International Journal of Basic and Applied Sciences, 14 (5) (2025) 659-663

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/rtjzhc70
Research paper



Integration of YOLO-RTLite with Wireless Sensor Network Simulation in OMNeT++

Manjot Kaur [1] *, Rajneesh Randhawa [2], Jagroop Kaur [3]

[1] Research Scholar, Department of Computer Science, Punjabi University, Patiala
[2] Assistant Professor, Department of Computer Science, Punjabi University, Patiala
[3] Associate Professor, Department of Computer Science and Engineering, Punjabi University, Patiala
**Corresponding author E-mail: manjotkaur1699@gmail.com*

Received: August 1, 2025, Accepted: September 9, 2025, Published: September 17, 2025

Abstract

Real-time decision-making in smart surveillance and traffic management systems requires efficient coordination between visual detection components and wireless sensor networks. In this context, the authors propose an integrated simulation framework that connects YOLO-YOLO-YOLO-RTLite-based object detection with a wireless sensor network. The WSN is simulated in OMNeT++, using MQTT as the communication protocol for sending alerts. The system is meant for curved or blind road sections. It uses an object detector on one side to warn vehicles on the other side about oncoming hazards. The framework is evaluated based on metrics such as end-to-end delay, throughput, system stability, and alert reliability. Results confirm the system's ability to function consistently under real-time constraints. A significant contribution of this work is the novel integration of live visual input from YOLO with OMNeT++ simulations, enabling event-driven behavior in WSN triggered by actual object detections.

Keywords: *Alert Generation; OMNeT++; Object Detection; Real-Time Simulation; Wireless Sensor Network.*

1. Introduction

Wireless Sensor Networks (WSN) play an important role in real-time monitoring and alert systems. They are especially useful in safety-related areas like road safety. With the rise of deep learning, it's now possible to combine object detection models like YOLO with WSN to improve hazard detection and warning systems.

This work focuses on integrating a lightweight object detection model, YOLO-RTLite, into a WSN using the OMNeT++ simulation framework. YOLO-RTLite is an optimized version of YOLOv4-Tiny. It includes changes like a spatial convolutional layer and updated route layers. The model is also accelerated using TensorRT to support real-time use.

YOLO-RTLite is used to detect objects such as vehicles, animals, or pedestrians. Once detected, alerts are sent using the MQTT protocol to nodes within the simulated WSN. The simulation helps evaluate how these alerts travel across the network under different settings.

This setup is designed to support road safety at blind turns and curved paths, where one node detects the object and another alerts the driver. The system helps model real-world safety scenarios and shows how vision-based detection can work together with network simulations for better accident prevention and faster response.

1.1. Related Work

The integration of YOLO-based object detection with Wireless Sensor Networks (WSN) is increasingly explored to develop intelligent, responsive, and scalable systems for road safety. YOLO's real-time detection capability is complemented by the distributed communication and sensing infrastructure of WSN, enabling end-to-end safety solutions across diverse environments. In [1], the SmartDENMA system combined YOLO-based detection with V2X communication through Decentralized Environmental Notification Messages (DENMs) to improve pedestrian safety. Similarly, [2] encoded YOLO outputs into standardized formats for integration with intelligent traffic systems. Thakur et al. [3] introduced a LEACH-YOLO-ResNet framework for forest fire detection, emphasizing hierarchical clustering and energy-efficient communication. While effective for large-scale monitoring, this approach focused mainly on energy conservation and did not directly optimize real-time delay. In contrast, our YOLO-RTLite–OMNeT++ framework achieves an average end-to-end delay of ~110 ms, making it suitable for time-critical road safety applications.

Other domain-specific applications have also been explored. For example, [6] presented a YOLO-based model integrated with sensor nodes for aiding visually impaired individuals, and [7] proposed a wireless sensor network–based framework for area-based speed control and traffic monitoring to enhance road safety. Standard WSN protocols such as IEEE 802.15.4 and LoRaWAN have been used with YOLO in [8] to enable low-power traffic control, while [9] reported VANET-based systems where YOLO and WSN jointly supported

Copyright © Manjot Kaur et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

emergency vehicle safety. Edge-based deployments of YOLO were studied in [10], showing how local inference can reduce delay. However, such systems were limited in scalability to a multi-hop WSN. Our integration with OMNeT++ overcomes this by simulating multi-hop routing (2–3 hops) while retaining real-time responsiveness. Similarly, Akash et al. [11] developed a YOLO-based collision avoidance system with distance estimation, demonstrating high detection reliability. Their work, however, was tailored to a fixed vehicular scenario, whereas our system leverages MQTT as a publish–subscribe bridge, ensuring modularity and reuse across different WSN topologies, such as curved-road hazard alerts or animal crossing detection.
Lightweight YOLO variants optimized through pruning and structural modifications were explored in [12] and [13], targeting deployment on resource-constrained devices. While these works focused on architectural optimization, our contribution emphasizes full system-level integration of vision, communication, and simulation. Advanced studies such as ESRGAN–YOLO integration [14] and UAV-based pothole detection [15] illustrate the breadth of YOLO applications but remain distinct from our focus on modular, low-delay WSN-driven road safety.

2. System Architecture The Architecture Comprises Three Core Modules

YOLO Detection Module: A Python-based script uses YOLO-RTLite to detect objects from video input and generates alert messages including class, confidence, and timestamps. Detection occurs continuously across video frames, and alert messages are published when predefined object classes such as cars, buses, or pedestrians are identified.
MQTT Communication Bridge: A Mosquitto MQTT broker is deployed locally to ensure low-delay communication between the detection module and the simulated WSN. MQTT is a lightweight messaging protocol that enables fast, publish–subscribe communication between detection modules and network nodes. Each alert message is published on a specific topic (e.g., road/alert) and immediately becomes available to subscribed nodes in OMNeT++.
OMNeT++ WSN Simulation: OMNeT++ simulates a wireless sensor network with nodes arranged to model a roadside sensor deployment. A custom YoloListener node subscribes to the MQTT topic, receives incoming alerts, and forwards them across the WSN. The network simulates a multi-hop routing scenario with 2–3 hops between the entry node and the gateway. Alerts are visualized, logged, and timestamped to enable performance evaluation.

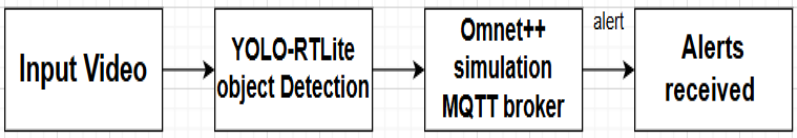


Fig. 1: Block Diagram of Integrated System

The overall workflow of the system is illustrated in Figure 1. The block diagram shows how the YOLO-RTLite detection module processes video frames and generates alerts, which are transmitted through MQTT. These alerts are then passed into the OMNeT++ simulation, where they initiate corresponding network-level events, enabling evaluation of performance parameters such as delay and throughput.

3. Implementation Details

The system consists of a modular pipeline integrating a YOLO-based Python detection module with a simulated OMNeT++ WSN via MQTT. Below is a detailed step-by-step overview of how the system operates from video input to in-simulation packet generation.
Step 1: Object Detection Using YOLO-RTLite
A Python script (detect_alert.py) loads the YOLO-RTLite model using OpenCV and PyTorch. It processes a video stream frame-by-frame, applying object detection in real time. When predefined classes such as person, car, or bicycle are detected with confidence greater than 0.5, the script generates a structured alert.
Step 2: Alert Generation and Publishing over MQTT
Detected object data is packaged in a JSON format containing fields such as class, confidence, and timestamp. The script publishes these alerts to a local MQTT broker (Mosquitto) on the topic yolo/alerts using the paho-mqtt client. This decouples the detection layer from the simulation environment.
Step 3: MQTT Subscription and OMNeT++ Integration
A custom OMNeT++ simulation module named YoloListener (defined in both NED and C++) is subscribed to the MQTT topic. It acts as an entry point for alerts into the simulated WSN. Upon receiving a message, the module parses the content and injects it as a cMessage into the simulation for further processing.
Step 4: In-Network Alert Propagation in OMNeT++
The simulated WSN (defined in your_network.ned) includes multiple static nodes connected in a 2–3 hop topology. Upon receiving an alert, the listener node forwards it to neighboring nodes based on preconfigured routing logic. This mimics real-world alert propagation in roadside sensor deployments.
Step 5: Performance Logging and Analysis
Each alert's arrival time is logged and compared against its original timestamp to compute metrics such as end-to-end delay, throughput, and jitter. Logs are exported for further analysis and visualization. The core pipeline of the proposed system is summarized below in pseudocode. It captures the end-to-end workflow from object detection using YOLO-RTLite, publishing alerts via MQTT, receiving them in OMNeT++ through a custom listener module, and forwarding them across the simulated WSN.

```
// YOLO-RTLite Python Detection Module

function main():
 load YOLO-RTLite model using config and weights
 open video stream
 while video has frames:
```

```
 frame ← read next frame
 detections ← detect_objects(frame)
 for each detection in detections:
 if confidence > threshold:
 alert ← {
 "class": detected_class,
 "confidence": confidence_score,
 "timestamp": current_time()
 }
 publish_to_mqtt(topic="yolo/alerts", message=alert)
 close video stream
 disconnect MQTT
// OMNeT++ MQTT Subscriber Module (YoloListener)
on initialize():
 broker ← read_parameter("mqttBroker")
 port ← read_parameter("mqttPort")
 topic ← "yolo/alerts"
 mqtt_client ← connect_to_broker(broker, port)
 subscribe(mqtt_client, topic)
on message_received(message):
 alert ← parse_JSON(message)
 packet ← create_cMessage_from(alert)
 send(packet, output_gate)
on finish():
 disconnect mqtt_client
// OMNeT++ WSN Simulation Logic
on cMessage arrival at node:
 if node is not a gateway:
 forward(packet) to next_hop
 else:
 log timestamp, compute delay
```

4. Performance Evaluation

Timestamps were collected from the Python script and OMNeT++ subscriber node to evaluate performance.

Table 1: System Performance Parameters

Parameter	Value	Description
Total Alerts Processed	33 alerts	Received during simulation
System Duration	~1.1 seconds	Time from first to last alert
Alert Delivery Delay	100–120 ms	From detection to WSN reception
Average Delay	110 ms	Mean delay across all alerts
Alert Rate	~30 alerts/second	Based on alert frequency
Throughput	~30 messages/second	Equal to alert rate
Packet Delivery Ratio	100%	No loss observed
Packet Loss	0%	No message drop
Network Path Length	2–3 hops	From MQTT entry to WSN gateway
Delay Variation (Jitter)	±10 ms	Minor variance across alerts

Table 2: Alert Reception and Delay Metrics

Sr No.	Class	Confidence	Receive Timestamp (UNIX)	Detection to Reception Delay (ms)
1	person	0.93	1751031955.953714	100
2	car	0.72	1751031955.954223	100
3	bike	0.83	175101959.954223	110
4	bus	0.62	1751031957.048453	120

Figure 2 shows a snapshot of the OMNeT++ simulation running alongside the YOLO-RTLite detection module. The video demonstrates how detected objects generate alerts, which are immediately published via MQTT and displayed in the simulation. This confirms the real-time integration of vision-based detection with the network environment.

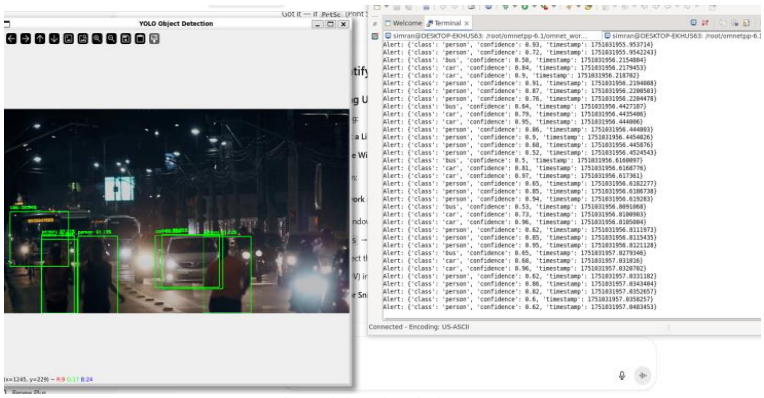


Fig. 2: Snapshot of the Simulation on OMNeT++.

5. Novel Contributions

Live vision-to-network integration: This work presents the first integration of a live object detection module (YOLO-RTLite) with OMNeT++ WSN simulation using MQTT as the bridge. It enables a hybrid simulation environment where real-time events from the perception layer directly influence the behavior of network nodes.
Event-driven simulation using visual data: Unlike traditional WSN simulations that rely on synthetic data or traffic models, this system triggers network events based on actual video frame analysis, providing realism and relevance to road safety scenarios.
End-to-end real-time pipeline: The system achieves real-time performance (delay < 120 ms) across the full pipeline: video frame capture, object detection, alert publication, WSN message routing, and reception.
Reusable architecture: The modularity of the pipeline allows it to be extended to various other applications, such as animal crossing detection, autonomous vehicle alert systems, and environmental hazard monitoring.
Foundation for digital twin simulation: The architecture and implementation methodology serve as a foundational step toward building a full digital twin for road infrastructure, where virtual systems mimic real-world input-output behavior using live or recorded data streams.

6. Limitations

This study demonstrates the integration of YOLO-RTLite with a simulated WSN, but several constraints remain. The simulation was restricted to the application and network layers and did not include physical layer aspects such as signal attenuation, interference, or fading. These omissions limit the ability to capture communication uncertainties that often affect real deployments. Incorporating such models in future work would allow a more realistic assessment of performance.
The network topology used in the experiments was static and small in scale. While sufficient for testing feasibility, it does not represent dynamic conditions such as node mobility, varying density, or congestion. Extending the system to support adaptive routing and dynamic topologies would provide better insight into scalability and robustness.
In the current setup, YOLO-RTLite runs externally and transmits alerts into OMNeT++ through MQTT. This simplifies the design but does not reflect the energy cost of running detection directly on sensor nodes. Future work could examine energy-efficient deployment of lightweight YOLO variants within sensor hardware to balance detection accuracy with node lifetime.
Finally, resilience factors such as packet drops, node failures, and malicious activity were not considered. Evaluating the system under these conditions will be essential to verify its suitability for safety-critical road applications.

7. Conclusion and Future Directions

This work demonstrates a novel simulation-based system that integrates computer vision, real-time messaging, and network simulation. The system successfully delivers real-time alerts with high reliability and minimal delay, making it suitable for intelligent transportation and safety monitoring systems. Future work will include dynamic WSN topologies, failure injection, and digital twin implementation for predictive simulations. The following are the major areas that make room for future research and study.

7.1. Optimizing MQTT for Congested Networks

Future work should investigate how MQTT performs under heavy traffic conditions in dense WSNs. Questions include whether adaptive quality-of-service or priority-based message handling can guarantee the timely delivery of safety-critical alerts.

7.2. Adaptive Routing Protocols in Dynamic Topologies

A key research direction is exploring adaptive routing methods that account for node mobility, failures, or variable density while maintaining strict delay constraints. Integrating AI-driven or reinforcement learning–based routing into OMNeT++ could improve scalability and resilience.

7.3. Energy-Aware Deployment of Detection Models

Further studies could examine lightweight YOLO variants running directly on sensor nodes, balancing detection accuracy with energy consumption. Research should focus on combining model pruning or quantization with duty-cycling strategies to extend node lifetime.

7.4. Resilience and Security in Safety-Critical Scenarios

The framework should be tested under adverse conditions such as packet loss, node compromise, or malicious activity. Future work may explore cryptographic or trust-based mechanisms to ensure secure and reliable alert delivery.

7.5. Toward Digital Twin Integration

An important step forward is extending this pipeline into a digital twin framework, combining live detection data with predictive simulations. This would allow proactive evaluation of system behavior under scenarios such as sudden congestion or multi-vehicle collisions.

References

[1] A. Dadashev, Á. Török, SmartDENM—a system for enhancing pedestrian safety through machine vision and V2X communication, *Electronics* 14(5) (2025) 1026. https://doi.org/10.3390/electronics14051026

[2] C. Saha, T.H. Tran, S. Syamal, Enhancing automotive safety through advanced object behaviour tracking for intelligent traffic and transport system, in: *Proc. IEEE Int. Workshop on Metrology for Automotive (MetroAutomotive)*, Bologna, Italy, (2024) 70-75. https://doi.org/10.1109/MetroAutomotive61329.2024.10615654

[3] S.N. Thakur, A. Sinha, M. Sikarwar, B. Kumar, K. Bhardwaj, T. Raj, R. Kundu, A. Alkhayyat, Wireless sensor network based integrated LEACH protocol with YOLO and feed-forward ResNet for detecting real-time forest fire to promote environmental sustainability, in: *Proc. 2024 4th Int. Conf. on Innovative Practices in Technology and Management (ICIPTM)*, Noida, India, (2024) 1-6. https://doi.org/10.1109/ICIPTM59628.2024.10563730

[4] A. Fascista, Toward integrated large-scale environmental monitoring using WSN/UAV/crowdsensing: a review of applications, signal processing, and future perspectives, *Sensors* 22(5) (2022) 1824. https://doi.org/10.3390/s22051824

[5] M.N. Mowla, N. Mowla, A.F.M.S. Shah, K.M. Rabie, T. Shongwe, Internet of things and wireless sensor networks for smart agriculture applications: a survey, *IEEE Access* 11 (2023) 145813-145852. https://doi.org/10.1109/ACCESS.2023.3346299

[6] M. Kamarunisha, K. Jayasri, S. Sharin, M. Priyadharshini, T. Anusuya, R. Malathi, Implementation of YOLO-based object detection and sensor integration model for visually challenged person, in: *Proc. 2024 Int. Conf. on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, Chennai, India, (2024) 1-7. https://doi.org/10.1109/ICSES63760.2024.10910328

[7] M. Rychlicki, Z. Kasprzyk, M. Pełka, A. Rosiński, Use of wireless sensor networks for area-based speed control and traffic monitoring, *Applied Sciences* 14(20) (2024) 9243. https://doi.org/10.3390/app14209243

[8] A. Hazarika, N. Choudhury, M.M. Nasralla, S.B.A. Khattak, I.U. Rehman, Edge ML technique for smart traffic management in intelligent transportation systems, *IEEE Access* 12 (2024) 25443-25458. https://doi.org/10.1109/ACCESS.2024.3365930

[9] N. Murali, D.D. Stalin, Advanced VANET technology for emergency vehicle safety in time-critical scenario, in: *Proc. 2023 Eighth Int. Conf. on Science Technology Engineering and Mathematics (ICONSTEM)*, Chennai, India, (2023) 1-8. https://doi.org/10.1109/ICONSTEM56934.2023.10142332

[10] A. Hazarika, N. Choudhury, M.M. Nasralla, S.B.A. Khattak, I.U. Rehman, Edge ML technique for smart traffic management in intelligent transportation systems, *IEEE Access* 12 (2024) 25443-25458. https://doi.org/10.1109/ACCESS.2024.3365930

[11] S.A. Dhinakar Raj, A. Singh, T.S. Shekhawat, M. Muthanna, K. Sivasankaran, Collision avoidance system using YOLO-based object detection and distance estimation, in: *Proc. 2025 3rd Int. Conf. on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, Bengaluru, India, (2025) 178-184. https://doi.org/10.1109/IDCIOT64235.2025.10915167

[12] H. Ouyang, T. Li, C. Wang, Y. Gu, F. Yang, K. Deng, Improving road defect detection precision and efficiency with structural pruning techniques, in: *Proc. 2024 21st Int. Computer Conf. on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, Chengdu, China, (2024) 1-6. https://doi.org/10.1109/ICCWAMTIP64812.2024.10873743

[13] J. Zhu, Y. Wu, T. Ma, Multi-object detection for daily road maintenance inspection with UAV based on improved YOLOv8, *IEEE Transactions on Intelligent Transportation Systems* 25(11) (2024) 16548-16560. https://doi.org/10.1109/TITS.2024.3437770

[14] N. Rout, G. Dutta, V. Sinha, A. Dey, S. Mukherjee, G. Gupta, Improved pothole detection using YOLOv7 and ESRGAN, *arXiv* (2023).

[15] S.F.M. Radzi, M.A.A. Rahman, M.K.A.M. Yusof, N.S.M. Haniff, R.F. Rahmat, Computationally enhanced UAV-based real-time pothole detection using YOLOv7-C3ECA-DSA algorithm, *IEEE Access* 13 (2025) 99092-99111. https://doi.org/10.1109/ACCESS.2025.3573651