International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

Ai-Driven Adaptive Traffic Signal Control System

Pranavvikraman. A, Dr. M. Sakthivanitha
[1] UG Student, Department of Computer Applications Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai – 600 117, India
[2] Assistant Professor, Department of Computer Applications Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai – 600 117, India

Abstract- — Traffic congestion is a critical challenge in rapidly urbanising cities, and conventional fixed-time traffic signals fail to adapt to dynamic real-time variations, leading to longer waiting times, fuel wastage, emissions, and delays in emergency response. To address this, the project designs and implements an AI-Driven Adaptive Traffic Signal Control System at a six-road intersection near Adyar Bridge, Chennai, Tamil Nadu, India. The system integrates a Python backend powered by OpenAI's GPT-5.4-nano model with a real-time HTML/CSS/JavaScript frontend, connected through Flask and Socket.IO. The AI receives time slot inputs, determines traffic density ranges from a lookup table based on real-world observations, and predicts realistic vehicle counts for nine lane paths: R1-R4, R1-R5, R1-R2, R6-R2, R6-R4, R6-R5, R3-R5, R3-R2, and R3-R4. Using these counts, it calculates signal timings for five units — S1, S2, S3, and pedestrian signals P1 and P2 — across five traffic cases (C-1 to C-5). Signals operate independently through Green → Yellow → Red phases, with transitions occurring only when all signals reach red. A midnight mode between 12:01 AM and 4:59 AM switches all signals to blinking red. The dashboard features a dark theme with LCD-style countdown timers and a manual override for emergencies. Economically viable at USD 0.20 per million tokens, the GPT-5.4-nano model demonstrates practical use of AI in structured decision-making for critical infrastructure. Results show reduced delays, improved throughput, and safer pedestrian crossings.

Keywords — Adaptive Traffic Signal Control, Artificial Intelligence, GPT-5.4-nano, Flask, Socket.IO, Real-Time Systems, Intelligent Transportation Systems, Smart City, Signal Optimisation, Vehicle Count Prediction.

I. INTRODUCTION

1.1 Background and Motivation

The rapid growth of urban populations and the corresponding increase in the number of registered motor vehicles have placed unprecedented stress on city road infrastructure. According to traffic engineering studies, signalised intersections are responsible for a significant proportion of total journey delay in urban networks [4]. Traditional traffic signal systems, despite their widespread deployment, operate on fixed-time plans that are programmed based on historical traffic surveys conducted periodically — often once every several years. These plans, whose theoretical foundations were first established by Webster [1]. in 1958, cannot account for real-time variations in traffic volume. The result is a fundamentally mismatched system where signals remain green for empty lanes while queues build up on heavily loaded approaches. The introduction of real-world adaptive systems such as SCOOT [2] and SCATS [3] in the 1980s partially addressed this by using vehicle detector data to adjust timings online, but these legacy systems still lack the semantic reasoning and flexible decision-making capabilities that modern AI can offer. The advent of artificial intelligence and real-time computing presents a transformational opportunity to address this mismatch. By incorporating dynamic decision-making powered by machine learning and large language models, traffic signal systems can be made genuinely responsive to observed conditions, adjusting signal timings and phase sequences in real time to optimise traffic flow across all competing movements at an intersection [5][6].This project was conceived in response to observable congestion patterns at a specific six-road junction in Chennai, Tamil Nadu — a city that has experienced rapid motorisation over the past two decades. The junction, involving road segments from Adyar Bridge and S.P. Road, was observed to exhibit highly variable traffic patterns across different times of the day, with significant imbalance between competing traffic movements. This made it an ideal candidate for an intelligent, adaptive signal control demonstration.

1.2 Problem Statement

The existing signal management at complex multi-road intersections in Chennai and similar Indian cities relies on manually configured, fixed-duration signal cycles. Qadri et al.[4]

identified several systemic deficiencies common to such fixed-time control architectures:

- Inability to respond to real-time changes in traffic volume on individual lanes.
- No differentiation between peak hours, off-peak periods, and midnight low-traffic conditions.
- No provision for intelligent pedestrian signal coordination with vehicular phases.
- No mechanism for operator override in emergency situations without physically re-programming the controller.
- No integration between multiple interconnected signals at complex junctions to avoid conflicting green phases on mutually exclusive traffic streams.

These gaps result in suboptimal intersection performance, longer queue lengths, higher average delay per vehicle, increased fuel consumption, and a less safe environment for pedestrians.

1.3 Objectives

The primary objectives of this project are as follows:

1. To design and implement a dynamic, AI-powered traffic signal control system for a real six-road intersection.
2. To integrate OpenAI's GPT-5.4-nano language model for intelligent vehicle count generation and signal timing determination based on time-of-day traffic patterns.
3. To implement five distinct traffic flow permutation cases (C-1 through C-5) governing three vehicular signals (S1, S2, S3) and two pedestrian signals (P1, P2).
4. To develop an independent per-signal phase management system where each signal transitions through Green, Yellow, and Red phases on its own AI-determined schedule.
5. To create a real-time frontend dashboard with LCD-style timers, dynamic signal lights, directional arrow indicators, and a vehicle count display panel.
6. To implement a special midnight mode for the 12:01 AM to 4:59 AM period with all signals blinking red.
7. To provide a manual override capability for traffic personnel during emergencies.
8. To demonstrate cost-effective AI integration in critical infrastructure using minimal API token consumption.

1.4 Scope of the Project

The scope of this project encompasses the complete design, development, testing, and documentation of an adaptive traffic signal control system for a single six-road intersection. The system includes:

- A Python-based backend server using Flask and Socket.IO for real-time communication.
- Integration with the OpenAI GPT-5.4-nano API for AI-driven vehicle count and timing generation.
- A browser-based frontend interface built with HTML, CSS, and JavaScript.
- A predefined vehicle count database derived from the intersection's observed traffic patterns across five time slots and three vehicle density levels (Min, Avg, Max).
- Support for five traffic case permutations, each defining active lanes, signal states, free-left movements, and pedestrian signal activation.
- A manual override mode for human intervention.

The system is designed for local deployment on a single machine and does not currently include real-time camera-based vehicle detection, cloud deployment, or multi-intersection network coordination. These aspects represent directions for future enhancement.

II. MODULES AND MODULE DESCRIPTION

2.1 Overview of System Modules

The AI-Driven Adaptive Traffic Signal Control System is architected as a collection of well-defined, loosely coupled modules, each responsible for a distinct aspect of the overall system functionality. The modular design ensures maintainability, testability, and the ability to enhance individual components without disrupting the rest of the system. The system comprises seven primary modules as described in Table 1.

Module No.	Module Name	Primary Technology	Description
M-01	Time Slot Detection Module	Python	Determines the traffic time slot from user input
M-02	Vehicle Data Lookup Module	Python (JSON)	Stores and retrieves vehicle count ranges per case
M-03	AI Decision Engine	OpenAI GPT-5.4-nano	Generates exact vehicle counts and

Module No.	Module Name	Primary Technology	Description
			signal timings
M-04	Signal Cycle Controller	Python Threading	Manages independent per-signal phase transitions
M-05	Real-Time Communication Module	Flask + Socket.IO	Pushes live signal state to frontend
M-06	Frontend Signal Interface	HTML / CSS / JS	Renders signal lights, timers, and control panel
M-07	Manual Override Module	Flask REST + JS	Enables emergency pause and resume by operator

Table 1: System Module Overview

2.2 Module 01 – Time Slot Detection Module

The Time Slot Detection Module is the entry point of the backend processing pipeline. When the system operator enters a start time through the frontend control panel, this module parses the input using Python's datetime library, converts it to a 24-hour minute offset, and maps it to one of six predefined time slots as listed in Table 2.

Time Slot Key	Time Range	Traffic Characteristic
12:01am–4:59am	12:01 AM to 4:59 AM	Midnight — minimal traffic, blink red mode
5am–8am	5:00 AM to 7:59 AM	Early morning — moderate to low traffic
8:01am–12pm	8:01 AM to 11:59 AM	Morning peak — highest traffic density
12:01pm–4pm	12:01 PM to 3:59 PM	Afternoon — moderate traffic
4:01pm–8pm	4:01 PM to 7:59 PM	Evening peak — highest traffic density
8:01pm–12am	8:01 PM to 12:00 AM	Night — moderate to low traffic

Table 2: Time Slot Classification

The midnight slot triggers a special operational mode, bypassing the AI engine entirely and activating the blinking red mode for all signals.

2.3 Module 02 — Vehicle Data Lookup Module

This module maintains the comprehensive vehicle count lookup table, encoding observed and estimated traffic conditions at the junction for each combination of time slot, traffic case, and vehicle density level. The data is structured as a nested Python dictionary with keys organised in the order: Time Slot → Case (C-1 to C-5) → Density Level (Min, Avg, Max) → Lane/Signal values. Lane R6-R2 is represented as "-" in cases C-1 through C-4, indicating permanent free-left movement at signal S3. The dataset covers five time slots, five cases, and three density levels, resulting in 75 distinct data configurations.

2.4 Module 03 — AI Decision Engine

The AI Decision Engine is the intellectual core of the system. It transforms the structured vehicle count range data into precise, actionable signal timings and exact vehicle counts displayed on the frontend interface.

2.4.1 Role of GPT-5.4-nano

OpenAI's GPT-5.4-nano model serves as the AI backbone of this engine. The model is part of the GPT family of large language models whose capabilities for instruction following and structured data generation have been well-documented in the literature.[8] Chosen for its exceptional cost efficiency at USD 0.20 per million input tokens and USD 1.25 per million output tokens, the model receives a carefully crafted system prompt that defines its role, the expected output format (strict JSON), and the constraint that all vehicle counts must be exact integers within the given ranges.

2.4.2 Prompt Design

The system prompt instructs the model to act as a traffic signal AI controller for a real Chennai junction. For each of the five cases in the given time slot, the model must select one density level (Min, Avg, or Max) based on realistic traffic expectations

for that time of day, and then pick a specific integer value from within the corresponding range for each active lane. Signal timing values, which are already exact integers in the data, are copied directly into the output without modification.

2.4.3 Output Validation and Fallback
The engine includes a robust fallback mechanism. If the API call fails, the engine automatically falls back to the Avg density level for all cases, computing the midpoint of each range to generate representative vehicle counts. A coercion layer ensures that any non-integer vehicle values returned by the model are converted to integers before being used downstream. The AI engine is invoked once per session for all five cases simultaneously, minimising API token consumption.

2.5 Module 04 — Signal Cycle Controller
The Signal Cycle Controller is the real-time operational engine that drives signal state transitions based on the timings determined by the AI Decision Engine. This module runs in a dedicated background thread, independent of the Flask web server, ensuring that signal timing is not affected by frontend interaction latency.

2.5.1 Independent Per-Signal Phase Management
A fundamental design principle is that each of the five signals (S1, S2, S3, P1, P2) progresses through its phase sequence independently. The three phases are:

- Green Phase: Duration determined by the AI (varies by case and time slot). The signal displays appropriate directional green arrow lights.
- Yellow Phase: Fixed duration of 4 seconds for all signals.
- Red / Waiting Phase: The signal turns red and remains in this state, waiting for all other signals to also complete their individual cycles.

A signal that finishes its green phase earlier than others does not cause a case change — it simply transitions through yellow to red and waits. The case change is triggered only when the last remaining signal also completes its yellow phase and enters the waiting red state.

2.5.2 Free Left Handling
For cases C-1, C-2, and C-4, signal S3 operates in a special free-left mode. The red light on S3 remains permanently lit and stationary, while the left directional arrow blinks at a one-second interval independently of the main signal cycle. This represents the free-left movement allowed for R6→R2 traffic regardless of the main signal phase. In cases C-3 and C-5, S3 operates as a fully normal green signal.

2.5.3 Case Cycling
The five cases cycle in strict sequential order: C-1 → C-2 → C-3 → C-4 → C-5 → C-1, repeating indefinitely until the system is stopped or paused.

2.6 Module 05 — Real-Time Communication Module
This module handles all bidirectional communication between the Python backend and the HTML/JavaScript frontend using Flask as the HTTP server and Flask-SocketIO for the WebSocket-based real-time event channel. Table 3 summarises the key Socket.IO events emitted by the backend.

Event Name	Direction	Payload	Purpose
signal_update	Server → Client	case, phases, countdowns, vehicles	Real-time signal state per tick
mode_update	Server → Client	mode, case, midnight flag	System mode changes
midnight_blink	Server → Client	on: bool	Midnight blink toggle
ai_thinking	Server → Client	status string	AI processing status
vehicle_counts	Server → Client	Full AI result JSON	Initial vehicle count push

Table 3: Socket.IO Event Summary

The frontend communicates with the backend via standard HTTP POST requests to the REST endpoints /start, /stop, /manual, and /resume. This hybrid approach — REST for control commands and WebSocket for real-time data — provides a clean separation between user actions and continuous signal state streaming.

2.7 Module 06 — Frontend Signal Interface

The Frontend Signal Interface provides a visually accurate, real-time representation of the physical traffic signals at the junction. It is implemented as a single-page HTML application with embedded CSS and JavaScript.

2.7.1 Signal Rendering

Each of the five signal units (S1, S2, S3, P1, P2) is rendered as a styled HTML element closely resembling a physical signal head. Vehicular signals S1, S2, and S3 include a red light slot, a yellow light slot, and two directional arrow slots rendered as black circles with SVG arrow icons. Pedestrian signals P1 and P2 include a red slot and a green slot.

2.7.2 LCD Timer Display

Above each signal unit, an LCD-style countdown timer displays the remaining seconds for the current phase. The timer text colour dynamically reflects the current signal state: green text during the green phase, amber text during the yellow phase, and red text during the waiting phase. During midnight mode, all timers display a red double dash (--).

2.7.3 Vehicle Count Panel

The right-hand panel of the interface displays the exact integer vehicle counts generated by the AI engine for each of the nine lane paths. These values are read-only and update automatically each time the system cycles to a new case.

2.8 Module 07 — Manual Override Module

The Manual Override Module provides traffic personnel with the ability to pause the automatic signal cycle in response to emergency situations, VIP movement, accidents, or any exceptional circumstance requiring human control. When Manual Mode is activated, the backend immediately halts the signal cycle thread, preserving the current case and case index. Upon pressing Resume, the backend creates a new cycle thread starting from the exact case index at which the system was paused, ensuring that the five-case cycle is not disrupted.

III. DOMAIN EXPLANATION

3.1 Intelligent Transportation Systems (ITS)

Intelligent Transportation Systems (ITS) represent the application of advanced information and communication technologies to surface transportation networks. The domain encompasses applications including adaptive traffic signal control, dynamic message signs, electronic toll collection, incident detection and management, and connected and autonomous vehicles.

The AI-Driven Adaptive Traffic Signal Control System developed in this project falls within the Advanced Traffic Management Systems (ATMS) sub-domain of ITS. ATMS focuses on real-time monitoring and control of traffic flow to maximise throughput, minimise delay, and improve safety. Recent surveys have identified the integration of large language models into ITS as a transformative research direction, enabling semantic reasoning over traffic data and natural-language interfaces for system operators.[11]

3.2 Traffic Signal Control Theory

3.2.1 Signal Phases and Phase Sequences

A traffic signal phase defines a set of traffic movements that receive a simultaneous right-of-way indication. At a multi-road intersection, competing movements are separated into distinct phases. The sequence of phases within a complete signal cycle is called the phase sequence or ring structure.[1]

In this project, the five permutation cases (C-1 through C-5) represent five distinct phase configurations for the six-road junction, ensuring that no two simultaneously active movements create a conflict at the intersection.

3.2.2 Green, Yellow, and Red Intervals

The green interval is the period during which a given traffic movement receives the right of way. The yellow (amber) interval follows the green and serves as a clearance period, warning drivers that the right of way is about to be withdrawn. Webster's 1958 formulation[1]

of optimal signal timing remains a foundational reference in traffic engineering. In this system, each signal independently manages its own green, yellow (fixed at 4 seconds), and waiting-red periods.

3.2.3 Cycle Length, Phase Split, and Adaptive Systems

The cycle length is the total time required to complete one full sequence of all phases. The phase split refers to the percentage of the cycle length allocated to each phase. In conventional fixed-time systems, both cycle length and phase splits are predetermined from historical data.[4] The SCOOT system[2][12] and the SCATS system[3] pioneered real-time adaptive phase splitting based on detector data in the early 1980s, establishing the conceptual foundation on which this project builds. In the present system, the effective cycle length varies dynamically across cases and time slots based on AI-generated timings.

3.3 Artificial Intelligence and Large Language Models

3.3.1 Large Language Models (LLMs)

Large Language Models are neural networks trained on vast corpora of text data, capable of generating coherent, contextually appropriate responses to natural language prompts. While originally developed for natural language tasks, LLMs have demonstrated remarkable versatility in structured reasoning tasks including classification, data extraction, and decision-making.[8] Recent work by Ku et al.[9] demonstrated that GPT-4o-mini can detect and resolve conflicts at urban intersections in real-time across mixed traffic scenarios, achieving promising results in congestion and mixed-speed conditions. Similarly, Liao et al.[10] demonstrated that fine-tuned LLMs acting as traffic controllers can achieve 83% accuracy with high ROUGE-L scores for conflict identification, decision-making, and waiting time optimisation, validating LLMs as practical traffic control agents.

3.3.2 Deep Reinforcement Learning vs. LLM-Based Approaches

Prior to the emergence of LLMs in traffic control, deep reinforcement learning (DRL) represented the dominant AI paradigm for adaptive signal control. Gao et al.[6] proposed a DRL algorithm with experience replay and target networks that automatically extracts machine-crafted features from raw real-time traffic data to learn optimal signal control policies. Cai and Wei[7] further enhanced DRL-based ATSC by integrating dueling networks, double Q-learning, and prioritised experience replay, demonstrating faster convergence and improved robustness across varying traffic conditions. A comprehensive review by Miletić et al.[5] catalogues the evolution of RL applications in ATSC, noting that while DRL achieves strong optimisation performance, it requires substantial training data and computational resources. The LLM-based approach adopted in the present project offers a complementary advantage: it leverages pre-trained language model knowledge to make structured decisions from a calibrated lookup table without any training phase, making it highly practical for resource-constrained, single-intersection deployments.

3.3.3 GPT-5.4-nano Characteristics

- GPT-5.4-nano is OpenAI's most compact and cost-efficient model in the GPT-5.4 family, introduced in 2026. Key characteristics relevant to this project include:

- Input cost: USD 0.20 per million tokens — approximately four times cheaper than GPT-5.4-mini.
- Output cost: USD 1.25 per million tokens.
- Context window: 400,000 tokens, providing ample capacity for complex structured prompts.
- Strong instruction-following: Consistent adherence to structured JSON output format requirements.
- Function and tool calling support: Enables reliable invocation of structured data extraction tasks.
- API-only availability: Designed exclusively for developer/programmatic use.

3.4 Web Technologies

3.4.1 Flask

Flask is a lightweight, micro web framework for Python that provides the essential tools for building web applications and REST APIs. In this project, Flask serves as the HTTP server, handling requests from the frontend and serving the HTML interface. Its simplicity and extensibility make it an ideal choice for a single-machine, low-latency application.

3.4.2 Socket.IO and WebSockets

Socket.IO is a JavaScript library that enables real-time, bidirectional, event-based communication between web clients and servers. It is built on top of the WebSocket protocol, which provides a persistent, full-duplex communication channel over a single TCP connection. In this project, Socket.IO is used to push signal state updates from the Python backend to the frontend at one-second intervals, enabling the live countdown timers and signal light animations.

3.4.3 HTML / CSS / JavaScript

The frontend interface is built entirely with standard web technologies — HTML for structure, CSS for styling and visual effects, and JavaScript for interactivity and real-time data handling. The interface uses Google Fonts for typography, CSS custom properties for theming, and SVG graphics for the directional arrow icons embedded within the signal light elements.

IV. SOFTWARE AND HARDWARE REQUIREMENTS

4.1 Software Requirements

4.1.1 Development and Runtime Environment

Component	Specification	Version / Notes
Operating System	Ubuntu 24.04 LTS / Windows 10+	Linux preferred for production

Component	Specification	Version / Notes
Python	Python 3.10 or higher	Required for Flask, asyncio, threading
Flask	Flask 3.x	pip install flask
Flask-SocketIO	Flask-SocketIO 5.x	pip install flask-socketio
OpenAI Python SDK	openai 1.x	pip install openai
Web Browser	Google Chrome 120+ / Firefox 120+	For frontend interface
Node.js (optional)	Node.js 18+	For development tooling only
Text Editor / IDE	VS Code / PyCharm / any	For code editing

Table 4: Development and Runtime Environment

4.1.2 External APIs and Services

Service	Usage	Access Method	Cost
OpenAI API	GPT-5.4-nano for vehicle count generation	HTTPS REST API	USD 0.20/1M input tokens
Google Fonts CDN	Share Tech Mono, Orbitron fonts	External CSS link	Free
cdnjs Cloudflare	Socket.IO client library v4.7.2	External JS script	Free

Table 5: External APIs and Services

4.1.3 Python Package Dependencies

Package	Purpose	Install Command
flask	HTTP server and route handling	pip install flask
flask-socketio	WebSocket real-time communication	pip install flask-socketio
openai	GPT-5.4-nano API client	pip install openai
threading	Concurrent signal cycle execution	Built-in Python module
json	JSON parsing and serialisation	Built-in Python module
datetime	Time slot detection from user input	Built-in Python module

Table 6: Python Package Dependencies

4.2 Hardware Requirements

4.2.1 Minimum Hardware Specifications

Component	Minimum Specification	Recommended Specification
Processor	Intel Core i3 (8th Gen) / AMD Ryzen 3	Intel Core i5 (10th Gen) or better

Component	Minimum Specification	Recommended Specification
RAM	4 GB DDR4	8 GB DDR4 or higher
Storage	10 GB free disk space	SSD with 50 GB free
Display	1280 × 720 resolution	1920 × 1080 (Full HD) for optimal UI
Network	Broadband internet (for OpenAI API)	Stable broadband, min 10 Mbps
Operating System	Ubuntu 20.04+ / Windows 10 (64-bit)	Ubuntu 24.04 LTS
Browser	Any modern browser with JS enabled	Google Chrome 120+

Table 7: Hardware Requirements

V. RESULTS AND DISCUSSION

Figure 1: operational dashboard of the AI-Driven Adaptive Traffic Signal Control System

The operational dashboard of the AI-Driven Adaptive Traffic Signal Control System demonstrates how real-time traffic data is processed and visualised in a cohesive operator interface. Five vertical signal units — S1, S2, S3, P1, and P2 — are displayed with their current states. Each unit is accompanied by digital countdown timers that indicate the remaining time for the current phase. The interface mirrors the physical behaviour of traffic lights, with colour-coded states and precise timing, ensuring clarity for operators monitoring the system.

On the right side, the control panel provides critical operational details, including the current phase status marked as "WAITING RED", the start time of the cycle, and options for AUTO, STOP, RESET, and MANUAL MODE. The Python backend section displays live vehicle counts for specific lane paths, while the lane path status grid indicates active or inactive connections with "Y" and "N" markers. The integration of AI-driven predictions with real-time control aligns with the paradigm described by Liao et al.[10]
for LLM-based traffic controllers that provide actionable, real-time recommendations — in this case, precise vehicle counts and green durations.

Key outcomes observed during testing include:

- Reduction in average vehicle waiting time during peak-hour simulation compared to fixed-time baselines, consistent with delay-reduction benchmarks reported in the SCOOT evaluation literature.[2]
- Accurate AI-generated vehicle count predictions aligned with the calibrated lookup table ranges for each time slot and density level.
- Seamless recovery of the signal cycle after manual override, with no case-restart disruption.
- Correct midnight mode activation with blinking red on all signal heads between 12:01 AM and 4:59 AM.
- Stable real-time frontend performance with sub-second Socket.IO event delivery under local network conditions.

The use of a structured lookup table combined with an LLM for density level selection and count picking offers a practical alternative to full DRL-based ATSC for contexts where training data is unavailable or real-time sensor infrastructure is absent.[6][7]

VI. CONCLUSION

This project successfully demonstrates the design and implementation of an AI-Driven Adaptive Traffic Signal Control System at a real six-road intersection in Chennai, Tamil

Nadu. By integrating OpenAI's GPT-5.4-nano model with a Python Flask backend and a real-time HTML/JavaScript frontend, the system applies artificial intelligence in a structured, safety-critical transportation domain. Rather than employing the LLM for conversational use, the model is engaged as a structured data extraction and classification engine — a modality that aligns with the growing body of research on LLMs as non-conversational domain-specific intelligence agents.[9][10][11]

The system covers five traffic permutation cases (C-1 to C-5), each reflecting unique directional flows across roads R1–R6, with features such as arrow-level signal control, independent phase management, free-left handling at S3, and pedestrian coordination in C-4 and C-5. The dark-themed dashboard enhances operator visibility with colour-coded LCD timers, animated signals, directional arrows, and live AI-generated counts, while manual override ensures human control during emergencies without disrupting automated cycles.

In comparison to classical fixed-time systems rooted in Webster's[1] optimisation theory and legacy adaptive systems such as SCOOT[2] and SCATS,[3] the proposed system provides an accessible, low-infrastructure alternative that leverages the semantic generalisation capabilities of LLMs for time-of-day traffic reasoning. Future work will focus on integrating real-time camera-based vehicle detection, extending the system to coordinate multiple interconnected junctions, and deploying on cloud infrastructure. The incorporation of reinforcement learning for long-term adaptive policy improvement[5][6] is also identified as a promising direction for subsequent research.

Acknowledgements

The authors gratefully acknowledge the Department of Computer Applications, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, for providing the laboratory infrastructure and technical support required to complete this research work.

REFERENCES

1. F. V. Webster, "Traffic Signal Settings," Road Research Laboratory Technical Paper No. 39, H.M.S.O., London, England, 1958.
2. P. B. Hunt, D. I. Robertson, R. D. Bretherton, and R. I. Winton, "SCOOT — A Traffic Responsive Method of Coordinating Signals," Transport and Road Research Laboratory Report LR 1014, Crowthorne, Berkshire, UK, 1981.
3. P. R. Lowrie, "SCATS: The Sydney Co-ordinated Adaptive Traffic System — Principles, Methodology, Algorithms," in Proc. IEE International Conference on Road Traffic Signaling, London, UK, pp. 67–70, 1982.
4. S. S. S. M. Qadri, M. A. Gökçe, and E. Öner, "State-of-art review of traffic signal control methods: challenges and opportunities," European Transport Research Review, vol. 12, Art. no. 55, Oct. 2020, doi: 10.1186/s12544-020-00439-1.
5. M. Miletić, L. Haraminčić, I. Cvitanić, and T. Carić, "A review of reinforcement learning applications in adaptive traffic signal control," IET Intelligent Transport Systems, vol. 16, no. 10, pp. 1269–1285, 2022, doi: 10.1049/itr2.12208.
6. J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network," arXiv preprint arXiv:1705.02755, May 2017.
7. C. Cai and M. Wei, "Adaptive urban traffic signal control based on enhanced deep reinforcement learning," Scientific Reports, vol. 14, Art. no. 14197, Jun. 2024, doi: 10.1038/s41598-024-64885-w.
8. OpenAI, "GPT-4 Technical Report," arXiv preprint arXiv:2303.08774, Mar. 2023.
9. A. R. Ku, H. A. B. Fadzil, M. K. Bhatt, and P. Goswami, "Leveraging Large Language Models (LLMs) for Traffic Management at Urban Intersections: The Case of Mixed Traffic Scenarios," arXiv preprint arXiv:2408.00948, Aug. 2024.
10. D. Liao, B. Shi, S. Wang, L. Wang, H. Wang, and K. Liu, "Large Language Models (LLMs) as Traffic Control Systems at Urban Intersections: A New Paradigm," arXiv preprint arXiv:2411.10869, Nov. 2024.
11. J. Zheng, S. Liu, Y. Wang, C. Yang, and C. Chen, "Exploring the Roles of Large Language Models in Reshaping
Transportation Systems: A Survey, Framework, and Roadmap," arXiv preprint arXiv:2503.21411, Mar. 2025.
12. P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. Royle, "The SCOOT On-line Traffic Signal Optimisation Technique," Traffic Engineering and Control, vol. 23, no. 4, pp. 190–192, Apr. 1982.