

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CƠ BẢN I



**BÁO CÁO BÀI TẬP LỚN:
LẬP TRÌNH IOT VÀ ỨNG DỤNG**

**Giảng viên hướng dẫn
Họ và tên sinh viên
Mã sinh viên
Lớp
Nhóm**

**Nguyễn Quốc Uy
Nguyễn Văn Đức
B22DCCN236
D22CQH TT05
16**

Hà Nội – 2025

Mục Lục

Chương 1: Giới thiệu	3
I. Tổng quan dự	3
1. Mục đích của dự án IoT	3
2. Các tính năng chính của hệ thống	3
II. Công nghệ sử dụng	4
Chương 2: Giao diện	5
1. Trang DashBoard	6
2. Trang DataSensor	6
3. Trang Action History	8
4. Trang Profile	9
Chương 3: Thiết kế tổng quan và chi tiết	9
I. Kiến trúc tổng quan	9
II. Backend	11
III. Kết nối MQTT	15
IV. Mô hình dữ liệu	16
V. Lập trình ESP32	17
Chương 4: Kết quả	19
I. Chức năng đã hoàn thành	19
II. Hiệu suất hệ thống	20
III. Cải tiến trong tương lai	21

Chương 1: Giới thiệu

I. Tổng quan dự

1. Mục đích của dự án IoT

- Dự án IoT được phát triển nhằm mục đích tạo ra một hệ thống giám sát và điều khiển thông minh cho môi trường trong nhà. Hệ thống này kết hợp công nghệ Internet of Things (IoT) với giao diện người dùng trực quan, cho phép người dùng theo dõi các thông số môi trường và điều khiển các thiết bị từ xa một cách dễ dàng và hiệu quả.

Các mục tiêu chính của dự án bao gồm:

- Giám sát môi trường: Thu thập và hiển thị dữ liệu về nhiệt độ, độ ẩm và ánh sáng trong thời gian thực.
- Điều khiển thiết bị: Cung cấp khả năng điều khiển từ xa các thiết bị như đèn LED, quạt và điều hòa không khí.
- Phân tích dữ liệu: Lưu trữ và phân tích dữ liệu lịch sử để đưa ra các xu hướng và thông tin chi tiết về môi trường.
- Giao diện thân thiện: Cung cấp một dashboard trực quan và dễ sử dụng cho người dùng.

2. Các tính năng chính của hệ thống

- Hiển thị dữ liệu thời gian thực:
 - + Biểu đồ động hiển thị nhiệt độ, độ ẩm và cường độ ánh sáng.
 - + Cập nhật liên tục từ các cảm biến được kết nối.
- Điều khiển bật tắt các thiết bị từ xa
- Bảng điều khiển tương tác:
 - + Giao diện trực quan cho phép người dùng dễ dàng xem và điều khiển thiết bị.
 - + Hiển thị trạng thái hiện tại của tất cả các thiết bị được kết nối.
- Lưu trữ và truy xuất dữ liệu lịch sử
 - + Lưu trữ tất cả dữ liệu cảm biến và hành động điều khiển
 - + Cung cấp khả năng xem và phân tích dữ liệu lịch sử

- API Restful

- + Cung cấp API cho phép tích hợp với các hệ thống và ứng dụng khác.

- + Hỗ trợ truy vấn dữ liệu và điều khiển thiết bị thông qua các yêu cầu HTTP.2

- Khả năng mở rộng

- + Thiết kế module cho phép dễ dàng thêm các loại cảm biến và thiết bị mới

- + Hỗ trợ nhiều phòng và khu vực khác nhau trong một hệ thống Dự án IoT này không chỉ cung cấp một giải pháp toàn diện cho việc giám sát và điều khiển môi trường trong nhà, mà còn đặt nền tảng cho việc phát triển và mở rộng trong tương lai, hướng tới một hệ sinh thái nhà thông minh hoàn chỉnh.

II. Công nghệ sử dụng

1.Backend (Node.js,Express.js)

- Node.js: Nền tảng chạy JavaScript phía máy chủ, được sử dụng để xây dựng server.

- Express.js: Framework web cho Node.js, giúp xây dựng API RESTful một cách nhanh chóng và hiệu quả.

2.Frontend(React.js)

- React.js: Thư viện JavaScript để xây dựng giao diện người dùng, giúp tạo ra các ứng dụng web động và hiệu quả.

- React Router: Thư viện định tuyến cho React, giúp xây dựng ứng dụng một trang (SPA) với nhiều route.

- Chart.js: Thư viện JavaScript để vẽ biểu đồ, được sử dụng để hiển thị dữ liệu cảm biến dưới dạng đồ thị.

- Vite: Là công cụ phục vụ phát triển và build, giúp khởi động tức thì, thay đổi nóng (HMR) cực nhanh, và tạo bản build production tối ưu hơn so với công cụ truyền thống.

- Tailwindcss: Là framework CSS utility-first giúp xây UI nhanh bằng cách ghép các utility class trực tiếp trong JSX

3.Cơ sở dữ liệu

MySQL – Hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được sử dụng để:

- Lưu trữ dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng) theo thời gian thực.
- Ghi lại lịch sử hoạt động của các thiết bị (bật/tắt đèn, quạt, điều hòa).

4. Giao thức truyền thông MQTT

- MQTT (Message Queuing Telemetry Transport): Giao thức truyền thông nhẹ, được sử dụng để truyền dữ liệu giữa ESP32 và server.
- HTTP/HTTPS: Giao thức truyền tải siêu văn bản, được sử dụng trong API Restful để giao tiếp giữa frontend và backend.³

5. Phần cứng ESP32, DHT11

- Arduino IDE: Môi trường phát triển tích hợp được sử dụng để lập trình cho ESP32.
- ESP32: Vi điều khiển có khả năng kết nối WiFi và Bluetooth, được sử dụng làm thiết bị IoT chính.
- DHT11: Cảm biến nhiệt độ và độ ẩm, được sử dụng để đo các thông số môi trường.
- LDR (Light Dependent Resistor): Cảm biến ánh sáng, được sử dụng để đo cường độ ánh sáng.

6. Công cụ phát triển quản lý mã nguồn

- Git: Hệ thống quản lý phiên bản phân tán, được sử dụng để quản lý mã nguồn.
- Github: Nền tảng lưu trữ mã nguồn trực tuyến, được sử dụng để lưu trữ và chia sẻ mã nguồn dự án.
- Postman: Công cụ phát triển API, được sử dụng để kiểm thử và tài liệu hóa API.

Việc sử dụng các công nghệ này cho phép xây dựng một hệ thống IoT mạnh mẽ, có

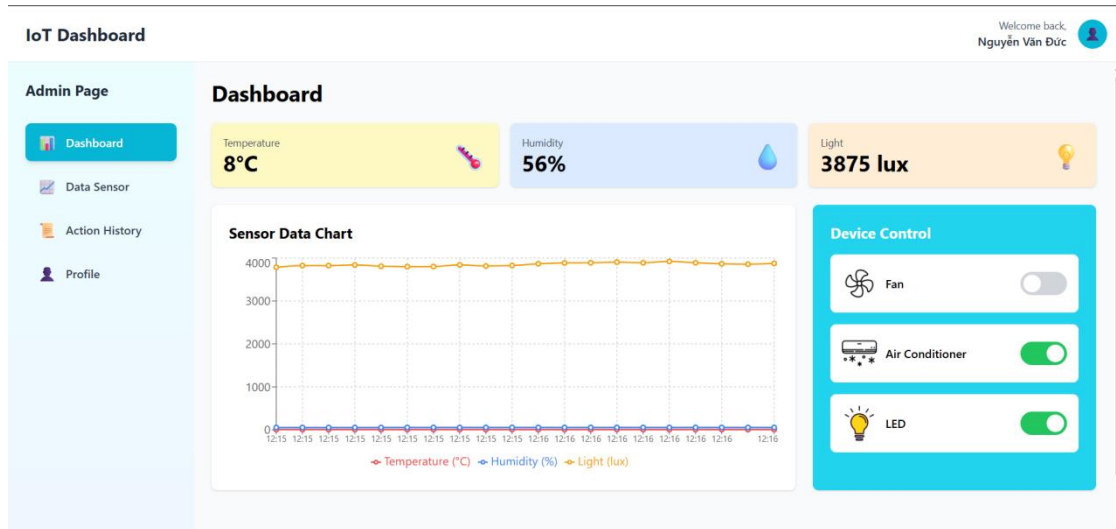
khả năng mở rộng và dễ bảo trì. Sự kết hợp giữa các công nghệ frontend hiện đại,

backend mạnh mẽ và thiết bị IoT linh hoạt tạo nên một giải pháp toàn diện cho việc

giám sát và điều khiển môi trường thông minh.

Chương 2: Giao diện

1. Trang DashBoard



Hình 1 Trang DashBoard

- Bảng thông số hiện tại:
 - + Nhiệt độ: Hiện thị nhiệt độ hiện tại với biểu tượng nhiệt kế.
 - + Độ ẩm: Hiện thị độ ẩm hiện tại với biểu tượng giọt nước.
 - + Ánh sáng: Hiện thị cường độ ánh sáng hiện tại với biểu tượng mặt trời.
- Biểu đồ theo dõi:
 - + Đường màu đỏ thể hiện nhiệt độ theo thời gian
 - + Đường màu xanh dương thể hiện độ ẩm.
 - + Đường màu vàng thể hiện cường độ ánh sáng
- Điều khiển thiết bị:
 - + Quạt: khi tắt sẽ hiện màu đỏ còn khi bật sẽ chuyển màu xanh và quạt quay
 - + Điều hòa: khi tắt sẽ hiện màu đỏ còn khi bật sẽ chuyển màu xanh và hiện đậm hơn
 - + Đèn : khi tắt sẽ hiện màu đỏ còn khi bật sẽ chuyển màu xanh và bóng đèn phát sáng
- Giao diện này cho phép người dùng:
 - + Theo dõi các thông số môi trường theo thời gian thực.
 - + Xem xu hướng thay đổi của nhiệt độ, độ ẩm và ánh sáng qua thời gian
 - + Điều khiển các thiết bị như quạt, điều hòa và đèn một cách hiệu quả

2. Trang DataSensor

IoT Dashboard

Welcome back, Nguyễn Văn Đức

Admin Page

Dashboard

Data Sensor

Action History

Profile

Tim kiếm:

Nhập nội dung tìm kiếm...

Tim kiếm theo:

Tất cả

Tim kiếm

ID	Temperature (°C) ↑↓	Humidity (%) ↑↓	Light (lux) ↑↓	Time ↑↓
1	28.4°C	87%	2295 lux	2025-10-10 01:50:30
2	28.4°C	88%	2311 lux	2025-10-10 01:50:32
3	28.4°C	87%	2311 lux	2025-10-10 01:50:34
4	28.4°C	87%	2320 lux	2025-10-10 01:50:36
5	28.3°C	88%	2313 lux	2025-10-10 01:50:38
6	28.3°C	87%	2306 lux	2025-10-10 01:50:40
7	28.4°C	88%	2311 lux	2025-10-10 01:50:42
8	28.3°C	87%	2303 lux	2025-10-10 01:50:44
9	28.3°C	88%	2305 lux	2025-10-10 01:50:46
10	28.4°C	87%	2315 lux	2025-10-10 01:50:48

Hiện thị 1 - 10 / 8588

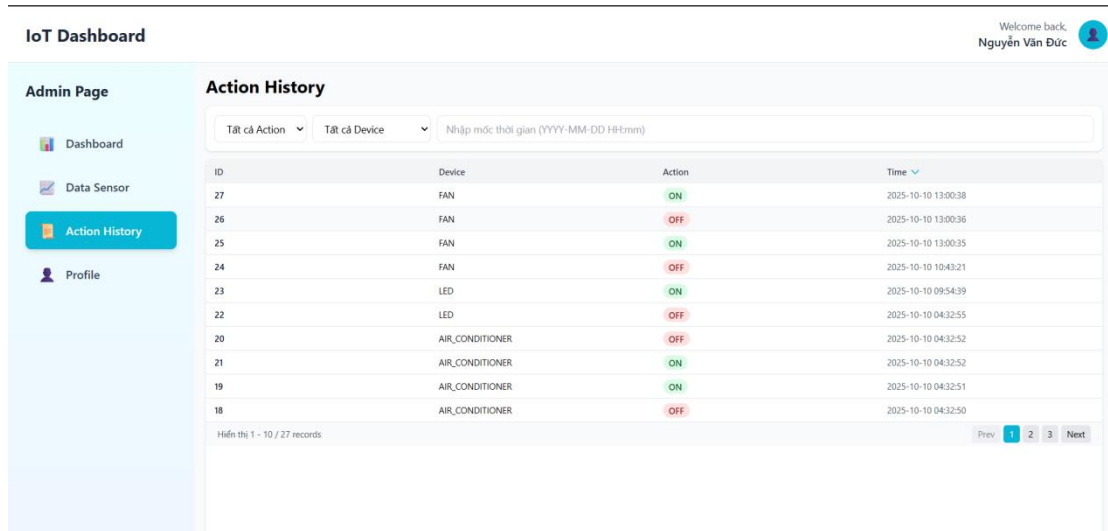
Prev 1 2 3 4 ... 859 Next

Hình 2: Trang DataSensor

- Thanh tìm kiếm:
 - + Ô nhập liệu cho phép tìm kiếm theo ID, nhiệt độ, độ ẩm, ánh sáng hoặc thời gian theo giờ, phút, giây
 - + Dropdown menu để chọn phạm vi tìm kiếm như Tất cả, Nhiệt độ, Độ ẩm, Ánh sáng hoặc thời gian
 - + Nút search để thực hiện tìm kiếm
- Bảng dữ liệu cảm biến:
 - + Hiện thị dữ liệu dưới dạng bảng với các cột
 - + ID: Số định danh duy nhất cho mỗi bản ghi
 - + Nhiệt độ (°C): Nhiệt độ được đo
 - + Độ ẩm (%): Độ ẩm được đo
 - + Ánh sáng (lux); Cường độ ánh sáng được đo
 - + Thời gian: Thời gian ghi nhận dữ liệu
 - + Mỗi cột có biểu tượng mũi tên lên/xuống cho phép sắp xếp dữ liệu
 - + Có tính năng phân trang và có thể chọn page size theo dropdown
- Trang Data Sensor này cho phép người dùng
 - + Xem dữ liệu cảm biến chi tiết theo thời gian.
 - + Tìm kiếm dữ liệu cụ thể dựa trên các tiêu chí khác nhau.

- + Sắp xếp dữ liệu theo bất kỳ cột nào để phân tích xu hướng.
- + Theo dõi sự thay đổi của các thông số môi trường theo thời gian.

3. Trang Action History



IoT Dashboard Welcome back, Nguyễn Văn Đức

Admin Page

- Dashboard
- Data Sensor
- Action History**
- Profile

Action History

Tất cả Action ▼ Tất cả Device ▼ Nhập mốc thời gian (YYYY-MM-DD HH:mm)

ID	Device	Action	Time
27	FAN	ON	2025-10-10 13:00:38
26	FAN	OFF	2025-10-10 13:00:36
25	FAN	ON	2025-10-10 13:00:35
24	FAN	OFF	2025-10-10 10:43:21
23	LED	ON	2025-10-10 09:54:39
22	LED	OFF	2025-10-10 04:32:55
20	AIR_CONDITIONER	OFF	2025-10-10 04:32:52
21	AIR_CONDITIONER	ON	2025-10-10 04:32:52
19	AIR_CONDITIONER	ON	2025-10-10 04:32:51
18	AIR_CONDITIONER	OFF	2025-10-10 04:32:50

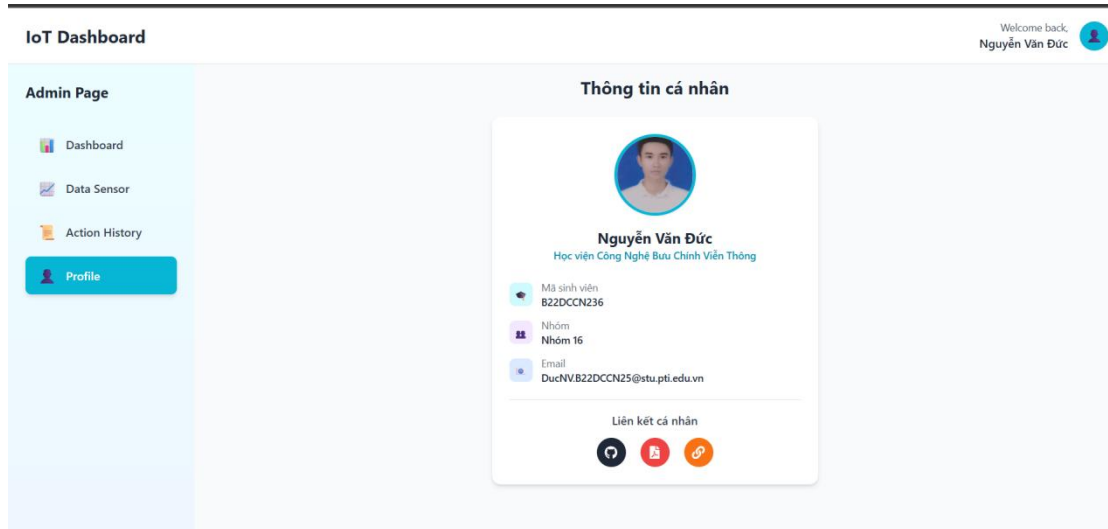
Hiện thị 1 - 10 / 27 records Prev 1 2 3 Next

Hình 3: Trang Action History

- Thanh tìm kiếm:
 - + Ô nhập liệu cho phép tìm kiếm theo thời gian theo giờ, phút, giây
 - + Dropdown menu để chọn loại thiết bị như Đèn, Quạt, Điều hòa hoặc Tất cả
 - + Dropdown menu để chọn hành động ON/ OFF
 - + Khi nhập text tự động thực hiện tìm kiếm thực hiện tìm kiếm.
- Bảng lịch sử hành động:
 - + Hiển thị dữ liệu dưới dạng bảng với các cột:
 - + ID: Số định danh duy nhất cho mỗi hành động.
 - + Thiết bị: gồm quạt, điều hòa và đèn
 - + Hành động: Trạng thái thiết bị (Bật/Tắt)
 - + Thời gian: Thời gian thực hiện hành động
 - + Mỗi cột có biểu tượng mũi tên lên/xuống cho phép sắp xếp dữ liệu
 - + Có tính năng phân trang
- Trang Action History này cho phép người dùng:
 - + Xem lịch sử chi tiết về các hành động điều khiển thiết bị.

- + Tìm kiếm các hành động cụ thể dựa trên loại thiết bị hoặc thời gian.
- + Theo dõi thứ tự và tần suất của các hành động điều khiển.
- + Phân tích mô hình sử dụng thiết bị theo thời gian.

4. Trang Profile



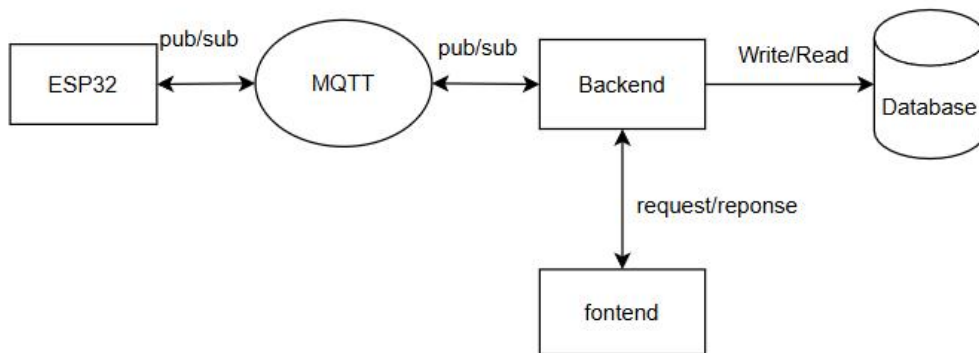
Hình 4: Trang Profile

- Thẻ thông tin cá nhân
- Ảnh đại diện
- Thông tin cơ bản:
 - + Tên : Nguyễn Văn Đức
 - + Mã SV: B22DCCN236
 - + Email: DucNV.B22DCCN236@stu.ptit.edu.vn
 - + Nhóm 16
 - + PDF tài liệu: liên kết tài liệu dạng PDF
 - + API Docs: liên kết đến tài liệu API
 - + Github: liên kết đến github project

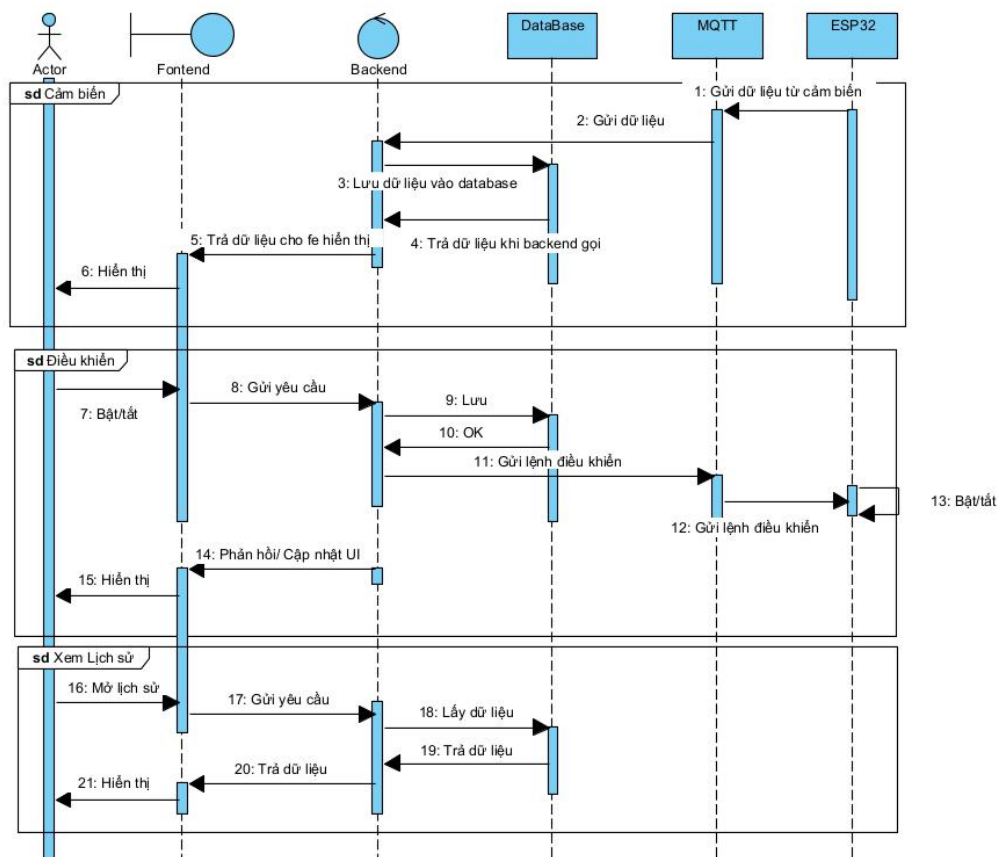
Chương 3: Thiết kế tổng quan và chi tiết

I.Kiến trúc tổng quan

- Sơ đồ tổng quan kiến trúc:



- + ESP32: Thiết bị phần cứng đọc dữ liệu từ cảm biến và điều khiển các thiết bị như quạt, đèn LED, và điều hòa.
 - + MQTT Broker: Trung gian truyền tin giữa ESP32 và Backend.
 - + Backend (Node.js): Xử lý logic nghiệp vụ, lưu trữ dữ liệu, và cung cấp API cho Frontend.
 - + Database (MySQL): Lưu trữ dữ liệu cảm biến và lịch sử hành động.
 - + Frontend (React): Giao diện người dùng để hiển thị dữ liệu và điều khiển thiết bị.
- Mô tả luồng dữ liệu và tương tác giữa các thành phần:



- Luồng dữ liệu cảm biến: ESP32 đọc dữ liệu từ cảm biến (nhiệt độ, độ ẩm, ánh sáng) → ESP32 gửi dữ liệu qua MQTT đến MQTT Broker → Backend nhận dữ liệu từ MQTT Broker và lưu vào database → Frontend định kỳ gọi API từ Backend để lấy dữ liệu mới nhất và hiển thị
- Luồng dữ liệu điều khiển thiết bị: Người dùng tương tác với giao diện trên Frontend để điều khiển thiết bị → Frontend gửi yêu cầu điều khiển đến Backend thông qua API → Backend lưu lại hành động vào database ,xử lý yêu cầu và gửi lệnh điều khiển qua MQTT đến ESP32 → ESP32 nhận lệnh và thực hiện điều khiển thiết bị tương ứng.

II.Backend

- Cấu trúc thư mục và file
 - API endpoints và chức năng
 - Dữ liệu cảm biến
 - + GET /api/sensors: Nhận tất cả dữ liệu cảm biến bao gồm Nhiệt độ, Độ ẩm và Ánh sáng.
- Input

GET http://localhost:3000/api/sensors Send

Params Authorization Headers (6) Body Scripts Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Output

200 OK • 19 ms • 641.51 KB Save Response

JSON Preview Visualize

```

1 {
2   "data": [
3     {
4       "id": 7088,
5       "temperature": 28.3,
6       "humidity": 95,
7       "light": 3984,
8       "created_at": "2025-10-17 09:31:48"
9     },
10    {
11      "id": 7087,
12      "temperature": 28.3,
13      "humidity": 95,
14      "light": 3983,
15      "created_at": "2025-10-17 09:31:46"
16    },
17    {
18      "id": 7086,
19      "temperature": 28.3,
20      "humidity": 95,
21      "light": 3975,
22      "created_at": "2025-10-17 09:31:44"
23    }
24  ]
25 }
```

- Tìm kiếm dữ liệu cảm bằng bộ lọc theo loại dữ liệu, thời gian và sắp xếp

Input

GET http://localhost:3000/api/sensors?page=1&limit=10&sortField=temperature&sortOrder=dsec&search=2025-10-16&searchField=time Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	
<input checked="" type="checkbox"/>	page	1		
<input checked="" type="checkbox"/>	limit	10		
<input checked="" type="checkbox"/>	sortField	temperature		
<input checked="" type="checkbox"/>	sortOrder	dsec		🔍 Type 🗑
<input checked="" type="checkbox"/>	search	2025-10-16		
<input checked="" type="checkbox"/>	searchField	time		
	Key	Value	Description	

Output:

```
Body Cookies Headers (8) Test Results 200 OK 15 ms 1.26 KB Save Response
{} JSON Preview Visualize
1 {
2   "data": [
3     {
4       "id": 612,
5       "temperature": 28.6,
6       "humidity": 95,
7       "light": 2617,
8       "created_at": "2025-10-16 22:31:05"
9     },
10    {
11      "id": 624,
12      "temperature": 28.6,
13      "humidity": 95,
14      "light": 2611,
15      "created_at": "2025-10-16 22:31:29"
16    },
17    {
18      "id": 626,
19      "temperature": 28.6,
20      "humidity": 95,
21      "light": 2610,
22      "created_at": "2025-10-16 22:31:33"
23    },
24    {
25      "id": 610,
26      "temperature": 28.6,
27      "humidity": 95,
```

- Lấy dữ liệu mới nhất từ cảm biến
- + Get api/sensors/latest : Nhận dữ liệu mới nhất từ cả biến

```
GET http://localhost:3000/api/sensors/latest Send
Params Authorization Headers (6) Body Scripts Settings Cookies
Body Cookies Headers (8) Test Results 200 OK 6 ms 359 B Save Response
{} JSON Preview Visualize
1 {
2   "id": 7183,
3   "temperature": 28.3,
4   "humidity": 95,
5   "light": 4095,
6   "created_at": "2025-10-17 09:34:58"
7 }
```

- Lịch sử hành động
- Get: /api/history: Lấy tất cả lịch sử hành động

```
GET http://localhost:3000/api/actions/history Send
Params Authorization Headers (6) Body Scripts Settings Cookies
Body Cookies Headers (8) Test Results 200 OK 7 ms 3.37 KB Save Response
{} JSON Preview Visualize
1 [
2   {
3     "id": 41,
4     "device": "FAN",
5     "action": "ON",
6     "created_at": "2025-10-17 02:09:17"
7   },
8   {
9     "id": 40,
10    "device": "FAN",
11    "action": "OFF",
12    "created_at": "2025-10-17 02:01:56"
13  },
14  {
15    "id": 39,
16    "device": "FAN",
17    "action": "ON",
18    "created_at": "2025-10-17 02:01:56"
19  },
20  {
21    "id": 38,
22    "device": "FAN",
23    "action": "OFF",
24    "created_at": "2025-10-17 02:00:17"
25  },
26  ]
```

- Tìm kiếm dữ liệu lịch sử hành động bằng bộ lọc theo thiết bị, hành động và ngày, sắp xếp.

Input

GET http://localhost:3000/api/actions/history?device=FAN&action=ON&time=2025&sortOrder=dsec

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
device	FAN	
action	ON	
time	2025	
sortOrder	dsec	

Output

200 OK • 12 ms • 934 B • Save Response

JSON Preview Visualize

```

1 [
2   {
3     "id": 41,
4     "device": "FAN",
5     "action": "ON",
6     "created_at": "2025-10-17 02:09:17"
7   },
8   {
9     "id": 39,
10    "device": "FAN",
11    "action": "ON",
12    "created_at": "2025-10-17 02:01:56"
13  },
14  {
15    "id": 37,
16    "device": "FAN",
17    "action": "ON",
18    "created_at": "2025-10-17 01:59:50"
19  },
20  {
21    "id": 31,
22    "device": "FAN",
23    "action": "ON",
24    "created_at": "2025-10-17 01:59:50"
25  }
26 ]

```

- Lấy trạng thái thiết bị:

+ Get /api/actions/states: Lấy trạng thái hiện tại của thiết bị

200 OK • 5 ms • 388 B • Save Response

JSON Preview Visualize

```

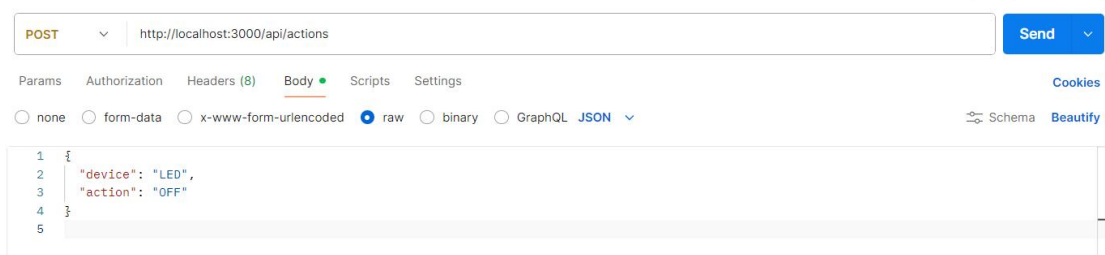
1 [
2   {
3     "device_name": "FAN",
4     "state": "ON"
5   },
6   {
7     "device_name": "AIR_CONDITIONER",
8     "state": "OFF"
9   },
10  {
11    "device_name": "LED",
12    "state": "OFF"
13  }
14 ]

```

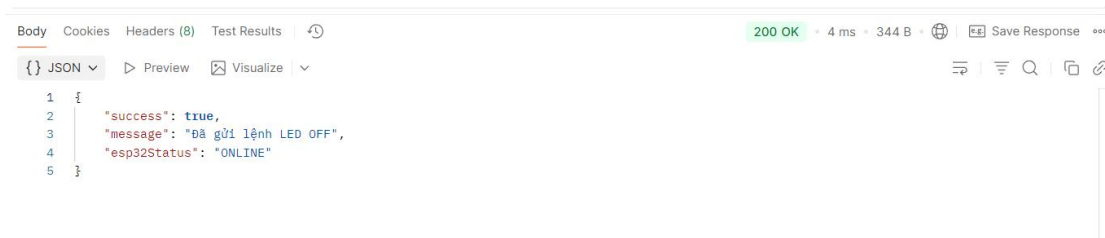
- Gửi lệnh điều khiển thiết bị :

+Post /api/actions: Gửi lệnh để bật tắt thiết bị

Input:



Output:



- Xử lý dữ liệu cảm biến và điều khiển thiết bị

+ Dữ liệu cảm biến được nhận qua MQTT và lưu vào database config/ mqttClient.js

+ Lệnh điều khiển thiết bị được gửi qua API, xử lý trong routes/actions.js và gửi qua mqtt đến ESP32

III.Kết nối MQTT

1.Cấu hình MQTT broker

Sử dụng thư viện mqtt để kết nối MQTT broker

- Cấu hình kết nối được định nghĩa trong config/mqttClient.js:

```
const mqttClient = mqtt.connect("mqtt://localhost:1884", {
  username: "scwud",
  password: "Abc@1234",
});
```

- Đăng kí những topic cần thiết:

```
mqttClient.on("connect", () => {
  console.log("MQTT Broker connected");

  mqttClient.subscribe("dataSensor", (err) => {
    if (err) console.error("❌ Lỗi subscribe dataSensor:", err);
    else console.log("✅ Subscribed: dataSensor");
  });

  mqttClient.subscribe("controlLED", (err) => {
    if (err) console.error("❌ Lỗi subscribe controlLED:", err);
    else console.log("✅ Subscribed: controlLED");
  });

  mqttClient.subscribe("esp32/status", (err) => {
    if (err) console.error("❌ Lỗi subscribe esp32/status:", err);
    else console.log("✅ Subscribed: esp32/status");
  });
});
```

- Xử lý tin nhắn nhận được

```
mqttclient.on("message", (topic, message) => {
  const msg = message.toString();
```

- Nhận dữ liệu cảm biến và lưu vào database

```
// --- Nhận dữ liệu cảm biến ---
if (topic === "dataSensor") {
  // Cập nhật trạng thái ESP32 online khi nhận được dữ liệu
  const wasOffline = !esp32Status.isOnline;
  esp32Status.isOnline = true;
  esp32Status.lastSeen = Date.now();

  if (wasOffline) {
    console.log("🟢 ESP32 ONLINE (vừa kết nối lại)");
  }

  const data = JSON.parse(msg);
  const { temp, humidity, light, status } = data;
  // 🔍 Nếu chỉ có "status": "ONLINE" hoặc "ALIVE" (heartbeat)
  if (!temp && !humidity && (status === "ONLINE" || status === "ALIVE")) {
    console.log("💖 [Heartbeat] ESP32 vẫn online");
    esp32Status.lastSeen = Date.now();
    return;
  }
  // 📊 Nếu có dữ liệu cảm biến đầy đủ
  if (temp !== undefined && humidity !== undefined && light !== undefined) {
    const sql =
      "INSERT INTO sensor_data (temperature, humidity, light) VALUES (?, ?, ?)";
    db.query(sql, [temp, humidity, light], (err, result) => {
      if (err) {
        console.error("❌ Lỗi ghi DB:", err);
      } else {
        console.log("📄 Dữ liệu sensor lưu vào DB - ID:", result.insertId);
      }
    });
  }
}
```

- Khôi phục trạng thái khi kết nối lại esp32

```
// --- Xử lý controlled topic ---
else if (topic === "controlled") {
  if (msg === "GET_STATE") {
    console.log("📡 ESP32 yêu cầu trạng thái thiết bị");
    const sql = "SELECT device_name, state FROM device_state";
    db.query(sql, (err, results) => {
      if (err) return console.error("❌ Lỗi đọc trạng thái:", err);
      results.forEach(({ device_name, state }) => {
        const command = `${device_name}_${state}`;
        mqttClient.publish("controlled", command);
        console.log("📤 Gửi lệnh:", command);
      });
    });
  }
}
```

IV. Mô hình dữ liệu

- Cấu trúc bảng dữ liệu cảm biến

- ```
CREATE TABLE IF NOT EXISTS sensor_data (
 id INT AUTO_INCREMENT PRIMARY KEY,
 temperature FLOAT NOT NULL,
 humidity FLOAT NOT NULL,
 light INT NOT NULL,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 INDEX idx_created_at (created_at)
);
```

- Cấu trúc bảng lịch sử hành động :

```
-- Bảng lịch sử hành động
CREATE TABLE IF NOT EXISTS action_history (
 id INT AUTO_INCREMENT PRIMARY KEY,
 device VARCHAR(50) NOT NULL,
 action VARCHAR(10) NOT NULL,
 created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
 INDEX idx_device (device),
 INDEX idx_created_at (created_at)
);
```

- Cấu trúc bảng trạng thái thiết bị :

```
-- Bảng trạng thái thiết bị
CREATE TABLE IF NOT EXISTS device_state (
 id INT AUTO_INCREMENT PRIMARY KEY,
 device_name VARCHAR(50) UNIQUE NOT NULL,
 state VARCHAR(10) NOT NULL DEFAULT 'OFF',
 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

## V. Lập trình ESP32

### - Cấu hình kết nối Wifi và MQTT

```

5 #define WIFI_SSID "Scwud"
6 #define WIFI_PASSWORD "14015353"
7 #define MQTT_SERVER "192.168.144.94"
8 #define MQTT_PORT 1884
9 #define MQTT_USER "scwud"
10 #define MQTT_PASSWORD "Abc@1234"
11 // Topic
12 #define TOPIC_SENSOR "dataSensor"
13 #define TOPIC_CONTROL "controlled"
14
15 #define DHTPIN 25
16 #define DHTTYPE DHT11
17 #define FAN_PIN 18
18 #define AIR_CONDITIONER_PIN 19
19 #define LED_PIN 21
20 #define LIGHT_SENSOR_PIN 36

```

## - Đọc dữ liệu cảm biến và gửi qua MQTT:

```

void loop() {
 // Kiểm tra WiFi
 if (WiFi.status() != WL_CONNECTED) {
 Serial.println("⚠️ WiFi mất kết nối, đang kết nối lại...");
 setup_wifi();
 return;
 }
 // Kiểm tra MQTT
 if (!client.connected()) {
 reconnect();
 return;
 }
 client.loop();
 unsigned long now = millis();
 // 📡 Gửi dữ liệu cảm biến mỗi 2 giây
 if (now - lastRead >= 2000) {
 lastRead = now;

 float h = dht.readHumidity();
 float t = dht.readTemperature();
 int lightValue = analogRead(LIGHT_SENSOR_PIN);

 if (isnan(h) || isnan(t)) {
 Serial.println("⚠️ Lỗi đọc DHT11!");
 return;
 }
 char payload[200];
 snprintf(payload, sizeof(payload),
 "{\"temp\":%.1f,\"humidity\":%.1f,\"light\":%d,\"status\":\"ALIVE\"}",
 t, h, lightValue);
 client.publish(TOPIC_SENSOR, payload);
 Serial.print("📡 [Sensor] ");
 Serial.println(payload);

 lastAliveSignal = now;
 }
 // ❤️ Gửi heartbeat mỗi 5 giây (để server biết ESP32 vẫn online)
 if (now - lastAliveSignal >= 5000 && !firstConnection) {
 client.publish(TOPIC_SENSOR, "{\"status\":\"ALIVE\"}");
 Serial.println("❤️ [Heartbeat] ESP32 vẫn online");
 lastAliveSignal = now;
 }
}

```

## - Nhận lệnh điều khiển và thực thi :

```

void callback(char* topic, byte* message, unsigned int length) {
 String msg;
 for (int i = 0; i < length; i++) msg += (char)message[i];
 Serial.print("📩 Nhận lệnh trên topic [");
 Serial.print(topic);
 Serial.print("]: ");
 Serial.println(msg);

 // Kiểm tra tên thiết bị
 if (msg.startsWith("FAN_")) {
 digitalWrite(FAN_PIN, msg.endsWith("ON") ? HIGH : LOW);
 Serial.println(msg.endsWith("ON") ? "🌀 FAN BẬT" : "🌀 FAN TẮT");
 }
 else if (msg.startsWith("AIR_CONDITIONER_")) {
 digitalWrite(AIR_CONDITIONER_PIN, msg.endsWith("ON") ? HIGH : LOW);
 Serial.println(msg.endsWith("ON") ? "❄️ AC BẬT" : "❄️ AC TẮT");
 }
 else if (msg.startsWith("LED_")) {
 digitalWrite(LED_PIN, msg.endsWith("ON") ? HIGH : LOW);
 Serial.println(msg.endsWith("ON") ? "💡 LED BẬT" : "💡 LED TẮT");
 }
}

```

## Chương 4: Kết quả

### I. Chức năng đã hoàn thành

Liệt kê các tính năng đã triển khai thành công

- Thu thập và hiển thị dữ liệu cảm biến:
  - + Hệ thống đã thành công trong việc thu thập dữ liệu từ các cảm biến nhiệt độ, độ ẩm và ánh sáng thông qua ESP32.
  - + Dữ liệu được truyền qua MQTT và lưu trữ trong cơ sở dữ liệu MySQL.
  - + Frontend hiển thị dữ liệu cảm biến dưới dạng bảng và biểu đồ thời gian thực.
- Điều khiển thiết bị từ xa:
  - + Người dùng có thể điều khiển quạt, đèn LED và điều hòa thông qua giao diện web.
  - + Lệnh điều khiển được gửi từ Frontend đến Backend, sau đó truyền qua MQTT đến ESP32.
  - + ESP32 thực hiện lệnh điều khiển và gửi phản hồi về trạng thái thiết bị.
- Lưu trữ và hiển thị lịch sử hành động:

- + Hệ thống ghi lại tất cả các hành động điều khiển thiết bị.
- + Người dùng có thể xem lịch sử hành động với các tùy chọn lọc và phân trang.
- Giao diện người dùng thân thiện:
  - + Frontend được phát triển bằng React, cung cấp giao diện trực quan và dễ sử dụng.
  - + Người dùng có thể dễ dàng chuyển đổi giữa các chế độ xem khác nhau: Dashboard, Data Sensor, Action History và Profile.
- API linh hoạt:
  - + Backend cung cấp các API endpoint cho phép truy xuất dữ liệu cảm biến và lịch sử hành động với nhiều tùy chọn lọc, sắp xếp và phân trang

## **II. Hiệu suất hệ thống**

### **1. Đánh giá về tốc độ phản hồi**

- Thời gian phản hồi API:
  - + Các API endpoint có thời gian phản hồi trung bình dưới 200ms cho các truy vấn cơ bản.
  - + Đối với các truy vấn phức tạp hơn (ví dụ: tìm kiếm với nhiều điều kiện), thời gian phản hồi vẫn dưới 500ms.
- Độ trễ điều khiển thiết bị:
  - + Thời gian từ khi người dùng gửi lệnh điều khiển đến khi thiết bị thực sự thay đổi trạng thái trung bình khoảng 1-2 giây.
  - + Phần lớn độ trễ này là do thời gian truyền tin qua MQTT và xử lý trên ESP32.
- Cập nhật dữ liệu thời gian thực:
  - + Dữ liệu cảm biến được cập nhật trên giao diện người dùng mỗi 2 giây, đảm bảo thông tin luôn mới nhất

### III. Cải tiến trong tương lai

#### - Tăng cường bảo mật:

- + Triển khai hệ thống xác thực và phân quyền người dùng.
- + Sử dụng SSL/TLS cho kết nối MQTT và API.
- + Thêm cơ chế mã hóa dữ liệu nhạy cảm trong cơ sở dữ liệu.

#### - Cải thiện khả năng mở rộng:

- + Triển khai cơ chế caching để giảm tải cho database.
- + Xem xét sử dụng cơ sở dữ liệu NoSQL như MongoDB cho dữ liệu cảm biến để tăng hiệu suất khi lưu trữ dữ liệu lớn.
- + Triển khai load balancing cho Backend khi số lượng request tăng cao.

#### - Nâng cao trải nghiệm người dùng:

- + Thêm tính năng thông báo và cảnh báo khi các chỉ số vượt ngưỡng.
- + Phát triển ứng dụng di động để người dùng có thể theo dõi và điều khiển từ xa dễ dàng hơn.
- + Tích hợp trí tuệ nhân tạo để đưa ra các đề xuất tối ưu hóa năng lượng dựa trên dữ liệu cảm biến.

#### - Mở rộng khả năng tương thích:

- + Hỗ trợ thêm các loại cảm biến và thiết bị IoT khác.
- + Phát triển SDK để cho phép các nhà phát triển bên thứ ba tích hợp với hệ thống.

#### - Tối ưu hóa năng lượng:

- + Triển khai chế độ ngủ sâu cho ESP32 khi không cần thu thập dữ liệu.
- + Điều chỉnh tần suất gửi dữ liệu dựa trên mức độ thay đổi của các chỉ số.

#### - Cải thiện khả năng phân tích:

- + Thêm các công cụ phân tích dữ liệu nâng cao để cung cấp insights về xu hướng và mẫu tiêu thụ năng lượng.

- + Tích hợp machine learning để dự đoán và tối ưu hóa việc sử dụng năng lượng.
- Tăng cường độ tin cậy:
  - + Triển khai cơ chế backup và khôi phục dữ liệu tự động.
  - + Cải thiện logging và monitoring để phát hiện và xử lý sự cố nhanh chóng