INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 10 ISSUE: 10 | OCT - 2026 SJIF RATING: 8.659 ISSN: 2582-3930

Hazardous Electric Pole Fault Detection and Protection System Using Arduino, GSM and GPS

Sudhamalla Mallesh
Department of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, India
mallesh.ece4@gmail.com

Sanda Ramu
Department of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, India
ramukanna999999@gmail.com

A Rishendhar
Department of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, India
rishendher@gmail.com

T Shiva Sai
Department of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, India
tshivasai0143@gmail.com

Indirala Ruthvik
Department of Electronics and Communication Engineering
CMR Technical Campus
Hyderabad, India
237r1a0423@cmrtc.ac.in

***Abstract*—Broken conductors, cracked insulators and leakage onto the pole body are a leading cause of electric shock accidents along overhead distribution lines, because a live fault usually stays invisible until someone touches it. This paper presents a low-cost Hazardous Electric Pole Fault Detection and Protection System built around an Arduino Uno microcontroller. A sensing stage watches the pole line for abnormal conditions. When a fault is confirmed, the controller drives a relay to cut off the supply to the affected section, lights a warning LED and sounds a buzzer so that people nearby are alerted at once. In parallel, a GPS receiver reads the pole's coordinates and a GSM modem sends an SMS carrying the fault type and a map link to the maintenance office, which removes the need to search for the faulty pole. The firmware is written in Arduino C++ and follows a simple detect, protect, alert and report sequence. Because every component is inexpensive and widely available, the system can be fitted to existing poles in rural and semi-urban feeders without major changes to the network. The design aims to shorten the time between a fault occurring and the line being made safe, and so to reduce the risk to the public and to line workers.**

Keywords*—Electric pole, fault detection, line protection, Arduino Uno, GSM, GPS, relay, public safety, embedded system.

I. INTRODUCTION

Electric poles carry power to homes, farms and streets, and they stand in places where people walk, children play and animals graze. When a conductor snaps, an insulator fails or a stay wire touches a live line, the pole and the ground around it can become energised without any visible sign. Accidents of this kind usually happen because nobody knows about the fault, and the supply stays on until a passer-by or a lineman discovers it the hard way.

In most distribution networks today, a faulty section is found only when consumers phone the utility or when a patrol crew happens to see the damage. Locating the exact pole can take hours on long rural feeders, and during that time the hazard remains. Protective devices such as fuses and circuit breakers at the substation do operate on heavy short circuits, but a high-resistance earth fault or a conductor lying on dry ground may draw too little current to trip them. A system that works at the level of the individual pole, and that tells people where the problem is, therefore fills a real gap.

Embedded electronics now make such a system affordable. A microcontroller board can read a sensor, make a decision in milliseconds and switch a relay, while a cellular modem and a satellite positioning receiver can report the event together with its location. Open-source platforms such as Arduino keep both the hardware and the programming effort low, which suits small utilities, local bodies and student projects alike [1], [2].

This paper describes a Hazardous Electric Pole Fault Detection and Protection System that combines these ideas on an Arduino Uno. The controller continuously monitors the pole line. On detecting a fault it (i) isolates the section through a relay, (ii) warns the public locally with an LED and a buzzer, and (iii) sends the pole's GPS coordinates to the maintenance team in an SMS over the GSM network. The remaining sections review related work, describe the hardware architecture and the control method, explain the implementation, discuss the expected behaviour of the prototype, and conclude with possible extensions.

II. RELATED WORK

Research on monitoring overhead distribution lines falls into a few broad groups. The first group covers conventional protection, in which over-current relays, fuses and reclosers at the substation or along the feeder disconnect the line when the current exceeds a set value. These devices are dependable for bolted short circuits, but they act on a whole feeder section and do not tell the operator which pole is affected. Reliability studies of distribution networks also show that a large share of outage time is spent locating the fault rather than repairing it [6].

A second group of studies uses microcontroller-based monitoring. Here a low-cost board reads current, voltage or tilt sensors mounted on or near the pole and flags abnormal readings. Many of these systems stop at local indication, such as an LED or an alarm, and therefore help only people who happen to be standing close to the pole.

A third group adds remote communication. GSM modems, which send SMS messages over the existing cellular network, are popular for this purpose because they need no new infrastructure and work in areas without internet coverage [4]. Some designs report only that a fault has occurred, which still leaves the crew with a long search. Others attach a location, usually obtained from a GPS receiver that outputs standard NMEA sentences [5], so that the pole can be reached directly.

Taken together, earlier work tends to treat protection, local warning and remote reporting as separate problems. The system proposed here combines all three on one Arduino Uno: a relay isolates the faulty section, an LED and buzzer protect people nearby, and a GSM and GPS pair reports the exact location. The aim is a single compact unit that is simple to

build, simple to maintain and cheap enough to be installed on many poles.

III. SYSTEM ARCHITECTURE

The system is organised around the Arduino Uno, which reads the fault-sensing input and controls four outputs: the relay, the LED, the buzzer and the two communication modules. Each part has a single, clear job, so the unit is easy to build, test and repair. Fig. 1 shows how the blocks connect.

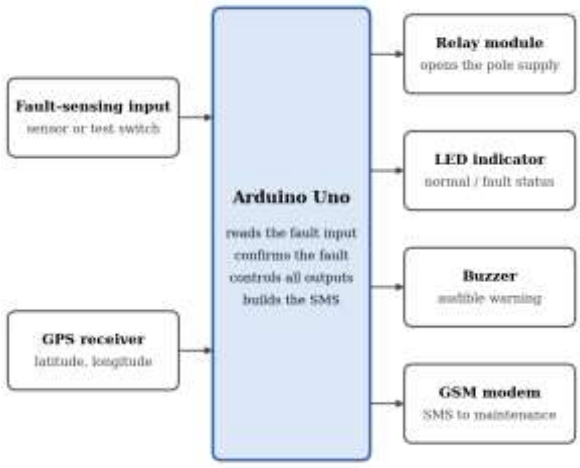


Fig. 1. Block diagram of the proposed pole fault detection and protection system.

The role of each hardware element is as follows.

- **Arduino Uno.** An ATmega328P-based board running at 5 V and 16 MHz, with 14 digital and 6 analogue pins and a hardware serial port [3]. It reads the fault input, runs the decision logic and coordinates all other modules.
- **Fault-sensing input.** A sensing stage connected to an analogue or digital pin that tells the controller when the pole line is abnormal. A current sensor, a voltage divider on the line, or a fault-simulation switch can be used, depending on the prototype.
- **Relay module.** A 5 V relay driven from a digital pin, usually through a transistor or an opto-isolated board. Its contacts sit in series with the supply to the pole section, so energising the relay coil opens the line and de-energises the section.
- **GPS receiver.** A module such as the u-blox NEO-6M that outputs NMEA position sentences over a serial link [5], [7]. It gives the latitude and longitude of the pole.
- **GSM modem.** A module such as the SIM800L or SIM900 that accepts AT commands over a serial link and sends SMS messages through a SIM card [4].
- **LED indicator.** Shows the state of the unit locally: steady for normal operation and blinking or a different colour for a fault.
- **Buzzer.** Gives an audible alarm during a fault so that people near the pole move away.

Because the Uno has only one hardware serial port, the GPS and GSM modules share the available serial resources: the GSM modem is placed on a software serial port and the GPS on a second one, or one of the two uses the hardware port while the other uses software serial. A typical pin allocation is shown in Table I and can be changed to suit the build.

TABLE I. Typical pin allocation of the prototype

Module	Arduino pin	Direction
Fault-sensing input	A0 or D2	Input
GPS TX / RX	D4 / D3	Software serial
GSM TX / RX	D7 / D8	Software serial
Relay IN	D9	Output
LED	D10 (220 Ω resistor)	Output
Buzzer	D11	Output

The GSM modem draws short current peaks of up to about 2 A when it transmits, so it should be powered from a separate regulated supply with a common ground, and not from the 5 V pin of the Arduino [4].

IV. PROPOSED METHODOLOGY

The system follows a four-stage sequence: detect, protect, alert and report. The stages run in a loop, so the unit keeps watching the line after a fault has been handled.

Detection. In normal operation the controller samples the fault-sensing input at a fixed interval. A reading outside the permitted band counts as a fault only if it persists for a few consecutive samples. This simple debounce prevents a short surge or electrical noise from tripping the line without cause.

Protection. Once a fault is confirmed, the controller energises the relay at once. The relay contacts open the supply to the affected section, so the conductor and pole are no longer live. Isolation is the first action, before any message is sent, because removing the hazard is more urgent than reporting it.

Local alert. Directly after isolation the controller switches the LED from its normal state to a fault pattern and turns on the buzzer. This warns anyone close to the pole while the maintenance crew is still on its way.

Remote report. The controller then reads the latest valid position from the GPS module by parsing the NMEA sentences [5]. It sends an AT command sequence to the GSM modem to compose an SMS [4]. The message contains the pole identifier, the fault type, the time of the event and a map link built from the latitude and longitude. If no valid GPS fix is available, the message uses the last stored position and says so, and if the first send fails the modem retries a set number of times.

Reset. When the fault input returns to the normal band and stays there for a set period, the controller can release the relay, switch off the buzzer and restore the LED to its normal state. In a utility setting this step would normally need a manual or authorised reset, to ensure the crew has cleared the cause first.

The complete procedure is summarised in the following algorithm.

1. Initialise the serial ports, the GPS and GSM modules, and the output pins. Keep the relay off, so the line is energised, and set the LED to normal.
2. Read the fault-sensing input.
3. If the reading is within the permitted band, return to step 2.
4. If the reading is abnormal, repeat it for the debounce count. If it becomes normal again, return to step 2.
5. Energise the relay to isolate the pole section.
6. Switch on the buzzer and set the LED to the fault pattern.
7. Read the GPS data and extract the latitude and longitude.
8. Compose and send the fault SMS through the GSM modem, with retries on failure.

9. Wait for the fault condition to clear and for an authorised reset, then restore normal operation and return to step 2.

The SMS sent to the maintenance office has this form:

```
ALERT: Pole P-014 FAULT detected. Supply isolated.
Location: https://maps.google.com/?q=<lat>,<lon>
Time: <hh:mm:ss>
```

V. IMPLEMENTATION

The prototype was built in two steps, hardware first and firmware second, with each module tested alone before the full system was assembled. This order made it easy to find wiring and configuration faults early.

Hardware assembly. The Arduino Uno, GPS receiver, GSM modem, relay module, LED and buzzer were wired as listed in Table I. The LED was placed in series with a 220 Ω resistor. The relay module was connected so that its contacts break the supply to the load that represents the pole section. The GSM modem received its own supply with a common ground, and its serial lines were connected to the Arduino through a level-compatible connection, since many GSM modules use 3.3 V logic [4]. For bench testing, a lamp or a low-voltage load stands in for the pole line, so no mains voltage is handled while the logic is being checked.

Firmware. The program was written in the Arduino IDE in C++ [1]. It uses the SoftwareSerial library for the two serial links and a GPS parsing library such as TinyGPS++ to decode the NMEA data [5], [8]. The code is divided into small functions: one reads and debounces the fault input, one switches the relay, LED and buzzer, one returns the current position, and one sends an SMS using standard AT commands. A main loop calls these functions in the order set out in the algorithm in Section IV.

Module testing. The GSM modem was first tested by sending a plain SMS to a mobile phone. The GPS module was then tested outdoors, where it needs a clear view of the sky, until it returned a valid fix. The relay, LED and buzzer were tested with a short test sketch. Only after each module worked on its own were they combined into the full program.

Safety during testing. The fault condition was created in a controlled way, using a switch or a low-voltage simulation, and never by shorting a live line. Any work with real mains conductors should be done only by qualified personnel, with proper isolation and protective equipment.

VI. RESULTS AND DISCUSSION

The prototype was checked against the operating conditions in Table II. For every condition, the table lists the response the design is meant to produce, and the bench tests verify that the unit follows this response.

TABLE II. Expected system response under different line conditions

Condition	Relay	LED	Buzzer	SMS with GPS location
Normal line	Off (line live)	Normal	Off	Not sent
Short disturbance (below debounce count)	Off	Normal	Off	Not sent
Confirmed fault	On (section isolated)	Fault pattern	On	Sent
Fault, no GPS fix	On	Fault pattern	On	Sent with last stored position, marked as such
Fault cleared, authorised reset	Released	Normal	Off	Optional “restored” message

Three observations follow from these tests. First, isolation, local warning and remote report all start from a single detection event, so the three protective actions cannot drift out of step. Second, the debounce count filters out brief disturbances and avoids needless trips, which matters because every unnecessary trip cuts supply to consumers. Third, the SMS carries a map link, so the maintenance crew can go straight to the pole instead of searching along the feeder.

The main limits of the prototype are also worth stating. SMS delivery depends on mobile network coverage at the pole, and a GPS fix can take from several seconds to a few minutes after a cold start, especially indoors or under tree cover. The detection quality depends on the chosen sensing stage and on how its thresholds are set for the feeder in question. A field installation would also need weather-proof housing, a battery backup so the unit still reports when the line goes dead, and approval from the utility.

VII. CONCLUSION AND FUTURE WORK

This paper presented a Hazardous Electric Pole Fault Detection and Protection System built on an Arduino Uno with a relay, GSM modem, GPS receiver, LED and buzzer. On detecting a fault, the system isolates the affected section, warns people near the pole and sends the pole’s location to the maintenance office by SMS. Combining these three actions in one low-cost unit shortens the time that a fallen or leaking conductor stays live and removes the need to search for the faulty pole. Since all parts are cheap and easy to obtain, the design is suitable for rural and semi-urban feeders where more advanced monitoring is not affordable.

Several improvements are possible. A proper current and voltage sensing stage with adjustable thresholds would make detection more precise. A tilt or vibration sensor could report a pole that has fallen. A solar panel with a battery would keep the unit working through an outage. A LoRa or NB-IoT link could replace SMS in areas with weak GSM coverage, and a cloud dashboard could show all poles on a map. Finally, a field trial on a real feeder, under utility supervision, would give measured data on reliability and response time.

REFERENCES

1. M. Banzi and M. Shiloh, Getting Started with Arduino, 3rd ed. Sebastopol, CA, USA: Maker Media, 2014.
2. S. Monk, Programming Arduino: Getting Started with Sketches, 2nd ed. New York, NY, USA: McGraw-Hill Education, 2016.
3. Microchip Technology Inc., ATmega328P Data Sheet. Chandler, AZ, USA: Microchip Technology.
4. SIMCom Wireless Solutions, SIM800 Series AT Command Manual. Shanghai, China: SIMCom Wireless Solutions.
5. National Marine Electronics Association, NMEA 0183 Interface Standard. Severna Park, MD, USA: NMEA.
6. IEEE, IEEE Guide for Electric Power Distribution Reliability Indices, IEEE Std. 1366-2012. New York, NY, USA: IEEE, 2012.
7. u-blox AG, NEO-6 Series GPS Modules Data Sheet. Thalwil, Switzerland: u-blox AG.
8. M. Hart, “TinyGPS++: A New Full-Featured NMEA Parsing Library for Arduino,” open-source software library.
9. C. Ofori, J. C. Attachie, and F. Obeng-Adjapong, “A GSM-Based Fault Detection on Overhead Distribution Lines,” Jurnal Nasional

Teknik Elektro, vol. 12, no. 2, Jul. 2023, doi: 10.25077/jnte.v12n2.986.2023.

10. P. Nikhare, S. K. Singh, V. P. Khurana, S. Kaiwart, and S. M. D. Ankit Kumar Sethi, "IOT Based Fault Detection in Overhead Transmission Line," International Journal of Creative Research Thoughts, vol. 11, no. 5, pp. h956–h960, May 2023.
11. "Detection and Location of High Impedance Faults in Multiconductor Overhead Distribution Lines Using Power Line Communication Devices," IEEE Transactions on Smart Grid, vol. 6, no. 2, pp. 894–902, Mar. 2015, doi: 10.1109/TSG.2014.2365855.
12. S. Sonwalkar, S. Gopashetty, S. Kagwade, and C. Umarani, "Arduino Based Underground Cable Fault Detector Using GSM," Journal of Advancement in Software Engineering and Testing, vol. 3, no. 2, 2020.
13. "Underground Cable Fault Detection and Monitoring System," in Proceedings of the 2025 6th International Conference for Emerging Technology (INCET), May 2025, doi: 10.1109/INCET64471.2025.11140826.
14. O. S. Alegu, C. H. Ubaka, C. U. Ogbuka, and P. A. Nnoli, "Design and Construction of a Prototype Arduino-Based Fault Detection System for the Agu-Awka 132 KV Transmission Line Using GSM Feedback," Iconic Research and Engineering Journals, vol. 9, no. 5, pp. 2141–2149, Nov. 2025, doi: 10.64388/IREV9I5-1712302.