Astik Sarkar, 2026, 14:6
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Vision-Based Cyber-Physical Traffic Signal Control Using YOLOV8 Multi-Parameter Congestion Scoring With Arduino UNO And ESP32–MQTT Prototypes

Astik Sarkar
School Of Computer Science And Engineering Vellore Institute Of Technology
Vellore, India Astik.Sarkar2024@Vitstudent.Ac.In

Abstract—Vehicles at urban crossings frequently linger at a red light while the crossing road is empty because fixed-time signal systems are unable to adjust to the way traffic demand varies throughout the day. We introduce a camera-based Cyber-Physical System (CPS) in this study that completes the loop between the signal head and the road scene. Traffic footage within an operator-selected region of interest (ROI) is processed by YOLOv8 and OpenCV. A weighted congestion score is created by combining three indicators—vehicle count, occupancy ratio, and queue depth—that are obtained from the detections. A guarded rule assigns a HIGH or LOW state and provides a microcontroller with the corresponding phase plan when the score, count, and queue all hit their respective criteria. Two hardware variations are powered by the same software pipeline: an ESP32 that subscribes to commands via MQTT and an Arduino UNO that is provided via a USB serial link. Infrared car sensors are not used in either version. In bench tests, a run with 12 detected vehicles, an occupancy ratio of 0.51, and a queue depth of 168.6 pixels yielded a score of 0.46, was marked HIGH, and created a plan on the LED signal head that was 20 seconds green, 2 seconds amber, and 10 seconds red. The project is now at the prototype stage; further work will include field tests as well as quantitative accuracy and latency assessments.

Keywords: YOLOv8, congestion estimation, ESP32, Arduino UNO, MQTT, intelligent transportation systems, adaptive traffic signal control, and cyber-physical systems

I. INTRODUCTION

Similar to other expanding metropolitan areas, the number of cars in Indian cities has increased more quickly than the capacity of the roads. Longer travel times, increased fuel use, and increased tailpipe emissions at every busy intersection are the results. Although the formation and propagation of such queues are explained by traffic-flow theory [3], the majority of installed controllers continue to function on pre-timed cycles. Pre-timed cycles waste time and road space by giving an approach the same green time whether ten vehicles are queued or none. Adaptive control substitutes a cycle that reacts to measured demand for the fixed cycle. Inductive loops or infrared detectors installed next to or buried within each lane are the foundation of traditional adaptive systems. These sensors only provide a single point of presence, require civil work to install and maintain, and don't provide much information regarding the length of a wait. In contrast, a single camera can be relocated without the need for road construction and can watch an entire approach at once. It is now feasible to extract vehicle positions from such video in real time on standard computer hardware thanks to developments in deep-learning detectors.

In order for a digital decision to change a physical process, a Cyber-Physical System combines sensing, processing, communication, and actuation [1]. CPS research views this connection as the foundation of several autonomous and networked systems [2]. The concept naturally translates to a signalized intersection: a microcontroller changes the lamps, a computer estimates the traffic condition, a communication link transmits the decision, and a camera and detector make up the sensing layer. Building and demonstrating such a loop at bench scale is the aim of this work. Its specific goals are to: (i) identify vehicles from video using YOLOv8 within a configurable ROI; (ii) characterize congestion using three complementary indicators rather than a single count; (iii) fuse the indicators into a guarded HIGH/LOW decision and a weighted score; and

© 2026 Astik Sarkar, This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

(iv) demonstrate that the same decision layer can drive two distinct controller and communication paths, a networked ESP32–MQTT link and a wired Arduino UNO link. Any behavioral differences between the two prototypes can be traced to the controller and its link rather than the detector because the analysis code is shared. The work is linked to the sustainability objectives of smart-city infrastructure because shorter idle times at crossings also lower the amount of fuel burned while waiting.

II. THE MAIN CONTRIBUTIONS OF THIS PAPER ARE AS FOLLOWS

- a five-layer CPS architecture that allows for the independent replacement of each layer by separating perception, traffic analysis, decision-making, communication, and actuation;
- a cautious decision rule that rejects a single deceptive indicator in conjunction with a multi-parameter congestion score that incorporates vehicle count, occupancy ratio, and queue depth;
- two functional hardware prototypes running the identical analysis and decision software, one via MQTT to an ESP32 and the other via USB serial to an Arduino UNO; and
- An explanation of the stability, safety, and calibration procedures required to advance the prototypes toward field use.

This is how the remainder of the paper is structured. Related work is reviewed in Section II. The technique, which includes the indicators, the congestion score, and the decision rule, is explained in Section III. The two prototypes are shown in Section IV. The results and conclusions are presented in Section V, design decisions, constraints, and extensions are covered in Section VI, and the paper is concluded and future work is outlined in Section VII.

III. RELATED WORK

Lee [1] and Rajkumar et al. [2] were the first to identify the design issues of connecting computation with physical processes. The

traffic-flow models that link density, flow, and queue expansion are presented by Treiber and Kesting [3]. In terms of perception, YOLO made real-time detection possible by reformulating object detection as a single regression pass [4], and the Ultralytics YOLOv5 codebase transformed it into a popular toolkit [5]. Since then, YOLO-family detectors have been used in moving-object traffic monitoring [9], speed estimates [8], model comparison for vehicle identification [7], signal-timing optimization [6], and vehicle categorization [10]. YOLOv8 and YOLOv10 have been benchmarked against one another [12], modified YOLOv8 networks enhance detection in challenging scenarios [11], [14], and the YOLOv8-Nano form has been investigated for hardware with constrained processing power [13]. A road segment can be continually monitored using counting and tracking pipelines built on YOLOv8 [16], [17], and YOLOv8 has also been utilized for more extensive monitoring in intelligent transportation [19]. The edge-deployed controller of Kazmi et al., which determines signal timings by combining YOLO detection with passenger-car-equivalent density, is the work most similar to ours [15]. Additionally, YOLOv11-based flow optimization [18] and AI-assisted adaptive monitoring [20] have emerged. The focus of typical studies is summarized in Table I. The majority of this research ends with simulation, counting, or detection. The current work is different in three ways: it drives actual signal lamps; it reuses one decision layer across two controller and communication options; and it scores congestion by combining count, occupancy, and queue depth.

Ref.	Main task	Traffic measure	Signal output
[7], [12]	Detector comparison	Detections	None
[8]	Speed estimation	Vehicle speed	None
[10], [11]	Detection, classification	Detections, class	None
[13]	Lightweight detection	Detections	None
[16], [17]	Counting	Vehicle count	None
[15]	Adaptivesignal timing	PCE-based density	Signal timing
[18]	Flow optimization	Detections, flow	Signal timing

Ref.	Main task	Traffic measure	Signal output
This work	Closed-loop CPS	Count,occupancy, queue	LED signal via two controllers

TABLE I. Focus of Representative Related Studies

IV. METHODOLOGY

1. *System Architecture*

The five layers of the system are perception, traffic analysis, decision, communication, and actuation, as shown in Fig. 1. The operator annotates the ROI that includes the technique under study once a computer gets recorded or live footage. In that regard, YOLOv8 recognizes cars while OpenCV handles frame reading, geometry, annotation, and display. The resulting indicators are supplied to a scoring module, which then selects a signal plan. The plan is then transmitted by USB serial to an Arduino UNO (Prototype I) or an ESP32 (Prototype II) via a MQTT broker. Microcontrollers do not run neural networks; instead, the computer handles the intricate calculations. The processing flow for Prototype I is shown in Fig. 2, and the pipeline for Prototype II is shown in Fig. 3.

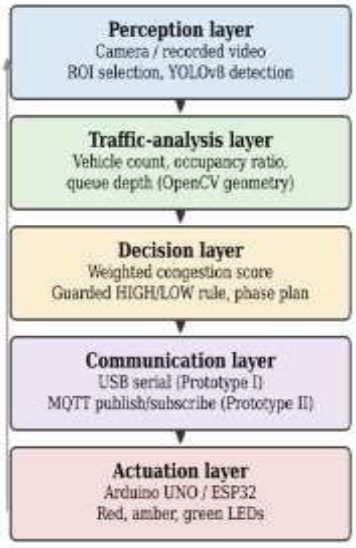


Figure 1 shows the suggested system's five-layer CPS architecture. The response from the physical signal to the observed traffic scene is shown by the arrow on the left.

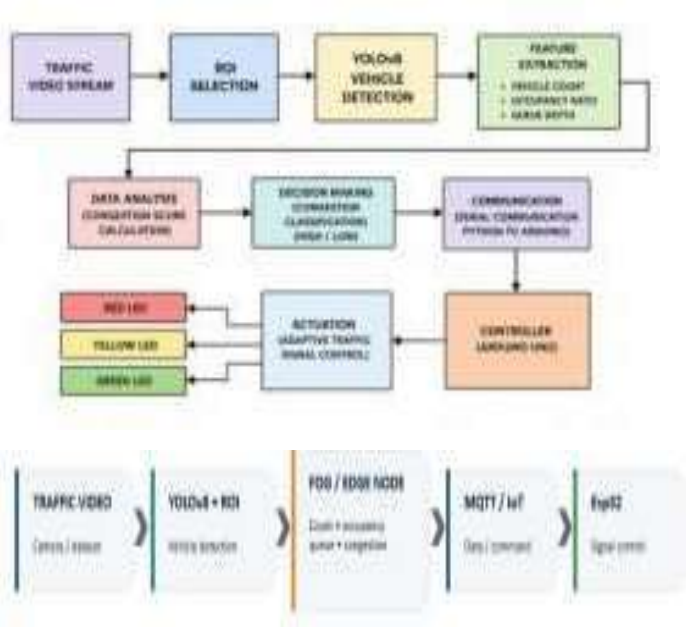


Fig. 1. Processing flow of Prototype I, from video input and ROI selection to LED actuation through the Arduino UNO.

Figure 2 shows the prototype II pipeline, where a fog/edge node calculates the indications and sends commands to the ESP32 via MQTT.

There is a useful advantage to this layer separation. The controller firmware does not need to know how congestion was measured, nor does the perception and analysis code need to know which controller is connected. Therefore, it is possible to upgrade one side without rewriting the other, for instance by switching to a different microcontroller or a newer detector.

2. Video Acquisition And Roi Selection

A camera or a recorded file is used to read video frames. The operator draws a rectangular ROI around the lanes of the approach that needs to be controlled on the first frame. By limiting the analysis to this area, one program can adjust to various camera heights and road layouts while excluding walkways, buildings, and opposing lanes. The stop position from which the queue is measured is indicated by a reference line inside the ROI. If the camera is moved, the ROI and reference line must be repainted because they are fixed for a particular camera view.

3. Vehicle Detection

Only four vehicle classes—car, motorcycle, bus, and truck—remain after applying YOLOv8 to every frame. When a bounding box's center is inside the ROI and its confidence is higher than the selected threshold, it is approved. The acceptable boxes in the current analysis window are indicated by B = {b1,..., bN}. According to recent vehicle-monitoring research [10]–[14], detection quality naturally changes with viewing angle, lighting, occlusion, resolution, and model weights. Larger YOLOv8 models can be utilized when accuracy is more crucial than speed by running the detector on the computer instead of the microcontroller, which simplifies the embedded side.

4. Traffic Indicators

The approach's current condition is described by three indications. The number of acceptable detections, or the size of B, is represented by the vehicle count N. The percentage of the ROI area occupied by car boxes is known as the occupancy ratio.

Or = min(1, Σi A(bi ∩ R) / A(R)) (1) where A(·) represents area in pixels and R is the ROI.

Overlapping boxes are prevented by the maximum limit of 1.

A bus makes a greater contribution than a motorcycle since occupancy increases with both the quantity and size of vehicles. The distance in pixels between the reference line and the farthest identified vehicle in the queue is known as the queue depth, or dq:

dq = maxi dist(bi, ℓ) (2)

It shows the extent to which the build-up has spread upstream. A few buses and numerous motorcycles cannot be distinguished by a count alone; occupancy and queue depth provide a more comprehensive picture of the same situation.

5. Normalization And Weighted Congestion Score

Each indicator is normalized to the range of 0–1 prior to weighting. While the count and queue depth are split by reference values Nref and Lref, which are fixed for the camera view, the occupancy ratio is already a fraction: Qd = min(1, dq/Lref)

and Vc = min(1, N/Nref) (3) The congestion score CS is :

CS = 0.20Vc + 0.40Or + 0.40Qd (4) when Or is taken straight from (1).

Because occupancy and queue depth represent the actual amount of road space used, they both have a weight of 40%, whereas the raw count carries a weight of 20%. A single threshold is important across runs because the weights add up to one and each term falls between 0 and 1, which also affects the score. Additionally, the weighting restricts the impact of any one indicator: occupancy or queue depth alone can increase CS by up to 0.40, whereas count alone can increase CS by up to 0.20.

Every camera view has its reference values set once. Taking Lref, which is the ROI's length along the direction of travel measured from the reference line, and Nref, which is the number of vehicles the ROI can accommodate when the approach is completely queued, is a sensible option. With these options, Vc and Qd only go close to 1 when the approach's visible portion is full, and the score remains similar even when the ROI is resized.

6. *Guarded Decision Rule*

The following criteria must all be met simultaneously for an approach to be classified as HIGH:

CS ≥ 0.30 AND N ≥ 5 AND dq ≥ 130 px (5)

LOW is the term for any other combination. One inflated indicator cannot alter the state on its own if all three conditions are met. For instance, a single long truck close to the stop line can result in a high occupancy, but the approach is still LOW if the count is less than five. Similarly, a long queue depth may result from a few dispersed cars distant from the stop line, but the score and count prevent the decision from changing. The behavior of the rule in typical instances is shown in Table II. The camera and ROI geometry determine the 130 px threshold, which needs to be adjusted for a new view.

Scenario	CS ≥ 0.30	N ≥ 5	dq ≥ 130 px	State
Dense queue at stop line	Yes	Yes	Yes	HIGH

Scenario	CS ≥ 0.30	N ≥ 5	dq ≥ 130 px	State
Single long truck	Possible	No	Possible	LOW
Many vehicles, short queue	Possible	Yes	No	LOW
Light, fr ee-flowing traffic	No	Possible	No	LOW

TABLE II. Behavior of the Guarded Rule in Typical Cases

7. *Signal Phase Sequencing*

As illustrated in Fig. 4, the decision module chooses the appropriate phase plan and transmits it to the controller once the state is known. The controller uses a 20-second green, 2-second amber, and 10-second red scheme for the HIGH state. The signal never jumps straight between go and stop since amber is displayed in both transitions—between green and red and between red and green. Every new analysis window closes the CPS loop by refreshing the indicators and possibly generating a new command. The signal output influences the traffic that the camera sees in the subsequent window. Algorithm 1 summarizes the entire process.

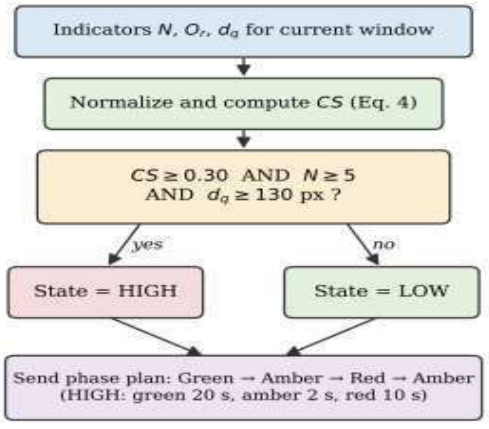


Fig. 2. Decision flow from the traffic indicators to the phase plan sent to the controller.

Algorithm 1: Vision-based adaptive signal control loop

Input: video stream, ROI *R*, reference line ℓ, *Nref*, *Lref*

Output: phase plan sent to the controller 1: for each analysis window do

2: read frame; run YOLOv8 detection

3: keep car, motorcycle, bus, truck boxes with center in *R*

4: compute *N*, *Or* by (1) and *dq* by (2) 5: normalize by (3); compute *CS* by (4)

6: if condition (5) holds then state ← HIGH 7: else state ← LOW

8: select phase plan for state (green → amber → red → amber) 9: send plan via USB serial (Prototype I) or MQTT (Prototype II) 10: controller drives the LEDs; display overlay and summary

11: end for

V. PROTOTYPE IMPLEMENTATIONS

1. *Common Software Pipeline*

One Python application based on OpenCV and the Ultralytics YOLOv8 package is shared by both prototypes. ROI selection, detection, indication extraction, scoring, the decision rule, and the on-screen overlay are all managed by the program. The output stage is the only one that varies; it either broadcasts the chosen plan to a MQTT topic or sends it to a serial port. The two prototypes are directly comparable because they maintain the same pipeline.

2. Prototype I: Arduino Uno Over Usb Serial

ROI selection, detection, indication extraction, scoring, and the decision rule are all carried out using a laptop in Prototype I. The chosen plan is sent to an Arduino UNO via a USB cable as a brief serial message. After decoding the message, the Arduino uses current-limiting resistors to power red, amber, and green LEDs that are installed on a breadboard (Fig. 7). There is no need for an additional supply because the board is powered by the USB wire. Although this configuration is simple to construct and troubleshoot, the distance between the computer and the controller is constrained by the need for a cable.

3. Prototype Ii: Esp32 Over Mqtt

Video input, ROI handling, detection, scoring, and the decision rule are all unchanged in Prototype II. The final link is the only difference: an ESP32 subscribing to a MQTT topic parses each message and adjusts its LED outputs once the computer publishes the chosen plan to that subject (Fig. 8). For Internet of Things devices, MQTT is a lightweight publish/subscribe protocol. The vision program does not need to know the controller's address because a broker stands in between the publisher and the subscribers. Later on, additional controllers, a remote dashboard, or a cloud logger can be added without changing the analysis code. Depending on the network, the broker may operate locally or remotely. Analytics and cloud storage are suggested but not yet put into practice. Table III compares the two variations.

Attribute	Prototype I	Prototype II
Perception	YOLOv8 + OpenCV	YOLOv8 + OpenCV
Indicators	Count, occupancy, queue	Count, occupancy, queue
Decision logic	Eqs. (4) and (5)	Eqs. (4) and (5)
Controller	Arduino UNO	ESP32
Link	USB serial (wired)	MQTT (Wi-Fi)
Coupling	Direct, point to point	Via broker, decoupled
IR vehicle sensors	None	None
Actuation	Red/amber/green LEDs	Red/amber/green LEDs
Scalability	Local, single unit	Networked, IoT-ready

TABLE III. Comparison of the Two Prototypes

4. Results And Findings

5. *Test Procedure*

A multi-lane metropolitan road's recorded traffic footage was used to test both prototypes on the bench. The application processed the video, the operator drew the ROI and reference line on the first frame of each run, and the attached controller received the final plan. Four features of each prototype were examined: the computation and display of indicators, the issuance of decisions, the controller's receipt of commands, and the switching of LEDs in the ordered sequence.

A. Summary of Detection Output and Analysis The on-screen overlay for a single frame is displayed in Fig. 5. The running count, occupancy, and queue depth are presented at the top, while accepted boxes are indicated in green, the ROI in red, and the signal reference line in blue. These values vary as cars enter and exit the ROI because they are recalculated for each frame. The program

sends the plan to the controller and prints a summary at the conclusion of each run (Fig. 6).

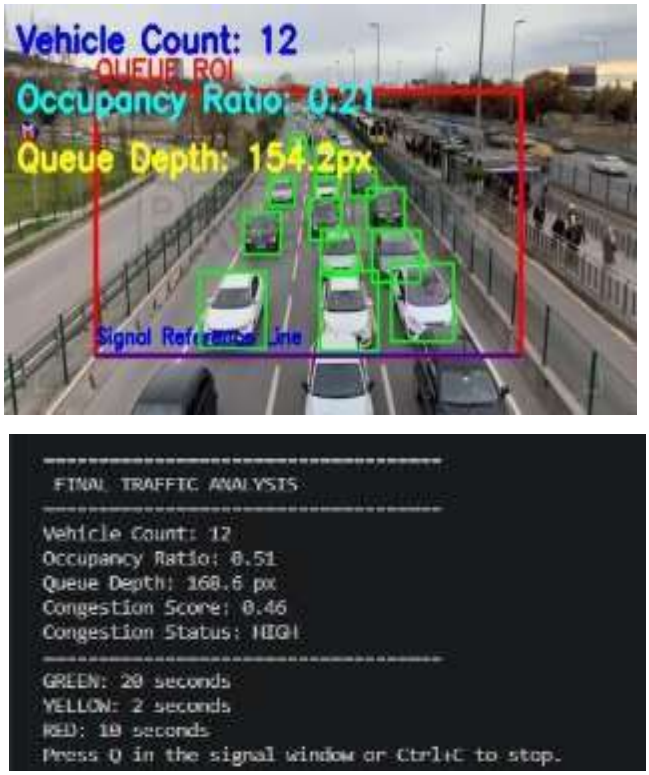


Figure 3: Annotated detection output with live traffic indicators, the ROI, and the signal reference line.

Figure 3 shows the final analysis summary for Run B, including the issued signal timings, congestion status, and calculated score.

6. Observed Runs

Two observed runs are listed in Table IV. Since all three of the requirements listed in (5) were met in both

situations, the state was HIGH and the 20/2/10 s plan was provided. For instance, in Run B, the queue depth of 168.6 px is roughly 30% above the 130 px criterion, the score of 0.46 is clearly more than 0.30, and the count of 12 is more than double the minimum of five. As a result, the choice is not affected by slight variations in any one indicator from frame to frame.

Quantity	Run A	Run B
Vehicle count N	13	12
Occupancy ratio Or	0.39	0.51
Queue depth dq (px)	160.8	168.6
Congestion score CS	0.44	0.46
Condition (5) satisfied	Yes	Yes
State	HIGH	HIGH
Green / amber / red (s)	20 / 2 / 10	20 / 2 / 10

TABLE IV. Observed Outputs from Two Bench Runs

7. Hardware Actuation

Both boards' LEDs switched in the order that was instructed. The Arduino signal head moving through amber, red, amber, and green is depicted in Fig. 7, while the ESP32 board driving the same sequence via MQTT commands is shown in Fig. 8. Together, they verify that a video-based decision is transmitted to the physical signal via either link. The functional checks are compiled in Table

I. The shared decision layer is independent of the controller and communication path, as demonstrated by the four checks that both prototypes passed.

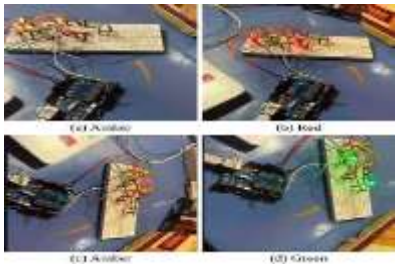


Fig. 3. Prototype I (Arduino UNO) stepping through the signal cycle: (a) amber, (b) red, (c) amber and (d) green.

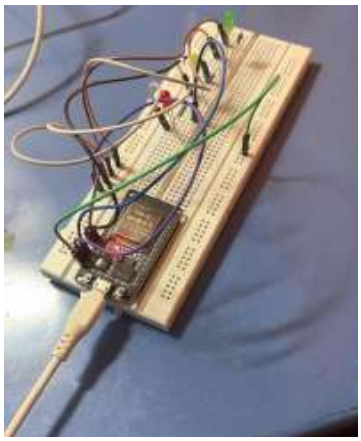


Fig. 4. Prototype II: ESP32 on a breadboard driving the LED signal head from MQTT commands.

Function checked	Prototype I	Prototype II
Indicators computed and displayed	Pass	Pass
HIGH/LOW decision issued	Pass	Pass
Command received by controller	Pass	Pass
LEDs switched in commanded order	Pass	Pass

TABLE V. Functional Verification of the Prototypes

These findings are not a statistical assessment; rather, they show proper functioning behavior. At this point, no claims are made regarding detection precision and recall, frame rate, command latency, MQTT delivery rate, power consumption, or real-world delay reduction.

VI. DISCUSSION

1. Choice Between The Two Prototypes

The prototypes can be directly compared since perception and scoring are the same in both builds. For a classroom or lab demonstration, the Arduino route is faster because it doesn't require a network and the serial link is easy to watch while debugging. Because other devices can join the same broker and messages are detached from the sender, the ESP32 approach is more appropriate for networked intersections. In order

to deploy the processing unit in a cabinet or at a nearby edge node, it also cuts the cable that connects the computer to the signal head.

2. Measurement In Image Space

Although two of its inputs are assessed in picture space, the three-indicator score provides more information than a count. Thresholds cannot be replicated between sites since a queue of 130 pixels corresponds to varied road lengths for various camera heights and angles. This restriction would be eliminated and the same thresholds may be used at each intersection by mapping pixels to meters using a per-camera calibration, such as a homography fitted to recognized lane markers. It makes sense to tune the weights in (4) on labeled footage since they were selected by design rather than fitted to data.

3. Decision Stability

The state may alternate between successive windows when the score is close to 0.30. Hysteresis, which uses a larger threshold to enter the HIGH state than to exit it, and smoothing the indications over multiple frames are two easy fixes. Enforcing a minimum length for each plan would help avoid abrupt shifts that can cause confusion for drivers.

4. Comparison With Fixed-Time Control

Table VI contrasts the proposed approach with conventional fixed-time control and with point-sensor actuated control. The main advantages of the vision-based design are that it observes the whole approach rather than a single point, that it measures how far the queue extends, and that its decisions can be supervised remotely in the MQTT variant. Its main costs are the need for a computer at or near the junction and the sensitivity of video detection to weather and lighting.

Aspect	Fixed-time	Point sensor	Proposed
Responds to demand	No	Partly	Yes
Queue length known	No	No	Yes, in pixels
Vehicle size reflected	No	No	Yes, via occupancy
Road works to install	None	Per lane	Camera mount only
Remote supervision	No	Depends	Yes (Prototype II)
Weather sensitivity	None	Low	Moderate

TABLE VI. Fixed-Time, Point-Sensor and Proposed Control

5. Extension To A Full Intersection

The prototypes control one approach at a time. A full four-way intersection can be handled by drawing one ROI and reference line per approach, computing a congestion score for each, and allocating longer green intervals to the approaches with higher scores within fixed minimum and maximum limits. With MQTT, each signal head can subscribe to its own topic, so the same broker can serve every approach. Coordinating neighboring intersections would follow the same pattern, with each junction publishing its scores so that upstream signals can anticipate arriving platoons.

6. Safety And Deployment

The LED setup is not a certified signal controller and must not be connected to public road equipment. A deployable version would need conflict-free phase interlocks, a safe default state, watchdog recovery, handling of lost links, message timestamps with rejection of stale commands, and authenticated MQTT traffic. If the camera view is blocked or the link fails, the controller should fall back to a conservative fixed-time plan.

7. Sustainability And Privacy

Replacing in-road sensors with a camera avoids repeated excavation, and reducing idle time at red lights lowers fuel use and emissions. Because the video is processed on the local computer and only short control messages leave it, no images need to be transmitted over the network. This reduces bandwidth and limits the exposure of identifiable information about road users.

VII. CONCLUSION AND FUTURE SCOPE

A camera-based CPS for adaptive signal control was designed and demonstrated with two controller options. A single YOLOv8 pipeline extracts vehicle count, occupancy ratio and queue depth, fuses them into a weighted congestion score, applies a guarded HIGH/LOW rule and issues a phase plan. Prototype I delivers that plan to an Arduino UNO over USB serial,

and Prototype II delivers it to an ESP32 over MQTT; neither uses infrared vehicle sensors. Bench tests confirmed that a congestion decision made from video reached and switched the physical lamps through both paths.

Future work will test both prototypes on the same footage and report detection precision and recall, inference rate, end-to-end latency and MQTT recovery after disconnection.

Queue depth will be calibrated in meters, the weights of the congestion score will be tuned on labeled data, and hysteresis will be added to the decision rule.

Trials will then be extended to several approaches and intersections. Finally, the ESP32–MQTT design will serve as the basis for a cloud dashboard, historical analytics and coordinated multi-junction control.

VIII. ACKNOWLEDGMENT

We would like to thank the Vellore Institute of Technology for providing the laboratory facilities to create and test the prototypes. Authors' contributions A. Sarkar created and implemented the software pipeline and both hardware prototypes, performed the bench testing and produced the manuscript; A. Victor supervised the work, directed the approach and examined the manuscript.

REFERENCES

1. E. A. Lee, "Cyber physical systems: Design challenges," in Proc. 11th IEEE Int. Symp. Object-Oriented Real-Time Distributed Computing (ISORC), Orlando, FL, USA, 2008, pp. 363–369, doi: 10.1109/ISORC.2008.25.
2. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in Proc. 47th Design Automation Conf. (DAC), 2010, pp. 731–736, doi: 10.1145/1837274.1837461.
3. M. Treiber and A. Kesting, Traffic Flow Dynamics: Data, Models and Simulation. Berlin, Germany: Springer, 2013, doi: 10.1007/978-3-642-32460-4.
4. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
5. G. Jocher et al., "YOLOv5," Ultralytics, 2020. [Online]. Available: https://github.com/ultralytics/yolov5
6. K. Harding and W. Baxter, "Optimizing traffic light control using YOLOv8 for real-time traffic management," IEEE, 2024, doi: 10.1109/ICDV61346.2024.10616901.
7. R. Sharma and M. Jain, "Comparative analysis of YOLOv5 and YOLOv8 for vehicle detection," IEEE, 2024, doi: 10.1109/SMART63812.2024.10882512.
8. V. Ceronmani Sharmila, A. Gosain, and A. R. Srivastava, "Real-time speed estimation in urban traffic using YOLOv8," IEEE, 2024, doi: 10.1109/ICMNWC63764.2024.10872042.
9. A. Singh, M. Z. Ur Rahman, P. Rani, N. K. Agrawal, R. Sharma, and E. Kariri, "Smart traffic monitoring through real-time moving object detection," IEEE Trans. Consum. Electron., vol. 70, no. 4, 2024, doi: 10.1109/TCE.2024.3445728.
10. J. Castillo, R. Rosculete, R. G. Babayans, A. Merzoian, K. Groves, and G. Calva, "A real-time traffic monitoring system based on YOLOv8 for vehicle detection and classification," in Proc. IEEE Global Engineering, Science and Technology Conf. (GESS), 2024, doi: 10.1109/GESS63533.2024.10784465.
11. X. Liu, Y. Wang, D. Yu, and Z. Yuan, "YOLOv8-FDD: A real-time vehicle detection method based on improved YOLOv8," IEEE Access, vol. 12, pp. 136280–136296, 2024, doi: 10.1109/ACCESS.2024.3453298.
12. A. S. Geetha, M. A. R. Alif, M. Hussain, and P. Allen, "Comparative analysis of YOLOv8 and YOLOv10 in vehicle detection," Vehicles, vol. 6, no. 3, 2024, doi: 10.3390/vehicles6030065.
13. M. Bakirci, "Real-time vehicle detection using YOLOv8-Nano for intelligent transportation systems," Traitement du Signal, vol. 41, no. 4, 2024, doi: 10.18280/ts.410407.

14. J. Klikowski and M. Woźniak, "Vehicle detection and tracking based on improved YOLOv8," IEEE Access, 2025, doi: 10.1109/ACCESS.2025.3538556.
15. M. Kazmi, H. M. Kidwai, H. R. Khan, S. A. Qazi, and K. Arshad, "An edge-deployed real-time adaptive traffic light control system using YOLO-based vehicle detection and PCE-aware density estimation," IEEE Access, 2025, doi: 10.1109/ACCESS.2025.3602844.
16. X. Duan, S. Xue, and Z. Zhou, "Design and implementation of road vehicle counting based on YOLOv8," ACM, 2025, doi: 10.1145/3711129.3711275.
17. M. M. Rahman, U. Ayman, M. A. Hossain, and A. Sohel, "Real-time vehicle type detection and counting using YOLOv8," Emerging Science Journal, vol. 9, no. 2, 2025, doi: 10.28991/ESJ-2025-09-02-023.
18. F. M. Talaat, R. M. Elbalka, S. Sweidan, and S. A. Gamel, "Smart traffic management system using YOLOv11 for real-time vehicle detection and dynamic flow optimization in smart cities," Neural Comput. Appl., 2025, doi: 10.1007/s00521-025-11434-9.
19. "Utilizing YOLOv8 for enhanced traffic monitoring in intelligent transportation systems," Digital Signal Processing, Elsevier, 2024. [Online]Available:https://www.sciencedirect.com/science/article/pii/S 105120042400594X
20. "Intelligent traffic monitoring and management for real-time adaptive signal control," SSRN Working Paper, 2025. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5597770