BÁO CÁO THỰC HÀNH

Môn học: Công nghệ Internet of Thing hiện đại IoT

Buổi báo cáo: Lab 04

Tên chủ đề: Sử dụng Dashboard và hoàn thiện cơ bản mô hình IoT

GVHD: Trần Văn Như Ý

Ngày thực hiện: 16/04/2025

THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)
Lớp: NT531.P21.2

Nhóm 6

STT	Họ và tên	MSSV	Email
1	Đào Công Sơn	22521249	22521249@gm.uit.edu.vn
2	Lê Khấu Hữu Tài	22521275	22521275@gm.uit.edu.vn
3	Hoàng Thế Anh Tài	22521274	22521274@gm.uit.edu.vn

1. ĐÁNH GIÁ KHÁC:

Nội dung	Kết quả
Tổng thời gian thực hiện bài thực hành trung bình	3 ngày
Link Video thực hiện *(nếu có)*	https://drive.google.com/drive/folders/1csX2ybui4TdqkemlpGWvNudZJqeDSkE5?usp=drive_link
Ý kiến *(nếu có)* + Khó khăn + Đề xuất …	
Điểm tự đánh giá	10/10

BÁO CÁO CHI TIẾT

Câu 1: Xây dựng ứng dụng IoT để giám sát chất lượng không khí và cung cấp dữ liệu theo thời gian thực trên nền tảng opensource IoT platform (Thingsboard, Openremote…)

• Tại phần thiết bị, sinh viên thực hiện sử dụng 2 loại thiết bị cảm biến DHT22/DHT11 và MQ-135 để lấy 3 loại dữ liệu cảm biến là nhiệt độ, độ ẩm, chất lượng không khí. Trên Wemos D1 sử dụng cảm biến MQ-135, đồng thời lập trình để gửi các giá trị cảm biến đến hệ thống thông qua giao thức HTTP. Trên Raspberry sử dụng DHT22/DHT11, đồng thời lập trình để gửi các giá trị cảm biến đến hệ thống thông qua giao thức MQTT.

• Biểu diễn dữ liệu: Cấu hình dashboard trên IoT platform để hiện thị dữ liệu theo thời gian thực từ Wemos D1 và Raspberry. Sử dụng các biểu đồ (charts/graphs) để thể hiện dữ liệu theo thời gian.

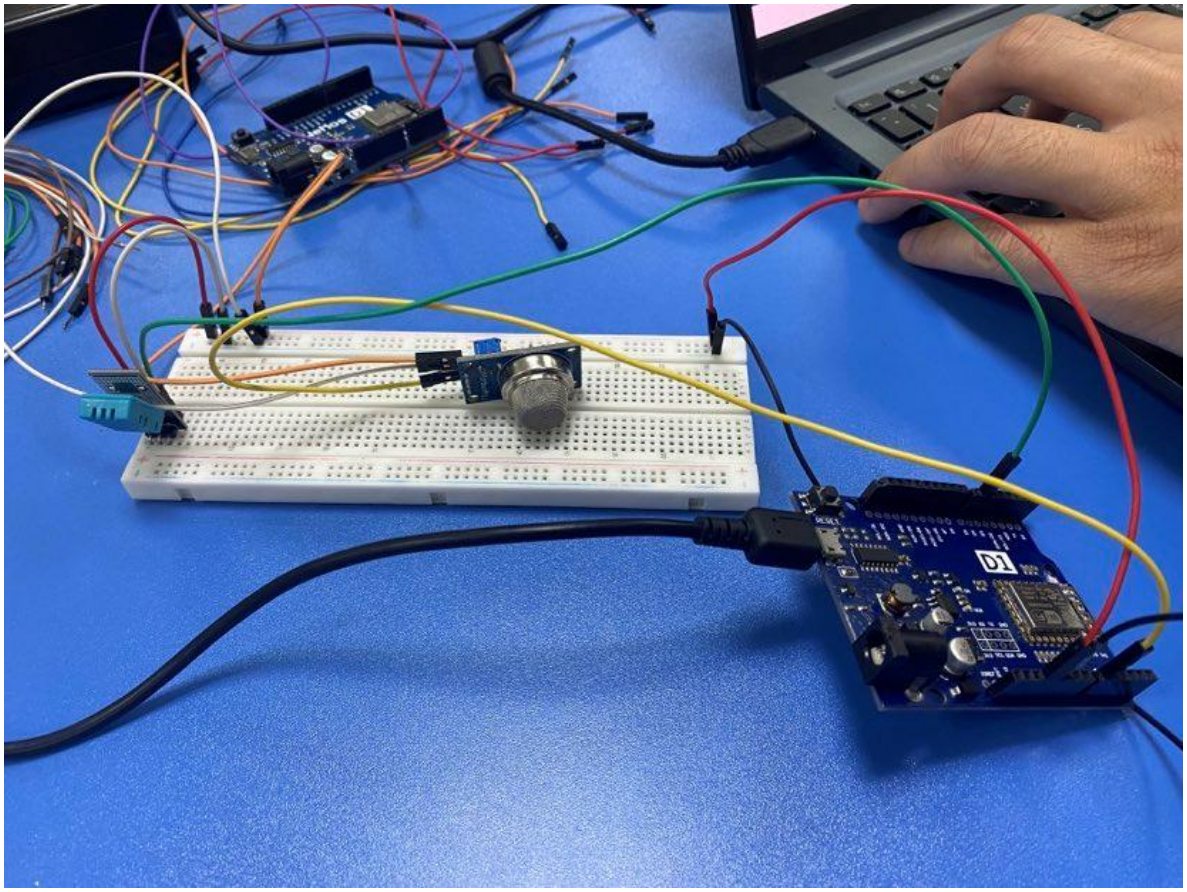
a. Cài đặt

Các thiết bị và thành phần được sử dụng:

- Wemos D1
- Dây nối
- Cảm biến DHT11 và MQ-135
- 1 Laptop (Chạy API server)
- 1 instance server – GCP (Chạy database PostgreSQL)
- 1 WEB UI để hiển thị dữ liệu từ API server

Cách nối dây:

- Kết nối DHT11 và MQ-135 với Wemos D1:
  - Chân data của DHT11 nối với chân D4 của WemosD1
  - VCC của DHT11 -> 5v của WemosD1
  - GND của DHT11 -> GND của WemosD1
  - AO của MQ-135 nối chân A0 trên WemosD1
  - VCC của MQ-135 -> 5v của WemosD1
  - GND của MQ-135 -> GND của WemosD1



b. Giải thích code nạp vào Wemos D1

Đoạn code được nạp vào board Wemos D1 để thu thập dữ liệu từ cảm biến DHT11 (nhiệt độ và độ ẩm) và cảm biến MQ-135 (chất lượng không khí), sau đó gửi dữ liệu này đến API server thông qua giao thức HTTP POST.

Giải thích chi tiết:

1. **Kết nối WiFi:**
   - Sử dụng thông tin SSID và mật khẩu để kết nối WiFi.
   - In địa chỉ IP của thiết bị sau khi kết nối thành công.
2. **Đọc dữ liệu cảm biến:**
   - DHT11 được sử dụng để đo nhiệt độ và độ ẩm.
   - MQ-135 đo chất lượng không khí thông qua chân analog A0.
3. **Gửi dữ liệu đến server:**
   - Dữ liệu được đóng gói dưới dạng JSON (`temperature`, `humidity`, `airQuality`).
   - Gửi dữ liệu qua HTTP POST đến URL server được chỉ định.
4. **Xử lý phản hồi:**
   - In mã phản hồi HTTP và nội dung phản hồi từ server (nếu có).
   - Báo lỗi nếu không gửi được dữ liệu.
5. **Chu kỳ hoạt động:**
   - Lặp lại việc thu thập và gửi dữ liệu mỗi 10 giây.

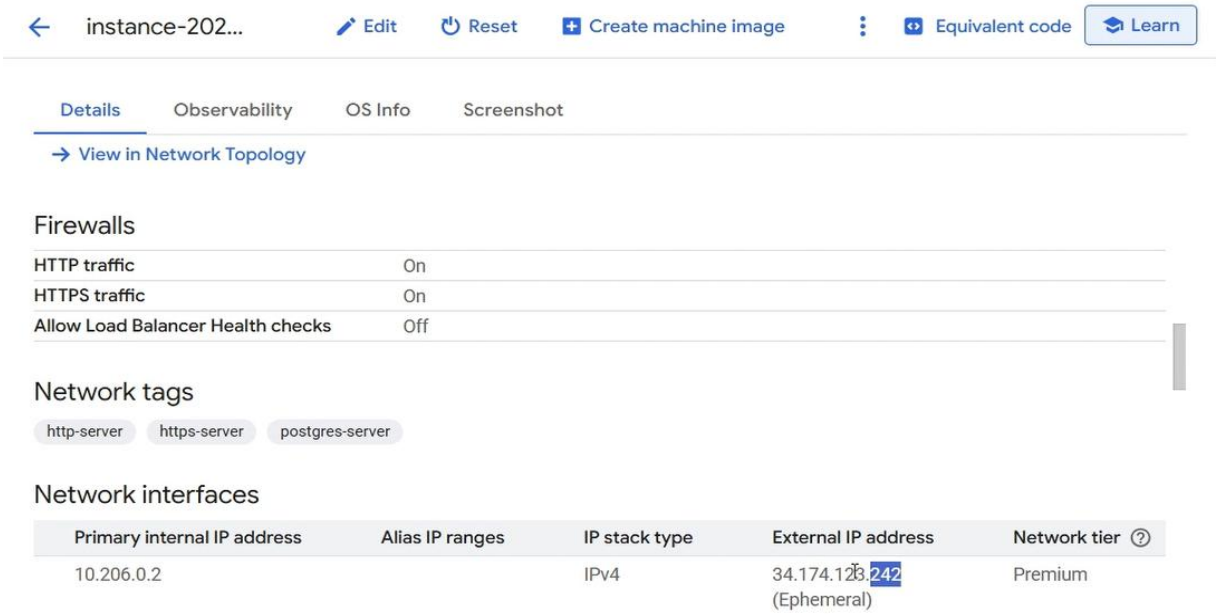
C: > Users > Windows > Downloads > lab04_mq135.ino

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

#define DHTPIN D4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

// Thông tin WiFi
const char* ssid = "UiTiOt-E3.1";
const char* password = "UiTiOtAP";

// URL API Server
const char* serverURL = "http://172.31.9.235:5000/data"; // Thay bằng IP của laptop bạn thay vì localhost

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("Connected!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  dht.begin(); // Khởi tạo cảm biến DHT
}
```

```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    HTTPClient http;

    // Sử dụng cú pháp mới cho begin() với WiFiClient
    http.begin(client, serverURL);
    http.addHeader("Content-Type", "application/json");

    // Đọc dữ liệu từ cảm biến
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();
    int airQuality = analogRead(A0); // Dữ liệu từ MQ-135

    // Kiểm tra nếu dữ liệu hợp lệ
    if (isnan(temperature) || isnan(humidity)) {
      Serial.println("Failed to read from DHT sensor!");
    } else {
      // JSON payload
      String jsonPayload = "{\"temperature\":" + String(temperature) +
                           ",\"humidity\":" + String(humidity) +
                           ",\"airQuality\":" + String(airQuality) + "}";

      Serial.println("Sending data: " + jsonPayload);

      // Gửi HTTP POST
      int httpResponseCode = http.POST(jsonPayload);
      if (httpResponseCode > 0) {
        String response = http.getString();
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);
        Serial.println("Response: " + response);
      } else {
        Serial.print("Error sending data. Error code: ");
        Serial.println(httpResponseCode);
      }
    }

    http.end();
  }
  delay(10000); // Gửi dữ liệu mỗi 10 giây
}
```

c. Khởi tạo server Database PostgreSQL

Tạo Compute Engine instance trên GCP:



SSH vào VM và chạy container database PostgreSQL:

```
docker run --name iot_database -e POSTGRES_USER=admin -e POSTGRES_PASSWORD=admin -e POSTGRES_DB=iot -p 5432:5432 -d iot
```

Khởi tạo Database thành công:

```
Windows@database-IoT:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED       STATUS            PORTS                                       NAMES
52b1411d4a16   postgres   "docker-entrypoint.s…"   2 hours ago   Up About an hour   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp   iot_database
```

d. Giải thích code chạy API server (Laptop)

Đoạn mã Python xây dựng một API server sử dụng Flask để quản lý dữ liệu IoT. Các chức năng chính:

1. **Cấu hình cơ bản:**
   - Sử dụng Flask để tạo server.
   - Kết nối với cơ sở dữ liệu PostgreSQL thông qua thư viện **psycopg2**.
   - Thông tin kết nối được lưu trong **DB_CONFIG**.
2. **Hàm init_db:**
   - **Tạo bảng sensor_data** trong cơ sở dữ liệu **nếu chưa tồn tại**.
   - Bảng lưu các thông số: nhiệt độ, độ ẩm, chất lượng không khí, và thời gian.
3. **API nhận dữ liệu (POST /data):**
   - Nhận dữ liệu JSON từ thiết bị IoT (nhiệt độ, độ ẩm, chất lượng không khí).
   - Kiểm tra dữ liệu đầu vào, lưu vào bảng **sensor_data** trong cơ sở dữ liệu.
   - Trả về phản hồi JSON xác nhận lưu thành công hoặc báo lỗi.
4. **API lấy dữ liệu (GET /data):**
   - Truy vấn 50 bản ghi mới nhất từ bảng sensor_data.
   - Trả về dữ liệu dưới dạng JSON để hiển thị trên giao diện người dùng.

5. **Chạy server:**
   - Server chạy trên cổng 5000 với chế độ debug.
   - Hàm **init_db** được gọi khi khởi động để đảm bảo cơ sở dữ liệu được thiết lập.

Mục đích: API server này đóng vai trò trung gian, nhận dữ liệu từ thiết bị IoT, lưu trữ vào cơ sở dữ liệu, và cung cấp dữ liệu cho giao diện hiển thị.

C: > Users > Windows > Downloads > lab04.py > ...

```python
from flask import Flask, request, jsonify
import psycopg2
from flask_cors import CORS
from psycopg2.extras import RealDictCursor

# Khởi tạo Flask app
app = Flask(__name__)
CORS(app)

# Cấu hình kết nối database PostgreSQL
DB_CONFIG = {
    "host": "34.174.123.242",  # Docker container PostgreSQL chạy trên localhost
    "database": "iot",     # Tên database
    "user": "admin",       # Tên người dùng
    "password": "admin"    # Mật khẩu
}

# Hàm kết nối với database
def get_db_connection():
    conn = psycopg2.connect(
        host=DB_CONFIG["host"],
        database=DB_CONFIG["database"],
        user=DB_CONFIG["user"],
        password=DB_CONFIG["password"]
    )
    return conn
```

```python
# API nhận dữ liệu từ thiết bị Wemos D1
@app.route('/data', methods=['POST'])
def receive_data():
    try:
        # Lấy dữ liệu JSON từ request
        data = request.get_json()
        temperature = data.get('temperature')
        humidity = data.get('humidity')
        air_quality = data.get('airQuality')

        # Kiểm tra dữ liệu đầu vào
        if temperature is None or humidity is None or air_quality is None:
            return jsonify({"error": "Missing data fields"}), 400

        # Kết nối với database và lưu dữ liệu
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO sensor_data (temperature, humidity, air_quality) VALUES (%s, %s, %s)",
            (temperature, humidity, air_quality)
        )
        conn.commit()
        cursor.close()
        conn.close()

        return jsonify({"message": "Data received and saved"}), 200

    except Exception as e:
        print(f"Error: {e}")
        return jsonify({"error": "Internal server error"}), 500
```

```python
# API lấy dữ liệu từ database để hiển thị trên UI
@app.route('/data', methods=['GET'])
def get_data():
    try:
        # Kết nối với database
        conn = get_db_connection()
        cursor = conn.cursor(cursor_factory=RealDictCursor)

        # Truy vấn dữ liệu cảm biến
        cursor.execute("SELECT temperature, humidity, air_quality, timestamp FROM sensor_data ORDER BY timestamp DESC LIMIT 50")
        rows = cursor.fetchall()
        cursor.close()
        conn.close()

        # Trả về dữ liệu dưới dạng JSON
        return jsonify(rows), 200

    except Exception as e:
        print(f"Error: {e}")
        return jsonify({"error": "Internal server error"}), 500
```

```python
# Hàm khởi tạo database (chạy một lần để tạo bảng nếu chưa có)
def init_db():
    try:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS sensor_data (
                id SERIAL PRIMARY KEY,
                temperature REAL,
                humidity REAL,
                air_quality REAL,
                timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            )
        ''')
        conn.commit()
        cursor.close()
        conn.close()
        print("Database initialized successfully!")
    except Exception as e:
        print(f"Error initializing database: {e}")


# Chạy server
if __name__ == '__main__':
    init_db()  # Khởi tạo database nếu cần
    app.run(debug=True, host='0.0.0.0', port=5000)
```

e. Giải thích code UI visualize Data

Đoạn mã HTML tạo giao diện cho một IoT Data Visualization Dashboard với các thành phần chính:

1. Phần `<head>`:

- Thiết lập thông tin cơ bản như mã hóa ký tự (`UTF-8`), khả năng hiển thị trên thiết bị di động (`viewport`).
- Đặt tiêu đề trang là "IoT Dashboard".
- Liên kết tệp CSS (`styles.css`) để định dạng giao diện.
- Tích hợp thư viện Chart.js từ CDN để vẽ biểu đồ.

2. Phần `<body>`:

- Header: Hiển thị tiêu đề chính "IoT Data Visualization Dashboard".
- Main: Chứa 3 phần biểu đồ:
  - Biểu đồ nhiệt độ (°C) với thẻ `<canvas>` có `id="temperatureChart"`.
  - Biểu đồ độ ẩm (%) với thẻ `<canvas>` có `id="humidityChart"`.
  - Biểu đồ chất lượng không khí (PPM) với thẻ `<canvas>` có `id="airQualityChart"`.
- Footer: Hiển thị thông tin bản quyền "IoT Dashboard © 2025".

3. Script:

- Liên kết tệp JavaScript (`script.js`) để xử lý logic và hiển thị biểu đồ.

Mục đích: Giao diện được thiết kế để hiển thị dữ liệu IoT dưới dạng biểu đồ trực quan.

UI.html | script.js | API-server.py | styles.css

UI.html > html > body > main > div.chart-section

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>IoT Dashboard</title>
    <link rel="stylesheet" href="styles.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <header>
        <h1>IoT Data Visualization Dashboard</h1>
    </header>
    <main>
        <div class="chart-section">
            <h2>Temperature (°C)</h2>
            <canvas id="temperatureChart"></canvas>
        </div>
        <div class="chart-section">
            <h2>Humidity (%)</h2>
            <canvas id="humidityChart"></canvas>
        </div>
        <div class="chart-section">
            <h2>Air Quality (PPM)</h2>
            <canvas id="airQualityChart"></canvas>
        </div>
    </main>
    <footer>
        <p>IoT Dashboard &copy; 2025</p>
    </footer>
    <script src="script.js"></script>
</body>
</html>
```

Đoạn mã JavaScript thực hiện việc lấy dữ liệu từ API và hiển thị dữ liệu IoT dưới dạng biểu đồ bằng thư viện **Chart.js**. Các chức năng chính:

1. Khai báo URL API:

- `API_URL` chứa địa chỉ API cung cấp dữ liệu IoT.

2. Hàm `fetchData`:

- Gửi yêu cầu đến API để lấy dữ liệu.
- Trả về dữ liệu dưới dạng JSON hoặc thông báo lỗi nếu có vấn đề.

3. Hàm `renderCharts`:

- Lấy **dữ liệu từ API** thông qua **`fetchData`.**
- Xử lý dữ liệu (đảo ngược thứ tự, tách dữ liệu thành các mảng `timestamps`, `temperatures`, `humidities`, `airQualities`).
- Tạo **3 biểu đồ** (nhiệt độ, độ ẩm, chất lượng không khí) với các tùy chỉnh như màu sắc, nhãn, và tiêu đề.

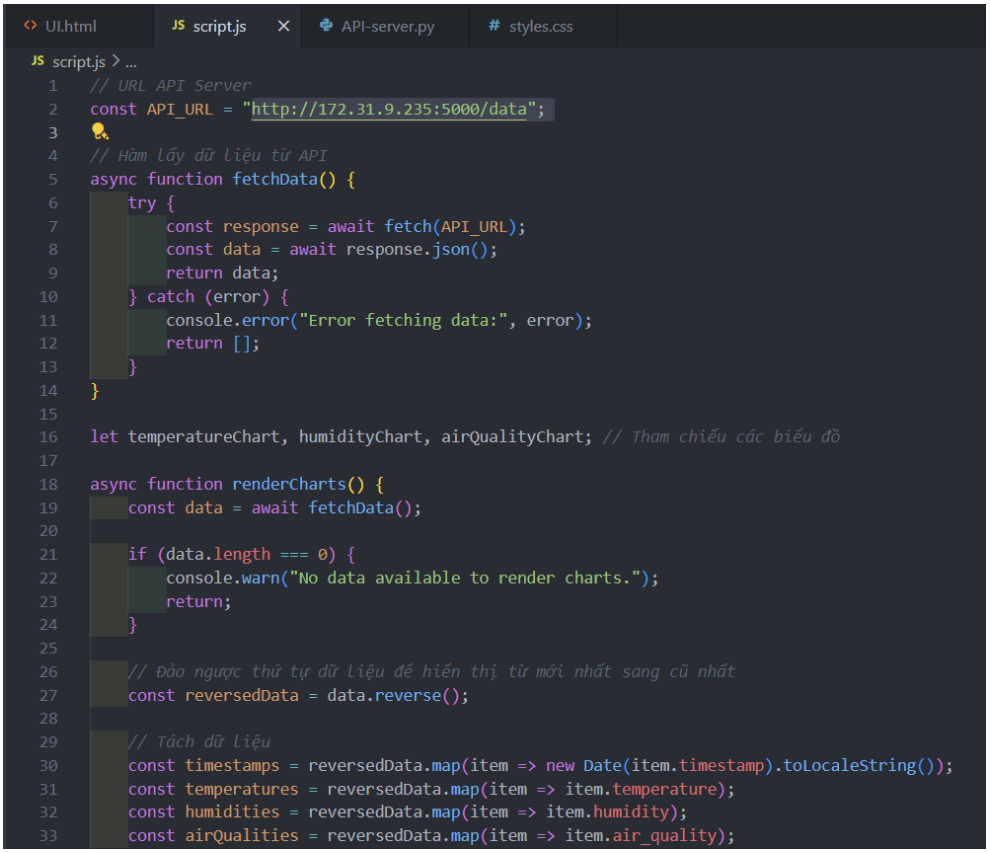
4. Cập nhật biểu đồ:

- Xóa biểu đồ cũ trước khi tạo biểu đồ mới để tránh lỗi.
- Tự động làm mới dữ liệu và **cập nhật biểu đồ** mỗi 10 giây bằng **`setInterval`.**

5. Kết nối với giao diện:

- Các biểu đồ được hiển thị trên các thẻ `<canvas>` trong file HTML (`temperatureChart`, `humidityChart`, `airQualityChart`).

Mục đích: Hiển thị dữ liệu IoT theo thời gian thực, trực quan hóa các thông số như nhiệt độ, độ ẩm, và chất lượng không khí.



```
// Kiểm tra và xóa biểu đồ cũ trước khi tạo biểu đồ mới
if (temperatureChart) temperatureChart.destroy();
if (humidityChart) humidityChart.destroy();
if (airQualityChart) airQualityChart.destroy();

// Biểu đồ nhiệt độ
const temperatureCtx = document.getElementById('temperatureChart').getContext('2d');
temperatureChart = new Chart(temperatureCtx, {
    type: 'line',
    data: {
        labels: timestamps,
        datasets: [{
            label: 'Temperature (°C)',
            data: temperatures,
            borderColor: 'rgba(255, 99, 132, 1)',
            backgroundColor: 'rgba(255, 99, 132, 0.2)',
            borderWidth: 2,
            tension: 0.4,
            fill: true
        }]
    },
    options: {
        responsive: true,
        plugins: {
            legend: { display: true, position: 'top' }
        },
        scales: {
            x: { title: { display: true, text: 'Timestamp' } },
            y: { title: { display: true, text: '°C' } }
        }
    }
});

// Biểu đồ độ ẩm
```

Đoạn mã CSS định dạng giao diện các thành phần cho IoT Dashboard bao gồm:

- Định dạng chung cho body
- Phần header
- Phần main
- Tiêu đề biểu đồ (chart-section h2)
- Phần footer
- Biểu đồ (canvas)

Mục đích: Tạo giao diện hiện đại, dễ nhìn, và tập trung vào việc hiển thị biểu đồ dữ liệu IoT.

UI.html | script.js | API-server.py | styles.css

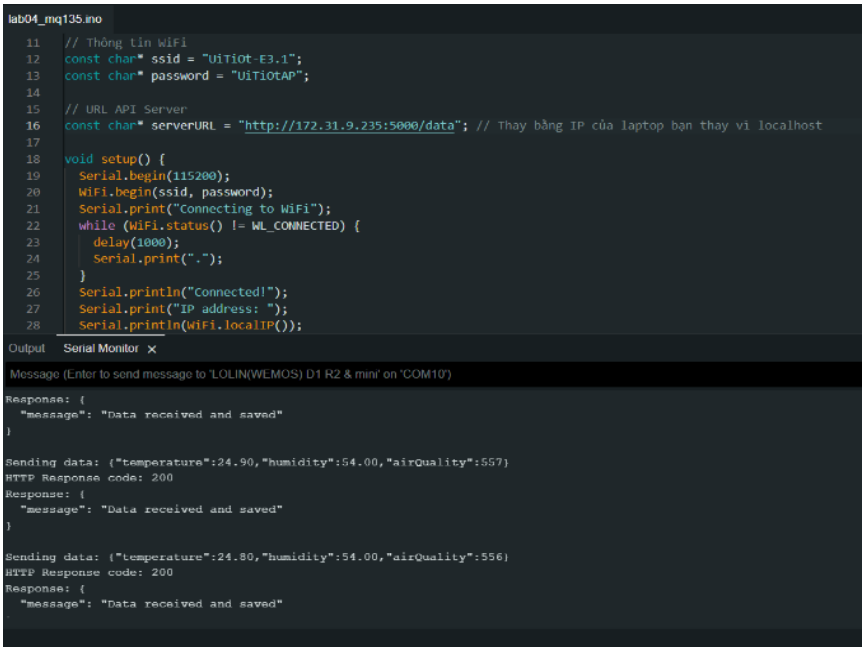
styles.css > *

```
/* General Reset */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Body Styling */
body {
    font-family: 'Arial', sans-serif;
    background-color: #f4f4f9;
    color: #333;
    line-height: 1.6;
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}

/* Header Styling */
header {
    background-color: #007bff;
    color: white;
    text-align: center;
    padding: 1rem 0;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

header h1 {
    font-size: 2rem;
    font-weight: bold;
}

/* Main Section */
main {
    flex: 1;
    padding: 2rem;
    width: 90%;
```

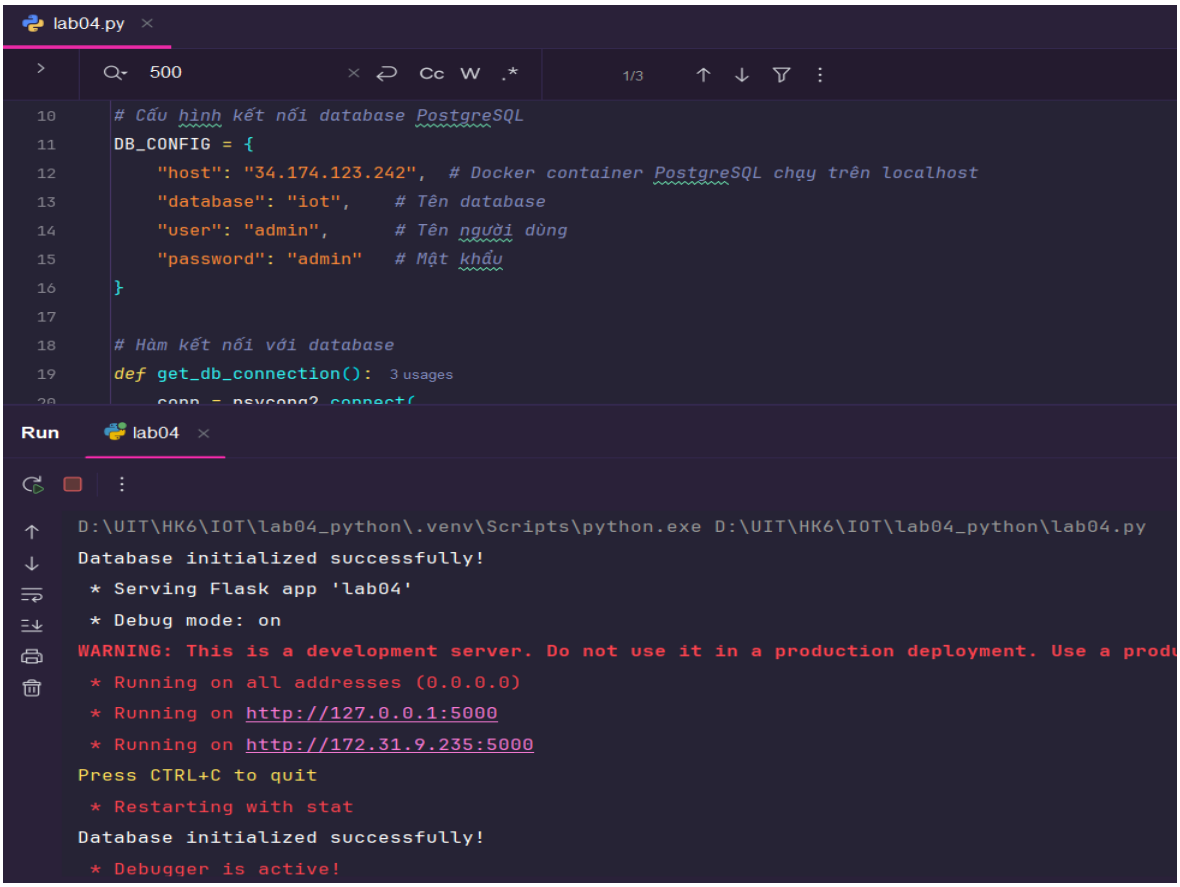
f. Kết quả

UI:

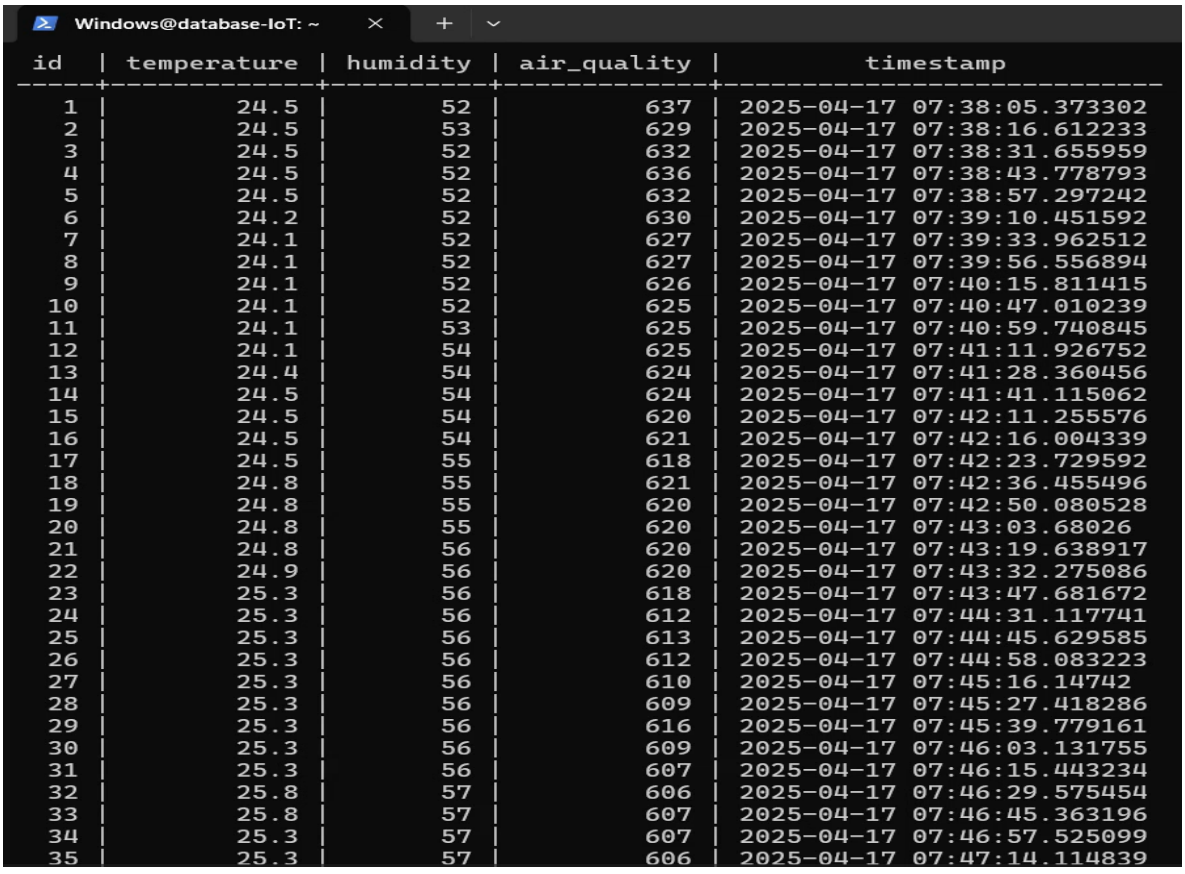
WemosD1:



API server:



Database:



id	temperature	humidity	air_quality	timestamp
1	24.5	52	637	2025-04-17 07:38:05.373302
2	24.5	53	629	2025-04-17 07:38:16.612233
3	24.5	52	632	2025-04-17 07:38:31.655959
4	24.5	52	636	2025-04-17 07:38:43.778793
5	24.5	52	632	2025-04-17 07:38:57.297242
6	24.2	52	630	2025-04-17 07:39:10.451592
7	24.1	52	627	2025-04-17 07:39:33.962512
8	24.1	52	627	2025-04-17 07:39:56.556894
9	24.1	52	626	2025-04-17 07:40:15.811415
10	24.1	52	625	2025-04-17 07:40:47.010239
11	24.1	53	625	2025-04-17 07:40:59.740845
12	24.1	54	625	2025-04-17 07:41:11.926752
13	24.4	54	624	2025-04-17 07:41:28.360456
14	24.5	54	624	2025-04-17 07:41:41.115062
15	24.5	54	620	2025-04-17 07:42:11.255576
16	24.5	54	621	2025-04-17 07:42:16.004339
17	24.5	55	618	2025-04-17 07:42:23.729592
18	24.8	55	621	2025-04-17 07:42:36.455496
19	24.8	55	620	2025-04-17 07:42:50.080528
20	24.8	55	620	2025-04-17 07:43:03.68026
21	24.8	56	620	2025-04-17 07:43:19.638917
22	24.9	56	620	2025-04-17 07:43:32.275086
23	25.3	56	618	2025-04-17 07:43:47.681672
24	25.3	56	612	2025-04-17 07:44:31.117741
25	25.3	56	613	2025-04-17 07:44:45.629585
26	25.3	56	612	2025-04-17 07:44:58.083223
27	25.3	56	610	2025-04-17 07:45:16.14742
28	25.3	56	609	2025-04-17 07:45:27.418286
29	25.3	56	616	2025-04-17 07:45:39.779161
30	25.3	56	609	2025-04-17 07:46:03.131755
31	25.3	56	607	2025-04-17 07:46:15.443234
32	25.8	57	606	2025-04-17 07:46:29.575454
33	25.8	57	607	2025-04-17 07:46:45.363196
34	25.3	57	607	2025-04-17 07:46:57.525099
35	25.3	57	606	2025-04-17 07:47:14.114839

g. **Link video Demo:**

UI-Database:
https://drive.google.com/file/d/1dXAbP-_RCjknI5Vypg4f3j0UN6cyJjJC/view?usp=sharing
APIServer-WemosD1:
https://drive.google.com/file/d/11k23FmOysRly56xvFBbB0PpGTtNxrhtM/view?usp=sharing
TestAPI:
https://drive.google.com/file/d/1G-No5bWZHAur_fLd0iQ_OhA1ljYDuNag/view?usp=sharing

YÊU CẦU CHUNG

1) Đánh giá

- Chuẩn bị tốt các yêu cầu đặt ra trong bài thực hành.
- Sinh viên hiểu và tự thực hiện được bài thực hành, trả lời đầy đủ các yêu cầu đặt ra.
- Nộp báo cáo kết quả chi tiết những đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả *(nếu có)*; giải thích cho quan sát *(nếu có)*.
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

2) Báo cáo

- File .PDF hoặc .docx. Tập trung vào nội dung, giải thích.
- Nội dung trình bày bằng Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Avo)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
- Đặt tên theo định dạng: LabX_MSSV1. (trong đó X là Thứ tự buổi Thực hành).
  Ví dụ: Lab01_21520001
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT