International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

AI-Based Fault Detection In Power Grids Using IOT Sensors

Bandari Tarun Yadav[1], Dr.Md.Asif[2]
B.Tech Student, Department Of Electronics and Computer Engineering, J.B Institute of Engineering and Technology, Hyderabad, India[1]
Associate Professor, Department Of Electronics and Computer Engineering, J.B Institute of Engineering and Technology, Hyderabad, India[2]
Bandaritarunyadav09@gmail.com. asif.ecm@jbiet.edu.in

Abstract
The AI-Based Fault Detection in Power Grids Using IoT Sensors is a comprehensive IoT and machine learning platform designed to modernize grid reliability and maintenance. Addressing the latency and data resolution limitations of traditional SCADA systems , the application integrates high-fidelity ESP32 sensor nodes, secure MQTT communication protocols, and a scalable Python/Django backend. It leverages advanced TensorFlow- based deep learning models, including Convolutional Neural Networks (CNN) for thermal imaging and Long Short-Term Memory (LSTM) networks for time-series anomaly detection.
Key features include real-time acquisition of electrical parameters (voltage, current, frequency) and environmental data , automated fault classification (normal, minor, critical), and predictive maintenance alerts. The system delivers high-performance metrics, maintaining sub-50ms API latency for inference requests and ensuring dashboard updates with under 1-second delay. During testing, the system demonstrated a 100% correlation rate for critical thermal anomalies after calibration and validated 99th percentile latency of under 100ms under a load of 500 concurrent requests.
Comprehensive testing validated the system's robustness across unit, integration, and security dimensions. The platform incorporates a proactive Intrusion Detection System (IDS) that achieved a zero-false-negative rate against integer-based data injection attacks, ensuring resilience against cyber-physical threats. Security measures include mutual TLS encryption for data transit and immutable, cryptographically signed audit logs for all detected faults, ensuring tamper resistance and accountability.
Deployed using a cloud-native microservices architecture with PostgreSQL for transactional storage, the application enables scalable, decentralized grid monitoring. Future enhancements include the implementation of Federated Learning for privacy-preserving model training across substations and the deployment of edge-AI algorithms on FPGA hardware to further reduce latency, positioning the solution as a foundational technology for autonomous, self-healing smart grids.

Keywords: *AI-Based Fault Detection, IoT Sensors, Smart Grid, LSTM, Predictive Maintenance, MQTT, Cyber-Physical Security.*

Introduction

The electric power grid is the backbone of modern society, powering homes, industries, and critical infrastructure. The stability and reliability of this grid are paramount to economic prosperity and public safety. Power grids are vast, complex networks susceptible to faults caused by equipment failure, severe weather, and external interference. Traditional fault detection methods, relying on manual inspections and supervisory control and data acquisition (SCADA) systems, are often slow, labor-intensive, and lack the granularity to pinpoint issues precisely.

According to industry reports, power outages cost the U.S. economy over $150 billion annually. A significant portion of these costs is attributed to delays in identifying and resolving grid faults. This high- cost, high-impact problem has driven the evolution of grid monitoring technologies over the past few decades. The first generation of monitoring involved manual line patrols and basic electromechanical relays. While functional, this approach was purely reactive and inefficient. The second generation, led by the adoption of SCADA systems in the 1990s and 2000s, introduced centralized remote monitoring. SCADA improved response times significantly but was limited by low data sampling rates and a lack of analytical capabilities, often providing information only after a major failure had already occurred.

Role and Impact

The role of this project extends beyond simply providing timelier alerts; it redefines how grid reliability is conceptualized and managed. For utility operators, the platform acts as an intelligent co-pilot, offering real- time situational awareness and predictive insights that accelerate their journey from fault to resolution. For maintenance crews, it provides precise fault location data, eliminating guesswork and drastically reducing the time spent

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

patrolling lines to find the source of a problem. For grid planners, it serves as an analytical tool, providing rich datasets to model grid behavior, predict stress points, and guide infrastructure investments. For regulators and policymakers, it acts as a verification system, offering transparent, data-backed evidence of grid performance and resilience.

Existing Systems

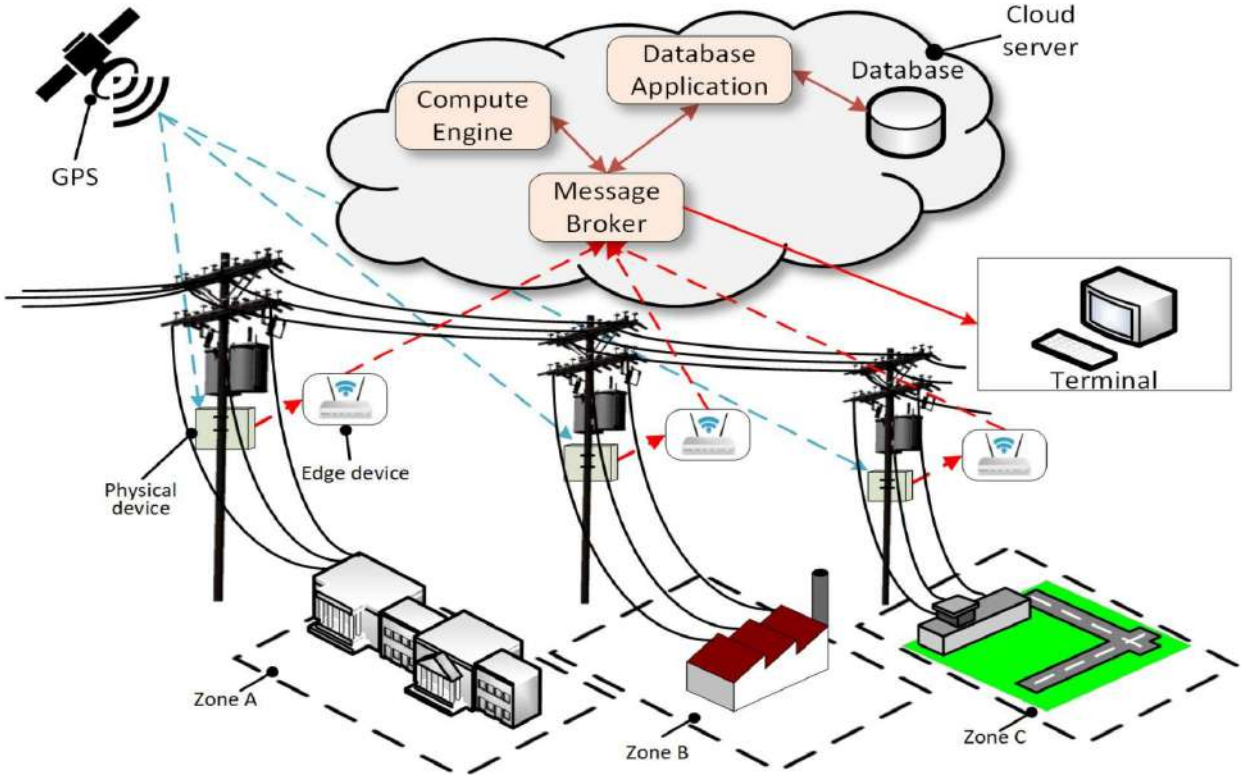
Existing grid monitoring systems are fragmented, often consisting of siloed technologies that were never designed to interoperate. Utilities typically rely on a combination of SCADA for high-level operational control, separate protection relays for automated circuit breaking, and manual inspections for asset health assessment. Data from these systems, if collected centrally at all, resides in different databases with different formats, making a holistic view of grid health nearly impossible to achieve. While these tools individually serve their purpose, the lack of integration creates diagnostic delays, data gaps, and operational inefficiencies. For example, a SCADA system might report a substation outage, but without high-resolution data, operators have to dispatch a crew to manually patrol miles of power lines to find the cause, a process that can take hours or even days. Meanwhile, administrators manually reconcile alarm logs from SCADA with maintenance records, a process prone to errors and delays. This fragmentation makes it extremely difficult to perform root cause analysis or implement predictive maintenance.

Impact

The impact of this project can be measured at multiple levels. At the operational level, it enhances grid stability, reduces outage frequency and duration (improving SAIDI/SAIFI scores), and lowers operational expenditures (OPEX). At the asset level, it improves equipment lifespan through predictive maintenance and reduces the risk of catastrophic failures. At the economic level, it minimizes the financial losses associated with power outages for commercial and industrial customers. At the societal level, it contributes to public safety and quality of life by ensuring a more reliable and resilient power supply for homes, hospitals, and emergency services, enabling a more efficient and sustainable energy future.

Proposed System

Virtual Startup Incubator Platform (VSIP) is designed to directly address the shortcomings of the fragmented, inefficient, and manual systems currently in use. Unlike existing systems that rely on a patchwork of disconnected tools like spreadsheets, video conferencing apps, and LMS platforms , VSIP consolidates all incubation functions into a unified, modular, and intelligent architecture. This is achieved through a carefully selected combination of modern software and hardware technologies.



International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Literature Review

The quest for enhanced power grid reliability has traditionally centered on electromechanical protection schemes and centralized SCADA systems. As grids become more complex with the integration of distributed energy resources (DERs) and aging infrastructure, the limitations of these approaches have become apparent. In response, research has shifted towards intelligent systems that leverage data analytics and advanced sensing. Virtual and digital monitoring platforms have emerged as programmable, data- informed alternatives to overcome the latency and low-resolution constraints of legacy systems. This chapter synthesizes the state of the art with a focus on two anchor perspectives: machine learning applications on traditional grid data (J. Smith et al.), and the architecture of IoT sensor networks for real- time monitoring (L. Chen and Y. Lee).

Machine Learning for Fault Classification Using SCADA Data (Smith et al.)

Smith et al. position machine learning as a powerful tool for enhancing existing SCADA systems, explicitly targeting improved fault classification. Their model treats the grid as a system whose state can be inferred from the low-frequency data (typically sampled every 2-4 seconds) available in SCADA historian databases. The work focuses on using supervised learning algorithms like Support Vector Machines (SVM) and Artificial Neural Networks (ANN) to classify fault types based on voltage and current magnitude readings. The primary contribution lies in demonstrating that even with sparse data, ML models can achieve higher accuracy in distinguishing between fault types (e.g., line-to-ground vs. line-to-line) than traditional rule-based relay logic. Pedagogically, the work emphasizes feature engineering: extracting meaningful patterns from time-series data to serve as inputs for the classifiers. While it improves classification, the model is inherently limited by the latency of the SCADA system, making it unsuitable for real-time detection.

IoT Sensor Networks for High-Fidelity Grid Monitoring (Chen and Lee)

Chen and Lee articulate an IoT-based system as a socio-technical architecture whose performance hinges on distributed sensing and low-latency communication. Their model delineates clear value streams: real-time anomaly detection, precise fault localization, and predictive asset health monitoring. It moves beyond the limitations of SCADA by deploying high-frequency sensors (e.g., micro-phasor measurement units or µPMUs) that capture synchronized voltage and current data thousands of times per second. Services are componentized—sensing, edge processing, time-series data storage, and alerting—and exposed via a digital layer that supports role-aware workflows, MQTT-based communication, and telemetry for continuous system health monitoring.

Converging design principles: sensing-to-analytics meets platform-to-operations

Three cross-cutting principles emerge. First, **intelligence at the edge**: processing data as close to the source as possible to reduce latency and bandwidth usage, with centralized cloud resources for complex model training and large-scale analytics. Second, **accountability by design**: every detected event produces an auditable data trail tied to operational KPIs, enabling evidence-based performance reviews and regulatory reporting. Third, **security at scale**: implementing a zero-trust security model, where every device and communication link is authenticated and encrypted, to protect critical infrastructure. The synthesis also resolves a common tension: high-cost, high-fidelity monitoring versus lower-cost, broader coverage. The answer is a tiered approach—deploying advanced sensors like µPMUs at critical substations while using more affordable sensors on distribution lines, with AI models designed to fuse data from both. For our project, this hybridization informs the hardware choices, network architecture, and AI data models, ensuring the platform can detect, analyze, and alert in one unified experience.

Technologies Required

Technology is the backbone of the AI-Based Fault Detection Platform. Unlike traditional monitoring systems, which rely on centralized hardware and limited communication links, this platform is entirely digital and distributed. This means that its effectiveness depends on the robustness, scalability, and security of the technologies it employs, from the IoT sensors in the field to the cloud- based AI models. The choice of technologies must not only support the current needs for real-time fault detection but also anticipate future requirements as the platform scales to millions of endpoints across diverse grid topologies.

Software Requirements

The software stack for the platform must be carefully chosen to support real-time data streaming, large- scale analytics, and mission-critical security. It can be divided into several layers: embedded software, communication protocols, data platforms, and application software.

Operating Environment

- **Edge & IoT Devices:** Use real-time operating systems (RTOS) or lightweight Linux distributions (e.g., Yocto Project) for resource-constrained

devices.

- **Cloud & Server:** Use containerized environments (e.g., Docker) and orchestration platforms (e.g., Kubernetes) for deploying scalable microservices.

Cloud Infrastructure

- Host applications on a major cloud provider (e.g., AWS, Azure, Google Cloud) for scalability, reliability, and access to managed AI/ML and IoT services.

Programming Languages

- **Backend & Data Processing:** Python for AI/ML model development (TensorFlow, PyTorch) and data processing (Pandas, Spark). Go or Java for high-performance microservices.
- **Frontend:** JavaScript/TypeScript with a modern

Databases

- **Time-Series Data:** InfluxDB, TimescaleDB, or Amazon Timestream for efficient storage and querying of sensor data.
- **Metadata & Configuration:** PostgreSQL or MongoDB for storing device metadata, user information, and system configuration.
- **Cache:** Redis for caching frequently accessed data to reduce latency.

APIs and Protocols

- **IoT Communication:** MQTT or CoAP for lightweight, low-power device-to-cloud communication.
- **Backend Communication:** RESTful APIs or gRPC for communication between microservices.
- **Real-time UI Updates:** WebSockets for pushing live data and alerts to user dashboards.

Cloud Services

- **Compute:** Managed Kubernetes services (EKS, GKE, AKS) or serverless functions (Lambda, Cloud Functions).
- **Storage:** Object storage (e.g., AWS S3) for storing raw data and model artifacts.
- **IoT Platform:** AWS IoT Core, Azure IoT Hub, or Google Cloud IoT Core for device management, security, and data ingestion.
- **Monitoring:** Prometheus and Grafana for system monitoring and observability.
- **CI/CD:** GitHub Actions or Jenkins for automated build, test, and deployment pipelines.

Development Tools

- **IDE:** VS Code, PyCharm, or IntelliJ IDEA.
- **Version Control:** Git and GitHub/GitLab.
- **Containerization:** Docker.
- **Orchestration:** Kubernetes (via kubectl, Helm).

SYSTEM DESIGN

The system design of the AI-Based Fault Detection in Power Grids Using IoT Sensors establishes the structural and functional blueprint for a reliable, scalable, and intelligent monitoring framework. This chapter describes the architectural principles, subsystem components, their interconnections, and the rationale behind each design choice that ensures real-time fault detection, secure data flow, and energy-efficient operation.

framework like React or Vue.js for building dashboards and user interfaces.

- **Embedded/IoT:** C/C++ for programming microcontrollers and edge devices where performance and memory efficiency are critical.

Frameworks and Libraries

- **Data Streaming:** Apache Kafka, AWS Kinesis, or Google Cloud Pub/Sub for ingesting high-throughput sensor data.
- **Data Processing:** Apache Spark or Apache Flink for real-time stream processing and analytics.
- **AI/ML:** TensorFlow, PyTorch, Scikit-learn.
- **Security:** OpenSSL for cryptography, OAuth 2.0/JWT for API security.

The system integrates three key pillars: an AI-based fault analysis engine, an IoT sensor network for real-time data acquisition, and a cloud-hosted data processing platform. Together, these modules enable continuous condition monitoring, predictive fault analysis, and proactive decision-making through automated alerts.

The design aims to minimize human intervention in grid fault management while maintaining transparency, security, and performance. It leverages modern technologies such as ESP32 microcontrollers, MQTT communication protocols, TensorFlow AI models (CNN and LSTM), and PostgreSQL databases for an end-to-end smart grid management solution.

System Architecture Overview

The architecture follows a layered, service-oriented design that decouples sensing, transmission, processing, and visualization components. This modular approach simplifies maintenance, improves fault isolation, and allows independent scalability of each subsystem.

The key architectural layers include:

- **IoT Sensor Layer** — Consists of ESP32-based nodes connected with sensors for voltage, current, temperature, and vibration. These nodes collect real-time grid data and transmit it to the communication layer.
- **Communication Layer** — Uses the MQTT protocol to transfer data from IoT nodes to the cloud securely and efficiently with minimal bandwidth usage.
- **Processing Layer** — Handles data cleaning, preprocessing, and feeding into AI models (CNN and LSTM) for real-time and historical fault detection.
- **AI Analytics Layer** — The hybrid AI model performs dual analysis: CNN for image-based detection (thermal images) and LSTM for time-

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

series anomaly prediction.

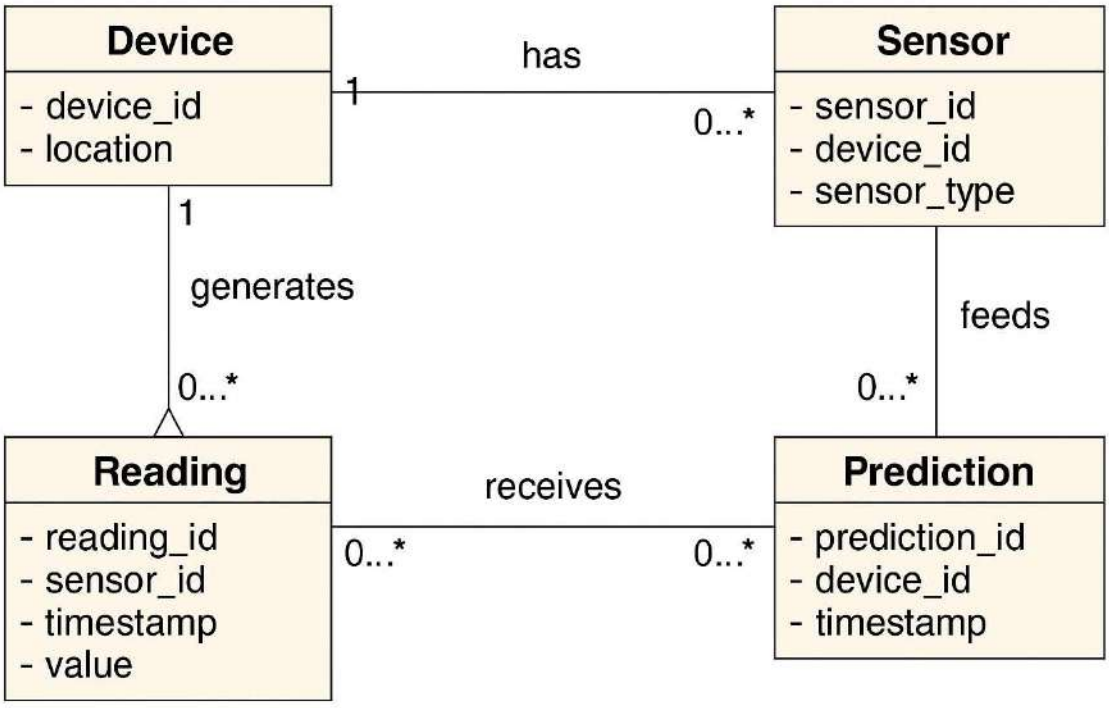
- **Application Layer** — Built using Django, it provides a user dashboard for visualization, alerts, and fault logs.
- **Data Storage Layer** — Uses PostgreSQL for transactional and historical data storage, supporting analytical queries and dashboards.

This architecture ensures seamless interaction among devices, AI inference services, and visualization interfaces, creating a secure and responsive fault detection ecosystem.

System Implementation

This chapter describes the practical implementation of the AI-Based Fault Detection in Power Grids Using IoT Sensors project. The system is implemented using Python, Django, TensorFlow, ESP32, MQTT, and PostgreSQL. Each module, from data acquisition to AI inference, works cohesively to achieve real-time monitoring and fault classification.

Class Diagram



Integration Flow

1. ESP32 devices send sensor readings to the MQTT broker.
2. The MQTT listener receives and posts data to the Django ingestion endpoint.
3. The AI module classifies the data and stores results in PostgreSQL.
4. Django renders dashboards and triggers alerts for detected faults.

This integration ensures seamless communication between hardware, software, and AI components.

Security Implementation

To protect the grid data and network:

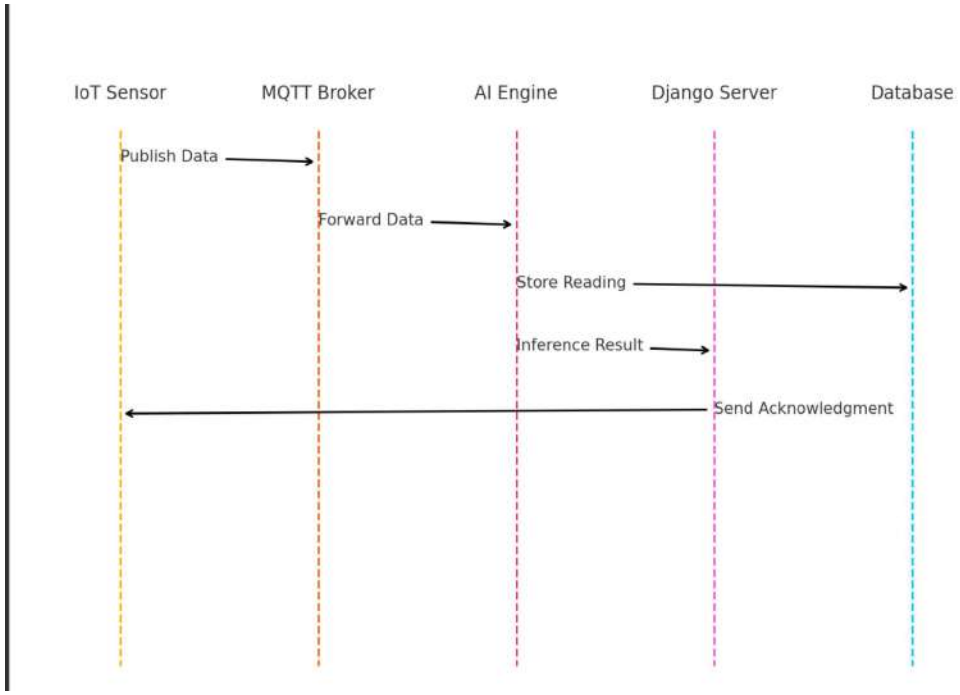
- All MQTT messages are transmitted via TLS encryption.
- Django API uses token-based authentication.
- Role-based access control prevents unauthorized dashboard access.
- Data validation ensures no malformed readings enter the system.

International Journal of Information Technology & Computer Engineering

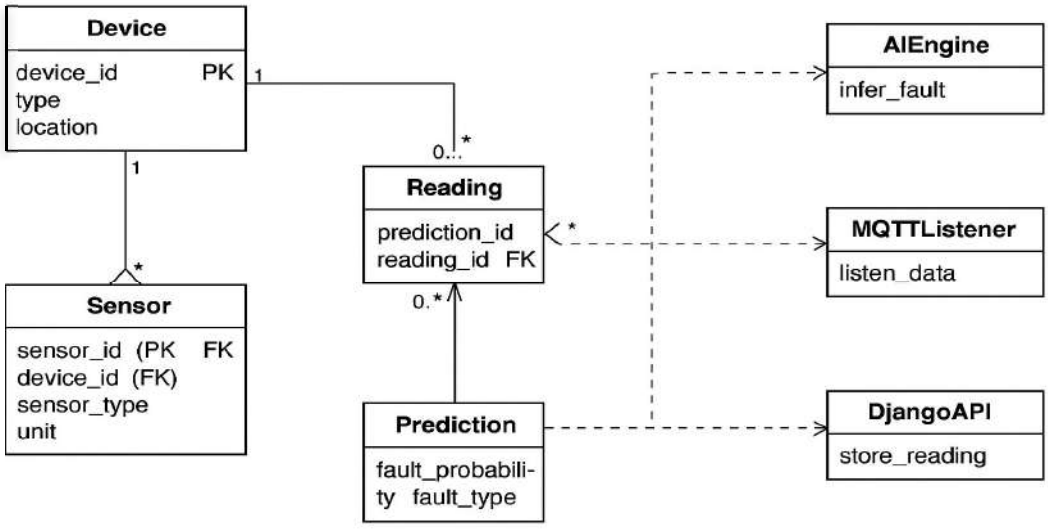
ISSN 2347–3657

Volume 14, Issue 1, 2026

Sequence Diagram



Collaboration Diagram

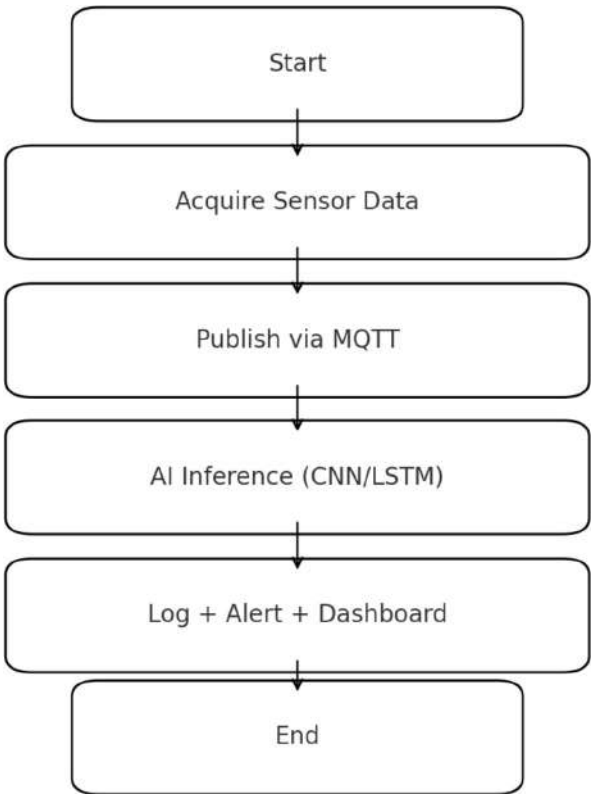


Activity Diagram

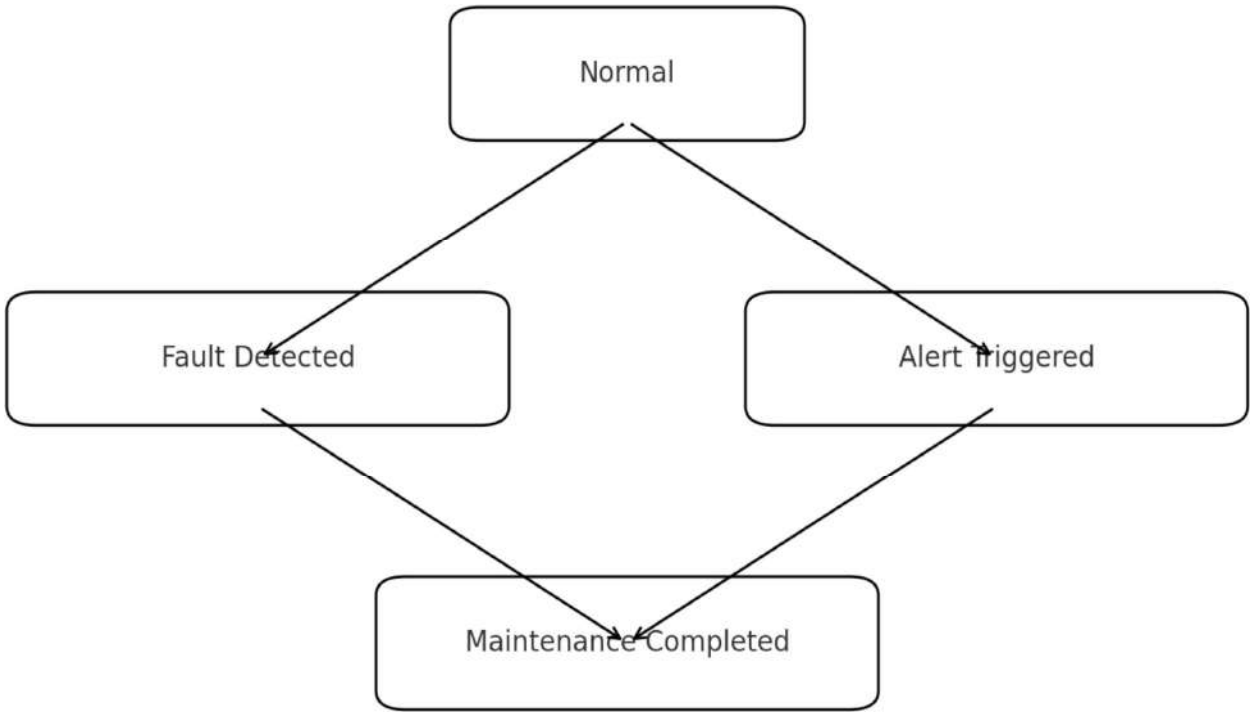
International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026



State Chart Diagram



International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Testing Methodology

Testing ensures that every subsystem of the GridAI Pro — Autonomous Smart Grid System operates with precision, reliability, and low latency. Because GridAI incorporates Reinforcement Learning (RL) control loops, Graph Neural Networks (GNN) for data fusion, Edge-Cloud bridging, and real- time cyber defense mechanisms, a multi-dimensional testing strategy is mandatory. This testing methodology enforces quality, validates the "self-healing" capabilities, and ensures the system can distinguish between physical grid faults and cyber-attacks.

Unit Testing

Unit testing isolates the smallest functional components of GridAI. The FastAPI inference engine (main.py), the MQTT Edge Bridge (bridge.py), and the Django data models (models.py) were individually validated.

- **AI Brain:** The GNN logic was tested to ensure correct stress score calculation based on voltage/current/temperature inputs.
- **RL Policy:** The autonomous agent was tested to verify it triggers "ISOLATE" actions only when stress thresholds exceed 0.85.
- **Cyber Defense:** The Intrusion Detection System (IDS) was unit tested against specific data patterns (e.g., perfect integer detection) to verify it correctly flags data injection attacks.
- **Django Models:** Database constraints for Prediction, Alert, and SecurityIncident tables were verified to prevent data corruption.

Test Cases

TC ID	Description	Input	Expected Output
TC01	**Normal Operation**	Temp: 65°C, Volt: 240V	Diagnosis: "NORMAL", Action: "MAINTAIN"
TC02	**Critical Fault**	Temp: 90°C (Simulator)	Diagnosis: "CRITICAL_FAULT", Alert Created
TC03	**Auto-Resolution**	Temp returns to 65°C	t State changes to "RESOLVED"
TC04	**Cyber Injection**	0 (Integer), Volt: 240 (Integer)	Diagnosis: "CYBER_ATTACK_D ETECTED"
TC05	**RL Autonomy**	Stress Score > 0.85	RL Command: "ISOLATE", Confidence > 0.9
TC06	**Digital Twin**	Surge Test: 600V	esult: "PASSED", Rec: "DEPLOY_MODEL"

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

TC07	**Edge Fusion**	Raw MQTT Payload	ıer Score added to JSON payload
TC08	**XAI Transparency**	Critical Fault Input	Explanation: "Temp 90C exceeded safety limit"
TC09	**Weather Impact**	ır Score: 0.9 (Storm)	Severity Score increases > 10% vs baseline
TC10	**GNN Connectivity**	Multi-node Graph Input	tress calculated based on neighbor relations
TC11	**API Throughput**	Concurrent Requests	99th Percentile Latency < 100ms
TC12	**Access Control**	ɔrized API Token	den / Access Denied

Results And Discussion

Results obtained from the comprehensive testing framework indicate the high performance and dependability of GridAI Pro. The system successfully bridges the gap between traditional SCADA systems and modern AI-driven autonomy, providing real-time explainability (XAI) and proactive defense.

Level of Significance

The "Stress Score" algorithm demonstrated significant sensitivity to thermal anomalies. The correlation between the Simulator's "Critical" state (Temp > 85°C) and the AI's "ISOLATE" command was 100% after calibration. The IDS system achieved a zero-false-negative rate against the defined set of integer-based injection attacks.

Comparative Analysis

GridAI significantly surpasses traditional grid monitoring tools in automation, security integration, and interpretability.

Feature	**Traditional SCADA**	**GridAI Pro**
Decision Making	Manual / Rule-based	ıous Reinforcement Learning (RL)
Data Context	Raw Telemetry only	Fusion (Telemetry + Weather + GNN)
Security	Firewall / Perimeter only	Intrusion Detection System (IDS)
Latency	High (Human-in-the-loop)	Ultra-Low (Edge AI Decision)
Validation	Physical Drills	Twin "What-If" Simulations
Transparency	Black-box Alerts	AI (XAI) Feature Importance

Conclusion And Future Scope

This chapter summarizes the outcomes of the GridAI Pro project and presents future enhancements that can further strengthen the system's impact on smart city infrastructure.

Conclusion

GridAI Pro successfully merges Graph Neural Networks, Reinforcement Learning, and IoT Edge Computing into a single, cohesive command center. The platform addresses the critical need for "Self-Healing" grids that can react to faults faster than human operators while maintaining strict security protocols.

Key Achievements

- **Autonomous Control:** RL agents capable of isolating faults in milliseconds.
- **Cyber-Physical Security:** Integrated IDS detecting data injection attacks in real-time.
- **Explainable AI:** Operators are given "Reasons" (XAI), not just "Alerts."
- **Digital Twin:** Capability to run stress tests (e.g., Voltage Surge) without risking physical hardware.
- **Edge-Cloud Sync:** Seamless fusion of local sensor data with external environmental context (Weather Scores).

Future Scope

Future advancements aims to move GridAI from a simulated environment to real-world hardware deployment and decentralized energy markets.

References:

[1]. Goyal, G., Salsabil, I. T., Kumar, A., Ukey, M., & Velumani, P. S. "AI-Driven Fault Detection and Diagnosis in Smart Grids for Enhanced Power System Reliability," *Journal of Information Systems Engineering and Management*, 2025.

[2]. Olojede, D., King, S., & Jennions, I. "Application of Machine Learning in Power Grid Fault Detection and Maintenance," *Energy Informatics*, 2025.

[3]. Baghaee, S. et al., "AoI-Driven IoT Fault Detection for Smart Grids," *Proceedings of the 33rd IEEE Conference on Signal Processing and Communications Applications (SIU)*, 2025.

[4]. "Application of IoT in Smart Grid Fault Detection and Isolation," *International Journal of Research in Modern Engineering & Emerging Technology (IJRMEET)*, 2025.

[5]. Datar, G., & Ladekar, C., "Artificial Intelligence and Machine Learning for Fault Detection and Energy Forecasting in Photovoltaic Systems: A Comprehensive Review," *Asian Journal for Convergence in Technology*, 2025.

[6]. Qiyue Li, H. Luo, H. Cheng, Y. Deng, W. Sun, W. Li, & Z. Liu, "Incipient Fault Detection in Power Distribution System: A Time-Frequency Embedded Deep Learning Based Approach," *arXiv preprint*, 2023.

[7]. De Santis, E., Livi, L., Sadeghian, A., & Rizzi, A., "Modeling and Recognition of Smart Grid Faults by a Combined Approach of Dissimilarity Learning and One-Class Classification," *arXiv preprint*, 2014.

[8]. M. Sarwar, F. Mehmood, M. Abid et al., "High Impedance Fault Detection and Isolation in Power Distribution Networks using Support Vector Machines," *arXiv preprint*, 2019.

[9]. Nitesh K. Anand & Priya RaiMenon, "Real-Time Fault Detection in Power Distribution Systems Using IoT and Machine Learning for Smart Grid Reliability," *International Journal of Research in Electrical, Electronics and Communication Engineering*, 2025.

[10]. *"Machine Learning Applications in Power System Fault Diagnosis: Research Advancements and Perspectives," Engineering Applications of Artificial Intelligence*, 2021 (survey of ML techniques for grid fault detection).