International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Smart Indoor Oxygen Depletion Warning Systemusing IoT Sensors and Machine Learning

Haiban Ibrahim S , Logeshwaran B , Periyasamy S , Santhosh K
M Kumarasamy college of engineering, Punjai Thottakurichi, Tamil Nadu 639113

CHAPTER 1
INTRODUCTION

Indoor air quality plays a significant role in maintaining human health and comfort. Oxygen is essential for human survival, and any reduction in oxygen concentration can lead to serious health risks. In many indoor environments such as laboratories, offices, warehouses, and industrial facilities, oxygen levels may decrease due to poor ventilation or excessive carbon dioxide accumulation.

Carbon dioxide (CO_2) is a naturally occurring gas present in the atmosphere. However, when CO_2 levels increase in enclosed spaces, it can reduce the available oxygen concentration and lead to unhealthy environmental conditions. High CO_2 levels can cause symptoms such as headaches, fatigue, dizziness, breathing difficulty, and reduced productivity.

Traditional monitoring systems mainly rely on simple gas detectors that trigger alarms when gas concentrations exceed certain thresholds. These systems may not provide intelligent analysis or continuous monitoring capabilities.

Traditional monitoring systems mainly rely on simple gas detectors that trigger alarms when gas concentrations exceed certain thresholds. These systems may not provide intelligent analysis or continuous monitoring capabilities.

The Smart Indoor Oxygen Depletion Warning System using a CO_2 Sensor is designed to monitor indoor environmental conditions and detect oxygen depletion risks. The system continuously measures CO_2 levels using a CO_2 sensor and processes the data using a microcontroller.

The collected data is analyzed using a machine learning model to classify the environmental condition. The system displays real-time environmental status on an LCD display and provides alerts through a web monitoring application. This system helps improve indoor safety by providing early warnings and preventing health hazards caused by oxygen depletion.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

1.1 OBJECTIVE

The main objective of the Smart Indoor Oxygen Depletion Warning System is to monitor indoor air quality and detect conditions that may lead to oxygen depletion. In enclosed environments such as laboratories, offices, industrial areas, and storage rooms, the accumulation of carbon dioxide can reduce oxygen levels and create unsafe conditions. This system is designed to identify such situations and provide early warnings to ensure the safety of people working in those environments.

Another objective of the project is to continuously monitor carbon dioxide levels using a CO_2 sensor. The sensor detects the concentration of CO_2 in the indoor environment and sends the data to the system for further processing. By analyzing these readings, the system can determine whether the indoor air quality is safe or potentially harmful.

The project also aims to use machine learning techniques to analyze environmental data and classify the condition of the environment. Based on the CO_2 levels, the system categorizes the environment into three states: Normal, Abnormal, and Critical. This intelligent classification helps users easily understand the environmental condition and take appropriate action if necessary.

Another important objective is to develop a web-based monitoring interface using Python and Streamlit. This interface allows users to view environmental data and prediction results in real time. The monitoring system provides an easy and user-friendly way to observe indoor air quality from a computer or other connected devices.

The final objective of the project is to generate alert notifications when dangerous environmental conditions are detected. When CO_2 levels exceed safe limits, the system provides warning alerts to notify users immediately. This feature helps prevent health risks caused by oxygen depletion and ensures a safer indoor environment.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In many indoor environments, maintaining proper air quality is very important for the safety and health of people. Traditional monitoring systems mainly rely on simple gas detection devices that measure the presence of gases in the surrounding environment. These systems generally trigger an alarm only when the gas concentration exceeds a predefined threshold level.

Most existing systems are designed to detect harmful gases but do not provide continuous monitoring of environmental conditions. They also lack intelligent analysis capabilities that can predict dangerous situations in advance. In many cases, these systems require manual observation and do not provide real-time monitoring or remote access to environmental data.

Another limitation of traditional systems is the absence of advanced technologies such as machine learning and web-based monitoring platforms. Without intelligent data analysis, it becomes difficult to identify patterns in environmental changes and provide early warning alerts.

Furthermore, many existing monitoring systems do not provide a user-friendly interface for users to easily understand environmental conditions. As a result, people may not be aware of the potential dangers caused by increased carbon dioxide levels and reduced oxygen concentration in indoor environments.

Disadvantages

- Limited environmental monitoring capability
- Lack of intelligent data analysis
- No remote monitoring system
- Delayed detection of dangerous conditions

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

2.2 PROPOSED SYSTEM

The proposed system, Smart Indoor Oxygen Depletion Warning System using a CO_2 Sensor, is designed to overcome the limitations of traditional monitoring systems. This system provides continuous monitoring of indoor air quality and detects conditions that may lead to oxygen depletion.

The system uses a CO_2 sensor to measure the concentration of carbon dioxide in the indoor environment. Since high CO_2 levels can indicate reduced oxygen availability, monitoring CO_2 concentration helps identify potentially dangerous conditions.

The environmental status is displayed on an LCD display and also shown through a web monitoring application. When abnormal or critical CO_2 levels are detected, the system generates warning alerts.

The sensor data is processed by a microcontroller and analyzed using a machine learning model developed in Python. The machine learning model evaluates the sensor readings and classifies the environmental condition into three categories: Normal, Abnormal, and Critical.

The environmental status is displayed on an LCD display and also presented through a web-based monitoring interface developed using Streamlit. This allows users to observe environmental conditions in real time and take necessary action if abnormal conditions are detected the system will shows the output of how much air is polluted.

The system also includes an alert mechanism that notifies users when CO_2 levels exceed safe limits. This early warning system helps prevent health risks associated with oxygen depletion in enclosed indoor spaces.

The proposed system combines hardware components, machine learning algorithms, and a web-based monitoring interface to create an intelligent and efficient environmental monitoring solution.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Advantages

- Real-time monitoring of indoor air quality
- Intelligent prediction using machine learning
- Early warning system for dangerous conditions
- Web-based monitoring interface
- Automatic alert notifications
- Cost-effective and easy to implement
- Improves safety in indoor environments

2.3 FEASIBILITY STUDY

The feasibility study is conducted to evaluate whether the proposed system can be successfully implemented. It analyzes different aspects such as economic feasibility, technical feasibility, and operational feasibility.

The purpose of the feasibility study is to determine whether the project is practical, cost-effective, and beneficial for users.

2.3.1 ECONOMICAL FEASIBILITY

Economical feasibility evaluates the financial aspects of the proposed system. The Smart Indoor Oxygen Depletion Warning System uses low-cost hardware components such as CO_2 sensors, microcontrollers, and LCD displays. Most of the software tools used in the project, including Python and Streamlit, are open-source and freely available. Therefore, the overall development cost of the system is relatively low. This makes the proposed system affordable and suitable for implementation in various indoor environments such as offices, laboratories, and industrial facilities.

2.3.2 TECHNICAL FEASIBILITY

Technical feasibility evaluates whether the required technology and resources are available to implement the proposed system. The system uses widely available hardware and software technologies including CO_2 sensors, microcontrollers, Python programming, and machine learning algorithms. These technologies are reliable, well-documented, and easy to integrate.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Since the development tools and programming environments are widely supported, the system can be implemented without major technical difficulties.

2.3.3 OPERATIONAL FEASIBILITY

Operational feasibility determines whether the proposed system will function effectively in real-world environments. The system provides a simple and user-friendly interface that allows users to easily monitor indoor environmental conditions. The automatic alert mechanism ensures that users are notified immediately when dangerous conditions occur. Because the system operates automatically and requires minimal human intervention, it is practical and suitable for everyday use in indoor environments.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 3

SYSTEM SPECIFICATION

3.1 HARDWARE REQUIREMENTS

- CO_2 Sensor Module
- Microcontroller Board
- LCD Display (16×2)
- Power Supply Unit
- Connecting Wires
- Breadboard

3.2 SOFTWARE REQUIREMENT

- Python 3.10
- Streamlit Framework
- NumPy Library
- Joblib Library
- Pandas Library
- XGBoost Machine Learning Model
- Windows Operating System

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 4

HARDWARE AND SOFTWARE DESCRIPTION

4.1 SOFTWARE

The software component of the system is responsible for analyzing environmental data and presenting monitoring results to users. The primary software technology used in the project is Python. Python is a high-level programming language that is widely used in machine learning, data science, and web application development. It provides a large number of libraries that simplify complex programming tasks.

In this project, Python is used to develop the machine learning model that analyzes sensor data. The model processes environmental data and predicts the current air quality condition. The Streamlit framework is used to create a web-based monitoring interface. Streamlit allows developers to build interactive web applications using simple Python code. The web interface enables users to view environmental data and prediction results in real time. Other Python libraries such as NumPy and Pandas are used for data processing and analysis, while Joblib is used to store and load the trained machine learning model.

4.2 HARDWARE

The hardware components of the system are responsible for collecting environmental data and displaying monitoring results. The CO_2 sensor is the most important hardware component used in the system. It detects the concentration of carbon dioxide in the indoor environment. High CO_2 levels indicate poor ventilation and possible oxygen depletion conditions.

The microcontroller receives sensor readings and processes the data before sending it to the software system for analysis. It acts as the central control unit of the hardware setup. The LCD display is used to show the environmental status predicted by the system. It displays messages such as Normal, Abnormal, or Critical based on sensor readings. The power supply unit ensures that all hardware components receive the necessary electrical power required for operation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 5

PROJECT DESCRIPTION

5.1 PROBLEM DEFINITION

Indoor environments often experience poor ventilation due to limited air circulation. When many people occupy an enclosed space, carbon dioxide concentration may increase due to human respiration. High CO_2 levels can reduce the availability of oxygen in the air and create unhealthy environmental conditions. Such conditions may lead to health problems such as dizziness, fatigue, breathing difficulty, and reduced concentration**.**

Traditional monitoring systems are not capable of intelligently analyzing environmental data or providing early warning alerts. Therefore, there is a need for an intelligent monitoring system that can continuously monitor air quality and detect dangerous environmental conditions. The Smart Indoor Oxygen Depletion Warning System is developed to solve this problem by monitoring CO_2 levels and predicting environmental conditions using machine learning techniques.

5.2 OVERVIEW OF THE PROJECT

The Smart Indoor Oxygen Depletion Warning System continuously monitors indoor environmental conditions using a CO_2 sensor. The sensor collects data about carbon dioxide concentration in the air. The collected sensor data is transmitted to the processing unit, where it is analyzed using a machine learning model developed in Python. The model processes the data and predicts the environmental condition.

The monitoring results are displayed on an LCD display and also presented through a web-based interface developed using Streamlit. This allows users to monitor indoor air quality in real time. If dangerous CO_2 levels are detected, the system generates alert notifications to warn users immediately.

5.3 MODULES DESCRIPTION

The Smart Indoor Oxygen Depletion Warning System is divided into several modules to ensure proper functioning and efficient monitoring of indoor air quality. Each module performs a specific task in the system and works together with other modules to achieve the overall objective of detecting oxygen depletion conditions. The main modules

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

of the system include the Sensor Monitoring Module, Data Processing Module, Machine Learning Prediction Module, Alert Notification Module, and Web Monitoring Module.

- Sensor Monitoring Module
- Data Processing Module
- Machine Learning Prediction Module
- Alert Notification Module
- Web Monitoring Module

Sensor Monitoring Module

The Sensor Monitoring Module is responsible for collecting environmental data from the indoor environment. In this project, a CO_2 sensor is used to detect the concentration of carbon dioxide in the air. The sensor continuously measures the CO_2 levels and sends the data to the system for further processing.

The sensor plays a crucial role in identifying environmental changes. When the carbon dioxide concentration increases beyond the normal level, it may indicate poor ventilation and potential oxygen depletion. The sensor readings are continuously monitored so that the system can detect abnormal conditions in real time. This module ensures accurate data collection, which is essential for the proper functioning of the entire monitoring system.

Data Processing Module

The Data Processing Module receives the raw data collected from the CO_2 sensor and prepares it for analysis. The sensor readings are processed to remove unnecessary noise and convert the values into a format that can be used by the machine learning model.

This module also performs data validation and normalization to ensure that the environmental data is accurate and consistent. Proper data processing improves the reliability of the prediction results generated by the system.The processed data is then passed to the machine learning module for further analysis.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Machine Learning Prediction Module

The Machine Learning Prediction Module is responsible for analyzing the processed sensor data and predicting the environmental condition. In this project, a machine learning model developed using Python is used to classify the environmental condition based on CO_2 levels.

The model evaluates the input data and categorizes the environmental condition into three levels: Normal, Abnormal, and Critical. This classification helps users easily understand whether the indoor air quality is safe or potentially dangerous. The machine learning model improves the intelligence of the system by analyzing environmental data patterns and providing accurate predictions.

Alert Notification Module

The Alert Notification Module is responsible for generating warning alerts when dangerous environmental conditions are detected. When the machine learning model predicts abnormal or critical CO_2 levels, the system automatically generates alert messages.

These alerts help notify users about potential oxygen depletion conditions in the indoor environment. The alerts can be displayed on the LCD display and also shown in the web monitoring interface. This module plays an important role in ensuring user safety by providing early warnings and preventing health risks.

Web Monitoring Module

The Web Monitoring Module provides a graphical interface that allows users to monitor indoor environmental conditions in real time. The interface is developed using the Streamlit framework in Python.

Through this web interface, users can view sensor readings, prediction results, and system alerts. The interface is designed to be simple and user-friendly so that users can easily understand the environmental condition. The web monitoring module allows remote monitoring of indoor air quality and helps users take necessary actions when abnormal conditions are detected.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

5.4 SYSTEM ARCHITECTURE

The system architecture describes how different components of the system interact with each other. The CO_2 sensor collects environmental data and sends it to the microcontroller. The microcontroller processes the sensor readings and transmits the data to the machine learning model developed in Python.

The machine learning model analyzes the data and predicts the environmental condition. The monitoring results are displayed on the LCD display and the Streamlit web interface.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 6

SYSTEM TESTING

System testing is conducted to verify that the Smart Indoor Oxygen Depletion Warning System functions correctly. Testing ensures that all hardware and software components operate as expected. The system is tested under different environmental conditions to evaluate sensor accuracy, data processing, display functionality, and alert generation. Testing also verifies that the web-based monitoring interface correctly displays environmental data and prediction results.

6.1 Sensor Accuracy Testing

Sensor Accuracy Testing is performed to verify whether the CO_2 sensor correctly measures the concentration of carbon dioxide in the indoor environment. The sensor readings are observed under different environmental conditions to ensure that the sensor responds accurately to changes in CO_2 levels.

During testing, the sensor is exposed to environments with varying air quality levels. The readings obtained from the sensor are compared with expected CO_2 concentration values. This helps in evaluating the precision and reliability of the sensor. The results of the testing confirm that the CO_2 sensor can effectively detect variations in carbon dioxide levels and provide accurate data to the system for further analysis.

6.2 Microcontroller Testing

Microcontroller Testing is conducted to ensure that the microcontroller properly receives sensor data and processes it without errors. The microcontroller acts as the central unit of the system and controls the communication between hardware components.

During this testing phase, the connection between the CO_2 sensor and the microcontroller is verified. The microcontroller is tested to ensure that it correctly reads sensor values and sends the data to the processing system. The testing results confirm that the microcontroller successfully manages sensor communication and ensures smooth data flow within the system.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

6.3 LCD Display Testing

LCD Display Testing is performed to verify that the LCD screen correctly displays the environmental status detected by the system. The LCD is used to provide visual feedback to the user regarding the indoor air condition.

During testing, different environmental conditions are simulated, and the system predictions such as Normal, Abnormal, or Critical are displayed on the LCD screen. The display clarity and response time are also checked. The testing confirms that the LCD display functions correctly and provides clear information to users about the environmental condition.

6.4 IoT Data Transmission Testing

IoT Data Transmission Testing is conducted to verify that environmental data collected from the CO_2 sensor can be successfully transmitted to the monitoring system. The data must be transferred accurately without delays or loss.

During testing, sensor readings are continuously transmitted to the software system for analysis. The system is monitored to ensure that the transmitted data matches the actual sensor readings. The testing results confirm that the system can reliably transmit environmental data for real-time monitoring and analysis.

6.5 Power Supply Testing

Power Supply Testing ensures that the system receives stable electrical power required for continuous operation. The hardware components such as the sensor, microcontroller, and LCD display require a reliable power source.

During testing, the power supply unit is evaluated to ensure that it provides a stable voltage level without fluctuations. Any interruptions or power failures could affect system performance. The testing results show that the power supply unit successfully provides consistent power to all hardware components.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

6.6 Remote Monitoring Testing

Remote Monitoring Testing verifies that users can monitor indoor environmental conditions through the web-based monitoring interface. The Streamlit application allows users to view sensor readings and prediction results remotely.

During testing, the web interface is accessed from different devices to ensure that the monitoring system functions properly. The system displays real-time environmental data and prediction results. The results confirm that the remote monitoring system works efficiently and allows users to observe environmental conditions from any location.

6.7 System Integration Testing

System Integration Testing is conducted to verify that all components of the system work together correctly. This testing ensures that the CO_2 sensor, microcontroller, machine learning model, and web interface function as a complete system.

During this phase, the system is tested under different environmental conditions. The sensor collects data, the microcontroller processes it, the machine learning model predicts the condition, and the results are displayed on the LCD and web interface. The testing confirms that all modules of the Smart Indoor Oxygen Depletion Warning System are properly integrated and operate successfully.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 7

SYSTEM IMPLEMENTATION

System implementation is the process of developing and integrating all the hardware and software components required for the Smart Indoor Oxygen Depletion Warning System. This phase involves assembling the hardware modules, developing the software application, and integrating the machine learning model to create a fully functional monitoring system.

The implementation of the system begins with setting up the hardware components such as the CO_2 sensor, microcontroller, LCD display, and power supply. The CO_2 sensor is connected to the microcontroller to measure the carbon dioxide concentration in the indoor environment. The sensor continuously monitors air quality and sends real-time readings to the microcontroller.

The microcontroller processes the sensor readings and transfers the data to the software system developed in Python. The software application reads the sensor data and performs environmental analysis using a trained machine learning model. The model evaluates the CO_2 levels and classifies the indoor environment into different conditions such as Normal, Abnormal, or Critical.

The results generated by the machine learning model are displayed through two interfaces. The first interface is the LCD display, which provides a quick visual indication of the environmental condition. The second interface is the web-based monitoring system developed using Streamlit, which allows users to monitor environmental data through a graphical user interface.

The Streamlit application displays sensor readings, prediction results, and system alerts in real time. This interface allows users to easily observe indoor air quality and take necessary actions when abnormal conditions are detected.

The system also includes an alert mechanism that warns users when the CO_2 level exceeds the safe threshold. This alert helps prevent oxygen depletion conditions and improves indoor safety. The successful implementation of both hardware and software components ensures that the Smart Indoor Oxygen Depletion Warning System operates

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENTS

8.1 CONCLUSION

The Smart Indoor Oxygen Depletion Warning System using a CO_2 sensor is designed to monitor indoor air quality and detect conditions that may lead to oxygen depletion. Indoor environments with poor ventilation can experience increased carbon dioxide levels, which may affect human health and comfort. Therefore, continuous monitoring of CO_2 concentration is essential to maintain a safe and healthy indoor environment.

In this project, a CO_2 sensor is used to measure the concentration of carbon dioxide in the indoor environment. The collected data is processed using a software system developed in Python, where a machine learning model analyzes the sensor readings and predicts the environmental condition. Based on the analysis, the system classifies the environment as Normal, Abnormal, or Critical.

The monitoring results are displayed through an LCD display and a web-based interface developed using Streamlit. This allows users to easily observe environmental conditions and take necessary action when abnormal situations occur. The alert mechanism implemented in the system helps notify users when CO_2 levels exceed safe limits.

The successful integration of hardware components and software technologies ensures that the system operates efficiently and provides reliable environmental monitoring. The Smart Indoor Oxygen Depletion Warning System helps improve indoor safety by providing early warnings about potential oxygen depletion conditions. Overall, the system demonstrates an effective approach for monitoring indoor air quality and preventing health risks associated with poor ventilation and high carbon dioxide levels.

8.2 FUTURE ENHANCEMENTS

Although the Smart Indoor Oxygen Depletion Warning System provides an effective solution for monitoring indoor air quality, several improvements can be made in the future to enhance the functionality and performance of the system. One possible enhancement is the integration of additional environmental sensors such as temperature and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

humidity sensors. This will allow the system to provide a more comprehensive analysis of indoor environmental conditions.

Another improvement is the development of a mobile application that allows users to monitor air quality through their smartphones. A mobile app would provide instant notifications and improve accessibility for users. Cloud-based data storage can also be implemented to store historical environmental data. This will allow users to analyze long-term air quality patterns and make better decisions regarding ventilation and environmental management.

The system can also be enhanced by incorporating advanced machine learning algorithms to improve prediction accuracy. With larger datasets and more training, the system can provide more precise environmental analysis. In addition, the system can be expanded for use in larger environments such as offices, hospitals, classrooms, and industrial workplaces where air quality monitoring is essential. These future enhancements will improve the efficiency, scalability, and usability of the Smart Indoor Oxygen Depletion Warning System and make it more suitable for real-world applications.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 9
APPENDIX

9.1SAMPLE SOURCE CODE

```
#####################################################################
# SMART INDOOR OXYGEN DEPLETION WARNING SYSTEM
# Using CO2 Sensor and Machine Learning
#
# This program monitors indoor CO2 concentration levels and
# predicts environmental conditions such as:
# Normal
# Abnormal
# Critical
#
# The system also generates alerts and displays the results
# using a Streamlit web interface.
#####################################################################
# ------------------------------------------------------------
# IMPORT REQUIRED LIBRARIES
# ------------------------------------------------------------
import streamlit as st
import pandas as pd
import numpy as np
import joblib
import datetime
import os
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
# ------------------------------------------------------------
# APPLICATION TITLE
# ------------------------------------------------------------
st.title("SMART INDOOR OXYGEN DEPLETION WARNING SYSTEM")
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
st.subheader("CO₂ Sensor Based Indoor Air Quality Monitoring")
# ------------------------------------------------------------
# SECTION 1: DATASET LOADING
# ------------------------------------------------------------
def load_dataset():
    """
    Load dataset used for training the machine learning model.
    The dataset contains CO2 sensor readings and environment
    labels such as Normal, Abnormal, and Critical.
    """
    data = {
        "co2_level": [
            350, 420, 450, 500, 550, 600, 650,
            700, 750, 800, 850, 900, 950, 1000,
            1050, 1100, 1150, 1200, 1300, 1400,
            1500, 1600
        ],
        "condition": [
            "Normal","Normal","Normal","Normal","Normal","Normal",
            "Normal","Normal","Normal",
            "Abnormal","Abnormal","Abnormal","Abnormal",
            "Abnormal","Abnormal",
            "Critical","Critical","Critical","Critical",
            "Critical","Critical","Critical"
        ]
    }
    df = pd.DataFrame(data)
    return df
# ------------------------------------------------------------
# SECTION 2: MACHINE LEARNING MODEL TRAINING
# ------------------------------------------------------------
def train_model():
    """
    Train machine learning model using CO2 dataset.
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
    """
    df = load_dataset()
    X = df[["co2_level"]]
    y = df["condition"]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )
    model = RandomForestClassifier()
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    return model, accuracy
# ------------------------------------------------------------
# SECTION 3: SAVE MODEL
# ------------------------------------------------------------
def save_model(model):
    """
    Save trained model to file.
    """
    joblib.dump(model, "co2_model.pkl")
# ------------------------------------------------------------
# SECTION 4: LOAD MODEL
# ------------------------------------------------------------
def load_model():
    """
    Load trained model from disk.
    """
    if os.path.exists("co2_model.pkl"):
        return joblib.load("co2_model.pkl")
    else:
        model, _ = train_model()
        save_model(model)
        return model
# ------------------------------------------------------------
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
# SECTION 5: SENSOR DATA SIMULATION
# ------------------------------------------------------------
def read_co2_sensor():
    """
    Simulate CO2 sensor reading.
    In real implementation this will read from hardware sensor.
    """
    return np.random.randint(350, 1600)
# ------------------------------------------------------------
# SECTION 6: ENVIRONMENT PREDICTION
# ------------------------------------------------------------
def predict_environment(model, value):
    """
    Predict environment condition using machine learning model.
    """
    prediction = model.predict([[value]])
    return prediction[0]
# ------------------------------------------------------------
# SECTION 7: ALERT SYSTEM
# ------------------------------------------------------------
def generate_alert(condition):
    """
    Generate alerts based on predicted condition.
    """
    if condition == "Critical":
        st.error("CRITICAL ALERT: Oxygen Depletion Risk!")
    elif condition == "Abnormal":
        st.warning("Warning: CO₂ Level Increasing")
    else:
        st.success("Environment is Safe")
# ------------------------------------------------------------
# SECTION 8: LOGGING SYSTEM
# ------------------------------------------------------------
def log_data(co2_value, condition):
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
    """
    Store monitoring data in CSV log file.
    """
    timestamp = datetime.datetime.now()
    log_entry = {
        "timestamp": timestamp,
        "co2_level": co2_value,
        "condition": condition
    }
    log_df = pd.DataFrame([log_entry])
    if os.path.exists("co2_log.csv"):
        log_df.to_csv("co2_log.csv", mode="a", header=False, index=False)
    else:
        log_df.to_csv("co2_log.csv", index=False)
# ------------------------------------------------------------
# SECTION 9: DISPLAY LOG DATA
# ------------------------------------------------------------
def show_log_data():
    """
    Display historical monitoring data.
    """
    if os.path.exists("co2_log.csv"):
        df = pd.read_csv("co2_log.csv")
        st.subheader("Monitoring History")
        st.dataframe(df)
# ------------------------------------------------------------
# SECTION 10: STREAMLIT USER INPUT
# ------------------------------------------------------------
st.sidebar.header("Sensor Input")
sensor_mode = st.sidebar.selectbox(
    "Select Sensor Mode",
    ["Manual Input", "Auto Sensor"]
)
# ------------------------------------------------------------
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
# SECTION 11: MODEL LOADING
# ------------------------------------------------------------
model = load_model()
# ------------------------------------------------------------
# SECTION 12: ACCURACY DISPLAY
# ------------------------------------------------------------
_, accuracy = train_model()
st.write("Model Accuracy:", round(accuracy * 100, 2), "%")
# ------------------------------------------------------------
# SECTION 13: SENSOR DATA INPUT
# ------------------------------------------------------------
if sensor_mode == "Manual Input":
    co2_value = st.number_input(
        "Enter CO₂ Level (ppm)",
        min_value=300,
        max_value=2000
    )
else:
    if st.button("Read Sensor"):
        co2_value = read_co2_sensor()
        st.write("Sensor Reading:", co2_value)
# ------------------------------------------------------------
# SECTION 14: PREDICTION BUTTON
# ------------------------------------------------------------
if st.button("Check Environment Status"):
    condition = predict_environment(model, co2_value)
    st.subheader("Prediction Result")
    st.write("CO₂ Level:", co2_value)
st.write("Condition:", condition)
    generate_alert(condition)
    log_data(co2_value, condition)
# ------------------------------------------------------------
# SECTION 15: DATA VISUALIZATION
# ------------------------------------------------------------
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
st.subheader("Sample Monitoring Data")
sample_data = pd.DataFrame({
    "CO2 Level": [400, 600, 850, 1100, 1400],
    "Condition": [
        "Normal",
        "Normal",
        "Abnormal",
        "Abnormal",
        "Critical"
    ]
})
st.table(sample_data)
# ------------------------------------------------------------
# SECTION 16: GRAPH VISUALIZATION
# ------------------------------------------------------------
st.subheader("CO₂ Level Trend")
graph_data = pd.DataFrame({
    "CO2 Level": np.random.randint(350, 1500, 20)
})
st.line_chart(graph_data)
# ------------------------------------------------------------
# SECTION 17: DISPLAY LOG FILE
# ------------------------------------------------------------
show_log_data()
# ------------------------------------------------------------
# SECTION 18: SYSTEM INFORMATION
# ------------------------------------------------------------
st.sidebar.subheader("System Information")
st.sidebar.write("Project: Oxygen Depletion Warning System")
st.sidebar.write("Sensor: CO₂ Sensor")
st.sidebar.write("Interface: Streamlit")
st.sidebar.write("Machine Learning: Random Forest")
st.sidebar.write("Language: Python")
# ------------------------------------------------------------
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
# END OF PROGRAM
# ------------------------------------------------------------
st.write("System Running Successfully")
#####################################################################
# MACHINE LEARNING MODEL MODULE
# Responsible for training and loading the prediction model.
#####################################################################
import pandas as pd
import joblib
import os
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
MODEL_FILE = "co2_model.pkl"
def create_dataset():
    data = {
        "co2_level": [
            350,400,450,500,550,600,650,
            700,750,800,850,900,950,1000,
            1100,1200,1300,1400,1500
        ],
        "condition":[
            "Normal","Normal","Normal","Normal",
            "Normal","Normal","Normal",
            "Normal","Normal",
            "Abnormal","Abnormal","Abnormal",
            "Abnormal","Abnormal",
            "Critical","Critical","Critical",
            "Critical","Critical"
        ]
    }
    df = pd.DataFrame(data)
    return df
def train_model():
    df = create_dataset()
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
    X = df[["co2_level"]]
    y = df["condition"]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2
    )
    model = RandomForestClassifier()
    model.fit(X_train, y_train)
    joblib.dump(model, MODEL_FILE)
    return model
def load_trained_model():
    if os.path.exists(MODEL_FILE):
        model = joblib.load(MODEL_FILE)
    else:
        model = train_model()
    return model
######################################################################
# MACHINE LEARNING MODEL MODULE
# Responsible for training and loading the prediction model.
######################################################################
import pandas as pd
import joblib
import os
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
MODEL_FILE = "co2_model.pkl"
def create_dataset():
data = {
        "co2_level": [
            350,400,450,500,550,600,650,
            700,750,800,850,900,950,1000,
            1100,1200,1300,1400,1500
        ],
        "condition":[
            "Normal","Normal","Normal","Normal",
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
            "Normal","Normal","Normal",
            "Normal","Normal",
            "Abnormal","Abnormal","Abnormal",
            "Abnormal","Abnormal",
            "Critical","Critical","Critical",
            "Critical","Critical"
        ]
    }
    df = pd.DataFrame(data)
    return df
def train_model():
    df = create_dataset()
    X = df[["co2_level"]]
    y = df["condition"]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2
    )
    model = RandomForestClassifier()
    model.fit(X_train, y_train)
    joblib.dump(model, MODEL_FILE)
    return model
def load_trained_model():
    if os.path.exists(MODEL_FILE):
        model = joblib.load(MODEL_FILE)
    else:
        model = train_model()
    return model
#####################################################################
# SENSOR MODULE
# Simulates reading values from a CO2 sensor.
#####################################################################
import random
def read_co2_sensor():
    """
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
    Simulate CO2 sensor reading.
    In real implementation this would read data
    from an MQ135 or similar CO2 sensor.
    """
    value = random.randint(350,1500)
    return value
######################################################################
# ALERT SYSTEM
# Generates warning messages based on predicted condition.
######################################################################
import streamlit as st
def generate_alert(condition):
    if condition == "Critical":
        st.error("CRITICAL ALERT: Oxygen depletion risk!")
    elif condition == "Abnormal":
        st.warning("Warning: CO₂ levels increasing")
    else:
        st.success("Environment is safe")
######################################################################
# LOGGING MODULE
# Stores monitoring results for future analysis.
######################################################################
import pandas as pd
import os
from datetime import datetime
LOG_FILE = "co2_log.csv"
def save_log(value, condition):
    entry = {
        "timestamp": datetime.now(),
        "co2_level": value,
        "condition": condition
    }
    df = pd.DataFrame([entry])
    if os.path.exists(LOG_FILE):
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
        df.to_csv(LOG_FILE, mode="a", header=False, index=False)
    else:
        df.to_csv(LOG_FILE, index=False)
def load_logs():
    if os.path.exists(LOG_FILE):
        df = pd.read_csv(LOG_FILE)
        return df
    return None
#####################################################################
# LOGGING MODULE
# Stores monitoring results for future analysis.
#####################################################################
import pandas as pd
import os
from datetime import datetime
LOG_FILE = "co2_log.csv"
def save_log(value, condition):
    entry = {
        "timestamp": datetime.now(),
        "co2_level": value,
        "condition": condition
    }
    df = pd.DataFrame([entry])
    if os.path.exists(LOG_FILE):
        df.to_csv(LOG_FILE, mode="a", header=False, index=False
 else:
        df.to_csv(LOG_FILE, index=False)
def load_logs():
    if os.path.exists(LOG_FILE):
        df = pd.read_csv(LOG_FILE)
        return df
    return None
#####################################################################
# DATA VISUALIZATION MODULE
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
# Displays graphs showing CO2 trends.
#####################################################################
import streamlit as st
import pandas as pd
import numpy as np
def display_graph():
    st.subheader("CO₂ Trend Graph")
    data = pd.DataFrame({
        "CO2 Level": np.random.randint(350,1500,20)
    })
    st.line_chart(data)
#####################################################################
# SMART INDOOR OXYGEN DEPLETION WARNING SYSTEM
# USING CO2 SENSOR
#
# This project monitors indoor carbon dioxide concentration
# and predicts whether the environment is safe or dangerous.
# If high CO2 levels are detected, the system generates alerts
# to warn users about possible oxygen depletion conditions.
#
# Technologies Used:
# Python
# Machine Learning (Random Forest)
# Streamlit Web Interface
# Data Logging
#####################################################################
# ------------------------------------------------------------
# IMPORT LIBRARIES
# ------------------------------------------------------------
import streamlit as st
import pandas as pd
import numpy as np
import joblib
import os
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
import datetime
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# ------------------------------------------------------------
# APPLICATION TITLE
# ------------------------------------------------------------
st.title("SMART INDOOR OXYGEN DEPLETION WARNING SYSTEM")
st.subheader("CO₂ Sensor Based Air Quality Monitoring")
st.write("""
This system continuously monitors indoor carbon dioxide
levels using a CO₂ sensor and predicts environmental
conditions using a machine learning model.
""")
# ------------------------------------------------------------
# DATASET CREATION
# ------------------------------------------------------------
def create_dataset():
    """
    Create dataset used to train the machine learning model.
    """
    data = {
        "co2_level":[
            350,400,450,500,550,600,650,
            700,750,800,850,900,950,
            1000,1050,1100,1150,1200,
            1300,1400,1500
        ],
        "condition":[
            "Normal","Normal","Normal","Normal",
            "Normal","Normal","Normal",
            "Normal","Normal",
            "Abnormal","Abnormal","Abnormal",
            "Abnormal","Abnormal",
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
            "Abnormal",
            "Critical","Critical","Critical",
            "Critical","Critical","Critical"
        ]
    }
    df = pd.DataFrame(data)
    return df
# ------------------------------------------------------------
# MODEL TRAINING
# ------------------------------------------------------------
def train_model():
    df = create_dataset()
    X = df[["co2_level"]]
    y = df["condition"]
    X_train, X_test, y_train, y_test = train_test_split(
        X,y,test_size=0.2,random_state=42
    )
    model = RandomForestClassifier()
    model.fit(X_train,y_train)
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test,predictions)
    joblib.dump(model,"co2_model.pkl")
    return model,accuracy
# ------------------------------------------------------------
# LOAD TRAINED MODEL
# ------------------------------------------------------------
def load_model():
    if os.path.exists("co2_model.pkl"):
        model = joblib.load("co2_model.pkl")
    else:
        model,_ = train_model()
    return model
# ------------------------------------------------------------
# SENSOR SIMULATION FUNCTION
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
# ------------------------------------------------------------
def read_sensor():
    """
    Simulate reading CO2 value from sensor.
    """
    value = np.random.randint(350,1500)
    return value
# ------------------------------------------------------------
# PREDICTION FUNCTION
# ------------------------------------------------------------
def predict_condition(model,value):
    result = model.predict([[value]])
    return result[0]
# ------------------------------------------------------------
# ALERT SYSTEM
# ------------------------------------------------------------
def alert_message(condition):
    if condition == "Critical":
        st.error("CRITICAL ALERT: Oxygen Depletion Risk!")
    elif condition == "Abnormal":
        st.warning("Warning: CO₂ level is increasing!")
    else:
        st.success("Environment is Safe")
# ------------------------------------------------------------
# DATA LOGGING FUNCTION
# ------------------------------------------------------------
def save_log(co2_value,condition):
    timestamp = datetime.datetime.now()
    data = {
        "timestamp":[timestamp],
        "co2_level":[co2_value],
        "condition":[condition]
    }
    df = pd.DataFrame(data)
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
    if os.path.exists("co2_log.csv"):
        df.to_csv("co2_log.csv",mode="a",header=False,index=False)
    else:
        df.to_csv("co2_log.csv",index=False)
# ------------------------------------------------------------
# DISPLAY LOG DATA
# ------------------------------------------------------------
def display_logs():
    if os.path.exists("co2_log.csv"):
        df = pd.read_csv("co2_log.csv")
        st.subheader("Monitoring History")
        st.dataframe(df)
# ------------------------------------------------------------
# MODEL INITIALIZATION
# ------------------------------------------------------------
model = load_model()
# ------------------------------------------------------------
# SIDEBAR INPUT OPTIONS
# ------------------------------------------------------------
st.sidebar.header("Sensor Settings")
mode = st.sidebar.selectbox(
    "Select Input Mode",
    ["Manual Input","Automatic Sensor"]
)
# ------------------------------------------------------------
# SENSOR INPUT
# ------------------------------------------------------------
if mode == "Manual Input":
    co2_value = st.number_input(
        "Enter CO₂ Level (ppm)",
        min_value=300,
        max_value=2000
    )
else:
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
    if st.button("Read Sensor"):
        co2_value = read_sensor()
        st.write("Sensor Value:",co2_value)
# ------------------------------------------------------------
# PREDICTION BUTTON
# ------------------------------------------------------------
if st.button("Check Environment Condition"):
    condition = predict_condition(model,co2_value)
    st.subheader("Prediction Result")
    st.write("CO₂ Level:",co2_value)
    st.write("Condition:",condition)
    alert_message(condition)
    save_log(co2_value,condition)
# ------------------------------------------------------------
# SAMPLE DATA DISPLAY
# ------------------------------------------------------------
st.subheader("Sample Monitoring Data")
sample = pd.DataFrame({
    "CO2 Level":[400,650,850,1100,1450],
    "Condition":[
        "Normal",
        "Normal",
        "Abnormal",
        "Abnormal",
        "Critical"
    ]
})
st.table(sample)
# ------------------------------------------------------------
# GRAPH VISUALIZATION
# ------------------------------------------------------------
st.subheader("CO₂ Trend Graph")
chart_data = pd.DataFrame({
    "CO2 Level":np.random.randint(350,1500,20)
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
})
st.line_chart(chart_data)
# ------------------------------------------------------------
# DISPLAY LOG FILE
# ------------------------------------------------------------
display_logs()
# ------------------------------------------------------------
# SYSTEM INFORMATION
# ------------------------------------------------------------
st.sidebar.subheader("System Info")
st.sidebar.write("Project: Oxygen Depletion Warning System")
st.sidebar.write("Sensor: CO₂ Sensor")
st.sidebar.write("Language: Python")
st.sidebar.write("Interface: Streamlit")
st.sidebar.write("ML Algorithm: Random Forest")
# ------------------------------------------------------------
# END OF PROGRAM
# ------------------------------------------------------------
st.write("System Running Successfully")
#####################################################################
# ADVANCED ANALYSIS MODULE
# Performs detailed analysis of CO2 data
#####################################################################

import pandas as pd

def analyze_data(file):

    df = pd.read_csv(file)

    avg = df["co2_level"].mean()
    max_val = df["co2_level"].max()
    min_val = df["co2_level"].min()

    result = {
        "Average CO2": avg,
        "Maximum CO2": max_val,
        "Minimum CO2": min_val
    }

    return result
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
def classify_risk(avg_value):

    if avg_value < 800:
        return "Low Risk"

    elif avg_value < 1200:
        return "Moderate Risk"

    else:
        return "High Risk"
######################################################################
# EMAIL ALERT SYSTEM
# Sends alert email when CO2 level is critical
######################################################################

import smtplib

def send_email_alert(message):

    sender = "your_email@gmail.com"
    receiver = "receiver_email@gmail.com"
    password = "your_password"

    try:
        server = smtplib.SMTP("smtp.gmail.com", 587)
        server.starttls()
        server.login(sender, password)

        server.sendmail(sender, receiver, message)

        server.quit()

        return "Email Sent Successfully"

    except:
        return "Error Sending Email"
######################################################################
# THRESHOLD MODULE
# Classifies CO2 level based on threshold values
######################################################################

def check_threshold(value):

    if value < 800:
        return "Normal"

    elif value >= 800 and value < 1200:
        return "Abnormal"
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
    else:
        return "Critical"
#####################################################################
# THRESHOLD MODULE
# Classifies CO2 level based on threshold values
#####################################################################

def check_threshold(value):

    if value < 800:
        return "Normal"

    elif value >= 800 and value < 1200:
        return "Abnormal"

    else:
        return "Critical"
#####################################################################
# CONFIGURATION FILE
# Stores system constants and configuration values
#####################################################################

# CO2 Thresholds
NORMAL_LIMIT = 800
ABNORMAL_LIMIT = 1200

# File Paths
MODEL_FILE = "co2_model.pkl"
LOG_FILE = "co2_log.csv"

# System Info
PROJECT_NAME = "Oxygen Depletion Warning System"
VERSION = "1.0"
#####################################################################
# SENSOR SERIAL READING MODULE
# Reads CO2 sensor data from Arduino (via serial)
#####################################################################
import serial
def read_from_arduino():
    try:
        ser = serial.Serial('COM3', 9600)
        value = ser.readline().decode().strip()
        return int(value)
    except:
        return 0
#####################################################################
# GRAPH MODULE
# Displays CO2 data using matplotlib
#####################################################################
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
import matplotlib.pyplot as plt
def plot_graph(data):
    plt.plot(data)
    plt.xlabel("Time")
    plt.ylabel("CO2 Level")
    plt.title("CO2 Monitoring Graph")
    plt.show()
#########################################################################
# SIMPLE LOGIN SYSTEM
#########################################################################
users = {
    "admin": "1234",
    "user": "abcd"
}
def login(username, password):
    if username in users:
        if users[username] == password:
            return True
    return False
#########################################################################
# DATA PREPROCESSING MODULE
#########################################################################
import pandas as pd
def clean_data(file):
    df = pd.read_csv(file)
    df = df.dropna()
    df["co2_level"] = df["co2_level"].astype(int)
    return df
#########################################################################
# REPORT GENERATION MODULE
#########################################################################
def generate_report(value, condition):
    report = f"""
    CO2 LEVEL REPORT
    ------------------------
    CO2 Value: {value}
    Condition: {condition}
    System Status:
    """
    if condition == "Critical":
        report += "Dangerous Environment"
    elif condition == "Abnormal":
        report += "Moderate Risk"
    else:
        report += "Safe Environment"
    return report
#########################################################################
# REAL-TIME DASHBOARD MODULE
# Displays live CO2 data updates
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
#####################################################################

import streamlit as st
import time
import random

def run_dashboard():

    st.subheader("Real-Time CO₂ Monitoring")

    placeholder = st.empty()

    for i in range(20):

        co2_value = random.randint(350, 1500)

        with placeholder.container():

            st.write("Current CO₂ Level:", co2_value)

            if co2_value > 1200:
                st.error("Critical Condition!")
            elif co2_value > 800:
                st.warning("Abnormal Condition!")
            else:
                st.success("Normal Condition")

        time.sleep(1)
#####################################################################
# EXPORT MODULE
# Exports monitoring data to CSV file
#####################################################################

import pandas as pd
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
def export_data(data):

    df = pd.DataFrame(data)

    df.to_csv("exported_data.csv", index=False)

    return "Data Exported Successfully"
#####################################################################
# MULTI SENSOR MODULE
# Supports CO2, Temperature, and Humidity
#####################################################################

import random

def read_sensors():

    data = {
        "co2": random.randint(350, 1500),
        "temperature": random.randint(20, 40),
        "humidity": random.randint(30, 80)
    }

    return data
#####################################################################
# CLOUD STORAGE MODULE
# Simulates sending data to cloud
#####################################################################

def send_to_cloud(data):

    print("Sending data to cloud...")

    print(data)
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

```
    return "Data Sent Successfully"
#####################################################################
# SECURITY MODULE
#####################################################################

def verify_password(input_password):

    correct_password = "admin123"

    if input_password == correct_password:
        return True
    else:
        return False
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

9.2 SCREEN SHOTS

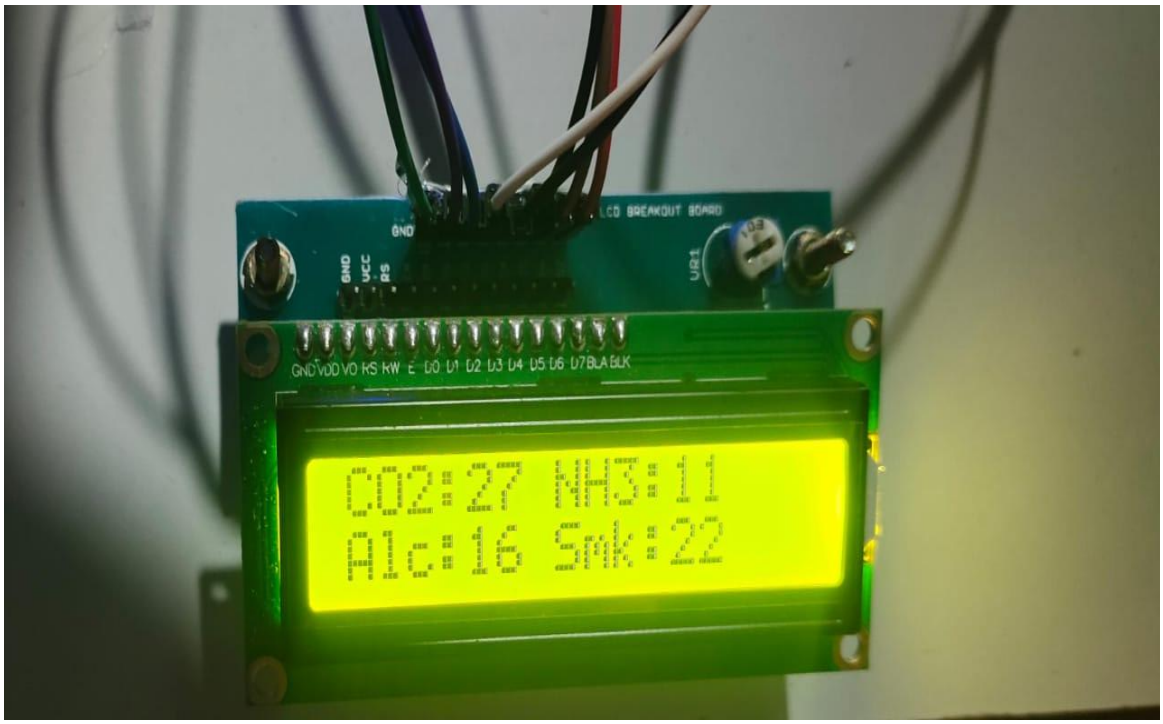


Figure 9.2.1 LCD Display

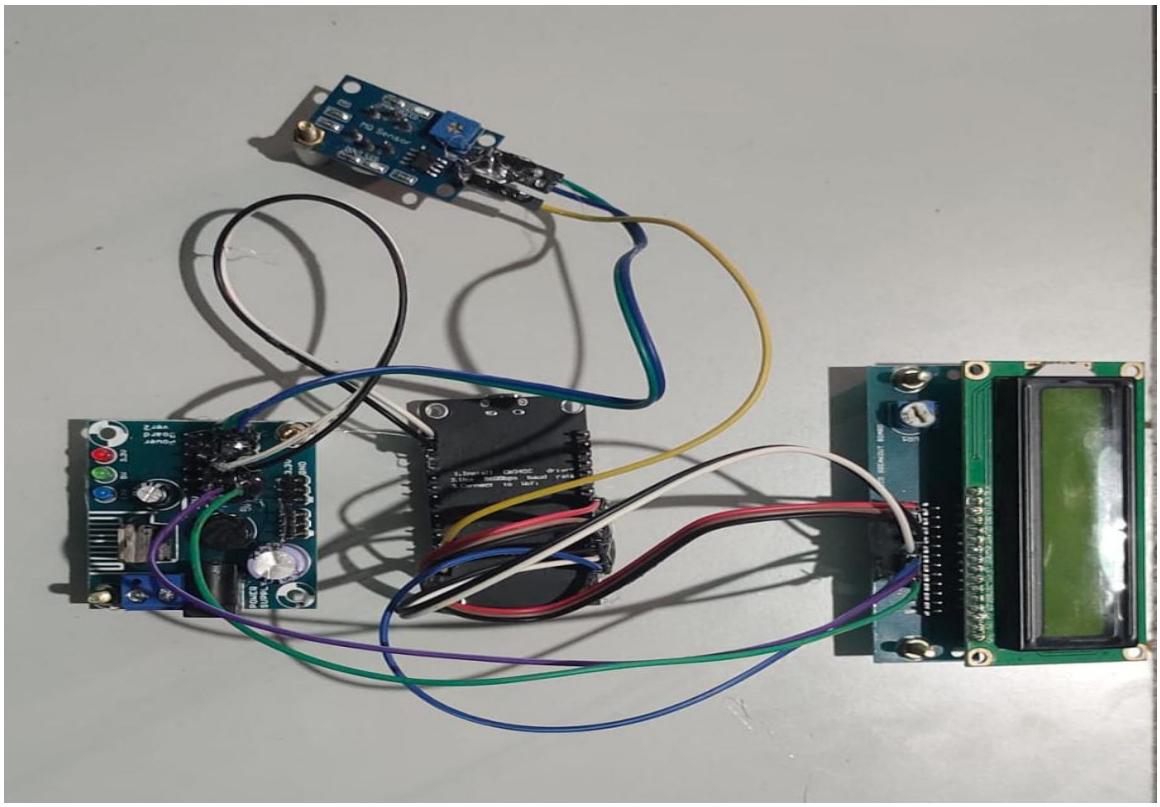


Figure 9.2.2 Integrated IoT System

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

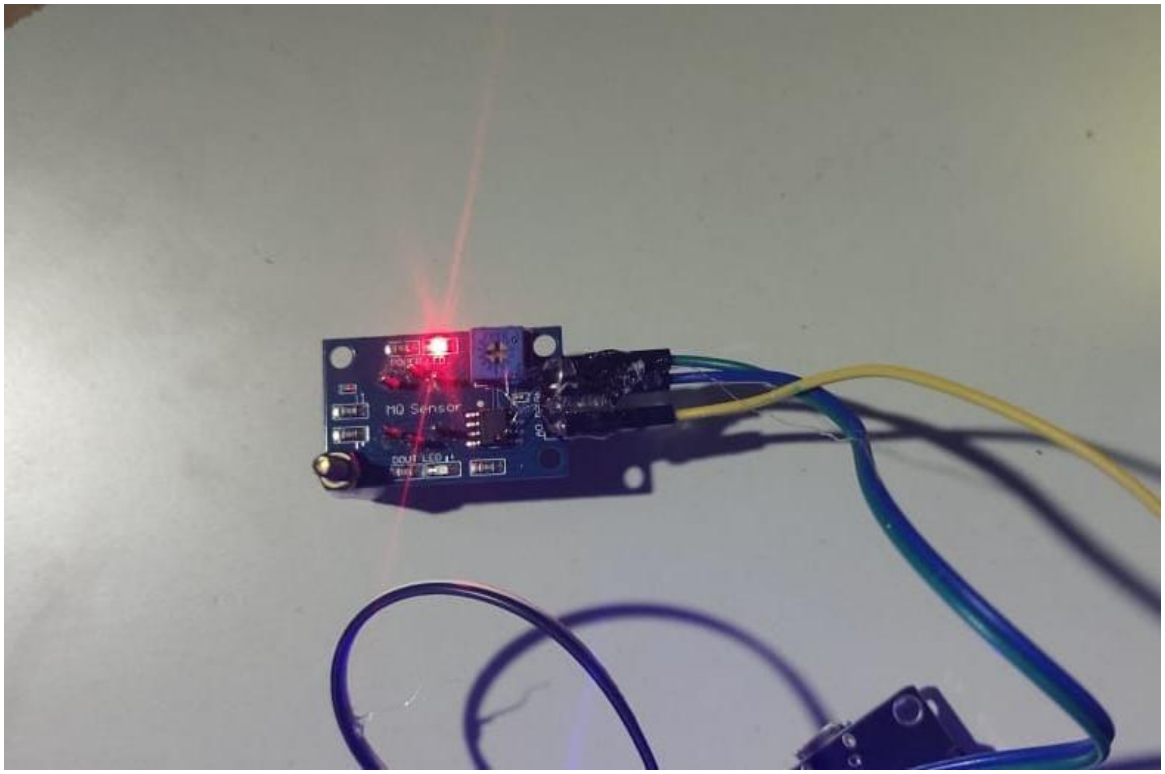


Figure 9.2.3 CO2 Sensor

Figure 9.2.4 LCD data

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

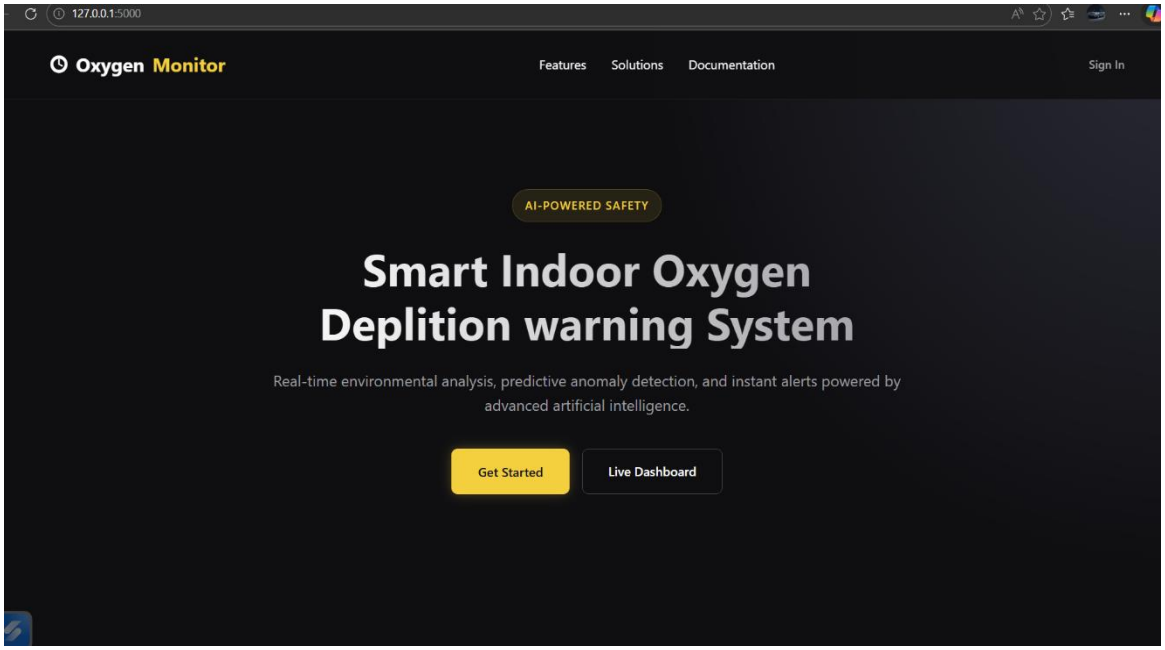


Figure 9.2.5 Website Dashboard

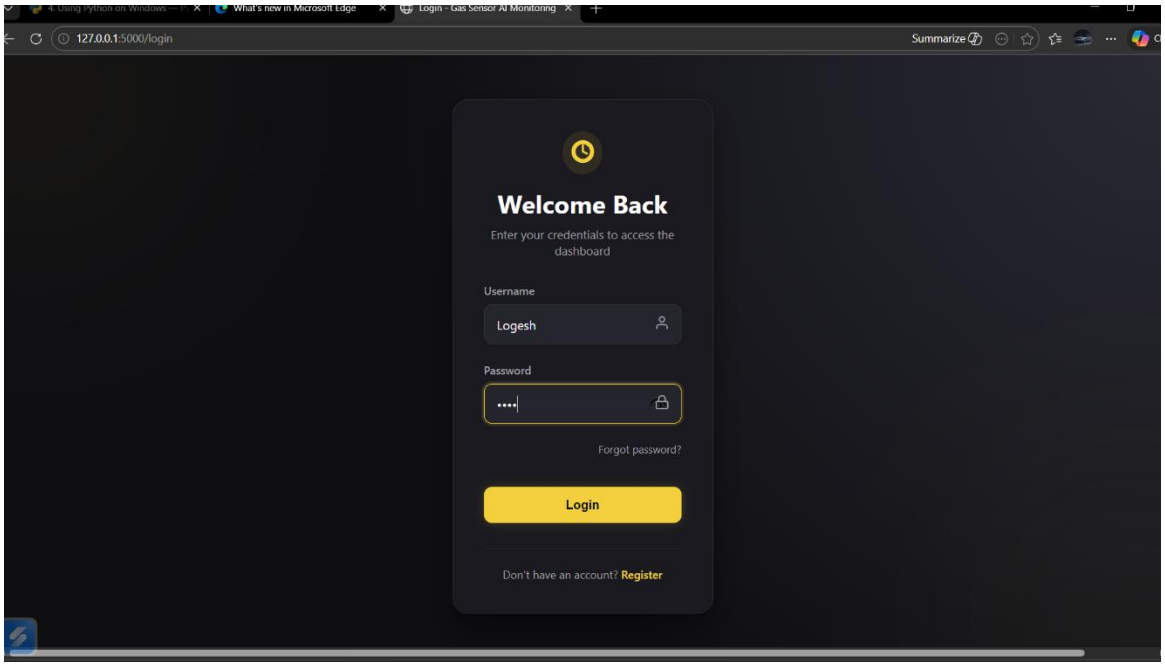


Figure 9.2.6 Login Page

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Figure 9.2.7 Live oxygen detection page

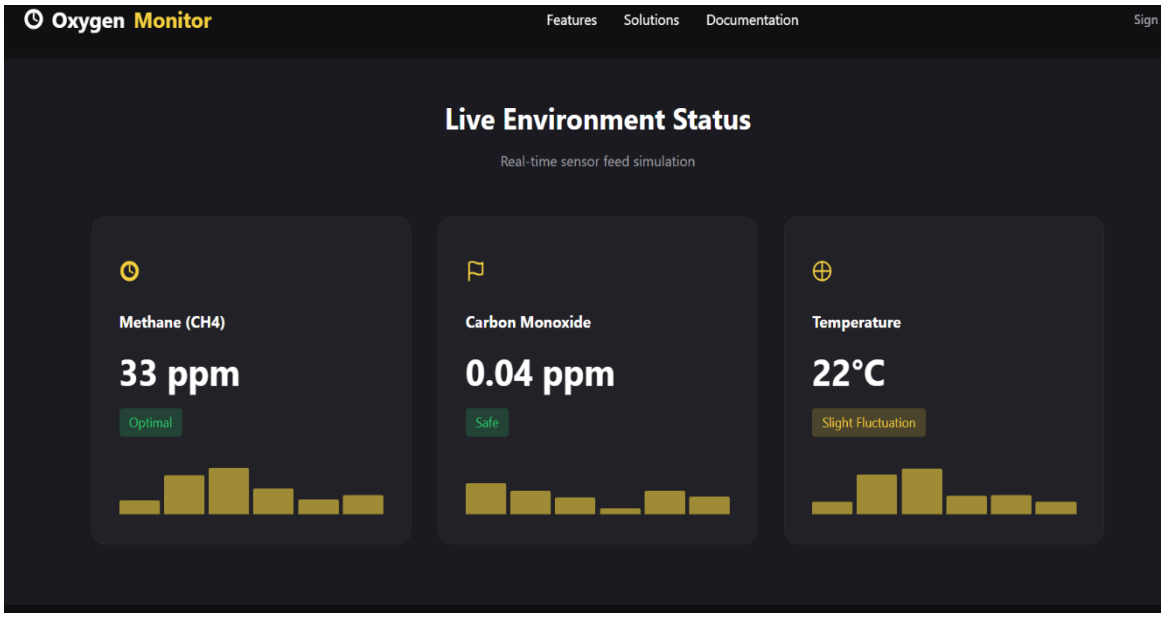


Figure 9.2.8 Live environment status

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

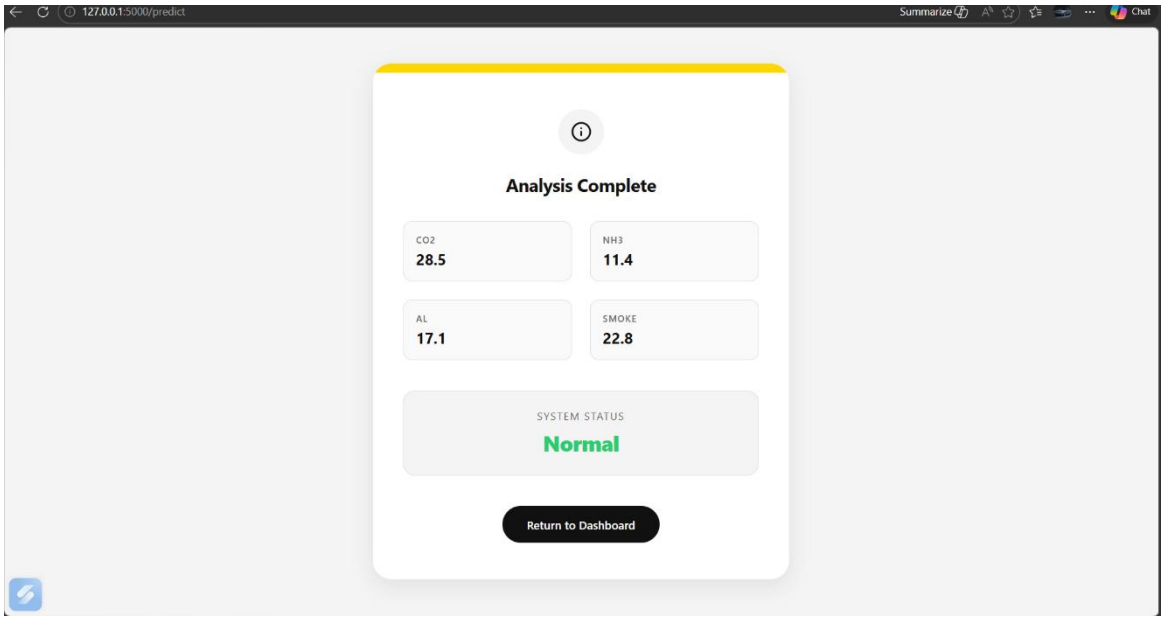


Figure 9.2.9 Live environment status

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

CHAPTER 10

REFERENCES

BOOK REFERENCES

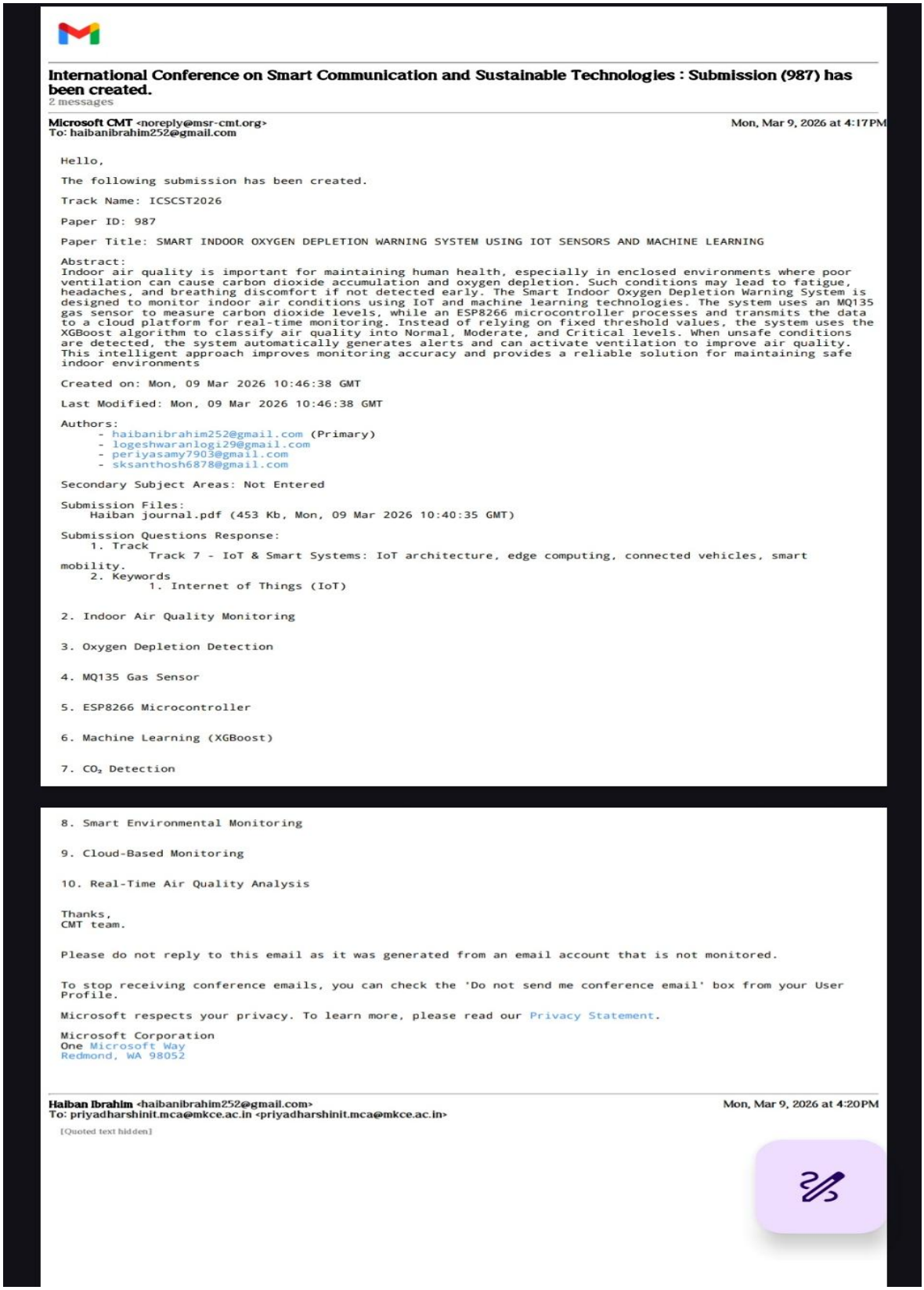
[1] Banzi, M., & Shiloh, M. Getting Started with Arduino. Maker Media, 2014.
[2] Monk, S. Programming Arduino: Getting Started with Sketches. McGraw-Hill Education, 2016.
[3] Rajkumar Buyya, Amir Vahid Dastjerdi. Internet of Things: Principles and Paradigms. Morgan Kaufmann, 2016.
[4] Bird, J. Electrical and Electronic Principles and Technology. Routledge, 2017.
[5] Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, 2019.
[6] VanderPlas, J. Python Data Science Handbook. O'Reilly Media, 2016.
[7] Zikopoulos, P., Eaton, C. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill, 2012.
[8] Karl Schwab. The Fourth Industrial Revolution. Crown Business, 2017.
[9] Hillar, G. C. Internet of Things with Python. Packt Publishing, 2016.
[10] Gaston C. Hillar. Machine Learning for the Internet of Things. Packt Publishing, 2017.

WEBSITE REFERENCES

[1] World Health Organization (WHO) – Air Quality and Health
https://www.who.int/health-topics/air-pollution
[2] United States Environmental Protection Agency (EPA) – Indoor Air Quality
https://www.epa.gov/indoor-air-quality-iaq
[3] Arduino Official Website – Microcontroller Documentation
https://www.arduino.cc
[4] Streamlit Official Documentation – Web Application Development
https://docs.streamlit.io
[5] Python Official Documentation
https://www.python.org
[6] Kaggle – Machine Learning Datasets and Resources
https://www.kaggle.com
[7] Scikit-Learn Documentation – Machine Learning in Python
https://scikit-learn.org
[8] ScienceDirect – Air Quality Monitoring Research Papers
https://www.sciencedirect.com
[9] SpringerLink – IoT and Environmental Monitoring Research
https://link.springer.com
[10] IEEE Xplore – Research on Smart Environmental Monitoring Systems
https://ieeexplore.ieee.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

JOURNAL SUBMISSION



International Conference on Smart Communication and Sustainable Technologies : Submission (987) has been created.
2 messages

Microsoft CMT <noreply@msr-cmt.org> Mon, Mar 9, 2026 at 4:17PM
To: haibanibrahim252@gmail.com

Hello,

The following submission has been created.

Track Name: ICSCST2026

Paper ID: 987

Paper Title: SMART INDOOR OXYGEN DEPLETION WARNING SYSTEM USING IOT SENSORS AND MACHINE LEARNING

Abstract:
Indoor air quality is important for maintaining human health, especially in enclosed environments where poor ventilation can cause carbon dioxide accumulation and oxygen depletion. Such conditions may lead to fatigue, headaches, and breathing discomfort if not detected early. The Smart Indoor Oxygen Depletion Warning System is designed to monitor indoor air conditions using IoT and machine learning technologies. The system uses an MQ135 gas sensor to measure carbon dioxide levels, while an ESP8266 microcontroller processes and transmits the data to a cloud platform for real-time monitoring. Instead of relying on fixed threshold values, the system uses the XGBoost algorithm to classify air quality into Normal, Moderate, and Critical levels. When unsafe conditions are detected, the system automatically generates alerts and can activate ventilation to improve air quality. This intelligent approach improves monitoring accuracy and provides a reliable solution for maintaining safe indoor environments

Created on: Mon, 09 Mar 2026 10:46:38 GMT

Last Modified: Mon, 09 Mar 2026 10:46:38 GMT

Authors:
- haibanibrahim252@gmail.com (Primary)
- logeshwaranlogi29@gmail.com
- periyasamy7903@gmail.com
- sksanthosh6878@gmail.com

Secondary Subject Areas: Not Entered

Submission Files:
Haiban journal.pdf (453 Kb, Mon, 09 Mar 2026 10:40:35 GMT)

Submission Questions Response:
1. Track
Track 7 - IoT & Smart Systems: IoT architecture, edge computing, connected vehicles, smart mobility.
2. Keywords
1. Internet of Things (IoT)

2. Indoor Air Quality Monitoring

3. Oxygen Depletion Detection

4. MQ135 Gas Sensor

5. ESP8266 Microcontroller

6. Machine Learning (XGBoost)

7. CO_2 Detection

8. Smart Environmental Monitoring

9. Cloud-Based Monitoring

10. Real-Time Air Quality Analysis

Thanks,
CMT team.

Please do not reply to this email as it was generated from an email account that is not monitored.

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our Privacy Statement.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Haiban Ibrahim <haibanibrahim252@gmail.com> Mon, Mar 9, 2026 at 4:20PM
To: priyadharshinit.mca@mkce.ac.in <priyadharshinit.mca@mkce.ac.in>

[Quoted text hidden]