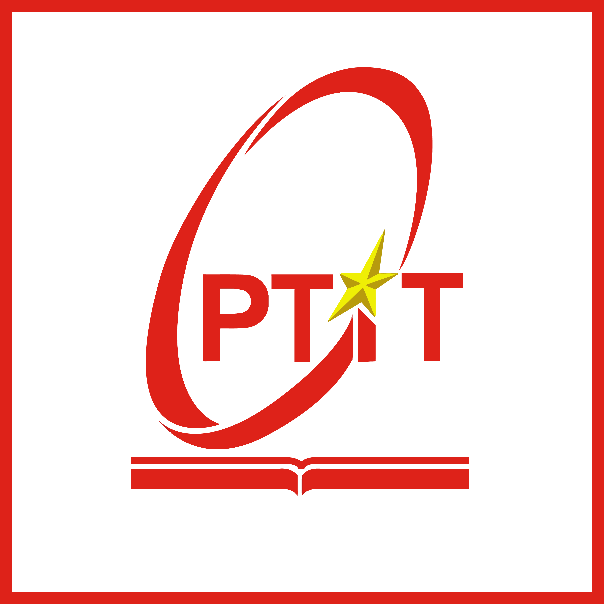
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN**

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



**BÁO CÁO BÀI TẬP LỚN**

**MÔN HỌC: IOT VÀ ỨNG DỤNG**

|  |  |  |
| --- | --- | --- |
| **Giảng viên** | **:** | **Nguyễn Quốc Uy** |
| **Sinh viên** | **:** | **Nguyễn Anh Đức** |
| **Lớp** | **:** | **D21CNPM02** |
| **Mã sinh viên** | **:** | **B21DCCN244** |
| **Số điện thoại** | **:** | **0349831890** |

**Tháng 9/2024**

**Mục lục**

[Lời cảm ơn 2](#_Toc177740683)

[Chương 1. Giới thiệu 3](#_Toc177740684)

[1.1. Đặt vấn đề 3](#_Toc177740685)

[1.2. Mục tiêu và phạm vi đề tài 3](#_Toc177740686)

[1.3. Định hướng giải pháp 3](#_Toc177740687)

[1.4. Bố cục bài tập lớn 5](#_Toc177740688)

[Chương 2. Giao diện và thiết kế tổng thể 7](#_Toc177740689)

[2.1. Giao diện 7](#_Toc177740690)

[2.2. Thiết kế tổng thế 8](#_Toc177740691)

[Chương 3. Chi tiết về đề tài 9](#_Toc177740692)

[3.1. Các thiết bị phần cứng 9](#_Toc177740693)

[3.2. MQTT 14](#_Toc177740694)

[3.3. Backend 17](#_Toc177740695)

[3.4. Frontend 18](#_Toc177740696)

[Chương 4. Mã nguồn 23](#_Toc177740697)

[4.1. Embedded Code 23](#_Toc177740698)

[4.2. Mã Nguồn 27](#_Toc177740699)

[Chương 5. Kết quả thực nghiệm 35](#_Toc177740700)

[5.1. Tổng quan 35](#_Toc177740701)

[5.2. Đo lường các chỉ số như nhiệt độ, độ ẩm, ánh sáng thời gian thực 35](#_Toc177740702)

[5.3. Điều khiển các thiết bị điện 36](#_Toc177740703)

[5.4. Lưu trữ và đưa ra dữ liệu cho người dùng 38](#_Toc177740704)

[Tài liệu tham khảo 41](#_Toc177740705)

# Lời cảm ơn

Trước hết, em xin gửi lời cảm ơn chân thành đến Học viện Công nghệ Bưu chính Viễn thông cùng khoa CNTT1 đã tạo điều kiện đưa môn học IoT và ứng dụng vào chương trình đào tạo. Đặc biệt, em xin gửi lời tri ân sâu sắc đến thầy Nguyễn Quốc Uy, người đã tận tâm giảng dạy, truyền đạt những kiến thức quý báu và giúp em hoàn thiện bản thân trong suốt quá trình học tập.

Nhờ sự hướng dẫn tận tình và chỉ bảo chu đáo của thầy, em đã tiếp thu được nhiều kiến thức bổ ích, rèn luyện được phương pháp làm việc nghiêm túc, hiệu quả. Những bài học này thực sự có ý nghĩa lớn đối với sự phát triển của em sau này. Không những vậy, em còn hoàn thành bài tập lớn của môn học này một cách toàn diện, nhờ vào sự hỗ trợ và chỉ dẫn từ thầy.

Em xin chân thành cảm ơn thầy và chúc thầy luôn mạnh khỏe và thành công trong sự nghiệp giảng dạy cũng như trong cuộc sống!

# Chương 1. Giới thiệu

## 1.1. Đặt vấn đề

Trong xu thế phát triển nhanh chóng của công nghệ, IoT (Internet of Things) đang dần trở thành một phần quan trọng trong việc kết nối và điều khiển các thiết bị điện tử qua Internet. Việc áp dụng IoT vào giám sát các chỉ số môi trường như nhiệt độ, độ ẩm, ánh sáng và điều khiển thiết bị điện đang ngày càng được chú trọng, đặc biệt trong các hệ thống nhà thông minh.

Đề tài này đặt mục tiêu xây dựng một hệ thống IoT cho phép theo dõi các chỉ số môi trường cũng như điều khiển thiết bị điện một cách tự động và hiệu quả.

## 1.2. Mục tiêu và phạm vi đề tài

**1.2.1. Mục tiêu**:

* Phát triển một hệ thống IoT có khả năng giám sát các chỉ số môi trường như nhiệt độ, độ ẩm và cường độ ánh sáng.
* Điều khiển các thiết bị điện (quạt, điều hòa, đèn) dựa trên các chỉ số môi trường được thu thập.
* Xây dựng giao diện người dùng thân thiện, cho phép giám sát và điều khiển hệ thống một cách dễ dàng.

**1.2.2. Phạm vi**:

* Hệ thống sẽ sử dụng các loại cảm biến thông dụng và kết nối với nền tảng IoT để xử lý dữ liệu và đưa ra các quyết định điều khiển tự động.

## 1.3. Định hướng giải pháp

**1.3.1. Phân Tích Yêu Cầu**

Bước đầu tiên trong quy trình phát triển hệ thống là phân tích yêu cầu. Ở giai đoạn này, cần xem xét chi tiết các yêu cầu hệ thống để đảm bảo hệ thống đáp ứng đầy đủ các mục tiêu đặt ra. Trước hết, cần xác định rõ các cảm biến cần thiết để thu thập dữ liệu môi trường, chẳng hạn như cảm biến đo nhiệt độ, độ ẩm và ánh sáng. Đồng thời, lựa chọn các thiết bị điều khiển như quạt, điều hòa không khí, và đèn để có thể điều chỉnh môi trường dựa trên dữ liệu thu thập được.

Ngoài ra, cần đánh giá phương thức kết nối giữa các thành phần của hệ thống. Điều này bao gồm lựa chọn giao thức truyền thông phù hợp, chẳng hạn như MQTT, để đảm bảo việc giao tiếp giữa các thiết bị và hệ thống trung tâm diễn ra suôn sẻ. Cần đặc biệt lưu ý đến các yếu tố như độ tin cậy, tính ổn định của kết nối, và khả năng mở rộng hệ thống trong tương lai khi số lượng cảm biến và thiết bị tăng lên.

**1.3.2. Thiết Kế Hệ Thống**

Giai đoạn thiết kế hệ thống là bước quan trọng trong việc lập kế hoạch chi tiết cho cả phần cứng và phần mềm của hệ thống. Ở bước này, cần tạo ra sơ đồ tổng thể của hệ thống, xác định rõ các module phần cứng và phần mềm, đồng thời xây dựng cơ chế tương tác giữa chúng.

Về phần cứng, cần lựa chọn các linh kiện và cảm biến thích hợp, thiết kế mạch điện tử, và bố trí các linh kiện trên bảng mạch sao cho hiệu quả. Đối với phần mềm, cần phát triển giao diện người dùng trực quan, xây dựng các thuật toán xử lý dữ liệu và thiết kế hệ thống lưu trữ, phân tích dữ liệu một cách chặt chẽ. Sự tương thích giữa phần cứng và phần mềm phải được đảm bảo, bao gồm sự phù hợp về giao thức truyền thông, điện áp của các linh kiện và khả năng tích hợp phần mềm.

Hơn nữa, cần tính đến khả năng mở rộng của hệ thống, giúp dễ dàng thêm mới các cảm biến hoặc thiết bị trong tương lai mà không làm ảnh hưởng đến hiệu suất hoặc hoạt động của hệ thống.

**1.3.3. Phát Triển và Thử Nghiệm**

Giai đoạn phát triển hệ thống bao gồm việc triển khai toàn bộ các thành phần dựa trên bản thiết kế đã được phê duyệt. Quá trình này bao gồm việc lắp đặt phần cứng, lập trình phần mềm và tích hợp chúng lại với nhau thành một hệ thống hoàn chỉnh. Trong suốt quá trình phát triển, cần thực hiện các bước kiểm tra liên tục để đảm bảo rằng tất cả các thành phần đều hoạt động đồng bộ và đúng như kỳ vọng.

Sau khi hệ thống được lắp ráp, tiến hành thử nghiệm là một bước thiết yếu nhằm kiểm tra tính năng và độ ổn định của hệ thống. Các kịch bản thử nghiệm cần được xây dựng bao quát các tình huống thực tế và giả định những lỗi có thể xảy ra. Việc này sẽ giúp đánh giá khả năng xử lý dữ liệu từ các cảm biến, khả năng điều khiển thiết bị một cách chính xác, và hiệu suất chung của hệ thống trong thời gian dài.

Kết quả thử nghiệm sẽ giúp xác định những điểm cần tối ưu hóa hoặc cải tiến để nâng cao hiệu suất và độ ổn định của hệ thống. Cuối cùng, việc biên soạn tài liệu hướng dẫn sử dụng và bảo trì chi tiết sẽ giúp người dùng cuối có thể sử dụng và quản lý hệ thống một cách dễ dàng và hiệu quả.

## 1.4. Bố cục bài tập lớn

**Chương 1: Giới thiệu**

Chương này cung cấp cơ sở để các phần tiếp theo của báo cáo đi sâu vào các chi tiết kỹ thuật và phân tích cụ thể hơn

**Chương 2: Giao diện và thiết kế tổng thể**

Chương này tập trung vào việc trình bày các giao diện người dùng của hệ thống và thiết kế tổng thể. Nó bao gồm mô tả các giao diện người dùng chính, thiết kế tổng thể của hệ thống, cùng với các sơ đồ khối và sơ đồ kết nối giữa các thành phần. Mục tiêu là cung cấp cái nhìn tổng quát về cấu trúc hệ thống và cách các thành phần tương tác với nhau.

**Chương 3: Chi tiết hệ thống**

Chương này đi sâu vào mô tả chi tiết từng thành phần của hệ thống. Nó bao gồm các cảm biến, bộ điều khiển, và các thành phần quan trọng khác như broker MQTT, backend, và frontend. Mục đích là cung cấp thông tin đầy đủ về cấu trúc và chức năng của từng phần của hệ thống, làm rõ cách chúng góp phần vào hoạt động tổng thể của hệ thống.

**Chương 4: Mã Nguồn**

Chương này trình bày mã nguồn của hệ thống, tổ chức các thư mục và tập tin, và giải thích cách thức hoạt động của phần mềm. Nó bao gồm việc mô tả cấu trúc mã nguồn, các module chính, và cách mà mã nguồn thực hiện các chức năng của hệ thống.**Chương 5: Kết quả**

Chương này đánh giá kết quả thử nghiệm hệ thống, so sánh với các mục tiêu đã đặt ra, và đưa ra nhận xét về tính hiệu quả của hệ thống. Nó bao gồm phân tích các dữ liệu thu được từ quá trình thử nghiệm, đánh giá hiệu suất và độ ổn định của hệ thống, và đưa ra các kết luận về sự thành công của dự án.

# Chương 2. Giao diện và thiết kế tổng thể

## 2.1. Giao diện

Chúng ta sẽ có tổng cộng 4 giao diện với các chức năng khác nhau;

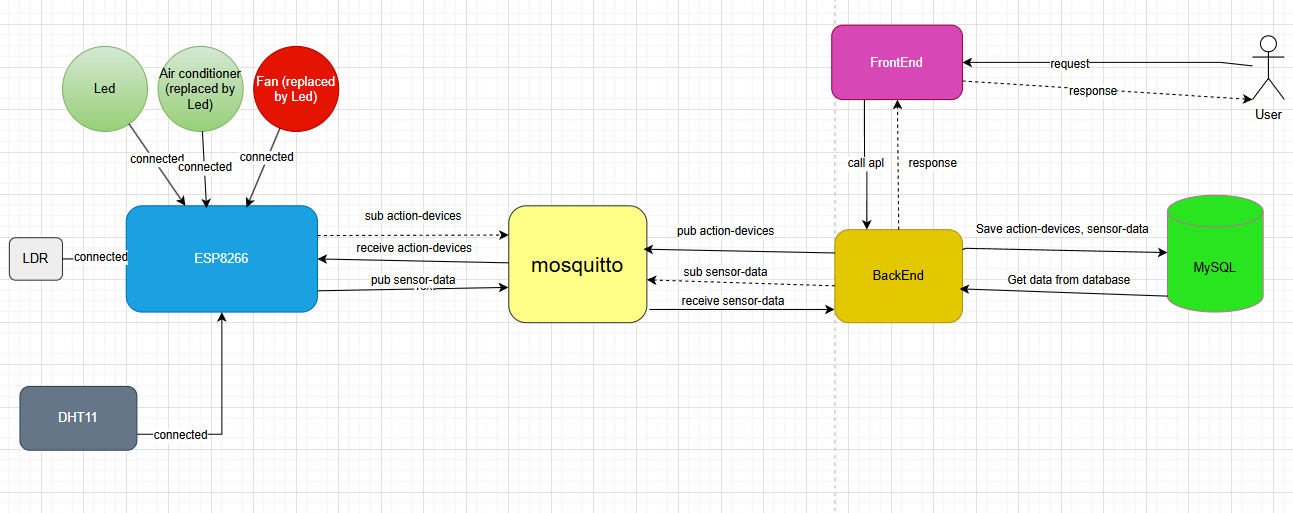
**Giao diện Dashboard:** hiển thị các thông số như nhiệt độ, độ ẩm và ánh sáng một cách rõ ràng, cho phép người dùng quan sát sự thay đổi của chúng. Các biểu đồ sẽ thể hiện sự biến động của các thông số này, giúp người dùng dễ dàng nhận diện các xu hướng và biến đổi quan trọng. Màn hình này cũng bao gồm các nút điều khiển để bật hoặc tắt các thiết bị điện như quạt, điều hòa và đèn, tạo sự thuận tiện trong việc quản lý các thiết bị này.

**Giao diện Data Sensor:** người dùng có thể xem và quản lý dữ liệu lịch sử của các cảm biến trên màn hình này. Các chức năng phân trang, tìm kiếm và sắp xếp được tích hợp để người dùng dễ dàng điều hướng qua dữ liệu, và nhanh chóng tìm kiếm thông tin cần thiết

**Giao diện Action History** màn hình này tập trung vào việc lưu trữ và quản lý lịch sử các hành động liên quan đến việc bật hoặc tắt các thiết bị điện. Người dùng có thể xem lại tất cả các hành động đã thực hiện, với các tùy chọn phân trang và tìm kiếm theo khoảng thời gian để dễ dàng theo dõi các sự kiện cụ thể. Đồng thời, màn hình cũng cho phép lọc dữ liệu theo từng thiết bị, giúp người dùng xác định các thay đổi và hoạt động của từng thiết bị trong hệ thống.

**Giao diện Profile:** Đây là nơi lưu trữ thông tin cá nhân chi tiết. Màn hình Profile cung cấp cái nhìn tổng quan về thông tin cá nhân như tên, ảnh đại diện, mã sinh viên, lớp và các thông tin liên quan khác như liên kết đến GitHub và báo cáo.

## 2.2. Thiết kế tổng thế



***Hình*** ***1: Thiết kế tổng thế của hệ thống***

# Chương 3. Chi tiết về đề tài

## 3.1. Các thiết bị phần cứng

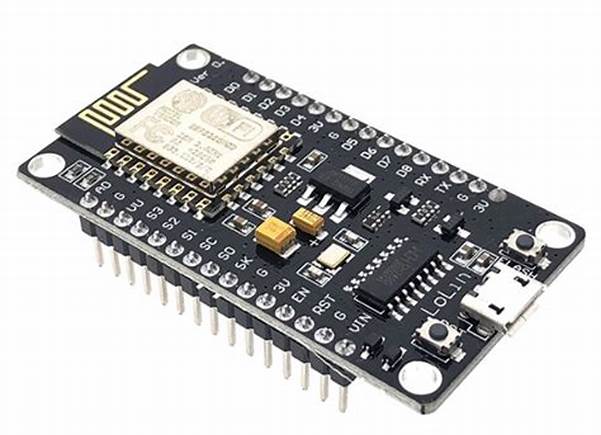
Trong hệ thống IoT được triển khai, các thành phần phần cứng đóng vai trò quan trọng trong việc thu thập dữ liệu, xử lý và điều khiển các thiết bị điện tử. Những thành phần cốt lõi trong dự án bao gồm các cảm biến để theo dõi các thông số môi trường như nhiệt độ, độ ẩm và ánh sáng. Các cảm biến này được kết nối với các vi điều khiển như ESP8266, giúp truyền dữ liệu lên mạng thông qua giao thức MQTT. Ngoài ra, các thiết bị đầu ra như đèn LED, quạt hoặc các thiết bị điện tử khác được tích hợp để phản hồi hoặc thực hiện các lệnh điều khiển từ xa từ hệ thống. Những thành phần này phối hợp chặt chẽ để tạo ra một mạng lưới IoT thông minh, tự động hóa và giám sát hiệu quả.

**3.1.1. ESP8266**:

ESP8266 là một vi điều khiển Wi-Fi mạnh mẽ, được sử dụng để kết nối hệ thống IoT với mạng không dây. Nó đảm nhận nhiệm vụ giao tiếp với các thiết bị cảm biến và thiết bị điều khiển, đồng thời gửi và nhận dữ liệu thông qua MQTT (Message Queuing Telemetry Transport).

**Chức năng**:

* Thu thập dữ liệu từ các cảm biến.
* Điều khiển các thiết bị như quạt, đèn, và máy điều hòa không khí dựa trên yêu cầu từ máy chủ hoặc người dùng.
* Gửi dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng) đến máy chủ thông qua broker MQTT.



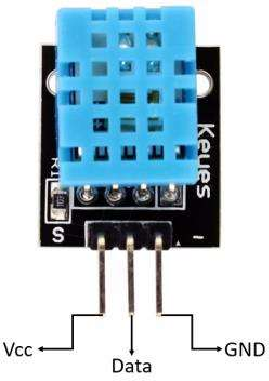
Hình : Module ESP8266

**3.1.2. Cảm biến nhiệt độ và độ ẩm**:

Cảm biến DHT11 được sử dụng để đo nhiệt độ và độ ẩm trong môi trường. Đây là loại cảm biến có độ chính xác tương đối cao và được sử dụng phổ biến trong các ứng dụng IoT để giám sát môi trường.

**Chức năng**:

* Đo lường nhiệt độ và độ ẩm của môi trường xung quanh.
* Truyền dữ liệu đến ESP8266 để xử lý và gửi lên máy chủ.



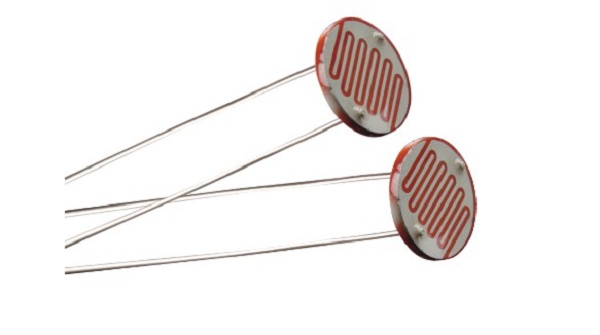
Hình : Ảnh về cảm biến DHT11

**3.1.3. Cảm biến ánh sáng**:

LDR là một loại điện trở thay đổi giá trị của nó dựa trên cường độ ánh sáng chiếu vào. Khi cường độ ánh sáng thay đổi, điện trở của LDR cũng thay đổi, từ đó giúp hệ thống nhận biết mức độ ánh sáng trong môi trường.

**Chức năng**:

* Đo mức độ ánh sáng xung quanh.
* Gửi dữ liệu đến ESP8266 để xử lý và quyết định các hành động điều khiển thiết bị (như bật/tắt đèn).



Hình : Ảnh cảm biến ánh sáng (quang trở)

**3.1.4. Đèn LED**:

Ba đèn LED được sử dụng để tượng trưng cho ba thiết bị điện khác nhau: **quạt, điều hòa và đèn chiếu sáng**.

**Chức năng**:

* Nhận lệnh điều khiển từ ESP8266 để bật hoặc tắt.
* Được điều khiển tự động dựa trên dữ liệu thu thập từ các cảm biến (ví dụ, bật đèn khi cường độ ánh sáng giảm).



Hình : Ảnh đèn LED

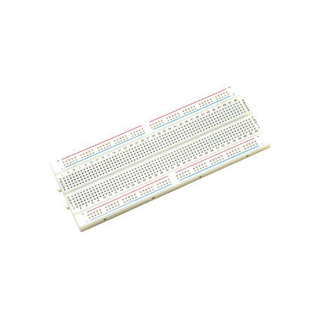
**3.1.5. Các thiết bị phụ trợ khác**:

* **Dây nối**: Dây nối (Jumper wires) là những dây dẫn nhỏ dùng để kết nối các thành phần điện tử với nhau trên breadboard hoặc kết nối với các module như ESP8266.



Hình : Dây nối

* **Breadboard**: Breadboard là một bảng mạch không hàn, cho phép kết nối tạm thời các thành phần điện tử với nhau mà không cần phải hàn mạch.



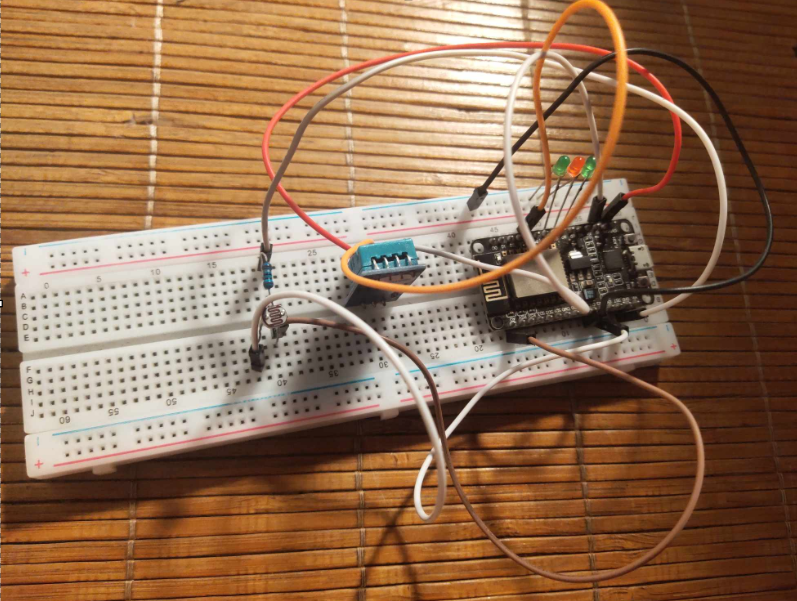
Hình : Breadboard

* **Điện trở**: Điện trở là một linh kiện điện tử giúp giới hạn dòng điện trong mạch để bảo vệ các thiết bị và điều chỉnh mức điện áp cho phù hợp với các cảm biến hoặc thiết bị.



Hình : Điện trở

Dưới đây là hệ thống phần cứng hoàn chỉnh:



Hình : Hệ thống phần cứng hoàn chỉnh

## 3.2. MQTT

**3.2.1. Lý thuyết về MQTT**

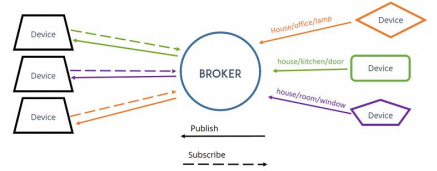
MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp / thuê bao), được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Nó dựa trên một Broker (tạm dịch là “Máy chủ môi giới”) “nhẹ” (khá ít xử lý) và được thiết kế có tính mở (tức là không đặc trưng cho ứng dụng cụ thể nào), đơn giản và dễ cài đặt.

**Mô hình Pub/Sub và Cơ chế hoạt động của MQTT**

* Client
  + Publisher - Nơi gửi thông điệp
  + Subscriber - Nơi nhận thông điệp
* Broker - Máy chủ môi giới

Trong đó Broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ Client (Publisher/Subscriber). Nhiệm vụ chính của Broker là nhận thông điệp (message) từ Publisher, xếp vào hàng đợi rồi chuyển đến một địa điểm cụ thể. Nhiệm vụ phụ của Broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs, ....

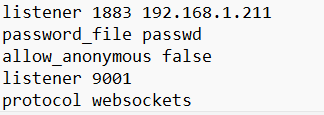
Client thì được chia thành hai nhóm là Publisher và Subscriber. Client chỉ làm ít nhất một trong 2 việc là publish các thông điệp (message) lên một/nhiều topic cụ thể hoặc subscribe một/nhiều topic nào đó để nhận message từ topic này.



MQTT Clients tương thích với hầu hết các nền tảng hệ điều hành hiện có: MAC OS, Windows, Linux, Android, iOS, ...

**3.2.2. Sử dụng MQTT trong dự án**

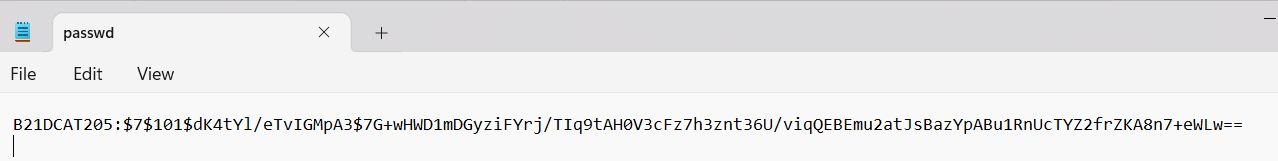
Ban đầu em sẽ cấu hình cho mqtt với các thông số dưới đây



Hình : Cấu hình file config mosquito.conf

Trong cấu hình của broker MQTT, dòng lệnh sau được sử dụng để thiết lập các thông số quan trọng:

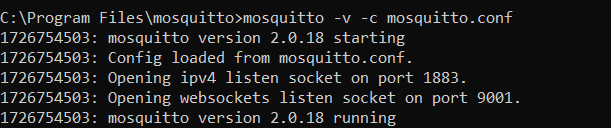
* **listener 1883 192.168.1.211**: Cấu hình một listener cho broker MQTT sử dụng giao thức TCP/IP (giao thức chuẩn của MQTT). Cổng 1883 là cổng mặc định cho MQTT qua TCP. 192.168.1.211 là địa chỉ IP mà broker sẽ lắng nghe. Chỉ những client kết nối qua địa chỉ IP này mới có thể kết nối với broker.
* **password\_file passwd**: Broker yêu cầu xác thực bằng username và password.File passwd chứa thông tin xác thực (username và password) cho các client.
* **allow\_anonymous false:** Cấu hình không cho phép kết nối ẩn danh (anonymous). Điều này có nghĩa là tất cả các client phải có thông tin đăng nhập hợp lệ (username và password) để kết nối.
* **listener 9001**: Cấu hình một listener khác cho broker nhưng lần này sử dụng giao thức WebSocket.Cổng 9001 là cổng được sử dụng để lắng nghe các kết nối MQTT qua WebSocket.
* **protocol websockets:** Cấu hình này chỉ rõ rằng cổng 9001 sẽ sử dụng giao thức WebSocket thay vì TCP thông thường.
* Với file **passwd** đã cấu hình xong username và password (mật khẩu đã được mã hóa).



Hình : File passwd chứa tài khoản với mật khẩu được mã hóa

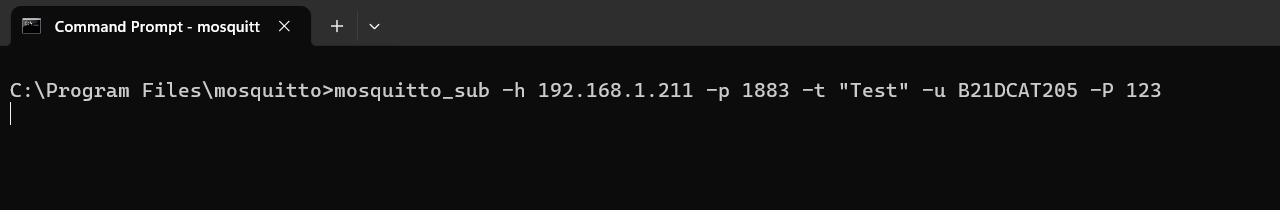
Thử nghiệm publish và subcribe trên broker đã cấu hình:

* Khởi động cấu hình broker:



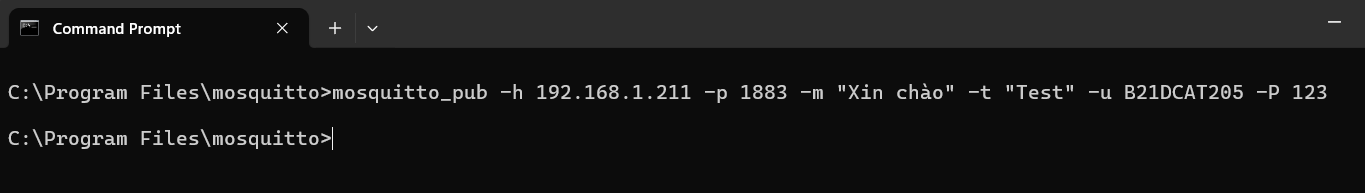
Hình : Khởi động broker

* Tiến hành thử nghiệm **publish** và **subcribe** để gửi, nhận dữ liệu

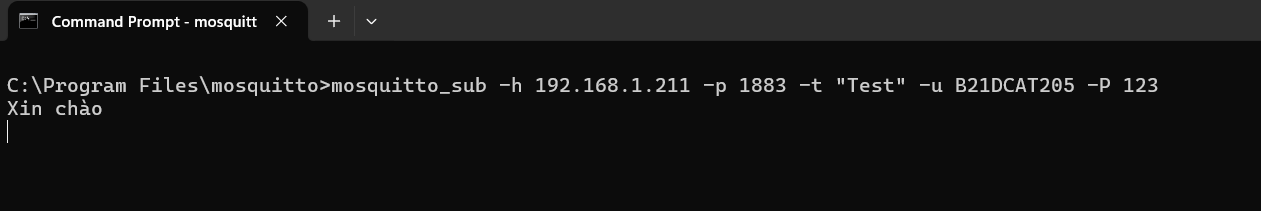


Hình : sub dữ liệu

* Nhận dữ liệu



Hình : pub dữ liệu



Hình : Kết quả

## 3.3. Backend

Trong dự án này, backend được xây dựng bằng **ExpressJS**, (phiên bản 4.19.2) làm framework để xây dựng API và xử lý các yêu cầu HTTP. Đây là một framework phổ biến, nhẹ và linh hoạt giúp dễ dàng xây dựng các ứng dụng web và API.

**Chức năng của Backend**

Backend thực hiện các nhiệm vụ chính sau:

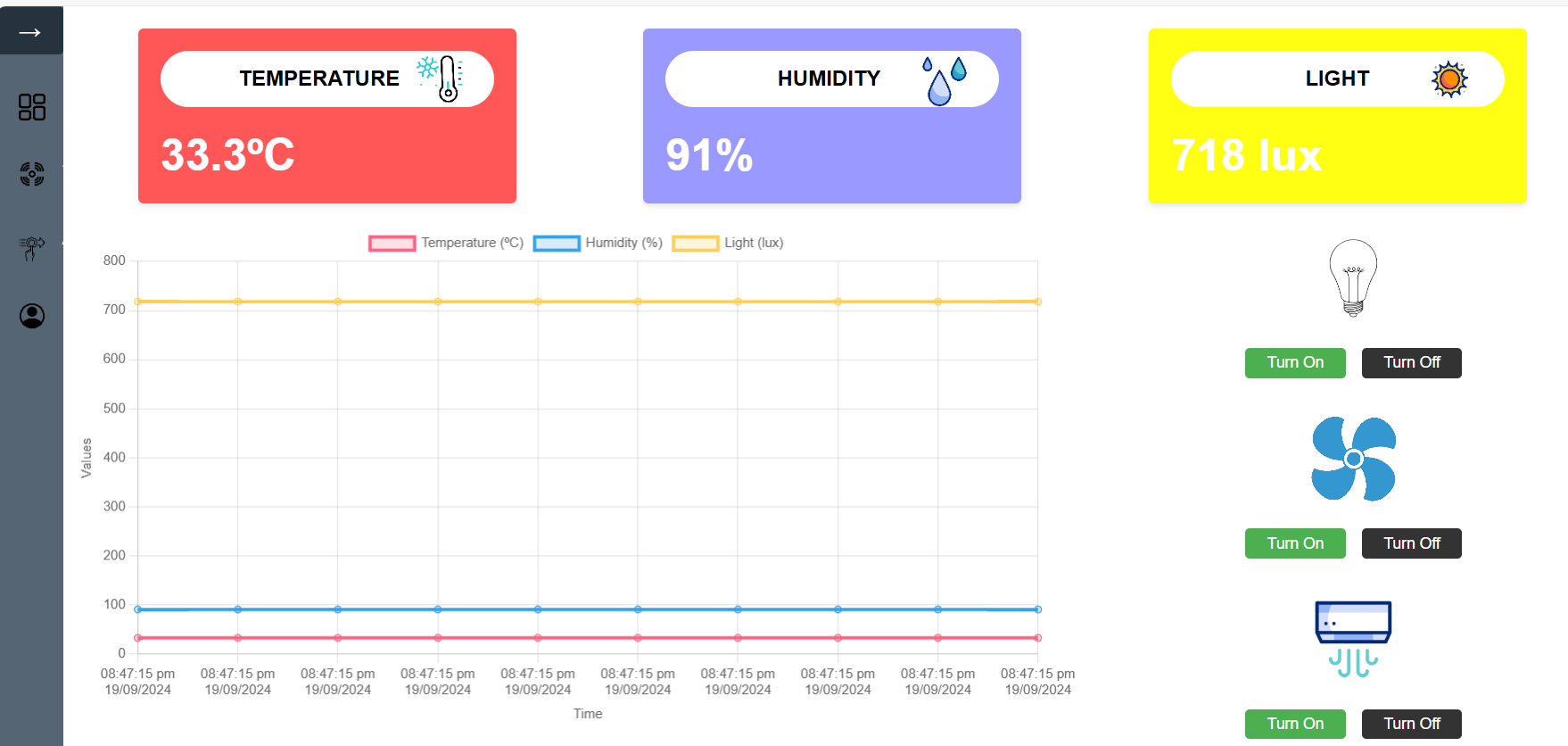
* Kết nối với MQTT Broker: Backend sử dụng MQTT để kết nối với MQTT Broker và nhận/lệnh điều khiển cho các thiết bị IoT.
* Gửi lệnh điều khiển: Server có thể gửi các lệnh MQTT tới các thiết bị IoT như đèn, quạt, điều hòa (bật/tắt thiết bị, thay đổi trạng thái).
* Nhận dữ liệu từ thiết bị: Thiết bị IoT có thể gửi dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng) về server thông qua MQTT. Server có thể xử lý hoặc lưu trữ các dữ liệu này.
* Phục hồi trạng thái thiết bị: Khi trang web được tải lại hoặc server khởi động lại, server sẽ tự động lấy lại trạng thái của các thiết bị từ cơ sở dữ liệu hoặc local storage và hiển thị trạng thái hiện tại của thiết bị.
* Lấy dữ liệu cảm biến: API có thể trả về dữ liệu cảm biến từ các thiết bị, chẳng hạn như nhiệt độ, độ ẩm, ánh sáng theo thời gian thực.
* Lưu trữ dữ liệu lịch sử: Dữ liệu về lịch sử điều khiển thiết bị (bật/tắt, trạng thái) hoặc dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng) được lưu trữ vào cơ sở dữ liệu MySQL để phân tích và hiển thị.
* Truy xuất dữ liệu lịch sử: Bạn có thể truy xuất dữ liệu lịch sử các hành động đã thực hiện với thiết bị (thời gian, trạng thái thay đổi), hoặc lấy dữ liệu cảm biến từ cơ sở dữ liệu.

## 3.4. Frontend

Front-end của hệ thống được xây dựng bằng **HTML/ CSS** đơn thuần với các chức năng điều khiển thiết bị, giám sát dữ liệu cảm biến, lịch sử hành động, và thông tin cá nhân. Giao diện này có thiết kế tương tác với người dùng, cho phép hiển thị dữ liệu theo thời gian thực và điều khiển các thiết bị thông qua các nút bật/tắt.

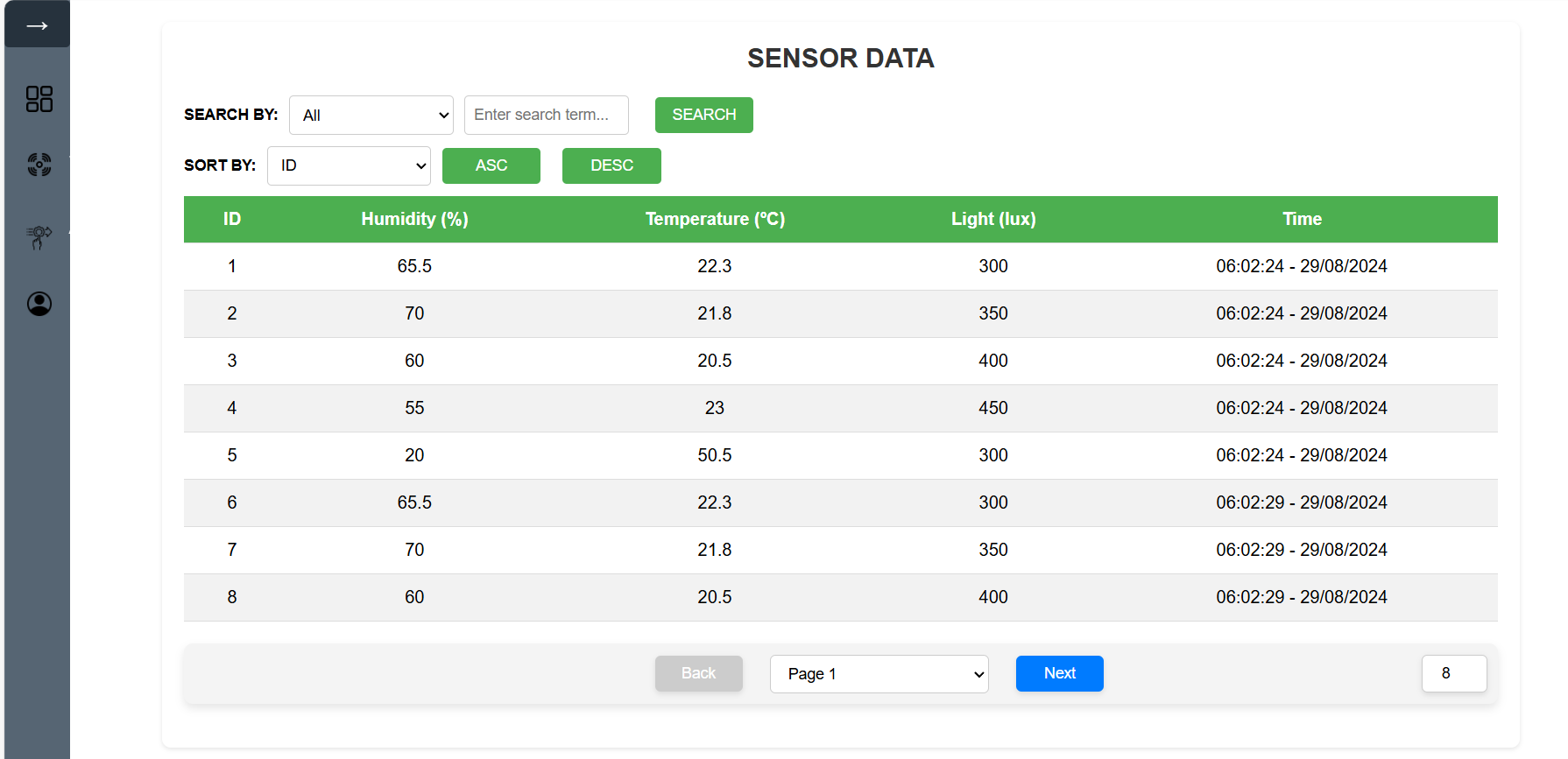
**Các thành phần chính và chức năng:**

* Sidebar (Thanh điều hướng): Đây là thanh điều hướng bên trái, giúp người dùng chuyển đổi giữa các phần khác nhau của ứng dụng như Dashboard, Sensor Data, Action History, và Profile. Mỗi mục trong sidebar có biểu tượng và tên đại diện cho chức năng của nó.
* Dashboard (Bảng điều khiển): Hiển thị dữ liệu cảm biến hiện tại và quá khứ, bao gồm nhiệt độ, độ ẩm, và ánh sáng. Dữ liệu này được cập nhật theo thời gian thực, với các biểu tượng và màu sắc đại diện cho từng loại dữ liệu (nhiệt độ – đỏ, độ ẩm – xanh, ánh sáng – vàng). Có thêm các nút điều khiển thiết bị như bật/tắt đèn, quạt, và máy điều hòa, mô phỏng quá trình tương tác với các thiết bị IoT.Biểu đồ thể hiện các giá trị biến động theo thời gian.



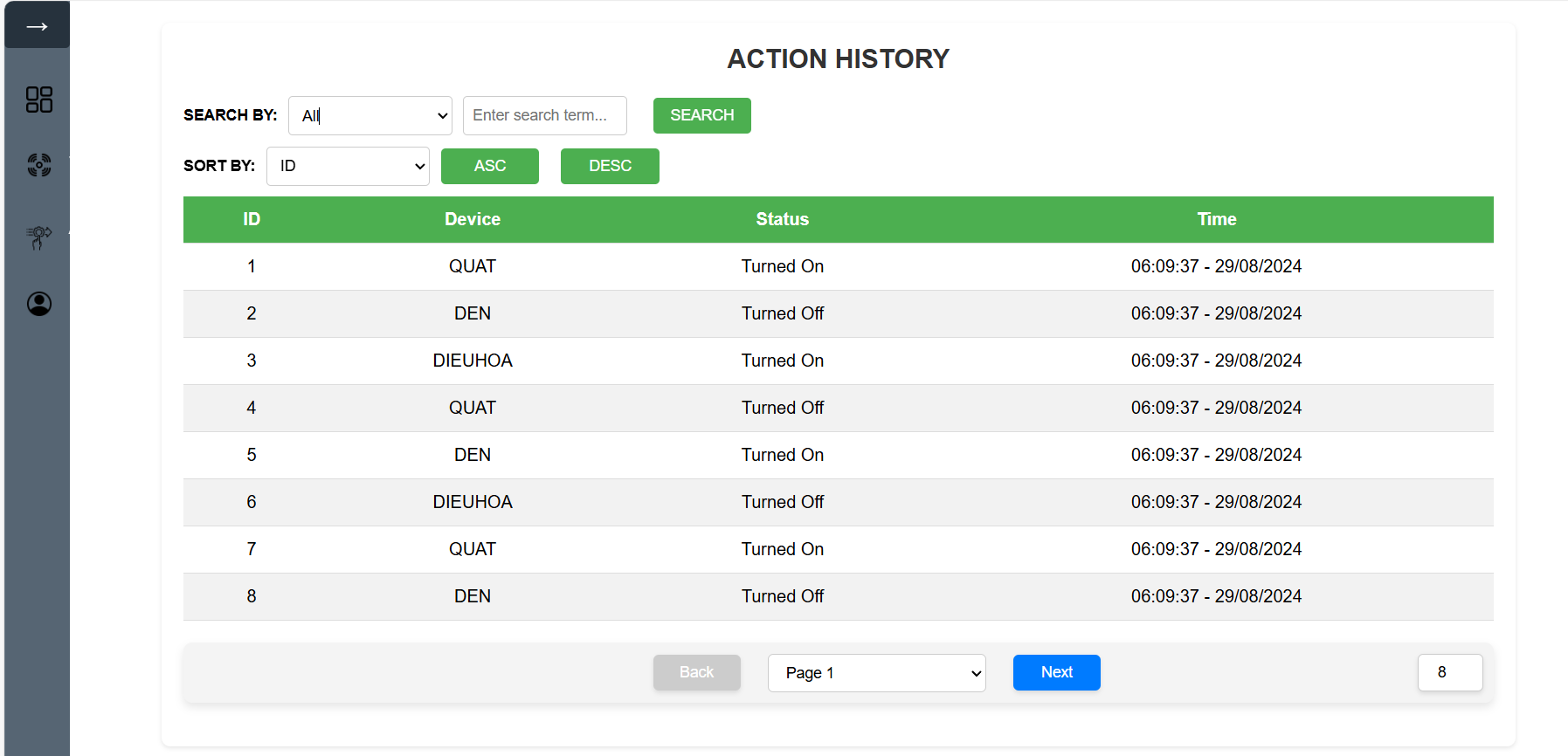
Hình : Giao diện Dashboard

* Sensor Data (Dữ liệu cảm biến): Cho phép người dùng tìm kiếm, sắp xếp, và phân trang dữ liệu lịch sử cảm biến dựa trên các tiêu chí như ID, độ ẩm, nhiệt độ, ánh sáng, và thời gian. Người dùng có thể lọc dữ liệu bằng các từ khóa và sắp xếp thứ tự tăng/giảm cho từng tiêu chí.Có hệ thống phân trang để quản lý và hiển thị dữ liệu dễ dàng hơn.



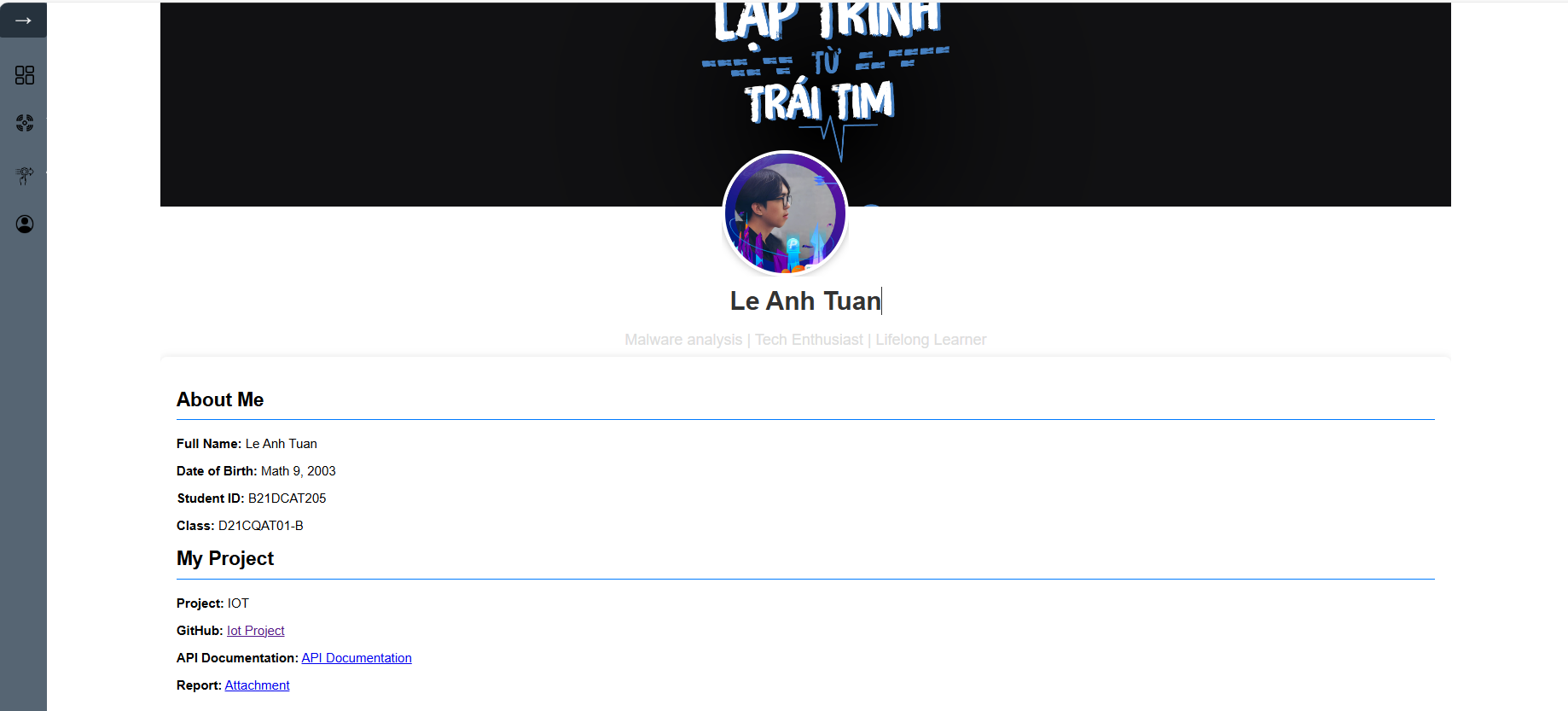
Hình : Giao diện Sensor Data

* Action History (Lịch sử hành động): Hiển thị lịch sử các hành động tương tác với thiết bị, bao gồm ID thiết bị, trạng thái (bật/tắt), và thời gian hành động.Tương tự phần Sensor Data, người dùng có thể tìm kiếm, sắp xếp và phân trang dữ liệu lịch sử hành động.



Hình : Giao diện Action History

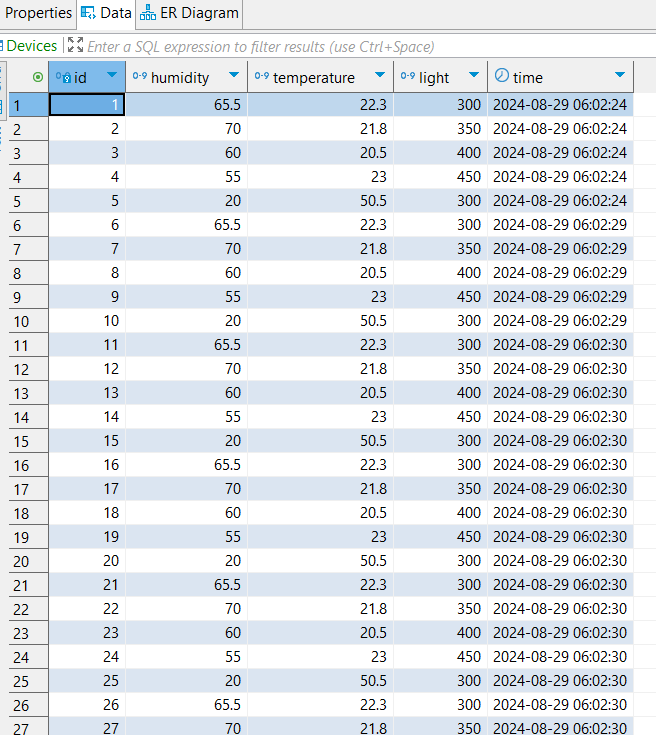
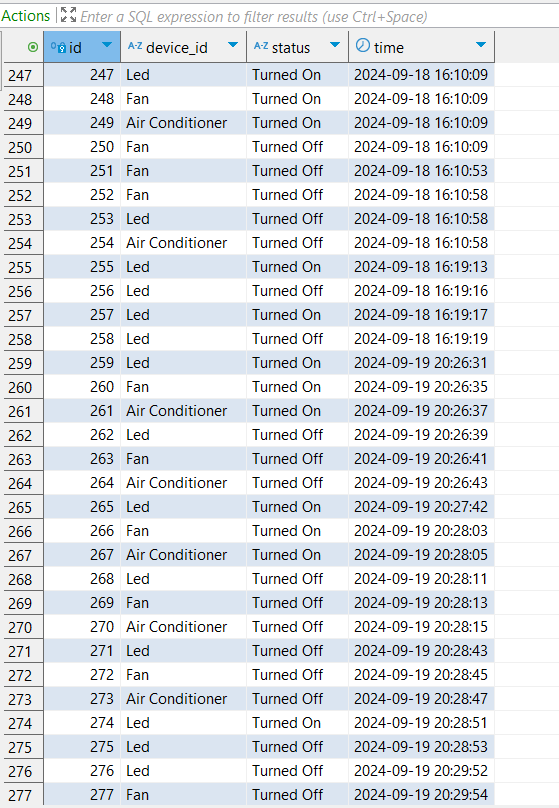
* Profile (Trang hồ sơ cá nhân): Hiển thị thông tin cá nhân của người dùng, bao gồm họ tên, ngày sinh, mã sinh viên, lớp học.Có các liên kết đến GitHub của dự án, tài liệu API, và báo cáo dự án.



Hình : Giao diện Profile

**3.5 Cơ sở dữ liệu**

Trong dự án này, em đã sử dụng **Docker** để chứa dữ liệu, với cơ sở dữ liệu được sử dụng đó là **MySQL**, trong cơ sở dữ liệu có lưu 2 bảng: bảng thứ nhất đó là bảng lưu các thông số cảm biến ánh sáng, nhiệt độ, độ ẩm, bảng còn lại lưu lịch sử hành động bật tắt thiết bị.Các thao tác cơ sở dữ liệu có thể được sử dụng trực tiếp thông qua **Dbeaver** - phần mềm hỗ trợ các hệ quản trị cơ sở dữ liệu Oracle, MySQL, IBM DB2, SQL Server, PostgreSQL, Java DB, …

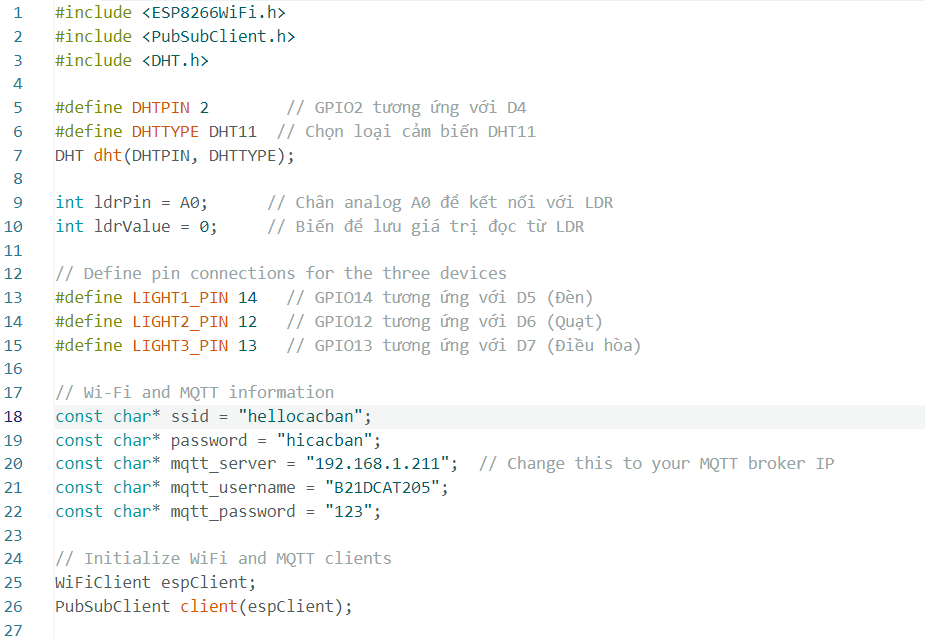
Hình : Dữ liệu 2 bảng

# Chương 4. Mã nguồn

## 4.1. Embedded Code

**4.1.1. Khai báo thư viện và định nghĩa các biến**

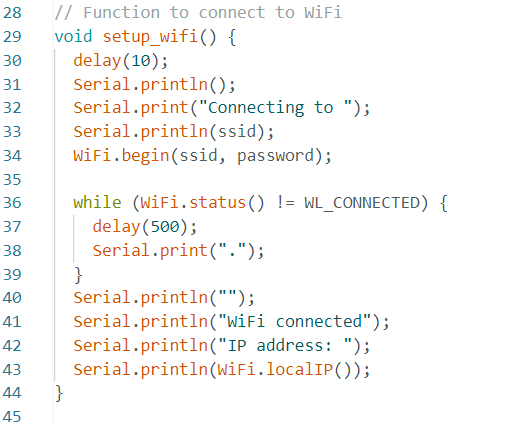
* ESP8266WiFi.h: Thư viện này dùng để kết nối module ESP8266 với các mạng Wi-Fi.
* PubSubClient.h: Thư viện này dùng để thực hiện giao thức MQTT (Message Queuing Telemetry Transport) cho việc gửi nhận tin nhắn.
* DHT.h: Thư viện này dùng để giao tiếp với cảm biến DHT, đo nhiệt độ và độ ẩm.
* Định nghĩa cảm biến và chân
* Định nghĩa chân cho các thiết bị
* Cấu hình Wi-Fi và MQTT
* Khởi tạo client Wi-Fi và MQTT



Hình : Khai báo thư viện và các biến

**4.1.2. Kết nối Wifi**

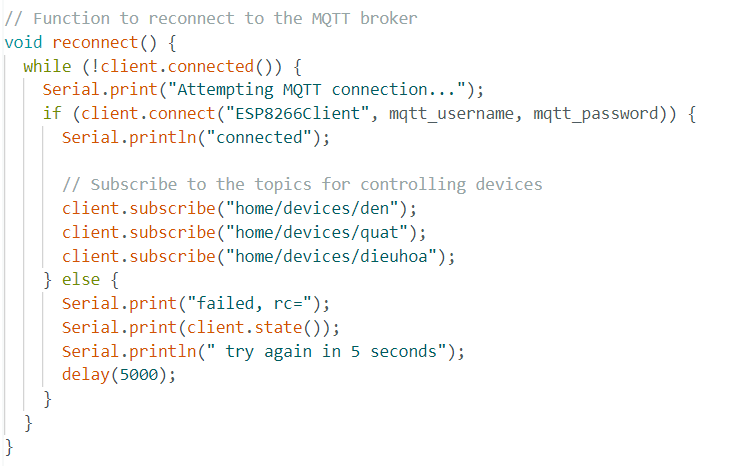
Hàm **setup\_wifi** tự động quản lý việc kết nối Wi-Fi cho ESP8266, hiển thị thông tin trạng thái kết nối và đảm bảo rằng thiết bị có thể giao tiếp với mạng Internet trước khi thực hiện các chức năng khác.



Hình : Hàm kết nối Wifi

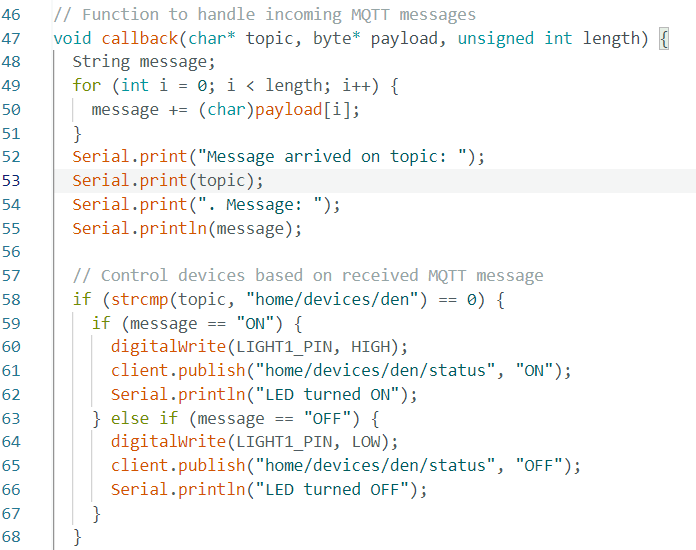
**4.1.3. Kết nối MQTT**

Hàm **reconnect** có nhiệm vụ thiết lập lại kết nối giữa ESP8266 và broker MQTT khi mất kết nối. Khi kết nối thành công, hàm sẽ đăng ký cho client để nhận tin nhắn từ các chủ đề cụ thể liên quan đến các thiết bị (đèn, quạt, điều hòa). Điều này cho phép ESP8266 có thể nhận và xử lý các lệnh từ broker MQTT.



Hình : Hàm kết nối với MQTT Broker

Hàm callback là nơi xử lý các tin nhắn MQTT đến từ broker. Nó đọc nội dung và chủ đề của tin nhắn, in ra thông tin cho người dùng, và điều khiển các thiết bị (đèn, quạt, điều hòa) dựa trên lệnh nhận được. Hàm này rất quan trọng trong việc thực hiện các lệnh từ xa trong các ứng dụng IoT

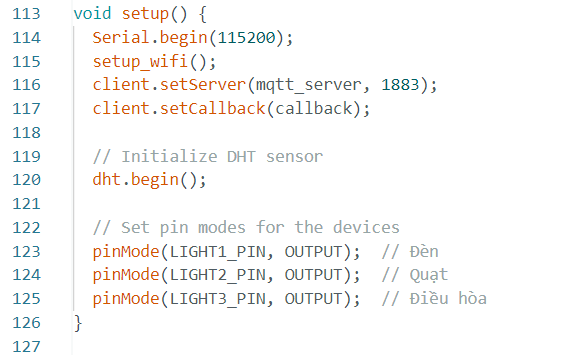
 

Hình : Hàm callback()

**4.1.4. Hàm setup() và loop()**

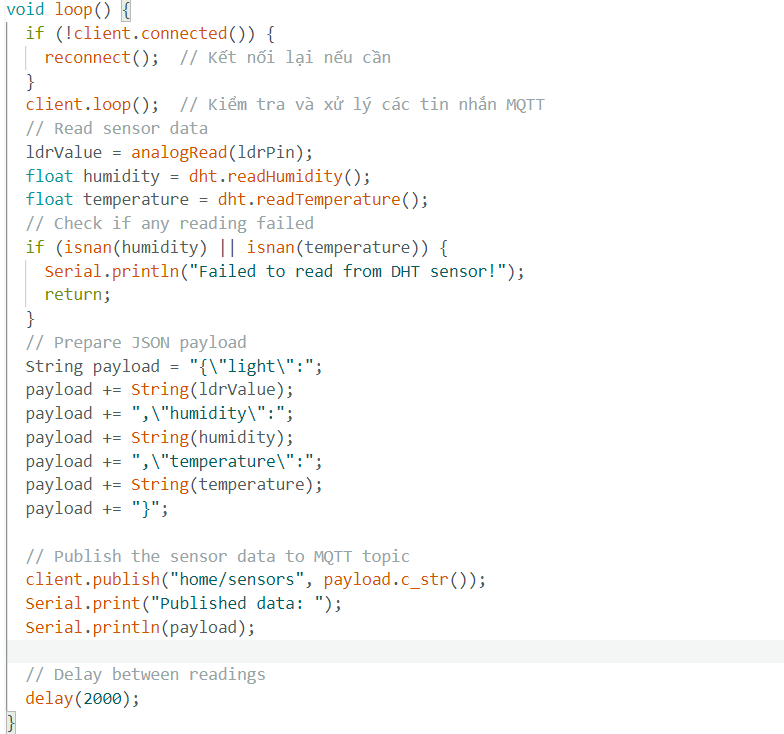
Hàm setup() và loop() là hai hàm chính trong chương trình Arduino, được sử dụng để khởi tạo và điều khiển các hoạt động của vi điều khiển ESP8266.

**Hàm setup():**Hàm setup thực hiện các bước khởi tạo cần thiết để ESP8266 có thể hoạt động trong môi trường IoT. Nó thiết lập giao tiếp Serial, kết nối Wi-Fi, cấu hình MQTT, khởi động cảm biến, và định nghĩa chế độ cho các chân GPIO. Tất cả các bước này đảm bảo rằng ESP8266 sẵn sàng để gửi và nhận dữ liệu, cũng như điều khiển các thiết bị khi nhận lệnh từ broker MQTT.

****

Hình : hàm setup()

**Hàm loop():** Hàm loop là phần chính của chương trình, thực hiện các nhiệm vụ liên tục như kiểm tra và duy trì kết nối MQTT, đọc dữ liệu từ cảm biến bao gồm ánh sáng, độ ẩm và nhiệt độ, sau đó gửi dữ liệu đến broker dưới dạng json và quản lý thời gian giữa các lần đọc là 2s. Hàm này rất quan trọng đảm bảo thiết bị luôn hoạt động và truyền tải thông tin chính xác.

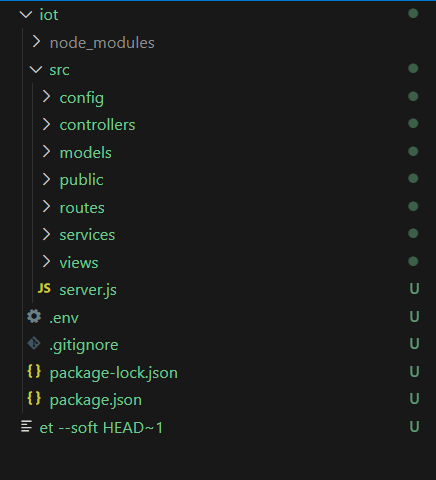


Hình : Hàm loop

## 4.2. Mã Nguồn

**4.2.1. Tổng quan về cấu trúc code**

Dưới đây là cấu trúc chính của backend:



Hình : Cấu trúc thư mục backend

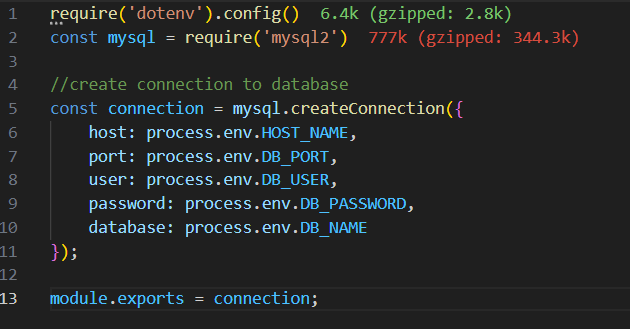
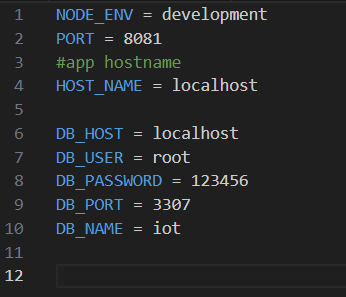
Phần Backend được tổ chức thành các thư mục chính:

1. **Config**

****

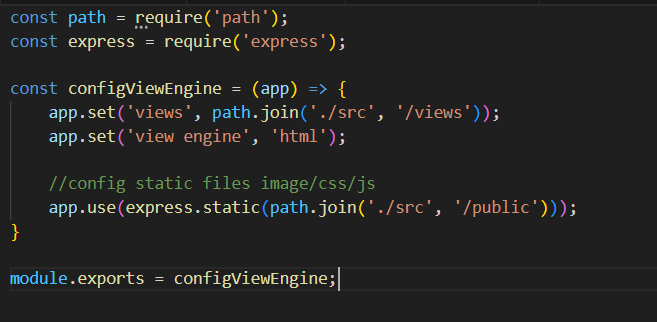
Hình : thư mục config

* **Chức năng database.js:** Tệp này nói về việc thiết lập kết nối giữa ứng dụng Node.js và cơ sở dữ liệu MySQL. Nó sử dụng thông tin nhạy cảm như địa chỉ host, cổng, tên người dùng, mật khẩu và tên cơ sở dữ liệu, được lưu trữ an toàn trong một tệp **.env**

Hình : file database.js (bên trái) và file .env(bên phải)

* **Chức năng viewEngine.js:** Thiết lập thư mục views để chứa các tệp giao diện (HTML) trong ./src/views. Chọn HTML làm view engine để render giao diện.Cấu hình thư mục tệp tĩnh (hình ảnh, CSS, JS) tại ./src/public để phục vụ trực tiếp cho người dùng.



Hình : viewEngine.js

**b. Models:**

* **Chức năng:** Tệp này định nghĩa các hàm để tương tác với cơ sở dữ liệu, bao gồm việc truy xuất dữ liệu và lưu trữ thông tin liên quan đến cảm biến và trạng thái hành động của thiết bị. Tất cả các truy vấn cơ sở dữ liệu được thực hiện thông qua kết nối đã được cấu hình trong tệp ../config/database.

**getAllData(callback):**

* Mục đích: Lấy tất cả dữ liệu từ bảng Devices (thường chứa thông tin cảm biến như nhiệt độ, độ ẩm, ánh sáng).
* Chức năng: Thực hiện câu truy vấn SELECT \* FROM Devices để lấy toàn bộ dữ liệu từ bảng này và trả kết quả về qua callback.

**getAllAction(callback):**

* Mục đích: Lấy toàn bộ dữ liệu từ bảng Actions (có thể chứa lịch sử các hành động được thực hiện trên các thiết bị).
* Chức năng: Thực hiện câu truy vấn SELECT \* FROM Actions để lấy dữ liệu từ bảng này và trả về qua callback.

**insertSensorData(temperature, humidity, light, time, callback):**

* Mục đích: Lưu dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng, và thời gian) vào bảng Devices.
* Chức năng: Thực hiện câu truy vấn INSERT INTO Devices để chèn một bản ghi mới với các giá trị nhiệt độ, độ ẩm, ánh sáng và thời gian vào bảng Devices.

**insertActionHistory(device\_id, status, time, callback):**

* Mục đích: Lưu trạng thái hành động của thiết bị (mã thiết bị, trạng thái, và thời gian) vào bảng Actions.
* Chức năng: Thực hiện câu truy vấn INSERT INTO Actions để chèn một bản ghi mới với các giá trị device\_id, status và thời gian vào bảng Actions.

**c. Controllers**

* **Chức năng:** Tệp này định nghĩa các **controller functions** trong Node.js để xử lý các yêu cầu HTTP và tương tác với **data model** (cơ sở dữ liệu). Nó gồm các hàm lấy dữ liệu từ cơ sở dữ liệu, lưu dữ liệu cảm biến, lưu lịch sử hành động của thiết bị, và render trang web.

**getData(req, res):**

* Mục đích: Lấy tất cả dữ liệu từ bảng Devices (thông tin cảm biến) để hiển thị trên biểu đồ.
* Chức năng: Gọi hàm getAllData từ dataModel, lấy dữ liệu và trả về dưới dạng JSON. Nếu có lỗi, trả về mã lỗi 500 kèm thông báo lỗi.

**getAction(req, res)**

* Mục đích: Lấy dữ liệu từ bảng Actions (lịch sử hành động của thiết bị).
* Chức năng: Gọi hàm getAllAction từ dataModel, trả về dữ liệu dưới dạng JSON. Nếu có lỗi, trả về mã lỗi 500 kèm thông báo lỗi.

**saveSensorData(temperature, humidity, light, callback):**

* Mục đích: Lưu dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng) vào bảng Devices.
* Chức năng: Kiểm tra tính hợp lệ của dữ liệu, sau đó gọi hàm insertSensorData từ dataModel để lưu dữ liệu vào cơ sở dữ liệu. Nếu dữ liệu không hợp lệ, trả về mã lỗi 400 với thông báo lỗi.

**saveDeviceAction(device\_id, status, callback):**

* Mục đích: Lưu lịch sử hành động của thiết bị (mã thiết bị và trạng thái) vào bảng Actions.
* Chức năng: Kiểm tra tính hợp lệ của dữ liệu, sau đó gọi hàm insertActionHistory từ dataModel để lưu vào cơ sở dữ liệu. Nếu dữ liệu không hợp lệ, trả về mã lỗi 400 với thông báo lỗi.

**getWeb(req, res):**

* Mục đích: Render trang web với template app.ejs.
* Chức năng: Gọi hàm res.render('app.ejs') để hiển thị giao diện người dùng dựa trên tệp app.ejs.

**d. Thư mục Public:**

* **Chức năng:** Thư mục này chứa 3 thư mục con bao gồm css,js,img. Css là thư mục xử lý cho việc css cho dashboard, Sensor-data, action-history và profile và naviigator. Img là file ảnh chứa các ảnh cho toàn bộ project này, js là nơi xử lý logic cho cho dashboard, Sensor-data, action-history và profile cũng như thanh điều hướng.
* **Các tệp tin chính:**
* **action\_history.js** : Đoạn mã cung cấp tính năng tìm kiếm, sắp xếp, phân trang và giới hạn số lượng bản ghi hiển thị cho bảng lịch sử hành động thiết bị, giúp người dùng dễ dàng tương tác với dữ liệu trực tiếp trên giao diện web.
* **chart.js**: Đoạn mã này tạo ra một biểu đồ thời gian thực hiển thị dữ liệu nhiệt độ, độ ẩm và ánh sáng, với khả năng tự động cập nhật dữ liệu và giới hạn số điểm hiển thị, nhằm mang đến trải nghiệm trực quan và tương tác cho người dùng.
* **devices.js**: File này kết nối mqtt tới cổng 9001 mở websockets , thực hiện chức năng pub dữ liệu khi mà thiết bị thay đổi trạng thái từ web theo thời gian thực.
* **sensor\_data.js:** Đoạn mã này cung cấp một giao diện người dùng để tìm kiếm, sắp xếp và phân trang dữ liệu cảm biến một cách hiệu quả, giúp người dùng dễ dàng quản lý và theo dõi thông tin cần thiết.
* **sidebar.js:** Đoạn mã này thực hiện các chức năng liên quan đến việc quản lý hiển thị các phần của giao diện người dùng và tương tác với thanh sidebar.

**e. routes**

* **Chức năng chính:** quản lý và tổ chức các tuyến đường của ứng dụng, giúp ứng dụng dễ dàng mở rộng và bảo trì.
* **Các tệp chính;**
* **swagger.js:** tài liệu hóa các API, giúp người dùng hiểu rõ hơn về cách sử dụng và dữ liệu yêu cầu**.**
* **web.js:** quản lý các định tuyến API và trang web một cách có tổ chức, giúp dễ dàng mở rộng và bảo trì ứng dụng.

**f. Services**

* **Chức năng:** Thư mục chứa file kết nối mqtt như mqttClient.js và mqttServer.ino
* **Các tệp chính:**
* **mqttClient.js**: Mã này thiết lập một kết nối MQTT, đăng ký theo dõi các chủ đề liên quan đến cảm biến và thiết bị, và lưu trữ thông tin nhận được từ các tin nhắn MQTT vào cơ sở dữ liệu thông qua các hàm điều khiển tương ứng.

**g. Views:** Thư mục chứa file app.ejs tạo ra một giao diện người dùng tương tác với trình duyệt, cho phép người dùng theo dõi các dữ liệu cảm biến, quản lý thiết bị, xem lịch sử hành động và truy cập thông tin cá nhân.

**h. File Server.js:** thiết lập một ứng dụng web với API tài liệu, hỗ trợ phục vụ các route và tài nguyên tĩnh.

**4.2.2. Luồng hoạt động:**

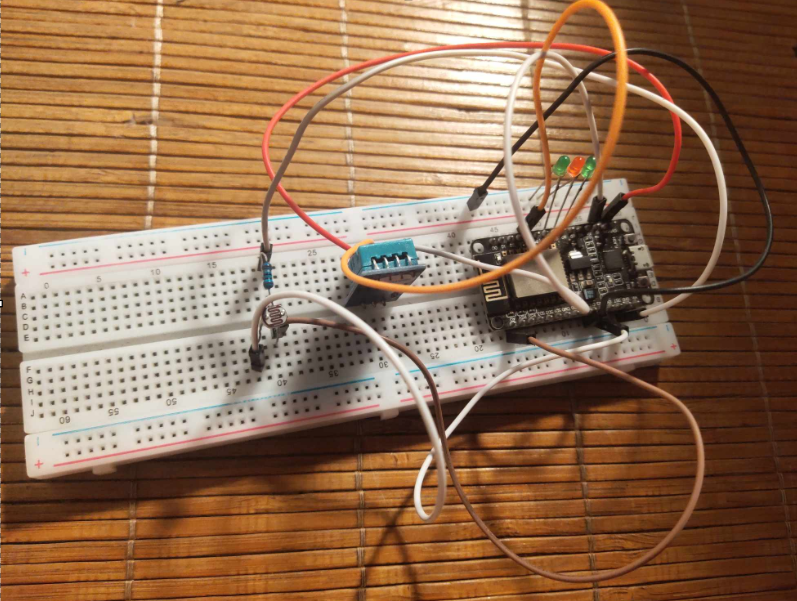
* Sử dụng câu lệnh npm start để khởi động chương trình.
* Giao diện web hiển thị với trang dashboard ban đầu, thanh điều hướng, các trạng thái đèn, điều hòa và quạt ở trạng thái off.Đối với các thông số về ánh sáng, nhiệt độ, độ ẩm sẽ được pub từ code embed sang mosquitto, mosquitto chuyển thông tin tới backend, backend sẽ lấy được dữ liệu mới nhất, front end call api từ backend lên và hiển thị dữ liệu hiện tại.Dữ liệu sẽ được hiển thị trong cả biểu đồ trực quan, các thanh màu trong dashboard thể hiện màu sắc tương đương với giá trị hiện tại của ánh sáng, nhiệt độ và độ ẩm.Ngoài ra frondend cũng gọi api để lấy danh sách các trạng thái bật tắt thiết bị.
* Khi thực hiện bật tắt thiết bị, thông tin hành động sẽ được pub vào mosquitto, mosquitto chuyển tiếp tới code embed xử lý phần cứng thực hiện bật tắt tương ứng
* Sau khi bật tắt xong, tại code embed sẽ thực hiện gửi pub tới backend và backend sẽ thực hiện lưu thông tin trạng thái thiết bị vào cơ sở dữ liệu (đối với các trạng thái sensor lưu tương tự chỉ có điều là được lưu liên tục)

# Chương 5. Kết quả thực nghiệm

## 5.1. 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.

**Tổng quan về phần cứng:**

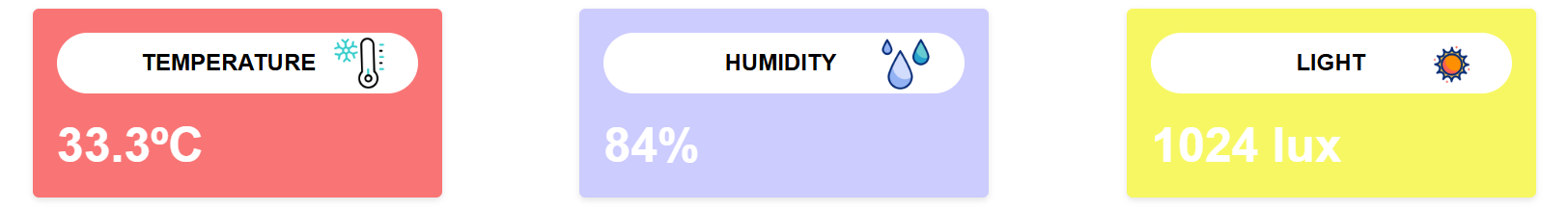


Hình : Toàn bộ hệ thống phần cứng

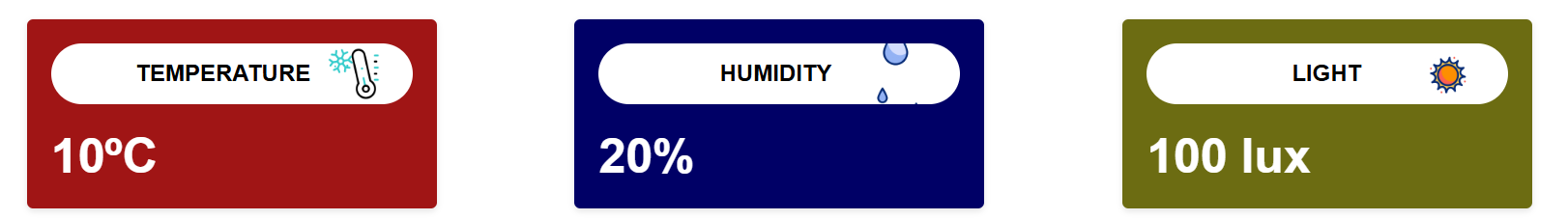
## 5.2. Đo lường các chỉ số như nhiệt độ, độ ẩm, ánh sáng thời gian thực

Em đã hoàn thành việc đo lường và thu thập các chỉ số môi trường như nhiệt độ, độ ẩm, và ánh sáng trong thời gian thực.

* **Với nhiệt độ, độ ẩm:** Dùng cảm biến DHT11 để đo
* **Với ánh sáng:** Dùng quang trở để đo



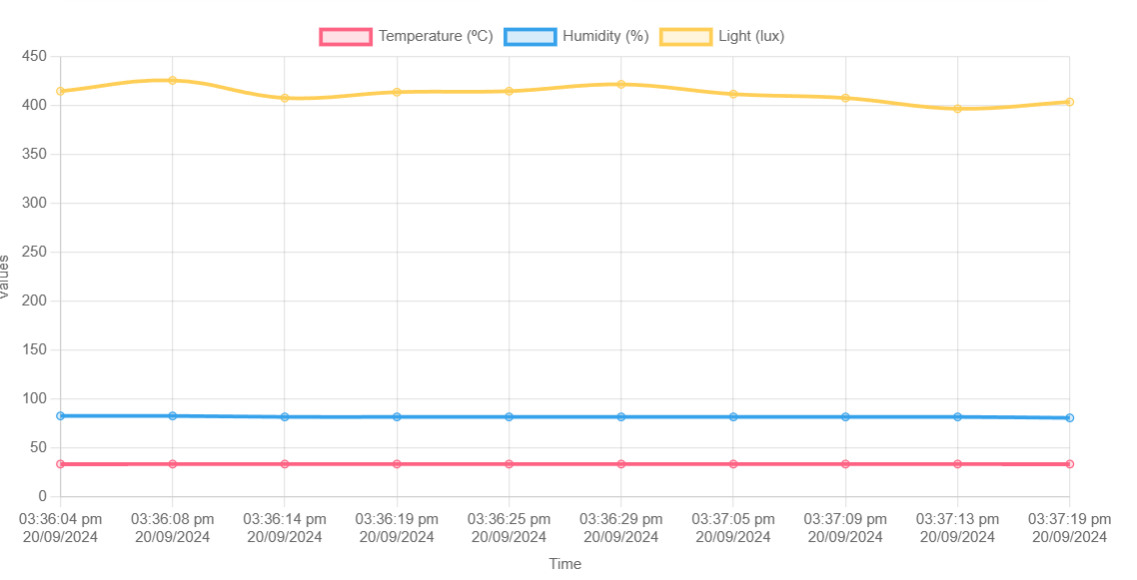
Hình : Hiển thị các thông số khi ở mức cao



Hình : Hiển thị các thông số khi ở mức thấp

Tông màu chủ đạo của nhiệt độ là màu đỏ, độ ẩm là màu xanh, ánh sáng là màu vàng, khi các thông số tăng thì giá trị màu cũng sẽ nhạt dần hơn và sáng hơn và ngược lại khi giảm, màu sẽ đậm hơn và tối hơn.

Ngoài ra, hệ thống cũng cung cấp đồ thị thời gian thực, phản ánh sự biến đổi của các giá trị này theo từng thời điểm, giúp người dùng dễ dàng quan sát và phân tích dữ liệu.

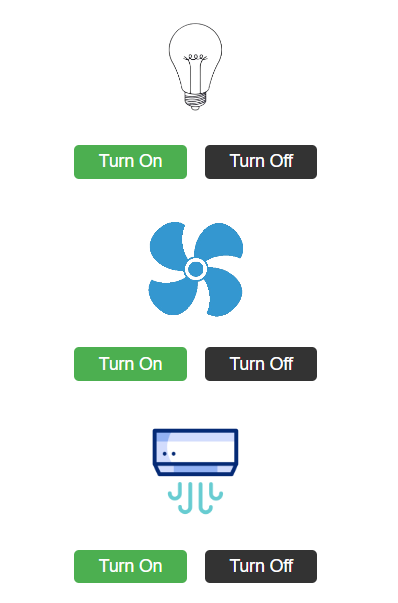
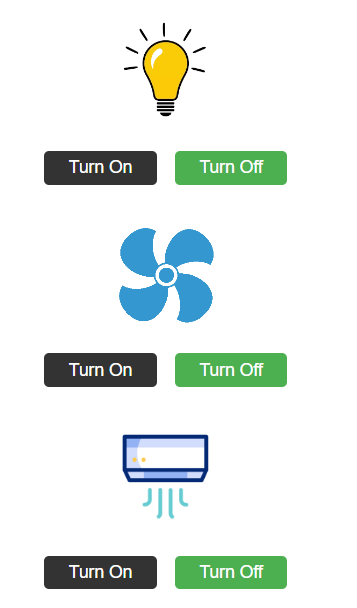


Hình : Đồ thị phản ánh sự thay đổi các chỉ số

## 5.3. Điều khiển các thiết bị điện

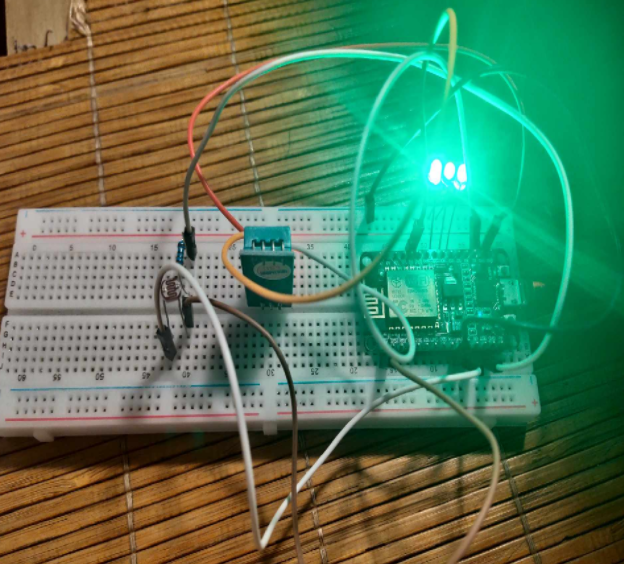
Khi tắt các thiết bị:

Đối với các thiết bị như quạt, điều hòa và đèn sẽ có 2 nút, một nút tắt và bật.Khi bấm vào nút nào thì tại nút đó có màu đen,chữ màu trắng và bên còn lại sẽ là nền xanh và chữ trắng.

Hình : Tắt (bên trái) và bật (bên phải) các thiết bị

Hệ thống khi bật các thiết bị:

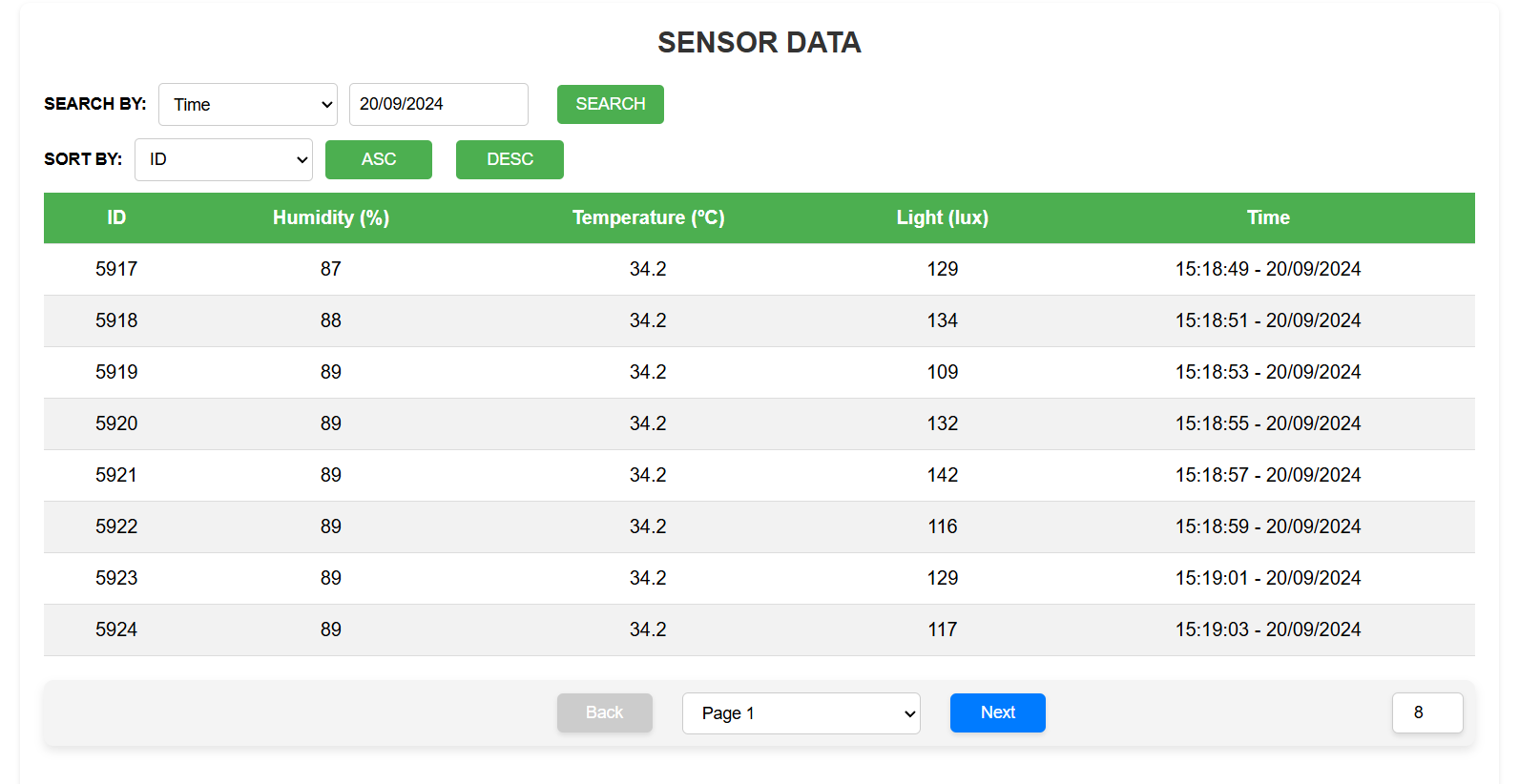


Hình :Các thiết bị phần cứng(led) sáng

## 5.4. Lưu trữ và đưa ra dữ liệu cho người dùng

**5.4.1. Data Sensor**

Ta thấy rằng các thông số sẽ liên tục được cập nhật trong sensor data

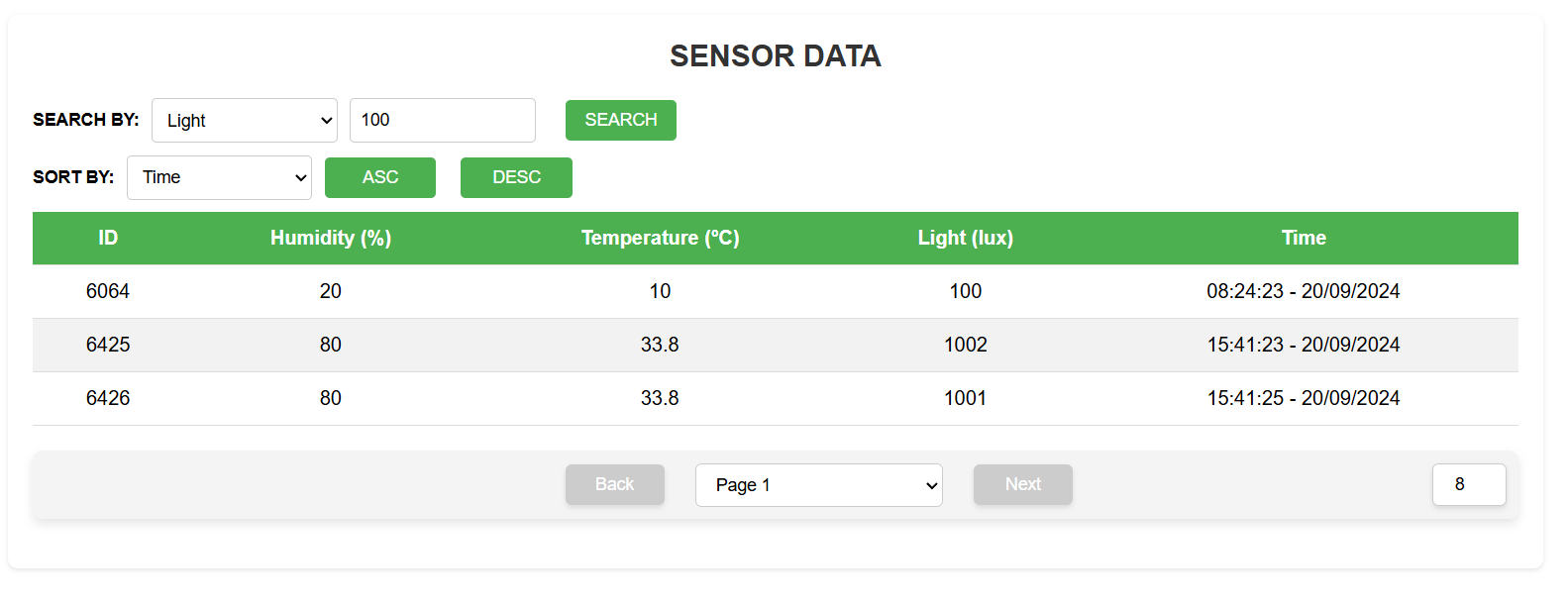


Hình : Trang Data Sensor

Ngoài ra còn có các chức năng khác như:

* Phân trang
* Tìm kiếm thông số theo loại chỉ số
* Tìm kiếm trong khoảng thời gian

Dưới đây là ví dụ minh họa cho tìm kiếm thông số với giá trị 100 cho chỉ số Ánh sáng page size là 8 và sắp xếp tăng dần thời gian

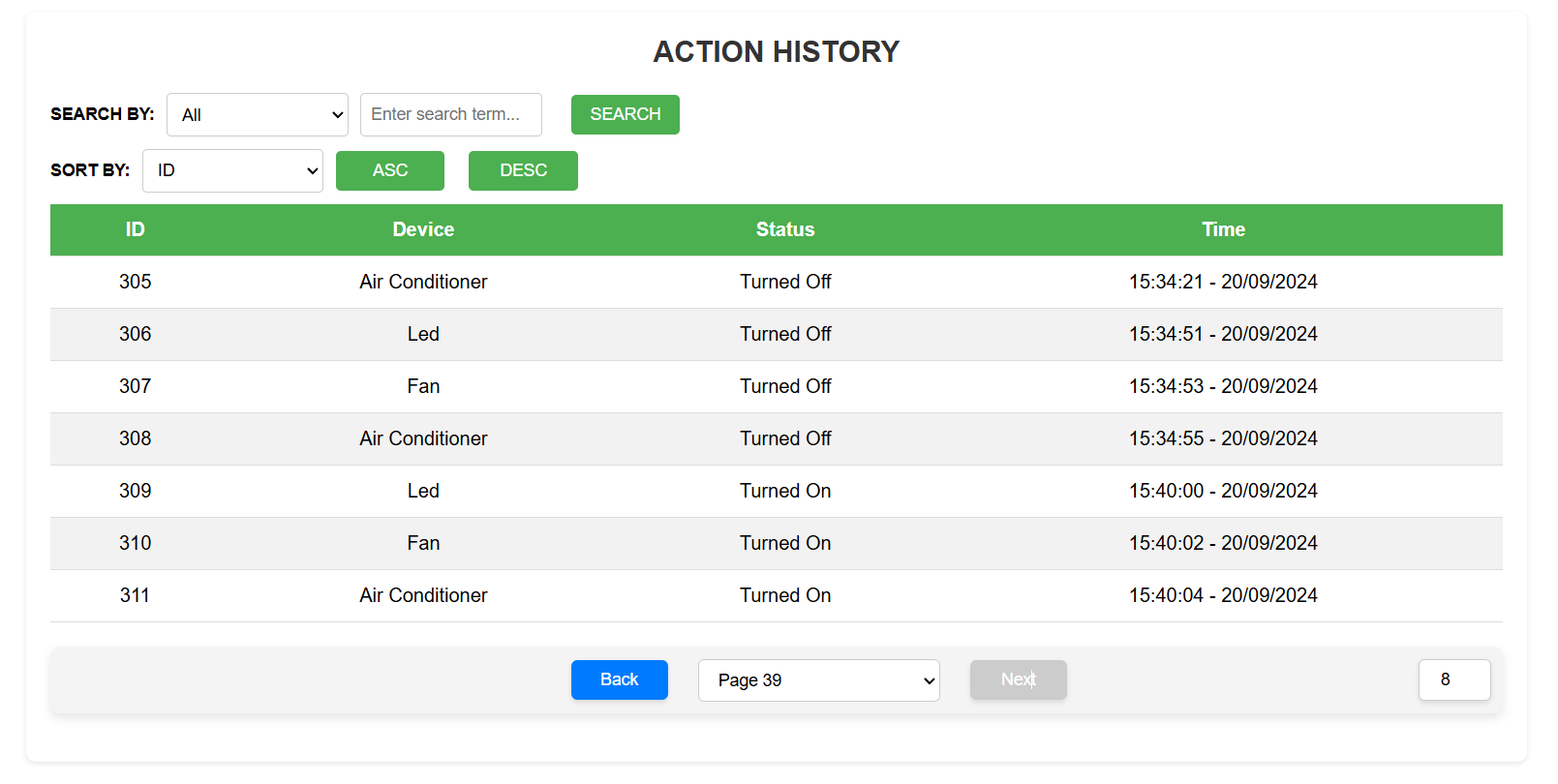


Hình : Ảnh minh họa ví dụ

**5.4.2. Action History**

Trang này lưu trữ và hiển thị thông tin về lịch sử các hành động bật/tắt thiết bị từ trước đến nay. Người dùng có thể kiểm tra các thao tác điều khiển thiết bị đã thực hiện, đảm bảo rằng mọi thay đổi trạng thái của các thiết bị đều được ghi lại và có thể kiểm tra lại khi cần thiết.

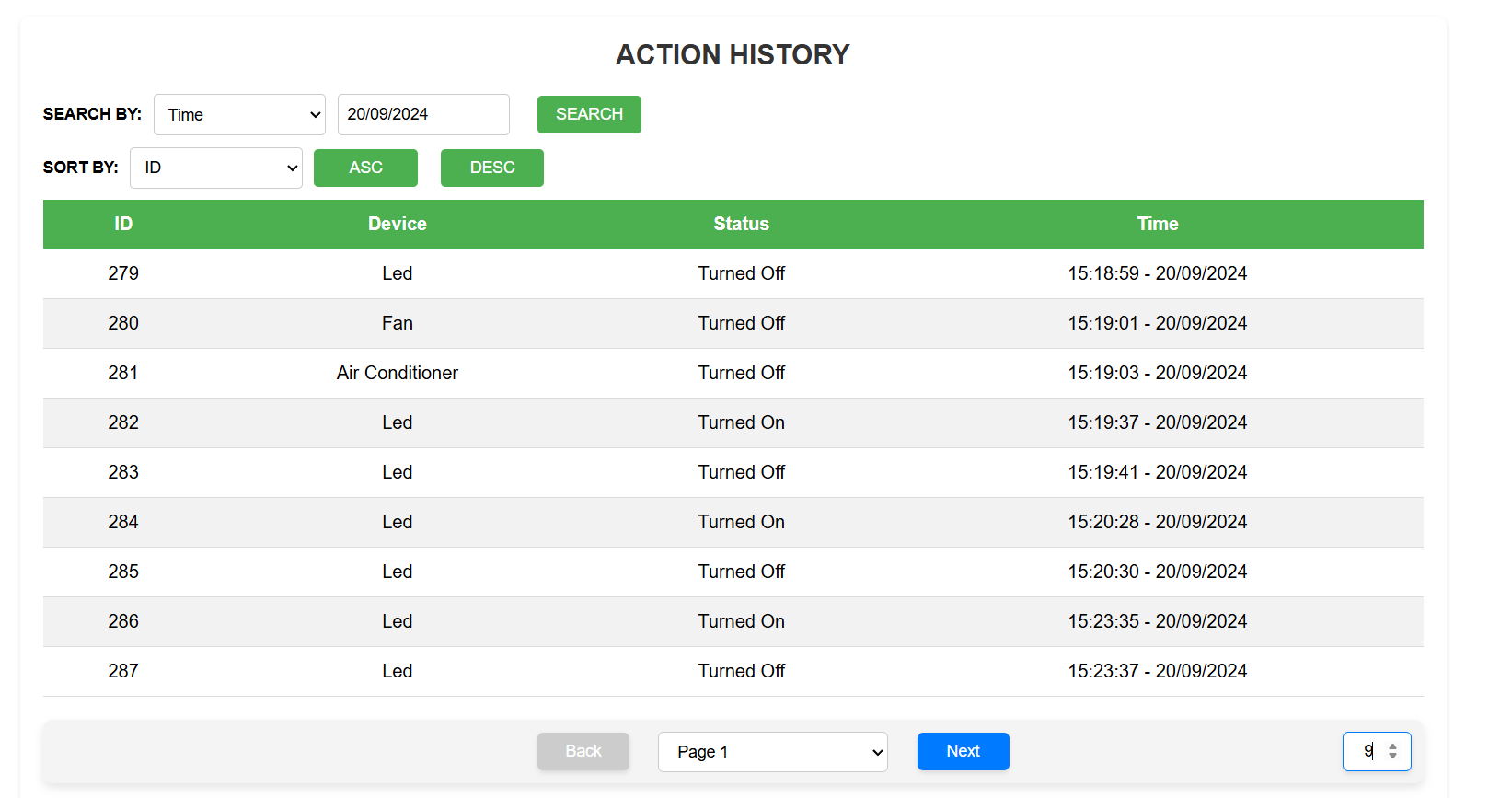
Tương tự như sensor data thì action history cũng cập nhật liên tục các hành động bật tắt thiết bị như quạt, điều hòa và đèn



Hình : Trang Action History

Ngoài ra, còn có chức năng tìm kiếm theo khoảng thời gian và lọc theo tên thiết bị và phân trang. Dưới đây là ví dụ:

Dưới đây là một ví dụ về việc sort tăng dần theo id, tìm kiếm theo thời gian ngày 20/09/2024



Hình : Ví dụ tìm kiếm và lọc của Action History

# Tài liệu tham khảo

* [1] [MQTT là gì? Vai trò của MQTT trong IoT (viblo.asia)](https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot-V3m5WL3bKO7)
* [2] [Mosquitto MQTT Broker on Windows Installation - Ultimate Guide | Cedalo](https://cedalo.com/blog/how-to-install-mosquitto-mqtt-broker-on-windows/)