

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



BÁO CÁO BÀI TẬP LỚN
IoT và Ứng dụng

Chủ đề: Xây dựng hệ thống giám sát và điều khiển các thiết bị

Khoa Công nghệ thông tin 1

Họ và tên: Nguyễn Quang Minh

Mã sinh viên: B21DCCN531

Lớp sinh viên: D21CNPM2

Nhóm lớp: Nhóm 06

Giảng viên hướng dẫn: Nguyễn Quốc Uy

Hà Nội, 2024

Mục lục

I . Giới thiệu	3
1.1. Đặt vấn đề.....	3
1.2. Mục tiêu bài toán	3
1.3. Phạm vi bài toán	3
1.3.1 Thiết bị sử dụng	3
1.3.2 Công nghệ sử dụng	8
II. Giao diện, thiết kế tổng thể.....	11
2.1. Giao diện web.....	11
2.2. Giao diện phần cứng.....	13
III. Thiết kế chi tiết.....	14
3.1. Sequence Diagram	14
IV. Thực nghiệm và kết quả	14
4.1. Embedded Code.....	14
4.2. FrontendCode	18
4.3. BackendCode	20
4.4. Kết quả	24
V. Kết luận	24

I . Giới thiệu

1.1. Đặt vấn đề

Internet of Things (IoT), hay còn gọi là Internet vạn vật, là một mạng lưới các thiết bị thông minh được kết nối với nhau qua internet. Các thiết bị này có khả năng thu thập, trao đổi dữ liệu và thực hiện các tác vụ một cách tự động mà không cần sự can thiệp của con người. IoT đã và đang thay đổi cách chúng ta sống và làm việc, từ việc quản lý nhà thông minh, chăm sóc sức khỏe, đến việc tối ưu hóa quy trình sản xuất trong công nghiệp.

Ứng dụng của IoT rất đa dạng và phong phú. Trong lĩnh vực y tế, các thiết bị đeo thông minh có thể theo dõi sức khỏe và gửi dữ liệu đến bác sĩ để theo dõi và chẩn đoán từ xa. Trong cuộc sống hàng ngày, IoT giúp tạo ra các ngôi nhà thông minh với các thiết bị có thể điều khiển từ xa như đèn, điều hòa, và hệ thống an ninh. Bên cạnh đó IoT còn giúp chúng ta xem được các chỉ số ví dụ như temperature, humidity, light,.. dựa trên các cảm biến điện tử.

1.2. Mục tiêu bài toán

Việc xây dựng hệ thống giám sát các thông số về nhiệt độ, độ ẩm, ánh sáng và điều khiển các thiết bị như đèn, điều hòa, quạt nhằm đạt được các mục tiêu:

- Giám sát thời gian thực: Hệ thống cần thu thập dữ liệu từ các cảm biến nhiệt độ, độ ẩm, và ánh sáng một cách liên tục và chính xác. Dữ liệu thu thập được cần được hiển thị dưới dạng biểu đồ và các chỉ số trực quan để người dùng dễ dàng theo dõi. Điều này giúp người dùng có cái nhìn tổng quan về tình trạng môi trường và có thể đưa ra các quyết định kịp thời.

- Cảnh báo và phản ứng nhanh: Người dùng có thể thiết lập các ngưỡng cảnh báo cho nhiệt độ, độ ẩm, và ánh sáng. Khi các thông số vượt quá ngưỡng, hệ thống sẽ gửi cảnh báo ngay lập tức. Hệ thống cũng có khả năng tự động điều khiển các thiết bị như đèn, điều hòa, và quạt dựa trên các thông số thu thập được. Ví dụ, nếu nhiệt độ quá cao, hệ thống sẽ tự động bật điều hòa để duy trì môi trường thoải mái và an toàn.

1.3. Phạm vi bài toán

1.3.1 Thiết bị sử dụng

a. Node MCU ESP 8266

- Thông số kỹ thuật:

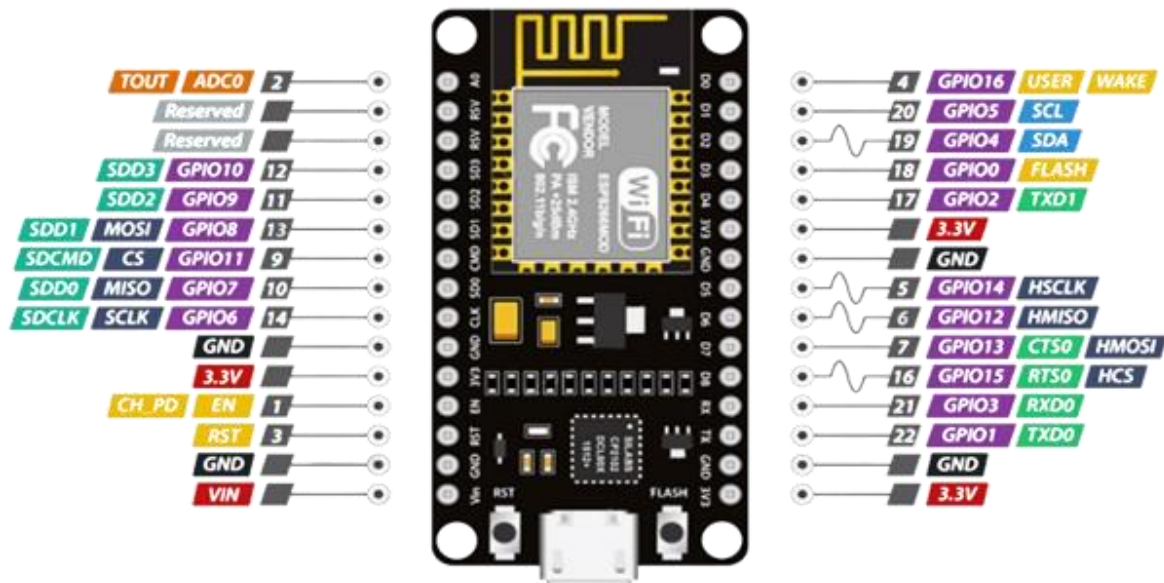
- + Hỗ trợ chuẩn 802.11 b/g/n.

- + 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.

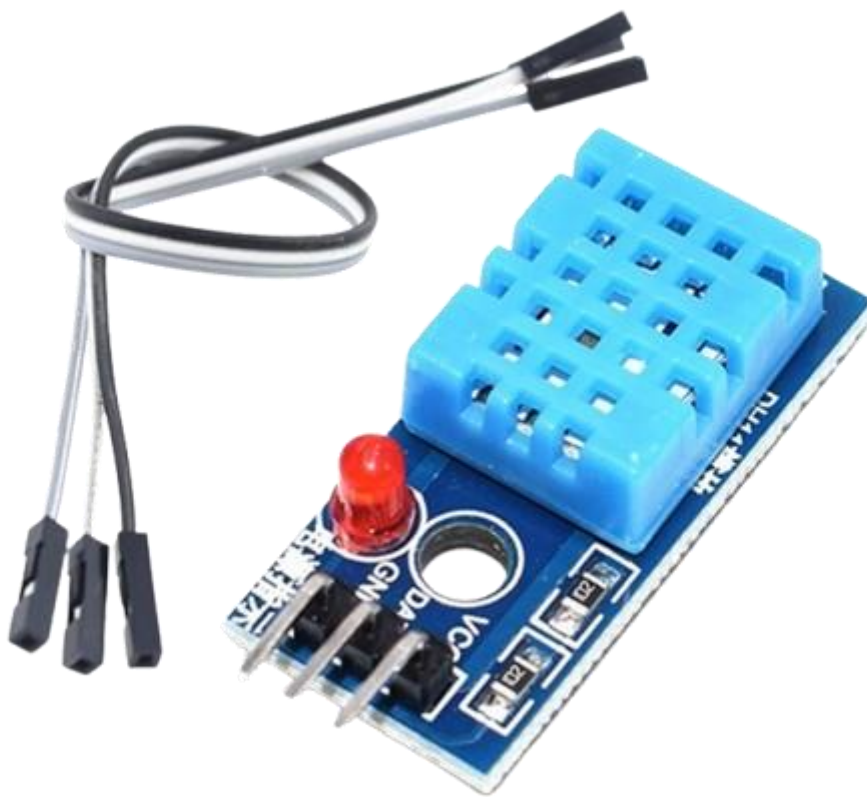


Hình 1: Module esp8266 CP2102

- 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 chạy.
- + USB: Cổng USB được sử dụng để kết nối board với máy tính để lập trình và cung cấp nguồn.

b. 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 $\pm 5\%$ RH
- Phạm vi cảm biến nhiệt độ: $0^{\circ}\text{C} \sim 50^{\circ}\text{C}$, sai số $\pm 2^{\circ}\text{C}$
- Tần số lấy mẫu tối đa: 1Hz (1 giây 1 lần)
- Kích thước: 23 * 12 * 5 mm



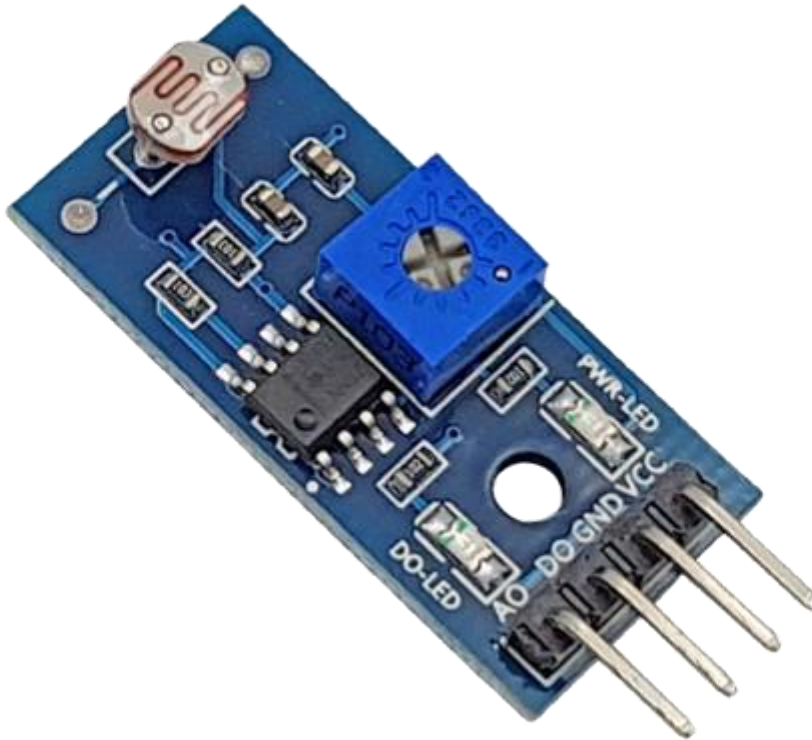
Hình 2: Cảm biến nhiệt độ , độ ẩm DHT11

c. 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 .

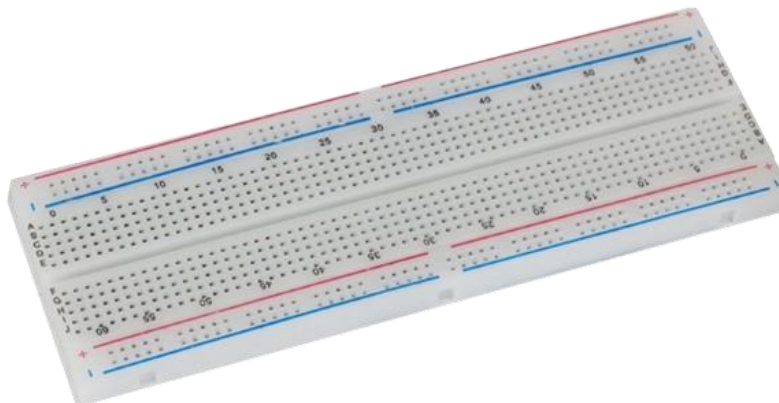
+ Kích thước: 32 x 14mm



Hình 3: Cảm biến ánh sáng

d. 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 4: Board test

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



Hình 5: Điện trở

f. Dây nối chân các thiết bị



Hình 6: Dây nối

g. Led tự đổi màu nhấp nháy 5mm, 7 màu 2 chân (1,5-3V)



Hình 7: Đèn LED

1.3.2 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 module Arduino.
- Nó có các phiên bản cho các hệ điều hành như MAC, Windows, Linux và chạy trên nền tảng Java đi kèm với các chức năng và lệnh có sẵn đóng vai trò quan trọng để gỡ lỗi, chỉnh sửa và biên dịch mã trong môi trường.
- Có rất nhiều các module Arduino như Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro và nhiều module khác. Mỗi module chứa một bộ vi điều khiển trên bo mạch được lập trình và chấp nhận thông tin dưới dạng mã.
- Môi trường IDE chủ yếu chứa hai phần cơ bản: Trình chỉnh sửa và Trình biên dịch, phần đầu sử dụng để viết mã được yêu cầu và phần sau được sử dụng để biên dịch và tải mã lên module Arduino.

Môi trường này hỗ trợ cả ngôn ngữ C và C ++

b. Mosquitto

Mosquitto là một MQTT Broker mã nguồn mở cho phép thiết bị truyền nhận dữ liệu theo giao thức MQTT version 5.0, 3.1.1 và 3.1 – Một giao thức nhanh, nhẹ theo mô hình publish/subscribe được sử dụng rất nhiều trong lĩnh vực Internet of Things. Mosquitto cung cấp một thư viện viết bằng ngôn ngữ C để triển khai các MQTT Client và có thể dễ dàng sử dụng bằng dòng lệnh: “mosquitto_pub” và “mosquitto_sub”.

- Ưu điểm:

- + Ưu điểm nổi bật của Mosquitto là tốc độ truyền nhận và xử lý dữ liệu nhanh, độ ổn định cao, được sử dụng rộng rãi và phù hợp với những ứng dụng embedded.
- + Mosquitto rất nhẹ và phù hợp để sử dụng trên tất cả các thiết bị.

+ Ngoài ra, Mosquitto cũng được hỗ trợ các giao thức TLS/SSL (các giao thức nhằm xác thực server và client, mã hóa các message để bảo mật dữ liệu).

- Nhược điểm:

+ Một số nhược điểm của mosquitto là khó thiết kế khi làm những ứng dụng lớn và ít phương thức xác thực thiết bị nên khả năng bảo mật vẫn chưa tối ưu.

c. ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, chuyên dùng để xây dựng giao diện người dùng (UI). ReactJS đặc biệt hữu ích cho các ứng dụng đơn trang (SPA) nơi mà trải nghiệm người dùng cần phải nhanh và tương tác.



Hình 8: ReactJS

- Ứng dụng của ReactJS trong dự án giám sát IoT:

+ Tạo giao diện người dùng trực quan: Sử dụng các component để hiển thị dữ liệu từ các cảm biến dưới dạng biểu đồ và các chỉ số trực quan. Ví dụ, bạn có thể sử dụng thư viện như Chart.js kết hợp với ReactJS để tạo các biểu đồ thời gian thực.

+ Cập nhật dữ liệu theo thời gian thực: Sử dụng Virtual DOM để cập nhật giao diện một cách nhanh chóng và hiệu quả khi dữ liệu thay đổi. Điều này đảm bảo rằng người dùng luôn có thông tin mới nhất về các thông số môi trường.

+ Điều khiển thiết bị từ xa: Tạo các component điều khiển để người dùng có thể bật/tắt đèn, điều hòa, quạt từ xa thông qua giao diện web. Bạn có thể sử dụng các thư viện như Axios để gửi yêu cầu HTTP đến máy chủ điều khiển các thiết bị.

=> ReactJS không chỉ giúp bạn xây dựng một giao diện người dùng mạnh mẽ và linh hoạt mà còn giúp bạn dễ dàng mở rộng và bảo trì ứng dụng trong tương lai. Với các lợi ích và tính năng mạnh mẽ của ReactJS, bạn có thể tạo ra một hệ thống giám sát và điều khiển IoT hiệu quả và tiện lợi

d. NodeJS, ExpressJS

NodeJS là một môi trường chạy JavaScript phía server, cho phép bạn xây dựng các ứng dụng mạng hiệu suất cao và mở rộng. ExpressJS là một framework web tối giản và linh hoạt cho NodeJS, cung cấp một bộ công cụ mạnh mẽ để xây dựng các ứng dụng web và API.



Hình 9: NodeJS và EXpressJS

- Ứng dụng của NodeJS, ExpressJS trong IoT

+ Xử lý dữ liệu từ cảm biến: Sử dụng NodeJS để thu thập và xử lý dữ liệu từ các cảm biến IoT. Ở đây ta sẽ sử dụng thư viện mqtt để kết nối và nhận dữ liệu từ các thiết bị IoT.

+ Tạo API RESTful: Sử dụng ExpressJS để tạo các API RESTful cho phép giao tiếp giữa frontend và backend. Các API này có thể được sử dụng để gửi dữ liệu cảm biến đến frontend và nhận các lệnh điều khiển từ người dùng.

+ Quản lý trạng thái thiết bị: Sử dụng ExpressJS để quản lý trạng thái của các thiết bị như đèn, điều hòa, và quạt. Bạn có thể tạo các endpoint để bật/tắt thiết bị dựa trên dữ liệu cảm biến hoặc lệnh từ người dùng.

=> NodeJS và ExpressJS không chỉ giúp bạn xây dựng một backend mạnh mẽ và linh hoạt mà còn giúp bạn dễ dàng mở rộng và bảo trì ứng dụng trong tương lai. Với các lợi ích và tính năng mạnh mẽ của NodeJS và ExpressJS, bạn có thể tạo ra một hệ thống giám sát và điều khiển IoT hiệu quả và tiện lợi.

e. MySQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được sử dụng rộng rãi để lưu trữ và quản lý dữ liệu. MySQL nổi tiếng với hiệu suất cao, độ tin cậy và khả năng mở rộng, làm cho nó trở thành lựa chọn phổ biến cho các ứng dụng web và IoT.

- Ứng dụng của MySQL trong dự án giám sát IoT

+ Lưu trữ dữ liệu cảm biến: MySQL có thể lưu trữ dữ liệu từ các cảm biến một cách hiệu quả, cho phép bạn truy xuất và phân tích dữ liệu một cách dễ dàng. Bạn có thể thiết kế các bảng cơ sở dữ liệu để lưu trữ thông tin về nhiệt độ, độ ẩm, ánh sáng và trạng thái của các thiết bị.

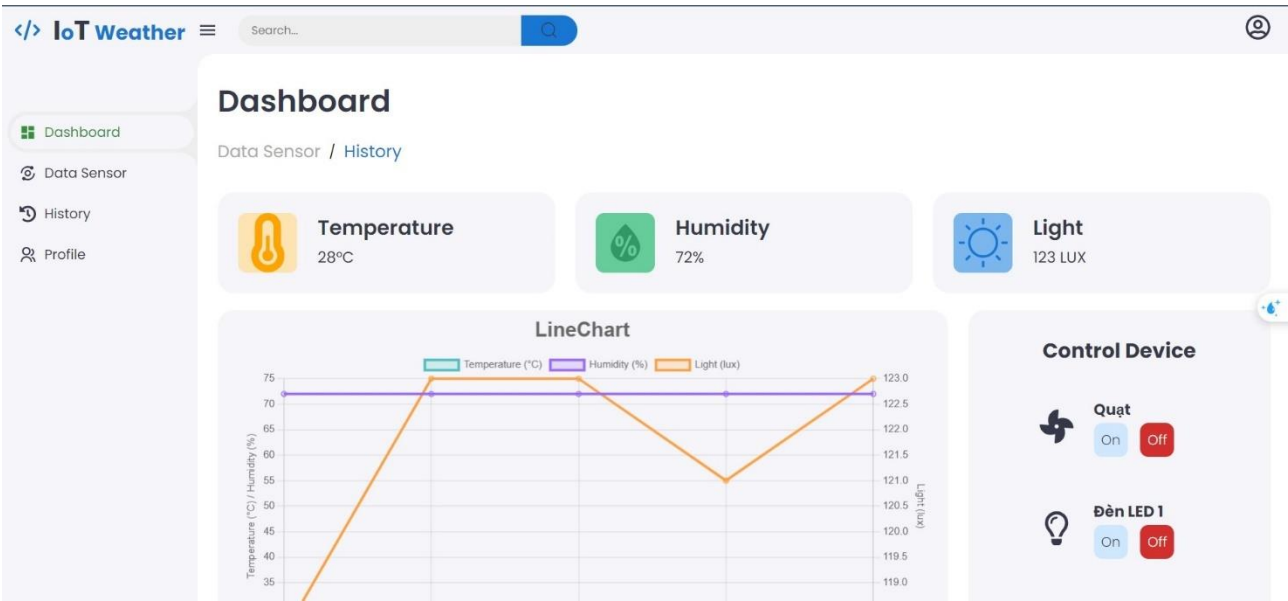
+ Quản lý người dùng và quyền truy cập: MySQL cho phép bạn quản lý người dùng và quyền truy cập, đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập và điều khiển các thiết bị.

+ Tích hợp với backend: Sử dụng NodeJS và ExpressJS, bạn có thể dễ dàng kết nối và tương tác với cơ sở dữ liệu MySQL. Ví dụ, bạn có thể tạo các API để thêm, cập nhật, và truy xuất dữ liệu từ MySQL, giúp frontend có thể hiển thị dữ liệu thời gian thực và điều khiển các thiết bị.

II. Giao diện, thiết kế tổng thể

2.1. Giao diện web

a. Giao diện trang Dashboard



Hình 10: Giao diện Dashboard

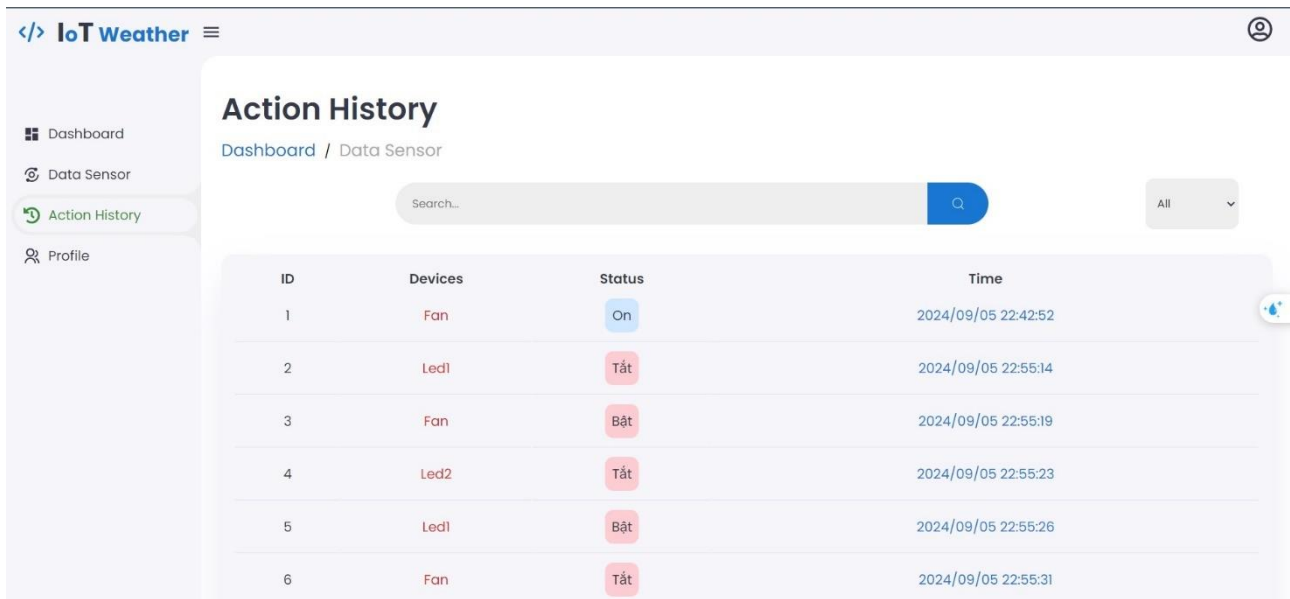
b. Giao diện trang Data Sensor

The screenshot shows the 'Data Sensor' page. It has a sidebar with 'Data Sensor' selected. The main area is titled 'Data Sensor' with a breadcrumb 'Dashboard / Action History'. It includes a search bar and a dropdown menu set to 'All'. Below is a table of sensor data.

ID	Temperature	Humidity	Light	Time
1	30	70	500	2024/09/20 09:51:02
2	20	60	600	2024/09/20 09:51:26
3	29	74	136	2024/09/22 22:08:37
4	29	74	139	2024/09/22 22:08:47
5	29	74	135	2024/09/22 22:08:57
6	29	74	139	2024/09/22 22:09:07
7	28	72	148	2024/09/23 20:19:21
8	28	72	146	2024/09/23 20:19:31
9	28	72	141	2024/09/23 20:19:41

Hình 11: Giao diện DataSensor

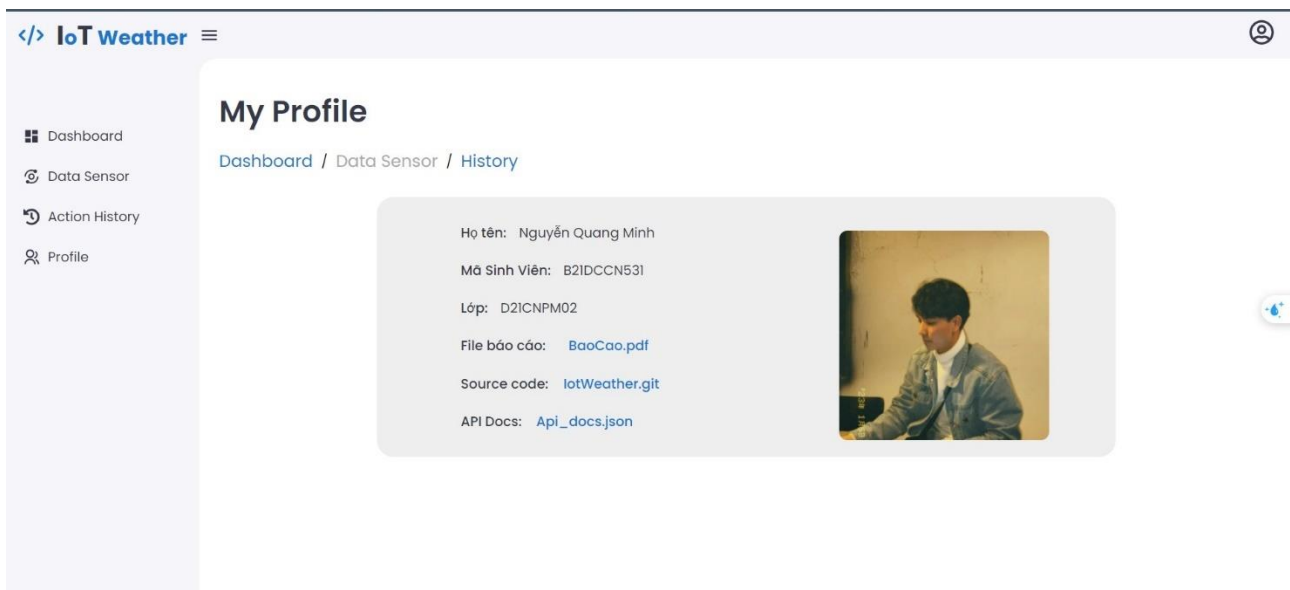
c. Giao diện trang Action History



ID	Devices	Status	Time
1	Fan	On	2024/09/05 22:42:52
2	Led1	Tắt	2024/09/05 22:55:14
3	Fan	Bật	2024/09/05 22:55:19
4	Led2	Tắt	2024/09/05 22:55:23
5	Led1	Bật	2024/09/05 22:55:26
6	Fan	Tắt	2024/09/05 22:55:31

Hình 12: Giao diện Action History

d. Giao diện trang Profile



My Profile

Dashboard / Data Sensor / History

Họ tên: Nguyễn Quang Minh

Mã Sinh Viên: B21DCCN531

Lớp: D21CNPM02

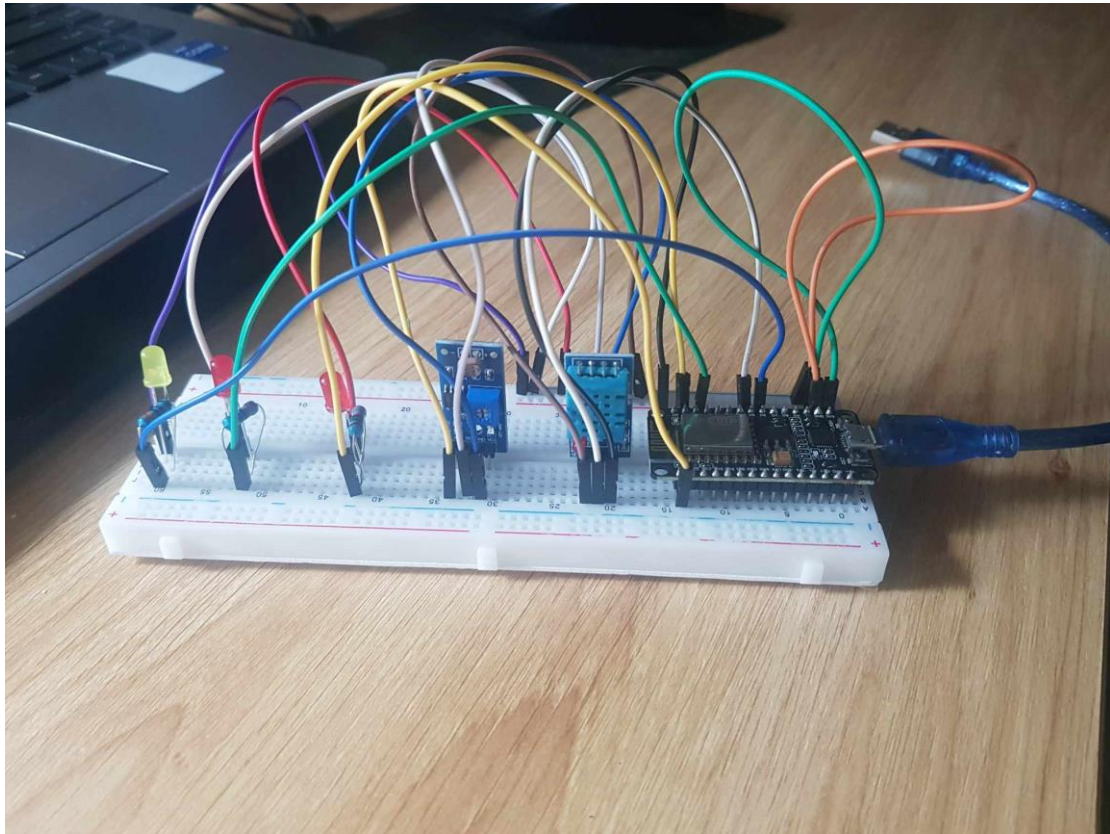
File báo cáo: [BaoCao.pdf](#)

Source code: [IoTWeather.git](#)

API Docs: [Api_docs.json](#)

Hình 13: Giao diện trang Profile

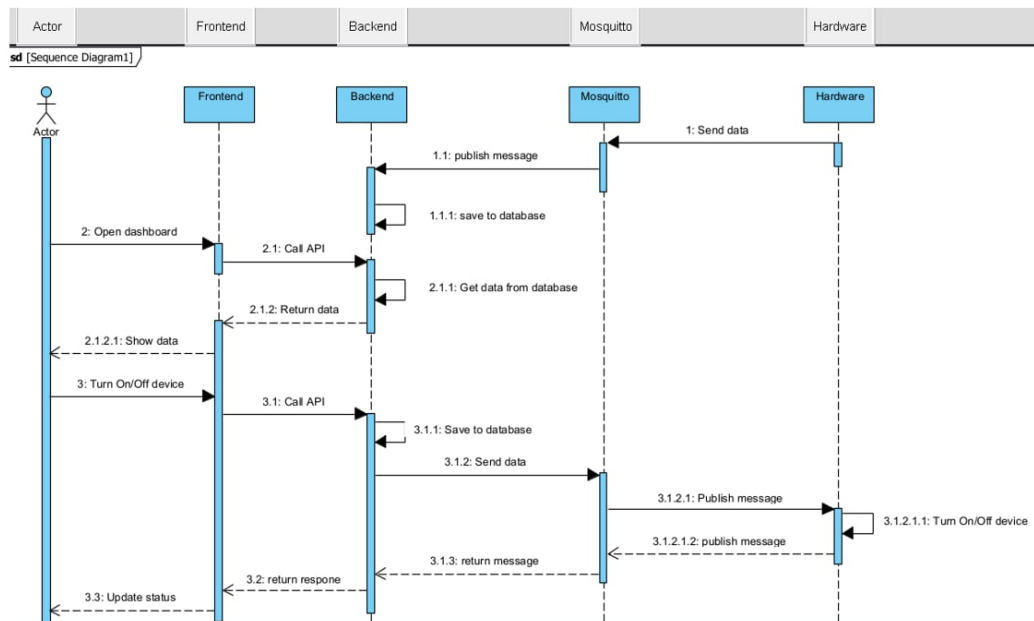
2.2. Giao diện phần cứng



Hình 14: Giao diện phần cứng

III. Thiết kế chi tiết

3.1. Sequence Diagram



Hình 15: Sequence Diagram

IV. Thực nghiệm và kết quả

4.1. Embedded Code

```
1 #include "DHT.h"
2 #include <ESP8266WiFi.h>
3 #include <Ticker.h>
4 #include <AsyncMqttClient.h>
5 #include <ArduinoJson.h>
```

- Khai báo các thư viện cần thiết gồm:

+ ESP8266: Thư viện hỗ trợ wifi cho NodeMCU ESP8266

+ DHT.h: Thư viện này cung cấp các chức năng để kết nối ESP8266 với mạng WiFi, cho phép thiết bị giao tiếp qua internet.

+ Ticker.h: Thư viện này cho phép bạn tạo các tác vụ định kỳ (periodic tasks) mà không cần sử dụng delay(), giúp chương trình chạy mượt mà hơn.

+ AsyncMqttClient.h: Thư viện này cung cấp các chức năng để kết nối và giao tiếp với một MQTT broker một cách không đồng bộ (asynchronous), giúp gửi và nhận dữ liệu từ các thiết bị IoT.

+ ArduinoJson.h: Thư viện này giúp bạn dễ dàng tạo và phân tích dữ liệu JSON.


```
#define WIFI_SSID "Loi"
#define WIFI_PASSWORD "0976300109"
```

- Khai báo các thông tin về Wifi

```
#define MQTT_HOST IPAddress(192, 168, 1, 29)
#define MQTT_PORT 1893
#define MQTT_PUB_SENSOR "datasensor" // topic
#define MQTT_USERNAME "minh"
#define MQTT_PASSWORD "b21dccn531"
```

- Khai báo các thông tin về MQTT broker

```
#define DHTPIN 14
#define LIGHT_SENSOR_PIN A0
#define LED1_PIN D1
#define LED2_PIN D2
#define FAN_PIN D6
#define DHTTYPE DHT11
```

- Cấu hình cảm biến và thiết bị

- + DHTPIN: Chân GPIO14 kết nối với cảm biến DHT11.
- + LIGHT_SENSOR_PIN: Chân analog kết nối với cảm biến ánh sáng.
- + LED1_PIN và LED2_PIN: Chân GPIO1 và 2 điều khiển đèn LED.
- + FAN_PIN: Chân GPIO điều khiển quạt.
- + DHTTYPE: Loại cảm biến DHT (ở đây là DHT11)

```
AsyncMqttClient mqttClient;
Ticker mqttReconnectTimer;

WiFiEventHandler wifiConnectHandler;
WiFiEventHandler wifiDisconnectHandler;
Ticker wifiReconnectTimer;
```

- + AsyncMqttClient mqttClient: được sử dụng để quản lý kết nối và giao tiếp với MQTT broker một cách không đồng bộ (asynchronous). Nó giúp gửi và nhận dữ liệu từ các thiết bị IoT mà không làm gián đoạn các tác vụ khác.
- + Ticker mqttReconnectTimer: được sử dụng để tạo ra các tác vụ định kỳ. mqttReconnectTimer sẽ được sử dụng để thử kết nối lại với MQTT broker nếu kết nối bị mất.

- + WiFiEventHandler wifiConnectHandler và WiFiEventHandler wifiDisconnectHandler:
wifiConnectHandler sẽ xử lý các sự kiện khi thiết bị kết nối thành công với mạng WiFi, trong khi wifiDisconnectHandler sẽ xử lý các sự kiện khi thiết bị bị ngắt kết nối khỏi mạng WiFi.
- + Ticker wifiReconnectTimer: là một đối tượng Ticker được sử dụng để thử kết nối lại với mạng WiFi nếu kết nối bị mất.

```
void onMessage(char* topic, char* payload, AsyncMqttClientMessageProperties properties,
    DynamicJsonDocument doc(1024);
    deserializeJson(doc, payload);

    String device = doc["device"];
    String status = doc["status"];
    Serial.printf("Message arrived: [%s,%s]\n", device, status);

    // Điều khiển LED dựa trên thông tin JSON
    if (device == "LED1") {
        digitalWrite(LED1_PIN, (status == "On") ? HIGH : LOW);
    } else if (device == "LED2") {
        digitalWrite(LED2_PIN, (status == "On") ? HIGH : LOW);
    }
    else {
        digitalWrite(FAN_PIN, (status == "On") ? HIGH:LOW);
    }
}
```

- Hàm xử lý khi nhận được tin nhắn về điều khiển thiết bị từ MQTT broker
 - + DynamicJsonDocument doc(1024): Tạo một đối tượng JSON với kích thước bộ nhớ 1024 byte.
 - + deserializeJson(doc, payload): Phân tích chuỗi JSON từ payload và lưu trữ trong đối tượng doc.
 - + Kiểm tra tin nhắn nhận được là thiết bị nào, status nào để bật tắt đúng thiết bị.

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

- + Serial.begin(115200): Khởi tạo giao tiếp Serial với tốc độ baud rate 115200, giúp bạn theo dõi các thông báo và dữ liệu từ ESP8266 trên Serial Monitor.


```
dht.begin();

pinMode(LED1_PIN, OUTPUT);
pinMode(LED2_PIN, OUTPUT);
pinMode(FAN_PIN, OUTPUT);
```

- Khởi tạo cảm biến DHT và cấu hình chân GPIO

```
wifiConnectHandler = WiFi.onStationModeGotIP(onWifiConnect);
wifiDisconnectHandler = WiFi.onStationModeDisconnected(onWifiDisconnect);
```

- Xử lý sự kiện Wifi khi kết nối thành công và khi thiết bị ngắt kết nối

```
mqttClient.onConnect(onMqttConnect);
mqttClient.onDisconnect(onMqttDisconnect);
mqttClient.onMessage(onMessage);

mqttClient.setServer(MQTT_HOST, MQTT_PORT);
mqttClient.setCredentials(MQTT_USERNAME, MQTT_PASSWORD);
```

- Xử lý sự kiện MQTT khi kết nối thành công, ngắt kết nối và khi nhận được tin nhắn từ MQTT broker.

```
hum = dht.readHumidity();
temp = dht.readTemperature();
light = 1024 - analogRead(LIGHT_SENSOR_PIN);
```

- Đọc dữ liệu từ cảm biến

```
DynamicJsonDocument doc(1024);
doc["temperature"] = temp;
doc["humidity"] = hum;
doc["light"] = light;
```

- Tạo đối tượng JSON

```

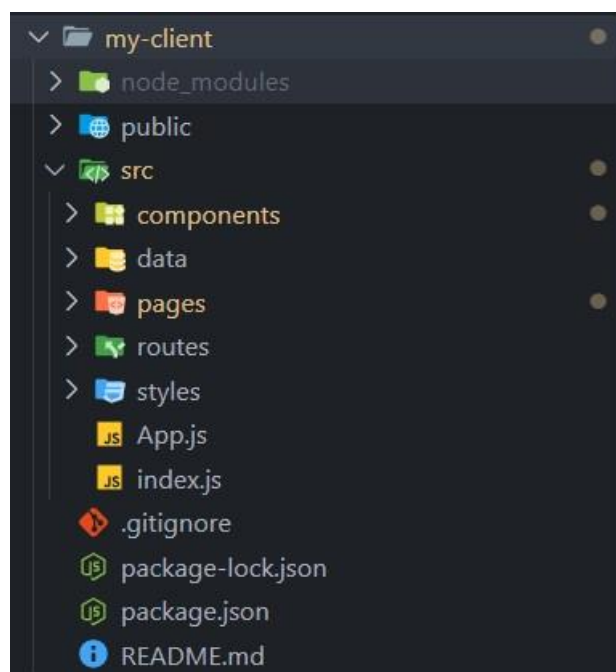
mqttClient.publish(MQTT_PUB_SENSOR, 1, true, jsonString.c_str());
Serial.print("Topic: ");
Serial.println(MQTT_PUB_SENSOR);
Serial.print("Message: ");
Serial.println(jsonString.c_str());
}

```

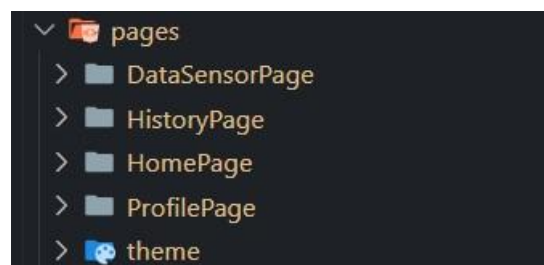
- Gửi dữ liệu đọc từ cảm biến qua MQTT

4.2. FrontendCode

- Cấu trúc thư mục bên phía FrontEnd:



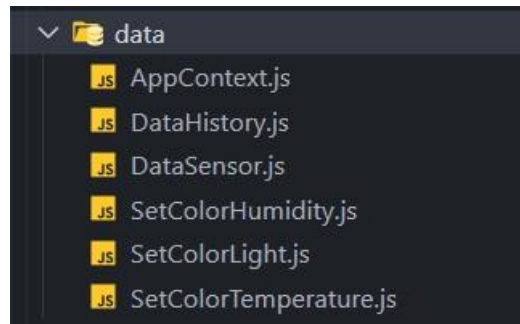
- Thư mục pages sẽ bao gồm các trang có trên web: dashboard, data sensor, action history, profile và cấu hình layout của các trang như: sidebar, navbar và component chính của mỗi trang.



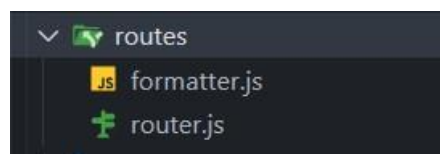
- Thư mục components bao gồm các components ở các pages, chúng ta sẽ chia các pages thành các components nhỏ để dễ quản lý.



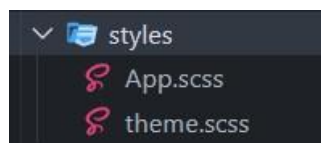
- Thư mục data bao gồm dữ liệu chúng ta fetch từ phía server và ở đây chúng ta thiết lập các hàm để giúp đổi màu nhiệt độ , độ ẩm, ánh sáng.



- Thư mục routes chứa các hàm và đường dẫn đến các trang web được chúng ta định nghĩa sẵn



- Thư mục styles chứa các style chung cho trang web, bao gồm màu sắc chung, các css chung cho button, input, breakcrule, ...



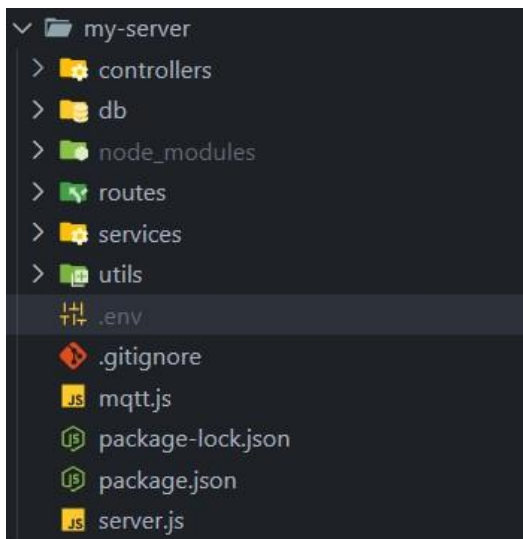
File app.js: sử dụng để thiết lập link đến các router, components và truyền vào các component props phù hợp.

```
const userRoute = [
  {
    path: ROUTER.USER.HOME,
    component: <HomePage dataSensor={dataSensor.data} dataHistory={dataHistory.data}/>,
  },
  {
    path: ROUTER.USER.STATICS,
    component: <DataSensorPage dataSensor={dataSensor.data} />,
  },
  {
    path: ROUTER.USER.HISTORY,
    component: <HistoryPage dataHistory={dataHistory.data} />,
  },
  {
    path: ROUTER.USER.PROFILE,
    component: <ProfilePage />,
  },
];

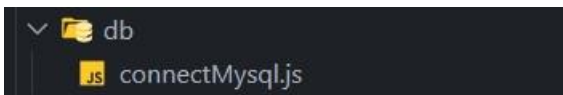
return (
  <AppProvider>
    <Router>
      <Layout>
        <Routes>
          {userRoute.map((route, index) => (
            <Route key={index} path={route.path} element={route.component} />
          ))}
        </Routes>
      </Layout>
    </Router>
  </AppProvider>
);
```

4.3. BackendCode

- Cấu trúc thư mục bên phía Backend



- Thư mục lưu trữ các kết nối đến database

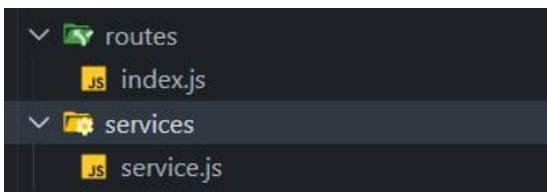
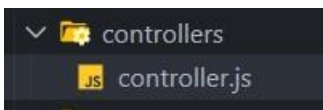


- Connect đến mysql với host, user, password, database

```
const mysql = require("mysql"); 272k (gzipped: 272k)

const conn = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "password",
  database: "ecommerce",
});
```

- Thiết kế viết RestfullAPI theo mô hình MVC gồm 3 thư mục là routes, controllers, services với các api tương ứng cho bảng datasensor, action history, api publish về temperature, humidity, light.



- Thư mục utils chứa các hàm để hỗ trợ việc format data ở đây ta chỉ cần dùng hàm formatter để định dạng lại thời gian pub, sub để trả lại data cho client



- Thực hiện kết nối đến MQTT broker thông qua baseUri và option: user, pass . Và thiết lập nhận subscribe với các topic là datasensor và action_history

```
const baseUri = "mqtt://localhost:1893";
const option = {
  username: "minh",
  password: "b21dccn531",
};

const client = mqtt.connect(baseUri, option);

client.on("connect", () => {
  console.log("Kết nối thành công đến MQTT server");
  client.subscribe("datasensor", (err) => {
    if (!err) {
      console.log("Subscribed to topic datasensor");
    } else {
      throw err;
    }
  });
  client.subscribe("action_history", (err) => {
    if (!err) {
      console.log("Subscribed to topic action_history");
    } else {
      throw err;
    }
  });
});
```


- Nhận tin nhắn với các topic tương ứng và thực hiện lưu chúng vào cơ sở dữ liệu mySql

```
client.on("message", (topic, message) => {
  console.log(`Topic ${topic}: ${message}`);

  let data;
  data = JSON.parse(message.toString());
  if (topic === "datasensor") {

    client.publish("datasensor_client" , JSON.stringify(data));
    const { temperature, humidity, light } = data;
    const query = `INSERT INTO datasensor (temperature, humidity, light, time)

    conn.query(query, [temperature, humidity, light], (err, result) => {
      if (err) {
        throw err;
      }
      console.log("Thêm vào database thành công với topic datasensor");
    });
  }
});
```

- Các cấu hình cần thiết bên server:
- + Thực hiện khai báo port để khởi động server
- + Thực hiện kết nối đến database và mqtt
- + Khai báo các middleware cần thiết
- + Chạy server, lắng nghe tại cổng port 8000

```
const express = require('express');
const app = express();
const cors = require('cors'); 4.5k (gzipped: 1.9k)
require('./db/connectMysql');
require('./mqtt');
const dataSensor = require('./routes/Data/data.router');
const dataMqtt = require('./routes/PubSubMqtt/pubsub.router');

app.get('/', (req, res, next) => {
  console.log("Hello");
})
app.use(cors({
  origin: 'http://localhost:3000',
  methods: ['GET', 'POST'],
}));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use('/api', dataSensor);
app.use('/api', dataMqtt);

const port = 8000;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
})
```

4.4. Kết quả

Trong quá trình phát triển hệ thống IoT, dự án này đã thực hiện thành công các chức năng đo lường thời gian thực, điều khiển thiết bị điện tử 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.

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 mới cập nhật UI tương ứng.

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

V. Kết luận

Việc xây dựng hệ thống giám sát các thông số về nhiệt độ, độ ẩm, ánh sáng và điều khiển các thiết bị như đèn, điều hòa, quạt là một ứng dụng quan trọng của IoT, mang lại nhiều lợi ích thiết thực. Hệ thống này không chỉ giúp giám sát môi trường một cách chính xác và liên tục mà còn cung cấp khả năng điều khiển từ xa, tối ưu hóa việc sử dụng năng lượng và tăng cường tiện nghi, an toàn cho người dùng.

Sử dụng các công nghệ hiện đại như ReactJS cho frontend, NodeJS và ExpressJS cho backend, cùng với MySQL để quản lý cơ sở dữ liệu, chúng ta có thể xây dựng một hệ thống mạnh mẽ, linh hoạt và dễ dàng mở rộng. ReactJS giúp tạo ra giao diện người dùng trực quan và tương tác, trong khi NodeJS và ExpressJS cung cấp nền tảng vững chắc cho việc xử lý dữ liệu và điều khiển thiết bị. MySQL đảm bảo việc lưu trữ và quản lý dữ liệu một cách hiệu quả và bảo mật.

Tóm lại, việc áp dụng các công nghệ này không chỉ giúp chúng ta xây dựng một hệ thống giám sát và điều khiển IoT hiệu quả mà còn mở ra nhiều cơ hội phát triển và ứng dụng trong các lĩnh vực khác nhau, từ quản lý tòa nhà, sản xuất công nghiệp đến nông nghiệp thông minh. Đây là một bước tiến quan trọng trong việc tận dụng sức mạnh của IoT để cải thiện chất lượng cuộc sống và hiệu quả công việc.