

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**



**BÁO CÁO BTL CUỐI KỲ**  
**MÔN HỌC: IOT VÀ ỨNG DỤNG**

**Giảng viên hướng dẫn : Nguyễn Quốc Uy**

**Sinh viên thực hiện : Nguyễn Đình Đồng**

**Mã sinh viên : B21DCCN231**

**Số điện thoại : 0384617563**

***Hà Nội, 2024***

## LỜI CẢM ƠN

Trước tiên, em xin được bày tỏ sự trân trọng và biết ơn sâu sắc đối với thầy Nguyễn Quốc Uy, giảng viên hướng dẫn em trong suốt thời gian qua. Nhờ sự chỉ dẫn tận tình và những bài giảng thú vị của thầy, em đã có thêm nhiều hiểu biết và cái nhìn sâu sắc về lĩnh vực IoT và ứng dụng. Do chưa có nhiều kinh nghiệm làm đề tài và còn hạn chế về kiến thức, nên bài tập lớn của em chắc chắn sẽ không tránh khỏi những thiếu sót. Em rất mong nhận được sự nhận xét, đánh giá và ý kiến đóng góp của thầy để dự án của em được cải thiện hơn và hoàn thiện hơn trong tương lai.

Cuối cùng, em xin kính chúc thầy nhiều sức khỏe, thành công và hạnh phúc trong công việc cũng như cuộc sống. Em hy vọng sẽ tiếp tục nhận được sự hướng dẫn và hỗ trợ từ thầy trong những lần tới.

Em xin chân thành cảm ơn!

# MỤC LỤC

<b>I. Giới thiệu .....</b>	<b>3</b>
1. Mô tả đề tài và mục đích đề tài .....	3
2. Thiết bị sử dụng .....	4
3. Công nghệ sử dụng .....	9
<b>II. Giao diện và thiết kế tổng thể .....</b>	<b>13</b>
1. Giao diện Web .....	13
2. Giao diện API Doc .....	15
3. Bảng mạch.....	15
<b>III. Code .....</b>	<b>16</b>
1. Embedded Code .....	16
2, Code Client .....	27
3. Code Server.....	28
<b>IV. Tổng kết.....</b>	<b>30</b>

## **I. Giới thiệu**

### **1. Mô tả đề tài và mục đích đề tài**

Đề tài này tập trung vào việc phát triển hệ thống giám sát và điều khiển nhiệt độ, độ ẩm, ánh sáng và các thiết bị quạt, đèn, điều hòa, hướng đến việc phát triển một hệ thống web thông minh cho phép người dùng theo dõi và điều khiển các thông số môi trường (như nhiệt độ, độ ẩm, ánh sáng) và các thiết bị điện tử như quạt, đèn, điều hòa. Hệ thống này sẽ sử dụng các cảm biến để thu thập dữ liệu môi trường theo thời gian thực và cung cấp cho người dùng khả năng quản lý các thiết bị điện tử từ xa qua giao diện web. Ngoài ra, người dùng còn có thể thiết lập các kịch bản tự động như tự bật/tắt đèn, quạt, điều hòa dựa trên các điều kiện môi trường đã định sẵn. Hệ thống sử dụng NodeJS làm nền tảng cho phần Backend, cung cấp các API để ứng dụng có thể lấy thông tin về nhiệt độ, độ ẩm, ánh sáng và trạng thái của các thiết bị. Đồng thời, hệ thống cũng hỗ trợ các API để điều khiển bật/tắt các thiết bị từ xa, đảm bảo sự tiện lợi và linh hoạt cho người dùng. Cơ sở dữ liệu MySQL được sử dụng để lưu trữ toàn bộ dữ liệu liên quan, bao gồm thông tin về môi trường, trạng thái thiết bị và lịch sử tương tác của người dùng. Người dùng không chỉ có thể dễ dàng điều khiển các thiết bị từ xa mà còn có thể liên tục theo dõi các chỉ số môi trường và hoạt động của thiết bị qua các biểu đồ cập nhật theo thời gian thực, cũng như xem lại lịch sử sử dụng. Hệ thống này hứa hẹn sẽ mang đến trải nghiệm quản lý và giám sát thông minh, tiện lợi và toàn diện cho người dùng.

## 2. Thiết bị sử dụng

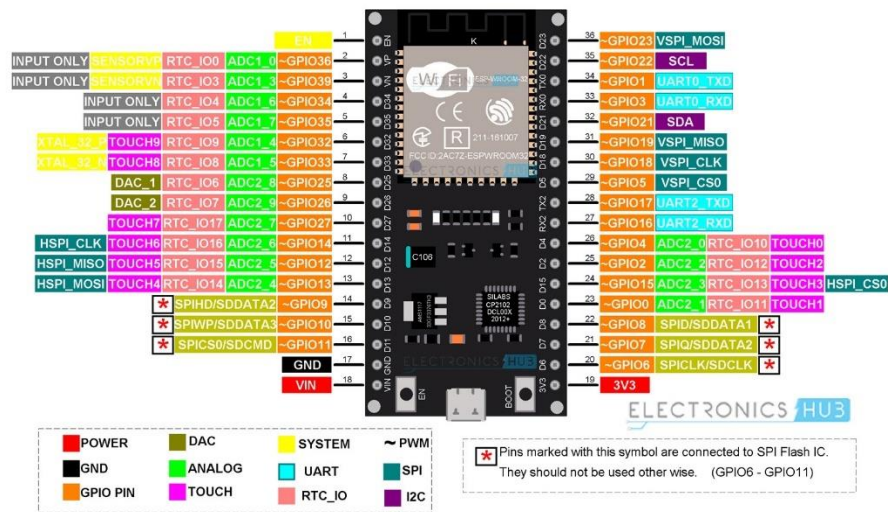
### a. Node MCU ESP32



#### - Thông số kỹ thuật:

- + CPU: Dual-core hoặc Single-core Tensilica Xtensa LX6, tốc độ xung nhịp từ 80 MHz đến 240 MHz
- + RAM: 520 KB SRAM
- + Bộ nhớ Flash: Tùy chọn từ 4 MB đến 16 MB (hoặc ngoài)
- + Wi-Fi: Chuẩn 802.11 b/g/n (2.4 GHz), hỗ trợ chế độ AP, STA
- + Bluetooth: Phiên bản 4.2, hỗ trợ Bluetooth Classic và BLE
- + GPIO: 34 chân GPIO, hỗ trợ ADC (18 kênh, 12 bit), 2 kênh DAC
- + Giao tiếp: SPI, I2C, I2S, UART, PWM, CAN Bus
- + Nguồn: Điện áp hoạt động từ 2.2V đến 3.6V (thông thường 3.3V)
- + Chế độ tiết kiệm năng lượng: Hỗ trợ Deep Sleep, Light Sleep, Modem Sleep
- + Cảm biến tích hợp: Cảm biến Hall và cảm biến nhiệt độ

## - Sơ đồ chân ESP 32



## b. Cảm biến nhiệt độ, độ ẩm DHT11



## - Thông số kỹ thuật:

- + Điện áp hoạt động: 3V – 5V DC
- + Dòng điện tiêu thụ: 2.5 mA
- + Phạm vi cảm biến độ ẩm: 20% - 90% RH, sai số  $\pm 5\%RH$
- + Phạm vi cảm biến nhiệt độ:  $0^{\circ}C \sim 50^{\circ}C$ , sai số  $\pm 2^{\circ}C$
- + Tần số lấy mẫu tối đa: 1Hz (1 giây 1 lần)
- + Kích thước: 23 x 12 x 5 mm

- Sơ đồ chân kết nối DHT11:

Chân	Mô tả
VCC	- Cung cấp nguồn điện (5V hoặc 3.3V) - Kết nối với chân 3V3 của ESP32
DATA	- Chân dữ liệu, dùng để giao tiếp với vi điều khiển - Kết nối với chân GPIO4 trên ESP32
GND	- Chân nối đất - Kết nối với chân GND của ESP32

***c. Cảm biến cường độ ánh sáng quang trở MS-CDS05***



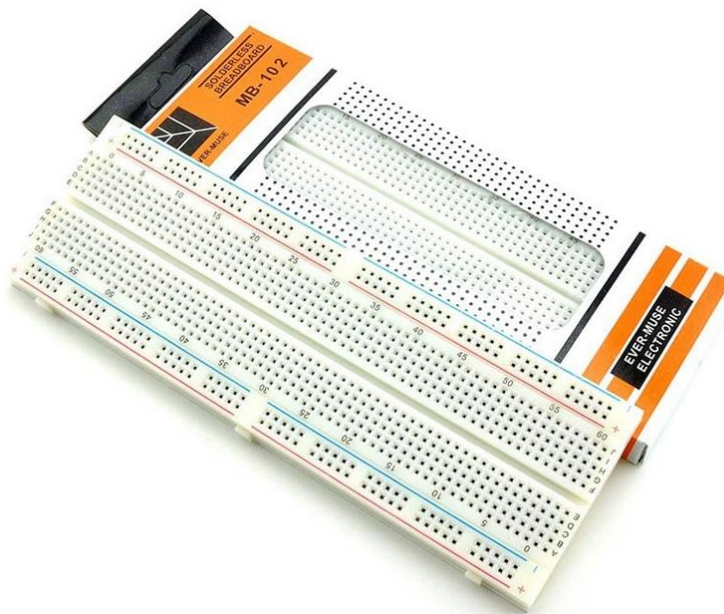
- Thông số kỹ thuật:

- + Điện áp hoạt động: 3V – 5V DC
- + Kết nối: 4 chân (2 chân cấp nguồn VCC và GND, 2 chân tín hiệu ra AO và DO)
- + Hỗ trợ cả tín hiệu Analog và TTL
- + Đầu ra Analog 0V – 5V tỷ lệ với cường độ ánh sáng đầu vào, đầu vào TTL hoạt động mức thấp
- + Độ nhạy sáng cao tùy chỉnh bằng điện trở
- + Kích thước: 32 x 14 mm

- Sơ đồ chân kết nối MS-CDS05

Chân	Mô tả
VCC	- Cung cấp nguồn điện (5V hoặc 3.3V) - Kết nối với chân 3V3 của ESP32
GND	- Chân nối đất - Kết nối với chân GND của ESP32
AO	- Chân tín hiệu analog, dùng để giao tiếp với vi điều khiển - Kết nối với chân GPIO34 trên ESP32
DO	- Chân tín hiệu số, dùng để giao tiếp với vi điều khiển - Không cần kết nối

***d. Board test MB-102***



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.

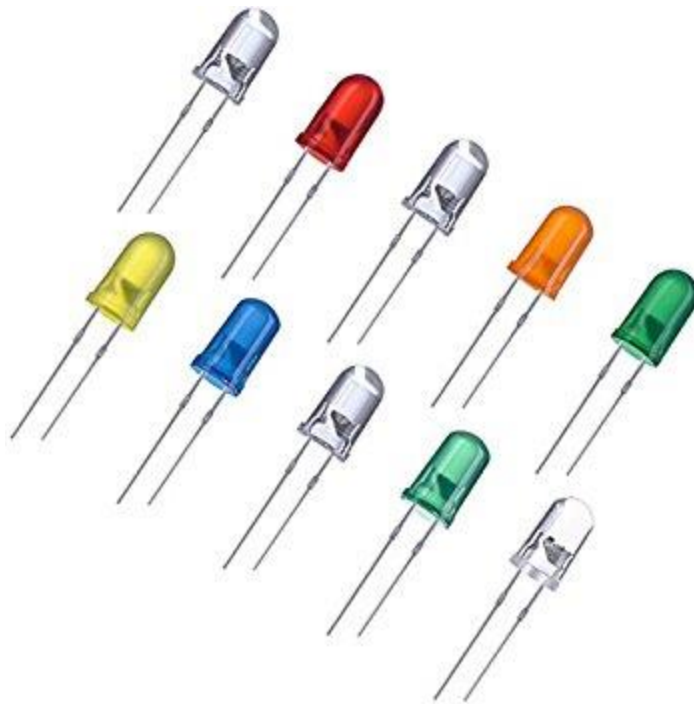


*e. Dây nối*

---



*f. Led màu 5mm*



### 3. Công nghệ sử dụng

#### *a. Trình biên dịch Arduino IDE*

Arduino IDE là một phần mềm mã nguồn mở, chủ yếu được sử dụng để viết và biên dịch mã vào các module Arduino. Phần mềm này có các phiên bản cho nhiều hệ điều hành như macOS, Windows, Linux, và được xây dựng trên nền tảng Java. Arduino IDE đi kèm với nhiều chức năng và lệnh có sẵn, đóng vai trò quan trọng trong việc gỡ lỗi, chỉnh sửa, và biên dịch mã. Các module Arduino phổ biến như Arduino Uno, Arduino Mega, và Arduino Leonardo đều sử dụng bộ vi điều khiển trên bo mạch có thể lập trình để nhận và thực hiện các lệnh từ mã được tải lên.

Môi trường phát triển IDE bao gồm hai phần chính: trình chỉnh sửa và trình biên dịch. Trình chỉnh sửa được sử dụng để viết mã chương trình, trong khi trình biên dịch có chức năng kiểm tra, biên dịch và tải mã lên module Arduino. Phần mềm hỗ trợ lập trình bằng ngôn ngữ C và C++, cung cấp sự linh hoạt cho lập trình viên trong việc phát triển các ứng dụng trên nền tảng Arduino.

#### *b. Visual Studio Code*

Visual Studio Code (VS Code) là một trình soạn thảo mã nguồn mở, đa nền tảng do Microsoft phát triển, hỗ trợ lập trình viên phát triển các ứng dụng bằng nhiều ngôn ngữ như JavaScript, Python, C++, Java, HTML, và CSS. Với giao diện trực quan, dễ sử dụng, VS Code phù hợp cho cả người mới bắt đầu và lập trình viên chuyên nghiệp. Trình soạn thảo này tích hợp Git, giúp quản lý mã nguồn dễ dàng thông qua các thao tác như commit, push, pull và xem thay đổi trực tiếp.

Ngoài ra, VS Code còn có kho extension phong phú, giúp mở rộng tính năng để hỗ trợ nhiều công nghệ và công cụ phát triển. Tính năng **IntelliSense** cung cấp tự động hoàn thành mã thông minh, gợi ý các biến, hàm và đối tượng, giúp cải thiện tốc độ lập trình. VS Code cũng tích hợp công cụ gỡ lỗi cho nhiều ngôn ngữ như Node.js, Python, và C++, cho phép lập trình viên dễ dàng kiểm tra và sửa lỗi ngay trong môi trường làm việc. VS Code có thể chạy trên nhiều hệ điều hành như Windows, macOS và Linux, giúp nó trở thành một trong những trình soạn thảo mã nguồn phổ biến nhất nhờ tốc độ nhanh, nhẹ và khả năng tùy chỉnh linh hoạt.

### ***c. ReactJS***

ReactJS là một thư viện JavaScript mã nguồn mở do Facebook phát triển, được sử dụng để xây dựng giao diện người dùng (UI) cho các ứng dụng web. React dựa trên kiến trúc thành phần (component-based), cho phép tạo ra các thành phần giao diện tái sử dụng, giúp ứng dụng dễ quản lý và mở rộng. Thư viện này sử dụng Virtual DOM để cập nhật giao diện nhanh hơn và cải thiện hiệu suất.

Với cú pháp JSX, lập trình viên có thể viết HTML trực tiếp trong JavaScript, giúp mã nguồn dễ đọc. React sử dụng cơ chế ràng buộc dữ liệu một chiều, đảm bảo việc quản lý dữ liệu đơn giản và dễ dự đoán. Ngoài ra, React có thể được render phía server, hỗ trợ tốt hơn cho SEO so với các thư viện chỉ render phía client.

### ***d. NodeJS***

Node.js là một nền tảng mã nguồn mở, đa nền tảng được phát triển dựa trên JavaScript V8 Engine của Google. Nó cho phép lập trình viên xây dựng các ứng dụng phía server bằng JavaScript, ngôn ngữ trước đây chỉ dùng cho phát triển phía client. Node.js nổi bật với kiến trúc hướng sự kiện (event-driven) và cơ chế xử lý bất đồng bộ (non-blocking I/O), giúp các ứng dụng chạy mượt mà với hiệu suất cao, đặc biệt phù hợp cho các hệ thống yêu cầu xử lý nhiều kết nối đồng thời như web server, ứng dụng thời gian thực, và API.

Một trong những ưu điểm của Node.js là nó sử dụng một môi trường chung cho cả phía client và server, cho phép lập trình viên dễ dàng chia sẻ mã nguồn và các thư viện. Hệ sinh thái phong phú với hàng nghìn module có sẵn từ npm (Node Package Manager) giúp việc phát triển ứng dụng nhanh chóng và dễ dàng hơn. Nhờ sự linh hoạt và mạnh mẽ, Node.js được sử dụng rộng rãi trong các ứng dụng web, dịch vụ API, và các ứng dụng yêu cầu hiệu suất cao.

### ***e. Mosquitto***

Mosquitto là một MQTT Broker mã nguồn mở, hỗ trợ giao thức MQTT phiên bản 5.0, 3.1.1 và 3.1. Đây là giao thức nhanh và nhẹ, hoạt động theo mô hình publish/subscribe, rất phổ biến trong các ứng dụng Internet of Things (IoT). Mosquitto cung cấp thư viện viết bằng ngôn ngữ C để triển khai các MQTT Client và có thể dễ dàng thao tác qua dòng lệnh như "mosquitto\_pub" và "mosquitto\_sub."

<b>Ưu điểm</b>	<b>Nhược điểm</b>
Mosquitto nổi bật nhờ tốc độ truyền nhận và xử lý dữ liệu nhanh chóng, độ ổn định cao, và được sử dụng rộng rãi, đặc biệt phù hợp với các ứng dụng nhúng (embedded). Nó rất nhẹ, có thể hoạt động trên nhiều thiết bị khác nhau. Ngoài ra, Mosquitto hỗ trợ các giao thức bảo mật TLS/SSL, giúp xác thực server và client cũng như mã hóa dữ liệu để tăng cường bảo mật.	Một số hạn chế của Mosquitto bao gồm khó khăn trong việc thiết kế các ứng dụng lớn và việc thiếu các phương thức xác thực đa dạng cho thiết bị, dẫn đến khả năng bảo mật chưa thực sự tối ưu.

### ***f. Socket.io***

Socket.IO là một thư viện JavaScript mạnh mẽ cho phép giao tiếp theo thời gian thực giữa client và server. Nó xây dựng trên giao thức WebSocket nhưng cung cấp nhiều tính năng bổ sung và khả năng phục hồi tự động khi kết nối gặp sự cố. Socket.IO có thể được sử dụng trong các ứng dụng web, mobile và desktop, mang đến khả năng truyền dữ liệu hai chiều hiệu quả.

Đặc điểm nổi bật của Socket.IO:

- + Kết nối hai chiều: Socket.IO cho phép client và server giao tiếp lẫn nhau một cách tức thì, hỗ trợ cả việc gửi và nhận dữ liệu mà không cần thiết lập lại kết nối.
- + Tự động phục hồi: Nếu kết nối bị mất, Socket.IO sẽ tự động cố gắng kết nối lại, giúp duy trì tính liên tục trong giao tiếp.

- + Hỗ trợ sự kiện: Socket.IO sử dụng mô hình sự kiện, cho phép lập trình viên dễ dàng xử lý các sự kiện cụ thể, như khi có dữ liệu mới đến hoặc khi một kết nối mới được thiết lập.
- + Khả năng tương thích: Socket.IO hỗ trợ nhiều giao thức truyền thông khác nhau, tự động chuyển đổi giữa chúng để đảm bảo kết nối tốt nhất có thể.

### ***Áp dụng Socket.IO trong lập trình IoT kết hợp với web:***

Trong lĩnh vực Internet of Things (IoT), Socket.IO có thể được sử dụng để tạo ra các ứng dụng web tương tác mạnh mẽ, cho phép giám sát và điều khiển thiết bị IoT theo thời gian thực. Dưới đây là một số ứng dụng cụ thể:

- + Giám sát dữ liệu cảm biến: Các thiết bị IoT có thể gửi dữ liệu cảm biến (như nhiệt độ, độ ẩm) đến server qua Socket.IO. Người dùng có thể xem dữ liệu này trong thời gian thực trên giao diện web mà không cần làm mới trang.
- + Điều khiển thiết bị từ xa: Người dùng có thể gửi lệnh đến các thiết bị IoT thông qua giao diện web, và Socket.IO sẽ gửi lệnh này đến thiết bị ngay lập tức, cho phép điều khiển tức thì (ví dụ: bật/tắt đèn, quạt).
- + Thông báo sự kiện: Socket.IO có thể gửi thông báo đến client ngay khi có sự kiện quan trọng xảy ra (như thiết bị gặp sự cố hoặc có thay đổi trong trạng thái), giúp người dùng có thể phản ứng kịp thời.

Với khả năng cung cấp giao tiếp hai chiều và hỗ trợ thời gian thực, Socket.IO là công cụ lý tưởng để phát triển các ứng dụng IoT kết hợp với web, tạo ra trải nghiệm người dùng liền mạch và tương tác nhanh chóng.

## II. Giao diện và thiết kế tổng thể

### 1. Giao diện Web



*Giao diện Dashboard*

The Data Sensors interface displays a table of sensor data. The table has columns for ID, Temp, Humi, Light, and Time. The data is sorted by Time in descending order. The table is paginated with 10 items per page. The current page is 1 of 198.

ID	Temp ▲	Humi ▲	Light ▲	Time
1973	30	77	207	25/09/2024 09:38:34
1972	30	77	202	25/09/2024 09:38:33
1971	30	77	201	25/09/2024 09:38:32
1970	30	77	205	25/09/2024 09:38:31
1969	30	77	201	25/09/2024 09:38:30
1968	30	77	209	25/09/2024 09:38:29
1967	30	77	203	25/09/2024 09:38:28
1966	30	77	203	25/09/2024 09:38:27
1965	30	77	204	25/09/2024 09:38:26
1964	30	77	203	25/09/2024 09:38:25

*Giao diện Data Sensor*

Dashboard

Data Sensor

Action History

Profile

ACTION HISTORY

9:41:10 AM

Page Size:

10

Search by Date:

ID	TYPE	State	Time
452	FAN	ON	25/09/2024 09:35:44
451	AC	ON	25/09/2024 09:35:42
450	LED	ON	25/09/2024 09:35:40
449	FAN	OFF	25/09/2024 08:52:05
448	AC	OFF	25/09/2024 08:52:04
447	LED	OFF	25/09/2024 08:52:03
446	FAN	ON	25/09/2024 08:52:00
445	LED	ON	25/09/2024 08:51:59
444	LED	OFF	25/09/2024 08:51:58
443	AC	ON	25/09/2024 08:51:55

1

2

3

4

...

45

*Giao diện Action History*

Dashboard

Data Sensor

Action History

Profile

PROFILE

9:56:00 AM

Nguyễn Đình Đồng

821DCCN231

Công nghệ phần mềm

Học viện Công nghệ Bưu chính Viễn thông

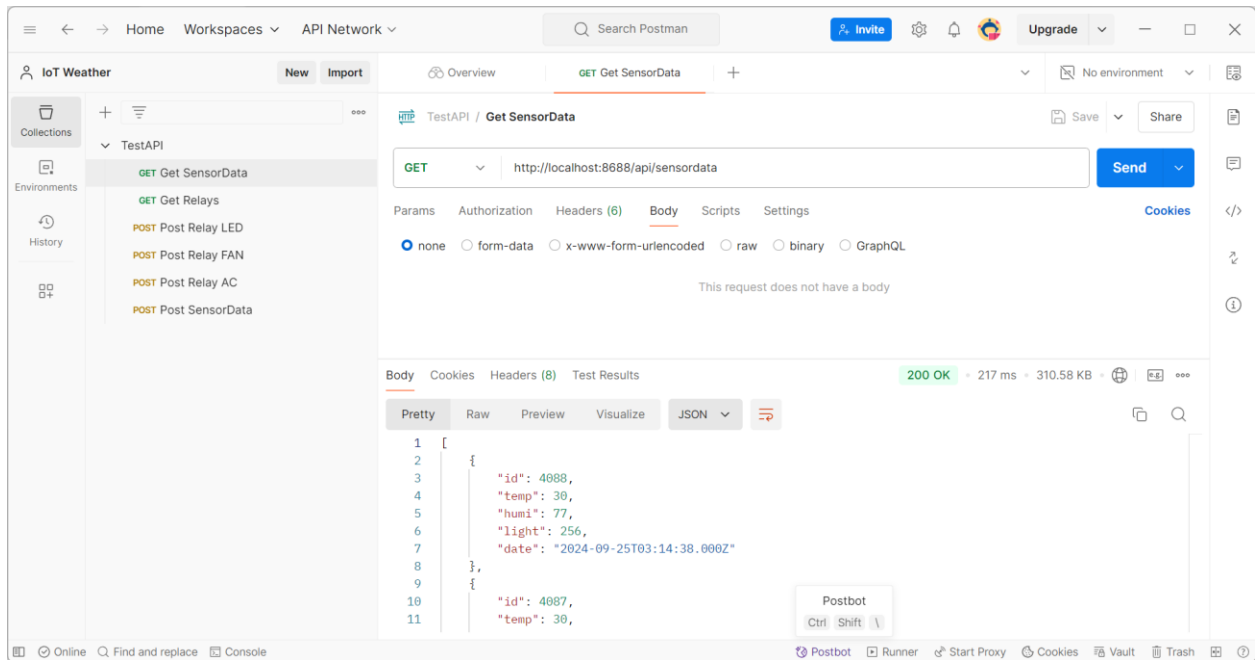
Google Drive

GitHub

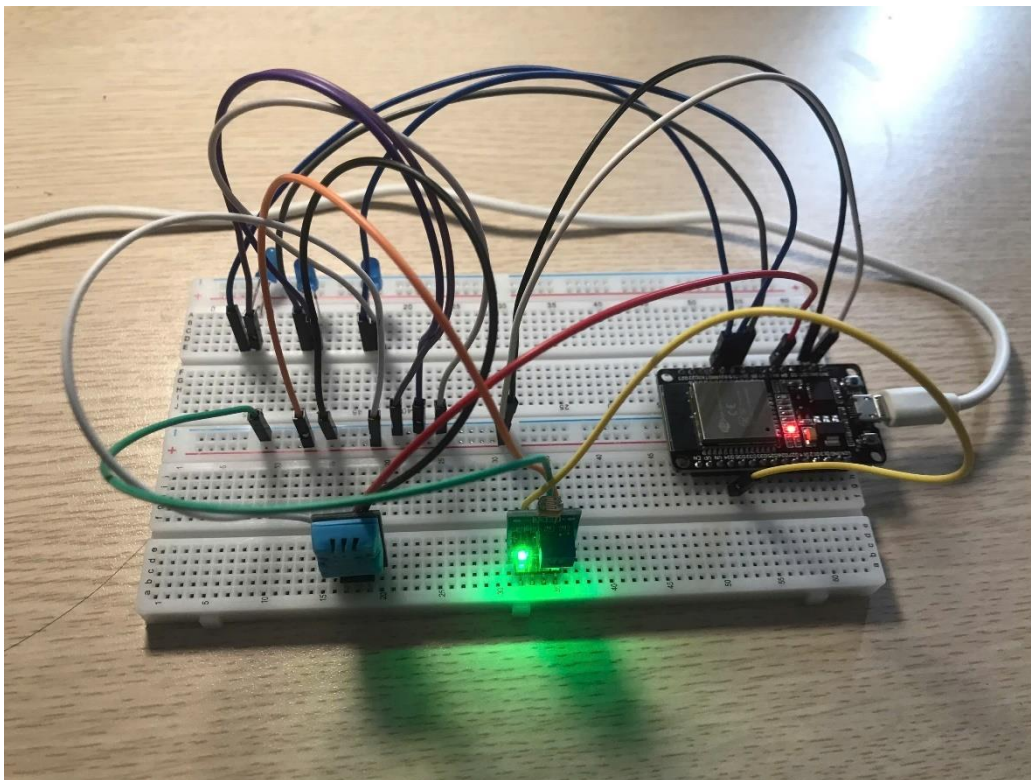
Báo cáo

*Giao diện Profile*

## 2. Giao diện API Doc



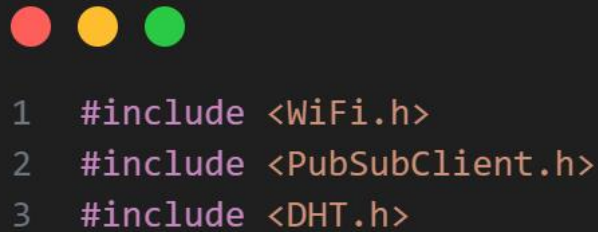
## 3. Bảng mạch





### III. Code

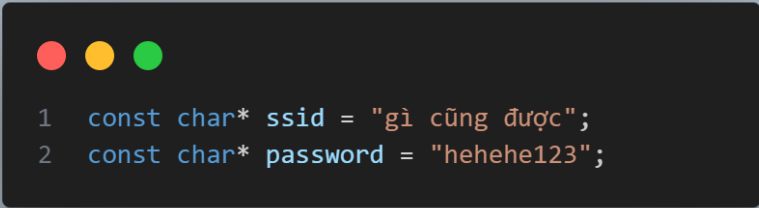
#### 1. Embedded Code



```
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3  #include <DHT.h>
```

- Khai báo thư viện

- + WiFi.h : Thư viện kết nối ESP32 với mạng WiFi
- + PubSubClient.h : Thư viện hỗ trợ giao thức MQTT
- + DHT.h : Thư viện cho cảm biến DHT (đọc nhiệt độ và độ ẩm)



```
1  const char* ssid = "gì cũng được";
2  const char* password = "hehehe123";
```

- Định nghĩa các thông tin WiFi như tên WiFi, mật khẩu để ESP32 có thể kết nối

```
1 // Cấu hình MQTT
2 const char* mqtt_server = "192.168.19.105";
3 const int mqtt_port = 1883;
4 const char* mqtt_topic = "sensor";
5
6 // Thông tin xác thực MQTT
7 const char* mqtt_user = "DongND";
8 const char* mqtt_pass = "dong1808";
```

- Cấu hình MQTT bao gồm: địa chỉ máy chủ, port, topic mà ESP32 sử dụng để gửi và nhận dữ liệu, tiếp đến là thông tin xác thực gồm user và password để đăng nhập vào máy chủ MQTT

```
1 // Chủ đề để gửi trạng thái relay
2 const char* relay_status_topic = "relay_status";
3
4 // Các chủ đề MQTT cho điều khiển relay
5 const char* relay_1_topic = "relay_1";
6 const char* relay_2_topic = "relay_2";
7 const char* relay_3_topic = "relay_3";
```

- Chủ đề điều khiển relay: Các chủ đề để điều khiển ba relay (điện áp có thể được bật/tắt) và gửi trạng thái của chúng.



```
1 #define DHTPIN 4
2 #define DHTTYPE DHT11
3 DHT dht(DHTPIN, DHTTYPE);
```

- Cấu hình cảm biến DHT11: Chân kết nối cảm biến DHT và kiểu cảm biến được sử dụng



```
1 #define RELAY_1_PIN 21
2 #define RELAY_2_PIN 19
3 #define RELAY_3_PIN 18
```

- Cấu hình 3 relay tương ứng LED, FAN, AC vào 3 cổng kết nối

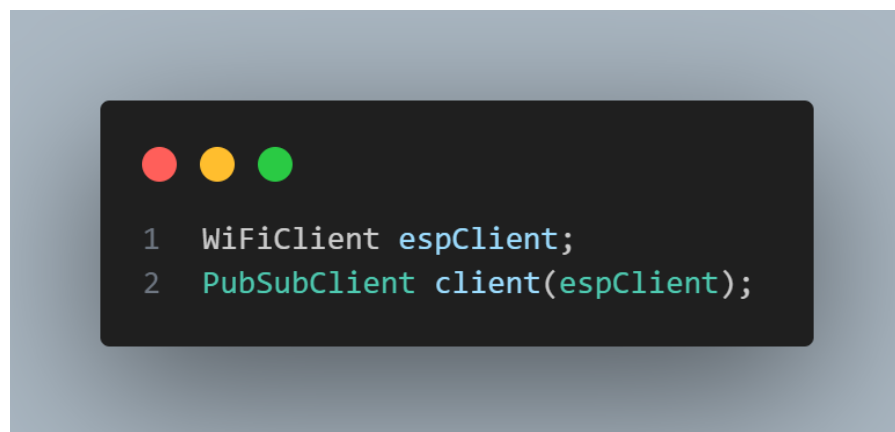


```
1 #define LIGHT_SENSOR_PIN 34
2 #define maxLux 500
```

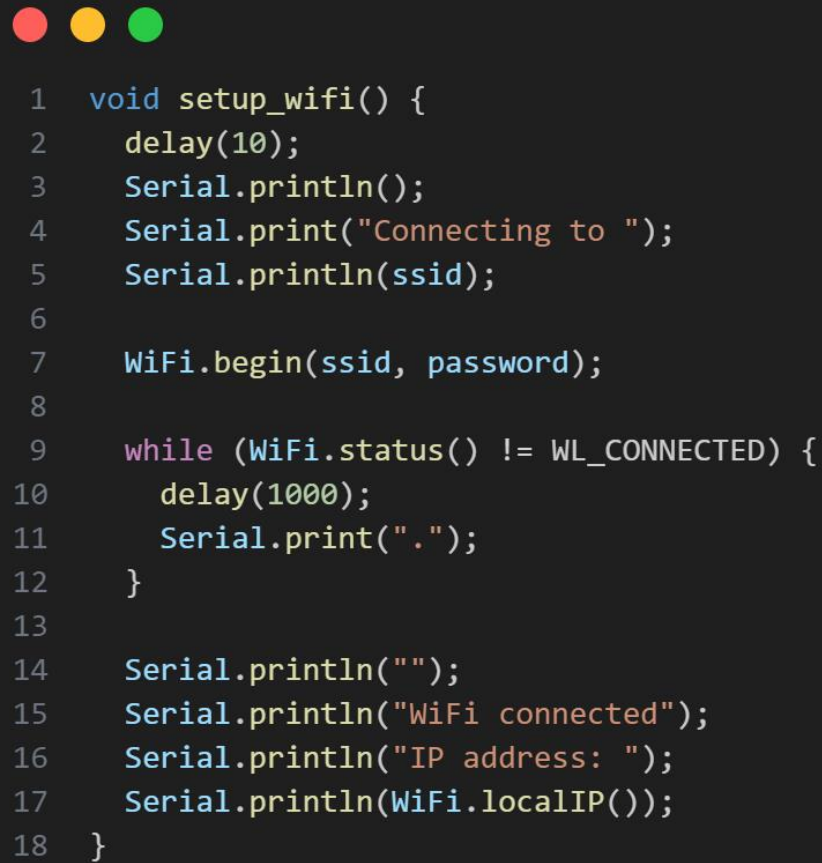
- Cấu hình cảm biến ánh sáng bao gồm cổng kết nối và maxLux



- Khai báo biến trạng thái để lưu trữ trạng thái của 3 relay




- Khởi tạo MQTT client: Tạo đối tượng client để xử lý giao tiếp MQTT



```
1 void setup_wifi() {
2     delay(10);
3     Serial.println();
4     Serial.print("Connecting to ");
5     Serial.println(ssid);
6
7     WiFi.begin(ssid, password);
8
9     while (WiFi.status() != WL_CONNECTED) {
10         delay(1000);
11         Serial.print(".");
12     }
13
14     Serial.println("");
15     Serial.println("WiFi connected");
16     Serial.println("IP address: ");
17     Serial.println(WiFi.localIP());
18 }
```

- Hàm kết nối WiFi: Dùng để kết nối ESP32 với WiFi, và in ra địa chỉ IP khi kết nối thành công

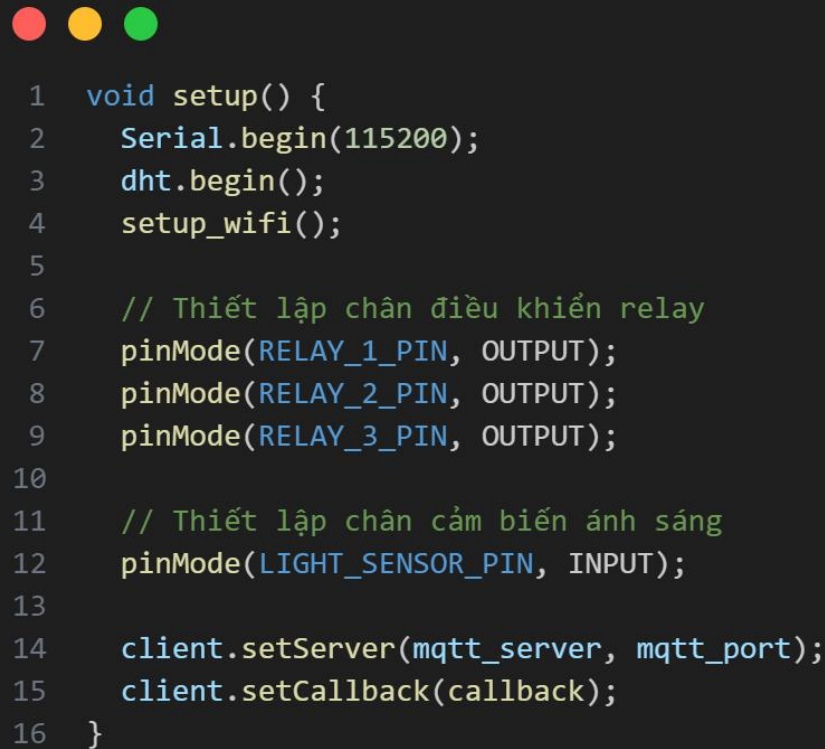


```
1  if (String(topic) == relay_1_topic) {  
2      if (msg == "1") {  
3          digitalWrite(RELAY_1_PIN, HIGH);  
4          state_1 = 1;  
5          Serial.println("Relay 1 ON");  
6      } else if (msg == "0") {  
7          digitalWrite(RELAY_1_PIN, LOW);  
8          state_1 = 0;  
9          Serial.println("Relay 1 OFF");  
10     }  
11 }
```

- Hàm xử lý relay\_1 dựa trên thông điệp nhận được, tương tự đối với relay\_2 và relay\_3

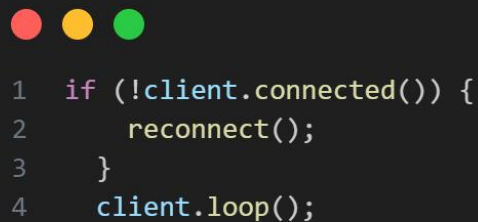
```
1 void reconnect() {
2     while (!client.connected()) {
3         Serial.print("Attempting MQTT connection...");
4         if (client.connect("ESP32Client", mqtt_user, mqtt_pass)) {
5             Serial.println("connected");
6
7             client.subscribe(relay_1_topic);
8             client.subscribe(relay_2_topic);
9             client.subscribe(relay_3_topic);
10        } else {
11            Serial.print("failed, rc=");
12            Serial.print(client.state());
13            Serial.println(" try again in 5 seconds");
14            delay(5000);
15        }
16    }
17 }
```

- Hàm kết nối MQTT với thông tin xác thực bao gồm user và password, kiểm tra và thiết lập kết nối đến máy chủ MQTT, đồng thời đăng ký các chủ đề để nhận tin nhắn



```
1 void setup() {  
2     Serial.begin(115200);  
3     dht.begin();  
4     setup_wifi();  
5  
6     // Thiết lập chân điều khiển relay  
7     pinMode(RELAY_1_PIN, OUTPUT);  
8     pinMode(RELAY_2_PIN, OUTPUT);  
9     pinMode(RELAY_3_PIN, OUTPUT);  
10  
11     // Thiết lập chân cảm biến ánh sáng  
12     pinMode(LIGHT_SENSOR_PIN, INPUT);  
13  
14     client.setServer(mqtt_server, mqtt_port);  
15     client.setCallback(callback);  
16 }
```

- Hàm setup: Khởi tạo Serial, cảm biến DHT, thiết lập WiFi, chân cho relay và cảm biến ánh sáng, cùng với cấu hình cho MQTT



```
1 if (!client.connected()) {  
2     reconnect();  
3 }  
4 client.loop();
```

- Hàm kiểm tra kết nối MQTT





```
1 // Đọc dữ liệu từ DHT11
2 float h = dht.readHumidity();
3 float t = dht.readTemperature();
4
5 // Kiểm tra nếu có lỗi khi đọc cảm biến
6 if (isnan(h) || isnan(t)) {
7     Serial.println("Failed to read from DHT sensor!");
8     return;
9 }
```

- Hàm đọc giá trị độ ẩm (humidity) và nhiệt độ (temperature) từ cảm biến DHT11. Hai giá trị này sẽ được sử dụng để gửi qua MQTT



```
1 int sensorValue = analogRead(LIGHT_SENSOR_PIN);
2 float voltage = sensorValue * (3.3 / 4095.0);
3 float light = 500 - (voltage / 3.3) * maxLux;
```

- Hàm đọc giá trị analog của cảm biến ánh sáng, sau đó tính toán dựa vào điện áp rồi xuất ra giá trị ánh sáng, chuẩn hóa rồi đưa ra giá trị hợp lý trong phạm vi tối đa được định nghĩa



```
1 String payload = "{\"temperature\": ";
2   payload += String(t);
3   payload += ", \"humidity\": ";
4   payload += String(h);
5   payload += ", \"light\": ";
6   payload += String(light);
7   payload += ", \"state_1\" : ";
8   payload += String(state_1);
9   payload += ", \"state_2\" : ";
10  payload += String(state_2);
11  payload += ", \"state_3\" : ";
12  payload += String(state_3);
13  payload += "}";
```

- Tạo chuỗi JSON: Payload JSON được tạo ra để chứa tất cả các dữ liệu cần gửi đi, bao gồm nhiệt độ, độ ẩm, ánh sáng và trạng thái của ba relay. Chuỗi JSON sẽ có định dạng chuẩn để dễ dàng gửi và phân tích trên máy chủ nhận.



```
1 Serial.print("Publishing message: ");  
2 Serial.println(payload);  
3 client.publish(mqtt_topic, (char*) payload.c_str());
```

- Gửi dữ liệu qua MQTT:

- + Trước khi gửi, hàm in ra thông điệp để người dùng có thể theo dõi
- + Dữ liệu JSON được gửi đến chủ đề MQTT được cấu hình trước đó

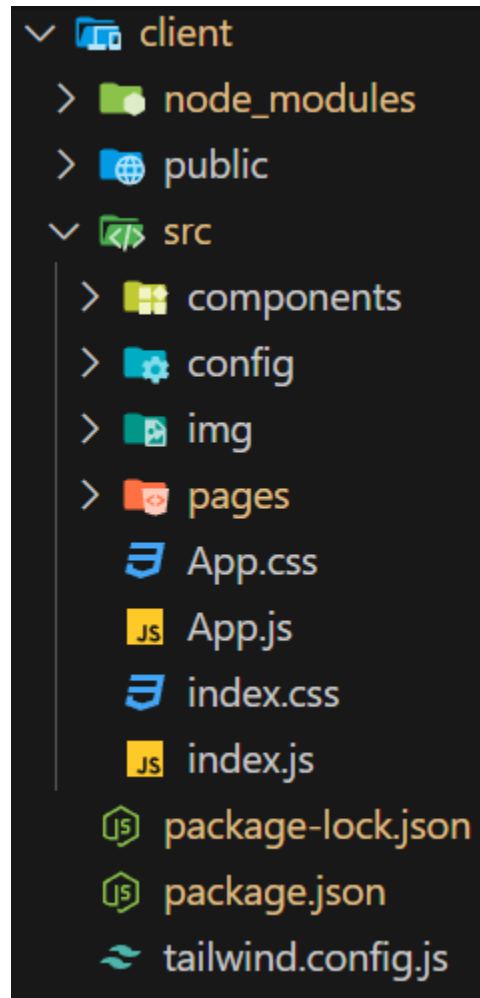


```
1 delay(1000);
```

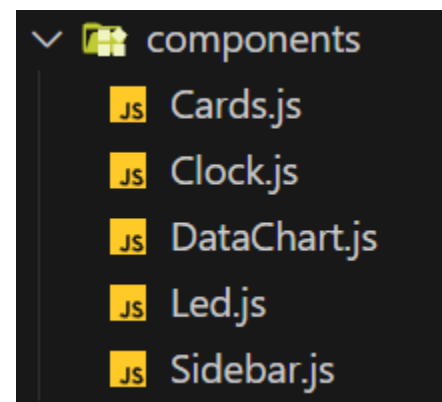
- Hàm dừng việc thực hiện trong 1 giây trước khi lặp lại quá trình đọc và gửi dữ liệu. Điều này giúp tránh việc gửi dữ liệu quá thường xuyên, giảm tải cho máy chủ và băng thông mạng

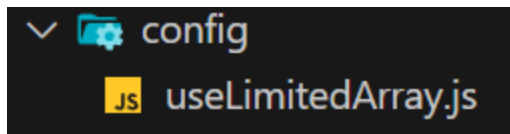
## 2. Code Client

- Cấu trúc thư mục:



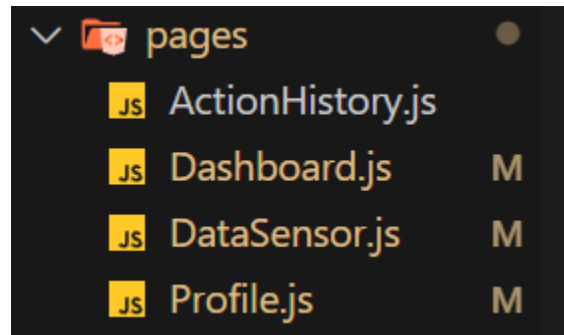
- Components: sử dụng để tổ chức và lưu trữ các thành phần (components) của trang web. Các thành phần này là các khối xây dựng cơ bản của giao diện Dashboard.





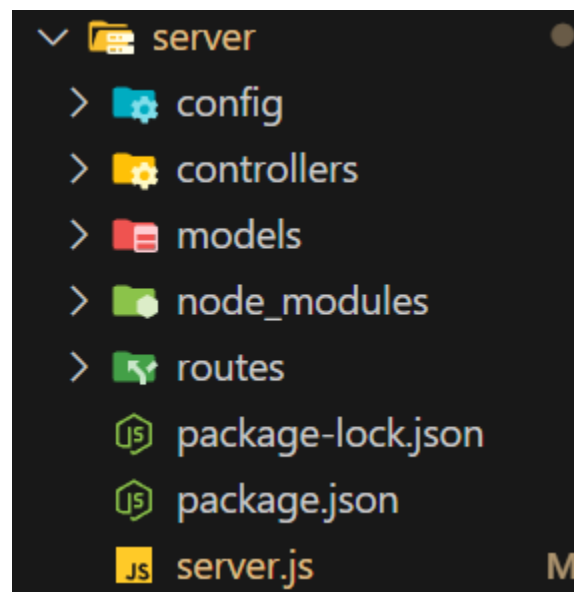
- Config: chứa file định nghĩa một custom Hook trong React để quản lý mảng có kích thước giới hạn

- Pages: được sử dụng để chứa các thành phần đại diện cho các trang khác nhau của ứng dụng, là các trang mà người dùng có thể điều hướng đến

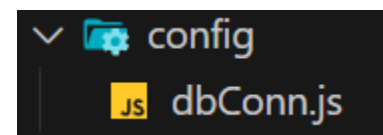


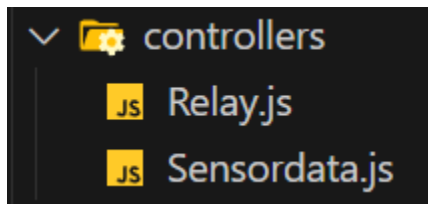
### 3. Code Server

- Cấu trúc thư mục:



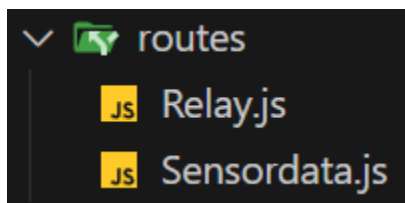
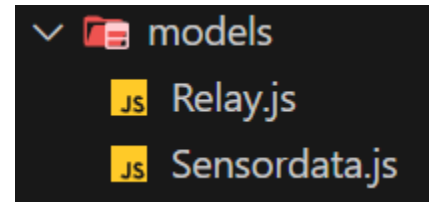
- Config: chứa file dùng để thiết lập kết nối với cơ sở dữ liệu MySQL





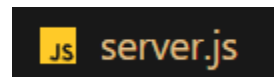
- Controller: chứa các file chịu trách nhiệm xử lý các yêu cầu HTTP, tương tác với mô hình dữ liệu và trả về phản hồi cho người dùng

- Model: sử dụng để tổ chức các tệp tin chứa các mô hình dữ liệu. Các mô hình này đại diện cho cấu trúc dữ liệu và các phương thức để tương tác với cơ sở dữ liệu



- Route: thường được sử dụng để tổ chức các tệp định tuyến. Các tệp định tuyến này định nghĩa các đường dẫn URL và ánh xạ chúng tới các phương thức trong Controller tương ứng

- Cuối cùng là file server.js: có nhiệm vụ nhận dữ liệu cảm biến, lưu trữ chúng vào cơ sở dữ liệu MySQL và phát dữ liệu này qua WebSocket cho các client kết nối



- Chức năng chính của server.js:

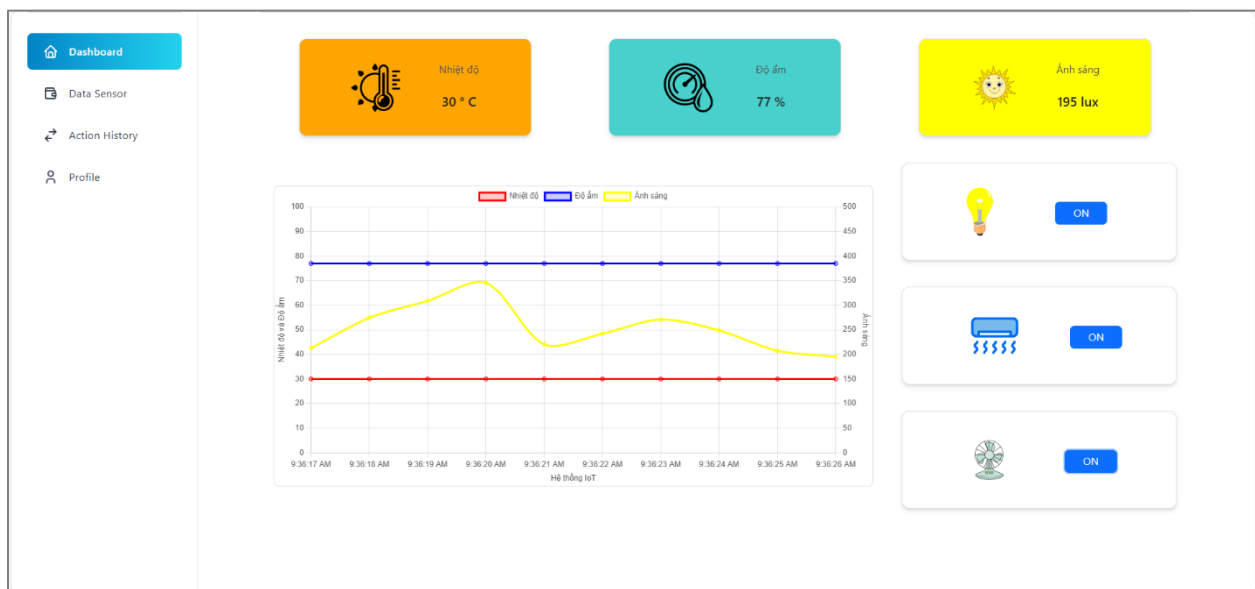
- + Phân tích dữ liệu cảm biến từ thông điệp JSON.
- + Chèn dữ liệu cảm biến vào cơ sở dữ liệu MySQL.
- + Phát dữ liệu cảm biến qua WebSocket cho các client.

## IV. Tổng kết

### 1. Tổng quan

Trong quá trình phát triển hệ thống IoT, em đã thực hiện thành công các chức năng đo lường 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.

### 2. Theo dõi thông tin theo thời gian thực



- Hệ thống đã đo được các thông số và truyền lên ứng dụng để hiển thị cho người dùng một cách tương đối, biểu đồ liên tục cập nhật theo thời gian thực
- Khi bật hoặc tắt đèn, ứng dụng chờ phản hồi từ hardware xong thì cập nhật giao diện tương ứng

### 3. Thông tin về các hoạt động bật tắt thiết bị

Dashboard

Data Sensor

Action History

Profile

ACTION HISTORY

9:41:10 AM

Page Size:

10

Search by Date:

ID	TYPE	State	Time
452	FAN	ON	25/09/2024 09:35:44
451	AC	ON	25/09/2024 09:35:42
450	LED	ON	25/09/2024 09:35:40
449	FAN	OFF	25/09/2024 08:52:05
448	AC	OFF	25/09/2024 08:52:04
447	LED	OFF	25/09/2024 08:52:03
446	FAN	ON	25/09/2024 08:52:00
445	LED	ON	25/09/2024 08:51:59
444	LED	OFF	25/09/2024 08:51:58
443	AC	ON	25/09/2024 08:51:55

1

2

3

4

...

45

- Hệ thống đã lưu thành công các dữ liệu đọc từ hardware, đồng thời người dùng có thể sắp xếp, tìm kiếm các hành động với các thiết bị

### 4. Thông tin về dữ liệu được đọc từ phần cứng

Dashboard

Data Sensor

Action History

Profile

DATA SENSORS

9:38:33 AM

Page Size:10

Filter:All

Search:Enter search term

ID	Temp ▲	Humi ▲	Light ▲	Time
1973	30	77	207	25/09/2024 09:38:34
1972	30	77	202	25/09/2024 09:38:33
1971	30	77	201	25/09/2024 09:38:32
1970	30	77	205	25/09/2024 09:38:31
1969	30	77	201	25/09/2024 09:38:30
1968	30	77	209	25/09/2024 09:38:29
1967	30	77	203	25/09/2024 09:38:28
1966	30	77	203	25/09/2024 09:38:27
1965	30	77	204	25/09/2024 09:38:26
1964	30	77	203	25/09/2024 09:38:25

1

2

3

4

...

198

- Hệ thống lưu thành công dữ liệu đọc được từ phần cứng  
- Người dùng có thể tìm kiếm, sắp xếp và lọc các dữ liệu