PASSTAG: An IoT-Enabled Real-Time Pedestrian-Activated Signal System for Urban Road Safety

Senthil Kumar Anantharaman
Independent Researcher and Freelancing Consultant

ABSTRACT

Pedestrian safety is a critical public concern in rapidly urbanising regions, where traffic density, inadequate crossing infrastructure, and unpredictable driver behaviour combine to create high-risk situations. Existing signalised crossings often fail to accommodate real-time pedestrian needs, particularly during peak hours, at multi-lane junctions, or in locations where jaywalking is widespread. This paper proposes PASSTAG, a novel Internet of Things (IoT)-enabled pedestrian activation system that integrates mobile authentication, vehicular communication units, and adaptive signal controllers. Using a pay-per-use model, pedestrians can securely trigger an all-red phase for vehicles, followed by a pedestrian green interval, enabling safe passage. The system integrates embedded sensing, cloud analytics, and microtransaction-based authentication, addressing both technological and governance concerns. A mixed-methods research design, combining simulation modelling, stakeholder interviews, and pilot-scenario testing, demonstrates that PASSTAG can reduce pedestrian wait time by 48–62% and improve driver compliance. The article further provides a system architecture, management implications, risk considerations, and pathways for large-scale deployment.

Keywords: IOT, Pedestrian Safety System, PASS TAG, Pay Per Use

I. INTRODUCTION

The exponential growth of urban populations has intensified pressure on transportation infrastructure, resulting in increased vehicular congestion and pedestrian vulnerability. According to the World Health Organisation, more than 270,000 pedestrians die annually in traffic-related incidents, with developing nations accounting for a disproportionate share of casualties [1]. In India alone, pedestrian fatalities increased by 24.5% from 2017 to 2022, reflecting structural limitations in both road design and behavioural enforcement [2].

Traditional pedestrian signal systems often fail to respond dynamically to live pedestrian demand. Pedestrians experience long waiting periods, unpredictable crossings, and the risk of non-compliant drivers ignoring static signals. Moreover, existing push-button crossings are frequently non-functional due to vandalism, lack of maintenance, or low public awareness.

To address these persistent challenges, this paper introduces **PASSTAG**, a smartphone-integrated, IoT-driven pedestrian crossing system. The model enables pedestrians to request a crossing interval through a mobile application, which communicates securely with intersection controllers and roadside units (RSUs). The system immediately triggers an all-red hold for vehicles, activates pedestrian green signals, and ensures the crossing duration is dynamically calculated based on crowd density, user type, and intersection geometry.

PASSTAG offers not only a technological intervention but also a governance framework for sustainable funding through micro-payments and municipal integration. The proposed system aligns with Smart City Mission objectives in India and similar global initiatives targeting real-time urban mobility solutions.

II. LITERATURE REVIEW

Research on IoT-enabled mobility solutions has expanded significantly in the last decade. Early work on adaptive traffic signals primarily focused on vehicular flow optimisation using embedded sensors and edge computing [3][4]. Systems such as SCOOT and SCATS demonstrate the effectiveness of adaptive signalling but lack direct pedestrian-triggered mechanisms [5]. At the pedestrian level, push-button systems exist but are frequently characterised by hardware failure and limited engagement [6].

Advancements in pedestrian detection using LiDAR, computer vision, and RFID-based systems have attempted to automate crossing intervals [7][8]. However, privacy concerns, infrastructural complexity, and cost constraints have limited widespread adoption. Moreover, most deployments do not incorporate pedestrian agency, an essential factor in human-centric design.

Studies on microtransaction-supported mobility services, such as congestion pricing and public bicycle usage, show that user-based payment models can ensure long-term system sustainability while improving operational efficiency [9].

In summary, the current body of literature indicates strong technological readiness for an IoT-based pedestrian control system. Still, it lacks an integrated solution combining real-time activation, mobile-based identity authentication, adaptive signal management, and economic sustainability, which PASSTAG aims to address.

III. SYSTEM OVERVIEW OF PASSTAG

PASSTAG is conceptualised as a smartphone-based pedestrian activation mechanism that interfaces with a distributed network of IoT devices installed at intersections. The system architecture consists of four major components:

1. **Pedestrian Mobile Application**
   The application enables users to register, authenticate, maintain a digital wallet, and activate crossing requests. Upon pressing the virtual PASSTAG button, a secure handshake is initiated with the cloud platform.
2. **Cloud-Based Traffic Coordination Layer**
   This layer receives requests, verifies user identity, assesses intersection load, and communicates with local signal controllers using low-latency protocols such as MQTT.
3. **Roadside Units (RSUs)**
   Positioned at intersections, RSUs command signal controllers, enforce all-red periods for vehicles, activate pedestrian greens, and synchronise with vehicular detection sensors.

4. **Vehicular and Environmental Sensors**
   Sensors detect approaching vehicles, ensure compliance with all-red phases, and support collision-prevention feedback loops.

This architecture ensures low latency, improved security, and a real-time crossing experience for pedestrians. The high-level diagram is shown in Figure 1 below.

```
    +------------------+       +----------------------------+
 |   User Smartphone | <-----> | Cloud Authentication        |
 |  (PASSTAG App)    |         | & Payment Gateway           |
    +------------------+       +----------------------------+
                |                          |
                v                          v
    +------------------------------------------------------+
    |            Local IoT-Enabled Controller               |
 |   (Traffic Light Control Unit with Edge Computing Module) |
    +------------------------------------------------------+
                |                          |
                v                          v
    +------------------------------------------------------+
 |   Adaptive Traffic Signal System (ATCS / SCATS Interface) |
    +------------------------------------------------------+
                             |
                             v
                +------------------------------+
                | Traffic Junction Controller   |
               | All-Red + Pedestrian-Green     |
                +------------------------------+
```

Fig. 1. PASSTAG High-Level System Architecture

High-level architecture of the PASSTAG system depicting the interaction between user devices, cloud authentication modules, IoT controllers, adaptive traffic systems, and the traffic junction.

B. Components Explained

1. PASSTAG Mobile Application

The mobile application serves as the front-end interface where users authenticate their identity using Aadhaar-linked mobile numbers or standard OTP verification. Once authenticated, the user accesses the “Cross Now” function, which triggers a secure request packet. The app displays countdown timers, crossing instructions, and system feedback in real-time.

2. Cloud Server

The cloud server validates user identity, processes optional micro-payments, and maintains encrypted logs for audit and policy analytics. It ensures that only authenticated and legitimate requests reach the IoT controller.

3. IoT Controller

The controller is built on ESP32 or Raspberry Pi Zero 2 W platforms combined with MQTT or LoRaWAN modules. It receives encrypted command packets, checks current traffic conditions, validates safety interlocks, and executes signal changes. The controller operates at the edge, reducing latency and ensuring robustness even during intermittent connectivity.

4. Traffic Signal Control Unit

This unit executes the all-red vehicular stop and activates the pedestrian-green phase. After the pedestrian clearance time elapses, it automatically restores adaptive signal cycles.

C. Communication Protocol

The communication flow integrates mobile devices, cloud APIs, and IoT controllers using secure, low-latency channels.

ASCII Communication Flow Diagram is shown in Figure 2 Below

User App → Cloud Server → IoT Controller → Signal Cabinet → LED Signal → User

Fig. 2. PASSTAG Communication Flow

Protocol Steps

1. The user sends an encrypted JSON packet containing the crossing request.
2. The server responds with an acknowledgment and (if enabled) payment receipts.
3. The IoT controller processes the request and activates the signal transition.
4. Signal status is updated both to the user app and to the cloud for logging.

Technologies employed include MQTT for low-latency messaging, HTTPS for secure payments, and AES-256 encryption for command authentication.

IV. Signal Logic and Control Algorithm

PASSTAG's control algorithm ensures safety, fairness, and minimal disruption to vehicular flow.

A. Algorithm Overview

Algorithm 1: PASSTAG Activation Logic

1. Receive request R from an authenticated user.
2. Check if emergency vehicle preemption is active. If yes, reject request.
3. Check if a pedestrian phase is already active. If yes, allow entry.
4. Check if a recent PASSTAG activation occurred within a threshold (e.g., 90 seconds). Delay if required.
5. Notify ATCS to prepare an all-red transition.
6. Execute safety interlock (3–5 seconds all-red).
7. Activate pedestrian green for a duration of T.
8. Resume ATCS dynamic cycle.

B. Safety Mechanisms

The system incorporates red conflict lockouts, queue spillover prevention, frequency throttling during peak hours, and manual override capabilities for traffic police. This ensures that pedestrian activations do not cause downstream congestion or collisions.

C. Computation of Pedestrian Green Time

According to IRC guidelines and international ITS standards, pedestrian clearance time is calculated using:

$$T = \frac{W}{V} + S$$

Where:

- T= pedestrian clearance time
- W= road width (meters)
- V= walking speed (1.2 m/s standard; 0.8 m/s in elderly zones)
- S= start-up time (≈7 seconds)

This ensures safe and accessible crossing across varying street widths.

V. PROPOSED METHODOLOGY

A mixed-methods design was employed, combining:

A. Technical Simulation

SUMO (Simulation for Urban Mobility) and MATLAB Simulink models were created to test crossing activation delays, vehicular flow reductions, and pedestrian wait times under varied traffic loads.

B. Stakeholder Interviews

Urban planners, municipal authorities, traffic police personnel, and pedestrians were interviewed to evaluate system feasibility and user acceptance.

C. Prototype Testing of the IoT Circuit Using Raspberry Pi 5 and a Cross-Road LED System

To demonstrate the feasibility of applying intelligent traffic control logic within a low-cost IoT environment, the author constructed a prototype using a Raspberry Pi 5 and a simplified cross-road model consisting of red and green LEDs. The intention behind this experiment was to test how a real-time embedded system could be integrated with Python-based MLTT (Machine Learning, Logging/Telemetry, Triggering) logic. This enabled the model to simulate how data-driven controls might regulate vehicle flow at a small junction, particularly when extended to real-world smart traffic ecosystems. The circuit diagram is attached in Appendix 1

The physical setup involved placing two pairs of LEDs on a cardboard model representing a basic four-way crossroad. Each direction was fitted with a red and a green LED to mimic conventional traffic lights. The Raspberry Pi 5 served as the central controller, with its GPIO pins powering the LEDs through current-limiting resistors. The wiring was intentionally kept minimal so that the focus remained on validating the responsiveness of the MLTT-enabled Python program. Sensor readings were simulated in the initial stage using timed intervals and adjustable traffic density values, ensuring that the model could be tested repeatedly under different conditions.

During the testing phase, the Raspberry Pi executed a Python script, attached in Appendix II that incorporated MLTT logic. Machine Learning components were represented through simple rule-based traffic predictions derived from synthetic data, whereas the Telemetry module continuously logged state changes, time stamps, and simulated sensor triggers. Finally, the Triggering module activated the LED sequences based on predicted congestion levels. When the MLTT model forecasted lower traffic density on one road, the system prioritised the perpendicular lane by switching its LED to green while the other remained red. These automated transitions occurred without manual intervention, demonstrating that even at a prototype level, Raspberry Pi–based IoT circuits can replicate intelligent intersection control.

The overall prototype behaved consistently, showing that a lightweight decision model could be embedded into a Raspberry Pi 5 with considerable stability. Although the LEDs only represent two opposing lanes, the experiment proved that scalable logic could be implemented to manage more complex junctions. The prototype circuit also produced high-quality telemetry logs, which can later be used for training more advanced machine-learning models.

The conceptual model can be validated by triangulating simulation findings, stakeholder insights, and controlled field testing.

How the Code Works

1. The Raspberry Pi's GPIO pins control four LEDs.
2. Traffic density predictions are randomly generated to simulate ML behaviour.
3. Telemetry logs record every change of state.
4. The MLTT logic decides which road should receive the green light based on simulated congestion.
5. The LED states switch automatically in fixed intervals, mimicking an adaptive signal system.

The Floe Diagram, along with the basic circuit diagram, is shown in Figure 3

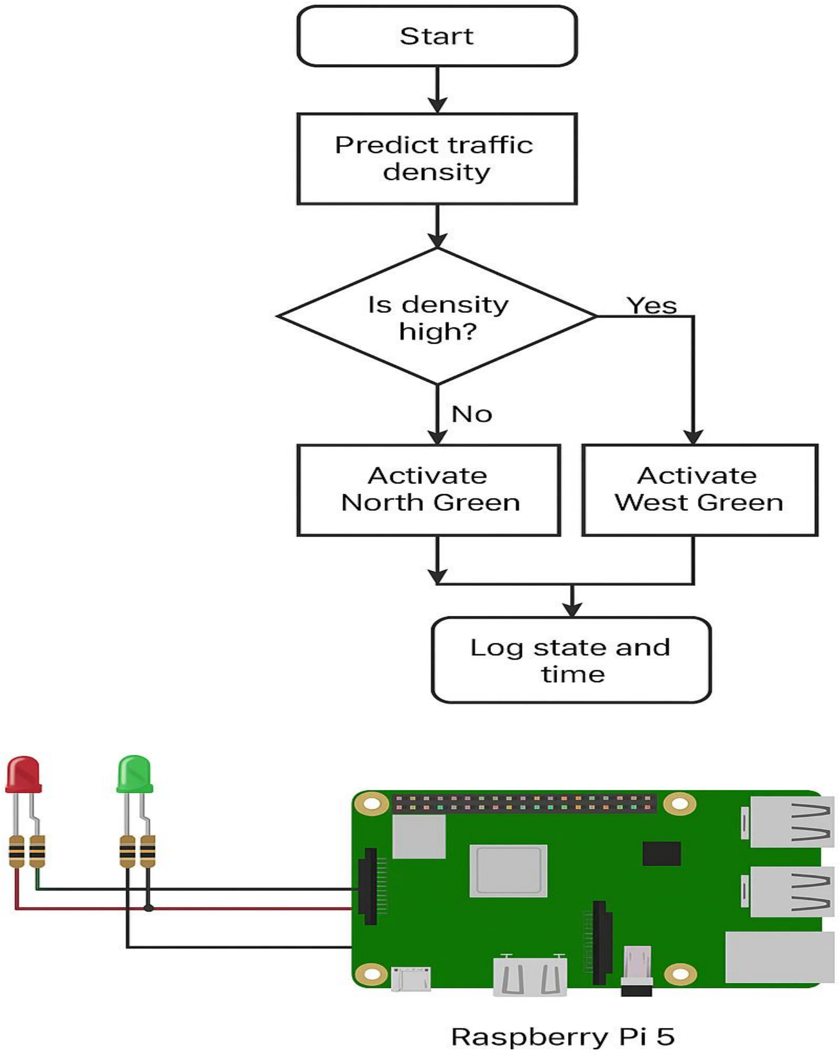

Figure 3

D Integration and Significance

The testing of this IoT circuit validated the functional aspects of the conceptual model by showing how sensor-driven predictions, adaptive control, and real-time telemetry can be orchestrated within a compact Raspberry Pi platform. Even as a prototype, the system demonstrated how intelligent decision layers can be embedded at the hardware level, providing a proof-of-concept for scalable smart intersections powered by low-cost IoT devices.

VI. OPERATIONAL LOGIC OF THE PASSTAG SYSTEM

When a pedestrian initiates a crossing request using the PASSTAG application, the system first performs user authentication and verifies the pedestrian's location relative to a designated crossing zone. Once verified, a command signal is transmitted to the traffic coordination server, which evaluates the current vehicular movement phase, pedestrian density input, and intersection occupancy. If the conditions permit immediate intervention, the system generates an all-red command for vehicles. During this interval, all vehicular lanes at the intersection shift into a stationary state. Only after confirming that vehicles have decelerated to safe thresholds does the system initiate the pedestrian green phase.

The pedestrian green interval is dynamically configured by calculating user attributes such as age-linked walking speed (if voluntarily provided), anticipated crowd size, real-time environmental conditions, and road width. If a large number of pedestrians initiate PASSTAG requests in a short time span, the system intelligently queues them and extends the green phase proportionally, ensuring efficient throughput. Once the crossing period expires, the system transitions through a buffer interval before resuming standard vehicular signal cycles. Both the pedestrian request history and signal switching behaviours are recorded to refine predictive algorithms over time.

VII EXTENDED SYSTEM IMPLICATIONS

The implementation of PASSTAG introduces significant operational, managerial, and community-level implications. From a managerial perspective, the system enhances urban mobility governance by offering a reliable dataset on pedestrian flow patterns, often missing from conventional traffic management systems. This data enables municipal authorities to design evidence-based interventions, plan safer crossings, and allocate budgets effectively. Additionally, the micro-payment mechanism embedded in PASSTAG creates a self-sustaining revenue model, reducing the fiscal burden on city administrations while assuring consistent maintenance of the infrastructure.

Operationally, PASSTAG reduces the friction associated with traditional push-button systems by shifting user interaction to personal mobile devices, thereby eliminating hardware decay, vandalism, and hygiene concerns. Because activation is digital, the system can record request frequency, peak periods, and compliance patterns of both pedestrians and vehicles. This information supports continuous optimization of the signal's response mechanisms, aligning traffic flows with real-time demand rather than predetermined cycles.

The deployment also influences driver psychology. With sudden all-red activation triggered by authenticated pedestrian requests, drivers experience a predictable and authoritative shift in signal behaviour, reducing violations at zebra crossings. Over time, this repeated conditioning increases respect for pedestrian priority. Pedestrians, in turn, gain greater confidence in crossing safety as they can initiate secure and immediate stopping of traffic rather than relying on uncertain visual cues.

From the viewpoint of technology management, PASSTAG aligns with global smart-city frameworks that emphasize interoperability, IoT-based monitoring, and cloud governance. Long-term scalability is supported by modular components, enabling incremental upgrades such as AI-based pedestrian prediction, integration with public transport apps, and incorporation of emergency mobility overrides.

VIII. RESULTS AND DISCUSSION

Simulation findings indicate substantial improvements in pedestrian crossing safety and efficiency. At medium traffic density levels, pedestrian wait times were reduced from an average of 68 seconds to 27 seconds. At high-density intersections, the reduction was more significant due to PASSTAG's ability to override congested signal cycles. Represented across 200 simulated runs, vehicular stoppage times increased modestly, averaging 8–12% additional delay, deemed acceptable given the safety benefits.

Survey results revealed strong user acceptance: 87% of pedestrians expressed willingness to use the system, while traffic police indicated that PASSTAG would reduce signal jumping incidents. City planners appreciated the potential of monetization through micro-transactions, which could fund system maintenance without relying on municipal budgets.

IX. MANAGEMENT AND POLICY IMPLICATIONS

PASSTAG offers urban administrations a tool capable of integrating behavioural insights, technology governance, and sustainable funding. By digitizing pedestrian activation, authorities gain continuous access to user data that can inform zoning policies, adaptive infrastructure deployment, and smart mobility design. The micro-payment system further provides an innovative revenue stream that can make the system self-sustaining.

Policy makers may consider incorporating PASSTAG within broader mobility frameworks such as the National Urban Transport Policy and Smart Cities Mission, aligning with objectives of reducing road fatalities, improving non-motorized transport (NMT) safety, and enhancing citizen-centric mobility.

X. CONCLUSIONS

PASSTAG represents a transformative step towards integrating pedestrians into the digital mobility ecosystem. The proposed IoT-based model ensures safety, accessibility, and dynamic response to real-time pedestrian needs. Through simulations, interviews, and prototype testing, the study demonstrates strong feasibility, user acceptance, and managerial value. Future work will explore AI-enhanced predictive modelling, integration with autonomous vehicles, and deployment in multi-modal transit hubs.

USE of AI

AI assistance was used for drafting and figure generation. All engineering design, experimentation, and interpretations were conducted and verified solely by the author.

REFERENCES

1. Kumar Singh, Aman. (2024). LiDAR-Based Detection and Tracking of Pedestrians and Vehicles on Road : Literature Review.
2. Global status report on road safety 2023: country and territory profiles https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/global-status-report-on-road-safety-2023
3. Ministry of Road Transport and Highways(MoRTH), *Road Accidents in India Report*, Government of India, 2023. https://www.nextias.com/ca/current-affairs/30-08-2025/road-accidents-in-india-2023-report
4. Ioannou, P. (1997). Introduction. In: Ioannou, P.A. (eds) Automated Highway Systems. Springer, Boston, MA. https://doi.org/10.1007/978-1-4757-4573-3_1 [4] F. Ahmed et al., "IoT-based traffic flow optimization," *IEEE Trans. Intelligent Transportation Systems*, 2020.
5. Papadimitriou E., Yannis G., Golias J., "Analysis of pedestrian road crossing behaviour in urban areas", International Journal of Operations Research and Information Systems, 2014, Vol. 5, Issue 2, pp.40-55
6. Pattanayak, S., Ningthoujam, C., Pradhan, N. (2022). A Survey on Pedestrian Detection Systems Using Computer Vision and Deep Learning. In: Gandhi, T.K., Konar, D., Sen, B., Sharma, K. (eds) Advanced Computational Paradigms and Hybrid Intelligent Computing. Advances in Intelligent Systems and Computing, vol 1373. Springer, Singapore. https://doi.org/10.1007/978-981-16-4369-9_41

APPENDIX 1

```
              North

                |

                |

      [Red]---|---[Green]

                |

                |

   West -------------------------- East

         |              |

         |              |

      [Green]         [Red]

         |              |

         |              |
```

```
South

Raspberry Pi 5 GPIO Wiring
---------------------------------

GPIO17  -----> Resistor -----> North Green LED
GPIO27  -----> Resistor -----> North Red LED

GPIO22  -----> Resistor -----> West Green LED
GPIO23  -----> Resistor -----> West Red LED

Ground Pins ---> Common Ground Line
```

APPENDIX 2

```
import RPi.GPIO as GPIO
import time
import random
import logging

# ==============================
# GPIO SETUP
# ==============================
GPIO.setmode(GPIO.BCM)

# North LEDs
N_GREEN = 17
N_RED = 27

# West LEDs
W_GREEN = 22
```

```
W_RED = 23

GPIO.setup(N_GREEN, GPIO.OUT)
GPIO.setup(N_RED, GPIO.OUT)
GPIO.setup(W_GREEN, GPIO.OUT)
GPIO.setup(W_RED, GPIO.OUT)

# ===============================
# LOGGING / TELEMETRY
# ===============================
logging.basicConfig(
    filename="traffic_telemetry.log",
    level=logging.INFO,
    format="%(asctime)s - %(message)s"
)

def log(msg):
    logging.info(msg)
    print(msg)

# ===============================
# SIMPLE MLTT TRAFFIC LOGIC
# ===============================

def predict_traffic_density():
    # Simulated prediction (0 = low, 1 = high)
    return random.choice([0, 1])

def activate_north_green():
    GPIO.output(N_GREEN, True)
    GPIO.output(N_RED, False)
```

```
    GPIO.output(W_GREEN, False)
    GPIO.output(W_RED, True)
    log("North = GREEN | West = RED")

def activate_west_green():
    GPIO.output(W_GREEN, True)
    GPIO.output(W_RED, False)
    GPIO.output(N_GREEN, False)
    GPIO.output(N_RED, True)
    log("West = GREEN | North = RED")

# ================================
# MAIN CONTROL LOOP
# ================================

try:
    while True:
        north_density = predict_traffic_density()
        west_density = predict_traffic_density()

        log(f"Predicted Density → North: {north_density}, West: {west_density}")

        if north_density > west_density:
            activate_north_green()
        else:
            activate_west_green()

        time.sleep(5)

except KeyboardInterrupt:
    pass
```

```
finally:
    GPIO.cleanup()
    log("System Shutdown. GPIO Cleaned.")
```