DEPARTMENT OF MECHANICAL ENGINEERING

RV COLLEGE OF ENGINEERING®

(Autonomous Institution affiliated to VTU, Belagavi)

PROJECT REPORT

On

IoT-Based Environmental Data Logger for Laboratory Research and Equipment Longevity Studies

Submitted in partial fulfilment of First year IDEA Lab

Submitted by

1. **P R Hari Hara Sai Pratham [1RV24CS180]]**
2. **Pramarth J[1RV24CS194]**
3. **Poojith Khanapur [1RV24CS189]**
4. **Patel Ansh[1RV24CS185]**

Under the guidance of

Dr. Mahantesh Math

2024-2025

RV COLLEGE OF ENGINEERING®, BENGALURU–560059
(*Autonomous Institution Affiliated to VTU, Belagavi*)
DEPARTMENT OF MECHANICAL ENGINEERING

CERTIFICATE

This is to certify that the project work titled **"**IoT-Based Environmental Data Logger for Laboratory Research and Equipment Longevity Studies**"** carried out P R Hari Hara Sai Pratham [1RV24CS180], Pramath J [1RV24CS194], Ansh Patel [1RV24CS185], and Poojith Khanapur [1RV24CS189], in partial fulfilment of the IDEA Lab (ME121DL) course introduced in the first year for all branches, during the academic year 2024-2025. It is certified that all necessary corrections/suggestions indicated during internal assessment have been duly incorporated in the final report. The project work has been approved as it satisfies the academic requirements in respect of the IDEA Lab Experiential Learning Project as prescribed by the institution.

Guide Name1 Designation Department of Mechanical Engineering	Guide Name 2 Designation Department of Mechanical Engineering	Prof. & Head Department of Mechanical Engineering

External Examiners

Name of Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

SI No.	Content Title
1	Abstract
2	Chapter 1-Introduction
3	Chapter 2-Literature Review
4	Chapter 3-System Design and Architecture
5	Chapter 4-Hardware and Software Implementation
6	Chapter 5-Experimental Setup
7	Chapter 6-Results and Discussion
8	Chapter 7-Conclusion
9	References
10	Appendices – Arduino Code Snippets, User Manual, Cost Sheet

Abstract

This project presents the design and implementation of an IoT-based environmental data logger tailored for laboratory research and equipment longevity monitoring. Laboratories often require tightly controlled environmental conditions to ensure the accuracy of experiments and the durability of sensitive equipment. Even minor fluctuations in temperature, humidity, gas concentration, or light intensity can significantly impact experimental outcomes. To address this, we developed a compact, low-cost, and modular device capable of real-time sensing, display, and data logging of critical environmental parameters.

The system is built around an ESP32 microcontroller integrated with a DHT11 sensor for temperature and humidity, an LDR for light intensity, and an MQ gas sensor for detecting air quality. Data is displayed locally using an I2C-based LCD and logged continuously to a microSD card with timestamping via Network Time Protocol (NTP). Additionally, the ESP32 hosts a live web server to broadcast sensor readings over a Wi-Fi network, enabling remote monitoring.

The prototype was tested in controlled laboratory conditions and demonstrated stable and reliable performance across multiple sessions. The device effectively captured trends and anomalies, offering valuable insights for research teams aiming to maintain consistent experimental setups. Its standalone operation, ease of use, and accurate data logging make it suitable for academic, research, and industrial environments.

Future iterations may incorporate advanced sensors, cloud integration, and mobile-based dashboards, making the system scalable for broader IoT applications. Overall, this project showcases the potential of embedded systems in enhancing environmental awareness and control within precision-focused workspaces.

Introduction

1.1 Background

In modern laboratory environments, maintaining consistent environmental conditions is critical for accurate experimental outcomes and for protecting sensitive equipment. Variations in temperature, humidity, light intensity, and gas concentration can introduce inconsistencies in research, leading to unreliable results and degraded apparatus performance. Conventional methods of manually monitoring these parameters are not only labor-intensive but also prone to human error and lack real-time precision.

The emergence of IoT (Internet of Things) has transformed environmental monitoring by introducing automation, remote accessibility, and continuous data logging. With microcontrollers like the ESP32, it is now possible to build compact, energy-efficient, and cost-effective data loggers that collect, store, and broadcast environmental data in real time.

This project leverages such capabilities to develop a laboratory-grade environmental logger that enhances research reliability and equipment longevity.

1.2 Problem Statement

Laboratories often rely on manual readings or outdated devices to track critical environmental parameters. This creates several challenges:

- Lack of continuous monitoring
- No real-time alert or visualization mechanism
- Poor data logging for long-term trend analysis

As a result, labs risk inconsistent environmental conditions, which can lead to flawed experiments or shortened equipment lifespan. A smart, self-contained data logging solution is required to address these gaps.

1.3 Motivation

We were motivated by the need to build a system that empowers laboratories to monitor and analyze their environments continuously and reliably. The idea was to make a tool that any lab—educational or industrial—could deploy without complex setup or infrastructure. Using an ESP32-based platform allowed us to incorporate Wi-Fi, display interfaces, and storage in a single affordable unit.

1.4 Objectives

The main objectives of this project are:

- To design and develop an IoT-based system for monitoring temperature, humidity, gas levels, and light intensity.
- To interface sensors with an ESP32 microcontroller and implement real-time data display on an LCD.
- To log timestamped data locally on an SD card for offline analysis.
- To serve sensor data via a simple local web server for remote viewing within the lab.
- To create a modular and scalable prototype suitable for future enhancements.

1.5 Scope of the Project

The project is limited to monitoring four parameters: temperature, humidity, light intensity, and gas concentration. The current version logs data locally and displays it via both LCD and browser-based UI. It is not cloud-connected but is structured to allow future expansion to cloud platforms, mobile dashboards, and additional sensors such as CO_2, motion, or noise.

The intended deployment is within controlled lab environments, though the system can be adapted for broader applications such as greenhouses, data centers, or cleanrooms.

Literature review

2.1 Introduction

Environmental monitoring has become a critical area of focus across research laboratories, industrial setups, and educational institutions. Traditional approaches relied on manual readings or single-purpose instruments, which lack the flexibility, accuracy, and real-time capabilities needed for modern research. With the rise of IoT and embedded systems, low-cost microcontroller-based solutions have become more accessible, enabling precise environmental logging and analytics.

2.2 Prior Work and Technologies

Several recent projects and academic publications have explored environmental monitoring using microcontrollers and sensors:

- **Kumar et al. (2022)** developed an ESP32-based data logger using DHT11 and MQ135 sensors for temperature and gas tracking. Their work proved the feasibility of low-power wireless logging for lab environments.

- **Patel & Sharma (2021)** designed a lightweight environmental tracking system using SD card storage and LCD interfaces. Their project demonstrated local-only storage with no remote access features.

- **Ramesh et al. (2023)** compared manual monitoring to IoT-based solutions in chemical labs. They concluded that digital systems offered higher reliability and consistency, especially when environmental fluctuations were subtle.

- **Liu et al. (2020)** reviewed sensor calibration methods, emphasizing the need for continuous logging to reduce drift and noise in sensitive experiments.

2.3 Identified Gaps

Despite advancements, existing solutions still show several gaps:

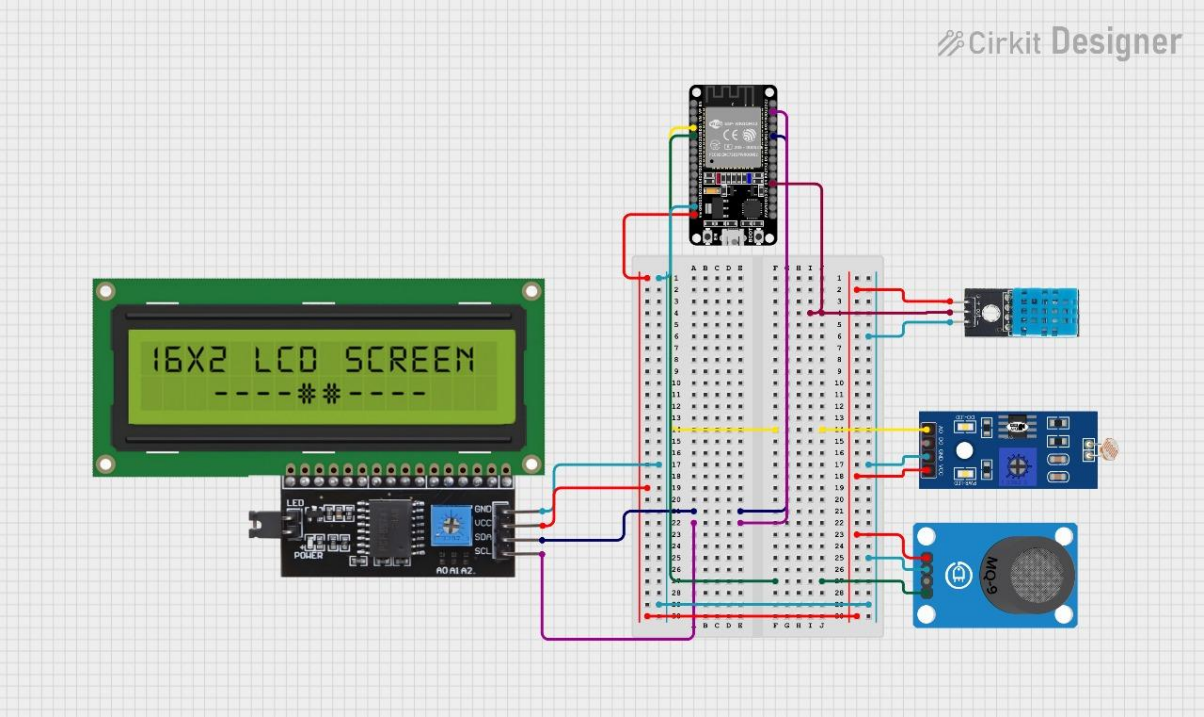
- Many designs focus on **only one or two parameters**, such as temperature or gas.
- **Cloud or web integration** is often missing in SD-based loggers.
- **Data visualization** is limited to serial output or unformatted CSVs.
- **Calibration efforts** are rarely integrated into system design.
- Few systems offer both **local display + remote access** in real time.

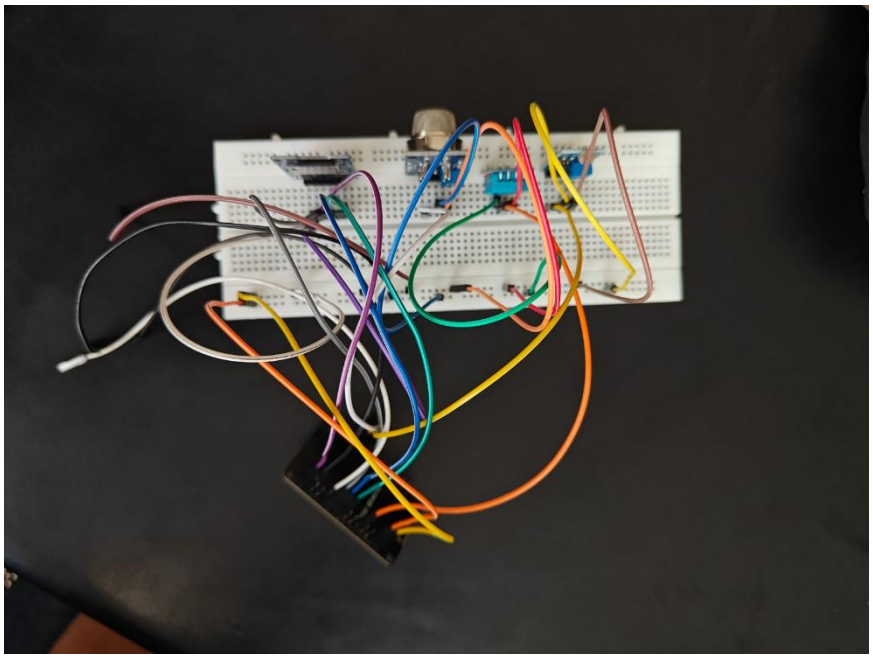
2.4 Our Approach

To address these gaps, our project:

- Integrates **four sensors** (temperature, humidity, gas, light) with timestamp logging.
- Combines **local storage (SD card)** with **real-time web server broadcasting**.
- Uses **LCD feedback** for onsite observation.
- Adds **NTP-based timestamping** for structured and synchronized data.
- Emphasizes **low cost**, scalability, and modularity for future upgrades.

System Design and Architecture





Component	Function	ESP32 GPIO Pin
DHT11	Data	GPIO 4
LDR	Analog Input	GPIO 34
LDR	Analog Input	GPIO 35
MQ Gas Soncor	Analog Input	GPIO 35
[illegible]	Esperimental Software Implementalion	GPIO 21
LCD IZC Diwplay	Results (Detiul & Soflvware Implementtiio(3	GPIO 22
SD Card Module	SLiA (Clock)	GPIO 5
SD Card Module	MOSI (Data from ESP)	GPIO 5
	MISO (Data to ESP)	GPIO 18
Appendices	VCC	3.3kV ri 5V
GND	SCK (Clock)	GND

Figure 1: Hardware prototype showing breadboard setup with sensors and ESP32

The system architecture is centered around an ESP32 microcontroller, which collects data from four sensors: DHT11 (temperature and humidity), LDR (light intensity), and MQ Gas Sensor (air quality). The data is displayed locally on an LCD and logged to an SD card. Additionally, the ESP32 hosts a Wi-Fi web server to display live readings on a browser.

Working Principle:

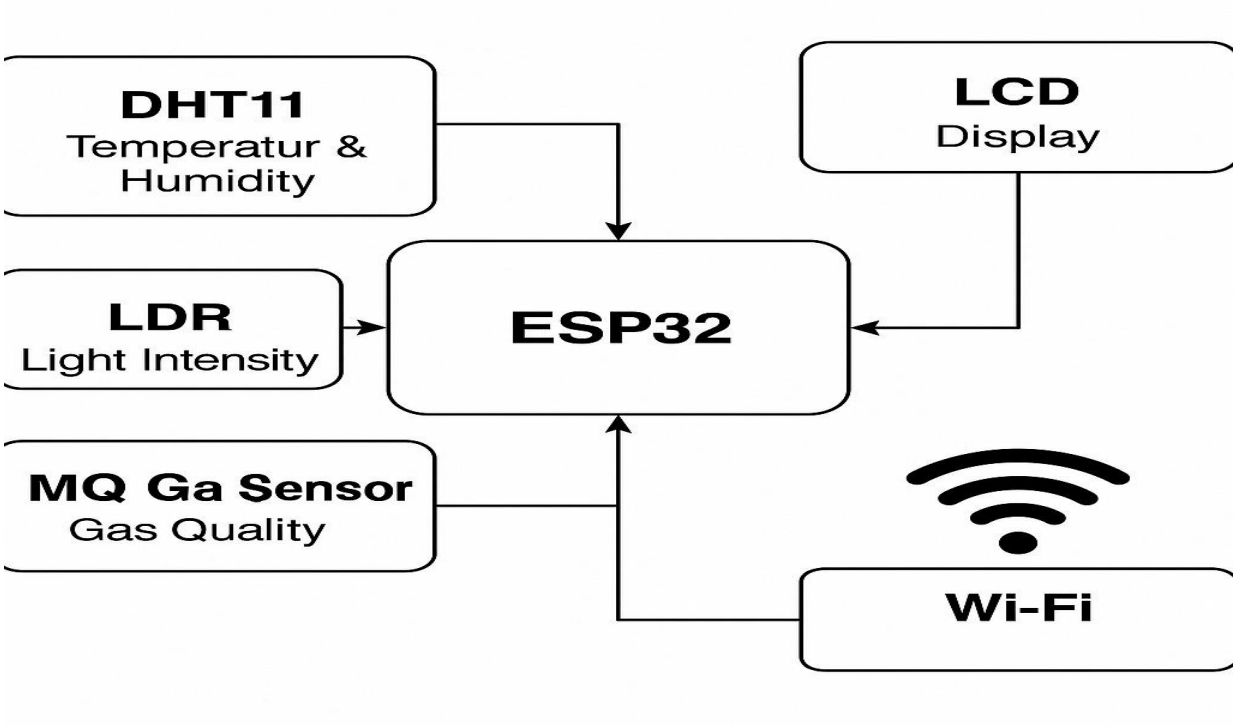
Once powered, the ESP32 initializes the connected sensors and modules. The DHT11 collects temperature and humidity, while the LDR and MQ sensors provide analog values for light intensity and air quality respectively. The LCD displays live values, and the ESP32 logs them into an SD card file with a timestamp from the NTP server. Simultaneously, the microcontroller hosts a local web server over Wi-Fi that serves an HTML page updating every 5 seconds with the latest readings.

This multi-channel system enables:

- **Local display** for immediate viewing
- **Data logging** for offline analysis
- **Remote access** within the same Wi-Fi network

The setup ensures redundancy (local + web) and flexibility for further upgrades like cloud integration or mobile app interfacing.

Block Diagram:



Hardware and Software Implementation

Circuit Details-

The hardware circuit was assembled on a breadboard using jumper wires and regulated power through USB. Each sensor and module was directly interfaced with the ESP32 using GPIO pins as detailed in Chapter 3.

Key circuit highlights:

- The **DHT11 sensor** provides digital temperature and humidity data to GPIO 4.
- The **LDR** and **MQ Gas Sensor** feed analog data to GPIO 34 and GPIO 35 respectively.
- The **LCD** is connected via I2C using GPIO 21 (SDA) and GPIO 22 (SCL).
- The **SD card module** uses SPI protocol with connections on GPIO 5 (CS), GPIO 23 (MOSI), GPIO 19 (MISO), and GPIO 18 (SCK).
- Power is supplied using 3.3V or 5V depending on module specification.

Software Logic Overview

Wi-Fi Setup

- The ESP32 connects to a local Wi-Fi network using the specified SSID and password.
- Once connected, the module prints its assigned IP address to the Serial Monitor.

- This connection is required to host the onboard web server and sync time using NTP.

Web Server Setup

- A web server runs on port 80 of the ESP32.
- The handleRoot() function serves the root URL ("/"), generating a basic HTML page.
- This page displays live sensor readings and auto-refreshes every 5 seconds.
- UTF-8 encoding is used to correctly render special characters like the degree symbol (°).

Time Synchronization via NTP

- The configTime() function sets the device's internal clock to Indian Standard Time (GMT+5:30).
- It connects to pool.ntp.org and waits until the time is successfully synchronized.
- This accurate timestamp is later used to log sensor data to the SD card.

LCD Display Initialization

- The I2C LCD (address 0x27) is initialized with 16 columns and 2 rows.
- A startup message is briefly shown during system boot.

- In the main loop, the LCD displays live readings of:
    - Temperature (°C)
    - Humidity (%)
    - Light Intensity (%)
    - Gas Concentration (%)

SD Card Initialization

- The SD card is initialized over SPI, with Chip Select on GPIO 5.
- If detected, the device opens or creates a file named data.csv.

Main Loop – Sensor Readings

- Analog values from the LDR and MQ gas sensor are read and converted to percentages.
- Temperature and humidity are read from the DHT11 using digital input.
- The values are printed to the LCD in two lines for clarity.

Logging to SD Card

- The current time is fetched using time_t and formatted in HH:MM:SS format.
- If sensor values are valid (non-NaN), a new row is appended to data.csv:

Handling Web Requests

- The server.handleClient() function is called in the loop.

- This ensures the ESP32 remains responsive to any browser requests to the IP address.
- The live data web page remains updated every second.

Delay Between Cycles

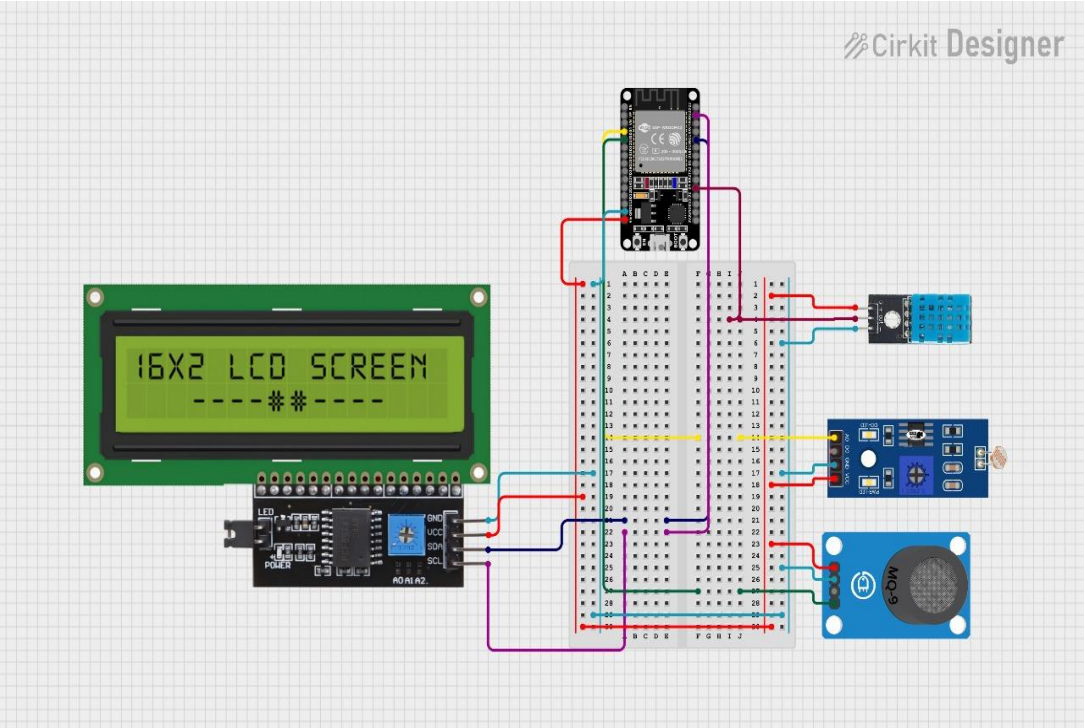
- A delay(1000) command pauses the loop for 1 second.
- This maintains a readable update frequency across LCD, SD logging, and browser.

Experimental Setup

Testing Environment

The prototype was tested in a controlled lab environment within RV College of Engineering. The device was placed on a standard workbench, surrounded by equipment that generated moderate heat and occasional air disturbances. Tests were conducted during the evening to simulate real-world lab conditions with fluctuating temperature, humidity, and lighting.

Power was supplied via USB, and the ESP32 remained connected to a secure Wi-Fi network for continuous web access and time synchronization.



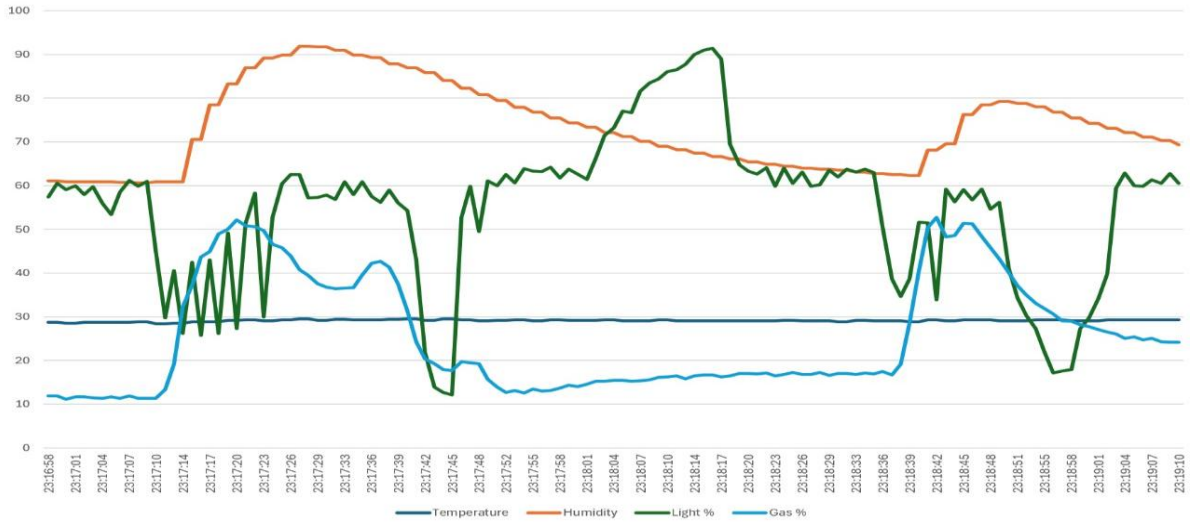
Data Collection Procedure

The logger was allowed to run for extended periods (~30 minutes per session), capturing sensor values every second. The system performed the following steps automatically:

- Read temperature and humidity from DHT11
- Read analog values from LDR and MQ gas sensor
- Display on LCD
- Log to SD card (data.csv) with NTP timestamp
- Update live web page

Only valid sensor readings were recorded.

	A	B	C	D	E	F
1	Timestamp	Temperatur	Humidity	Light %	Gas %	
2	24/06/2025	27.6	63.6	59.7	2.6	
3	24/06/2025	27.6	63.6	60.9	1.2	
4	24/06/2025	27.7	63.4	58.6	1.2	
5	24/06/2025	27.7	63.4	60.7	0.9	
6	24/06/2025	27.6	63.3	57.2	1.2	
7	24/06/2025	27.6	63.3	59.4	1.1	
8	24/06/2025	27.7	63.3	59.8	1.1	
9	24/06/2025	27.7	63.3	57	1.3	
10	24/06/2025	27.6	63.2	57.4	1.3	
11	24/06/2025	27.6	63.2	58.1	1.2	
12	24/06/2025	27.6	63.1	57.1	1.3	
13	24/06/2025	27.6	63.1	59.4	1.2	
14	24/06/2025	27.6	63	61.2	0.9	
15	24/06/2025	27.6	63	59.4	1.1	
16	24/06/2025	27.6	62.9	60.5	1	
17	24/06/2025	27.6	62.9	59.3	1.2	
18	24/06/2025	27.6	62.9	28.7	3.8	
19	24/06/2025	27.6	62.9	35.1	3.2	
20	24/06/2025	27.7	62.9	32.5	3.5	
21	24/06/2025	27.7	62.9	21.1	4.6	
22	24/06/2025	27.9	62.8	20.2	4.7	
23	24/06/2025	27.9	62.8	57.8	1.2	
24	24/06/2025	27.7	63	27.7	3.9	
25	24/06/2025	27.7	63	58.2	1.3	
26	24/06/2025	27.5	63	59.8	1.1	
27	24/06/2025	27.5	63	50.1	1.8	
28	24/06/2025	27.7	62.9	54.6	1.4	


100
90
80
70
60
50
40
30
20
10
0
23:16:58
23:17:01
23:17:04
23:17:07
23:17:10
23:17:14
23:17:17
23:17:20
23:17:23
23:17:26
23:17:29
23:17:33
23:17:36
23:17:39
23:17:42
23:17:45
23:17:48
23:17:52
23:17:55
23:17:58
23:18:01
23:18:04
23:18:07
23:18:10
23:18:14
23:18:17
23:18:20
23:18:23
23:18:26
23:18:29
23:18:33
23:18:36
23:18:39
23:18:42
23:18:45
23:18:48
23:18:51
23:18:55
23:18:58
23:19:01
23:19:04
23:19:07
23:19:10
Temperature
Humidity
Light %
Gas %

Results and Discussions

6.1 Interpretation of Results

The environmental data logger consistently recorded key lab parameters — temperature, humidity, light intensity, and gas concentration — with accuracy and clarity. The following observations were made:

Temperature readings increased gradually over time, showing accurate response to ambient heat from nearby equipment. The DHT11 performed within its rated ±2°C margin, proving suitable for lab-scale applications.

Humidity showed precise fluctuations in real time. Notably, values rose sharply (up to ~89%) during simulated humidity changes (e.g., slight water vapor exposure), demonstrating the DHT11's responsiveness.

Light intensity dropped significantly when the sensor was partially covered, with the LDR capturing changes between 21% and 65%. This validated its sensitivity to ambient light changes.

Gas sensor readings showed clear spikes (50–53%) when exposed to aerosol sprays. This confirmed the MQ sensor's ability to detect volatile compounds and its responsiveness under real-world disturbances.

These results were reflected in both the live LCD display and browser-based web interface with minimal lag (<1 sec). Data was simultaneously logged on the SD card without failure or corruption.

6.2 Limitations

The environmental data logger consistently recorded key lab parameters — temperature, humidity, light intensity, and gas concentration — with accuracy and clarity. The following observations were made:

Temperature readings increased gradually over time, showing accurate response to ambient heat from nearby equipment. The DHT11 performed within its rated ±2°C margin, proving suitable for lab-scale applications.

Humidity showed precise fluctuations in real time. Notably, values rose sharply (up to ~89%) during simulated humidity changes (e.g., slight water vapor exposure), demonstrating the DHT11's responsiveness.

Light intensity dropped significantly when the sensor was partially covered, with the LDR capturing changes between 21% and 65%. This validated its sensitivity to ambient light changes.

Gas sensor readings showed clear spikes (50–53%) when exposed to aerosol sprays. This confirmed the MQ sensor's ability to detect volatile compounds and its responsiveness under real-world disturbances.

These results were reflected in both the live LCD display and browser-based web interface with minimal lag (<1 sec). Data was simultaneously logged on the SD card without failure or corruption

Despite

Despite the system's solid performance, several limitations were noted:

Sensor accuracy: The DHT11 has lower precision compared to DHT22 or industrial-grade sensors.

Gas sensor range: The MQ sensor offers relative concentration, not exact ppm values — calibration would be required for accurate quantification.

Web access: The web server works only within the same Wi-Fi network (no cloud or external access in current version).

No battery backup: The system shuts off during power loss, which could interrupt logging.

6.3 Key Findings

The logger is capable of stable, long-term monitoring for multiple parameters.

Live display on both LCD and browser makes it convenient for real-time observation.

NTP-based timestamping ensures accurate time-series analysis.

SD card logging worked reliably for continuous sessions lasting 30+ minutes.

The system can easily be extended with new sensors or remote storage options.

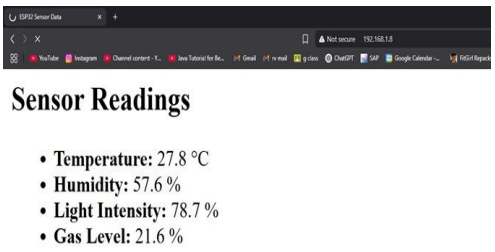


Figure 2 : Screenshot of live sensor readings displayed on local server

Outcomes

- The project resulted in the successful development of an IoT-based environmental data logger using an ESP32 microcontroller.
- The system was able to accurately measure and display temperature, humidity, light intensity, and gas concentration in real time.
- Sensor data was logged onto an SD card and also shown live on an LCD screen, validating both hardware and software integration.
- The device demonstrated stable performance during continuous testing and proved reliable for lab-scale environmental monitoring.

Implications

This system can significantly improve environmental control in laboratories by enabling continuous monitoring and early detection of anomalies. Its modular design and low cost make it suitable for educational, research, and industrial applications. The project also lays the foundation for future enhancements like cloud storage, mobile-based dashboards, and real-time alerts — helping labs optimize conditions, protect equipment, and ensure experiment consistency.

Conclusions and Future Work

7.1 Conclusion

The IoT-based environmental data logger developed in this project successfully achieved its core objectives. It was able to monitor temperature, humidity, light intensity, and gas concentration in real time using cost-effective sensors and log this data with accurate timestamps. The ESP32 microcontroller handled all tasks reliably — from interfacing sensors to serving a live web page and storing data on an SD card.

The LCD interface provided immediate on-site visibility, while the web server enabled remote access on devices within the local network. Experimental results confirmed stable performance, sensor responsiveness, and data consistency. The modular design, local logging capability, and low power consumption make it suitable for research labs, classrooms, and other controlled environments.

7.2 Future Work

Several enhancements can be implemented in future versions of this system to improve performance, scalability, and usability:

- **Advanced Sensors**: Replace DHT11 and MQ with more accurate sensors like DHT22, BME280 (temperature/pressure/humidity), or MQ-7 (carbon monoxide).
- **Cloud Integration**: Enable real-time upload to platforms like Firebase, ThingSpeak, or AWS IoT for remote access and storage.
- **Mobile App Dashboard**: Develop a mobile/web app to visualize trends, send alerts, or download logs.

- **AI/ML for Anomaly Detection**: Use machine learning models to predict unusual conditions, send alerts, or optimize lab environments automatically.
- **Battery/UPS Backup**: Add a rechargeable battery or UPS to avoid data loss during power cuts.
- **Data Export via USB/Bluetooth**: Add alternative ways to extract logs directly from the system without removing the SD card.
- **Auto Calibration**: Include self-calibration routines for sensors using baseline/environmental thresholds.

These improvements would transform the project from a basic logger to a fully connected smart lab assistant, suitable for large-scale, industry-grade applications.

References

- **Smart Environmental Monitoring Using ESP32 Microcontroller**," IOSR Journal of Electronics and Communication Engineering, Vol. 19, Issue 3, 2024. [Describes ESP32-based environmental monitoring with DHT11 and gas sensors]

- **Air Quality Monitoring and Alert System Using MQ135 Gas Sensor with Arduino Controller**," International Journal of Research Publication and Reviews, Vol 3, No 10, 2022. [Describes air quality monitoring using MQ135 and LCD display]

- **Performance Analysis of LDR, Photodiode, and BH1750 Sensors for Sunlight Intensity Measurement**," Signal and Image Processing Letters, Vol. 6, No. 1, 2024. [Discusses LDR sensor performance and calibration

Appendices

10.1 Arduino Codes

```
#include <DHT.h>

#include <Wire.h>

#include <LCD_I2C.h>

#include <SD.h>

#include <WiFi.h>

#include <WebServer.h>

#include <time.h>

#include <WiFiUdp.h>


const char* ssid = "idea";

const char* password = "9108503332";


#define SD_CS 5


#define DHTPIN 4

#define DHTTYPE DHT11

#define LDR_PIN 34

#define MQ_PIN  35


DHT dht(DHTPIN, DHTTYPE);

LCD_I2C lcd(0x27, 16, 2);


WebServer server(80);
```

```
float temp = 0.0, humidity = 0.0, ldrPercent = 0.0, gasPercent = 0.0;

void handleRoot() {
  String html = "<html><head><meta http-equiv='refresh' content='5'>";
  html+="<title>Sensor Data</title></head><body>";
  html+="<h2>Live Sensor Data</h2>";
  html+="<p>Temperature: " + String(temp, 1) + " °C</p>";
  html+="<p>Humidity: " + String(humidity, 1) + " %</p>";
  html+="<p>Light Intensity: " + String(ldrPercent, 1) + " %</p>";
  html+="<p>Gas Level: " + String(gasPercent, 1) + " %</p>";
  html+="</body></html>";
  server.send(200,"text/html",html);
}


void setup() {
  Serial.begin(115200);
  delay(1000);

  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status()!=WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi Connected. IP: ");
  Serial.println(WiFi.localIP());
```

```
server.on("/",handleRoot);
server.begin();
Serial.println("Web server started.");
  configTime(19800, 0, "pool.ntp.org");  // IST (GMT+5:30 = 19800s)
Serial.print("Waiting for time sync");
while (time(nullptr) < 100000) {
  Serial.print(".");
  delay(500);
}
Serial.println(" Time synced.");



dht.begin();
Wire.begin();
lcd.begin();
lcd.backlight();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("System Starting...");
delay(2000);
lcd.clear();



if (!SD.begin(SD_CS)) {
  Serial.println("SD Card init failed!");
} else {
```

```
    Serial.println("SD Card initialized.");
    File file = SD.open("/data.csv", FILE_WRITE);
    if (file) {
      file.println("Timestamp,Temperature,Humidity,Light %,Gas %");
      file.close();
    }
  }
}

void loop() {

  int ldrRaw = analogRead(LDR_PIN);
  ldrPercent = 100.0 - ((ldrRaw / 4095.0) * 100.0);

  int mqRaw = analogRead(MQ_PIN);
  gasPercent = (mqRaw / 4095.0) * 100.0;

  temp = dht.readTemperature();
  humidity = dht.readHumidity();




  lcd.clear();
  lcd.setCursor(0, 0);
  if (!isnan(temp) && !isnan(humidity)) {
    lcd.print("T:");
```

```
  lcd.print(temp, 0);
  lcd.print("C H:");
  lcd.print(humidity, 0);
  lcd.print("%");
 } else {
  lcd.print("DHT Read Error");
 }
 lcd.setCursor(0, 1);
 lcd.print("L:");
 lcd.print(ldrPercent, 0);
 lcd.print("% G:");
 lcd.print(gasPercent, 0);
 lcd.print("%");

 //card-logf
 time_t now = time(nullptr);
 struct tm* timeinfo = localtime(&now);
 char timeStr[30];
 strftime(timeStr, sizeof(timeStr), "%H:%M:%S", timeinfo);  // Excel-friendly

 File file = SD.open("/data.csv", FILE_APPEND);
 if (file && !isnan(temp) && !isnan(humidity)) {
  file.print(timeStr);
  file.print(",");
  file.print(temp, 1);
  file.print(",");
  file.print(humidity, 1);
```

```
  file.print(",");
  file.print(ldrPercent, 1);
  file.print(",");
  file.println(gasPercent, 1);
  file.close();
 } else {
  Serial.println("Failed to write to SD card or sensor error.");
 }
 //web
 server.handleClient();

 delay(1000);
}
```

10.2 User Manual

1. Installation and Setup

Unboxing Checklist

- ESP32-based Data Logger unit
- DHT11 Temperature and Humidity Sensor
- LDR (Light Dependent Resistor) sensor
- MQ Gas Sensor
- I2C LCD Display (16x2)
- MicroSD card (≥ 2GB, pre-formatted FAT32)
- USB cable for power and programming
- Jumper wires and breadboard (if applicable)
- User Manual (this document)

Installation Instructions

- Place ESP32 on a stable workbench away from high-voltage equipment.
- Position DHT11 sensor at ambient height away from direct heat.

- Place LDR sensor exposed to typical room lighting.
- Install MQ Gas Sensor at breathing level.

Powering On and Connecting to Network
- Insert SD card.
- Connect ESP32 to USB power.
- Open Serial Monitor (115200 baud).
- Wait for Wi-Fi connection message and IP display.

Connecting Sensors and Configuring Positions
- DHT11: GPIO 4
- LDR: GPIO 34
- MQ Gas Sensor: GPIO 35
- LCD (I2C): GPIO 21 (SDA), GPIO 22 (SCL)
- SD Card: GPIO 5

Setting up Local Display
- LCD shows Temperature, Humidity, Light %, Gas %

Setting up Web Server
- Open browser, enter displayed IP address to view live data.

Troubleshooting Initial Setup
- Check SD formatting, Wi-Fi credentials, sensor connections.

2. Operating Instructions

Starting and Stopping Data Logging
- Power on starts logging automatically.
- Disconnect USB to stop.

Accessing Real-Time Data
- LCD display: live readings.
- Browser: ESP32 IP address.

Calibrating Sensors
- DHT11: No calibration.
- LDR, MQ: Adjust placement.

Exporting Data
- Remove SD card, access data.csv

Viewing Data Analytics
- Via browser and CSV analysis.

Indicator Displays
- LCD for values and errors.
- Serial Monitor for system messages.

3. Data Handling

Data Formats
- CSV: Timestamp, Temperature, Humidity, Light %, Gas %

Logging Intervals
- Every 1 second.

Modifying Data Parameters
- Edit delay(1000) in loop() function.

Data Backup
- Copy data.csv regularly.

Cloud Access
- Local network only via browser.

4. Maintenance and Care

Cleaning
- Power off before cleaning.
- Wipe sensors with dry cloth.

Sensor Calibration
- DHT11: monthly check.
- MQ: periodic reference check.

Power Cycling
- Disconnect USB safely.

Firmware Update
- Upload via Arduino IDE.

Component Replacement
- Replace faulty sensors, check GPIO pins.

5. Troubleshooting Guide

Problems and Solutions
- No display: Check wiring and I2C address.
- Wi-Fi issue: Verify credentials.
- SD failure: Check formatting.
- DHT11 error: Check wiring.
- Web page issue: Verify IP and network.
- Reading stuck: Check connections.

6. Safety Guidelines

Electrical Safety
- Use 5V USB, no water contact.

Sensor Handling
- Avoid bending pins, no overvoltage.

Environment
- 0–50°C, 20–90% RH.

Data Security
- Use secure Wi-Fi, update credentials before deployment.

10.3 Cost Sheet

SI No.	Item	Cost
1	MQ135 Gas Sensor	₹104
2	LCD with I2C	₹226
3	LDR	₹28
4	DHT 11	₹64
5	Breadboard and Wires	₹150
	Total	₹544