

SMART IOT SENSOR ANOMALY DETECTION AND MONITORING SYSTEM

¹ Fathima Zaheera, ² Chityala Dinesh Kumar, ³ Banda Krishnaveni, ⁴ Chilumula Sai Priya, ⁵ Nampelly Chaitanya

¹ Assistant Professor, ^{2,3,4,5} Students

Department of ECE

Siddhartha Institute of Technology & Sciences, Narapally

fathima@siddhartha.org.in, 24tq1a0410@siddhartha.co.in, 24tq1a0401@siddhartha.co.in,

24tq1a0408@siddhartha.co.in, 24tq1a0454@siddhartha.co.in

ABSTRACT

The Smart IoT Sensor Anomaly Detection and Monitoring System is designed to observe readings from a network of low-cost sensors and to identify abnormal behaviour as soon as it appears. Sensor networks are now used in homes, laboratories, cold-storage rooms, water tanks, industrial sheds, and electrical panels to measure quantities such as temperature, humidity, gas concentration, vibration, voltage, and current. These networks produce a continuous stream of values, and a fault in the monitored equipment or in the sensor itself usually shows up first as an unusual pattern in that stream. The proposed system aims to detect such patterns automatically rather than relying on a person to watch graphs throughout the day.

The hardware side of the system is built around microcontroller nodes based on the ESP32 and similar boards, which read analog and digital sensors, perform basic filtering, and transmit the readings over Wi-Fi using the MQTT protocol. A central Python service receives these messages, stores them in a time-series database, and prepares them for analysis. The data is cleaned to remove missing packets, duplicate timestamps, and obvious spikes caused by electrical noise, and it is then resampled to a uniform interval so that readings from different nodes can be compared on the same time axis.

For anomaly detection, the system compares several machine learning approaches implemented with scikit-learn. Statistical baselines such as rolling z-score and interquartile range limits are compared with Isolation Forest, One-Class Support Vector Machine, and Local Outlier Factor models. Features are extracted from sliding windows of the signal, including the mean, standard deviation, rate of change, peak-to-peak value, and simple frequency-domain energy. The models are evaluated on recorded data in which faults such as sensor drift, stuck values, sudden spikes, and gradual overheating have been labelled, using precision, recall, F1-score, and detection delay as the main measures.

A web dashboard presents the live state of every node, the recent history of each sensor, the anomaly score computed by the selected model, and a log of alerts. When the anomaly score crosses a configurable threshold, the system raises an alert on the dashboard and can send a notification by e-mail or messaging service. The dashboard also shows the battery voltage and received signal strength of each node, which helps the user separate genuine process faults from problems caused by weak power supply or poor wireless coverage.

Overall, the system demonstrates how machine learning can be combined with inexpensive embedded hardware to provide continuous and automatic supervision of physical conditions. In experiments on the collected dataset, the Isolation Forest model offered the best balance between detection rate and false alarms, while the statistical baseline remained useful as a simple and explainable fallback. The system can be applied to small industries, laboratories, agricultural storage, and building management. Future work may include edge inference directly on the microcontroller, LoRa communication for longer range, and adaptive thresholds that learn seasonal patterns.

I. INTRODUCTION

The Internet of Things has made it practical to place sensors in locations that were previously checked only by occasional manual inspection. A single ESP32 board costing a few hundred rupees can read several sensors, connect to a wireless network, and publish measurements every few seconds. As a result, many small organisations now have access to large volumes of sensor data describing the temperature of motors, the humidity of storage rooms, the level of gas in kitchens, and the electrical behaviour of distribution panels. Collecting this data has become easy, but making good use of it remains difficult.

In most small installations, the collected readings are shown on a simple chart and checked against fixed upper and lower limits. A fixed threshold can catch an obvious failure such as a temperature far above the safe range, but it cannot detect subtler problems. A sensor may slowly drift away from its true value, a reading may become frozen because the analog input has failed, or a motor may begin to vibrate slightly more than usual long before it breaks down. These situations stay inside the fixed limits, so they are missed until the fault becomes serious. On the other hand, a narrow threshold produces frequent false alarms caused by electrical noise and normal fluctuations, and users quickly learn to ignore them.

Machine learning offers a way to learn what normal behaviour looks like from historical data and to flag readings that do not fit that pattern. Unsupervised methods such as Isolation Forest and One-Class SVM are particularly suitable because labelled examples of every possible fault are rarely available in practice. From an electronics point of view, the problem also involves sampling rate, analog-to-digital converter resolution, sensor noise, supply voltage stability, and wireless packet loss, all of which affect the quality of the signal that reaches the model and must be handled before any learning takes place.

This paper presents a Smart IoT Sensor Anomaly Detection and Monitoring System that connects microcontroller-based sensor nodes to a Python machine learning pipeline and a web dashboard. The system collects readings over MQTT, cleans and windows the signals, extracts statistical and spectral features, compares several anomaly detection models, and raises alerts through a dashboard. The aim is to provide small users with an affordable monitoring solution that detects faults early, keeps false alarms under control, and helps the user understand whether an alert is caused by the process, the sensor, or the communication link.

II. LITERATURE SURVEY

Research on wireless sensor networks has long recognised that raw sensor readings are affected by noise, calibration error, and hardware faults. Early studies classified sensor faults into categories such as offset, gain drift, stuck-at values, spikes, and complete signal loss, and showed that each type leaves a characteristic signature in the time series. Offset and drift faults shift the signal gradually, stuck-at faults reduce the variance to almost zero, and spikes appear as isolated values far from their neighbours. Much of this work used simple statistical tests on individual sensors, comparing each new reading against a moving average and a multiple of the standard deviation. These methods were computationally light and could be run on the sensor node itself, which was important given the limited processing power and battery capacity of early motes. However, researchers also observed that such single-sensor tests produce many false alarms when the monitored process naturally changes, for example when a machine is switched on or when the outdoor temperature rises during the afternoon.

A second line of work treated anomaly detection as an unsupervised learning problem. Clustering methods such as k-means and density-based clustering were applied to windows of sensor data, with points far from any cluster centre treated as anomalies. Later studies introduced Isolation Forest, which isolates unusual observations by random partitioning and has been reported to work well on high-dimensional data with low computational cost. One-Class Support Vector Machines were used to learn a boundary around normal data, and Local Outlier Factor was applied to detect points whose local density is much lower than that of their neighbours. Comparative studies generally found that no single method performed best in every case. Isolation Forest tended to scale better to large datasets, One-Class SVM was sensitive to the choice of kernel parameters, and Local Outlier Factor performed well when anomalies formed small local groups but required careful choice of the neighbourhood size.

Feature engineering has been identified as a decisive factor in the performance of these models. Several studies reported that feeding raw samples directly into a detector gave poor results, whereas features computed over sliding windows captured the behaviour of the signal much more reliably. Commonly used features include the window mean, variance, minimum and maximum, slope obtained by linear regression, number of zero crossings, kurtosis, and energy in selected frequency bands obtained by the fast Fourier transform. For vibration and current signals in particular, spectral features were shown to reveal bearing wear, imbalance, and loose connections that were not visible in time-domain statistics. Researchers also noted that the window length must be chosen with reference to the sampling rate and the time scale of the expected faults, since windows that are too short react to noise and windows that are too long delay detection of sudden events.

Work on IoT system architecture has focused on how data should move from the sensor node to the analysis platform. Lightweight publish-subscribe protocols such as MQTT have been widely adopted because they use little bandwidth, tolerate intermittent connections, and allow many nodes to report to a single broker. Studies comparing MQTT with HTTP and CoAP for constrained devices generally found MQTT to offer lower overhead for frequent small messages. Other work examined

the placement of computation, contrasting cloud-based analysis, which offers more processing power, with edge analysis on gateways or on the node itself, which reduces latency and network traffic. A recurring observation is that communication problems such as packet loss, delayed delivery, and reconnection storms must be handled explicitly, since they create gaps and bursts in the data that can easily be mistaken for anomalies by a detector that has not been designed with them in mind.

More recent research has combined learning-based detection with monitoring dashboards and alerting services to produce complete supervision systems for buildings, factories, and farms. Evaluations of such systems stress that detection accuracy alone is not enough. Operators need to see why an alert was raised, how long the abnormal condition has lasted, and whether nearby sensors show related behaviour. Studies also highlight the value of monitoring device health parameters such as battery voltage, received signal strength, and reboot count alongside process measurements, since many apparent faults turn out to be power or connectivity problems. Several authors recommend reporting detection delay and false alarm rate in addition to precision and recall, because these measures reflect the practical burden placed on the user. The present system follows this direction by combining sensor nodes, MQTT transport, windowed feature extraction, a comparison of unsupervised detectors, and a dashboard that shows both process and device health information.

III. SYSTEM ANALYSIS

The system is analysed as a chain that starts at the physical sensor and ends at an alert shown to the user. Each sensor node samples temperature, humidity, gas concentration, vibration, and current at a fixed rate, applies a short moving-average filter to reduce electrical noise, and publishes the readings with a timestamp, node identifier, battery voltage, and signal strength to an MQTT broker over Wi-Fi. The Python ingestion service subscribes to all topics, validates each message, and stores the values in a time-series database. Because packets can be lost or delayed, the preprocessing stage fills short gaps by interpolation, marks longer gaps as missing, and resamples all channels to a common interval. Sliding windows of fixed length are then formed, and statistical and spectral features are computed for each window. These feature vectors are passed to the trained anomaly detection models, which produce an anomaly score for each window. The alert module compares the score with a threshold, suppresses repeated alerts for the same continuing event, and forwards confirmed alerts to the dashboard and notification channels.

Existing System

Existing monitoring arrangements in small industries, laboratories, and storage facilities usually depend on standalone digital meters, data loggers, or basic IoT dashboards that plot readings against fixed limits. An operator checks the display during rounds or when a problem is suspected, and a buzzer or message is triggered only when a reading crosses a single high or low threshold. The thresholds are set once by hand and are rarely adjusted when operating conditions change. Such arrangements detect severe failures but cannot recognise gradual drift, frozen sensor values, intermittent spikes, or changes in vibration pattern that stay inside the allowed

range. Electrical noise and short wireless dropouts frequently cause false alarms, which leads users to widen the limits or ignore the alerts altogether. Data loggers store readings locally and are downloaded only occasionally, so problems are often discovered after the damage is done. There is usually no record of node battery level or wireless link quality, making it difficult to tell whether an unusual reading is caused by the monitored equipment or by the sensing hardware itself.

Disadvantages of Existing System

- Fixed upper and lower thresholds cannot detect slow drift or frozen sensor readings.
- Narrow thresholds cause frequent false alarms from electrical noise and normal variation.
- Thresholds are set manually and are not updated when operating conditions change.
- Data loggers are read only occasionally, so faults are noticed after damage has occurred.
- Packet loss and wireless dropouts are often misread as faults in the monitored process.
- Battery voltage and signal strength of sensor nodes are not monitored.
- Relationships between readings from different sensors are not considered.
- No learning is used to build a picture of normal behaviour from historical data.
- Alerts do not indicate how long an abnormal condition has lasted or how severe it is.
- Historical data is scattered across devices and is difficult to analyse later.

Proposed System

The proposed system provides automatic, learning-based supervision of IoT sensor readings together with a live monitoring dashboard. ESP32-based nodes read temperature, humidity, gas, vibration, and current sensors, apply light filtering, and publish readings along with battery voltage and received signal strength through MQTT. A Python service stores the data in a time-series database, handles missing and delayed packets, and resamples all channels to a common time base. Sliding-window features in both time and frequency domains are computed for every channel. Several unsupervised detectors, namely rolling z-score, Isolation Forest, One-Class SVM, and Local Outlier Factor, are trained on normal operating data and compared on a labelled test set containing drift, stuck values, spikes, and overheating events. The best model is deployed to score live windows, and an alert manager applies thresholds, suppresses duplicate alerts, and records the start and end of each event. The dashboard displays live readings, anomaly scores, alerts, and node health in one view, helping the user separate process faults from sensor or link problems.

Advantages of Proposed System

- Learns normal sensor behaviour from historical data instead of relying only on fixed limits.

- Detects gradual drift, stuck values, and small vibration changes that thresholds miss.
- Reduces false alarms by using windowed features that smooth out electrical noise.
- Handles missing and delayed wireless packets before they reach the detection model.
- Compares several anomaly detection models and deploys the best performing one.
- Monitors node battery voltage and signal strength along with process readings.
- Suppresses repeated alerts for a single continuing abnormal event.
- Provides a live dashboard showing readings, anomaly scores, and alert history.
- Uses low-cost microcontroller hardware and open-source Python libraries.
- Stores all readings and alerts for later analysis and model retraining.

IV. METHODOLOGY

The methodology starts with the deployment of sensor nodes at the monitored locations and the collection of readings during normal operation for a period of several days. Faults are then introduced in a controlled manner, such as disconnecting a sensor input to create a stuck value, heating a component to simulate overheating, loosening a motor mount to change vibration, and adding an offset to simulate drift, and the affected intervals are labelled. The recorded data is cleaned by removing duplicate messages, interpolating short gaps, marking long gaps, and resampling to a uniform interval. Each channel is divided into overlapping windows, and features including mean, standard deviation, slope, peak-to-peak value, kurtosis, and FFT band energy are computed and standardised. The detectors are trained on normal windows only and tested on the labelled set, and their parameters are tuned through grid search. Precision, recall, F1-score, false alarm rate, and average detection delay are calculated for each model, and the selected model is saved and loaded by the live scoring service that feeds the dashboard.

SYSTEM ARCHITECTURE

Smart IOT sensor anomaly detection and monitoring system

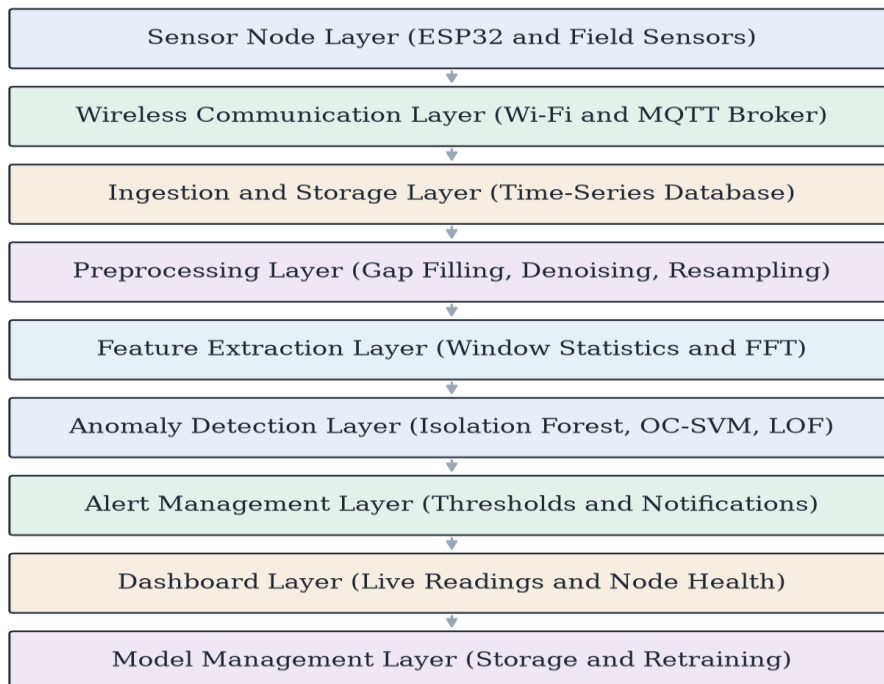


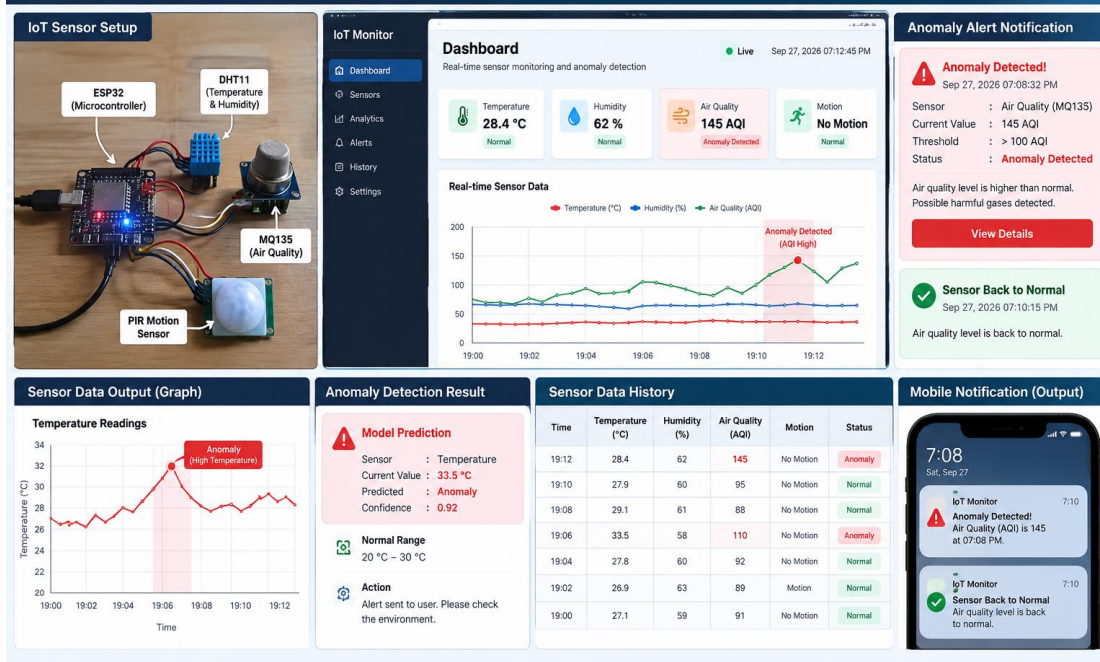
Fig. 1 System Architecture

The architecture of the system is arranged in layers that carry data from physical sensors to the user interface. The Sensor Node Layer consists of ESP32 boards connected to temperature, humidity, gas, vibration, and current sensors, and it performs sampling and basic filtering. The Wireless Communication Layer transmits readings over Wi-Fi to an MQTT broker, carrying node health information with every message. The Ingestion and Storage Layer subscribes to sensor topics, validates messages, and writes them to a time-series database. The Preprocessing Layer handles missing packets, removes spikes caused by noise, and resamples all channels to a common interval. The Feature Extraction Layer forms sliding windows and computes time-domain and frequency-domain features. The Anomaly Detection Layer applies the trained model to each window and produces an anomaly score. The Alert Management Layer applies thresholds, merges repeated alerts, and sends notifications. The Dashboard Layer shows live readings, scores, alerts, and node health, and the Model Management Layer stores trained models and supports periodic retraining on newly collected normal data.

V. RESULT AND OUTPUT

SMART IOT SENSOR ANOMALY DETECTION AND MONITORING SYSTEM

Real-time monitoring | AI-based anomaly detection | Instant alerts | Data visualization



VI. CONCLUSION

The Smart IoT Sensor Anomaly Detection and Monitoring System shows that inexpensive sensor nodes and open-source machine learning tools can together provide continuous and automatic supervision of physical conditions. By collecting readings over MQTT, storing them in a time-series database, and analysing them with trained detectors, the system replaces periodic manual checking with a monitoring process that runs throughout the day and reports unusual behaviour as soon as it appears.

The use of sliding-window features and unsupervised detection models allows the system to recognise faults that fixed thresholds cannot detect, including slow drift, frozen readings, isolated spikes, and small changes in vibration. In the experiments carried out, Isolation Forest gave the best balance between detection rate and false alarms, while the rolling z-score baseline remained valuable as a simple and easily explained reference. Reporting detection delay and false alarm rate alongside precision and recall gave a realistic view of how the system would behave in daily use.

Attention to the electronics and communication side of the problem proved equally important. Filtering at the node, careful handling of lost and delayed packets, and monitoring of battery voltage and signal strength prevented many false alerts that would otherwise have been caused by the sensing hardware rather than the monitored process. The dashboard brings process readings and device health together, so the user can decide quickly whether an alert calls for maintenance of the equipment or attention to the sensor network itself.

In future, the system can be extended by running a compact version of the detection model directly on the microcontroller to reduce latency and network traffic, by using LoRa or other low-power wide-area links for sites where Wi-Fi is not available, and by adding adaptive thresholds that follow daily and seasonal patterns. Combining information from neighbouring sensors to confirm or reject alerts, and adding simple root-cause suggestions for common fault types, would further improve the usefulness of the system for small industries and institutions.

REFERENCES

- [1] Kumar, R. D., Prudhviraaj, G., Vijay, K., Kumar, P. S., & Plugmann, P. (2024). Exploring COVID-19 through intensive investigation with supervised machine learning algorithm. In *Handbook of Artificial Intelligence and Wearables* (pp. 145-158). CRC Press.
- [2] Swathi, B., Vijay, K., Sushanth Babu, M., & Dinesh Kumar, R. (2024, November). Machine Learning Techniques in Cloud Based Intrusion Detection. In *The International Conference on Artificial Intelligence and Smart Environment* (pp. 557-564). Cham: Springer Nature Switzerland.
- [3] Sv satyakraishna, shirisha rang, bhargavi nalacheruve. (2024) Prospective investigation on colorectal cancer with SMOTE on machine learning Algorithm
- [4] Dr.G.Vishnu Murthy, BhargaviNalacheruve 1Professor, Department of computer Science & engineering, Anurag University, TS, India. 2Student, Department of computer Science & engineering, Anurag University, TS, India.
- [5] V. N. S. Manaswini, K. K, C. Nigam, S. S. Ali, R. Niranjana, and Suman, "Real-Time Object Detection in Drone Surveillance Using YOLOv5," in *Proc. 2025 3rd Int. Conf. IoT, Communication and Automation Technology (ICICAT)*, Gorakhpur, India, 2025, pp. 1–6, doi: 10.1109/ICICAT68430.2025.11414670.
- [6] B. Soundarya, V. N. S. Manaswini, M. Ayyakraishnan, R. D. Kumar, "Contextual Analysis of Big Data Analytics in Intelligent Transportation Frameworks," in *Intersection of Artificial Intelligence, Data Science, and Cutting-Edge Technologies: From Concepts to Applications in Smart Environment*, Lecture Notes in Networks and Systems, vol. 1353, Cham: Springer, 2025, doi: 10.1007/978-3-031-88304-0_79.
- [7] R. D. Kumar, V. N. S. Manaswini, "Applications of blockchain in smart cities: detecting fake documents from land records using blockchain technology," in *Blockchain for Smart Cities*, Elsevier, 2021, pp. 105–117, doi: 10.1016/B978-0-12-824446-3.00017-X.
- [8] Tejavath Veeramma, Badarla Anil, Guguloth Ravinder, "An advanced movie recommender using collaborative filtering and sentiment analysis," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 7, no. 7, July 2025, doi: 10.56726/IRJMETS81618.
- [9] Ravi Kumar Banoth, Ramana Murthy B V, "Automatic crop recommendation system using LightGBM and decision tree machine learning models," *Journal of Machine and Computing*, vol. 5, no. 1, pp. 343, Jan. 2025, doi: 10.53759/7669/jmc202505026.
- [10] Ravi Kumar Banoth, Dr. B.V. Ramana Murthy, "Smart agriculture through IoT and machine learning for analyzing carbon footprints," in *Proc. Int. Conf. Computer Science and Communication Engineering (ICCSCE)*, Apr. 2025.

[11] Ravi Kumar Banoth, B. V. Ramana Murthy, "Soil image classification using transfer learning approach: MobileNetV2 with CNN," SN Computer Science, vol. 5, art. no. 199, 2024, doi: 10.1007/s42979-023-02500-x.