# **STUDENT NAME: ANISH BANSAL STUDENT ID: 221435713 TASK 9.1P**

**RESEARCH REPORT**

# 1. Introduction

## Problem Identification: Real-Time Fire and Gas Leak Detection in Industrial Zones

Industrial safety is a major global concern, particularly in zones that handle flammable materials, chemicals, or gases. Fires and gas leaks pose serious threats not only to the infrastructure but also to human lives. In the past decade, numerous tragic incidents have occurred due to delayed detection of fire or gas leaks, leading to catastrophic damage, loss of life, and significant financial losses. Current fire and gas leak detection systems in industrial areas are either insufficient in terms of real-time alerting or are prohibitively expensive and inaccessible to smaller industrial units.

**Objective**: This research aims to propose an innovative, affordable, and scalable solution for real-time fire and gas leak detection using IoT sensors, cloud-based analytics, and real-time visualisation dashboards. The goal is to provide early detection, efficient alert mechanisms, and detailed data analytics that can mitigate risks and prevent damage.

# 2. Literature Review on Real-Time Fire and Gas Leak Detection

## A. Mid-Wave Infrared Imaging for Gas Leak Detection

Research by Sun et al. (2024) introduces an advanced technology for gas leak detection using mid-wave infrared (MWIR) imaging combined with visual algorithms. The MWIRGas-YOLO network effectively identifies gas leaks by focusing on low-contrast gas plumes and segments them within the visual field. By using a cooled MWIR system, the method achieves high sensitivity and rapid response, outperforming other mainstream algorithms. The global attention mechanism (GAM) enhances feature fusion, and the transfer learning model utilises visible light smoke features to provide the model with a better understanding of infrared gas characteristics. This method shows a significant improvement over existing technologies in terms of precision and detection capabilities.

### Advantages:

* High precision in identifying low-contrast gas leaks.
* Ability to segment the gas plume within complex scenes.

### Shortcomings:

* It may require a significant computational resource.
* Primarily focuses on gas leak detection and may not be optimised for fire detection.

## B. Ultrasonic-Based Gas Leak Detection

Kim et al. (2024) present a novel ultrasonic sensor-based gas leak detection system for industrial settings. A Monte Carlo simulation approach was used to compare traditional concentration-based gas detectors with the proposed ultrasonic system. The results showed that the combination of the existing concentration-measurement detectors and ultrasonic sensors greatly enhanced detection probabilities, especially for combustible gases in complex environments like oil refining and petrochemical plants.

### Advantages:

* Improved detection probabilities when combined with traditional sensors.
* The ability to detect gas leaks over a range of environmental conditions.

### Shortcomings:

* The system may still be affected by environmental noise, requiring further optimisation.
* Specific implementation challenges exist in large, varied environments like oil refineries​.

C. AI-Enabled IoT Framework for LPG Leak Detection  
Soumya Ranjan Samal et al. (2023) proposed an AI and IoT-based framework to detect gas leakage during LPG transportation and predict its consequences. This approach focuses on using edge devices for real-time detection, quick assessment, and informing disaster management teams. It includes an AI model that predicts the risk and impact radius based on the type of leak detected, offering a rapid response system for transportation hazards. The prototype demonstrated efficient disaster management capabilities by quickly identifying leaks and predicting their consequences.​

## Identified Gaps in Existing Solutions

* **Limited Detection Range**: Both infrared imaging and ultrasonic-based systems need enhancements for broader industrial applicability, particularly in detecting both fires and gas leaks simultaneously.
* **Cost and Scalability Issues**: Existing methods, such as MWIR systems and AI-based pattern recognition, are often costly and complex, limiting their feasibility for small-to-medium enterprises.
* **Environmental Interference**: Ultrasonic and traditional contact-based systems are sensitive to environmental conditions, leading to challenges in ensuring consistent, accurate detection.

These gaps highlight the need for a holistic, scalable solution that can cost-effectively detect both fire and gas leaks in diverse industrial environments without requiring extensive computational power or being overly sensitive to environmental variables.

# 3. Proposed Solution and Methodology

## Overview

This report proposes an **integrated IoT-based fire and gas leak detection system** that is both cost-effective and scalable. The system will utilise a network of sensors to monitor real-time data on temperature, humidity, smoke, and flammable gases. The captured data will be streamed to a cloud-based platform (Arduino IoT Cloud) for storage, visualisation, and real-time alerts. The system will also incorporate machine learning to predict potential fire or gas leaks based on historical data patterns.

## Solution Components and Design

**a) Data Sources and Destinations**

* **Sensors**: Multiple sensors will be used to detect environmental changes:
  + **DHT22**: Measures temperature and humidity, crucial for understanding environmental conditions that can lead to fire hazards.
  + **MQ135**: Detects a variety of gases, including CO2, ammonia, benzene, and smoke.
  + **Flame Sensor Module**: An infrared-based sensor to detect visible flames in case of a fire.
* **Data Destination**: The data from all sensors will be transmitted to the **Arduino IoT Cloud**, which will serve as the primary storage and analytics hub.

**b) Data Types**

* **Temperature & Humidity**: Environmental data indicating the likelihood of a fire.
* **Gas Concentration (PPM)**: The concentration of harmful or flammable gases.
* **Smoke/Flame Presence**: Alerts for the presence of visible flames or smoke particles.

**c) Data Capture Protocol**

* **MQTT Protocol**: A lightweight messaging protocol designed for IoT, enabling efficient data transfer from sensors to the Arduino IoT Cloud.

**d) Data Logging and Storage**

* Data from the sensors will be logged locally on a microSD card for backup purposes and simultaneously streamed to the Arduino IoT Cloud for real-time analysis and historical data tracking.

**e) Storage and Cloud Integration**

* **Arduino IoT Cloud** will serve as the main storage hub for the collected data, allowing real-time access and analytics. A hybrid approach will be used where a mix of **local memory** (microSD) and **cloud storage** will ensure data redundancy.

**f) Data Dashboard for Real-Time Monitoring**

* **Plotly Dash Framework**: A web-based dashboard will be developed using Plotly Dash for live visualisation. The dashboard will feature:
  + Real-time line graphs for temperature, humidity, gas concentration, and flame detection.
  + Alert sections indicate when a certain threshold is breached (for example, gas concentration above a safe level).
  + Export options for full data and visual graphs.

**g) Data Analytics and Pattern Recognition**

* **Threshold-Based Alerts**: Alerts will be triggered in real-time whenever sensor readings exceed predefined safe limits.
* **Predictive Analytics**: A machine learning model will be trained on historical data to predict potential fire or gas leak incidents. This model will use factors like temperature spikes, gas concentration surges, and humidity changes to generate early warnings.
* **Notification Mechanisms**: The system will notify stakeholders (industrial managers safety officers) through SMS, email, or cloud-based alerts.

**h) Visualization and Notification Mechanism**

* **Flow Diagram**:
  1. **Sensor Data Collection**: Data is collected from DHT22, MQ135, and Flame sensors.
  2. **Data Transmission via MQTT**: Sensor data is sent to Arduino IoT Cloud for processing.
  3. **Data Logging**: Local logging on microSD and cloud storage in Arduino IoT.
  4. **Dashboard Update**: Real-time visualisation on Plotly Dash.
  5. **Alert Trigger & Notification**: Alerts are sent based on threshold values and analytics.

# 4.Use Case Overview

The proposed IoT-based fire and gas leak detection system is designed for use in industrial settings, specifically small-to-medium-sized manufacturing plants or storage facilities handling flammable materials. These facilities often lack affordable and scalable safety monitoring systems, leaving them vulnerable to fires and gas leaks.

## Target Users

* **Industrial Safety Officers**: Monitoring the system for potential hazards.
* **Plant Managers**: Receiving alerts to prevent incidents from escalating.
* **Small-to-Medium Enterprises (SMEs)**: Industrial zones where affordability and scalability are key concerns.

**Example Scenario** A small factory storing flammable chemicals has installed the system. During routine operations, a sensor detects an abnormal increase in gas concentration. The system transmits this data to the cloud in real-time and triggers an alert on the dashboard. An SMS alert is sent to the plant manager, who takes immediate action to contain the leak before it escalates into a fire.

## Detailed System Working

**1. Sensor Network and Data Collection**

The foundation of the system is a network of IoT sensors that work together to monitor environmental conditions in industrial settings. The sensors are strategically chosen for their ability to detect specific parameters that may indicate a fire or gas leak.

* **DHT22 Temperature & Humidity Sensor**:
  + *Function*: Measures the surrounding temperature and humidity, both of which are critical indicators of environmental changes that could lead to a fire.
  + *Working*: The DHT22 continuously records temperature and relative humidity and transmits these values to the microcontroller.
* **MQ135 Gas Sensor**:
  + *Function*: Detects multiple types of gases, such as CO2, ammonia, benzene, and other volatile organic compounds (VOCs).
  + *Working*: The MQ135 continuously samples air quality and provides a concentration reading (measured in parts per million) to identify any dangerous buildup of gases.
* **Flame Sensor**:
  + *Function*: Uses infrared detection to identify the presence of visible flames.
  + *Working*: The sensor detects infrared radiation emitted by flames, enabling the system to respond quickly to fire outbreaks.

**2. Data Transfer Using the Arduino Nano 33 IoT**

The central control unit of the system is the **Arduino Nano 33 IoT microcontroller**, which integrates WiFi and BLE (Bluetooth Low Energy) connectivity to efficiently manage the data collected by the sensors. The Arduino microcontroller plays a key role in receiving sensor readings, processing them, and transmitting data to cloud storage for further analysis and visualization.

* **Data Acquisition**: The microcontroller gathers sensor data at regular intervals (e.g., every second) from all connected sensors.
* **Data Filtering and Pre-Processing**: Basic filtering is performed to eliminate noise or erroneous readings from the sensors. This helps improve accuracy and avoid false positives.

**3. Data Transfer Protocol – MQTT**

The **MQTT (Message Queuing Telemetry Transport) protocol** is used to transfer data from the Arduino Nano to the cloud. MQTT is a lightweight protocol designed for IoT applications, making it ideal for low-bandwidth scenarios and efficient real-time data transfer.

* **Broker Setup**: The Arduino Nano acts as a client that publishes sensor data to a topic hosted on the Arduino IoT Cloud (broker).
* **Data Transfer to Cloud**: Every time a new reading is collected, the Arduino Nano sends this data through MQTT to the cloud, ensuring that it is available for monitoring and analytics in real-time.

**4. Data Storage and Logging**

The system uses a **hybrid storage approach** to ensure reliability and data redundancy:

* **Local Storage (MicroSD Card)**: The Arduino Nano logs sensor data to a **microSD card** for local backup. This ensures that data is not lost in case of connectivity issues or cloud failures.
* **Cloud Storage (Arduino IoT Cloud)**: Data is also streamed to the **Arduino IoT Cloud**, serving as the primary storage and analytics hub for real-time access and analysis. The cloud allows scalable data storage, making it easy to track historical patterns and trends.

**5. Data Visualization and Real-Time Monitoring**

To provide a user-friendly monitoring experience, a **dashboard** is developed using **Plotly Dash** – an open-source, web-based visualization framework for Python.

* **Real-Time Data Visualization**: The dashboard will feature interactive line graphs displaying sensor data over time, such as:
  + **Temperature and Humidity** readings (from the DHT22 sensor).
  + **Gas Concentration** levels (from the MQ135 sensor).
  + **Flame Detection Status** (from the Flame sensor).
* **Alert Mechanism**: When any sensor reading exceeds predefined safe limits (e.g., high gas concentration, extreme temperature), an alert is triggered on the dashboard. The alert section will provide detailed information on which sensor detected the anomaly and the exact reading.

**6. Analytics and Alert Mechanisms**

The system employs both threshold-based alerts and predictive analytics to ensure timely detection and prevent potential hazards.

* **Threshold-Based Alerts**: Alerts are generated whenever sensor readings cross certain safety thresholds (e.g., high gas concentration). These alerts are displayed on the dashboard in real-time.
* **Predictive Analytics for Early Detection**: Using historical data collected in the cloud, a **machine learning model** is trained to identify potential patterns that may indicate the likelihood of fire or gas leaks. For instance, temperature spikes combined with a sudden rise in gas concentration may indicate a potential fire.
  + The model is developed using Python libraries such as **Scikit-learn**.
  + Real-time predictions are provided on the dashboard to inform users of any potential risks.

**7. Notifications and Alert System**

The notification system is designed to provide immediate communication to relevant stakeholders to facilitate a prompt response.

* **Multiple Notification Channels**: Alerts are sent via SMS, email, or cloud-based push notifications when safety thresholds are breached.
* **Multi-Sensor Verification**: To prevent false positives, data is cross-verified from multiple sensors before triggering an alert (e.g., verifying gas concentration with temperature and humidity).

**8. Flow of Operations Summary**

The following flow diagram can help visualize the data flow and decision-making in the system:

1. **Data Collection**: Sensors (DHT22, MQ135, Flame sensor) collect environmental data.
2. **Data Processing and Transmission**: The Arduino Nano receives sensor data and processes it. The processed data is transmitted to the cloud using MQTT.
3. **Data Logging and Storage**: Sensor readings are stored locally on a microSD card and sent to the Arduino IoT Cloud for backup and analytics.
4. **Dashboard Visualization**: The Plotly Dash-based dashboard provides real-time graphs and alerts for users to monitor the environment.
5. **Alert and Notification**: If sensor data exceeds safe limits or the predictive model identifies potential risks, notifications are sent to stakeholders, allowing quick action to mitigate hazards.

**9. System Updates and Improvements**

The system is designed to be updated regularly:

* **Refining Thresholds**: Safety thresholds can be adjusted based on historical data to improve alert accuracy.
* **Machine Learning Model Updates**: The predictive model can be retrained with new data to enhance its ability to detect patterns that may indicate potential hazards.
* **Modular Sensor Design**: Additional sensors (e.g., smoke sensors, air pressure sensors) can be added to improve the system's accuracy and applicability across different industrial settings.

# 5. Project Budget and Resources

## Hardware Requirements

To ensure an efficient and robust solution, the hardware selected is compatible with data processing needs and seamless integration with the Arduino IoT Cloud for real-time data acquisition and alerts. Each component's pricing is based on current market rates and vendor availability.

**1. Hardware Costs**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Component** | **Specification** | **Vendor/Source** | **Quantity** | **Price per Unit (USD)** | **Total Cost (USD)** |
| Arduino Nano 33 IoT | Microcontroller with built-in WiFi/BLE | Arduino Store | 1 | 30 | 30 |
| DHT22 Sensor | Measures temperature and humidity | Adafruit | 1 | 10 | 10 |
| MQ135 Gas Sensor | Air quality sensor for CO2, NH3, etc. | Amazon | 1 | 12 | 12 |
| Flame Sensor | UV flame detection | SparkFun | 1 | 5 | 5 |
| MicroSD Card Module | Data logging module for Arduino | Adafruit | 1 | 8 | 8 |
| 32GB MicroSD Card | Data storage for local logging | Sandisk | 1 | 8 | 8 |
| Jumper Wires and Breadboard | For prototyping connections | Amazon/Electronics Shop | 1 set | 10 | 10 |
| 5V Power Adapter | Stable power supply for Arduino board | Local Electronics Store | 1 | 5 | 5 |
| Enclosure for Hardware | Protective case for hardware setup | SparkFun | 1 | 20 | 20 |
| **Subtotal Hardware Cost** |  |  |  |  | **108 USD** |

## Software and Cloud Costs

The system requires software tools for coding, cloud storage, and data visualisation, leveraging both free and subscription-based tools for an optimal solution.

|  |  |  |  |
| --- | --- | --- | --- |
| **Software/Tool** | **Specification/Use** | **License/Cost per Month (USD)** | **Total Cost (USD)** |
| **Arduino IDE** | Programming Arduino devices | Free | 0 |
| **Arduino IoT Cloud** | Cloud platform for IoT devices and data | Free (Basic Plan) | 0 |
| **Python Libraries** | Data handling and visualization (Pandas, Plotly) | Free (Open Source) | 0 |
| **Plotly Dash** | Interactive data dashboard for Python | Free | 0 |
| **VNC Viewer or SSH Client** | Remote access to Arduino setup | Free | 0 |
| **Miscellaneous Software** | Libraries, Debugging tools | Free | 0 |

**Subtotal Software Cost**: **0 USD**

## Operational Costs

* **Internet/WiFi Subscription**: Ensuring stable internet connectivity is essential for real-time data streaming. Estimated **10 USD/month** over the project development (2 months) = **20 USD**.
* **Testing and Calibration Materials**: For accurate sensor calibration (e.g., gas calibration kits, flame simulation tools). Estimated **30 USD**.
* **Shipping/Import Duties for Components**: If sourced from multiple vendors, anticipate shipping costs. Estimated **15 USD**.

**Subtotal Operational Cost**: **65 USD**

**Total Project Budget**

|  |  |
| --- | --- |
| **Category** | **Total Cost (USD)** |
| Hardware | 108 |
| Software | 0 |
| Operational | 65 |
| Grand Total | 173 USD |

## Budget Justification and Cost Efficiency

1. **Hardware**: The selected components are cost-efficient, ensuring necessary functionality without over-expenditure. The Arduino Nano 33 IoT is chosen for its onboard WiFi/BLE, eliminating the need for additional connectivity modules. The DHT22 and MQ135 are reliable yet affordable for environmental monitoring.
2. **Software**: All required software tools are open-source or free, significantly reducing the overall project costs while maintaining robust functionality for coding, cloud integration, and data visualisation.
3. **Operational Costs**: Essential expenses like internet connectivity and sensor calibration are kept minimal but ensure accuracy and reliability in data collection.
4. **Human Resources**: The bulk of the budget is allocated to programming and system integration. With high hourly rates justified by the specialised skills required, the project emphasises expertise in programming for IoT, data visualisation, and machine learning.

This well-rounded budget optimally utilises hardware, software, operational expenses, and skilled labour to deliver a reliable, efficient, and innovative IoT solution for real-time environmental monitoring and safety alerts.

# Detailed Development Plan and Budget

The development plan involves setting up hardware and software components, programming the microcontroller (Arduino), integrating cloud storage, and developing a visualisation dashboard using Plotly Dash. Below is a breakdown of programming efforts and a Gantt chart detailing the project timeline.

**Week 1-2**:

* **Sensor Selection & Hardware Setup**
  + Assemble and test the sensors (DHT22, MQ135, and Flame sensor).
  + Configure the Arduino Nano 33 IoT for sensor data collection and connectivity.
  + Estimated time: **5-7 hours**.

**Week 3-4**:

* **Arduino Programming (Microcontroller Setup)**
  + Write and test code to collect data from sensors.
  + Set up MQTT protocol for data transfer to the Arduino IoT Cloud.
  + Estimated time: **5-7 hours**.

**Week 5**:

* **Back-End Data Processing (Python)**
  + Develop a Python script to visualize the data on the Plotly Dash dashboard.
  + Implement real-time data analysis and create threshold-based alerts.
  + Estimated time: **7-15 hours**.

**Week 6**:

* **Data Storage & Logging**
  + Implement local logging on a microSD card and cloud storage.
  + Integrate data storage capabilities with the Python script and Arduino IoT Cloud.
  + Estimated time: **7-10 hours**.

**Week 7**:

* **Machine Learning Model Development**
  + Collect historical data and clean it for use in predictive modeling.
  + Train and test a basic machine learning model to identify hazard patterns.
  + Estimated time: **7-15 hours**.

**Week 8**:

* **Testing & Validation**
  + Systematically test the entire system: hardware, software, and data flow.
  + Validate real-time alerts and ensure that the dashboard is accurate.
  + **Documentation & User Guide Creation**
    - Document the development process and create a guide for setting up and using the system.
  + Estimated time: **10 hours** (testing) + **5 hours** (documentation).

## b) Gantt Chart - Development Timeline

A Gantt chart is provided below to outline the timeline and dependencies of tasks, with an estimated man-hour allocation per phase:

### **Gantt Chart Breakdown**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Start Date** | **Duration (Days)** | **Estimated Effort (Hours)** | **End Date** |
| **Sensor Selection & Hardware Setup** | Oct 3, 2024 | 2 | 5-7 | Oct 5, 2024 |
| **Arduino Programming (Microcontroller Setup)** | Oct 5, 2024 | 2 | 5-7 | Oct 7, 2024 |
| **Back-End Data Processing (Python)** | Oct 7, 2024 | 2 | 7-15 | Nov 9, 2024 |
| **Data Storage & Logging** | Nov 9, 2024 | 3 | 7-10 | Nov 12, 2024 |
| **Machine Learning Model Development** | Nov 12, 2024 | 3 | 7-15 | Nov 15, 2024 |
| **Testing & Validation** | Nov 15, 2024 | 2 | 10 | Nov 17, 2024 |
| **Documentation & User Guide Creation** | Nov 17, 2024 | 3 | 5 | Nov 20, 2024 |

### **Total Hours Calculation**

6 + 6 + 11 + 8.5 + 11 + 10 + 5 = **57.5 hours**

So, the total estimated effort is approximately **58 hours**

# 6. Ethical Considerations and Data Privacy

## a) Ethical Issues in Data Collection and Usage

* **Data Sensitivity and Security**: The collection of sensor data, particularly in sensitive industrial environments, raises concerns over data privacy and security. Unauthorised access to this data can lead to severe breaches in safety protocols and potential exploitation.
* **Real-Time Monitoring and False Positives**: Real-time monitoring must be accurate, as false positives in the detection of gas leaks or fires can lead to unnecessary panic, resource wastage, and operational downtimes.

## b) Mitigation Strategies for Data Security

1. **Encryption of Data Streams**: All data transmissions between sensors, the microcontroller, and the cloud will be encrypted using TLS/SSL protocols to ensure secure communication.
2. **Data Anonymization**: In case of data sharing with external stakeholders, sensitive data will be anonymised to ensure privacy compliance and mitigate the risks of information leakage.
3. **Access Control & Authentication**: A robust access control mechanism will be implemented for all users and administrators accessing the system. Two-factor authentication (2FA) and role-based access controls will ensure that only authorised personnel can access critical data.
4. **Periodic Audits and Reviews**: Regular system audits and data privacy reviews will be conducted to identify and fix potential vulnerabilities.
5. **Compliance with Legal Frameworks**: The solution will comply with local data privacy regulations (e.g., GDPR, CCPA) to ensure that data collection, processing, and storage practices are ethically sound.

## c) Handling Emergency Scenarios and False Positives

* **Multi-Sensor Verification**: In case of a threshold breach (e.g., gas concentration), data from multiple sensors will be cross verified before triggering alerts, reducing the likelihood of false positives.
* **Automated Alert Acknowledgment**: Users will be required to acknowledge alerts within the dashboard, ensuring that they are not ignored. Alert history will be logged to analyse past incidents and refine system thresholds for future accuracy.

# Conclusion and Future Work

The proposed IoT-based fire and gas leak detection system aims to provide a cost-effective, scalable, and secure solution to mitigate industrial safety risks. With real-time data visualisation, predictive analytics, and secure data handling, this approach strives to fill the gaps left by existing solutions. Future work will focus on refining the predictive model, integrating advanced analytics (e.g., anomaly detection), and expanding the system to other industrial applications, ensuring it remains robust and adaptable to evolving safety requirements.

# REFERENCES

**Sun, et al. (2024).** "Mid-Wave Infrared Imaging for Gas Leak Detection." <https://www.mdpi.com/1424-8220/24/13/4345>

**Kim, et al. (2024).** "Ultrasonic-Based Gas Leak Detection."

<https://www.mdpi.com/2227-9717/12/1/67>

**Samal, S. R., et al. (2023).** "AI-Enabled IoT Framework for LPG Leak Detection."

<https://www.mdpi.com/1424-8220/23/14/6473>

# PANOPTO LINK

<https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=82cfa9b0-bd5a-4cad-8f0a-b1fd00a976cb>

# GITHUB LINK