Edge-To-Application Response Support

4.1 Monitoring and Disruption-Awareness Interface

The application layer is implemented in ArkTS and ArkUI using DevEco Studio. It provides four functional views: live robot-state monitoring, environmental sensing, route and task configuration, and disruption-warning presentation. The interface displays the current operating state, event category, confidence score, source timestamp, and fallback action in a unified view, allowing the operator to assess whether route adjustment, additional inspection, or manual intervention is required.

Figure 3 presents the edge-to-application monitoring and warning workflow. The live-monitoring page overlays the traffic-sign-derived restriction and warning indicators evaluated in this study, together with the auxiliary road-edge cue. The environmental page displays temperature, humidity, gas-sensor output, and ultrasonic distance at sensor-appropriate update intervals. The route page supports waypoint review and task configuration, while the warning page displays the event type, confidence, source timestamp, route state, and current fallback action. The report service archives warning records and patrol history but does not participate in latency-critical control.

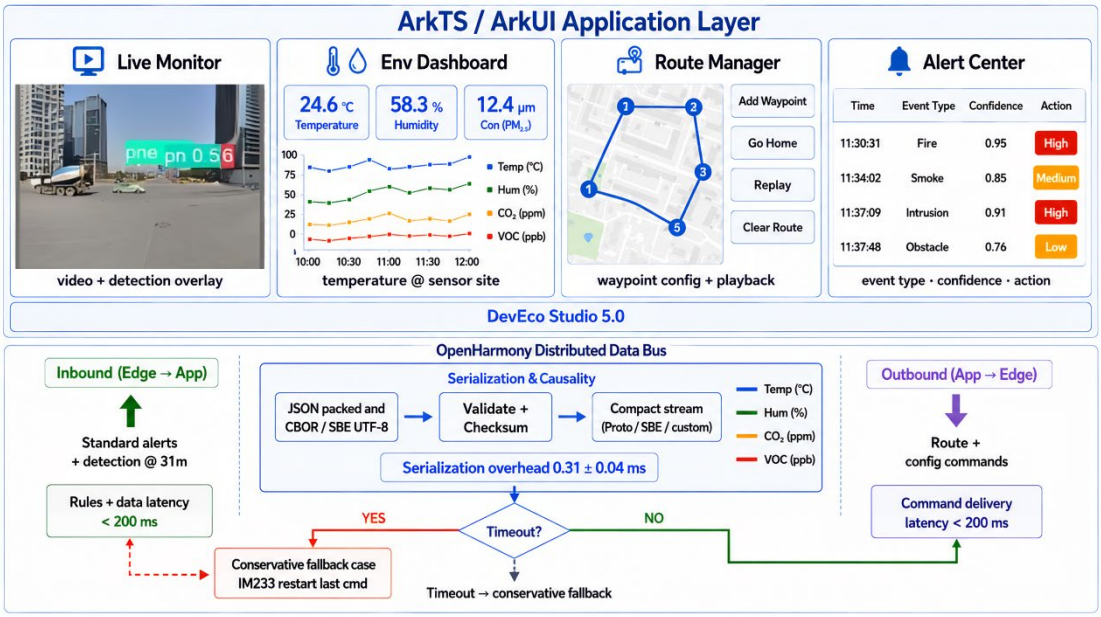

Figure 3. Edge-to-application monitoring and disruption-warning workflow

4.2 Event Publication and Fallback Logic

Patrol and monitoring commands are serialized as validated JSON messages at the application boundary and converted into compact internal command structures before entering the SparkLink SLE path. The WS63 controller executes motion commands locally and reports the resulting state, while the Jetson Nano performs event assessment and publishes structured warning information. This arrangement preserves local control continuity while allowing device

discovery, state synchronization, and operator supervision to be maintained through OpenHarmony distributed services.

A typical command contains the device type, action, target, timestamp, sequence number, and checksum. Its JSON representation occupies 78–82 bytes in UTF-8 and expands to approximately 96 bytes after validation fields are appended. The measured serialization and validation overhead on the Jetson Nano is 0.31 ± 0.04 ms. Malformed fields, expired timestamps, and checksum mismatches cause message rejection. When a valid command or visual observation is temporarily unavailable, the WS63 controller enters a conservative fallback state by reducing patrol speed, increasing observation frequency, preserving the latest valid state, and notifying the operator rather than executing stale input.

The fallback logic is implemented as a four-state machine. NORMAL executes the configured route; WARNING retains normal connectivity but reduces speed after a high-confidence event; DEGRADED is entered after an observation or communication timeout and applies the reduced-speed, high-scan policy; RECOVERY requires a fresh frame with a valid sequence, timestamp, and checksum before the normal command stream is restored. The operator interface displays the active state and reason code, so a loss of confidence is visible rather than silently masked.

5. Experimental Evaluation

5.1 Test Setup

The prototype consists of a WS63 controller board, a Jetson Nano edge node, geared DC motors with encoder feedback, ultrasonic and infrared sensing modules, a gas sensor, and a temperature-humidity sensor. The software stack includes OpenHarmony 5.0 Release SDK, HDF, DevEco Studio 5.0, OpenCV, PyTorch, TensorRT, and the ArkTS/ArkUI application. The system was evaluated as an integrated manufacturing-oriented monitoring prototype, while communication, motion control, visual perception, power consumption, northbound publication, and end-to-end warning performance were measured separately to isolate the contribution of each subsystem.

Communication-latency tests were conducted in an office-like indoor environment with 5–7 active Wi-Fi access points operating on channels 1, 6, and 11. The observed received-signal-strength range was approximately −65 to −45 dBm, and the ambient temperature was maintained at 25±2C. Twenty measurement blocks were used, and the tested communication paths were interleaved in randomized order within each block to reduce temporal confounding. Board-level power was measured at the regulated inputs using an inline DC analyzer during five paired 20 min runs after system warm-up. Jetson Nano power measurements covered continuous camera acquisition, preprocessing, inference, post-processing, and telemetry transmission, whereas WS63 measurements covered sensing, local control, frame construction, and state reporting.

The Jetson Nano operated in MAXN mode with the GPU clock fixed at 921 MHz and the CPU clock fixed at 1.43 GHz, while the WS63 was configured at 160 MHz although the SoC supports up to 240 MHz [19]. Detector deployment used TensorRT FP16 with batch size 1. Detector latency includes image preprocessing, TensorRT inference, and non-maximum suppression, but excludes northbound interface rendering. Board-level power measurements additionally include continuous camera input and telemetry transmission; they are not GPU-only or kernel-only energy measurements. Electrical power was logged at 1 kS/s using an analyzer with ±0.05% full-scale accuracy. The complete measurement and evaluation protocol is summarized in Table 1.

Table 1. Measurement and evaluation protocol

Experiment	Configuration
Communication latency	3 m LOS; 20-byte binary matched-payload tests for Wi-Fi/UDP, BLE, and SLE; deployed Wi-Fi+JSON full-stack baseline; 1000 RTTs/link; 20 randomized time blocks; RSSI −65 to −45 dBm; 25 ± 2 °C.
Motion control	12 m route; 20 laps; 2.3 kg nominal gross mass; 2.6 kg ballast condition; reference speeds of 0.15, 0.25, and 0.35 m/s; dry tile and concrete surfaces.
Power	Jetson MAXN, GPU 921 MHz, CPU 1.43 GHz, TensorRT FP16 batch 1; WS63 160 MHz; 1 kS/s analyzer; five paired 20 min runs.
End-to-end safety events	500 visual-event trials comprising 206 normal, 102 night, and 192 partial-occlusion cases; 1280×720 video at 15 FPS; warning success threshold of 200 ms; conservative fallback enabled.

5.2 Communication Latency and Motion Performance

Communication latency was evaluated through repeated command round-trip tests under the 3 m line-of-sight conditions summarized in Table 1. SparkLink SLE, Wi-Fi/UDP, and BLE transmitted the same 20-byte application frame between the WS63 controller and the Jetson Nano. The deployed Wi-Fi-plus-JSON path was evaluated separately as a full-stack implementation reference because it additionally includes JSON serialization and validation overhead. Each communication path was tested using 1000 round trips organized into 20 randomized measurement blocks of 50 samples. The blocks were used to balance the test order and examine temporal stability, while the reported latency, jitter, tail latency, and packet-loss values were calculated from the complete set of measurements. For latency samples Li, the mean round-trip delay, jitter, and relative latency reduction are defined as follows:

$$\bar{L} = \frac{1}{N}\sum_{i=1}^{N} L_i, \quad (3)$$

$$\sigma_L = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(L_i - \bar{L})^2} \quad (4)$$

The relative latency reduction with respect to baseline path b is

$$\eta_L = \frac{\bar{L}_{base} - \bar{L}_{SL}}{\bar{L}_{base}} \times 100\% \quad (5)$$

Table 2 reports the communication results under the matched-frame and deployed full-stack conditions. In the matched-frame comparison, SparkLink SLE achieved a mean round-trip latency of 4.8 ± 1.2 ms, compared with 14.6 ± 3.2 ms for Wi-Fi/UDP and 17.2 ± 3.4 ms for BLE. According to (5), these results correspond to latency reductions of 67.1% and 72.1%, respectively. The deployed Wi-Fi-plus-JSON path required 23.4 ± 5.8 ms, corresponding to an implementation-level reduction of 79.5% for the compact SLE path. SparkLink SLE also achieved a 99th-percentile latency of 7.1 ms and a measured packet-loss rate of 0.3%. Table 3 summarizes the absolute and relative latency differences between SparkLink SLE and the three reference paths. These results are limited to the stated short-range indoor line-of-sight environment and should not be generalized to non-line-of-sight, dense-multipath, or high-interference manufacturing conditions.

Table 2. Matched-frame and full-stack communication results

Communication path	Mean RTT (ms)	Jitter (ms)	P99 RTT (ms)	Packet loss (%)	Comparison basis
Wi-Fi/UDP binary	14.6	3.2	22.8	0.9	Matched 20-byte frame
BLE binary	17.2	3.4	28.5	1.3	Matched 20-byte frame
SparkLink SLE binary	4.8	1.2	7.1	0.3	Matched 20-byte frame
Wi-Fi + JSON	23.4	5.8	41.6	1.8	Deployed full stack

Table 3. Descriptive latency differences relative to SparkLink SLE

Reference path	Absolute latency difference (ms)	Relative latency reduction (%)	Comparison type	Reference path
Wi-Fi/UDP binary	9.8	67.1	Matched-frame	Wi-Fi/UDP binary
BLE binary	12.4	72.1	Matched-frame	BLE binary
Wi-Fi + JSON	18.6	79.5	Full-stack implementation	Wi-Fi + JSON

Motion-control experiments were conducted over 20 repeated patrol laps on a 12 m campus-like route with mild terrain variation and obstacle interactions. The speed-tracking and path-following metrics are reported as the steady-state speed error

$$e_{ss} = \frac{|v_{ref} - \bar{v}|}{v_{ref}} \times 100\% \quad (6)$$

and the mean lateral trajectory deviation

$$d_{traj} = \frac{1}{N_p}\sum_{i=1}^{N_p}|y_i - y_i^{ref}| \quad (7)$$

where vref is the reference speed, v is the measured steady-state speed, yi is the sampled lateral position, and y_i^* is the reference path position. Using (6) and (7), the PID loop limited the speed error to 1.7 ± 0.4% of the target while constraining trajectory deviation to 4.2 ± 0.6 cm under the nominal 2.3 kg gross mass. A supplemental ballast test at the 0.25 m/s reference speed produced steady-state speed errors of 1.6% and 2.1% for gross masses of 2.3 and 2.6kg, respectively. Table 4 consolidates these motion-control results and shows that the fixed-gain controller remains stable but gradually loses regulation quality as payload increases. This behavior is consistent with the broader mobile-robot control literature: The observed payload sensitivity is consistent with reported limitations of fixed-gain control on low-cost mobile robots [20].These results satisfy the design targets in Section 2.1 for the nominal patrol configuration but also show that heavier loads would benefit from retuning or gain scheduling. They are also consistent with recent studies on moving robots, which show that stable local links and near-edge computation reduce the risk of control degradation in dynamic operation. The southbound controller therefore remains responsible for fast actuation, whereas the edge node handles the more computationally intensive inference tasks.

Table 4. Motion-control dynamics and payload sensitivity

Condition	Speed error (%)	Rise time (s)	Overshoot (%)
0.15 m/s, 2.3 kg	1.4 ± 0.3	0.42 ± 0.05	2.6 ± 0.7
0.25 m/s, 2.3 kg	1.6 ± 0.4	0.47 ± 0.06	3.1 ± 0.8
0.35 m/s, 2.3 kg	2.0 ± 0.5	0.54 ± 0.07	3.8 ± 1.0
0.25 m/s, 2.6 kg	2.1 ± 0.4	0.51 ± 0.06	3.7 ± 0.9

5.3 Visual Perception and Edge Deployment

The perception model is derived from YOLOv10-N [21]. GSConv replaces selected standard convolutions in the neck to reduce feature-transport cost [22], while MSAA is inserted into selected feature-fusion stages to strengthen multi-scale feature weighting. Recent embedded traffic-sign studies have also emphasized the trade-off among recognition accuracy, latency, and hardware resources [23].Figure 4 illustrates the modified YOLOv10-N architecture, including the GSConv neck blocks and the MSAA-enhanced feature-fusion stages.

The implemented MSAA block is a lightweight convolutional channel-spatial attention mechanism rather than token-based self-attention. It operates after feature concatenation at selected neck scales, uses pooled channel descriptors and spatially aggregated responses to generate bounded weights, and applies residual reweighting before the subsequent C2f block. The insertion points were selected at the medium- and high-resolution fusion stages because small and partially occluded signs depend most strongly on cross-scale feature retention. The associated parameter and latency costs are quantified in the component-ablation results presented later in this section.

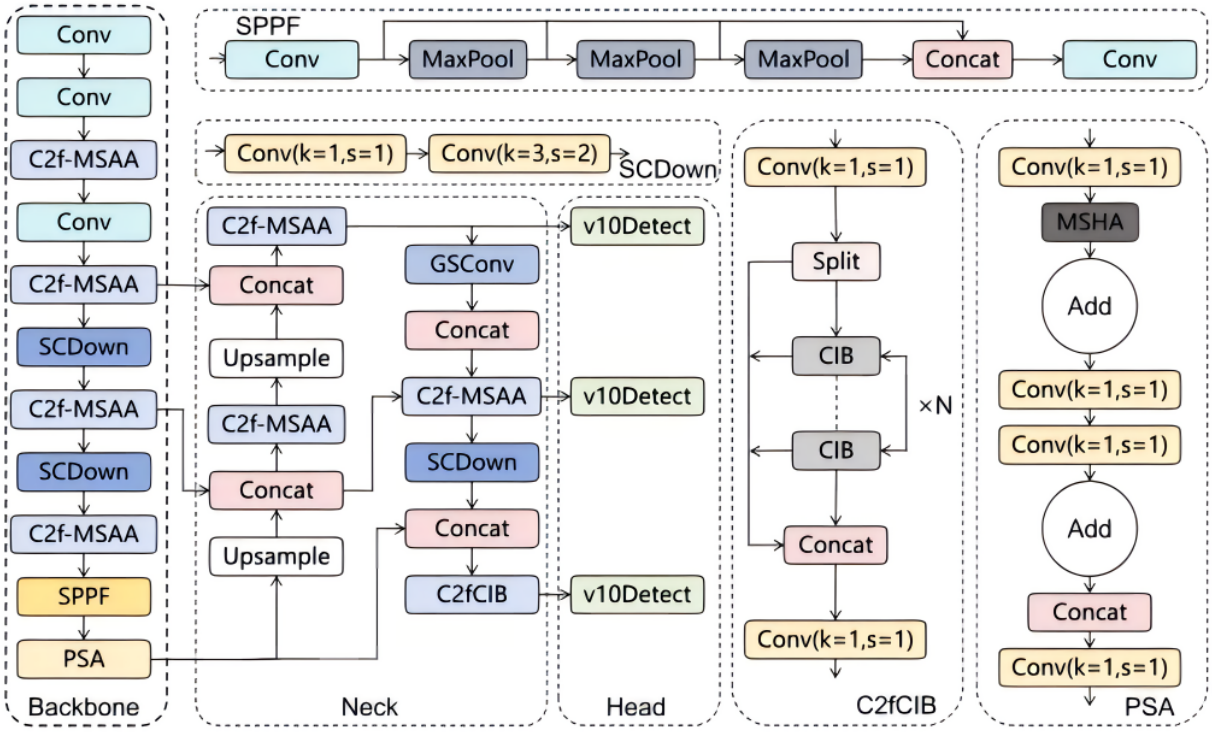

Figure 4. Modified YOLOv10-N architecture with GSConv neck blocks and MSAA feature-fusion module

Experiments were conducted on the TT100K benchmark [24]. The filtered protocol contains 45 classes with at least 100 annotated instances and 8,941 images, comprising 5,962 development images and 2,979 held-out test images. The development set was divided into 5,366 training images and 596 validation images using a fixed stratified split, while the test set was used only for final evaluation. The detector was trained for 300 epochs using SGD with momentum 0.9, weight decay 5×10-4, cosine learning-rate decay from 0.001 to 0.0001, 640 × 640 inputs, and COCO-pretrained initialization. Augmentation included color jittering, Mosaic composition, and class-aware horizontal flipping. For direction-sensitive signs, labels were remapped to the corresponding mirrored class; flipping was disabled when no valid mirrored class existed. Each detector configuration was evaluated using five fixed random seeds. Table 5 summarizes the dataset split, training configuration, augmentation strategy, model-selection criterion, and evaluation protocol.

Table 5. Detector training and evaluation configuration

Item	Configuration
Dataset protocol	TT100K filtered to 45 classes with ≥100 instances; 5,962 development / 2,979 test images
Development split	5,366 train / 596 validation, fixed stratified split
Initialization	COCO-pretrained YOLOv10-N weights
Epochs / batch	300 / 16
Optimizer	SGD; momentum 0.9; weight decay 5×10^{-4}
Learning rate	Cosine schedule; 0.001 initial, 0.0001 minimum
Input	640×640
Augmentation	Color jitter; Mosaic; class-aware horizontal flipping for direction-sensitive signs
Model selection	Highest validation mAP50:95; test set used once for final evaluation
Random seeds	Five fixed seeds per configuration; Table 6 reports the arithmetic mean.

The evaluation metrics are defined as follows:

$$P = \frac{TP}{TP+FP} \times 100\% \quad (8)$$

$$R = \frac{TP}{TP+FN} \times 100\% \quad (9)$$

$$\text{mAP} = \frac{1}{C}\sum_{i=1}^{C} A\,P_i \quad (10)$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively, and C is the number of evaluated classes. The paper reports mAP50 at an IoU threshold of 0.50 and mAP50:95 averaged over IoU thresholds from 0.50 to 0.95 in increments of 0.05. Precision and recall are calculated at the confidence threshold selected on the validation set. Jetson throughput includes preprocessing, TensorRT FP16 inference, and non-maximum suppression with batch size 1.

Table 6 presents the lightweight baseline comparison and component ablation. Relative to YOLOv10-N, GSConv reduces the parameter count from 2.3 M to 1.9 M and increases throughput from 24.6 to 30.1 FPS, while producing a limited 0.20-point improvement in mAP50. MSAA alone increases mAP50 from 80.84% to 82.31% and mAP50:95 from 48.37% to 50.12%, although its throughput is slightly lower than that of the baseline. The combined GSConv+MSAA model achieves 83.96% mAP50, 51.74% mAP50:95, 76.58% recall, 2.1 M parameters, and 28.3 FPS. Compared with YOLOv10-N, these values correspond to gains of 3.12 points in mAP50, 3.37 points in mAP50:95, 3.68 points in recall, and 3.7 FPS, together with a reduction of 0.2 M parameters. Across five fixed seeds, the standard deviation remained within 0.5 mAP points and 0.6 recall points. The earlier p-value statement has been removed because the original records did not preserve a sufficiently explicit statistical-test specification.

Table 6. Lightweight detector comparison and component ablation on TT100K

Model	Precision (%)	Recall (%)	mAP50 (%)	mAP50:95 (%)	Params (M)	FPS	Latency (ms)
YOLOv5-N	81.11	70.11	78.59	45.32	2.6	26.437.88	
YOLOv8-N	82.35	69.18	78.71	46.10	3.2	23.742.19	
YOLOv10-N	82.06	72.90	80.84	48.37	2.3	24.640.65	
YOLOv10-N + GSConv	81.98	72.11	81.04	48.62	1.9	30.133.22	
YOLOv10-N + MSAA	82.18	75.21	82.31	50.12	2.4	23.941.84	
YOLOv10-N + GSConv + MSAA	82.32	76.58	83.96	51.74	2.1	28.335.34	

YOLO11-N is a relevant newer lightweight reference [23,25]. However, direct numerical comparison is not reported because available studies use different dataset filters, preprocessing pipelines, TensorRT versions, clock settings, and measurement boundaries. The performance claims in Table 6 are therefore restricted to the baselines evaluated under the identical protocol; an identical-protocol YOLO11-N comparison remains future work.

Figure 5 provides representative qualitative examples. The combined model recovers several small or partially occluded traffic signs missed by the lightweight baselines and reduces selected duplicate detections. These examples are illustrative and should be interpreted together with the aggregate metrics and failure analysis rather than as independent evidence. The Hough-based road-edge module is an auxiliary route-processing component [26] and is not included in the detector metrics reported in Table 6.

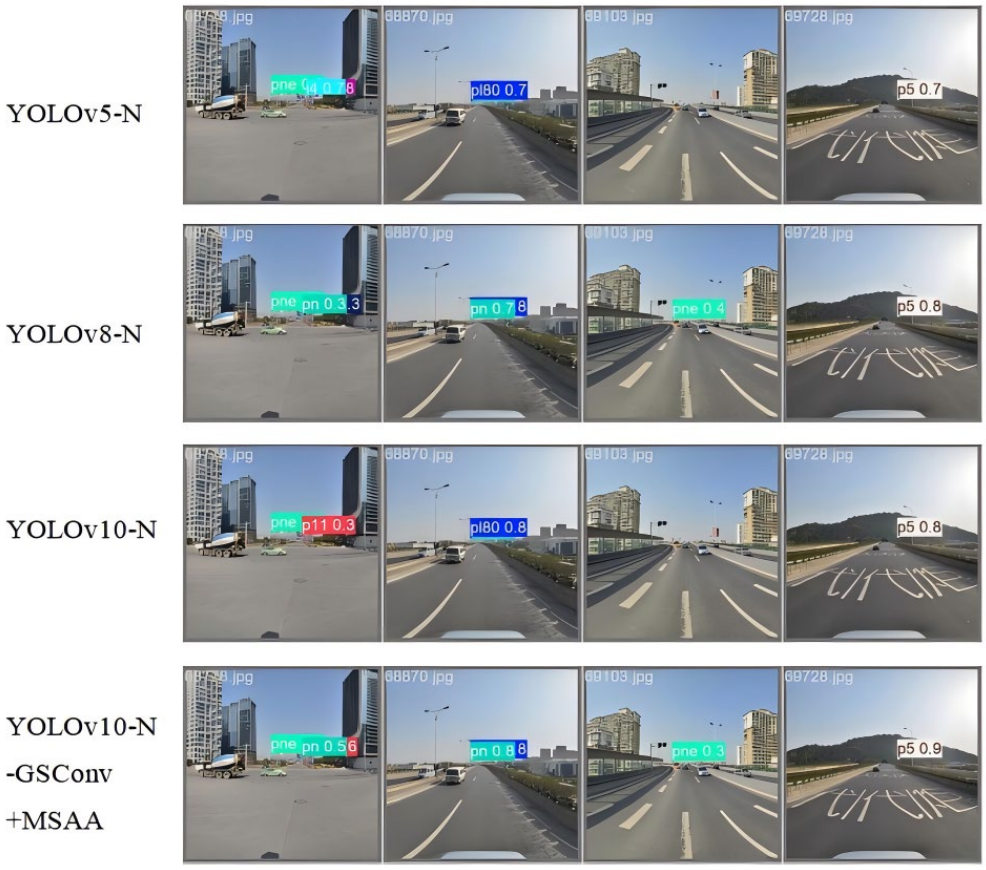

Figure 5. Representative TT100K detections produced by the lightweight baselines and the combined GSConv+MSAA model

5.4 Power Consumption and Energy Efficiency

Power measurements were conducted to quantify the energy-efficiency gains of the lightweight cross-layer design on both the Jetson Nano edge node and the WS63 southbound controller. For the Jetson-side comparison, the baseline YOLOv10-N model and the proposed YOLOv10-N-GSConv+MSAA model were deployed under the same input resolution and clock configuration. The reported Jetson values are board-level averages collected during sustained camera input, inference, post-processing, and telemetry transmission rather than instantaneous GPU-only power snapshots. For the WS63-side comparison, a conventional periodic-polling communication mode was used as the baseline, whereas the proposed mode used compact event-driven packet aggregation together with the OpenHarmony distributed bus service to reduce unnecessary synchronization overhead. Here, the effective sense-control-report transaction rate on WS63 denotes the number of completed sense-control-report cycles per second during stable patrol operation.

The energy-efficiency metrics reported in Table 7 are defined as

$$\eta_{\mathrm{inf}} = \frac{f_{\mathrm{inf}}}{P_{\mathrm{avg}}}, \qquad \eta_{loop} = \frac{f_{loop}}{P_{avg}} \tag{11}$$

where finf is inference throughput in FPS, floop is the controller sense-control-report transaction rate in cycles/s, and Pavg is the measured average electrical power.

Table 7. Board-level power and energy efficiency over five paired 20 min runs

Platform	Configuration	Throughput	Power	Success rate	Energy efficiency
Jetson Nano	YOLOv10-N	24.6 ± 0.4 FPS	9.42 ± 0.11 W	—	2.61 FPS/W
Jetson Nano	GSConv + MSAA	28.3 ± 0.5 FPS	9.18 ± 0.09 W	—	3.08 FPS/W
WS63	Periodic polling	42.0 ± 0.8 cycles/s	0.72 ± 0.02 W	97.8 ± 0.4%	58.3 cycles/s/W
WS63	Event-driven reporting	46.0 ± 0.7 cycles/s	0.66 ± 0.01 W	98.8 ± 0.3%	69.7 cycles/s/W

Across five paired runs, the combined detector increased Jetson throughput from 24.6 ± 0.4 to 28.3 ± 0.5 FPS while reducing average board power from 9.42 ± 0.11 to 9.18 ±

0.09 W. According to (11), inference energy efficiency increased from 2.61 to 3.08 FPS/W, corresponding to an 18.0% improvement. On WS63, event-driven reporting increased the sense-control-report transaction rate from 42.0 ± 0.8 to 46.0 ± 0.7 cycles/s and reduced controller-board power from 0.72 ± 0.02 to 0.66 ± 0.01 W. Controller-side energy efficiency therefore increased from 58.3 to 69.7 cycles/s/W. The absolute power reductions are modest; the principal benefit is the increase in useful work per unit energy under fixed clock settings.

5.5 End-to-End Patrol Validation

Recent patrol studies have investigated visual alert recognition and autonomous route planning [27,28]; the present evaluation instead focuses on end-to-end warning delivery and continuity-oriented fallback.The complete pipeline was evaluated with motion control, environmental sensing, camera streaming, edge inference, northbound publication, and fallback behavior enabled simultaneously. The robot-state stream was published at 5 Hz, while slower environmental channels retained their sensor-appropriate update periods. Table 8 reports application-level delay from edge publication to the rendered northbound state.

Table 8. Northbound monitoring performance

Signal	Update period (s)	Application delay (ms)	Success rate (%)
Robot state stream	0.2	78 ± 9	100
Temperature / humidity	2	62 ± 7	100
Gas concentration	2	68 ± 8	100
Ultrasonic range	1	54 ± 6	100

The end-to-end visual warning task was limited to traffic-sign-derived disruption indicators represented in the filtered TT100K protocol. The evaluated events were grouped into stop/yield, speed-restriction, prohibitory, and road-warning categories. These categories were used as controlled proxies for route restrictions and warning conditions rather than as confirmed manufacturing-equipment failures. A trial was counted as correct only when the event group was recognized and the corresponding warning was rendered within the 200 ms success threshold. The 500 visual trials comprised 206 normal, 102 night, and 192 partial-occlusion cases. Table 9 reports the aggregate end-to-end recognition and alert-latency results.

Table 9. End-to-end visual disruption-warning results

Scenario (trials)	Recognition accuracy (%)	Mean latency (ms)	P95 latency (ms)
Normal (206)	96.1	82 ± 14	109
Night (102)	93.1	101 ± 19	132
Partial occlusion (192)	92.2	114 ± 20	142
Overall (500)	94.0	97.9 ± 22.6	137

Table 10 further summarizes the alert-latency distribution, visual-error composition, dominant failure conditions, and fallback behavior.

Table 10. Alert-latency distribution and visual-error composition

Metric	Summary
Latency distribution	Mean 97.9 ± 22.6 ms; median 94 ms; P95 137 ms; P99 145 ms.
Visual errors	30/500 trials: 21 false negatives (70%) and 9 false positives (30%).
Dominant misses	Targets <32×32 px, >50% partial occlusion, or low local contrast.
Dominant false alarms	Reflective backgrounds, glare, and visually similar sign patterns.
Fallback behavior	Patrol speed reduced and scan frequency increased until the next valid observation; 100% fallback execution success.

Table 11 makes the proxy mapping explicit. The mapping is functional rather than semantic: a visual class is used to trigger a patrol action with a similar operational meaning. For example, a prohibitory sign is treated as a no-entry or hazardous-zone cue, not as evidence of a machine defect. Accordingly, the table is an operational interpretation aid and not empirical validation of equipment-anomaly recognition.

Table 11. Functional mapping of proxy groups to representative manufacturing patrol conditions and response actions

Proxy group	Illustrative manufacturing condition	System action	Interpretation boundary
Stop / yield	Temporary aisle closure; vehicle or personnel crossing	Stop or reduce speed; notify operator	Route-control cue only
Speed restriction	Shared workspace; low-speed safety zone	Apply lower speed setpoint	Does not estimate process risk
Prohibitory	No-entry area; hazardous or maintenance zone	Hold position or request route change	Not an equipment-failure diagnosis

Proxy group	Illustrative manufacturing condition	System action	Interpretation boundary
Road warning	Obstacle, spill, floor damage, or maintenance activity	Increase scan frequency; raise warning	Requires domain-specific retraining for deployment

5.6 Limitations and Validity Considerations

Internal validity is limited by implementation coupling: the deployed SLE and Wi-Fi-plus-JSON paths differ in both bearer and serialization, which is why a matched binary comparison is reported separately. External validity is limited to one robot, one Jetson Nano, a predominantly indoor campus-like route, line-of-sight short-range radio tests, dry surfaces, and moderate illumination. Construct validity is limited to traffic-sign-derived safety events and environmental sensor thresholds rather than general manufacturing-event recognition.

The absence of an identical-protocol YOLO11-N benchmark and a manufacturing-domain image dataset limits comparative and construct validity. Both additions require complete retraining, deployment, and end-to-end evaluation under the same protocol. Accordingly, the present claims are restricted to cross-layer engineering feasibility, and identical-protocol model comparison and manufacturing-domain validation remain necessary next steps.

6. Conclusion

Manufacturing resilience depends on the timely detection of route restrictions, environmental hazards, and other disruption indicators before they propagate into longer operational interruptions. To support this requirement, this paper presents an OpenHarmony-based AI-driven edge-intelligent monitoring framework integrating local embedded control, SparkLink SLE telemetry, Jetson-based perception, and an ArkTS/ArkUI supervision interface.

Under the stated test conditions, SparkLink SLE achieved 4.8 ± 1.2 ms round-trip latency, corresponding to reductions of 67.1% relative to matched binary Wi-Fi/UDP, 72.1% relative to BLE, and 79.5% relative to the deployed Wi-Fi-plus-JSON stack. The combined detector achieved 83.96% mAP50, 51.74% mAP50:95, and 28.3 FPS with 2.1 M parameters. The nominal controller maintained an aggregate speed error of 1.7 ± 0.4% and a trajectory deviation of 4.2 ± 0.6 cm. The Jetson and WS63 configurations achieved energy efficiencies of 3.08 FPS/W and 69.7 sense-control-report cycles/s/W, respectively. Across 500 visual safety-event trials, overall recognition accuracy was 94.0%, with mean alert latency of 97.9 ± 22.6 ms and a 99th-percentile latency of 145 ms.

These findings support the feasibility of the integrated design under a constrained single-robot indoor and campus-like deployment. They do not establish general superiority over ROS 2 or other robotics stacks, performance under non-line-of-sight or electromagnetic-stress conditions, or manufacturing-equipment anomaly detection. The traffic-sign-derived classes are controlled proxies for route and warning actions, and domain deployment requires retraining and validation on representative factory events.

Future work will evaluate non-line-of-sight and interference conditions, incorporate IMU/UWB-based multi-sensor fusion, replace the fixed-gain PID controller with payload-aware control, and extend the architecture to coordinated multi-robot patrol. An identical-protocol YOLO11-N benchmark, validation on real manufacturing images, dynamic-obstacle navigation, additional edge platforms, and public release of reproducibility artifacts are also planned before wider deployment claims are made.

Acknowledgements

The authors thank Neusoft Institute Guangdong for providing the embedded-development environment, hardware integration support, and campus-road test conditions used in this study.

Funding

This work was supported by the Project of Guangdong Provincial Association of Higher Education (ID: 24GYB79) and the Project of Guangdong Provincial Association of Teaching Management in Colleges and Universities (ID: GDZLGL25013).

Authors' Contributions

Boqiang Zhang and Yuxin Zhuo contributed equally to the system design, software implementation, experiment execution, and manuscript drafting. Weiting He supervised the study, refined the technical route, and revised the manuscript. Zhiwei Chen and Zhibin Xian supported hardware integration, application development, and experimental measurements. Chengming Huang contributed to result verification, reference checking, and language polishing. Jiamin Chen and Zonghui Huang contributed to experimental support, data organization, and manuscript review. All authors read and approved the final manuscript.

Competing Interests

The authors declare that they have no competing interests.

Availability Of Data and Materials

The TT100K traffic-sign dataset used for detector training and evaluation is publicly available from the TT100K benchmark project. On reasonable request, the corresponding author can provide the fixed train/validation split, detector configuration files, inference-export settings, communication frame definition, latency and power logs, and ArkTS interface configuration used to reproduce the reported results. The materials are not currently hosted in a public repository.

References

[1] J. Liu, G. Xie, J. Wang, S. Li, C. Wang, F. Zheng, and Y. Jin, "Deep industrial image anomaly detection: A survey," Machine Intelligence Research, vol. 21, no. 1, pp. 104–135, 2024, doi: 10.1007/s11633-023-1459-z.

[2] L. Lin, J. Guo, and L. Liu, "Multi-scene application of intelligent inspection robot based on computer vision in power plant," Scientific Reports, vol. 14, art. 10657, 2024, doi: 10.1038/s41598-024-56795-8.

[3] J. Lee, S. Siahpour, X. Jia, and P. Brown, "Introduction to resilient manufacturing systems," Manufacturing Letters, vol. 32, pp. 24-27, 2022, doi: 10.1016/j.mfglet.2022.02.002.

[4] J. Leng, J. Xie, R. Li, X. Zhou, X. Gu, Q. Liu, X. Chen, W. Zhang, and A. Kusiak, "Resilient manufacturing: A review of disruptions, assessment, and pathways," Journal of Manufacturing Systems, vol. 79, pp. 563-583, 2025, doi: 10.1016/j.jmsy.2025.02.006.

[5] V. Shukla, A. Shukla, S. P. S. K., and S. Shukla, "A systematic survey: Role of deep learning-based image anomaly detection in industrial inspection contexts," Frontiers in Robotics and AI, vol. 12, art. 1554196, 2025, doi: 10.3389/frobt.2025.1554196.

[6] W. Chen, W. Chi, S. Ji, H. Ye, J. Liu, Y. Jia, J. Yu, and J. Cheng, "A survey of autonomous robots and multi-robot navigation: Perception, planning and collaboration," Biomimetic Intelligence and Robotics, vol. 5, no. 2, art. 100203, 2025, doi: 10.1016/j.birob.2024.100203.

[7] K. Kobayashi, S. Ueno, and T. Higuchi, "Multi-robot patrol with continuous connectivity and assessment of base station situation awareness," Journal of Robotics and Mechatronics, vol. 36, no. 3, pp. 526–537, 2024, doi: 10.20965/jrm.2024.p0526.

[8] Z. R. Madin, J. Lawry, and E. R. Hunt, "Collective anomaly perception during multi-robot patrol: Constrained interactions can promote accurate consensus," in Proc. 39th ACM/SIGAPP Symposium on Applied Computing, 2024, pp. 630-637, doi: 10.1145/3605098.3635975.

[9] M. Groshev, G. Baldoni, L. Cominardi, A. de la Oliva, and R. Gazda, "Edge robotics: Are we ready? An experimental evaluation of current vision and future directions," Digital Communications and Networks, vol. 9, no. 1, pp. 166–174, 2023, doi: 10.1016/j.dcan.2022.04.032.

[10] N. Tahir and R. Parasuraman, "Edge computing and its application in robotics: A survey," Journal of Sensor and Actuator Networks, vol. 14, no. 4, art. 65, 2025, doi: 10.3390/jsan14040065.

[11] J. L. Mela and C. García Sánchez, "Yolo-based power-efficient object detection on edge devices for USVs," Journal of Real-Time Image Processing, vol. 22, art. 108, 2025, doi: 10.1007/s11554-025-01682-2.

[12] J. Krejčí, M. Babiuch, J. Suder, V. Krys, and Z. Bobovský, "Latency-sensitive wireless communication in dynamically moving robots for urban mobility applications," Smart Cities, vol. 8, no. 4, art. 105, 2025, doi: 10.3390/smartcities8040105.

[13] M. L. Gambo, A. Danasabe, B. Almadani, F. Aliyu, A. Aliyu, and E. Al-Nahari, "A systematic literature review of DDS middleware in robotic systems," Robotics, vol. 14, no. 5, art. 63, 2025, doi: 10.3390/robotics14050063.

[14] OpenAtom OpenHarmony, "OpenHarmony Project Overview," OpenHarmony Documentation, 2026. [Online]. Available: https://github.com/openharmony/docs/blob/master/en/OpenHarmony-Overview.md. Accessed: Jul. 30, 2026.

[15] M. Gao, L. Wan, R. Shen, Y. Gao, J. Wang, Y. Li, and B. Vucetic, "SparkLink: A short-range wireless communication protocol with ultra-low latency and ultra-high reliability," The Innovation, vol. 4, no. 2, art. 100386, 2023, doi: 10.1016/j.xinn.2023.100386.

[16] B. Liu, Z. Ren, Y. Zhang, M. Li, J. Liang, B. He, J. Li, H. Wu, and J. Zhou, "Unveiling SparkLink Low Energy: A comparative measurement study," ACM Transactions on Internet of Things, vol. 7, no. 1, art. 6, pp. 1-24, 2026, doi: 10.1145/3776559.

[17] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics, vol. 7, no. 66, art. eabm6074, 2022, doi: 10.1126/scirobotics.abm6074.

[18] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ROS2," in Proceedings of the 13th International Conference on Embedded Software, Pittsburgh, PA, USA, 2016, pp. 1–10, doi: 10.1145/2968478.2968502.

[19] HiSilicon, "Hi3863V100: Wi-Fi 6 and NearLink multi-mode IoT SoC," 2026. [Online]. Available: https://www.hisilicon.com/en/products/connectivity/short-range-iot/wifi-nearlink-ble/hi3863v100. Accessed: Jul. 30, 2026.

[20] L. Mérida-Calvo, A. San-Millán Rodríguez, F. Ramos, and V. Feliu-Batlle, "Advanced motor control for improving the trajectory tracking accuracy of a low-cost mobile robot," Machines, vol. 11, no. 1, art. 14, 2023, doi: 10.3390/machines11010014.

[21] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "YOLOv10: Real-time end-to-end object detection," in Advances in Neural Information Processing Systems, vol. 37, 2024, doi: 10.52202/079017-3429.

[22] H. Li, J. Li, H. Wei, Z. Liu, Z. Zhan, and Q. Ren, "Slim-neck by GSConv: A lightweight-design for real-time detector architectures," Journal of Real-Time Image Processing, vol. 21, no. 3, art. 62, 2024, doi: 10.1007/s11554-024-01436-6.

[23] R. Reveles-Martínez, H. Gamboa-Rosales, E. Sánchez-Femat, J. Saldívar-Pérez, T. Ibarra-Pérez, L. C. Reveles-Gómez, O. A. Guirette-Barbosa, J. I. Galván-Tejada, C. E. Galván-Tejada, and J. M. Celaya-Padilla, "Benchmarking YOLOv8 to YOLOv11 architectures for real-time traffic sign recognition in embedded 1:10 scale autonomous vehicles," Technologies, vol. 13, no. 11, art. 531, 2025, doi: 10.3390/technologies13110531.

[24] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016, pp. 2110–2118, doi: 10.1109/CVPR.2016.232.

[25] Ultralytics, "Ultralytics YOLO11," 2024. [Online]. Available: https://docs.ultralytics.com/models/yolo11/. Accessed: Jul. 30, 2026.

[26] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," Communications of the ACM, vol. 15, no. 1, pp. 11–15, 1972, doi: 10.1145/361237.361242.

[27] Y. Shi, X. Zhang, Q. Wang, and X. Liu, "Public security patrol and alert recognition for police patrol robots based on improved YOLOv8 algorithm," Mathematical and Computational Applications, vol. 30, no. 5, art. 97, 2025, doi: 10.3390/mca30050097.

[28] Q. Zou, H. Wang, T. Zhang, Z. Li, and Y. Zhuang, "Research on path planning method for autonomous patrol robots," Electronics, vol. 13, no. 14, art. 2865, 2024, doi: 10.3390/electronics13142865.