Research Trends International Journal of Technology and Innovation
An International Peer-Reviewed, Refereed & Open-Access Journal
ISSN : 3144-3273
Available Online at : https://rtijti.com/
doi : https://doi.org/10.5281/zenodo.23721929

Web-Controlled Smart Street Light Fault Detection System Using ESP32 and IoT

C. P. Divate[1], N. R. Bhokare[2], S. S. Chougule[3], Shruti Nandkumar Bavade[4], Ashvita Kumar Kudachi[5], Udayan Ravsaheb Babar[6], Yash Dipak Jasud[7], Gavandi Sartaj Khalil[8]

[1] Academics Dean, [2] Head of Department, [3] Project Guide, [4, 5, 6, 7, 8] Students
Department of Computer Engineering, ATS Sanjay Bhokare Group of Institute, Miraj, Maharashtra, India

ARTICLE INFO

Article History:
Received : 23-10-2026
Accepted : 26-10-2026
Published : 28-10-2026

Publication Issue
Volume 1, Issue 4
October – December 2026

Page Number
44-48

ABSTRACT

Street lighting is a public-safety service and a large part of municipal energy use, yet faults such as burnt-out lamps and cut wires are still found mainly through night patrols or complaints from residents. Lamps can also stay lit in daylight when a timer or switch fails.

This paper presents the design of a web-controlled smart street light fault detection system based on the Internet of Things. Each pole carries an ESP32 or NodeMCU node with a Light Dependent Resistor (LDR) for switching the lamp from ambient light, an ACS712 sensor for measuring the load current, and a relay for remote switching. The firmware compares the commanded relay state with the measured current to separate open-circuit faults (blown lamp or broken wire) from overcurrent faults (short circuit). Readings are sent over MQTT or HTTP to a Node.js or Flask back end with a MySQL or PostgreSQL database, and a responsive web dashboard shows the status of each pole, raises fault alerts, and lets operators override zones remotely.

The system is intended to reduce manual patrolling, shorten the time taken to respond to faults, and cut daytime energy waste. A plan for evaluating the prototype is also given.

Index Terms - Internet of Things, Smart Street Light, Fault Detection, ESP32, ACS712, MQTT, Web Dashboard, Smart City

I. INTRODUCTION

Public lighting is both a safety service and a large consumer of urban energy. In many municipalities a faulty lamp is found only when a maintenance crew patrols at night or when residents complain. This leaves dark stretches on roads for days, delays repairs, and wastes energy when lamps stay lit in daylight because a control has failed.

The Internet of Things (IoT) can turn this passive infrastructure into a monitored network of assets. A low-cost sensing node on every pole can measure the state of its lamp and report it to a central server. A web dashboard then shows operators which pole has failed, and where, without sending a crew to look.

1.1 Background and Motivation

Several street light monitoring systems have been reported [1]-[5], but many of them depend on GSM modules, dedicated mobile applications, or extra GPS hardware on each pole. The aim of this work is a design that a student team can build from common parts (ESP32, LDR, ACS712 current sensor, and a relay module) and that a municipality could try on a few poles before a larger rollout.

1.2 Contributions

The main contributions of this work are as follows:

Copyright © 2026 The Author(s) : This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

- An ESP32-based pole node that switches the lamp from ambient light and measures its current.
- A firmware rule that separates open-circuit and overcurrent faults by comparing the relay state with the measured current.
- A web platform with role-based access for administrators and field engineers that raises fault tickets and allows remote zone control.
- A plan and a set of measures for evaluating the prototype.

1.3 Organisation of the Paper

Section II reviews related work and Section III states the problem and the objectives. Section IV describes the system design and Section V the software. Section VI gives the evaluation plan and feasibility, Section VII concludes the paper, and Section VIII lists future work.

II. RELATED WORK

A. Cloud-Based Fault Alerts: Sharan Kumar and Vadivel described an IoT street light fault detection and management system in which sensor readings from each lamp are sent to a cloud platform. Abnormal behaviour, such as a lamp that stays off after dusk, is flagged automatically, which removes the need for manual scouting [1].

B. Mobile Application with Secured Diagnosis: Kanthi and Dilli proposed a smart streetlight system that uses a master and slave arrangement of nodes across poles together with a secured mobile application for fault detection and diagnosis, with attention to the power drawn by the system [2].

C. ESP32 Edge Diagnosis: Bhisikar et al. used the ESP32 and its built-in Wi-Fi to build diagnostic nodes that check line parameters at the pole, tell a cut wire apart from a failed lamp, and send reports to supervisors [3].

D. Fault Location with WSN and GPS: Sudha et al. combined a wireless sensor network with GPS so that the coordinates of a failed street light are sent to a monitoring station in real time and crews can go straight to the pole [4].

E. Centralised Monitoring: Vamsi et al. presented a sensor-enabled centralised monitoring system in which the readings from many poles are collected in one place and faults are displayed to the operator [5].

A comparison of the related work is given in Table 1, arranged from the most recent study to the oldest.

Table 1. Comparative analysis of related work

Ref.	Year	Technique	Key Finding	Limitation
[4]	2025	WSN + GPS fault location	Sends coordinates of failed poles	GPS module on every pole
[3]	2024	ESP32 edge diagnosis	Separates cut wire from lamp failure	No remote zone control
[5]	2024	Centralised sensor monitoring	One view of many poles	Limited operator workflow
[1]	2024	IoT cloud fault alerts	Removes manual scouting	No fault ticket workflow
[2]	2023	Master and slave WSN + mobile app	Secured fault diagnosis	Depends on a mobile app

2.1 Research Gap

These works show that pole-level sensing, cloud telemetry, and dashboards are practical for street light monitoring. Few of them combine, in one low-cost design, local LDR-based switching, current-based fault classification on the node itself, role-based web access with fault tickets, and remote override through relays. The proposed system brings these together.

III. PROBLEM STATEMENT AND OBJECTIVES

3.1 Challenges Identified from the Literature

Usability: operators need a clear dashboard that works on control-room screens and on phones, especially during emergencies [1].

Trust in sensor data: nodes installed outdoors must be checkable through status history and self-test routines [2].

Latency: delays in telemetry or relay commands reduce the value of real-time control [3].

Scalability: thousands of poles sending data at the same time must not slow down the server or the dashboard [5].

Security: remote switching must be protected from unauthorised access through authentication and encryption [2].

3.2 Objectives

- Detect lamp and wiring faults automatically at pole level and report them to the dashboard.
- Switch lamps automatically from ambient light to avoid daytime energy waste.
- Allow authorised users to switch single poles or whole zones on or off from the web.

- Store telemetry, fault tickets, and asset records in a relational database.
- Provide a responsive dashboard that works on desktop and mobile devices.
- Define measures for fault detection, latency, availability, and usability.

IV. SYSTEM DESIGN

4.1 Architecture

The system has three layers, as shown in Figure 1. The pole node senses light and current and drives the lamp relay. The cloud layer receives telemetry, applies access control, stores data, and forwards commands. The web dashboard shows the status of the network and accepts operator actions.

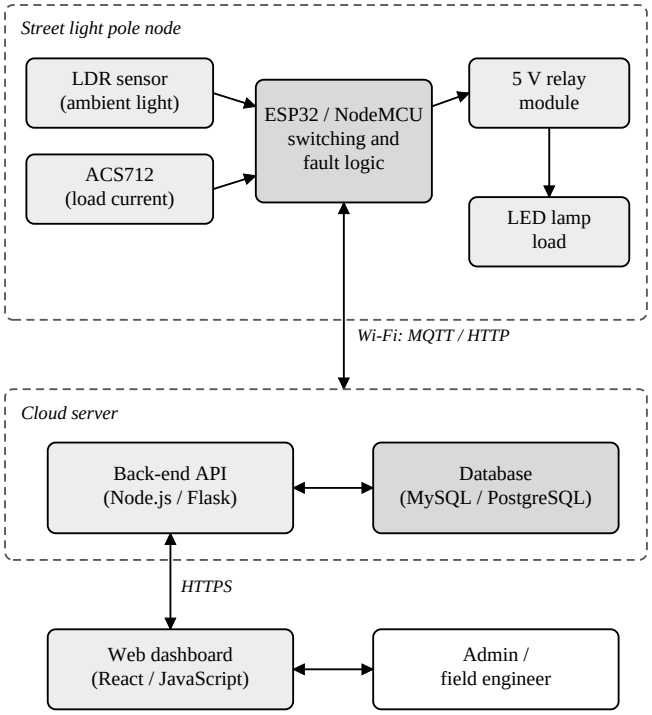


Figure 1. Architecture of the proposed street light fault detection system.

4.2 Pole Node Hardware

Each node uses an ESP32 (or a NodeMCU ESP8266) [8] as the controller. An LDR measures ambient light, and the firmware compares it with a threshold to switch the lamp on at dusk and off at dawn. An ACS712 Hall-effect sensor [7] is placed in series with the lamp to measure its current. A 5 V relay module switches the LED lamp and can also be driven by commands from the web server. The parts of the node are listed in Table 2.

Table 2. Components of the pole node

Component	Role in the node
ESP32 / NodeMCU	Controller; runs the switching and fault logic and sends telemetry over Wi-Fi
LDR	Measures ambient light for dusk and dawn switching
ACS712	Measures the lamp current for fault detection
5 V relay module	Switches the lamp on local or remote command
LED lamp	Street light load being monitored

4.3 Fault Detection Logic

The firmware runs a diagnostic loop that compares the commanded relay state with the measured current I. Table 3 lists the cases and Figure 2 shows the loop. If the relay is ON but the current is close to 0 A, the lamp circuit is open, which means a blown lamp or a broken line. If the current is above a safe limit, a short circuit or overload is assumed and the relay is opened to protect the circuit. The limits are set during calibration.

Table 3. Fault classification rules

Relay	Measured current	Diagnosis and action
ON	Within normal range	Healthy; report status
ON	≈ 0 A	Open circuit (blown lamp or broken wire); raise fault ticket
ON	Above safe limit	Overcurrent or short circuit; open relay and raise alert
OFF	≈ 0 A	Healthy (daytime or manual off)

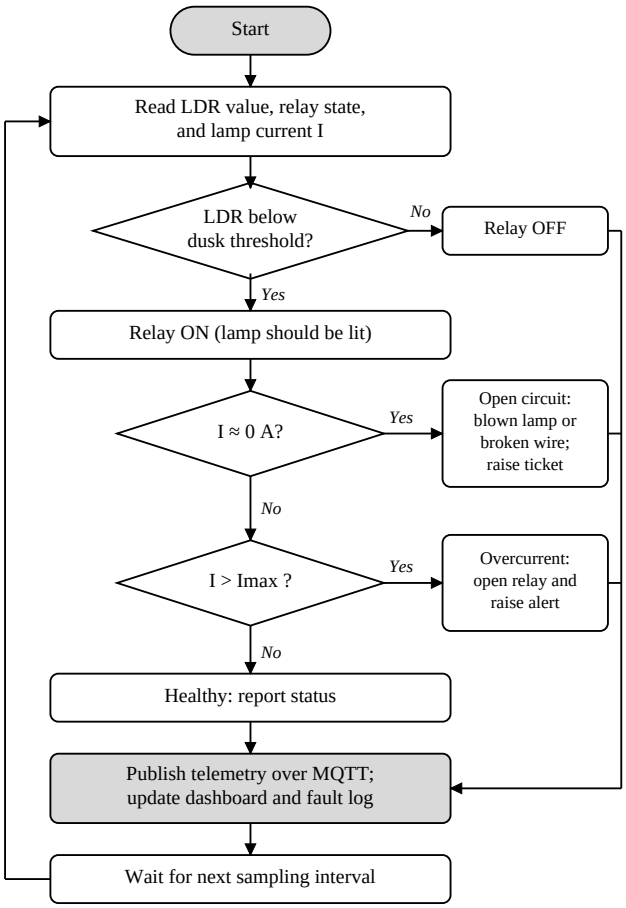


Figure 2. Working flow of the fault detection loop on the pole node.

4.4 Web Control and Operating Modes

Three access points are provided. Administrators monitor the status of all poles, handle escalated faults, change configuration limits, and send override commands. Field engineers register their profiles and assigned areas, accept fault tickets, track their daily schedule, and mark repairs as complete from the field. The pole node publishes telemetry and fault reports and receives control commands.

By default the nodes work on their own from the LDR, so central action is rarely needed. The override console lets operators bypass the LDR schedule in special cases such as storms or maintenance work. On the dashboard each node has an asset code (for example Node-01) and a status colour: green for healthy and red for a detected fault, with time-stamped entries added to the fault log.

V. SOFTWARE IMPLEMENTATION

Telemetry is sent from the nodes to the server using MQTT [6] or HTTP. The back end is written in Node.js or Python (Flask/Django) and handles authentication, telemetry intake, creation of fault tickets, and routing of relay commands. A relational database (MySQL or PostgreSQL) stores pole profiles, location data, telemetry history, fault tickets, and user accounts.

The front end uses HTML5, CSS3, and JavaScript, with React or Angular for a responsive dashboard that shows charts and status colours on monitors and smartphones. The back end can be deployed on AWS, Google Cloud, or Azure to support many nodes at once. Communication uses TLS, and remote switching commands need an authenticated session. Development follows an Agile process with version control and repeated hardware and software testing.

VI. EVALUATION PLAN AND FEASIBILITY

6.1 Test Procedure

The prototype will be checked by removing the lamp while the relay is ON to confirm that an open-circuit fault appears on the dashboard, by creating an overload to check overcurrent protection, by covering and uncovering the LDR to check switching, by sending web commands to check remote override, and by disconnecting the network to check recovery. Usability will be tested with a small group of operators.

6.2 Performance Measures

The measures planned for the evaluation are listed in Table 4. Measured values will be reported after testing of the prototype is complete.

Table 4. Planned performance measures

Measure	How it is measured
Fault detection accuracy	Injected faults correctly classified divided by total injected faults
Fault alert latency	Time from fault occurrence to alert on the dashboard
Relay command latency	Time from web command to relay change
Dashboard load time	Page load time on desktop and mobile
Uptime and data loss	Server availability and share of telemetry packets lost
Usability	Operator survey after the test sessions
Security	Check of the TLS setup and access control

6.3 Feasibility

The parts and tools chosen (ESP32, LDR, ACS712, relay modules, React, Node.js, PostgreSQL) are cheap and easy to obtain, so the system is technically and economically feasible as a college project and can be extended for municipal use. Operation is simple because the dashboard uses clear colour coding. Electrical safety rules for mains-powered lamp circuits and data protection requirements must be followed.

VII. CONCLUSION

This paper described a web-controlled street light fault detection system that combines an ESP32 pole node, LDR-based switching, current-based fault classification, MQTT or HTTP telemetry, and a web dashboard with remote control. The design aims to replace manual patrols with automatic detection, reduce repair delays, and avoid daytime energy waste. Measured results will be reported once testing of the prototype is complete.

VIII. FUTURE ENHANCEMENT AND DIRECTIONS

- Building the prototype and carrying out the evaluation planned in Section VI.
- GPS-based location of each pole, as in [4].
- Cloud analytics on stored telemetry for predictive maintenance.
- Encrypted mesh or GSM links for areas without Wi-Fi coverage.

ACKNOWLEDGMENT

The authors thank the Department of Computer Engineering, ATS Sanjay Bhokare Group of Institute, Miraj, for laboratory facilities and support.

REFERENCES

[1] R. P. Sharan Kumar and R. Vadivel, "IoT based smart street light fault detection management system," *World Journal of Advanced Research and Reviews*, vol. 21, no. 3, pp. 1659-1666, 2024, doi: 10.30574/wjarr.2024.21.3.0685.

[2] M. Kanthi and R. Dilli, "Smart streetlight system using mobile applications: Secured fault detection and diagnosis with optimal powers," *Wireless Networks*, vol. 29, no. 5, pp. 2015-2028, 2023, doi: 10.1007/s11276-023-03278-9.

[3] S. Bhisikar, P. Jagtap, S. Sonawane, H. Sawant, and P. Pisal, "Enhanced streetlight management: Using IoT and ESP32 for automated fault detection," in *Proc. 2024 IEEE 5th India Council Int. Subsections Conf. (INDISCON)*, 2024, pp. 1-6, doi: 10.1109/INDISCON62179.2024.10744375.

[4] M. Sudha, A. Gandhi, G. Ajay, P. Deepika, and M. Kishore Kumar, "IoT-based smart street light fault detection and real-time fault location identification system using wireless sensor networks and GPS," *International Journal of Latest Technology in Engineering, Management & Applied Science*, vol. 14, no. 8, pp. 964-972, 2025, doi: 10.51583/IJLTEMAS.2025.1408000125.

[5] V. Hima Vamsi, A. Supraja Reddy, P. Sathish, B. Neeraja, and M. Vinodh Kumar, "Sensor enabled centralised monitoring system for streetlight fault detection using IoT," *Sensing and Imaging*, vol. 25, Art. no. 47, 2024, doi: 10.1007/s11220-024-00500-6.

[6] A. Banks and R. Gupta, Eds., "MQTT Version 3.1.1," OASIS Standard, Oct. 2014.

[7] Allegro MicroSystems, "ACS712: Fully integrated, Hall-effect-based linear current sensor IC," datasheet.

[8] Espressif Systems, "ESP32 series datasheet," datasheet.