ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC & KĨ THUẬT MÁY TÍNH

Thực tập Đồ án môn học Đa ngành

Green Sense - Nông trại thông minh

Nhóm: Nhóm 83

Giảng viên hướng dẫn: Thầy Mai Đức Trung

Sinh viên thực hiện: Dương Minh Hiếu - 2210978
Trần Anh Khôi - 2211694
Dương Hải Lâm - 2211807
Trần Nguyễn Thanh Lâm - 2211822
Hoàng Nhật Linh - 2211847
Nguyễn Thành Nhân - 2212366
Hồ Tấn Phát - 2212507

Thành phố Hồ Chí Minh, tháng 6 năm 2025

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Danh sách thành viên & Phân chia công việc

STT	Họ và tên	MSSV	Vị trí	Khối lượng công việc
1	Dương Minh Hiếu	2210978	Embedded System Engineer	14.3%
2	Trần Anh Khôi	2211694	Backend Developer	14.3%
3	Dương Hải Lâm	2211807	Frontend Developer	14.3%
4	Trần Nguyễn Thanh Lâm	2211822	Backend Developer	14.3%
5	Hoàng Nhật Linh	2211847	Frontend Developer	14.3%
6	Nguyễn Thành Nhân	2212366	AI Engineer & Backend Developer	14.3%
7	Hồ Tấn Phát	2212507	AI Engineer & Backend Developer	14.3%

Bảng 1: Danh sách thành viên và phân chia công việc

2 Giới thiệu đề tài

2.1 Bối cảnh & động lực

Nhà kính được sử dụng để điều chỉnh điều kiện khí hậu cho những cây trồng nhạy cảm hoặc không phải cây bản địa của khí hậu tự nhiên địa phương. Những cây trồng này có thể yêu cầu một môi trường ổn định và kiểm soát chặt chẽ về nhiệt độ, độ ẩm, ánh sáng và các yếu tố khí hậu khác để phát triển tốt. Tuy nhiên, các yếu tố bên ngoài như ánh sáng mặt trời, gió và mưa có ảnh hưởng mạnh mẽ và thay đổi không lường trước được đến nhà kính, khiến việc duy trì môi trường thích hợp luôn là một công việc tốn nhiều công sức và thời gian.

Vì vậy, các nhà kính thương mại hiện đại ngày nay không còn là những cơ sở sản xuất đơn giản mà đã trở thành các cơ sở sản xuất công nghệ cao. Các thiết bị hiện đại như màn che động cơ tự động, hệ thống điều khiển khí hậu nhân tạo, hệ thống chiếu sáng tự động và các cảm biến môi trường đã được tích hợp để giám sát và điều khiển điều kiện bên trong nhà kính một cách chính xác. Những hệ thống này thường được điều khiển bởi máy tính, giúp tự động hóa các quy trình và tối ưu hóa điều kiện sinh trưởng cho cây trồng, từ đó giảm thiểu sự can thiệp thủ công và nâng cao hiệu quả sản xuất.

Bên cạnh đó, các kỹ thuật tiên tiến như cảm biến nhiệt độ, độ ẩm, áp suất và ánh sáng được sử dụng để thu thập dữ liệu liên tục, giúp người dùng có thể phân tích và điều chỉnh điều kiện một cách chủ động. Những dữ liệu này không chỉ phục vụ việc điều khiển trong thời gian thực, mà còn là cơ sở để đánh giá hiệu suất và điều chỉnh chiến lược canh tác trong dài hạn.

2.2 Mục tiêu của dự án

Dự án hướng đến việc phát triển một nguyên mẫu hệ thống giám sát và điều khiển nhà kính thông minh, kết hợp các tiến bộ trong công nghệ Internet, IoT (Internet of Things) và kết nối không dây. Mục tiêu chính bao gồm:

- Xây dựng một hệ thống có khả năng giám sát các yếu tố môi trường quan trọng bên trong nhà kính như nhiệt độ, độ ẩm và cường độ ánh sáng.
- Cho phép người dùng điều khiển từ xa các thiết bị như quạt thông gió, máy bơm nước, đèn chiếu sáng hoặc hệ thống phun sương.
- Hiển thị dữ liệu thời gian thực và cung cấp giao diện trực quan để quản lý nhà kính một cách tiện lợi thông qua trình duyệt web hoặc ứng dụng di động.
- Góp phần giảm thiểu lao động thủ công, nâng cao hiệu suất sản xuất và tăng độ tin cậy trong quá trình canh tác nông nghiệp công nghệ cao.

2.3 Phạm vi dự án

Trong phạm vi của dự án này, nhóm sẽ tập trung vào phát triển một nguyên mẫu chức năng cơ bản với các thành phần sau:

- Phần cứng: Hệ thống cảm biến để thu thập dữ liệu (nhiệt độ, độ ẩm, ánh sáng), vi điều khiển (như ESP32), các thiết bị điều khiển (quạt, đèn, bơm...).
- Phần mềm nhúng: Chương trình chạy trên vi điều khiển để thu thập dữ liệu từ cảm biến và điều khiển các thiết bị đầu ra theo lệnh từ người dùng hoặc theo điều kiện định sẵn.

- Hệ thống mạng và lưu trữ: Kết nối không dây qua Wi-Fi để truyền dữ liệu đến máy chủ; lưu trữ dữ liệu cảm biến để phục vụ hiển thị và phân tích.
- Giao diện người dùng (Web Dashboard): Hiển thị các thông số môi trường theo thời gian thực, điều khiển từ xa các thiết bị trong nhà kính và cung cấp thông báo trạng thái.

Dự án sẽ không đi sâu vào các giải pháp xử lý dữ liệu nâng cao như AI, machine learning hay điều khiển mờ trong giai đoạn đầu, mà chỉ tập trung vào việc xây dựng một hệ thống giám sát và điều khiển thông minh ở mức cơ bản, khả thi và dễ triển khai.

3 Cơ sở lý thuyết và công nghệ

3.1 Kiến thức nền tảng

3.1.1 Mô hình phát triển phần mềm

Trong suốt quá trình thực hiện dự án, nhóm áp dụng Scrum Framework, một mô hình phát triển phần mềm linh hoạt thuộc phương pháp Agile. Scrum chia quá trình phát triển thành các chu kỳ ngắn gọi là Sprint (thường kéo dài từ 1–2 tuần), trong đó nhóm tập trung hoàn thành một tập hợp các chức năng cụ thể. Việc áp dụng Scrum giúp nhóm dễ dàng thích ứng với các thay đổi yêu cầu, cải thiện chất lượng phần mềm thông qua kiểm tra và phản hồi liên tục, đồng thời thúc đẩy sự phối hợp chặt chẽ giữa các thành viên.

Các thành phần chính trong quy trình Scrum nhóm đã áp dụng bao gồm:

- Product Backlog: Danh sách các yêu cầu và chức năng cần phát triển.
- Sprint Planning: Phiên lập kế hoạch cho từng Sprint.
- Daily Standup: Họp nhanh mỗi ngày để cập nhật tiến độ và xử lý vướng mắc.
- Sprint Review và Sprint Retrospective: Đánh giá kết quả và cải tiến quy trình sau mỗi Sprint.

3.1.2 Kiến trúc phần mềm

Hệ thống phần mềm của dự án được phát triển theo mô hình kiến trúc MVC (Model – View – Controller). Mô hình này giúp tách biệt rõ ràng giữa ba thành phần chính:

- Model: Quản lý dữ liệu và logic nghiệp vụ.
- View: Giao diện người dùng, nơi hiển thị thông tin và nhận tương tác.
- Controller: Điều phối luồng dữ liệu giữa Model và View, xử lý yêu cầu từ người dùng.

Cách tổ chức này giúp tăng tính mô-đun, dễ bảo trì và mở rộng hệ thống trong tương lai, đồng thời phù hợp với phương pháp phát triển theo nhóm với nhiều thành viên phụ trách các phần khác nhau.

3.1.3 MCP (Model – Context – Protocol)

Model Context Protocol (MCP) là một giao thức mở được phát triển bởi Anthropic nhằm chuẩn hóa cách các ứng dụng AI kết nối và tương tác với nguồn dữ liệu cũng như công cụ bên ngoài. MCP được thiết kế để giải quyết vấn đề phân mảnh trong việc tích hợp AI với các hệ thống khác nhau.

Các thành phần chính bao gồm:

- MCP Client: Là ứng dụng AI hoặc công cụ AI (như Claude, ChatGPT), đảm nhiệm việc khởi tạo kết nối và gửi yêu cầu đến MCP Server, đồng thời xử lý và hiển thị kết quả từ server
- MCP Server: Là dịch vụ cung cấp quyền truy cập vào nguồn dữ liệu hoặc công cụ cụ thể, thực hiện việc xử lý các yêu cầu từ client và trả về dữ liệu được định dạng, quản lý xác thực và phân quyền truy cập
- Transport Layer: là lớp giao tiếp giữa client và server, thường sử dụng JSON-RPC over HTTP/WebSocket để đảm bảo tính bảo mật và tin cậy trong truyền tải dữ liệu

Luồng hoạt động:

1. **Input Processing**: Người dùng nhập câu lệnh tiếng Việt (ví dụ: ”Tắt quạt nếu nhiệt độ dưới 28 độ”)
2. **Intent Recognition**: Protocol Layer sử dụng LLM để:
   - Phân tích ngữ nghĩa câu lệnh
   - Xác định thiết bị đích (quạt)
   - Trích xuất điều kiện (nhiệt độ < 28°C)
   - Xác định hành động (tắt)
3. **Context Evaluation**: Truy xuất Context Layer để:
   - Kiểm tra trạng thái hiện tại của hệ thống
   - Đánh giá điều kiện đã đặt ra
   - Xác định tính khả thi của lệnh
4. **Action Execution**: Nếu điều kiện thỏa mãn:
   - Chuyển đổi lệnh thành API calls
   - Gửi tín hiệu điều khiển đến thiết bị IoT
   - Cập nhật trạng thái trong Model Layer
5. **Feedback Loop**:
   - Ghi lại kết quả thực thi
   - Cập nhật Context Layer với thông tin mới
   - Gửi phản hồi xác nhận cho người dùng

Việc áp dụng MCP trong tính năng AI giúp nhóm dễ dàng mở rộng khả năng hiểu và xử lý ngôn ngữ tự nhiên, giảm sự phụ thuộc vào các câu lệnh cứng, đồng thời tách biệt rõ logic diễn dịch, trạng thái và giao tiếp hệ thống. Đây là bước đầu trong việc tạo ra một hệ thống nhà kính thông minh có thể hiểu, phản hồi và hành động dựa trên ngôn ngữ con người – hướng tới trải nghiệm người dùng tự nhiên hơn.

3.1.4 Giao tiếp với thiết bị IoT

Việc giao tiếp với thiết bị IoT được thực hiện thông qua giao thức MQTT, sử dụng nền tảng Adafruit IO làm MQTT broker trung gian. Backend gửi các tín hiệu điều khiển thiết bị (như bật/tắt, thay đổi ngưỡng...) bằng cách publish dữ liệu đến các topic MQTT trên Adafruit IO. Yolo:Bit thực hiện việc subscribe các topic này để nhận lệnh và phản hồi trạng thái. Hệ thống cũng thu thập dữ liệu cảm biến từ thiết bị gửi ngược về, đảm bảo hai chiều truyền thông qua MQTT diễn ra ổn định và theo thời gian thực.

Việc tích hợp MQTT – Adafruit IO mang lại giải pháp giao tiếp nhẹ, hiệu quả, thời gian thực và dễ triển khai cho hệ thống nhà kính thông minh, đặc biệt phù hợp với môi trường mạng không ổn định hoặc tài nguyên phần cứng hạn chế.

3.2 Công nghệ sử dụng

3.2.1 Frontend

Phần giao diện người dùng (frontend) của hệ thống được xây dựng bằng React, một thư viện JavaScript phổ biến do Facebook phát triển. React cho phép phát triển giao diện tương tác, có khả năng cập nhật dữ liệu theo thời gian thực mà không cần tải lại toàn bộ trang. Việc sử dụng component-based architecture của React giúp nhóm dễ dàng tái sử dụng mã nguồn và tăng tốc độ phát triển giao diện. Backend

3.2.2 Backend

Phần xử lý logic và API phía máy chủ được hiện thực bằng Flask, một micro-framework viết bằng Python. Flask nhẹ, linh hoạt, dễ tích hợp với nhiều thư viện mở rộng và đặc biệt phù hợp với các dự án nhỏ hoặc nguyên mẫu (prototype). Backend chịu trách nhiệm nhận dữ liệu từ cảm biến (qua Wi-Fi), lưu trữ vào cơ sở dữ liệu, xử lý lệnh điều khiển từ người dùng, và trả về dữ liệu cho frontend hiển thị.

Việc giao tiếp giữa Adafruit IO và thiết bị Yolo:Bit được lập trình bằng ngôn ngữ C++, thông qua thư viện hỗ trợ giao thức MQTT dành cho vi điều khiển ESP32. Trong đó, Yolo:Bit thực hiện việc subscribe đến các topic tương ứng trên Adafruit IO để nhận lệnh điều khiển từ hệ thống, đồng thời publish dữ liệu cảm biến thu thập được về lại broker. Toàn bộ quá trình này được thực thi trực tiếp trên firmware chạy C++ nhằm đảm bảo hiệu suất xử lý, độ trễ thấp và tính ổn định trong môi trường nhúng tài nguyên hạn chế.

3.2.3 Cơ sở dữ liệu

Nhóm sử dụng MongoDB làm hệ quản trị cơ sở dữ liệu. Đây là một cơ sở dữ liệu NoSQL dạng tài liệu, cho phép lưu trữ dữ liệu dưới dạng BSON (Binary JSON). MongoDB đặc biệt phù hợp với các hệ thống IoT do khả năng lưu trữ linh hoạt, không cần định nghĩa cấu trúc cứng nhắc, và dễ dàng mở rộng khi lượng dữ liệu lớn lên theo thời gian. Các công cụ hỗ trợ

3.2.4 Các công cụ hỗ trợ

Trong quá trình phát triển, nhóm sử dụng GitHub làm nền tảng quản lý mã nguồn. GitHub hỗ trợ làm việc nhóm hiệu quả thông qua các chức năng như:

- Quản lý phiên bản (version control)
- Tạo nhánh phát triển độc lập (branching)
- Gộp mã (pull request) và rà soát mã (code review)
- Theo dõi tiến độ công việc bằng các công cụ như Projects, Issues và Milestones

4 Danh sách thiết bị IoT

4.1 Cảm biến nhiệt độ độ ẩm DHT20

- **Ứng dụng:** Đo nhiệt độ và báo cáo lên máy chủ IoT. Nhiệt độ là một chỉ số quan trọng cho nhiều hoạt động bảo trì nhà kính, bao gồm nhưng không giới hạn ở việc có tưới nước hay không, tưới bao nhiêu, có triển khai màn che nắng hay không, v.v.
- **Đầu vào:** Nhiệt độ không khí - cảm biến được gắn trên mặt đất, tốt nhất là treo ở trung tâm thể tích của nhà kính.
- **Đầu ra:** Giá trị nhiệt độ, được báo cáo lên bộ điều khiển, bộ điều khiển sẽ chuyển tiếp tới máy chủ IoT.

4.2 Cảm biến độ ẩm đất

- **Ứng dụng:** Đo độ ẩm của đất và sử dụng kết quả từ cảm biến để quyết định việc tưới nước.
- **Đầu vào:** Độ ẩm của đất.
- **Đầu ra:** Dữ liệu số hóa về độ ẩm của đất.

4.3 Cảm biến ánh sáng

- **Ứng dụng:** Giúp nhận biết và đo lường cường độ ánh sáng của môi trường nhà kính. Có vai trò quyết định việc mở đèn thêm.
- **Đầu vào:** Ánh sáng.
- **Đầu ra:** Số liệu về cường độ ánh sáng.

4.4 Động cơ bơm nước

- **Ứng dụng:** Dùng để bơm nước, phục vụ cho việc tưới cây.
- **Đầu vào:** Nguồn điện.
- **Đầu ra:** Nếu nhận được điện, động cơ sẽ bắt đầu việc bơm nước để tưới cây.

4.5 Động cơ Servo SG90S

- **Ứng dụng:** Sử dụng để điều khiển mái che, có mục đích che nắng.
- **Đầu vào:** Lệnh điều khiển từ người dùng, hoặc dữ liệu từ cảm biến ánh sáng
- **Đầu ra:** Đóng và mở mái che dựa vào dữ liệu được truyền đến.

4.6 Đèn 4 LED RGB

- **Ứng dụng:** Được sử dụng như một đèn quang hợp tượng trưng.
- **Đầu vào:** Lệnh từ người dùng, hoặc dữ liệu từ cảm biến ánh sáng.
- **Đầu ra:** Bật hoặc tắt đèn tùy theo dữ liệu được truyền đến.

4.7 Yolo:Bit

Ứng dụng: Đóng vai trò máy chủ của hệ thống, nhận các thông tin từ các cảm biến và gửi dữ liệu nhận được lên máy chủ MQTT.

4.8 Mạch mở rộng

Ứng dụng: Cung cấp thêm các port kết nối cho các cảm biến gắn với Yolo:Bit

5 Phân tích và thiết kế hệ thống

5.1 Phân tích yêu cầu

5.1.1 Yêu cầu chức năng

Về mặt thiết bị IoT:

- Cung cấp báo cáo khí hậu trực tiếp dưới dạng biểu đồ và nhật ký, giúp theo dõi nhiệt độ, độ ẩm và ánh sáng trong nhà kính.
- Tưới nước tự động theo độ ẩm đất và loại cây, đảm bảo cây phát triển khỏe mạnh mà không cần tưới thủ công thường xuyên.
- Duy trì điều kiện khí hậu lý tưởng (nhiệt độ, thông gió, chiếu sáng) bằng phương thức thủ công, tự động hoặc theo lịch trình cài đặt trước.
- Phát hiện và báo cáo các sự cố khi không thể duy trì điều kiện môi trường trong ngưỡng cho phép, giúp người dùng can thiệp kịp thời.
- Nhắc nhở định kỳ các công việc bảo trì thủ công như kiểm tra nguồn nước, vệ sinh cửa sổ và đảm bảo hệ thống thông gió hoạt động hiệu quả.

Về mặt Web App:

- Cho phép người dùng điều khiển thủ công các chức năng cụ thể trong nhà kính, như tưới cây, bật/tắt đèn và kích hoạt hệ thống cảnh báo, giúp linh hoạt trong việc quản lý môi trường trồng trọt.
- Cung cấp số liệu khí hậu hiện tại cùng với dữ liệu lịch sử, giúp người dùng theo dõi sự thay đổi môi trường và đưa ra quyết định phù hợp cho cây trồng.
- Thực hiện phân tích thống kê và cung cấp báo cáo chi tiết về tình trạng và hiệu suất hoạt động của hệ thống, giúp tối ưu hóa quá trình vận hành và bảo trì.

5.1.2 Yêu cầu phi chức năng

Về mặt thiết bị IoT:

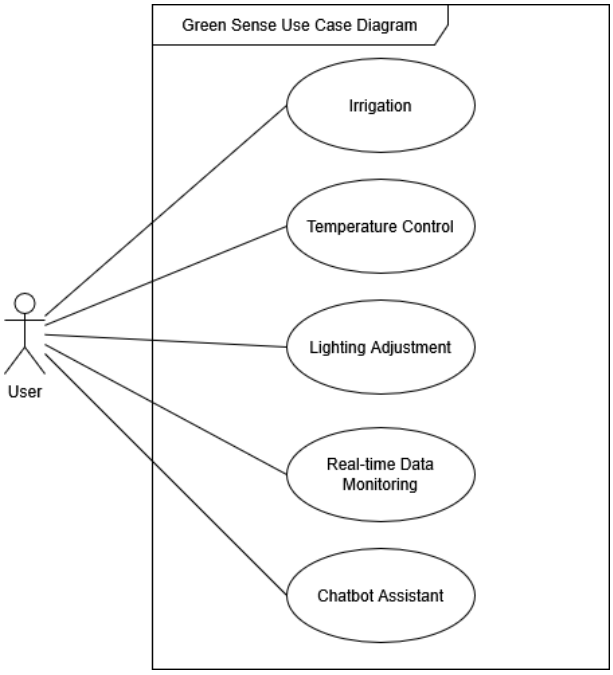
- Hệ thống phải hoạt động liên tục 24/7 với thời gian gián đoạn theo lịch trình không quá 2 giờ mỗi tháng và không có gián đoạn đột xuất quá 10 phút.
- Độ trễ của bộ truyền động khi thực hiện lệnh không được vượt quá 5 giây trong 95% trường hợp kiểm tra.
- Hệ thống phải ghi nhật ký dữ liệu lịch sử liên tục, lưu trữ ít nhất 180 ngày với độ chính xác $\pm 1\%$ và không mất dữ liệu quá 0,1% trong quá trình vận hành.
- Cơ sở dữ liệu phải hỗ trợ lưu trữ tối thiểu 500MB và có thể mở rộng lên đến 2GB mà không ảnh hưởng đến hiệu suất truy xuất dữ liệu (tốc độ phản hồi dưới 1 giây cho 95% truy vấn).
- Hướng dẫn sử dụng phải dễ hiểu, cho phép người dùng mới vận hành hệ thống trong vòng 10 phút mà không cần trợ giúp. Giao diện LCD và nút bấm phải phản hồi trong vòng 1 giây sau khi thao tác.

Về mặt Web App:

- Hệ thống phải dễ sử dụng cho mọi độ tuổi, đặc biệt hướng đến nông dân, người làm vườn và quản lý nông nghiệp, với thời gian đào tạo không quá 15 phút.
- Ứng dụng web phải tương thích với các trình duyệt phổ biến, bao gồm Chrome, Firefox, Edge và Safari, với hiệu suất ổn định trên cả máy tính và thiết bị di động.
- Kết nối với máy chủ phải hoàn tất trong vòng 2 giây, trong khi các thao tác giao diện cơ bản (bấm nút, cuộn trang) phải phản hồi trong vòng 500ms.
- Hỗ trợ lưu trữ dữ liệu cục bộ tối đa 100MB, đồng thời đồng bộ dữ liệu với máy chủ mà không làm gián đoạn hoạt động của ứng dụng.
- Người dùng phải có thể xác thực bằng mật khẩu và/hoặc hình vẽ mẫu, với mật khẩu được lưu trữ an toàn theo tiêu chuẩn mã hóa SHA-256 hoặc cao hơn.

5.2 Use Case Diagram

5.2.1 Toàn bộ hệ thống



Hình 1: Biểu đồ use case cho hệ thống Green Sense

5.2.2 Bảng mô tả cho từng use case

Use case 1 - Tưới nước

Usecase	**Tưới nước**
Use case ID	1
Actor	Người dùng của hệ thống
Description	Hệ thống cung cấp ba chế độ tưới: – Tự động: Duy trì độ ẩm đất theo mức cài đặt sẵn. – Lập lịch: Tưới theo lịch do người dùng đặt. – Thủ công: Người dùng tự bật/tắt tưới.
Precondition	Hệ thống đang hoạt động bình thường và được trang bị ít nhất một máy bơm, một cảm biến độ ẩm đất và một nguồn nước
Postcondition	Dữ liệu độ ẩm đất được ghi lại theo tần suất mà kế hoạch hiện tại kiểm tra độ ẩm. Ngoài ra, hệ thống cũng lưu trữ nhật ký tất cả các lần tưới, kèm theo thông tin về sự kiện kích hoạt tưới.
Normal Flow	Quy trình hoạt động dự kiến là người dùng ủy quyền việc tưới tiêu cho hệ thống. 1. Người dùng mở trang web của hệ thống. 3. Người dùng chọn ”Tưới tiêu” từ danh sách tính năng trên giao diện chính. 4. Người dùng chọn ”Tự động” trong phần Chế độ. 5. Người dùng đặt mức độ ẩm mục tiêu theo bảng giá trị khuyến nghị trên màn hình hoặc điều chỉnh bằng thanh trượt để chọn giá trị tùy chỉnh. 6. Hệ thống đánh giá từng phép đo nhận được và thực hiện hành động nếu cần. Cụ thể, nếu độ ẩm thấp hơn 50% so với mức mục tiêu, hệ thống sẽ tưới trong 5 giây.
Alternative Flow	Tại bước 4, nếu người dùng chọn tưới theo lịch: 4.1 Người dùng chọn thêm một nhiệm vụ tưới mới. 4.2 Người dùng chọn khoảng thời gian lập lịch (theo chế độ Cron hoặc chế độ Interval). Biểu mẫu tương ứng sẽ hiển thị. 4.3 Người dùng đặt thời gian tưới trong lịch trình đó và đặt thời gian bơm nước cần sử dụng. 4.4 Hệ thống tưới theo lịch trình đã đặt. Nếu hành động tự động được bật, hệ thống sẽ theo dõi độ ẩm đất mỗi khi có dữ liệu mới và có thể gửi cảnh báo qua thông báo nếu cần. Tại bước 4, nếu người dùng chọn tưới hoàn toàn thủ công: 4.1 Người dùng chỉ định hệ thống bật bơm nước trong khoảng thời gian do người dùng tự thiết lập Chi tiết về giao diện ứng dụng có thể thay đổi trong quá trình phát triển.
Exception Flow	Tại bước 6, nếu hệ thống phát hiện sự thay đổi bất thường (hoặc không có thay đổi) trong độ ẩm đất khi đang tưới: a. Hệ thống không phát hiện sự thay đổi độ ẩm nhanh như mong đợi hoặc không có thay đổi nào. b. Hệ thống gửi thông báo đẩy đến ứng dụng trên điện thoại, cảnh báo về sự cố với nguồn nước (hết nước hoặc tắc nghẽn) hoặc bơm. c. Nếu cơ chế dự phòng được kích hoạt, hệ thống thực hiện hành động mạnh hơn (bật bơm trong 20 giây).

Bảng 2: Bảng mô tả cho use case Tưới nước

Use case 2 - Điều chỉnh nhiệt độ

Usecase	**Điều chỉnh nhiệt độ**
Use case ID	2
Actor	Người dùng của hệ thống
Description	Với tính năng kiểm soát nhiệt độ, hệ thống cung cấp 3 lựa chọn: – **Chế độ tự động:** Hệ thống cố gắng duy trì nhiệt độ bên trong nhà kính ở mức lý tưởng. – **Chế độ theo lịch:** Tuân theo lịch trình mà người dùng cài đặt, kèm theo tùy chọn dự phòng an toàn (nếu muốn) để hệ thống tự điều chỉnh khi có biến động bất thường. – **Chế độ thủ công:** Người dùng toàn quyền điều khiển đóng/mở các tấm che nắng (là những tấm che trên trần có khả năng cản sáng; trong mô hình này, chúng ta dùng động cơ mô phỏng việc cuốn/mở tấm che), bất cứ khi nào họ muốn.
Precondition	Hệ thống đang vận hành bình thường. Hệ thống được trang bị cảm biến DHT20 để đo nhiệt độ và độ ẩm. Tùy chọn, hệ thống có lắp các tấm che nắng kết nối với bộ truyền động (Động cơ Servo SG90S) để cuộn vào hoặc mở ra.
Postcondition	Nhiệt độ (đo bởi cảm biến) và trạng thái của tấm che nắng được ghi nhận vào cơ sở dữ liệu của hệ thống theo định kỳ. Mọi hành động điều chỉnh nhiệt độ (do người dùng thực hiện hoặc do hệ thống tự động) đều được ghi lại vào lịch sử.
Normal Flow	1. Người dùng mở trang web của hệ thống. 2. Người dùng chọn “Temperature control” (Kiểm soát nhiệt độ) từ danh sách tính năng trên màn hình chính. 3. Người dùng chọn “Scheduled mode” (Chế độ theo lịch). 4. Người dùng chọn khung giờ mong muốn và chọn góc độ để tấm che nắng mở hoặc đóng.
Alternative Flow 1	3.1. Tại bước 3, nếu người dùng chọn “Automatic” (Tự động): 3.1.1 Người dùng chỉ định khoảng nhiệt độ mục tiêu. 3.1.2 Hệ thống bắt đầu kiểm tra mức nhiệt mỗi khi có dữ liệu mới từ cảm biến. Ví dụ, nếu nhiệt độ bên trong nhà kính thấp hơn khoảng mục tiêu từ 3 đến 5 độ C, hệ thống sẽ mở các tấm che nắng theo góc mà người dùng đã cài đặt từ trước
Alternative Flow 2	3.2. Tại bước 3, nếu người dùng chọn “Manual” (Thủ công): 3.2.1 Người dùng chỉ định góc độ mà tấm che nắng sẽ mở và chọn ”điều khiển tấm che nắng” 3.2.2 Tấm che nắng sẽ được mở với góc độ mà người dùng đã cài đặt

Bảng 3: Bảng mô tả cho use case Điều chỉnh nhiệt độ

Use case 3 - Điều chỉnh ánh sáng

Usecase	**Điều chỉnh ánh sáng**
Use case ID	3
Actor	Người dùng của hệ thống
Description	Chúng ta có thể tùy chỉnh cho phù hợp các mức độ ánh sáng phù hợp trong nhà trồng cây với 3 chế độ: tự động điều chỉnh phù hợp với điều kiện môi trường, lập lịch các khoảng thời gian trong ngày hoặc các ngày trong tuần, hoặc người dùng có thể điều chỉnh ánh sáng của đèn trực tiếp.
Precondition	Hệ thống hoạt động bình thường và luôn được trang bị cảm biến ánh sáng và một mạch role gắn vào nguồn chiếu sáng của nguồn chiếu sáng.
Postcondition	Hệ thống ánh sáng hoạt động đúng như cách người dùng thiết lập. Các hành động điều chỉnh mức độ sáng bởi người dùng hay hệ thống đều được lưu vào lịch sử theo dõi.
Normal Flow	1. Người dùng mở trang web của hệ thống. 2. Người dùng chọn “Điều chỉnh chế độ sáng” từ danh sách các tính năng trên màn hình chính. 3. Người dùng chọn “Lập lịch”. 4. Hệ thống cung cấp một thanh trượt dựa trên phạm vi biểu thị 24h trong ngày. 5. Người dùng chọn khoảng thời gian mong muốn trong ngày để đèn hoạt động và các mức độ để đèn bật (với cường độ sáng tăng từ 1 đến 4, tắt đèn là 0).
Alternative Flow 1	3.1. Ở bước 3, nếu người dùng chọn ‘Tự động’ trên thanh tùy chọn: 3.1.1. Người dùng chọn cụ thể mức độ sáng mục tiêu 3.1.2. Hệ thống bắt đầu kiểm tra và nhận dữ liệu về sau mỗi đơn vị thời gian. Nếu mức độ sáng không phù hợp với cấu hình không đúng với mục tiêu, hệ thống sẽ điều chỉnh đến khi nào cảm biến ánh sáng nhận được dữ liệu với mức độ sáng theo cách người dùng thiết lập
Alternative Flow 2	3.2. Ở bước 3, nếu người dùng chọn ‘Thủ công’ trên thanh tùy chọn: 3.2.1. Người dùng thiết lập độ sáng cho đèn (với cường độ sáng tăng từ 1 đến 4, tắt đèn là 0) và chọn “Điều khiển đèn” 3.2.2. Đèn sẽ được bật/tắt theo thiết lập của người dùng.

Bảng 4: Bảng mô tả cho use case Điều chỉnh ánh sáng

Use case 4 - Theo dõi dữ liệu theo thời gian thực và thống kê hệ thống

Usecase	**Theo dõi dữ liệu theo thời gian thực và thống kê hệ thống**
Use case ID	4
Actor	Người dùng của hệ thống
Description	Ứng dụng cho phép người dùng xem dữ liệu đã ghi dưới dạng báo cáo tổng hợp (trong một khoảng thời gian) với các tính toán tổng hợp và biểu đồ tổng quan về các bản ghi trước đó.
Precondition	Trang web được kết nối với cơ sở dữ liệu và hệ thống vẫn đang ghi lại thống kê.
Postcondition	Không có.
Normal Flow	1. Người dùng mở trang web của hệ thống. 2. Người dùng chọn 'Thống kê' từ danh sách các tính năng trên màn hình chính. 3. Hệ thống trích xuất dữ liệu hàng ngày làm chế độ báo cáo mặc định và hiển thị cho người dùng. Báo cáo bao gồm biểu đồ đường về mức ánh sáng, độ ẩm và nhiệt độ trong khoảng thời gian đó; các hành động đáng chú ý được thực hiện thủ công hoặc tự động trên hệ thống nhà kính; và trạng thái mới nhất của nhà kính (các phép đo nhận được gần nhất). 4. Người dùng có thể chọn khoảng thời gian ưa thích bằng cách chọn 'Khoảng thời gian' trên thanh tùy chọn. 5. Hệ thống cung cấp lịch chi tiết để người dùng chọn ngày bắt đầu và ngày kết thúc cho báo cáo. 6. Người dùng chọn khoảng thời gian ưa thích và nhấn 'Xác nhận'. 7. Hệ thống tải lại trang báo cáo với chế độ hiển thị biểu đồ phù hợp và theo dõi hành động. 8. Người dùng nhấn 'Hoàn tất' để quay lại trang chính.
Alternative Flow	
Exception Flow	

Bảng 5: Bảng mô tả cho use case Theo dõi dữ liệu theo thời gian thực và thống kê hệ thống

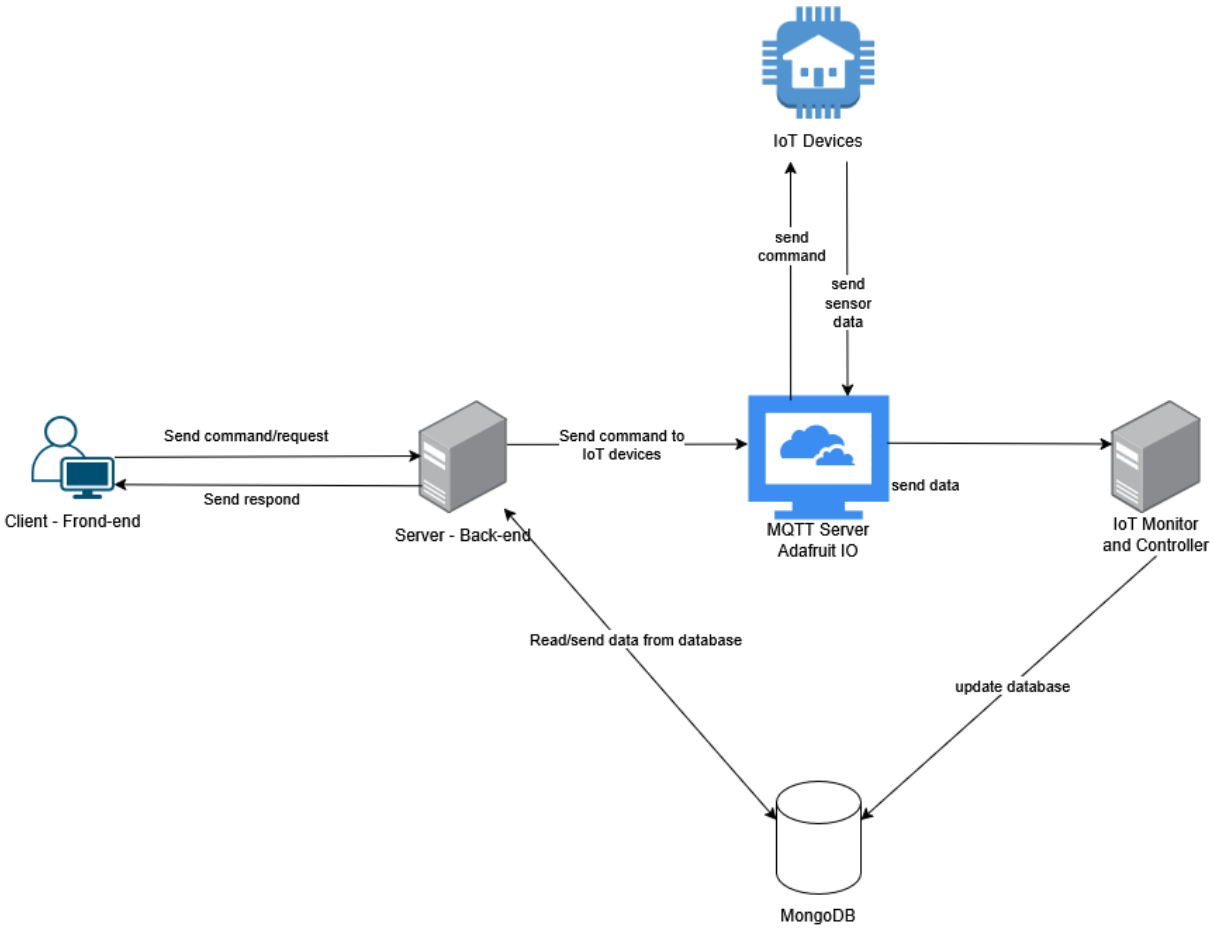
Use case 5 - Điều khiển IoT Device trong hệ thống bằng AI

Usecase	**Điều khiển IoT Device trong hệ thống bằng AI**
Use case ID	5
Actor	Người dùng của hệ thống
Description	Người dùng sử dụng khung chat trên giao diện web để gửi lệnh điều khiển thiết bị IoT bằng ngôn ngữ tự nhiên. Hệ thống AI sẽ phân tích lệnh, ánh xạ với API đã được huấn luyện và thực hiện lệnh phù hợp.
Precondition	- Người dùng đã đăng nhập hệ thống. - AI được cấp quyền truy cập vào hệ thống web server. - AI đã được huấn luyện với tài liệu API tương ứng.
Postcondition	- Hệ thống thực hiện đúng lệnh điều khiển IoT nếu AI hiểu được lệnh. - Nếu lệnh không hợp lệ hoặc không hiểu được, AI sẽ thông báo lỗi cho người dùng mà không thực hiện hành động.
Normal Flow	1. Người dùng truy cập vào trang web hệ thống. 2. Người dùng chọn tính năng ”Control IoT with AI helper” từ thanh công cụ. 3. Giao diện chat được hiển thị. 4. Người dùng nhập lệnh điều khiển bằng ngôn ngữ tự nhiên. 5. AI phân tích ngữ nghĩa lệnh và ánh xạ với lời gọi API phù hợp. 6. AI gửi lời gọi API đến server điều khiển IoT Device. 7. Thiết bị IoT phản hồi và thực hiện hành động.
Alternative Flow	
Exception Flow	- Tại bước 5, nếu AI không hiểu lệnh (do chưa được huấn luyện hoặc lệnh không rõ ràng): + AI thông báo lỗi: “Tôi chưa hiểu rõ lệnh này. Bạn có thể diễn đạt lại?” + Không thực hiện gọi API.

Bảng 6: Bảng mô tả cho use case Điều khiển IoT Device trong hệ thống bằng AI

5.3 Thiết kế hệ thống

5.3.1 Kiến trúc tổng thể



Hình 2: Kiến trúc tổng thể của hệ thống Green Sense

Các thành phần chính:

- *Frontend (React)*: Cung cấp giao diện người dùng trực quan để giám sát trạng thái nhà kính, điều khiển thiết bị, và tương tác với tính năng điều khiển thiết bị bằng ngôn ngữ tự nhiên qua AI. Frontend gửi các yêu cầu đến backend qua API REST để thực hiện các thao tác.
- *Backend (Flask)*: Xử lý logic nghiệp vụ, quản lý dữ liệu và tiếp nhận các lệnh từ frontend. Backend có trách nhiệm điều phối hoạt động của thiết bị IoT thông qua giao thức MQTT và kết nối với cơ sở dữ liệu MongoDB để lưu trữ trạng thái và lịch sử hoạt động của hệ thống.
- *Database (MongoDB)*: Lưu trữ thông tin về trạng thái hiện tại của các thiết bị, dữ liệu cảm biến thu thập được và lịch sử hoạt động để phục vụ cho việc giám sát, phân tích và báo cáo.
- *Thiết bị IoT (tích hợp trên Yolo:Bit)*: Thiết bị vật lý chịu trách nhiệm thu thập dữ liệu cảm biến như nhiệt độ, độ ẩm, ánh sáng và thực thi các lệnh điều khiển nhận được từ hệ thống. Thiết bị giao tiếp với backend thông qua giao thức MQTT trên nền tảng Adafruit IO. Firmware của thiết bị được phát triển bằng ngôn ngữ C++ để đảm bảo hiệu suất và ổn định trong môi trường nhúng.
- *IoT Monitor and Controller (Python)*: Là module độc lập được phát triển bằng Python, có nhiệm

vụ lắng nghe các dữ liệu cảm biến thời gian thực từ nền tảng Adafruit IO. Khi nhận dữ liệu, module này sẽ thực hiện việc bật/tắt các thiết bị nếu chế độ điều khiển đang được đặt ở trạng thái tự động. Đồng thời, nó sẽ lưu trữ dữ liệu thu thập được lên MongoDB để đảm bảo tính nhất quán và theo dõi trạng thái hệ thống.

- *Nền tảng MQTT – Adafruit IO*: Là dịch vụ trung gian đóng vai trò broker MQTT, kết nối backend và thiết bị IoT, đảm bảo việc truyền nhận dữ liệu và lệnh điều khiển được diễn ra một cách nhanh chóng, ổn định và theo thời gian thực.

Luồng dữ liệu và tương tác giữa các thành phần:

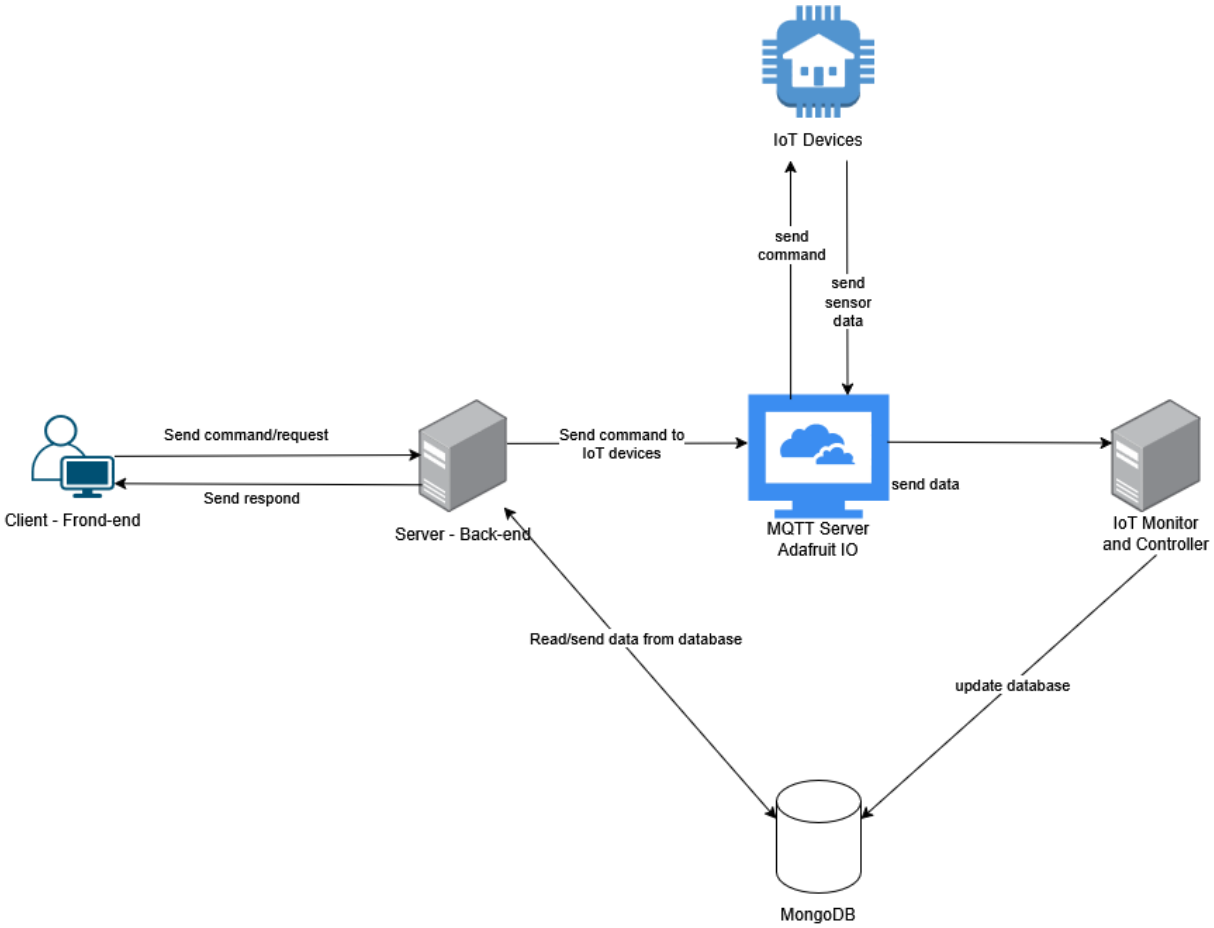
1. Người dùng tương tác với giao diện frontend để thực hiện giám sát và điều khiển nhà kính.
2. Frontend gửi yêu cầu thông qua API REST đến backend xử lý.
3. Backend chuyển lệnh điều khiển đến thiết bị IoT qua giao thức MQTT trên Adafruit IO.
4. Thiết bị IoT nhận lệnh và thực thi các hành động, đồng thời gửi dữ liệu cảm biến về lại Adafruit IO.
5. Module IoT Monitor and Controller nhận dữ liệu cảm biến từ Adafruit IO, quyết định hành động bật/tắt thiết bị nếu ở chế độ tự động, và cập nhật dữ liệu vào MongoDB.
6. Backend truy xuất dữ liệu từ MongoDB để phản hồi trạng thái và lịch sử hoạt động cho frontend hiển thị.

5.3.2 Thiết kế cơ sở dữ liệu

Hệ thống sử dụng cơ sở dữ liệu để lưu trữ ba loại thông tin chính:

1. Lịch sử hoạt động của các thiết bị IoT
2. Dữ liệu thu thập từ các cảm biến (sensors)
3. Thông tin về các tác vụ được người dùng cấu hình trong chế độ lập lịch hoạt động của thiết bị

Do sử dụng MongoDB – một hệ quản trị cơ sở dữ liệu NoSQL dạng document – nên cấu trúc cơ sở dữ liệu không tuân theo mô hình quan hệ truyền thống. Thay vào đó, thiết kế được thể hiện dưới dạng Class Diagram, nhằm mô tả trực quan các trường dữ liệu có trong từng collection.



Hình 3: Class Diagram mô tả cấu trúc cơ sở dữ liệu MongoDB của hệ thống

Mô hình dữ liệu bao gồm các collection chính sau:

- Activity: Lưu trữ các hoạt động của thiết bị với các thuộc tính gồm:
  - `time` (datetime): Thời điểm ghi nhận hoạt động
  - `data` (int): Giá trị định lượng của hoạt động
  - `activity_type` (str): Loại hoạt động được thực hiện
- Humid, Light, Temperature: Mỗi collection lưu dữ liệu tương ứng từ cảm biến độ ẩm, ánh sáng và nhiệt độ. Các trường dữ liệu gồm:
  - `time` (datetime): Thời điểm ghi nhận
  - `data` (int): Giá trị đo được từ cảm biến
- Scheduler_job: Lưu thông tin về các job được người dùng lập lịch cho thiết bị. Các trường dữ liệu gồm:
  - `job_id` (str): Mã định danh của job
  - `device_id` (str): Mã thiết bị tương ứng
  - `action` (str): Hành động sẽ được thực hiện
  - `action_options` (dict): Các tùy chọn cấu hình cho hành động

- `trigger` (str): Kiểu kích hoạt job (ví dụ: theo thời gian, sự kiện,...)
- `trigger_options` (dict): Các tùy chọn cấu hình cho trigger

Thiết kế này đảm bảo tính mở rộng, linh hoạt và dễ tích hợp với các hệ thống IoT, đồng thời phù hợp với đặc điểm lưu trữ phi quan hệ của MongoDB.

6 Triển khai và thực hiện

6.1 Các tính năng đã triển khai

6.1.1 Tính năng điều khiển thiết bị thủ công

Mục tiêu

Tính năng điều khiển thiết bị một cách thủ công cho phép người dùng chủ động điều chỉnh các thiết bị trong nhà kính như servo, đèn (light) và máy bơm (pump) theo ý muốn, thay vì phụ thuộc vào chế độ tự động. Điều này giúp người dùng kiểm soát tốt hơn trong những tình huống đặc biệt, hoặc khi cần can thiệp trực tiếp vào hoạt động của thiết bị.

Quy trình hoạt động - Lấy ví dụ là điều khiển động cơ Servo:

1. Người dùng nhập lệnh điều khiển thông qua giao diện frontend, ví dụ như chọn góc mở mong muốn của servo.
2. Frontend gửi yêu cầu POST đến API `/api/adafruit/servo` trên backend Flask, kèm theo dữ liệu góc (`angle`) do người dùng nhập.

```
@ada_fruit.route('/api/adafruit/servo', methods=['POST'])
def route_adafruit_servo():
    ...
    angle = int(data['angle'])
    return ctl_adafruit_servo(angle)

```

3. Flask Backend kiểm tra và xử lý dữ liệu:
   - Xác minh dữ liệu đầu vào (kiểm tra có tồn tại `angle` hay không).
   - Kiểm tra tính hợp lệ của góc: chỉ chấp nhận giá trị từ 0 đến 180 độ.
   - Nếu dữ liệu hợp lệ, backend sử dụng thư viện `Adafruit_IO` để gửi giá trị góc lên feed `servo` trên nền tảng Adafruit IO.

```
def route_adafruit_servo(angle):
    servo_feed = aio.feeds('servo')
    ...
    aio.send_data(servo_feed.key, angle)

```

4. Thiết bị IoT (Yolo:Bit) đã đăng ký lắng nghe (subscribe) topic `servo` sẽ nhận được dữ liệu mới, sau đó:
   - Diễn giải giá trị góc nhận được.
   - Điều khiển servo mở theo đúng góc đã được chỉ định.

Cách phát triển tính năng điều khiển thủ công cho các thiết bị khác như đèn (light) và máy bơm (pump) được thực hiện tương tự như với servo:

- Tạo một API riêng cho từng thiết bị, nhận giá trị điều khiển (ví dụ: ON/OFF hoặc phần trăm độ sáng).
- Đẩy dữ liệu tương ứng lên feed Adafruit IO đã định danh sẵn (ví dụ: `light, pump`).
- Thiết bị Yolo:Bit lắng nghe từng feed tương ứng và thực hiện hành động điều khiển phù hợp.

6.1.2 Tính năng điều khiển thiết bị tự động

Mục tiêu

Tính năng điều khiển thiết bị ở chế độ tự động được phát triển nhằm giúp hệ thống nhà kính có khả năng phản ứng linh hoạt và chủ động trước các điều kiện môi trường thay đổi. Thay vì yêu cầu người dùng điều khiển thủ công từng thiết bị, hệ thống sẽ tự động đưa ra quyết định kích hoạt hoặc điều chỉnh thiết bị như máy bơm nước (pump), đèn chiếu sáng (light), hoặc động cơ servo dựa trên các ngưỡng điều kiện do người dùng cài đặt trước đó.

Việc điều khiển tự động giúp tối ưu hóa hoạt động vận hành, giảm thiểu sự can thiệp thủ công và đảm bảo cây trồng luôn được chăm sóc trong điều kiện phù hợp nhất. Đây là một trong những bước quan trọng trong việc nâng cấp hệ thống nhà kính truyền thống thành một hệ thống nông nghiệp thông minh ứng dụng công nghệ IoT.

Quy trình hoạt động - Lấy ví dụ là điều khiển máy bơm nước

1. Người dùng bật chế độ tự động cho thiết bị pump
   Thông qua giao diện frontend, người dùng có thể chuyển chế độ hoạt động của thiết bị sang ”automatic” bằng cách gửi một yêu cầu POST đến API `/api/mode/pump/automation`.
2. Người dùng cấu hình các thông số tự động
   Yêu cầu gửi kèm theo hai tham số chính trong phần `automatic_options`:
   - `threshold`: ngưỡng độ ẩm đất. Nếu độ ẩm đo được thấp hơn ngưỡng này, hệ thống sẽ kích hoạt pump.
   - `duration`: khoảng thời gian (tính bằng giây) mà máy bơm sẽ hoạt động khi được kích hoạt.
3. Backend ghi nhận chế độ và cấu hình tự động
   Flask backend lưu trạng thái chế độ hoạt động và các tham số `threshold`, `duration` vào một tệp JSON cấu hình (`IOT_ENV`), dùng để truy xuất nhanh khi có sự kiện xảy ra.
4. Sensor đo độ ẩm gửi dữ liệu thời gian thực lên Adafruit IO
   Thiết bị cảm biến độ ẩm đất gửi giá trị đo được lên feed riêng trên nền tảng Adafruit IO.
5. Module giám sát tự động nhận và xử lý dữ liệu (IoT Monitor and Controller) Một lớp giám sát mang tên `humidListener` lắng nghe dữ liệu từ feed cảm biến. Khi có dữ liệu mới, hệ thống thực hiện các bước:
   - Kiểm tra xem thiết bị pump hiện đang ở chế độ `automatic`.
   - Đối chiếu dữ liệu độ ẩm mới với ngưỡng `threshold` đã cài đặt.
   - Nếu độ ẩm < threshold: pump được bật lên.
   - Pump hoạt động trong đúng khoảng `duration` giây như người dùng cài đặt, sau đó được tự động tắt.

Hiện thực tính năng

1. API ghi nhận chế độ tự động và cấu hình:

```
@IOT_mode.route('/api/mode/pump/automation', methods=["POST"])
def pump_automation():
    ...
    return ctl_pump_automation(automatic_options)
```

Trong đó `ctl_pump_automation()` sử dụng hàm `change_mode()` để lưu lại trạng thái tự động và các tùy chọn vào hệ thống.

2. Lớp giám sát và điều khiển tự động:

```
def update(self, data):
    if get_mode('pump') != 'automatic':
        return
    ...
    if float(data) < float(pump_automatic_options['threshold']):
        # Turn pump on
        ...
        time.sleep(duration)
        # Turn pump off
```

Ghi chú về cách hoạt động các thiết bị khác

Cách thức hoạt động tự động của light và servo được thiết kế và triển khai tương tự như với pump:

- Người dùng cấu hình điều kiện kích hoạt (ví dụ: ánh sáng dưới ngưỡng sẽ bật đèn, nhiệt độ vượt ngưỡng sẽ đóng mở servo).
- Hệ thống theo dõi dữ liệu cảm biến liên quan theo thời gian thực.
- Khi điều kiện được thỏa mãn, thiết bị sẽ tự động được bật hoặc điều chỉnh trạng thái tương ứng.
- Sau khi thực hiện, hệ thống cập nhật lại trạng thái và tiếp tục lắng nghe vòng lặp mới.

6.1.3 Tính năng đặt lịch điều khiển thiết bị

Mục tiêu

Tính năng đặt lịch được xây dựng nhằm giúp người dùng thiết lập sẵn các thời điểm hệ thống sẽ tự động điều khiển thiết bị mà không cần can thiệp thủ công. Điều này đặc biệt hữu ích trong các tình huống lặp lại hằng ngày hoặc định kỳ, chẳng hạn như bật đèn vào mỗi sáng sớm, mở máy bơm theo chu kỳ tưới, hoặc điều chỉnh servo vào các mốc cố định. Tính năng này góp phần nâng cao tính tự động hóa và tính cá nhân hóa trong vận hành nhà kính thông minh.

Quy trình hoạt động - Lấy ví dụ là tính năng đặt lịch cho thiết bị Light

1. Người dùng chọn đặt lịch trên giao diện web
   Tại giao diện frontend, người dùng lựa chọn thiết bị (ví dụ: `light`) và mở chức năng ”Lập lịch hoạt động”.
2. Hệ thống hiển thị danh sách các job đã được cài đặt trước đó
   Nếu chưa có job nào, danh sách sẽ hiển thị trống. Nếu đã có, danh sách hiện đầy đủ các job đang hoạt động để người dùng chỉnh sửa hoặc xoá.
3. Thêm job mới bằng cách nhấn dấu ”+”
   Khi người dùng nhấn nút ”+”, một modal sẽ hiển thị, cho phép cấu hình thông tin của job mới.
4. Chọn kiểu đặt lịch (Trigger mode)
   Người dùng có thể chọn giữa hai chế độ đặt lịch:
   - Interval: lặp lại theo khoảng thời gian (ví dụ: mỗi 1 giờ 30 phút).
   - Cron: lặp lại theo lịch cố định (ví dụ: 6 giờ sáng mỗi ngày).

5. Nhập thông tin thời gian
   Tuỳ vào kiểu trigger đã chọn, người dùng sẽ được cung cấp các trường thông tin khác nhau:
   - Với interval: các tùy chọn gồm `weeks`, `days`, `hours`, `minutes`, `seconds`.
   - Với cron: các trường như `year`, `month`, `day`, `day_of_week`, `hour`, `minute`, `second`.

   Ví dụ:
   - `interval`: `{ "hours": 1, "minutes": 30 }` → lặp mỗi 1 giờ 30 phút.
   - `cron`: `{ "hour": 6 }` → thực hiện lúc 6h sáng mỗi ngày.
6. Chọn hành động (action) và thông số
   Ví dụ với thiết bị `light`, người dùng chọn cường độ ánh sáng (`intensity`) cần đặt.
7. Thêm job mới
   Sau khi hoàn tất cấu hình, người dùng bấm "Add Job", hệ thống sẽ lưu lại job vào bộ lập lịch và thực thi theo đúng thời gian định sẵn.

Hiện thực tính năng

- Backend sử dụng APScheduler để quản lý các job chạy nền.
- API chính để tạo job là:

```
POST /api/job/add
```

- Các trường yêu cầu:

```
{
    "device_id": "light",
    "trigger": "interval" ặhoc "cron",
    "trigger_options": { ... },
    "action": "set_intensity",
    "action_options": {
        "intensity": 3
    }
}
```

- Tại backend, hàm `ctl_schedule_light_add_job()` sẽ:
  - Xác thực dữ liệu.
  - Thêm job mới vào APScheduler.
  - Gọi đến `ctl_adafruit_light()` để gửi giá trị điều khiển lên Adafruit IO khi đến thời điểm chạy.
  - Lưu thông tin job vào mongoDB thông qua hàm thuộc module Model - `model_schedule_insert_job()`.

Cách phát triển tính năng đặt lịch cho các thiết bị pump và servo cũng được thực hiện tương tự, chỉ khác ở action và thông số cấu hình (ví dụ: độ mở của servo, thời gian bật pump).

6.1.4 Tính năng hiển thị dữ liệu cảm biến dưới dạng đồ thị

Mục tiêu

Tính năng này cho phép người dùng theo dõi dữ liệu lịch sử từ các cảm biến môi trường (độ ẩm đất, nhiệt độ, ánh sáng) một cách trực quan và dễ hiểu thông qua các biểu đồ dạng tuyến tính. Việc trực quan hóa dữ liệu giúp người dùng dễ dàng đánh giá điều kiện sinh trưởng của cây trồng trong các khung thời gian khác nhau, đồng thời hỗ trợ phân tích để đưa ra các quyết định điều chỉnh hệ thống.

Quy trình hoạt động

1. Người dùng truy cập Dashboard
   Tại giao diện người dùng, khi người dùng điều hướng đến trang thống kê (dashboard), hệ thống sẽ hiển thị ba biểu đồ tương ứng với ba loại cảm biến:
   - Cảm biến ánh sáng (light-sensor)
   - Cảm biến nhiệt độ (temperature-sensor)
   - Cảm biến độ ẩm đất (humid-sensor)
2. Hiển thị mặc định theo ngày hiện tại
   Khi trang dashboard được tải, hệ thống sẽ mặc định lấy dữ liệu của ngày hiện tại để hiển thị biểu đồ cho từng loại cảm biến. Các giá trị được vẽ trên biểu đồ tuyến tính theo trục thời gian (giờ:phút:giây) trên trục hoành và giá trị cảm biến trên trục tung.
3. Chọn ngày xem dữ liệu với Date Picker
   Người dùng có thể chọn một ngày cụ thể khác bằng cách sử dụng lịch (date picker) ở khung bên phải. Khi ngày được thay đổi, hệ thống sẽ thực hiện gọi các API tương ứng để lấy dữ liệu cảm biến của ngày đã chọn, và cập nhật lại biểu đồ theo dữ liệu mới.

Hiện thực tính năng

- Frontend (React)
  Thành phần `Statistics.jsx` là giao diện chính cho dashboard thống kê. Dữ liệu được phân tách và quản lý bằng React `useState()` cho từng loại cảm biến (soilMoisture, temperature, light). Việc gọi API thực hiện thông qua thư viện `axios`.
- API gọi dữ liệu cảm biến
  Các endpoint được gọi có dạng:

```
/api/sensor_data/<sensor_type>?year=YYYY&month=MM&date=DD

```

  Trong đó <sensor_type> là humid, temperature, hoặc light.
- Cập nhật biểu đồ khi thay đổi ngày
  Thành phần `Calendar` được sử dụng để thay đổi `selectedDate`. Khi người dùng chọn ngày mới, `useEffect()` sẽ tự động kích hoạt các truy vấn dữ liệu tương ứng cho từng cảm biến và cập nhật lại dữ liệu trục hoành (`xAxis`) và trục tung (`yAxis`) của biểu đồ.
- Biểu đồ riêng biệt Ba biểu đồ được đóng gói trong ba component:
  - `SoilMoistureCard`
  - `TemperatureCard`
  - `LightCard`

  Mỗi component nhận dữ liệu qua props `xAxis`, `yAxis`, và `latest`, sau đó hiển thị biểu đồ trực quan cho người dùng.

6.1.5 Tính năng điều khiển thiết bị thông qua AI Chatbot

Mục tiêu

Phát triển hệ thống AI Chatbot thông minh có khả năng điều khiển các thiết bị IoT trong môi trường smart farm thông qua giao tiếp tự nhiên. Hệ thống cho phép người dùng tương tác bằng ngôn ngữ tự nhiên để điều khiển các thiết bị như máy bơm, đèn LED, và servo motor mà không cần phải hiểu rõ về lệnh kỹ thuật phức tạp.

Quy trình hoạt động

1. Tiếp nhận yêu cầu từ người dùng: Người dùng nhập lệnh bằng ngôn ngữ tự nhiên qua giao diện người dùng (UI).
2. Phân tích ngôn ngữ tự nhiên: AI Chatbot sẽ phân tích câu lệnh người dùng để trích xuất ý định (intent) và thông tin liên quan (ví dụ: thiết bị, trạng thái).
3. Gửi yêu cầu điều khiển: Dữ liệu được gửi đến máy chủ điều khiển thiết bị (Control Server).
4. Thực thi lệnh: Máy chủ điều khiển thực hiện yêu cầu tương ứng với các thiết bị IoT qua giao thức nội bộ.
5. Phản hồi kết quả: Hệ thống gửi phản hồi về trạng thái hoặc kết quả thực thi về UI để hiển thị cho người dùng.

Hiện thực tính năng

1. Xử lý Function Calling với AI Model
   Đoạn code dưới đây thực hiện việc gửi yêu cầu của người dùng đến model Gemini 2.0 Flash cùng với danh sách các function có thể gọi. Model sẽ phân tích yêu cầu và quyết định có cần gọi function nào không. Nếu có, hệ thống sẽ thực thi function đó và trả về kết quả.

```
def call_model(self, state: MessagesState):
    tools = types.Tool(function_declarations=self.tools_list)
    config = types.GenerateContentConfig(tools=[tools])

    response = self.client.models.generate_content(
        model="gemini-2.0-flash",
        contents=state["messages"][-1].content,
        config=config,
    )

    # ểKim tra ếnu có function call
    if response.candidates[0].content.parts[0].function_call:
        function_call = response.candidates[0].content.parts[0].function_call
        data = self.function_calling_excution(function_call.name, function_call.args)

        return {
            "messages": state["messages"] + [AIMessage(content=json.dumps(data,
    ensure_ascii=False))]
        }

```

2. Thực thi Function Calling
   Khi AI model quyết định gọi một function, đoạn code này sẽ phân tích tên function để xác định thiết bị cần điều khiển và tên function thực tế, sau đó chuyển tiếp lệnh đến client tương ứng.

```
def function_calling_excution(self, function_name, arguments):
    client = self.listTool[function_name.split('_')[0]]
    functionName = function_name[function_name.index('_') + 1:]
    data = client.function_calling(functionName, arguments)
    return data

```

3. Giao tiếp WebSocket Real-time
   Đây là handler xử lý tin nhắn từ server WebSocket. Khi nhận được lệnh 'discover', client sẽ cập nhật danh sách công cụ có thể sử dụng. Đối với các message khác, nó sẽ lưu trữ dữ liệu phản hồi và kích hoạt event để thông báo cho thread chính.

```
def on_message(self, ws, message):
    data = json.loads(message)
    if data['function'] == 'discover':
        for el in data["data"]:
            el["name"] = self.client_name + '_' + el["name"]
            el["description"] = self.description + '.' + el["description"]
            self.tools.append(el)
    else:
        self.response_data = data
        self.response_event.set()

```

4. Định nghĩa Function Schema cho Thiết bị
   Ví dụ về một Function Schema cho việc điều khiển đèn LED:

```
light_on_intensity = {
    "name": "light_on_intensity",
    "description": "Active light on by specify intensity",
    "parameters": {
        "type": "object",
        "properties": {
            "intensity": {
                "type": "string",
                "description": "The intensity of light. When user want to turn off light,
 agent set the intensity to 0",
                'enum': [str(i) for i in range(0, 5)],
                'default': "0",
            },
        },
        "required": ['intensity']
    },
}

```

   Đây là schema định nghĩa function cho việc điều khiển đèn LED. Schema này giúp AI model hiểu được function này có tác dụng gì, cần tham số gì và kiểu dữ liệu như thế nào. Điều này rất quan trọng để model có thể gọi đúng function với đúng tham số.

5. WebSocket Server Handler

```
@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    while True:
        data = await websocket.receive_text()
```

```
        request = json.loads(data)

        data = handle_request(request)

        response = {
            "function": request['function'],
            "data": data
        }

        await websocket.send_text(json.dumps(response))

```

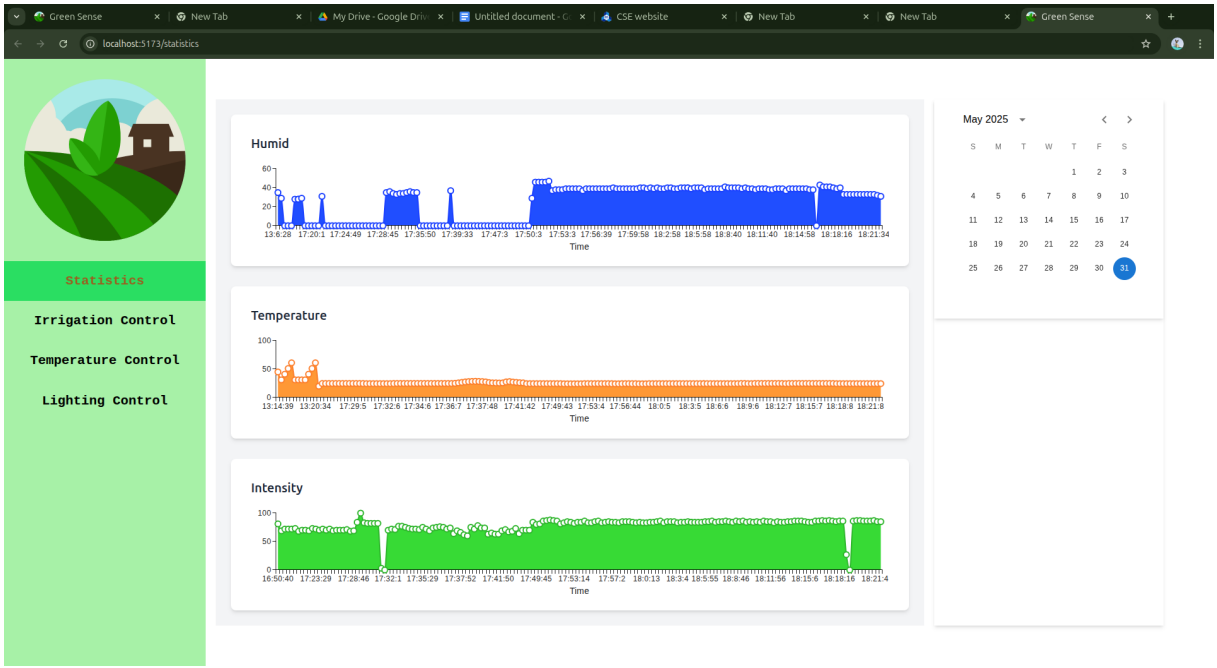
Đây là endpoint WebSocket server sử dụng FastAPI. Nó liên tục lắng nghe các yêu cầu từ client, xử lý chúng thông qua hàm `handle_request()`, và gửi phản hồi trở lại client. Điều này tạo ra kết nối real-time giữa chatbot và server điều khiển thiết bị.

6. Xử lý Yêu cầu Điều khiển Thiết bị
   Hàm `handle_request` phân loại và xử lý các yêu cầu. Nếu là yêu cầu 'discover', nó trả về danh sách các function có thể gọi. Nếu là yêu cầu điều khiển thiết bị, nó sẽ sử dụng reflection để gọi method tương ứng trong class `controlDevice` với các tham số được truyền vào.

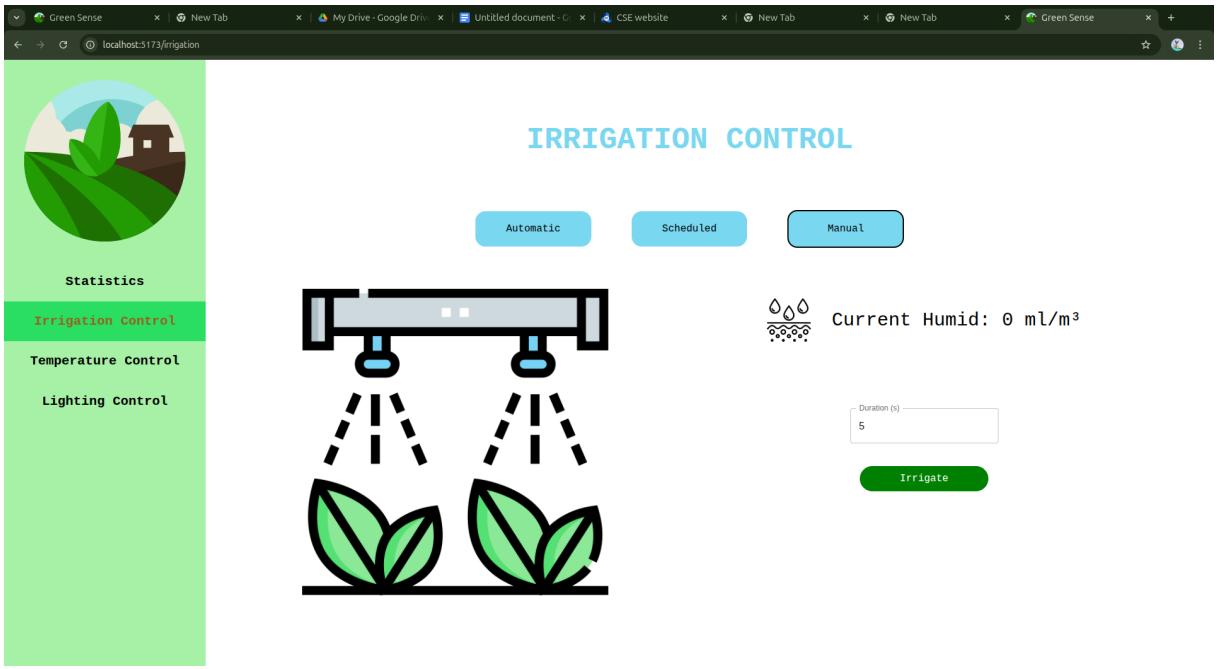
```
def handle_request(request):
    if request['function'] == 'discover':
        return [pump_on_duration, light_on_intensity, servo_on_angle]
    else:
        f = getattr(tools, request['function'])
        arguments = request.get('arguments') if request.get('arguments') != None else {}
        return f(**arguments)

```

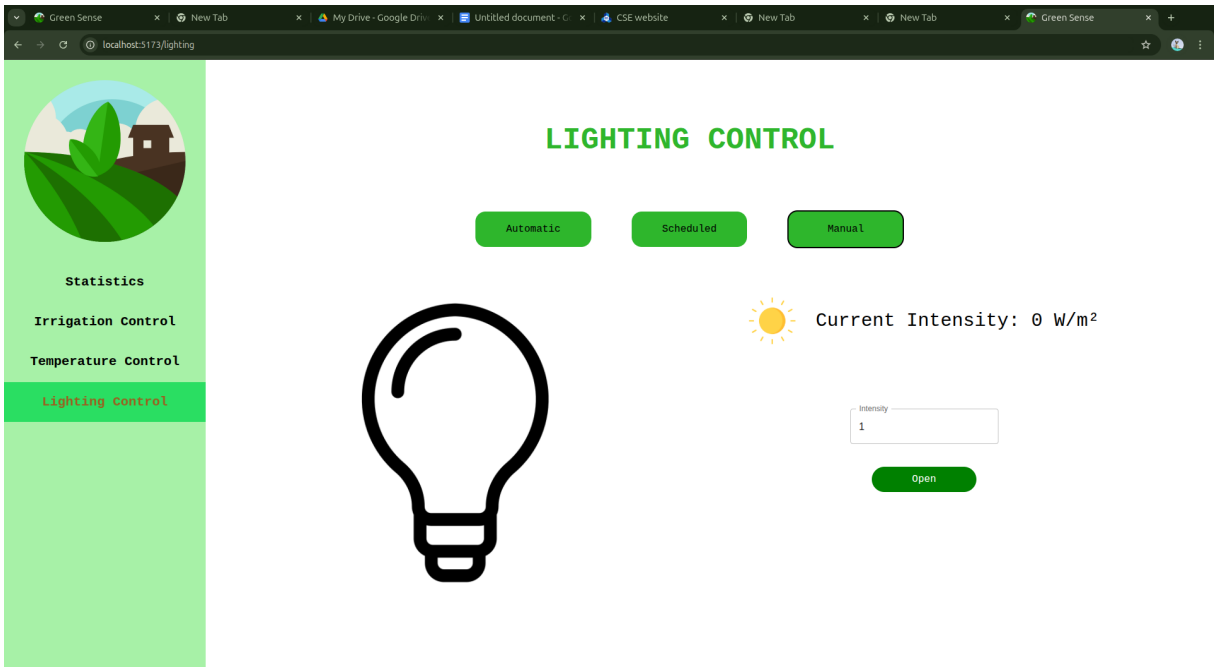
6.2 Giao diện hệ thống



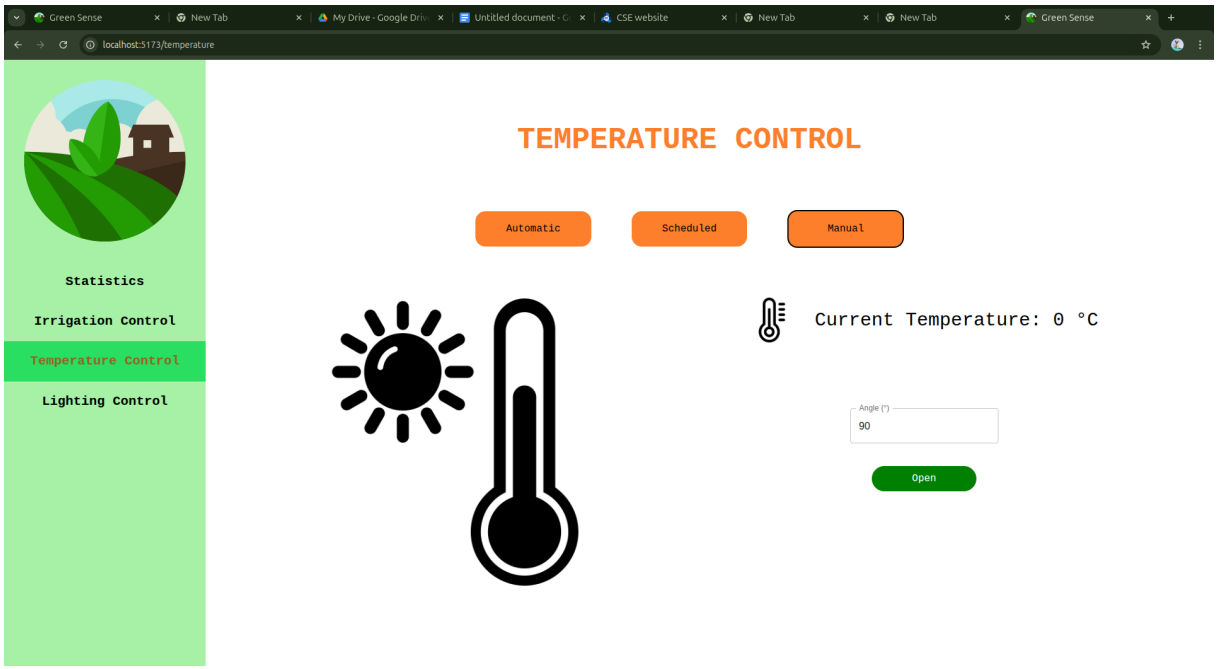
Hình 4: Màn hình tính năng Dashboard



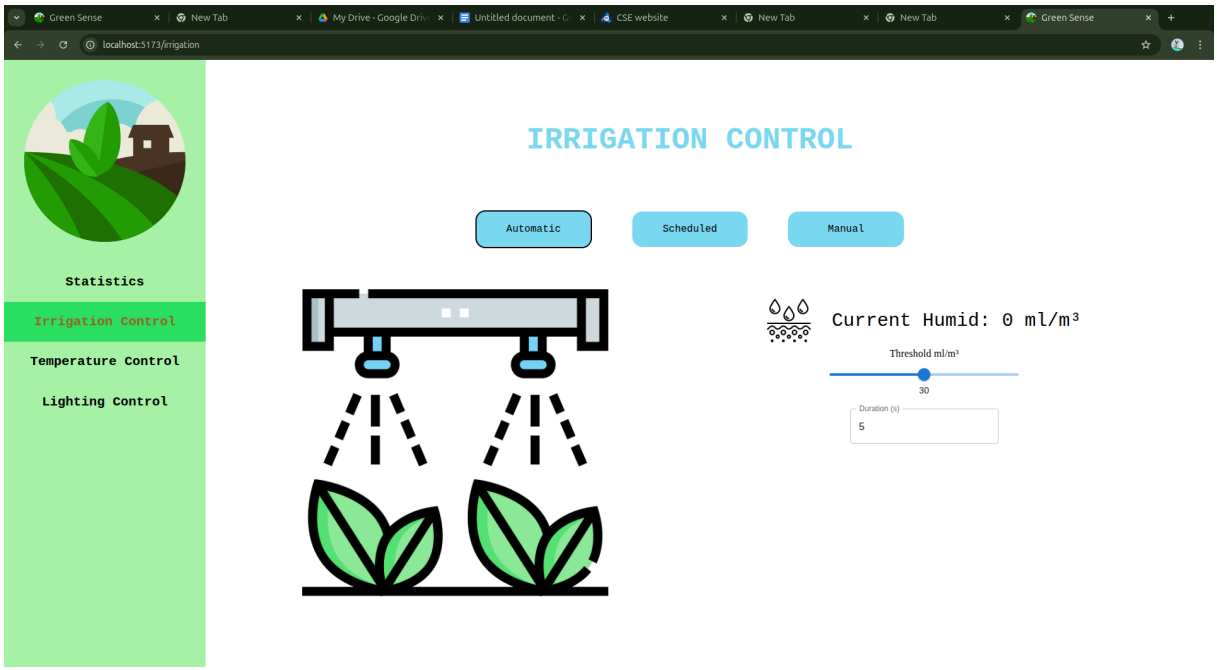
Hình 5: Màn hình điều khiển máy bơm nước thủ công



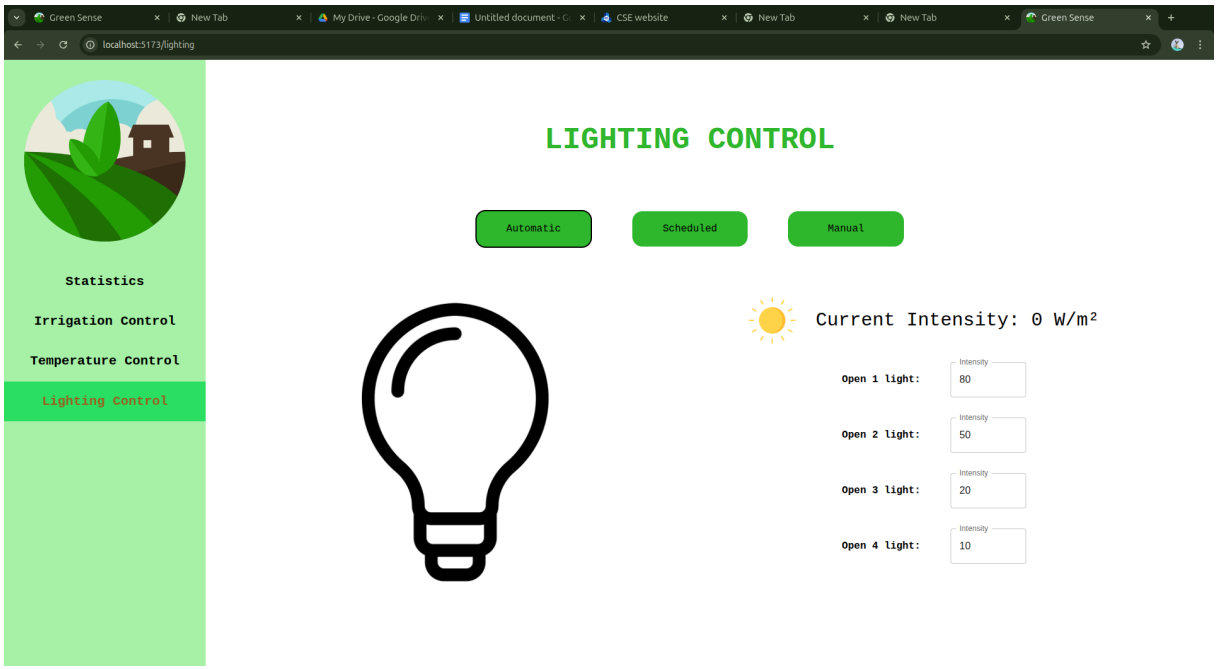
Hình 6: Màn hình điều khiển đèn thủ công



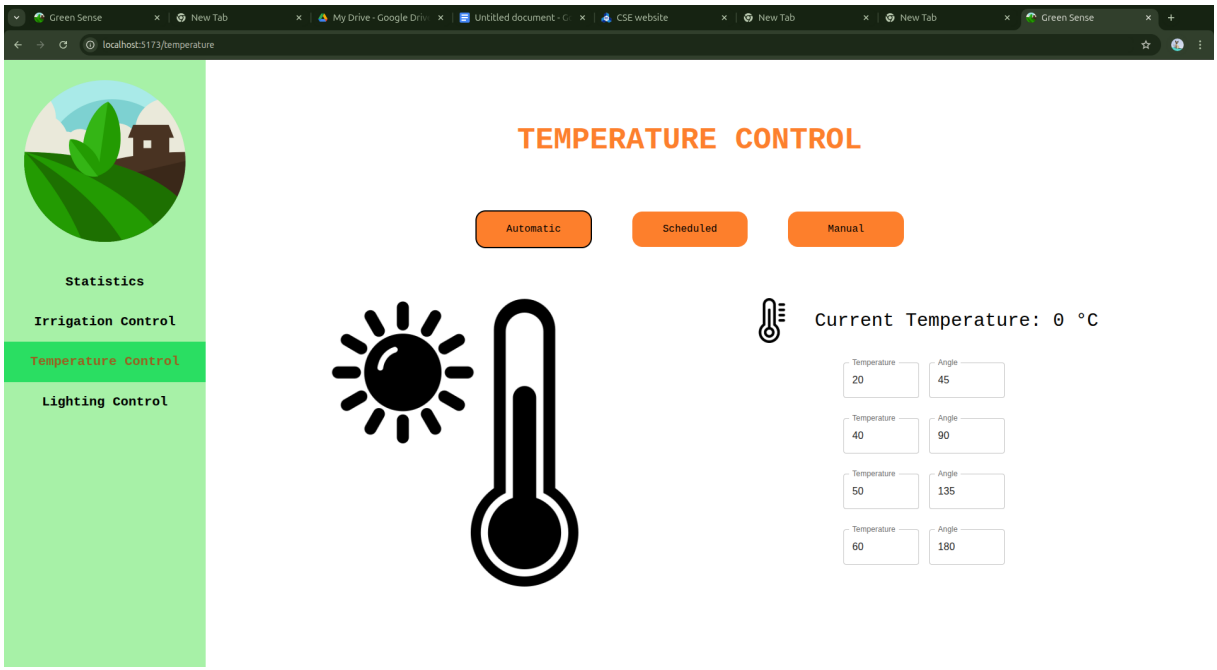
Hình 7: Màn hình điều khiển màn che (động cơ servo) thủ công



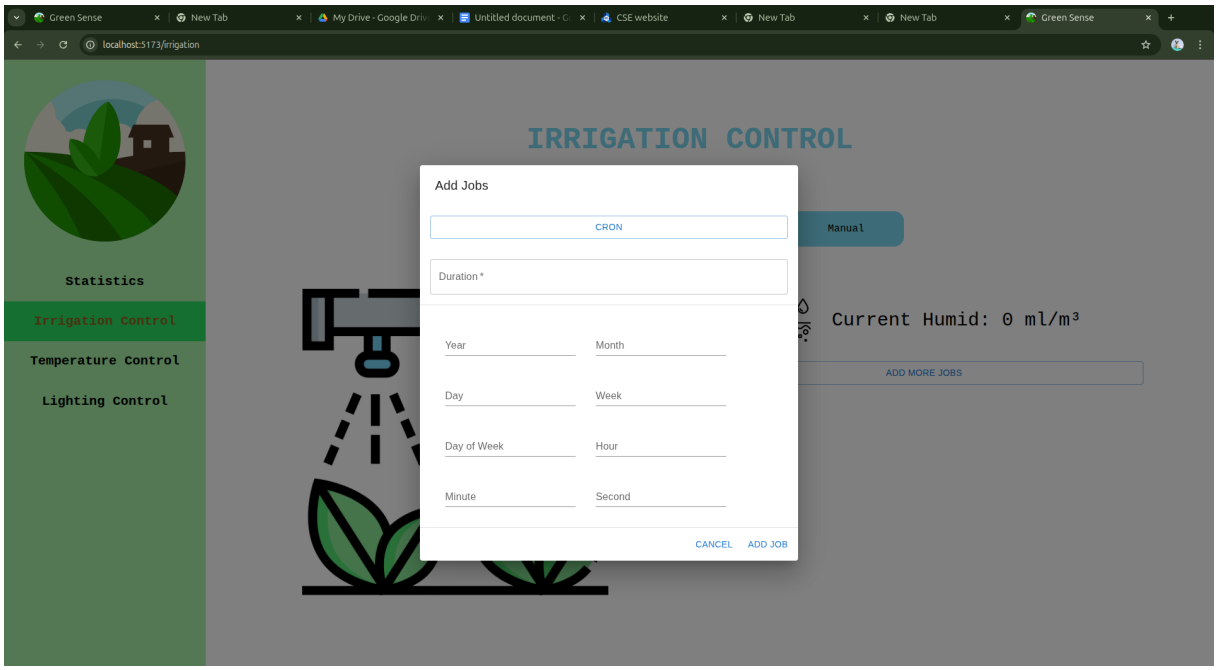
Hình 8: Màn hình cài đặt điều khiển máy bơm nước tự động



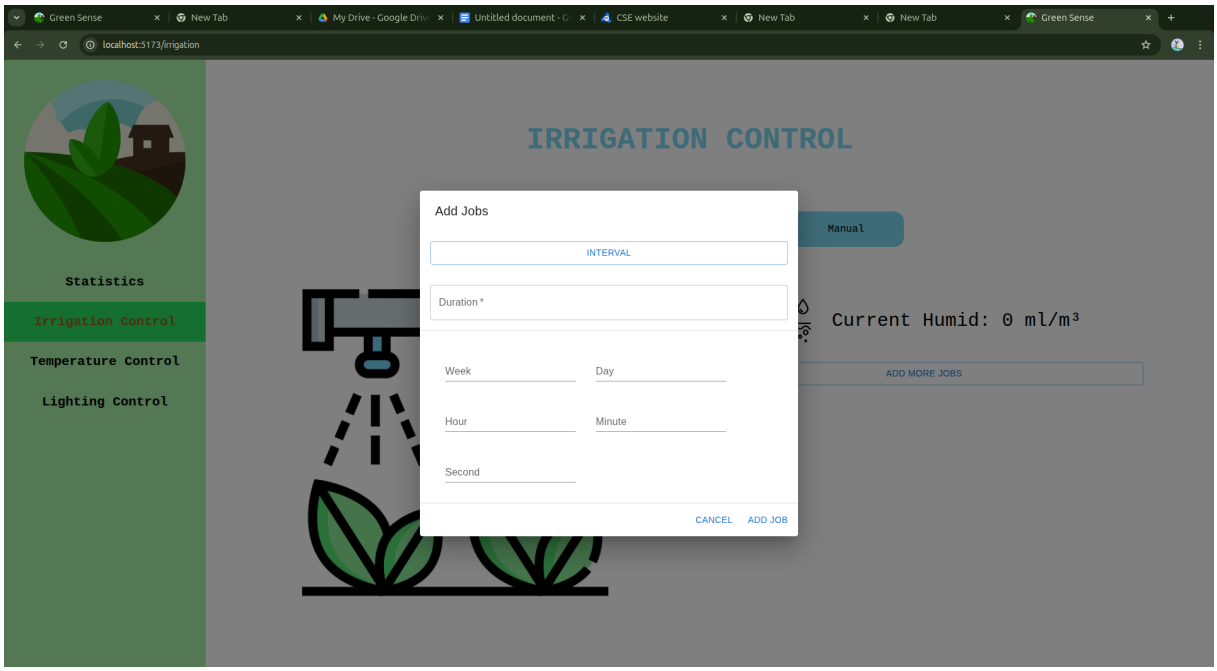
Hình 9: Màn hình cài đặt điều khiển đèn tự động



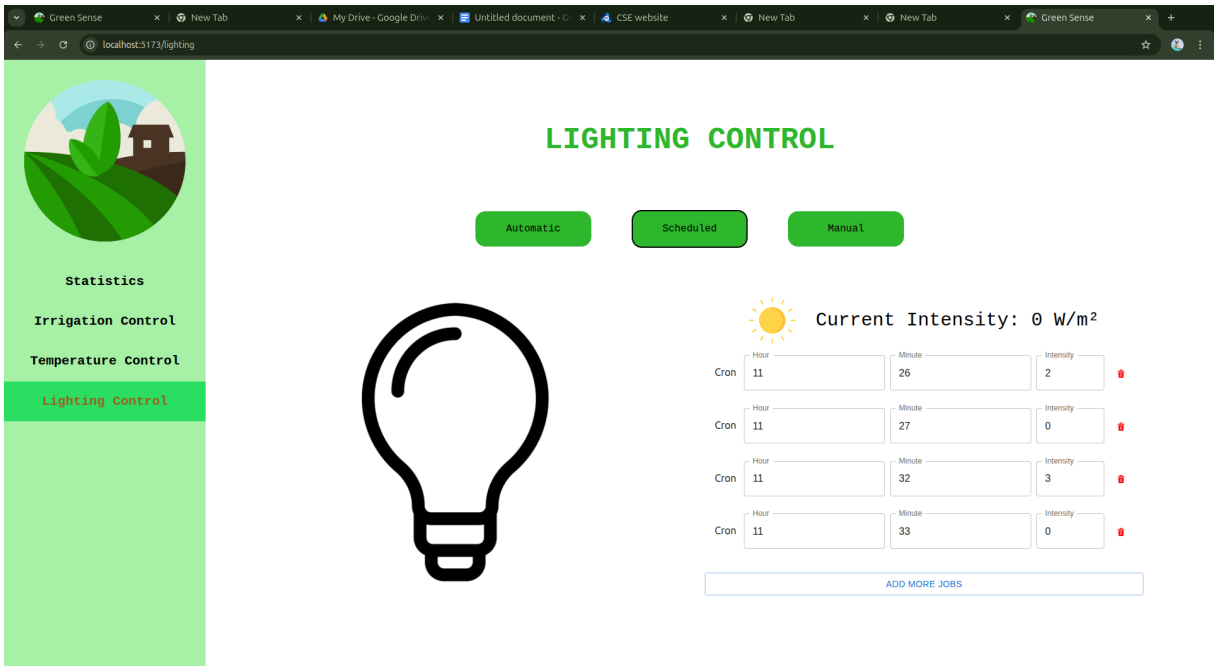
Hình 10: Màn hình cài đặt điều khiển màn che (động cơ servo) tự động



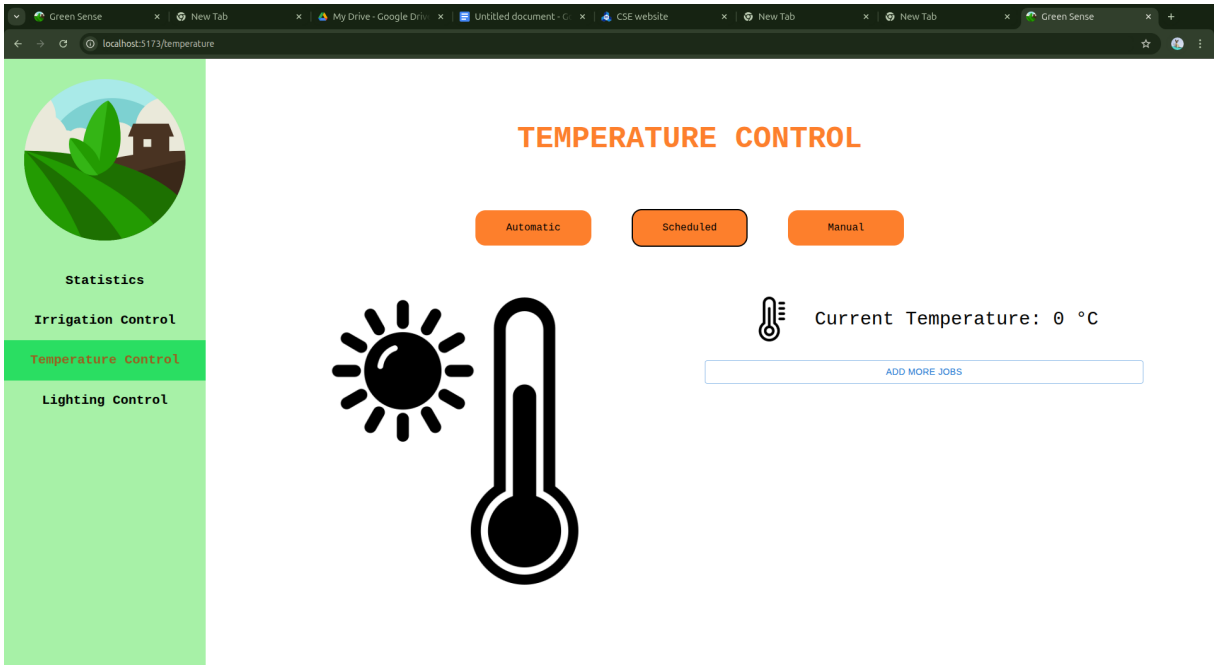
Hình 11: Màn hình đặt lịch hoạt động cho máy bơm nước - theo chế độ cron



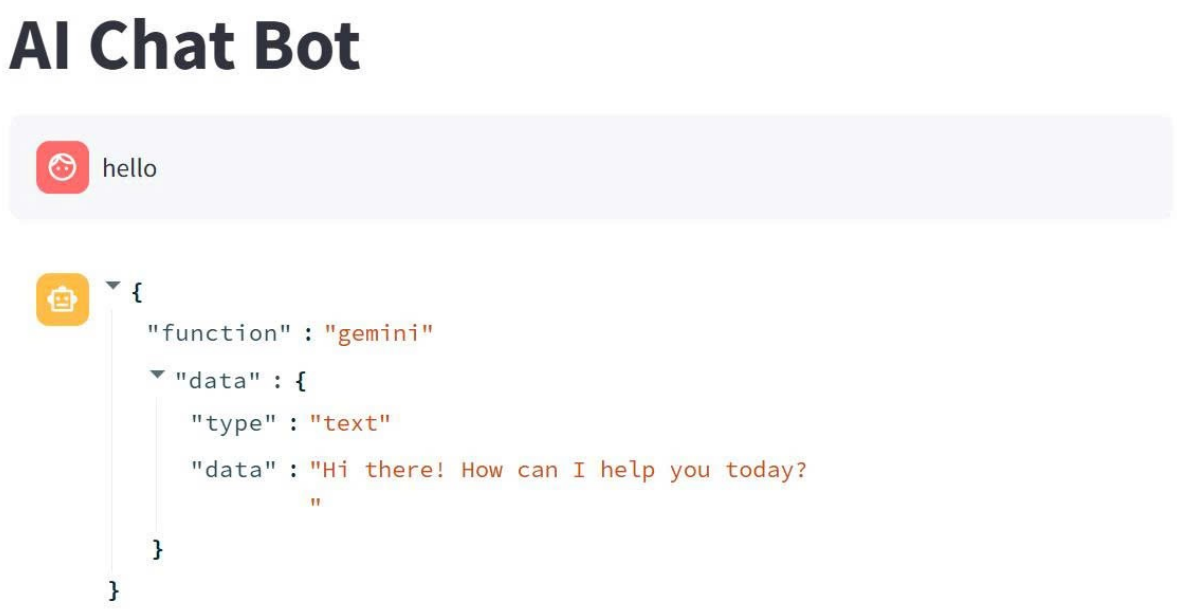
Hình 12: Màn hình đặt lịch hoạt động cho máy bơm nước - theo chế độ interval



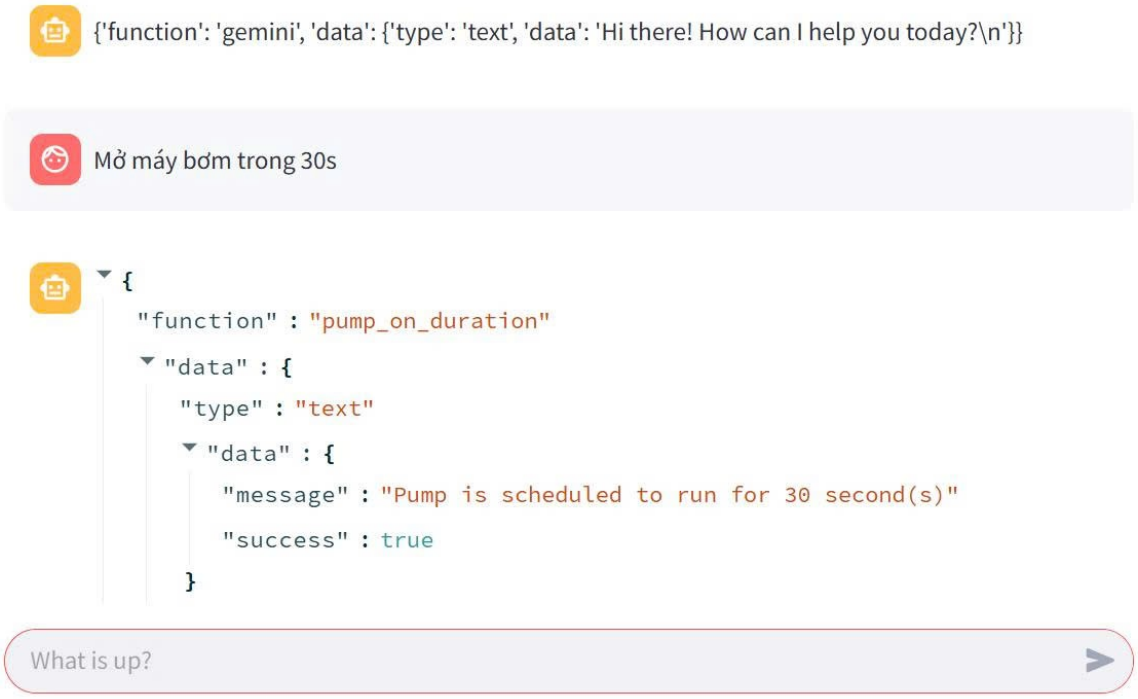
Hình 13: Màn hình đặt lịch hoạt động cho đèn



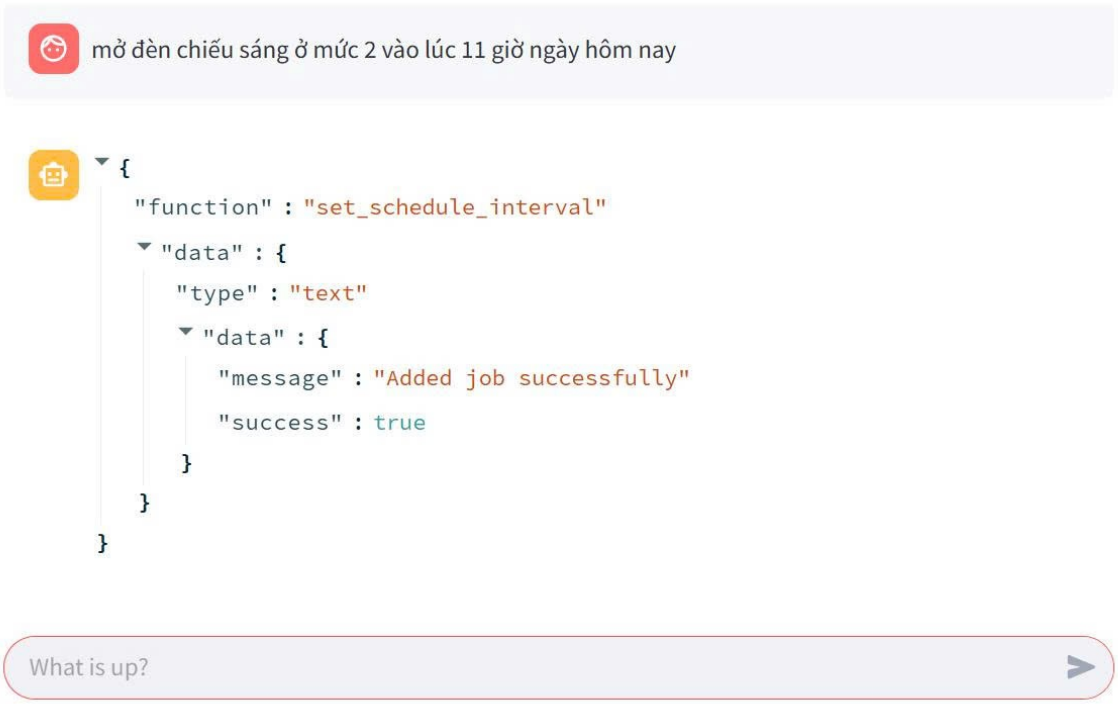
Hình 14: Màn hình đặt lịch hoạt động cho màn che (động cơ servo)



Hình 15: Màn hình khởi động AI Chatbot



Hình 16: Điều khiển máy bơm (pump) bằng ngôn ngữ tự nhiên



Hình 17: Đặt lịch mở đèn bằng ngôn ngữ tự nhiên

Hình 18: Lấy dữ liệu của cảm biến độ ẩm đất bằng ngôn ngữ tự nhiên

7 Kết luận và hướng phát triển

7.1 Kết luận chung

Dự án phát triển hệ thống giám sát và điều khiển nhà kính thông minh đã hoàn thành các mục tiêu chính đề ra. Hệ thống nguyên mẫu được xây dựng thành công dựa trên sự kết hợp của các công nghệ hiện đại như Internet of Things (IoT), trí tuệ nhân tạo (AI) và giao thức MQTT thông qua nền tảng Adafruit IO. Việc áp dụng các công nghệ này giúp hệ thống có khả năng thu thập dữ liệu thời gian thực và điều khiển các thiết bị trong nhà kính một cách hiệu quả và chính xác.

Giao diện người dùng được thiết kế thân thiện và trực quan, cho phép người dùng dễ dàng giám sát các thông số môi trường như nhiệt độ, độ ẩm, ánh sáng cũng như thực hiện các thao tác điều khiển từ xa thông qua website. Đặc biệt, việc tích hợp tính năng điều khiển bằng ngôn ngữ tự nhiên thông qua AI đã nâng cao trải nghiệm người dùng, giúp thao tác trở nên đơn giản và tiện lợi hơn.

Kiến trúc phần mềm được xây dựng theo mô hình MVC và MCP, đảm bảo tính linh hoạt và mở rộng cho hệ thống trong tương lai. Sự phối hợp nhịp nhàng giữa các thành phần frontend, backend, thiết bị IoT và module điều khiển tự động đã giúp hệ thống vận hành ổn định, giảm thiểu sự can thiệp thủ công và nâng cao hiệu suất quản lý nhà kính.

Kết quả thử nghiệm thực tế cho thấy hệ thống hoạt động ổn định và đáp ứng tốt các yêu cầu kỹ thuật, góp phần giảm tải công việc quản lý thủ công, tối ưu hóa điều kiện khí hậu cho cây trồng, từ đó giúp nâng cao năng suất và chất lượng sản phẩm nông nghiệp trong môi trường nhà kính.

7.2 Hướng phát triển

Hiện tại, hệ thống chỉ hỗ trợ hoạt động với một bộ thiết bị duy nhất, sử dụng các feed Adafruit IO đã được cấu hình sẵn. Trong tương lai, dự án sẽ được mở rộng để hỗ trợ đồng thời nhiều bộ thiết bị, giúp tăng khả năng quản lý và giám sát cho các nhà kính có quy mô lớn hoặc nhiều khu vực khác nhau.

Hệ thống cần được bổ sung các tính năng cảnh báo và thông báo tự động đến điện thoại của người dùng, có thể qua tin nhắn SMS hoặc ứng dụng di động. Điều này nhằm kịp thời cảnh báo khi xảy ra sự cố hoặc các điều kiện bất thường trong nhà kính, giúp người quản lý nhanh chóng có biện pháp xử lý phù hợp.

Mở rộng tích hợp thêm các loại cảm biến đa dạng hơn như cảm biến CO2, cảm biến đất, hoặc camera giám sát để cung cấp dữ liệu toàn diện hơn về môi trường nhà kính, từ đó nâng cao hiệu quả điều khiển và chăm sóc cây trồng.

Nâng cao khả năng phân tích dữ liệu và dự báo thông minh dựa trên Machine Learning để đưa ra các khuyến nghị tối ưu hóa điều kiện trồng trọt, giúp tăng năng suất và chất lượng sản phẩm.

Phát triển ứng dụng di động với giao diện thân thiện, cho phép người dùng dễ dàng giám sát và điều khiển nhà kính mọi lúc mọi nơi, đồng thời tích hợp tính năng điều khiển giọng nói dựa trên AI nhằm tăng tính tiện lợi.

Tối ưu hóa giao thức truyền thông và bảo mật hệ thống để đảm bảo dữ liệu được truyền tải an toàn, tránh nguy cơ tấn công hoặc truy cập trái phép.

8 Phụ lục

Mã nguồn hệ thống:

- Frontend: https://github.com/HK242-DADN-GreenSense/GreenSense-FE
- Backend: https://github.com/HK242-DADN-GreenSense/GreenSense-BE
- AI Chatbot: https://github.com/HK242-DADN-GreenSense/AI_Chatbox
- IoT: https://github.com/HK242-DADN-GreenSense/Embedded-ESP32
- Report: https://github.com/HK242-DADN-GreenSense/report

Video Demo:

- Video link: https://drive.google.com/file/d/1ksycE5RTZAWw1gkSqknD27hsfJCQHASX/views