

FenceGuard: An IoT and TinyML-Based Smart Unauthorized Electric Fence Monitoring System with GPS-Assisted Fault Localization

Prof. Malar K¹, Rithika P², Rakshith J² & Hemanth Kumar S²

*1Assistant Professor, Department of Information Science and Engineering
Sri Sairam College of Engineering, Bengaluru*

2Final-year B.E(Information Science and Engineering), Sri Sairam College of Engineering, Bengaluru

DOI: doi.org/10.34293/iejcsa.v4i4.165

Abstract - Unauthorized intrusion, wire damage, and electrical faults in perimeter electric fences remain challenging problems in agricultural and rural environments, particularly when fence boundaries extend over large areas and cannot be inspected continuously. Conventional monitoring approaches primarily depend on manual inspection or predefined electrical thresholds, which provide limited discrimination between different abnormal operating conditions under changing environmental and load conditions. This paper proposes FenceGuard, an Internet of Things (IoT)-enabled smart electric-fence monitoring architecture that combines voltage and current sensing, edge machine learning, GPS-assisted localization, cloud monitoring, cellular alerting, and solar-powered operation. The proposed system uses an ESP32 microcontroller as the field processing node. Electrical measurements obtained from voltage and current sensing circuits are processed locally and supplied to a lightweight machine-learning model deployed using TensorFlow Lite Micro. The model is designed to classify five operating conditions: normal operation, voltage drop, wire cut, short circuit, and tampering. When an abnormal condition is identified, the system activates a local buzzer, obtains geographical information through a NEO-6M GPS module, and transmits the event through Wi-Fi to Firebase for visualization through a Flutter dashboard. GSM communication is additionally proposed as a backup alert mechanism when conventional network connectivity is unavailable. A solar-battery power subsystem is incorporated to support deployment in remote agricultural locations. Unlike cloud-dependent monitoring architectures, the proposed design performs primary fault classification at the edge, reducing dependence on continuous network connectivity. The paper presents the motivation, research gap, system architecture, hardware design, TinyML methodology, communication framework, and proposed evaluation methodology. Experimental accuracy and performance measurements are intentionally reserved for subsequent prototype-validation work.

Index Terms: Internet of Things, TinyML, edge machine learning, ESP32, electric fence monitoring, fault classification, GPS localization, Firebase, smart agriculture, solar-powered IoT.

INTRODUCTION

Electric fences are widely used for agricultural boundaries, livestock protection, restricted areas, and rural perimeter security. Although such systems act as a physical deterrent, the effectiveness of an electric fence depends on the continuity and electrical condition of its conductors. Wire cuts, voltage degradation, short circuits, and unauthorized interference can reduce fence effectiveness without being immediately visible to the

operator. In large agricultural installations, manually inspecting several kilometers of fencing is labor-intensive and may result in considerable detection delays.

Existing monitoring approaches have introduced sensors and IoT connectivity to provide remote visibility into fence conditions. However, many monitoring systems depend on predefined threshold values. A threshold-based system may identify a condition when a measured parameter crosses a predetermined limit, but electrical characteristics can vary with operating conditions, leakage, environmental factors, and changes in the monitored fence. Consequently, distinguishing between different fault conditions using only fixed thresholds can be difficult.

Machine learning provides an alternative by learning relationships between electrical measurements and known operating conditions. However, implementing machine-learning inference on a remote server introduces dependence on communication infrastructure. In rural agricultural environments, network connectivity may be intermittent, making cloud-dependent fault classification less suitable for immediate local detection. Edge machine learning, particularly TinyML, enables lightweight models to execute directly on resource-constrained microcontrollers. In the proposed FenceGuard architecture, the ESP32 performs sensor acquisition, preprocessing, and local classification, and voltage and current measurements are treated as the primary electrical signatures from which the operating condition of the fence is classified.

The proposed system further addresses the problem of locating detected abnormalities. A GPS module is associated with the sensing node so that a classified event can be linked to geographical coordinates. The system also incorporates local and remote notification mechanisms: a buzzer provides immediate local indication, while Wi-Fi and Firebase support remote monitoring through a Flutter application, and GSM is considered as a secondary communication mechanism for cellular alerts. Because the system is designed for remote deployment, it incorporates a solar-battery power subsystem. The resulting architecture combines electrical sensing, edge intelligence, localization, communication, visualization, and renewable power within a single proposed monitoring framework.

The primary contributions of this work are as follows. First, the paper proposes an IoT architecture for continuous electric-fence condition monitoring using voltage and current measurements. Second, it presents a TinyML-based edge classification approach designed for ESP32 deployment. Third, it integrates GPS information with classified fence events for zone-level localization. Fourth, it adopts a hybrid communication architecture in which Wi-Fi and Firebase support dashboard monitoring and GSM serves as a backup alert mechanism. Fifth, it describes a solar-battery power architecture intended for remote agricultural deployment. Finally, it defines a proposed evaluation methodology for future prototype validation.

RELATED WORK AND RESEARCH GAP

Existing research relevant to FenceGuard can be grouped into electric-fence monitoring, IoT-based fault detection, edge machine learning for agriculture, GPS/GSM-assisted asset localization, cloud-connected agricultural systems, and solar-powered sensing. Each group is examined below, with emphasis on what the work contributes relative to the others and what it leaves unaddressed.

A. Electric Fence and Perimeter Monitoring

Conventional fence monitoring approaches generally depend on manual inspection or electrical threshold conditions. Manual inspection becomes inefficient as the monitored boundary increases in length, and threshold-based systems, although capable of rapid detection of significant electrical deviations, require predetermined limits and may not provide sufficient discrimination among multiple fault conditions. Sufian et al. [1] present a LoRaWAN-based electric-fence fault sensing system for elephant-conflict mitigation that combines high-voltage sensing, short-circuit detection, and gyroscope-based tilt sensing, reporting high field-validated detection rates on a 50 m test segment; however, the system relies on fixed thresholds rather than learned classification and does not incorporate machine-learning-based discrimination between fault types. Related electric-fence alerting work for human-wildlife conflict, such as the eleAlert system [9], similarly focuses on binary intrusion alerting rather than multi-class electrical fault diagnosis. A Blynk-based IoT electric-fence notification system

[10] uses an ACS712 current sensor and battery-level monitoring to notify farmers of fence status via a mobile app, but again relies on threshold logic rather than a trained classifier and does not address GPS localization or edge inference.

B. IoT-Based Fault Detection and Cloud-Connected Monitoring

IoT-based monitoring improves accessibility by transferring sensor information to remote users. Such systems can provide dashboards and historical records, but architectures that perform the primary classification remotely remain dependent on network availability. This limitation is particularly important for agricultural deployments where cellular or Wi-Fi connectivity may not be continuously reliable. Kiarie [2] proposes a low-cost time-domain reflectometer approach for locating faults along a fence conductor, offering finer-grained physical fault localization than node-level GPS but requiring specialized reflectometry hardware not typically available in low-cost IoT deployments.

C. TinyML and Edge Machine Learning in Agriculture

TinyML provides a means of executing lightweight machine-learning models directly on microcontrollers. The use of TensorFlow Lite Micro allows a trained model to be converted and deployed on resource-constrained hardware, as formalized by David et al. [3] and analyzed on the ESP32's Xtensa LX6 core by Zim [13]. Recent agricultural applications demonstrate the feasibility of this approach: a four-layer edge-first TinyML framework for

precision irrigation achieves high predictive accuracy for irrigation decisions using an ESP32 inference node without cloud dependency

[11]; ESP32-based CNN classification of crop-field imagery has been shown to exceed 87% accuracy under quantized, memory-constrained deployment [12]; and a transfer-learning TinyML framework evaluated on an ESP32 MCU board demonstrates DNN/LSTM inference for soil-moisture forecasting with quantified energy and latency trade-offs [14]. These works confirm that ESP32-class hardware can support multi-class, learned classification under real energy and memory constraints, a foundation FenceGuard builds on by targeting electrical fault signatures rather than agronomic or imaging signals. In FenceGuard, this capability is used to shift the primary classification stage for electric-fence conditions from the cloud to the ESP32.

D. GPS/GSM-Assisted Localization and Backup Communication

GPS technology has also been widely used in IoT and livestock-monitoring applications, typically for continuous asset tracking rather than event-triggered fault localization. GNSS-based livestock and fenceless-farming trackers combine satellite positioning with cellular or LPWAN connectivity to report animal location in near real time, and GSM/GPRS-based trackers are commonly used as a fallback communication path when Wi-Fi or broadband connectivity is unavailable. In the proposed system, however, GPS is not treated merely as a continuous telemetry feature. The geographical information is associated with the classified event so that the operator can identify the monitoring node or affected zone, reducing power consumption relative to continuous tracking while still providing actionable location context. Related work on GPS-based agricultural tractor monitoring [6], [7] and localization technology reviews for precision farming [8] further motivate node-level rather than continuous-tracking localization for stationary infrastructure.

E. Solar-Powered IoT Sensing

Solar-battery power subsystems are a common design choice for remote agricultural IoT nodes, since replacing batteries across large or inaccessible boundaries is impractical. Solar-powered agricultural sensor nodes with energy-harvesting capability have been shown to substantially extend deployment lifetime [15], and PV-powered remote monitoring designs report data-acquisition failure rates reduced to well under 1% through dynamic power and communication management [16]. Analyses of solar energy harvesting for wireless sensor networks further show that duty-cycled, harvested power can theoretically sustain indefinite network lifetime under favorable irradiance [17]. FenceGuard adopts the same solar-battery principle, applying it specifically to a fence-monitoring node that must remain operational for edge inference, GPS acquisition, and GSM transmission, each with different peak current demands.

F. Comparative Summary and Research Gap

Table I summarizes representative related systems against FenceGuard along the dimensions most relevant to this work.

Table 1 Comparison of FenceGuard with Representative Related Systems

System	Sensing	Classification	Localization	Connectivity	Power
eleAlert [9]	Contact/current on fence	Threshold	None	GSM/SMS	Not stated
Blynk Fence [10]	ACS712 + battery	Threshold	None	Wi-Fi	Battery only
LoRaWAN [1]	10 kV, short-circuit, gyro	Threshold, validated	Gateway	LoRaWAN	Not stated
TinyML Irrigation [11]	Moisture, temp, pH, light	Gradient boosting	N/A	ESP32, no cloud	Not stated
TinyML-TL [14]	Soil/environment sensors	DNN/LSTM, TL	N/A	Wireless node	Not stated
Solar WSN [15], [17]	Moisture, temp, humidity	N/A	N/A	Wi-Fi/RF	Solar + harvesting
FenceGuard	Voltage + current	5-class TinyML	Per-node GPS	Wi-Fi/Firebase + GSM	Solar + Li-ion

As Table I indicates, existing electric-fence monitoring systems [1], [9], [10] rely on threshold-based detection and rarely combine learned multi-class fault classification with event-level GPS localization. Conversely, TinyML systems demonstrated for agriculture [11], [12], [14] validate ESP32-based edge inference but are applied to irrigation and imaging tasks rather than electrical fence fault signatures, and do not incorporate GPS or dual-path (Wi-Fi/GSM) alerting. Solar-powered IoT works [15]-[17] address power autonomy but are not integrated with edge classification or localization in a single architecture.

Based on these areas, the research gap addressed by this work is the lack of a single proposed architecture specifically combining electrical fence sensing, TinyML-based local classification, GPS-assisted event localization, IoT monitoring, cellular backup alerts, and solar-powered operation. The proposed contribution is therefore not the individual invention of these technologies, but their integration into a fence-specific edge-intelligence architecture designed for remote agricultural monitoring.

PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement

Large agricultural fence boundaries are difficult to inspect continuously using manual methods. When a wire is cut, electrical performance changes, or unauthorized interference occurs, the operator may not immediately know that an abnormal condition has occurred or which monitoring zone is affected. A monitoring system should therefore provide continuous acquisition of electrical fence parameters, automatic classification of different abnormal conditions, local detection without requiring continuous cloud connectivity,

identification of the monitoring location, immediate local alerting, remote event monitoring, and operation in areas with limited electrical infrastructure.

B. System Objectives

The proposed FenceGuard system is designed to achieve the following objectives. It measures fence voltage and current continuously and uses a lightweight machine-learning model, executed locally on an ESP32, to classify fence conditions, identifying normal operation, voltage drop, wire cut, short circuit, and tampering. Detected abnormal events are associated with GPS coordinates, and local alerts are provided using a buzzer. Events are synchronized with Firebase through Wi-Fi, and remote visualization is provided through a Flutter dashboard, with GSM-based backup notification available when conventional connectivity fails. Finally, the system supports remote operation using solar-battery power.

PROPOSED FENCEGUARD ARCHITECTURE

The proposed architecture is divided into five functional layers, shown in Fig. 1: the sensing layer, the edge intelligence layer, the localization and alert layer, the IoT/cloud layer, and the power layer. In the overall operational sequence, the electrical condition of the fence is captured by voltage and current sensing, passed to the ESP32 for preprocessing and TinyML classification, followed by GPS acquisition and alerting, transmission over Wi-Fi or GSM, and finally delivery to Firebase and the Flutter dashboard.

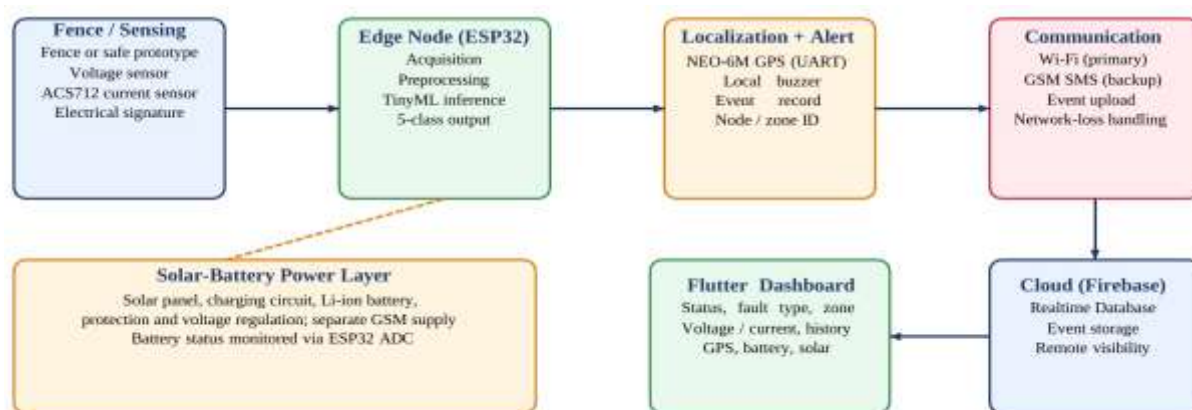


Figure 1 System Architecture of FenceGuard

A. Sensing Layer

The sensing layer consists primarily of a voltage sensor and an ACS712 current sensor. The voltage sensor measures the electrical voltage associated with the monitored fence or safe prototype equivalent, while the ACS712 provides a measurement proportional to current. The two measurements form the input electrical signature for the classification model, and the use of both parameters is intended to provide more information than relying on a single electrical measurement. For prototype development, a safe low-voltage fence simulation or controlled test load is required. The high-voltage output of an actual

electric fence must not be connected directly to the ESP32 or a breadboard, and appropriate isolation, scaling, protection, and signal conditioning are required for any real deployment.

B. Edge Intelligence Layer

The ESP32 serves as the central processing node. It receives sensor measurements, performs preprocessing, executes the TinyML model, and controls subsequent event processing. The model is trained offline using labeled voltage-current measurements, converted to TensorFlow Lite, and subsequently prepared for TensorFlow Lite Micro deployment on the ESP32. The proposed five output classes are normal operation, voltage drop, wire cut, short circuit, and tampering; in effect, the voltage and current inputs pass through preprocessing and the TinyML model to yield a fence condition.

C. Event Processing Layer

If the model identifies normal operation, monitoring continues. If an abnormal condition is identified, the ESP32 initiates an event-processing sequence in which the buzzer is activated, GPS coordinates are acquired, an event record is generated, and the record is passed to the communication stage. The event record can contain the event type, voltage, current, latitude, longitude, timestamp, monitoring-node or zone identifier, and device status, as listed in Table II. This approach separates immediate classification from remote monitoring.

Table 2 Event Record Parameters

Parameter	Description	Parameter	Description
Device ID	Monitoring-node identifier	Latitude	GPS latitude
Zone	Fence monitoring zone	Longitude	GPS longitude
Condition	Classified fence state	Timestamp	Event time
Voltage	Measured voltage	Status	Current event state
Current	Measured current		

D. Localization Layer

A NEO-6M GPS module is connected to the ESP32 using a UART interface. When an abnormal event is detected, the controller obtains the current latitude and longitude and associates these coordinates with the event. The proposed localization should be interpreted as monitoring-node or zone localization, rather than claiming that a single GPS module can identify the exact physical point of a fault along an entire fence. In a multi-node deployment, each node can represent a particular fence zone, allowing the event coordinates and node identity to identify the affected region.

E. Alert Layer

FenceGuard uses two forms of alerting. For the local alert, a buzzer is activated when the classified condition is abnormal, allowing personnel near the monitoring node to receive an immediate indication without depending on network connectivity. For the remote alert, Wi-Fi communication is used as the primary communication path to Firebase, while GSM is proposed as a backup cellular communication mechanism for sending SMS alerts. Thus, the primary path runs from the ESP32 over Wi-Fi to Firebase and then to the Flutter dashboard, and, when required, the backup path runs from the ESP32 through GSM as an SMS to the user.

HARDWARE DESIGN

The proposed hardware consists of an ESP32 DevKit, voltage sensor, ACS712 current sensor, NEO-6M GPS, buzzer, GSM module, solar panel, rechargeable Li-ion battery, charging circuitry, voltage regulation, and supporting electronics.

A. ESP32 Microcontroller

The ESP32 is selected as the central field controller because it can combine sensor acquisition, local processing, wireless communication, and peripheral control within a single embedded platform. Its responsibilities include sensor data acquisition, preprocessing, TinyML inference, GPS communication, buzzer control, Wi-Fi communication, GSM communication, Firebase synchronization, and temporary local event handling during communication interruptions.

B. Voltage and Current Sensing

The voltage sensor provides the voltage feature used by the classification model, and the ACS712 provides current measurements. Together, these measurements describe the electrical behavior of the monitored system. A conceptual fault-detection sequence therefore begins with an electrical condition that changes voltage and current, which the ESP32 acquires and the TinyML model classifies.

C. GPS Module and Buzzer

The NEO-6M provides latitude and longitude information and communicates with the ESP32 through UART. The GPS subsystem is activated or read when an abnormal event requires location information. The buzzer provides local indication of an abnormal condition: when the prediction is normal, the buzzer remains off, and when the prediction is anything other than normal, the buzzer is switched on.

D. GSM Communication and Hardware Data Flow

A SIM800L/SIM900-class GSM module can provide cellular backup communication, transmitting an SMS containing the detected condition, zone, and location information. GSM is treated as a separate power subsystem because cellular modules can experience relatively high current demand during transmission and should not be powered directly

from an ESP32 GPIO. The complete field-node data flow begins with the fence, whose electrical condition is captured by the sensors and passed to the ESP32 for TinyML-based fault classification. For abnormal conditions, the flow continues through the buzzer and GPS to Wi-Fi or GSM, then to Firebase or SMS, and finally to the Flutter dashboard or the user, with Table II listing the parameters carried by each event record.

TINYML-BASED FAULT CLASSIFICATION

The central intelligence of FenceGuard is the TinyML classification stage, which is described below in terms of dataset preparation, preprocessing, model training, and embedded inference.

A. Dataset Preparation

The proposed model requires a labeled dataset containing electrical measurements acquired under known operating conditions. Each sample contains voltage and current values associated with one of the five predefined classes: normal operation, voltage drop, wire cut, short circuit, and tampering. Multiple measurements should be collected under each condition to represent variations within a class rather than a single fixed electrical signature.

B. Preprocessing

Before classification, voltage and current measurements undergo preprocessing, which can include scaling, normalization, filtering, and conversion to the input representation required by the selected model. The same preprocessing procedure used during training must be applied during ESP32 inference to maintain consistency.

C. Model Training

Model training is performed offline using TensorFlow. The proposed development pipeline starts from labeled sensor data, proceeds through TensorFlow training, model conversion, and quantization, and ends with deployment as a TensorFlow Lite Micro model on the ESP32.

D. Embedded Inference

During operation, the ESP32 periodically acquires voltage and current measurements, preprocesses them, and passes them to the embedded model, which outputs one of the five predefined classes. Even when remote communication is temporarily unavailable, the ESP32 can continue local classification and activate the local alert mechanism.

IOT, CLOUD, AND APPLICATION INTEGRATION

A. Firebase Backend

Firebase Realtime Database is proposed as the cloud backend for storing and synchronizing detected events. The Firebase layer is not responsible for primary fault

classification; instead, it acts as the remote data and visualization layer, maintaining local intelligence with remote visibility rather than remote intelligence dependent on continuous connectivity.

B. Flutter Dashboard

The Flutter application provides the user-facing monitoring interface, displaying current fence status, detected fault type, monitoring zone, voltage and current, event history, GPS location, battery status, and solar charging status. The dashboard architecture separates field-level sensing and classification from user-level visualization.

SOLAR-POWERED OPERATION

Agricultural fence monitoring nodes may need to operate in locations where conventional electrical power is unavailable. FenceGuard therefore incorporates a solar-battery power subsystem in which the solar panel feeds a solar-compatible charging circuit, which charges a Li-ion battery, which in turn supplies the electronics through protection and voltage regulation. The battery provides stored energy during periods of low solar generation or nighttime operation. The solar panel should not be connected directly to the ESP32; a suitable charging circuit and regulated power path are required. Battery monitoring can be incorporated using a voltage-sensing circuit connected to an ESP32 ADC, with the resulting battery status transmitted to Firebase and displayed on the dashboard. The complete power subsystem therefore supports three functions: energy generation through solar power, energy storage through the rechargeable battery, and regulated power delivery to the embedded electronics.

METHODOLOGY AND WORKFLOW

A. Complete System Operation

The operation of FenceGuard proceeds through seven stages, shown in Fig. 2. In the first stage, electrical sensing, the voltage and current sensors acquire measurements from the monitored fence or safe prototype equivalent. In the second stage, edge processing, the ESP32 preprocesses the measurements and supplies them to the TinyML classifier. In the third stage, fault classification, the model determines whether the condition is normal, a voltage drop, a wire cut, a short circuit, or tampering. In the fourth stage, local response, the ESP32 activates the buzzer if the condition is abnormal. In the fifth stage, localization, the ESP32 obtains GPS coordinates and associates them with the event. In the sixth stage, communication, the event is transmitted via Wi-Fi to Firebase, while GSM can provide a backup SMS notification. In the seventh stage, remote visualization, Firebase synchronizes the event with the Flutter dashboard.

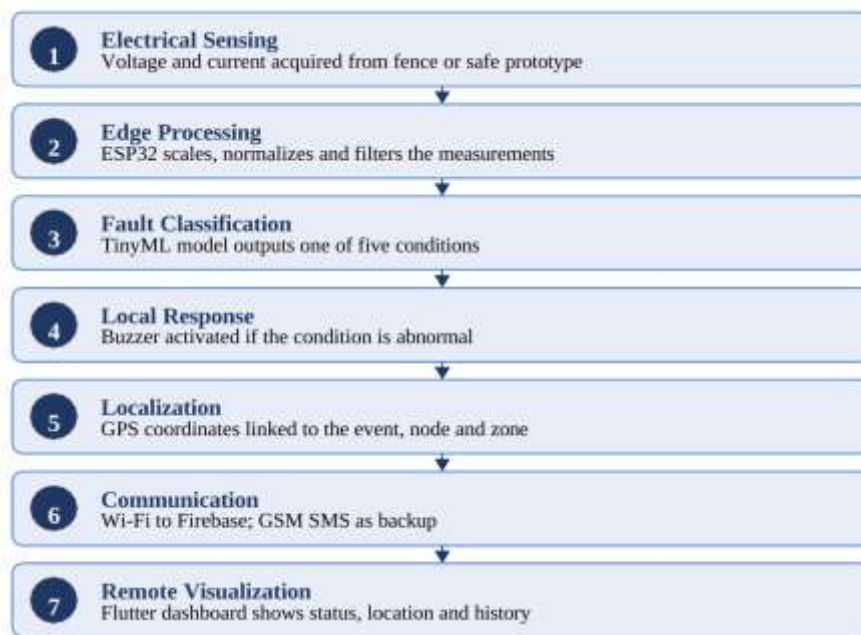


Figure 2 Workflow of the FenceGuard System

PROPOSED EVALUATION METHODOLOGY

Since this paper focuses on the proposed architecture rather than experimental validation, the evaluation methodology summarized in Table III is defined for future prototype testing. For classification evaluation, the five operating conditions will be generated in a safe prototype environment and evaluated using accuracy, precision, recall, F1-score, and a confusion matrix, with particular attention to classes producing similar electrical signatures. Embedded performance will be assessed on the ESP32 using inference latency, model size, memory utilization, sensor sampling time, and overall response time. The GPS subsystem will be evaluated by comparing reported coordinates with known reference positions to determine localization error. The communication subsystem will be evaluated by measuring event transmission time, Firebase synchronization time, dashboard update time, GSM SMS delivery behavior, and system behavior during temporary network loss. Finally, future power testing should measure ESP32, sensor, GPS, and GSM power consumption, battery capacity, solar charging behavior, and operating duration.

Table 3 Proposed Evaluation Plan for Future Prototype Validation

Subsystem	Proposed Metrics
Classification	Accuracy, precision, recall, F1-score, confusion matrix
Embedded performance	Inference latency, model size, memory utilization, sampling time, response time
Localization	Localization error against known reference positions
Communication	Transmission time, Firebase sync time, dashboard update time, SMS delivery, behavior during network loss
Power	Component power consumption, battery capacity, solar charging behavior, operating duration

Embedded performance Inference latency, model size, memory utilization, sampling time, response time

Communication Transmission time, Firebase sync time, dashboard update time, SMS delivery, behavior during network loss

EXPECTED ADVANTAGES AND LIMITATIONS

A. Expected Advantages

The proposed FenceGuard architecture offers several expected benefits. Continuous electrical sensing can reduce dependence on manual fence inspection, while TinyML-based edge inference enables local fault classification without continuous cloud connectivity. GPS provides the location of the monitoring node or affected zone, while the buzzer, Wi-Fi/Firebase, and GSM provide multiple alert mechanisms. The Flutter dashboard enables remote monitoring, and the solar-battery subsystem supports deployment without reliable grid power.

B. Limitations

The proposed classifier has not yet been experimentally validated; therefore, accuracy and other performance metrics cannot be claimed at this stage. GPS identifies the monitoring node or zone rather than the exact fault point along a continuous fence. Real deployment also requires appropriate electrical isolation and protection for the sensing circuitry, and solar-panel sizing, battery endurance, GSM power requirements, and long-term outdoor reliability require further validation.

FUTURE WORK

Future work will focus on prototype implementation and experimental validation. A diverse voltage-current dataset will be collected for the five operating classes, followed by comparison of lightweight machine-learning models for ESP32 deployment using TensorFlow Lite Micro. The complete architecture will then be integrated and evaluated using classification accuracy, precision, recall, F1-score, inference latency, memory usage, GPS error, communication latency, and power consumption. The TinyML approach will also be compared with fixed-threshold detection, while multi-node deployment will be investigated for monitoring larger fence boundaries.

CONCLUSION

This paper presented FenceGuard, a proposed IoT and TinyML-based architecture for electric-fence monitoring. The system combines voltage and current sensing with local machine-learning inference on an ESP32 to classify normal operation, voltage drop, wire cut, short circuit, and tampering. The architecture integrates GPS for event-associated location information, a buzzer for local alerts, Wi-Fi and Firebase for remote monitoring, GSM for backup notification, and solar-battery power for remote deployment.

The current work focuses on the proposed architecture and methodology; therefore, experimental accuracy and performance results are reserved for subsequent prototype validation. Future implementation will evaluate classification performance, embedded efficiency, localization accuracy, communication latency, and power consumption.

REFERENCES

- [1] M. S. Sufian Bin Mohammad et al., "Development of Electric Fence Fault Sensing and Monitoring System with LoRaWAN IoT," *IIUM Engineering Journal*, vol. 27, no. 1, pp. 48–80, 2026.
- [2] P. Kiarie, "A low-cost Raspberry Pi based time domain reflectometer for fault detection in electric fences," *IET Science, Measurement & Technology*, 2024, doi: 10.1049/smt2.12183.
- [3] R. David et al., "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems," *Proc. Machine Learning and Systems*, vol. 3, 2021.
- [4] A. Maier, A. Sharp, and Y. Vagapov, "Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things," *Proc. ITA*, 2017, pp. 143–148, doi: 10.1109/ITECHA.2017.8101926.
- [5] "TinyML on Microcontrollers: Enabling Energy-Efficient, Real-Time, Privacy-Preserving Incremental Learning for Embedded Systems," *Proc. ICSCN*, 2025, doi: 10.1109/ICSCN67106.2025.11308574.
- [6] T. Phasinam et al., "Real-Time Monitoring and Positioning of Agricultural Tractors Using a Low-Cost GPS and IoT Device," *International Journal of Geoinformatics*, vol. 21, no. 1, pp. 111–120, 2024, doi: 10.52939/ijg.v21i1.3799.
- [7] T. Phasinam et al., "IoT and radio telemetry based wireless engine control and real-time position tracking system for an agricultural tractor," *Discover Internet of Things*, 2023, doi: 10.1007/s43926-023-00035-4.
- [8] Z. Diao et al., "Localization technologies for smart agriculture and precision farming: A review," *Computers and Electronics in Agriculture*, vol. 236, 2025, Art. no. 110464.
- [9] L. Wijesinghe et al., "Electric Fence Intrusion Alert System (eleAlert)," presented at *IEEE Region 10 Conf. (TENCON)*, 2020.
- [10] "Towards IoT-based Notification System for Agriculture Electric Fence," *IEEE Conference Publication*, 2020, doi: 10.1109/IEEECONF.2020.9250958.
- [11] K. Taueatsoala, C. Daniels, and A. J. Ramsunar, "TinyML-Enabled IoT for Sustainable Precision Irrigation," *arXiv:2601.13054*, 2026.
- [12] "TinyML Classification for Agriculture Objects with ESP32," *Sensors and Actuators Reports (MDPI)*, 2025.

- [13] M. Z. H. Zim, "TinyML: Analysis of Xtensa LX6 microprocessor for Neural Network Applications by ESP32 SoC," arXiv:2106.10652, 2021.
- [14] A. Hayajneh et al., "Tiny machine learning on the edge: A framework for transfer learning empowered unmanned aerial vehicle assisted smart farming," IET Smart Cities, 2024, doi: 10.1049/smc2.12072.
- [15] "Solar-Powered Smart Agricultural Monitoring System Using Internet of Things Devices," presented at IEEE conference, 2018.
- [16] "Design of an Improved IoT-Based PV-Powered Soil Remote Monitoring System with Low Data Acquisition Failure Rate," Future Internet (MDPI), vol. 17, no. 12, Art. no. 538, 2025.
- [17] "Maximization of wireless sensor network lifetime using solar energy harvesting for smart agriculture monitoring," Ad Hoc Networks (Elsevier), 2019.
- [18] "A LoRa IoT Framework with Machine Learning for Remote Livestock Monitoring in Smart Agriculture," arXiv:2510.07322, 2025.