Design and Implementation of an Edge AI-Based Smart IoT System for Real-Time Environmental Monitoring and Intelligent Decision Making

[1]Ms.V. Revathy, [2]Mrs. Megha U. Vakani, [3]Dr. S. Sivakumar, [4]Mrs. Palak Modi, [5]Ms. Pooja Vora
[6]Ms. Ami Patel
[1]*Assistant Professor, Department of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai*
[2]*Assistant Professor, Department of Electronics and Communication Engineering, Government Engineering College, Bhavnagar*
[3]*Professor, Department of Electrical and Electronics Engineering, AKT memorial college of Engineering and Technology, Kallakurichi*
[4,5]*Assistant Professor, Department of Computer Science Engineering, SAL College of Engineering, Ahmedabad*
[6]*Assistant Professor, Department of IT, SAL Institute of Diploma Studies, Ahmedabad*
doi.org/10.64643/IJIRT209654

***Abstract*—Environmental monitoring supports public health, industrial safety, agriculture, and sustainable development. Conventional IoT monitoring systems often transmit sensor readings to remote servers for analysis, creating dependence on network availability and potentially delaying alerts. This paper proposes an Edge Artificial Intelligence (Edge AI)-based smart Internet of Things system for real-time environmental monitoring and intelligent decision-making. The architecture integrates environmental sensors, an embedded edge controller, a lightweight machine learning model, wireless communication, and a remote dashboard. Temperature, relative humidity, particulate matter, and suitable gas-related measurements are acquired, validated, and processed locally. An anomaly-detection module identifies unusual patterns and supports local alert generation, while cloud connectivity enables historical storage and remote visualization. The design also considers sensor faults, temporary network outages, data buffering, and secure communication. A practical evaluation plan is defined for detection performance, processing latency, communication overhead, energy use, and operational reliability. The proposed framework offers an implementable basis for smart-campus, building, industrial, and urban monitoring. Quantitative performance claims require validation through a working prototype and documented experiments.**

***Index Terms*—Edge AI, Internet of Things, Environmental Monitoring, Machine Learning, Anomaly Detection, Embedded Systems, Intelligent Decision-Making.**

I. INTRODUCTION

Environmental conditions influence human health, workplace safety, agricultural productivity, and the suitability of indoor and outdoor spaces. Changes in temperature, humidity, airborne particles, and harmful gases may require timely attention. Continuous monitoring can help identify undesirable conditions and support corrective action. Manual inspections and standalone instruments may be insufficient when data must be collected continuously across multiple locations.

The Internet of Things (IoT) enables connected sensors, embedded controllers, communication networks, and monitoring applications to work together. However, systems that depend entirely on cloud processing may experience delays or lose analytical capability during communication interruptions. Transmitting every sensor observation can also increase network traffic.

Edge computing processes information near its source, and Edge AI enables machine learning inference to run on an embedded device or nearby edge computer. For environmental monitoring, local inference can identify unusual patterns and initiate alerts without requiring every decision to be made by a remote server. Cloud services can still support long-term storage, historical analysis, and remote supervision.

1.1 Problem Statement

Many monitoring systems emphasize data collection and dashboard display rather than local intelligent analysis. Fixed thresholds can generate false alerts during ordinary fluctuations, while cloud-dependent inference may be unavailable during network failures. An integrated design is therefore needed to validate sensor readings, detect abnormal conditions locally, distinguish sensor faults from environmental events, and provide timely alerts.

1.2 Objectives and Contributions

This work proposes a modular system that combines multi-parameter sensing, edge preprocessing, lightweight anomaly detection, local alert management, offline buffering, and cloud-assisted visualization. Its intended contributions are an integrated architecture, a practical deployment workflow, and an evaluation plan covering detection quality, latency, communication demand, energy use, and reliability. These contributions describe the proposed design; measured benefits must be established experimentally.

II. RELATED WORK

IoT environmental monitoring systems collect measurements through connected sensors and present them through dashboards or data platforms. Their usefulness depends on sensor quality, calibration, installation, maintenance, and reliable communication. Measurements can be affected by noise, sensor drift, environmental interference, and component ageing, making validation an essential part of the system.

Edge computing moves processing closer to the data source and can reduce dependence on remote infrastructure. Shi and colleagues discuss its opportunities and challenges, including support for applications requiring timely processing [1]. Embedded machine learning frameworks also enable supported models to run on resource-constrained devices. TensorFlow Lite for Microcontrollers is one example [2].

Machine learning can identify patterns across several sensor channels that simple threshold rules may not capture. Supervised classifiers require representative labelled examples, whereas anomaly-detection approaches may be useful when abnormal events are rare. Isolation Forest is one candidate for detecting observations that differ from learned normal behavior. Model suitability depends on the quality of training data, environmental variation, hardware resources, and the consequences of false alarms.

The practical challenge is not simply combining sensors and machine learning, but coordinating data validation, inference, decision rules, communication, and fault handling. The proposed framework addresses these functions within a single architecture and defines how they can be evaluated.

III. PROPOSED SYSTEM ARCHITECTURE

The system contains five connected layers: environmental sensing, edge data acquisition, edge AI processing, intelligent decision-making, and communication with the application layer. Sensor measurements flow to the edge controller for validation and preprocessing. The machine learning module analyses the prepared data, and the decision module determines whether a local alert is required. Selected observations and events are then transmitted to a dashboard when connectivity is available.

3.1 Hardware and Software

A prototype may use an ESP32 development board or an equivalent controller, a temperature and humidity sensor, a particulate matter sensor where air-quality monitoring is required, and a suitable gas sensor for the target application. A regulated power supply and a local LED, buzzer, or display can support operation. Optional local storage can retain records during network outages.

Sensor interfaces may include I²C, SPI, UART, or compatible analog and digital connections. Electrical compatibility, grounding, sensor warm-up, signal conditioning, and power requirements must be checked. A general-purpose gas sensor should not be treated as a calibrated concentration instrument for every gas. The exact controller and sensors must be

selected according to the required accuracy, operating environment, memory, and power constraints.

Firmware may be developed using Arduino or ESP-IDF. Model development can use Python and suitable machine learning libraries, with a compatible embedded inference framework for deployment. MQTT is a candidate for lightweight messaging, while HTTPS may suit request-response communication. The dashboard may be developed through an IoT platform or a custom web application.

IV. METHODOLOGY

4.1 Data Acquisition and Preprocessing

The system collects timestamped environmental measurements and records device identity and sensor health. Before inference, the controller checks for missing values, stale readings, communication errors, and observations outside the sensor's specified operating range. Suitable filtering and feature preparation are applied consistently with the model's training procedure. Excessive smoothing should be avoided because it may hide short-duration events.

4.2 Machine Learning and Decision-Making

A lightweight anomaly-detection model is proposed. Isolation Forest may be evaluated when the available training data mainly represent normal conditions; a compact decision tree or another classifier may be more suitable when reliable labelled events are available. Training and validation should use representative observations, and the final evaluation should use independent test data. Where readings are collected over time, separate monitoring sessions or chronological splits can reduce data leakage.

The decision module interprets model output alongside sensor validity and application-specific rules. It can classify the condition as normal, warning, or critical and use persistence checks to reduce repeated nuisance alerts. Urgent safety conditions should not be delayed by persistence rules. Sensor faults must be reported separately from environmental anomalies, and invalid data must never be interpreted automatically as evidence of safety.

4.3 Edge Deployment and Offline Operation

After validation, the model is converted or implemented in a format supported by the selected edge device. The deployed system must use the same feature definitions and preprocessing settings as the training environment. Model loading, memory use, inference time, and stability should be tested on the target hardware.

When the network is unavailable, local acquisition, inference, and alerting should continue where hardware and software remain operational. If storage is available, the device can buffer observations and synchronize them after reconnection. The firmware should handle storage exhaustion, sensor disconnection, unsuccessful model initialization, and unexpected resets.

4.4 Security and Reliability

The system should use authenticated communication, encrypted data transfer where supported, protected device credentials, and controlled dashboard access. Reliability features may include watchdog handling, sensor health checks, timestamp validation, bounded data queues, and documented calibration procedures. Safety-critical deployments require suitable certified sensors and protective mechanisms in addition to machine learning.

V. EXPERIMENTAL EVALUATION

A working prototype should be tested in a controlled setting and, where feasible, at a real monitoring location. Tests should cover normal environmental variation, relevant abnormal conditions, sensor faults, and temporary network disconnection. Reference instruments should be used when measurement accuracy is being assessed. Potentially hazardous tests must follow appropriate laboratory and safety procedures.

A cloud-based baseline should use comparable sensor inputs and a clearly documented processing approach. The timing boundaries, sampling configuration, test duration, and reporting policy should be consistent wherever possible. The evaluation should report the hardware, sensor models, software environment, model configuration, dataset characteristics, and test conditions.

5.1 Evaluation Measures

Detection performance should include accuracy, precision, recall, F1-score, missed events, and false alarms. Because anomalies may be uncommon, overall accuracy alone may be misleading. Processing

latency should distinguish local inference time from end-to-end alert delivery. Communication overhead should be measured from transmitted data volumes during equivalent monitoring periods. Energy consumption should be measured with a suitable instrument and should state whether the complete prototype or selected components are included.

Reliability tests should verify continued local operation during network outages, successful buffering and synchronization, and handling of sensor or storage failures. Repeated tests and clear sample counts should be reported to make the results reproducible.

VI. RESULTS AND DISCUSSION

The following results are illustrative sample values intended only to show the format and style of a results section. They are not verified experimental findings and must be replaced with measurements from the implemented system before submission.

Evaluation item	Edge AI system	Cloud baseline
Classification accuracy	94.2%	92.1%
Precision	93.5%	91.0%
Recall	92.8%	90.4%
F1-score	93.1%	90.7%
Median inference latency	38 ms	210 ms
End-to-end alert latency	120 ms	480 ms
Data transmitted per hour	1.8 MB	6.5 MB
Energy consumption per hour	2.4 Wh	2.7 Wh
Monitoring availability during network outage	Local alerts continued	Remote inference unavailable

Functional testing should first verify sensor acquisition, preprocessing, model loading, inference, alert generation, logging, and dashboard communication. Predictive performance should then be assessed on independent data containing normal and abnormal conditions. The discussion should explain missed events and false alarms and investigate possible causes such as noisy sensors, insufficient training data, or environmental changes.

Local inference avoids the network round trip required for cloud inference, but the actual latency advantage depends on hardware, model complexity, and network conditions. Transmitting selected measurements and event summaries may reduce traffic, although the reduction depends on the reporting policy. Likewise, local processing does not automatically reduce total energy use because sensors and wireless interfaces may dominate consumption. These effects must be established from measurements rather than assumed.

VII. LIMITATIONS AND FUTURE SCOPE

The system's performance depends on sensor calibration, training data coverage, hardware resources, power availability, and maintenance. Models may not generalize to locations with different environmental patterns. False alerts may result from sensor drift or changing conditions. Remote visualization still depends on network availability, while offline retention depends on storage capacity. Machine learning anomaly detection is not a substitute for certified safety equipment or mandatory protective interlocks.

Future work may investigate model compression, adaptive anomaly detection, explainable predictions, secure firmware updates, and long-term field evaluation. Testing across different edge platforms could clarify trade-offs between detection quality, latency, memory, and energy use. Integration with building management systems or distributed monitoring networks may also be explored.

VIII. CONCLUSION

This paper proposes an Edge AI-based smart IoT framework for real-time environmental monitoring and intelligent decision-making. The design integrates multi-parameter sensing, edge preprocessing, lightweight machine learning, local alert generation, wireless communication, and remote visualization. It is intended to support local monitoring without continuous cloud-based inference and to maintain

essential operation during temporary network interruptions. The framework provides an implementable basis for smart-campus, building, industrial, agricultural, and urban monitoring. Its actual benefits must be established through prototype testing, measured comparisons, and documented results.

REFERENCES

[1]. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016. doi: 10.1109/JIOT.2016.2579198.

[2]. R. David et al., "TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems," Proceedings of Machine Learning and Systems, vol. 3, 2021. Available: https://proceedings.mlsys.org/.

[3]. A. Banks, E. Briggs, K. Borgendale, and R. Gupta, eds., "MQTT Version Five: An OASIS Standard," OASIS, 2019. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[4]. Espressif Systems, "ESP32 Series Datasheet and Technical Documentation." Available: https://www.espressif.com/en/products/socs/esp32/documentation.