TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

**KHOA ĐIỆN TỬ**

**Bộ môn: Công nghệ Thông tin.**

**BÀI TẬP LỚN**

MÔN HỌC

**XÂY DỰNG HỆ THỐNG IOT**

**(INTERNET OF THING)**

Sinh viên: Phạm Quang Trường – K214580106129

Lớp: K57KMT

Giáo viên hướng dẫn: Nguyễn Văn Huy

**Thái Nguyên – 2025**

|  |  |
| --- | --- |
| **TRƯỜNG ĐHKTCN** | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA ĐIỆN TỬ** | ***Độc lập - Tự do - Hạnh phúc*** |

**BÀI TẬP LỚN**

**MÔN HỌC** **XÂY DỰNG HỆ THỐNG IOT**

BỘ MÔN CÔNG NGHỆ THÔNG TIN

*Sinh viên: Phạm Quang Trường*

*Lớp*: *K57KMT* *Ngành: Tin học công nghiệp*

*Giáo viên hướng dẫn: Nguyễn Văn Huy*

*Ngày giao đề*  *Ngày hoàn thành*

*Tên đề tài : Điều khiển và giám sát hệ thống thông gió.*

*Yêu cầu :*

* Giao diện web: Giám sát nhiệt độ, độ ẩm, trạng thái relay, điều khiển tự động/tay.
* Quản lý dữ liệu: Lưu vào SQLite, truy vấn theo thời gian, gửi Firebase khi cần.
* Bảo mật: MQTT có xác thực, bảo vệ bằng WireGuard VPN, xác thực đăng nhập.
* Tự động hóa: Điều khiển relay theo ngưỡng nhiệt độ/độ ẩm, lưu lịch sử thay đổi.
* Hiệu suất: Tối ưu Flask, xem xét nâng cấp database, kiểm tra độ ổn định.

|  |
| --- |
| **GIÁO VIÊN HƯỚNG DẪN** |
| *(Ký và ghi rõ họ tên)* |
|  |

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Thái Nguyên, ngày….tháng…..năm 20....

**GIÁO VIÊN HƯỚNG DẪN**

*(Ký ghi rõ họ tên)*

**MỤC LỤC**

[LỜI NÓI ĐẦU 5](#_Toc191254261)

[CHƯƠNG I. GIỚI THIỆU 6](#_Toc191254262)

[1.1. Khái niệm và ứng dụng của IoT. 6](#_Toc191254263)

[*1.1.1.* *Khái niệm IoT.* 6](#_Toc191254264)

[*1.1.2.* *Ứng dụng của IoT.* 6](#_Toc191254265)

[1.2. Giới thiệu một ứng dụng thực thế của IoT. 6](#_Toc191254266)

[1.3. Giới thiệu và phạm vi đề tài. 7](#_Toc191254267)

[*1.3.1.* *Giới thiệu đề tài.* 7](#_Toc191254268)

[*1.3.2.* *Mục tiêu và lý di thực hiện đề tài.* 7](#_Toc191254269)

[*1.3.3.* *Phạm vi và tính năng của sản phẩm.* 8](#_Toc191254270)

[CHƯƠNG 2: THIẾT KẾ XÂY DỰNG HỆ THỐNG 9](#_Toc191254271)

[2.1. Tổng quan. 9](#_Toc191254272)

[2.2. Thu thập dữ liệu. 9](#_Toc191254273)

[2.3. Lưu trữ dữ liệu: SQLite và Firebase. 13](#_Toc191254274)

[2.4. Xây dựng Web Service với Flask. 14](#_Toc191254275)

[*2.4.1. Cấu trúc chính của ứng dụng Flask.* 14](#_Toc191254276)

[*2.4.2. Cách thức hoạt động và giao tiếp với các thành phần khác.* 15](#_Toc191254277)

[2.5. Bảo mật (Sử dụng WireGuard VPN). 15](#_Toc191254278)

[CHƯƠNG 3: THỬ NGHIỆM HỆ THỐNG 18](#_Toc191254279)

[3.1. Giao diện giám sát và điều khiển. 18](#_Toc191254280)

[3.2. Quản lý dữ liệu. 19](#_Toc191254281)

[3.3. Bảo mật. 20](#_Toc191254282)

[3.4. Kết luận. 20](#_Toc191254283)

# LỜI NÓI ĐẦU

Trong thời đại công nghệ 4.0, các hệ thống giám sát và điều khiển từ xa ngày càng đóng vai trò quan trọng trong việc nâng cao hiệu suất và tiết kiệm năng lượng. Đặc biệt, trong lĩnh vực IoT (Internet of Things), việc ứng dụng các công nghệ tiên tiến giúp tối ưu hóa quá trình vận hành và quản lý thiết bị một cách hiệu quả.

Bài tập lớn này tập trung vào việc xây dựng một hệ thống giám sát và điều khiển thông gió sử dụng ESP8266, cảm biến DHT22, relay và giao thức MQTT. Hệ thống cho phép theo dõi nhiệt độ, độ ẩm theo thời gian thực, tự động điều khiển relay dựa trên ngưỡng cài đặt và lưu trữ dữ liệu vào cơ sở dữ liệu SQLite. Đồng thời, việc tích hợp Firebase giúp truy xuất dữ liệu từ xa, và WireGuard đảm bảo an toàn khi kết nối.

Trong quá trình thực hiện, nhóm đã nghiên cứu và áp dụng các công nghệ như Flask để xây dựng giao diện web, Paho MQTT để truyền nhận dữ liệu, và các giải pháp bảo mật để đảm bảo tính ổn định của hệ thống. Bài tập lớn không chỉ giúp củng cố kiến thức về IoT mà còn mở ra hướng phát triển các ứng dụng thông minh trong tương lai.

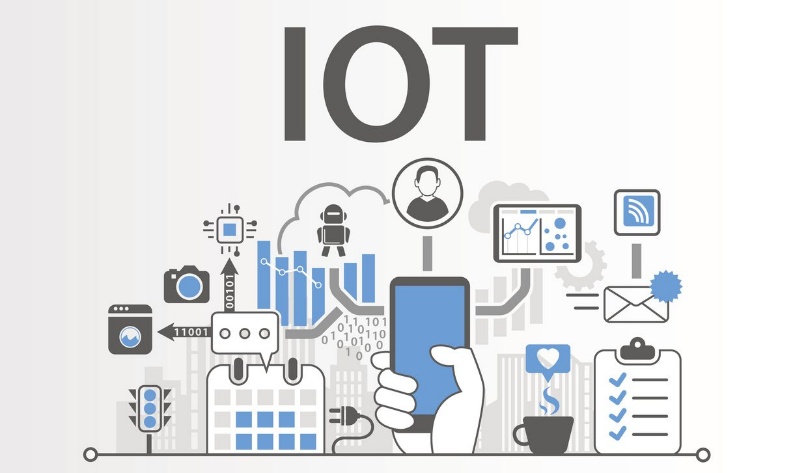
Hy vọng rằng kết quả đạt được sẽ là nền tảng cho các nghiên cứu và ứng dụng tiếp theo. Rất mong nhận được sự đóng góp ý kiến từ thầy cô và bạn bè để hoàn thiện hơn nữa.

Sinh viên thực hiện

# CHƯƠNG I. GIỚI THIỆU

* 1. **Khái niệm và ứng dụng của IoT.**
     1. *Khái niệm IoT.*

Internet of Things (IoT) là một khái niệm mô tả hệ thống các thiết bị và cảm biến được kết nối với nhau qua Internet để thu thập, trao đổi và xử lý dữ liệu. Các thiết bị này không chỉ thực hiện các nhiệm vụ đơn giản mà còn có khả năng tự động hóa và tương tác với môi trường xung quanh. IoT mở ra cơ hội ứng dụng rộng rãi trong nhiều lĩnh vực nhờ khả năng giám sát và điều khiển từ xa, góp phần tối ưu hóa tài nguyên và nâng cao chất lượng cuộc sống.



Hình 1.1: Vạn vận kết nối với nhau qua Internet.

* + 1. *Ứng dụng của IoT.*

IoT được ứng dụng trong nhiều lĩnh vực khác nhau, chẳng hạn:

* **Nhà thông minh:** Tự động điều chỉnh ánh sáng, nhiệt độ, an ninh và các thiết bị gia dụng dựa trên dữ liệu thu thập từ cảm biến.
* **Thành phố thông minh:** Quản lý giao thông, chiếu sáng đường phố, giám sát môi trường và an ninh công cộng.
* **Công nghiệp 4.0:** Tự động hóa dây chuyền sản xuất, giám sát máy móc và tối ưu hóa quy trình vận hành.
* **Y tế:** Giám sát bệnh nhân từ xa, quản lý dữ liệu sức khỏe và hỗ trợ chẩn đoán.
* **Nông nghiệp:** Tự động tưới tiêu, giám sát điều kiện đất và khí hậu, từ đó cải thiện năng suất canh tác.

Nhờ khả năng kết nối liên tục và xử lý dữ liệu thời gian thực, IoT đang trở thành nền tảng công nghệ hỗ trợ phát triển các giải pháp thông minh, tiết kiệm chi phí và tăng cường hiệu quả trong nhiều ngành nghề.

* 1. **Giới thiệu một ứng dụng thực thế của IoT.**

Một ví dụ điển hình của ứng dụng IoT là hệ thống **nhà thông minh**. Hệ thống này tích hợp nhiều thiết bị và cảm biến để tạo nên một môi trường sống tiện nghi và an toàn. Cụ thể:

Các thành phần chính:

* **Cảm biến:** Như cảm biến chuyển động, cảm biến nhiệt độ, cảm biến ánh sáng… để thu thập dữ liệu về môi trường sống.
* **Bộ xử lý trung tâm:** Các vi điều khiển có nhiệm vụ nhận dữ liệu từ các cảm biến, xử lý và gửi thông tin đến hệ thống điều khiển.
* **Thiết bị điều khiển:** Relay, ổ cắm thông minh, camera giám sát,… được kích hoạt theo các lệnh điều khiển từ hệ thống.
* **Giao diện người dùng:** Ứng dụng di động hoặc giao diện web cho phép người dùng theo dõi và điều khiển các thiết bị trong nhà từ xa.

Quy trình hoạt động:

* Các cảm biến trong nhà thu thập dữ liệu liên tục (ví dụ: nhiệt độ, ánh sáng, chuyển động), gửi về bộ xử lý trung tâm để phân tích. Dựa trên các tiêu chí đã được lập trình, hệ thống sẽ tự động điều chỉnh hoạt động của các thiết bị như bật/tắt đèn, điều chỉnh nhiệt độ phòng, hoặc kích hoạt báo động an ninh. Qua đó, người dùng có thể theo dõi và kiểm soát mọi hoạt động của nhà mình thông qua giao diện web hoặc ứng dụng di động.

Lợi ích:

* Hệ thống nhà thông minh giúp tối ưu hóa tiêu thụ năng lượng, tăng cường an ninh, và tạo ra sự tiện nghi, linh hoạt cho người sử dụng.
  1. **Giới thiệu và phạm vi đề tài.**
     1. *Giới thiệu đề tài.*

Đề tài báo cáo này hướng đến việc xây dựng một hệ thống thông gió thông minh dựa trên nền tảng IoT, với các thành phần chính bao gồm:

* **ESP8266/ESP32:** Vi xử lý tích hợp Wi-Fi, đóng vai trò quan trọng trong việc kết nối và truyền tải dữ liệu.
* **Cảm biến DHT22:** Dùng để đo nhiệt độ và độ ẩm, cung cấp dữ liệu môi trường chính xác.
* **Relay:** Thiết bị điều khiển bật/tắt hệ thống thông gió dựa trên các điều kiện môi trường.
* **Web Service sử dụng Flask và giao thức MQTT:** Xây dựng giao diện giám sát và điều khiển từ xa, đảm bảo quá trình truyền tải dữ liệu ổn định và hiệu quả.
* **Raspberry Pi:** Đóng vai trò làm server chạy web và lưu trữ cơ sở dữ liệu.
  + 1. *Mục tiêu và lý di thực hiện đề tài.*

Mục tiêu: Xây dựng một hệ thống thông gió IoT có khả năng thu thập dữ liệu môi trường (nhiệt độ, độ ẩm) và điều khiển hoạt động của hệ thống thông gió tự động hoặc thủ công thông qua giao diện web.

Lý do chọn đề tài:

* Nhu cầu giám sát và điều chỉnh môi trường sống ngày càng cao nhằm tiết kiệm năng lượng và tối ưu hóa điều kiện không gian.
* Ứng dụng IoT trong việc điều khiển thông gió giúp cải thiện chất lượng không khí, tạo môi trường sống trong lành và giảm chi phí vận hành.
  + 1. *Phạm vi và tính năng của sản phẩm.*

Phạm vi của sản phẩm:

* Hệ thống được thiết kế để thu thập và xử lý dữ liệu môi trường ở một khu vực nhỏ như một phòng hoặc một khu vực cụ thể trong nhà, làm mẫu trước khi triển khai rộng rãi.

Tính năng chính:

* **Thu thập dữ liệu:** Ghi nhận thông tin nhiệt độ và độ ẩm từ cảm biến DHT22, truyền dữ liệu qua giao thức MQTT.
* **Điều khiển hệ thống thông gió:** Sử dụng relay để bật/tắt thiết bị thông gió dựa trên dữ liệu thu thập được hoặc theo lệnh của người dùng.
* **Giao diện giám sát và điều khiển:** Xây dựng giao diện web bằng Flask, hiển thị dữ liệu thời gian thực và cung cấp các chức năng điều khiển từ xa.
* **Lưu trữ và phân tích dữ liệu:** Lưu trữ dữ liệu dưới dạng realtime và dữ liệu lưu trữ lâu dài nhằm phục vụ việc phân tích xu hướng và tối ưu hóa hoạt động của hệ thống.

# CHƯƠNG 2: THIẾT KẾ XÂY DỰNG HỆ THỐNG

## 2.1. Tổng quan.

Sơ đồ khối hệ thống:

* **Cảm biến và ESP8266:** Thu thập dữ liệu môi trường (nhiệt độ, độ ẩm) từ cảm biến DHT22, xử lý và gửi dữ liệu qua giao thức MQTT.
* **Relay:** Nhận lệnh điều khiển từ ESP8266 để bật/tắt hệ thống thông gió.
* **MQTT Broker:** Là trung tâm truyền tải dữ liệu giữa các thiết bị (ESP8266) và Web Service.
* **Web Service (Flask):** Giao diện hiển thị dữ liệu realtime, cung cấp các API điều khiển và lưu trữ dữ liệu.
* **Lưu trữ dữ liệu:** Dữ liệu được ghi vào cơ sở dữ liệu SQLite để lưu trữ lịch sử và đẩy dữ liệu lên Firebase Realtime Database.

A diagram of a computer network

AI-generated content may be incorrect.

Hình 2.1: Sơ đồ khối hệ thống thông gió.

Dữ liệu vào ra giữa các khối:

* **Dữ liệu vào (Input):**
  + Dữ liệu môi trường: nhiệt độ, độ ẩm (được thu thập từ DHT22).
  + Lệnh điều khiển từ người dùng (qua Web Service) gửi xuống hệ thống.
* **Dữ liệu ra (Output):**
  + Hiển thị dữ liệu realtime trên giao diện web.
  + Gửi lệnh kích hoạt relay để bật/tắt thông gió.
  + Lưu trữ dữ liệu cho việc phân tích xu hướng hoạt động hệ thống.

## 2.2. Thu thập dữ liệu.

Lý thuyết tổng quan:

* Chức năng: Thu thập dữ liệu môi trường là bước nền tảng của hệ thống IoT, cho phép giám sát liên tục và cập nhật trạng thái của môi trường.
* Ý nghĩa: Dữ liệu chính xác, kịp thời giúp hệ thống tự động phản ứng (ví dụ: kích hoạt quạt thông gió khi nhiệt độ vượt mức) và cung cấp thông tin để phân tích, tối ưu hóa hoạt động.

Kết nối phần cứng:

* Cảm biến DHT22:
  + Loại tín hiệu: Số (digital signal).
  + Đặc tính: Đo nhiệt độ và độ ẩm với độ chính xác tương đối cao; tốc độ cập nhật dữ liệu định kỳ (thường từ 1 đến 2 giây).
* Relay:
  + Loại tín hiệu: Số (digital signal).
  + Đặc tính: Đóng/mở mạch để cho hoặc chặn dòng điện đi qua.
* Kết nối phần cứng ESP8266:
  + Chân điều khiển: Kết nối chân IN của module relay với chân D4 của ESP8266. Dây dữ liệu nối chân tín hiệu của DHT22 vào chân D3 của ESP8266.
  + Nguồn và đất: Đảm bảo module relay được cấp nguồn phù hợp (thường là 5V hoặc 3.3V, tùy module) và nối đất đúng cách.

Thuật toán thu thập và xử lý dữ liệu:

* Khởi tạo cảm biến: Thiết lập các tham số và kiểm tra kết nối với DHT22.
* Đọc dữ liệu: Lấy giá trị nhiệt độ và độ ẩm từ cảm biến theo chu kỳ định sẵn.
* Lọc và hiệu chuẩn: Kiểm tra dữ liệu đầu vào, loại bỏ nhiễu nếu cần và hiệu chuẩn theo ngưỡng đã xác định.
* Gửi dữ liệu: Chuyển dữ liệu đã xử lý lên MQTT broker để phục vụ các chức năng giám sát realtime.

Code ESP8266

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <DHT.h>

#include <ArduinoJson.h>

const char\* ssid = "RiNo House";

const char\* password = "12345678@";

const char\* mqtt\_server = "nekotrang.duckdns.org";

const int mqtt\_port = 1883;

#define DHTPIN D4

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

#define RELAY\_PIN D3

WiFiClient espClient;

PubSubClient client(espClient);

const char\* topicData = "nhom28/data";

const char\* topicRelay = "nhom28/relay";

unsigned long lastPublishTime = 0;

const long publishInterval = 5000;

float lastTemperature = NAN;

float lastHumidity = NAN;

int lastRelayState = -1;

void setup() {

  Serial.begin(115200);

  dht.begin();

  pinMode(RELAY\_PIN, OUTPUT);

  digitalWrite(RELAY\_PIN, LOW);

  setup\_wifi();

  client.setServer(mqtt\_server, mqtt\_port);

  client.setCallback(callback);

}

void loop() {

  if (!client.connected()) {

    reconnect();

  }

  client.loop();

  unsigned long currentMillis = millis();

  if (currentMillis - lastPublishTime >= publishInterval) {

    lastPublishTime = currentMillis;

    float temperature = dht.readTemperature();

    float humidity = dht.readHumidity();

    int relayState = digitalRead(RELAY\_PIN);

    if ((!isnan(temperature) && temperature != lastTemperature) ||

        (!isnan(humidity) && humidity != lastHumidity) ||

        (relayState != lastRelayState)) {

      lastTemperature = temperature;

      lastHumidity = humidity;

      lastRelayState = relayState;

      StaticJsonDocument<200> doc;

      doc["temperature"] = temperature;

      doc["humidity"] = humidity;

      doc["relay"] = relayState;

      char buffer[512];

      serializeJson(doc, buffer);

      client.publish(topicData, buffer);

      Serial.println("Đã gửi dữ liệu: " + String(buffer));

    }

  }

}

void setup\_wifi() {

  Serial.println("Đang kết nối WiFi...");

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(1000);

    Serial.print(".");

  }

  Serial.println("\nWiFi đã kết nối!");

}

void callback(char\* topic, byte\* payload, unsigned int length) {

  String message;

  for (int i = 0; i < length; i++) {

    message += (char)payload[i];

  }

  Serial.println("Nhận MQTT từ " + String(topic) + ": " + message);

  StaticJsonDocument<200> doc;

  DeserializationError error = deserializeJson(doc, message);

  if (error) {

    Serial.println("Lỗi phân tích JSON");

    return;

  }

  if (String(topic) == topicRelay && doc.containsKey("relay")) {

    int relayState = doc["relay"];

    digitalWrite(RELAY\_PIN, relayState ? HIGH : LOW);

    Serial.println("Relay: " + String(relayState));

  }

}

void reconnect() {

  while (!client.connected()) {

    if (client.connect("ESP8266Client")) {

      client.subscribe(topicRelay);

      Serial.println("Đã subscribe: " + String(topicRelay));

    } else {

      delay(5000);

    }

  }

}

## 2.3. Lưu trữ dữ liệu: SQLite và Firebase.

Lưu Trữ Nội Bộ với SQLite: Ở phía server, dữ liệu nhận từ MQTT sẽ được lưu vào SQLite để lưu trữ lịch sử. Một số hàm chính:

* Khởi tạo cơ sở dữ liệu: Nếu chưa có database sẵn thì hệ thống sẽ tự tạo ra để lưu dữ liệu.

def init\_db():  
 conn = sqlite3.connect('sensor\_data.db')  
 c = conn.cursor()  
 c.execute('''CREATE TABLE IF NOT EXISTS sensor\_data  
 (id INTEGER PRIMARY KEY AUTOINCREMENT,  
 temperature REAL,  
 humidity REAL,  
 relay INTEGER,  
 timestamp DATETIME)''')  
 c.execute("CREATE INDEX IF NOT EXISTS idx\_timestamp ON sensor\_data(timestamp)")  
 conn.commit()  
 conn.close()

* Lưu dữ liệu vào SQLite: Khi nhận được dữ liệu từ ESP8266 thông qua giao thức MQTT thì sẽ được lưu vào database.

def save\_to\_db(temperature, humidity, relay):  
 conn = sqlite3.connect('sensor\_data.db')  
 c = conn.cursor()  
 timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')  
 c.execute("INSERT INTO sensor\_data (temperature, humidity, relay, timestamp) VALUES (?, ?, ?, ?)",  
 (temperature, humidity, relay, timestamp))  
 conn.commit()  
 conn.close()

* Đẩy Dữ Liệu Lên Firebase: Sử dụng thư viện Pyrebase để gửi dữ liệu sensor kèm timestamp lên Firebase Realtime Database.

def push\_to\_firebase(temperature, humidity, relay):  
 global last\_firebase\_data  
 temp\_margin = 0.5  
 humi\_margin = 2.0  
 if last\_firebase\_data is not None:  
 if (abs(temperature - last\_firebase\_data["temperature"]) < temp\_margin and  
 abs(humidity - last\_firebase\_data["humidity"]) < humi\_margin and  
 relay == last\_firebase\_data["relay"]):  
 print("Dữ liệu không thay đổi nên không cần gửi lên Firebase")  
 return  
  
 timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')  
 data\_firebase = {  
 "temperature": temperature,  
 "humidity": humidity,  
 "relay": relay,  
 "timestamp": timestamp  
 }  
 db\_firebase.child("sensor\_data").push(data\_firebase)  
 last\_firebase\_data = {"temperature": temperature, "humidity": humidity, "relay": relay}  
 print("Dữ liệu được gửi lên Firebase:", data\_firebase)

**2.4. Xây dựng Web Service với Flask.**

Phần này tập trung vào việc tạo ra một ứng dụng web sử dụng Flask, cho phép:

* **Hiển thị dữ liệu realtime:** Giao diện chính (index) sẽ hiển thị thông tin cảm biến, trạng thái relay và biểu đồ cập nhật liên tục.
* **Truy xuất dữ liệu lịch sử:** Endpoint **/get\_history\_data** cho phép người dùng lọc và xem dữ liệu theo khoảng thời gian mong muốn.
* **Điều khiển hệ thống:** Endpoint **/control** nhận các lệnh điều khiển như bật/tắt relay, chuyển chế độ tự động hoặc cập nhật ngưỡng nhiệt độ, độ ẩm.
* **Giao tiếp dữ liệu:** Endpoint **/get\_all\_data** cung cấp dữ liệu mới nhất và danh sách các bản ghi từ SQLite để cập nhật giao diện realtime.

### 2.4.1. Cấu trúc chính của ứng dụng Flask.

**Trang chủ (index):**

* Endpoint / dùng hàm index() để render trang HTML chính (index.html) chứa giao diện người dùng với biểu đồ, bảng điều khiển, và các nút chức năng.

**Lấy dữ liệu realtime:** Endpoint /get\_all\_data trả về một JSON gồm:

* latest\_data: Dữ liệu cảm biến mới nhất (nhiệt độ, độ ẩm, trạng thái relay).
* records: Danh sách các bản ghi lưu trong SQLite (để hiển thị biểu đồ realtime).
* auto\_mode và thresholds: Thông tin cấu hình chế độ tự động và các ngưỡng điều khiển.

**Điều khiển hệ thống:** Endpoint /control nhận các yêu cầu POST từ giao diện để thực hiện các thao tác:

* set\_relay: Điều khiển trạng thái relay (bật/tắt) và tắt chế độ tự động.
* toggle\_auto: Bật hoặc tắt chế độ tự động điều khiển relay dựa trên ngưỡng nhiệt độ, độ ẩm.
* set\_thresholds: Cập nhật ngưỡng nhiệt độ và độ ẩm.

**Lấy dữ liệu lịch sử:**

* Endpoint /get\_history\_data cho phép người dùng truyền khoảng thời gian (start và end) dưới dạng chuỗi datetime và trả về danh sách các bản ghi trong khoảng thời gian đó.

### 2.4.2. Cách thức hoạt động và giao tiếp với các thành phần khác.

**Giao tiếp với MQTT:**

* Khi nhận được lệnh điều khiển từ giao diện, Flask gọi MQTT client (được khởi tạo từ module paho-mqtt) để gửi thông điệp điều khiển đến các thiết bị. Ví dụ: khi người dùng nhấn nút bật relay, endpoint /control gửi lệnh MQTT tới topic nhom28/relay.

**Lưu trữ dữ liệu:**

* Dữ liệu cảm biến nhận được qua MQTT đã được lưu vào cơ sở dữ liệu SQLite để phục vụ việc lưu trữ lịch sử và cũng được đẩy lên Firebase. Endpoint /get\_all\_data và /get\_history\_data truy xuất dữ liệu từ SQLite để cung cấp cho giao diện web.

**Cập nhật giao diện:**

* Giao diện web sử dụng các API trên Flask để lấy dữ liệu mới nhất và cập nhật biểu đồ thời gian thực (Chart.js) cũng như các phần hiển thị dữ liệu cảm biến.

## 2.5. Bảo mật (Sử dụng WireGuard VPN).

2.5.1. Giới thiệu về WireGuard.

Trong hệ thống IoT, bảo mật giao tiếp giữa các thành phần là yếu tố thiết yếu để đảm bảo tính toàn vẹn và riêng tư của dữ liệu. Một trong những giải pháp hiện đại và hiệu quả là sử dụng VPN với WireGuard. WireGuard cung cấp một kênh truyền tải mã hóa mạnh mẽ, giúp bảo vệ dữ liệu khỏi các tấn công từ bên ngoài và đảm bảo chỉ có các thiết bị đã được xác thực mới có thể truy cập vào hệ thống. Một số ưu điểm của việc sử dụng WireGuard VPN:

* **Hiệu năng cao:** WireGuard được thiết kế nhẹ và tối ưu, cho phép xử lý nhanh chóng trên cả hệ thống server và các thiết bị nhúng.
* **Cấu hình đơn giản:** Cấu hình của WireGuard chỉ cần tạo một tệp cấu hình cho từng thành phần với khóa riêng và khóa công khai, giúp dễ dàng triển khai và quản lý.
* **Bảo mật tiên tiến:** WireGuard sử dụng các thuật toán mã hóa hiện đại, bảo vệ dữ liệu khi truyền tải qua mạng, kể cả khi hệ thống đang hoạt động qua Internet công cộng.

2.5.2. Triển khai WireGuard trong Hệ Thống IoT.

Để bảo mật giao tiếp giữa các thành phần (MQTT Broker, Flask Server và Client), ta triển khai WireGuard như sau:

* **Trên Server:** Cài đặt WireGuard trên máy chủ chạy Flask và MQTT Broker.
* **Trên Client:** Cài đặt WireGuard trên máy chủ dựa trên hệ điều hành hiện tại.

Cấu hình WireGuard:

* **Tạo khóa bảo mật:** Trên cả server và client, tạo cặp khóa riêng và khóa công khai.
* **Cấu hình trên server:**

**A computer screen with white text

AI-generated content may be incorrect.**

Hình 2.2. Cấu hình trên server.

* **Cấu hình trên Client:**

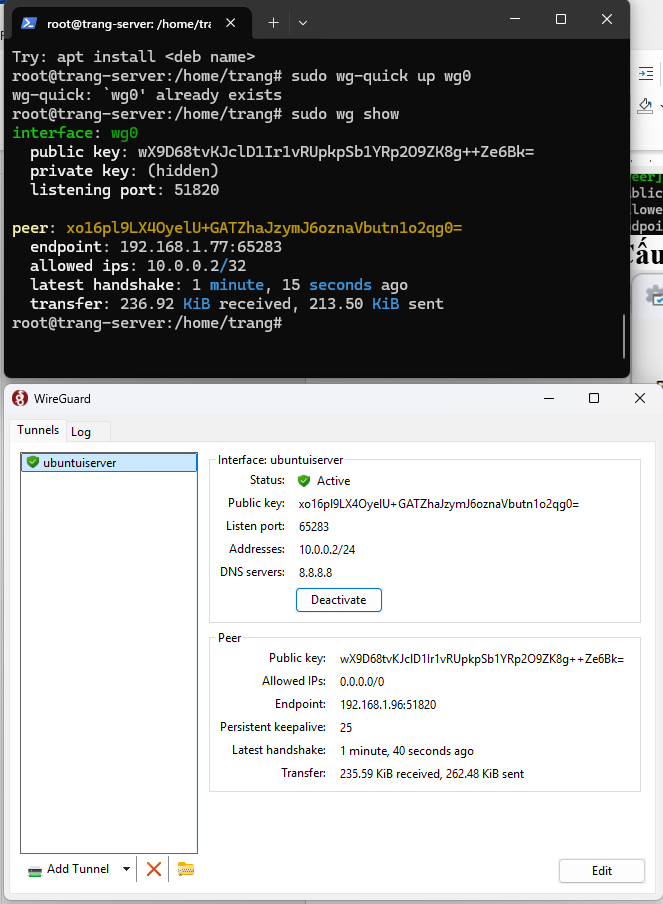
**A screenshot of a computer

AI-generated content may be incorrect.**

Hình 2.3. Cấu hình trên Client.

**2.5.3. Kết nối và kiểm tra.**

Để kiểm tra kết nối thì chỉ cần active cấu hình và thử ping đến địa chỉ của Server.



Hình 2.4. Kiểm tra kết nối từ Server đã thấy Client kết nối đến.

Việc tích hợp WireGuard giúp hệ thống IoT của bạn có một lớp bảo mật bổ sung bằng cách mã hóa toàn bộ lưu lượng truyền tải và ngăn chặn truy cập trái phép từ bên ngoài. Điều này đặc biệt quan trọng khi dữ liệu được truyền qua các kênh mạng công cộng, tạo nên một hệ thống giám sát và điều khiển tin cậy và an toàn.

# CHƯƠNG 3: THỬ NGHIỆM HỆ THỐNG

## 3.1. Giao diện giám sát và điều khiển.

Screens screenshot of a graph

AI-generated content may be incorrect.

Hình 3.1. Giao diện giám sát và điều khiển.

Chạy chương trình ở Server và tiến hành quá trình thử nghiệm:

* Mở trang chủ trên trình duyệt <Địa chỉ ip của Server>:5000.
* Quan sát biểu đồ cập nhật liên tục với dữ liệu cảm biến.

A graph with red and blue lines

AI-generated content may be incorrect.

Hình 3.2. Biểu đồ hiển thị giá trị nhiệt độ và độ ẩm mới nhất.

* Thực hiện các thao tác điều khiển (bấm nút bật/tắt, thay đổi ngưỡng) và kiểm tra phản hồi từ hệ thống thông qua giao diện.

Giao diện hiển thị dữ liệu mượt mà, cập nhật liên tục theo chu kỳ đã định (2 giây); các lệnh điều khiển được xử lý nhanh chóng qua API /control.

## 3.2. Quản lý dữ liệu.

Mục tiêu:

* Đánh giá quá trình lưu trữ dữ liệu vào SQLite và thuật toán giảm tải khi đẩy dữ liệu lên Firebase.

Quá trình thử nghiệm:

* Gửi dữ liệu cảm biến qua MQTT và kiểm tra việc ghi nhận các bản ghi vào SQLite.

A screen shot of a graph

AI-generated content may be incorrect.

Hình 3.3. Lịch sử được lấy trực tiếp từ SQLite.

* Quan sát log hệ thống để xác minh rằng hàm push\_to\_firebase chỉ đẩy dữ liệu khi có sự thay đổi vượt qua ngưỡng (ví dụ: nhiệt độ thay đổi ≥ 0.5°C, độ ẩm thay đổi ≥ 2% hoặc trạng thái relay thay đổi).

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.4. Các bản ghi trên Firebase được gửi lên từ Server.

Dữ liệu được lưu trữ chính xác, đầy đủ và có thể truy xuất theo khoảng thời gian. Thuật toán giảm tải hoạt động hiệu quả, giúp tránh ghi dữ liệu không cần thiết lên Firebase.

## 3.3. Bảo mật.

Mục tiêu:

* Kiểm tra hiệu quả của bảo mật sử dụng WireGuard VPN trong việc mã hóa và bảo vệ giao tiếp giữa các thành phần hệ thống.

Quá trình thử nghiệm:

* Thiết lập WireGuard trên server và client/gateway theo cấu hình đã định.
* Kiểm tra kết nối VPN qua lệnh wg và theo dõi log để xác minh rằng các kết nối đã được mã hóa.
* Thực hiện các bài test đơn giản (ví dụ: cố gắng truy cập từ một IP không nằm trong VPN) để đảm bảo chỉ có các thiết bị xác thực mới có thể giao tiếp được với server.

Đánh giá: Hệ thống giao tiếp qua VPN được mã hóa toàn bộ, đảm bảo dữ liệu không bị rò rỉ và chỉ các thiết bị đã cấu hình đúng mới được truy cập.

## 3.4. Kết luận.

Hệ thống thông gió đã làm được:

* Hệ thống IoT đã hoạt động ổn định với giao diện giám sát realtime và khả năng điều khiển relay hiệu quả.
* Dữ liệu cảm biến được lưu trữ chính xác vào SQLite và đẩy lên Firebase một cách hợp lý nhờ thuật toán giảm tải.
* Việc tích hợp bảo mật bằng WireGuard tạo ra một kênh an toàn cho giao tiếp giữa các thành phần hệ thống.

Điểm mạnh:

* Giao diện trực quan, dễ sử dụng với cập nhật dữ liệu liên tục.
* Quá trình lưu trữ dữ liệu hoạt động hiệu quả, cho phép truy xuất lịch sử đầy đủ.
* Bảo mật mạnh mẽ với WireGuard, đảm bảo an toàn thông tin khi truyền tải qua mạng.

Những điểm chưa tốt và cần cải tiến:

* Tối ưu hiệu suất cập nhật dữ liệu trên giao diện để giảm độ trễ.
* Cải thiện thông báo, phản hồi lệnh điều khiển trên giao diện để tăng trải nghiệm người dùng.
* Tinh chỉnh thuật toán lọc dữ liệu để xử lý tốt hơn các biến động nhỏ không cần thiết.
* Phát triển thêm các chức năng bảo mật nâng cao, tự động hóa quản lý khóa để dễ dàng mở rộng khi số lượng thiết bị tăng.
* Hệ thống và thuật toán chỉ áp dụng cho một dự án nhỏ, cần cải tiến để có thể hoạt động ở một dự án lớn hơn.