API DOC

1. Tổng quan

Đây là API dành cho việc điều khiển các thiết bị và thu thập dữ liệu cảm biến từ mạch ESP8266 thông qua MQTT. API này cho phép điều khiển các thiết bị như đèn, quạt, TV và ghi nhận các thông tin về trạng thái thiết bị cũng như dữ liệu cảm biến (nhiệt độ, độ ẩm, độ sáng).

2. Cấu trúc API

2.1. Action Log API

Endpoint: `/action-log`

Mô tả: Ghi nhận lịch sử các hành động điều khiển thiết bị.

Phương thức: `GET`

- **URL**: `localhost:6060/action-log`
- **Mô tả**: Trả về danh sách các hành động đã ghi nhận (bật/tắt thiết bị) và render ra trang actions_log.ejs.
- **Response mẫu**:

```
{

  result: [

    {

      id: 1,

      device_name: 'light',

      action: 'off',

      action_time: 2024-09-02T18:43:32.000Z

    },

    {

      id: 2,

      device_name: 'fan',

      action: 'on',
```

```
      action_time: 2024-09-02T18:43:33.000Z
    }
  ],
  pagination: { page: 0, page_size: 2, total_count: 130 },
  filters: { from: null, to: null }
}
```

2.2. History API

Endpoint: `/history`

Mô tả: Cung cấp lịch sử dữ liệu cảm biến như nhiệt độ, độ ẩm.

Phương thức: `GET`

- **URL**: `localhost:6060/history`
- **Mô tả**: Trả về danh sách các bản ghi cảm biến, bao gồm nhiệt độ, độ ẩm, và ánh sáng và render ra trang history.ejs.
- **Response mẫu**:

```
{
  historyData: [
    {
      id: 3375,
      time: 2024-09-22T16:58:56.000Z,
      temperature: 29.3,
      humidity: 92,
      light: 3009.96
    },
    {
      id: 3374,
      time: 2024-09-22T16:58:53.000Z,
```

```
      temperature: 29.3,
      humidity: 92,
      light: 3019.63
    }
  ],
  filters: {
    temperatureFrom: null,
    temperatureTo: null,
    humidityFrom: null,
    humidityTo: null,
    lightFrom: null,
    lightTo: null,
    timeFrom: null,
    timeTo: null
  },
  pagination: { page: 8, page_size: 2, total_count: 3391 }
}
```

2.3. Home API

Endpoint: `/home`

Mô tả: Cung cấp thông tin về trạng thái hiện tại của các thiết bị trong hệ thống.

Phương thức: `GET`

- **URL**: `localhost:6060/`
- **Mô tả**: Trả về trạng thái hiện tại của các thiết bị như đèn, quạt, TV và render trang home.ejs.
- **Response mẫu**:

```
{
  "light": "off",
  "fan": "on",
```

```
  "television": "off"
}
```

2.4. Profile API

Endpoint: `/profile`

Mô tả: Quản lý thông tin người dùng liên quan đến hệ thống.

Phương thức: `GET`

- **URL**: `localhost:6060/profile`
- **Mô tả**: Render ra trang profile hiển thị thông tin người làm project.

3. MQTT Integration

Hệ thống sử dụng MQTT để giao tiếp giữa server và ESP8266 cho việc điều khiển thiết bị và thu thập dữ liệu cảm biến.

3.1. Subscribe to MQTT Topics

Các topic được hệ thống subscribe để nhận dữ liệu từ các thiết bị ESP8266:

- **Topic điều khiển đèn**: `TOPIC_LIGHT_RES_STATUS_SUB_ESP8266`
- **Topic điều khiển quạt**: `TOPIC_FAN_RES_STATUS_SUB_ESP8266`
- **Topic điều khiển TV**: `TOPIC_TELEVISION_RES_STATUS_SUB_ESP8266`
- **Topic cảm biến nhiệt độ, độ ẩm và ánh sáng**: `TOPIC_TEM_HUMI_SUB_ESP8266`

Khi nhận được thông tin từ các topic này, hệ thống sẽ:

- Cập nhật trạng thái thiết bị tương ứng vào cơ sở dữ liệu (`DeviceStatus.updateStatus`).
- Ghi lại hành động vào `ActionLog`.
- Gửi dữ liệu đến front-end thông qua `Socket.IO`.

3.2. Publish to MQTT Topics

Hệ thống có thể điều khiển các thiết bị thông qua các topic MQTT:

- **Topic điều khiển đèn**: `TOPIC_LIGHT_REQ_PUB_ESP8266`
- **Topic điều khiển quạt**: `TOPIC_FAN_REQ_PUB_ESP8266`
- **Topic điều khiển TV**: `TOPIC_TELEVISION_REQ_PUB_ESP8266`

Ví dụ: khi người dùng trên giao diện web bật đèn, hệ thống sẽ gửi lệnh đến topic tương ứng để điều khiển đèn.

4. Socket.IO Events

4.1. Event: Light Control

- **Event**: `TOPIC_LIGHT_CONTROL_SUB_FRONT`
- **Mô tả**: Điều khiển bật/tắt đèn từ giao diện người dùng.
- **Dữ liệu**: `{ "status": "on" }`
- **Hành động**: Gửi yêu cầu qua MQTT để bật/tắt đèn và phát sự kiện trạng thái cho front-end.

4.2. Event: Fan Control

- **Event**: `TOPIC_FAN_CONTROL_SUB_FRONT`
- **Mô tả**: Điều khiển bật/tắt quạt từ giao diện người dùng.
- **Dữ liệu**: `{ "status": "off" }`
- **Hành động**: Gửi yêu cầu qua MQTT để bật/tắt quạt và phát sự kiện trạng thái cho front-end.

4.3. Event: Television Control

- **Event**: `TOPIC_TELEVISION_CONTROL_SUB_FRONT`
- **Mô tả**: Điều khiển bật/tắt TV từ giao diện người dùng.
- **Dữ liệu**: `{ "status": "on" }`
- **Hành động**: Gửi yêu cầu qua MQTT để bật/tắt TV và phát sự kiện trạng thái cho front-end.

5. Data Models

5.1. ActionLog

Ghi lại lịch sử các hành động điều khiển thiết bị.

- **Trường**:
  - `device` (string): Tên thiết bị.
  - `action` (string): Hành động (bật/tắt).
  - `timestamp` (datetime): Thời gian hành động.

5.2. DeviceStatus

Lưu trữ trạng thái hiện tại của các thiết bị trong hệ thống.

- **Trường**:
  - `device` (string): Tên thiết bị.
  - `status` (string): Trạng thái hiện tại của thiết bị (bật/tắt).

5.3. DataLog

Lưu trữ dữ liệu cảm biến.

- **Trường**:
  - `time` (datetime): Thời gian thu thập dữ liệu.
  - `temperature` (float): Nhiệt độ (°C).
  - `humidity` (float): Độ ẩm (%).
  - `lux` (float): Cường độ ánh sáng (lux).

6. Kết luận

Tài liệu này mô tả các chức năng chính của API cho hệ thống IoT điều khiển thiết bị và thu thập dữ liệu cảm biến. Các API cho phép giao tiếp hai chiều giữa front-end và back-end thông qua HTTP, MQTT, và Socket.IO.