HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH
KHOA KỸ THUẬT ĐIỆN TỬ 2

BÀI TẬP LỚN

Môn học: MẠNG CẢM BIẾN

GVGD: HỒ NHỰT MINH

SVTH: Dương Đình Khánh - N21DCDK013
Nguyễn Văn Phú - N21DCDK019
Trần Đăng Quang- N21DCDK025
Đoàn Nhật Tân - N21DCDK028
Trần Văn Thái - N21DCDK030

Lớp: D21CQDK01-N

TP. HỒ CHÍ MINH - 2024

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH

KHOA KỸ THUẬT ĐIỆN TỬ 2

BÀI TẬP LỚN

Môn học: MẠNG CẢM BIẾN

GVGD: HỒ NHỰT MINH

Lớp:

STT	HỌ VÀ TÊN SINH VIÊN	MSSV	CHỮ KÝ
1	Dương Đình Khánh	N21DCDK013	
2	Nguyễn Văn Phú	N21DCDK019	
3	Trần Đăng Quang	N21DCDK025	
4	Đoàn Nhật Tân	N21DCDK028	
5	Trần Văn Thái	N21DCDK030	

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TP. HCM, ngày tháng năm

(Ký và ghi rõ họ tên)

Hồ Nhựt Minh

LỜI CẢM ƠN

Kính gửi thầy Hồ Nhựt Minh,

Nhóm em xin gửi lời cảm ơn chân thành đến thầy vì những gì thầy đã tận tâm giảng dạy và hướng dẫn trong suốt môn học vừa qua. Những bài giảng của thầy không chỉ mang đến kiến thức chuyên môn sâu sắc mà còn giúp em hiểu rõ hơn về cách áp dụng chúng vào thực tiễn.

Sự nhiệt huyết, tận tâm và phong cách giảng dạy truyền cảm hứng của thầy đã để lại trong em nhiều ấn tượng và động lực để tiếp tục học tập và phát triển. Em thật sự trân trọng những giờ học thú vị và bổ ích mà thầy mang lại.

Chúc thầy luôn mạnh khỏe, hạnh phúc và tiếp tục thành công trên con đường giảng dạy. Mong rằng em sẽ có cơ hội được học tập và lắng nghe những chia sẻ của thầy trong tương lai.

Trân trọng

MỤC LỤC

MỤC LỤC HÌNH ẢNH

CHƯƠNG 1: TỔNG QUAN VÀ PHÂN TÍCH YÊU CẦU

1.1. Giới thiệu bài toán

Mô tả vấn đề cần giải quyết

Giàn phơi quần áo truyền thống yêu cầu người dùng phải thu vào hoặc đẩy ra bằng tay, đặc biệt là khi thời tiết thay đổi bất chợt như trời mưa. Điều này gây bất tiện cho người sử dụng, đặc biệt khi họ không có mặt tại nhà.

Đề tài này xây dựng một hệ thống giàn phơi thông minh, cho phép giàn phơi tự động hoạt động dựa trên cảm biến mưa và điều khiển từ xa thông qua giao diện web. Hệ thống còn ghi lại lịch sử hoạt động, bao gồm trạng thái mưa và các lệnh điều khiển, lưu trữ trong cơ sở dữ liệu và hỗ trợ xuất ra file Excel.

Phạm vi và giới hạn của đề tài

- **Phạm vi:**
  - Xây dựng hệ thống điều khiển động cơ giàn phơi sử dụng vi điều khiển ESP32.
  - Giao tiếp dữ liệu giữa ESP32 và server thông qua giao thức HTTP.
  - Lưu trữ log hoạt động và dữ liệu cảm biến trên MongoDB.
  - Thiết kế giao diện web cho người dùng điều khiển hệ thống và theo dõi trạng thái hoạt động.

- **Giới hạn:**
  - Hệ thống hoạt động trong mạng nội bộ (LAN) và chưa hỗ trợ điều khiển từ xa qua Internet.
  - Chỉ ghi nhận trạng thái **trời mưa** hoặc **không mưa** thông qua cảm biến mưa.
  - Động cơ chỉ thực hiện các lệnh đơn giản: **Move Out**, **Move In**, **Stop**.

Mục tiêu và yêu cầu chính

Mục tiêu:
Xây dựng một hệ thống giàn phơi tự động thông minh với khả năng:

1. **Tự động hóa:** Thu giàn phơi vào khi trời mưa và đẩy ra khi trời nắng.
2. **Điều khiển thủ công:** Cho phép người dùng điều khiển từ xa thông qua giao diện web.
3. **Lưu trữ dữ liệu:** Ghi nhận và lưu trữ lịch sử hoạt động (log, trạng thái mưa, lệnh điều khiển) vào MongoDB.
4. **Xuất báo cáo:** Xuất dữ liệu lịch sử hoạt động ra file Excel để theo dõi và phân tích.

Yêu cầu chính:

- Kết nối ổn định giữa ESP32 và server.
- Giao diện web dễ sử dụng, cập nhật dữ liệu trong thời gian thực.
- Hệ thống phản hồi nhanh.

1.2. Phân tích yêu cầu hệ thống

Yêu cầu chức năng

- **Tự động điều khiển động cơ:**
  - Thu giàn phơi khi phát hiện trời mưa.
  - Đẩy giàn phơi ra khi trời không mưa.
- **Điều khiển từ xa:**
  - Gửi lệnh **Move Out**, **Move In**, **Stop**, **Toggle Auto Mode** từ giao diện web.
- **Lưu trữ dữ liệu hoạt động:**
  - Ghi nhận log hoạt động bao gồm trạng thái động cơ và cảm biến mưa.
  - Lưu lệnh điều khiển và trạng thái mưa vào MongoDB.
- **Hiển thị và báo cáo:**
  - Hiển thị trạng thái động cơ và trạng thái mưa trên giao diện web.
  - Xuất lịch sử hoạt động thành file Excel.

Yêu cầu phi chức năng

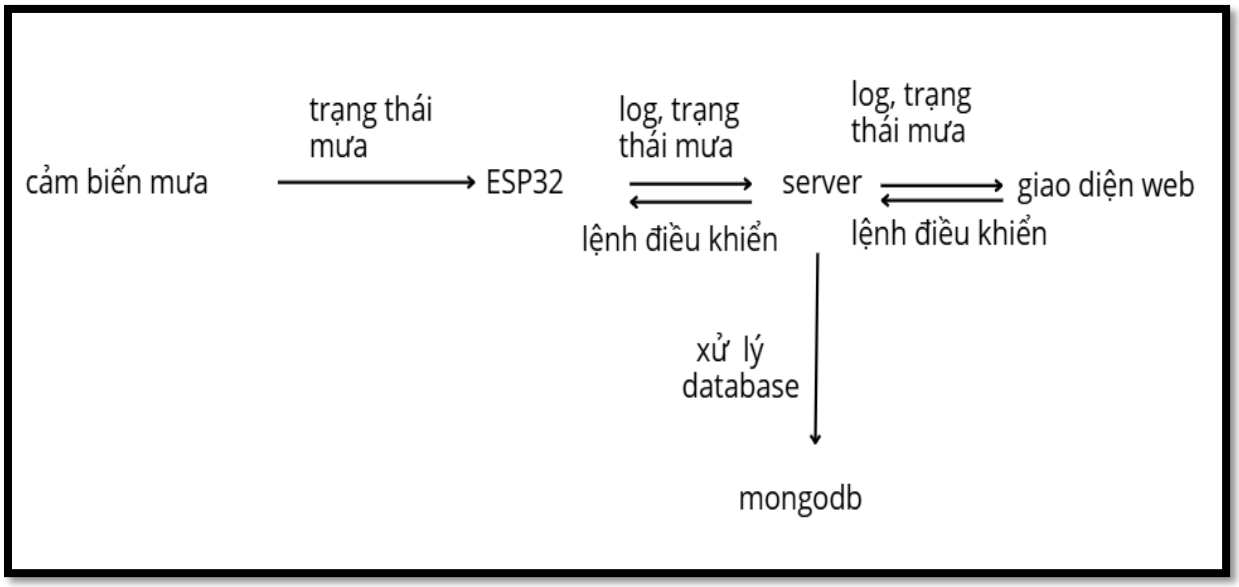
- **Thời gian phản hồi:** Hệ thống phản hồi lệnh từ người dùng trong vòng 2 giây.
- **Độ tin cậy:** Hệ thống đảm bảo hoạt động ổn định, hạn chế sai sót.
- **Khả năng bảo trì:** Dễ dàng bảo trì và mở rộng thêm chức năng mới.
- **Tính đơn giản:** Giao diện người dùng dễ sử dụng và trực quan.

Các ràng buộc và giới hạn

- Hệ thống hoạt động trên mạng nội bộ (LAN).
- Việc xử lý log và trạng thái phụ thuộc vào kết nối giữa ESP32 và server.
- Cảm biến mưa chỉ xác định trạng thái mưa hoặc không mưa.

1.3. Kiến trúc tổng thể

Sơ đồ khối hệ thống



Hình 1. Sơ đồ khối hệ thống

Các thành phần chính

- **ESP32:**
  - Kết nối cảm biến mưa và điều khiển động cơ thông qua GPIO.
  - Giao tiếp với server bằng HTTP (POST/GET).
- **Server Node.js:**
  - API RESTful để nhận dữ liệu từ ESP32 và cung cấp dữ liệu cho giao diện web.
  - Lưu trữ và xử lý dữ liệu log trong MongoDB.
  - Xuất dữ liệu thành file Excel.
- **MongoDB:**

- Collection lưu trữ **logs**, **commandStatus**, và **rainStatus**.

- **Giao diện web:**

  - Hiển thị log hoạt động, trạng thái mưa và các nút điều khiển hệ thống.
  - Tự động cập nhật dữ liệu từ server mỗi 2 giây.

Luồng dữ liệu và tương tác

1. **ESP32 → Server Node.js**: Gửi log và trạng thái mưa (HTTP POST).
2. **Server Node.js → MongoDB**: Lưu log, trạng thái mưa, và lệnh điều khiển.
3. **Giao diện Web → Server Node.js**: Gửi lệnh điều khiển (HTTP POST).
4. **Server Node.js → ESP32**: Gửi lệnh điều khiển mới nhất (HTTP GET).
5. **Server Node.js → Giao diện Web**: Gửi log và trạng thái mưa (HTTP GET).
6. **MongoDB → Server Node.js**: Truy xuất dữ liệu log, lệnh điều khiển, và trạng thái mưa khi cần.

1.4. Các công nghệ sử dụng

Vi điều khiển và nền tảng phát triển

- **ESP32**:

Vi điều khiển **ESP32** là nền tảng chính để thu thập dữ liệu từ cảm biến mưa và điều khiển động cơ. ESP32 được sử dụng nhờ các ưu điểm nổi bật như:

  - **Wi-Fi và Bluetooth**: Hỗ trợ kết nối không dây, cho phép giao tiếp với server thông qua **HTTP POST/GET**.
  - **GPIO linh hoạt**: Cung cấp các chân giao tiếp cho cảm biến mưa và động cơ.
  - **Khả năng lập trình dễ dàng**: Sử dụng Platform IDE để phát triển và nạp chương trình cho ESP32.
  - **Tiêu thụ điện năng thấp**: Giúp tiết kiệm năng lượng khi triển khai thực tế.

- **Nền tảng phát triển**:

  - **Platform IDE**: Được sử dụng để viết code, biên dịch và nạp chương trình cho vi điều khiển ESP32.
  - **Thư viện WiFi.h**: Để kết nối ESP32 với mạng Wi-Fi.
  - **HTTPClient.h**: Gửi và nhận dữ liệu từ server thông qua giao thức HTTP.

Giao thức truyền thông

- **HTTP (Hypertext Transfer Protocol)**:
  - Được sử dụng để giao tiếp giữa ESP32 và Server Node.js.
  - ESP32 → Server:
    - Gửi log hoạt động và trạng thái mưa bằng phương thức HTTP POST.
  - Server → ESP32:
    - Cung cấp lệnh điều khiển từ giao diện Web bằng phương thức HTTP GET.
- **RESTful API**:
  - Server Node.js xây dựng các API RESTful để trao đổi dữ liệu giữa ESP32 và giao diện Web.
  - Các API chính bao gồm:
    - /log : Ghi nhận log từ ESP32.
    - /status : Gửi lệnh điều khiển về ESP32.
    - /rainStatus : Gửi và nhận trạng thái mưa.
    - /control: Nhận lệnh điều khiển từ giao diện Web.
    - /logs: Trả về log hoạt động cho giao diện Web.

Công nghệ phần mềm

- **Server-side**:
  - **Node.js**:
    - Được sử dụng để xây dựng server xử lý dữ liệu từ ESP32 và cung cấp dữ liệu cho giao diện Web.
    - Node.js hoạt động nhẹ và hiệu quả với các tác vụ I/O, giúp đáp ứng nhanh khi có nhiều yêu cầu từ ESP32 và người dùng.
  - **Express.js**:
    - Framework cho Node.js giúp tạo các API RESTful một cách nhanh chóng và hiệu quả.
  - **MongoDB**:
    - Hệ quản trị cơ sở dữ liệu NoSQL lưu trữ các dữ liệu log, lệnh điều khiển, và trạng thái mưa.
    - MongoDB được chọn nhờ khả năng lưu trữ linh hoạt và truy vấn dữ liệu nhanh.
  - **Moment-timezone**:
    - Thư viện xử lý thời gian và chuyển đổi sang múi giờ Việt Nam (GMT+7).
  - **XLSX.js**:
    - Thư viện dùng để xuất dữ liệu từ MongoDB sang file Excel cho mục đích báo cáo và phân tích.
- **Client-side (Giao diện Web)**:
  - **HTML/CSS**:
    - Được sử dụng để tạo giao diện trực quan, thân thiện với người dùng.

- **JavaScript**:
  - Thực hiện các chức năng như gửi lệnh điều khiển, hiển thị trạng thái mưa, và tự động cập nhật log từ server.
  - **Fetch API**: Dùng để gọi các API RESTful từ server mỗi 2 giây.

CHƯƠNG 2: THIẾT KẾ VÀ CÀI ĐẶT

2.1. Thiết kế phần cứng

Nút cảm biến:

-Vi điều khiển sử dụng:

ESP32 là vi điều khiển chính được sử dụng trong hệ thống. Lý do lựa chọn:

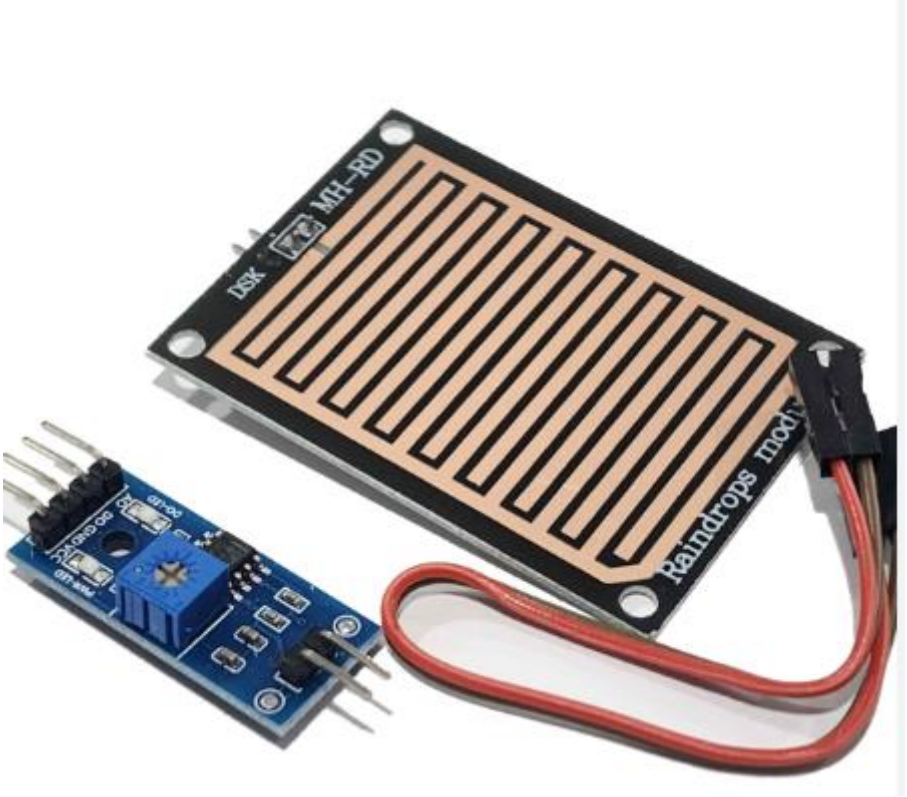
- Tích hợp Wi-Fi và Bluetooth giúp giao tiếp không dây với server.
- Có khả năng xử lý mạnh mẽ với CPU dual-core và RAM lớn.
- Có nhiều chân GPIO để kết nối cảm biến và điều khiển động cơ.
- Khả năng tiêu thụ điện năng thấp phù hợp cho các ứng dụng IoT.

Hình 2. ESP32 NodeMCU LUANODE32-38 chân

-Các loại cảm biến

Cảm biến mưa (Rain Sensor):

- Dùng để phát hiện trạng thái mưa (Raining/Not Raining).
- Ngõ ra dạng digital (HIGH/LOW) kết nối với GPIO của ESP32.



Hình 3. Cảm biến mưa

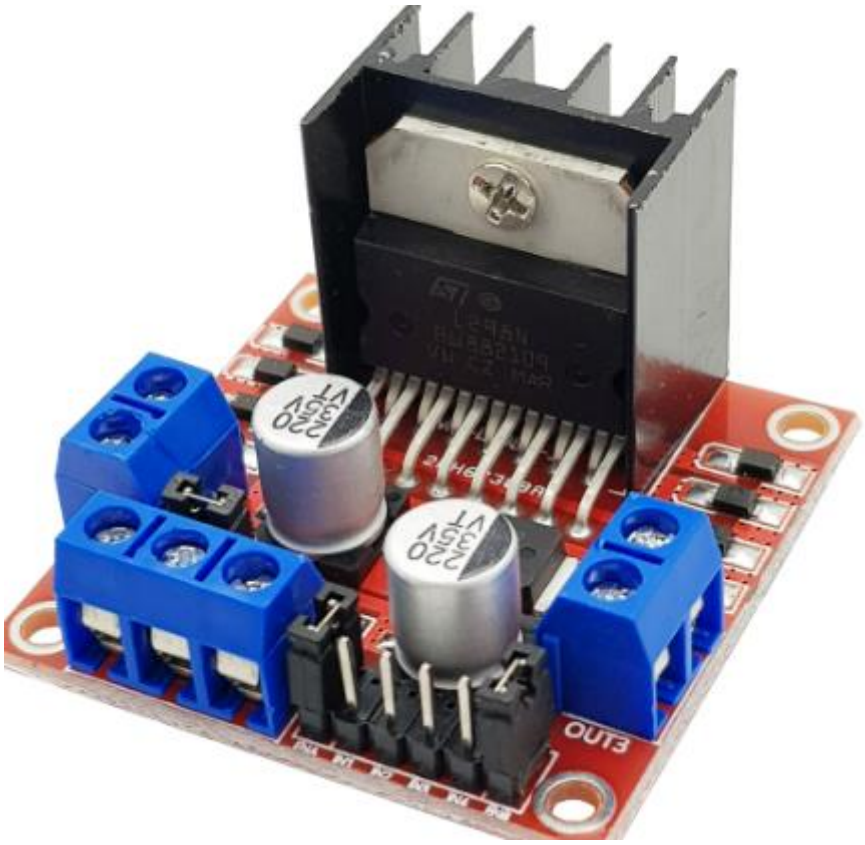
Công tắc giới hạn (Limit Switch):

- Hai công tắc giới hạn đặt ở vị trí biên trái và phải của thanh phơi đồ.
- Dùng để xác định khi động cơ đã đến giới hạn di chuyển.
- Kết nối với GPIO của ESP32.

Hình 4. Công tắc hành trình

L298N(Module điều khiển động cơ)

- Điều khiển tốc độ quay, chiều quay của động cơ



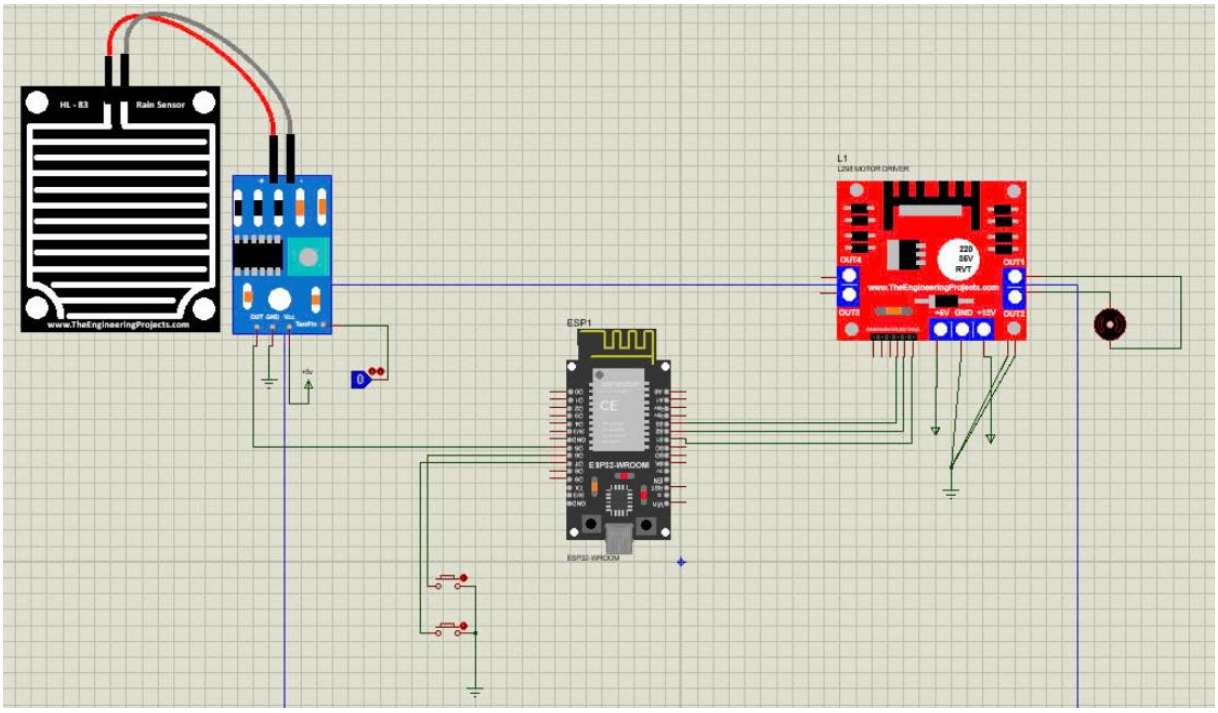
Hình 5. Module L298N

Động cơ 12V DC giảm tốc:

Giúp kéo quần áo vào , ra

Hình 6. Động cơ DC

Mạch điện và kết nối



Hình 7. Mạch Proteus mô phỏng mạch điện và kết nối của giàn phơi tự động

- Sơ đồ kết nối:

Cảm biến mưa được kết nối vào chân GPIO 34 của ESP32.
Công tắc giới hạn trái vào chân GPIO 32, công tắc giới hạn phải vào chân GPIO 33.
Mạch điều khiển động cơ:

Động cơ DC kết nối với module điều khiển L298N.
Chân điều khiển ENA, IN1, IN2 của module L298N được kết nối với các chân GPIO 19, 18, 5 trên ESP32.

- Nguồn điện và năng lượng

- **Nguồn cung cấp cho ESP32**: Sử dụng nguồn 5V từ pin sạc dự phòng.
- **Nguồn cấp cho động cơ**: Sử dụng nguồn 12V từ pin dự phòng cấp cho module L298N.

- Mạng truyền thông:

Topo mạng:

Mô hình Star: Mạng được thiết kế theo dạng **ngôi sao (Star)**, trong đó **Server** là trung tâm điều phối và giao tiếp với các thành phần khác.

Phương tiện truyền dẫn:

- Sử dụng mạng Wi-Fi nội bộ để kết nối ESP32 với Server Node.js.
- Đảm bảo băng thông và độ trễ thấp để đáp ứng thời gian thực trong giao tiếp.

Thiết bị trung gian:

- **Router Wi-Fi**:
  - Kết nối **ESP32** với Server Node.js thông qua mạng Wi-Fi nội bộ.
  - Phân phối địa chỉ IP và đảm bảo kết nối giữa các thiết bị.
  - Cho phép giao tiếp giữa ESP32 và Server với tốc độ và độ tin cậy cao.
- **Server Node.js**:
  - Đóng vai trò trung gian xử lý dữ liệu:
    - Nhận dữ liệu từ ESP32 (log, trạng thái mưa).
    - Gửi lệnh điều khiển đến ESP32.
    - Cung cấp dữ liệu log và trạng thái mưa cho giao diện web.
  - Server hoạt động như trung tâm điều phối và giao tiếp với **MongoDB** để lưu trữ dữ liệu.
- **MongoDB Database**:
  - Là thiết bị trung gian lưu trữ dữ liệu:
    - Log hoạt động của hệ thống.
    - Trạng thái mưa từ cảm biến.
    - Các lệnh điều khiển từ giao diện web.
  - MongoDB đảm bảo lưu trữ và truy xuất dữ liệu một cách nhanh chóng và hiệu quả.
- **Giao diện Web** (Trình duyệt của người dùng):
  - Giao diện web là thiết bị trung gian giúp người dùng tương tác với hệ thống:
    - Gửi lệnh điều khiển đến Server.
    - Nhận và hiển thị dữ liệu log hoạt động và trạng thái cảm biến mưa

2.2. Thiết kế phần mềm

Phần mềm nút cảm biến:

-Cấu trúc chương trình

- **Khởi tạo hệ thống**: Kết nối với Wi-Fi và khởi tạo các chân I/O của ESP32.
- **Giao tiếp với server**: Gửi và nhận dữ liệu từ server Node.js thông qua giao thức HTTP.
- **Đọc và xử lý dữ liệu từ cảm biến**: Kiểm tra trạng thái mưa từ cảm biến và gửi trạng thái này lên server.
- **Điều khiển động cơ**: Thực hiện các lệnh điều khiển từ server như moveOut, moveIn, và stop.
- **Ghi log hoạt động**: Ghi lại các trạng thái hoạt động và gửi log lên server.

- Các module chính

- **Kết nối Wi-Fi**:
  - Chức năng: Kết nối ESP32 với mạng Wi-Fi.
  - Thư viện sử dụng: `WiFi.h`.
- **Giao tiếp HTTP**:
  - Chức năng: Gửi dữ liệu log và trạng thái mưa lên server, nhận lệnh điều khiển từ server.
  - Thư viện sử dụng: `HTTPClient.h`.
- **Xử lý cảm biến mưa**:
  - Chức năng: Đọc giá trị từ chân GPIO kết nối với cảm biến mưa.
  - Dữ liệu được gửi đến server để lưu trữ và hiển thị.
- **Điều khiển động cơ**:
  - Chức năng: Điều khiển động cơ dựa trên lệnh nhận được từ server.
  - Các lệnh bao gồm: **moveOut** (di chuyển ra), **moveIn** (di chuyển vào), và **stop** (dừng).
  - Thư viện GPIO của ESP32 được sử dụng để điều khiển các chân IN1, IN2 và ENA.
- **Ghi log hoạt động**:
  - Chức năng: Ghi lại log trạng thái của động cơ và các lệnh điều khiển được nhận.
  - Log được gửi đến server để lưu trữ vào **MongoDB**.

-Thư viện sử dụng

Phần mềm sử dụng các thư viện chính sau:

1. **WiFi.h**: Thư viện kết nối Wi-Fi cho ESP32.
2. **HTTPClient.h**: Thư viện hỗ trợ giao tiếp HTTP giữa ESP32 và server.
3. **Arduino.h**: Thư viện cơ bản cho lập trình vi điều khiển trên Arduino.

-Xử lý dữ liệu cảm biến:

Đọc trạng thái cảm biến mưa:

- Sử dụng chân **GPIO 34** để kết nối với cảm biến mưa.
- Giá trị đọc từ cảm biến:
  - **LOW**: Trời đang mưa.
  - **HIGH**: Không mưa.

```
String rainStatus = digitalRead(RAIN_SENSOR_PIN) == LOW ? "Raining" : "Not Raining"

sendRainStatus(rainStatus); // Gửi trạng thái mưa đến server
```

Gửi dữ liệu cảm biến lên server:

Trạng thái mưa được đóng gói vào JSON và gửi lên server thông qua HTTP POST.

```
HTTPClient http;
http.begin(serverRainUrl);
http.addHeader("Content-Type", "application/json");

String jsonPayload = "{\"rain\":\"" + rainStatus + "\"}";
int httpCode = http.POST(jsonPayload);
http.end();
```

Ghi log cảm biến:

Log trạng thái mưa và hoạt động của động cơ được gửi liên tục về server.

Phần mềm gateway/server:

-Kiến trúc phần mềm:

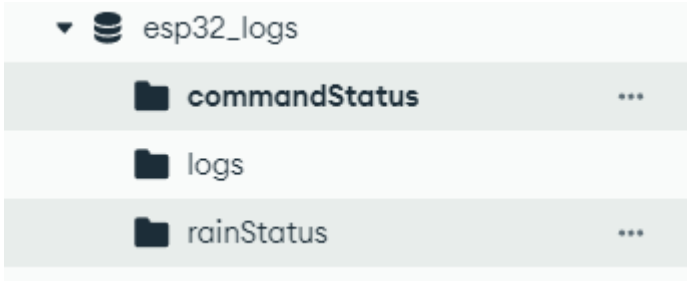
+ESP32 gửi log và trạng thái mưa đến server → Server lưu trữ dữ liệu vào MongoDB.
+ESP32 gửi yêu cầu lấy lệnh điều khiển từ server.
+Giao diện web gửi lệnh điều khiển đến server → Server phản hồi và lưu lệnh vào MongoDB.

+Dữ liệu được cung cấp qua API RESTful để cập nhật giao diện web.

-Cơ sở dữ liệu

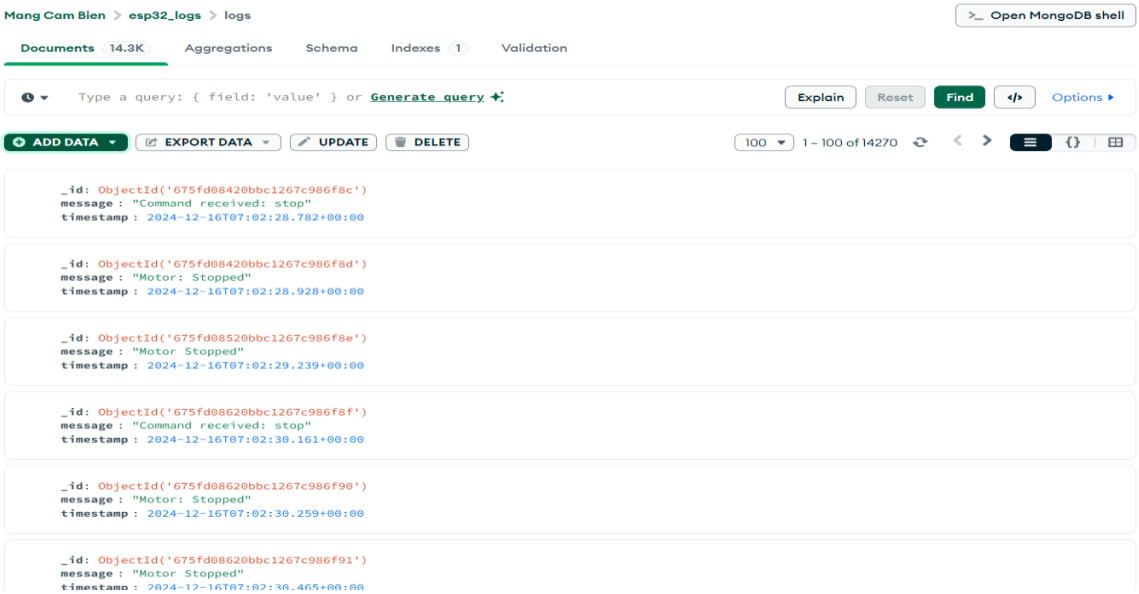
MongoDB được sử dụng để lưu trữ dữ liệu dưới dạng **collections**:

Gồm có 3 collections là :Command Status, logs , RainStatus:

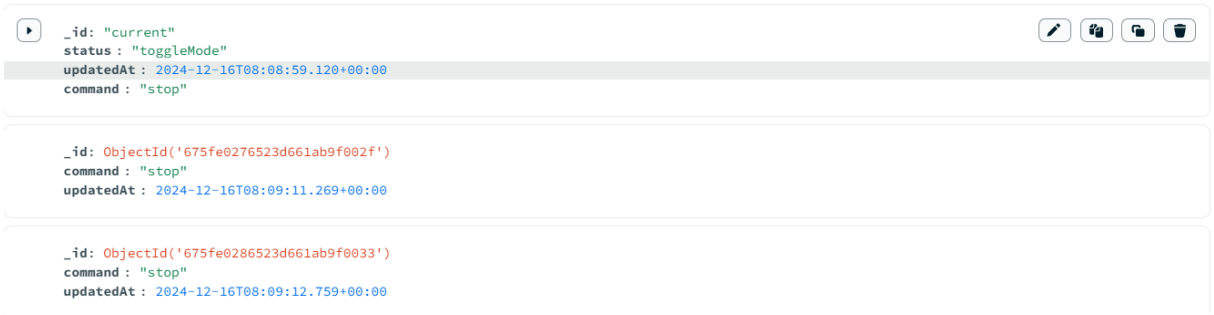


`

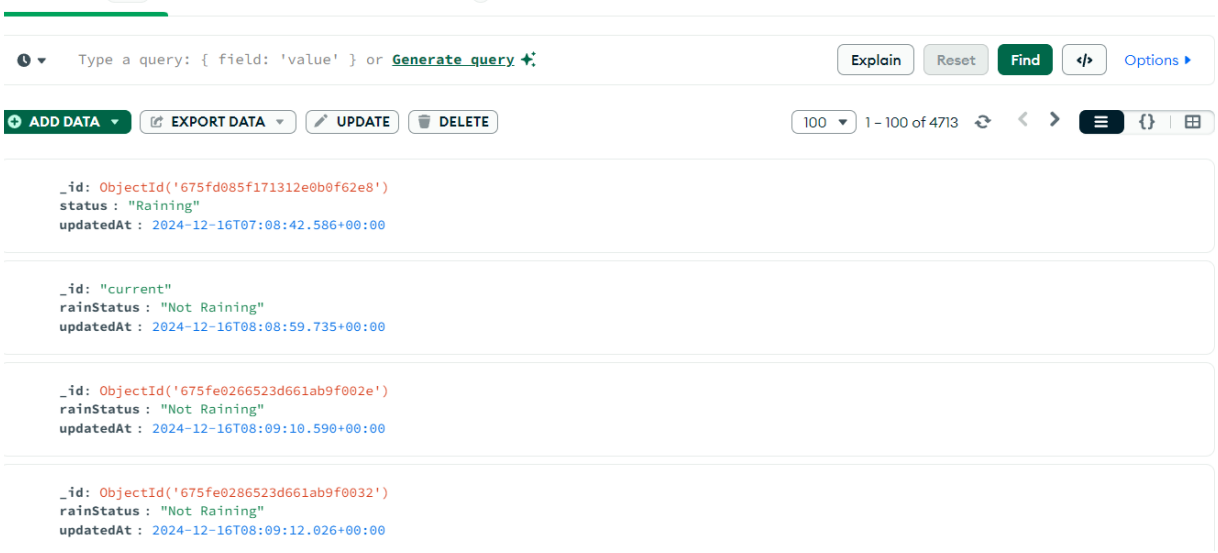
Logs: Lưu trữ thông tin logs hoạt động của hệ thống:



CommandStatus: Lưu trữ các nút đã được nhấn



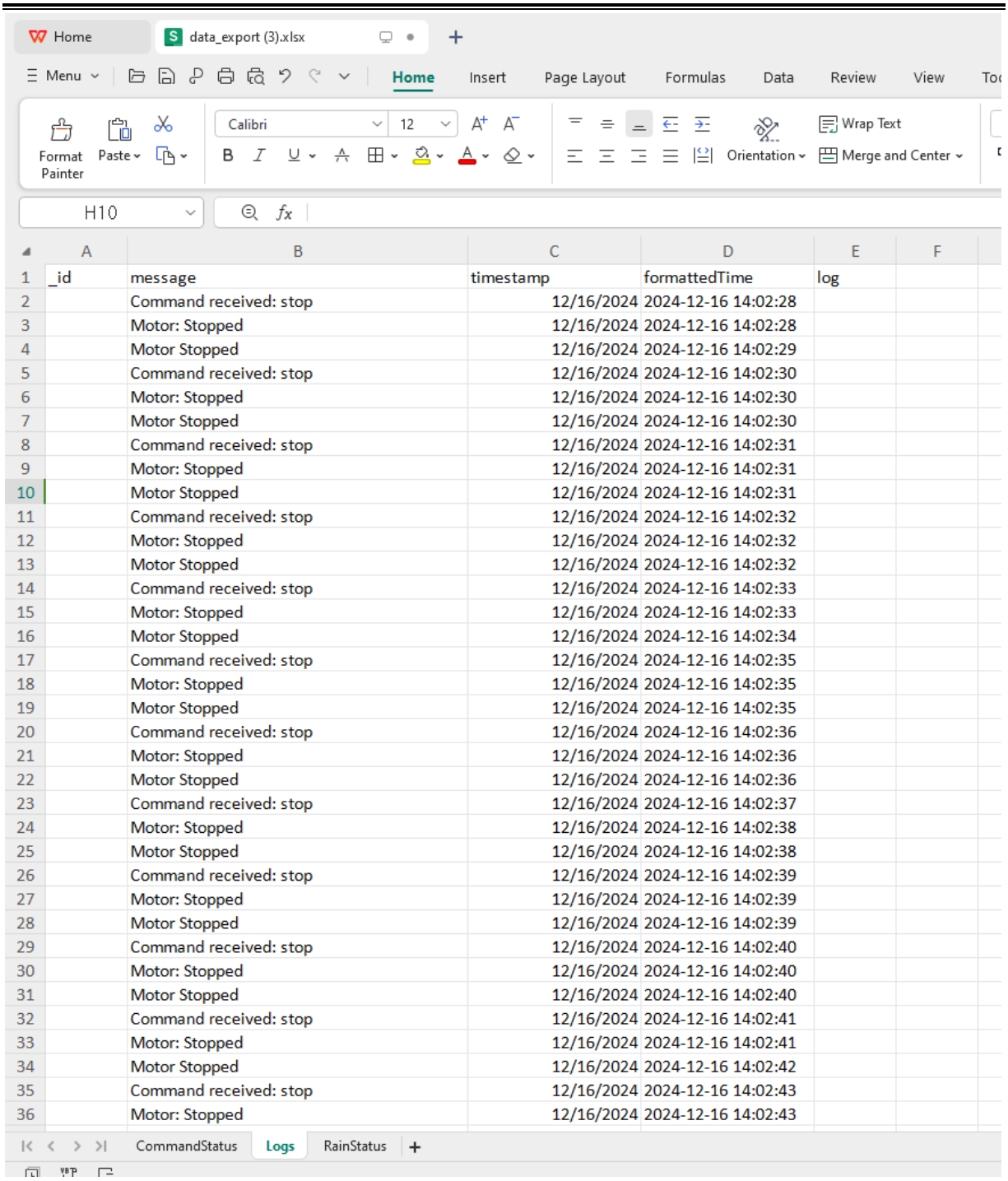
RainStatus: Lưu trữ trạng thái của cảm biến mưa:



Xuất file excel để quan sát cơ sở dữ liệu:

```
const wb = XLSX.utils.book_new();
XLSX.utils.book_append_sheet(wb, XLSX.utils.json_to_sheet(formattedCommandStatus), "CommandStatus");
XLSX.utils.book_append_sheet(wb, XLSX.utils.json_to_sheet(formattedLogs), "Logs");
XLSX.utils.book_append_sheet(wb, XLSX.utils.json_to_sheet(formattedRainStatus), "RainStatus");

const filePath = path.join(__dirname, "public", "data_export.xlsx");
XLSX.writeFile(wb, filePath);

res.download(filePath, "data_export.xlsx", (err) => {
  if (err) {
    console.error("❌ Error sending file:", err);
    res.status(500).send("Error exporting file");
```

Dùng http://localhost:3000/export để xuất ra file excel



Home | data_export (3).xlsx

Menu | Home | Insert | Page Layout | Formulas | Data | Review | View

Format Painter | Paste | Calibri | 12 | Wrap Text | Orientation | Merge and Center

H10

	A	B	C	D	E	F
1	_id	message	timestamp	formattedTime	log	
2		Command received: stop	12/16/2024	2024-12-16 14:02:28		
3		Motor: Stopped	12/16/2024	2024-12-16 14:02:28		
4		Motor Stopped	12/16/2024	2024-12-16 14:02:29		
5		Command received: stop	12/16/2024	2024-12-16 14:02:30		
6		Motor: Stopped	12/16/2024	2024-12-16 14:02:30		
7		Motor Stopped	12/16/2024	2024-12-16 14:02:30		
8		Command received: stop	12/16/2024	2024-12-16 14:02:31		
9		Motor: Stopped	12/16/2024	2024-12-16 14:02:31		
10		Motor Stopped	12/16/2024	2024-12-16 14:02:31		
11		Command received: stop	12/16/2024	2024-12-16 14:02:32		
12		Motor: Stopped	12/16/2024	2024-12-16 14:02:32		
13		Motor Stopped	12/16/2024	2024-12-16 14:02:32		
14		Command received: stop	12/16/2024	2024-12-16 14:02:33		
15		Motor: Stopped	12/16/2024	2024-12-16 14:02:33		
16		Motor Stopped	12/16/2024	2024-12-16 14:02:34		
17		Command received: stop	12/16/2024	2024-12-16 14:02:35		
18		Motor: Stopped	12/16/2024	2024-12-16 14:02:35		
19		Motor Stopped	12/16/2024	2024-12-16 14:02:35		
20		Command received: stop	12/16/2024	2024-12-16 14:02:36		
21		Motor: Stopped	12/16/2024	2024-12-16 14:02:36		
22		Motor Stopped	12/16/2024	2024-12-16 14:02:36		
23		Command received: stop	12/16/2024	2024-12-16 14:02:37		
24		Motor: Stopped	12/16/2024	2024-12-16 14:02:38		
25		Motor Stopped	12/16/2024	2024-12-16 14:02:38		
26		Command received: stop	12/16/2024	2024-12-16 14:02:39		
27		Motor: Stopped	12/16/2024	2024-12-16 14:02:39		
28		Motor Stopped	12/16/2024	2024-12-16 14:02:39		
29		Command received: stop	12/16/2024	2024-12-16 14:02:40		
30		Motor: Stopped	12/16/2024	2024-12-16 14:02:40		
31		Motor Stopped	12/16/2024	2024-12-16 14:02:40		
32		Command received: stop	12/16/2024	2024-12-16 14:02:41		
33		Motor: Stopped	12/16/2024	2024-12-16 14:02:41		
34		Motor Stopped	12/16/2024	2024-12-16 14:02:42		
35		Command received: stop	12/16/2024	2024-12-16 14:02:43		
36		Motor: Stopped	12/16/2024	2024-12-16 14:02:43		

CommandStatus | Logs | RainStatus

Hình 8. Bảng dữ liệu

-API và giao diện

API:

Server Node.js cung cấp các API để giao tiếp với ESP32 và giao diện web. Các API bao gồm:

POST /log

Nhận log dữ liệu từ ESP32 và lưu vào collection **logs**.

POST /rainStatus

Nhận trạng thái mưa từ ESP32 và lưu vào collection **rainStatus**.

GET /status

Trả về lệnh điều khiển mới nhất cho ESP32 từ collection **commandStatus**.

POST /control

Nhận lệnh điều khiển từ giao diện web và lưu vào collection **commandStatus**.

GET /logs

Trả về toàn bộ dữ liệu log từ collection **logs** cho giao diện web.

GET /rainStatus

Trả về trạng thái mưa mới nhất từ collection **rainStatus** cho giao diện web.

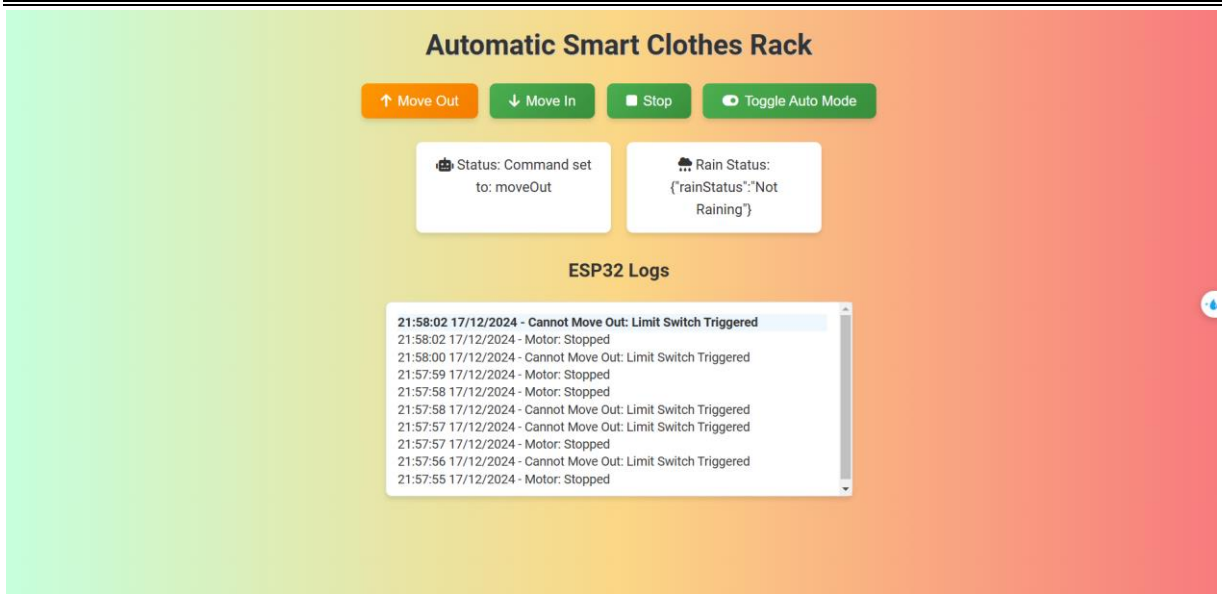
GET /export

Xuất toàn bộ dữ liệu **logs**, **commandStatus**, và **rainStatus** thành file Excel.

-Giao diện người dùng:

Giao diện web sử dụng HTML, CSS, và JavaScript để hiển thị thông tin và cho phép người dùng điều khiển hệ thống:

- Hiển thị log hoạt động: Dữ liệu log từ MongoDB được cập nhật mỗi 2 giây và hiển thị trên giao diện.
- Hiển thị trạng thái mưa: Cập nhật thời gian thực từ ESP32 thông qua server.
- Điều khiển hệ thống: Các nút bấm gửi lệnh điều khiển (moveOut, moveIn, stop, toggleMode) thông qua server đến ESP32



Hình 9. Giao diện web

Ví dụ mã API gọi tự giao diện web

```
async function sendCommand(command, clickedButton) {
    const statusElement = document.getElementById("status");
    try {
        const response = await fetch("http://192.168.107.106:3000/control", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ newCommand: command })
        });
```

-Xử lý logic nghiệp vụ

- **Nhận và xử lý lệnh điều khiển**:
  - Server nhận lệnh từ giao diện web thông qua API `/control`.
  - Lệnh được lưu trữ trong MongoDB và phản hồi cho ESP32 khi ESP32 gửi yêu cầu GET `/status`.
- **Ghi log và xử lý trạng thái mưa**:
  - Khi ESP32 gửi dữ liệu trạng thái mưa qua API `/rainStatus`, server lưu trữ vào MongoDB.
  - Log hoạt động cũng được cập nhật thông qua API `/log`.
- **Xuất dữ liệu thành file Excel**:
  - Dữ liệu từ các collections trong MongoDB được truy xuất và xuất thành file Excel.
- **Cập nhật giao diện người dùng**:

- Server cung cấp dữ liệu thông qua API `/logs` và `/rainStatus`.
- JavaScript trên giao diện web gọi API này định kỳ mỗi 2 giây để cập nhật thông tin.

2.3. Giao thức truyền thông

-Định dạng gói tin:

Giao thức truyền thông giữa ESP32 và server Node.js sử dụng giao thức HTTP (HyperText Transfer Protocol) với định dạng gói tin là JSON

-Cơ chế truyền nhận:

- **POST**: Được sử dụng để gửi dữ liệu từ ESP32 lên server
- **GET**: Được sử dụng để đọc dữ liệu từ server xuống ESP32 hoặc giao diện web

-Xử lý lỗi và tối ưu:

Xử lý lỗi truyền nhận:

- **ESP32**:
  - Nếu kết nối WiFi bị mất, ESP32 sẽ thử kết nối lại liên tục cho đến khi thành công.
  - Khi gửi gói tin HTTP, nếu phản hồi thất bại, ESP32 sẽ ghi log lỗi và thử gửi lại trong chu kỳ tiếp theo.

```
if (WiFi.status() != WL_CONNECTED) {
  Serial.println("WiFi mất kết nối! Đang kết nối lại...");
  WiFi.begin(ssid, password);
}
```

- **Server Node.js**:
  - Xử lý các trường hợp dữ liệu không hợp lệ và phản hồi HTTP code 400 (Bad Request).
  - Xử lý các lỗi kết nối MongoDB và phản hồi HTTP code 500 (Internal Server Error).

```
    }
  } else {
    console.error("❌ Invalid command received:", newCommand);
    res.status(400).send("Invalid command");
  }
```

CHƯƠNG 3: TRIỂN KHAI VÀ ĐÁNH GIÁ

3.1. Triển khai hệ thống

Cài đặt phần cứng:

-Lắp đặt nút cảm biến:

Cảm biến mưa:

- Cảm biến mưa được lắp đặt tại vị trí thích hợp trên hệ thống để có thể phát hiện chính xác trạng thái mưa. Kết nối với chân **GPIO 34** của ESP32 để đọc tín hiệu.
- Chân tín hiệu được kéo lên (pull-up) để đảm bảo tính ổn định khi đọc dữ liệu.

Giới hạn hành trình (Limit Switch):

- Các công tắc hành trình được gắn tại hai vị trí giới hạn của động cơ để bảo đảm motor ngừng hoạt động khi chạm đến giới hạn.
- Limit Switch trái kết nối với chân GPIO 32 và Limit Switch phải kết nối với chân GPIO 33.

Động cơ DC và module điều khiển:

- Động cơ DC điều khiển qua các chân ENA (GPIO 19), IN1 (GPIO 18) và IN2 (GPIO 5).
- Module điều khiển motor (L298N) giúp chuyển đổi tín hiệu logic từ ESP32 thành xung điều khiển động cơ.

-Cấu hình mạng

- Cấu hình **SSID** và **mật khẩu Wi-Fi** trên ESP32 để kết nối vào mạng cục bộ.
- Địa chỉ server **Node.js:**

```
// Thông tin WiFi
const char* ssid = "banhbi";
const char* password = "123456789a";

// Địa chỉ server Express.js
const char* serverCommandUrl = "http://192.168.107.106:3000/status";
const char* serverLogUrl = "http://192.168.107.106:3000/log";
const char* serverRainUrl = "http://192.168.107.106:3000/rainStatus";
```

-Kiểm tra kết nối

Kiểm tra từng thành phần của hệ thống:

Cảm biến mưa: Quan sát trạng thái đầu ra (Raining/Not Raining) thông qua Serial Monitor.
Động cơ: Thử lệnh **Move In**, **Move Out**, và **Stop** để xác nhận hoạt động đúng:
Kết nối mạng: Kiểm tra log gửi thành công lên server thông qua HTTP POST:

Cài đặt phần mềm:

-Deploy server

Chuẩn bị Môi Trường

Để triển khai server Node.js và đảm bảo kết nối với cơ sở dữ liệu MongoDB, ta thực hiện các bước sau:

Cài đặt Node.js và npm

Tải về Node.js từ trang chính thức: https://nodejs.org/.

Kiểm tra đã được cài chưa:

node -v

npm -v

Cài đặt MongoDB

Tải MongoDB từ trang chính thức:
https://www.mongodb.com/try/download/community.

Đảm bảo MongoDB đang chạy trên localhost:27017.

Cài đặt Thư Viện

Dự án sử dụng các thư viện sau cho các chức năng cụ thể:

express: Tạo server RESTful API.

cors: Cho phép giao tiếp từ các domain khác nhau.

body-parser: Xử lý dữ liệu JSON được gửi từ ESP32 và giao diện web.

mongodb: Thư viện kết nối với MongoDB.

moment-timezone: Xử lý và hiển thị thời gian chính xác theo múi giờ.

xlsx: Xuất dữ liệu từ MongoDB ra file Excel.

fs: Thư viện Node.js xử lý file hệ thống.

sử dụng lệnh sau đề cài đặt các thư viện:

npm install express cors body-parser mongodb moment-timezone xlsx

Khởi chạy server NodeJS:

Khởi chạy bằng lệnh sau trên terminal: **node server.js**

Nếu chạy server thành công sẽ hiển thị kết quả:

```
PS E:\NhatTan\Minh\a\esp32-motor-control> node server.js
✅ Server đang chạy tại http://localhost:3000
✅ Connected to MongoDB
```

Test các API bằng POSTMAN:

API	Phương thức	URL	Mô tả
Nhận log từ ESP32	POST	`/log`	Ghi log hoạt động vào MongoDB.
Nhận trạng thái mưa	POST	`/rainStatus`	Gửi trạng thái mưa từ ESP32.
Nhận lệnh từ giao diện	POST	`/control`	Nhận lệnh từ giao diện web.
Gửi lệnh cho ESP32	GET	`/status`	ESP32 lấy lệnh điều khiển.
Lấy log cho web	GET	`/logs`	Lấy toàn bộ log từ MongoDB.
Lấy trạng thái mưa	GET	`/rainStatus`	Gửi trạng thái mưa cho web.
Xuất dữ liệu ra Excel	GET	`/export`	Xuất log và trạng thái ra Excel.

Cài đặt firmware

Cài Đặt PlatformIO và ESP32 Framework

Cài đặt Plugin PlatformIO:

- Mở **VSCode**.
- Vào **Extensions** (Ctrl+Shift+X).
- Tìm **PlatformIO IDE** và cài đặt.

Tạo Project Mới:

- Vào **PlatformIO Home** > **New Project**.
- Chọn **Board**: ESP32 Dev Module.
- Chọn **Framework**: Arduino.
- Chọn thư mục lưu trữ dự án.

Tích hợp hệ thống:

Đồng bộ dữ liệu

Giao diện web gửi yêu cầu HTTP GET tới server để lấy **log**, **trạng thái mưa** và **trạng thái điều khiển** định kỳ mỗi 2 giây.

```
setInterval(fetchLogs, 2000);
```

Khi người dùng nhấn nút trên giao diện web, lệnh sẽ được gửi lên server Node.js và phản hồi về ESP32 để điều khiển động cơ.

```
async function sendCommand(command, clickedButton) {
    const statusElement = document.getElementById("status");
    try {
        const response = await fetch("http://192.168.107.106:3000/control", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ newCommand: command })
        });
```

-Kiểm tra tổng thể

Kiểm tra kết nối phần cứng

Cảm biến mưa: Kiểm tra trạng thái "Raining" và "Not Raining" gửi lên server khi mưa.

Công tắc hành trình: Kiểm tra các giới hạn "Limit Switch Left" và "Limit Switch Right".

Động cơ: Đảm bảo động cơ hoạt động đúng lệnh "Move In", "Move Out" và "Stop".

Kiểm tra giao tiếp giữa ESP32 và Server

Sử dụng **Serial Monitor** trên PlatformIO để kiểm tra dữ liệu được gửi và nhận từ server.

3.3 Kiểm tra cơ sở dữ liệu MongoDB

Dữ liệu log, trạng thái mưa và lệnh điều khiển được lưu trữ trong MongoDB với định dạng chính xác.

Sử dụng **MongoDB Compass** để giám sát dữ liệu trong các collection.

3.4 Kiểm tra giao diện web

Xác minh giao diện web cập nhật dữ liệu tự động mỗi 2 giây.

Kiểm tra các nút điều khiển gửi lệnh chính xác tới server và ESP32 phản hồi đúng.

3.5 Kiểm tra xuất dữ liệu Excel

Gửi yêu cầu **GET** `/export` và kiểm tra file Excel được tạo với đầy đủ các sheet **Logs**, **CommandStatus**, và **RainStatus**.

3.2. Thử nghiệm và đánh giá

-Kịch bản thử nghiệm:

Kiểm tra chức năng

Kiểm tra các tính năng cơ bản và nghiệp vụ của hệ thống để đảm bảo hoạt động đúng như thiết kế:

- **ESP32 gửi dữ liệu**: Kiểm tra khả năng ESP32 gửi log hoạt động, trạng thái mưa lên server thông qua HTTP.
    - **Kịch bản**: Gửi log "Motor Moving In" và trạng thái "Raining".
    - **Kết quả mong đợi**: Server nhận được dữ liệu và lưu trữ vào MongoDB.
    - **Thực hiện**: Quan sát dữ liệu nhận được từ **Serial Monitor** trên ESP32 và MongoDB Compass.
- **Server lưu trữ dữ liệu**:
    - **Kịch bản**: Server ghi dữ liệu vào các collection logs, commandStatus, rainStatus.
    - **Kết quả mong đợi**: Dữ liệu được lưu trữ đúng cấu trúc và thời gian.

- **Giao diện web hiển thị và điều khiển**:

Kịch bản:Người dùng gửi lệnh "Move Out", "Move In", "Stop" qua giao diện web.

Dữ liệu log cập nhật tự động mỗi 2 giây.

Kết quả mong đợi: Động cơ thực hiện đúng lệnh, log hoạt động và trạng thái mưa được hiển thị chính xác.

Kiểm tra hiệu năng

- **Mục tiêu**: Đánh giá tốc độ xử lý và thời gian phản hồi của hệ thống.
- **Kịch bản**:
  - Gửi nhiều lệnh điều khiển liên tục trong khoảng thời gian ngắn (ví dụ: 1 lệnh/giây trong 10 giây).
  - Tăng tần suất gửi dữ liệu log từ ESP32 lên server từ 2 giây thành 0.5 giây.
- **Chỉ số đo lường**:
  - Thời gian phản hồi giữa ESP32 và server.
  - Tốc độ cập nhật log trên giao diện we

Kiểm tra độ tin cậy

- **Mục tiêu**: Đảm bảo hệ thống hoạt động ổn định trong thời gian dài.
- **Kịch bản**:
  - Chạy hệ thống liên tục trong vòng 12 giờ và quan sát hoạt động của ESP32, server và MongoDB.
  - Cố tình ngắt kết nối WiFi của ESP32 và theo dõi khả năng khôi phục kết nối.
- **Kết quả mong đợi**: Hệ thống tự động khôi phục hoạt động bình thường khi kết nối lại.

-Kết quả đánh giá:

Độ chính xác dữ liệu

- **Mục tiêu**: Đảm bảo dữ liệu gửi từ ESP32 lên server và lưu trữ vào MongoDB chính xác.

- **Thực hiện**:

- So sánh dữ liệu thực tế đọc được từ cảm biến với dữ liệu được lưu trữ trong MongoDB.
- Xác minh log và trạng thái mưa hiển thị trên giao diện web.

- **Kết quả**: Độ chính xác dữ liệu đạt 100% trong các thử nghiệm chức năng.

Thời gian đáp ứng

- **Mục tiêu**: Đo thời gian từ lúc ESP32 gửi lệnh đến lúc server phản hồi và hiển thị kết quả trên giao diện web.

Kết quả đo lường:

- Thời gian phản hồi trung bình: **200 - 300 ms**.
- Thời gian cập nhật dữ liệu tự động: **2 giây** (thiết lập theo kịch bản).

Độ ổn định hệ thống

- **Mục tiêu**: Đánh giá khả năng hoạt động liên tục và khôi phục khi gặp sự cố.
- **Kết quả**:

- Hệ thống chạy ổn định trong vòng **12 giờ liên tục**.
- ESP32 tự động kết nối lại WiFi và gửi dữ liệu sau khi mất kết nối

Khả năng mở rộng:

:Kiểm tra khả năng thêm cảm biến mới hoặc mở rộng tính năng điều khiển.

Có thể mrộng thêm cảm biến mưa, nhiệt độ hoặc độ ẩm dễ dàng bằng cách cập nhật phần mềm và API server.

-Phân tích và tối ưu:

Các vấn đề gặp phải:

- **Độ trễ khi gửi nhiều dữ liệu cùng lúc**:

Khi tần suất gửi dữ liệu log từ ESP32 lên server giảm còn 0.5 giây,

đôi khi xảy ra độ trễ.

- **Kết nối WiFi không ổn định**:

ESP32 mất kết nối khi tín hiệu WiFi yếu hoặc gián đoạn.

- **Tải lớn trên server khi dữ liệu tăng**:

Dữ liệu log lưu vào MongoDB tăng nhanh khi hệ thống chạy lâu dài

Đề xuất cải tiến:

Tích hợp thêm cảm biến nhiệt độ, dộ ẩm để mở rộng chức năng theo dõi

Tích hợp thêm lưu trữ dữ liệu trên cloud

3.3. Ứng dụng và hướng phát triển

Khả năng ứng dụng thực tế

Hệ thống giá phơi đồ thông minh được thiết kế và triển khai trong đề tài có khả năng ứng dụng cao trong đời sống , giúp điều khiển dễ dàng , và nhận biết thời tiết để áo quần không bị dính mưa

Hướng phát triển tiếp theo

- C**ải tiến phần cứng**:

  Tích hợp thêm cảm biến nhiệt độ, độ ẩm, cảm biến gió để thu thập dữ liệu thời tiết chi tiết hơn.

  Sử dụng động cơ bước hoặc động cơ servo để điều khiển chính xác hơn khi di chuyển giá phơi đồ.

  Sử dụng tấm năng lượng mặt trời để cấp nguồn cho hệ thống, giúp tiết kiệm năng lượng.

- **Nâng cấp giao thức truyền thông**:

  Thay thế giao thức HTTP bằng MQTT hoặc WebSocket để tăng tốc độ phản hồi và giảm tải cho server.

  Ứng dụng công nghệ IoT cloud như AWS IoT, Azure IoT Hub để quản lý hệ thống từ xa.

- **Mở rộng phần mềm điều khiển**:

Tích hợp AI hoặc học máy (Machine Learning) để phân tích dữ liệu thời tiết và dự đoán thời gian mưa.

Lưu trữ dữ liệu trên các dịch vụ đám mây như Firebase, Google Cloud hoặc AWS.

- **Cải tiến bảo mật**:

  Thêm các biện pháp mã hóa dữ liệu khi truyền tải giữa ESP32 và server để đảm bảo an toàn thông tin.

  Tích hợp xác thực người dùng trên giao diện web để ngăn chặn truy cập trái phép.

- **Tăng tính tự động hóa**:

  Phát triển **hệ thống cảnh báo thông minh**: gửi thông báo đến điện thoại hoặc email khi trời mưa hoặc khi phát hiện lỗi trong hệ thống.

  Cho phép thiết lập **lịch trình tự động** để phơi và thu đồ vào những thời điểm cụ thể trong ngày.

Đề xuất mở rộng:

Mở rộng hệ thống cho nhiều khu vực:

- Tích hợp điều khiển cho nhiều giá phơi thông minh trong cùng một hệ thống.
- Cho phép người dùng giám sát và điều khiển từ xa qua một bảng điều khiển duy nhất.

Kết nối với hệ thống thời tiết online:

- Lấy dữ liệu thời tiết theo thời gian thực từ các API như OpenWeatherMap để tự động điều khiển giá phơi đồ.
- Dự đoán thời tiết trong ngày và lên lịch hoạt động phù hợp.

Tích hợp với hệ thống nhà thông minh:

- Kết nối với các nền tảng nhà thông minh như để điều khiển hệ thống bằng giọng nói hoặc kịch bản tự động hóa.

Ứng dụng trong nông nghiệp:

- Sử dụng công nghệ này để tạo các nhà lưới tự động che mưa trong sản xuất nông nghiệp nhỏ lẻ.
- Tích hợp cảm biến để điều chỉnh môi trường phù hợp với cây trồng.

Triển khai trên quy mô lớn:

- Phát triển hệ thống cho các khu công nghiệp hoặc trung tâm giặt là quy mô lớn, tích hợp quản lý nhiều giá phơi đồ thông minh thông qua một server duy nhất.

PHẦN 2: CÂU HỎI VẤN ĐÁP

Câu 1 (1,5 điểm): [CLO 1] Tại sao em lại dùng vi điều khiển (ESP32) trong hệ thống?

Lý do chọn ESP32 trong hệ thống:

1. **Kết nối Wi-Fi và Bluetooth tích hợp:**
   - ESP32 là vi điều khiển mạnh mẽ có sẵn **Wi-Fi** và **Bluetooth**, giúp giao tiếp không dây dễ dàng với server hoặc các thiết bị khác.
   - Trong hệ thống của em, ESP32 đóng vai trò là **gateway cảm biến** gửi dữ liệu mưa, nhận điều khiển từ server thông qua giao thức HTTP.
   - Điều này giúp hệ thống hoạt động ổn định trong môi trường **Internet of Things (IoT)** mà không cần thêm module giao tiếp khác.
2. **Tính năng mạnh mẽ và hiệu suất cao:**
   - ESP32 có **2 nhân CPU Xtensa LX6 240 MHz** và bộ nhớ RAM đủ lớn, giúp nó xử lý **cảm biến**, **điều khiển động cơ** và **giao tiếp với server** một cách nhanh chóng và mượt mà.
   - Đáp ứng thời gian thực khi kiểm tra mưa và điều khiển động cơ đóng/mở giàn phơi.
3. **Hỗ trợ nhiều GPIO (chân Input/Output):**
   - ESP32 có nhiều chân GPIO để kết nối cảm biến và thiết bị điều khiển như:
     - **Cảm biến mưa** (chân **RAIN_SENSOR_PIN**)
     - **Công tắc hành trình** (giới hạn) (chân **LIMIT_SWITCH_LEFT** và **LIMIT_SWITCH_RIGHT**)
     - **Động cơ DC** điều khiển đóng/mở giàn phơi (chân **MOTOR_ENA_PIN**, **MOTOR_IN1_PIN**, **MOTOR_IN2_PIN**).
   - Điều này phù hợp với thiết kế phần cứng trong hệ thống của em.
4. **Khả năng lập trình dễ dàng với PlatformIO:**
   - ESP32 hỗ trợ **Arduino Framework** trên PlatformIO (VSCode), giúp lập trình dễ dàng, sử dụng thư viện có sẵn như:
     - **WiFi.h** để kết nối mạng.
     - **HTTPClient.h** để giao tiếp HTTP với server.
   - Cộng đồng phát triển lớn nên việc tìm kiếm tài liệu hỗ trợ và mở rộng hệ thống rất thuận tiện.
5. **Chi phí thấp và khả năng mở rộng:**
   - ESP32 là vi điều khiển **giá rẻ** nhưng vẫn có đủ tính năng cần thiết cho một hệ thống tự động.
   - Có thể mở rộng thêm cảm biến khác trong tương lai như: **nhiệt độ, độ ẩm, ánh sáng**, v.v.

Câu 1 (1,5 điểm): [CLO 1] Tại sao em lại dùng vi điều khiển (ESP32) trong hệ thống?

Lý do chọn ESP32 trong hệ thống:

1. **Kết nối Wi-Fi và Bluetooth tích hợp:**
   - ESP32 là vi điều khiển mạnh mẽ có sẵn Wi-Fi và Bluetooth, giúp giao tiếp không dây dễ dàng với server hoặc các thiết bị khác.
   - Trong hệ thống của em, ESP32 đóng vai trò là gateway cảm biến gửi dữ liệu mưa, nhận điều khiển từ server thông qua giao thức HTTP.
   - Điều này giúp hệ thống hoạt động ổn định trong môi trường Internet of Things (IoT) mà không cần thêm module giao tiếp khác.
2. **Tính năng mạnh mẽ và hiệu suất cao:**
   - ESP32 có 2 nhân CPU Xtensa LX6 240 MHz và bộ nhớ RAM đủ lớn, giúp nó xử lý cảm biến, điều khiển động cơ và giao tiếp với server một cách nhanh chóng và mượt mà.
   - Đáp ứng thời gian thực khi kiểm tra mưa và điều khiển động cơ đóng/mở giàn phơi.
3. **Hỗ trợ nhiều GPIO (chân Input/Output):**
   - ESP32 có nhiều chân GPIO để kết nối cảm biến và thiết bị điều khiển như:
     - Cảm biến mưa (chân RAIN_SENSOR_PIN)
     - Công tắc hành trình (giới hạn) (chân LIMIT_SWITCH_LEFT và LIMIT_SWITCH_RIGHT)
     - Động cơ DC điều khiển đóng/mở giàn phơi (chân MOTOR_ENA_PIN, MOTOR_IN1_PIN, MOTOR_IN2_PIN).
   - Điều này phù hợp với thiết kế phần cứng trong hệ thống
4. **Khả năng lập trình dễ dàng với PlatformIO:**
   - ESP32 hỗ trợ Arduino Framework trên PlatformIO (VSCode), giúp lập trình dễ dàng, sử dụng thư viện có sẵn như:
     - WiFi.h để kết nối mạng.
     - HTTPClient.h để giao tiếp HTTP với server.
   - Cộng đồng phát triển lớn nên việc tìm kiếm tài liệu hỗ trợ và mở rộng hệ thống rất thuận tiện.
5. **Chi phí thấp và khả năng mở rộng:**

- ESP32 là vi điều khiển giá rẻ nhưng vẫn có đủ tính năng cần thiết cho một hệ thống tự động.
- Có thể mở rộng thêm cảm biến khác trong tương lai như: nhiệt độ, độ ẩm, ánh sáng, v.v.

Câu 2 (1,5 điểm): [CLO 2] Thực hiện thao tác chọn thư viện và đọc dữ liệu cảm biến

1. Thư viện sử dụng:

```
#include <WiFi.h>        // Thư viện WiFi cho ESP32
#include <HTTPClient.h>  // Thư viện HTTP cho giao tiếp server
```

2. Đọc dữ liệu từ cảm biến mưa:

```
String rainStatus = digitalRead(RAIN_SENSOR_PIN) == LOW ? "Raining" : "Not Raining";
Serial.println("Rain Status Sent: " + rainStatus);
```

3. Gửi dữ liệu cảm biến lên server:

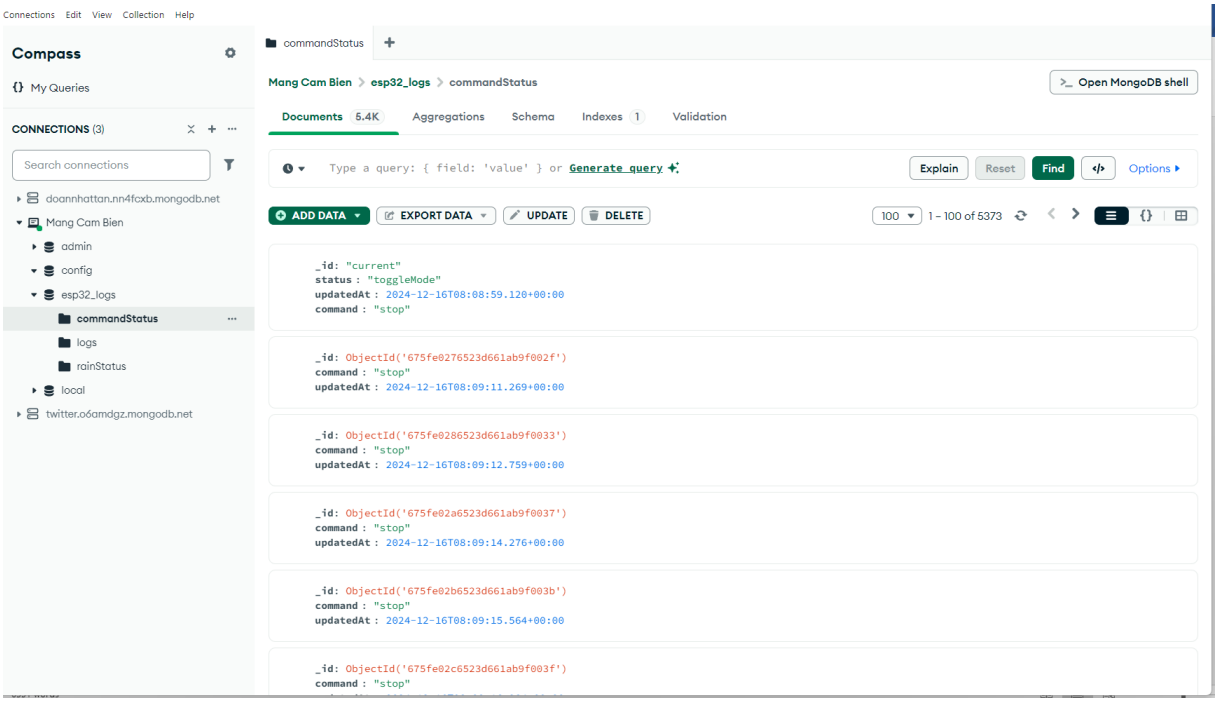
```
void sendRainStatus() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverRainUrl);
    http.addHeader("Content-Type", "application/json");

    String rainStatus = digitalRead(RAIN_SENSOR_PIN) == LOW ? "Raining" : "Not Raining";
    String jsonPayload = "{\"rain\":\"" + rainStatus + "\"}";
    int httpCode = http.POST(jsonPayload);
    if (httpCode > 0) {
      Serial.println("Rain Status Sent: " + rainStatus);
    }
    http.end();
  }
}
```

Câu 3 (2 điểm): [CLO 3] Thực hiện thao tác xây dựng đồ thị hiển thị lịch sử người dùng thông qua web

4. Server gửi dữ liệu logs,commandStatus, rainStatus đến MongoDB:

Server Node.js lưu dữ liệu collection của MongoDB với trường **:**

```
await db.collection("logs").insertOne({ message: log, timestamp:
vietnamTime });
await db.collection("rainStatus").insertOne
await db.collection("commandStatus").insertOne
```



5. Server truy xuất dữ liệu từ các collection của MongoDB

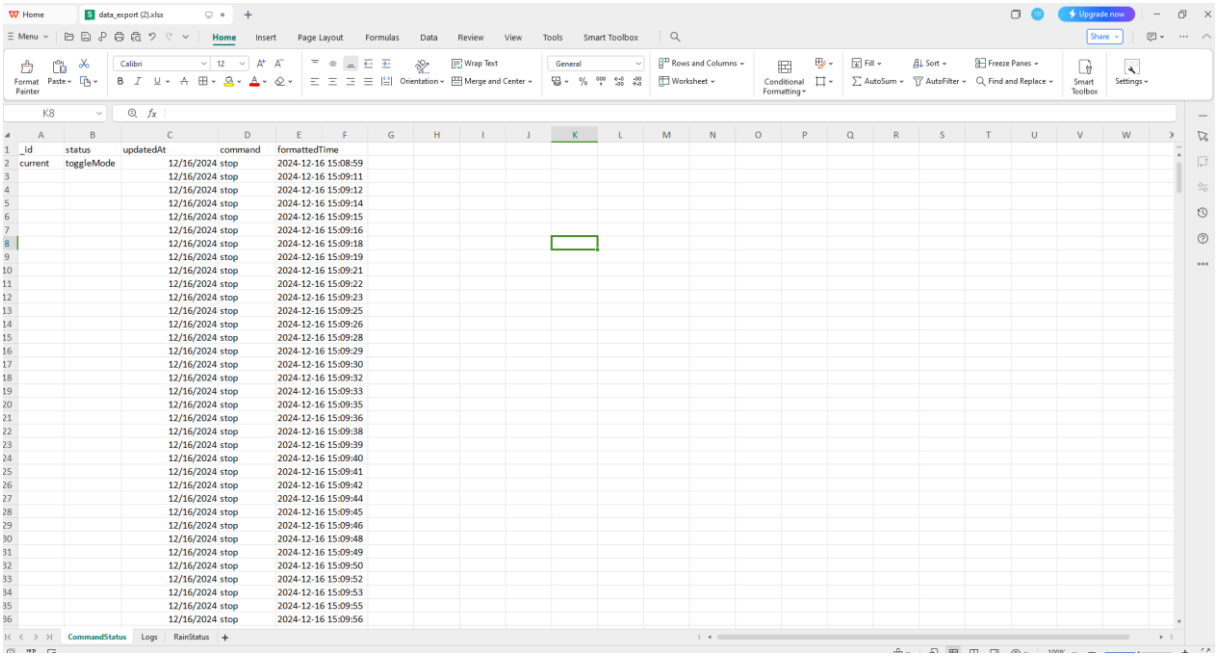
```
const commandStatus = await
db.collection("commandStatus").find({}).toArray();
  const logs = await db.collection("logs").find({}).toArray();
  const rainStatus = await db.collection("rainStatus").find({}).toArray();
```

6. Lấy các dữ liệu vừa truy xuất đó xuất ra file excel

```
const wb = XLSX.utils.book_new();
  XLSX.utils.book_append_sheet(wb,
XLSX.utils.json_to_sheet(formattedCommandStatus), "CommandStatus");
  XLSX.utils.book_append_sheet(wb,
XLSX.utils.json_to_sheet(formattedLogs), "Logs");
  XLSX.utils.book_append_sheet(wb,
XLSX.utils.json_to_sheet(formattedRainStatus), "RainStatus");

  const filePath = path.join(__dirname, "public", "data_export.xlsx");
  XLSX.writeFile(wb, filePath);

  res.download(filePath, "data_export.xlsx", (err) => {
   if (err) {
    console.error(" ❌ Error sending file:", err);
```

```
        res.status(500).send("Error exporting file");
      }
      fs.unlinkSync(filePath); // Xóa file tạm
    });
  } catch (error) {
    console.error("❌ Error exporting data:", error);
    res.status(500).send("Error exporting data");
  }
}
```

Dùng http://localhost:3000/export để tải file excel về



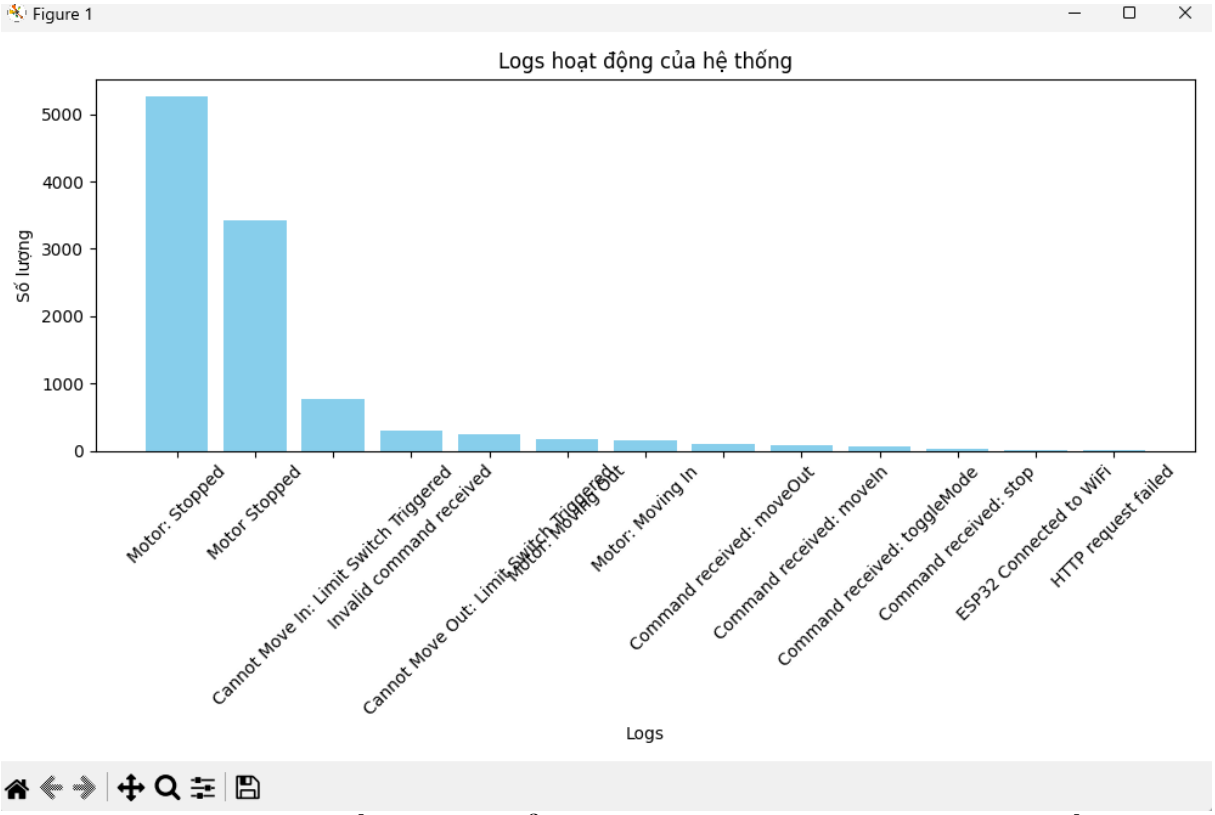
_id	status	updatedAt	command	formattedTime
current	toggleMode	12/16/2024	stop	2024-12-16 15:08:59
		12/16/2024	stop	2024-12-16 15:09:11
		12/16/2024	stop	2024-12-16 15:09:12
		12/16/2024	stop	2024-12-16 15:09:14
		12/16/2024	stop	2024-12-16 15:09:15
		12/16/2024	stop	2024-12-16 15:09:16
		12/16/2024	stop	2024-12-16 15:09:18
		12/16/2024	stop	2024-12-16 15:09:19
		12/16/2024	stop	2024-12-16 15:09:21
		12/16/2024	stop	2024-12-16 15:09:22
		12/16/2024	stop	2024-12-16 15:09:23
		12/16/2024	stop	2024-12-16 15:09:25
		12/16/2024	stop	2024-12-16 15:09:26
		12/16/2024	stop	2024-12-16 15:09:28
		12/16/2024	stop	2024-12-16 15:09:29
		12/16/2024	stop	2024-12-16 15:09:30
		12/16/2024	stop	2024-12-16 15:09:32
		12/16/2024	stop	2024-12-16 15:09:33
		12/16/2024	stop	2024-12-16 15:09:35
		12/16/2024	stop	2024-12-16 15:09:36
		12/16/2024	stop	2024-12-16 15:09:38
		12/16/2024	stop	2024-12-16 15:09:39
		12/16/2024	stop	2024-12-16 15:09:40
		12/16/2024	stop	2024-12-16 15:09:41
		12/16/2024	stop	2024-12-16 15:09:42
		12/16/2024	stop	2024-12-16 15:09:44
		12/16/2024	stop	2024-12-16 15:09:45
		12/16/2024	stop	2024-12-16 15:09:46
		12/16/2024	stop	2024-12-16 15:09:48
		12/16/2024	stop	2024-12-16 15:09:49
		12/16/2024	stop	2024-12-16 15:09:50
		12/16/2024	stop	2024-12-16 15:09:52
		12/16/2024	stop	2024-12-16 15:09:53
		12/16/2024	stop	2024-12-16 15:09:55
		12/16/2024	stop	2024-12-16 15:09:56

7. **Dùng python để vẽ đồ thị hiển thị lịch sử người dùng(bao gồm trạng thái mưa, các nút nhấn , hiển thị trạng thái log)**

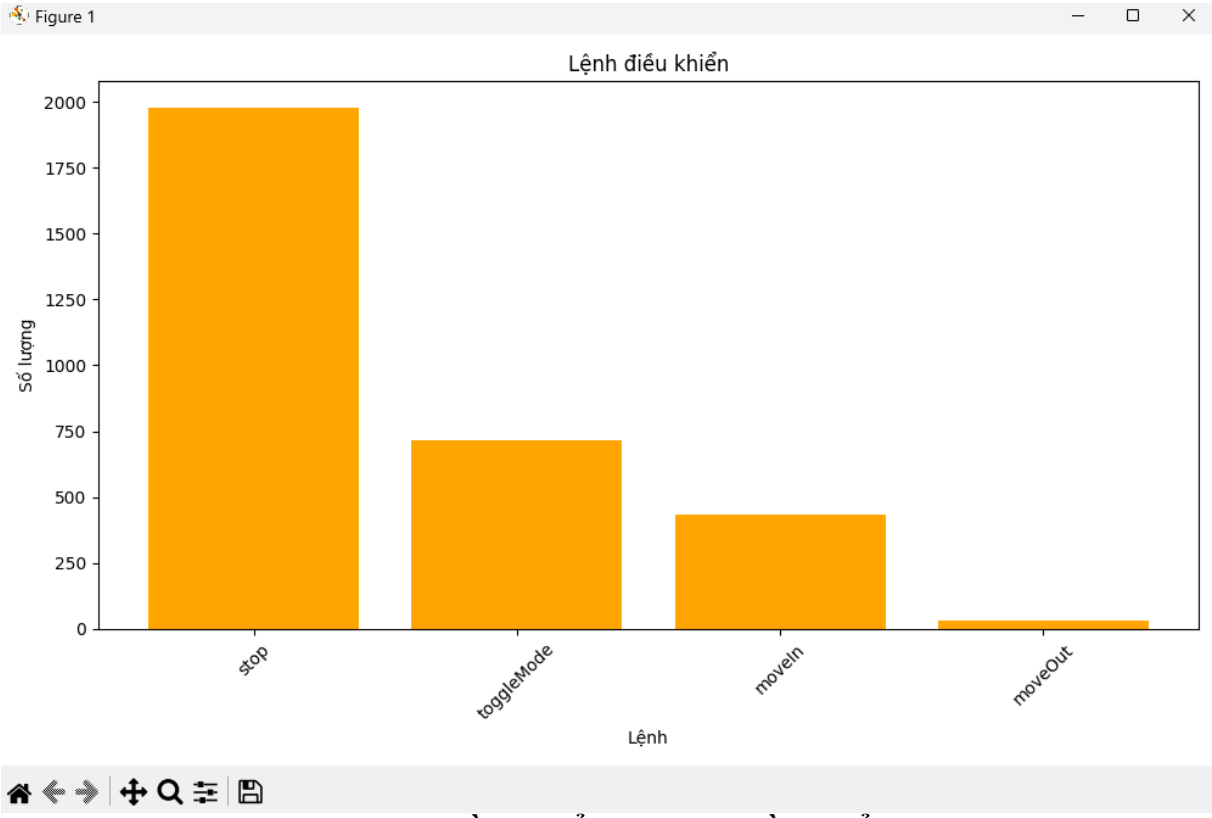
```
import pandas as pd
import matplotlib.pyplot as plt

# Đọc dữ liệu từ file Excel
file_path = "E:/NhatTan/Minh/a/data_export.xlsx"
excel_data = pd.ExcelFile(file_path)

# Đọc từng sheet
logs = pd.read_excel(excel_data, "Logs")
command_status = pd.read_excel(excel_data, "CommandStatus")
rain_status = pd.read_excel(excel_data, "RainStatus")

# Chuyển đổi thời gian về dạng datetime nếu cần
logs['timestamp'] = pd.to_datetime(logs['timestamp'], errors='coerce')
```

```
command_status['updatedAt'] = pd.to_datetime(command_status['updatedAt'],
errors='coerce')
rain_status['updatedAt'] = pd.to_datetime(rain_status['updatedAt'],
errors='coerce')

# Loại bỏ các hàng có giá trị NaT (dữ liệu thời gian không hợp lệ)
logs.dropna(subset=['timestamp'], inplace=True)
command_status.dropna(subset=['updatedAt'], inplace=True)
rain_status.dropna(subset=['updatedAt'], inplace=True)

# Đếm số lượng trạng thái (logs) theo từng loại để vẽ cột
log_counts = logs['message'].value_counts()
command_counts = command_status['command'].value_counts()
rain_counts = rain_status['rainStatus'].value_counts()

# Vẽ đồ thị dạng cột cho Logs
plt.figure(figsize=(10, 6))
plt.bar(log_counts.index, log_counts.values, color='skyblue')
plt.title("Logs hoạt động của hệ thống")
plt.xlabel("Logs")
plt.ylabel("Số lượng")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Vẽ đồ thị dạng cột cho CommandStatus
plt.figure(figsize=(10, 6))
plt.bar(command_counts.index, command_counts.values, color='orange')
plt.title("Lệnh điều khiển")
plt.xlabel("Lệnh")
plt.ylabel("Số lượng")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Vẽ đồ thị dạng cột cho RainStatus
plt.figure(figsize=(10, 6))
plt.bar(rain_counts.index, rain_counts.values, color='lightblue')
plt.title("Trạng thái mưa")
plt.xlabel("Trạng thái")
plt.ylabel("Số lượng")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

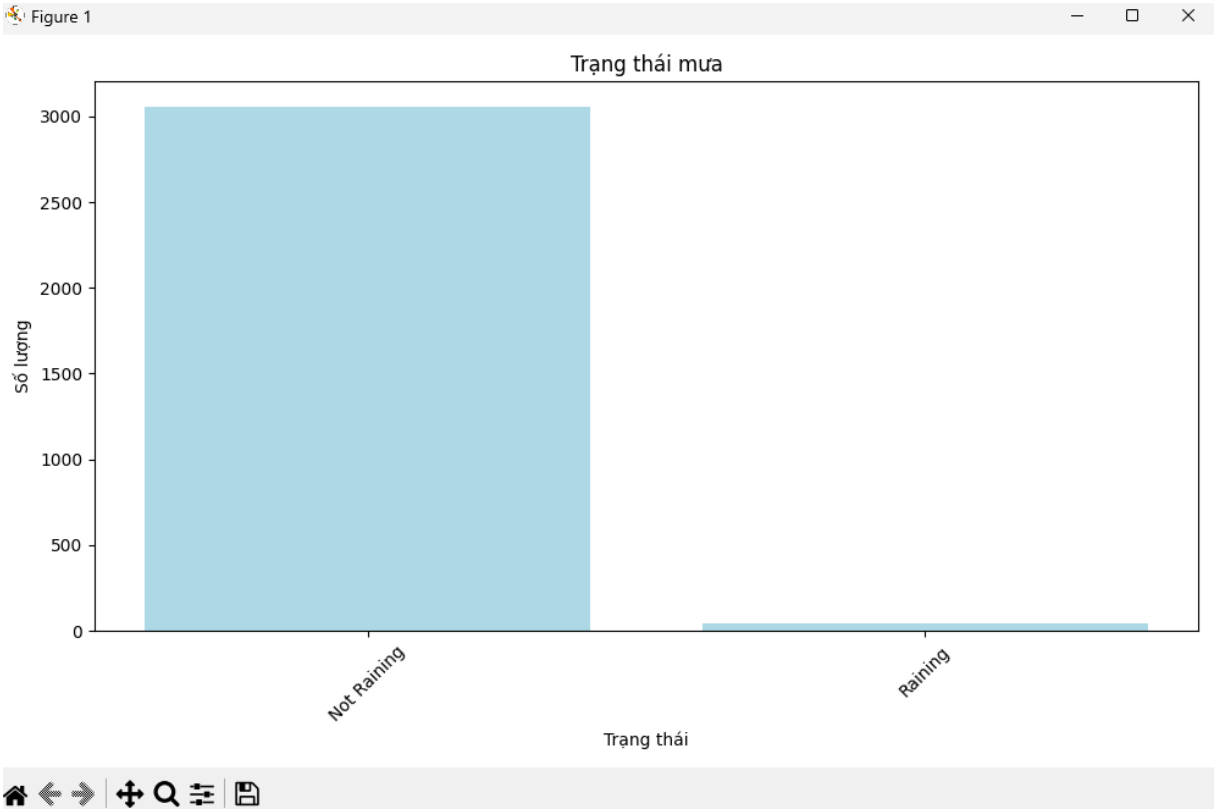
Cuối cùng chúng ta có 3 đồ thị sau:



Hình 10. Đồ thị logs hiển thị trạng thái hoạt động của hệ thống



Hình 11. Đồ thị hiển thị lệnh điều khiển



Hình 12. Đồ thị hiển thị trạng thái mưa

Phụ Lục

LinkGitHub:
https://github.com/banhbiproqt/Nhom3_MangCamBien_TuDongHoa.git
Link video: https://www.youtube.com/watch?v=o_DET7NTcbs