HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CƠ BẢN I

ĐỀ TÀI: SMART HOME

GIẢNG VIÊN: NGUYỄN QUỐC UY

NHÓM LỚP HỌC: 06

HỌ VÀ TÊN: NGUYỄN ANH DUY

MSV: B21DCCN298

LỚP: D21CNPM5

Hà Nội – 2024

LỜI CẢM ƠN

Em xin chân thành gửi lời cảm ơn sâu sắc đến thầy **Nguyễn Quốc Uy** đã tận tình hướng dẫn và truyền đạt kiến thức trong suốt quá trình học môn **Internet of Things (IoT)**. Những bài giảng của thầy không chỉ cung cấp nền tảng vững chắc về IoT mà còn giúp em hiểu rõ hơn về cách áp dụng công nghệ này vào thực tiễn.

Nhờ sự tận tâm của thầy, em đã có thể hoàn thành dự án này một cách tốt đẹp. Những chỉ dẫn chi tiết và sự hỗ trợ của thầy đã giúp em vượt qua những khó khăn trong quá trình nghiên cứu và triển khai. Đây là một bước tiến quan trọng trong quá trình học tập và phát triển của em.

Em xin kính chúc thầy luôn dồi dào sức khỏe và thành công trong sự nghiệp giảng dạy, tiếp tục truyền lửa và cảm hứng cho các thế hệ sinh viên sau.

GIỚI THIỆU ĐỀ TÀI

Hệ thống IoT SmartHome nhỏ gọn này được thiết kế để cung cấp giải pháp tự động hóa và quản lý môi trường trong không gian gia đình hoặc văn phòng. Hệ thống tích hợp các cảm biến và thiết bị thông minh giúp theo dõi và điều khiển các thông số môi trường như nhiệt độ, độ ẩm và ánh sáng, đồng thời cho phép điều khiển từ xa các thiết bị điện tử như quạt, đèn, và điều hòa không khí.

Contents

I. Tính năng

1. Các tính năng chính của hệ thống:

- Giám sát nhiệt độ, độ ẩm, ánh sáng: Hệ thống trang bị các cảm biến đo nhiệt độ, độ ẩm và ánh sáng, cung cấp dữ liệu theo thời gian thực về môi trường bên trong ngôi nhà. Thông tin này giúp người dùng có cái nhìn tổng quát về tình trạng không gian, từ đó điều chỉnh các thiết bị cho phù hợp.
- Điều khiển thông minh quạt, đèn và điều hòa: Người dùng có thể dễ dàng điều khiển quạt, đèn và điều hòa từ xa qua giao diện web. Hệ thống cho phép bật/tắt thiết bị hoặc điều chỉnh các chế độ hoạt động theo yêu cầu.
- Tự động hóa môi trường: Dựa trên các dữ liệu từ cảm biến, hệ thống có thể tự động điều chỉnh các thiết bị để đảm bảo môi trường luôn thoải mái. Ví dụ, hệ thống có thể tự động bật quạt khi nhiệt độ vượt quá ngưỡng cho phép, hoặc bật đèn khi ánh sáng quá thấp.

2. Cách thức thu thập và truyền dữ liệu:

a. Cảm biến nhiệt độ độ ẩm DHT11

- Cách thu thập dữ liệu của cảm biến DHT11 như sau:

  + Cảm biến DHT11 có khả năng thu thập dữ liệu về nhiệt độ và độ ẩm từ môi trường xung quanh. Nó sử dụng các linh kiện điện tử cảm biến để đo độ ẩm và nhiệt độ.

  + Để đo độ ẩm, DHT11 sử dụng một tấm điện dung để đo độ ẩm không khí. Khi độ ẩm thay đổi, giá trị điện dung cũng thay đổi, và cảm biến sẽ chuyển đổi sự thay đổi đó thành giá trị số để gửi đi.

  + Để đo nhiệt độ, DHT11 sử dụng một điện trở nhiệt (thermistor). Khi nhiệt độ thay đổi, điện trở của thermistor cũng thay đổi, và cảm biến sẽ chuyển đổi thành giá trị số đại diện cho nhiệt độ.

- Cách truyền dữ liệu:

  + Dữ liệu nhiệt độ và độ ẩm được gửi từ DHT11 đến vi điều khiển ESP32 qua giao tiếp 1-Wire.

+ Sau khi nhận dữ liệu, vi điều khiển sẽ xử lý và gửi tiếp dữ liệu lên hệ thống qua các giao thức như MQTT để hiển thị hoặc lưu trữ trên máy chủ.

b. Cảm biến ánh sáng

- Cách thu thập dữ liệu:

  + Cảm biến ánh sáng đo cường độ ánh sáng trong môi trường thông qua sự biến đổi dòng điện hoặc điện trở.
  + Loại cảm biến này thường sử dụng một điện trở quang (photoresistor) hoặc diode quang để phát hiện lượng ánh sáng. Khi cường độ ánh sáng thay đổi, giá trị điện trở của cảm biến cũng thay đổi theo, từ đó biến đổi thành tín hiệu điện tương ứng.
  + Cảm biến ánh sáng thường được sử dụng để điều khiển mức độ sáng của đèn hoặc tự động bật/tắt đèn dựa trên cường độ ánh sáng xung quanh.

- Cách truyền dữ liệu:

  + Dữ liệu ánh sáng được truyền từ cảm biến tới vi điều khiển dưới dạng tín hiệu điện tương tự (analog). Vi điều khiển sẽ chuyển đổi tín hiệu này thành giá trị số tương ứng với mức độ sáng.

  + Tín hiệu ánh sáng sau đó có thể được xử lý để thực hiện các hành động cụ thể, ví dụ như bật đèn khi cường độ ánh sáng quá thấp. Vi điều khiển sẽ truyền thông tin qua giao thức MQTT để giám sát hoặc điều khiển thiết bị từ xa.

3. Cách thức xử lý dữ liệu – tính năng thông minh xử lý dữ liệu:

a. Cảm biến nhiệt độ độ ẩm DHT11

- Vi điều khiển nhận dữ liệu từ DHT11 và tính toán nhiệt độ và độ ẩm hiện tại. Dữ liệu này được sử dụng để hiển thị tình trạng nhiệt độ và độ ẩm môi trường.
- Khi nhiệt độ hoặc độ ẩm vượt ngưỡng thiết lập trước (ví dụ: nhiệt độ > 40°C, độ ẩm > 70%), vi điều khiển có thể tự động điều khiển quạt hoặc điều hòa để điều chỉnh môi trường.

b. Cảm biến ánh sáng:

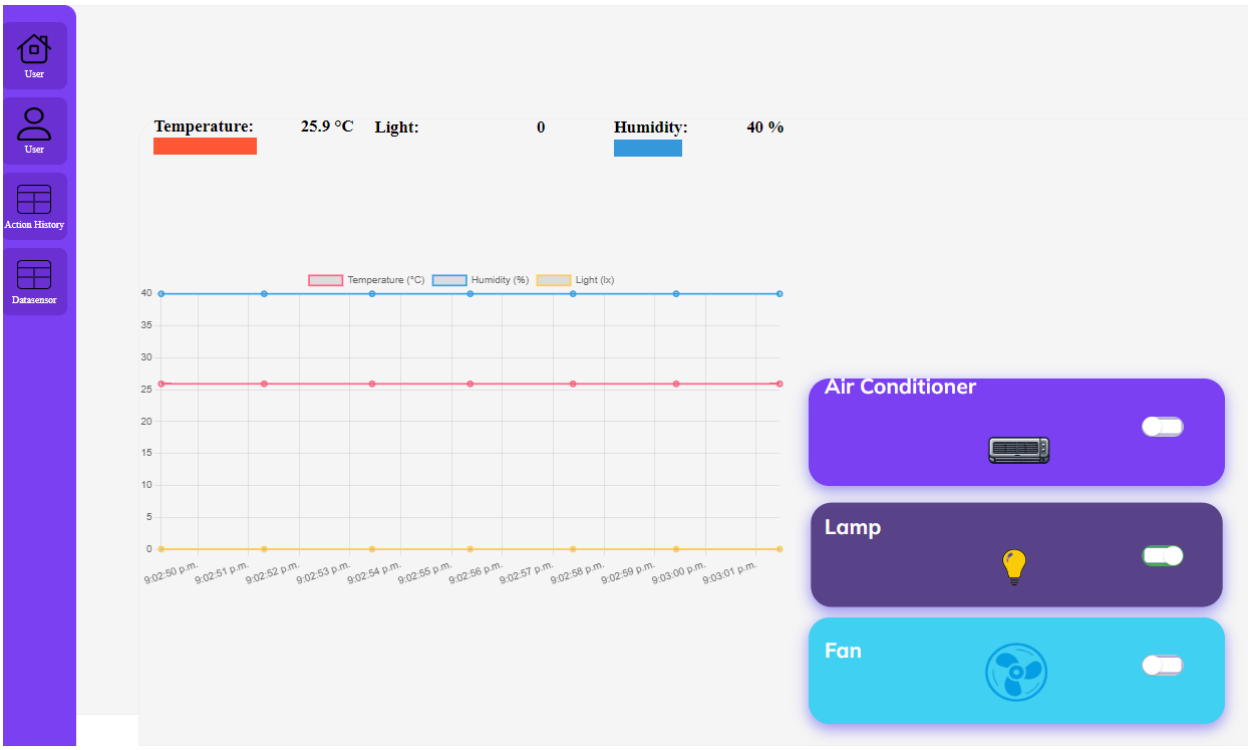
- Dữ liệu ánh sáng thu thập được sẽ được vi điều khiển sử dụng để điều chỉnh các thiết bị chiếu sáng trong không gian.
- Khi ánh sáng quá thấp, hệ thống sẽ tự động bật đèn, và ngược lại, khi ánh sáng đủ mạnh, đèn sẽ được tắt để tiết kiệm năng lượng.

c. Tính năng thông minh

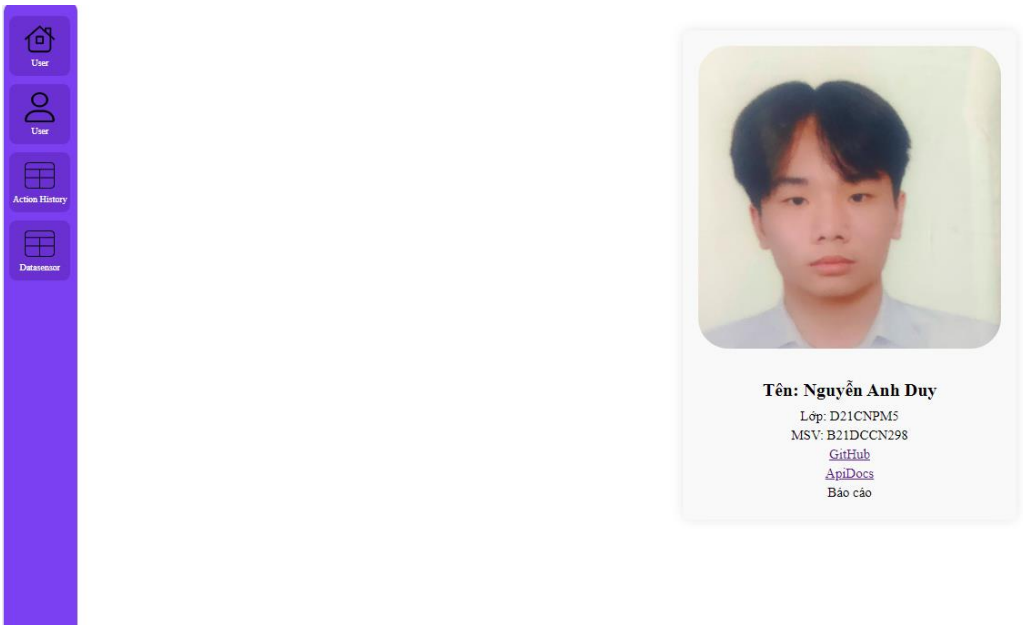
- **Bật quạt khi nhiệt độ quá cao**: Khi cảm biến DHT11 đo được nhiệt độ vượt quá 40°C, hệ thống sẽ tự động bật quạt để làm mát không gian. Hệ thống sẽ kiểm tra trạng thái của quạt trước khi ra lệnh bật để tránh trường hợp quạt đã được bật sẵn.
- **Bật điều hòa khi độ ẩm quá cao**: Nếu độ ẩm trong không gian vượt quá 70%, hệ thống sẽ tự động bật điều hòa để giảm độ ẩm, giúp môi trường trở nên thoáng mát hơn. Vi điều khiển sẽ đảm bảo điều hòa chỉ được bật khi cần thiết.
- **Bật đèn khi ánh sáng quá thấp**: Khi ánh sáng trong không gian giảm xuống dưới 300 lux, cảm biến ánh sáng sẽ kích hoạt hệ thống bật đèn để tăng cường ánh sáng trong phòng, đảm bảo đủ ánh sáng cho các hoạt động.

II. Giao diện

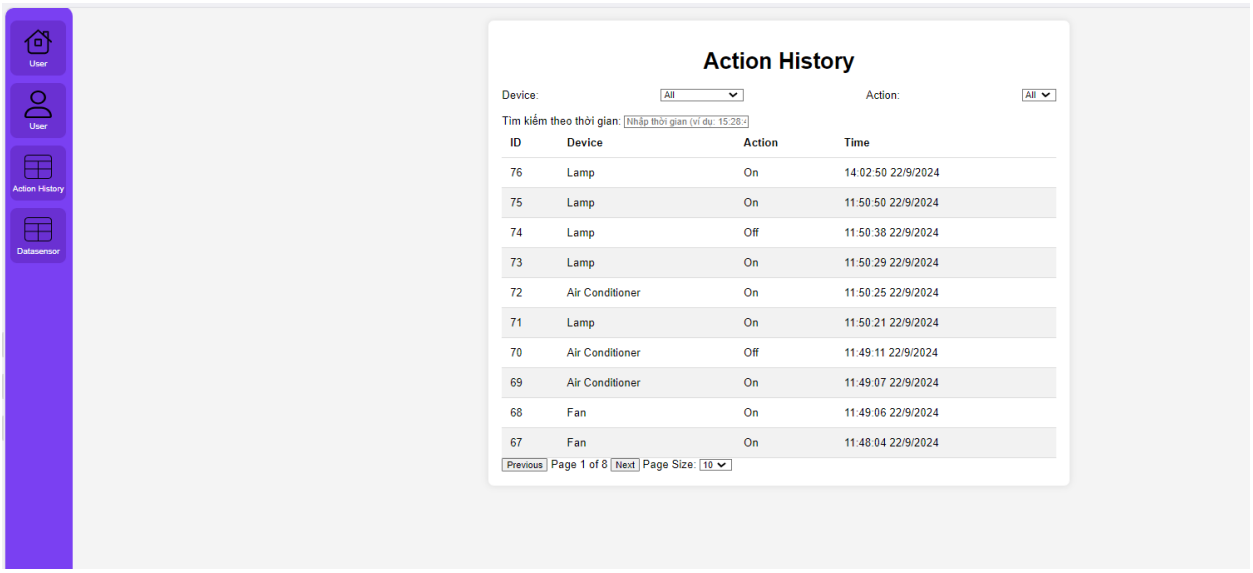
1. Giao diện trang chủ



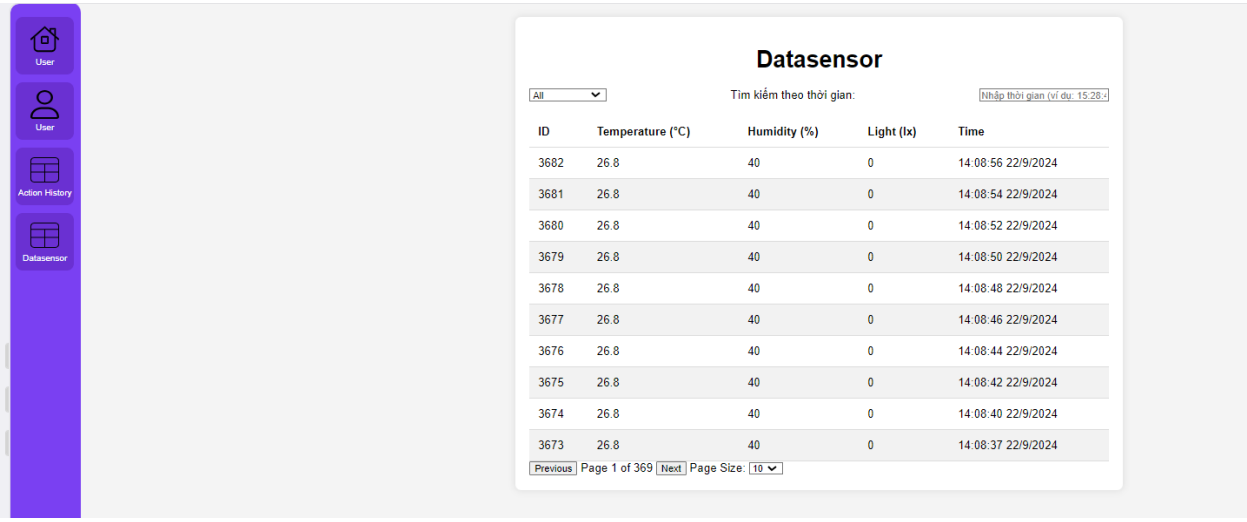
2. Giao diện trang cá nhân



3. Giao diện Action History



4. Giao diện Datasenser



III. Thiết kế hệ thống

1. Database

Bảng actor

	id	temperature	humidity	light	time
▶	1	32.4	61	4095	2024-09-22 09:37:15
	2	32.4	61	3776	2024-09-22 09:37:17
	3	32.4	61	3895	2024-09-22 09:37:19
	4	32.4	61	4051	2024-09-22 09:37:21
	5	32.4	61	3987	2024-09-22 09:37:23
	6	32.4	61	3887	2024-09-22 09:37:25
	7	32.4	61	4095	2024-09-22 09:37:27
	8	32.4	61	3543	2024-09-22 09:37:29
	9	32.4	61	3921	2024-09-22 09:37:31
	10	32.4	61	4095	2024-09-22 09:37:33
	11	32.4	61	4095	2024-09-22 09:37:35
	12	32.4	61	3953	2024-09-22 09:37:37

- Mục đích: Bảng này lưu trữ các thông tin liên quan đến hành động điều khiển các thiết bị trong hệ thống. Mỗi hành động được ghi lại khi có sự thay đổi trạng thái của thiết bị (bật, tắt, hoặc các trạng thái khác).
- Các trường trong bảng:

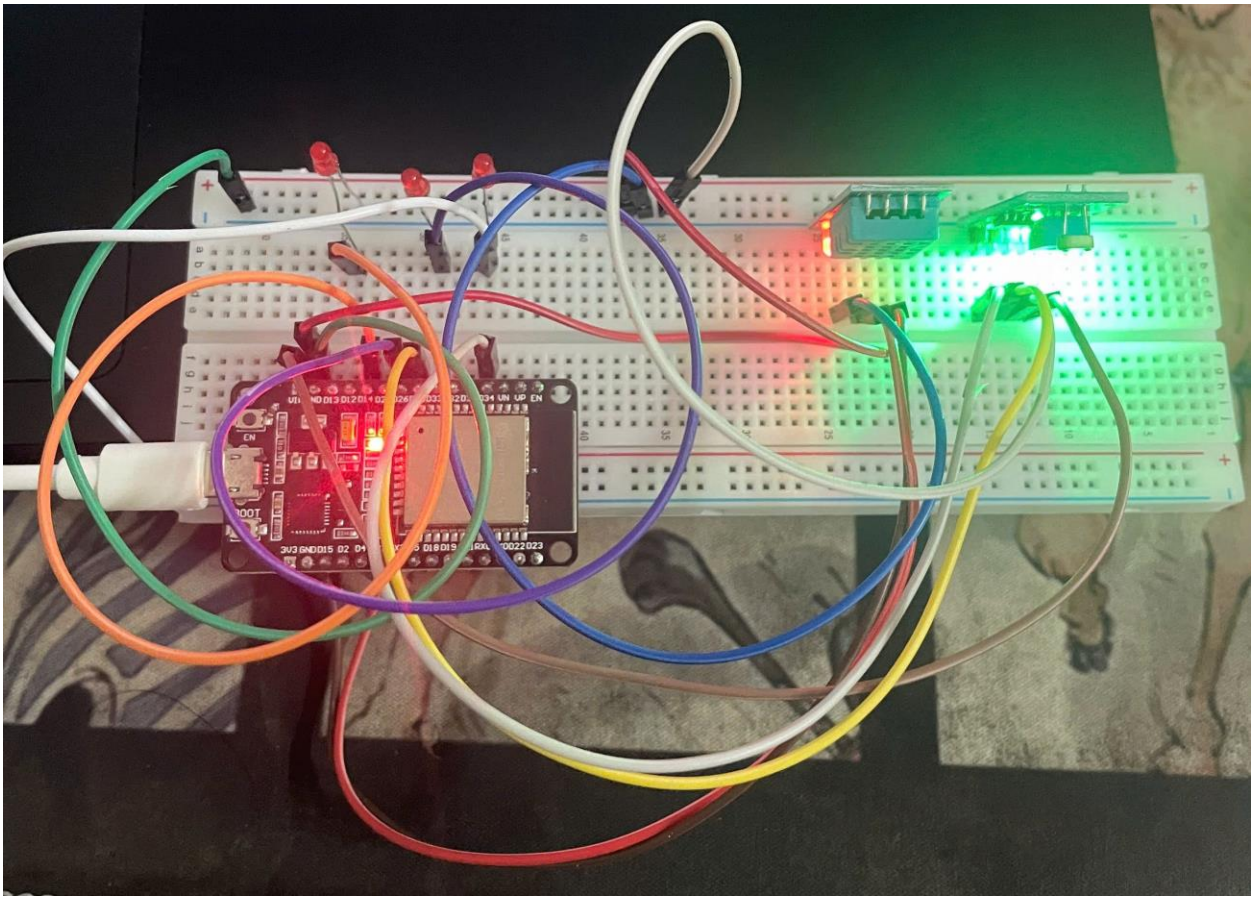
- id: Khóa chính của bảng, là giá trị số duy nhất để định danh mỗi hành động.
- device: Tên của thiết bị (ví dụ: quạt, đèn, điều hòa) đang được điều khiển.
- action: Hành động thực hiện trên thiết bị (ví dụ: "On", "Off").
- time: Thời gian hành động được thực hiện. Thời gian này được lưu theo định dạng thời gian UTC.

Bảng sensers

id	temperature	humidity	light	time
1	32.4	61	4095	2024-09-22 09:37:15
2	32.4	61	3776	2024-09-22 09:37:17
3	32.4	61	3895	2024-09-22 09:37:19
4	32.4	61	4051	2024-09-22 09:37:21
5	32.4	61	3987	2024-09-22 09:37:23
6	32.4	61	3887	2024-09-22 09:37:25
7	32.4	61	4095	2024-09-22 09:37:27
8	32.4	61	3543	2024-09-22 09:37:29
9	32.4	61	3921	2024-09-22 09:37:31
10	32.4	61	4095	2024-09-22 09:37:33
11	32.4	61	4095	2024-09-22 09:37:35
12	32.4	61	3953	2024-09-22 09:37:37

- Mục đích: Bảng này lưu trữ các dữ liệu cảm biến thu thập được từ hệ thống, bao gồm các thông tin về nhiệt độ, độ ẩm và mức ánh sáng từ các cảm biến trong môi trường.
- Các trường trong bảng:
    - id: Khóa chính của bảng, là giá trị số duy nhất để định danh mỗi lần ghi lại dữ liệu cảm biến.
    - temperature: Nhiệt độ đo được từ cảm biến nhiệt độ, được ghi lại dưới dạng số thập phân (ví dụ: 32.4°C).
    - humidity: Độ ẩm đo được từ cảm biến, được ghi dưới dạng phần trăm (ví dụ: 61%).
    - light: Mức ánh sáng đo được từ cảm biến ánh sáng, được ghi dưới dạng giá trị số (ví dụ: 4095 đại diện cho mức ánh sáng tối đa).
    - time: Thời gian ghi nhận dữ liệu cảm biến, được lưu theo định dạng thời gian UTC.

2. Mô tả chi tiết về hệ thống



a. Tổng quan hệ thống

- Hệ thống được xây dựng trên nền tảng ESP32 – một vi điều khiển phổ biến cho các ứng dụng IoT (Internet of Things), kết hợp với các cảm biến và thiết bị ngoại vi khác. Hệ thống này có chức năng thu thập dữ liệu từ các cảm biến nhiệt độ, độ ẩm, và ánh sáng, sau đó truyền dữ liệu đến máy chủ qua giao thức MQTT để giám sát và điều khiển các thiết bị như quạt, đèn LED.

b. Thành phần của hệ thống

- Vi điều khiển ESP32: Là trung tâm của hệ thống, có nhiệm vụ thu thập dữ liệu từ các cảm biến, xử lý dữ liệu và truyền dữ liệu qua giao thức MQTT. ESP32 còn có khả năng kết nối WiFi để giao tiếp với các thiết bị khác.
- Cảm biến nhiệt độ và độ ẩm DHT11: DHT11 được sử dụng để đo nhiệt độ và độ ẩm không khí. Cảm biến này truyền dữ liệu về nhiệt độ (°C) và độ ẩm (%) đến ESP32.
- Cảm biến ánh sáng: Đo cường độ ánh sáng trong môi trường. Dữ liệu từ cảm biến ánh sáng cho phép hệ thống nhận diện mức độ sáng để thực hiện các hành động tự động, chẳng hạn như bật/tắt đèn.
- Đèn LED: Được sử dụng để hiển thị trạng thái hệ thống hoặc có thể mô phỏng thiết bị điều khiển (ví dụ: quạt, đèn). Các đèn LED sẽ bật/tắt dựa trên điều kiện từ dữ liệu cảm biến hoặc các lệnh điều khiển từ máy chủ.

c. Cách thức hoạt động

- Thu thập dữ liệu cảm biến: Hệ thống thu thập dữ liệu từ cảm biến DHT11 và cảm biến ánh sáng. Dữ liệu này bao gồm nhiệt độ, độ ẩm, và mức ánh sáng trong thời gian thực.
- Xử lý dữ liệu và ra quyết định:
- Nếu nhiệt độ vượt quá một ngưỡng nhất định (ví dụ: 40°C), hệ thống tự động bật quạt hoặc đèn LED để làm mát không gian.
- Nếu độ ẩm quá cao (ví dụ: trên 70%), hệ thống tự động bật điều hòa hoặc thiết bị làm giảm độ ẩm.
- Nếu ánh sáng quá thấp, đèn LED sẽ tự động được bật để tăng cường ánh sáng.
- Truyền dữ liệu và điều khiển từ xa: Dữ liệu từ các cảm biến sẽ được truyền qua giao thức MQTT đến một máy chủ để giám sát. Người dùng có thể điều khiển từ xa (bật/tắt các thiết bị) thông qua giao diện web hoặc ứng dụng di động.

d. Sơ đồ kết nối

- ESP32 được kết nối với:
- Cảm biến DHT11 để nhận dữ liệu nhiệt độ và độ ẩm.
- Cảm biến ánh sáng để đo cường độ ánh sáng.
- Đèn LED để hiển thị trạng thái hoặc điều khiển mô phỏng thiết bị như quạt, điều hòa.

- Các thành phần này được nối với nhau trên một bảng mạch breadboard như trong hình ảnh. Tất cả các dây kết nối từ các chân I/O của ESP32 được nối với cảm biến và đèn LED thông qua các dây nhảy.

e. Kết nối và giao thức truyền dữ liệu

- Giao thức MQTT: Được sử dụng để truyền dữ liệu giữa ESP32 và máy chủ. MQTT là một giao thức nhẹ, phù hợp với các ứng dụng IoT, giúp hệ thống truyền dữ liệu nhanh chóng và tiết kiệm tài nguyên.
- WiFi: ESP32 sử dụng WiFi để kết nối với máy chủ qua mạng cục bộ. Từ đó, các lệnh điều khiển từ người dùng có thể được gửi về ESP32 để điều khiển các thiết bị (đèn LED, quạt) dựa trên dữ liệu cảm biến.

3. Luồng xử lý

a. Luồng thu thập dữ liệu cảm biến:

- Cảm biến: Hệ thống bao gồm các cảm biến đo nhiệt độ, độ ẩm, và ánh sáng. Các cảm biến này liên tục ghi nhận dữ liệu về điều kiện môi trường.
- ESP32: Dữ liệu thu thập từ các cảm biến sẽ được vi điều khiển ESP32 xử lý và gửi qua giao thức MQTT đến một máy chủ trung gian (MQTT broker).
- Máy chủ: Khi ESP32 gửi dữ liệu, MQTT broker sẽ nhận và chuyển tiếp đến server, nơi dữ liệu sẽ được lưu vào cơ sở dữ liệu MySQL.
- Lưu trữ: Dữ liệu được lưu trong bảng `sensors` của cơ sở dữ liệu, bao gồm các trường temperature, humidity, light, và time để ghi lại thời điểm nhận dữ liệu.

b. Luồng hiển thị dữ liệu cảm biến:

- Frontend: Người dùng có thể theo dõi trực tiếp dữ liệu từ cảm biến qua giao diện web. Khi trang web được tải, một yêu cầu API sẽ được gửi từ frontend đến server để lấy dữ liệu từ cơ sở dữ liệu.
- API Server: Server xử lý yêu cầu lấy dữ liệu thông qua endpoint `/api/sensor-data`, truy vấn cơ sở dữ liệu MySQL và trả về dữ liệu cảm biến mới nhất.
- Cập nhật giao diện: Dữ liệu nhiệt độ, độ ẩm, ánh sáng sẽ được hiển thị trên giao diện người dùng, bao gồm cả việc cập nhật biểu đồ trực quan để người dùng dễ dàng theo dõi xu hướng thay đổi theo thời gian.

c. Luồng điều khiển thiết bị:
   - Giao diện người dùng: Người dùng có thể điều khiển các thiết bị (quạt, đèn, điều hòa) qua các nút bật/tắt trên giao diện web.
   - MQTT: Khi người dùng nhấn nút bật/tắt, một lệnh điều khiển sẽ được gửi từ web đến ESP32 thông qua giao thức MQTT. Lệnh này bao gồm trạng thái của thiết bị (bật hoặc tắt) và được gửi qua topic `esp32/ledControl`.
   - ESP32: ESP32 nhận lệnh từ MQTT broker và thay đổi trạng thái của các thiết bị tương ứng.
   - Cập nhật trạng thái: Sau khi trạng thái của thiết bị được thay đổi, ESP32 sẽ gửi phản hồi về web thông qua topic `esp32/leds`, giúp giao diện web hiển thị đúng trạng thái thiết bị.

d. Luồng cảnh báo và tự động điều khiển:
   - Cảnh báo nhiệt độ cao: Khi nhiệt độ đo được vượt quá 40°C, hệ thống tự động bật quạt và hiển thị cảnh báo trên giao diện người dùng.
   - Cảnh báo độ ẩm cao: Khi độ ẩm vượt quá 70%, hệ thống sẽ tự động bật điều hòa và cảnh báo người dùng.
   - Cảnh báo ánh sáng thấp: Nếu ánh sáng giảm xuống dưới 300 lux, hệ thống sẽ tự động bật đèn và cảnh báo người dùng.

e. Luồng lưu trữ và hiển thị lịch sử hành động:
   - Lưu trữ hành động: Mỗi khi có hành động điều khiển thiết bị (bật/tắt), dữ liệu này sẽ được ghi lại vào cơ sở dữ liệu trong bảng `actions` với thông tin thiết bị, hành động và thời gian.
   - Hiển thị lịch sử: Người dùng có thể xem lại lịch sử điều khiển thiết bị qua giao diện web. Dữ liệu được truy xuất từ cơ sở dữ liệu thông qua API `/api/actions` và hiển thị dưới dạng bảng.

f. Luồng xử lý dữ liệu liên tục và cập nhật:
   - Hệ thống liên tục lấy dữ liệu từ các cảm biến và cập nhật lên giao diện mỗi giây, cho phép người dùng theo dõi thông tin thời gian thực về nhiệt độ, độ ẩm và ánh sáng mà không cần tải lại trang.
   - Biểu đồ trực quan sẽ cập nhật dữ liệu mới nhất và giữ lại tối đa 20 điểm dữ liệu để đảm bảo tính trực quan và hiệu suất của biểu đồ.

4. API Docs

a. API lấy dữ liệu cảm biến

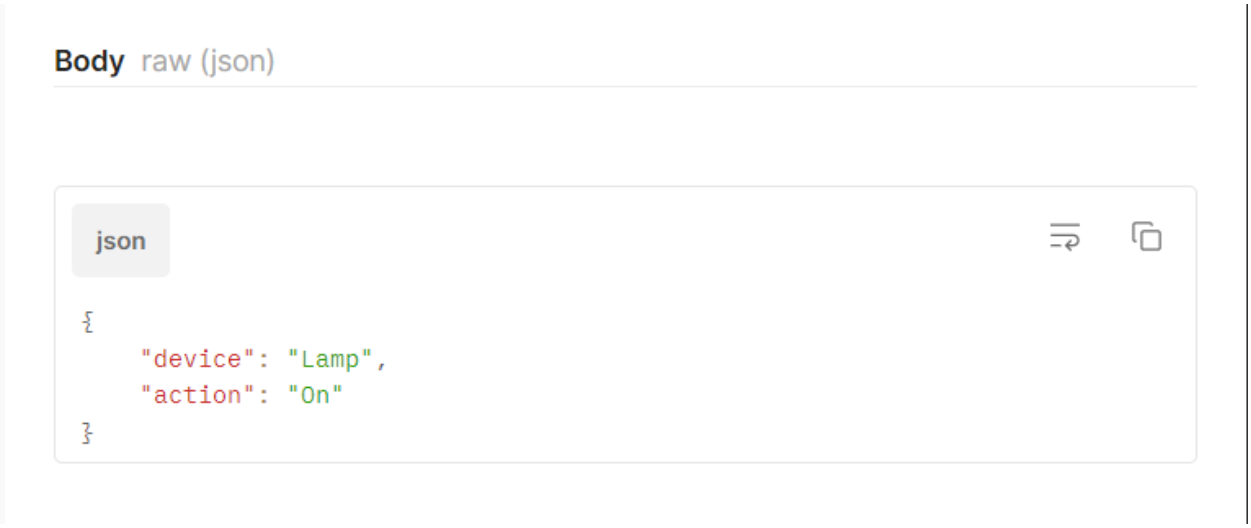
- Endpoint: GET /api/sensor-data
- Mô tả: Lấy tất cả dữ liệu cảm biến hiện tại, bao gồm nhiệt độ, độ ẩm và ánh sáng từ bảng sensors.
- Phản hồi:

b. API lấy lịch sử hành động

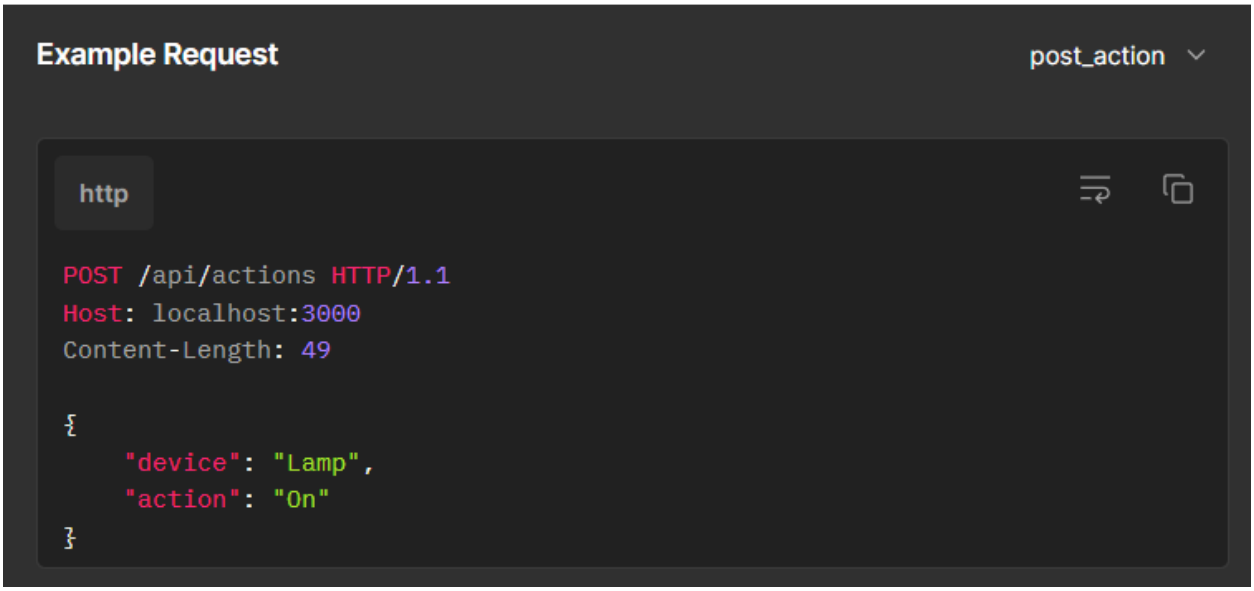
- Endpoint: GET /api/actions
- Mô tả: Lấy tất cả các hành động điều khiển thiết bị, bao gồm các thông tin về thiết bị, hành động thực hiện (bật/tắt) và thời gian.
- Phản hồi:

c. API điều khiển thiết bị

- Endpoint: POST /api/actions
- Mô tả: Gửi một hành động điều khiển thiết bị, như bật quạt, tắt đèn hoặc bật điều hòa.
- Request Body:
    - Content-Type: application/json
    - Request Body:



- Phản hồi:



IV. Source Code

1. Cấu trúc tổng quan

Hệ thống được xây dựng dựa trên mô hình client-server, trong đó:

- Client: Frontend được viết bằng HTML, CSS, và JavaScript, tương tác với backend thông qua các API và MQTT để nhận dữ liệu cảm biến và điều khiển các thiết bị (quạt, đèn, điều hòa).
- Server: Backend được xây dựng bằng Node.js với sự kết hợp của các thư viện như Express.js để quản lý API, MySQL để lưu trữ dữ liệu và MQTT để giao tiếp với các thiết bị IoT.

2. Đánh giá các thành phần chính

a. Backend (server.js):

- Sử dụng Express để triển khai các API, quản lý cơ sở dữ liệu MySQL và giao tiếp với MQTT broker.
- Quản lý MQTT:
  - Kết nối với MQTT broker, lắng nghe dữ liệu từ các cảm biến (nhiệt độ, độ ẩm, ánh sáng) và lưu trữ vào cơ sở dữ liệu `MySQL` thông qua API `/api/sensor-data` và `/api/actions`.
  - Tích hợp tốt MQTT để nhận và điều khiển thiết bị IoT thông qua các topic như `esp32/sensors`, `esp32/ledControl`, v.v.
- Quản lý MySQL:
  - Kết nối ổn định với MySQL, lưu trữ dữ liệu cảm biến và lịch sử hành động một cách hiệu quả. Việc triển khai các truy vấn SQL được thực hiện chính xác và phản hồi kịp thời các yêu cầu từ frontend.
- API:
  - Xây dựng tốt các API cho việc nhận và lưu dữ liệu từ cảm biến, cũng như lịch sử hành động điều khiển thiết bị.

b. Frontend (index.html, script.js, data.html, action.html):

- Giao diện người dùng:

    - Giao diện được xây dựng khá trực quan với các thành phần chính như thanh điều khiển (quạt, đèn, điều hòa) và biểu đồ hiển thị dữ liệu cảm biến theo thời gian.
    - Sử dụng `Chart.js` để hiển thị trực quan dữ liệu nhiệt độ, độ ẩm, ánh sáng theo thời gian là một lựa chọn hợp lý.
    - Mỗi thiết bị được điều khiển thông qua các nút bấm và trạng thái của thiết bị được cập nhật thông qua giao thức MQTT.
- Biểu đồ thời gian thực:
    - Biểu đồ cập nhật mỗi khi có dữ liệu mới từ MQTT và chỉ giữ lại 20 điểm dữ liệu gần nhất để đảm bảo hiệu suất.
- Cảnh báo tự động:
    - Cảnh báo tự động khi nhiệt độ vượt quá 40°C, độ ẩm vượt 70%, và ánh sáng dưới 300 lux. Cảnh báo được thực hiện một cách hợp lý, chỉ hiển thị một lần mỗi khi ngưỡng bị vượt qua, và điều khiển thiết bị tương ứng (quạt, điều hòa, đèn) bật tự động.
- Pagination:
    - Sử dụng tốt pagination để hiển thị dữ liệu lịch sử trên trang `data.html` và `action.html`, cho phép người dùng duyệt qua các dữ liệu cảm biến và lịch sử điều khiển một cách dễ dàng.

3. Đánh giá code Arduino

a. Cấu hình kết nối

- WiFi: ESP32 sẽ kết nối với mạng WiFi được cấu hình thông qua biến `ssid` và `password`.
- MQTT broker: Kết nối với broker tại địa chỉ IP `192.168.1.16` qua cổng 1883.

b. Thu thập dữ liệu cảm biến

- DHT11: Đo nhiệt độ và độ ẩm thông qua chân `DHTPIN 4`. Sau đó, giá trị được chuyển đổi thành chuỗi để gửi qua MQTT.
- Cảm biến ánh sáng: Đọc tín hiệu analog từ cảm biến ánh sáng kết nối với chân `lightSensorPin (34)`. Giá trị đọc được được đảo ngược để ánh sáng càng mạnh thì giá trị càng lớn (4095 là tối đa).

c. Giao tiếp qua MQTT

- Đăng ký topic và nhận lệnh điều khiển:

- Topic `esp32/ledControl` nhận lệnh điều khiển các đèn LED.
- Mỗi khi nhận lệnh qua MQTT, ESP32 sẽ kiểm tra các khóa trong dữ liệu JSON để bật/tắt các đèn tương ứng (LED 1, LED 2, LED 3).
- Phản hồi trạng thái thiết bị: Sau khi xử lý lệnh bật/tắt, ESP32 gửi phản hồi về trạng thái hiện tại của các LED thông qua topic `esp32/leds`.

d. Gửi dữ liệu cảm biến qua MQTT

- Sau khi đọc dữ liệu từ các cảm biến (nhiệt độ, độ ẩm, ánh sáng), ESP32 tạo một đối tượng JSON chứa giá trị của các cảm biến.
- Dữ liệu cảm biến này được gửi qua MQTT đến topic `esp32/sensors` mỗi 2 giây.

e. Kết nối lại với MQTT broker

- Trong hàm `loop()`, nếu kết nối MQTT bị mất, ESP32 sẽ tự động thử kết nối lại qua hàm `reconnect()`, đảm bảo hệ thống luôn duy trì kết nối với broker.

f. Điều khiển đèn LED

- ESP32 có thể nhận lệnh điều khiển LED từ MQTT, kiểm tra lệnh `ON` hoặc `OFF` cho từng đèn LED (ledPin1, ledPin2, ledPin3), và gửi phản hồi về trạng thái hiện tại của các LED qua MQTT.

V. Kết quả thu được

1. Kết nối và giao tiếp

- Hệ thống kết nối thành công với mạng WiFi và MQTT broker, đảm bảo việc thu thập dữ liệu và điều khiển thiết bị được thực hiện thông qua internet.
- MQTT được sử dụng để gửi dữ liệu cảm biến và nhận lệnh điều khiển thiết bị, đảm bảo phản hồi thời gian thực.

2. Thu thập và xử lý dữ liệu cảm biến

- Hệ thống đã thu thập và xử lý chính xác dữ liệu nhiệt độ, độ ẩm (DHT11) và ánh sáng từ các cảm biến.
- Cảm biến ánh sáng được xử lý đặc biệt để đảo ngược giá trị, giúp hệ thống phản hồi theo yêu cầu khi ánh sáng thay đổi.

3. Điều khiển thiết bị từ xa

- Người dùng có thể điều khiển các thiết bị (LED) từ xa thông qua các lệnh gửi qua MQTT. Hệ thống thực hiện chính xác lệnh bật/tắt đèn LED và gửi lại trạng thái của thiết bị về máy chủ.

- Các thiết bị phản hồi ngay lập tức sau khi nhận được lệnh điều khiển từ MQTT.

4. **Tự động hóa và cảnh báo**

- Hệ thống tự động bật quạt khi nhiệt độ vượt quá 40°C, bật điều hòa khi độ ẩm vượt 70%, và bật đèn khi ánh sáng quá thấp, giúp tự động điều chỉnh môi trường theo thời gian thực.

5. **Tính ổn định và mở rộng**

- Hệ thống có cơ chế tự động kết nối lại khi mất kết nối với MQTT broker, đảm bảo tính liên tục trong việc giám sát và điều khiển.
- Cơ sở hạ tầng có thể mở rộng dễ dàng để tích hợp thêm các cảm biến và thiết bị mới mà không ảnh hưởng đến hệ thống hiện tại.