

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: Nguyễn Tài Quân

Mã sinh viên: B21DCCN614

Hà Nội, 2024

MỤC LỤC

I. GIỚI THIỆU	3
1.1. IOT (INTERNET OF THINGS - MẠNG LƯỚI VẠN VẬT KẾT NỐI)	3
1.2. MÔ TẢ VÀ MỤC ĐÍCH CỦA ĐỀ TÀI.....	3
1.3. THIẾT BỊ SỬ DỤNG.....	4
1.4. CÔNG NGHỆ SỬ DỤNG	9
II. GIAO DIỆN.....	12
2.1. GIAO DIỆN WEBSITE.....	12
2.2. GIAO DIỆN API, API DOCS	14
2.3. GIAO DIỆN THIẾT BỊ.....	16
III. THIẾT KẾ CHI TIẾT	16
3.1. THIẾT KẾ HỆ THỐNG.....	16
3.2. SEQUENCE DIAGRAM.....	17
IV. CODE.....	17
4.1. ARDUINO CODE.....	17
V. KẾT QUẢ THU ĐƯỢC	20
5.1. TỔNG QUAN	20
5.2. TỪ DHT11 VÀ CẢM BIẾN ÁNH SÁNG	20
5.3. KHI NGƯỜI DÙNG ĐIỀU KHIỂN THIẾT BỊ.....	21
VI. TÀI LIỆU THAM KHẢO	21

I. Giới thiệu

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

1.1.1. Khái niệm

Mạng lưới vạn vật kết nối (IoT) là một hệ thống mạng lưới 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. IoT cho phép các thiết bị không chỉ kết nối với mạng internet, mà còn có khả năng truyền nhận thông tin và tương tác với nhau mà không cần sự can thiệp của con người.

1.1.2. Các thành phần của IoT

- Thiết bị IoT: Đây là các thiết bị thông minh được trang bị cảm biến, các công nghệ kết nối và khả năng thu thập, gửi và nhận dữ liệu.
- Mạng: Đảm bảo việc truyền dữ liệu giữa các thiết bị IoT và hệ thống xử lý dữ liệu.
- Hệ thống xử lý dữ liệu: Bao gồm các máy chủ và hệ thống phần mềm để lưu trữ, xử lý và phân tích dữ liệu từ các thiết bị IoT.
- Ứng dụng và dịch vụ: Cung cấp các ứng dụng và dịch vụ dựa trên dữ liệu từ các thiết bị IoT, nhằm giúp cải thiện cuộc sống và công việc của con người.

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 và cơ sở dữ liệu MySQL để lưu trữ và quản lý dữ liệu. Các cảm biến sẽ thu thập dữ

liệu nhiệt độ và độ ẩm trong thời gian thực, và người dùng có thể điều khiển các thiết bị điện tử 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:

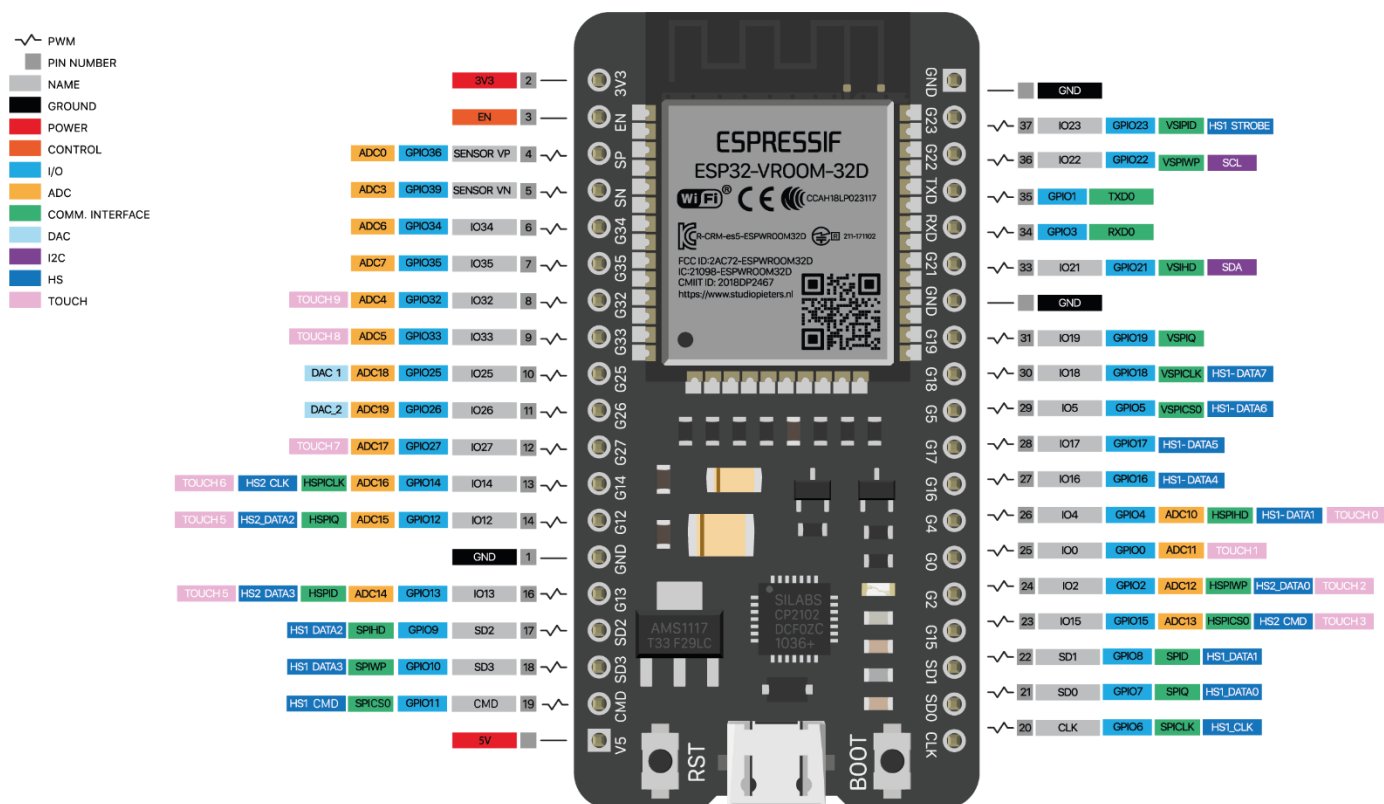
- Giám sát và cảnh báo:
 - Thu thập và hiển thị dữ liệu nhiệt độ, độ ẩm từ các cảm biến một cách trực quan trên website.
 - Cảnh báo người dùng khi các chỉ số môi trường vượt quá ngưỡng an toàn (ví dụ: nhiệt độ quá cao hoặc độ ẩm quá thấp).
- Điều khiển thiết bị từ xa:
 - Cho phép người dùng bật/tắt 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. ESP32

- Thông số kỹ thuật:
 - CPU: Xtensa Dual-Core LX6 microprocessor.
 - Chạy hệ 32 bit
 - Tốc độ xử lý 160MHZ up to 240 MHz
 - Tốc độ xung nhịp đọc flash chip 40mhz --> 80mhz (tùy chỉnh khi lập trình)
 - RAM: 520 KByte SRAM
 - Wi-Fi: 802.11 b/g/n/e/i
 - Bluetooth: v4.2 BR/EDR and BLE Hỗ trợ cả 2 giao tiếp TCP và UDP.

- Các cảm biến tích hợp trên ESP32: 1 cảm biến Hall, 1 cảm biến đo nhiệt độ, cảm biến chạm (điện dung) với 10 đầu vào khác nhau.
- Nhiệt độ hoạt động: -40 +85
- Điện áp hoạt động: 2.2 – 3.6V
- Số cổng GPIOs: 34



Hình 1: ESP32 WROOM

- Các chân ESP32:

- GND: Chân mẫu điện âm.
- 3V3: Chân nguồn cung cấp 3.3V.
- RST: Chân khởi động lại board.
- GPIO0 đến GPIO39: Các chân GPIO đa năng có thể được cấu hình làm đầu vào hoặc đầu ra.
- ADC1: GPIO32 đến GPIO39 và ADC2: GPIO0 đến GPIO15. Các chân ADC này có thể dùng để đọc tín hiệu tương tự với độ phân giải lên tới 12-bit.
- DAC Pins (Digital to Analog Converter - Chân Chuyển Đổi Tín Hiệu Số

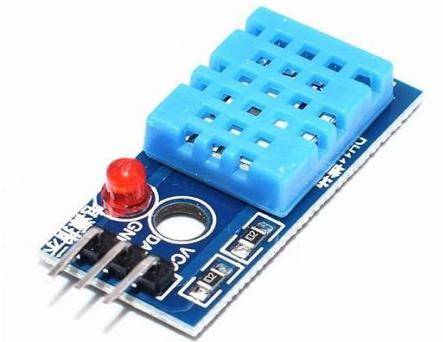
Sang Tương Tự

- 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 DHT11

- Thông số kỹ thuật:

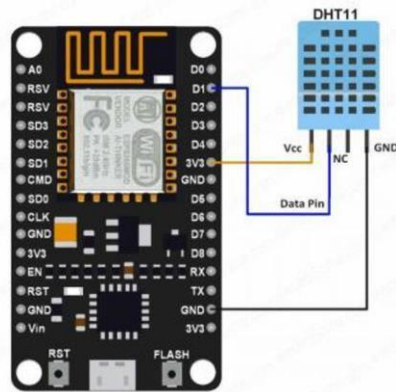
- Điện áp hoạt động: 3V - 5V DC
- Dòng điện tiêu thụ: 2.5mA
- Phạm vi cảm biến độ ẩm: 20% - 90% RH, sai số $\pm 5\% RH$
- Phạm vi cảm biến nhiệt độ: $0^{\circ}C \sim 50^{\circ}C$, sai số $\pm 2^{\circ}C$
- Tần số lấy mẫu tối đa: 1Hz (1 giây 1 lần)
- Kích thước: 23 * 12 * 5 mm



Hình 2: Cảm biến DHT11

- Sơ đồ chân DHT11:

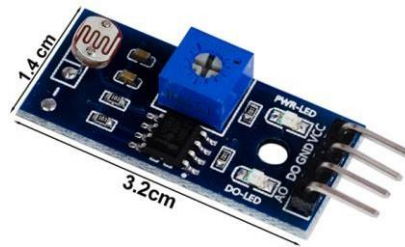
Số chân	Tên chân	Mô tả
1	VCC	Nguồn 3.3V đến 5V
2	Data	Đầu ra nhiệt độ độ ẩm thông qua dữ liệu nối tiếp
3	NC	Không có kết nối và do đó không sử dụng
4	Ground	Nối đất



Hình 3: Sơ đồ chân nối DHT11

1.3.3. Cảm biến cường độ ánh sáng quang trở

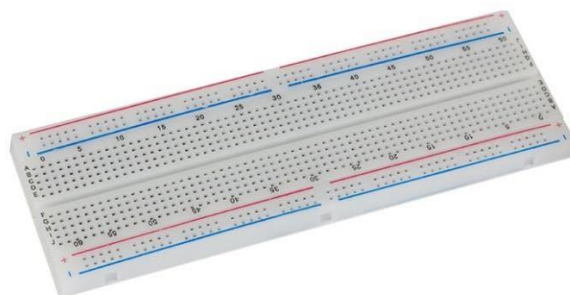
- Thông số kỹ thuật
 - Điện áp hoạt động: 3.3V – 5V
 - Kết nối 4 chân với 2 chân cấp nguồn (VCC và GND) và 2 chân tín hiệu ngõ ra (AO và DO).
 - Hỗ trợ cả 2 dạng tín hiệu ra Analog và TTL. Ngõ ra Analog 0– 5V tỷ lệ thuận với cường độ ánh sáng, ngõ TTL tích cực mức thấp.
 - Độ nhạy cao với ánh sáng được 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, 1 màu 2 chân (1,8-3V)



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: React

- React là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, dùng để xây dựng giao diện người dùng (UI). Được phát hành lần đầu tiên vào năm 2013, React đã nhanh chóng trở thành một trong những thư viện phổ biến nhất cho việc phát triển các ứng dụng web, nhờ cách tiếp cận hiệu quả để quản lý và cập nhật giao diện người dùng khi dữ liệu thay đổi.
- Đặc điểm chính:
 - Component-Based (Dựa trên các thành phần): React cho phép bạn chia giao diện người dùng thành các "component" độc lập. Mỗi component là một

phần nhỏ của giao diện, có thể tái sử dụng và tự quản lý trạng thái riêng. Ví dụ, một nút, một thanh điều hướng, hoặc một danh sách các bài viết đều có thể là một component.

- Virtual DOM (DOM Ảo): React sử dụng một "Virtual DOM" để tăng hiệu suất. Khi trạng thái của ứng dụng thay đổi, React tạo ra một bản sao ảo của DOM, so sánh nó với DOM thật để xác định sự khác biệt, và chỉ cập nhật các thành phần cần thiết thay vì làm mới toàn bộ DOM.
- Unidirectional Data Flow (Luồng dữ liệu một chiều): Đồng bộ hóa dữ liệu giữa giao diện và logic ứng dụng một cách tự động.
- JSX (JavaScript XML): React sử dụng một cú pháp gọi là JSX, cho phép viết mã HTML trong JavaScript. JSX giúp mô tả UI dễ hiểu hơn và cho phép tận dụng sức mạnh của JavaScript trong việc tạo giao diện

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.

1.4.3. 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.

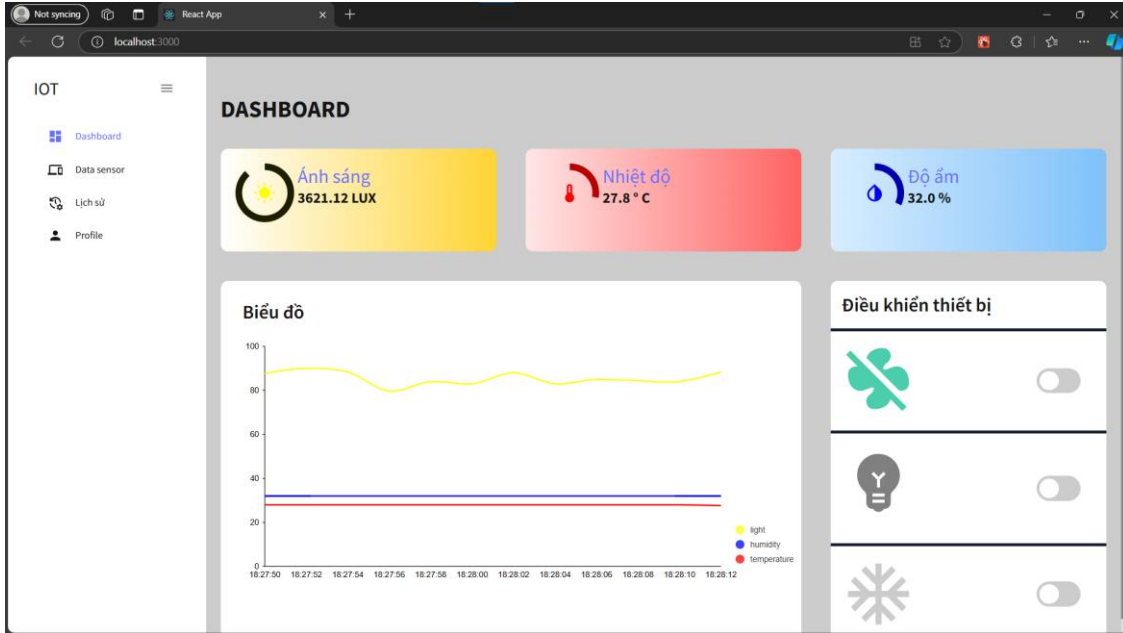
1.4.4. Mosquitto

- 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

- 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 9: Giao diện Dashboard

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

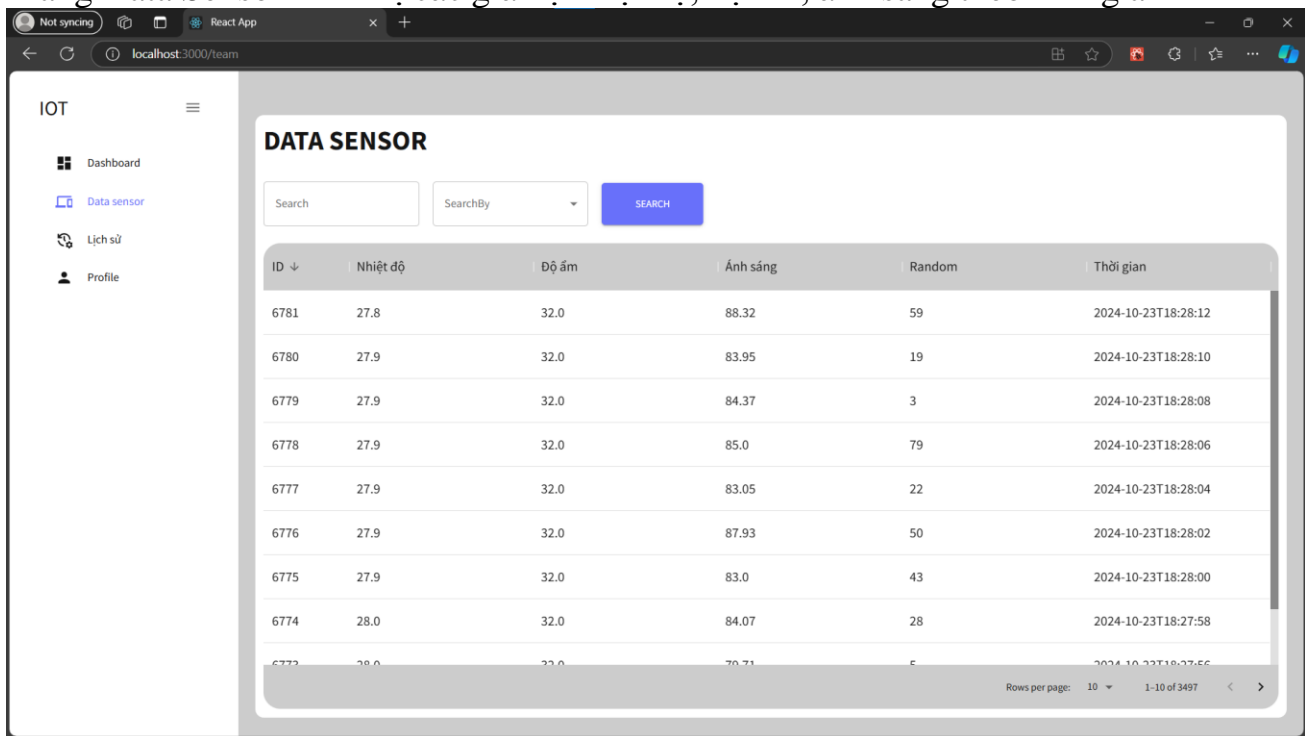
The screenshot shows the 'LỊCH SỬ TRẠNG THÁI THIẾT BỊ' (Device Status History) page. It features a search bar and a 'SEARCH' button. Below is a table with the following data:

ID	Name	Action	Time
406	light	off	2024-10-23T18:21:32
405	light	on	2024-10-23T18:21:30
404	humidity	off	2024-10-23T18:21:26
403	humidity	on	2024-10-23T18:21:24
402	humidity	off	2024-10-23T18:21:06
401	humidity	on	2024-10-23T18:21:04
400	random	off	2024-10-23T18:16:42
399	random	on	2024-10-23T18:16:40
398	humidity	off	2024-10-23T18:06:32

At the bottom right, there is a pagination control showing 'Rows per page: 10' and '1-10 of 406'.

Hình 10: Giao diện màn Action History

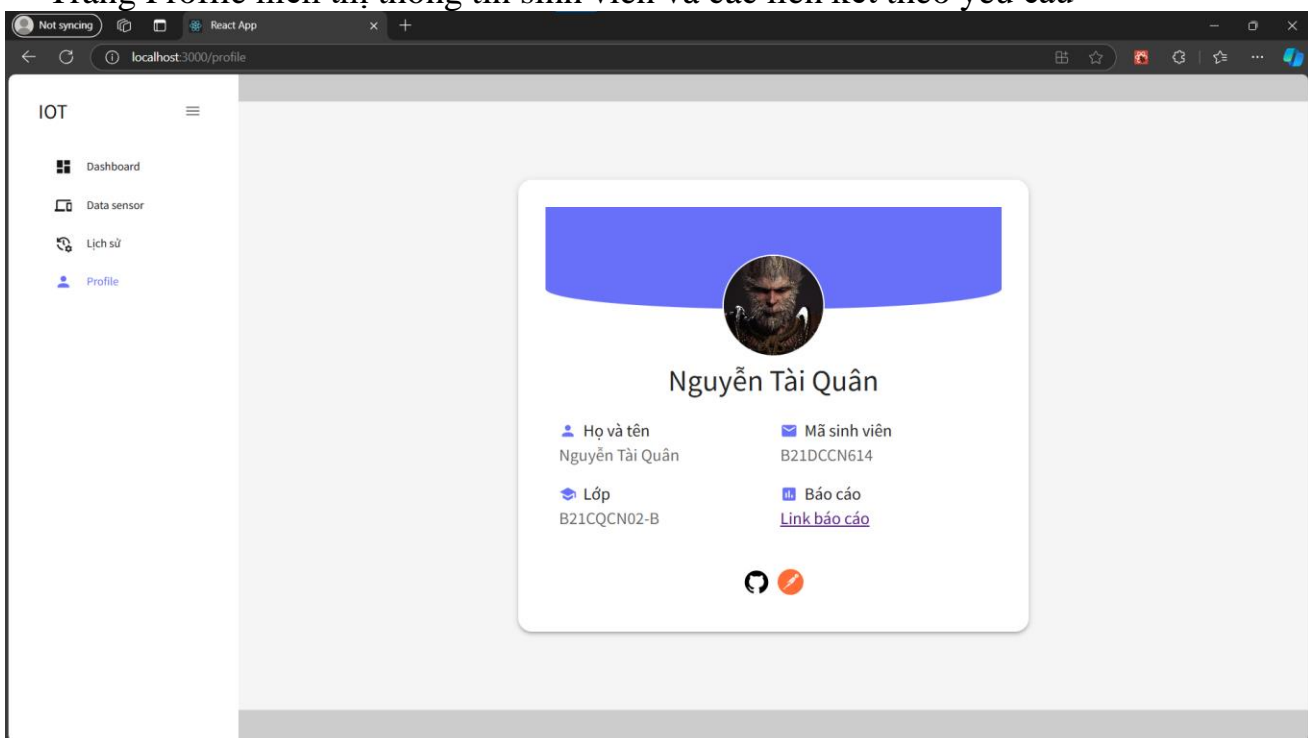
- Trang Data Sensor hiển thị các giá trị nhiệt độ, độ ẩm, ánh sáng theo thời gian



ID ↓	Nhiệt độ	Độ ẩm	Ánh sáng	Random	Thời gian
6781	27.8	32.0	88.32	59	2024-10-23T18:28:12
6780	27.9	32.0	83.95	19	2024-10-23T18:28:10
6779	27.9	32.0	84.37	3	2024-10-23T18:28:08
6778	27.9	32.0	85.0	79	2024-10-23T18:28:06
6777	27.9	32.0	83.05	22	2024-10-23T18:28:04
6776	27.9	32.0	87.93	50	2024-10-23T18:28:02
6775	27.9	32.0	83.0	43	2024-10-23T18:28:00
6774	28.0	32.0	84.07	28	2024-10-23T18:27:58
6773	28.0	32.0	70.71	5	2024-10-23T18:27:56

Hình 11: Giao diện màn Data Sensor

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



Nguyễn Tài Quân

Họ và tên: Nguyễn Tài Quân

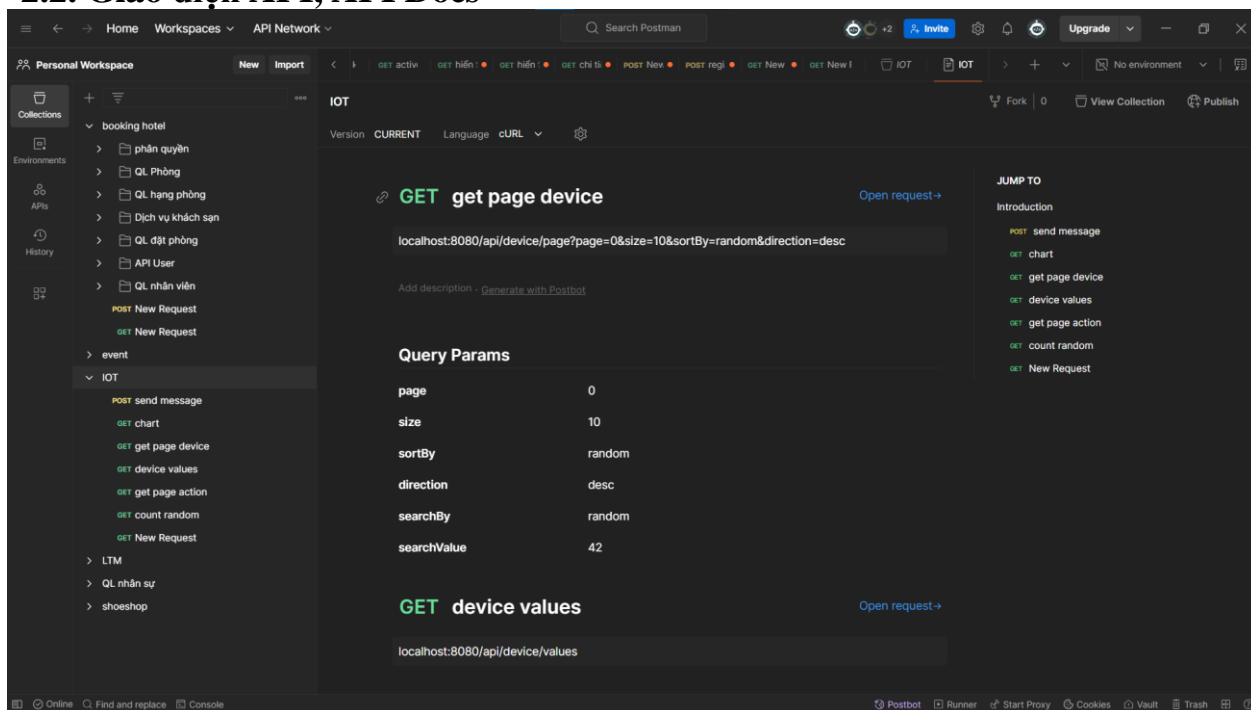
Mã sinh viên: B21DCCN614

Lớp: B21CQCN02-B

Báo cáo: [Link báo cáo](#)

Hình 12: Giao diện màn Profile

2.2. Giao diện API, API Docs

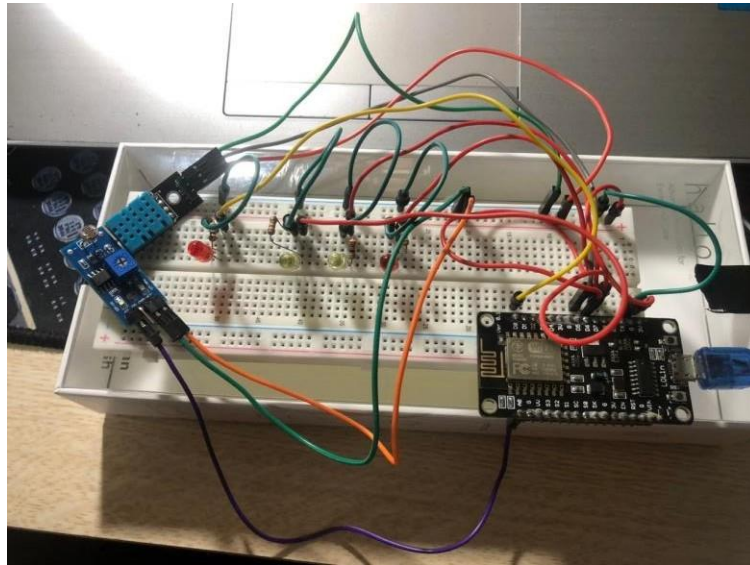


Hình 13: Giao diện API

```
"item": [  
  {  
    "name": "get page device",  
    "request": {  
      "method": "GET",  
      "header": [],  
      "url": {  
        "raw": "localhost:8080/api/device/page?page=0&size=10&sortBy=random&direction=desc",  
        "host": [  
          "localhost"  
        ],  
        "port": "8080",  
        "path": [  
          "api",  
          "device",  
          "page"  
        ],  
        "query": [  
          {  
            "key": "page",  
            "value": "0"  
          },  
          {  
            "key": "size",  
            "value": "10"  
          },  
          {  
            "key": "sortBy",  
            "value": "random"  
          },  
          {  
            "key": "direction",  
            "value": "desc"  
          },  
          {  
            "key": "searchBy",  
            "value": "random",  
            "disabled": true  
          }  
        ]  
      }  
    }  
  ]  
}
```

Hình 14: Giao diện API Docs

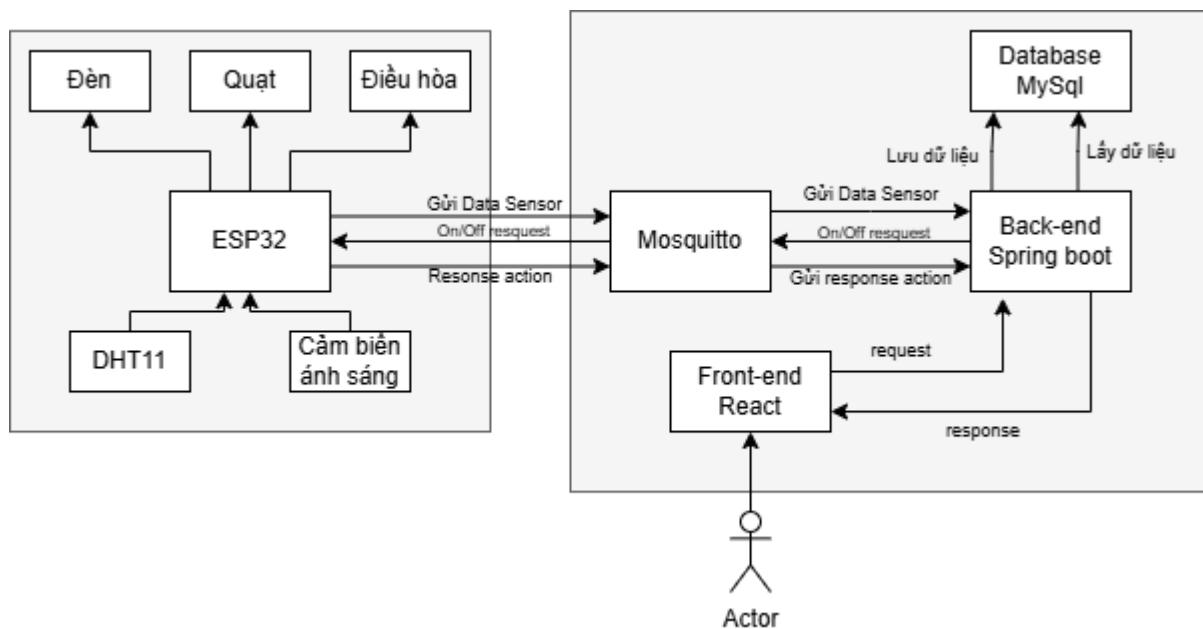
2.3. Giao diện thiết bị



Hình 15: Giao diện thiết bị

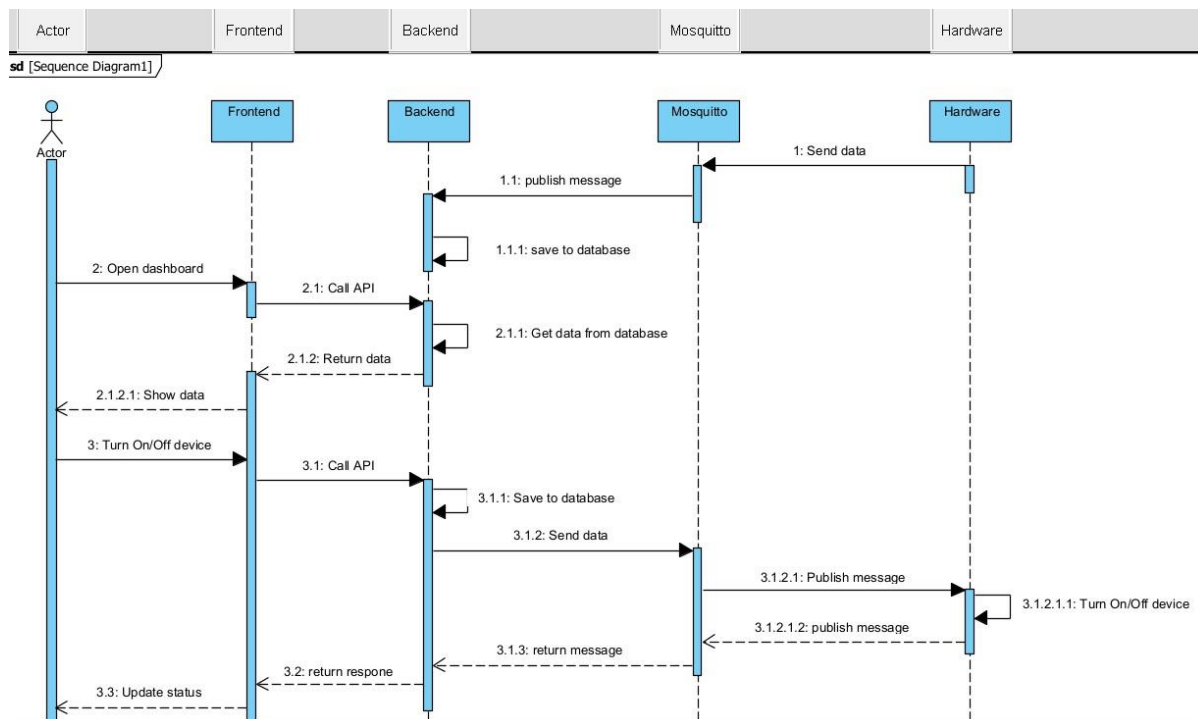
III. Thiết kế chi tiết

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 <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
```

- Khai báo các thư viện cần thiết gồm:

- DHT.h: Đọc dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT.
- WiFi.h: Thư viện kết nối với wifi
- PubSubClient.h: thư viện sử dụng mqtt

```
const char* ssid = "iPhone";
const char* password = "123456789";
```

- Định nghĩa các thông tin như tên wifi, mật khẩu wifi mà ESP32 sẽ kết nối

```
const char* mqtt_server = "172.20.10.2";
const char* mqtt_username = "taiquan"; // MQTT username
const char* mqtt_password = "b21dccn614"; // MQTT password
```

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

```
const int ledPin1 = 4;
const int ledPin2 = 21;
const int ledPin3 = 18;
```

- Định nghĩa các chân (pin) mà cảm biến ánh sáng và các thiết bị (quạt và đèn LED) được kết nối trên ESP32. Cụ thể:
 - Cảm biến ánh sáng được kết nối với chân analog A0 của ESP8266.
 - Chân GPIO số 4 được sử dụng để điều khiển quạt.
 - Chân GPIO số 21 được dùng để điều khiển LED 1.
 - Chân GPIO số 18 được dùng để điều khiển LED 2.

```
#define DHTPIN 14
#define AO_PIN 34
#define DHTTYPE DHT11
// LED Pin
```

- Định nghĩa chân kết nối với DHT11 là chân GPI14
- Setup kết nối wifi

```
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

- Nhận yêu cầu bật đèn và trả về thông báo đèn đã bật hay chưa

```
if (String(topic) == "esp32/output") {  
  if(messageTemp == "onLed1" && led1State == LOW){  
    Serial.println("onLed1");  
    led1State = HIGH;  
    digitalWrite(ledPin1, HIGH);  
    client.publish("response",String("onLed1").c_str());  
  }  
  else if(messageTemp == "offLed1" && led1State == HIGH){  
    led1State = LOW;  
    Serial.println("offLed1");  
    digitalWrite(ledPin1, LOW);  
    client.publish("response",String("offLed1").c_str());  
  }  
  else if(messageTemp == "onLed2" && led2State == LOW){  
    Serial.println("onLed2");  
    led2State = HIGH;  
    digitalWrite(ledPin2, HIGH);  
    client.publish("response",String("onLed2").c_str());  
  }  
  else if(messageTemp == "offLed2" && led2State == HIGH){  
    Serial.println("offLed2");  
    led2State = LOW;  
    digitalWrite(ledPin2, LOW);  
    client.publish("response",String("offLed2").c_str());  
  }  
  else if(messageTemp == "onLed3" && led3State == LOW){  
    Serial.println("onLed3");  
    led3State = HIGH;  
    digitalWrite(ledPin3, HIGH);  
    client.publish("response",String("onLed3").c_str());  
  }  
  else if(messageTemp == "offLed3" && led3State == HIGH){  
    Serial.println("offLed3");  
    led3State = LOW;  
    digitalWrite(ledPin3, LOW);  
    client.publish("response",String("offLed3").c_str());  
  }  
}
```

- Gửi data sensor

```
if (now - lastMsg > interval) {
  lastMsg = now;

  // Đọc cảm biến và gửi dữ liệu
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float lightValue = (float)analogRead(AO_PIN);
  int randomValue = random(0, 101);
  Serial.println(randomValue);

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.println(" %");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C");
  Serial.print("Light: ");
  Serial.println(lightValue);

  if (client.publish("data/sensor", ("{\"temperature\": " + String(t) + ", \"humidity\": " + String(h) + ", \"light\": " + String(lightValue) + ", \"random\": " + String(randomValue) + "}"))) {
    Serial.println("Data published successfully");
  }
}
```

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	Độ ẩm	Ánh sáng	Random	Thời gian
6781	27.8	32.0	88.32	59	2024-10-23T18:28:12
6780	27.9	32.0	83.95	19	2024-10-23T18:28:10
6779	27.9	32.0	84.37	3	2024-10-23T18:28:08
6778	27.9	32.0	85.0	79	2024-10-23T18:28:06
6777	27.9	32.0	83.05	22	2024-10-23T18:28:04

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

5.3. Khi người dùng điều khiển thiết bị

ID ↓	Name	Action	Time
406	light	off	2024-10-23T18:21:32
405	light	on	2024-10-23T18:21:30
404	humidity	off	2024-10-23T18:21:26
403	humidity	on	2024-10-23T18:21:24
402	humiditv	off	2024-10-23T18:21:06

- 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 Esp32, mosquito, DHT11, Arduino IDE:
- [esp32-nodemcu-mqtt-publish-dht11-arduino](#)
- [esp32-nodemcu-mqtt-publish-subscribe-dht22-readings/](#)