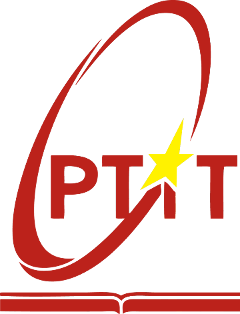
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**



**BÁO CÁO BTL CUỐI KỲ**

**Môn học:** IoT và ứng dụng

**Giảng viên:** Nguyễn Quốc Uy

**Thành viên:** Hoàng Mạnh Dũng

**Mã sinh viên:** B21DCCN268

**Số điện thoại:** 0377659771

***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 dẫn dắt em trong suốt thời gian qua. Do chưa có nhiều kinh nghiệm làm đề tài nhiều cũng như những hạn chế về kiến thức, trong BTL của em chắc chắn sẽ không tránh khỏi những thiếu sót. Rất mong nhận được sự nhận xét, ý kiến đóng góp và phê bình từ thầy để dự án của em được hoàn thiện hơn.

Lời cuối cùng, em xin kính chúc thầy nhiều sức khỏe, thành công và hạnh phúc.

Em xin chân thành cảm ơn!

Mục lục

[**LỜI CẢM ƠN** 2](#_Toc175723313)

[**DANH MỤC HÌNH ẢNH** 4](#_Toc175723314)

[**I. Giới thiệu** 5](#_Toc175723315)

[1. Mô tả đề tài và mục đích của đề tài 5](#_Toc175723316)

[2. Thiết bị sử dụng 5](#_Toc175723317)

[a. Node MCU ESP 8266 5](#_Toc175723318)

[b. Cảm biến nhiệt độ - độ ẩm 7](#_Toc175723319)

[c. Cảm biến cường độ ánh sáng quang trở 8](#_Toc175723320)

[d. Board Test MB-102 16.5x5.5 9](#_Toc175723321)

[e. Điện trở vạch 1/4W sai số 5% 250V 1R-10M 10](#_Toc175723322)

[g. Led tự đổi màu nhấp nháy 5mm, 7 màu 2 chân (1,5-3V) 11](#_Toc175723323)

[3. Công nghệ sử dụng 11](#_Toc175723324)

[a. Trình biên dịch Arduino IDE 11](#_Toc175723325)

[b. Android Studio 11](#_Toc175723326)

[c. Flutter 12](#_Toc175723327)

[d. Spring Boot 12](#_Toc175723328)

[e. Mosquitto 13](#_Toc175723329)

[**II. Giao diện, thiết kế thổng thể** 15](#_Toc175723330)

[1. Giao diện ứng dụng 15](#_Toc175723331)

[2. Giao diện API Docs 18](#_Toc175723332)

[3. Giao diện thiết bị 18](#_Toc175723333)

[**III. Thiết kế chi tiết** 19](#_Toc175723334)

[1. Sequence Diagram 19](#_Toc175723335)

[**IV. Code** 20](#_Toc175723336)

# **DANH MỤC HÌNH ẢNH**

*Hình 1: Node MCU ESP8266*

*Hình 2: Cảm biến nhiệt độ, độ ẩm DHT11*

*Hình 3: Sơ đồ chân nối DHT11*

*Hình 4: Sơ đồ giao tiếp vi điều khiển DHT11*

*Hình 5: Cảm biến ánh sáng*

*Hình 6: Board test*

*Hình 7: Điện trở*

*Hình 8: Dây nối*

*Hình 9: Led đổi màu*

*Hình 10, 11: Giao diện Dashboard*

*Hình 12, 13: Giao diện màn Action*

*Hình 13: Giao diện màn Data Sensor*

*Hình 14: Giao diện màn Profile*

*Hình 15: Giao diện API Docs*

*Hình 16: Giao diện thiết bị*

*Hình 17: Sequence Diagram của hệ thống*

*Hình 18 đến 31: Code embedded*

# **I. Giới thiệu**

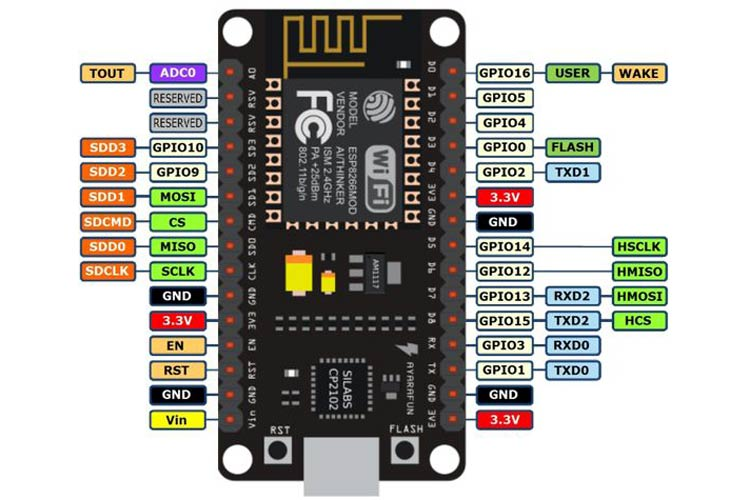
## 1. Mô tả đề tài và mục đích của đề tài

Đề tài này tập trung vào việc phát triển một hệ thống giám sát và điều khiển nhiệt độ, độ ẩm, cũng như các thiết bị như điều hòa, điện, quạt thông qua ứng dụng di động dành cho Android và iOS. Hệ thống sử dụng Spring Boot làm nền tảng cho phần backend, cung cấp các API để ứng dụng có thể lấy thông tin về nhiệt độ, độ ẩm 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 thông qua ứng dụng 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 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 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.

### b. Cảm biến nhiệt độ - độ ẩm

- Thông số kỹ thuật

  - Điện áp hoạt động: 3V - 5V DC

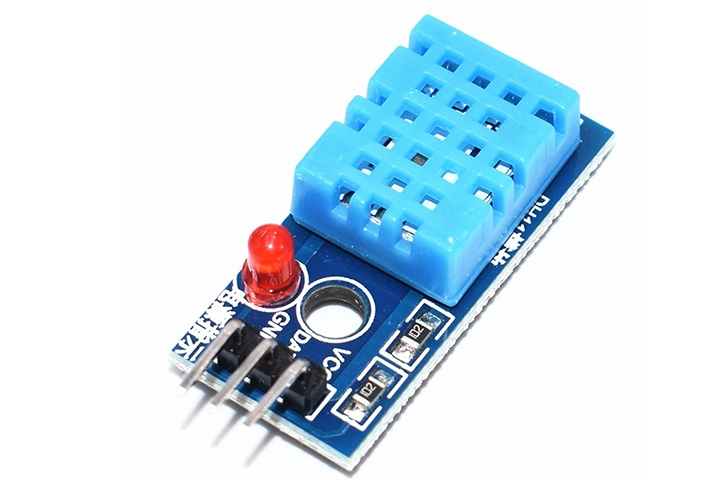
 - Dòng điện tiêu thụ: 2.5mA

 - Phạm vi cảm biến độ ẩm: 20% - 90% RH, sai số ±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)

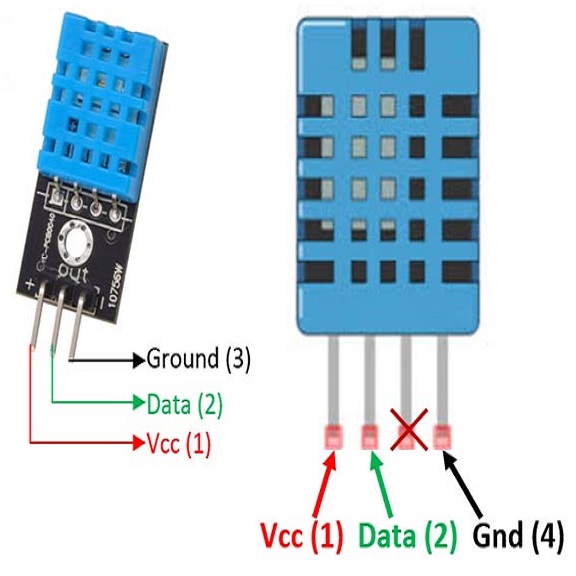
 - Kích thước: 23 \* 12 \* 5 mm



*Hình 2: Cảm biến DHT11*

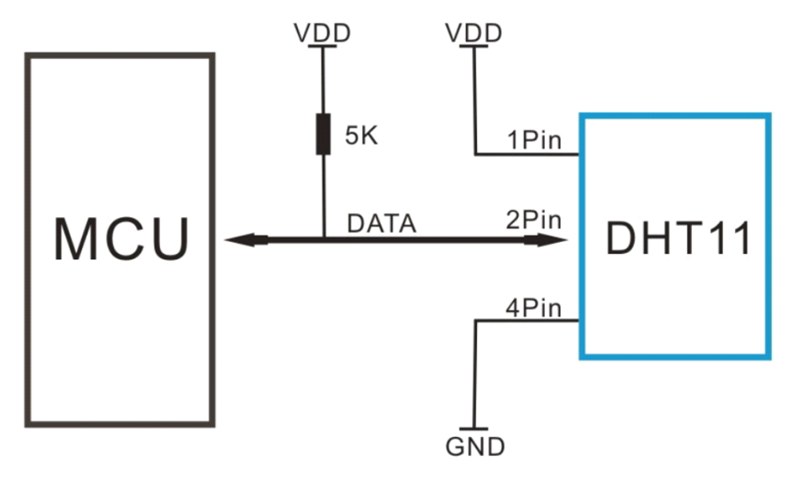
- Sơ đồ chân DHT11:

|  |  |  |
| --- | --- | --- |
| 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*

- Giao tiếp với vi điều khiển



*Hình 4: Sơ đồ giao tiếp vi điều khiển DHT11*

### c. Cảm biến cường độ ánh sáng quang trở

- Thông số kỹ thuật

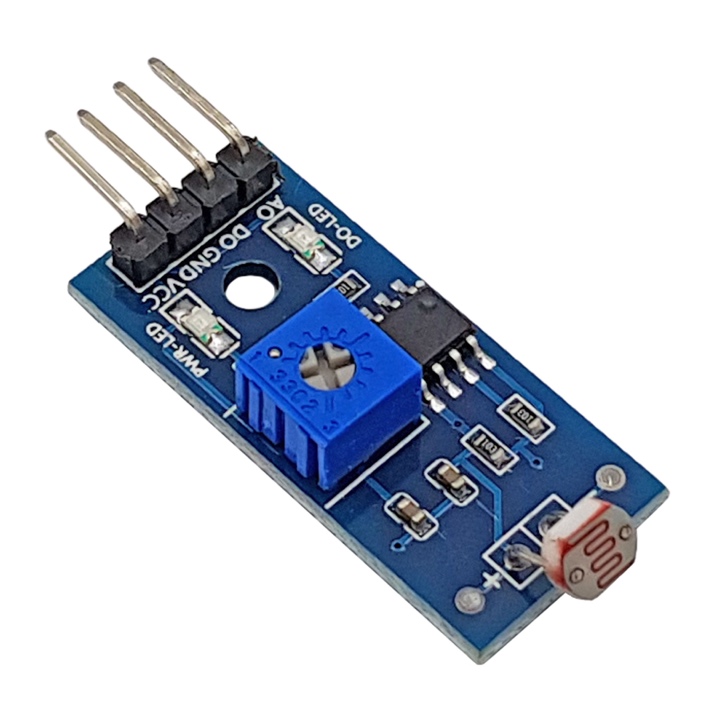
+ Điện áp hoạt động: 3.3V – 5V

+ 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 .

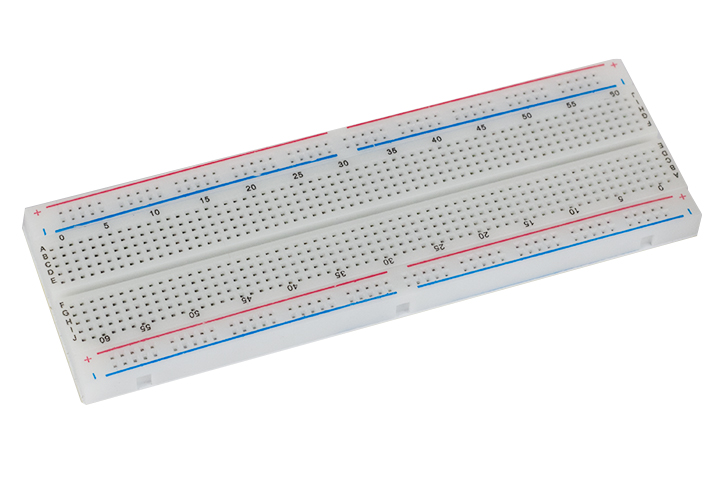
+ Kích thước: 32 x 14mm



*Hình 5: 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



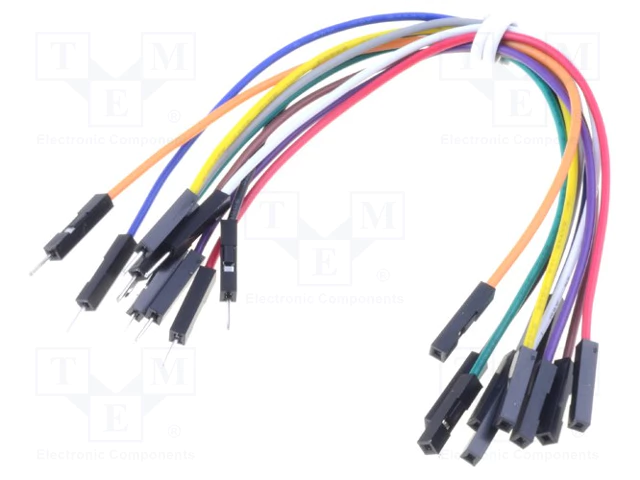
*Hình 6: Board test*

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



*Hình 7: Điện trở*

f. Dây nối chân các thiết bị

****

*Hình 8: Dây nối*

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



*Hình 9: Led đổi màu*

## 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 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. Android Studio

- Android Studio là một môi trường phát triển tích hợp (IDE) chính thức được sử dụng để phát triển ứng dụng Android. Nó được phát triển bởi Google và dựa trên IDE Java IntelliJ của hãng JetBrains. Android Studio cung cấp nhiều tính năng và công cụ giúp các nhà phát triển tạo ra các ứng dụng Android chất lượng cao.

- Các tính năng chính của Android Studio:

+ Giao diện người dùng trực quan: Android Studio có giao diện người dùng trực quan và dễ sử dụng. Nó cung cấp các tính năng như trình khám phá dự án, trình chỉnh sửa mã, trình gỡ lỗi và trình chạy ứng dụng.

+ Hỗ trợ các ngôn ngữ lập trình: Android Studio hỗ trợ các ngôn ngữ lập trình Java, Kotlin và C/C++. Nó cũng cung cấp các tính năng hỗ trợ cho các ngôn ngữ lập trình khác, chẳng hạn như Python và JavaScript.

+ Tiện ích mở rộng và thư viện: Android Studio có một cộng đồng lớn các nhà phát triển tạo tiện ích mở rộng và thư viện cho IDE. Điều này cung cấp cho các nhà phát triển khả năng mở rộng IDE để đáp ứng nhu cầu cụ thể của họ.

+ Tích hợp với công cụ khác: Android Studio tích hợp với các công cụ khác, chẳng hạn như Gradle, Android SDK và Android emulator. Điều này giúp các nhà phát triển tiết kiệm thời gian và công sức khi phát triển ứng dụng Android.

### c. Flutter

- Flutter là mobile UI framework của Google để tạo ra các giao diện chất lượng cao trên iOS và Android trong khoảng thời gian ngắn. Flutter hoạt động với những code sẵn có được sử dụng bởi các lập trình viên, các tổ chức.

- Flutter hoàn toàn miễn phí và cũng là mã nguồn mở.

- Các ứng dụng được xây dựng với Flutter hầu như không thể phân biệt với những ứng dụng được xây dựng bằng cách sử dụng Android SDK, cả về giao diện và hiệu suất. Hơn nữa, với những tinh chỉnh nhỏ, chúng có thể chạy trên thiết bị iOS.

- Chạy ở 60 fps, giao diện người dùng được tạo ra với Flutter thực thi tốt hơn nhiều so với những ứng dụng được tạo ra với các framework phát triển đa nền tảng khác chẳng hạn như React Native và Ionic

### d. Spring Boot

- Spring Boot là một framework Java được sử dụng để xây dựng các ứng dụng và dịch vụ web dễ dàng và nhanh chóng. Nền tảng cung cấp các cấu hình mặc định cho một số thư viện và bộ công cụ hỗ trợ xây dựng, triển khai, quản lý ứng dụng Spring-based.

- Cách Spring Boot hoạt động nhằm tối ưu hóa quy trình phát triển ứng dụng Java. Điều này sẽ giúp nhà phát triển tập trung vào việc xây dựng tính năng chính của ứng dụng mà không cần phải lo lắng về cấu hình phức tạp

- Những ưu điểm nổi bật của Spring Boot:

+ Tối ưu hóa quá trình phát triển: Spring Boot cung cấp cấu hình mặc định thông minh và tự động, giúp giảm thiểu việc cấu hình thủ công và tối ưu quá trình phát triển ứng dụng Java.

+ Tích hợp tốt: Spring Boot tích hợp tốt với nhiều công nghệ và thư viện khác trong hệ sinh thái Spring Framework. Nền tảng cho phép hệ thống dễ dàng tích hợp các module và dịch vụ khác nhau mà không cần phải lo lắng về cấu hình phức tạp.

+ Embedded server: Spring Boot đi kèm với các máy chủ nhúng như Tomcat, Jetty, hoặc Undertow. Đây là công cụ không thể thiếu trong việc triển khai ứng dụng một cách đơn giản mà không cần cấu hình thêm bất kỳ máy chủ nào khác.

+ Tự động cấu hình: Spring Boot sử dụng cơ chế cấu hình tự động thông minh, cho phép ứng dụng tự cấu hình dựa trên các thư viện và module được sử dụng.

+ Quản lý phụ thuộc: Spring Boot cung cấp các công cụ quản lý phụ thuộc mạnh mẽ như Maven hoặc Gradle, giúp quản lý các phụ thuộc của ứng dụng một cách hiệu quả.

+ Monitoring và quản lý: Spring Boot cung cấp các công cụ hỗ trợ giám sát và quản lý ứng dụng dễ dàng, bao gồm Spring Boot Actuator cho việc giám sát và quản lý ứng dụng.

### e. 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 lí 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.

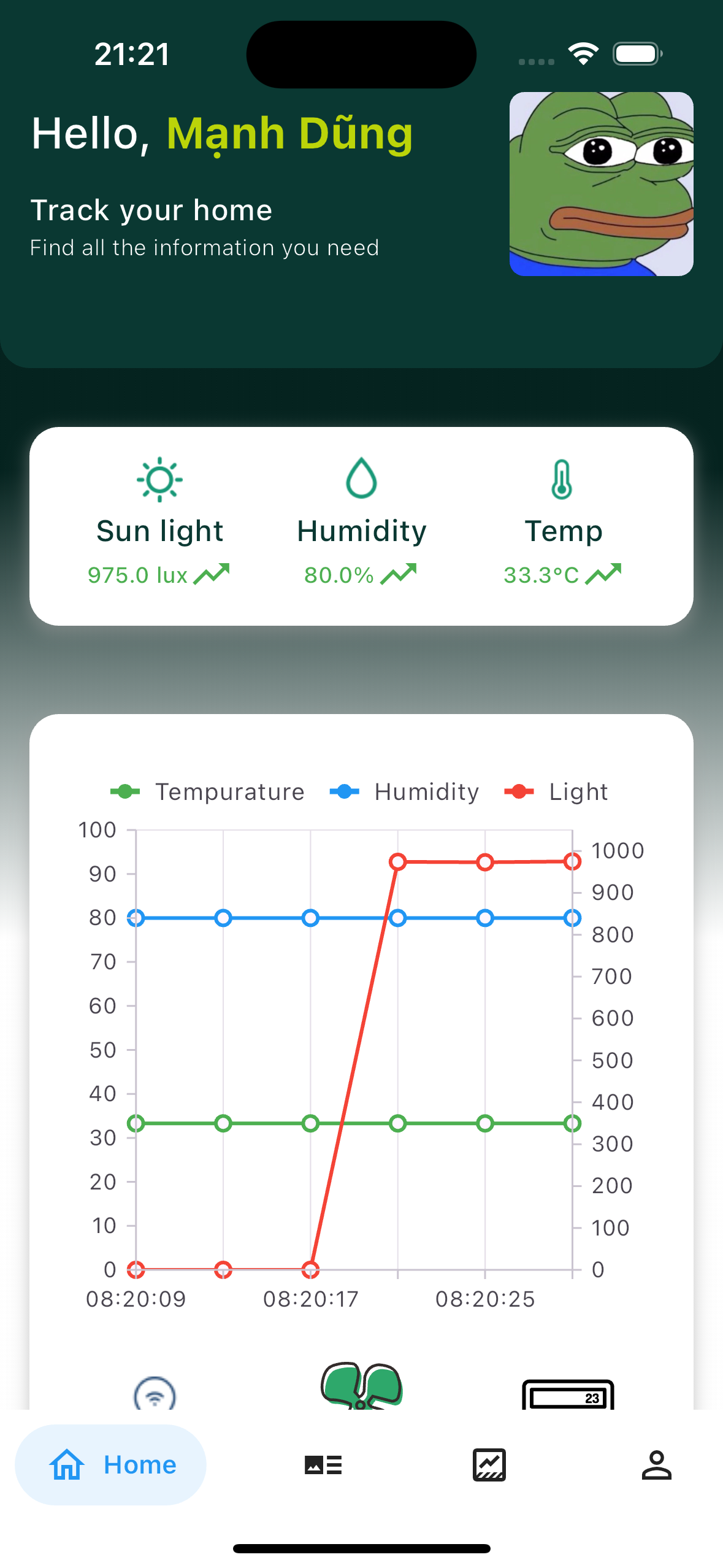
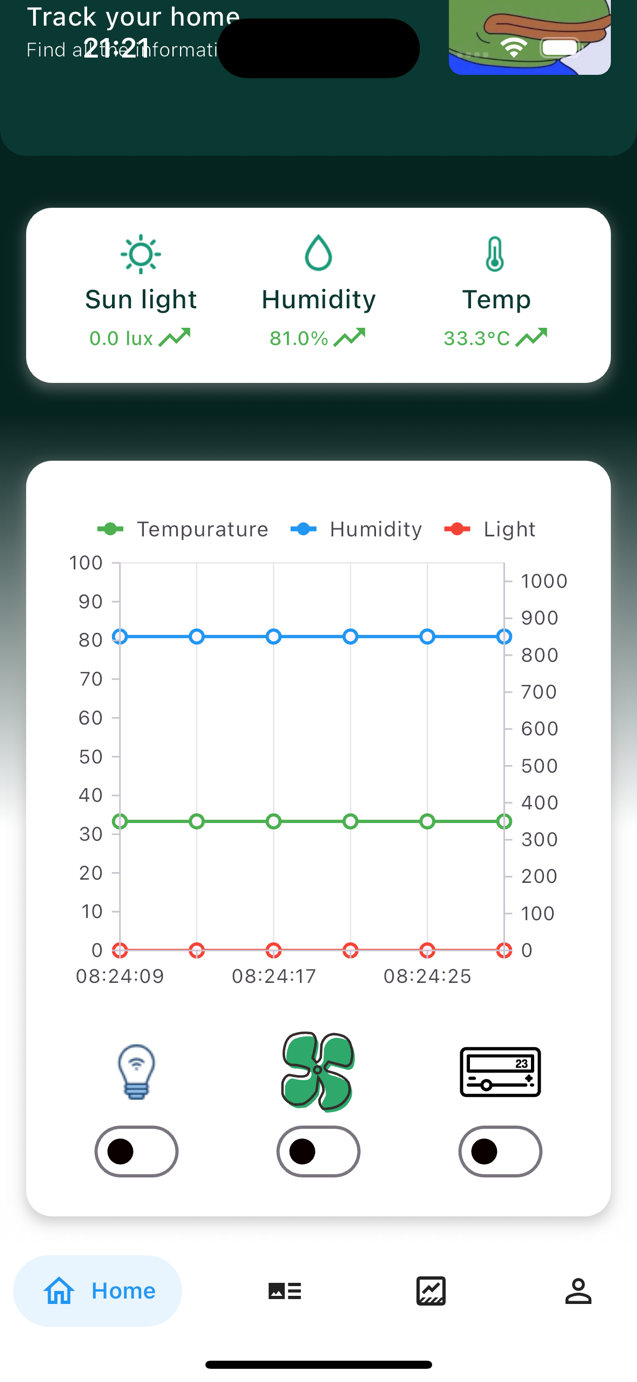
+ Ngoài ra, Mosquitto cũng được hỗ trợ các giao thức TLS/SSL (các giao thức nhằm xác thực server và client, mã hóa các message để bảo mật dữ liệu).

- Nhược điểm:

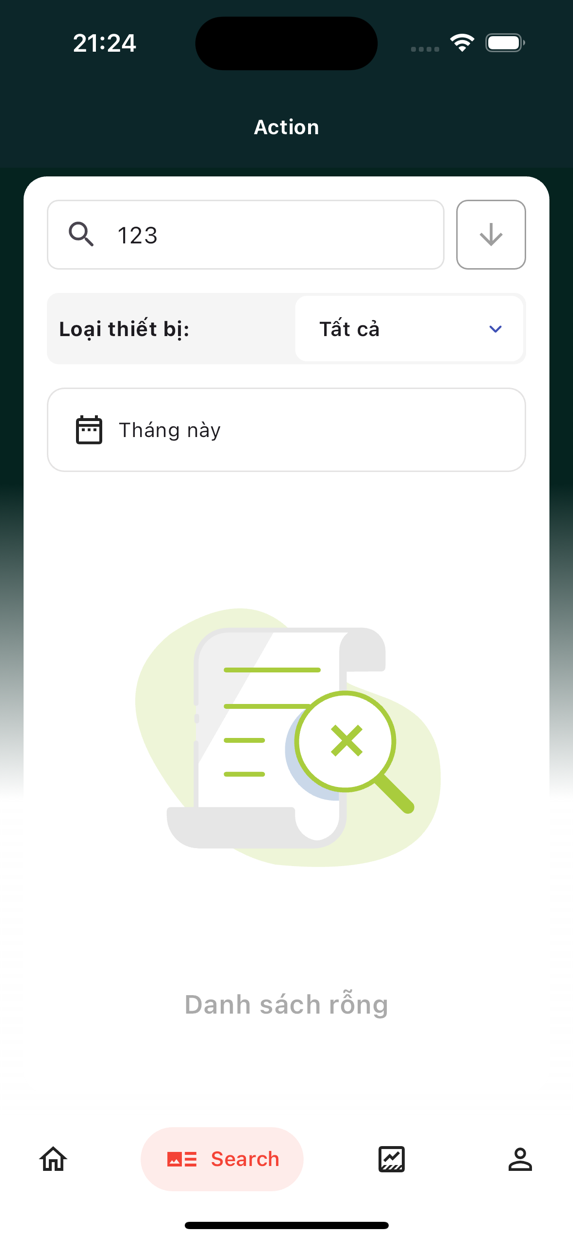
+ Một số nhược điểm của mosquitto là khó thiết kế khi làm những ứng dụng lớn và ít phương thức xác thực thiết bị nên khả năng bảo mật vẫn chưa tối ưu.

# **II. Giao diện, thiết kế thổng thể**

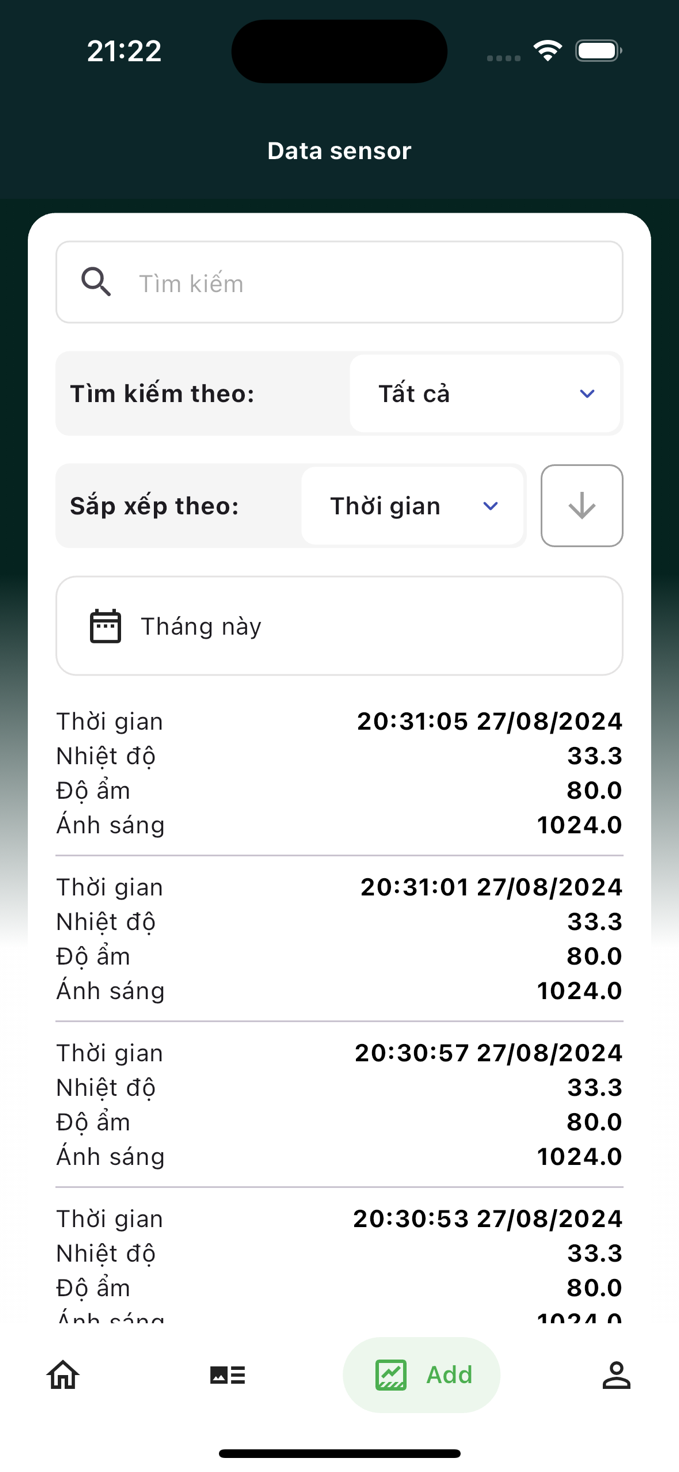
## 1. Giao diện ứng dụng

*Hình 10, 11: Giao diện dasboard*

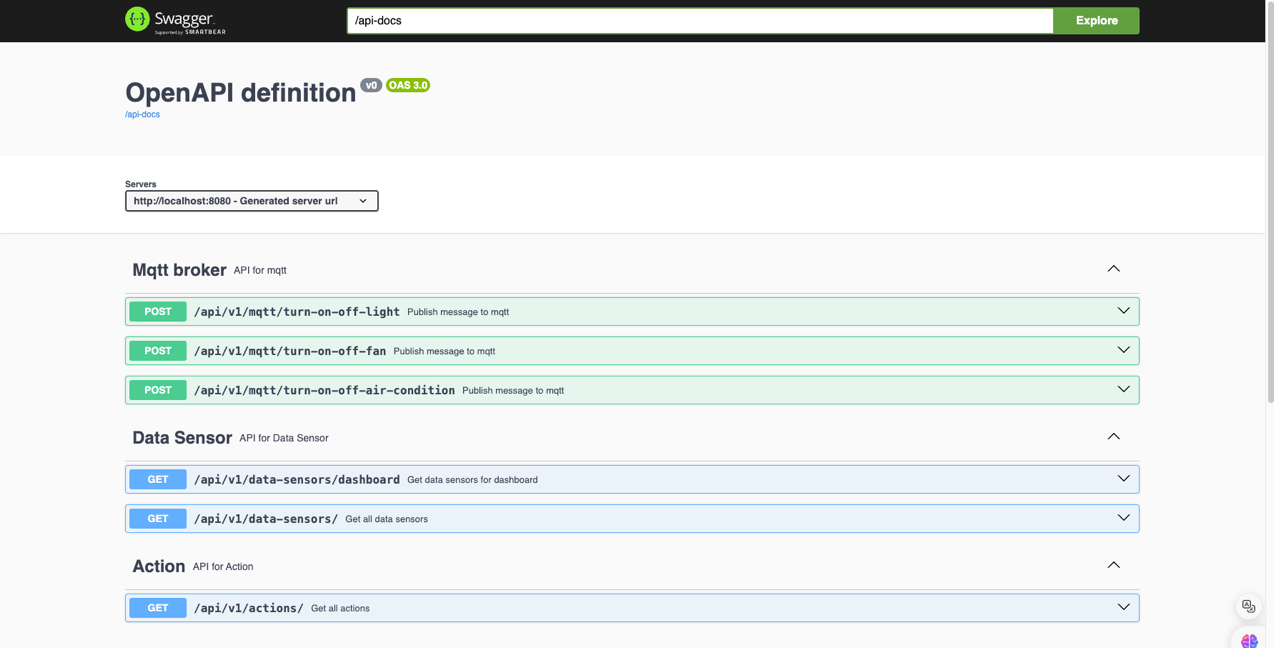
 

*Hình 12, 13: Giao điện màn Action*

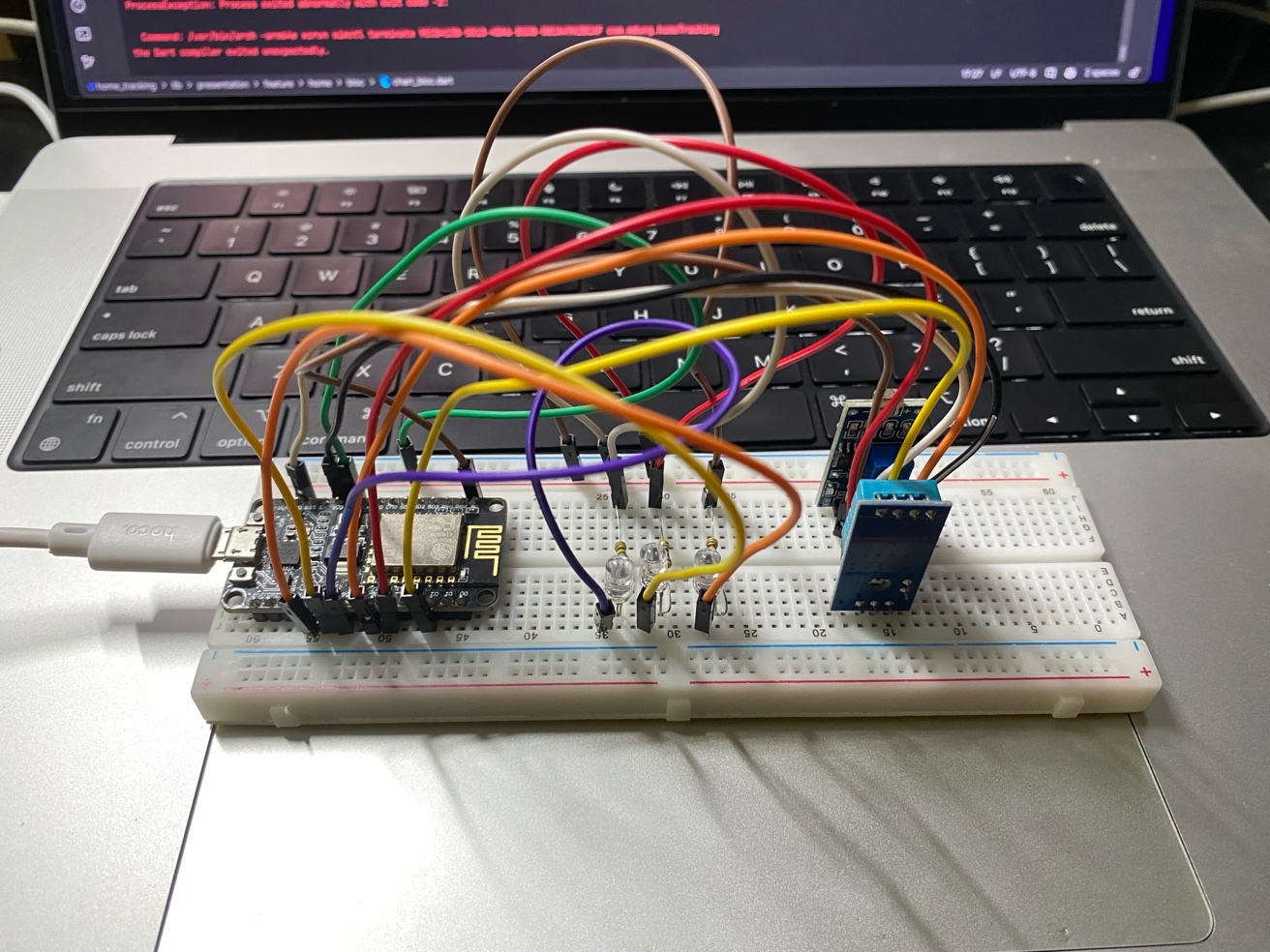
*Hình 13: Giao diện màn Data Sensor Hình 14: Giao diện màn Profile*

## 2. Giao diện API Docs



Hình 15: Giao diện API Docs

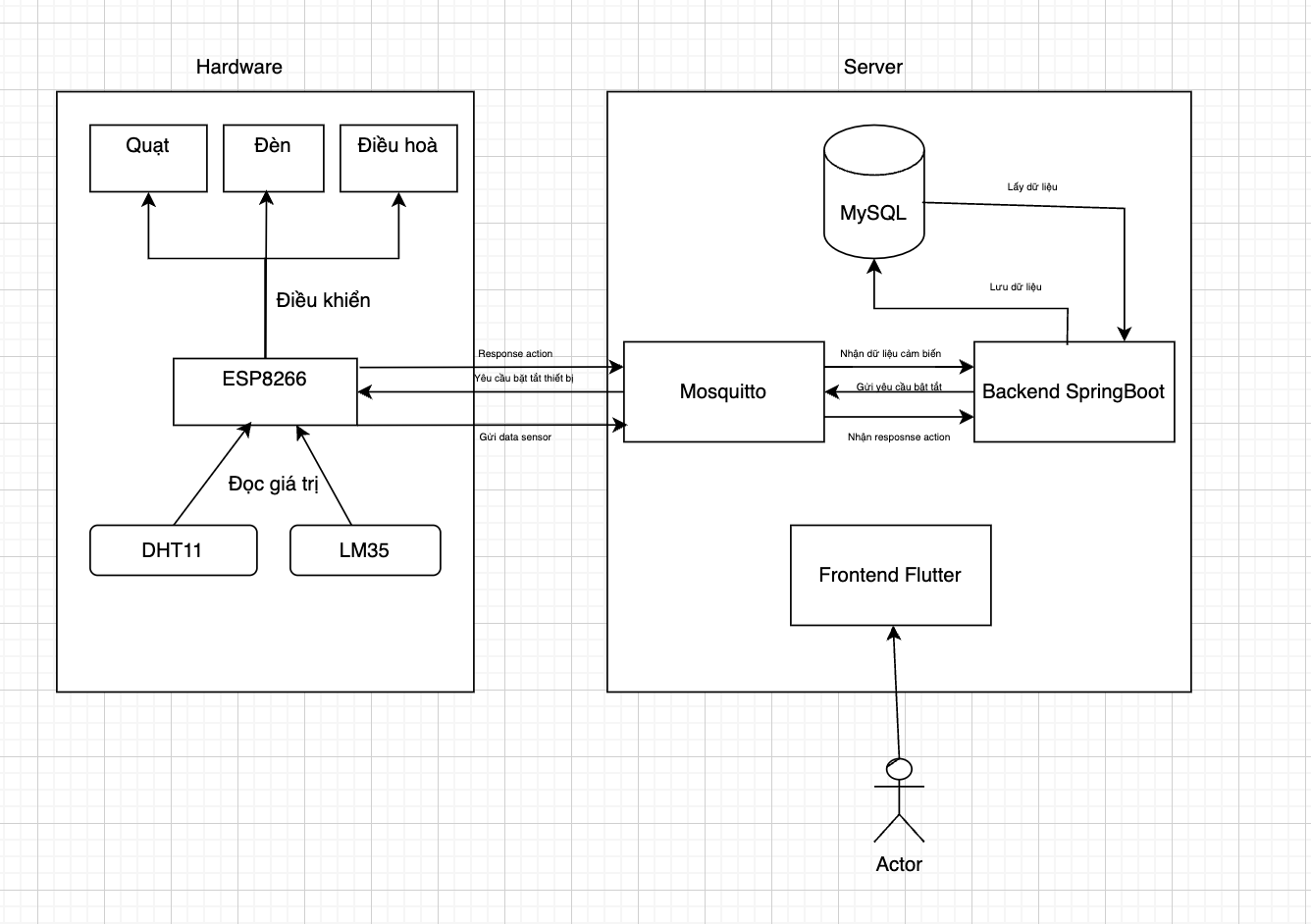
## 3. Giao diện thiết bị



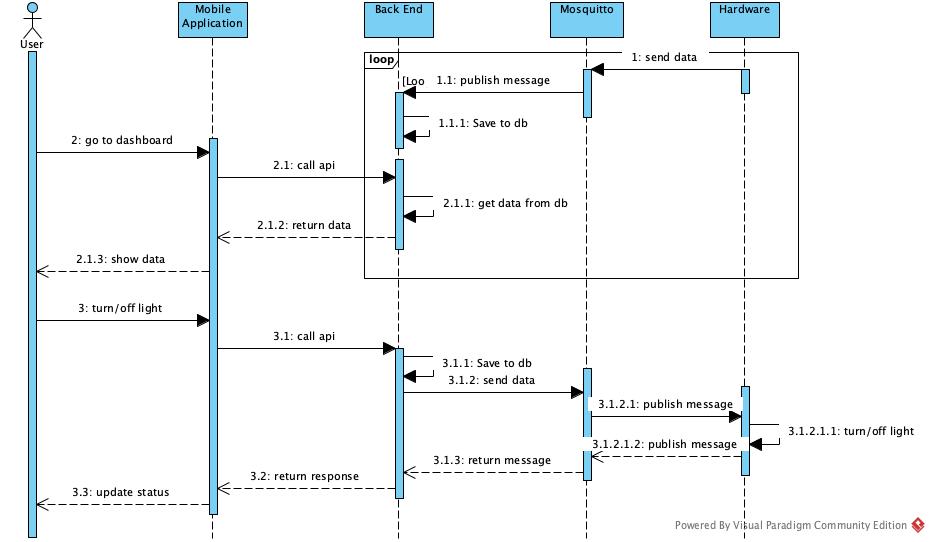
*Hình 16: Giao diện thiết bị*

# **III. Thiết kế chi tiết**

## 1. Thiết kế hệ thống



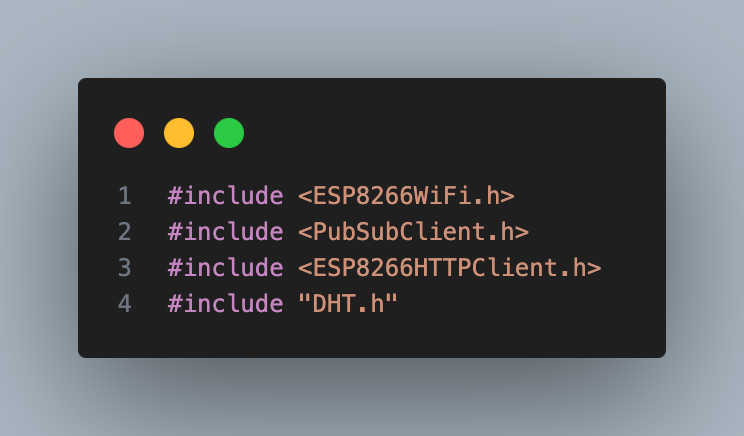
## 2. Sequence Diagram



*Hình 17: Sequence Diagram của hệ thống*

# **IV. Code**

## 1. Embedded code



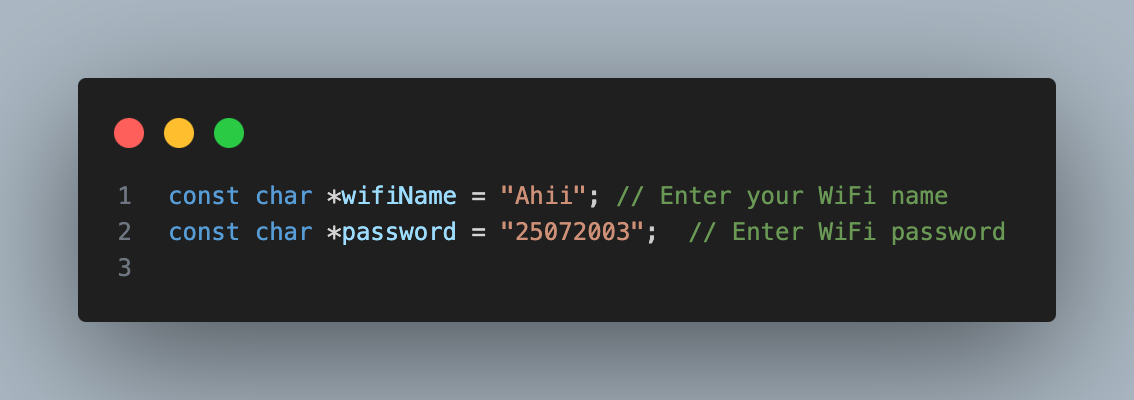
*Hình 18: Code embedded*

- Khai báo các thư viện cần thiết gồm:

+ ESP8266: Thư viện hỗ trợ wifi cho NodeMCU ESP8266

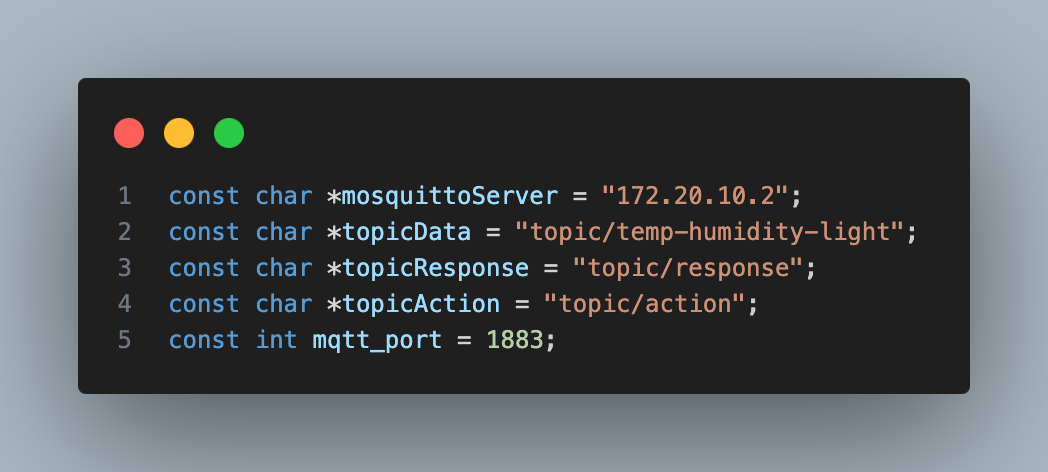
+ PubsubClient: Cung cấp các client để thực hiện các công việc publish, subcribe message với các server hỗ trợ giao thức MQTT

+ DHT.h: Thư viện dành cho các dòng cảm biến nhiệt độ độ ẩm có chi phí thấp của DHT



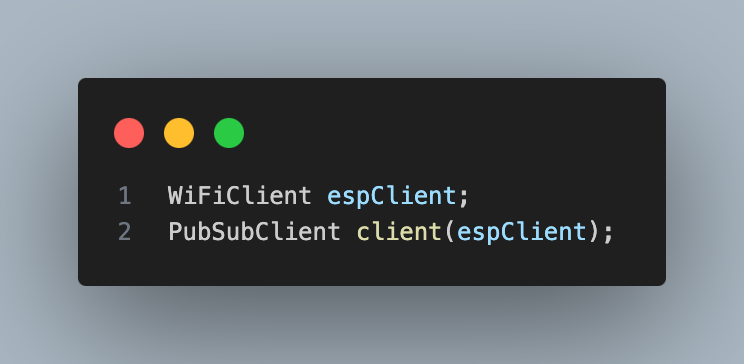
*Hình 19: Code embedded*

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



*Hình 20: Code embedded*

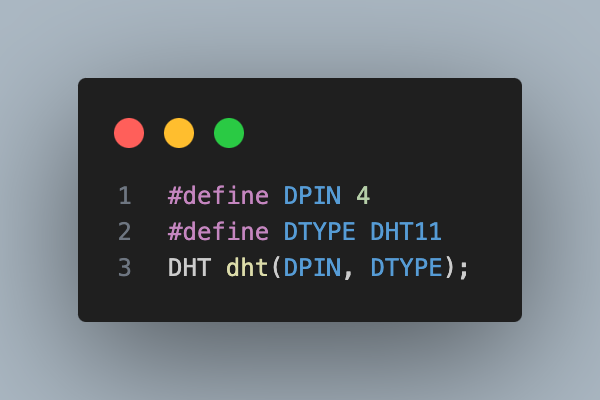
- Định nghĩa các thông tin về server của MQTT Broker (ở đây là Mosquitto), tên các topic, port



*Hình 21: Code embedded*

- WifiClient: Là một class trong thư viện ESP8266Wifi. Lớp này chịu trách nhiệm quản lý kết nối đến Wifi

- Pubsubclient: Là một class trong thư viện PubsubClient. Lớp này chịu trách nhiệm để thực hiện các thao tác liên quan đến giao thức MQTT. Client sẽ sử dụng kết nối TCP/IP của WifiClient để giao tiếp với máy chủ MQTT



*Hình 22: Code embedded*

- Định nghĩa chân kết nối với DHT11 là chân GPIO4, chân Data của DHT11 sẽ nối với chân D2 trong mạnh và định nghĩa loại DHT ở đây là DHT11



*Hình 23: Code embedded*

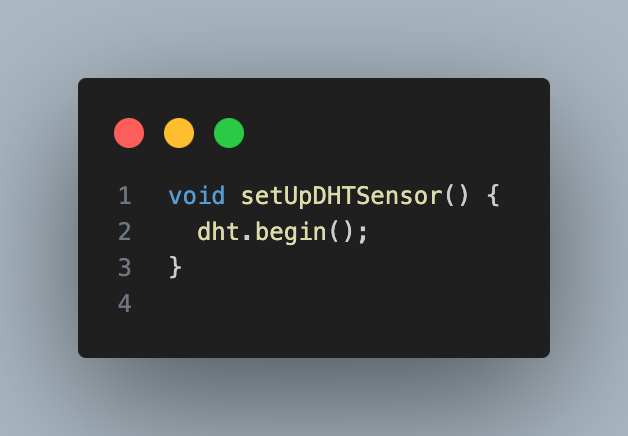
- Định nghĩa chân đọc giá trị của cảm biến ánh sáng (tín hiệu analog)



*Hình 24: Code embedded*

- Khởi tạo cổng kết nối trên vi điều khiển với tốc độ truyền dữ liệu là 115200 baud (bit/giây)

- Gọi các hàm khởi tạo chương trình



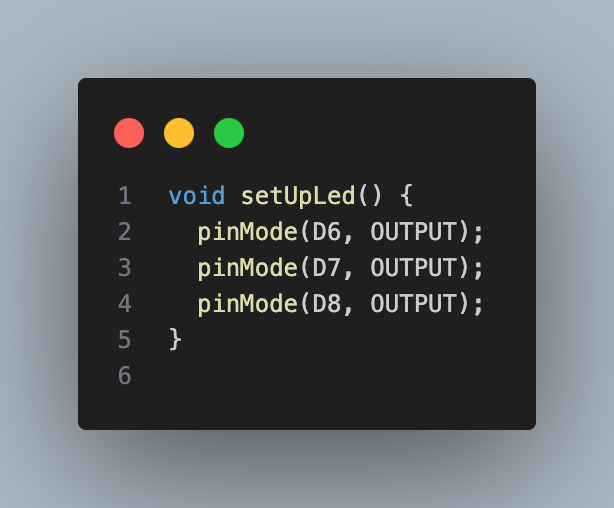
*Hình 25: Code embedded*

- Khởi tạo cảm biến DHT, vi điều khiển sẽ thiết lập các cấu hình cần thiết để bắt đầu đọc dữ liệu từ cảm biến



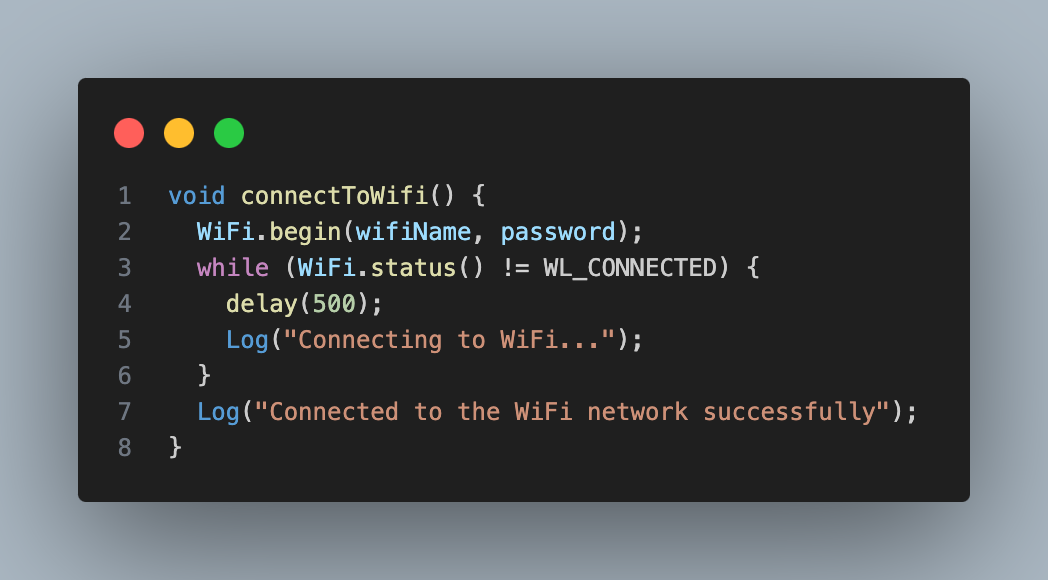
*Hình 26: Code embedded*

- Thiết lập chân LIGHT (chân A0) là chân đầu vào (tức là chân nhận dữ liệu từ cảm biến)



*Hình 27: Code embedded*

- Thiết lập chân đầu ra cho 3 đèn LED (Đầu ra sẽ cung cấp nguồn điện để cho đèn sáng)



*Hình 28: Code embedded*

- Tiến hành kết nối với Wifi

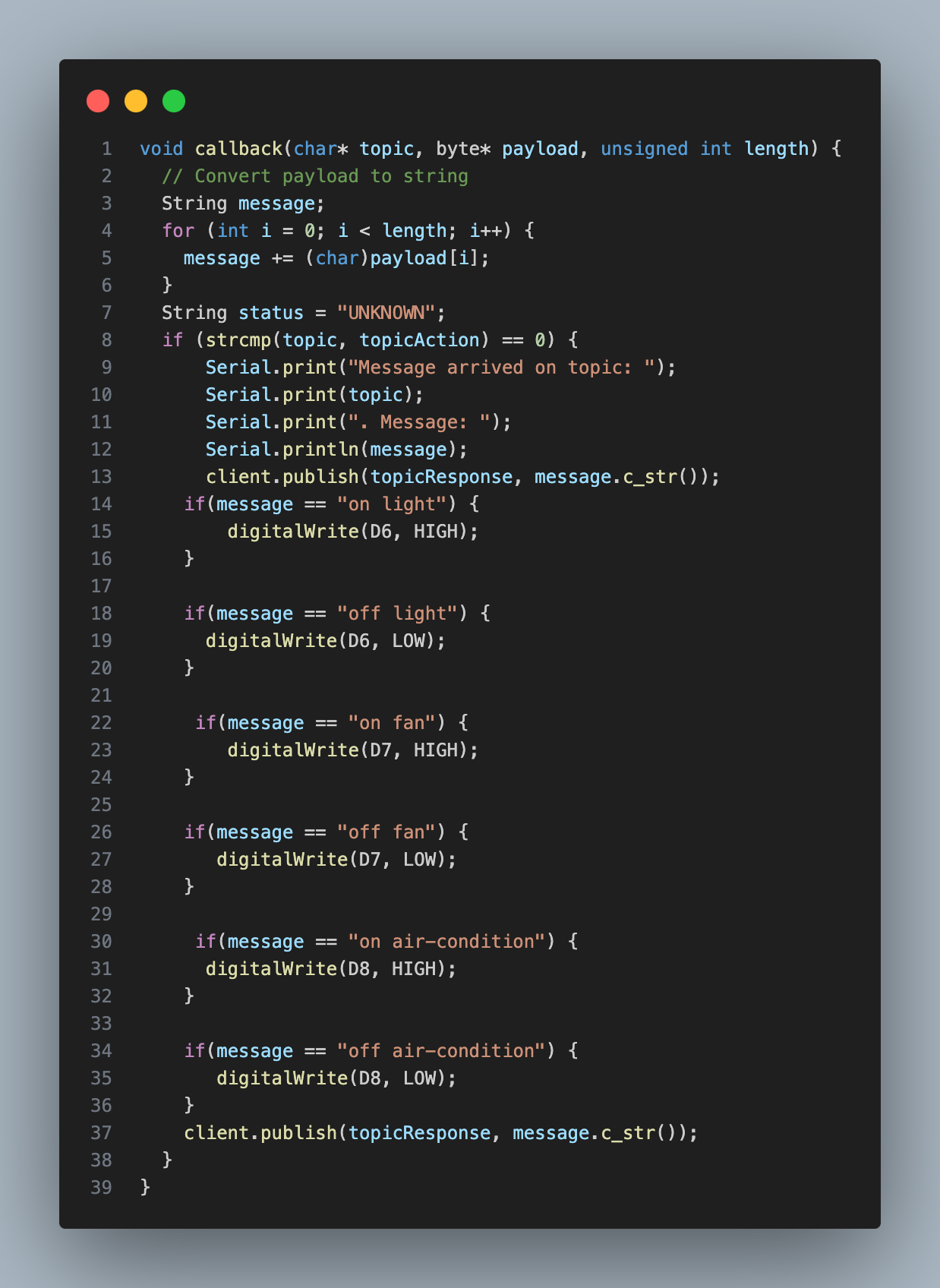


*Hình 29: Code embedded*

- Kết nối vi điều khiển ESP8266 đến MQTT Broker (Mosquitto) thông qua địa chỉ và port của Mosquitto

- Định nghĩa một callback để lắng nghe tất cả các tin nhắn mới nhận được trong số các topic mà thiết bị đã đăng ký

- Sau khi kết nối thành công, tiến hành đăng ký các topic để lắng nghe và phát



*Hình 30: Code embedded*

- Định nghĩa một callback để lắng nghe các message từ topic đã đăng ký.

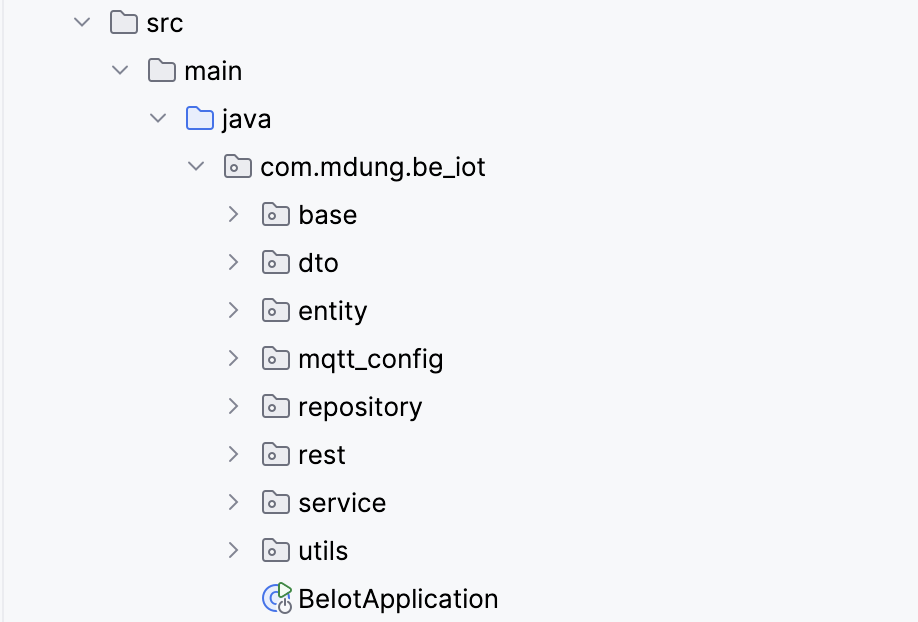
- Nếu nhận được message từ topic yêu cầu bật tắt các thiết bị, tiến hành bật tắt các thiết bị bằng các thiết lập các giá trị LOW hoặc HIGH với các chân cấp nguồn tương ứng cho thiết vị (LOW – 0V, HIGH – 3.3V)



*Hình 31: Code embedded*

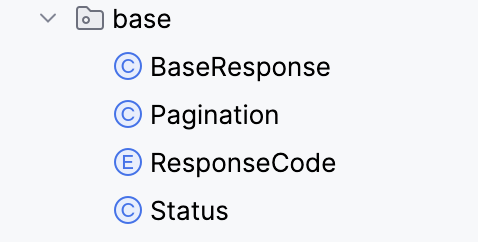
- Vòng lặp chính của chương trình sẽ liên tục đọc giá trị từ các cảm biến và ghép thành dạng Json rồi publish cho BE.

## 2. Backend code

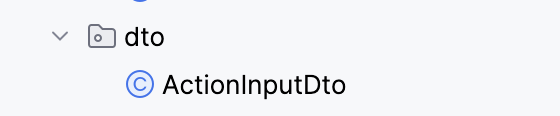


Cấu trúc thư mục của backend gồm các foldel như sau

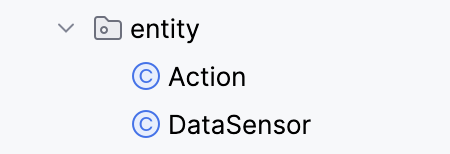
- Base: Định nghĩa các class dùng chung như response trả về, phân trang, status



- Dto: Gồm các data transfer object dùng để chuyển dữ liệu giữa client và server



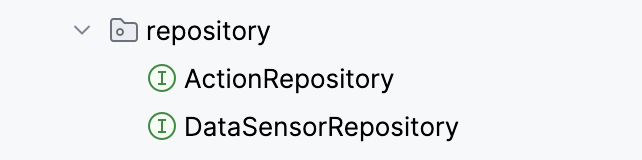
- Entity: Các thực thể của ứng dụng: Action, DataSensor



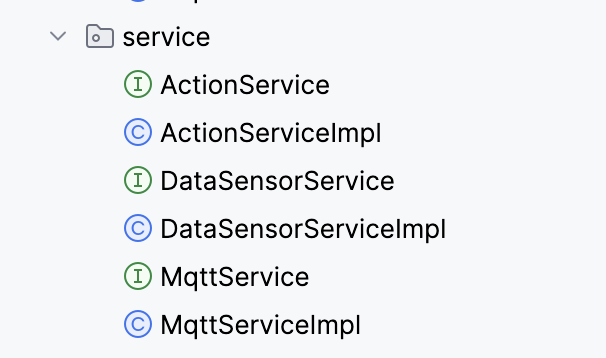
- Mqtt config: Gồm các file để config mqtt, có trách nhiệm gửi và nhận các message từ Mosquito



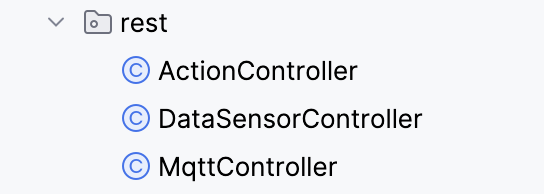
- Repository: Gồm các interface để giao tiếp với cơ sở dữ liệu



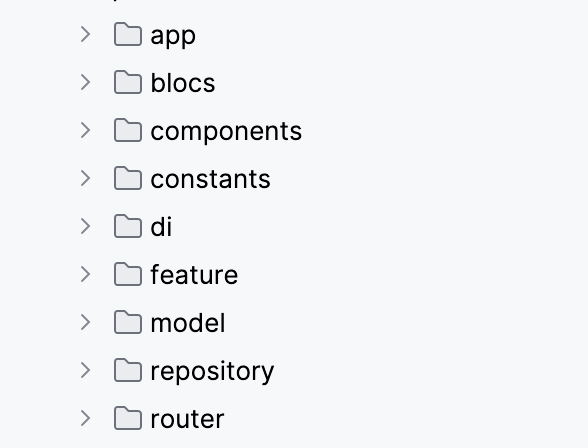
- Service: Là tầng nằm giữa controller, có trách nhiệm đưa dữ liệu từ repository đến controller



- Rest: Là các controller sử dụng trong các dịch vụ RestFullApi



## 3. Frontend Code



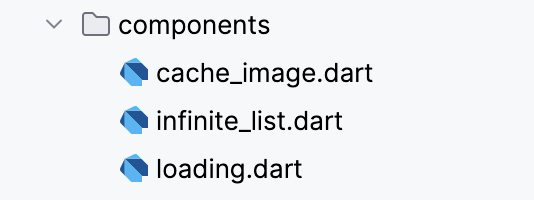
Cấu trúc thư mục như sau:

- App: Định nghĩa nơi bắt đầu khi khởi động ứng dụng

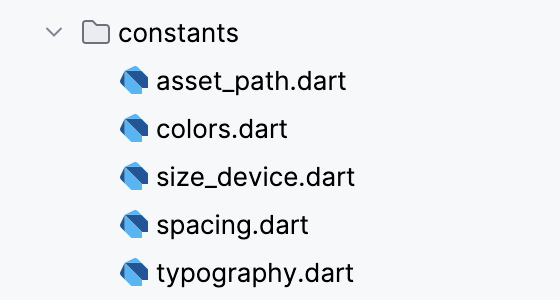
- Bloc: Gồm các file để quản lý state của các màn hình



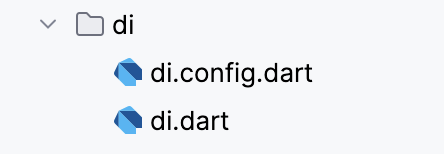
- Components: Định nghĩa các widget dùng chung trong toàn bộ ứng dụng



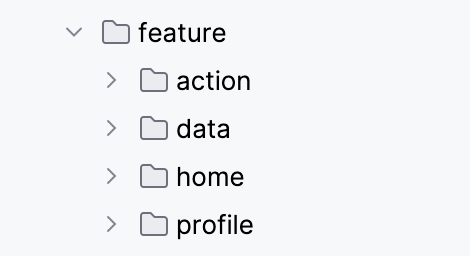
- Constant: Định nghĩa các hằng số như font chữ, font size, spacing, ….



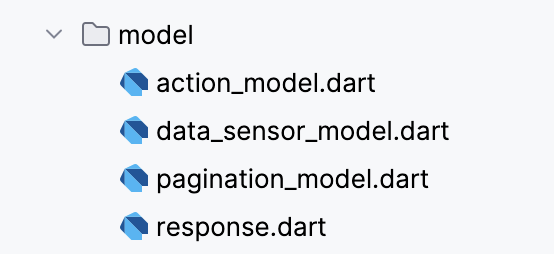
- Di: Phục vụ cho dependency injection



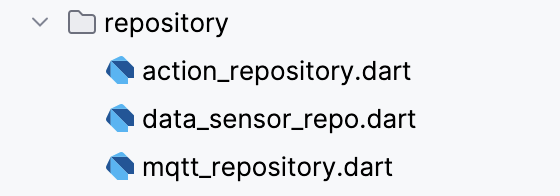
- Feature: Các tính năng chính của ứng dụng



- Model



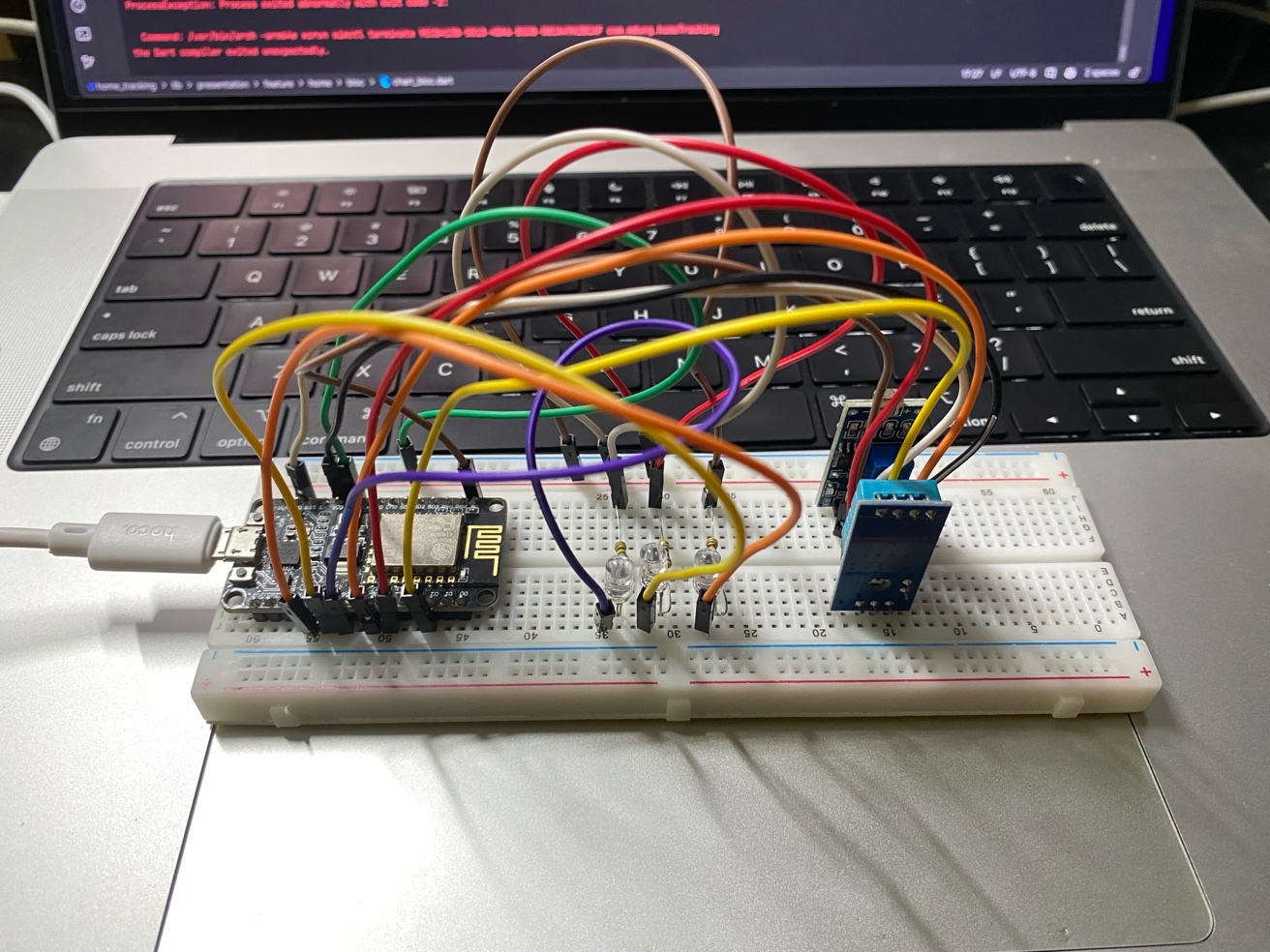
- Repository: Gồm các file để giao tiếp với backend



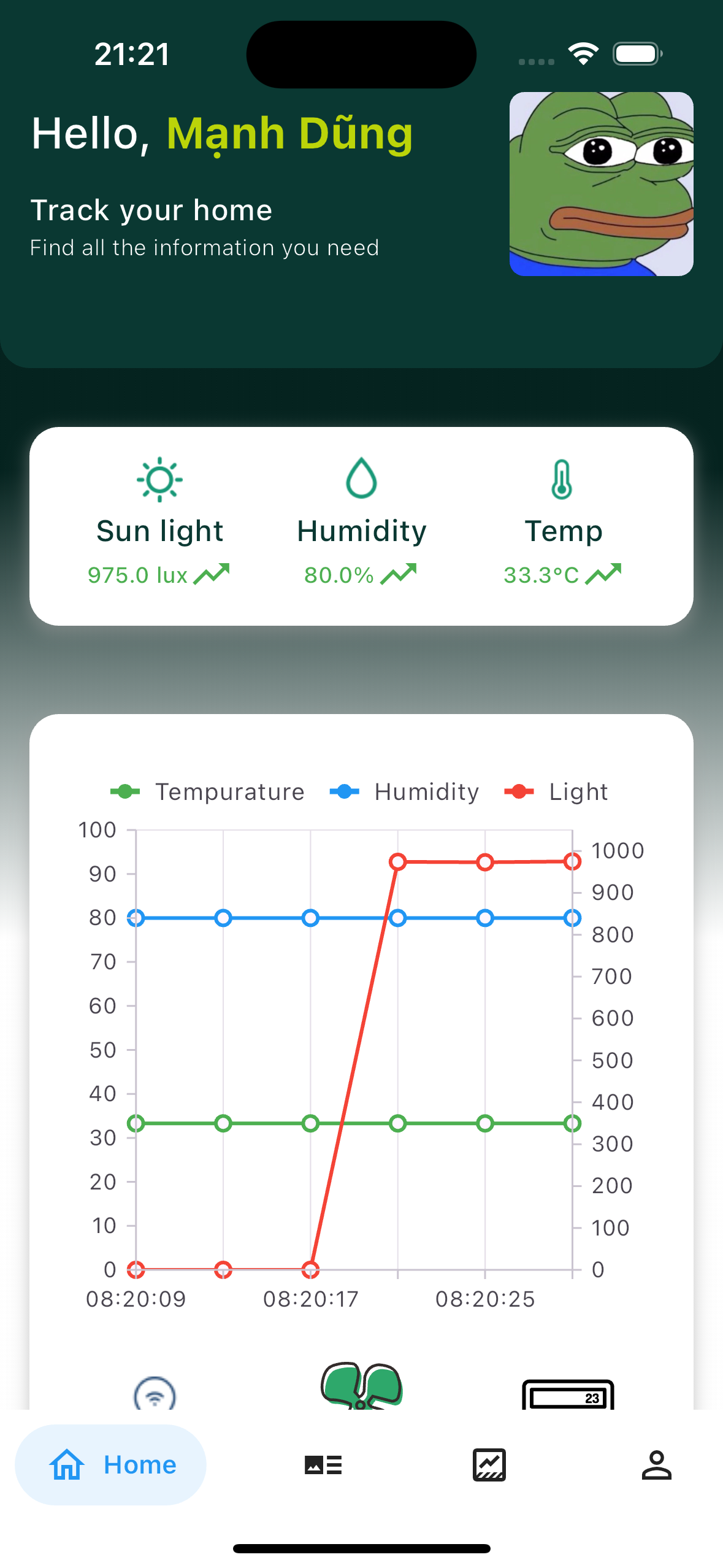
# **V. Kết quả**

## 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 hiện thị cho người dùng một cách tương đối, biểu đồ liên tục cập nhật theo thời gian thực

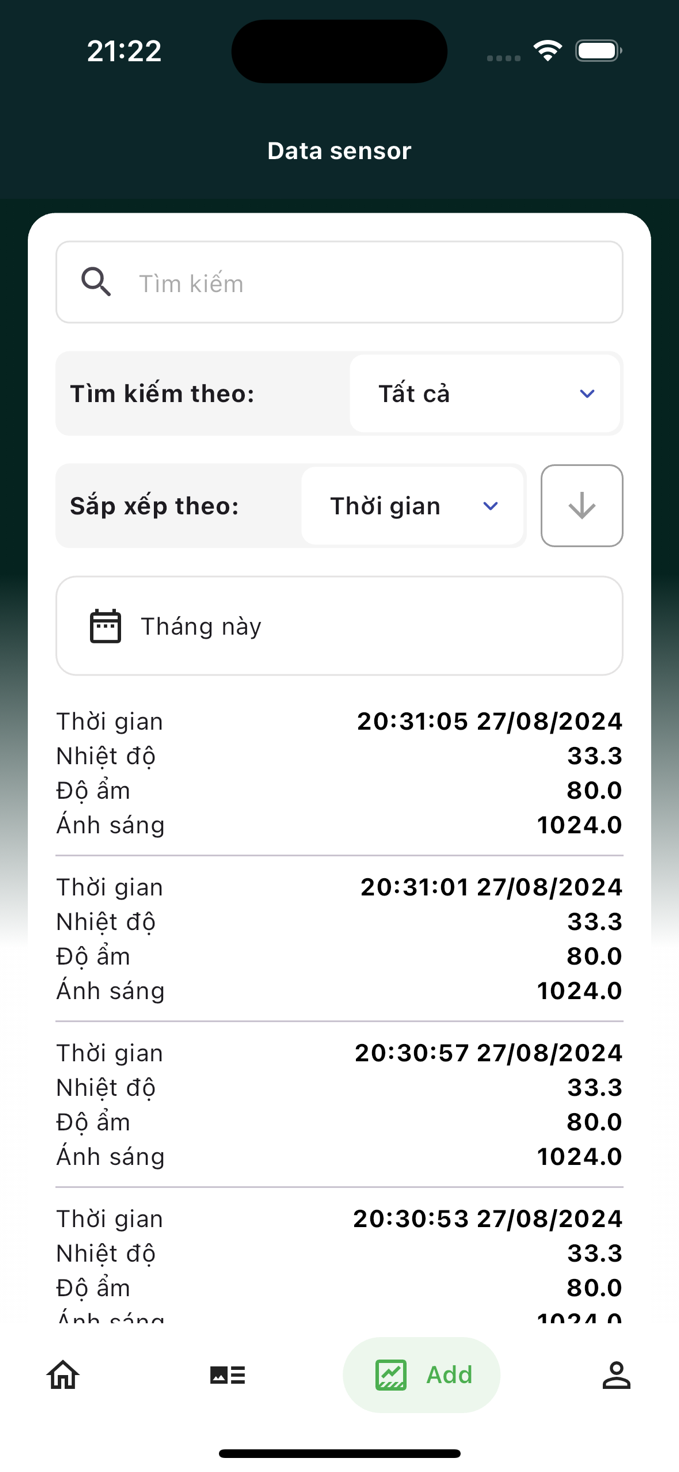
- Khi bật hoặc tắt đèn, ứng dụng chờ phản hồi từ hardware xong thì mới cập nhật UI tương ứng

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



- 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 đọc được từ phần cứng



- 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