iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85527

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

AI-Enabled Intelligent Traffic Control with Emergency Vehicle Detection: A Design-Stage Framework

Deepaly Thorat[1], Dipti Bhawar[2], Ragini Dighe[3], Anushka Kakade[4], Sudhanshu Badgujar[5]
Department of Electronics and Computer Engineering (ECE), Sir Visvesvaraya Institute Of Technology, Nashik
Savitribai Phule Pune University, Maharashtra, India

Abstract: ***Fixed-time traffic signals cannot respond to changing vehicle demand or to the urgent movement of emergency vehicles, and ambulances in particular can lose critical minutes at congested intersections. This paper proposes an AI-enabled Intelligent Traffic Control with Emergency Vehicle Detection framework that uses YOLOv8 to detect ambulances, fire trucks, and police vehicles from camera video, estimates lane-wise traffic demand, and requests priority for the emergency vehicle's lane through a controlled signal transition. Detection results travel through an ESP32 communication unit to an STM32 controller that runs a finite-state signal machine. At the same time, a MERN-stack dashboard shows live counts, signal states, alerts, and detection history. We review an earlier IoT-based base paper and related studies, define the design problem, and describe the architecture, decision logic, message format, safety rules, and fallback behaviour. The work is at the design stage, so no measured results are claimed. Instead, we set out an evaluation plan covering detection quality, response latency, waiting time, communication reliability, and signal safety. The framework is a starting point for an LED-signal prototype, not for use on public roads.***

Keywords: Intelligent Transportation System; YOLOv8; Adaptive Traffic Signal; Emergency Vehicle Prioritization; ESP32; STM32; Traffic Monitoring Dashboard .

I. INTRODUCTION

Road networks in growing cities carry more vehicles than their signal plans were designed for. A fixed-time signal gives each approach a preset share of green time, so a lightly used approach may hold the green while a long queue waits on another. The cost of this becomes serious when an ambulance, fire truck, or police vehicle reaches an intersection, because a few lost minutes can change the outcome of a medical or rescue call, as the base paper of this work argues [1].

Cameras and object-detection models offer a way to estimate traffic conditions without special equipment on every vehicle. The base paper [1] detects ambulances with RF, sound, or vision sensors on Arduino or Raspberry Pi hardware and creates a green corridor for them. A 2026 study [2] uses YOLO-based detection with an Arduino signal controller. Both point to difficulties that any camera-based design must face: lighting and obstruction, communication delay, safe signal transitions, and recovery after the emergency vehicle has passed. This paper proposes a design that builds on these studies. YOLOv8 running on a host computer detects vehicles and emergency-vehicle classes. An ESP32 carries the results to an STM32 controller that runs the signal state machine, and a MERN-stack dashboard (MongoDB, Express,

React, Node.js) shows live status. The design stresses safe behaviour: a low-confidence detection never overrides the signals on its own, priority is applied through a controlled transition, and a defined fallback applies when communication fails. The work is at the design stage, and we make no claims about measured performance.

II. LITERATURE SURVEY

A. The Base Paper: IoT-Based Emergency Signal Control

Nithyashree et al. [1] describe an IoT-based traffic signal system that gives ambulances priority. An RF transmitter inside the ambulance is the primary detection method, and sound sensors and camera-based recognition (YOLO or ResNet) are offered as additions. Infrared or ultrasonic sensors count vehicles, an Arduino or Raspberry Pi controller runs the override logic, and LEDs represent the signals in a scaled-down prototype. The authors also suggest MQTT-based reporting to a central dashboard and list evaluation metrics: detection accuracy, response time, reliability, traffic impact, and scalability.

©IJRASET: All Rights are Reserved

Their abstract refers to simulations and prototypes that show reduced ambulance waiting time, but the text we read gives no numerical results, so the size of the benefit cannot be judged.

The literature review in [1] lists limits of earlier work: no real-time ambulance detection, dependence on vision under poor conditions, heavy infrastructure needs, no coordination across intersections, limited fault tolerance, and weak secure communication.

B. Vision-Based Priority Using YOLO

Mohan et al. [2] describe a smart signal system with YOLO-based vehicle detection, a decision engine for traffic density, an ambulance override, and Arduinobased signal execution. They report tests on recorded traffic video and note that real-time deployment, model accuracy, and hardware capability remain challenges. Their ambulance priority uses a fixed-duration phase, which can hold the green longer than needed or end it before the vehicle has passed. This motivates a clearancechecked release with a timeout as a backstop.

C. Other Related Approaches

As summarised in [1], Aworinde et al. [3] adjusted signal timing from traffic volume on Arduino hardware but had no real-time emergency detection, and Chong et al. [4] used image processing for density-based control without ambulance detection. Djahel et al. [5] combined RFID, GPS, and vehicular ad hoc networks to prioritize emergency vehicles, which reduced delay but required heavy infrastructure. Jimenez-Moreno et al. [6] paired ResNet-50 with fuzzy logic to recognize emergency vehicles, with sensitivity to image quality and parameter tuning. Zerroug et al. [7] proposed ADSTLS, a visionbased adaptive system with fault-tolerant control that is complex to maintain. Vani et al. [8] gave priority to emergency vehicles in a density-based image-processing model that depended on environmental conditions, and Omina [9] used fuzzy logic for urban traffic without full ambulance preemption. We describe these works only as summarised in [1] and did not re-read the originals.

D. Detection and Embedded Platforms

YOLO [10] treats detection as a single-pass regression problem and runs fast enough for video, which suits perframe vehicle counting. OpenCV [11] supplies frame capture and image handling. The ESP32 [12] is a low-cost Wi-Fi microcontroller suited to message transfer, and STM32 devices [13] offer the deterministic timing that a signal state machine needs.

E. The Gap

Taken together, the sources leave several gaps that this paper tries to address in its design:

- RF detection [1] needs a transmitter in every emergency vehicle, while camera detection needs no on-vehicle equipment but is limited by lighting and occlusion.
- A fixed-duration priority phase [2] ignores whether the vehicle has actually passed.
- Message validation, stale-message rejection, and fallback behaviour are described only at a high level in the sources we read.
- The two closest papers [1], [2] focus on ambulances, and dashboards are described only briefly.

This paper does not claim to close these gaps. It proposes a design that targets them and a plan to test whether it does.

III. METHODOLOGY / APPROACH

A. Study Approach

This is a design-stage study. The design draws on the base paper [1], the 2026 study [2], and the project's working plan. No experiments were run, and no hardware or software performance was measured, so everything below is a proposal to be implemented and tested. Values such as thresholds and green-time limits are placeholders to be tuned.

B. System Architecture

The system has five layers: video acquisition, AI detection and traffic estimation, communication, embedded decision and signal actuation, and dashboard monitoring (Fig. 1). A camera supplies video to a host computer that runs the YOLOv8 pipeline with Python and OpenCV. The host summarises each frame as vehicle classes, counts, confidence values, and approach information. It sends the data to an Express.js API, which stores records in MongoDB and serves the React dashboard, and it sends control messages to an ESP32 [12] over Wi-Fi. The ESP32 passes validated messages to an STM32 [13] over a wired link such as UART, which the prototype will fix. The STM32 runs the signal state machine and drives the LEDs.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The MERN components handle the dashboard, APIs, and database; they do not perform detection. In a minimal first prototype, the ESP32 can be left out, and the host can talk to the STM32 over USB serial.

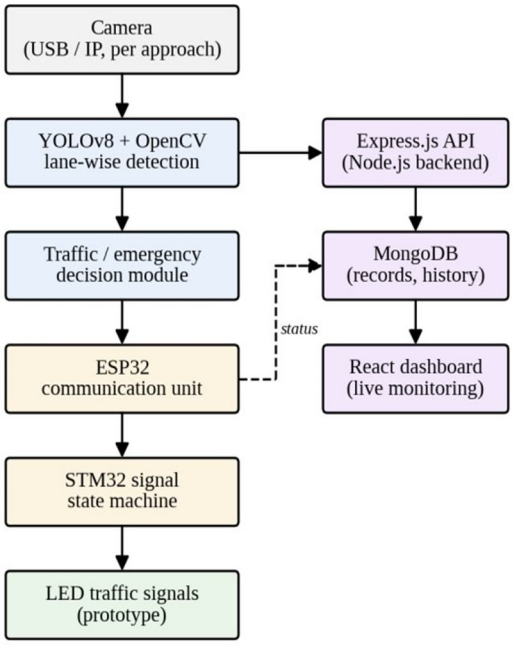


Fig. 1. Proposed functional flow and data paths.

C. Vehicle Detection and Traffic Estimation

YOLOv8 [14] processes each frame and returns a class, a confidence score, and a bounding box for every detection. The detector targets common vehicle classes and three emergency classes: ambulance, fire truck, and police vehicle. A region of interest is assigned to each approach, so detections inside it count toward that approach, and a tracker or counting line can reduce double counting. The output is an estimate: low-confidence or partly hidden detections must not trigger a signal change on their own. A model for the emergency classes needs suitable training images, including night and bad-weather scenes, and whether a public dataset is enough is still open.

D. Adaptive Signal Decision

In normal operation, the controller computes a demand score for each approach, $Di = \alpha Ni + \beta Qi + \gamma Wi(1)$ where N_i is the observed vehicle count, Q_i is an estimated queue measure, W_i is the time since the approach last had green, and α, β, γ are tunable weights. The approach with the highest eligible score receives green, subject to minimum and maximum green times, a yellow interval, and an all-red clearance interval. Equation (1) is a proposed formulation; its weights and limits need tuning and validation.

E. Emergency Vehicle Prioritization

An emergency request for approach i is raised only when an emergency class is detected there with confidence at least τ for k consecutive frames, $Ei(t) = 1 if ci(t - m) \geq \tau, m = 0, 1, \ldots, k - 1(2)$

where $c_i(t)$ is the highest emergency-class confidence in approach i at frame t. The controller then checks the current phase, ends it through yellow and all-red, and gives green to approach i (Fig. 2). Priority is released when a clearance condition is met, for example, no emergency detection in the approach for a set time after the vehicle leaves the region of interest, or when a configured timeout expires. Normal adaptive control then resumes. If the approach or the confidence is uncertain, the controller keeps normal operation and raises an operator alert instead of overriding. When emergency vehicles appear at several approaches at once, a configured policy decides the order, for example, earliest confirmed request first; the best policy is left open.

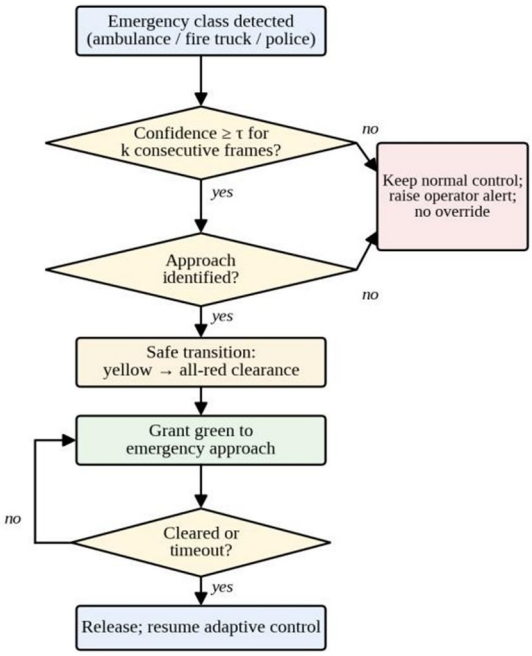


Fig. 2. Emergency-priority decision flow.

F. ESP32–STM32 Communication

The ESP32 handles data transfer while the STM32 keeps deterministic control. Table 1 lists the proposed message fields. The STM32 rejects malformed, stale, or duplicate messages. A communication timeout puts the controller into a predefined safe fallback, such as normal adaptive or fixed-time operation, and sends a dashboard notification. The exact protocol and interface are implementation choices to be fixed during prototyping.

Table 1. Proposed control message fields.

Field	Purpose
Message ID / sequence	Detect duplicate or missing messages
Approach ID	Identify the target approach
Vehicle counts	Support the demand estimate
Emergency flag and class	Priority request and vehicle type
Confidence / validity	Gate for any override
Timestamp	Reject stale messages
Integrity check	Detect corrupted messages

G. Signal Actuation and Safety

The STM32 runs a finite-state machine for the signal phases with green, yellow, and red states, and it never lets conflicting movements have green together. Emergency preemption is a state transition, not a direct switch. In the prototype, LEDs represent the lights. A student prototype must not be connected to operational traffic signals or toll barriers. Any real deployment would need certified controllers, safety engineering, and authorization from the road authority, and lies outside this paper.

H. MERN Dashboard

MongoDB stores detection records, density, signal events, and alerts. Express and Node.js expose the APIs, and React shows the interface summarised in Table 2. Authenticated access and audit logs are proposed for administrative actions, and any dashboard command must still pass the STM32's safety rules.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Table 2. Proposed dashboard sections.

Section	Information shown
Dashboard home	Overall system status
Live camera	Current video, if streamed
Vehicle detection	Detected vehicle type and count
Emergency alert	Vehicle type, approach, confidence
Lane monitoring	Lane-wise traffic density
Signal status	Red, yellow, or green per lane
Detection history	Earlier detections with timestamps
Statistics	Total vehicles and emergency detections

I. Components

Table 3 lists the main hardware. The software stack is Python, YOLOv8, and OpenCV for detection; Node.js, Express.js, React, and MongoDB for the dashboard; STM32CubeIDE for the controller; and Postman for API testing. The final choice of camera, interface, power supply, and boards must be checked against availability and implementation needs.

Table 3. Main hardware components.

Component	Role	Status
USB webcam or IP camera	Live traffic video	Required
Laptop / PC	Runs YOLOv8 and dashboard	Required
STM32 board	Signal control	Required
ESP32 board	Wireless communication	Optional for minimal prototype
LEDs, resistors, breadboard, wires	Prototype signals	Required
Road model, servo and barrier arm	Multi-lane model; tollbarrier demo	Optional

IV. EXPECTED OUTCOME

A. Current Status

The project is at the design stage. No detection accuracy, waiting-time reduction, latency, or reliability value has been measured, and none is claimed. Any figures reported by the earlier papers belong to those papers and are not evidence for this design.

B. Comparison with Previous Research

Table 4 sets the sources side by side, using the descriptions in [1] and [2] for the works cited there.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Table 4. Related studies and points relevant to this design.

Study	Approach	Point relevant here
[1]	RF primary, optional sound and vision; Arduino or Raspberry Pi; scaled prototype	Needs on-vehicle transmitter; no numerical results in the text
[2]	YOLO vehicle detection; Arduino; recorded video	Fixed-duration priority phase; real-time use untested
[3]	Volume-based timing; Arduino	No real-time emergency detection
[4]	Image-processing density control	No ambulance detection
[5]	RFID, GPS, and VANET	Heavy infrastructure
[6]	ResNet-50 with fuzzy logic	Sensitive to image quality and tuning
[7]	Vision-based ADSTLS	Complex to implement and maintain
[8]	Density-based image processing	Depends on environment
[9]	Fuzzy-logic control	No full ambulance preemption
This work	YOLOv8; ESP32 and STM32; MERN dashboard	Not yet implemented or tested

Compared with [1], the proposed design replaces onvehicle RF with camera detection. This removes the need for a transmitter in every emergency vehicle but makes the system depend on visibility. Compared with [2], it releases priority on a clearance check with a timeout instead of a fixed interval, and it adds message validation and a dashboard. These are differences in design, not demonstrated advantages.

C. Planned Evaluation

Table 5 groups the planned metrics. Emergency response time is

$$Tresp = tsignal - tdetect \quad (3)$$

where *tdetect* is the time the emergency detection is confirmed, and *tsignal* is the time green appears on the emergency approach. The proposed controller will be compared with a fixed-time baseline on the same video scenarios or prototype conditions, using the relative change in average waiting time,

$$\Delta W = (Wfixed - Wadaptive) / Wfixed \times 100\% \quad (4)$$

Results will be reported only after testing, with test conditions and sample sizes stated.

Table 5. Planned evaluation metrics.

Level	Metrics
Detection	Precision, recall, F1-score, confusion matrix per class
Timing	Inference time per frame; detection-to-signal latency; transition duration
Traffic	Average waiting time, queue length, throughput, green-time use
Reliability	Packet success rate, timeouts, recovery, continuousrun stability
Safety	No conflicting greens; yellow and all-red intervals respected

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

D. Challenges and Limitations

Camera-based detection is sensitive to lighting, weather, and obstruction [1]. Fire-truck and police-vehicle appearance varies by region, so a custom training set may be needed. The path from host to controller adds delay that must be measured. Only a single intersection is considered, so green corridors across junctions are not addressed. An LED prototype does not capture real signal hardware, driver behaviour, or pedestrian phases. The dashboard and message path need proper authentication. Finally, the weights and thresholds will affect results and must be tuned fairly for both the adaptive design and the baseline.

V. CONCLUSION

This paper presented a design-stage framework for AIenabled Intelligent Traffic Control with Emergency Vehicle Detection. The design joins YOLOv8 detection, ESP32 communication, STM32 signal control, LED prototype outputs, and a MERN monitoring dashboard. It defines a demand-based decision rule, a confidence-gated emergency workflow with clearance-checked release, and rules for message validation, timeouts, and safe phase changes. Because nothing has been implemented or measured yet, the contribution is architectural and methodological. Building the prototype and testing it against a fixed-time baseline are needed before any performance claim can be made.

VI. FUTURE SCOPE

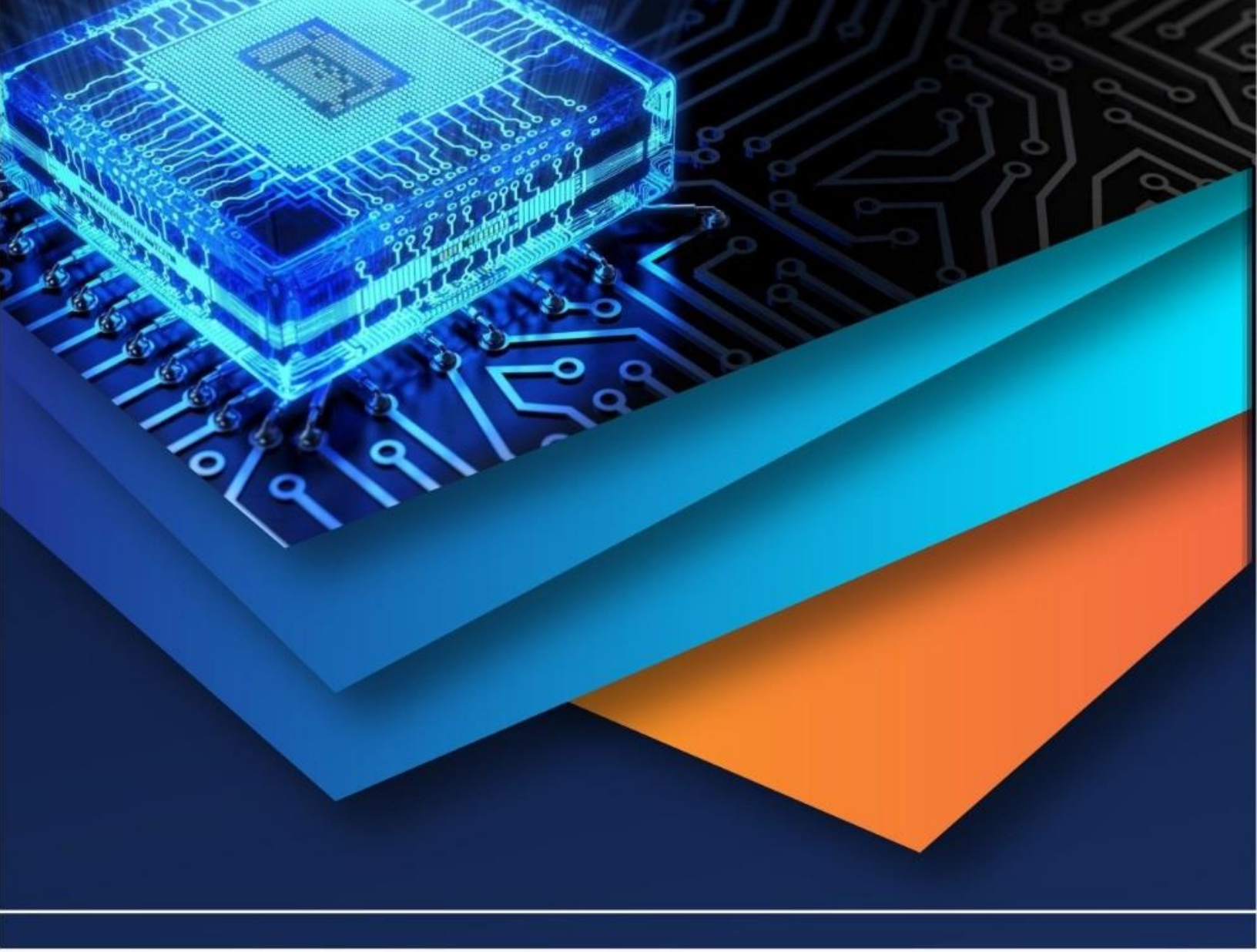
Implement the detection and embedded-control pipeline and test it in the three stages described above.
Collect day, night, and bad-weather video, including fire trucks and police vehicles, and tune the timing parameters.
Test communication fault handling and validate on a scaled intersection prototype.
Extend to multi-intersection coordination and GPS-based emergency route information.
Add secure device authentication and encrypted communication, and optimize edge inference.
Explore predictive traffic analysis and an optional toll-plaza barrier demonstration.

Limitations and benefits. This work has three limitations: it is a design with no experiments, its parameters are untested, and it covers one intersection. Its benefits are a complete camera-to-controller architecture with explicit safety and fallback rules, a clear evaluation plan, and a direct link to the base paper [1] that shows which of that paper's limits it targets.

REFERENCES

[1] Nithyashree R, Manoj HK, Huzaif Ahmad, Mohammed Haris, and Manohar M, "Smart IoT-Based Traffic Signal Control System with Emergency Vehicle Detection," International Journal of Innovative Research in Technology (IJIRT), vol. 12, issue 2, Jul. 2025.
[2] Abhinand Mohan, Akash V, Asiya Shafi, and Farsana AS, "Smart Traffic Signal System using AI-Based Vehicle Detection and Emergency Prioritization," International Journal of Engineering Research & Technology (IJERT), vol. 15, issue 03, Mar. 2026.
[3] H. O. Aworinde, O. O. Osunmakinde, and T. L. Afolabi, "Development of a Prioritized Traffic Light Control System for Emergency Vehicles," IJERT, vol. 8, no. 4, pp. 237–241, Apr. 2019.
[4] B. Chong, M. B. Mustaffa, A. W. A. Wahab, and N. A. Rahim, "Smart Traffic Light Control System Using Image Processing," in Proc. IEEE Int. Conf. on System Engineering and Technology (ICSET), 2016, pp. 8–12.
[5] S. Djahel, R. Doolan, G. Muntean, and J. Murphy, "A Communications-Oriented Perspective on Traffic Management Systems for Smart Cities: Challenges and Innovative Approaches," IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 125–151, 2015.
[6] R. Jimenez-Moreno et al., "Ambulance Detection for Smart Traffic Light Applications with Fuzzy Controller," Journal of Ambient Intelligence and Humanized Computing, vol. 11, no. 10, pp. 4307–4318, Oct. 2020.
[7] A. Zerroug, A. Guermazi, and H. H. Ben Ghezala, "Adaptive and Dynamic Smart Traffic Light System for Efficient Management of Regular and Emergency Vehicles at City Intersection," Procedia Computer Science, vol. 170, pp. 708– 713, 2020.
[8] R. Vani and B. S. Aithal, "Intelligent Traffic Control System with Priority to Emergency Vehicles," International Journal of Computer Applications, vol. 116, no. 23, pp. 7–12, Apr. 2015.
[9] J. A. Omina, "An Intelligent Traffic Light Control System Based on Fuzzy Logic Algorithm," International Journal of Computer Applications, vol. 179, no. 27, pp. 25–30, Mar. 2018.
[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
[11] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)