# Iot-Based Smart Street Light Fault Detection And Reporting System Using ESP32 And Blynk Cloud Platform

**Emil Nithish Rani M D**
Dept of Computer Applications
Dr. M.G.R. Educational and Research Institute Chennai,
Tamil Nadu, India

***Abstract-*** *Traditional street lighting operates on fixed schedules with no fault detection, causing energy waste, delayed maintenance, and safety risks. This paper proposes an IoT-Based Smart Street Light Fault Detection and Adaptive Brightness Control Framework leveraging ESP32 microcontrollers, LDR, ACS712 current sensors, voltage sensors, and PIR motion detectors. The Blynk cloud platform provides remote monitoring via mobile apps, while MQTT enables lightweight device-to-cloud communication. Threshold-based anomaly detection identifies lamp failures, voltage irregularities, and connectivity disruptions. Results show ~45% energy reduction and fault detection within 15 seconds. The modular architecture supports future AI-driven predictive maintenance, solar integration, and smart city interoperability.*

***Keywords:*** IoT, Smart Street Lighting, ESP32, Blynk Cloud, Adaptive Brightness, Fault Detection, MQTT, Energy Efficiency, Smart City.

## I. INTRODUCTION

Urban street lighting is a critical component of smart city infrastructure, influencing public safety, energy consumption, and municipal expenditure. Conventional systems operate at constant illumination regardless of ambient conditions, lacking sensor-based awareness, automated fault detection, and adaptive control. This results in significant energy wastage, delayed maintenance, and elevated operational costs.

Existing solutions—Zigbee, GSM-based control, SCADA—suffer from limited scalability, high infrastructure costs, or insufficient real-time responsiveness, and none unify energy optimization with fault detection under a mobile-friendly interface.

This paper proposes a unified IoT framework integrating ESP32 edge nodes, multi-sensor arrays, the Blynk cloud platform, and MQTT protocol to address these gaps.

**Primary Contributions:**

- Comprehensive IoT architecture unifying adaptive brightness control and automated fault detection.
- ESP32 edge nodes with multi-sensor integration for real-time environmental monitoring and autonomous decision-making
- Blynk cloud platform integration for cross-platform mobile monitoring, alerts, and control.
- Threshold-based and trend-analysis fault detection covering lamp burnout, ballast malfunction, and power irregularities.
- Scalable, modular architecture supporting AI-driven predictive analytics, renewable energy, and smart city integration.

## II. RELATED WORK

Early smart lighting relied on timer-based controls and photocells—basic on/off automation without fault detection. Zigbee mesh networks demonstrated reliable short-range low-power connectivity but required dedicated gateways and had limited interoperability. GSM-based systems provided wide-area coverage but incurred recurring costs and latency. Arduino/Raspberry Pi WiFi implementations proved adaptive lighting feasibility, yet lacked comprehensive fault detection and trend analysis.

Cloud platforms (ThingSpeak, Firebase, Blynk) emerged as IoT hosts; Blynk gained traction for its intuitive mobile builder and multi-hardware support, though its application in municipal street lighting remained unexplored. Energy-optimization algorithms (traffic density, time-of-day, weather dimming) achieved substantial savings but often compromised safety or required complex prediction models.

Existing solutions consistently exhibit one or more gaps: isolated objectives (energy OR fault detection, not both), lack of mobile-friendly maintenance interfaces, scalability

limitations, and underdeveloped security. The proposed framework addresses all these gaps in a single unified system.

## III. System Architecture

The system follows a four-layer architecture: Presentation, Application, Security, and Database layers, supporting distributed sensing, edge processing, cloud connectivity, and user interaction.

### A. Presentation Layer

Implemented via the Blynk mobile application framework for cross-platform (iOS/Android) real-time visualization. Dashboards display individual luminaire status, zone-level aggregation, energy metrics, and fault alerts. Supports multiple user roles (admin / maintenance), interactive remote brightness control, manual overrides, and geographic visualization via mapping services integration.

### B. Application Layer

Built on Node.js/Express. Handles data processing, business logic, workflow management, and inter-module coordination. Five major modules:

- Sensor Data Acquisition – collects real-time LDR, current, voltage, and PIR data from ESP32 nodes.
- Adaptive Brightness Control – dynamically adjusts PWM based on ambient light, time-of-day, and motion events.
- Fault Detection & Diagnostics – identifies lamp failure, electrical anomalies, and communication disruptions via threshold + trend analysis.
- Alert Management – priority-classifies faults; dispatches Blynk push, email, and SMS notifications.
- Energy Analytics & Reporting – aggregates consumption data, calculates efficiency KPIs, generates periodic reports.

### C. Security Layer

- TLS encryption on all ESP32 ↔ MQTT broker ↔ Blynk transmissions.
- MQTT username/password authentication with ACL for topic-level access control.
- Role-based access control (RBAC) restricting configuration to admins.
- Blynk token-based device authentication; unique token per hardware node.
- JWT authentication on all API endpoints; rate limiting to prevent DoS attacks.

### D. Database Layer (MongoDB)

| Collection | Contents |
|---|---|
| USER_CREDENTIALS | Bcrypt-hashed accounts, roles, auth tokens |
| SENSOR_READINGS | Time-stamped LDR, current, voltage, PIR data per node |
| OPERATIONAL_STATUS | Brightness level, active mode, connectivity per luminaire |
| FAULT_EVENTS | Anomaly severity, timestamp, node ID, resolution status |
| ENERGY_METRICS | Aggregated power consumption for efficiency analysis |

## IV. METHODOLOGY

### A. User Registration & Data Collection

Authorized users (admins, technicians, operators) are registered with bcrypt-hashed credentials and role assignments stored in MongoDB. Each ESP32 node integrates: LDR (ambient light), ACS712 current sensor (power monitoring), voltage divider (supply voltage), and HC-SR501 PIR (motion detection). Sensors polled every 30 s for environmental parameters; PIR monitored continuously with debounce logic. Calibration during deployment establishes operational baselines.

### B. Adaptive Brightness Algorithm

The on-device algorithm continuously evaluates ambient light against configurable thresholds:

- Daylight (LDR > dusk threshold): luminaires OFF.
- Dusk/Night (LDR < threshold): luminaires activated at base brightness sufficient for visibility.
- Motion detected: brightness elevated to 100% for 30 s, then smoothly dimmed back (gradual curve).
- Time-of-day weighting: higher base brightness during evening peak hours; reduced during late-night low-traffic periods.

### C. Fault Detection Processing

- Current near zero → lamp failure or ballast malfunction.
- Elevated current above threshold → short-circuit or ballast degradation.
- Voltage outside nominal ±5% band → power supply irregularity flagged.
- Missed heartbeat beyond threshold → connectivity fault classification.
- Trend analysis monitors gradual drift in addition to hard threshold crossings.

### D. Security & Verification Mechanism

Each ESP32 is provisioned with a unique device ID and authentication token. Unauthorized connection attempts are rejected and logged. Operational commands include sequence numbers and timestamps preventing replay attacks. All configuration changes, overrides, and adjustments are audit-logged with user ID and timestamp.

### E. Encryption & Data Protection

- WiFi: WPA2-PSK with strong passphrase.
- MQTT: TLS encryption over internet links.
- Blynk: end-to-end encryption (mobile ↔ hardware).
- ESP32 flash: WiFi credentials, MQTT addresses, tokens stored encrypted (Arduino EEPROM encryption library).
- DB credentials: bcrypt with salt; API: HTTPS with certificate pinning (anti-MITM).

### F. Reporting & Analytics

- Energy summaries by zone, time period, vs. baseline.
- Fault reports: detection timestamps, resolution time, root cause classification.
- Uptime statistics: availability % per luminaire and zone.
- Brightness adjustment frequency analysis for base-level optimization.
- Maintenance response time analytics (fault detection → technician acknowledgment).

## V. IMPLEMENTATION

### Technology Stack

| Layer | Technology | Role |
|---|---|---|
| Frontend | Blynk Mobile App, HTML Dashboard | UI, remote monitoring |
| Backend | Node.js, Express | Business logic, data processing, API |
| Database | MongoDB 6.0 | Time-series storage & analytics |
| Security | TLS/SSL, JWT, bcrypt | Encrypted comms & access control |
| Core (Edge) | ESP32, Arduino IDE, MQTT | Edge computing & sensor integration |
| Dev Tools | Arduino IDE 2.2.1, VS Code | Firmware & application development |
| MQTT Broker | Mosquitto 2.0.15 + TLS | Lightweight IoT messaging |

### Module Descriptions

#### 1) Authentication Module

Manages user registration, login, and RBAC. Bcrypt hashing before DB storage; JWT session tokens with configurable expiration. Supports TOTP-based MFA for admin accounts.

#### 2) Data Management Module

ESP32 firmware buffers sensor data during connectivity loss for resilient ingestion. Node.js RESTful API provides historical retrieval, WebSocket real-time streaming, and bulk export. Automated data retention archives records beyond configured thresholds while preserving aggregated long-term statistics.

#### 3) Processing & Analytics Module

Evaluates LDR readings, PIR events, and time-of-day parameters to compute optimal PWM dimming levels. Applies statistical analysis (baseline deviation, trend monitoring) to current/voltage streams. Energy analytics aggregate across spatial and temporal dimensions, generating efficiency metrics and cost projections.

#### 4) Monitoring Module

Blynk app and web dashboard display luminaire status with color-coded indicators (operational / brightness level / fault). Zone-level summaries provide management overview. Push notifications dispatched on fault detection with severity-based prioritization.

#### 5) Reporting Module

Automated daily / weekly / monthly reports distributed to configured email recipients. On-demand custom reports via configurable parameter selection with PDF, CSV, and JSON export. Municipal energy management system integration for city-wide sustainability reporting.

## VI. EXPERIMENTAL EVALUATION

### A. Test Environment

Controlled laboratory environment simulating municipal street lighting conditions:

- Hardware: ESP32 DevKit V1 + LDR, ACS712, HC-SR501 PIR, voltage divider, high-power LED arrays with MOSFET PWM drivers.
- Software: Arduino IDE 2.2.1, Node.js 18.16.0 / Express 4.18.2, MongoDB 6.0, Mosquitto MQTT 2.0.15 (TLS enabled), Blynk IoT platform.
- Network: Local WiFi infrastructure; 100 Mbps broadband internet.

### B. Test Cases & Results

| TC | Objective | Expected Output | Result |
|---|---|---|---|
| TC-01 | User Registration & Auth | Secure account creation + role assignment | PASS |
| TC-02 | Sensor Acquisition & Calibration | Accurate environmental measurements | PASS |
| TC-03 | Adaptive Brightness Control | Dynamic illumination adjustment | PASS |
| TC-04 | Motion-Based Activation | Brightness elevation on PIR detection | PASS |
| TC-05 | Fault Detection & Alert | Alert notification dispatch | PASS |
| TC-06 | Encrypted Transmission | Secure MQTT/TLS communication | PASS |
| TC-07 | Remote Monitoring (Blynk) | Real-time status display on mobile | PASS |
| TC-08 | Report Generation & Analytics | Automated report output | PASS |

### C. Performance Analysis

Key quantitative results:

- Energy savings: ~45% reduction vs. constant illumination; greatest during late-night low-traffic periods.
- Fault detection latency: failures identified within 15 s; mobile alert delivered within 5 s of detection.
- Voltage anomaly sensitivity: deviations as small as 5% from nominal detected, enabling early-stage degradation identification.
- System uptime: >99% connectivity during evaluation period.
- Scalability: multi-node simulation confirmed Node.js/MongoDB backend handles municipal-scale deployment with appropriate server resources.
- ESP32 processing: handled multi-sensor polling, brightness calculation, and MQTT within the 30 s sampling interval without degradation.

### D. Security Analysis

- TLS: packet capture confirmed encrypted MQTT payload transmission; no plaintext leakage.
- JWT: expired and malformed tokens correctly rejected; unauthorized API access prevented.
- bcrypt: credential hashing prevents reverse computation from DB exposure.
- Audit logs: comprehensive records of logins, config changes, overrides, and fault acknowledgments with user ID and timestamp.
- Integrity: checksum validation on sensor readings; corrupted/tampered packets discarded and flagged.

## VII. DISCUSSION

The framework successfully demonstrates that adaptive illumination control and automated fault detection can coexist within a single, cost-effective, municipal-grade architecture. ESP32 hardware with commodity sensors and the Blynk platform eliminates proprietary infrastructure dependency, lowering deployment costs.

The adaptive brightness algorithm balances safety with energy conservation; multi-parameter fault detection covers common failure modes; the Blynk mobile interface enables non-technical maintenance personnel to respond to alerts efficiently.

**Applicability beyond street lighting:**

- Parking lot and campus lighting networks.
- Industrial facility perimeter lighting.
- Recreational area illumination.
- Integration with GIS for spatial maintenance planning.

**Limitations:**

- WiFi dependency: limited range in areas with sparse network infrastructure (mitigation: cellular modem or LoRaWAN gateway).
- Threshold-based fault detection only; no ML-based predictive failure identification.
- No battery backup: functionality disrupted during extended power outages.

**Future Work:**

- ML-based predictive maintenance to detect degradation patterns before complete failure.
- Solar panel + battery storage for off-grid/resilient operation.
- LoRaWAN protocol support for extended deployment range.
- Smart city platform and municipal traffic management system integration.
- Computer vision-based pedestrian/vehicle detection to supplement PIR sensing.

## VIII. CONCLUSION

This paper presented an IoT-Based Smart Street Light Fault Detection and Adaptive Brightness Control Framework integrating ESP32 microcontrollers, multi-sensor

environmental monitoring, the Blynk cloud platform, and MQTT within a unified, practically deployable architecture.

Experimental evaluation confirmed: (1) ~45% energy reduction vs. conventional constant-output systems; (2) automated fault identification within 15 s with 5 s mobile alert dispatch; (3) intuitive Blynk interface enabling non-expert maintenance operation; (4) robust security via TLS + JWT + bcrypt + audit logging. The modular, scalable design supports future AI analytics, renewable energy integration, and advanced communication protocols

## REFERENCES

[1] Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[2] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An Information Framework for Creating a Smart City Through Internet of Things," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 112–121, Apr. 2014.

[3] P. Mohanty, S. P. Choudhury, and P. K. Raut, "IoT Based Smart Street Light Management System," *International Journal of Engineering Research and Technology*, vol. 9, no. 5, pp. 245–250, 2020.

[4] S. S. Kumar, S. S. Kumar, and A. Kumar, "Smart Street Light System Using IoT," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 9, no. 5, pp. 23–27, May 2020.

[5] H. T. Dath, T. T. H. Le, and H. H. Nguyen, "Design and Implementation of Smart Street Lighting System Based on IoT Technology," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 4, pp. 3865–3872, Aug. 2020.

[6] R. K. Kodali and S. Sorat, "An IoT Based Street Light Management System," *International Conference on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, pp. 1124–1129, 2017.

[7] S. S. Gaikwad, A. N. Gaikwad, and P. Potdar, "IoT Based Smart Street Light Management System," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 6, no. 4, pp. 3456–3461, Apr. 2018.

[8] S. Bhardwaj, A. Gupta, and R. Kumar, "IoT Based Smart Street Light System for Smart City," *International Journal of Engineering Research and Technology*, vol. 8, no. 6, pp. 234–239, 2019.

[9] P. P. Ray, "A Survey on Internet of Things Architectures," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, Jul. 2018.

[10] S. Li, L. Da Xu, and S. Zhao, "The Internet of Things: A Survey," *Information Systems Frontiers*, vol. 17, n