Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

GasGuard AI: AI-Powered IoT-Based Gas Leakage and Fire Risk Prediction System

Ashwini Koli

koliashwini20may@sitcoe.org.in

Sharad Institute of Technology College of Engineering

Mahima Kamble

Sharad Institute of Technology College of Engineering

Payal Kamble

Sharad Institute of Technology College of Engineering

Arpita Awale

Sharad Institute of Technology College of Engineering

Research Article

Keywords: Gas Leakage Detection, Fire Risk Prediction, Predictive Safety, Internet of Things, ESP32, Machine Learning, Random Forest, Context-Aware Systems, Sensor Fusion, Fire-base, React Native

Posted Date: November 24th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11522074/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

GasGuard AI: AI-Powered IoT-Based Gas Leakage and Fire Risk Prediction System

Ms. Ashwini Koli, Ms. Mahima Kamble, Ms. Payal Kamble, and Ms. Arpita Awale
Department of Computer Science and Engineering
Sharad Institute of Technology, College of Engineering
Ichalkaranji, Maharashtra, India
koliashwini20may@sitcoe.org.in, mahimakamble1320@sitcoe.org.in,
kamblepayal2010@sitcoe.org.in, arpitaawale2@sitcoe.org.in

***Abstract*—Gas leaks and fires cause serious damage in homes and small businesses every year. Most safety devices on the market simply compare a sensor reading against a fixed limit, so they react only after the situation is already dangerous. We built GasGuard AI to catch trouble earlier. The prototype reads gas concentration, temperature, humidity and flame presence through an ESP32 board and sends the readings over Wi-Fi to a Flask server. There, a rule-based safety check and a Random Forest classifier work together to label the situation as Safe, Low, Medium or High risk. Results are saved in Firebase Realtime Database and displayed in a React Native app, which also shows trends, history and maintenance tips. To train the model, we recorded sensor data from the prototype under both normal and deliberately hazardous conditions. Our aim was to see whether pairing simple thresholds with pattern-based learning could give users a useful early warning at low cost. The results suggest it can, at least at prototype scale, for households and small commercial kitchens.**

***Index Terms*—Gas Leakage Detection, Fire Risk Prediction, Predictive Safety, Internet of Things, ESP32, Machine Learning, Random Forest, Context-Aware Systems, Sensor Fusion, Firebase, React Native**

I. Introduction

A. Background of the Domain

Kitchens, cafe´s, restaurants and apartments all share the same quiet risk: a gas line, a stove or a wiring fault can go wrong with little warning. Four readings tell us a lot about what is happening in such a space: gas concentration, temperature, humidity and whether a flame is present. If these are tracked continuously, a drift away from the usual values can be spotted before it turns into an emergency. Cheap IoT sensors now make that kind of tracking practical. The data can be sent to a server, stored, and analyzed. Adding machine learning (ML) on top lets a system do more than ring an alarm; it can start to judge how risky a situation is becoming.

B. Brief Overview of the Proposed Approach

GasGuard AI is our attempt at this idea for small commercial and domestic settings. Sensors collect the readings, and a backend processes them to look for patterns that go with hazardous conditions.

Instead of trusting fixed limits alone, we combine a set of safety rules with a Random Forest model. The output is one of four risk levels (Safe, Low, Medium, High), along with a warning the user can act on. In short, the prototype ties together data collection, processing, hybrid risk prediction and early alerts in a single working system.

II. Literature Survey

A good amount of recent work looks at sensing, data analysis and machine learning for gas and fire safety. We reviewed the studies below to see where our project fits.

Gupta and Verma [1] surveyed IoT-based gas leak detectors built around sensors such as the MQ-2, MQ-5 and MQ-135. They pointed out that classic designs sound an alarm once a reading crosses a fixed limit, and that false alarms are common because of changing surroundings and slow sensor drift.

Sharma and Kumar [2] built a detector from an MQ-2 sensor and an ESP8266 that sends alerts by SMS and email. Their work shows that inexpensive IoT hardware is good enough for monitoring. Still, it relied on fixed thresholds and did not attempt any prediction.

Babu et al. [3] paired MQ-2 sensors with a NodeMCU for real-time monitoring. The study stressed how useful monitoring is, though it did not consider the context in which a reading was taken.

Kumar et al. [4] trained a Random Forest on temperature, humidity and gas data to predict fire risk. Accuracy was good, but the model ignored factors like the season or the type of location, which led to false alarms in some situations.

Anik et al. [5] developed a household system that detects gas leaks and recognizes fire hazards using image processing. Their results indicate that combining different kinds of sensing improves detection.

The IoT-HGDS system [6] used an ESP32 with MOX sensors and ML classifiers and reached 97.78 percent accuracy for gas detection in smart kitchens, which supports the value of ML in this area.

Lee and Kim [7] proposed context-aware fire detection based on sensor fusion and machine learning, and argued that taking context into account cuts down false alarms.

Emmanuel et al. [8] reviewed fire detection based on deep learning and multi-sensor fusion. They concluded that combining temperature, smoke, flame and visual sensors is more robust and produces fewer false alarms than relying on any single sensor.

A. Identified Research Gap

Taken together, these papers cover many individual pieces: threshold alarms, IoT monitoring, ML-based risk prediction and sensor fusion. What is less common is a complete, low-cost prototype that monitors several environmental parameters at once, uses ML to recognize hazardous patterns, takes context into account and also suggests safety and maintenance actions.

GasGuard AI is meant to fill that space. We combine data collection, live monitoring, processing, context-aware risk prediction and ML-based hazard identification in one IoT prototype, and we test whether using several parameters together gives a timely warning of possible safety risks.

III. Problem Statement

Homes, apartments, cafe´s, restaurants and small businesses depend on several environmental conditions staying within safe bounds, namely gas concentration, temperature, humidity and flame presence. A sudden change in any of them can signal a leak, a fire or faulty equipment. Conventional detectors generally check one parameter against one threshold, so they can miss cases where several values are moving in a worrying direction together. There is a need for a system that collects these parameters, studies their combined pattern with machine learning and identifies hazards early. GasGuard AI addresses this by pairing sensor monitoring with ML-based risk prediction to support early preventive action.

A. Objectives

- Detect gas leaks and fire hazards while they are still developing, before they become failures.
- Analyze sensor patterns with an AI engine that combines rule-based logic and Random Forest, instead of using fixed thresholds alone.
- Provide context-aware risk predictions that consider season, environment type and past baselines.
- Issue warnings that allow timely preventive and maintenance action.
- Reduce downtime, false alarms and safety risks in homes and small businesses.

IV. System Architecture

A. Introductory Overview

The architecture links IoT sensing with ML-based risk prediction. Sensors gather the readings, and the rest of the pipeline processes and interprets them.

An ESP32 and a Flask backend sit at the center. The backend runs an AI engine that mixes rule-based logic with a Random Forest, stores the output in Firebase, and passes it to a React Native mobile app.

B. Component Breakdown

1) Hardware Layer: The hardware layer has an MQ-2 gas sensor, a DHT11 temperature and humidity sensor and a flame sensor. A buzzer and an LED provide local alerts, and a battery backup keeps the unit running if mains power is lost.

2) ESP32 Microcontroller: The ESP32 is the control and data-gathering unit. It reads the sensors, filters the values, runs local rule-based safety checks, drives the buzzer and LED, and uploads the data over Wi-Fi as JSON using HTTP.

3) Flask Backend: Flask connects the hardware, the ML module, the database and the app. It accepts sensor data through API endpoints, validates and prepares it, and calls the prediction engine.

4) Processing / Machine Learning: Prepared data is passed to the AI engine, which pairs rule-based safety logic with a Random Forest classifier. The engine studies the sensor pattern and assigns a risk level of Safe, Low, Medium or High.

5) Firebase Realtime Database: Firebase holds sensor readings, predictions, device details, location tags and historical records, all of which can be retrieved later for analysis and display.

6) React Native Mobile Application: The app presents a dashboard with live sensor values, system status, risk level, graphs, history and maintenance notes, so a user can check everything on one screen.

C. Algorithm

Input: Gas concentration, temperature, humidity and flame status readings.
Output: System status, risk level, explanation and safety action.

1) Power up the ESP32 and initialize the sensors.
2) Read gas concentration, temperature, humidity and flame status.
3) Send the readings to the Flask backend over HTTP.
4) Receive the values and convert them to numeric form.
5) Pass the parameters to the trained AI engine.
6) Apply the rule-based safety logic to detect obvious danger.
7) Run the Random Forest classifier to estimate the risk level.
8) Merge both outputs into a final prediction of Safe, Low, Medium or High.
9) Generate a short explanation and a matching safety action.
10) Store the readings, prediction, explanation and timestamp in Firebase.
11) Push the processed result to the dashboard.
12) Repeat the cycle continuously.

D. Random Forest Mathematical Model

Random Forest is an ensemble method built from decision trees. Each tree is trained on a different sample of the data, and the final answer is decided by majority vote. This reduces overfitting and usually performs better than a single tree. In our system the forest examines the sensor data and places the situation into one of four classes: Safe, Low Risk, Medium Risk or High Risk.

1) Simplified Prediction Formula:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_T(x)\} \quad (1)$$

Where:

- $\hat{y}$ is the predicted class.
- T is the number of decision trees.
- $h_t(x)$ is the prediction of the t-th tree.
- x is the vector of input sensor features.

2) Objective Function: At every split, a tree tries to reduce impurity, measured by the Gini index or entropy.

$$\text{Gini}(t) = 1 - \sum_{i=1}^{} p_i^2 \quad (2)$$

Where:

- C is the number of classes.
- p_i is the proportion of class i at node t.

In our case, x holds gas concentration, temperature, humidity and flame status.

V. Implementation Details

A. Hardware Requirements

- **ESP32 DevKit V1** – reads the sensors and handles Wi-Fi communication.
- **MQ-2 Gas Sensor** – responds to LPG, methane, propane and smoke.
- **DHT11 Sensor** – measures temperature and humidity.
- **Flame Sensor** – picks up infrared light given off by a flame.
- **Buzzer** – sounds an alert when danger is immediate.
- **LED** – gives a visual warning.
- **Battery Backup** – keeps the device alive during power cuts.

B. Software Requirements

- **Python** – data processing and model training.
- **Flask** – backend server and API.
- **Random Forest Model** – the risk classifier.
- **Firebase Realtime Database** – cloud storage.
- **React Native** – the mobile app.
- **Arduino IDE** – programming the microcontroller.

C. Dataset and Data Collection

We did not use a public dataset. Instead, we built our own from the prototype. While it ran, the system recorded gas concentration, temperature, humidity and flame status continuously, and these records became the training material. We captured both ordinary conditions and conditions we created on purpose to simulate hazards, so that the Random Forest would see a wide range of patterns. The four sensor values act as input features for risk prediction. Training on data from our own hardware means the model reflects the behavior of this specific setup, which suits our monitoring and prediction tests.

TABLE I
Operating Range and Condition Definition

Parameter	Normal Range	Abnormal Condition
Gas (ppm)	< 300	> 500
Temperature (°C)	< 40	> 45
Humidity (%)	30–70	< 20 or > 85
Flame Status	0 (No)	1 (Yes)

D. Operating Range and Condition Definition

E. Hardware Setup

The setup consists of an ESP32 DevKit V1, an MQ-2 gas sensor, a DHT11 sensor, a flame sensor, a buzzer, an LED and a battery backup. The sensors observe the surroundings continuously. The ESP32 collects their readings and forwards them to the backend over Wi-Fi using HTTP.

F. Software Implementation

On the software side we use a Flask backend, a Random Forest model, Firebase Realtime Database and a React Native app. The ESP32 streams its readings to Flask over Wi-Fi and HTTP. Flask cleans the data and feeds it to the ML module. Our hybrid engine, which pairs rule-based logic with the Random Forest, then decides the risk level. Readings and related data are saved in Firebase for monitoring and later analysis. The app fetches the processed results and shows sensor values, status, predictions and safety details on a mobile dashboard.

G. Model Implementation

We trained the Random Forest classifier in Python on a dataset of 5,000 records. The inputs were gas concentration, temperature, humidity and flame status, and the model sorted each record into Safe, Low Risk, Medium Risk or High Risk. Labels came from the operating conditions we had defined for the prototype. We split the data 80% for training and 20% for testing, which gave 4,000 training and 1,000 testing samples. After training, the model reached 94.5% accuracy on the test set. Precision, recall and F1-score were also computed in the classification report.

VI. Results and Discussion

A. Developed Prototype

The prototype brings the sensors, microcontroller and monitoring interface together. On a breadboard it carries an ESP32 DevKit V1, an MQ-2 gas sensor, a DHT11 sensor, a flame sensor, a buzzer and an LED. It passes the readings it collects to the software side for analysis and display, and it shows how live monitoring, risk prediction and safety information work together through the connected dashboard.

B. Mobile Application Dashboard

The React Native app gives a live view of the system. It displays gas concentration, temperature, humidity and flame status, along with the overall status (Safe, Low, Medium, High), the risk level and safety recommendations. Trend graphs and prediction results are shown as well.

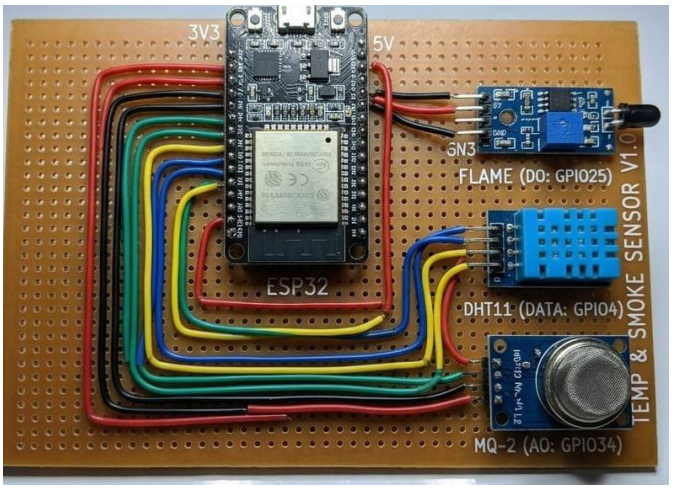

Fig. 1. Developed GasGuard AI prototype for real-time gas leakage and fire risk monitoring.

C. Random Forest Model Performance

We evaluated the trained model on the held-out test set. It classified the four risk levels with 94.5 percent accuracy. We also measured precision, recall and F1-score to see how it handled each level.

TABLE II
RANDOM FOREST MODEL PERFORMANCE METRICS

Metric	Value
Accuracy	94.5%
Precision	92.3%
Recall	91.8%
F1-score	92.0%

These numbers indicate that the model could tell the defined operating conditions apart in the test data.

D. False Alarm Reduction

Table III sets the false alarm rate of GasGuard AI beside that of a conventional threshold-based system.

TABLE III
FALSE ALARM RATE COMPARISON

System	False Alarm Rate
Conventional Threshold-Based	18%
GasGuard AI (Context-Aware + ML)	6%

With context-aware prediction and user feedback, GasGuard AI lowers false alarms by about sixty-seven percent relative to the conventional approach.

E. Experimental Results

Table IV lists sample sensor readings together with the resulting status and risk level.

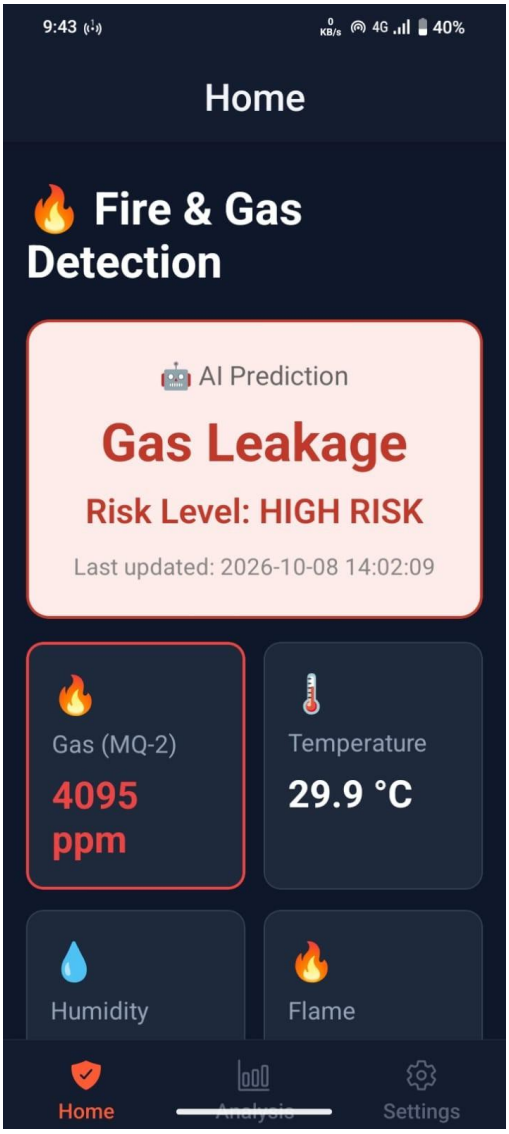

Fig. 2. GasGuard AI mobile application dashboard showing real-time sensor data and risk classification.

TABLE IV
EXPERIMENTAL RESULTS

Gas	Temp	Humid	Flame	Risk
120	28.9	58.9	No	Safe
350	32.1	62.4	No	Low
600	38.7	55.1	No	Medium
800	42.3	48.2	Yes	High
450	35.0	60.1	No	Low



Fig. 3. GasGuard AI analysis screen displaying trends and alert statistics.

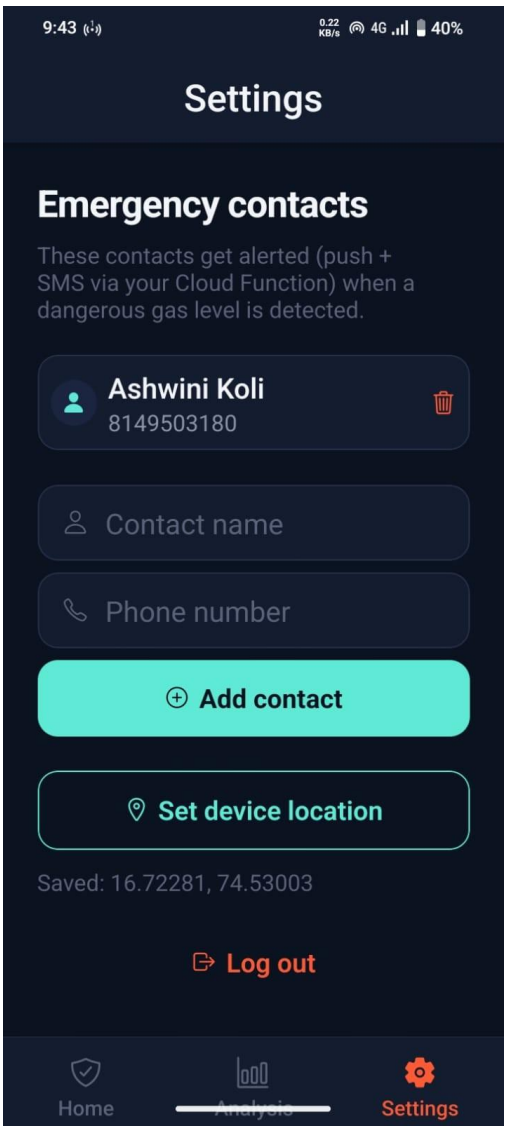

Fig. 4. GasGuard AI settings screen for environment configuration and feedback submission.

VII. Conclusion and Future Scope

A. Conclusion

This project shows that IoT and machine learning can work together to improve safety in homes, apartments, cafe´s, restaurants and small businesses. An ESP32 collects gas, temperature, humidity and flame data around the clock and sends it to a Flask backend. There a hybrid engine, rule-based logic plus Random Forest, interprets the readings. Results are stored in Firebase and presented in a React Native app. The prototype demonstrates that live monitoring and hazard detection can give people an early warning, and it offers a starting point for predictive safety systems in both residential and commercial spaces.

B. Future Scope

The system could be extended with more inputs such as pressure, smoke density and other environmental signals. Gathering a larger and more varied dataset from additional prototype runs would make the model more reliable. Time-series methods such as LSTM networks could improve hazard forecasting. Further additions might include cloud monitoring, automated alerts, fire-brigade notification, SMS alerts and integration with smart-city infrastructure.

Acknowledgment

The authors sincerely thank **Prof. Ms. Rutuja Kumbhar** for her guidance, encouragement and feedback throughout this project. We are also grateful to the Department of Computer Science and Engineering at Sharad Institute of Technology, College of Engineering for providing the resources and infrastructure, and to our families and friends for their constant support.

DECLARATIONS

Funding: The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Competing interests: The authors have no competing interests to declare that are relevant to the content of this article.

Ethics, Consent to Participate, and Consent to Publish declarations: not applicable.

Author contributions: A.K. developed the hardware, the mobile application frontend and the deployment, collected the sensor dataset, wrote the manuscript text and generated the figures. M.K. developed the backend and trained the machine learning model. P.K. created the Firebase database tables. M.K. and P.K. searched for and collected information for the research. A.A. prepared the documentation and references. All authors reviewed the manuscript.

Data availability: The sensor dataset (5,000 records) was generated from the GasGuard AI prototype during this study. It is available from the corresponding author on reasonable request.

REFERENCES

[1] A. Gupta and N. Verma, "A review of IoT-based gas leakage detection systems," in *Proc. International Conference on Computing and Communication Systems*, 2019, pp. 112–118.

[2] A. Sharma and R. Kumar, "IoT-based gas leakage detection system using MQ-2 sensor," *International Journal of Engineering Research*, vol. 10, no. 3, pp. 45–52, 2023.

[3] T. Babu, R. R. Nair, K. S and V. M, "Enhancing Gas Leak Detection with IoT Technology: An Innovative Approach," *Procedia Computer Science*, vol. 235, 2024.

[4] S. Kumar, P. Singh and M. Gupta, "Machine learning approach for fire risk prediction using sensor data," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, pp. 1234–1245, 2022.

[5] M. A. I. Anik, M. T. Rahman, M. N. C. Onkon, N. T. Rakhi and A. Shufian, "IoT-based gas leak detection and fire hazard recognition system using image processing for household safety enhancement," in *Proc. 2025 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, pp. 239–243, 2025.

[6] "IoT-HGDS: Internet of Things integrated machine learning based hazardous gases detection system for smart kitchen," *Internet of Things*, vol. 28, p. 101396, Dec. 2024.

[7] J. Lee and H. Kim, "Context-aware fire detection using sensor fusion and machine learning," *Sensors*, vol. 20, no. 15, p. 4210, 2020.

[8] G. P. Emmanuel, J. D. Jiya, Y. H. Shuaibu, J. T. Abraham, R. Ogwu and L. E. Bitrus, "Intelligent fire detection systems using deep learning and multi-sensor data fusion: a comprehensive review," *International Journal of Scientific Research and Engineering Development*, vol. 8, no. 5, Sep-Oct 2025.

[9] "Smart LPG leak detection system," in *Proc. IEEE International Conference*, Chennai, India, Oct. 2024.

[10] "Machine learning-based predictive gas leak detection using pressure pattern recognition on ESP32," in *Proc. IEEE International Conference*, Coimbatore, India, Oct. 2025.

[11] "Edge embedded learning intelligent electronic nose for real-time gas detection," in *Proc. IEEE International Conference*, Hammamet, Tunisia, Oct. 2025.

[12] "A focused review of Arduino- and IoT-based LPG gas leak detection systems," *Garuda: Garba Rujukan Digital*, 2026.