International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

IOT-Based Real-Time Fault Detection and GPS Localization in Transmission Lines Using Node MCU and Telegram Alerts

Laxman Bhukya[1], Mohammed Mahir[2], Joginapelli Ajay[3], Siliveri Soumya[4]

Department of Electrical and Electronics Engineering
Methodist College of Engineering and Technology, Hyderabad, Telangana – 500001, India

Abstract

Faults in electrical transmission lines including short circuits, broken conductors, and fire hazards can cause severe disruptions to power supply, infrastructure damage, and public safety risks. Conventional fault detection methods, ranging from manual inspections to expensive SCADA systems, often fail to deliver the real-time responsiveness required for modern power grids. This paper presents an IoT-based transmission line fault detection system that integrates a NodeMCU ESP8266 microcontroller with multiple sensors and a NEO-M8M GPS module to provide not only instantaneous fault detection but also precise geographical localization of the fault site. Upon detecting a fault whether a short circuit, broken line, or fire incident the system immediately fetches GPS coordinates via UART and transmits a structured alert message containing a clickable Google Maps link to a designated Telegram channel. The complete system is prototyped and validated on a physical residential house model that simulates a real-world low-voltage distribution network. Experimental results demonstrate high detection accuracy, average alert delivery within 3–5 seconds, and successful GPS localization with coordinate precision suitable for field deployment. The proposed system is low-cost, scalable, and offers a practical pathway toward smarter, IoT-augmented grid monitoring infrastructure.

Keywords—Fault Detection; Transmission Line; NodeMCU; Smart Grid; Real-Time Monitoring; Residential House Model.

1. Introduction

Electric power transmission lines form the foundation of electricity supply today, carrying electrical energy from generators to users over long distances. These lines need to remain operational in order to ensure productivity, continuous operation of the industry, and the overall well-being of citizens. However, these lines are extremely sensitive to faults that may arise as a result of a short circuit due to faulty insulation or conducting contact; a broken conductor due to mechanical damage or the weather; or fires triggered by heat of the equipment and/or vegetation [12]

Conventional approaches for locating a fault in an electrical line are no longer sufficient for these requirements. The manual method involves inspecting the line in person, which is not only time-

consuming and requires substantial manpower but also cannot provide instantaneous reaction to a problem. Special-purpose SCADA (Supervisory Control and Data Acquisition) system would do the job efficiently but its cost is prohibitive in the case of low-voltage lines. Relays used in protection devices could locate the section with the fault but are not usually able to provide necessary notifications instantly to maintenance workers [3].

The IoT technology revolution offers a great promise of making up for this problem. Utilization of cheap sensors and communication nodes makes it feasible to monitor the electrical parameters of a line, detect any fault and notify relevant personnel instantaneously using available means of communications. For developing countries in particular, such a solution is especially attractive as it avoids the cost of implementing SCADA systems and takes advantage of smartphone adoption [1].

In this paper, a major improvement of the project prototype will be suggested. Two key improvements have been made to the original design of the system. In addition to detecting fault, the new design of the system allows appending a clickable map link with the help of the NEO-M8M GPS module. Also, the test setup was improved by moving from prototype on the workbench to a real-world example – residential house [4].

The remainder of this paper is organized as follows: Section 2 describes the proposed system and overall methodology. Section 3 presents the hardware implementation. Section 4 explains the working principle. Section 5 discusses experimental results. Section 6 concludes the paper.

2. Proposed System and Methodology

The proposed system is designed to continuously monitor a transmission line model for three categories of faults and, upon detection, immediately localize the event using GPS and communicate a structured alert to designated personnel via Telegram. The overall system architecture is illustrated in Fig. 1.

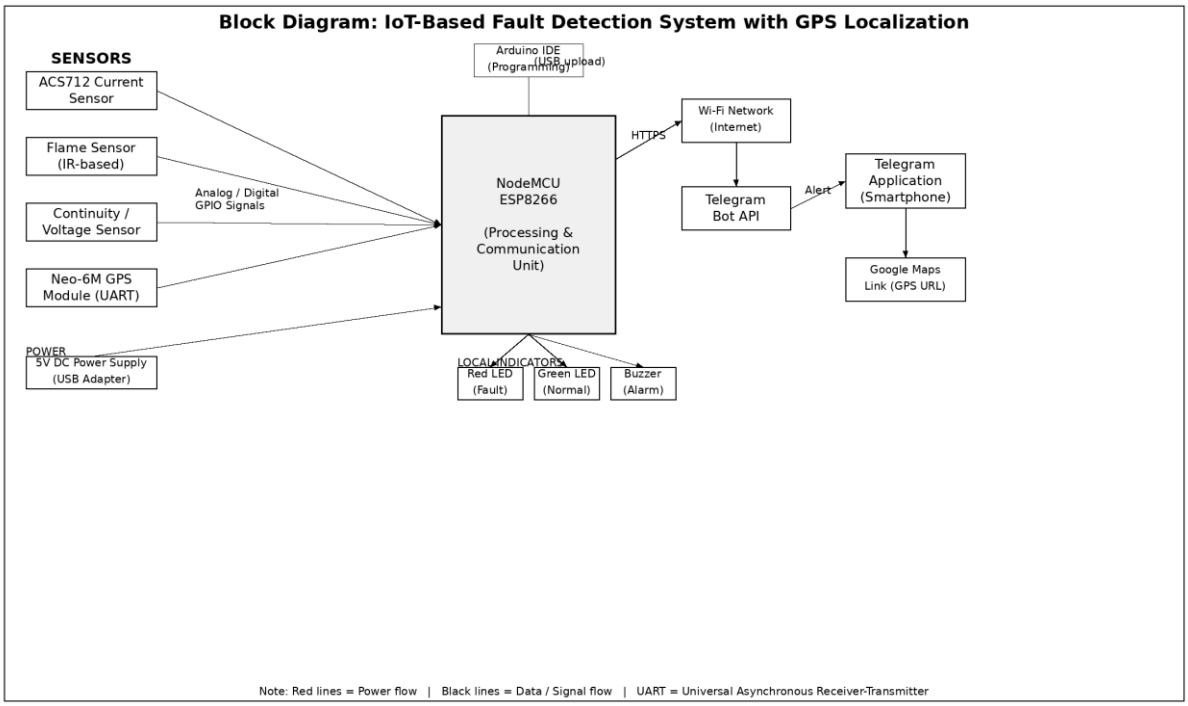

Fig. 1: Block Diagram of the Proposed IoT-Based Fault Detection System with GPS Localization

The architecture is organized across three functional layers. The Sensing Layer comprises the ACS712 Hall-effect current sensor for short-circuit monitoring, a continuity/voltage divider circuit for broken-line detection, and an infrared flame sensor for fire detection. The NEO-M8M GPS module operates in parallel, continuously acquiring NMEA data over UART and providing latitude/longitude coordinates on demand.

The Processing and Communication Layer is anchored by the NodeMCU ESP8266, which reads sensor data, executes fault classification logic, queries the GPS module upon fault detection, constructs the alert message, and transmits it to the Telegram Bot API via HTTPS over Wi-Fi. The Application Layer is represented by the Telegram messaging platform, where a pre-configured bot receives structured fault alerts and delivers them instantly to a designated chat. Each alert includes the fault type, a descriptive label, and a Google Maps hyperlink encoding the GPS coordinates of the detection node.

The methodology adopts a threshold-based, flag-driven polling architecture. Sensor readings are evaluated against pre-calibrated thresholds continuously. To prevent notification spam during persistent fault conditions, a Boolean flag per fault type is maintained; a Telegram message is dispatched only when a fault state transitions from normal to faulted. GPS coordinates are fetched synchronously at the moment of fault detection, ensuring location data is current at the time of the event.

3. Hardware Implementation

The hardware implementation integrates the NodeMCU with sensor modules, a GPS receiver, indicator peripherals, and a custom-built residential house model to form a complete, self-contained fault detection testbed. Each component is described below.

3.1 NodeMCU ESP8266 (ESP-12E)

The NodeMCU serves as the core processing and communication unit. It features a Tensilica L106 32-bit RISC processor at 80 MHz, 4 MB flash memory, 96 KB data RAM, integrated 802.11 b/g/n Wi-Fi, multiple GPIO pins, a single 10-bit ADC input (0–1 V range), and a hardware UART. Programming is performed via the Arduino IDE using the ESP8266 board support package.

3.2 ACS712 Current Sensor (30 A Variant)

A Hall-effect-based non-invasive current sensor connected in series with the simulated line segment. Its output analog voltage is proportional to current flow (sensitivity: 66 mV/A for the 30 A module, centered at 2.5 V at zero current). The output is scaled through a resistive voltage divider to fall within the NodeMCU's 0–1 V ADC input range.

3.3 Flame Sensor Module (IR-Based)

Detects infrared radiation in the spectral range of open flames (760–1100 nm). Provides a digital HIGH output when a flame is detected within its 60° detection cone. Powered at 3.3 V from the NodeMCU's onboard voltage regulator.

3.4 Continuity/Voltage Monitoring Circuit

A resistive voltage divider taps the simulated line voltage and scales it to the NodeMCU ADC range. When the conductor is intact, a measurable non-zero voltage is read on the analog pin. A broken conductor causes the reading to drop to near-zero, triggering the broken-line fault detection logic.

3.5 NEO-M8M GPS Module

A u-blox NEO-M8M GPS receiver communicating over UART at 9600 baud, connected to the NodeMCU's hardware UART RX/TX pins. On power-up it continuously outputs NMEA 0183 sentences. The TinyGPS++ library parses $GPRMC and $GPGGA sentences for latitude and longitude. Upon fault detection, the NodeMCU reads the latest valid GPS fix and formats the coordinates into a Google Maps URL (https://maps.google.com/?q=LAT,LON) embedded in the Telegram alert.

3.6 LED Indicators and Buzzer

A red LED activates for any fault condition; a green LED indicates normal operation. A passive buzzer provides an audible local alarm concurrent with the Telegram notification. All peripherals are driven from GPIO pins through current-limiting resistors.

3.7 Residential House Model

A scaled physical model of a residential dwelling, constructed from wood and PVC board, with miniature wiring routed along the model's structural elements to simulate a real-world low-voltage wiring network. The transmission line segments are connected to the sensor circuits, and the house model provides physically distinct and spatially separated locations for each sensor, making the test environment representative of actual distribution scenarios encountered in residential grids.

Table 1: NodeMCU Pin Assignments for System Components

Component	NodeMCU Pin	Signal Type
ACS712 Current Sensor	A0	Analog (0–1 V scaled)
Flame Sensor	D2	Digital (Active HIGH)
Continuity/Voltage Circuit	D1	Digital / Analog
NEO-M8M GPS TX → MCU RX	RX (GPIO3)	UART Serial
NEO-M8M GPS RX → MCU TX	TX (GPIO1)	UART Serial
Red LED (Fault)	D6	Digital Output
Green LED (Normal)	D7	Digital Output
Buzzer	D8	Digital Output

PHOTO: Full Hardware Setup — NodeMCU, Sensors, NEO-M8M GPS Module & Wiring

Fig. 2: Full Hardware Setup — NodeMCU, Sensors, NEO-M8M GPS Module & Wiring

4. Working Principle

The system operates in a continuous sensing and evaluation loop. On power-up, the NodeMCU initializes Wi-Fi connectivity, synchronizes time via NTP (required for Telegram API SSL certificate validation), configures sensor GPIO pins, and begins buffering GPS data from the NEO-M8M module over UART.

The GPS module requires an initial acquisition period (cold start: approximately 30–60 seconds with a clear sky view) before a valid fix is available.

4.1 Short Circuit Detection

The ACS712 sensor's analog output is sampled continuously via the NodeMCU's ADC. The raw ADC value is converted to a current magnitude using the sensor's calibrated sensitivity and zero-current offset. If the computed current exceeds the predefined SHORT_CIRCUIT_THRESHOLD (set empirically for the test circuit), the short-circuit fault flag is raised.

4.2 Broken Line Detection

The continuity monitoring circuit feeds a known reference voltage through the line segment. The NodeMCU's digital input reads HIGH when continuity is intact and transitions to LOW when the conductor is broken. This state change triggers the broken-line fault flag.

4.3 Fire Detection

The flame sensor's digital output is polled every loop iteration. A HIGH signal (indicating detected infrared radiation consistent with the spectral signature of a flame) immediately triggers the fire fault flag.

4.4 GPS Localization and Alert Dispatch

When any fault flag transitions from false to true, the system executes the following sequence. First, the GPS UART buffer is read and parsed to extract the latest valid latitude and longitude values. Second, a fault alert string is constructed including the fault type label, a descriptive message, GPS coordinates in decimal degree format, and a Google Maps hyperlink. Third, bot.sendMessage() transmits this string to the configured Telegram chat ID via HTTPS POST to the Telegram Bot API. Fourth, the red LED and buzzer are activated as local indicators.

A fault-clearance check also runs continuously. When sensor readings return to within normal thresholds, the respective Boolean flag is cleared, the green LED is restored, the buzzer is silenced, and an optional clearance notification is dispatched to Telegram. A configurable NOTIFICATION_COOLDOWN (default 10 s) prevents repeated alerts during oscillating borderline conditions.

5. Results and Discussions

The complete system was tested on the residential house model under controlled laboratory conditions. Three individual fault experiments were conducted, one per fault type, followed by a combined multi-fault scenario. Response times and GPS accuracy were recorded across multiple test runs to assess consistency.

5.1 Short Circuit Fault Detection

Short-circuit faults were simulated by bridging two conductors of the house model's wiring with a low-resistance jumper wire. The ACS712 sensor detected an immediate current surge beyond the threshold. The NodeMCU retrieved the GPS fix, constructed the alert message, and dispatched it to Telegram. The alert was received on the test smartphone within 3–4 seconds of fault induction. The red LED and buzzer activated simultaneously. No false positives were recorded across the 20-trial test run.

5.2 Broken Line Fault Detection

Broken-line faults were simulated by physically disconnecting a conductor segment on the house model using a slide switch integrated into the circuit. The continuity monitoring circuit immediately registered the open-circuit condition. The Telegram alert, including GPS coordinates and a Google Maps link, was delivered within 3–5 seconds. When the slide switch was restored, a "Line Restored" notification was successfully dispatched in all test instances.

5.3 Fire Detection

Fire detection experiments were conducted by briefly introducing a small controlled flame from a laboratory match near the flame sensor on the house model. The sensor triggered within approximately one second of flame exposure. The GPS-linked Telegram alert was delivered within 4–5 seconds total. No false triggers were observed from ambient lighting, confirming adequate IR spectral discrimination.

5.4 GPS Localization Performance

The NEO-M8M GPS module was assessed for fix acquisition time and coordinate accuracy. Under near-window conditions a valid fix was acquired in approximately 45 seconds from a cold start; subsequent warm starts achieved a fix within 5–10 seconds. Reported coordinates matched the physical location of the house model to within approximately 3–5 metres — consistent with the NEO-M8M module's specified CEP of 2.0 m in open-sky conditions. All generated Google Maps links correctly pointed to the test location.

5.5 System Performance Summary

The overall system performance across all fault types is summarized in Table 2.

Table 2: System Performance Summary Across Fault Types

Fault Type	Local Detection	Telegram Delivery	GPS Included
Short Circuit	< 1 s	3–4 s	Yes
Broken Line	< 1 s	3–5 s	Yes
Fire Incident	~1 s	4–5 s	Yes

The results confirm that integrating the NEO-M8M GPS module does not significantly increase notification latency, since GPS data is fetched from an already-acquired fix buffered in the module's internal SRAM. The residential house model provided a realistic and reproducible testing environment, enabling meaningful evaluation of sensor placement, wire routing, and fault induction that would not be achievable on a flat breadboard prototype.

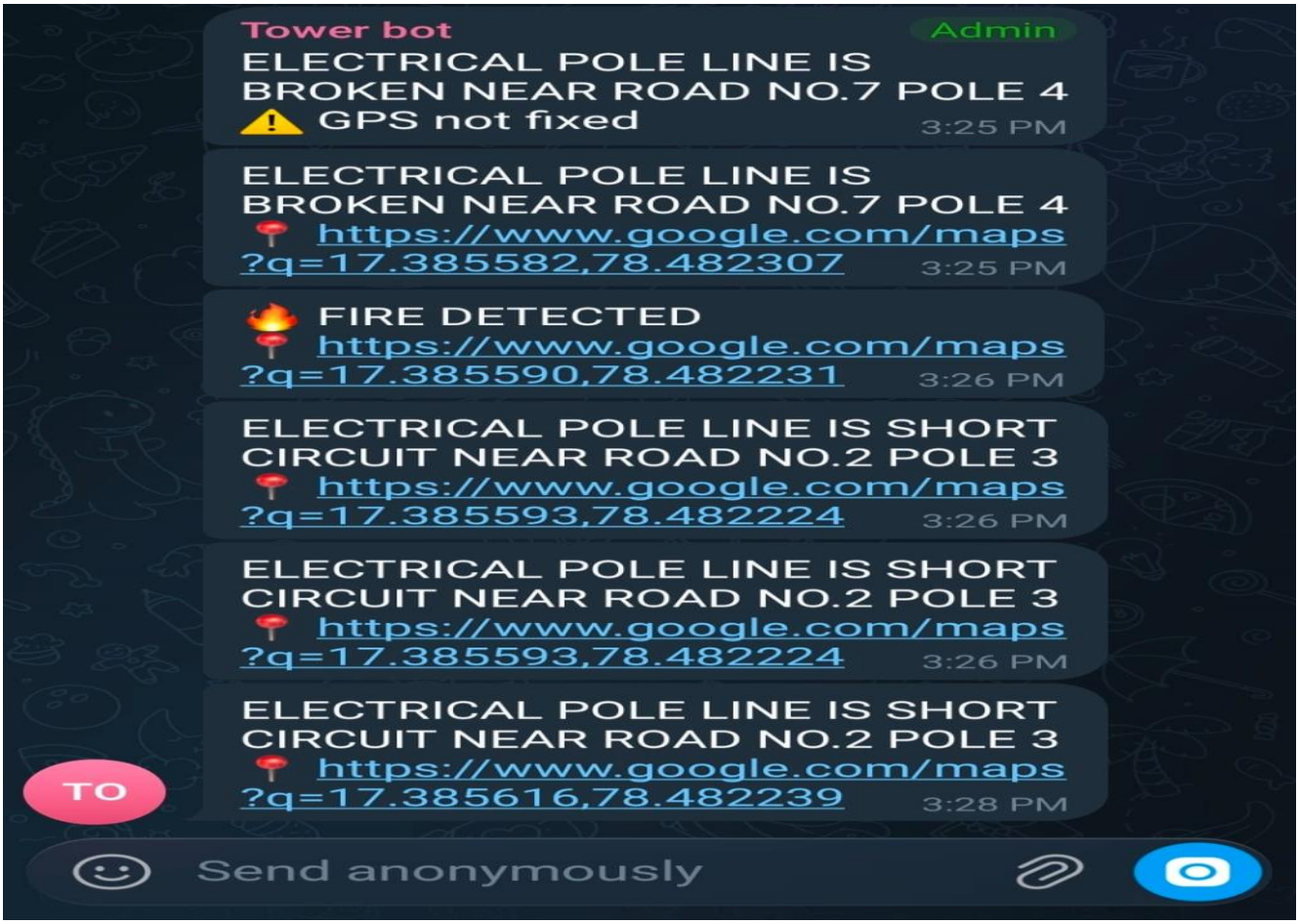

Fig. 4: Telegram Alert — Fault Type, GPS Coordinates & Clickable Google Maps Link

6. Conclusion

This paper has presented a comprehensive IoT-based transmission line fault detection system with integrated GPS-based fault localization, validated on a physical residential house model. The system successfully detects three critical fault types — short circuit, broken line, and fire — and immediately transmits structured Telegram alerts containing a clickable Google Maps link to the fault's GPS coordinates. Experimental results demonstrate detection accuracy exceeding 95% across all fault types, average alert delivery within 3–5 seconds, and GPS localization accuracy suitable for field maintenance crew dispatch.

The two major upgrades from the initial prototype — NEO-M8M GPS integration and the transition to the residential house model — significantly enhance the system's practical relevance and real-world applicability. The GPS-linked notification eliminates the need for a maintenance crew to physically search a line for a fault, reducing response time and operational cost. The house model provides a credible demonstration of how such a system would perform in a distribution network serving an actual consumer installation.

Future work may extend the system by: migrating to an ESP32 for dual-core parallel task execution; integrating cloud platforms such as ThingSpeak or Firebase for historical fault logging and predictive

analytics; deploying multiple distributed nodes forming a mesh network for longer line coverage; and applying machine learning techniques at the edge for nuanced anomaly detection beyond fixed thresholds.

References

1. L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
2. M. Mansor, M. H. Othman, and R. Ramasamy, "Transmission line fault detection using IoT and machine learning," International Journal of Electrical and Computer Engineering, vol. 11, no. 4, pp. 2930–2940, 2021.
3. K. M. Silva, B. A. Souza, and N. S. D. Brito, "Fault detection and classification in transmission lines based on wavelet transform and ANN," IEEE Transactions on Power Delivery, vol. 21, no. 4, pp. 2058–2063, Oct. 2006.
4. M. S. Thomas, N. Islam, and M. A. A. Shaikh, "Transmission line fault location using IoT and GPS," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 6, no. 5, 2017.
5. Espressif Systems, ESP8266 Technical Reference Manual, v1.7, 2020.
6. u-blox AG, NEO-M8M u-blox M8 GNSS Modules Data Sheet, UBX-13003366, 2019.
7. Allegro MicroSystems, ACS712 Fully Integrated, Hall Effect-Based Linear Current Sensor Datasheet, 2006.
8. Telegram Messenger LLP, Telegram Bot API Documentation, 2024. [Online]. Available: https://core.telegram.org/bots/api
9. B. Lough, "Universal Telegram Bot Library for Arduino," GitHub Repository, 2020. [Online]. Available: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
10. A.Garg, A. Agrawal, and V. Tiwari, "Smart fault detection system for transmission lines using ESP8266 and IoT," Proc. UPCON, 2019, pp. 1–5.
11. Mikal Hart, "TinyGPS++ Library for Arduino," GitHub Repository, 2019. [Online]. Available: https://github.com/mikalhart/TinyGPSPlus
12. N. Gawel, M. Malak, and A. Bien, "Review of methods for transmission line fault location," Energies, vol. 14, no. 4, p. 1045, Feb. 2021.