

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1**

o0o



IOT VÀ ỨNG DỤNG

Đề tài: Hệ thống theo dõi sức khỏe

Nhóm học phần: 15

Giảng viên hướng dẫn : Nguyễn Quốc Uy

Họ và tên sinh viên : Nguyễn Quốc Dương

Mã sinh viên : B22DCCN167

HÀ NỘI, 08/2025

LỜI CẢM ƠN

Em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Quốc Uy, giảng viên môn IoT và Ứng dụng lớp 15, đã tận tình truyền đạt kiến thức và định hướng trong suốt quá trình học tập. Nhờ sự chỉ dạy của thầy, em đã có thể hoàn thành báo cáo bài tập lớn này.

Em xin trân trọng cảm ơn!

Mục lục

I.	Giới thiệu	4
A.	Tổng quan	4
B.	Các chức năng chính của hệ thống	5
C.	Công nghệ và thiết bị sử dụng	5
1.	Phần cứng.....	5
2.	Phần mềm.....	5
II.	Giao diện	7
A.	Giao diện trang chủ	7
B.	Giao diện thống kê dữ liệu cảm biến	7
C.	Giao diện thống kê lịch sử điều khiển thiết bị ..	8
D.	Giao diện hồ sơ	8
III.	Phân tích, thiết kế hệ thống	9
A.	Kiến trúc hệ thống	9
B.	Các sơ đồ logic	10
1.	Sequence Diagram.....	10
2.	Sensor Activity Diagram.....	11
3.	Device Activity Diagram.....	11
C.	Thiết kế CSDL	12
D.	Chuẩn bị phần cứng	15
E.	Đầu nối mạch	17
F.	Code Arduino	18
G.	Code Back-end (Node/Express)	21
1.	Các module chính.....	22
H.	API	23
1.	Get latest data sensor.....	23
2.	Get 10 latest data sensor.....	24
3.	Get history of sensors.....	25
4.	Get history of actions.....	27
5.	Get device status.....	28
6.	Toggle devices.....	29
IV.	Đánh giá kết quả	30
A.	Chức năng đã hoàn thành	30
1.	Thu thập và hiển thị dữ liệu cảm biến.....	30
2.	Điều khiển thiết bị từ xa.....	31
3.	Lưu trữ và hiển thị lịch sử trạng thái thiết bị	31
4.	Xác thực và bảo mật người dùng.....	31
5.	API linh hoạt và tìm kiếm nâng cao.....	31
B.	Đánh giá hiệu suất	31

1.	Đánh giá độ chính xác của cảm biến.....	32
2.	Đánh giá về tốc độ phản hồi của hệ thống.....	32
3.	Độ trễ điều khiển thiết bị.....	32
4.	Cập nhật dữ liệu real-time.....	32
C.	Điểm cần cải thiện	32
1.	Xử lý lỗi chưa toàn diện.....	32
2.	Thiếu monitoring và logging.....	32
3.	Bảo mật cần tăng cường.....	32
D.	Đề xuất cải tiến	33
1.	Tăng cường bảo mật và hiệu suất.....	33
2.	Mở rộng khả năng phân tích dữ liệu.....	33
3.	Hỗ trợ đa thiết bị và đa người dùng.....	33
4.	Phát triển ứng dụng di động.....	33

I. Giới thiệu

A. Tổng quan

- Đề tài xây dựng một hệ thống IoT hỗ trợ giám sát và chẩn đoán sức khỏe cơ bản của người dùng. Phần phần cứng sử dụng vi điều khiển **ESP32** kết hợp với các cảm biến **DHT11** (đo nhiệt độ và độ ẩm) và **BH1750** (đo ánh sáng). Hệ thống cũng tích hợp đèn LED kèm điện trở, có thể bật/tắt từ xa nhằm mô phỏng khả năng điều khiển thiết bị ngoại vi.
- Dữ liệu từ các cảm biến và trạng thái điều khiển thiết bị được truyền qua giao thức **MQTT**, đảm bảo việc giao tiếp giữa ESP32 và hệ thống phần mềm diễn ra nhanh chóng, ổn định và dễ mở rộng. Ở phía backend, **NodeJS** chịu trách nhiệm xử lý, lưu trữ và cung cấp API. Phía frontend, **React** được sử dụng để xây dựng giao diện web, hỗ trợ hiển thị các chỉ số sức khỏe và điều khiển thiết bị theo thời gian thực.

B. Các chức năng chính của hệ thống

- Xem các thông tin nhiệt độ cơ thể, nhịp tim và nồng độ ô xi trong máu realtime qua cảm biến
- Điều khiển bật tắt 3 thiết bị đèn LED
- Xem lịch sử data sensor: Tìm kiếm và sắp xếp theo các giá trị + thời gian
- Xem lịch sử bật tắt thiết bị: Tìm kiếm và sắp xếp theo thời gian, loại thiết bị, hành động.

C. Công nghệ và thiết bị sử dụng

1. Phần cứng

- ESP32: Vi điều khiển chính của hệ thống, có khả năng kết nối Wi-Fi và Bluetooth, dùng để thu thập và gửi dữ liệu cảm biến lên server.
- Board Test 400 lỗ: Bảng mạch thử nghiệm (breadboard) dùng để lắp ráp mạch tạm thời mà không cần hàn.
- Cảm biến nhiệt độ và độ ẩm DHT11: Đo nhiệt độ cơ thể và gửi dữ liệu dạng số đến ESP32.
- Đèn LED và điện trở: Hiển thị trạng thái hoạt động hoặc cảnh báo của hệ thống.
- Dây jump đực – cái: Dây nối giữa các linh kiện và ESP32, giúp truyền tín hiệu điện.

2. Phần mềm

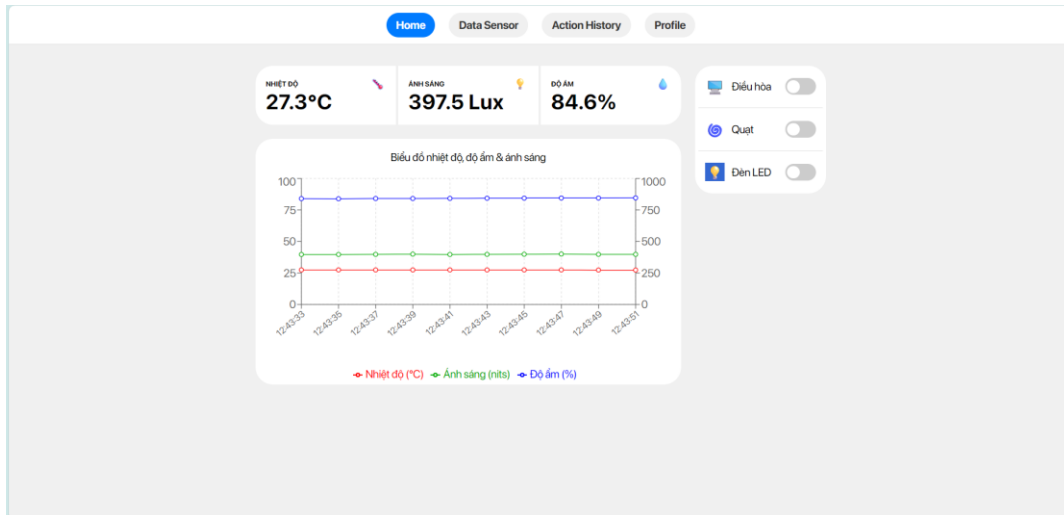
- Front-End: Sử dụng React (JavaScript Framework) để xây dựng giao diện người dùng, hiển thị dữ liệu cảm biến theo thời gian thực.
- Back-End: Sử dụng Node/Express (JavaScript) để xây dựng server, xử lý dữ liệu và cung cấp API cho giao diện.
- Database: Dùng MongoDB để lưu trữ dữ liệu cảm biến và thông tin

người dùng.

- MQTT Broker: Sử dụng Mosquitto làm broker trung gian giúp truyền dữ liệu giữa ESP32 và server.
- Postman API: Công cụ dùng để kiểm thử, phát triển và tài liệu hóa các API của hệ thống.
- Công cụ quản lý phiên bản : Git/Github để quản lý version mã nguồn.

II. Giao diện

A. Giao diện trang chủ



Hình 1: Giao diện trang chủ

- Trang Dashboard hiển thị dữ liệu cảm biến realtime , bên cạnh đó là thanh điều khiển bóng đèn LED , cùng với biểu đồ dữ liệu của cả 3 loại cảm biến trong 20s.
- Dữ liệu cảm biến có hiệu ứng biến đổi tùy vào chỉ số
- Các thiết bị chỉ thay đổi icon khi nhận được message trả về đã bật/tắt thành công

B. Giao diện thống kê dữ liệu cảm biến

The 'Data Sensor' interface displays a table of historical sensor data. At the top, there are tabs for 'Home', 'Data Sensor', 'Action History', and 'Profile'. Below the tabs, there is a search bar with a dropdown menu set to 'Tất cả' (All) and a search input field. To the right of the search bar are two buttons: 'Tìm kiếm' (Search) and 'Bỏ lọc' (Clear Filter). The table has five columns: 'STT' (Serial Number), 'Nhiệt độ' (Temperature), 'Độ ẩm' (Humidity), 'Ánh sáng' (Light), and 'Thời gian' (Time). The table contains 10 rows of data, each representing a sensor reading at a specific time. At the bottom of the table, there is a pagination bar showing 'Hiện thị: 10' (Display: 10) and 'bản ghi' (records), along with a total count of '1/1036' records.

STT	Nhiệt độ	Độ ẩm	Ánh sáng	Thời gian
1	27.3°C	85.6%	402.5 LUX	2025-10-17 12:44:23
2	27.3°C	85.2%	400 LUX	2025-10-17 12:44:21
3	27.3°C	85.4%	398.3 LUX	2025-10-17 12:44:19
4	27.3°C	85.4%	398.3 LUX	2025-10-17 12:44:17
5	27.3°C	85.3%	400.8 LUX	2025-10-17 12:44:15
6	27.3°C	85.3%	401.7 LUX	2025-10-17 12:44:13
7	27.3°C	85.2%	402.5 LUX	2025-10-17 12:44:11
8	27.3°C	85.2%	402.5 LUX	2025-10-17 12:44:09
9	27.3°C	85.2%	400 LUX	2025-10-17 12:44:07
10	27.3°C	85.1%	400 LUX	2025-10-17 12:44:05

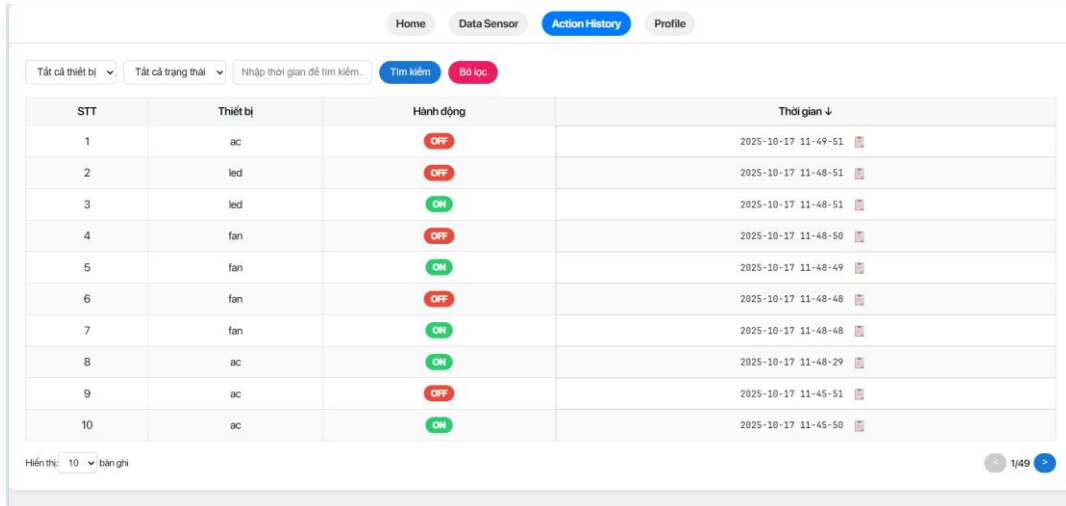
Hình 2: Giao diện thống kê dữ liệu cảm biến

- Hiển thị lịch sử cảm biến ghi nhận trong cơ sở dữ liệu, cho phép tìm

kiểm và sắp xếp theo các tiêu chí tắt cả, id, giá trị cảm biến và thời gian.

- Cho phép phân trang và chọn số bản ghi hiển thị

C. Giao diện thống kê lịch sử điều khiển thiết bị



The screenshot shows the 'Action History' page with a table of device control actions. The table has four columns: STT, Thiết bị, Hành động, and Thời gian. The data is as follows:

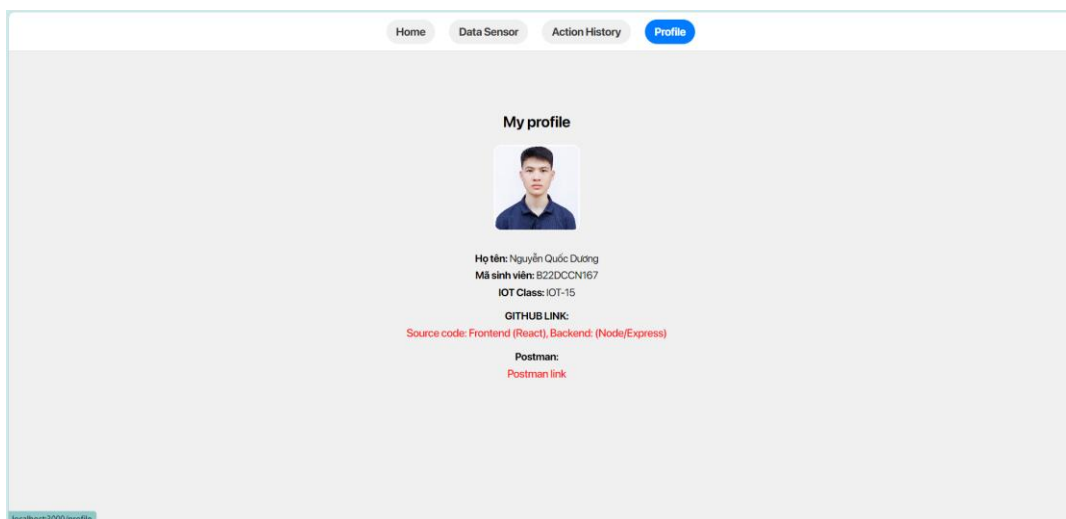
STT	Thiết bị	Hành động	Thời gian ↓
1	ac	OFF	2025-10-17 11-49-51
2	led	OFF	2025-10-17 11-48-51
3	led	ON	2025-10-17 11-48-51
4	fan	OFF	2025-10-17 11-48-50
5	fan	ON	2025-10-17 11-48-49
6	fan	OFF	2025-10-17 11-48-48
7	fan	ON	2025-10-17 11-48-48
8	ac	ON	2025-10-17 11-48-29
9	ac	OFF	2025-10-17 11-45-51
10	ac	ON	2025-10-17 11-45-50

At the bottom, there is a pagination control showing 'Hiển thị: 10 bản ghi' and a page number '1/49'.

Hình 3: Giao diện thống kê lịch sử điều khiển thiết bị

- Hiện thị lịch sử bật tắt thiết bị được ghi nhận trong cơ sở dữ liệu, cho phép tìm kiếm và sắp xếp theo các tiêu chí id, thời gian. Về thiết bị và trạng thái có dropdown để lựa chọn tìm kiếm.

D. Giao diện hồ sơ



Hình 4: Giao diện hồ sơ

- Hiện thị thông tin cá nhân kèm link GitHub mã nguồn dự án kèm theo file PDF báo cáo và API Docs (Postman).

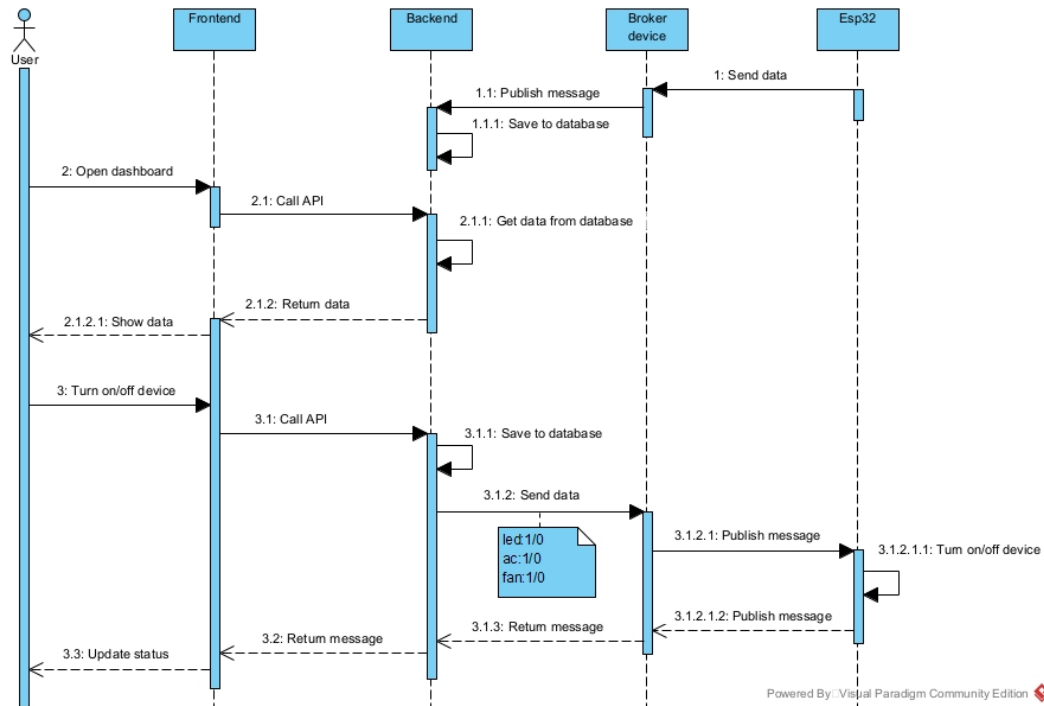
III. Phân tích, thiết kế hệ thống

A. Kiến trúc hệ thống

- Kiến trúc hệ thống này được chia thành ba tầng chính: tầng phần cứng, tầng back-end và tầng front-end.
- Tầng phần cứng bao gồm ESP32 với các chức năng như Wi-Fi, Bluetooth, cảm biến DS18B20, MAX30102, và SpO2 cùng với linh kiện như LED, pin điều khiển Board Test 400 và dây jump để kết nối các thành phần.
- Tầng back-end sử dụng Node/Express với JavaScript Framework, hỗ trợ giao diện người dùng theo thời gian thực.
- Toàn bộ hệ thống được điều khiển bởi người dùng và dữ liệu được truyền qua các giao thức như MQTT và HTTP giữa các tầng.

B. Các sơ đồ logic

1. Sequence Diagram

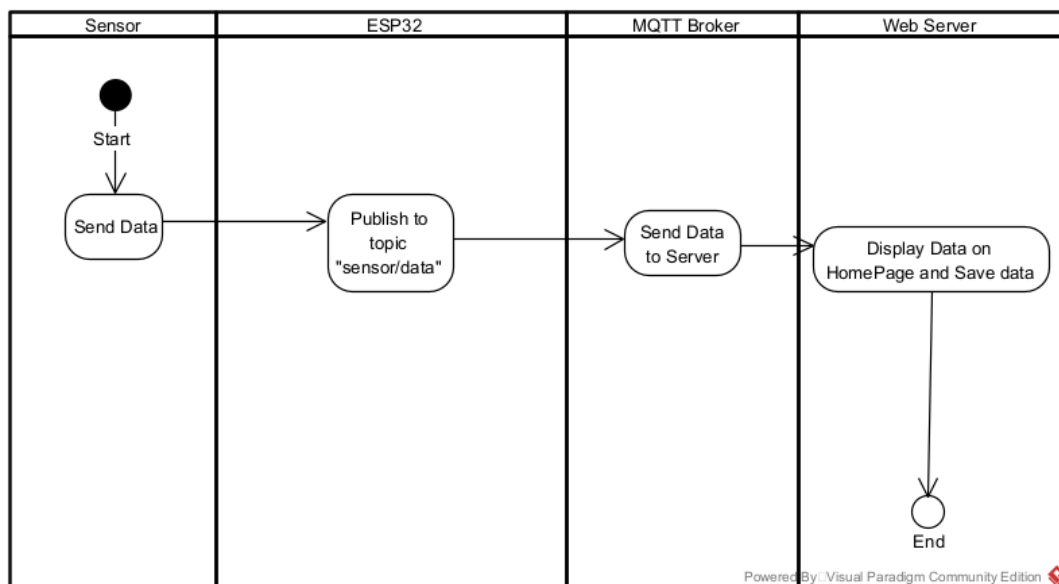


Hình 5: Sequence Diagram

- Cả máy chủ web và phần cứng (Esp32) cùng kết nối vào MQTT Broker(Mosquitto) thông qua Wifi
- Web Server subscribe topic data/sensor, state/device
- Esp32 subscribe topic device/action
- Sử dụng Arduino IDE điều khiển phần cứng
- Luồng dữ liệu từ cảm biến - > Web server:
 1. Cảm biến đọc dữ liệu đo được sau đó đẩy về Esp32
 2. Esp32 publish dữ liệu về topic data/sensor
 3. Web Server nhận dữ liệu được publish do đã subscribe vào topic data/sensor
 4. Web Server lưu dữ liệu hợp lệ vào database và hiển thị lên giao diện
- Luồng điều khiển thiết bị từ Web - > Esp32:
 5. Người dùng click nút On/Off thiết bị trên giao diện
 6. Web Server publish lệnh điều khiển lên topic device/action ở Broker

7. MQTT Broker publish message sang Esp32 đã subscribe trước đó
8. Esp32 thực hiện điều khiển bật tắt thiết bị
9. Điều khiển xong Esp32 trả về Response (Bật tắt thành công/ thất bại)
10. Response được publish đến topic state/device ở Broker
11. Broker publish response này đến Web Server
12. Server nhận phản hồi và xử lý (lưu vào database nếu thành công+ thay đổi hiệu ứng icon)

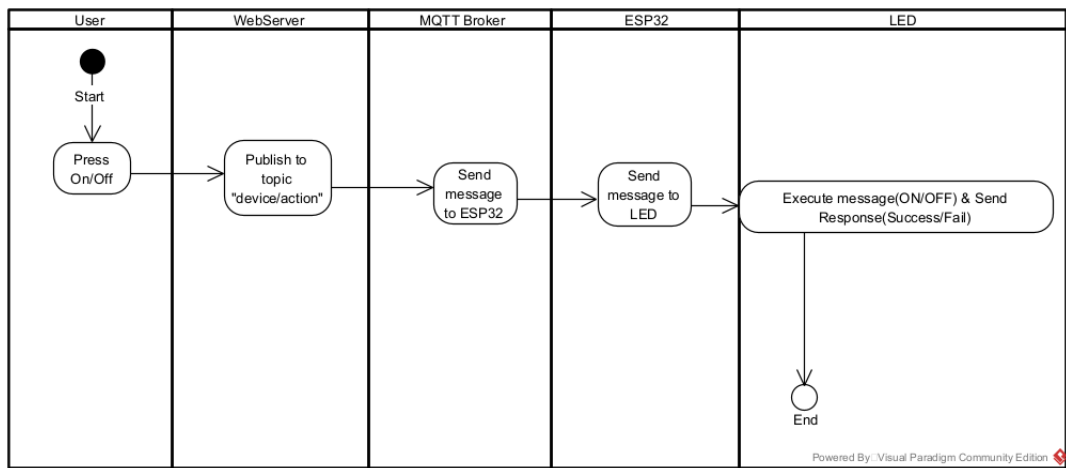
2. Sensor Activity Diagram



Hình 6: Sensor Activity Diagram

1. Sensor bắt đầu, thu thập và gửi dữ liệu.
2. ESP32 nhận dữ liệu và publish lên topic "sensor/data".
3. MQTT Broker chuyển dữ liệu này đến server.
4. Web Server nhận dữ liệu, hiển thị lên trang chủ và lưu trữ.
5. Quy trình kết thúc.

3. Device Activity Diagram

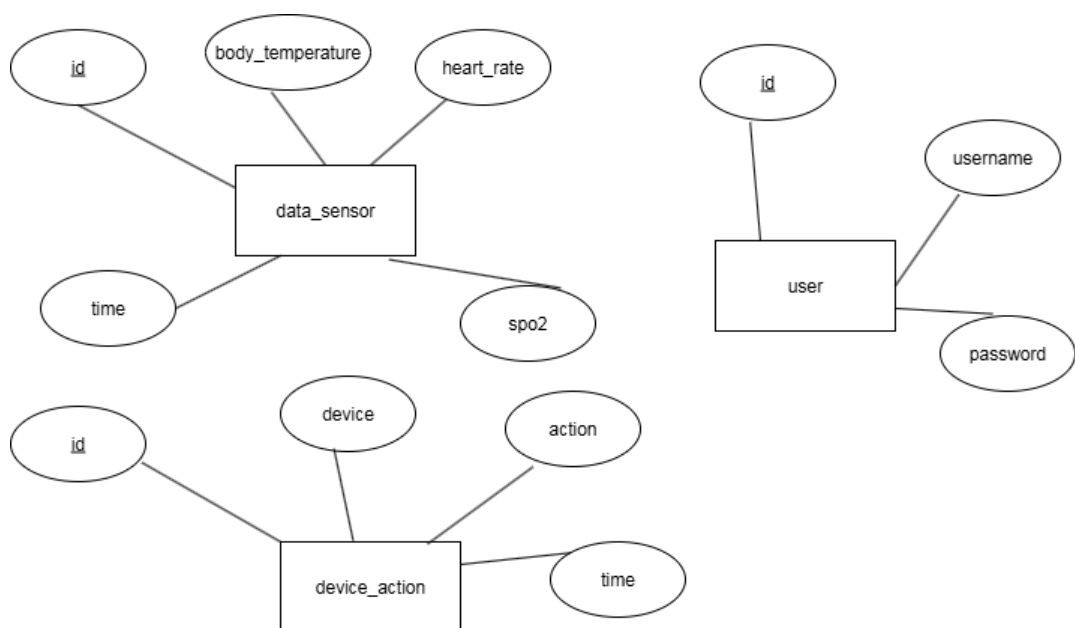


Hình 7: Device Activity Diagram

1. User nhấn nút On/Off.
2. Web Server publish lệnh điều khiển lên topic "device/action".
3. MQTT Broker gửi thông điệp này đến ESP32.
4. ESP32 nhận và gửi lệnh điều khiển tới LED.
5. LED thực thi lệnh (bật/tắt) và gửi phản hồi (thành công/thất bại).
6. Quy trình kết thúc.

C. Thiết kế CSDL


– ERD:



Hình 8: Lược đồ Entity-Relationship


– Bảng data_sensor: Lưu dữ liệu cảm biến ghi nhận được kèm thời

gian

data_sensor	
 id	bigint
body_temperature	double
heart_rate	double
spo2	double
time	datetime

Hình 9: Bảng data_sensor




- + id: bigint (Mã định danh duy nhất cho dữ liệu cảm biến)
 - + body_temperature: double (Nhiệt độ cơ thể được đo bằng cảm biến)
 - + heart_rate: double (Tần số nhịp tim được ghi nhận)
 - + spo2: double (Mức độ bão hòa oxy trong máu)
 - + time: datetime (Thời gian ghi nhận dữ liệu)
- Bảng device_action : Lưu lịch sử điều khiển thiết bị kèm thời gian




device_action	
 id	bigint
device	varchar
action	varchar
time	datetime

Hình 10: Bảng device_action

- + id: bigint (Mã định danh duy nhất cho hành động thiết bị)
- + device: varchar (Tên hoặc mã của thiết bị thực hiện hành động)
- + action: varchar (Loại hành động được thực hiện bởi thiết bị)
- + time: datetime (Thời gian thực hiện hành động)

D. Chuẩn bị phần cứng

Thiết bị	Hình ảnh
Esp32	 <p>Hình 11: Bo mạch ESP32</p>
Cảm biến nhiệt độ DS18B20	 <p>Hình 12: Cảm biến DS18B20</p>
Cảm biến nhịp tim, nồng độ Oxi Max30120	 <p>Hình 13: Cảm biến Max30120</p>

Đèn Led	 <p><i>Hình 14: Đèn Led mini</i></p>
Điện trở	 <p><i>Hình 15: Điện trở</i></p>
Dây jump đực-cái	<p>Đực-Đực 10cm</p>  <p><i>Hình 16: Dây jump đực-cái</i></p>

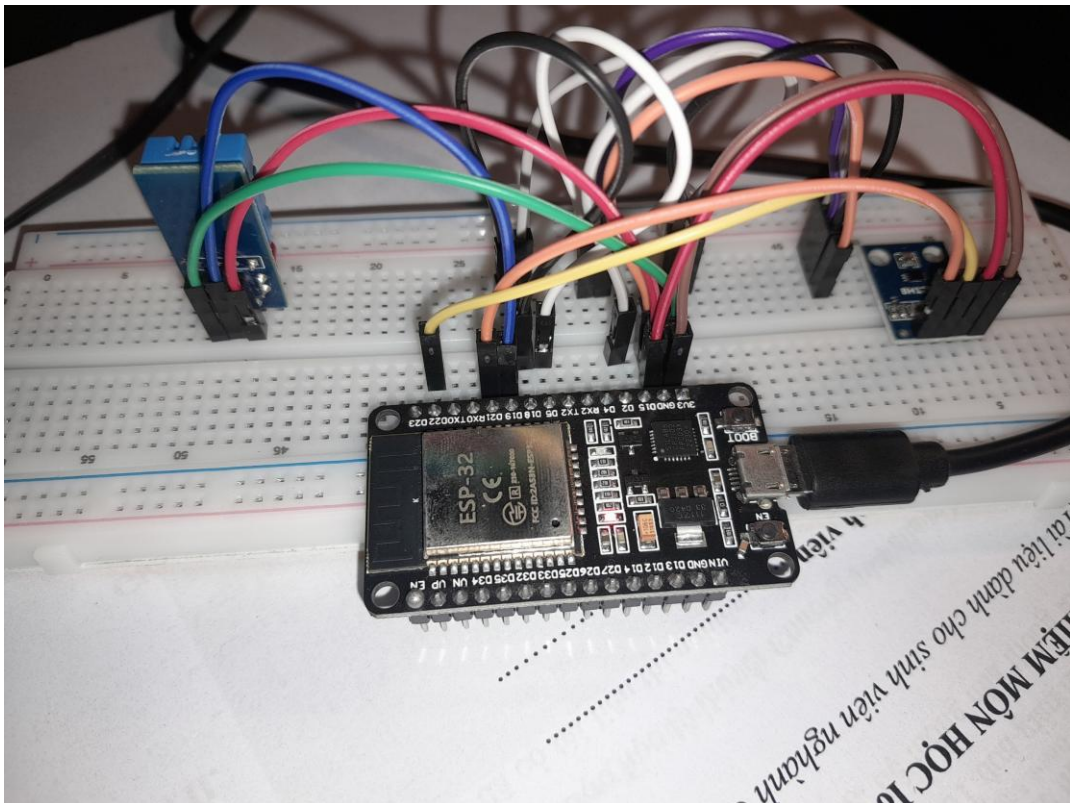
Board test 400 lỗ



Hình 17: Board test 400 lỗ

E. Đấu nối mạch

- Chân GND đấu vào cột (-) trên board test
- Chân 3V3 đấu vào cột (+) trên board test
- Chân Katode của 3 LED đấu vào cột (-)
- Chân Anode của 3 LED nối chung hàng với lần lượt là chân GP26, GP32, GP12
- 3 đèn LED mắc nối tiếp với 3 điện trở
- Dây đen của DS18B20 nối vào cột (-), dây đỏ vào cột (+), dây vàng nối vào chân GP33 có điện trở kéo
- Chân VCC của Max30120 đấu vào cột (+), chân GND vào cột (-), chân SCL đầu cùng hàng GP22, chân SCA đầu cùng hàng GP21.



Hình 18: Đầu nối phân cứng

F. Code Arduino

- Khai báo thư viện, define các biến kết nối wifi, địa chỉ và cổng MQTT Broker, topic để sub/pub dữ liệu

```

1
2 #include <WiFi.h>
3 #include <PubSubClient.h>
4 #include <Wire.h>
5 #include <BH1750.h>
6 #include <ArduinoJson.h>
7 #include "DHT.h"
8
9 // ==== WiFi + MQTT ====
10 const char* ssid = "sufjan stevens";           // tên wifi
11 const char* password = "sufjanstevens";       // password
12 const char* mqtt_server = "10.23.200.4";      // local broker trên máy
13 const int mqtt_port = 1883;
14
15 WiFiClient espClient;
16 PubSubClient client(espClient);

```

Hình 19: Khai báo thư viện, biến

- Khai báo chân kết nối các cảm biến, đèn.

```

18 // DHT11
19 #define DHTPIN 19 // GPIO4 = D4
20 #define DHTTYPE DHT22
21 DHT dht(DHTPIN, DHTTYPE);
22
23 // BH1750
24 BH1750 lightMeter;
25
26 // LED devices
27 #define LED_PIN 2
28 #define AC_PIN 5
29 #define FAN_PIN 18

```

Hình 20: Khai báo chân đèn, cảm biến

- Hàm `call_back` gọi khi nhận dữ liệu điều khiển thiết bị, sau khi nhận message điều khiển -> bật/tắt đèn tương ứng và gửi phản hồi về Broker

Hình 21: Hàm `call_back` được gọi khi nhận message

```

45 void callback(char* topic, byte* payload, unsigned int length) {
46     // Chuyển payload sang chuỗi
47     String msg;
48     for (int i = 0; i < length; i++) {
49         msg += (char)payload[i];
50     }
51     Serial.print("Message arrived [");
52     Serial.print(topic);
53     Serial.print("] ");
54     Serial.println(msg);
55
56     // Parse JSON
57     StaticJsonDocument<200> doc;
58     deserializeJson(doc, msg);
59
60     // nếu json có key led
61     if (doc.containsKey("led")) {
62         int ledState = doc["led"];
63         digitalWrite(LED_PIN, ledState ? HIGH : LOW);
64         Serial.print("LED set to: ");
65         Serial.println(ledState);
66     }
67
68     // nếu json có key "ac"
69     if (doc.containsKey("ac")) {
70         int acState = doc["ac"];
71         digitalWrite(AC_PIN, acState ? HIGH : LOW);
72         Serial.print("AC set to: ");
73         Serial.println(acState);
74     }
75
76     // nếu json có key "fan"
77     if (doc.containsKey("fan")) {
78         int fanState = doc["fan"];
79         digitalWrite(FAN_PIN, fanState ? HIGH : LOW);
80         Serial.print("Fan set to: ");
81         Serial.println(fanState);
82     }

```

– Hàm Setup

+ Setup cho kết nối wifi, cảm biến và khai báo chân đèn ở pinMode Output

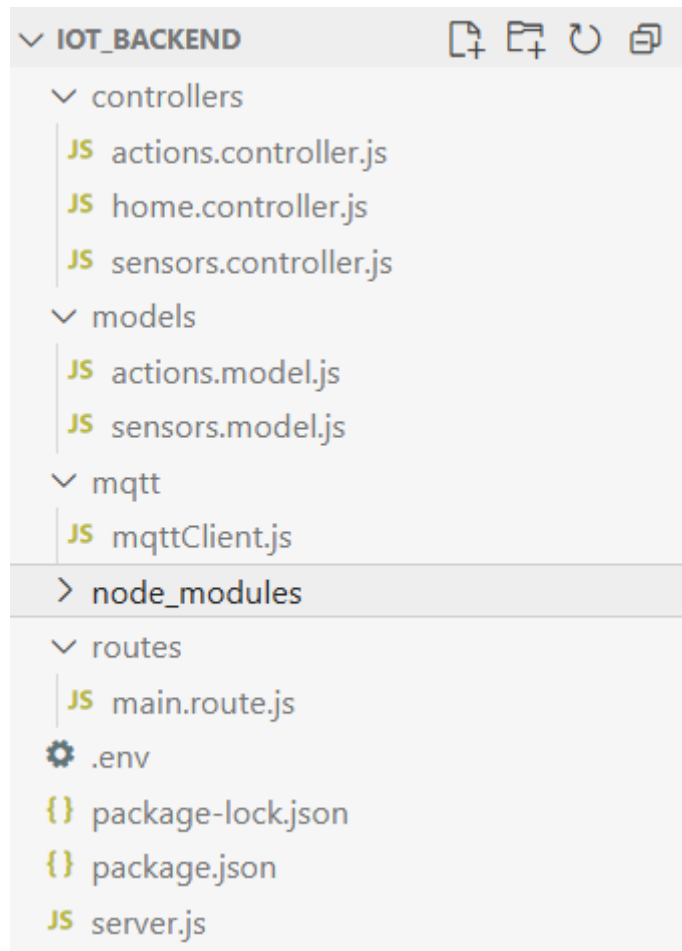
```

104 // ==== Setup ====
105 void setup() {
106     Serial.begin(115200);
107
108     pinMode(LED_PIN, OUTPUT);
109     pinMode(AC_PIN, OUTPUT);
110     pinMode(FAN_PIN, OUTPUT);
111
112     digitalWrite(LED_PIN, LOW);
113     digitalWrite(AC_PIN, LOW);
114     digitalWrite(FAN_PIN, LOW);
115
116     setup_wifi();
117
118     client.setServer(mqtt_server, mqtt_port);
119     client.setCallback(callback);
120
121     dht.begin();
122     Wire.begin(21, 22); // SDA=21, SCL=22
123     lightMeter.begin();
124
125 }

```

Hình 22: Setup wifi, cảm biến

G. Code Back-end (Node/Express)



Hình 23: Cấu trúc mã nguồn

1. Các module chính

- Module Dữ liệu Sensor (Sensor Data Management)
 - + Controller: DataSensorController
 - + Service: DataSensorServiceImpl
 - + Repository: DataSensorRepository
 - + Chức năng:
 - Thu thập dữ liệu sensor từ ESP32
 - Lưu trữ dữ liệu vào database
 - Truy vấn dữ liệu với phân trang
 - Tìm kiếm dữ liệu theo nhiều tiêu chí
 - Lấy dữ liệu 24h hoặc 5 phút gần nhất
- Module Điều khiển Thiết bị (Device Control)
 - + Entity: DeviceAction (id, device, action, time)
 - + Controller: DeviceActionController
 - + Service: DeviceActionServiceImpl

- + Repository: DeviceActionRepository
- + Chức năng:
 - Điều khiển LED qua MQTT
 - Lưu trữ lịch sử hành động thiết bị
 - Tìm kiếm lịch sử thiết bị
 - Lấy trạng thái thiết bị mới nhất
- Module MQTT Communication
 - + Service: MqttManagerService
 - + Config: MqttConfig
 - + Chức năng:
 - Kết nối MQTT Broker
 - Subscribe topics: data/sensor, state/device
 - Publish commands đến thiết bị
 - Xử lý dữ liệu real-time từ sensors
- Module WebSocket Real-time
 - + Handler: SensorWebSocketHandler
 - + Config: WebSocketConfig
 - + Chức năng:
 - Kết nối WebSocket với Frontend
 - Gửi dữ liệu sensor real-time
 - Broadcast thông báo lỗi
 - Cập nhật trạng thái MQTT

H. API

1. Get latest data sensor

- + GET request để lấy về thông tin gần nhất của 3 cảm biến nhiệt độ, độ ẩm, ánh sáng. GET <http://localhost:5000/api/home/latest>
- + Header Request:

Body	Headers (8)	200 OK
X-Powered-By	Express	
Access-Control-Allow-Origin	*	
Content-Type	application/json; charset=utf-8	
Content-Length	123	
ETag	W/"7b-dnXifXLNOAM1gdosuRkMSbkfqtI"	
Date	Fri, 31 Oct 2025 02:46:11 GMT	
Connection	keep-alive	
Keep-Alive	timeout=5	

Hình 24: Header Request

+ Response:

Example Request

Get latest data sensor

curl

curl --location 'http://localhost:5000/api/home/latest'

Example Response

Body Headers (8) 200 OK

```
{
  "_id": "690422f206c99f6a36ef0ec7",
  "temperature": 28.7,
  "humidity": 74.5,
  "light": 245,
  "time": "2025-10-31T02:46:10.530Z",
  "__v": 0
}
```

Hình 25: Response của API Get latest data sensor

2. Get 10 latest data sensor

GET request để lấy về 10 bản ghi gần nhất của 3 cảm biến nhiệt độ, độ ẩm, ánh sáng. GET

<http://localhost:5000/api/home/tenlatest>

+ GET <http://localhost:5000/api/home/tenlatest>

+ Header Request:

Body	Headers (8)	200 OK
X-Powered-By	Express	
Access-Control-Allow-Origin	*	
Content-Type	application/json; charset=utf-8	
Content-Length	831	
ETag	W/"33f-AwGuJvYk3pITTVs/fC/dcjMcN4c"	
Date	Fri, 31 Oct 2025 02:52:34 GMT	
Connection	keep-alive	
Keep-Alive	timeout=5	

Hình 26: Header Request

+ Response:

Example Response
<pre>[{ "temperature": 29.2, "humidity": 73.6, "light": 245, "time": "2025-10-31T02:52:14.708Z" }, { "temperature": 29.2, "humidity": 73.6, "light": 245, "time": "2025-10-31T02:52:16.724Z" }, { "temperature": 29.2, "humidity": 73.5, "light": 245, "time": "2025-10-31T02:52:18.736Z" }, { "temperature": 29.2, "humidity": 73.5, "light": 245, "time": "2025-10-31T02:52:20.756Z" }, { "temperature": 29.2, "humidity": 73.5, "light": 245, "time": "2025-10-31T02:52:22.707Z" },]</pre>

Hình 27: Response của API lấy 10 bản ghi dữ liệu gần nhất

3. Get history of sensors

- + API quản lý lịch sử sensor với các tính năng tìm kiếm, lọc, sắp xếp và phân trang
- + GET <http://localhost:5000/api/sensors>

+ Params:

Parameter	Type	Default	Description
<code>search</code>	string	-	Giá trị tìm kiếm (số cho sensor, datetime cho time)
<code>searchType</code>	string	<code>all</code>	Loại tìm kiếm: <code>all</code> , <code>temperature</code> , <code>humidity</code> , <code>light</code> , <code>time</code>
<code>sortBy</code>	string	<code>time</code>	Sắp xếp theo: <code>temperature</code> , <code>humidity</code> , <code>light</code> , <code>time</code>
<code>sortOrder</code>	string	<code>asc</code>	Thứ tự: <code>asc</code> (tăng dần), <code>desc</code> (giảm dần)
<code>page</code>	number	1	Trang hiện tại
<code>limit</code>	number	10	Số bản ghi/trang (max: 100)

Hình 27: Parameters của API

+ Header Request:

Body	Headers (8)	200 OK
	X-Powered-By	Express
	Access-Control-Allow-Origin	*
	Content-Type	application/json; charset=utf-8
	Content-Length	1466
	ETag	W/"5ba-9ql7vYsZIO7McVXYRFJMHRmsoc"
	Date	Fri, 31 Oct 2025 02:57:36 GMT
	Connection	keep-alive
	Keep-Alive	timeout=5

Hình 28 : Header Request

+ Response:

```

{
  "success": true,
  "data": [
    {
      "_id": "68f1b051c49b3b1b76270e00",
      "id": 7890,
      "temperature": 30,
      "humidity": 83.8,
      "light": 50,
      "time": "2025-10-17T02:56:17.867Z",
      "__v": 0
    },
    {
      "_id": "68f1b059c49b3b1b76270e0c",
      "id": 7894,
      "temperature": 30,
      "humidity": 83.9,
      "light": 50,
      "time": "2025-10-17T02:56:25.840Z",
      "__v": 0
    },
    {
      "_id": "68f1c53cc49b3b1b762721a5",
      "id": 9442,
      "temperature": 30.2,
      "humidity": 77.8,
      "light": 50,
      "time": "2025-10-17T04:25:32.096Z",
      "__v": 0
    }
  ]
}

```

Hình 29: Response của API với params

<http://localhost:5000/api/sensors?searchType=all&search=50>

4. Get history of actions

- + API trả về một các data liên quan đến lịch sử hoạt động của 3 thiết bị led/fan/ac.GET

<http://localhost:5000/api/actions>

- + Params:

Parameter	Type	Default	Description
device	string	all	Loại thiết bị: led , fan , ac , all
action	string	all	Trạng thái: on , off , all
search	string	-	Tìm kiếm thời gian (định dạng linh hoạt)
sortOrder	string	asc	Sắp xếp: asc (tăng dần), desc (giảm dần)
page	number	1	Trang hiện tại
limit	number	10	Số bản ghi/trang

Hình 30:Param

- + Header Request:

Body	Headers (8)	200 OK
X-Powered-By	Express	
Access-Control-Allow-Origin	*	
Content-Type	application/json; charset=utf-8	
Content-Length	975	
ETag	W/"3cf-ToeyrQEKMGUUt2/G6LUFA6Vnphl"	
Date	Fri, 31 Oct 2025 03:03:34 GMT	
Connection	keep-alive	
Keep-Alive	timeout=5	

Hình 31 : Header Request

+ Response:

```
{
  "success": true,
  "data": [
    {
      "_id": "68d4babafbd99173272cc4f6",
      "id": 10,
      "device": "led",
      "action": "off",
      "time": "2025-09-24T12:45:00.000Z"
    },
    {
      "_id": "68d4babafbd99173272cc4fc",
      "id": 16,
      "device": "led",
      "action": "off",
      "time": "2025-09-24T13:15:00.000Z"
    },
    {
      "_id": "68d4babafbd99173272cc502",
      "id": 22,
      "device": "led",
      "action": "off",
      "time": "2025-09-24T13:45:00.000Z"
    },
    {
      "_id": "68d4babafbd99173272cc508",
      "id": 28,
      "device": "led",
      "action": "off",
      "time": "2025-09-24T14:15:00.000Z"
    }
  ]
}
```

Hình 32: Response của API với params

<http://localhost:5000/api/actions?device=led&action=off&search=2025-09-24>

5. Get device status

- + GET request trả về trạng thái gần nhất của 3 thiết bị led/fan/ac.động của 3 thiết bị led/fan/ac
- + GET: <http://localhost:5000/api/home/devicestatus>
- + Header Request:

Body	Headers (8)	200 OK
	X-Powered-By	Express
	Access-Control-Allow-Origin	*
	Content-Type	application/json; charset=utf-8
	Content-Length	158
	ETag	W/"9e-W4LQ1S0Brt5/JrT305KsGoM6LXk"
	Date	Fri, 31 Oct 2025 03:45:07 GMT
	Connection	keep-alive
	Keep-Alive	timeout=5

Hình 33: Header Request

+ Response:

```
{
  "success": true,
  "status": {
    "ac": false,
    "fan": false,
    "led": false
  },
  "lastUpdated": "2025-10-31T03:45:07.979Z",
  "message": "Current device status fetched successfully"
}
```

Hình 34: Response của API

<http://localhost:5000/api/home/devicestatus>

6. Toggle devices

- + POST request bật tắt 3 thiết bị led/ac/fan
- + GET: <http://localhost:5000/api/home/toggle>
- + Header Request:

Example Response	
Body	Headers (8)
200 OK	
X-Powered-By	Express
Access-Control-Allow-Origin	*
Content-Type	application/json; charset=utf-8
Content-Length	180
ETag	W/"b4-N5TEv+4nrvxMe1yOE42n62KwbOI"
Date	Fri, 31 Oct 2025 03:49:52 GMT
Connection	keep-alive
Keep-Alive	timeout=5

Hình 35: Header Request

+ Body JSON:

```
--data '{
  "device": "led",
  "action": "off"
}'
```

+ Response:

Body	Headers (8)
200 OK	
<pre>{ "success": true, "message": "Action executed and confirmed by ESP32", "data": { "device": "led", "action": "on", "time": "2025-10-31T03:49:52.866Z", "_id": "690431e006c99f6a36ef1e5b", "__v": 0 } }</pre>	

Hình 36: Response của API <http://localhost:5000/api/home/toggle>

IV. Đánh giá kết quả

A. Chức năng đã hoàn thành

1. Thu thập và hiển thị dữ liệu cảm biến

- Hệ thống đã thành công trong việc thu thập dữ liệu cảm

biến thông qua ESP32 với các loại dữ liệu bao gồm nhiệt độ cơ thể, nhịp tim và nồng độ oxy trong máu (SpO2). Dữ liệu được publish qua MQTT broker và lưu trữ thành công trong database MySQL.

- Hệ thống hiển thị dữ liệu real-time cho người dùng thông qua WebSocket, cho phép quan sát các chỉ số sức khỏe theo thời gian thực. Người dùng có thể truy xuất dữ liệu lịch sử thông qua các API RESTful với khả năng phân trang và tìm kiếm linh hoạt.

2. Điều khiển thiết bị từ xa

- Hệ thống cho phép người dùng điều khiển các thiết bị LED thông qua giao diện Web một cách trực quan. Lệnh điều khiển được truyền theo luồng từ Frontend đến Backend, sau đó đến MQTT Broker và cuối cùng đến ESP32 để thực thi.

- Quá trình điều khiển có khả năng thực thi tức thời với độ trễ rất thấp do các thành phần được kết nối trong cùng mạng LAN.

3. Lưu trữ và hiển thị lịch sử trạng thái thiết bị

- Hệ thống có khả năng ghi nhận và lưu trữ tất cả các hành động điều khiển thiết bị cùng với thời gian thực hiện.

- Dữ liệu lịch sử được tổ chức theo cấu trúc rõ ràng với thông tin về thiết bị, hành động và thời gian, cho phép người dùng theo dõi và phân tích các hoạt động trong quá khứ.

4. Xác thực và bảo mật người dùng

- Hệ thống tích hợp JWT authentication cho phép đăng ký và đăng nhập người dùng một cách an toàn.

- Mật khẩu được mã hóa bằng BCrypt và tất cả các API được bảo vệ bằng token authentication.

- CORS được cấu hình để đảm bảo giao tiếp an toàn giữa Frontend và Backend.

5. API linh hoạt và tìm kiếm nâng cao

- Các API truy xuất dữ liệu có khả năng lọc, phân trang và tìm kiếm đa tiêu chí nhằm tối ưu hiệu năng và trải nghiệm người dùng.

- Hệ thống hỗ trợ tìm kiếm theo thời gian, giá trị số và kết hợp nhiều điều kiện khác nhau để đáp ứng nhu cầu truy vấn phức tạp.

B. Đánh giá hiệu suất

1. Đánh giá độ chính xác của cảm biến

- Đối với nhiệt độ cơ thể, độ chính xác dao động trong khoảng 0.1 độ C với khả năng phát hiện các giá trị không hợp lệ.
- Nhịp tim được đo với độ chính xác cao và có thể xử lý các trường hợp dữ liệu N/A khi cảm biến không đọc được giá trị. SpO2 được đo với độ chính xác phù hợp cho mục đích giám sát sức khỏe cơ bản.

2. Đánh giá về tốc độ phản hồi của hệ thống

- Các API truy vấn đơn giản có thời gian phản hồi dưới 100ms nhờ vào cấu trúc database được tối ưu và sử dụng JPA/Hibernate hiệu quả.
- Đối với các truy vấn phức tạp như tìm kiếm đa tiêu chí và phân trang, thời gian phản hồi trung bình dưới 200ms. WebSocket có khả năng gửi dữ liệu real-time với độ trễ tối thiểu, tuy nhiên có thể bị ảnh hưởng bởi chất lượng kết nối mạng.

3. Độ trễ điều khiển thiết bị

- Thời gian từ lúc người dùng gửi yêu cầu điều khiển đến lúc thiết bị thực sự thay đổi trạng thái gần như tức thời, thường dưới 50ms. Điều này đạt được nhờ việc sử dụng MQTT với QoS 0 và tất cả các thành phần được kết nối trong cùng mạng LAN.

4. Cập nhật dữ liệu real-time

- Dữ liệu sensor được cập nhật và hiển thị trên giao diện người dùng một cách tức thời thông qua WebSocket.
- Hệ thống có khả năng xử lý và broadcast dữ liệu đến nhiều client đồng thời mà không gây tắc nghẽn.

C. Điểm cần cải thiện

1. Xử lý lỗi chưa toàn diện

- Hệ thống thiếu global exception handler và structured error responses, có thể dẫn đến trải nghiệm người dùng không nhất quán khi xảy ra lỗi.

2. Thiếu monitoring và logging

- Hệ thống chưa có cơ chế monitoring hiệu suất và logging chi tiết, gây khó khăn trong việc debug và tối ưu hóa.

3. Bảo mật cần tăng cường

- Thiếu rate limiting cho API và chưa có cơ chế refresh token, có thể dẫn đến các vấn đề bảo mật trong môi trường production.

D. Đề xuất cải tiến

1. Tăng cường bảo mật và hiệu suất

- Triển khai rate limiting cho tất cả API endpoints để ngăn chặn tấn công DDoS. Thêm cơ chế refresh token và session management nâng cao. Tích hợp monitoring tools như Prometheus và Grafana để theo dõi hiệu suất hệ thống.

2. Mở rộng khả năng phân tích dữ liệu

- Phát triển các thuật toán phân tích xu hướng và phát hiện bất thường trong dữ liệu sức khỏe. Tích hợp machine learning để dự đoán các vấn đề sức khỏe tiềm ẩn dựa trên dữ liệu lịch sử.

3. Hỗ trợ đa thiết bị và đa người dùng

- Mở rộng hệ thống để hỗ trợ nhiều thiết bị IoT khác nhau và quản lý nhiều người dùng với các quyền hạn khác nhau. Tích hợp device discovery và auto-configuration.

4. Phát triển ứng dụng di động

- Xây dựng ứng dụng di động native hoặc cross-platform để người dùng có thể theo dõi sức khỏe mọi lúc mọi nơi. Tích hợp push notification để cảnh báo khi có dữ liệu bất thường.