ENVIRONMENT QUALITY MONITORING

Vedang Pratap Singh
Artificial Intelligence(Medical Engineering)
Amrita Viswa Vidyapeetham Coimbatore, India
cb.ai.u4aim24151@cb.students.amrita.edu

M. Sai Manaswini
Artificial Intelligence(Medical Engineering)
Coimbatore, India
cb.ai.u4aim24129@cb.students.amrita.edu

G. Ayyappa Harsha Vardhan
Artificial Intelligence(Medical Engineering)
Amrita Viswa Vidyapeetham Coimbatore, India
cb.ai.u4aim24155@cb.students.amrita.edu

O. Raghu Harsha Vardhan
Artificial Intelligence(Medical Engineering)
Amrita Viswa Vidyapeetham Coimbatore, India
cb.ai.u4aim24135@cb.students.amrita.edu

***Abstract*—This article presents a sophisticated hybrid air quality monitoring system developed using the ESP32 microcontroller, which is integrated without any issues with a real-time web dashboard and a Neural Network based analytical module. The system is designed to monitor principal environmental parameters around the clock, such as temperature, humidity, carbon dioxide (CO_2) levels, and ultraviolet (UV) radiation intensity levels. These readings are obtained with a combination of dependable and affordable sensors: the DHT11 for temperature and humidity, MQ135 for CO_2 sensing, and ML8511 for sensing UV intensity.**

The collected data is sent to a responsive web interface which allows for real-time visualization and monitoring of environmental conditions. In addition to mere data presentation, the system includes a CNN model which is trained on past sensor data to carry out intelligent analysis. The model can determine environmental anomalies, forecast pollution patterns, and classify air quality into given categories like Good, Moderate, and Hazardous.

With cost-effectiveness and scalability in its design, the suggested solution meets the urgent requirement for localized air quality monitoring within semi-urban areas like Ettimadai, Coimbatore. With the provision of affordable environmental information and public awareness, the system presents a major leap towards active environmental control and public well-being.

***Index Terms*—Air Quality, Neural Network, Temperature and humidity, UV intensity**

I. Introduction

Environmental quality is a key consideration that affects public health, stability of ecosystems, and general quality of life. Industrialization, rapid urbanization, and high emissions from vehicles have resulted in compromised air quality and climatic changes, creating a need for effective systems to monitor the environment. Conventional ways of environmental data gathering tend to be costly, location-based, and incapable of supporting predictive or real-time analytics. The latest developments in embedded systems and the Internet of Things (IoT) have enabled it to be possible to create smart, low-cost, and real-time environmental monitoring solutions. This project is centered on the design of a small, IoT-based environmental quality monitoring device that has three main sensors: the MQ135 gas sensor to detect pollutants such as CO_2, NH_3, and volatile organic compounds (VOCs); the DHT11 sensor to measure ambient temperature and humidity; and the ML8511 UV sensor to monitor ultraviolet radiation levels. These sensors are connected to an ESP32-WROOM-32 microcontroller, selected for its high-processing capabilities and built-in Wi-Fi connectivity, allowing real-time data transfer to cloud-based platforms like Blynk for visualization and remote access.

Along with real-time monitoring, the system has a predictive analytics module based on a feed-forward neural network. Upon training the model from time-series sensor data, the system is able to predict future environmental conditions, which is critical for early warning systems, data-driven urban planning, and personal health security. This combination of real-time sensing and machine learning-based prediction will help offer an integrated solution for environmental monitoring in smart cities, homes, and industries.

II. Literature Review

Several studies have demonstrated the potential of IoT in creating low-cost, real-time air quality monitoring systems. M. Deepshika et al. [1] developed a practical, real-time monitoring solution that collects data from various gas sensors to evaluate air quality conditions. Their system offers a simple yet effective approach to environmental monitoring using embedded systems.

1. Hassan et al. [2] introduced a robust IoT-based environment monitoring system that measures multiple environmental parameters including CO_2, temperature, and humidity. The system provides an efficient method for real-time air quality data collection, useful for both urban and indoor applications.

2. Al-Faris and Saputra Elsi [5] developed an IoT-based air quality monitoring system that is both portable and easy to use. Their system is designed to be accessible for everyday users, allowing communities to actively monitor air pollution in their surroundings. This approach not only supports local environmental assessment but also helps increase public awareness about the impact of poor air quality.

3. Jeffry Cerdan and Andrade-Arenas [6] designed an IoT-powered weather station for wetland environments, showcasing the adaptability of IoT technologies in diverse ecological

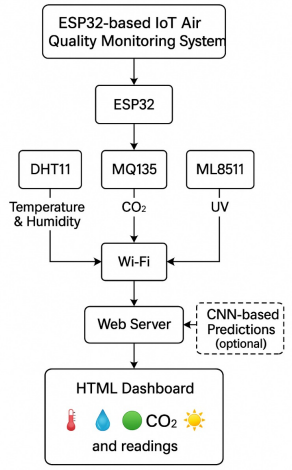

Fig. 1. Structure and Workflow

conditions. Their approach is useful for both weather and air quality monitoring in sensitive ecosystems.

4. Lakshmi Prabha and Vadivel [9] applied IoT technology to smart transportation by integrating air quality sensors into vehicles. Their system monitors pollution levels in real-time during travel, helping to manage in-transit air quality and protect passengers from harmful exposure.

5. AI is being used more and more to improve monitoring systems by helping them predict trends and recognize patterns.

6. Akbaba and Dişken [3] used feedforward neural networks to develop an indoor air quality detection system. Their work highlights how AI can identify patterns in pollutant levels and offer more accurate air quality assessments.

7. Jujjavarapu et al. [7] developed a forecasting model using deep neural networks to predict ambient air pollution. Their system showed promising results in predicting future pollutant concentrations, allowing authorities to take proactive measures.

8. Freeman et al. [8] also utilized deep learning to forecast air quality time series data. By comparing traditional and AI-based models, they demonstrated that deep learning methods significantly improve forecasting accuracy.

9. Alotaibi and Nassif [4] provided an in-depth analysis of AI's role in environmental monitoring. They explained how machine learning techniques are applied to analyze large environmental datasets, optimize the performance of monitoring systems, and enable real-time decision-making to manage pollution effectively.

III. METHODOLOGY

The system employs a hybrid solution of Internet of Things (IoT) hardware for real-time environmental monitoring and Neural Network model for smart data analysis. The process is separated into the following salient phases:

i functionality and processing capabilities. The sensors listed below were interfaced with the ESP32:

DHT11: To measure ambient temperature and humidity.

MQ135: To measure carbon dioxide (CO_2) and other volatile organic compounds (VOCs).

ML8511: To measure ultraviolet (UV) radiation intensity.

The sensors were interfaced using analog and digital GPIO pins, and the data was read periodically.

2. Firmware Development and Web Interface The ESP32 was implemented with the Arduino IDE. A local web server was implemented and run on the ESP32 itself, allowing real-time browser-based monitoring over Wi-Fi. The web interface provided current environmental readings with auto-refresh support and employed emoji-based symbols to indicate air quality status for increased user interaction.

3. NN-Based Analytical Model Three Neural Network models were trained on past environmental data gathered under different air quality conditions. The structure of the model was as follows:

Input Layer: Sensor values normalized.

Hidden Layers: dense layers for pattern extraction.

Output Layer: Multi-class classification of air quality levels — Safe, Moderate, or Dangerous.

The NN was implemented on a different server or local machine, which interacted with the ESP32 to give real-time anomaly detection and classification feedback.

4. Deployment and Testing The prototype was tested in semi-urban regions of Ettimadai, Coimbatore. The following environmental parameters were measured during testing:

CO_2 levels varied between 50 to 250 ppm.

UV radiation was measured between 25–35 mW/cm².

Temperature was around 31°C.

Humidity was around 55%.

The CNN model effectively identified air quality anomalies and enhanced classification accuracy compared to fixed threshold-based systems.

A. Problem statement

Motivation and Problem Statement:

- Limited public awareness: Many individuals are unaware of how daily environmental exposure affects their health. - Sparse coverage of monitoring stations: Traditional air monitoring units are limited in number and cannot cover remote or semi-urban regions like Ettimadai, Coimbatore. - Need for low-cost, portable monitoring: Most air quality devices are expensive or require high maintenance. - Rising pollution and UV exposure: Climate change and urbanization have increased the health risks associated with CO_2 buildup and UV radiation. - Education tool potential: This system can be used in academic environments to teach students about IoT, sensors, and environmental science.

B. System overview

- The suggested system brings together a hardware unit based on a microcontroller and a web-based dashboard, as well as an AI-backed analytical layer to track and analyze environmental conditions in real time. It is expected to be low-cost, user-friendly, and implementable in semi-urban settings like Ettimadai, Coimbatore.

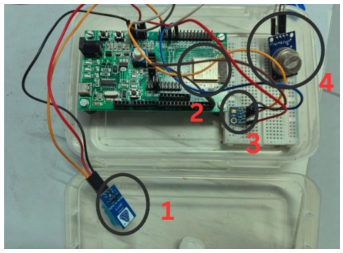

Fig. 2. Sensors connected with ESP32 microcontroller: 1) DHT11 temp and humidity sensor, 2) ESP32 microcontroller, 3) ML851 uv radiation sensor, 4) ML135 gas sensor

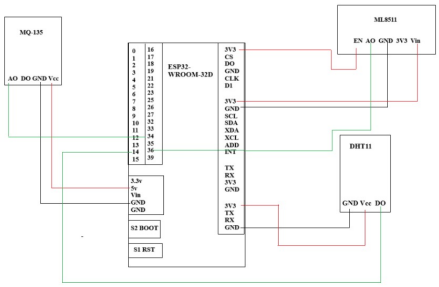

Fig. 3. Schemetic diagram of Interconnected sensor module

- The hardware setup revolves around the ESP32 microcontroller, selected due to its integrated Wi-Fi feature and low power consumption. The sensors chosen are inexpensive, simple to integrate, and suitable for monitoring important environmental parameters, such as:
  - ESP32: Wi-Fi microcontroller for central processing and data management. - DHT11: Temperature and humidity sensor for measuring ambient temperature and humidity. - MQ135: Gas and air quality sensor for estimating CO_2 and VOC concentration in the air. - ML8511: Ultraviolet (UV) sensor for measuring the intensity of UV radiation in mW/cm². -Web Server: Hosted on the ESP32, which trends real-time data through a browser-based dashboard.
- All sensors are wired to the ESP32 with GPIO pins and have their analog or digital output processed at set intervals.

C. Software Architecture

The system is coded in the Arduino IDE. Sensor data is gathered, processed, and served via a local web server running on the ESP32. The dashboard is viewable on any device on the same Wi-Fi network, such as smartphones, tablets, and PCs.

Major Features:

- **Automatic Refresh of Data:** The sensor data is updated every 5 seconds on the dashboard.
- **Responsive Design:** The dashboard is desktop- and mobile-friendly, without needing any other apps.
- **Data Normalization:** Sensor readings are preprocessed to make them compatible with the CNN for continuous real-time processing.

This modular design allows non-technical users and researchers alike to gain precise, real-time environmental insights in an easy-to-use manner.

D. Web server interface

Web Dashboard The web dashboard is the main user interface for displaying environmental data in real time. Its main features are:

Auto-Refresh Functionality: The dashboard refreshes automatically every 5 seconds so that the user can always view the latest sensor measurements without having to refresh the page manually.

Real-Time Data Display: It displays the main environmental parameters clearly and effectively, including:

Temperature: In degrees Celsius.

Humidity: Percentage of moisture in the air.

CO_2 Levels: Reports the level of carbon dioxide in parts per million (ppm).

Air Quality Status: Reports a qualitative status (e.g., Good, Moderate, Poor) based on sensor reading.

UV Radiation: Reports the strength of ultraviolet rays, helpful in determining sun exposure risk.

Responsive Design: The dashboard is accessible from any device (computer, tablet, or mobile phone) that shares the same Wi-Fi connection, which makes it easy for users to observe data on the move.

E. Reference table

TABLE I
SAFE AND DANGEROUS CO_2 LEVELS BASED ON ASHRAE & EPA GUIDELINES

CO (ppm)	Category	Health Impact
400–600	Ideal	Fresh outdoor air
600–1000	Acceptable	Slight fatigue possible
1000–2000	Poor	Drowsiness, discomfort
2000–5000	Unhealthy	Headaches, reduced concentration
>5000	Dangerous	Potential health hazard

TABLE II
UV INDEX CATEGORIES AND RECOMMENDED PROTECTIVE ACTIONS

UV Index	Category	Protective Actions
0–2	Low	Minimal risk
3–5	Moderate	Wear sunglasses
6–7	High	Use sunscreen, stay in shade
8–10	Very High	Protective clothing essential
>11	Extreme	Avoid sun exposure

IV. RESULTS

The ESP32-based Air Quality Monitoring System effectively integrates multiple environmental sensors and provides real-time data visualization through a web dashboard. The system employs the ESP32 microcontroller, leveraging its built-in Wi-Fi capability to collect environmental parameters

including temperature and humidity using the DHT11 sensor on GPIO14, CO_2 concentration using the MQ-135 gas sensor on GPIO32, and ultraviolet (UV) intensity via the ML8511 UV sensor on GPIO34. Upon successful connection to a local Wi-Fi network, the ESP32 hosts a web server that serves a dynamically updated HTML page, refreshing every five seconds. The dashboard displays live sensor readings using clear visual indicators such as emojis and labels for temperature (**thermometer**), humidity (**droplet**), CO_2 (**lungs**), Air Quality Index (AQI) (**chart-increasing**), and UV intensity (**sun**).

The system estimates CO_2 concentration using a regression-based calibration formula and categorizes the air quality index (AQI) into five qualitative levels: *Excellent*, *Good*, *Moderate*, *Poor*, and *Hazardous*. These are represented with intuitive indicators such as ✓, **!**, or **X** to denote the air quality status. In the event of sensor read errors, the dashboard provides immediate error feedback. Furthermore, the system incorporates automated Wi-Fi reconnection logic and reboots the device if connection failures persist, ensuring continuous operation.

Stylistically, the web interface is constructed using responsive HTML and CSS, enhancing readability and compatibility across multiple device types. This implementation demonstrates a practical and reliable Internet of Things (IoT) application for monitoring environmental air quality in real-time. Future enhancements may include cloud-based data storage, long-term trend visualization, mobile notifications, and integration with smart environmental control systems.

A. COMPARISON OF NN MODELS

In this project, we explored three types of neural networks—Feedforward Neural Network (FNN), Time-Delay Neural Network (TDNN), and Long Short-Term Memory (LSTM)—for the task of predicting future values of environmental parameters such as temperature, humidity, and UV index. The Feedforward Neural Network, being the simplest architecture, processes inputs independently without any sense of temporal context. While suitable for static data, it lacks the capability to model time-dependent patterns, making it less effective for time series forecasting. The TDNN, on the other hand, extends the feedforward design by incorporating delayed inputs, enabling it to capture short-term temporal dependencies. This makes it a lightweight yet efficient model for time series data when the influence of recent observations is significant. Finally, the LSTM network, a type of recurrent neural network (RNN), is specifically designed to learn both short- and long-term dependencies in sequential data through its memory cells and gating mechanisms. LSTMs are particularly powerful for capturing complex temporal dynamics, but they require larger datasets and more computational resources. In our study,TDNN offered a good balance between performance and complexity, making it suitable for moderate-length sequences.—while the LSTM consistently outperformed FNN and TDNN in terms of accuracy, particularly on longer datasets. but risked overfitting on smaller data.FNN, although simplest, was fastest to train and is adequate when temporal dependencies are minimal. Therefore, model selection in time series prediction should carefully balance complexity, memory requirements, and data availability.

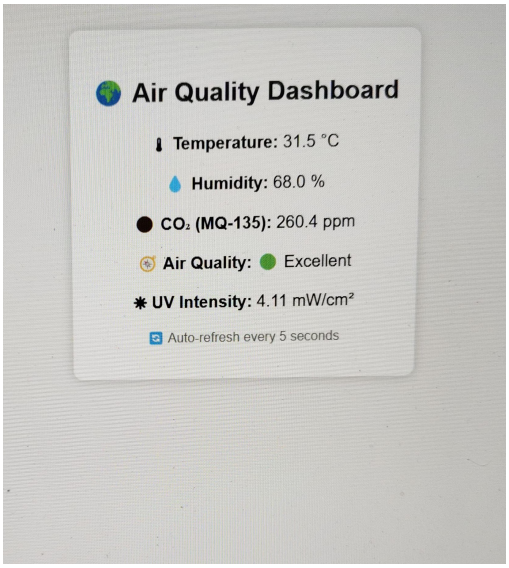

Fig. 4. Output in web-server

V. REFERENCES

REFERENCES

[1] M. Deepshika, M. Vikram, S. Ganesh and A. Deva Kalyan, "Real time air quality monitoring system, " 2021, [Online]. Available: https://ece.anits.edu.in/Project%20Reports%202021-22%20NAAC/Sec%20C/C16.pdf

[2] Mosfiqun Nahid Hassan, Farida Habib Semantha, Mohammed Rezwanul Islam, Abdul Hasib Siddique, Fahad Faisal and Mehedi Hasan , " An IoT based environment monitoring system" 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9316050

[3] Cihat Ediz Akbaba and Gökay Dişken, "Feedforward Neural Network-based indoor air quality detection system, " 2023. [Online].Available: https://www.researchgate.net/publication/380466076_Feedforward_Neural_Network-Based_Indoor_Air_Quality_Detection_System

[4] Emran Alotaibi and Nadia Nassif, "Artifcial intelligence in environmental monitoring: in-depth analysis" 2024. [Online]. Available: https://link.springer.com/article/10.1007/s44163-024-00198-1

[5] Muhammad Ghozi Al-Faris and Zulhipni Reno Saputra Elsi, "Air Quality Monitoring System Based Internet Of Things" 2024. [Online]. Available: https://jurnal.itscience.org/index.php/brilliance/article/download/4924/3749/22550

[6] Jeffry Ricaldi Cerdan and Laberiano Andrade-Arenas, "Design of a Weather Station with IoT in the Wetlands of the District of Ventanilla – Callao" 2022. [Online]. Available: https://www.academia.edu/download/94311527/ijetae0922_04.pdf

[7] Geethika Jujjavarapu, Siddhartha Duggirala, Anulekha Kavutarapu and Ravikishan Surapaneni, "Ambient Air Pollution Forecasting System using Deep Neural Networks" 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9316050

[8] Brian S. Freeman, Graham Taylor, Bahram Gharabaghi, and Jesse Thé, "Forecasting air quality time series using deep learning" 2018. [Online]. Available: https://www.researchgate.net/publication/320858715_Forecasting_Air_Quality_Time_Series_Using_Deep_Learning

[9] Lakshmi prabha S and R. Vadivel, "Sensorsync auto: Integrating IoT sensors for smart vehicle air quality monitoring". 2024 [Online]. Available:https://wjarr.co.in/wjarr-2024-0678

VI. APPENDIX

A. System Code

```
#include <WiFi.h>
#include <WebServer.h>
#include <DHT.h>
#include <MQUnifiedsensor.h>

const char* ssid = "Nee net vaduko mama";
const char* password = "chathan11";

#define DHTPIN 14
#define DHTTYPE DHT11
#define MQ135_PIN 34
#define ML8511_PIN 36

#define Board "ESP32"
#define Voltage_Resolution 3.3
#define ADC_Bit_Resolution 12
#define Type "MQ-135"

MQUnifiedsensor MQ135(Board, Voltage_Resolution,
    ADC_Bit_Resolution, MQ135_PIN, Type);

DHT dht(DHTPIN, DHTTYPE);
WebServer server(80);

void readDHT(float &temperature, float &humidity) {
  humidity = temperature = -1;
  for (int i = 0; i < 5; i++) {
    humidity = dht.readHumidity();
    temperature = dht.readTemperature();
    if (!isnan(humidity) && !isnan(temperature))
        break;
    delay(1000);
  }
}

float readUV() {
  int uvRaw = analogRead(ML8511_PIN);
  float voltage = uvRaw * (3.3 / 4095.0);
  float intensity = (voltage - 1.0) * 15.0;
  return (intensity > 0) ? intensity : 0;
}

String getAQILevel(float co2ppm) {
  if (co2ppm <= 400) return "       Excellent";
  else if (co2ppm <= 800) return "    Good";
  else if (co2ppm <= 1200) return "       Moderate";
  else if (co2ppm <= 2000) return "       Poor";
  else return "       Hazardous";
}

String getSensorData() {
  float temp, hum;
  readDHT(temp, hum);
  MQ135.update();
  float co2ppm = MQ135.readSensor();
  float uv = readUV();
  String aqi = getAQILevel(co2ppm);

  String html = "<!DOCTYPE html><html><head><meta
      charset='UTF-8'><title>ESP32 Air Quality
      Monitor</title>";
  html += "<meta http-equiv='refresh' content='5'>";
  html += "<style>body{font-family:Arial;background
      :#f4f4f4;padding:20px;text-align:center;}";
  html += "div{background:#fff;border-radius:10px;
      padding:20px;box-shadow:0 0 10px #ccc;display:
      inline-block;}</style></head><body>";
  html += "<div><h2>       Air Quality Dashboard</h2
      >";
  html += "<p><strong>       Temperature:</strong> "
       + (temp != -1 ? String(temp, 1) + "  C " : "
          Error") + "</p>";
  html += "<p><strong>       Humidity:</strong> " +
      (hum != -1 ? String(hum, 1) + " %" : "
      Error") + "</p>";
  html += "<p><strong>        C O   (MQ-135):</strong
      > " + (co2ppm > 0 ? String(co2ppm, 1) + " ppm"
       : "    Error") + "</p>";
  html += "<p><strong>       Air Quality:</strong> "
       + aqi + "</p>";
  html += "<p><strong>    UV Intensity:</strong> " +
       String(uv, 2) + " mW/cm </p>";
  html += "<p style='color:gray;font-size:0.8em;'>
            Auto-refresh every 5 seconds</p></div
      ></body></html>";

  return html;
}

void connectToWiFi() {
  Serial.print("Connecting to WiFi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  int retry = 0;
  while (WiFi.status() != WL_CONNECTED && retry <
      20) {
    delay(500);
    Serial.print(".");
    retry++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nWiFi connected!");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nFailed to connect to WiFi!
        Rebooting...");
    delay(3000);
    ESP.restart();
  }
}

void setup() {
  Serial.begin(115200);
  dht.begin();
  connectToWiFi();

  MQ135.init();
  MQ135.setRegressionMethod(1);
  MQ135.setA(116.6020682);
  MQ135.setB(-2.769034857);
  MQ135.setRL(10.0);
  MQ135.setR0(10.0);

  Serial.println("Calibrating MQ-135...");
  MQ135.update();
  float r0 = MQ135.calibrate(9.83);
  if (isnan(r0) || r0 <= 0) {
    Serial.println("Calibration failed. Using
        default R0.");
    MQ135.setR0(10.0);
  } else {
    MQ135.setR0(r0);
    Serial.print("Calibrated R0 = ");
    Serial.println(r0, 2);
  }

  server.on("/", []() {
    server.send(200, "text/html", getSensorData());
  });
  server.begin();
  Serial.println("Web server started.");
}
```

```
void loop() {
  server.handleClient();
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi lost. Reconnecting...");
    connectToWiFi();
  }
}
```

Listing 1. Air Quality Monitor Code for ESP32