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

**KHOA CÔNG NGHỆ THÔNG TIN**

**----------------------------------------------**

**Icon

Description automatically generated**

**BÁO CÁO BÀI TẬP LỚN NHẬP MÔN IOT**

|  |  |  |
| --- | --- | --- |
| **Giảng viên** | **:** | **Nguyễn Quốc UY** |
| **Nhóm lớp** | **:** | **06** |
| **Họ và tên** | **:** | **Nguyễn Đình Hải** |
| **Mã sinh viên** | **:** | **B21DCAT004** |

**Hà Nội, 11/2024**

Contents

[I. Mô tả hệ thống 1](#_Toc181803756)

[Phần Cứng 1](#_Toc181803757)

[Phần Mềm 1](#_Toc181803758)

[II. MQTT 1](#_Toc181803759)

[II. Xây dựng flash và esptool. 2](#_Toc181803760)

[IV. Giao diện người dùng. 5](#_Toc181803761)

[V. Xây dựng mô hình hệ thống. 8](#_Toc181803762)

[1. Thiết bị -> MQTT Broker -> Flask MQTT Client: 8](#_Toc181803763)

[2. Flask MQTT Client -> Backend/WebSocket: 8](#_Toc181803764)

[3. Backend -> Flask -> MQTT Broker (nếu cần): 8](#_Toc181803765)

[MQTT Broker 9](#_Toc181803766)

[Flask Server (HTTP Server) 9](#_Toc181803767)

[WebSocket 10](#_Toc181803768)

[Cơ Sở Dữ Liệu SQLite 10](#_Toc181803769)

[**** Bảng dữ liệu: 10](#_Toc181803770)

[ Tính năng: 10](#_Toc181803771)

[VI. Mô tả sơ đồ luồng hoạt động. 11](#_Toc181803772)

[Luồng 1: Khởi tạo luồng Sensor 11](#_Toc181803773)

[Luồng 2: Truy cập vào trang DashBoard 11](#_Toc181803774)

# Mô tả hệ thống

**Mục Đích**

Hệ thống sử dụng các cảm biến để thông báo nhiệt độ, độ ẩm và ánh sáng cho người dùng thông qua một trang web có thể truy cập trong mạng LAN. Trang web cung cấp các thông tin về nhiệt độ, độ ẩm và ánh sáng với giao diện dashboard dễ nhìn, cho phép người dùng xem trực tiếp các thông tin này theo thời gian thực.

**Các Thiết Bị Sử Dụng Trong Hệ Thống**

## Phần Cứng

 1 kit ESP8266 NodeMCU 0.9

 1 cảm biến nhiệt độ và độ ẩm DHT22  1 cảm biến ánh sáng

 3 đèn LED

 1 bảng mạch

 Các loại dây dẫn đi kèm

## Phần Mềm

 PlatformIO trên VSCode chạy trên nền Arduino  Mosquitto server trên Windows

 Framework Flask + Flask-SocketIO (Backend)  Framework ReactJS (Client)

# MQTT

Cài đặt mqtt set username và password cho người dùng.

C:\Program Files\mosquitto>mosquitto\_passwd.exe -c "D:\PTIT\_Project\IOTapp\mosquitto\passwdfile.t Password:

Reenter password:

C C

Setting port và sử dụng passwordfile.txt trong file config mosquitto.

listener 1993 0.0.0.0

password\_file D:\PTIT\_Project\IOTapp\mosquitto\passwdfile.txt allow\_anonymous false

Kết quả sau khi sửa port và sử dụng với conf mới.

C:\Program Files\mosquitto>mosquitto.exe -c mosquitto.conf -v 1730648360: mosquitto version 2.0.20 starting

1730648360: Config loaded from mosquitto.conf. 1730648360: Opening ipv4 listen socket on port 1993. 1730648360: mosquitto version 2.0.20 running

# II. Xây dựng flash và esptool.

**Ardruino and device.**

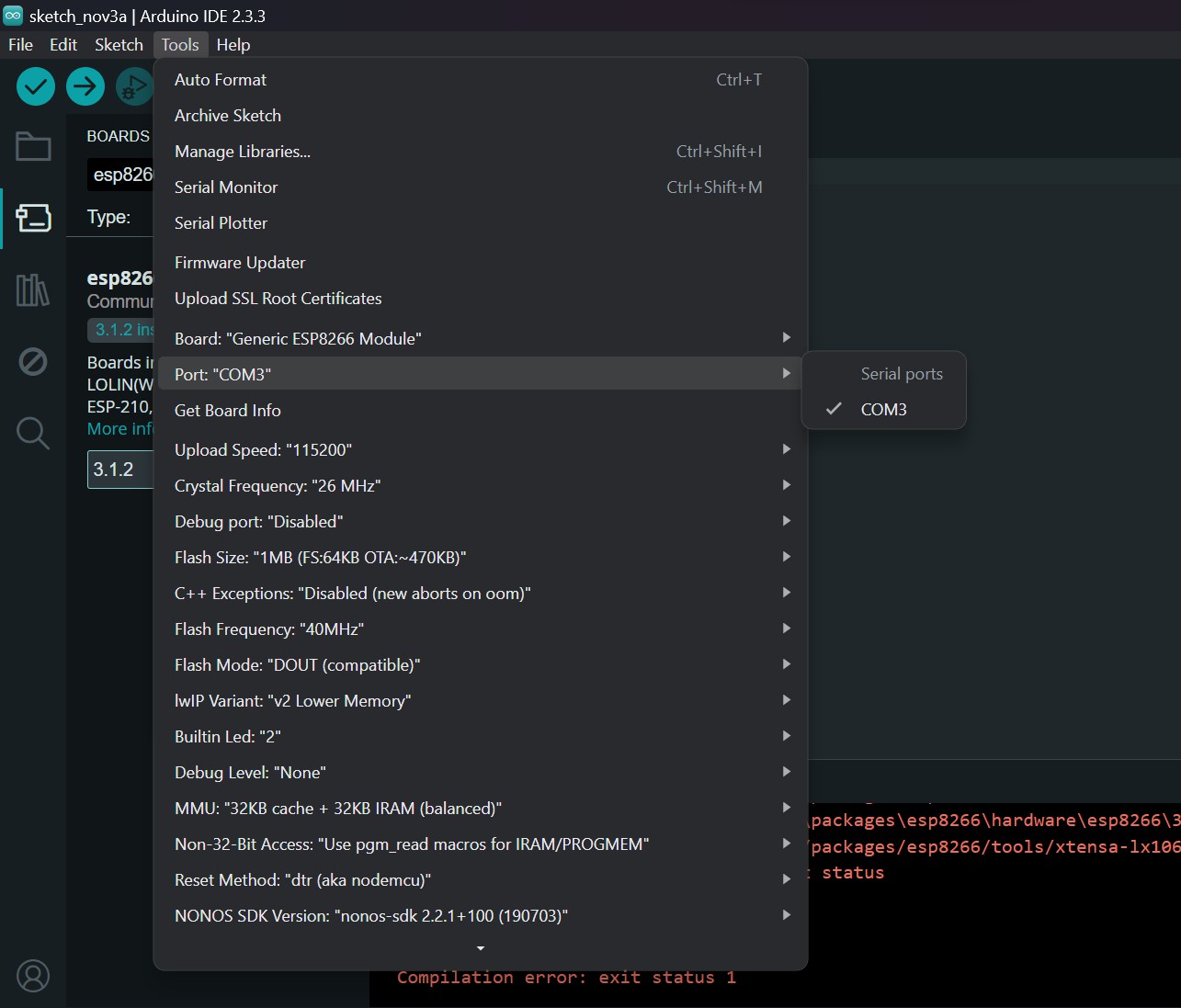
Cài đặt esptools trên https://micropython.org/download/ESP8266\_GENERIC/ pip install esptool

Sử dụng json này cho vào url để cài esp8266 trên arduno và cài đặt driver cho port arduino ide.

https://arduino.esp8266.com/stable/package\_esp8266com\_index.json

https://[www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers?tab=downloads](http://www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers?tab=downloads)

Cài arduino và xem cổng trên máy kết nối, check cổng kết nối trên device. Thường sẽ là COM3.



**Flash esp8266.**

esptool --port COM3 erase\_flash

esptool --port COM3 --baud 115200 write\_flash --flash\_size=detect 0x0 ESP8266\_GENERIC-20240602-v1

C C

Sau khi import file vào esp8266 tiếp tục cài mpremote

pip install mpremote

Tiếp tục thêm file vào esp8266

mpremote connect COM3 fs cp simple.py :

mpremote connect COM3 fs cp esp8266.py :

mpremote connect COM3 run esp8266.py

Kết Quả:

PS D:\PTIT\_Project\IOTapp\app\esp8266\dev\_mqtt> esptool --port COM3 erase\_flash esptool.py v4.8.1

Serial port COM3 Connecting....

Detecting chip type... Unsupported detection protocol, switching and trying again... Connecting....

Detecting chip type... ESP8266

Chip is ESP8266EX Features: WiFi Crystal is 26MHz

MAC: ec:64:c9:d3:dd:51

Uploading stub... Running stub...

Stub running...

Erasing flash (this may take a while)... Chip erase completed successfully in 7.5s Hard resetting via RTS pin...

PS D:\PTIT\_Project\IOTapp\app\esp8266\dev\_mqtt> esptool --port COM3 --baud 115200 write\_flash --f esptool.py v4.8.1

Serial port COM3

Connecting....

Detecting chip type... Unsupported detection protocol, switching and trying again... Connecting....

Detecting chip type... ESP8266

Chip is ESP8266EX Features: WiFi Crystal is 26MHz

MAC: ec:64:c9:d3:dd:51

Uploading stub... Running stub...

Stub running...

Configuring flash size... Auto-detected Flash size: 4MB

Flash will be erased from 0x00000000 to 0x0009afff...

Flash params set to 0x0040 Compressed 633048 bytes to 423611...

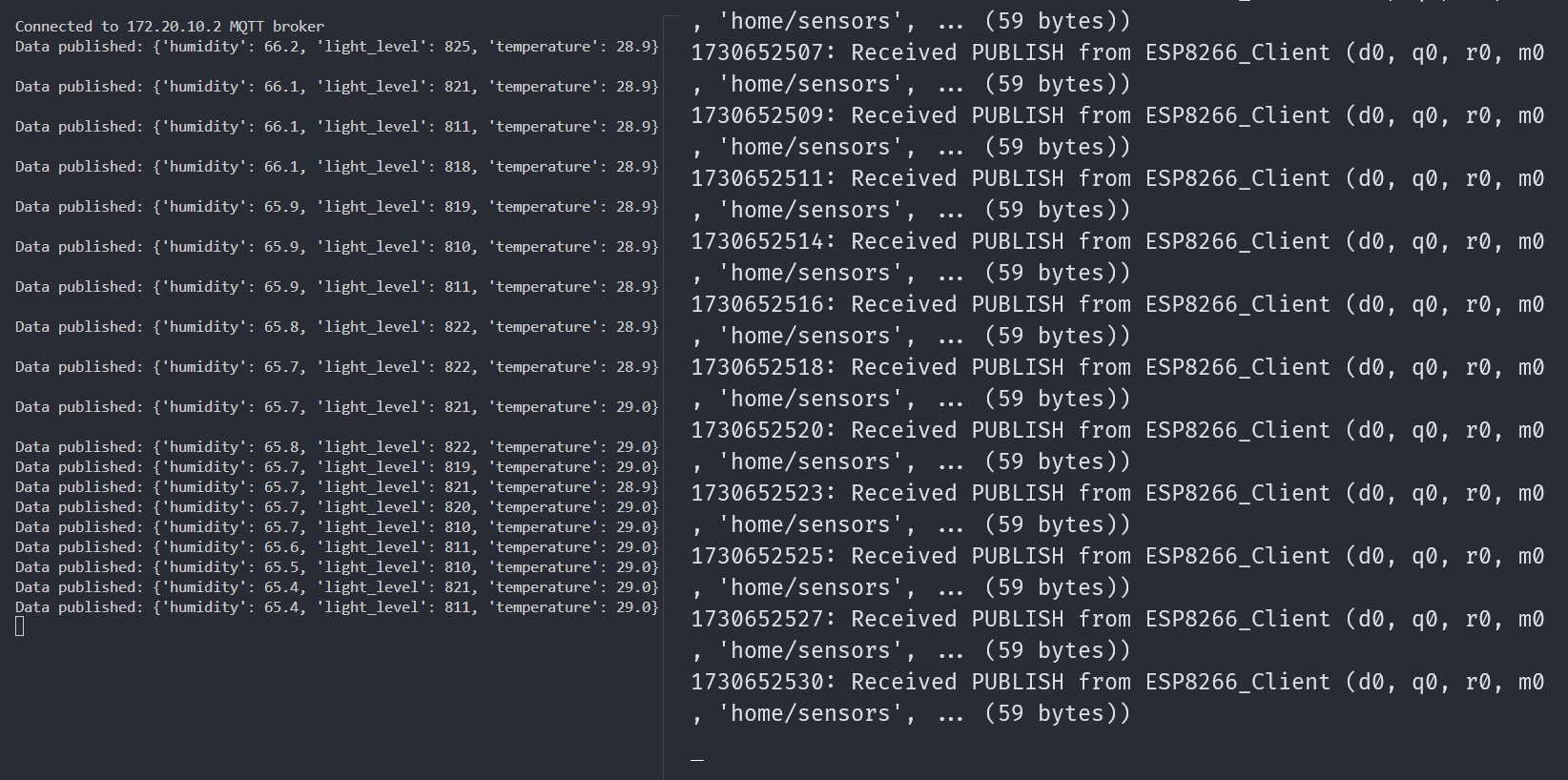
Wrote 633048 bytes (423611 compressed) at 0x00000000 in 37.8 seconds (effective 134.0 kbit/s)...

Hash of data verified.

Leaving...

Hard resetting via RTS pin...

 Sau khi nạp xong test thử và đã nhận được thông tin từ firmware.



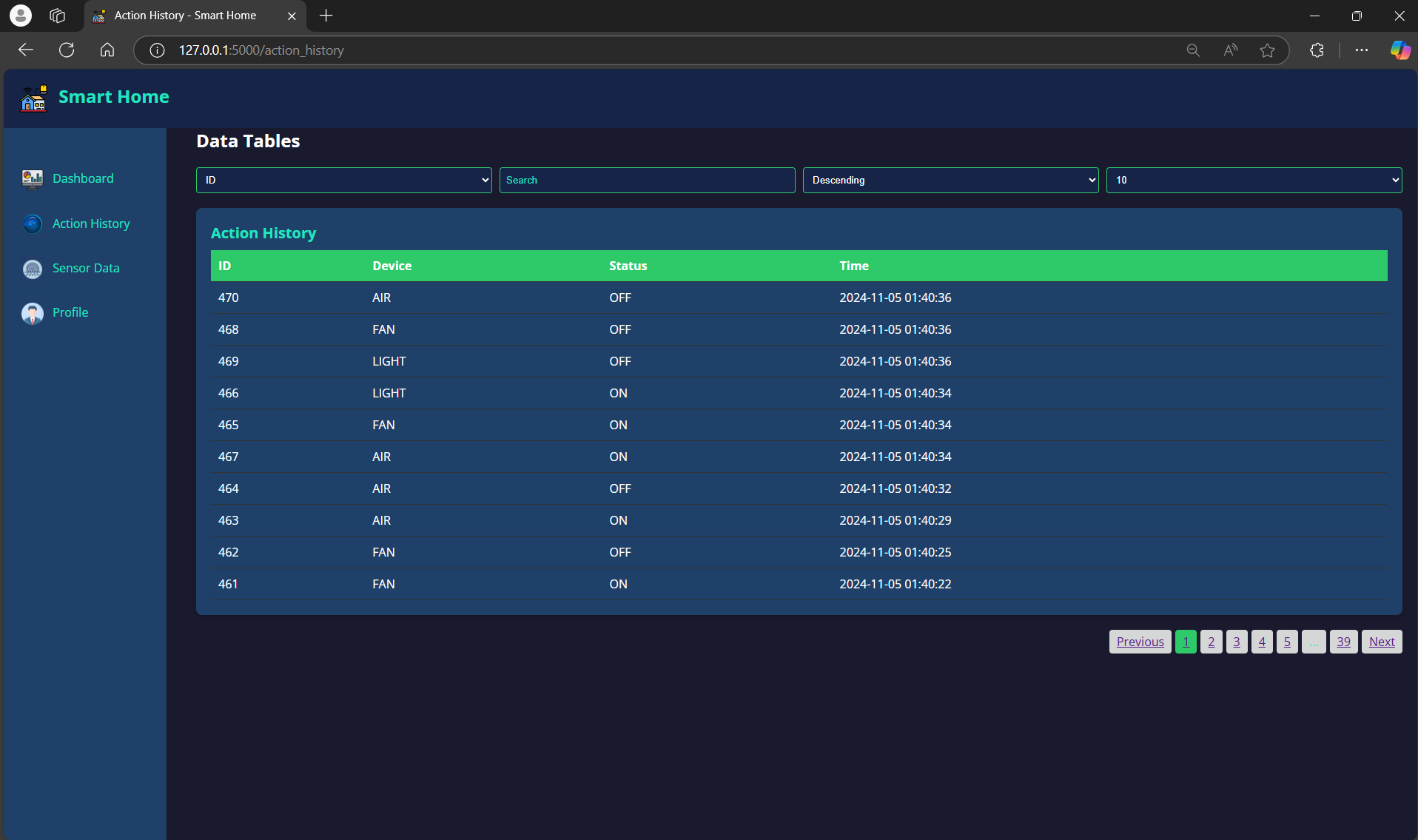
# Giao diện người dùng.

* 1. **Giao diện dashboard**

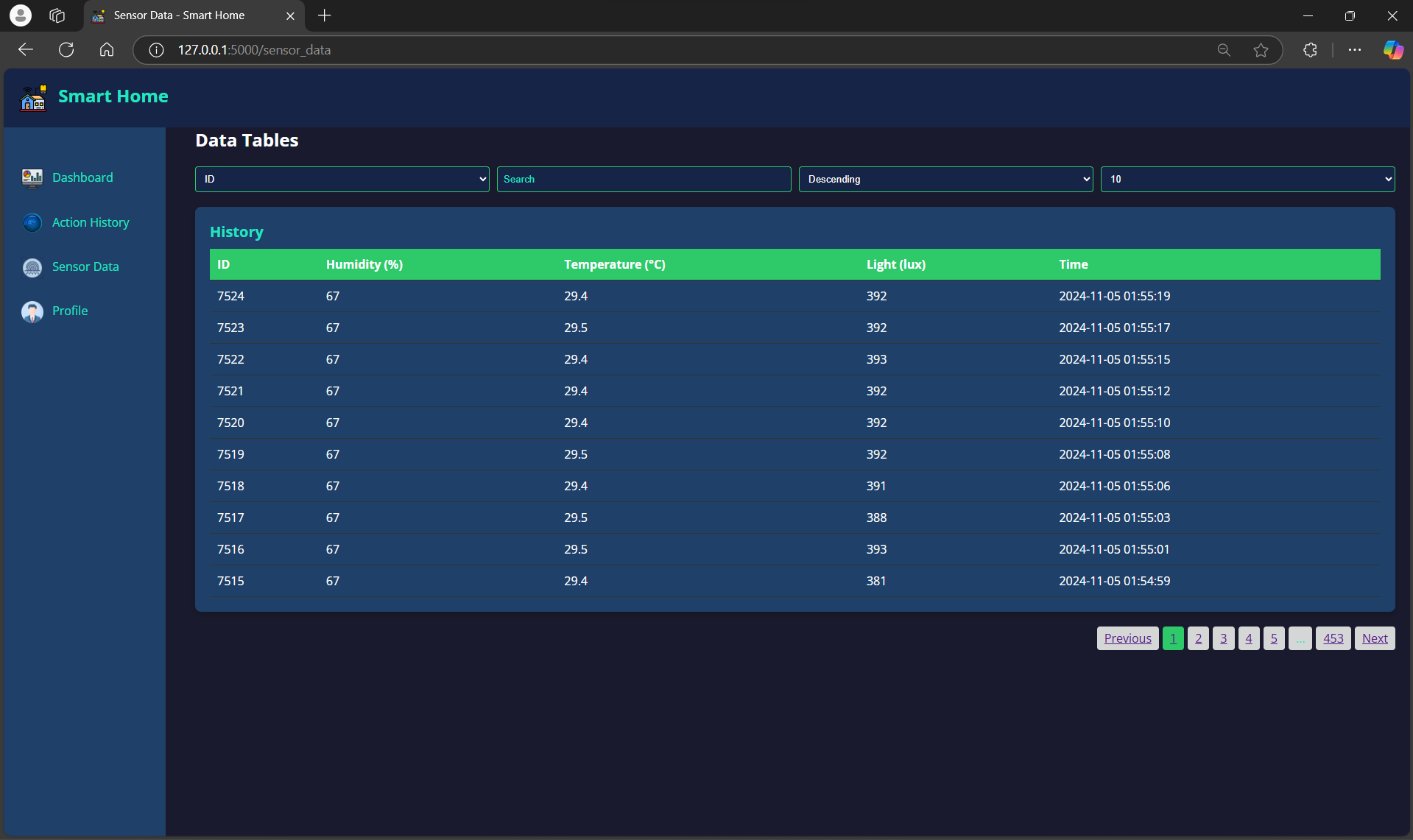
C C



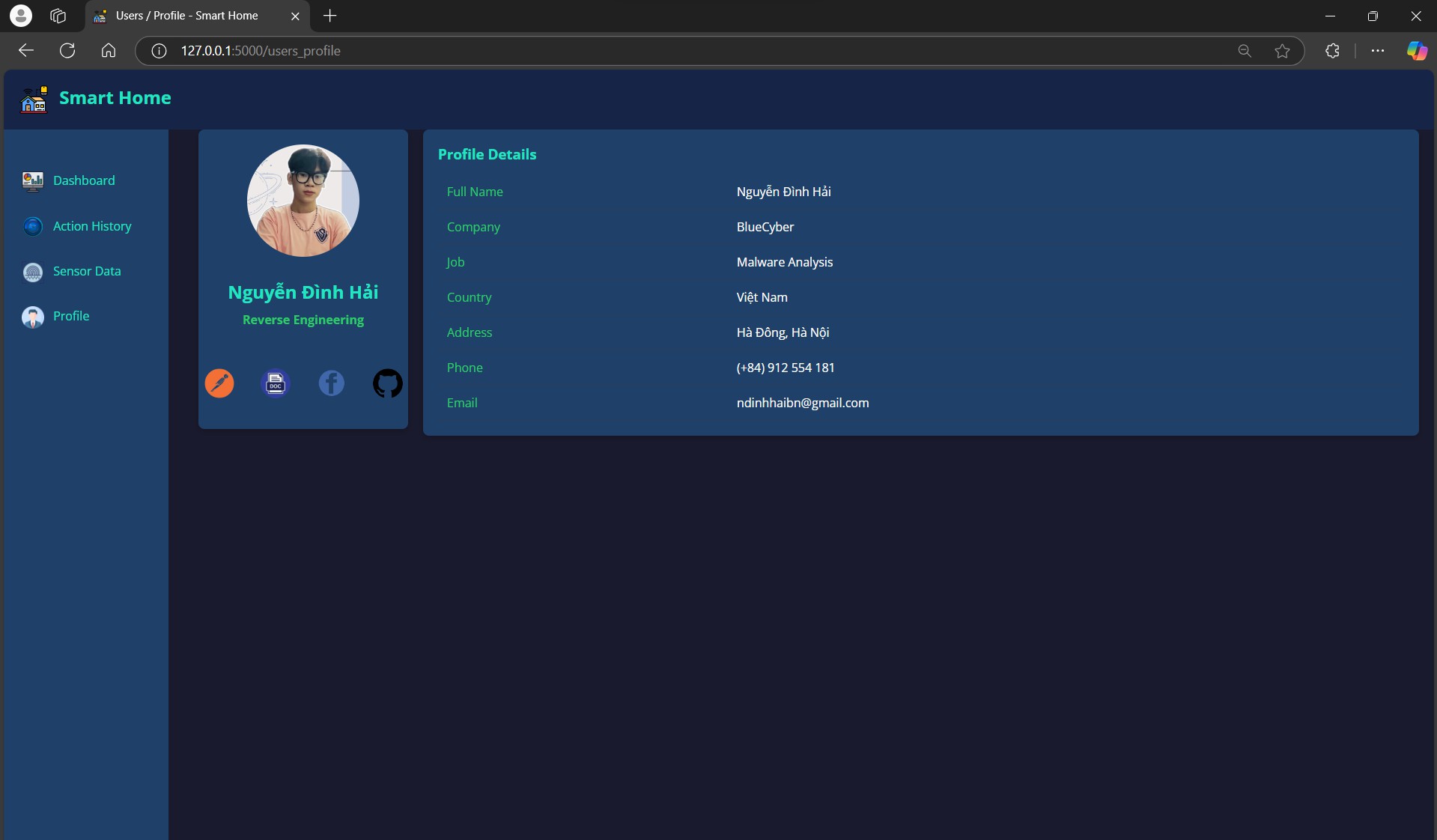
* 1. **Giao diện lịch sử điều khiển thiết bị- Action History**



* 1. **Giao diện ghi lại dữ liệu Sensor theo thời gian thực- Data Sensor**



* 1. **Giao diện profile**



# Xây dựng mô hình hệ thống.

**Sơ đồ Giao tiếp**

## Thiết bị -> MQTT Broker -> Flask MQTT Client:

 Thiết bị gửi dữ liệu lên MQTT broker.

 Flask MQTT Client nhận dữ liệu từ MQTT broker.

## Flask MQTT Client -> Backend/WebSocket:

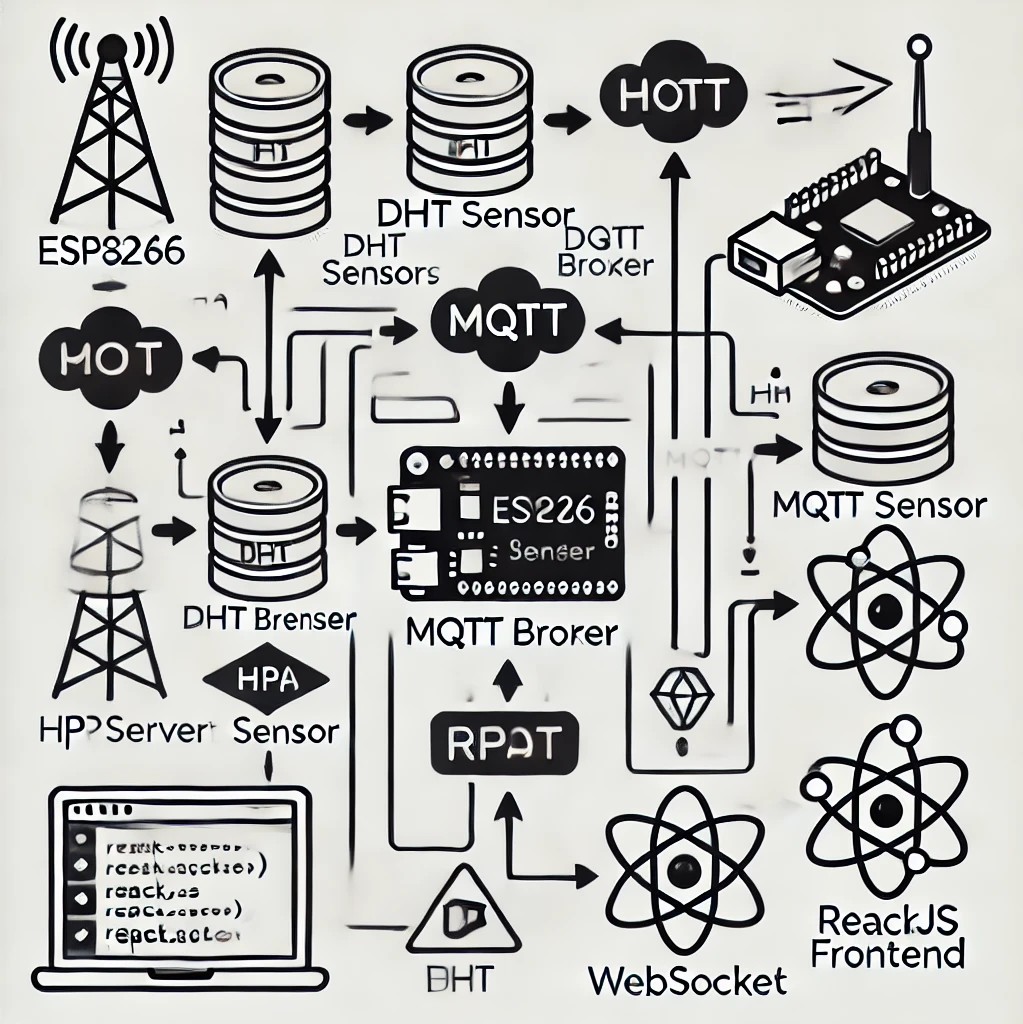
 Flask xử lý dữ liệu và gửi qua WebSocket hoặc HTTP API.

 Cập nhật thông tin tới các client hoặc lưu trữ vào backend.

## Backend -> Flask -> MQTT Broker (nếu cần):

 Flask có thể gửi tin nhắn đến các thiết bị qua MQTT broker.

 Publish tin nhắn đến các chủ đề tương ứng trên MQTT broker.



**Kiến Trúc Hệ Thống**

## MQTT Broker

 **Kết nối:** Kết nối tới mqtt://localhost:1993 .

 **Thông tin đăng nhập:** Yêu cầu đăng nhập với:

 **Username:** HaiND

 **Password:** B21DCAT004

## Flask Server (HTTP Server)

**Cổng:** Lắng nghe trên cổng 4444 .

**Routes:** Xử lý các yêu cầu từ client với các route sau:

 /api/v1/sensor/\*

 /api/v1/device/\*

## WebSocket

 **Máy chủ:** Được tạo trên cùng máy chủ với Flask.  **Thư viện:** Sử dụng eventlet để quản lý luồng.  **Chức năng:**

 Lắng nghe kết nối WebSocket từ client.

 Hiển thị thông tin cảm biến theo thời gian thực.

## Cơ Sở Dữ Liệu SQLite

 **Vị trí:** Tệp CSDL nằm ở thư mục gốc của server backend.

## Bảng dữ liệu:

 DataRealTime : Lưu trữ dữ liệu cảm biến thời gian thực.

 DeviceHistory : Lưu trữ lịch sử điều khiển thiết bị.

## Giao Diện Web Tính năng:

 Hiển thị đo lường theo thời gian thực của cảm biến DHT22.  Hiển thị thông tin lịch sử đo lường của cảm biến.

 Hiển thị thông tin lịch sử điều khiển thiết bị.  Hỗ trợ lọc và tìm kiếm dữ liệu để hiển thị.

**Thiết Kế Database**

Hệ thống bao gồm 2 bảng được sử dụng để lưu trữ dữ liệu của cảm biến và lịch sử điều khiển thiết bị, lần lượt là:

**Bảng DataRealTime**

CREATE TABLE `DataRealTime` (

`id` INT NOT NULL AUTO\_INCREMENT,

`temp` VARCHAR(255) NOT NULL,

`humidity` INT NOT NULL,

`light` INT NOT NULL,

`timestamp` DATETIME NULL DEFAULT NULL,

PRIMARY KEY (`id`)

);

**Bảng DeviceHistory**

CREATE TABLE `DeviceHistory` (

`id` INT NOT NULL AUTO\_INCREMENT,

`device\_name` VARCHAR(50) NOT NULL,

`command` VARCHAR(50) NOT NULL,

`status` VARCHAR(50) NOT NULL,

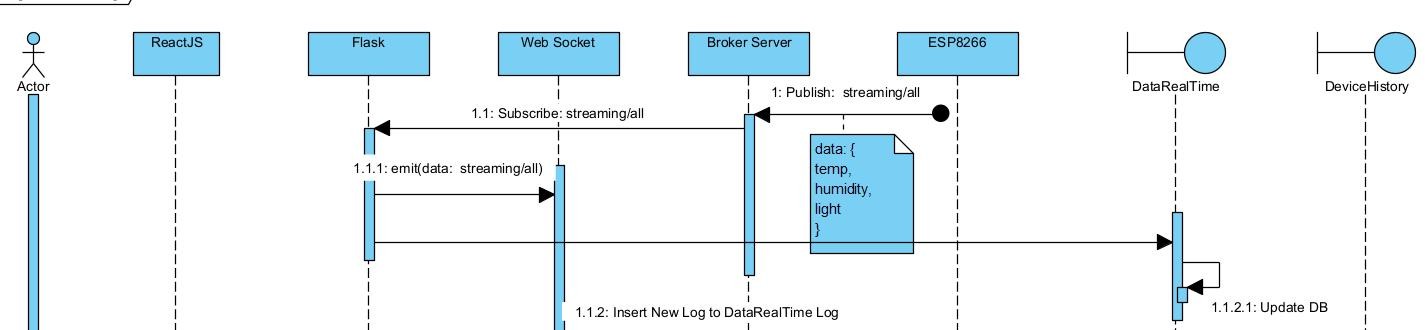
`timestamp` DATETIME NULL DEFAULT NULL, PRIMARY KEY (`id`)

);

# Mô tả sơ đồ luồng hoạt động.

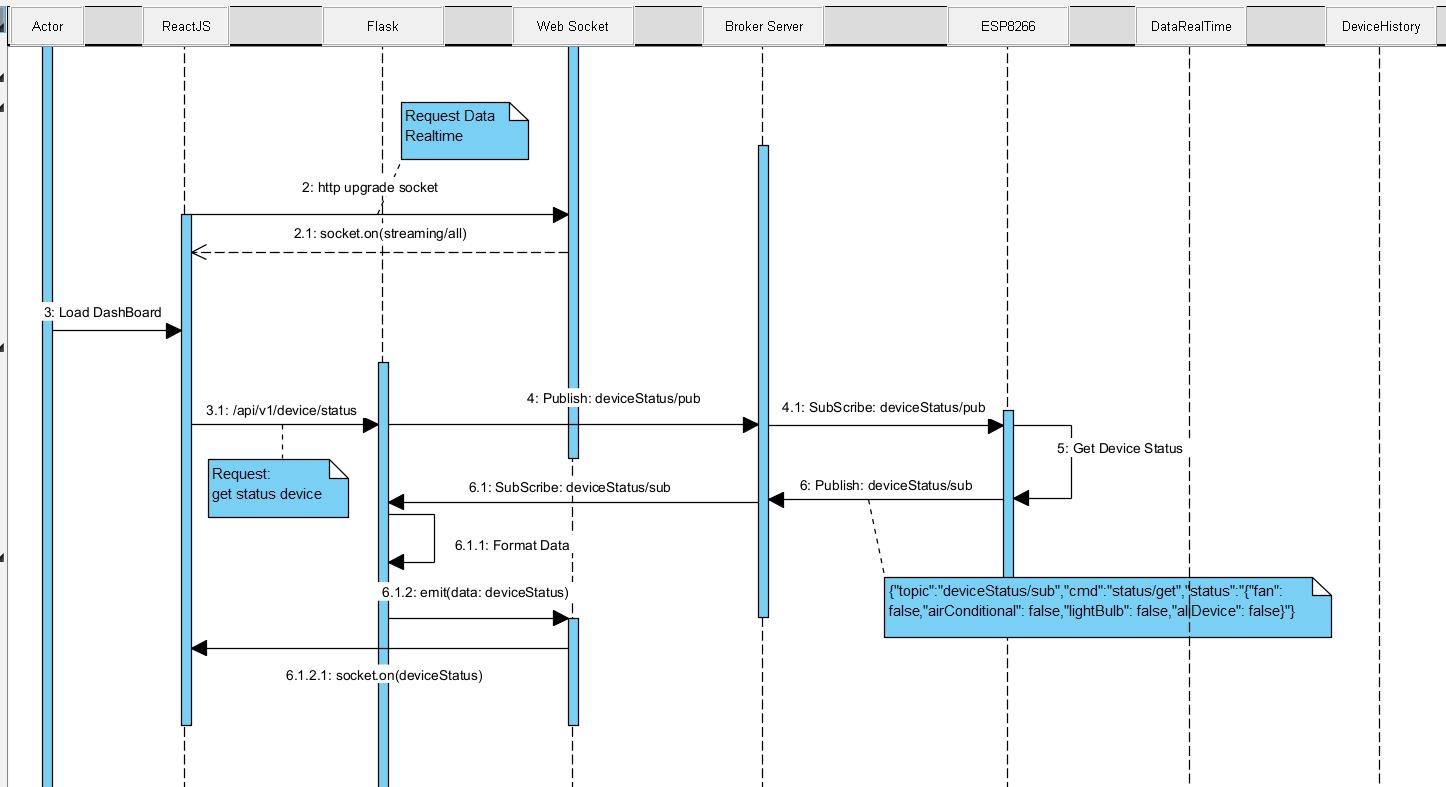
### Luồng 1: Khởi tạo luồng Sensor

Dữ liệu Sensor từ ESP về Flask, dữ liệu đến Flask được lưu vào DB với tên bảng là DATAREALTIME và đổ về Web Socket

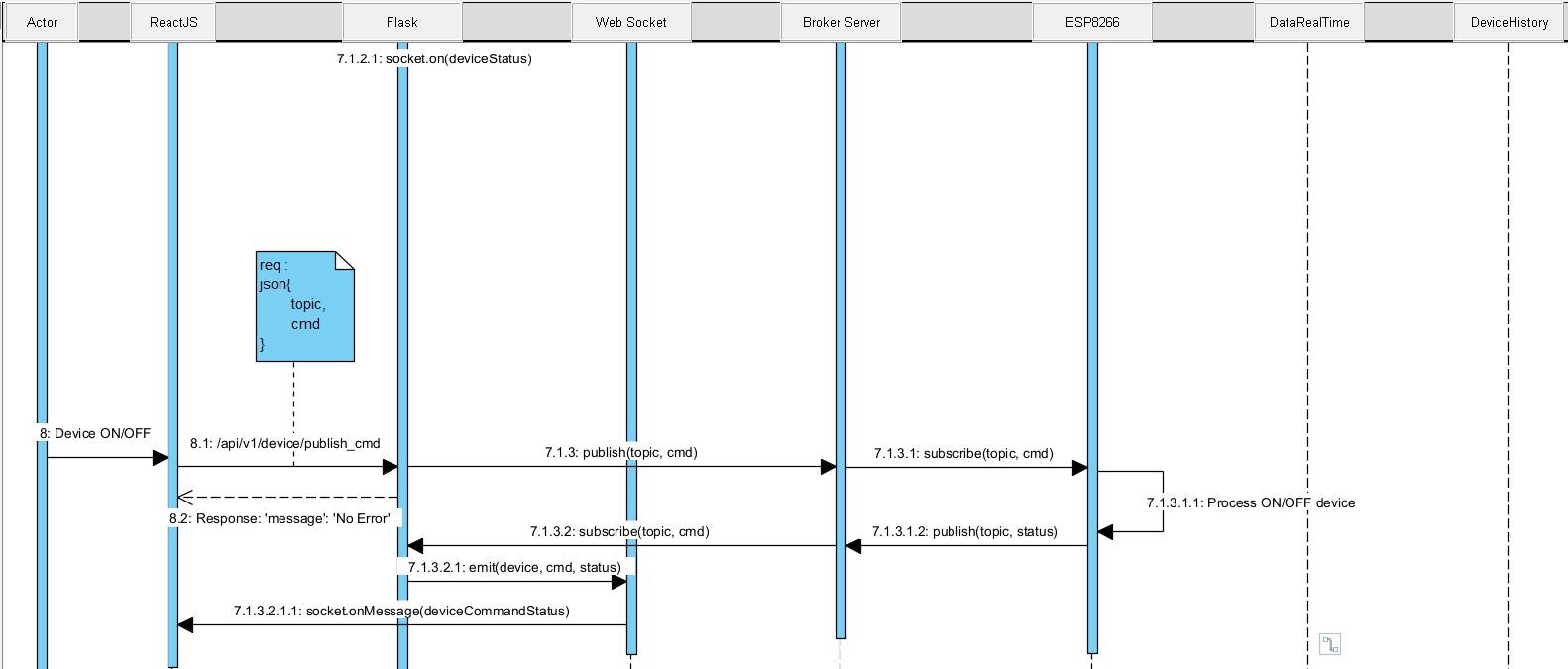


### Luồng 2: Truy cập vào trang DashBoard

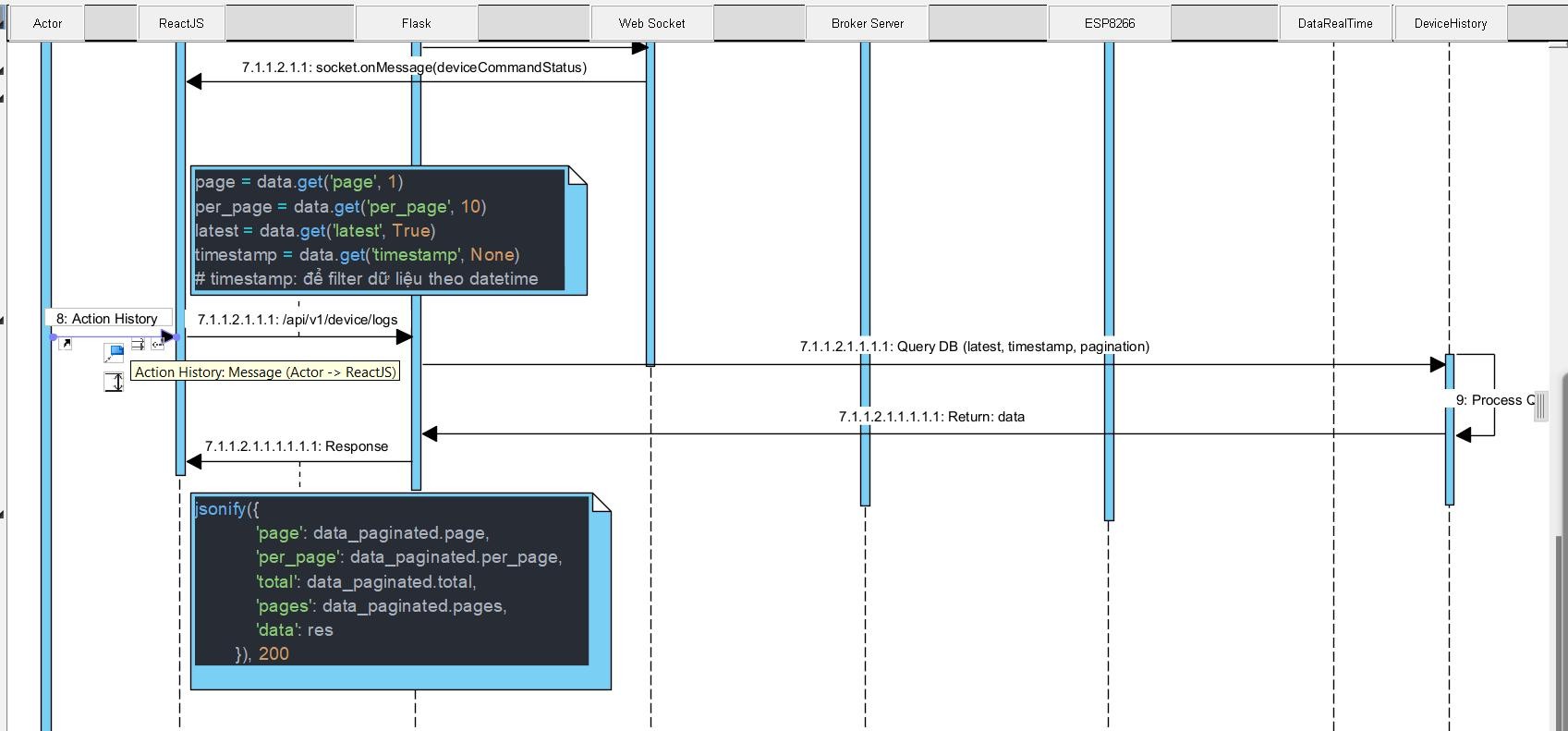
* Nhận dữ liệu Sensor qua WebSocket
* Nhận trạng thái ban đầu của device qua web socket



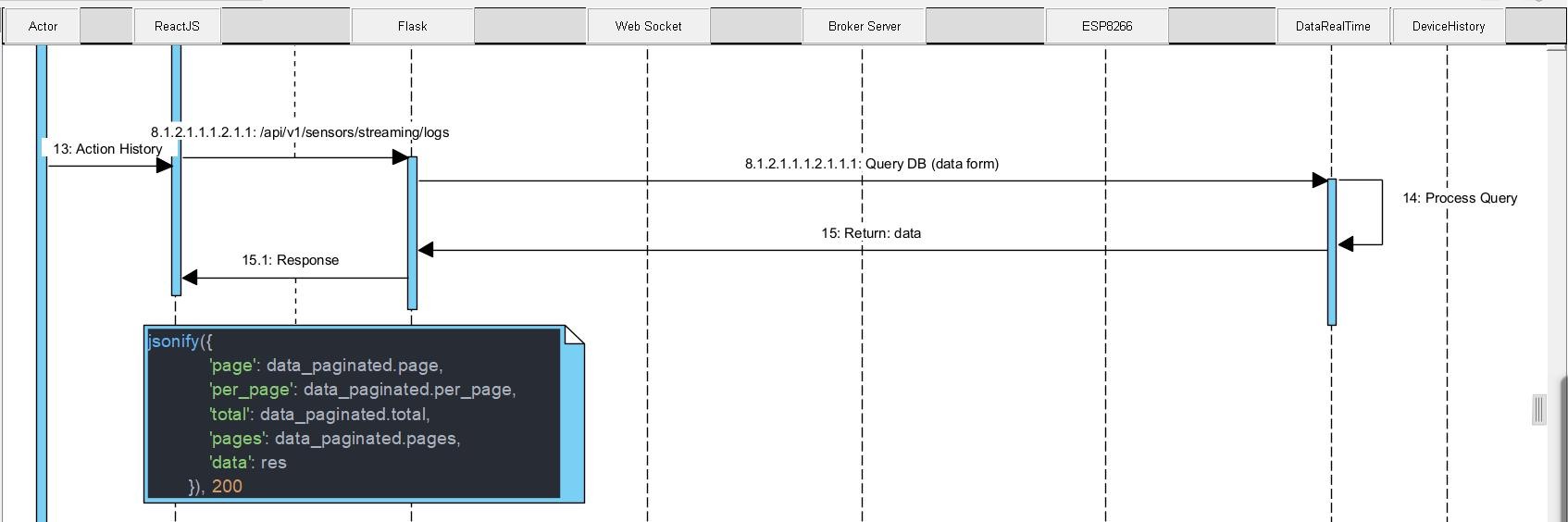
* + 1. Luồng 3:Điều khiển Device



* + 1. Luồng 4: Action History



* + 1. Luồng 5: Data Sensor



1. **API document.**

Nội dung chi tiết bên trong poshman. https://documenter.getpostman.com/view/39489332/2sAY4ye1DQ

API này được thiết kế nhằm cung cấp khả năng truy cập và tương tác với các dịch vụ thông qua các phương thức HTTP. Nó được xây dựng để hỗ trợ nhiều chức năng khác nhau, từ lấy dữ liệu cho đến gửi lệnh điều khiển các thiết bị từ xa. API hướng đến việc hỗ trợ các ứng dụng trong hệ thống IoT (Internet of Things), các ứng dụng quản lý thiết bị, và nhiều ứng dụng tương tác khác.

**Authentication**

Hiện tại, tất cả các yêu cầu gửi đến API không cần phải xác thực, nghĩa là người dùng có thể truy cập trực tiếp vào các endpoint mà không cần cung cấp bất kỳ thông tin đăng nhập hay mã thông báo (token) nào. Điều này giúp việc truy cập dữ liệu nhanh chóng và thuận tiện hơn, đặc biệt phù hợp cho các ứng dụng phát triển và thử nghiệm. Tuy nhiên, trong các phiên bản tương lai, API có thể triển khai cơ chế xác thực để đảm bảo an toàn và bảo mật cho dữ liệu nhạy cảm.

**Methods**

API của chúng tôi được thiết kế dựa trên các nguyên tắc RESTful, đảm bảo tính nhất quán và dễ sử dụng. Các phương thức HTTP chính được hỗ trợ bao gồm:

GET: Phương thức này được sử dụng để lấy thông tin từ hệ thống. Các yêu cầu GET có thể được sử dụng để truy xuất dữ liệu thiết bị, trạng thái của các thiết bị, lịch sử điều khiển, và nhiều thông tin khác. Với GET, người dùng chỉ có quyền truy vấn và không có khả năng chỉnh sửa dữ liệu.

POST: Phương thức POST được sử dụng để tạo mới dữ liệu trong hệ thống hoặc gửi các lệnh điều khiển thiết bị từ xa. Người dùng có thể gửi các yêu cầu POST với dữ liệu JSON trong phần body để điều khiển trạng thái của thiết bị, tạo mới bản ghi hoặc cấu hình thiết bị.

API hỗ trợ định dạng dữ liệu JSON cho cả yêu cầu (requests) và phản hồi (responses), giúp các ứng dụng có thể dễ dàng tích hợp và xử lý dữ liệu. Tất cả các yêu cầu đều tuân theo cấu trúc JSON, đảm bảo tính linh hoạt và khả năng mở rộng cho các ứng dụng khác nhau.

Các Tính Năng Chính

API bao gồm nhiều tính năng và endpoint khác nhau, được thiết kế để đáp ứng các yêu cầu đa dạng từ người dùng:

Lấy thông tin thiết bị: Người dùng có thể lấy danh sách các thiết bị có sẵn trong hệ thống, bao gồm thông tin cơ bản như tên, loại thiết bị, và trạng thái hiện tại.

Truy xuất lịch sử điều khiển: API cung cấp khả năng truy xuất lịch sử các lệnh đã được gửi đến các thiết bị, giúp người dùng theo dõi hoạt động và trạng thái thay đổi của hệ thống theo thời gian.

Điều khiển thiết bị từ xa: Người dùng có thể bật/tắt hoặc điều chỉnh trạng thái của các thiết bị từ xa thông qua các lệnh điều khiển, hỗ trợ kiểm soát linh hoạt và thuận tiện.

Phân trang và tìm kiếm: Các yêu cầu truy vấn dữ liệu có hỗ trợ phân trang và tìm kiếm, giúp người dùng dễ dàng tìm kiếm thông tin cụ thể hoặc quản lý lượng lớn dữ liệu mà không ảnh hưởng đến hiệu suất hệ thống.

**POST** <http://localhost:5000/api/v1/device/publish>

StartFragmentAPI này được sử dụng để điều khiển thiết bị từ xa bằng cách gửi lệnh bật/tắt thông qua topic và cmd. Người dùng có thể sử dụng API này để thay đổi trạng thái của thiết bị (như đèn, quạt, máy lạnh) trong hệ thống nhà thông minh.EndFragment

**Request**

Body: Yêu cầu phải gửi dữ liệu ở định dạng JSON, bao gồm:

cmd (string): Lệnh để điều khiển thiết bị. Giá trị có thể là "ON" để bật thiết bị hoặc "OFF" để tắt thiết bị.

topic (string): Chủ đề của thiết bị cần điều khiển. Ví dụ: "home/light" để điều khiển đèn trong nhà hoặc "home/all" để điều khiển tất cả thiết bị.

Body

raw (json)

json

{

"cmd": "OFF",

"topic": "home/light"

}

Example Request

http://localhost:5000/api/v1/device/publish

curl

curl --location 'http://localhost:5000/api/v1/device/publish' \

--data '

{

"cmd": "OFF",

"topic": "home/light"

}'

200 OK

Example Response

Body

Headers (5)

json

{

"device": {

"cmd": "OFF",

"output": "",

"status": "self.cmd",

"topic": "home/light"

},

"message": "Device turned OFF"

}

* 1. **Sensor**

API này được sử dụng để lấy thông tin dữ liệu từ các cảm biến trên thiết bị. Dữ liệu cảm biến bao gồm các giá trị về nhiệt độ, độ ẩm, ánh sáng, và thời gian thu thập. API hỗ trợ phân trang, tìm kiếm và sắp xếp kết quả để dễ dàng quản lý và truy xuất dữ liệu hiệu quả.

**GET** <http://localhost:5000/api/v1/sensor/data>

**Request**

page (int): Số trang hiện tại mà người dùng yêu cầu, dùng để phân trang kết quả. Giá trị mặc định là 1, đại diện cho trang đầu tiên.

limit (int): Số lượng bản ghi được trả về trên mỗi trang. Mặc định là 10, giúp giới hạn số lượng dữ liệu trả về để tối ưu hóa tốc độ phản hồi.

search (string): Từ khóa tìm kiếm dữ liệu theo một trường cụ thể (như temp, humidity, light, hoặc timestamp). Cho phép lọc dữ liệu theo một giá trị nhất định.

sort (string): Thứ tự sắp xếp kết quả trả về. Có thể sắp xếp tăng dần (ASC) hoặc giảm dần (DESC) dựa trên trường timestamp, giúp hiển thị dữ liệu mới nhất hoặc cũ nhất trước.

**Response**

Dữ liệu trả về bao gồm một danh sách các bản ghi cảm biến theo yêu cầu của người dùng. Mỗi bản ghi chứa các thông tin sau:

id (int): Số thứ tự của bản ghi trong cơ sở dữ liệu, giúp định danh duy nhất mỗi bản ghi.

humidity (float): Giá trị độ ẩm được ghi nhận bởi cảm biến, đại diện cho độ ẩm trong môi trường tại thời điểm thu thập.

light (float): Giá trị ánh sáng đo được từ cảm biến, biểu thị mức độ ánh sáng của môi trường.

temp (float): Giá trị nhiệt độ đo được từ cảm biến, cung cấp thông tin về nhiệt độ của môi trường.

timestamp (string): Thời điểm dữ liệu được ghi nhận từ cảm biến. Định dạng YYYY-MM-DD HH:MM:SS, giúp người dùng xác định chính xác thời gian thu thập dữ liệu.