HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA KỸ THUẬT ĐIỆN - ĐIỆN TỬ I

BÁO CÁO BÀI TẬP LỚN

MẠNG CẢM BIẾN

NỘI DUNG:

THIẾT KẾ HỆ THỐNG IOT

Giảng viên hướng dẫn : Thầy Nguyễn Quốc Uy

Sinh viên thực hiện : Nguyễn Minh Trung – B21DCDT226

MỤC LỤC

DANH SÁCH HÌNH ẢNH

DANH SÁCH BẢNG

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy Nguyễn Quốc Uy vì đã tận tâm hướng dẫn chúng em trong suốt quá trình thực hiện bộ môn mạng cảm biến.. Sự hỗ trợ tận tình của Thầy đã tạo điều kiện thuận lợi để em thực hiện và hoàn thiện sản phẩm. Em cũng xin cảm ơn các anh chị và các bạn cùng khóa đã luôn sẵn sàng hỗ trợ, chia sẻ và hợp tác trong quá trình thực hiện sản phẩm. Sự giúp đỡ và hợp tác của các bạn đã góp phần quan trọng vào việc hoàn thành sản phẩm đúng tiến độ. Mặc dù đã nỗ lực khắc phục khó khăn và thử thách, em nhận thấy rằng sản phẩm còn nhiều thiếu sót về nội dung và hình thức do kiến thức còn hạn chế. Em rất mong nhận được sự thông cảm và những góp ý quý báu từ quý Thầy Cô để cải thiện và hoàn thiện các mô hình trong tương lai. Một lần nữa em xin chân thành cảm ơn.

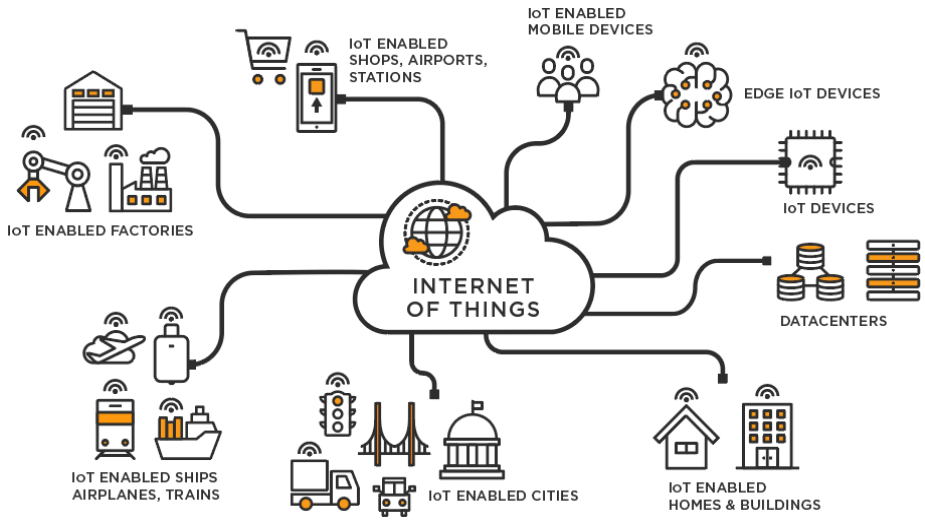
Chương 1. TỔNG QUAN VỀ CÔNG NGHỆ

1.1. Giới thiệu về IoT

1.1.1. Định nghĩa IoT

IoT (Internet of Things), hay còn gọi là Internet vạn vật, là một hệ thống liên kết các thiết bị điện tử, máy móc, và các chủ thể thông minh khác vào một mạng internet hoặc mạng nội bộ. Các thiết bị này có khả năng thu thập, chia sẻ dữ liệu và tương tác với nhau mà không cần sự can thiệp của con người. Sự phát triển của IoT bắt đầu từ những năm 1980 với sự ra đời của các thiết bị đầu tiên có khả năng kết nối. Qua các thập kỷ, IoT đã phát triển mạnh mẽ nhờ vào sự tiến bộ của công nghệ không dây, vi cơ điện tử và internet. Ngày nay, IoT được ứng dụng rộng rãi trong nhiều lĩnh vực như:

- Ứng dụng trong gia đình: Các thiết bị gia dụng thông minh phổ biến như đèn, tủ lạnh, và hệ thống an ninh có thể được điều khiển từ xa qua điện thoại di động.
- Ứng dụng trong lĩnh vực y tế: Các thiết bị y tế thông minh giúp theo dõi sức khỏe bệnh nhân từ xa, cung cấp dữ liệu liên tục cho bác sĩ và có thể cảnh báo kịp thời khi phát hiện bất thường và kịp thời xử lý khắc phục bất thường.
- Ứng dụng trong lĩnh vực công nghiệp: IoT giúp tối ưu hóa quy trình sản xuất, theo dõi và bảo trì thiết bị, và quản lý năng lượng hiệu quả. Các cảm biến thông minh trong nhà máy có thể giám sát máy móc và tự động thông báo về tình trạng cho cơ quan kiểm soát và bảo trì hệ thống, làm kiểm soát hiệu quả hơn.



Hình 1.1: ***Ứng dụng của Internet vạn vật trong cuộc sống***

1.1.2. Các thành phần chính của IoT

Các thiết bị cảm biến

Các thiết bị và cảm biến thông minh là thành phần cốt lõi của IoT, bao gồm các thiết bị vật lý có khả năng thu thập, xử lý và truyền dữ liệu. Ví dụ cảm biến nhiệt độ đo nhiệt độ môi trường và gửi dữ liệu đến bộ điều khiển hoặc máy chủ, trong khi cảm biến độ ẩm đo độ ẩm không khí và cung cấp thông tin này cho hệ thống. Ngoài ra, các thiết bị gia dụng thông minh như đèn, tủ lạnh, máy giặt và nhiều thiết bị khác có thể được điều khiển từ xa qua internet, tăng cường tiện ích và hiệu quả cho người sử dụng.

Hạ tầng mạng

Hạ tầng mạng cung cấp khả năng kết nối giữa các thiết bị IoT với nhau và với internet, sử dụng các công nghệ kết nối phổ biến như Wi-Fi, Bluetooth, ZigBee và LoRaWAN. Wi-Fi được sử dụng rộng rãi trong nhà và văn phòng để kết nối các thiết bị với mạng internet. Bluetooth thường được sử dụng cho kết nối ngắn hạn giữa các thiết bị cá nhân như điện thoại, đồng hồ thông minh. ZigBee là một giao thức không dây dành cho các ứng dụng yêu cầu băng thông thấp và tiêu thụ năng lượng thấp, được ứng dụng trong nhà thông minh. LoRaWAN là giao thức mạng diện rộng không dây cho các ứng dụng IoT yêu cầu truyền dữ liệu xa và tiêu thụ năng lượng thấp.

Bộ điều khiển và quản lý

Bộ điều khiển và quản lý chịu trách nhiệm thu thập dữ liệu từ các cảm biến, xử lý dữ liệu và thực hiện các hành động cần thiết dựa trên dữ liệu thu thập được. Các thành phần chính bao gồm gateway (cổng kết nối), thiết bị trung gian kết nối các thiết bị IoT với mạng internet, thực hiện việc truyền tải dữ liệu từ thiết bị đến các nền tảng IoT, và bộ điều khiển trung tâm, điều khiển và quản lý các thiết bị IoT, thực hiện các lệnh điều khiển từ người sử dụng nhằm các mục đích dựa trên dữ liệu thu thập được.

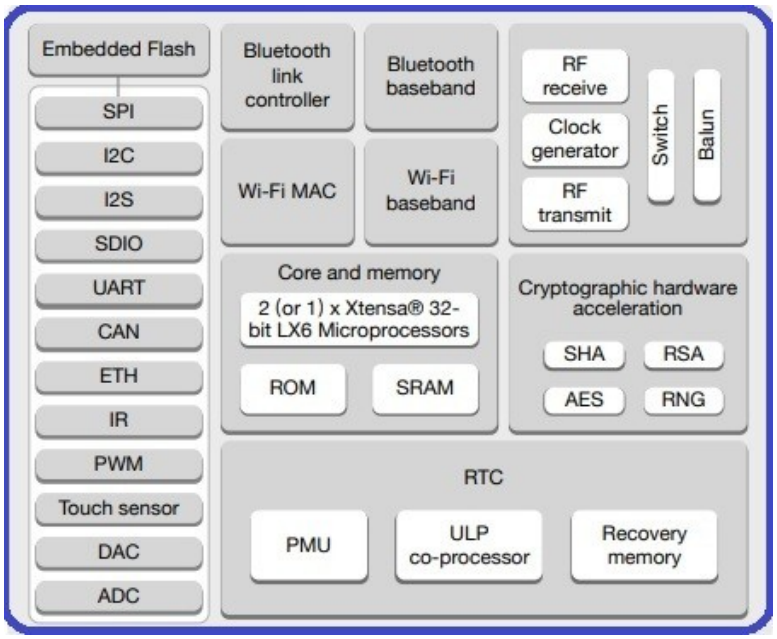
Lưu trữ và phân tích dữ liệu

Dữ liệu từ các thiết bị IoT sau khi đã thu thập cần được lưu trữ và phân tích để tạo ra các giá trị thực tiễn. Các thành phần chính bao gồm cơ sở dữ liệu, nơi lưu trữ, tổng hợp dữ liệu thu thập từ các thiết bị IoT, ví dụ như: InfluxDB, MongoDB và MySQL databases, và công cụ phân tích, được sử dụng để phân tích dữ liệu và đưa ra các báo cáo, như các công cụ máy học, trí tuệ nhân tạo và các công cụ phân tích dữ liệu lớn.

1.2. Công Nghệ Phần Cứng

1.2.1. Giới Thiệu Về Vi Điều Khiển ESP32

a. Tổng quan về vi điều khiển esp32



Hình 1.1: ***Hình ảnh sơ đồ khối chức năng ESP32***

ESP32 là một vi điều khiển giá rẻ, năng lượng thấp có hỗ trợ WiFi và dual-mode Bluetooth (tạm dịch: Bluetooth chế độ kép). Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 ở cả hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc antenna tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng. ESP32 được chế tạo và phát triển bởi Espressif Systems, và được sản xuất bởi TSMC bằng cách sử dụng công nghệ 40 nm. ESP32 là sản phẩm kế thừa từ vi điều khiển ESP8266.

b. Cấu hình của ESP32

Bảng 1.1: Bảng thông số kỹ thuật của ESP32

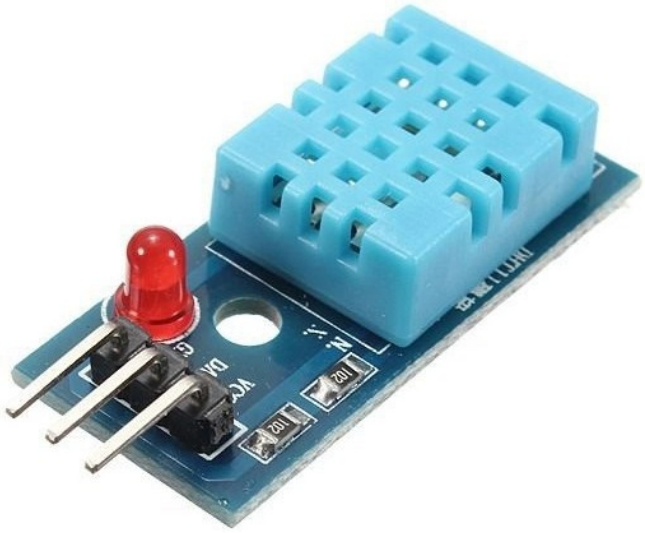
Danh Mục	Chi Tiết
CPU	Xtensa Dual-Core LX6 microprocessor chạy hệ 32-bit. Tốc độ xử lý 160 MHz lên đến 240 MHz. Tốc độ xung nhịp đọc flash chip 40 MHz – 80 MHz (tùy chỉnh khi lập trình)
RAM	520 KBytes SRAM liền chip – (8 KBytes RAM RTC tốc độ cao – 8 KBytes RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep))
Giao tiếp không dây	Wi-Fi: 802.11 b/g/n/e/i Bluetooth: v4.2 BR/EDR và BLE
Giao Tiếp	DACs 8-bit (digital to analog): 2 cổng ADC 12-bit (Analog to Digital): 16 cổng I2C: 2 cổng. UART: 3 cổng SPI: 3 cổng (1 cổng cho chip FLASH). I2S: 2 cổng SD card / SDIO / MMC host. Slave (SDIO / SPI) Giao diện Ethernet MAC với DMA chuyên dụng, hỗ trợ IEEE 1588 CAN bus 2.0. IR (TX/RX). Băm xung PWM (tất cả các chân)
Cảm Biến Tích Hợp	1 cảm biến Hall (cảm biến từ trường). 1 cảm biến đo nhiệt độ Cảm biến chạm (điện dung) với 10 đầu vào khác nhau
Nguồn Điện Hoạt Động	Nhiệt độ hoạt động: -40 - +85°C Điện áp hoạt động: 2.2 - 3.6V Số cổng GPIO: 34

c. Điểm mạnh của ESP32 so với các dòng khác

Có thể nói ESP32 là sự nâng cấp hoàn hảo của ESP8266, với ESP8266 phù hợp với các dự án nhỏ và tiết kiệm chi phí. ESP32 lại phù hợp với các dự án phức tạp hơn, tốc độ xử lý cao hơn và tích hợp nhiều ngoại vi mạnh mẽ hơn. ESP8266 là 17 chân GPIO, ADC độ phân giải 10 bit, 8 kênh PWM mềm trong khi đó ESP32 có tới 30/36 chân GPIO, 18 kênh ADC độ phân giải 12-bit, 16 kênh PWM mềm, Touch Sensor, Hall Effect Sensor, Ethernet MAC Interface, Cảm biến nhiệt độ được tích hợp sẵn.

Về bộ nhớ ESP32 có thêm 4MB External Flash và 520 KBytes SRAM (static random access memory) trong đó 8 KBytes RAM RTC tốc độ cao – 8 KBytes RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep). ESP32 hỗ trợ Bluetooth 4.2 và BLE (Bluetooth Low Energy). Việc hỗ trợ cả bluetooth khiến ESP32 có thể tương tác với các thiết bị như là bàn phím, chuột, điện thoại khi mà không có wifi. Ultra Low Power giải quyết vấn đề năng lượng cho ESP bởi vì sử dụng WiFi sẽ rất tốn điện đặc biệt khi chúng ta sử dụng pin phải tính toán rất kĩ.

1.2.2. Cảm Biến nhiệt độ, độ ẩm DHT11



Hình 1.1: ***Hình ảnh cảm biến nhiệt độ, độ ẩm DHT11***

a. Thông số kỹ thuật

Cảm biến DHT11 có thể hoạt động ở dải điện áp từ 3.3V đến 5V. Cho phép cảm biến tương thích với nhiều dòng vi điều khiển và hệ thống khác nhau, từ các vi điều khiển có điện áp thấp như ESP32 cho đến các hệ thống với điện áp cao hơn. Việc sử dụng điện áp trong khoảng này đảm bảo cảm biến hoạt động ổn định và chính xác.

Cảm biến DHT11 có khả năng đo độ ẩm trong khoảng từ 20% đến 90% RH (độ ẩm tương đối). Cho phép cảm biến được sử dụng trong nhiều môi trường khác nhau, từ các khu vực khô đến những nơi có độ ẩm cao. Dải đo này không phù hợp các điều kiện độ ẩm rất cao hoặc rất thấp, cần lưu ý khi chọn cảm biến cho các ứng dụng đặc biệt.

DHT11 có khả năng đo nhiệt độ trong khoảng từ 0°C đến 50°C. Dải đo này phù hợp với hầu hết các ứng dụng trong điều kiện môi trường bình thường. Nhưng nếu ứng dụng yêu cầu đo nhiệt độ ở ngoài dải này, người sử dụng có thể cần cân nhắc thay đổi một cảm biến khác có khả năng đo nhiệt độ trong phạm vi rộng hơn với dải rộng hơn.

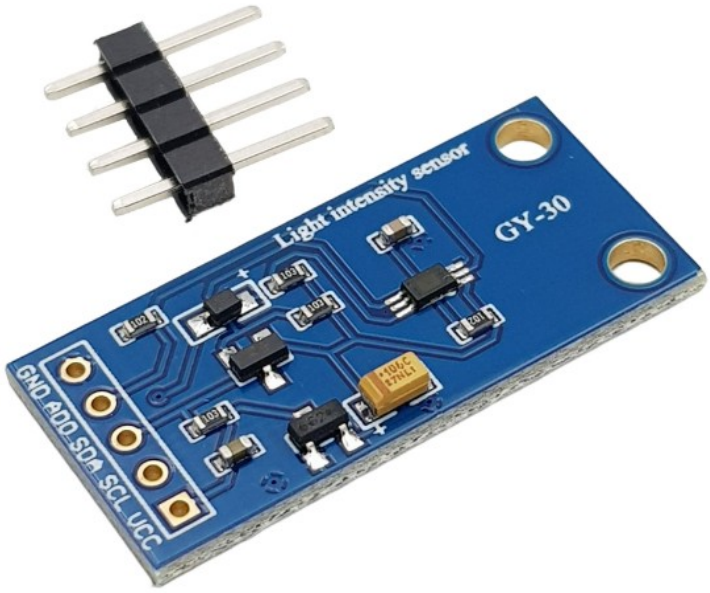
Độ chính xác của cảm biến DHT11 đối với đo độ ẩm là ±5% RH. Điều này có nghĩa là giá trị đo được có thể sai lệch lên hoặc xuống 5% so với giá trị thực tế. Mặc dù đây không phải là chỉ số cao nhất trong các cảm biến độ ẩm, nhưng với độ chính xác này đủ để người dùng ứng dụng cho nhiều ứng dụng yêu cầu đo độ ẩm cơ bản.

b. Thông số các chân

Bảng 1.1: Bảng thông số các chân cảm biến nhiệt độ DHT11

Chân	Ký hiệu	Mô tả	Giá trị
1	VCC	Nguồn cung cấp	3.3V - 5V
2	Data	Truyền dữ liệu	
3	NC	Không kết nối(Not Connected)	
4	GND	Đất	0V

1.2.3. Cảm biến ánh sáng BH1750



Hình 1.1: ***Hình ảnh cảm biến ánh sáng BH1750***

a. Thông số kỹ thuật

Cảm biến BH1750 hoạt động với điện áp từ 2.4V đến 3.6V, phù hợp cho nhiều dòng vi điều khiển và hệ thống khác nhau, bao gồm cả các thiết bị nhúng có điện áp thấp. Dải đo ánh sáng của BH1750 thường từ 0 đến 65535 lux, cho phép đo được cả môi trường sáng mạnh và môi trường tối. Độ chính xác của cảm biến BH1750 có thể đạt ±10% trong điều kiện lý tưởng, phù hợp cho các ứng dụng cơ bản đo độ sáng. Cảm

biến BH1750 thường có tần số lấy mẫu cao, từ 50 Hz đến 200 Hz, thích hợp cho các ứng dụng yêu cầu độ phản hồi nhanh với môi trường ánh sáng thay đổi nhanh. Kích thước nhỏ gọn, dễ tích hợp trong các dự án IoT và các hệ thống có không gian hạn chế.

b. Thông số các chân BH1750

Bảng 1.1: Bảng thông số các chân cảm biến ánh sáng BH1750

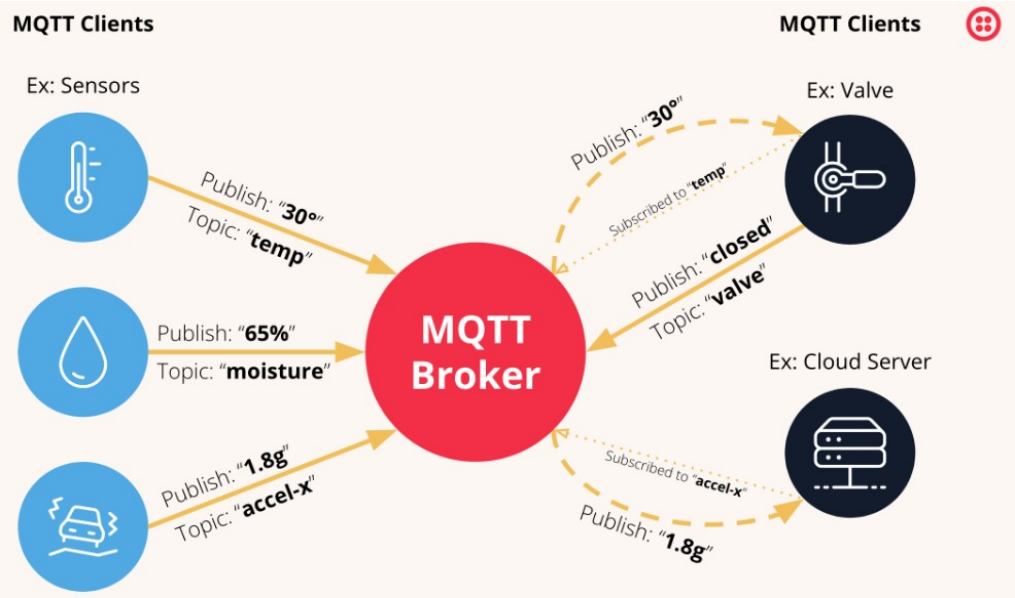
Chân	Ký hiệu	Mô tả	Giá trị
1	VCC	Nguồn cung cấp	2.4V – 3.6V
2	GND	Chân nối đất	
3	SCL	Chân Clock truyền tín hiệu đồng bộ	
4	SDA	Chân truyền dữ liệu cảm biến	
5	ADDR	Chân chọn địa chỉ I2C 0x23 0x5C	 GND VCC

1.3. Công nghệ phần mềm

1.3.1. Giới thiệu về MQTT (Message Queuing Telemetry Transport)

a. Đặc điểm của MQTT

MQTT là một giao thức truyền thông nhẹ được thiết kế đặc biệt cho các hệ thống IoT (Internet of Things), nơi mà các thiết bị cần trao đổi thông tin một cách nhanh chóng và tiết kiệm tài nguyên. MQTT sử dụng mô hình **publish/subscribe**, nghĩa là các thiết bị có thể gửi (publish) và nhận (subscribe) các thông điệp thông qua một máy chủ trung gian gọi là **broker**. Điều này khác với mô hình client-server truyền thống như HTTP, vì không có sự kết nối trực tiếp giữa người gửi và người nhận.



Hình 1.1: ***Giới thiệu về Mqtt***

Các đặc điểm nổi bật của MQTT:

- **Nhẹ (Lightweight)**: MQTT được tối ưu để sử dụng băng thông và tài nguyên hệ thống ít nhất có thể, phù hợp với các thiết bị có khả năng tính toán hạn chế như cảm biến, thiết bị di động, hay các thiết bị IoT.
- **Kết nối lâu dài**: MQTT hỗ trợ các kết nối liên tục, cho phép thiết bị có thể duy trì kết nối với broker trong thời gian dài mà không phải kết nối lại, tiết kiệm năng lượng cho các thiết bị IoT sử dụng pin.
- **Cơ chế giữ kết nối (Last Will and Testament - LWT)**: Khi một thiết bị bị mất kết nối đột ngột, MQTT broker có thể gửi thông điệp cuối cùng được cấu hình trước để thông báo về tình trạng mất kết nối của thiết bị.
- **Bảo mật**: MQTT hỗ trợ bảo mật thông qua cơ chế **SSL/TLS**, đảm bảo dữ liệu được truyền an toàn và bảo mật qua mạng. Ngoài ra, MQTT có thể sử dụng cơ chế **Xác thực người dùng** (User Authentication) để kiểm soát truy cập vào broker.

b. Cấu trúc của MQTT

1. **Broker**: Là máy chủ trung gian chịu trách nhiệm nhận các thông điệp từ các thiết bị gửi (publisher) và phân phối chúng cho các thiết bị nhận (subscriber) dựa trên chủ đề (topic).
2. **Publisher**: Thiết bị gửi thông điệp đến broker theo một chủ đề (topic) cụ thể.

3. **Subscriber**: Thiết bị nhận thông điệp từ broker sau khi đăng ký theo dõi (subscribe) một hoặc nhiều chủ đề (topic) nhất định.
4. **Topic**: Các thông điệp trong MQTT được phân loại và gửi đi theo các **chủ đề** (topics). Các topic có cấu trúc dạng cây (hierarchical structure), ví dụ: sensors/temperature/room1. Điều này giúp tổ chức dữ liệu và dễ dàng quản lý các loại thông tin khác nhau.
5. **Message**: Là các thông điệp chứa dữ liệu được gửi đi từ publisher tới broker, bao gồm phần payload (nội dung thông điệp) và header (các thông tin về giao thức và chủ đề).

c. Ưu điểm của MQTT trong hệ thống IoT

1. **Nhẹ và tiết kiệm băng thông**: Nhờ kích thước tin nhắn nhỏ và khả năng hoạt động tốt trên các mạng có độ trễ cao hoặc không ổn định, MQTT rất lý tưởng cho các thiết bị IoT.
2. **Tối ưu cho mạng không ổn định**: MQTT có thể duy trì kết nối ngay cả khi mạng gặp sự cố, nhờ vào khả năng giữ kết nối lâu dài và tái kết nối tự động.
3. **Bảo mật và mở rộng**: Với hỗ trợ SSL/TLS và xác thực, MQTT có thể triển khai trong các hệ thống yêu cầu bảo mật cao như giám sát y tế hoặc hệ thống tài chính.

1.3.2. Giới thiệu về Spring Boot

a. Tổng quan về Spring Boot

Spring Boot là một framework mã nguồn mở được phát triển bởi Pivotal Software, được xây dựng dựa trên nền tảng của Spring Framework. Mục tiêu của Spring Boot là đơn giản hóa quá trình phát triển ứng dụng Java bằng cách cung cấp một cách tiếp cận tự động hóa và dễ dàng cấu hình cho các ứng dụng web, microservices, và RESTful APIs. Spring Boot cho phép lập trình viên xây dựng và triển khai ứng dụng một cách nhanh chóng và dễ dàng mà không cần phải cấu hình quá nhiều.

Hình 1.1: ***Giới thiệu về Spring Boot***

b. Tính năng nổi bật của Spring Boot

Khởi tạo nhanh: Spring Boot cung cấp các cấu hình mặc định hợp lý và một bộ công cụ khởi tạo dự án thông qua Spring Initializr, giúp lập trình viên dễ dàng bắt đầu với dự án mới.

Tự động cấu hình: Spring Boot tự động cấu hình ứng dụng dựa trên các thư viện có sẵn trong classpath, giúp giảm thiểu công việc cấu hình thủ công.

Sử dụng cấu hình kiểu convention over configuration: Spring Boot sử dụng quy tắc "cấu hình theo quy ước" (convention over configuration) để giảm thiểu các cấu hình cần thiết, giúp lập trình viên tập trung vào logic của ứng dụng.

Tích hợp dễ dàng với các công nghệ khác: Spring Boot hỗ trợ tích hợp với nhiều công nghệ khác như Spring Data, Spring Security, Hibernate, và các thư viện bên thứ ba khác, giúp xây dựng ứng dụng dễ dàng và nhanh chóng.

Hỗ trợ microservices: Spring Boot rất phù hợp cho phát triển microservices nhờ khả năng triển khai độc lập và tích hợp với các giải pháp như Spring Cloud.

Cung cấp công cụ quản lý và giám sát: Spring Boot Actuator cung cấp các endpoint để giám sát và quản lý ứng dụng, giúp theo dõi tình trạng và hiệu suất của hệ thống.

c. Ứng dụng của Spring Boot

Trong hệ thống thu thập dữ liệu cảm biến, Spring Boot đóng vai trò là backend, xử lý việc thu thập, lưu trữ và cung cấp dữ liệu. Các ứng dụng cụ thể của Spring Boot bao gồm:

Thu thập dữ liệu từ các thiết bị cảm biến: Spring Boot sử dụng các API hoặc giao thức như MQTT để nhận dữ liệu từ các cảm biến nhiệt độ, độ ẩm và ánh sáng.

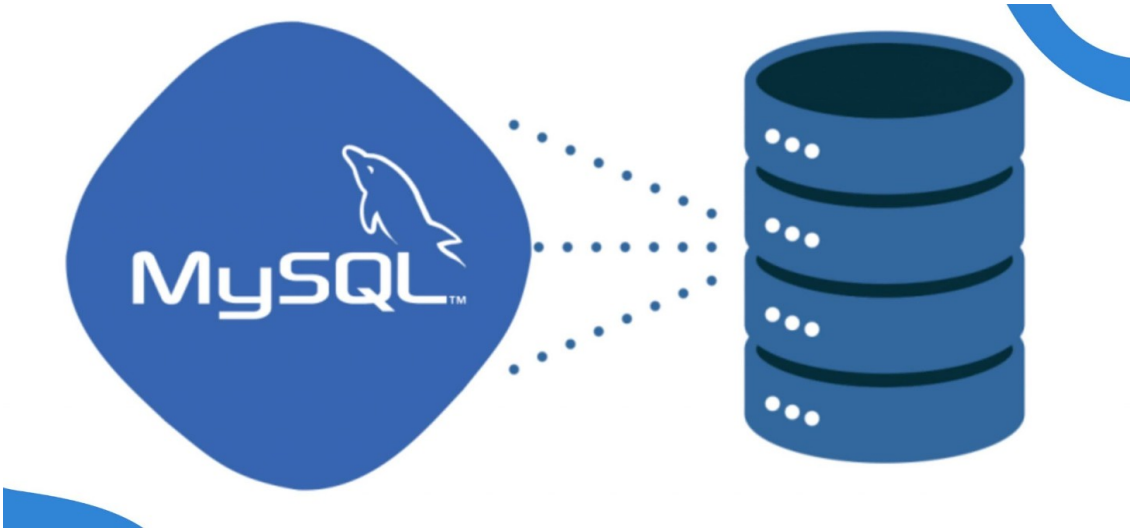
Lưu trữ dữ liệu vào cơ sở dữ liệu MySQL: Thông qua Spring Data JPA, Spring Boot có thể lưu dữ liệu cảm biến vào MySQL mà không cần viết câu lệnh SQL trực tiếp.

Cung cấp API cho front-end: Spring Boot tạo các endpoint RESTful API để cung cấp dữ liệu cảm biến cho phần giao diện web. Các API này có thể trả về dữ liệu thời gian thực hoặc lịch sử dữ liệu dưới dạng JSON để JavaScript xử lý và hiển thị.

Quản lý trạng thái thiết bị: Spring Boot cũng có thể quản lý trạng thái bật/tắt của các thiết bị như quạt, đèn và cập nhật các trạng thái này theo yêu cầu từ frontend.

1.3.3. Giới thiệu về MySQL

a. Tổng quan về MySQL



Hình 1.1: ***Giới thiệu về MySql***

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, được phát triển và duy trì bởi Oracle Corporation. Nó sử dụng ngôn ngữ truy vấn cấu trúc (SQL) để truy xuất và quản lý dữ liệu. MySQL được thiết kế với mục tiêu cung cấp hiệu suất cao, tính khả dụng, và độ bảo mật tốt, phù hợp với nhiều loại ứng dụng từ nhỏ đến lớn.

Tính năng nổi bật của MySQL:

- Hiệu suất cao: MySQL sử dụng nhiều kỹ thuật tối ưu hóa để xử lý các truy vấn nhanh chóng, đảm bảo hiệu suất tốt trong môi trường làm việc với khối lượng dữ liệu lớn.

- Khả năng mở rộng: MySQL hỗ trợ khả năng mở rộng với các công cụ như MySQL Cluster, cho phép người dùng quản lý khối lượng dữ liệu ngày càng tăng mà không làm giảm hiệu suất.
- Bảo mật mạnh mẽ: MySQL cung cấp các tính năng bảo mật như mã hóa, xác thực người dùng, và kiểm soát quyền truy cập, giúp bảo vệ dữ liệu khỏi các mối đe dọa tiềm tàng.
- Dễ dàng tích hợp: MySQL có khả năng tích hợp tốt với nhiều ngôn ngữ lập trình và công nghệ khác nhau, bao gồm Java, PHP, Python, và các ứng dụng web.

b. Ứng dụng của MySQL

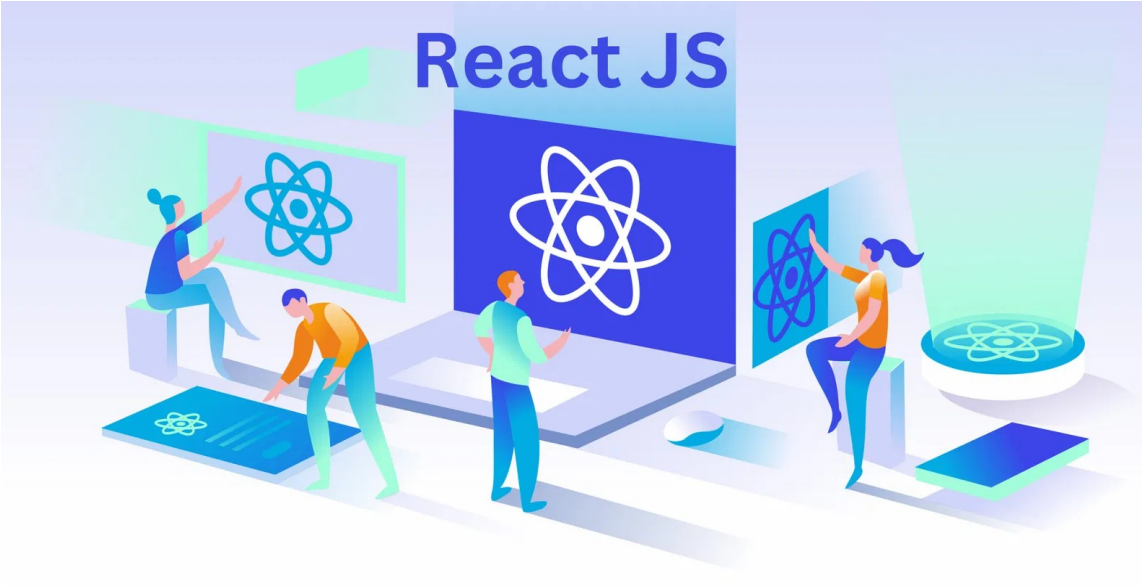
MySQL được sử dụng rộng rãi trong các ứng dụng web, hệ thống quản lý nội dung (CMS), và các dự án lớn như WordPress, Joomla, và Drupal. Nó cũng được sử dụng trong các ứng dụng doanh nghiệp để lưu trữ dữ liệu giao dịch, thông tin khách hàng, và phân tích dữ liệu.

Với sự phổ biến và cộng đồng lớn, MySQL là một lựa chọn hàng đầu cho nhiều nhà phát triển và tổ chức trong việc xây dựng và duy trì các hệ thống quản lý cơ sở dữ liệu hiệu quả.

1.3.4. Giới thiệu về React

a. Tổng quan về React

React là một thư viện JavaScript mã nguồn mở, được phát triển bởi Facebook, và hiện nay đã trở thành một trong những công cụ phổ biến nhất trong phát triển frontend. React được thiết kế để giúp lập trình viên xây dựng các giao diện người dùng một cách hiệu quả và dễ dàng. Thư viện này hoạt động dựa trên khái niệm component-based architecture, cho phép chia nhỏ giao diện thành các thành phần độc lập, có thể tái sử dụng và dễ dàng quản lý.



Hình 1.1: ***Giới thiệu về React***

b. Đặc điểm

Mỗi ứng dụng React được tổ chức thành các component. Mỗi component có thể quản lý trạng thái và logic riêng, giúp việc phát triển và bảo trì mã nguồn trở nên đơn giản hơn. React sử dụng JSX (JavaScript XML), một cú pháp cho phép kết hợp mã JavaScript và HTML trong cùng một tệp, giúp lập trình viên dễ dàng hình dung và viết mã cho giao diện.

React cung cấp một cách tiếp cận mạnh mẽ để quản lý trạng thái của các component thông qua useState và useEffect (Hooks). Điều này giúp lập trình viên dễ dàng theo dõi và cập nhật trạng thái mà không cần phải viết mã phức tạp. Ngoài ra, các thư viện quản lý trạng thái bên ngoài như Redux hoặc MobX có thể được tích hợp để quản lý trạng thái toàn cục cho các ứng dụng lớn.

React sử dụng Virtual DOM, một phiên bản ảo của DOM, để tối ưu hóa hiệu suất. Khi có sự thay đổi trong trạng thái, React sẽ cập nhật Virtual DOM trước, sau đó so sánh với DOM thật và chỉ cập nhật những phần cần thiết. Điều này giúp giảm thiểu số lượng thao tác trên DOM thật, từ đó cải thiện tốc độ và hiệu suất của ứng dụng.

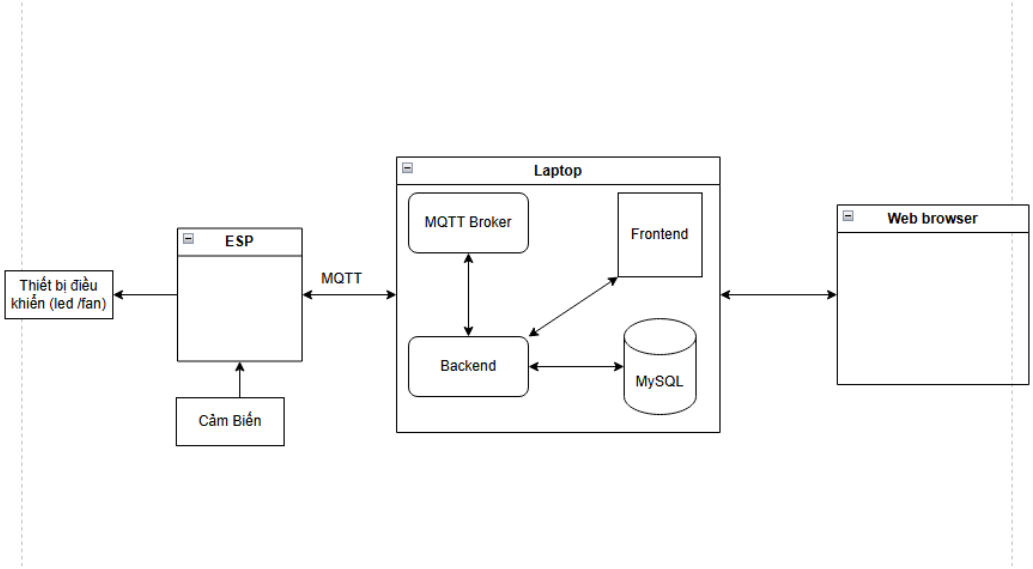
React hỗ trợ tích hợp với nhiều thư viện và công nghệ khác, cho phép mở rộng tính năng của ứng dụng một cách dễ dàng. Các thư viện như React Router giúp quản lý điều hướng trong ứng dụng, trong khi các thư viện như Axios cho phép thực hiện các yêu cầu HTTP một cách thuận tiện. Hệ sinh thái phong phú này giúp lập trình viên có thể xây dựng các ứng dụng phức tạp mà không gặp phải nhiều trở ngại.

React thường được sử dụng để phát triển Single Page Applications (SPAs), nơi mà người dùng có thể tương tác mà không cần tải lại trang. Nhiều ứng dụng web lớn như Facebook, Instagram và Airbnb đều được xây dựng dựa trên React, nhờ vào khả năng linh hoạt và hiệu suất cao mà nó mang lại. Ngoài ra, React cũng được sử dụng trong phát triển ứng dụng di động thông qua React Native, cho phép lập trình viên sử dụng mã nguồn JavaScript để xây dựng ứng dụng cho cả iOS và Android.

Chương 2. Thiết kế và triển khai hệ thống

2.1. Thiết kế luồng

2.1.1. Sơ đồ khối thiết kế luồng



Hình 1.1: ***Hình ảnh thiết kế luồng***

2.1.2. Thành phần và chức năng chính của hệ thống

Thành phần ESP

Cảm biến thu thập các dữ liệu môi trường như nhiệt độ, độ ẩm, ánh sáng và gửi dữ liệu này đến MQTT broker thông qua topic MQTT là data_sensor.

Thiết bị điều khiển (quạt, đèn LED) nhận các lệnh điều khiển từ hệ thống thông qua các topic MQTT tương ứng (action).

MQTT Broker

MQTT là giao thức truyền tin sử dụng trong hệ thống IoT để đảm bảo việc truyền thông nhanh chóng và nhẹ nhàng. MQTT broker (ở đây cấu hình là tcp://localhost:1996) đóng vai trò trung gian giữa cảm biến và hệ thống backend.

Backend (Spring Boot)

Hệ thống Spring Boot được cấu hình để nhận dữ liệu từ broker MQTT, lưu trữ dữ liệu vào cơ sở dữ liệu MySQL, và gửi các lệnh điều khiển tới thiết bị IoT thông qua các topic MQTT.

Các chức năng chính của backend bao gồm:

- Nhận và xử lý dữ liệu cảm biến.

- Lưu trữ dữ liệu cảm biến.
- Điều khiển thiết bị thông qua các yêu cầu từ người dùng.

Cơ sở dữ liệu MySQL

Dữ liệu cảm biến được lưu vào MySQL, cho phép hệ thống lưu trữ và truy vấn dữ liệu lịch sử môi trường một cách hiệu quả.

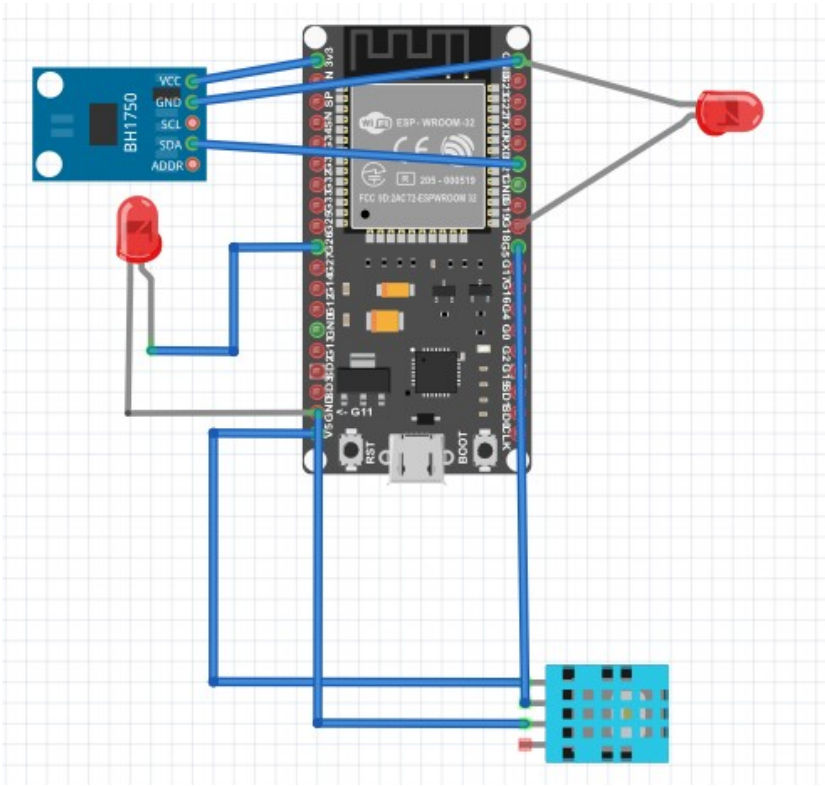
Frontend React

Giao diện React cho phép người dùng quan sát các dữ liệu cảm biến và điều khiển các thiết bị (bật/tắt quạt và đèn LED). Các thao tác điều khiển này được gửi tới backend qua các API HTTP.

2.2. Thiết kế giao diện và phần cứng

2.2.1. Thiết kế phần cứng

a. Sơ đồ kết nối chân phần cứng



Hình 2.1: ***Sơ đồ kết nối chân***

Sơ đồ kết nối cảm biến ánh sáng bh1750

Chân ESP	Chân BH1750
VCC	5V

GND	GND
SCL	PIN 22
SDA	PIN 21

Bảng 2.1: Sơ đồ kết nối chân cảm biến ánh sáng

Sơ đồ kết nối cảm biến nhiệt độ, độ ẩm dht11

Chân ESP	**Chân DHT11**
VCC	5V
GND	GND
SDA	PIN 4

Bảng 2.2: Sơ đồ kết nối chân cảm biến nhiệt độ, độ ẩm

b. Giới thiệu về khối phần cứng

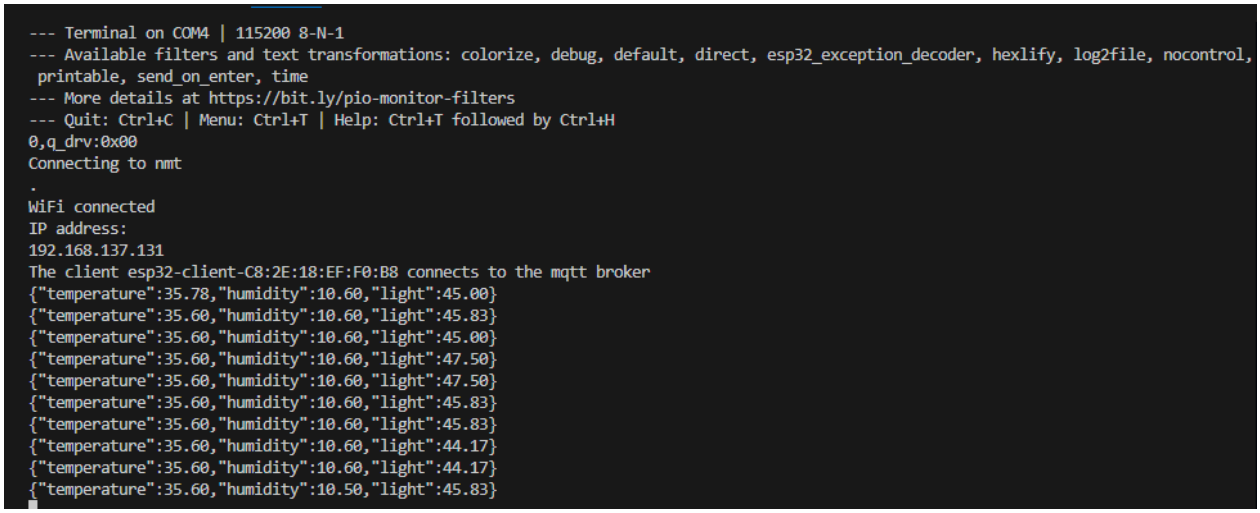
Chức năng

Khối cảm biến chịu trách nhiệm ghi nhận các thông tin từ môi trường như nhiệt độ, độ ẩm và ánh sáng. Đây là nguồn dữ liệu đầu vào cho toàn bộ hệ thống.

Các thành phần

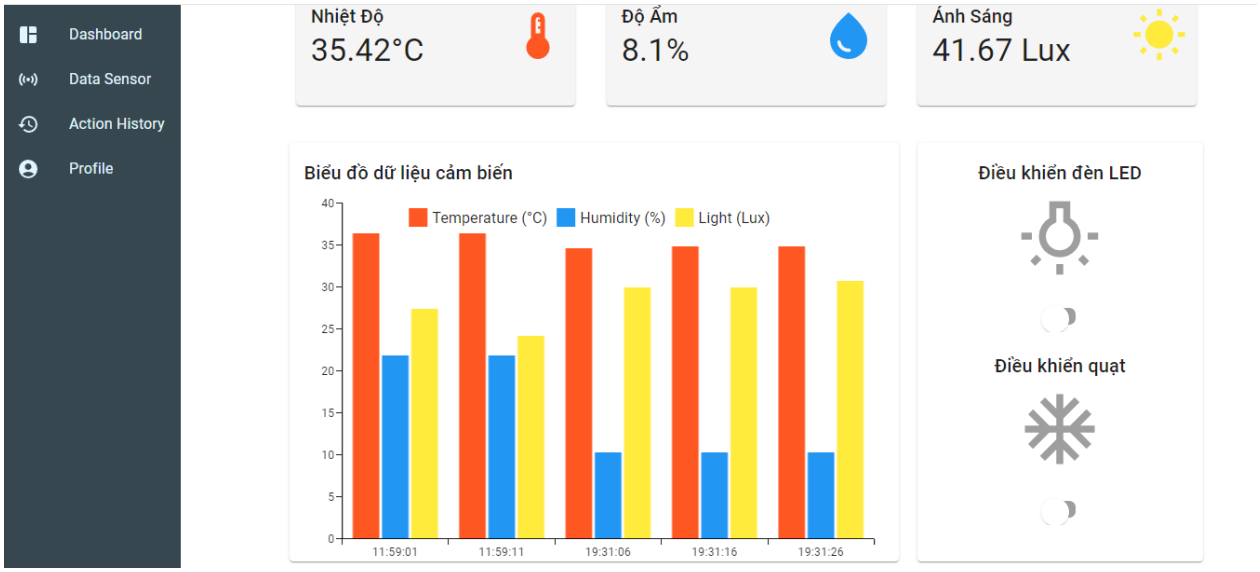
Trong hệ thống IoT sử dụng Spring Boot, ESP32, và các cảm biến như DHT11 và BH1750, cấu hình hệ thống đóng vai trò quan trọng trong việc thu thập dữ liệu môi trường và điều khiển thiết bị. ESP32 được tích hợp với cảm biến DHT11 để đo nhiệt độ và độ ẩm, cũng như cảm biến BH1750 để đo cường độ ánh sáng. Hệ thống còn bao gồm hai đèn LED được điều khiển từ xa qua giao thức MQTT. Khi dữ liệu từ các cảm biến được thu thập, ESP32 sẽ gửi thông tin qua MQTT broker đến ứng dụng Spring Boot, nơi các listener được cấu hình sẵn trong MqttConfig tiếp nhận thông điệp. Sau đó, dữ liệu được truyền vào service để xử lý và lưu trữ vào cơ sở dữ liệu MySQL thông qua các repository. Đồng thời, người dùng có thể gửi yêu cầu điều khiển hai đèn LED thông qua frontend React, từ đó Spring Boot sẽ gửi lệnh điều khiển qua MQTT đến ESP32 để bật hoặc tắt đèn. Quá trình hoạt động liên tục này giúp hệ thống phản hồi nhanh chóng với môi trường và cho phép điều khiển thiết bị từ xa một cách hiệu quả.

Luồng hoạt động



Hình 2.1: ***Hình ảnh dữ liệu từ esp đưa lên mqtt broker***

2.2.2. Thiết kế giao diện



Hình 2.1: ***Hình ảnh giao diện Dashboard***

Trong hệ thống IoT sử dụng frontend React, giao diện người dùng được thiết kế với bốn trang chính để theo dõi và quản lý dữ liệu cảm biến cũng như điều khiển thiết bị. Trang đầu tiên, "Dashboard", hiển thị trạng thái điều khiển của hai đèn LED và bao gồm ba thẻ (card) riêng biệt để hiển thị các giá trị mới nhất về nhiệt độ, độ ẩm (từ cảm biến DHT11), và cường độ ánh sáng (từ cảm biến BH1750). Ngoài ra, Dashboard cũng tích hợp một biểu đồ theo dõi các giá trị cảm biến này trong khoảng thời gian cụ thể, giúp người dùng dễ dàng quan sát xu hướng dữ liệu.

Trang thứ hai, “Data Sensor”cho phép người dùng theo dõi lịch sử cập nhật của các giá trị nhiệt độ, độ ẩm, và ánh sáng với chi tiết về thời gian ghi nhận. Trang thứ ba, “Action History”, ghi lại lịch sử điều khiển hai đèn LED theo thời gian, hiển thị khi nào thiết bị được bật hoặc tắt. Cuối cùng, trang thứ tư, “Profile”, cung cấp thông tin người dùng và các tùy chọn cài đặt cá nhân hóa. Toàn bộ hệ thống frontend này được kết nối với backend Spring Boot, đảm bảo việc hiển thị dữ liệu theo thời gian thực và quản lý lịch sử hiệu quả, cung cấp trải nghiệm trực quan và tiện lợi cho người dùng.

2.3. Thiết kế hệ thống Backend

Hệ thống backend được xây dựng bằng Spring Boot là phần quan trọng trong kiến trúc hệ thống IoT của bạn. Nó có trách nhiệm xử lý dữ liệu từ các cảm biến, lưu trữ vào cơ sở dữ liệu, và cung cấp API cho frontend để tương tác với dữ liệu và điều khiển thiết bị. Dưới đây là phân tích chi tiết hơn về từng thành phần và luồng hoạt động của hệ thống backend.

2.3.1. Chức năng

Xử lý dữ liệu từ cảm biến: Nhận và phân tích dữ liệu gửi từ broker MQTT.

Lưu trữ dữ liệu: Lưu trữ dữ liệu vào cơ sở dữ liệu MySQL để phục vụ cho việc truy xuất và phân tích sau này.

Cung cấp API: Cung cấp các RESTful API cho frontend để truy cập và hiển thị dữ liệu, cũng như thực hiện các hành động điều khiển thiết bị.

Điều khiển thiết bị: Gửi lệnh điều khiển đến thiết bị thông qua MQTT.

2.3.2. Các thành phần chính

Controller

Controller đóng vai trò quan trọng trong việc quản lý giao tiếp giữa client (frontend) và các dịch vụ xử lý dữ liệu. Chức năng chính của Controller là nhận yêu cầu từ client và gọi các service tương ứng để xử lý. Cụ thể, DataSensorController đảm nhận việc xử lý các yêu cầu liên quan đến dữ liệu cảm biến, trong khi ActionController quản lý các yêu cầu điều khiển thiết bị.

Luồng hoạt động bắt đầu khi Controller nhận yêu cầu từ frontend, tiếp theo là việc gọi các service phù hợp để thực hiện xử lý và cuối cùng trả về dữ liệu hoặc trạng thái cho client. Qua đó, Controller không chỉ giúp tách biệt các tầng trong ứng dụng

mà còn tối ưu hóa quy trình xử lý thông tin, đảm bảo tính hiệu quả và khả năng mở rộng của hệ thống.

Service

Lớp Service đóng vai trò trung tâm trong việc xử lý logic nghiệp vụ và quản lý dữ liệu. Chức năng chính của Service là tiếp nhận dữ liệu từ Controller và thực hiện các thao tác cần thiết liên quan đến nghiệp vụ. Cụ thể, DataSensorService chuyên trách việc xử lý các hoạt động liên quan đến dữ liệu cảm biến, trong khi ActionService đảm nhận việc xử lý các hành động điều khiển thiết bị.

Luồng hoạt động bắt đầu khi Service nhận dữ liệu từ Controller, sau đó thực hiện các thao tác như lưu trữ, truy xuất thông tin, và gửi lệnh điều khiển. Đặc biệt, Service còn gọi các repository để tương tác với cơ sở dữ liệu, đảm bảo việc quản lý dữ liệu diễn ra mượt mà và hiệu quả. Nhờ vào việc tách biệt rõ ràng giữa Controller và Service, ứng dụng có thể duy trì sự linh hoạt và dễ dàng mở rộng trong tương lai.

Repository

Lớp Repository đảm nhiệm vai trò giao tiếp với cơ sở dữ liệu MySQL, thực hiện các thao tác cơ bản trong quản lý dữ liệu theo mô hình CRUD (Create, Read, Update, Delete). Chức năng chính của Repository là nhận yêu cầu từ lớp Service và thực hiện các truy vấn SQL tương ứng. Cụ thể, DataSensorRepository giao tiếp với bảng lưu trữ dữ liệu cảm biến, trong khi ActionRepository tương tác với bảng lưu trữ các hành động điều khiển thiết bị. Luồng hoạt động bắt đầu khi Repository nhận yêu cầu từ Service, sau đó tiến hành thực hiện các truy vấn SQL cần thiết và cuối cùng trả kết quả về cho Service. Việc tách biệt rõ ràng giữa lớp Repository và các thành phần khác của ứng dụng không chỉ giúp tối ưu hóa quá trình xử lý dữ liệu mà còn nâng cao khả năng bảo trì và mở rộng của hệ thống.

MQTT Configuration

MQTT Configuration đóng vai trò thiết yếu trong việc thiết lập kết nối đến broker MQTT và xử lý thông điệp truyền đi. Chức năng chính của cấu hình này là tạo điều kiện cho các thành phần trong ứng dụng có thể giao tiếp hiệu quả qua giao thức MQTT. Một trong những thành phần quan trọng là MqttConfig, đảm nhận việc cấu hình kết nối đến broker và thiết lập các listener cho các topic. Luồng hoạt động bắt đầu khi broker gửi thông điệp mới, lúc này các listener sẽ kích hoạt và gọi phương thức

tương ứng trong service để xử lý dữ liệu. Quá trình này không chỉ giúp tối ưu hóa việc truyền tải thông điệp mà còn đảm bảo tính phản hồi nhanh chóng trong ứng dụng IoT, từ đó nâng cao khả năng tương tác và điều khiển các thiết bị kết nối.

2.3.3. Luồng hoạt động chi tiết

Khởi động ứng dụng:

- Ứng dụng Spring Boot được khởi động, và trong quá trình này, MqttConfig được thiết lập.
- Kết nối đến broker MQTT và đăng ký listener cho topic data_sensor.

Nhận dữ liệu từ broker:

- Khi có thông điệp mới được gửi từ cảm biến đến broker, broker sẽ chuyển tiếp thông điệp đó đến ứng dụng backend.
- MqttMessageListener sẽ nhận thông điệp này và gọi phương thức trong DataSensorService để xử lý.

Xử lý dữ liệu cảm biến:

- Trong DataSensorService, thông điệp JSON được phân tích bằng ObjectMapper của Jackson để chuyển đổi thành đối tượng Java (ví dụ: DataSensor).
- Kiểm tra tính hợp lệ của dữ liệu (ví dụ: kiểm tra giá trị có nằm trong khoảng cho phép không).
- Nếu dữ liệu hợp lệ, gọi DataSensorRepository để lưu trữ dữ liệu vào cơ sở dữ liệu MySQL.

Cung cấp API cho frontend:

Controller cung cấp các endpoint như:

- GET /api/sensor/data: Truy xuất danh sách dữ liệu cảm biến từ cơ sở dữ liệu.
- POST /api/action: Gửi lệnh điều khiển thiết bị.

Khi Frontend gửi yêu cầu đến các endpoint này:

- Controller gọi service tương ứng để xử lý yêu cầu.
- Service truy cập repository để lấy dữ liệu từ cơ sở dữ liệu hoặc thực hiện hành động điều khiển.
- Kết quả được trả về cho frontend.

Gửi lệnh điều khiển:

- Khi người dùng gửi lệnh điều khiển thông qua API, ActionController nhận yêu cầu.
- Gọi ActionService để xử lý lệnh.
- ActionService sẽ gửi lệnh đến broker MQTT để truyền đến thiết bị tương ứng.

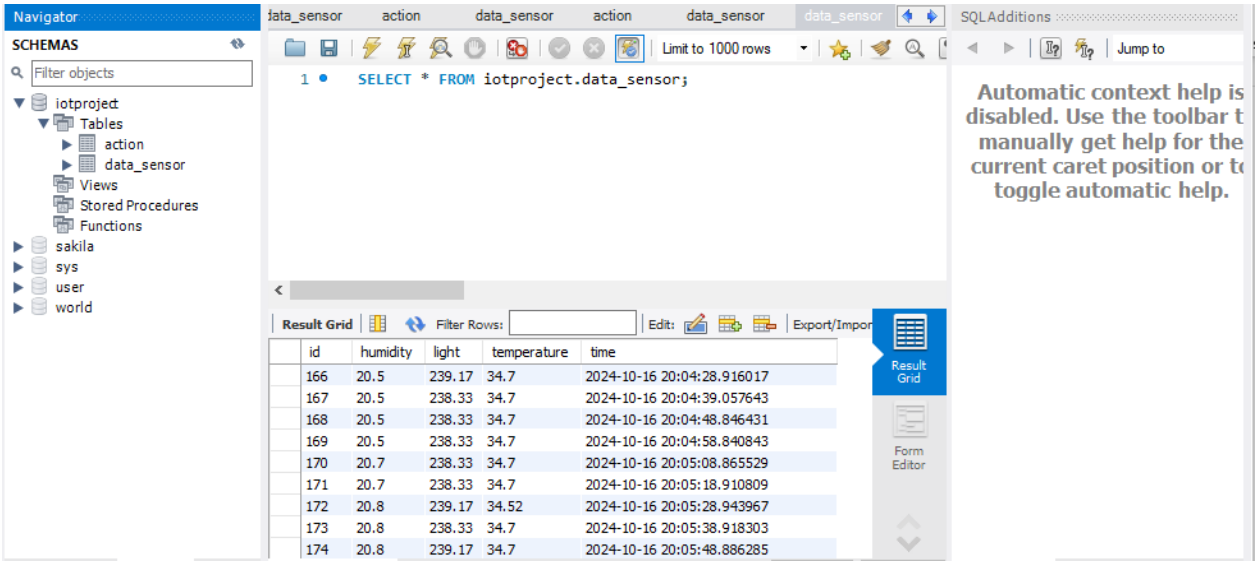
Kết quả đầu ra

- Dữ liệu từ cảm biến được lưu trữ trong cơ sở dữ liệu MySQL và có thể truy cập thông qua các API.
- Lệnh điều khiển được gửi đến thiết bị và được thực hiện.
- Hệ thống backend đảm bảo tính chính xác và khả năng mở rộng cho các yêu cầu từ frontend.

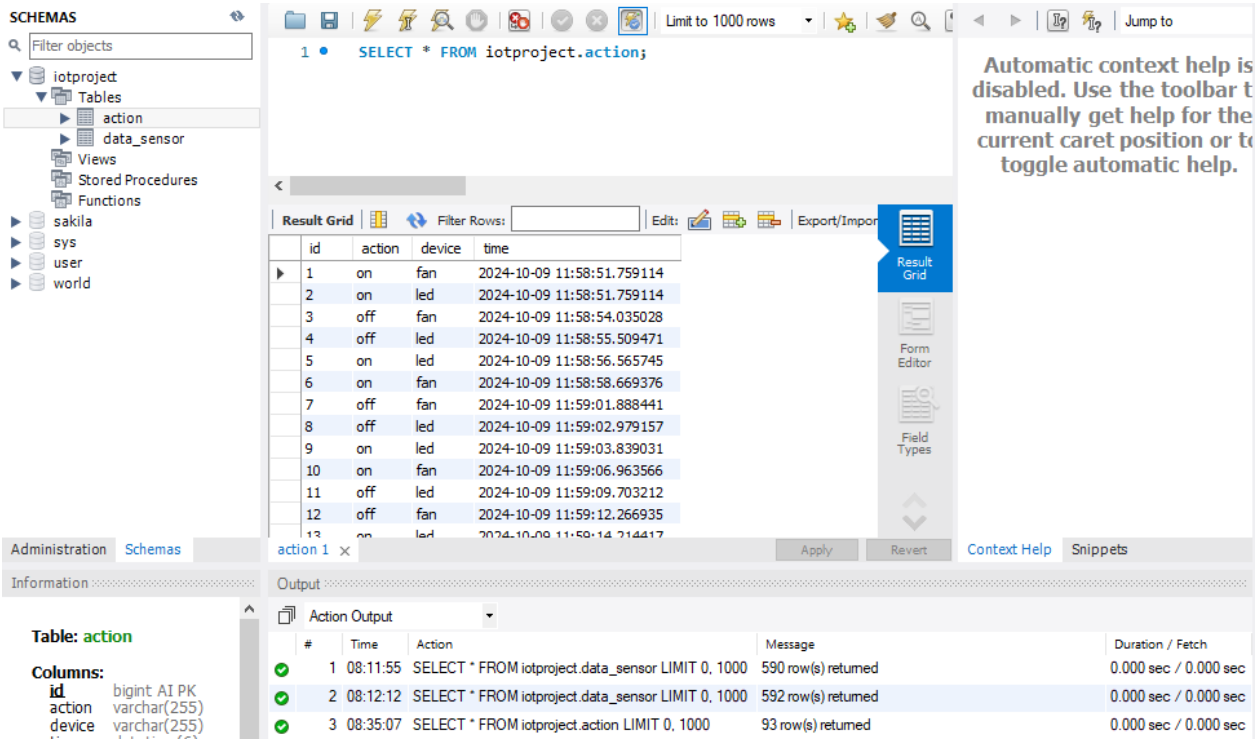
Chương 3. Kết quả và Đánh giá nhận xét

3.1. Hình ảnh của toàn bộ hệ thống:

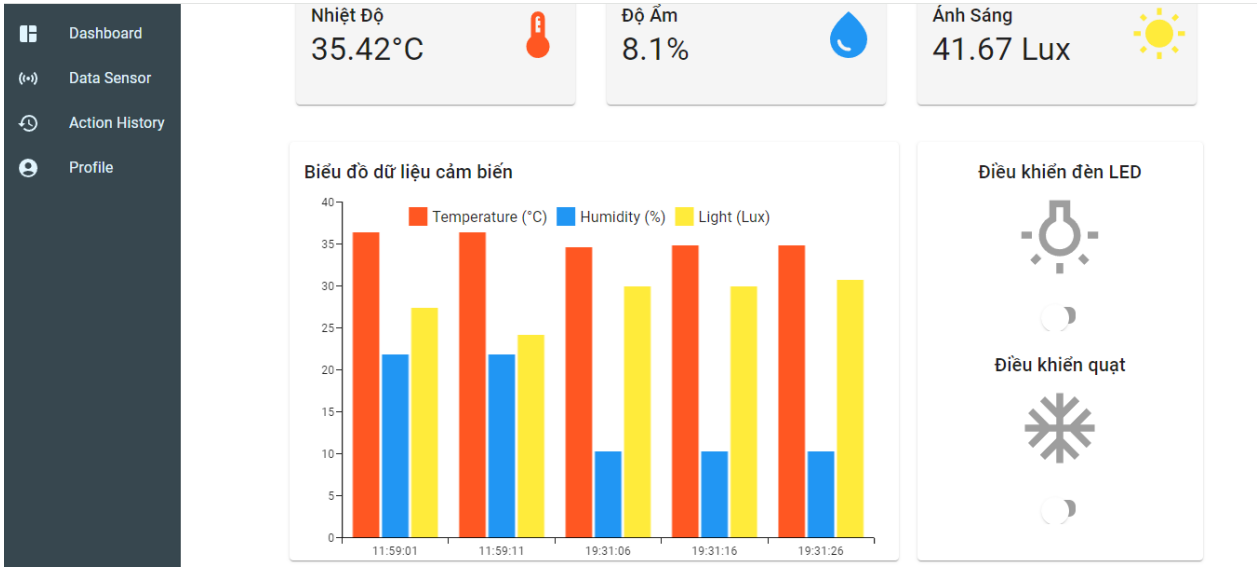
Sau khi dữ liệu được đưa vào cập nhật trong MySQL



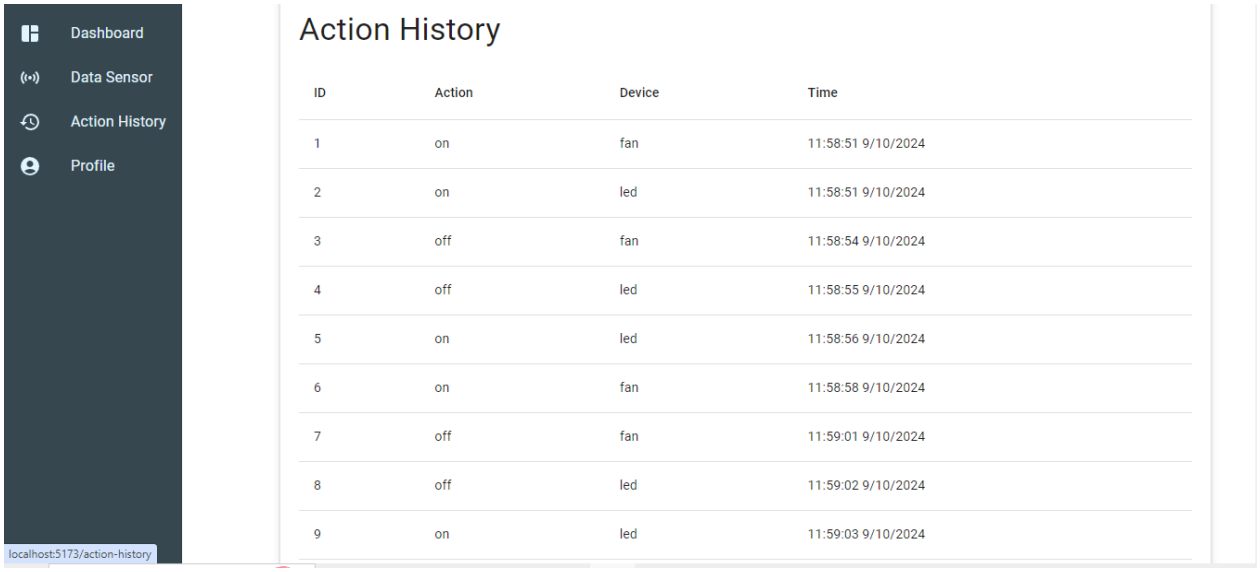
Hình 1.1: ***Hình ảnh dữ liệu data sensor được cập nhật trên MySQL***



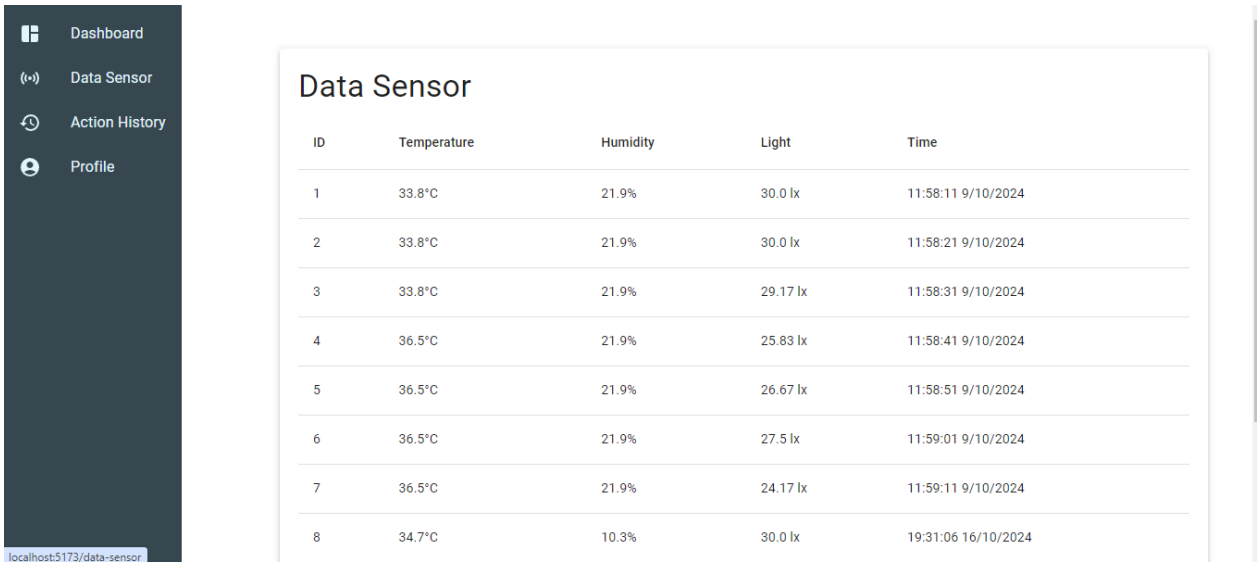
Hình 1.2: ***Hình ảnh dữ liệu action được cập nhật trên MySQL***



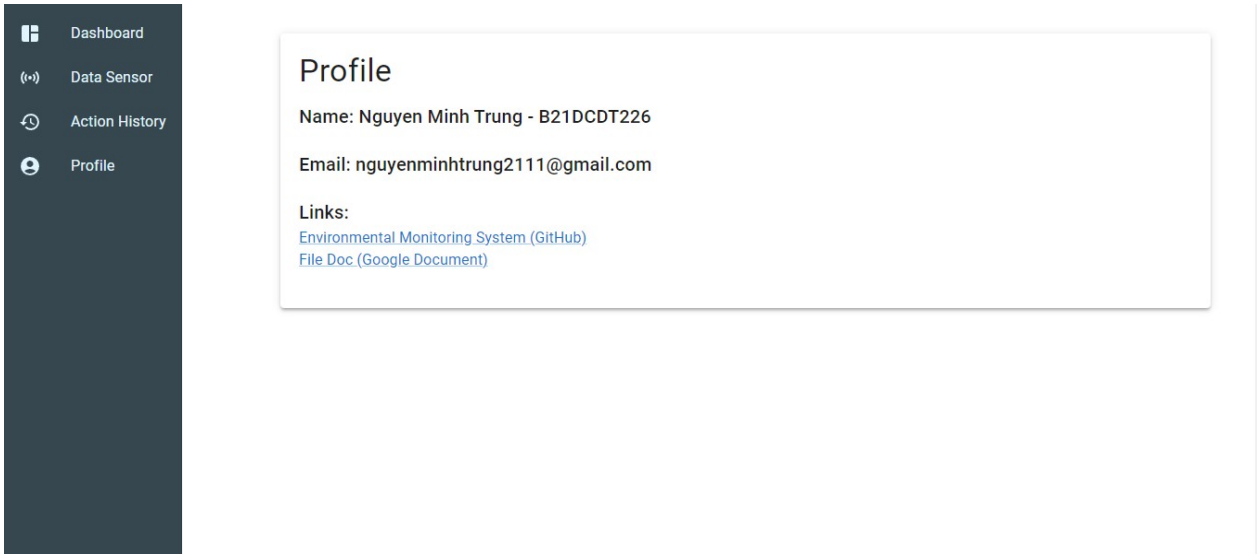
Hình 1.3: ***Giao diện Dashboard được cập nhật trên Frontend***



Hình 1.4: ***Hình ảnh dữ liệu Action được cập nhật trên Frontend***



Hình 1.5: ***Hình ảnh dữ liệu Data sensor được cập nhật trên Frontend***



Hình 1.6: ***Hình ảnh trang Profile***