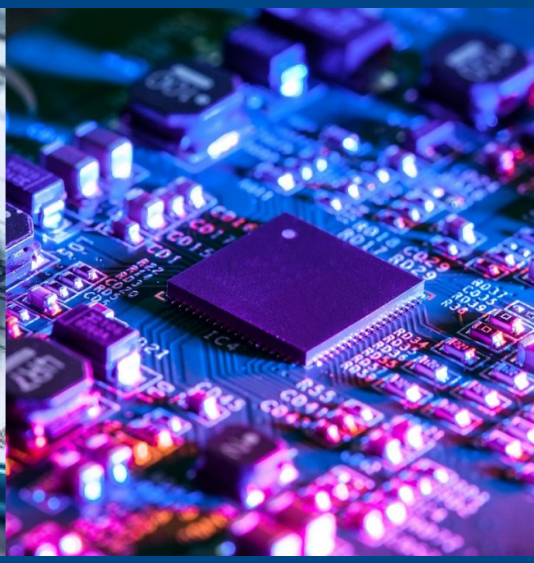
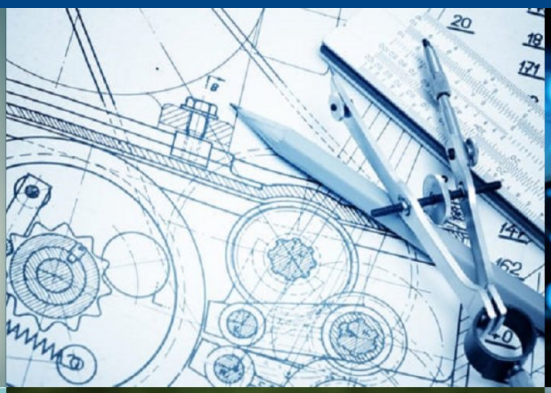
ISSN: 2582-7219

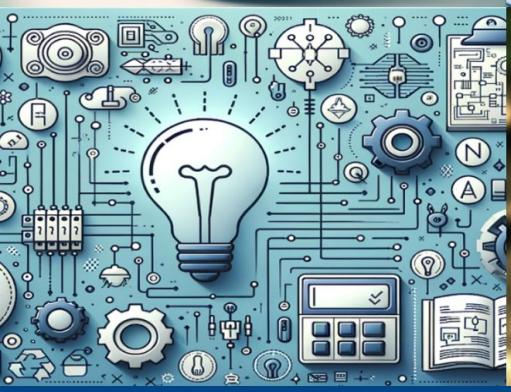




International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206 **Volume 9, Issue 4, April 2026**

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904434

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Smart Traffic Signal Management Using Image Pixel Density Classification and ESP32-Based System

Syed Mohammad Aslam, Syed Abhutakir A, Mohamed Sharif S B, Suhail Rashid N,Dr. K. Rajeswari,

Department of Electronics and Communication Engineering Aalim Muhammad Salegh College of Engineering, Chennai, Tamil Nadu, India

Supervisor, Associate Professor, Department of Electronics and Communication Engineering Aalim Muhammad Salegh College of Engineering, Chennai, Tamil Nadu, India

ABSTRACT: Traffic congestion in urban areas has long been one of the most stubborn challenges city planners face. Beyond the obvious frustration of sitting at red lights, it quietly drains fuel, raises emissions, and chips away at road safety every single day. The traffic signal systems most cities still rely on were designed for a different era — they run on fixed timers set once during installation and simply don't know what's happening on the road right now. This paper introduces a Smart Traffic Signal Management System (STSMS) that takes a practical, low-cost approach to fixing that problem. Instead of costly sensors or complex AI models requiring heavy computation, we use something much simpler: the pixel density of camera images as a stand-in for how busy a lane actually is.Two ESP32-CAM modules are mounted at a road intersection, each watching one lane and streaming live video wirelessly over WebSocket to a backend server. Every five seconds, the server grabs a frame from each camera, runs a quick pixel analysis to figure out which lane is busier, and adjusts the traffic signals accordingly. The busier lane gets a longer green light; the quieter one waits its turn — but not indefinitely. A built-in fairness rule steps in to guarantee that no lane sits on red for more than 60 seconds without getting at least a 15-second green window. On top of all this, a live web dashboard lets operators see both camera feeds, density readings, and signal countdowns in real time. The whole system was built for under $20 in parts, runs without any GPU, and can plug into cameras already installed at intersections — making it genuinely practical for cash-strapped municipalities.

KEYWORDS: Smart Traffic Management, ESP32, ESP32-CAM, Image Pixel Density, WebSocket, Adaptive Signal Control, Embedded Systems, Real-Time Traffic Monitoring, IoT Traffic Systems.

I. INTRODUCTION

Walk through the streets of any major Indian city during peak hours and the problem is immediately obvious. Vehicles queue at intersections for minutes on end, even when the crossing lane is nearly empty. The reason is simple: most traffic signals still operate on rigid pre-set cycles that have no awareness of actual traffic flow. They were configured based on historical data and haven't been meaningfully updated since.

The Internet of Things (IoT) has quietly changed what's possible here. Affordable microcontrollers like the ESP32 which packs a dual-core processor, built-in Wi-Fi, and Bluetooth into a chip that costs just a few dollars — make it realistic to build distributed sensing systems at road intersections without breaking the budget. Pair one with an ESP32-CAM module and you have a functional wireless camera node for under five dollars.

That's exactly what this project builds on. Rather than going down the route of computationally expensive object detection models that demand GPU hardware or cloud connectivity, we developed a method that uses the density of non-background pixels in a frame as a rough but effective measure of how congested a lane is. The system covers two opposing lanes of an intersection, dynamically adjusting signal durations based on live pixel analysis, while a web interface gives operators full visibility of what's happening at the junction.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. PROBLEM STATEMENT

The core issue with conventional traffic signal systems isn't hard to identify. They're set up once, based on traffic surveys conducted at a point in time, and then left to run on the same schedule regardless of what's actually happening on the road. This creates some predictable and frustrating outcomes:

- Lanes sitting nearly empty still get full green-light durations, while the busy lane piles up.
- Emergency vehicles have no way to trigger signal preemption because the system has no awareness of them.
- Intersections near schools, markets, or event venues see wildly asymmetric traffic at certain times — and fixed-timing systems handle this poorly.
- Smarter adaptive systems using inductive loop detectors or radar do exist, but they're expensive to install and maintain, and scaling them up is not straightforward.

What's missing is an affordable, infrastructure-light solution that can read real-time conditions and respond accordingly — without requiring specialized hardware or ongoing maintenance contracts.

III. LITERATURE REVIEW

Research into adaptive traffic signal control goes back several decades. Webster's foundational 1958 work [1] laid out mathematical frameworks for optimal signal cycle lengths — still referenced today, despite being inherently static in nature. Vehicle-actuated control, which uses inductive loop detectors to trigger signal changes when vehicles are detected at stop lines, represented an early step toward responsiveness [2].

Camera-based traffic analysis opened up new possibilities as hardware became cheaper. Buch et al. [3] surveyed the landscape of computer vision methods for traffic monitoring, covering everything from background subtraction to machine learning detection pipelines. Kastrinaki et al. [4] looked specifically at motion-based segmentation for surveillance applications, highlighting the persistent tension between accuracy and real-time performance.

Deep learning changed the ceiling for what's possible. YOLO, introduced by Redmon et al. [5], made real-time multi-class object detection feasible for the first time, and follow-up work applied YOLO variants to traffic management with genuinely impressive results [6]. But these systems need hardware like the NVIDIA Jetson or reliable cloud connectivity — raising real concerns about deployment cost and latency. IoT-based approaches have tried to bridge this gap. Karthik et al. [7] built a Raspberry Pi system using OpenCV for vehicle counting. More recently, Sharma and Patel [8] demonstrated a prototype using IR sensors on ESP32 hardware, though without any visual density estimation.

IV. PROPOSED SYSTEM

The STSMS is designed around four guiding principles: keep it wireless, keep it lightweight, make it fair, and make it transparent. By wireless, we mean no signal cables between cameras and the processing unit — everything runs over Wi-Fi. Lightweight means the pixel density method can run on an ordinary laptop or small server, no GPU needed. Fair means a lane can never be left on red indefinitely. Transparent means operators can see exactly what the system is doing through the web dashboard.

V. SYSTEM ARCHITECTURE

The architecture breaks down into four layers stacked on top of each other: sensing, communication, processing, and presentation. Figure 1 below illustrates the complete system architecture with data flow between each component.

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904434

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

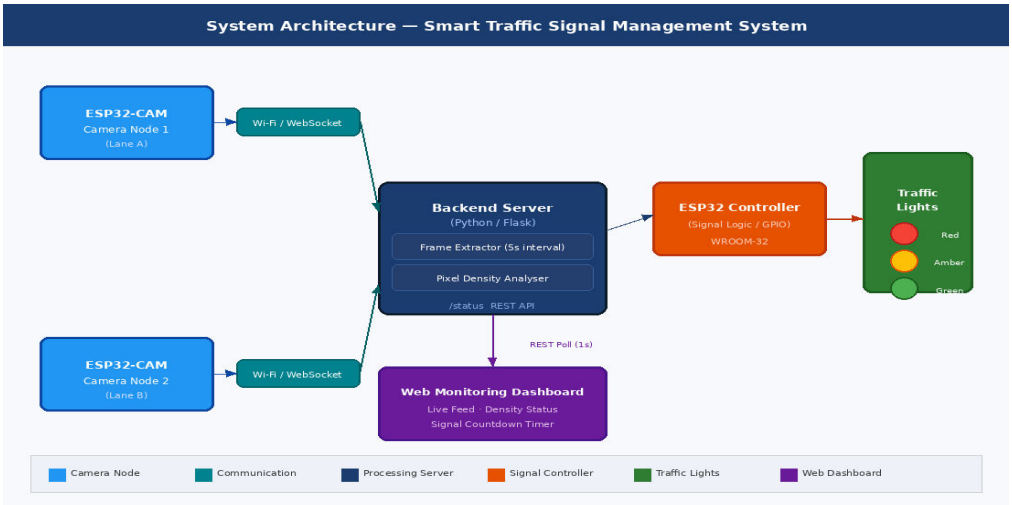


Figure 1: System Architecture Block Diagram — Smart Traffic Signal Management System

5.1 Sensing Layer

Each lane of the intersection gets its own ESP32-CAM module. These devices stream continuous MJPEG video over a dedicated WebSocket connection to the backend server, all on a local Wi-Fi network.

5.2 Communication Layer

WebSocket (RFC 6455) handles the data link between cameras and server. The persistent connection it maintains makes it well-suited to continuous streaming from embedded hardware — unlike HTTP polling, there's no overhead from repeatedly opening and closing connections.

5.3 Processing Layer

The backend runs on Python using Flask and OpenCV. It pulls one frame every five seconds from each camera stream, preprocesses it, and computes a pixel density score. That score feeds directly into the signal control logic, which decides which lane gets green and for how long, then sends the appropriate command to the ESP32 signal controller.

5.4 Presentation Layer

A simple web dashboard shows live video from both cameras, a density status label for each lane (Low, Moderate, or High), and a countdown timer for the current signal phase. It talks to the backend via REST API, polling for updates every second.

VI. METHODOLOGY

6.1 Image Pixel Density Classification

The working assumption is straightforward: more vehicles in a lane means more non-background pixels in the camera frame. The preprocessing pipeline goes like this — frame capture every five seconds, grayscale conversionto reduce dimensionality, Gaussian blur (5×5 kernel) to smooth noise, adaptive binary thresholding to separate foreground from background, and finally the density score D = (N_foreground / N_total) × 100%.

Table I: Pixel Density Classification and Signal Actions

Density Score (D)	Classification	Signal Duration
D < 20%	Low Traffic	15 seconds
20% – 50%	Moderate Traffic	30 seconds
D ≥ 50%	High Traffic	60 seconds

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904434

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

6.2 Signal Control Logic

The controller runs a two-state machine — either Lane A has green, or Lane B does. Density comparison determines which lane goes next and for how long. The fairness constraint overrides this logic if either lane has been sitting on red for over 60 consecutive seconds, forcing a minimum 15-second green phase before normal density-based operation resumes. Figure 2 below shows the complete signal control flowchart.

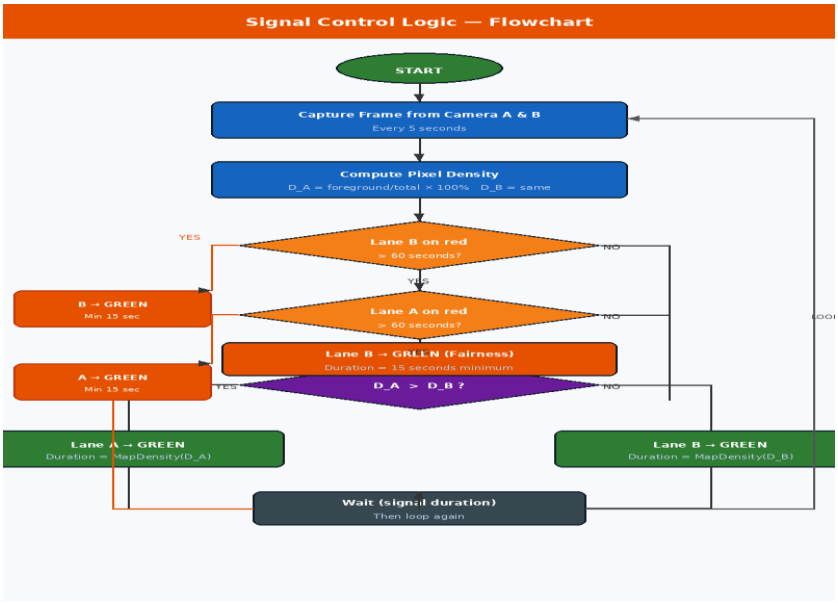


Figure 2: Signal Control State Machine — Flowchart with Fairness Logic

VII. HARDWARE COMPONENTS

Figure 3 illustrates the complete hardware connection diagram of the system, showing how the ESP32-CAM modules, backend server, signal controller, and traffic light board are interconnected.

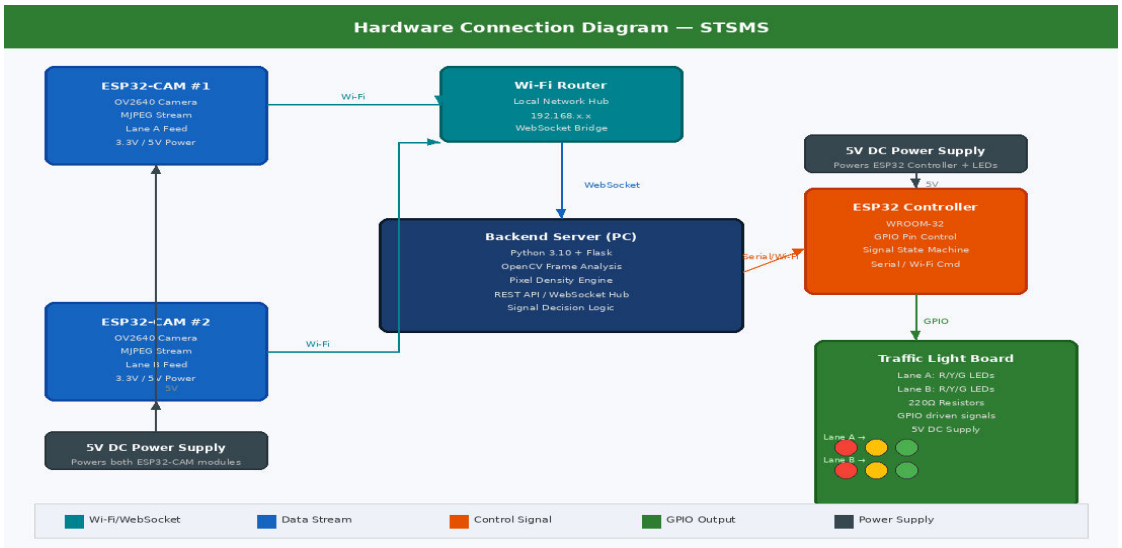


Figure 3: Hardware Connection Diagram — STSMS Component Layout

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Table II: Hardware Components and Specifications

Component	Specification / Role	Qty
ESP32 (WROOM-32)	Signal controller; drives traffic LEDs via GPIO	1
ESP32-CAM (AI-Thinker)	OV2640 2MP camera; MJPEG streaming over WebSocket	2
Traffic Signal LEDs	Red, Yellow, Green LEDs simulating traffic lights	6
Resistors (220fi)	Current limiting for LEDs	6
Wi-Fi Router	Local network for WebSocket communication	1
5V DC Supply	Powers ESP32 and ESP32-CAM modules	2
Backend Server (PC)	Python Flask server; frame analysis; signal decisions	1

VIII. SOFTWARE IMPLEMENTATION

8.1 ESP32-CAM Firmware

The firmware is written using the Arduino framework for ESP32, with the esp32-camera library handling video capture and ArduinoWebSockets managing the communication link. On startup, the device initializes the camera, connects to the local Wi-Fi network, opens a WebSocket connection to the backend, and begins transmitting JPEG-encoded frames at roughly 2 frames per second.

8.2 Backend Server

The server runs Python 3.10, using Flask, websockets, OpenCV (cv2), and NumPy. Frame buffers are maintained for both cameras. A background thread wakes every five seconds to compute density scores, update the signal state,
and refresh the /status endpoint, which returns a JSON payload containing density labels, current signal state, and countdown timer.

8.3 Web Dashboard

The monitoring interface is a single-page application in HTML5, styled with Tailwind CSS, and driven by plain JavaScript. Live video is rendered via the MJPEG stream endpoints; density badges and countdowns update from the /status API every second.

IX.ALGORITHM DESIGN

9.1 Pixel Density Computation

ALGORITHM: ComputePixelDensity(frame)
INPUT : Raw JPEG frame (bytes)
OUTPUT: Density score D (float, 0.0 – 100.0)
1. Decode JPEG bytes to BGR image array I
2. Convert I to grayscale image G
3. Apply Gaussian blur: G_blur = GaussianBlur(G, 5x5)
4. Binary threshold: mask = 1 if G_blur[x,y] > T else 0
5. N_foreground = count(mask == 1)
6. N_total = width(G) × height(G)
7. D = (N_foreground / N_total) × 100.0
8. RETURN D

9.2 Signal Control State Machine

ALGORITHM: SignalControlLoop()
INIT: active_lane='A'; FAIRNESS=60s; MIN_GREEN=15s
LOOP every 5 seconds:
D_A = ComputePixelDensity(frame_A) D_B = ComputePixelDensity(frame_B)
IF time_since_green_B >= FAIRNESS: active='B'; duration=MIN_GREEN ELIF time_since_green_A >= FAIRNESS:

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904434

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

active='A'; duration=MIN_GREEN ELIF D_A > D_B: active='A'; duration=MapDensity(D_A)
ELSE: active='B'; duration=MapDensity(D_B) SendCommand(active, duration) Wait(duration)
END LOOP

X. RESULTS AND DISCUSSION

The prototype was tested under three controlled scenarios. When both lanes had roughly equal density (around 30%), the system settled into an alternating rhythm with 30-second green phases on each side. In Scenario 2, where Lane A was heavily loaded (~65%) and Lane B was nearly empty (~8%), Lane A correctly received a 60-second green, and when the fairness timer expired, Lane B was granted its guaranteed 15-second phase. Scenario 3 tested dynamic switching: as density values shifted, the system updated its decision within the 5-second analysis window.Pixel density computation for a single VGA (640×480) frame took roughly 18 milliseconds on the test server — fast enough to have no meaningful impact on the 5-second decision cycle. The web dashboard streamed video with a 1–2 second lag and showed no synchronization issues.

Table III: Observed System Behavior Under Test Scenarios

Scenario	D_A (%)	D_B (%)	Result
Equal Density	~30%	~28%	Alternating 30s each
A Heavy, B Light	~65%	~8%	A: 60s; B: 15s fairness
Dynamic Switch	Varies	Varies	Switched within 5s

XI.ADVANTAGES OF THE SYSTEM

a. Cost-Effectiveness: The full prototype came together for under $20 — a fraction of what inductive loop detector installations typically run.
b. Infrastructure-Compatible: Works with pre-installed roadside cameras, making deployment far cheaper in cities that already have camera coverage.
c. Fully Wireless: Wi-Fi handles everything between cameras and server, eliminating the need for signal cables at intersections.
d. Built-In Fairness: No lane can be suppressed indefinitely — the system guarantees a minimum green window regardless of relative density.
e. Live Monitoring: Traffic authorities can watch feeds, density readings, and signal countdowns remotely through the web dashboard.
f. Scalable Design: The modular architecture can extend to three or four lanes with straightforward additions to the codebase.
g. Low Overhead: No GPU required. The entire processing stack runs comfortably on a modest server or desktop PC.

XII. LIMITATIONS

Like any prototype, the system has limitations that are worth being upfront about. The fixed binary threshold used in pixel classification is sensitive to lighting changes — performance under low-light conditions or at night would likely degrade without modification. Perspective distortion is another issue: vehicles closer to the camera naturally occupy more pixels, which can overweight their contribution to the density score. The current implementation is also constrained to exactly two opposing lanes. Vehicle type is entirely ignored — a single parked bus contributes the same pixel mass as several motorbikes. Finally, the centralized backend is a single point of failure.

XIII. FUTURE SCOPE

13.1 Deep Learning-Based Vehicle Detection

Swapping pixel density for YOLOv8 or MobileNet SSD would give the system genuine vehicle counting with type awareness, eliminating false positives from non-vehicle objects.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

13.2 Traffic Flow Prediction

The density readings the system already collects are time-series data — a natural input for an LSTM or Transformer model trained to forecast near-future congestion, shifting the system from reactive to proactive.

13.3 Emergency Vehicle Detection

A practical extension would combine audio classification for siren detection with visual recognition of emergency vehicle light patterns, triggering automatic signal preemption.

13.4 Edge Computing Deployment

Moving the backend processing onto an NVIDIA Jetson Nano or Raspberry Pi 5 mounted at the intersection would remove the network dependency entirely, with much lower latency.

13.5 Multi-Intersection Coordination

Coordinating signals across adjacent junctions via a cloud platform would enable green-wave progression along corridors — a proven way to reduce stop-and-go traffic.

13.6 Adaptive Threshold Calibration

An automated calibration routine adjusting the binary threshold based on rolling histogram analysis would make the system robust to time-of-day lighting changes and different weather conditions.

XIV. CONCLUSION

This paper set out to demonstrate that adaptive traffic signal control doesn't have to be expensive or computationally demanding to be genuinely useful. The Smart Traffic Signal Management System we built shows that a $20 hardware setup, a standard Python server, and a simple pixel density metric can together produce a system that responds meaningfully to real traffic conditions — prioritizing congested lanes, enforcing fairness for quiet ones, and giving operators a clear window into what's happening at the junction.The experimental results confirmed that the system correctly prioritizes heavy lanes, adapts its decisions within five seconds of changing conditions, and operates the web dashboard without any visible desynchronization. For resource-constrained municipalities looking for a practical first step toward smarter intersections, this system offers a credible starting point. The next natural steps — deep learning-based detection, LSTM-driven traffic prediction, and edge computing deployment — are all within reach, and the modular design makes adding them a matter of extension rather than rebuilding from scratch.

REFERENCES

[1] F. V. Webster, "Traffic Signal Settings," Road Research Technical Paper No. 39, HMSO, London, 1958.
[2] D. I. Robertson, "TRANSYT: A Traffic Network Study Tool," Report LR 253, Transport Research Laboratory, 1969.
[3] N. Buch, S. A. Velastin, and J. Orwell, "A Review of Computer Vision Techniques for the Analysis of Urban Traffic," IEEE Trans. Intell. Transp. Syst., vol. 12, no. 3, pp. 920–939, 2011.
[4] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A Survey of Video Processing Techniques for Traffic Applications," Image and Vision Computing, vol. 21, no. 4, pp. 359–381, 2003.
[5] J. Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE CVPR, 2016, pp. 779–788.
[6] A. Luvizon, D. Tabia, and D. Picard, "Vehicle Speed Estimation from Monocular Video Streams," IEEE Signal Process. Lett., vol. 24, no. 2, pp. 193–197, 2017.
[7] S. Karthik et al., "Smart Traffic Management System Using Image Processing," in Proc. ICEECCOT, 2018, pp. 1–5.
[8] R. Sharma and P. Patel, "IoT-Based Adaptive Traffic Signal Control Using ESP32 and IR Sensors," IJERT, vol. 9, no. 6, pp. 812–817, 2020.
[9] S. Gupte et al., "Detection and Classification of Vehicles," IEEE Trans. Intell. Transp. Syst., vol. 3, no. 1, pp. 37–47, 2002.
[10] J. Singh, A. Jain, and S. Arora, "Image Processing Based Intelligent Traffic Controller," UARJ, vol. 1, no. 1, pp. 78–83, 2012.
[11] P. Hunt et al., "The SCOOT On-Line Traffic Signal Optimisation Technique," Traffic Eng. Control, vol. 23, no. 4, pp. 190–192, 1982.
[12] M. Abdulhai et al., "Reinforcement Learning for True Adaptive Traffic Signal Control," J. Transp. Eng., vol. 129, no. 3, pp. 278–285, 2003.
[13] Espressif Systems, "ESP32 Technical Reference Manual," v5.1, 2023.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com