HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO BÀI TẬP LỚN

Môn học: IoT và ứng dụng

Giảng viên hướng dẫn: Nguyễn Quốc Uy

Họ và tên sinh viên: Hoàng Văn Minh Hiếu

Mã sinh viên: B21DCCN051

Số điện thoại: 0972165893

Hà Nội, 2024

MỤC LỤC

DANH MỤC HÌNH VẼ

I. Giới thiệu

1.1. IoT (Internet of Things - Mạng lưới vạn vật kết nối)

1.1.1. Khái niệm

Internet vạn vật (IoT) là một hệ thống kết nối trong đó các thiết bị và đối tượng thông minh có khả năng giao tiếp và trao đổi dữ liệu với nhau thông qua internet. Những thiết bị này không chỉ đơn thuần kết nối với mạng internet mà còn có thể tự động gửi và nhận thông tin, tương tác với nhau một cách linh hoạt mà không cần đến sự can thiệp trực tiếp của con người. Điều này giúp tạo ra một môi trường thông minh, nơi các thiết bị có thể điều khiển, giám sát và tối ưu hóa hoạt động của chúng một cách tự động và hiệu quả, từ đó nâng cao trải nghiệm và hiệu suất trong nhiều lĩnh vực khác nhau như nhà thông minh, y tế, giao thông và công nghiệp.

1.1.2. Các thành phần của IoT

- Thiết bị IoT: Bao gồm các cảm biến và thiết bị thông minh, có khả năng thu thập, gửi và nhận dữ liệu từ môi trường xung quanh.
- Mạng: Đảm bảo truyền tải dữ liệu từ các thiết bị IoT đến hệ thống xử lý thông qua các giao thức như Wi-Fi, Bluetooth, 4G/5G, hoặc mạng LPWAN.
- Hệ thống xử lý dữ liệu: Gồm các máy chủ và nền tảng đám mây, nơi dữ liệu được lưu trữ, phân tích và xử lý để đưa ra quyết định.
- Ứng dụng và dịch vụ: Các phần mềm, ứng dụng và bảng điều khiển giúp người dùng quản lý, giám sát và điều khiển các thiết bị IoT, tạo ra giá trị và cải thiện trải nghiệm người dùng.

1.1.3. Ứng dụng

IoT có thể được áp dụng trong nhiều lĩnh vực khác nhau, bao gồm:

- Nhà thông minh: Điều khiển ánh sáng, nhiệt độ, an ninh, thiết bị gia đình thông qua

điện thoại di động.

- Công nghiệp: Giám sát và quản lý các quy trình sản xuất, dự báo bảo trì thiết bị, tăng cường an toàn lao động.
- Giao thông: Điều khiển giao thông thông minh, quản lý đỗ xe, theo dõi vận chuyển hàng hóa.
- Y tế: Theo dõi sức khỏe, quản lý thuốc, giám sát bệnh nhân từ xa.
- Năng lượng: Theo dõi và quản lý tiêu thụ năng lượng, tối ưu hóa sử dụng tài nguyên.

1.2. Mô tả và mục đích của đề tài

1.2.1. Mô tả đề tài

Đề tài này tập trung vào việc xây dựng một hệ thống IoT để giám sát và điều khiển các yếu tố môi trường như nhiệt độ, độ ẩm và các thiết bị điện (điều hòa, đèn, quạt) thông qua một ứng dụng website. Hệ thống được phát triển với backend sử dụng Java Spring Boot, frontend sử dụng ReactJs và cơ sở dữ liệu MySQL để lưu trữ và quản lý dữ liệu. Các cảm biến sẽ thu thập dữ liệu nhiệt độ, độ ẩm và ánh sáng trong thời gian thực, và người dùng có thể điều khiển các thiết bị điện từ xa thông qua giao diện website.

1.2.2. Mục đích của đề tài

Mục đích của đề tài là xây dựng một hệ thống quản lý và điều khiển thông minh với các mục tiêu chính như sau:

- Giám sát và cảnh báo:
    - Thu thập và hiển thị dữ liệu nhiệt độ, độ ẩm từ các cảm biến một cách trực quan trên website.
    - Cảnh báo người dùng khi các chỉ số môi trường vượt quá ngưỡng an toàn (ví dụ: nhiệt độ quá cao hoặc độ ẩm quá thấp).
- Điều khiển thiết bị từ xa:

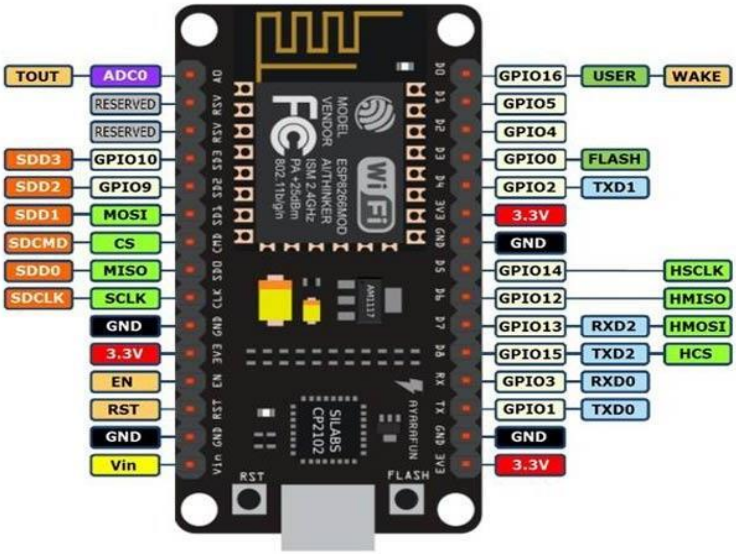
- Cho phép người dùng bật/tắt và điều chỉnh các thiết bị như điều hòa, đèn, quạt thông qua giao diện website.

- Quản lý và lưu trữ dữ liệu:
    - Lưu trữ dữ liệu giám sát (nhiệt độ, độ ẩm) và lịch sử hoạt động của các thiết bị vào cơ sở dữ liệu MySQL.
    - Cung cấp các báo cáo thống kê, biểu đồ phân tích dữ liệu giúp người dùng nắm bắt tình trạng hoạt động của hệ thống.

1.3. Thiết bị sử dụng

1.3.1. Node MCU ESP 8266

- Thông số kỹ thuật:
    - Hỗ trợ chuẩn 802.11 b/g/n.
    - Wi-Fi 2.4 GHz, hỗ trợ WPA/WPA2.
    - Chuẩn điện áp hoạt động: 3.3V.
    - Chuẩn giao tiếp nối tiếp UART với tốc độ Baud lên đến 115200.
    - Có 3 chế độ hoạt động: Client, Access Point, Both Client and Access Point.
    - Hỗ trợ các chuẩn bảo mật như: OPEN, WEP, WPA_PSK, WPA2_PSK, WPA WPA2 PSK.
    - Hỗ trợ cả 2 giao tiếp TCP và UDP.



Hình 1: MCU ESP8266

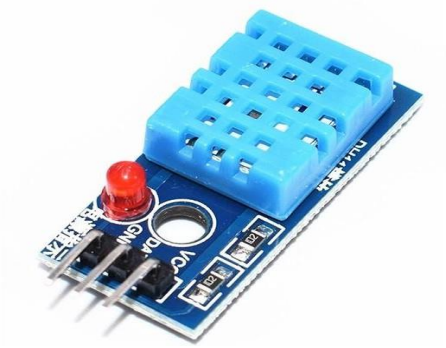
- Các chân ESP 8266:

  - VIN: Chân nguồn cung cấp cho board, có thể được kết nối với nguồn cung cấp từ 5V đến 12V.
  - GND: Chân mẫu điện âm.
  - 3V3: Chân nguồn cung cấp 3.3V.
  - RST: Chân khởi động lại board.
  - AO: Chân đo lường tín hiệu Analog-to-Digital Converter (ADC), được sử dụng để đo các tín hiệu điện áp tương tự.
  - DO - D8: Các chân kết nối đầu vào/số GPIO từ 0 đến 8, được sử dụng để điều khiển các thiết bị hoặc đọc các tín hiệu từ các cảm biến.
  - TXD: Chân truyền UART, được sử dụng để truyền dữ liệu từ board đến một thiết bị khác.

- RXD: Chân nhận UART, được sử dụng để nhận dữ liệu từ một thiết bị khác.
- CH_PD: Chân đưa board ra khỏi chế độ nghỉ, và bắt đầu chay.
- USB: Cổng USB được sử dụng để kết nối board với máy tính để lập trình và cung cấp nguồn.

1.3.2. Cảm biến nhiệt độ - độ ẩm

- Thông số kỹ thuật:
  - Điện áp hoạt động: 3V - 5V DC
  - Dòng điện tiêu thụ: 2.5mA
  - Phạm vi cảm biến độ ẩm: 20% - 90% RH, sai số ±5%RH
  - Phạm vi cảm biến nhiệt độ: 0°C ~ 50°C, sai số ±2°C
  - Tần số lấy mẫu tối đa: 1Hz (1 giây 1 lần)
  - Kích thước: 23 * 12 * 5 mm

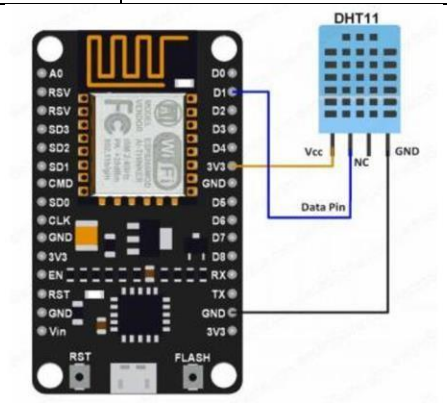


Hình 2: Cảm biến DHT11

- Sơ đồ chân DHT11:

Số chân	Tên chân	Mô tả

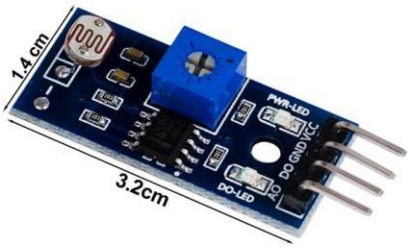
1	VCC	Nguồn 3.3V đến 5V
2	Data	Đầu ra nhiệt độ độ ẩm thông qua dữ liệu nối tiếp
3	NC	Không có kết nối và do đó không sử dụng
4	Ground	Nối đất



Hình 3: Sơ đồ chân nối DHT11

1.3.3. Cảm biến cường độ ánh sáng quang trở

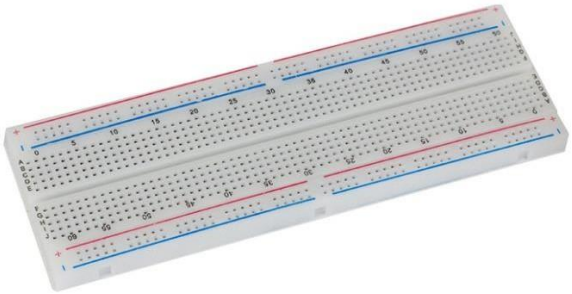
- Thông số kỹ thuật
  - Điện áp hoạt động: 3.3V – 5V
  - Kết nối 4 chân với 2 chân cấp nguồn (VCC và GND) và 2 chân tín hiệu ngõ ra (AO và DO).
  - Hỗ trợ cả 2 dạng tín hiệu ra Analog và TTL. Ngõ ra Analog 0– 5V tỷ lệ thuận với cường độ ánh sáng, ngõ TTL tích cực mức thấp.
  - Độ nhạy cao với ánh sáng được tùy chỉnh bằng biến trở .
  - Kích thước: 32 x 14mm



Hình 4: Cảm biến ánh sáng

1.3.4. Board Test MB-102 16.5x5.5

Dùng để test mạch trước khi làm mạch in hoàn chỉnh. Giúp bạn dễ dàng thực hiện các mạch điện thực tế trước khi hàn trực tiếp linh kiện lên board mạch đồng



Hình 5: Board test

1.3.5. Điện trở vạch 1/4W sai số 5% 250V 1R-10M

Hình 6: Điện trở

1.3.6. Dây nối chân các thiết bị

Hình 7: Dây nối

1.3.7. Led 5mm, 7 màu 2 chân (1,5-3V)

Hình 8: Led

1.4. Công nghệ sử dụng

1.4.1. Trình biên dịch Arduino IDE

- Arduino IDE là một môi trường phát triển tích hợp được sử dụng để viết và nạp mã cho các vi điều khiển Arduino. Đây là công cụ chính để lập trình các thiết bị phần cứng, cho phép giao tiếp với các cảm biến (như nhiệt độ, độ ẩm) và thiết bị điều khiển (như quạt, đèn, điều hòa).
- Tính năng chính:
    - Hỗ trợ các thư viện và hàm đặc biệt dành riêng cho lập trình IoT.
    - Dễ dàng nạp mã cho các bo mạch Arduino thông qua cổng USB.

- Giao diện đơn giản, dễ sử dụng cho việc lập trình và kiểm tra mã nguồn.

1.4.2. Frontend: ReactJS

- ReactJS là một thư viện JavaScript mạnh mẽ do Facebook phát triển, được sử dụng để xây dựng giao diện người dùng động và tương tác. ReactJS giúp tạo ra các ứng dụng web hiệu suất cao với khả năng giám sát và điều khiển thiết bị dễ dàng.
- Tính năng chính:
    - Component-based Architecture: Giao diện được chia thành các component nhỏ, có thể tái sử dụng, dễ dàng quản lý, bảo trì và mở rộng.
    - Hooks: Cung cấp các API như useState, useEffect giúp quản lý trạng thái và vòng đời của component một cách linh hoạt.
    - Virtual DOM: Tối ưu hóa hiệu suất bằng cách chỉ cập nhật những thay đổi cần thiết trên giao diện, giảm thiểu thao tác với DOM thật.
    - Two-way Data Binding (Controlled Components): Dữ liệu được đồng bộ hóa giữa giao diện và logic ứng dụng thông qua các component có kiểm soát.
    - React Router: Hỗ trợ quản lý điều hướng trong ứng dụng SPA (Single Page Application), giúp tạo ra trải nghiệm người dùng mượt mà.
    - HTTP Client (Axios/Fetch API): Tích hợp các phương thức giao tiếp với backend qua RESTful API, giúp lấy và cập nhật dữ liệu từ server.

1.4.3. Backend: Spring Boot

- Spring Boot là một framework mạnh mẽ để phát triển các ứng dụng Java, đặc biệt là các ứng dụng web và RESTful API. Spring Boot cung cấp môi trường phát triển nhanh chóng và đơn giản với các tính năng bảo mật và quản lý dữ liệu hiệu quả.
- Tính năng chính:

- RESTful API: Xây dựng các dịch vụ API để giao tiếp với frontend, cung cấp dữ liệu từ cơ sở dữ liệu và thực hiện các chức năng điều khiển thiết bị.
- Spring Security: Cung cấp bảo mật cho ứng dụng với các chức năng xác thực và phân quyền người dùng.
- Spring Data JPA: Tương tác với cơ sở dữ liệu một cách dễ dàng, hỗ trợ các thao tác CRUD (Create, Read, Update, Delete).
- Dependency Injection: Quản lý các thành phần và phụ thuộc trong ứng dụng một cách hiệu quả.

1.4.4. Database: MySQL

- MySQL là một hệ quản trị cơ sở dữ liệu quan hệ phổ biến, được sử dụng để lưu trữ và quản lý thông tin về nhiệt độ, độ ẩm và trạng thái của các thiết bị. MySQL cung cấp khả năng truy vấn và quản lý dữ liệu mạnh mẽ, phù hợp cho các ứng dụng có quy mô vừa và lớn.
- Tính năng chính:
    - Khả năng lưu trữ: Lưu trữ thông tin lịch sử hoạt động của các thiết bị và dữ liệu cảm biến.
    - Quan hệ dữ liệu: Tổ chức dữ liệu theo mô hình quan hệ, đảm bảo tính nhất quán và toàn vẹn dữ liệu.
    - Truy vấn nhanh chóng: Sử dụng các chỉ mục và khóa ngoại để tối ưu hóa hiệu suất truy vấn dữ liệu.
    - Bảo mật dữ liệu: Cung cấp các tính năng quản lý người dùng và phân quyền truy cập để đảm bảo an toàn dữ liệu.

1.4.5. Mosquitto

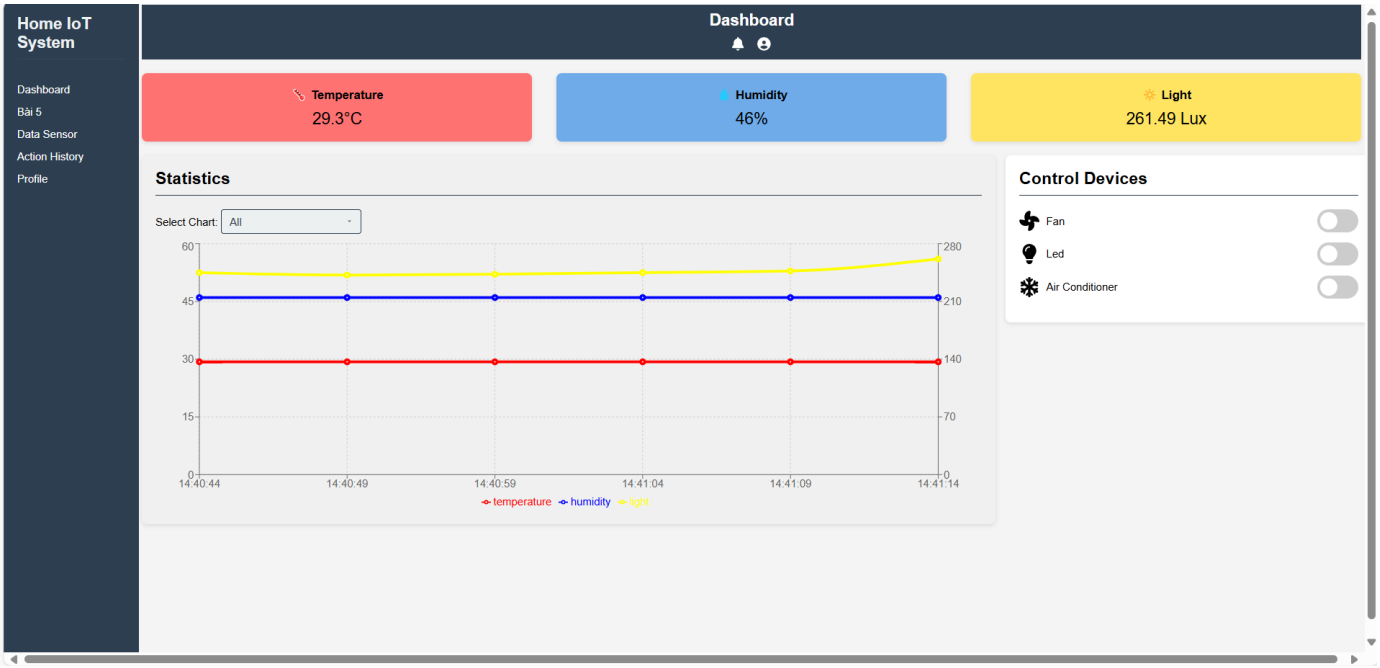
- Mosquitto là một MQTT broker mã nguồn mở, cho phép giao tiếp giữa các thiết bị

IoT với nhau và với server. Nó đảm bảo việc truyền tải thông tin nhanh chóng và ổn định, giúp hệ thống giám sát và điều khiển hoạt động trơn tru.

- Tính năng chính:
    - Giao tiếp MQTT: Hỗ trợ giao thức MQTT để truyền tải thông điệp giữa các thiết bị IoT và ứng dụng server.
    - Chất lượng dịch vụ (QoS): Cung cấp nhiều mức độ QoS để đảm bảo thông điệp được truyền tải một cách tin cậy.
    - Nhẹ và hiệu quả: Tiêu thụ ít tài nguyên hệ thống, phù hợp cho các thiết bị IoT có tài nguyên hạn chế.
    - Bảo mật: Hỗ trợ các cơ chế bảo mật như TLS và xác thực người dùng để đảm bảo an toàn trong việc truyền thông điệp.

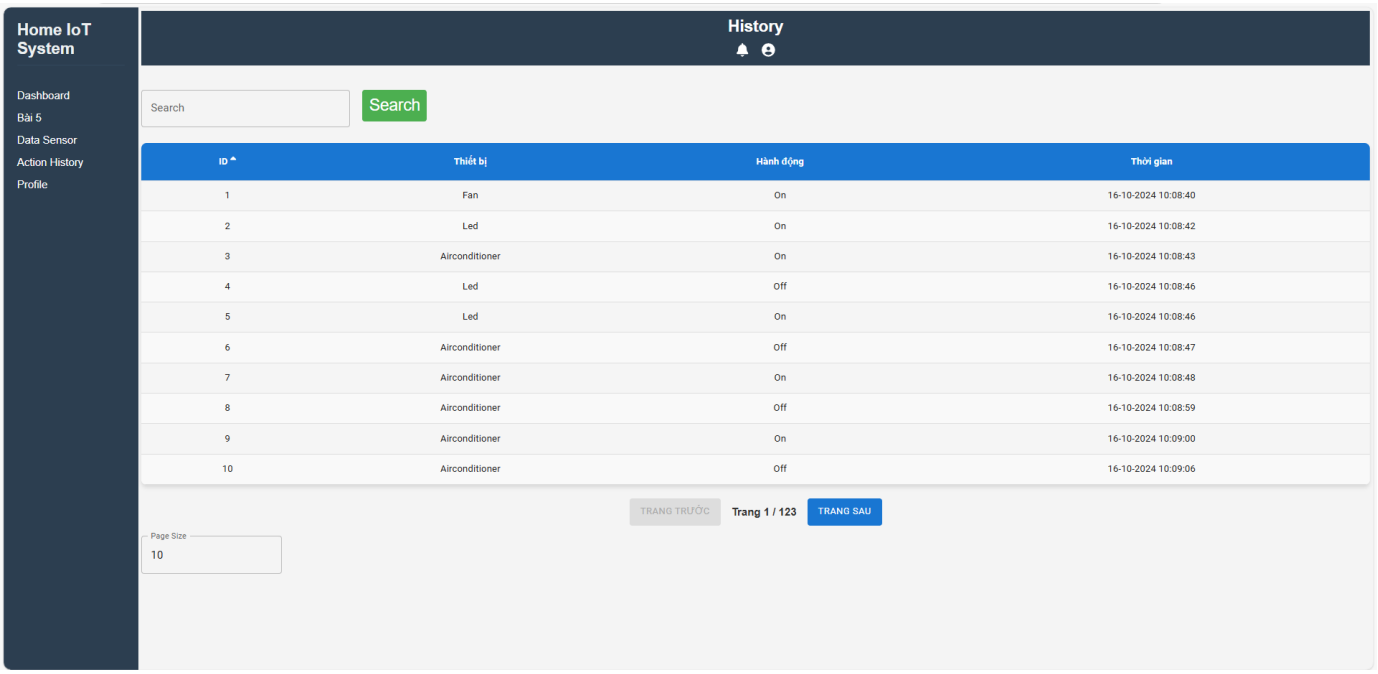
II. Giao diện

2.1. Giao diện website



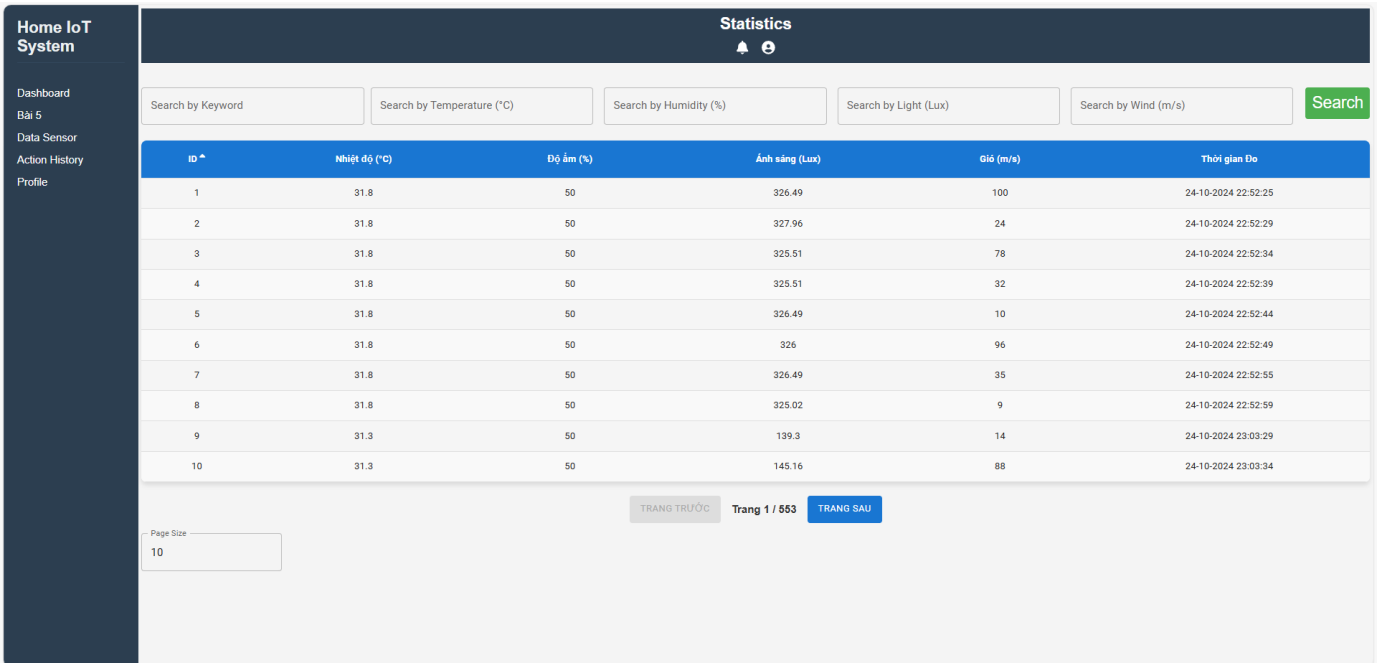
Hình 9: Giao diện Dashboard

- Trang Dashboard hiển thị nhiệt độ, độ ẩm, ánh sáng mà cảm biến đo được, cùng với biểu đồ thống kê 3 giá trị đó theo thời gian. Để điều khiển bật tắt các thiết bị có thể nhấn On/Off



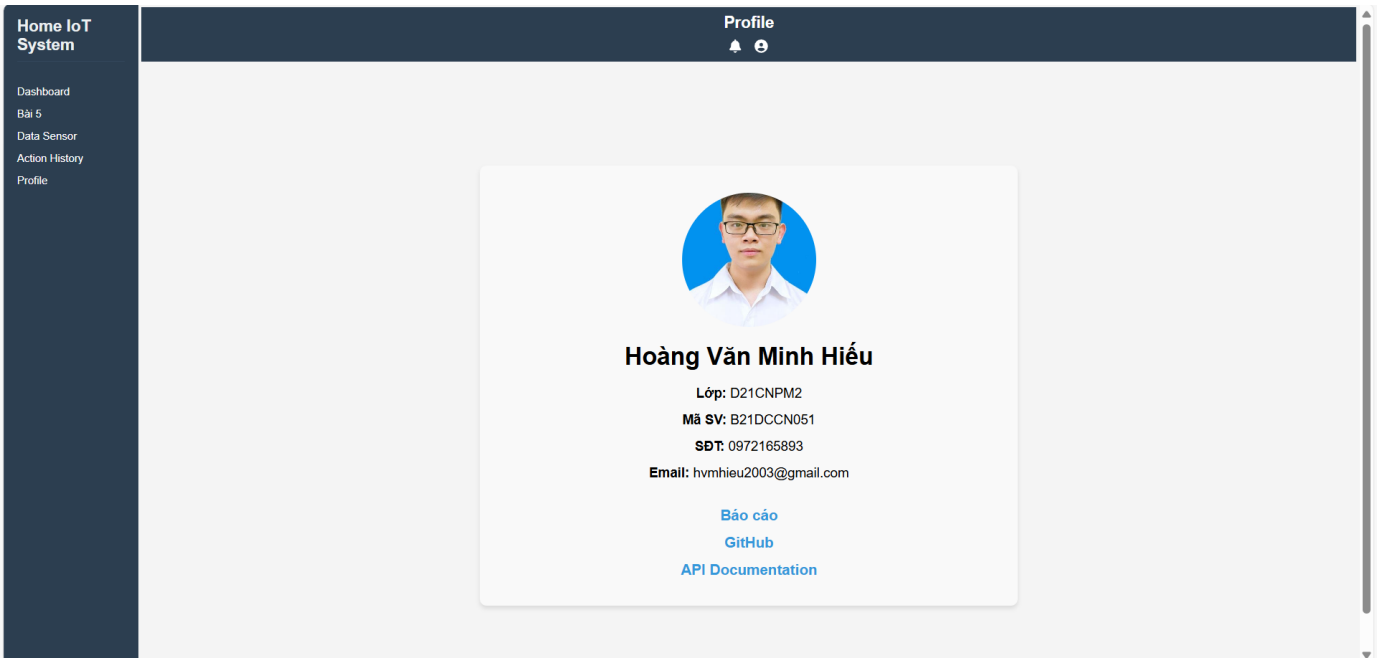
Hình 10: Action History

- Trang Action History hiển thị hoạt động bật tắt của các thiết bị theo thời gian.



Hình 11: Giao diện màn Data Sensor

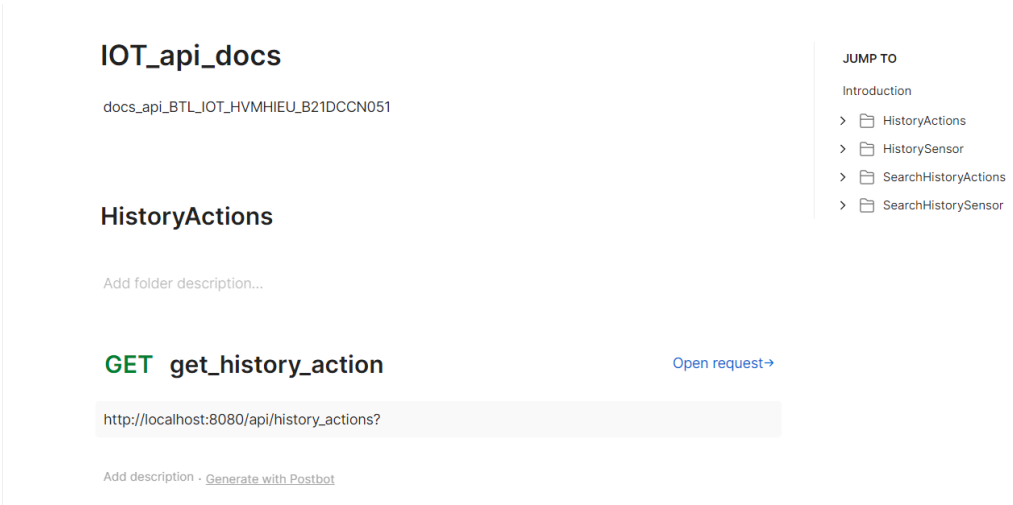
- Trang Data Sensor hiển thị các giá trị nhiệt độ, độ ẩm, ánh sang theo thời gian

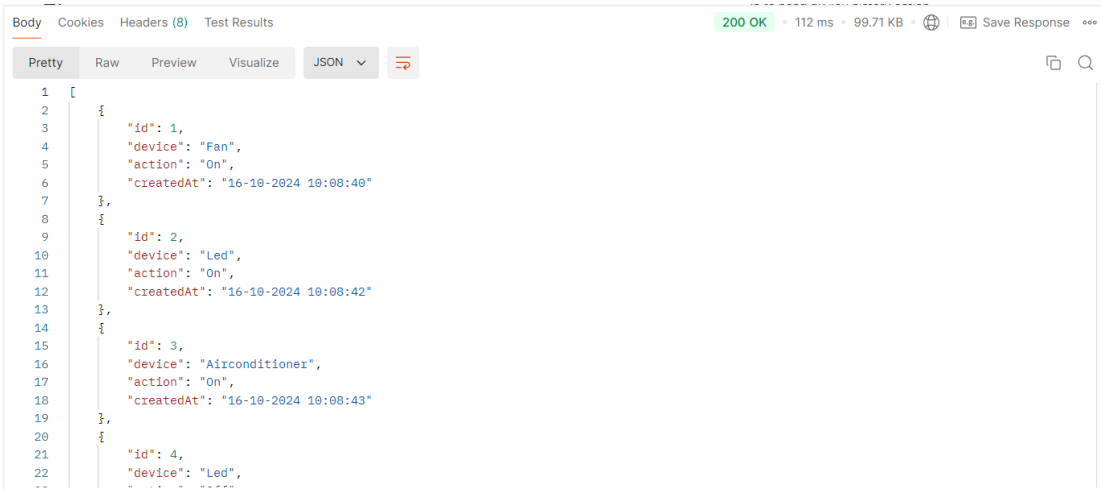


Hình 12: Giao diện màn Profile

- Trang Profile hiển thị thông tin sinh viên và các liên kết theo yêu cầu

2.2. Giao diện API, API Docs



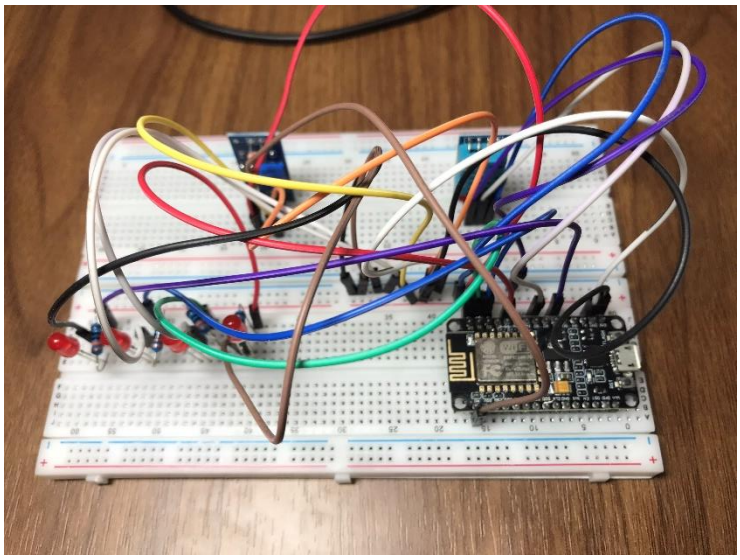


Hình 13: Giao diện API


```
{
  "collection": {
    "info": {
      "_postman_id": "35194488-ddf6-465e-b357-93ff462877d2",
      "name": "IOT_api_docs",
      "description": "docs_api_BTL_IOT_HVMHIEU_B21DCCN051",
      "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
      "updatedAt": "2024-11-13T14:17:28.000Z",
      "createdAt": "2024-10-15T16:14:08.000Z",
      "lastUpdatedBy": "39033111",
      "uid": "39033111-35194488-ddf6-465e-b357-93ff462877d2"
    },
    "item": [
      {
        "name": "HistoryActions",
        "item": [
          {
            "name": "get_history_action",
            "id": "2e09c652-1e00-4064-9570-7e0355d9a32e",
            "protocolProfileBehavior": {
              "disableBodyPruning": true
            },
            "request": {
              "method": "GET",
              "header": [],
              "url": {
                "raw": "http://localhost:8080/api/history_actions?",
                "protocol": "http",
                "host": [
                  "localhost"
                ],
                "port": "8080",
                "path": [
                  "api",
                  "history_actions"
                ],
                "query": [
                  {
                    "key": null,
                    "value": null,
                    "description": "in ra bảng dữ liệu history action",
                    "type": "text"
                  }
                ]
              }
            }
```


Hình 14: Giao diện API Docs

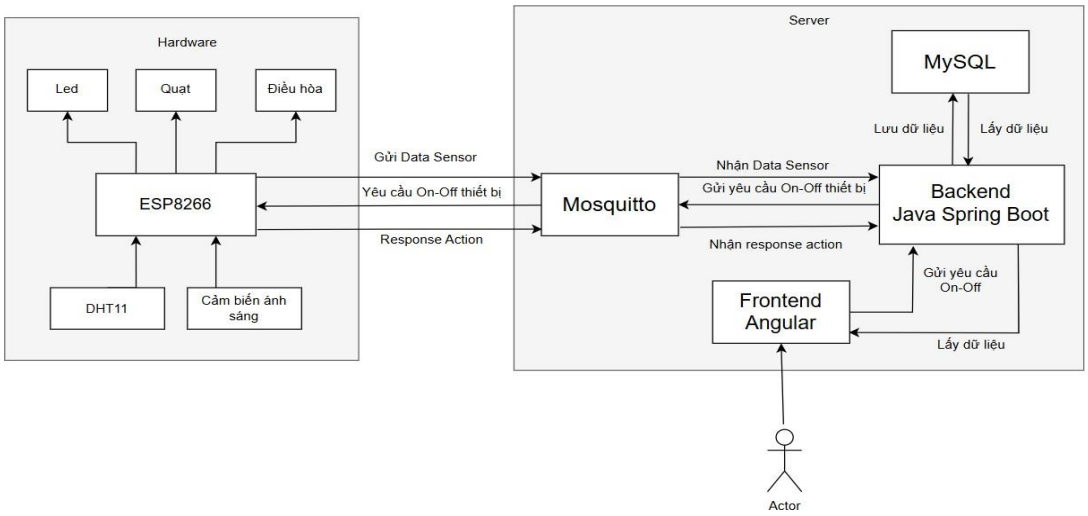
2.3. Giao diện thiết bị



Hình 15: Giao diện thiết bị

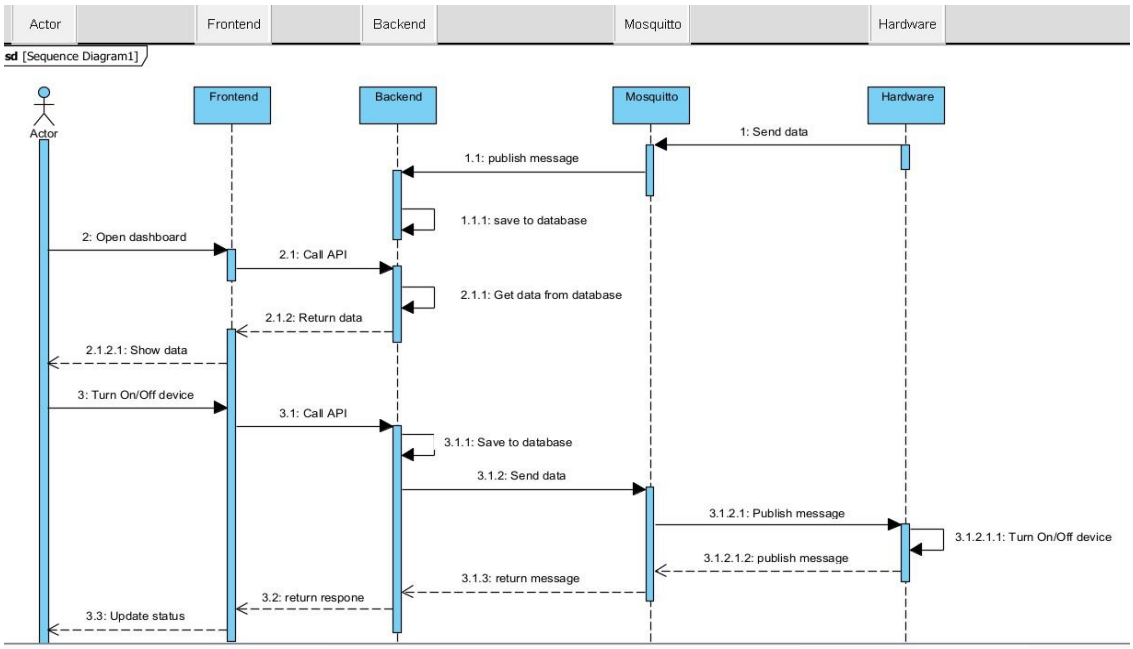
III. Thiết kế chi tiết

3.1. Thiết kế hệ thống



Hình 16: Thiết kế hệ thống

3.2. Sequence Diagram



Hình 17: Sequence Diagram của hệ thống

IV. Code

4.1. Arduino code

```
#include "DHT.h"
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
```

- Khai báo các thư viện cần thiết gồm:
    - DHT.h: Đọc dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT.
    - ESP8266WiFi.h: Kết nối ESP8266 với Wi-Fi.
    - PubSubClient.h: Thư viện hỗ trợ ESP8266 giao tiếp với MQTT broker, giúp gửi và nhận dữ liệu qua các chủ đề (topics).
    - ArduinoJson.h: Thư viện giúp phân tích cú pháp (parse) và tạo dữ liệu JSON, tiện lợi trong việc truyền và nhận dữ liệu có cấu trúc giữa các thiết bị

```
6   #define WIFI_SSID "1506"
7   #define WIFI_PASSWORD "lig1506b"
8
9   // #define WIFI_SSID "iPhone 16 Promax"
10  // #define WIFI_PASSWORD "Minhchau2008"
```

- Định nghĩa các thông tin như tên wifi, mật khẩu wifi mà ESP8266 sẽ kết nối

```
12  // Raspberry Pi Mosquitto MQTT Broker
13  #define MQTT_HOST "172.20.10.6"
14  #define MQTT_PORT 1884
15  #define MQTT_PUB_SENSOR "data_sensor"
16  #define MQTT_SUB_DEVICE_ACTION "device/action"
17  #define MQTT_PUB_WARN "count"
```

- Định nghĩa các thông tin về server của MQTT Broker (ở đây là Mosquitto), tên các topic, port

```
19  #define DHTPIN 14
20  #define LIGHT_SENSOR_PIN A0
21  #define LED1_PIN D2
22  #define FAN_PIN D1
23  #define AIRCONDITIONER_PIN D4
24  #define WARNING_LED_PIN D7  // 
25  #define DHTTYPE DHT11
26
```

- Định nghĩa các chân (pin) mà cảm biến ánh sáng và các thiết bị (quạt và đèn LED) được kết nối trên ESP8266. Cụ thể:
  - Cảm biến ánh sáng được kết nối với chân analog A0 của ESP8266.
  - Chân D1 được sử dụng để điều khiển quạt.
  - Chân D2 được dùng để điều khiển LED 1.
  - Chân D4 được dùng để điều khiển điều hoà.
- Định nghĩa chân kết nối với DHT11 là chân GPI14, chân Data của DHT11 sẽ nối với chân D5 trong mạnh và định nghĩa loại DHT ở đây là DHT11

```
DHT dht(DHTPIN, DHTTYPE);
float temp;
float hum;
int light;
float dustLevel; // Thêm biến độ bụi
int dustCount = 0;
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);

unsigned long previousMillis = 0;
unsigned long warningLedMillis = 0; // Biến lưu thời
const long interval = 5000;
const long blinkInterval = 500; // Thời gian nhấp nhá

bool led1Status = LOW;
bool fanStatus = LOW;
bool airConditionerStatus = LOW;
bool warningLedStatus = LOW;  // Trạng thái của đèn c
bool dustExceedThreshold = false;
bool warningLedBlinking = false;
unsigned long previousBlinkMillis = 0;
```

- Đoạn mã này khai báo các biến và đối tượng cần thiết để làm việc với cảm biến, MQTT, và kết nối Wi-Fi

```
27  #define MQTT_USERNAME "Hoang_Van_Minh_Hieu"
28  #define MQTT_PASSWORD "b21dccn051"
29
```

- Khai báo user, password của mosquitto

```
51
52  void connectToWifi() {
53    Serial.println("Connecting to Wi-Fi...");
54    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
55  }
```

- Định nghĩa các hàm để quản lý kết nối Wi-Fi cho ESP8266:
    - connectToWifi(): Hàm kết nối ESP8266 với Wi-Fi dựa trên SSID và mật khẩu được cung cấp.

```
void reconnectToMqtt() {
  while (!mqttClient.connected()) {
    Serial.println("Connecting to MQTT...");
    if (mqttClient.connect("ESP8266Client", MQTT_USERNAME, MQTT_PASSWORD)) {
      Serial.println("Connected to MQTT.");
      mqttClient.subscribe(MQTT_SUB_DEVICE_ACTION);
    } else {
      Serial.print("Failed to connect to MQTT, rc=");
      Serial.print(mqttClient.state());
      delay(2000);
    }
  }
}
```

- Định nghĩa các hàm để quản lý kết nối MQTT cho ESP8266, bao gồm kết nối, đăng ký chủ đề (topic) và xử lý mất kết nối:

- reconnectToMqtt(): Hàm đảm bảo kết nối lại với MQTT broker nếu bị ngắt, đồng thời đăng ký lắng nghe các thông điệp từ topic đã chỉ định.

```
// Callback xử lý message từ MQTT
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.printf("Message arrived [%s]: ", topic);

  if (strcmp(topic, MQTT_SUB_DEVICE_ACTION) == 0) {
    Serial.println("Received message for device/action");

    // Phân tích JSON từ payload
    DynamicJsonDocument doc(256);
    String message;
    for (int i = 0; i < length; i++) {
      message += (char)payload[i];
    }
    deserializeJson(doc, message);

    // Xử lý thiết bị
    if (doc.containsKey("device")) {
      String deviceName = doc["device"];
      String status = doc["status"];

      if (deviceName == "Led") {
        led1Status = (status == "On") ? HIGH : LOW;
        digitalWrite(LED1_PIN, led1Status);
      } else if (deviceName == "Fan") {
        fanStatus = (status == "On") ? HIGH : LOW;
        digitalWrite(FAN_PIN, fanStatus);
      } else if (deviceName == "Airconditioner") {
        airConditionerStatus = (status == "On") ? HIGH : LOW;
        digitalWrite(AIRCONDITIONER_PIN, airConditionerStatus);
      } else if (deviceName == "WarningLed") {
        warningLedStatus = (status == "On") ? HIGH : LOW;
        digitalWrite(WARNING_LED_PIN, warningLedStatus);
        warningLedBlinking = false; // Tắt chế độ nhấp nháy khi nhận lệnh bật/tắt
      } else if (deviceName == "WarningLedBlink") {
        warningLedBlinking = (status == "On"); // Bật chế độ nhấp nháy khi nhận lệnh "Blink"
      }

    }
  } else {
```

- Hàm callback() xử lý khi một thông điệp MQTT mới đến từ một chủ đề (topic) cụ thể.

Các bước hoạt động chính của hàm:

- In thông báo về thông điệp mới: In ra chủ đề (topic) và nội dung thông điệp nhận được.
- Phân tích JSON từ payload: Chuyển đổi payload nhận được thành đối tượng JSON để truy xuất dữ liệu.
- Xử lý thiết bị và trạng thái: Tách thông tin về thiết bị và trạng thái từ thông điệp JSON để điều khiển các thiết bị như đèn, quạt, điều hòa, và đèn cảnh báo.

```
if (deviceName == "Led") {
  led1Status = (status == "On") ? HIGH : LOW;
  digitalWrite(LED1_PIN, led1Status);
} else if (deviceName == "Fan") {
  fanStatus = (status == "On") ? HIGH : LOW;
  digitalWrite(FAN_PIN, fanStatus);
} else if (deviceName == "Airconditioner") {
  airConditionerStatus = (status == "On") ? HIGH : LOW;
  digitalWrite(AIRCONDITIONER_PIN, airConditionerStatus);
} else if (deviceName == "WarningLed") {
  warningLedStatus = (status == "On") ? HIGH : LOW;
  digitalWrite(WARNING_LED_PIN, warningLedStatus);
  warningLedBlinking = false; // Tắt chế độ nhấp nháy khi nhậ
} else if (deviceName == "WarningLedBlink") {
  warningLedBlinking = (status == "On"); // Bật chế độ nhấp n
}
```

- Kiểm tra loại thiết bị và trạng thái để bật/ tắt thiết bị trên hardware

```
void setup() {
  Serial.begin(115200);
  Serial.println();

  dht.begin();

  pinMode(LED1_PIN, OUTPUT);
  pinMode(FAN_PIN, OUTPUT);
  pinMode(AIRCONDITIONER_PIN, OUTPUT);
  pinMode(WARNING_LED_PIN, OUTPUT);  // Đặt chân đèn cảnh báo làm OUTPUT

  mqttClient.setServer(MQTT_HOST, MQTT_PORT);
  mqttClient.setCallback(callback);

  connectToWifi();
}
```

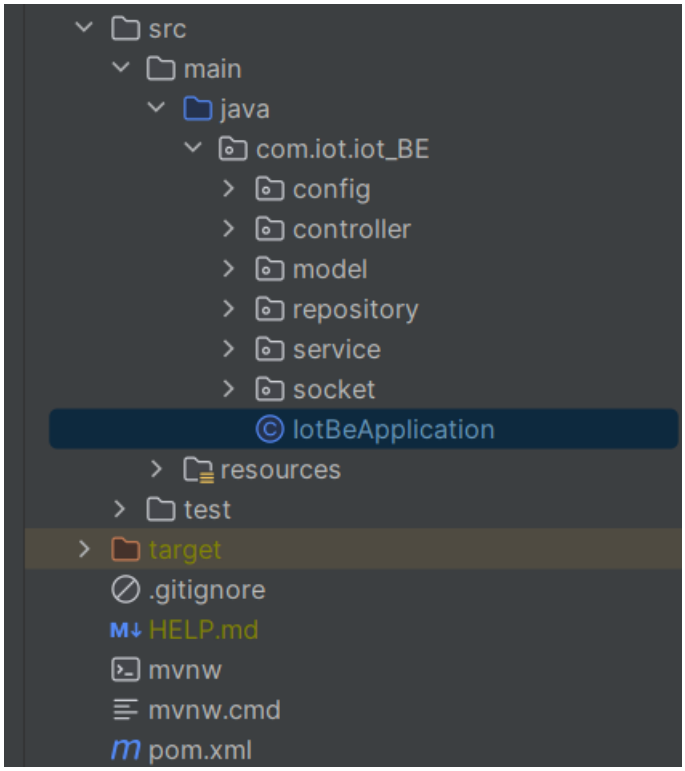
- Hàm setup() cấu hình các thiết lập cần thiết cho ESP8266 khi khởi động. Các bước thực hiện:

- Khởi tạo Serial: Bắt đầu giao tiếp serial với tốc độ 115200 để in thông báo debug.
- Khởi động cảm biến DHT: Khởi động cảm biến DHT để bắt đầu thu thập dữ liệu nhiệt độ và độ ẩm.
- Thiết lập chân điều khiển: Đặt các chân (pin) cho LED, quạt, điều hòa và đèn cảnh báo làm đầu ra.
- Cấu hình MQTT: Thiết lập máy chủ MQTT và chỉ định hàm callback xử lý thông điệp đến từ MQTT.
- Kết nối Wi-Fi: Bắt đầu kết nối ESP8266 với mạng Wi-Fi thông qua hàm connectToWifi().

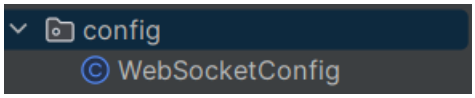
```
void loop() {
  if (!mqttClient.connected()) {
    reconnectToMqtt();
  }
  mqttClient.loop();

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Đọc dữ liệu từ cảm biến
    hum = dht.readHumidity();
    temp = dht.readTemperature();
    int analogValue = analogRead(LIGHT_SENSOR_PIN);
    float light = 500 - (analogValue / 1023.0) * 500;

    // Tạo giá trị độ bụi ngẫu nhiên (giả sử từ 0 đến 100%).
    dustLevel = random(0, 101);

    // Làm tròn giá trị nhiệt độ, độ ẩm, ánh sáng và độ bụi đến 2 chữ số thập phân
    float roundedTemp = round(temp * 100) / 100.0;
    float roundedHum = round(hum * 100) / 100.0;
    float roundedLight = round(light * 100) / 100.0;
    float roundedDust = round(dustLevel * 100) / 100.0;

    // Tạo đối tượng JSON chứa dữ liệu cảm biến
    DynamicJsonDocument doc(1024);
    doc["temperature"] = roundedTemp;
    doc["humidity"] = roundedHum;
    doc["light"] = roundedLight;
    doc["dust"] = roundedDust; // Thêm dữ liệu về độ bụi

    // Gửi dữ liệu lên MQTT broker
    String jsonData;
    serializeJson(doc, jsonData);
    mqttClient.publish(MQTT_PUB_SENSOR, jsonData.c_str());
    Serial.print("Published: ");
    Serial.println(jsonData);
  }
```

- Vòng lặp chính của chương trình sẽ liên tục đọc giá trị từ các cảm biến và ghép thành dạng Json rồi publish cho BE.

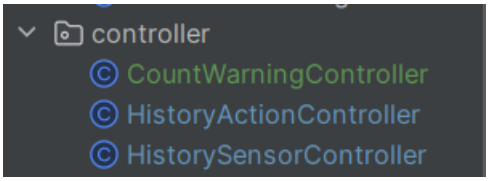
4.2. Backend code



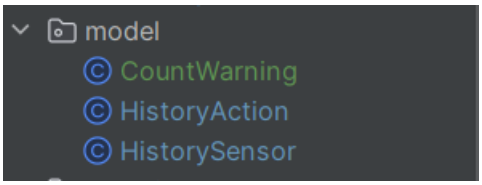
Cấu trúc thư mục của backend gồm:



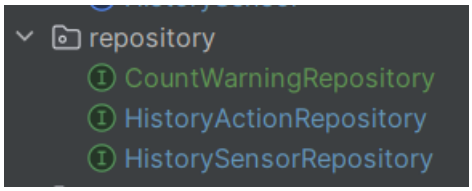
- Config: Chứa các cấu hình chung của ứng dụng, như cấu hình bảo mật, kết nối cơ sở dữ liệu, cấu hình MQTT, v.v.



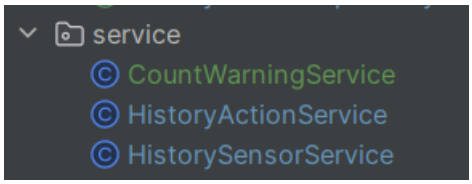
- Controller: Chứa các lớp điều khiển (controller) xử lý các yêu cầu HTTP từ phía người dùng, nhận và trả dữ liệu.



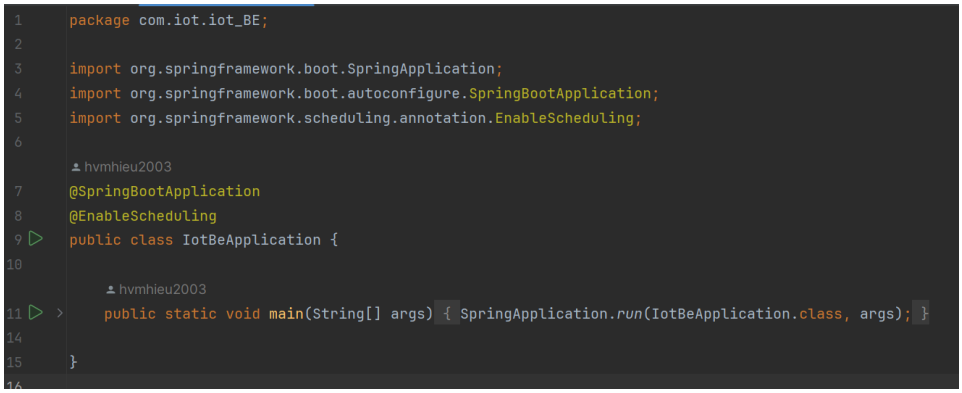
- Model: Chứa các lớp biểu diễn các thực thể (entity) tương ứng với các bảng trong cơ sở dữ liệu.



- Repository: Chứa các lớp giao tiếp với cơ sở dữ liệu, thường là các interface mở rộng JpaRepository để thực hiện các thao tác CRUD.



- Service: Chứa các lớp xử lý nghiệp vụ chính của ứng dụng, kết nối và xử lý dữ liệu từ repository và controller.



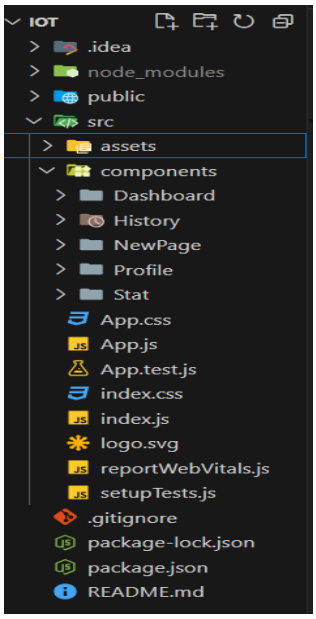
```
package com.iot.iot_BE;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

hvmhieu2003
@SpringBootApplication
@EnableScheduling
public class IotBeApplication {

    hvmhieu2003
    public static void main(String[] args) { SpringApplication.run(IotBeApplication.class, args); }

}
```

- IotBeApplication.java: Lớp chính của ứng dụng Spring Boot, nơi chứa phương thức main để khởi động ứng dụng.

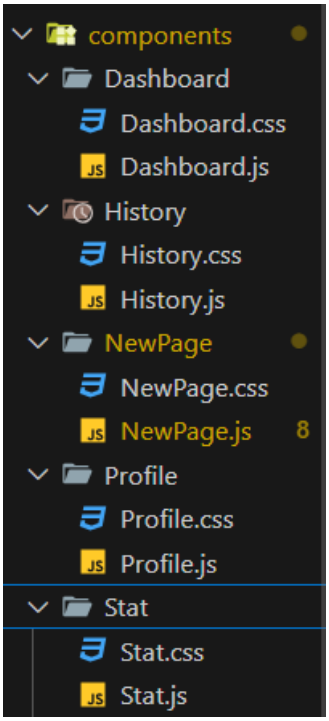
```java
@Service
public class MqttService {
    16 usages
    private static final Logger logger = LoggerFactory.getLogger(MqttService.class);
    1 usage
    @Autowired
    private HistorySensorRepository historySensorRepository;
    1 usage
    @Autowired
    private HistoryActionRepository historyActionRepository;
    1 usage
    @Autowired
    private CountWarningService countWarningService;
    7 usages
    private MqttClient client;
    no usages  hvmhieu2003 +1 *
    public MqttService() throws MqttException {
        // Tạo kết nối MQTT
        client = new MqttClient( serverURI: "tcp://192.168.1.49:1884", MqttClient.generateClientId());
        MqttConnectOptions options = new MqttConnectOptions();
        options.setUserName("Hoang_Van_Minh_Hieu");
        options.setPassword("b21dccn051".toCharArray());
        client.connect(options);
```

- MqttService.java: Chứa logic để xử lý các thông điệp nhận được từ máy chủ MQTT, có thể bao gồm việc đăng ký chủ đề và xử lý dữ liệu nhận được.

4.3. Frontend Code



Cấu trúc thư mục như sau:



- Dashboard, History, Profile, Stat: Các thư mục này chứa các thành phần (components) tương ứng với các tính năng cụ thể của trang. Mỗi thư mục có các tệp .js và .css để quản lý giao diện và logic của từng thành phần.

```jsx
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import './App.css';
import Dashboard from './components/Dashboard/Dashboard';
import Stat from './components/Stat/Stat'; // Import component Stat
import Profile from './components/Profile/Profile'; // Giả sử bạn có component Profile
import History from './components/History/History'; // Giả sử bạn có component History
import NewPage from './components/NewPage/NewPage';
function App() {
  return (
    <Router>

      <div className="App">
        <Routes>
          <Route path="/" exact element={<Dashboard />} />
          <Route path="/dashboard" element={<Dashboard />} />
          <Route path="/newpage" element={<NewPage />} />
          <Route path="/statistics" element={<Stat />} />
          <Route path="/history" element={<History />} />
          <Route path="/profile" element={<Profile />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

- App.js: Đây là file JavaScript chính của ứng dụng React, nơi chứa cấu trúc và logic cho

giao diện người dùng. File này thường định nghĩa các component, trạng thái (state), các sự kiện (event handlers) và cách thức ứng dụng phản ứng với các thay đổi. App.js là điểm khởi đầu của ứng dụng React, từ đó các component khác được nhúng hoặc điều hướng.

V. Kết quả thu được

5.1. Tổng quan

Hệ thống IoT đã thực hiện thành công các chức năng đo thông số môi trường theo thời gian thực, điều khiển thiết bị điện từ xa, và lưu trữ dữ liệu cho người dùng. Hệ thống đã được triển khai và thử nghiệm, đáp ứng tốt các yêu cầu đề ra.

5.2. Từ DHT11 và cảm biến ánh sáng

ID	Nhiệt độ (°C)	Độ ẩm (%)	Ánh sáng (Lux)	Gió (m/s)	Thời gian Đo
5525	30.2	44	202.83	97	13-11-2024 14:52:04
5524	30.2	44	244.87	94	13-11-2024 14:52:00
5523	30.2	44	238.51	51	13-11-2024 14:51:54
5522	30.2	43	245.36	0	13-11-2024 14:51:49
5521	30.2	43	258.55	10	13-11-2024 14:51:44
5520	30.2	43	256.6	92	13-11-2024 14:51:39
5519	30.2	43	262.46	17	13-11-2024 14:51:34
5518	30.2	43	265.4	46	13-11-2024 14:51:29
5517	30.2	43	263.93	56	13-11-2024 14:51:24
5516	30.2	43	263.44	62	13-11-2024 14:51:19

- Hệ thống đã đo được các thông số của nhiệt độ, độ ẩm, ánh sáng và truyền lên website theo thời gian thực để hiện thị cho người dùng một cách tương đối trên trang Data Sensor.

5.3. Khi người dùng điều khiển thiết bị

ID	Thiết bị	Hành động	Thời gian
1	Fan	On	16-10-2024 10:08:40
2	Led	On	16-10-2024 10:08:42
3	Airconditioner	On	16-10-2024 10:08:43
4	Led	Off	16-10-2024 10:08:46
5	Led	On	16-10-2024 10:08:46
6	Airconditioner	Off	16-10-2024 10:08:47
7	Airconditioner	On	16-10-2024 10:08:48
8	Airconditioner	Off	16-10-2024 10:08:59
9	Airconditioner	On	16-10-2024 10:09:00
10	Airconditioner	Off	16-10-2024 10:09:06

- Hệ thống đã lưu thành công các dữ liệu khi người dùng nhấn On/Off trên website và hiển thị trên trang Action History.

VI. Tài liệu tham khảo

- Tham khảo về Esp8266, mosquito, DHT11, Arduino IDE:

- esp8266-nodemcu-mqtt-publish-dht11 -arduino
- esp8266-nodemcu-mqtt-publish-subscribe-dht22-readings/

VII. Source code

- https://github.com/hvmhieu2003/BTL_IOT