

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



THỰC TẬP ĐỒ ÁN MÔN HỌC ĐA NGÀNH
HƯỚNG CÔNG NGHỆ PHẦN MỀM - CO3109

BÁO CÁO ĐỒ ÁN MÔN HỌC

*Đề tài: Xây dựng hệ thống giám sát
và điều khiển đèn chiếu sáng*

Lớp L03 - HK222

Giảng viên hướng dẫn Đỗ Thanh Thái

Nhóm Wednesday

Danh sách sinh viên thực hiện

1. Huỳnh Ngọc Như - 2010495
2. Lê Văn Vy - 2010805
3. Nguyễn Trí Hiếu - 2013153
4. Nguyễn Ngọc Trí - 2012286

LỜI MỞ ĐẦU

Lời đầu tiên nhóm Wednesday - lớp L03 xin gửi lời cảm ơn đến giảng viên hướng dẫn Dõ Thanh Thái vì những trao đổi kiến thức và những góp ý về đồ án môn học của thầy để từ đó nhóm có đủ nền tảng để nghiên cứu, phân tích, hiện thực và hoàn thành đề tài được giao.

Để hoàn thành tốt đề tài này, trước hết nhóm chúng em đã khảo sát, tìm hiểu thực tiễn để có thể thống nhất những ý tưởng ban đầu cho đề tài. Sau đó nhóm đã áp dụng những kiến thức đã được học về IoT, lập trình web, công nghệ phần mềm để hiện thực hệ thống. Ngoài ra nhóm còn tìm hiểu kỹ thêm các vấn đề thông qua những tài liệu tham khảo hay các diễn đàn trên mạng, đồng thời nhóm cũng đã kiểm thử nhiều lần hệ thống để đảm bảo hệ thống đạt hiệu quả cao.

Trong quá trình thực hiện đề tài, nhóm cũng gặp phải nhiều khó khăn như mất nhiều thời gian để giải quyết các vấn đề khó, quá trình mượn thiết bị xảy ra các sự cố ngoài ý muốn, hay giữa các thành viên có sự khác biệt về thời gian học tập, trong quá trình thực hiện đề tài có một vài thời điểm vẫn chưa thật sự thống nhất chung về mặt ý tưởng, hiện thực giải pháp... Tuy nhiên nhóm cũng đã cơ bản giải quyết được những khó khăn trên và cho ra kết quả đạt được như mong muốn.

Trong quá trình thực hiện đề tài chắc chắn nhóm sẽ không thể tránh được những thiếu sót. Chúng em mong rằng đề tài này sẽ được nhận những góp ý từ giảng viên hướng dẫn để nhóm có thể khắc phục, chỉnh sửa để đề tài hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

PHÂN CÔNG CÔNG VIỆC

TUẦN 1 (13 - 18/2/2023)

Nội dung công việc - Nhiệm vụ chung: Chọn và thống nhất đề tài, tìm hiểu các lý thuyết liên quan đến đề tài, bao gồm các thiết bị phần cứng và các công nghệ phần mềm.

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	- Xác định công nghệ phần mềm - Tìm hiểu giới thiệu tổng quan về Yolo:Bit	100%
2	Lê Văn Vỹ	- Xác định công nghệ phần mềm - Tìm hiểu các thành phần thiết bị	100%
3	Nguyễn Trí Hiếu	- Xác định công nghệ phần mềm - Tìm hiểu các thành phần thiết bị	100%
4	Nguyễn Ngọc Trí	- Tìm hiểu các thành phần thiết bị - Tìm hiểu về server, gateway và Ohstem	100%

TUẦN 2 (20 - 26/2/2023)

Nội dung công việc: Thảo luận, thống nhất các tính năng của hệ thống. Vẽ use-case diagram, đặc tả use-case. Khảo sát các thiết bị phần cứng, tìm hiểu IoT Gateway.

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	- Phân chia công việc - Vẽ use-case diagram, đặc tả use-case scenario	100%
2	Lê Văn Vỹ	- Vẽ use-case diagram - Đặc tả usecase scenario	100%
3	Nguyễn Trí Hiếu	- Phân tích các Stakeholders - Xác định Functional và Non-functional hệ thống	100%
4	Nguyễn Ngọc Trí	- Khảo sát chi tiết các thành phần thiết bị - Setup code gateway và Ohstem	100%

TUẦN 3 (27/2 - 5/3/2023)

Nội dung công việc: Thiết kế UI/UX design (Web app), hiện thực và tích hợp Iot Gateway bằng Python

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	- Thiết kế UI/UX phần Context Setup, Manual Setting	100%
2	Lê Văn Vy	- Thiết kế UI/UX phần Home - Tổng hợp thiết kế Figma	100%
3	Nguyễn Trí Hiếu	- Thiết kế UI/UX phần Login, Signup, User Detail	100%
4	Nguyễn Ngọc Trí	- Thiết kế UI/UX phần Statistics - Hiện thực gateway Python và test thiết bị	100%

TUẦN 4 (6 - 12/3/2023)

Do tuần này tất cả các thành viên trong nhóm đều có lịch thi giữa kì nên nhóm chỉ hiện thực vẽ activity diagram, còn lại tạm ngưng các hoạt động khác để tập trung ôn tập kiểm tra.

TUẦN 5 (13 - 19/3/2023)

Nội dung công việc: Hoàn thiện UI/UX design (Web app), hiện thực code giao diện, kiểm tra kết nối từ các thiết bị phần cứng tới server Adafruit, lên ý tưởng xây dựng và kết nối database.

Nhiệm vụ chung: Hoàn thiện thiết kế UI/UX (Web app).

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	- Lên ý tưởng xây dựng Database - Code hiện thực Frontend	100%
2	Lê Văn Vy	- Setup và hiện thực code Frontend	100%
3	Nguyễn Trí Hiếu	- Hiện thực code Frontend	100%
4	Nguyễn Ngọc Trí	- Kiểm tra kết nối dữ liệu lên Adafruit - Tìm hiểu cách kết nối dữ liệu giữa Adafruit IO server với Backend của web nhóm đang hiện thực	100%

TUẦN 6 (20 - 26/3/2023)

Nội dung công việc: Tiếp tục chỉnh sửa, hoàn thiện UI/UX phù hợp, tiếp tục hiện thực phần Frontend Web, bước đầu thiết kế Database Schema để có góc nhìn sơ bộ về những data cần thiết liên quan tới hệ thống, thiết kế Userflow.

Nhiệm vụ chung: Thống nhất tối ưu thiết kế UI/UX (Web app).

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	- Code hiện thực Frontend - Thiết kế bước đầu Database Schema	100%
2	Lê Văn Vỹ	- Hiện thực code Frontend - Thiết kế bước đầu Database Schema	100%
3	Nguyễn Trí Hiếu	- Hiện thực code Frontend - Thiết kế Userflow cho toàn bộ hệ thống Web	100%
4	Nguyễn Ngọc Trí	- Chính sửa, cải thiện code hiện thực phần cứng sao cho đồng bộ với web nhóm đang hiện thực	100%

TUẦN 7 (27/3 - 2/4/2023)

Nội dung công việc: Tối ưu UI/UX, debug các tính năng nhận đọc dữ liệu từ web, chỉnh sửa database schema, bắt đầu hiện thực phần backend.

Nhiệm vụ chung: Tối ưu UI/UX, chỉnh sửa và thống nhất Database schema.

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	- Tổ chức phần code Backend - Chính sửa giao diện Frontend	100%
2	Lê Văn Vỹ	- Chính sửa giao diện Frontend - Test các API từ server Adafruit IO (từ phía Web)	100%
3	Nguyễn Trí Hiếu	- Chính sửa giao diện Frontend - Code hiện thực Backend	100%
4	Nguyễn Ngọc Trí	- Test các API (từ phía Server Adafruit IO) - Sửa lỗi từ hệ thống phần cứng	100%

TUẦN 8 (3/4 - 9/4/2023)

Nội dung công việc: Thiết kế UI/UX cho Mobile App, hoàn thiện Frontend và Backend của Web, kiểm tra và chỉnh sửa các tính năng kết nối và xử lý dữ liệu.

Nhiệm vụ chung: Lên ý tưởng thiết kế UI/UX cho Mobile App, kiểm tra và chỉnh sửa các tính năng kết nối và xử lý dữ liệu.

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	<ul style="list-style-type: none"> - Thiết kế UI/UX Mobile App phần Context Setup, Manual Setting - Code hoàn thiện Frontend và Backend phần Web 	100%
2	Lê Văn Vỹ	<ul style="list-style-type: none"> - Thiết kế UI/UX Mobile App phần Home - Code hoàn thiện Frontend và Backend phần Web 	100%
3	Nguyễn Trí Hiếu	<ul style="list-style-type: none"> - Thiết kế UI/UX Mobile App phần Login, Signup, User Detail - Code hoàn thiện Frontend và Backend phần Web 	100%
4	Nguyễn Ngọc Trí	<ul style="list-style-type: none"> - Thiết kế UI/UX Mobile App phần Statistics - Hoàn thiện hệ thống phần cứng 	100%

TUẦN 9 (10/4 - 16/4/2023)

Nội dung công việc: Chốt lại phần cứng kết nối từ Yolobit qua Gateway đến Adafruit IO, kiểm tra tổng hợp các tính năng của cả hệ thống, code hiện thực phần mobile app.

Nhiệm vụ chung: Test tổng hợp các tính năng của cả hệ thống (gồm các thành phần Node, Gateway, Adafruit IO Server, Backend MongoDB và Web hiện thực của nhóm).

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	<ul style="list-style-type: none"> - Hiện thực Frontend mobile app. 	100%
2	Lê Văn Vỹ	<ul style="list-style-type: none"> - Hiện thực Frontend mobile app. 	100%
3	Nguyễn Trí Hiếu	<ul style="list-style-type: none"> - Hiện thực Frontend mobile app. 	100%
4	Nguyễn Ngọc Trí	<ul style="list-style-type: none"> - Tổng hợp code Yolobit, Gateway Python - Cố định giao diện, luồng hoạt động của phần Adafruit IO 	100%

TUẦN 10 (17/4 - 23/4/2023)

Nội dung công việc: Hoàn thiện phần mobile app, tổng hợp báo cáo.

STT	Tên thành viên	Phân công công việc	Đánh giá
1	Huỳnh Ngọc Như	- Hoàn thiện phần hiện thực Mobile app.	100%
2	Lê Văn Vỹ	- Hoàn thiện phần hiện thực Mobile app.	100%
3	Nguyễn Trí Hiếu	- Hoàn thiện phần hiện thực Mobile app.	100%
4	Nguyễn Ngọc Trí	- Tổng hợp báo cáo.	100%

TUẦN 11 (24/4 - 27/4/2023)

Nội dung công việc: Kiểm tra lần cuối toàn bộ hệ thống, hoàn thiện báo cáo và hoàn thành đề tài đồ án với giảng viên hướng dẫn.

Mục lục

1 Giới thiệu đề tài	1
1.1 Tên đề tài	1
1.2 Bối cảnh đề tài	1
1.3 Các ý tưởng ban đầu của đề tài	1
1.4 Cơ sở hiện thực đề tài	2
2 Phân tích yêu cầu	3
2.1 Các bên liên quan đến dự án (Stakeholder)	3
2.2 Yêu cầu chức năng	4
2.3 Yêu cầu phi chức năng	5
3 Danh sách thiết bị phần cứng	7
3.1 Cảm biến nhiệt độ độ ẩm DHT20	7
3.2 Cảm biến ánh sáng	9
3.3 Cảm biến hồng ngoại	11
3.4 Mạch Yolo:Bit	13
3.5 Grove shield - Mạch mở rộng cho Yolo:Bit	15
3.6 Remote điều khiển từ xa và mắt nhận hồng ngoại	16
3.7 Quạt mini	17
3.8 Màn hình LCD 16x2	18
3.9 Dây nối tín hiệu	20
4 Mô hình kiến trúc hệ thống	21
5 Use-case diagram	23
5.1 Biểu đồ Usecase cho toàn hệ thống	23
5.2 Biểu đồ Usecase cho các tính năng chính	24
5.2.1 Hiển thị dữ liệu thời gian thực	24
5.2.2 Hiển thị thống kê hoạt động	27
5.2.3 Hiển thị thông báo	29
5.3 Biểu đồ Usecase cho tính năng "Thiết lập ngữ cảnh"	31
6 Thiết kế UI/UX	33
6.1 Login & Signup	33
6.1.1 Web App	33
6.1.2 Mobile App	34
6.2 Homepage	35
6.2.1 Web App	35

6.2.2	Mobile App	36
6.3	User Detail	37
6.3.1	Web App	37
6.3.2	Mobile App	37
6.4	Manual Control	38
6.4.1	Web App	38
6.4.2	Mobile App	38
6.5	Statistics	39
6.5.1	Web App	39
6.5.2	Mobile App	40
6.6	Context Setup	40
6.6.1	Web App	40
6.6.2	Mobile App	43
6.7	User Flow	44
7	Activity Diagram: Context Setup	47
8	Thiết kế Database	49
9	Hiện thực phần cứng	50
9.1	Hiện thực Node	50
9.1.1	Sơ đồ dự kiến kết nối các thiết bị phần cứng	50
9.1.2	Hiện thực chương trình	50
9.2	Hiện thực Gateway	52
9.2.1	Kết nối với Yolo:Bit	52
9.2.2	Kết nối với Adafruit IO Server	53
9.3	Hiện thực Adafruit IO Server	54
9.3.1	Nhóm các khối điều khiển thiết bị, giám sát các trạng thái	54
9.3.2	Nhóm các khôi quan sát các thông số dữ liệu	55
10	Hiện thực Web & App - Design Pattern	57
10.1	Kiến trúc MVC (MVC pattern)	57
10.2	Observer Pattern	59
10.3	React Hooks - State Pattern	66
11	Hoàn thành sản phẩm	67
12	Đánh giá hệ thống	68
12.1	Ưu điểm	68
12.2	Nhược điểm	68

12.3 Hướng phát triển	69
13 Tổng kết đề tài	70
14 Tài liệu tham khảo	71

Danh mục hình ảnh

1	Cảm biến DHT20	7
2	Sơ đồ chân của cảm biến DHT20 theo datasheet đính kèm	7
3	Cảm biến ánh sáng	9
4	Hình ảnh quang điện trở GL5528	9
5	Sơ đồ chân của module cảm biến ánh sáng	10
6	Cảm biến hồng ngoại PIR	11
7	IC AS312	11
8	Sơ đồ chân của module cảm biến hồng ngoại PIR	12
9	Mạch Yolo:Bit	13
10	Các thành phần của Yolo:Bit	14
11	Sơ đồ chân của Yolo:Bit	14
12	Grove shield - Mạch mở rộng cho Yolo:Bit	15
13	Remote điều khiển từ xa và mắt nhận hồng ngoại	16
14	Quạt mini	17
15	Sơ đồ chân của module cánh quạt mini	17
16	Màn hình LCD 16x2	18
17	Màn hình hiển thị LCD 1602	18
18	Sơ đồ chân của màn hình LCD 1602	19
19	Dây nối tín hiệu	20
20	Hệ thống kiến trúc được đề xuất	21
21	Sơ đồ Usecase cho toàn hệ thống	23
22	Sơ đồ Usecase cho tính năng hiển thị dữ liệu thời gian thực	24
23	Sơ đồ Usecase cho tính năng hiển thị thống kê hoạt động	27
24	Sơ đồ Usecase cho tính năng hiển thị thông báo	29
25	Sơ đồ Usecase cho tính năng "Thiết lập ngữ cảnh"	31
26	Giao diện log in - Web App	33
27	Giao diện Sign up - Web App	34
28	Giao diện Sign in - Mobile App	34
29	Giao diện Sign up - Mobile App	35
30	Giao diện Homepage - Web App	35
31	Giao diện Homepage - Mobile App	36
32	Giao diện Notification của Homepage - Mobile App	36
33	Giao diện User Detail - Web App	37
34	Giao diện User Detail - Mobile App	37
35	Giao diện Manual Control - Web App	38
36	Giao diện Manual Control - Mobile App	39
37	Giao diện Statistics - Web App	39

38	Giao diện Statistics - Mobile App	40
39	Giao diện Context Setup trang chính - Web App	41
40	Giao diện cấu hình kịch bản Context Setup bước 1 - Web App	41
41	Giao diện cấu hình kịch bản Context Setup bước 2 - Web App	42
42	Giao diện hoàn thành kịch bản Context Setup - Web App	42
43	Các giao diện Context Setup phần 1 - Mobile App	43
44	Các giao diện Context Setup phần 2 - Mobile App	43
45	User Flow phần Login/Sign Up	44
46	User Flow phần Homepage	44
47	User Flow phần Container	45
48	User Flow phần Context Setup	45
49	User Flow phần Manual Settings	45
50	User Flow phần Statistics	46
51	Activity Diagram - Context Setup	47
52	Sơ đồ kết nối các thiết bị phần cứng	50
53	Cấu trúc dữ liệu từ Yolo:Bit gửi lên Gateway	52
54	Giao diện Dashboard Adafruit IO - Phần điều khiển thiết bị, giám sát các trạng thái	54
55	Giao diện Dashboard Adafruit IO - Phần quan sát các thông số dữ liệu	56
56	Danh sách các kênh Feeds trong server Adafruit IO	56
57	Cấu trúc thư mục của dự án với 2 phần backend (models, controller) và frontend tuân theo MVC pattern	58

1 Giới thiệu đề tài

1.1 Tên đề tài

Xây dựng hệ thống giám sát và điều khiển đèn chiếu sáng bằng cách sử dụng cảm biến ở các khu vực nhà vệ sinh, nhà kho dân cư, hành lang trường học,...

1.2 Bối cảnh đề tài

Trong thời đại công nghệ số ngày nay, việc xây dựng các hệ thống giám sát và điều khiển thông minh đã trở thành một trong xu hướng phát triển mạnh mẽ. Việc áp dụng các hệ thống này được đánh giá cao bởi tính hiệu quả và tiết kiệm năng lượng mà nó mang lại.

Trước đây, việc điều khiển đèn chiếu sáng tại các khu vực đặc biệt như nhà vệ sinh, nhà kho dân cư, hành lang trường học, các khu chung cư,... vẫn còn khá thô sơ. Đặc biệt những khu vực này còn tồn đọng nhiều vấn đề như quá vắng vẻ khi vào ban đêm, xảy ra những sự cố mà lại không được cảnh báo hay phát hiện kịp thời, hay thường không được để ý về quá trình bảo trì cơ sở vật chất,...

Với việc áp dụng công nghệ tiên tiến vào hệ thống giám sát và điều khiển đèn chiếu sáng, không chỉ các khu vực trên mà sau này khi hệ thống được cải tiến có thể mở rộng phạm vi sẽ trở nên an toàn hơn, tiết kiệm và thông minh hơn. Hệ thống giám sát và điều khiển đèn chiếu sáng sử dụng các thiết bị cảm biến sẽ tự động điều chỉnh các thông số phù hợp với điều kiện môi trường, từ đó giúp tiết kiệm năng lượng và giảm chi phí. Không chỉ vậy hệ thống còn giúp cho các khu vực trở nên xanh hơn, thân thiện hơn với môi trường, đồng thời còn giúp tạo ra một không gian sống và làm việc thông minh và tiện nghi hơn cho mọi người.

Dó là lí do nhóm chọn đề tài trên để hiện thực trong môn học thực tập đồ án lần này.

1.3 Các ý tưởng ban đầu của đề tài

Hệ thống có một số thiết bị như sau:

- Cảm biến ánh sáng giúp giám sát và phát hiện đèn bị hư.
- Cảm biến nhiệt độ, độ ẩm không khí giúp phát hiện sự thay đổi đột ngột về nhiệt độ/độ ẩm như đèn bị nóng quá mức hoặc nhiệt độ trong khu vực cao bất thường...
- Cảm biến hồng ngoại giúp phát hiện người để điều chỉnh bật/tắt đèn.
- Các thiết bị khác sẽ sử dụng: Yolo:Bit, mạch mở rộng Yolo:Bit, các thiết bị hỗ trợ cho quá trình quan sát hay điều khiển như remote, màn hình LCD,...

Hệ thống được nhóm hiện thực có một số tính năng sau:

- Hệ thống vận hành thông qua mạng không dây. Hệ thống dữ liệu đám mây. Phần mềm quản lý, theo dõi trên nền tảng thiết bị di động và web.
- Kiểm tra và phát ra cảnh báo (thay đổi màu sắc đèn và gửi cảnh báo) nếu: Thay đổi bất ngờ về nhiệt độ, độ ẩm; Đèn bị hư hỏng; Các trường hợp rủi ro được thiết lập trước ví dụ như: Khi có người ở liên tục quá lâu tại 1 khu vực (để phòng các trường hợp nguy cấp đột quy trong phòng tắm, phòng xông hơi...) hoặc khi có người đột nhập vào khu vực trong giờ giới nghiêm,...
- Điều khiển trực tiếp (bật tắt đèn, quạt thông hơi,...), thiết lập hoạt động của đèn/cảm biến theo lịch trình/thủ công. Điều khiển và thiết lập các thiết bị trên sơ đồ trực quan để có trải nghiệm dễ dàng hơn.
- Có Dashboard giúp hiển thị và theo dõi dữ liệu theo thời gian thực: nhiệt độ, độ ẩm, thời gian chiếu sáng của hệ thống đèn, phát hiện sự xuất hiện của con người.
- Thống kê thời gian hoạt động của hệ thống.

...

1.4 Cơ sở hiện thực đề tài

Để hiện thực được đề tài trên, nhóm đã tham khảo những tài liệu liên quan như hệ thống đề xuất, list hướng dẫn sử dụng các thiết bị và các giáo trình hướng dẫn thuộc lĩnh vực Internet of Things (IoT).

Nhóm tiến hành nghiên cứu và tổng hợp được những lý thuyết có thể ứng dụng trong đề tài thành file có link đính kèm dưới đây.

Link tài liệu cở sở lí thuyết

2 Phân tích yêu cầu

2.1 Các bên liên quan đến dự án (Stakeholder)

Các bên liên quan đến dự án này bao gồm:

- *User*: Phụ trách việc giám sát, thiết lập và điều khiển các hành vi của hệ thống
- *Guest*: Đối tượng sử dụng các tính năng giám sát, tự động hóa của hệ thống
- *Đơn vị phát triển hệ thống*: Phụ trách việc phân phối hệ thống ra thị trường
- *Đội ngũ kỹ thuật viên*: Phụ trách bảo trì, nâng cấp, sửa chữa hệ thống

Stakeholder	Nhu cầu/Vấn đề đặt ra	Giải pháp từ hệ thống
User	<ul style="list-style-type: none"> - Cần có công cụ để điều khiển được hệ thống chiếu sáng đèn phù hợp với nhu cầu của cá nhân (tăng tính cá nhân hóa). - Cần có công cụ giúp hiển thị các thông tin về nhiệt độ, độ ẩm theo thời gian thực. - Cần có công cụ giúp thống kê được các thông số về nhiệt độ, độ ẩm, độ sáng của đèn. - Cần có công cụ giúp thiết lập các cảnh báo sử dụng hệ thống đèn chiếu sáng. 	<ul style="list-style-type: none"> - Giúp điều chỉnh thời gian sáng, lịch trình bật/tắt của đèn dựa theo ngữ cảnh được cài đặt trước. - Giúp cập nhật thông số về nhiệt độ, độ ẩm của môi trường xung quanh khu vực giám sát theo thời gian thực với độ trễ thấp. - Giúp ghi lại lịch sử về các thông số nhiệt độ, độ ẩm cũng như thời gian hoạt động của đèn và hiển thị dưới dạng đồ thị trực quan. - Cài đặt ngữ cảnh được đi kèm với việc tự do lựa chọn việc điều khiển bật/tắt đèn thủ công, màu của đèn và khoảng thời gian đèn hoạt động tối đa.
Guest	<ul style="list-style-type: none"> - Cần có hệ thống giám sát đèn giúp tự động bật/tắt hoặc thay đổi màu sắc của đèn phù hợp với nhu cầu của người dùng. - Có thể giúp người dùng phát hiện tình huống không an toàn ở trong khu vực giám sát. 	<ul style="list-style-type: none"> - Giúp điều khiển bật/tắt đèn thủ công cũng như tự động hóa thông qua việc thiết lập ngữ cảnh hoạt động của đèn. - Giúp kiểm tra những tình huống bất thường để thông báo qua việc thay đổi màu sắc của đèn và gửi thông báo về thiết bị.

2.2 Yêu cầu chức năng

Đối với User:

- Có thể xem dữ liệu về thời gian đèn hoạt động trong ngày
- Có thể xem dữ liệu về nhiệt độ, độ ẩm, cường độ ánh sáng của môi trường xung quanh hệ thống theo thời gian thực
- Có thể xem biểu đồ thống kê thời gian hoạt động của đèn (phạm vi 1 tháng)
- Có thể xem biểu đồ thống kê biến thiên nhiệt độ, độ ẩm, cường độ ánh sáng (phạm vi 1 tháng)
- Có thể nhận cảnh báo về thiết bị khi xảy ra trường hợp nhiệt độ, độ ẩm hoặc cường độ ánh sáng vượt ngưỡng bình thường
- Có thể nhận thông báo về tình trạng bật/tắt của đèn
- Có thể thiết lập ngữ cảnh và hành vi tương ứng của đèn
- Có thể điều chỉnh màu sắc, bật/tắt đèn thủ công
- Có thể nhận được thông báo khi có người đi vào khu vực giám sát trong thời gian giới nghiêm hoặc ở trong khu vực giám sát vượt quá thời gian tối đa cho phép

Đối với Guest:

- Có thể đăng nhập vào hệ thống giám sát đèn
- Có thể thấy được sự thay đổi màu sắc của đèn khi xảy ra trường hợp khẩn cấp
- Có thể nhận được cảnh báo từ màu sắc đèn khi đi vào khu vực được giám sát trong thời gian giới nghiêm
- Có thể nhận được cảnh báo từ màu sắc đèn khi đi vào khu vực được giám sát vượt quá thời gian cho phép

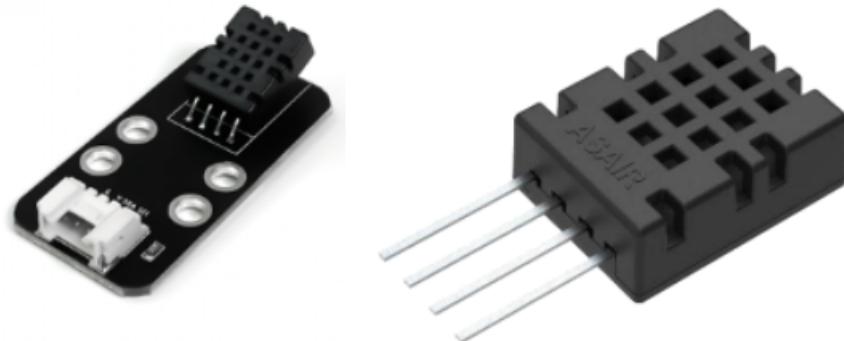
2.3 Yêu cầu phi chức năng

Security	<ul style="list-style-type: none"> - Password của người dùng phải được hash bằng MD5. - Hệ thống sẽ tạm deactivate tài khoản của người dùng nếu nhập sai password 5 lần liên tiếp. - Khi user quên mật khẩu và thực hiện chức năng lấy lại mật khẩu, mật khẩu mặc định của hệ thống phải được gửi về email được dùng để đăng ký tài khoản, hoặc gửi mã OTP về cho số điện thoại được sử dụng để đăng ký tài khoản. - Yêu cầu password của người dùng khi khởi tạo phải có độ dài tối thiểu 8 ký tự, bao gồm kí tự đặc biệt, chữ cái thường, chữ cái in hoa và số.
Performance	<ul style="list-style-type: none"> - Đối với màn hình input: tối đa 20 trường dữ liệu, không tính toán dữ liệu phức tạp, không tương tác với hệ thống ngoài, có thể lưu trữ dữ liệu trực tiếp ngay xuống database, không lưu trữ các tệp nội dung lớn như hình ảnh, video, tập tin quá 3 MB. - Đối với màn hình output: dữ liệu được truy vấn trực tiếp từ database, hạn chế những câu lệnh truy vấn phức tạp, những truy vấn từ hệ thống ngoài. Hiển thị tối đa 50 dòng dữ liệu, mỗi dòng tối đa 10 cột, có độ dài nhỏ hơn 100 ký tự.
Usability	<ul style="list-style-type: none"> - Tất cả thông tin quan trọng phải được hiển thị trong 1 màn hình (không cần phải thực hiện thêm thao tác cuộn). - Khi người dùng nhập dữ liệu chưa chính xác, chỉ cần cho phép chỉnh sửa lại dữ liệu bị sai, không bắt buộc phải nhập lại toàn bộ dữ liệu. Kiểm tra kiểu dữ liệu,... trên từng trường nhập và đưa ra cảnh báo tức thì nếu việc nhập liệu bị sai. - Giao diện của hệ thống cần có sự nhất quán, về mặt hình ảnh biểu tượng cũng như vị trí các đối tượng trên màn hình để người dùng làm quen dễ dàng hơn. - Người dùng có thể thành thạo các thao tác trên màn hình trong 15 phút sử dụng.

Audit	<ul style="list-style-type: none"> - Dữ liệu audit cần lưu trữ tách biệt trong một database riêng, độc lập hoàn toàn với database chính của hệ thống. - Dữ liệu của hệ thống phải được sao lưu định kỳ, và tồn tại trong vòng 30 ngày. - Các dữ liệu audit phải ở chế độ Read Only và không cho phép chỉnh sửa từ giao diện người dùng.
Supportability	<ul style="list-style-type: none"> - Hệ thống có thể được sử dụng hiệu quả trên các trình duyệt web (Opera, UC Browser, Safari, Microsoft Edge, Google Chrome, Samsung Internet Browser). - Hệ thống không chạy trên các phiên bản trình duyệt quá cũ.
Maintainability	<ul style="list-style-type: none"> - Cần nâng cấp, bảo trì hệ thống định kỳ 6 tháng một lần, mỗi lần bảo trì không quá 2 tiếng.
Scalability	<ul style="list-style-type: none"> - Server có khả năng upgrade cấu hình - Có khả năng tách database trên một server riêng và backend trên một server riêng
Localization	<ul style="list-style-type: none"> - Hệ thống sử dụng ngôn ngữ tiếng Việt

3 Danh sách thiết bị phần cứng

3.1 Cảm biến nhiệt độ độ ẩm DHT20



Hình 1: Cảm biến DHT20

Pins	Name	Describe	
1	VDD	Power supply(2.2v to 5.5v)	
2	SDA	Serial Data Bidirectional port	
3	GND	Ground	
4	SCL	Serial clock Bidirectional port	

Hình 2: Sơ đồ chân của cảm biến DHT20 theo datasheet đính kèm

DHT20 là sản phẩm nâng cấp mới của DHT11, được trang bị chip cảm biến ASIC chuyên dụng, cảm biến độ ẩm điện dung dựa trên silicon bán dẫn hiệu suất cao và cảm biến nhiệt độ trên chip tiêu chuẩn, đồng thời sử dụng định dạng tín hiệu đầu ra dữ liệu I²C tiêu chuẩn. Hiệu suất của nó đã được cải thiện rất nhiều và vượt quá mức độ tin cậy của cảm biến thế hệ trước (DHT11). Thế hệ sản phẩm nâng cấp mới đã được cải tiến để giúp hiệu suất của chúng ổn định hơn trong môi trường nhiệt độ cao và độ ẩm cao; đồng thời, độ chính xác, thời gian đáp ứng và phạm vi đo lường của sản phẩm đã được cải thiện rất nhiều.

Người dùng có thể ứng dụng cảm biến này vào các dự án điều khiển tự động, ghi nhận dữ liệu về nhiệt độ, độ ẩm trong môi trường xung quanh, làm máy hút ẩm,... và nhiều dự án khác.

Để sử dụng sản phẩm module cảm biến, người dùng cần biết cách lập trình để điều khiển cảm biến hoạt động được. Chúng ta sẽ sử dụng phần mềm OhStem App cùng với các board điều khiển như máy tính Yolo:Bit, mạch Arduino Easy Kit... để lập trình điều khiển DHT20.

Thông số kỹ thuật:

- Hỗ trợ điện áp rộng 2.2 đến 5.5VDC
- Chất liệu cảm biến: Nhựa ABS.
- Đầu ra nhiệt độ và độ ẩm tương đối
- Phạm vi đo nhiệt độ: -40 -> + 80 độ C
- Phạm vi đo độ ẩm: 0 -> 100% RH
- Độ chia nhỏ nhất T: 0.01 độ C RH; 0.024
- Hiệu suất cảm biến vượt trội, độ chính xác điển hình RH: ±3%, T: ±0,5 độ C
- Đầu ra kỹ thuật số được hiệu chỉnh và xử lý đầy đủ, giao thức I2C
- Cảm biến ổn định lâu dài
- Khả năng phản hồi nhanh và chống nhiễu

Link datasheet: [Data Sheet DHT20 - Humidity and Temperature Module](#)

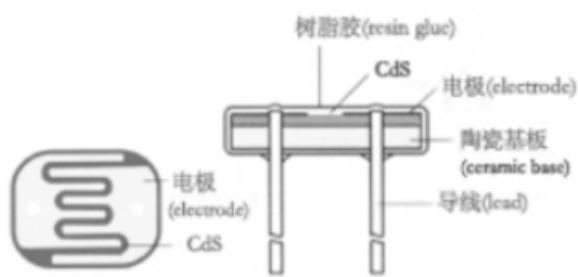
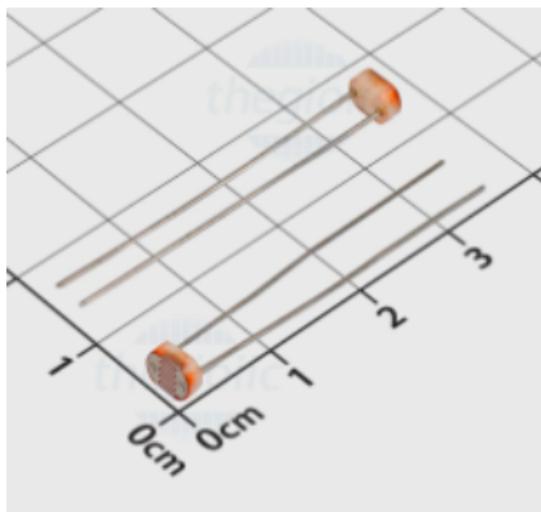
3.2 Cảm biến ánh sáng



Hình 3: Cảm biến ánh sáng

Cảm biến ánh sáng là thiết bị giúp nhận biết và phát hiện cường độ ánh sáng của môi trường xung quanh. Cảm biến này thích hợp để làm các ứng dụng cơ bản về nhận biết ánh sáng, biết được trời sáng hay trời tối và nhiều ứng dụng thú vị khác.

Cảm biến sử dụng con quang điện trở GL5528 có đường kính 5mm, là một điện trở được làm bằng vật liệu bán dẫn và độ dãn thay đổi theo sự thay đổi của ánh sáng. Điện trở quang được sử dụng rộng rãi trong nhiều ngành công nghiệp, chẳng hạn như đồ chơi, đèn, máy ảnh,...



Hình 4: Hình ảnh quang điện trở GL5528

Thông số kỹ thuật:

- Điện áp hoạt động: 3-5V
- Nhiệt độ môi trường: -30 -> +70 độ C
- Dòng cung cấp: 0.5-3mA
- Điện trở quang: GL5528
- Phạm vi giá trị đọc cảm biến Analog: 0 - 4095 hoặc %: 0 - 100
- Điện trở khi có ánh sáng: 20K Ohm
- Điện trở khi không có ánh sáng: 1M Ohm
- Thời gian phản hồi: 20-30 secs
- Bước sóng tối đa: 540 nm

STT	Chân	Chức năng
1	GND	Nối đất
2	VCC	Cấp nguồn
3	NC	T Không sử dụng
4	SIG	Tín hiệu ngõ ra của cảm biến

Hình 5: Sơ đồ chân của module cảm biến ánh sáng

3.3 Cảm biến hồng ngoại



Hình 6: Cảm biến hồng ngoại PIR

Cảm biến PIR (Passive infrared sensor) được sử dụng để phát hiện sự chuyển động của các vật thể phát ra bức xạ hồng ngoại (ví dụ như sự chuyển động của con người, con vật, các vật phát nhiệt,...).

Cảm biến này còn có tên gọi khác là cảm biến thân nhiệt chuyển động PIR. Người dùng có thể chỉnh được độ nhạy của cảm biến để giới hạn khoảng cách nhận dạng chuyển động trong mức khoảng cách xa hoặc gần, cũng như cường độ bức xạ của vật thể mong muốn.



Hình 7: IC AS312

Thông số kỹ thuật:

- Điện áp hoạt động: 5VDC
- Phạm vi phát hiện: góc 360 độ hình nón, độ xa tối đa 6m.
- Nhiệt độ hoạt động: 32-122 độ F (0-50 độ C)
- Mức tiêu thụ dòng: <= 50 uA

- IC: AS312
- Kiểu tín hiệu: Digital
- Kích thước của mạch: 24mm x 48mm x 16mm

STT	Chân	Chức năng
1	GND	Nối đất
2	VCC	Cấp nguồn
3	NC	Không sử dụng
4	SIG	Tín hiệu cảm biến

Hình 8: Sơ đồ chân của module cảm biến hồng ngoại PIR

3.4 Mạch Yolo:Bit



Hình 9: Mạch Yolo:Bit

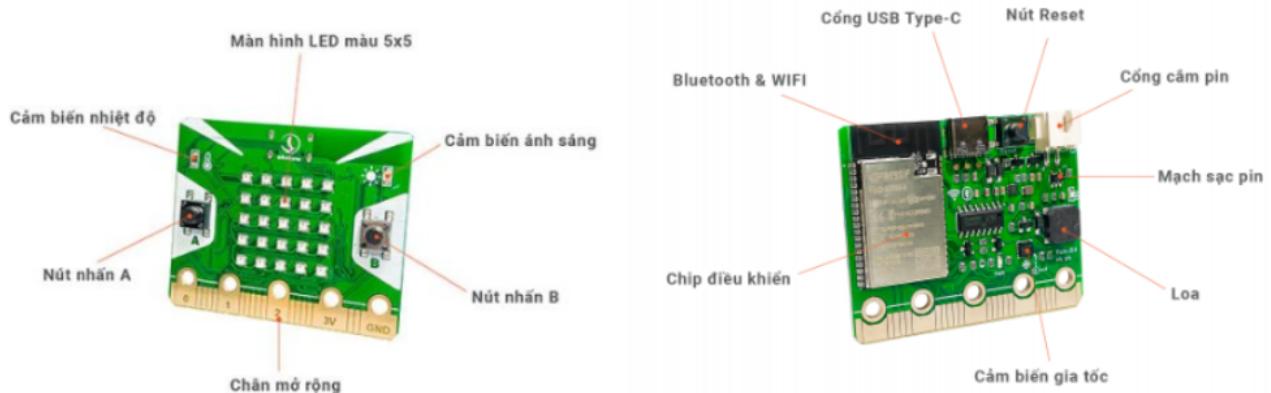
Yolo:Bit là một máy tính nhỏ gọn có thẻ lập trình, phù hợp để dạy lập trình & điện tử cơ bản cho trẻ em. Tích hợp màn hình LED đa màu, nút nhấn, loa phát nhạc và nhiều cảm biến thông minh để phục vụ việc lập trình và tương tác thực tế. Có ma trận LED đa màu 5×5 & 2 nút nhấn để làm nhiều ứng dụng vui nhộn. Được kết nối qua USB / Bluetooth.

Thông số kỹ thuật:

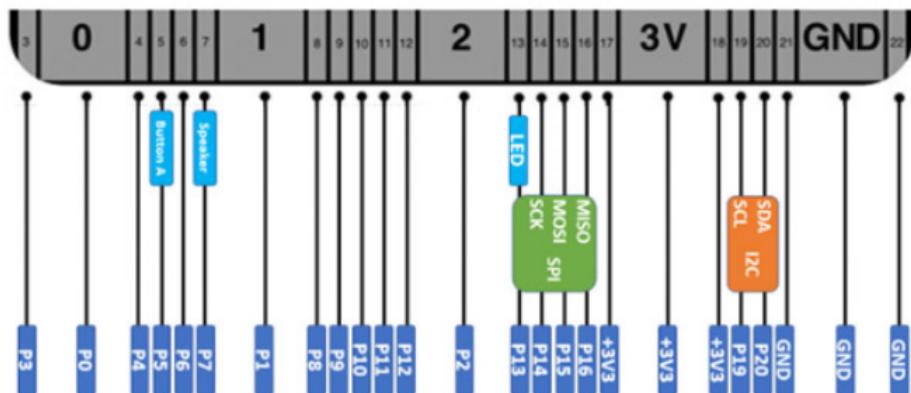
Trên bộ Kit học STEM Yolo:Bit có tích hợp nhiều bộ phận như:

- 25 đèn LED đa màu, được sắp xếp theo ma trận vuông 5×5 .
- 2 nút nhấn A, B để tương tác.
- Cảm biến ánh sáng.
- Cảm biến nhiệt độ và độ ẩm.
- Cảm biến gia tốc để đo góc nghiêng và hướng chuyển động.
- Loa phát nhạc.
- Nhiều chân cảm mở rộng, giúp người dùng dễ dàng kết nối với các module chức năng mở rộng khác để tạo thành nhiều dự án sáng tạo hơn.
- Lưu ý: Chỉ cắm pin có nguồn điện từ 3.7V – 6V vào Yolo:Bit
- Kích thước: 4cm x 5 cm.
- RAM: 8MB.

- Flash memory: 4MB.
- Nhỏ gọn, dễ dàng lập trình.
- An toàn và có độ bền cao.

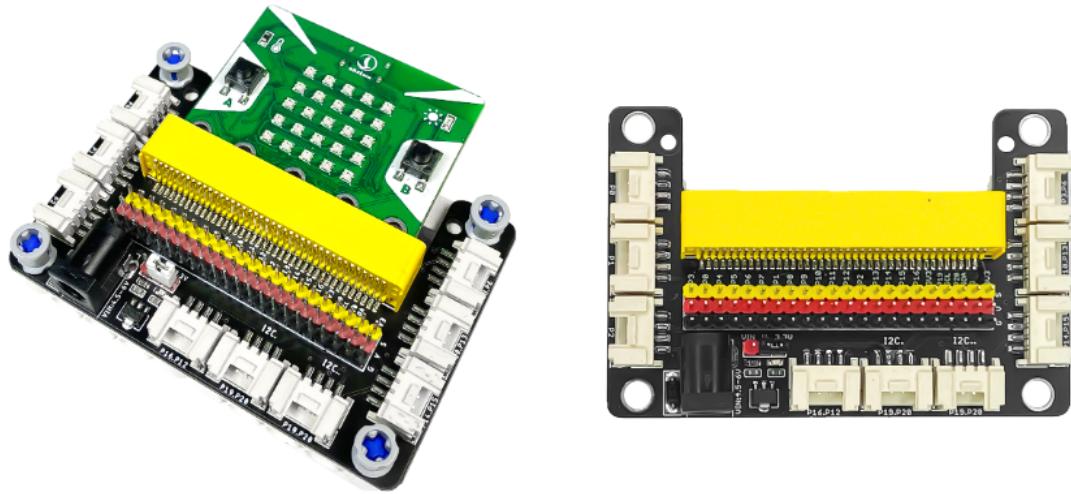


Hình 10: Các thành phần của Yolo:Bit



Hình 11: Sơ đồ chân của Yolo:Bit

3.5 Grove shield - Mạch mở rộng cho Yolo:Bit



Hình 12: Grove shield - Mạch mở rộng cho Yolo:Bit

Grove shield là shield mở rộng tương thích với Yolo:Bit, giúp mở rộng thêm 6 port kết nối chuẩn grove (3 port analog, 1 port I2C, 1 port UART).

Thông số kỹ thuật:

- 3 cổng Analog
- 1 cổng I2C
- 1 cổng UART
- Ngoài ra, tất cả các pin có thể sử dụng được từ Yolo:Bit còn được bố trí thành dạng chân cắm header, dễ dàng kết nối Yolo:Bit với tất cả các loại cảm biến trong hệ sinh thái module của OhStem nói riêng và các cảm biến trên thị trường nói chung.
- Các tín hiệu P0, P1, P2, P3, 3V, GND còn có thể sử dụng được với kẹp cá sấu.
- Trên mạch mở rộng của Yolo:Bit có tổng cộng 9 khe cắm khác nhau, phân loại như sau:
 - Khe cắm đa dụng một tín hiệu: P0, P1 và P2.
 - Khe cắm hai tín hiệu: P16/12, P14/15, P10/13, P3/6.
 - Khe giao tiếp I2C: I2C1 và I2C2, cùng là chân P19/20.

3.6 Remote điều khiển từ xa và mắt nhận hồng ngoại



Hình 13: Remote điều khiển từ xa và mắt nhận hồng ngoại

Trong trường hợp muốn tăng thông tin tương tác với người dùng, người dùng có thể tích hợp thêm remote hồng ngoại. Trên remote có sẵn nhiều nút nhấn, bao gồm cả chữ và số. Đây là công cụ mở rộng hữu dụng trên Yolo:Bit nhằm tăng số lượng nút nhấn trong một ứng dụng. Bàn phím số này gồm có 2 phần: Remote điều khiển từ xa và mắt nhận hồng ngoại.

Nguyên lý hoạt động của nó cũng giống với các remote trong dân dụng, như remote tivi hoặc máy lạnh chẳng hạn. Khi sử dụng remote, người dùng phải hướng nó vào mắt nhận thì việc kết nối mới ổn định và chính xác. Sóng hồng ngoại là sóng có hướng và có độ phản xạ tương đối thấp. Do đó, khi không hướng trực tiếp vào mắt nhận, xác suất nhận được tín hiệu là tương đối thấp.

Người dùng cần lưu ý lấy đúng mắt nhận hồng ngoại, bởi nó hơi giống với cảm biến ánh sáng. Đối với remote, bạn cần loại bỏ tấm chắn pin bằng plastic để pin có thể tiếp xúc với điện cực, lúc đó remote mới hoạt động. Khi sản xuất, tấm chắn plastic này hạn chế việc chạm phím khi vận chuyển, tránh hao pin hoặc gây hư hỏng remote điều khiển.

3.7 Quạt mini



Hình 14: Quạt mini

Động cơ quạt mini gồm 2 bộ phận: động cơ và cánh quạt giúp bạn có thể tạo thành một quạt máy mini và ứng dụng trong nhiều dự án sáng tạo khác nhau. Ví dụ: Bạn có thể lập trình để quạt tự động bật / tắt dựa theo nhiệt độ, tự động bật quạt khi có người,...

Động cơ là một thiết bị điện tử rất phổ biến trong cuộc sống của chúng ta (như quạt, động cơ xe, máy bơm nước...). Động cơ khi được cung cấp điện sẽ làm xoay trực động cơ, từ đó tạo nên nhiều ứng dụng khác nhau.

Ví dụ, khi bạn gắn cánh quạt vào trực động cơ thì chúng ta sẽ có được động cơ quạt mini; nếu ta gắn móc treo vào thì chúng ta sẽ có được cần câu...

Ta có thể kết hợp module động cơ này với những module cảm biến khác như: điều khiển bật/tắt quạt theo thời gian, đọc giá trị nhiệt độ để bật/tắt quạt,...

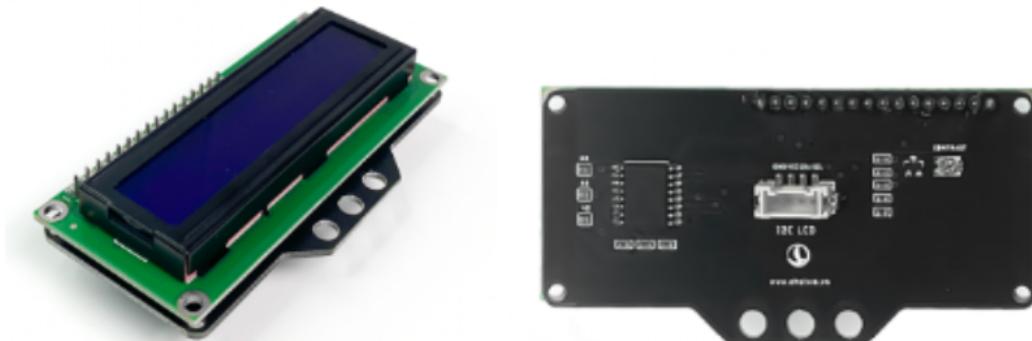
Thông số kỹ thuật:

- Điện áp hoạt động: 5V
- Tín hiệu điều khiển: 2 pins
- Kích thước của mạch: 24mm x 48mm x 16mm

STT	Chân	Chức năng
1	GND	Nối đất
2	VCC	Cấp nguồn
3	S2	Tín hiệu điều khiển quay nghịch
4	S1	Tín hiệu điều khiển quay thuận

Hình 15: Sơ đồ chân của module cánh quạt mini

3.8 Màn hình LCD 16x2



Hình 16: Màn hình LCD 16x2

Màn hình LCD 1602 là thiết bị điện tử lý tưởng cho những ai mới tiếp xúc và học về dự án. Thiết bị có thể hiển thị 2 dòng, mỗi dòng 16 ký tự. Bạn có thể lập trình để màn hình LCD1602 hiển thị bất kỳ ký tự, thông tin nào bạn cần. Module này có độ bền cao, đồng thời có tích hợp module giao tiếp I2C giúp việc giao tiếp được dễ dàng và nhanh chóng hơn rất nhiều.

Màn hình LCD 1602 có kèm module I2C sử dụng driver HD44780. Module này có khả năng hiển thị 2 dòng với mỗi dòng 16 ký tự. LCD 1602 có độ bền cao và rất phổ biến (có nhiều code mẫu). Nếu bạn là người mới học và làm dự án, đây sẽ là thiết bị điện tử phù hợp nhờ vào tính dễ sử dụng của chúng. Màn hình LCD được tích hợp module giao tiếp I2C giúp việc giao tiếp được dễ dàng và nhanh chóng hơn rất nhiều.



Hình 17: Màn hình hiển thị LCD 1602

Thông số kỹ thuật của màn hình LCD 1602:

- Điện áp MAX : 7V.
- Điện áp MIN : -0,3V.

- Hoạt động ổn định : 2.7-5.5V.
- Điện áp ra mức cao : > 2.4V.
- Điện áp ra mức thấp : < 0.4V.
- Dòng điện cấp nguồn : 350uA 600uA.
- Nhiệt độ hoạt động : -30 75 độ C.
- Bên trong LCD1602 có tích hợp sẵn chip VDK HD44780.
- Địa chỉ I2C: 0x27
- Màu: Xanh lá
- Kích thước lỗ bắt ốc: 3x M3
- Kích thước của mạch: 80mm x 42mm x 19mm
- Trọng lượng: 38g

STT	Chân	Chức năng
1	GND	Nối đất
2	VCC	Cấp nguồn
3	SDA	Truyền tín hiệu
4	SCL	Xung tín hiệu

Hình 18: Sơ đồ chân của màn hình LCD 1602

3.9 Dây nối tín hiệu



Hình 19: Dây nối tín hiệu

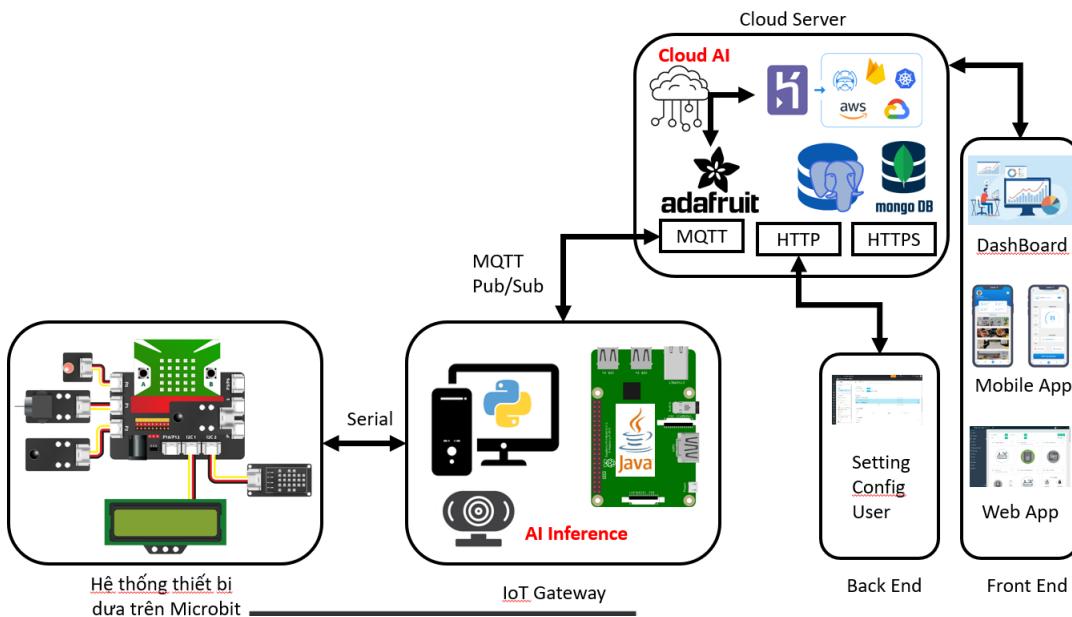
Sản phẩm dây nối tín hiệu được dùng để kết nối các module điện tử và bo mạch với nhau một cách đơn giản, không cần phải hàn gắn phức tạp như khi sử dụng các dây Jumper.

Thông số sản phẩm:

- Chất liệu: nhựa và kim loại
- Chiều dài dây: 20cm / 30cm / 40cm

4 Mô hình kiến trúc hệ thống

Mô hình kiến trúc hệ thống được đề xuất như hình ảnh dưới đây:



Hình 20: Hệ thống kiến trúc được đề xuất

Sau khi phân tích hệ thống được đề xuất cùng với quá trình bàn bạc để lựa chọn công nghệ sử dụng, nhóm quyết định đề xuất hệ thống cụ thể được mô tả như sau:

Hệ thống kiến trúc nhóm đề xuất tuân theo chuẩn mô hình kiến trúc IoT gồm 4 phần:

1. ***Node:***

Node bao gồm **YoloBit** đóng vai trò làm node trung tâm. YoloBit sẽ được gắn vào mạch Grove Shield mở rộng để có thể kết nối với các module cảm biến và thiết bị.

Các cảm biến sẽ đo và gửi các dữ liệu lên Yolo:Bit, sau đó Yolobit sẽ tiến hành gửi dữ liệu đã được nhận qua Gateway nhờ vào kết nối Serial USB.

Ngoài ra từ Yolo:Bit cũng sẽ tiến hành nhận các dữ liệu nhận được từ Gateway và tiến hành xử lý dữ liệu, gửi lệnh điều khiển các thiết bị hay hiển thị các thông tin ra LED LCD.

2. ***Gateway:***

Nhóm lựa chọn kiến trúc Gateway gồm các thiết bị máy tính như laptop, máy tính để bàn,... chạy chương trình hiện thực Gateway sử dụng ngôn ngữ Python vì tính phổ biến và linh hoạt cao.

Gateway sẽ tiến hành nhận về các gói dữ liệu từ Yolobit của Node để tiến hành phân tích gói, cuối cùng lấy dữ liệu vừa được phân tích để gửi lên các kênh Feed phù hợp của server **Adafruit IO** theo cơ chế *MQTT Publish*.

Ở chiều ngược lại, Gateway sẽ tiến hành nhận dữ liệu từ các kênh Feed của server Adafruit IO theo cơ chế *MQTT Subscribe* và tiến hành gửi xuống Yolo:Bit.

Như vậy Gateway đóng vai trò trung gian và điều phối luồng dữ liệu giữa Node và Server Adafruit IO.

3. *Server:*

Ở phần Server nhóm sẽ có Server Adafruit IO là nơi lưu trữ dữ liệu theo các kênh Feed. Điều này giúp phân chia và quản lý luồng dữ liệu một cách chặt chẽ, rõ ràng ứng với từng tính năng mà nhóm sẽ hiện thực. Adafruit IO kết nối với Gateway theo cơ chế MQTT. Ngoài ra Adafruit IO còn cung cấp Dashboard dễ sử dụng và có thể giúp nhóm quan sát hoạt động ở khía cạnh của cả hệ thống.

Ngoài Adafruit IO, phần Server của nhóm còn có Server MongoDB được xây dựng để nhóm hiện thực phần Backend trong thành phần cuối: *Các thiết bị để theo dõi dữ liệu và điều khiển từ xa*. Server này sẽ kết nối đồng bộ dữ liệu 2 chiều với Adafruit IO.

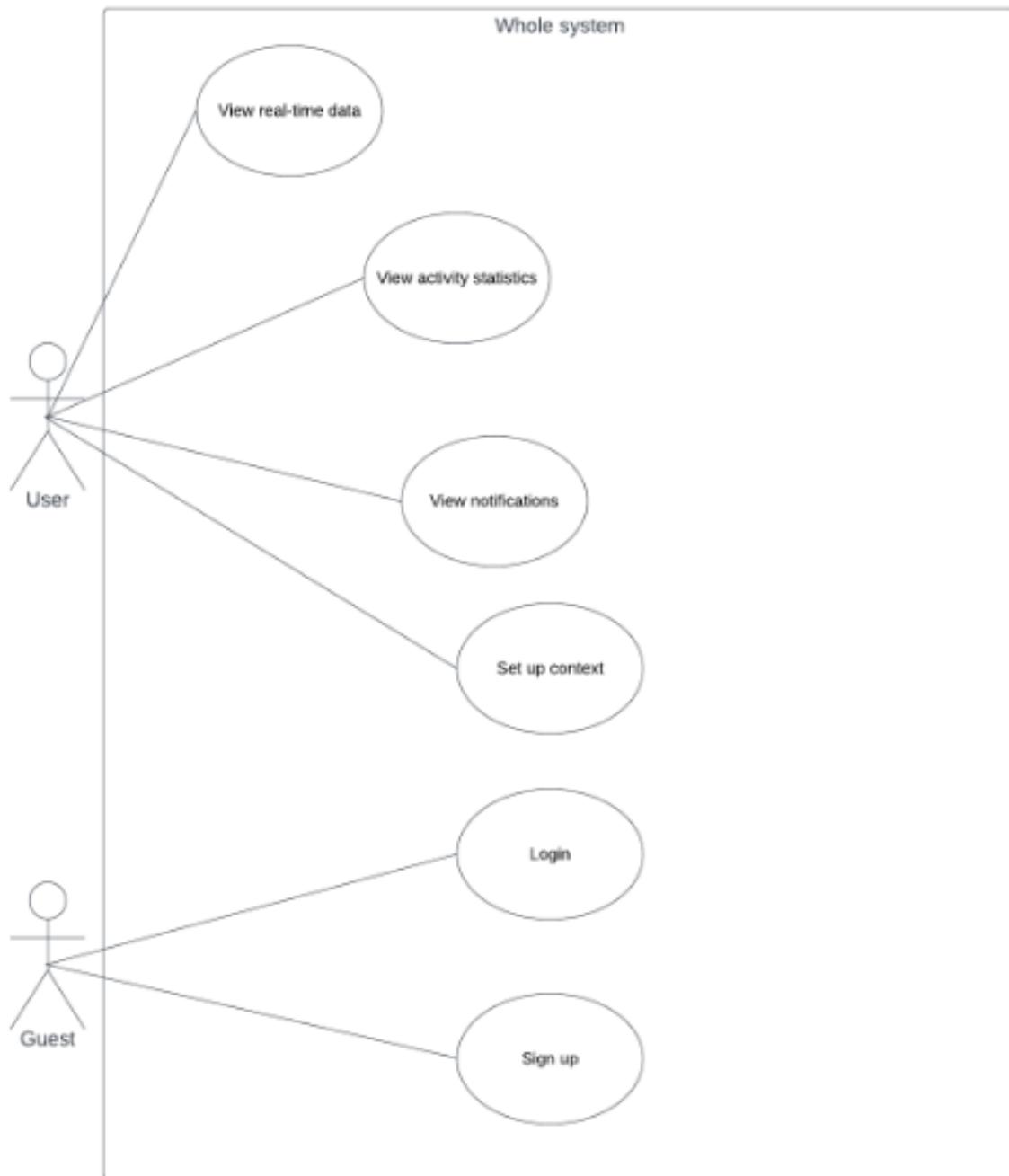
4. *Các thiết bị để theo dõi dữ liệu và điều khiển từ xa:*

Nhóm sẽ hiện thực ứng dụng để theo dõi dữ liệu và điều khiển từ xa ở cả Web App và Mobile App.

Ngoài ra các thiết bị điều khiển từ xa cũng sẽ bao gồm remote điều khiển và một chương trình AI dùng để nhận diện giọng nói sử dụng kết nối Bluetooth kết nối với Yolo:Bit.

5 Use-case diagram

5.1 Biểu đồ Usecase cho toàn hệ thống



Hình 21: Sơ đồ Usecase cho toàn hệ thống

Bảng danh sách các Actors:

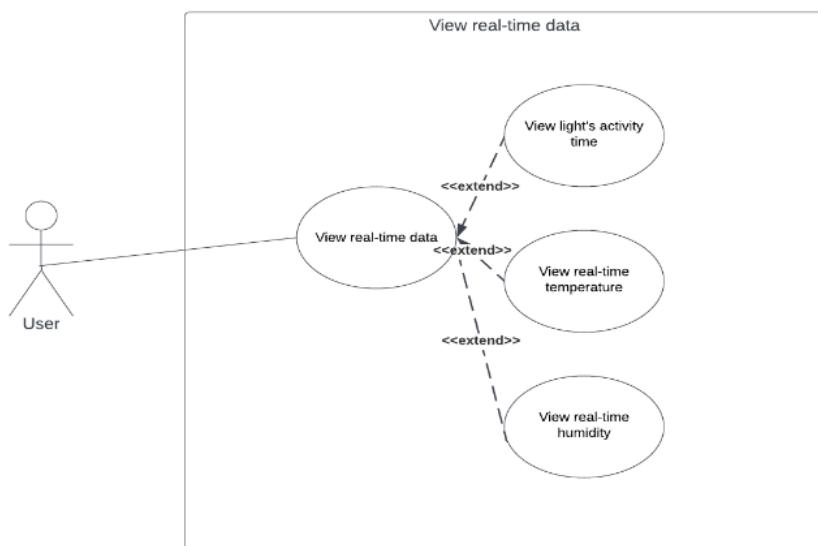
Actor ID	Tên Actor
1	- Khách hàng (Guest - chưa đăng nhập vào hệ thống)
2	- Người dùng (User - đã đăng nhập vào hệ thống)

Bảng mô tả các use case chính của hệ thống:

Use case ID	Tên Use case	Mô tả Use case
1	Sign up	Đăng ký vào hệ thống
2	Log in	Đăng nhập vào hệ thống
3	View real-time data	Xem, theo dõi các dữ liệu theo thời gian thực
4	View activity statistics	Xem, theo dõi các báo cáo thống kê về các hoạt động
5	View notifications	Xem các thông báo
6	Set up context	Thiết lập ngữ cảnh

5.2 Biểu đồ Usecase cho các tính năng chính

5.2.1 Hiển thị dữ liệu thời gian thực



Hình 22: Sơ đồ Usecase cho tính năng hiển thị dữ liệu thời gian thực

Use-case name	View real-time data
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem các dữ liệu thời gian thực
Trigger	User nhấp vào nút “View real-time data” trên dashboard
Preconditions	Không
Postconditions	Hệ thống hiển thị các dữ liệu theo thời gian thực cho User
Normal flow	<ol style="list-style-type: none"> Người dùng chọn nút “View real-time data”. Hệ thống hiển thị menu danh sách các dữ liệu được thu thập theo thời gian thực. Người dùng chọn xem chi tiết dữ liệu.
Alternative flows	Không
Exceptions	Không

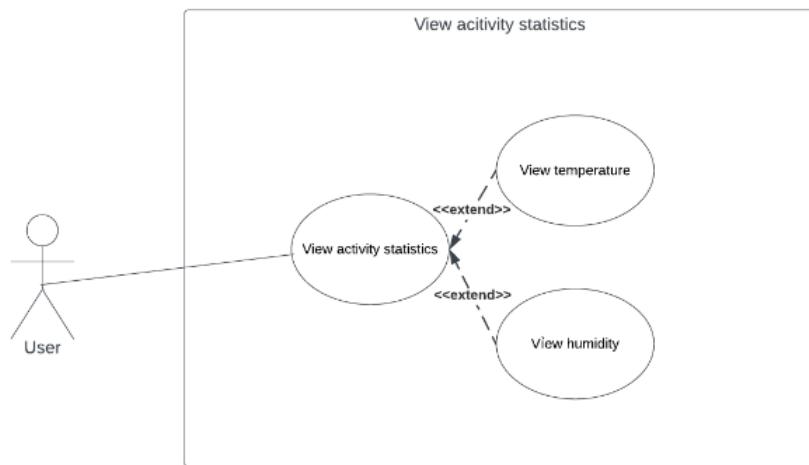
Use-case name	View light's activity time
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem thời gian hoạt động của đèn
Trigger	User nhấp vào nút “View light's activity time”
Preconditions	User đã nhấp vào nút “View real-time data”
Postconditions	Hệ thống hiển thị thời gian hoạt động của đèn cho User
Normal flow	<ol style="list-style-type: none"> Người dùng chọn nút “View light's activity time”. Hệ thống hiển thị thời gian thực hoạt động của đèn trong ngày.
Alternative flows	Không
Exceptions	Không

Use-case name	View real-time temperature
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem dữ liệu về nhiệt độ theo thời gian thực

Trigger	User nhấp vào nút “View real-time temperature”
Preconditions	User đã nhấp vào nút “View real-time data”
Postconditions	Hệ thống hiển thị dữ liệu về nhiệt độ theo thời gian thực cho User
Normal flow	1. Người dùng chọn nút “View real-time temperature”. 2. Hệ thống hiển thị thông tin dữ liệu về nhiệt độ theo thời gian thực.
Alternative flows	Không
Exceptions	Không

Use-case name	View real-time humidity
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem dữ liệu về độ ẩm theo thời gian thực
Trigger	User nhấp vào nút “View real-time humidity”
Preconditions	User đã nhấp vào nút “View real-time data”
Postconditions	Hệ thống hiển thị dữ liệu về độ ẩm theo thời gian thực cho User
Normal flow	1. Người dùng chọn nút “View real-time humidity”. 2. Hệ thống hiển thị thông tin dữ liệu về độ ẩm theo thời gian thực.
Alternative flows	Không
Exceptions	Không

5.2.2 Hiển thị thống kê hoạt động



Hình 23: Sơ đồ Usecase cho tính năng hiển thị thống kê hoạt động

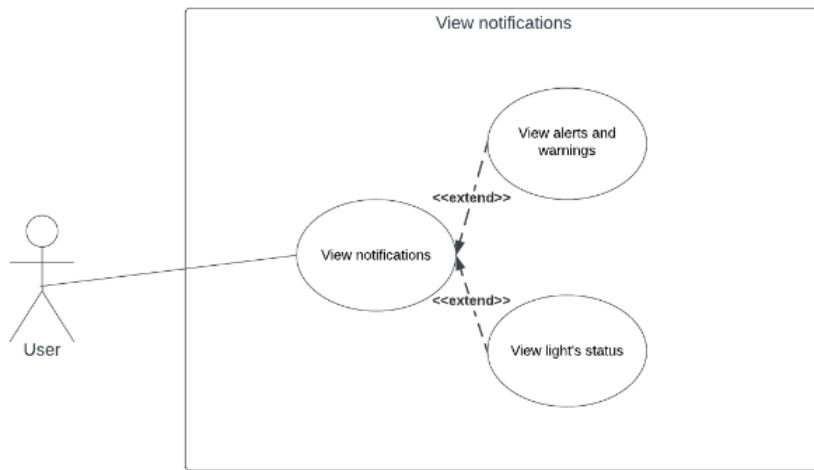
Use-case name	View activity statistics
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem các thống kê hoạt động
Trigger	User nhấn vào nút “View activity statistics” trên dashboard
Preconditions	Không
Postconditions	Hệ thống hiển thị các thống kê về các hoạt động cho User
Normal flow	<ol style="list-style-type: none"> Người dùng chọn nút “View activity statistics”. Hệ thống hiển thị menu danh sách các thống kê về các hoạt động. Người dùng chọn xem chi tiết hoạt động.
Alternative flows	Không
Exceptions	Không

Use-case name	View temperature
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem các thống kê về nhiệt độ

Trigger	User nhấn vào nút “View temperature”
Preconditions	User đã nhấn vào nút “View activity statistics” trên dashboard
Postconditions	Hệ thống hiển thị các thống kê về nhiệt độ cho User
Normal flow	<ol style="list-style-type: none"> 1. Người dùng chọn nút “View temperature”. 2. Hệ thống hiển thị menu danh sách các thống kê về nhiệt độ trong vòng 7 ngày, 30 ngày. 3. Người dùng chọn xem chi tiết hoạt động trong thời gian cụ thể.
Alternative flows	Không
Exceptions	Không

Use-case name	View humidity
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem các thống kê về độ ẩm
Trigger	User nhấn vào nút “View humidity”
Preconditions	User đã nhấn vào nút “View activity statistics” trên dashboard
Postconditions	Hệ thống hiển thị các thống kê về độ ẩm cho User
Normal flow	<ol style="list-style-type: none"> 1. Người dùng chọn nút “View humidity”. 2. Hệ thống hiển thị menu danh sách các thống kê về độ ẩm trong vòng 7 ngày, 30 ngày. 3. Người dùng chọn xem chi tiết hoạt động trong thời gian cụ thể.
Alternative flows	Không
Exceptions	Không

5.2.3 Hiển thị thông báo



Hình 24: Sơ đồ Usecase cho tính năng hiển thị thông báo

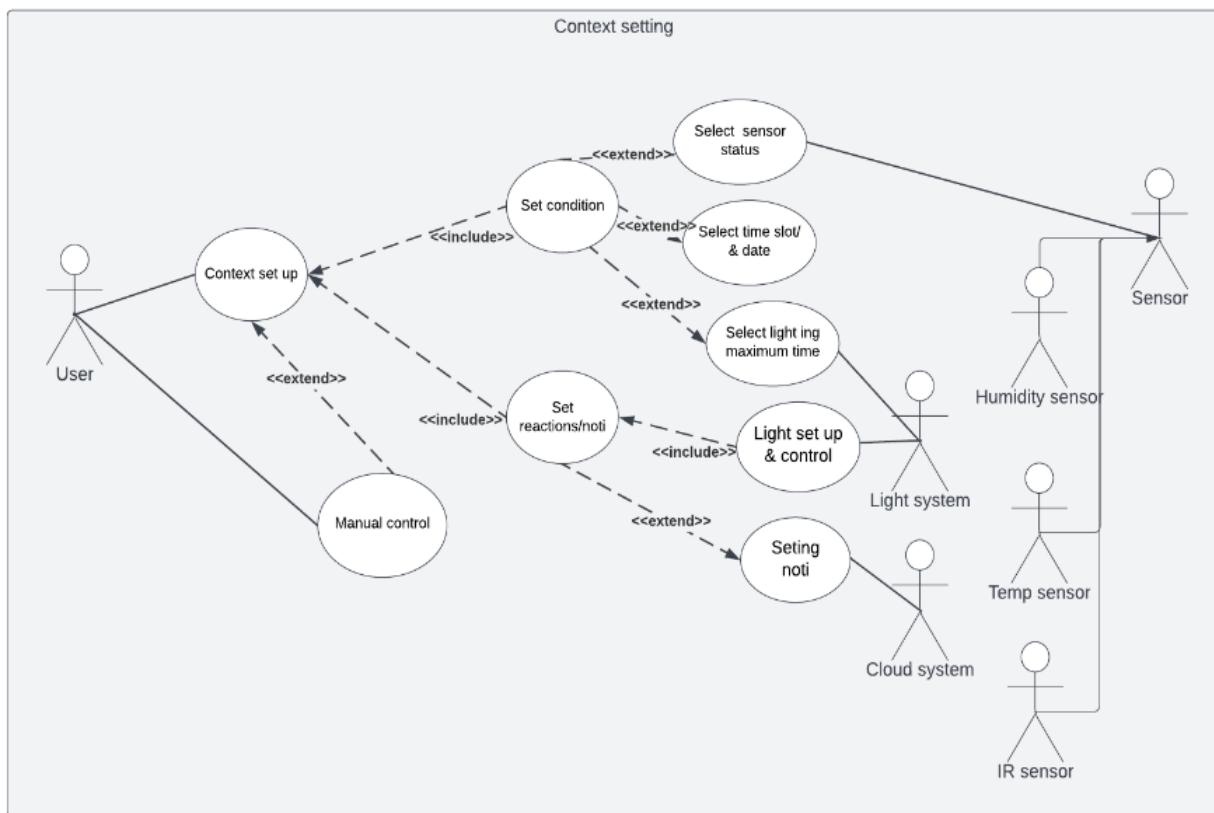
Use-case name	View notifications
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem các thông báo của hệ thống
Trigger	User nhấn vào nút “View notifications” hoặc icon thông báo trên dashboard/ header.
Preconditions	Không
Postconditions	Hệ thống hiển thị các thông báo cho User
Normal flow	1. Người dùng chọn nút “View notifications”. 2. Hệ thống hiển thị menu danh sách các thông báo của hệ thống. 3. Người dùng chọn xem chi tiết thông báo.
Alternative flows	Không
Exceptions	Không

Use-case name	View alerts and warnings
ID	UC-1
Created by	Lê Văn Vy
Actors	User

Description	User muốn xem các cảnh báo của hệ thống
Trigger	User nhấn vào nút “View alerts and warning”
Preconditions	User đã nhấn vào nút “View notifications”
Postconditions	Hệ thống hiển thị các thông kê về các hoạt động cho User
Normal flow	<ol style="list-style-type: none"> 1. Người dùng chọn nút “View alerts and warnings”. 2. Hệ thống hiển thị menu danh sách các cảnh báo.
Alternative flows	Không
Exceptions	Không

Use-case name	View light's status
ID	UC-1
Created by	Lê Văn Vy
Actors	User
Description	User muốn xem các thông kê hoạt động.
Trigger	User nhấn vào nút “View light's status” trên dashboard
Preconditions	User đã nhấn vào nút “View notifications”
Postconditions	Hệ thống hiển thị tình trạng đèn cho User
Normal flow	<ol style="list-style-type: none"> 1. Người dùng chọn nút “View light's status”. 2. Hệ thống hiển thị trạng thái của đèn.
Alternative flows	Không
Exceptions	Không

5.3 Biểu đồ Usecase cho tính năng "Thiết lập ngữ cảnh"



Hình 25: Sơ đồ Usecase cho tính năng "Thiết lập ngữ cảnh"

Use-case name	Context set up
ID	CS-02
Created by	Huỳnh Ngọc Như
Actors	User
Description	User thiết lập các ngữ cảnh để quản lý hệ thống đèn một cách tự động hóa, cảnh báo các trường hợp nguy cấp
Trigger	User nhấp vào mục "Context set up" trên dashboard
Preconditions	User đã login thành công
Postconditions	Hệ thống hoạt động tự động, nhận các tín hiệu từ cảm biến và thực hiện các tác vụ theo ngữ cảnh người dùng đã cài đặt trước hoặc đã chọn. Hệ thống đám mây ghi nhận các dữ liệu, thông báo đến người dùng trong trường hợp cần thiết
Normal flow	<ol style="list-style-type: none"> 1. User nhấp vào mục "Context set up" trên dashboard 2. User chọn các điều kiện cho ngữ cảnh

	<p>3. User chọn các tác vụ tương ứng với ngữ cảnh (thiết lập trạng thái đèn, thông báo khẩn cấp)</p> <p>4. Hệ thống ghi nhận ngữ cảnh, hoạt động tự động với các tín hiệu nhận được từ cảm biến. Hệ thống đám mây ghi nhận dữ liệu, khi cần thiết hệ thống có thể thông báo đến người dùng</p> <p>5. User có thể chọn các ngữ cảnh một cách ko tự động để kích hoạt trạng thái của hệ thống đèn theo ngữ cảnh</p>
Alternative flows	Không
Exceptions	Không

6 Thiết kế UI/UX

Ứng dụng của nhóm được thiết kế gồm có các phần chính như sau:

6.1 Login & Signup

6.1.1 Web App

Sign in

Let's start our journey!

Username

Password

Remember me [Forgot password?](#)

Sign in



Hình 26: Giao diện log in - Web App

Trang login sẽ được hiện ra khi bắt đầu truy cập vào website của ứng dụng.

Người dùng có thể tiến hành đăng nhập vào ứng dụng bằng cách nhập tên tài khoản và mật khẩu. Người dùng có thể nhấn vào nút "Remember me" để lưu tên tài khoản và mật khẩu để tiện lợi hơn cho các lần đăng nhập sau.

Trong trường hợp người dùng chưa có tài khoản để đăng nhập ứng dụng thì người dùng có thể chuyển tới trang đăng ký để đăng ký tài khoản như hình dưới đây.

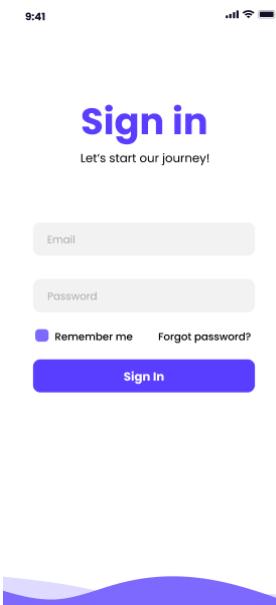


Hình 27: Giao diện Sign up - Web App

Người dùng sẽ nhập các trường thông tin bao gồm họ và tên, số điện thoại, email, username, mật khẩu,... để tiến hành đăng ký mới tài khoản để đăng nhập mật khẩu.

Nếu người dùng đã có tài khoản thì người dùng có thể nhấn vào nút Sign in.

6.1.2 Mobile App



Hình 28: Giao diện Sign in - Mobile App

9:41

Welcome

to our project

Fullname

Phone number

Email

Username

Password

Confirm password

Create Account

Already have an account? [Sign in](#)

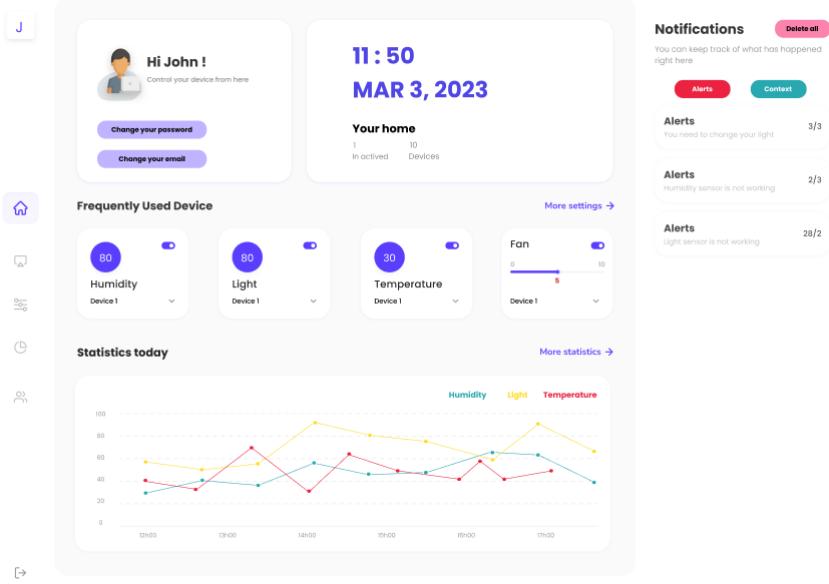


Hình 29: Giao diện Sign up - Mobile App

Mobile App phần Login - Signup được nhóm thiết kế tương tự với Web App.

6.2 Homepage

6.2.1 Web App



Hình 30: Giao diện Homepage - Web App

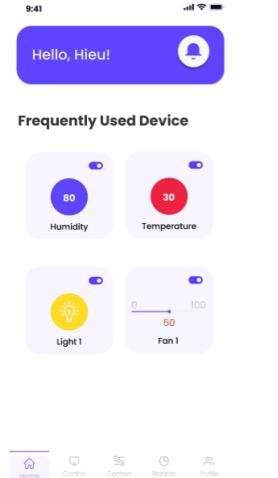
Sau khi người dùng đã đăng nhập thành công, ứng dụng sẽ hiển thị trang Homepage cho người dùng.

Ở góc bên trái sẽ bao gồm các nút tương ứng với các chức năng từ trên xuống gồm Homepage, Manual Control, Context Setup, Statistics, User Detail và cuối cùng là Logout.

Trang Homepage sẽ hiển thị đầy đủ các thông tin dữ liệu thực bao gồm thời gian hiện tại, bảng điều khiển cơ bản cho các thiết bị thường xuyên được sử dụng, các thông báo/cảnh báo từ hệ thống...

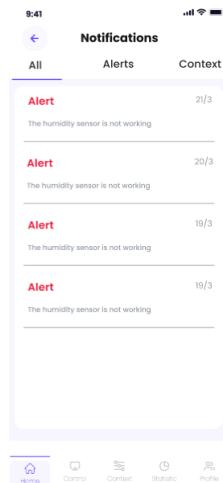
Tương tự với ý tưởng như vậy nhóm thiết kế Homepage cho Mobile App như các hình dưới đây.

6.2.2 Mobile App



Hình 31: Giao diện Homepage - Mobile App

Khi người dùng muốn xem các thông báo/cảnh báo từ hệ thống, người dùng sẽ nhấn vào nút chuông thông báo ở trên trang.



Hình 32: Giao diện Notification của Homepage - Mobile App

6.3 User Detail

Khi người dùng chuyển đến phần User Detail, hệ thống sẽ hiển thị thông tin cá nhân như: Họ và tên, email, số điện thoại, tên tài khoản... Người dùng có thể thay đổi các thông tin và có thể thiết lập thay đổi mật khẩu. Tính năng này được nhóm thiết kế ở Web App và Mobile App bằng các hình dưới đây:

6.3.1 Web App

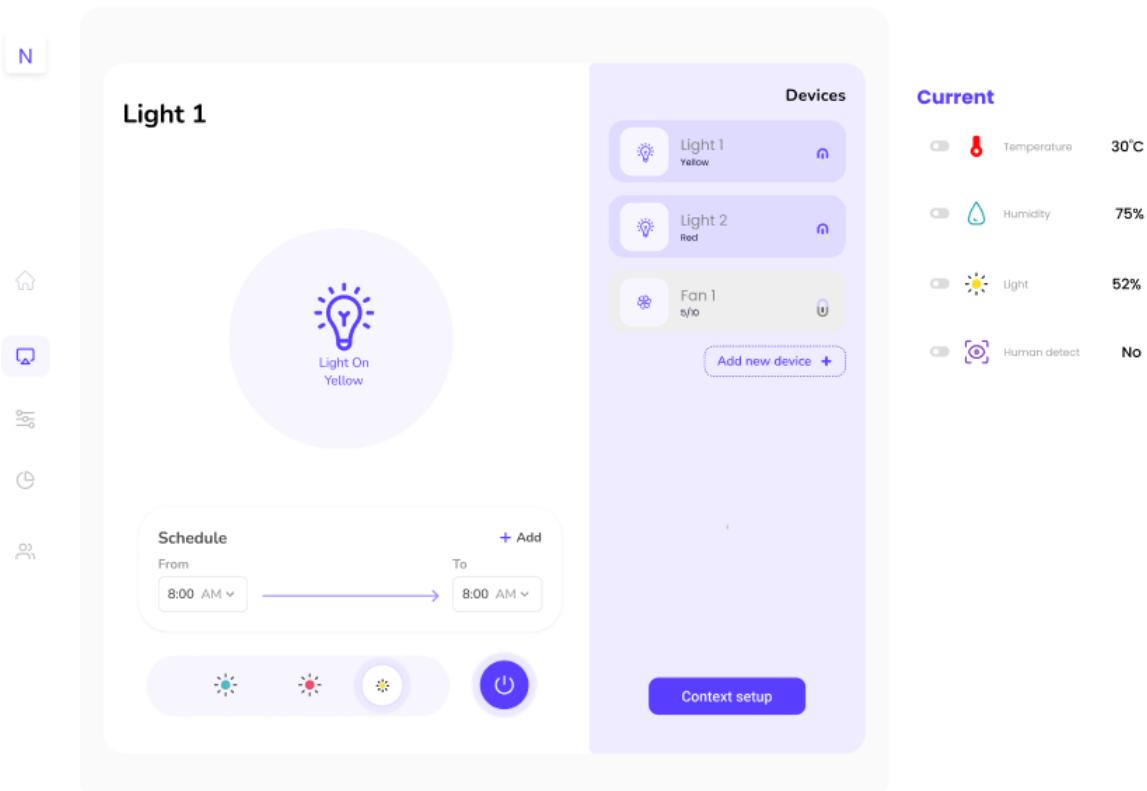
Hình 33: Giao diện User Detail - Web App

6.3.2 Mobile App

Hình 34: Giao diện User Detail - Mobile App

6.4 Manual Control

6.4.1 Web App



Hình 35: Giao diện Manual Control - Web App

Khi người dùng chuyển đến phần Manual Control, người dùng có thể điều khiển chi tiết từng thiết bị hay bật tắt các cảm biến.

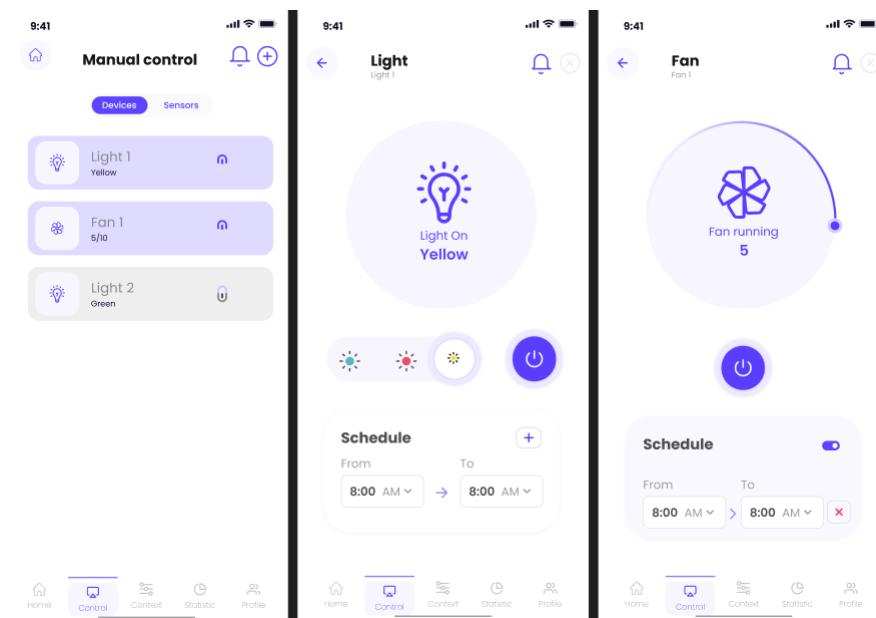
Ngoài ra, người dùng có thể đặt lịch bật tắt các thiết bị. Chẳng hạn người dùng có thể setup bật đèn chuyển sang màu vàng trong khoảng thời gian từ 8h sáng đến 10h sáng. Hệ thống sẽ lưu lại lịch và tiến hành điều chỉnh đèn khi đến giờ được cài đặt.

6.4.2 Mobile App

Mobile App cũng sẽ được thiết kế tương tự như với Web App.

Ban đầu khi người dùng chuyển đến phần Manual Control thì hệ thống sẽ hiển thị danh sách lịch thiết lập các thiết bị như hình bên trái dưới đây.

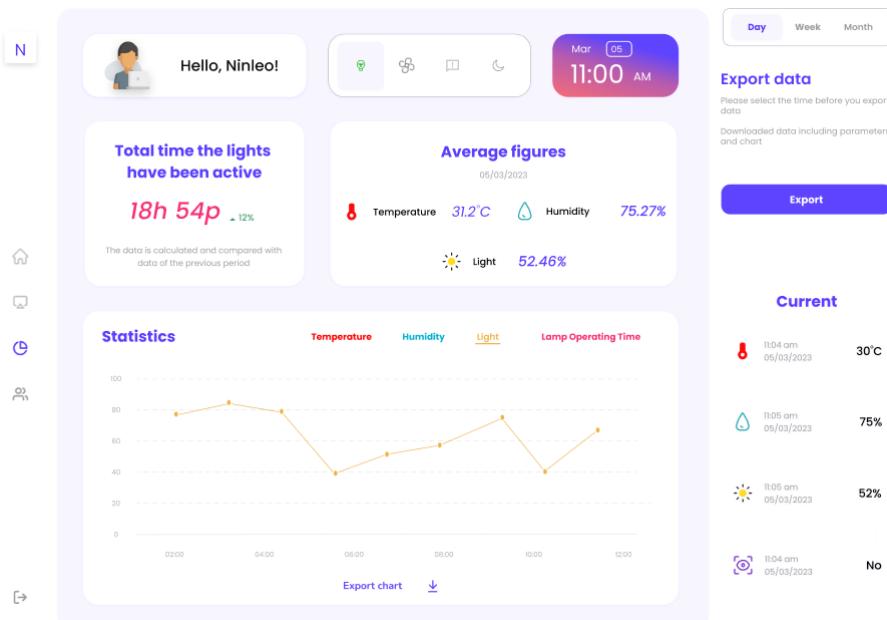
Người dùng có thể chọn đặt lịch hoặc điều khiển chi tiết từng thiết bị. Giao diện sẽ hiển thị như ở hai trang tiếp theo.



Hình 36: Giao diện Manual Control - Mobile App

6.5 Statistics

6.5.1 Web App



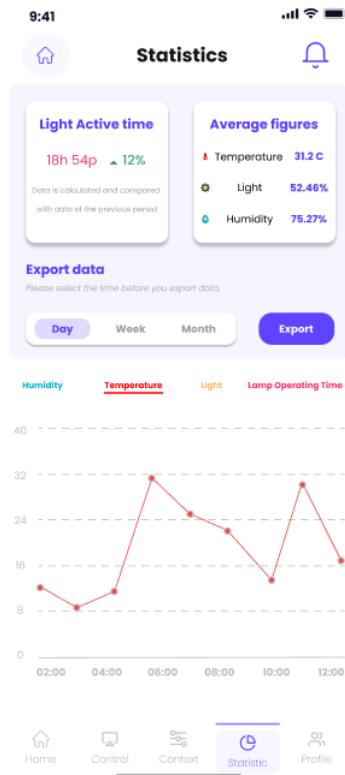
Hình 37: Giao diện Statistics - Web App

Người dùng có thể xem được chi tiết thống kê về nhiệt độ không khí, độ ẩm, cường độ ánh sáng và thời gian đèn ở chế độ bật tắt. Ngoài ra hệ thống cũng sẽ tính toán dữ liệu và hiển thị cho người dùng các kết quả tính toán trung bình hay tổng thời gian đèn được bật.

Trong trang này người dùng vẫn có thể xem được các dữ liệu thời gian thực quan trọng ở phần Current cũng như phát hiện có thông báo/cảnh báo quan trọng.

Cuối cùng người dùng cũng có thể tải về các dữ liệu nhờ vào chức năng Export data.

6.5.2 Mobile App



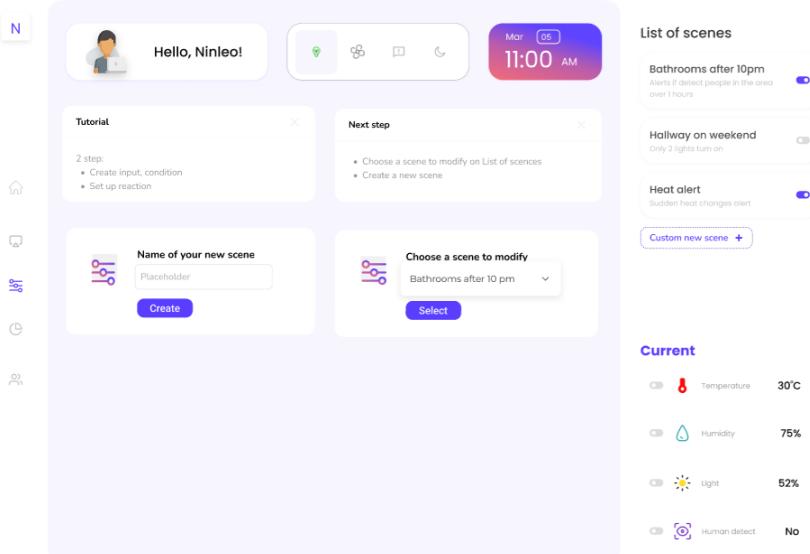
Hình 38: Giao diện Statistics - Mobile App

6.6 Context Setup

Dây là tính năng phức tạp nhất mà nhóm sẽ thực hiện. Chức năng chính là người dùng sẽ tạo một kịch bản để thiết lập các giá trị ngưỡng cảnh báo xảy ra và điều khiển các thiết bị trong một khoảng thời gian.

6.6.1 Web App

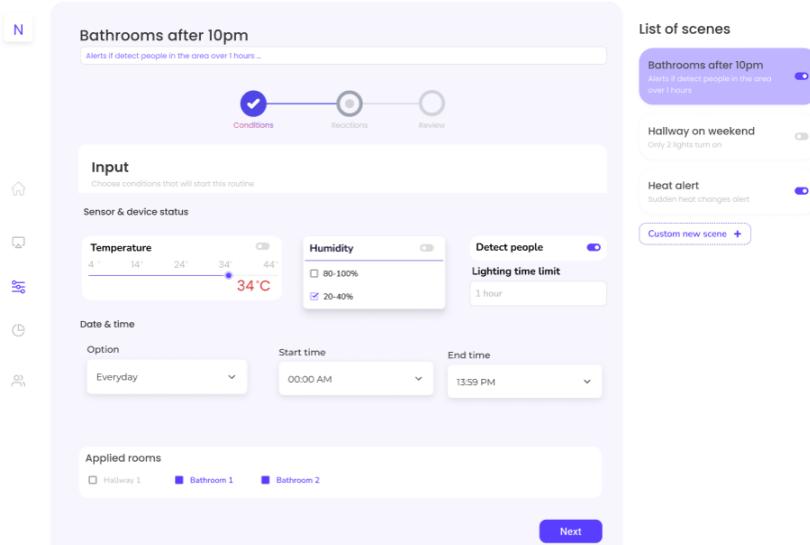
Khi người dùng chuyển đến phần Context Setup, giao diện sẽ hiển thị ban đầu bao gồm hướng dẫn sử dụng tính năng Context Setup và các lựa chọn gồm tạo một kịch bản mới hay chọn kịch bản để sửa đổi. Ngoài ra người dùng còn sẽ được thấy danh sách các kịch bản được bật hay tắt.



Hình 39: Giao diện Context Setup trang chính - Web App

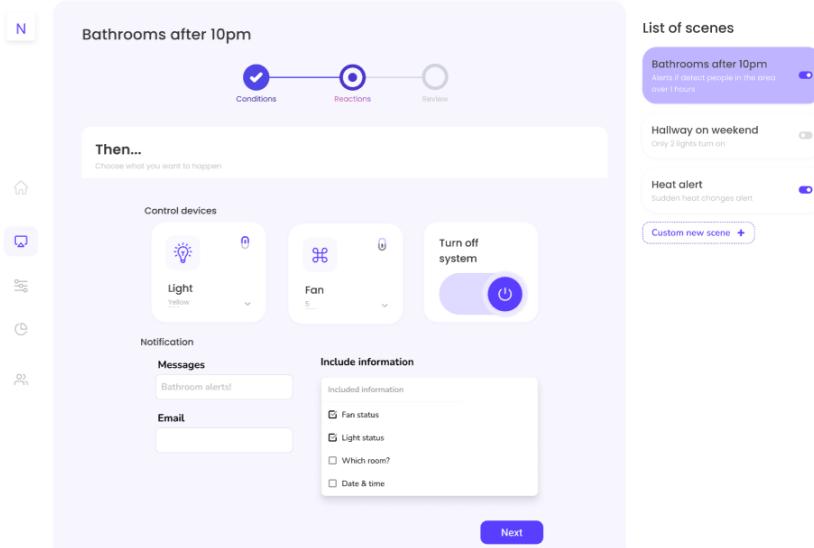
Khi người dùng lựa chọn tạo một kịch bản hay sửa đổi một kịch bản có sẵn thì đều phải tuân theo các bước sau:

- Dặt và chỉnh sửa tên kịch bản. Thiết lập các ngưỡng trên và dưới của các thông số nhiệt độ không khí, độ ẩm và cường độ ánh sáng. Người dùng có thể lựa chọn thiết lập hoặc không thiết lập các chế độ này.
- Tiếp theo người dùng có thể lựa chọn tắt hoặc bật chế độ cảnh báo hiện người.
- Người dùng có thể lựa chọn các quy tắc thời gian như Start time, End time, chế độ lặp lại...



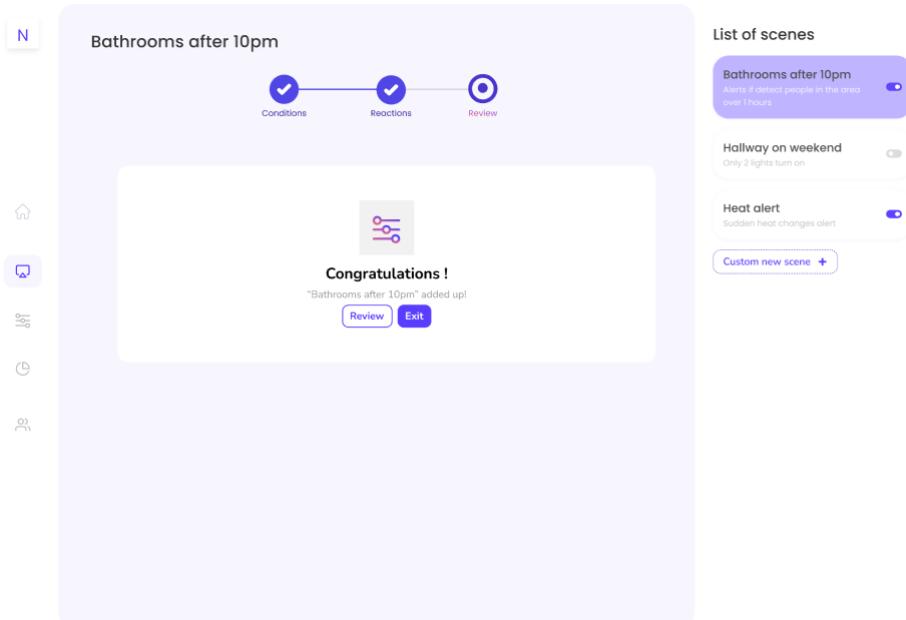
Hình 40: Giao diện cấu hình kịch bản Context Setup bước 1 - Web App

- Người dùng có thể thiết lập điều khiển thiết bị đèn hay quạt nếu như xảy ra tình trạng giá trị vượt ngưỡng cho phép hay hệ thống phát hiện người trong khoảng thời gian hoạt động của kịch bản.
- Cuối cùng người dùng có thể gửi message cảnh báo các thông tin cho email mà người dùng sẽ nhập.



Hình 41: Giao diện cấu hình kịch bản Context Setup bước 2 - Web App

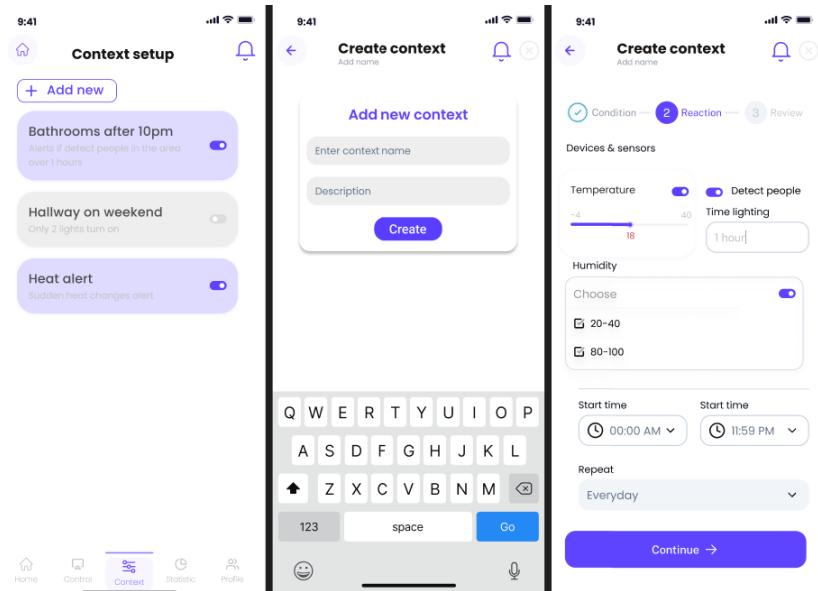
Khi người dùng đã hoàn thành xong các bước trên, hệ thống sẽ hiển thị giao diện cho biết cấu hình kịch bản Context Setup đã được cấu hình thành công, người dùng có thể review lại kịch bản vừa cấu hình hoặc nhấn nút **Exit** để trở về giao diện chính trang Context Setup.



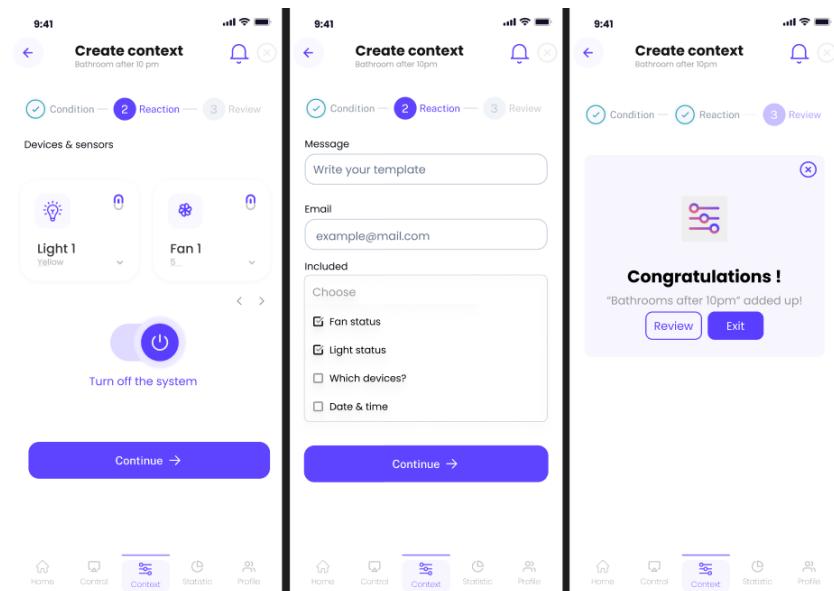
Hình 42: Giao diện hoàn thành kịch bản Context Setup - Web App

6.6.2 Mobile App

Giao diện ở Mobile App sẽ bao gồm các trang như ở các hình dưới đây



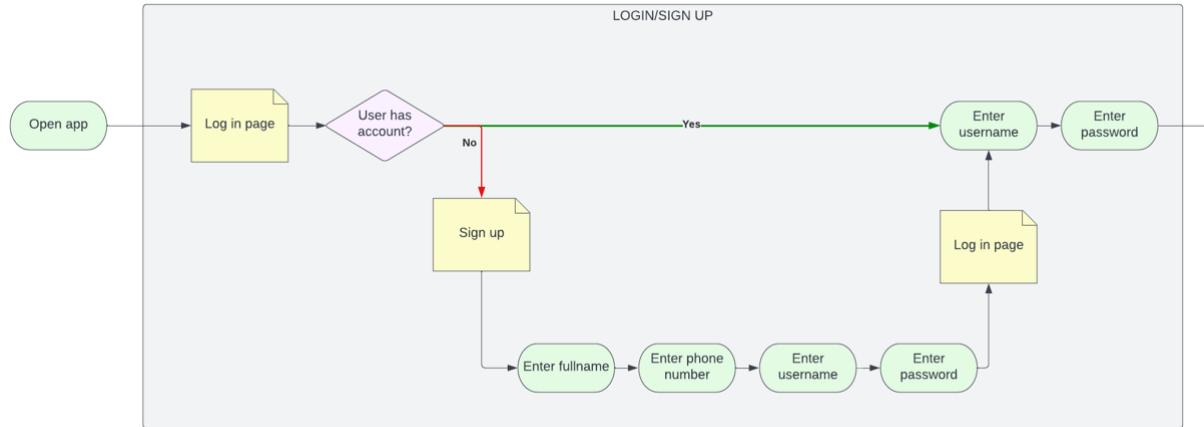
Hình 43: Các giao diện Context Setup phần 1 - Mobile App



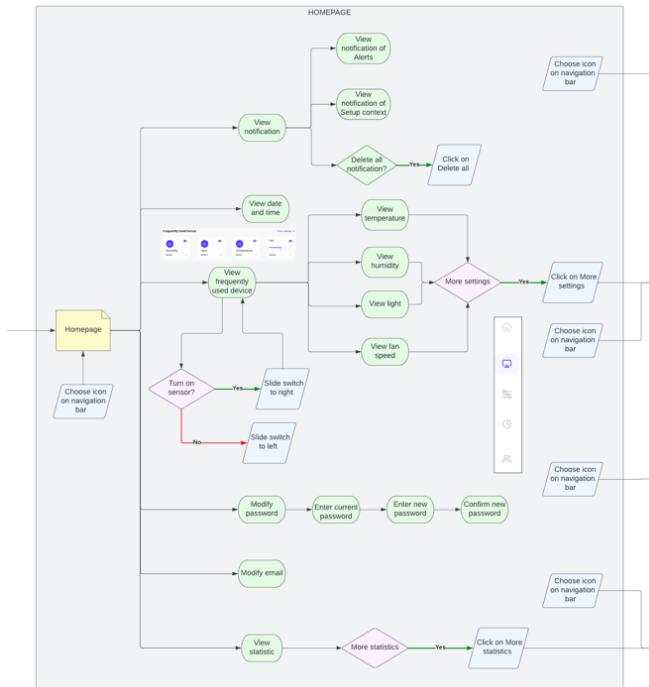
Hình 44: Các giao diện Context Setup phần 2 - Mobile App

6.7 User Flow

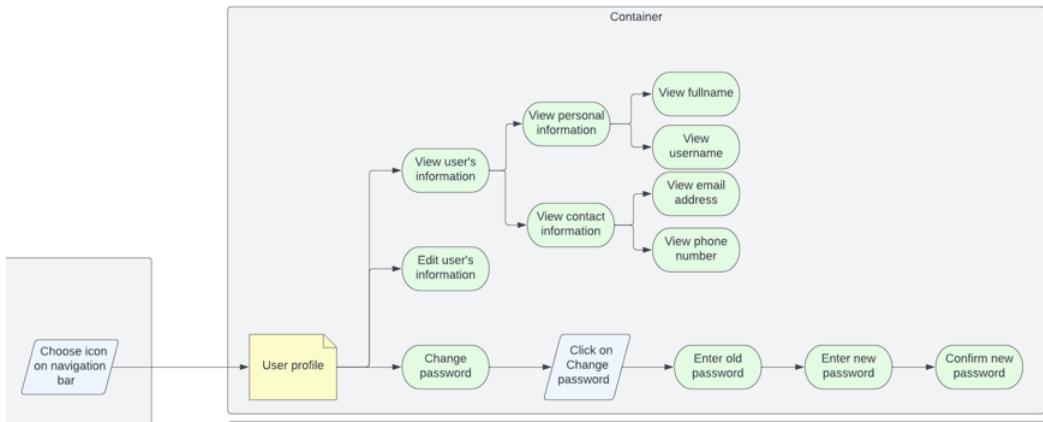
Để tổng hợp lại các tính năng thiết kế vừa được trình bày ở các phần trên cùng với khái quát lại luồng thực thi của người dùng, nhóm đã hiện thực lại sơ đồ User Flow như các hình sau:



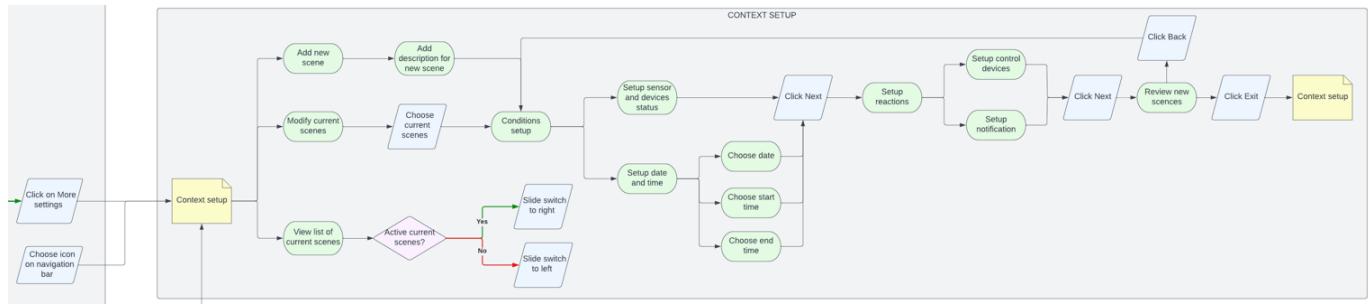
Hình 45: User Flow phần Login/Sign Up



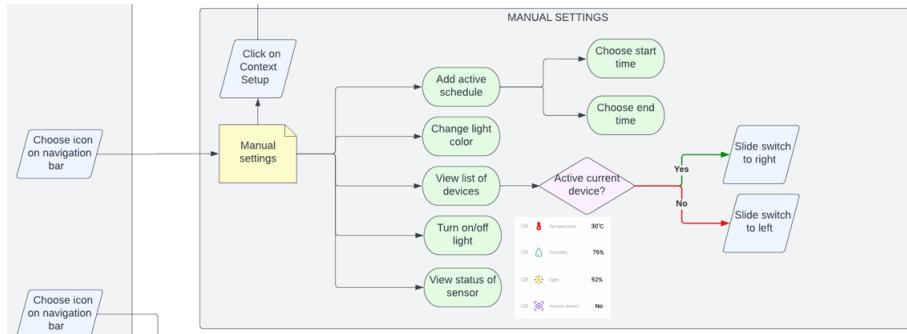
Hình 46: User Flow phần Homepage



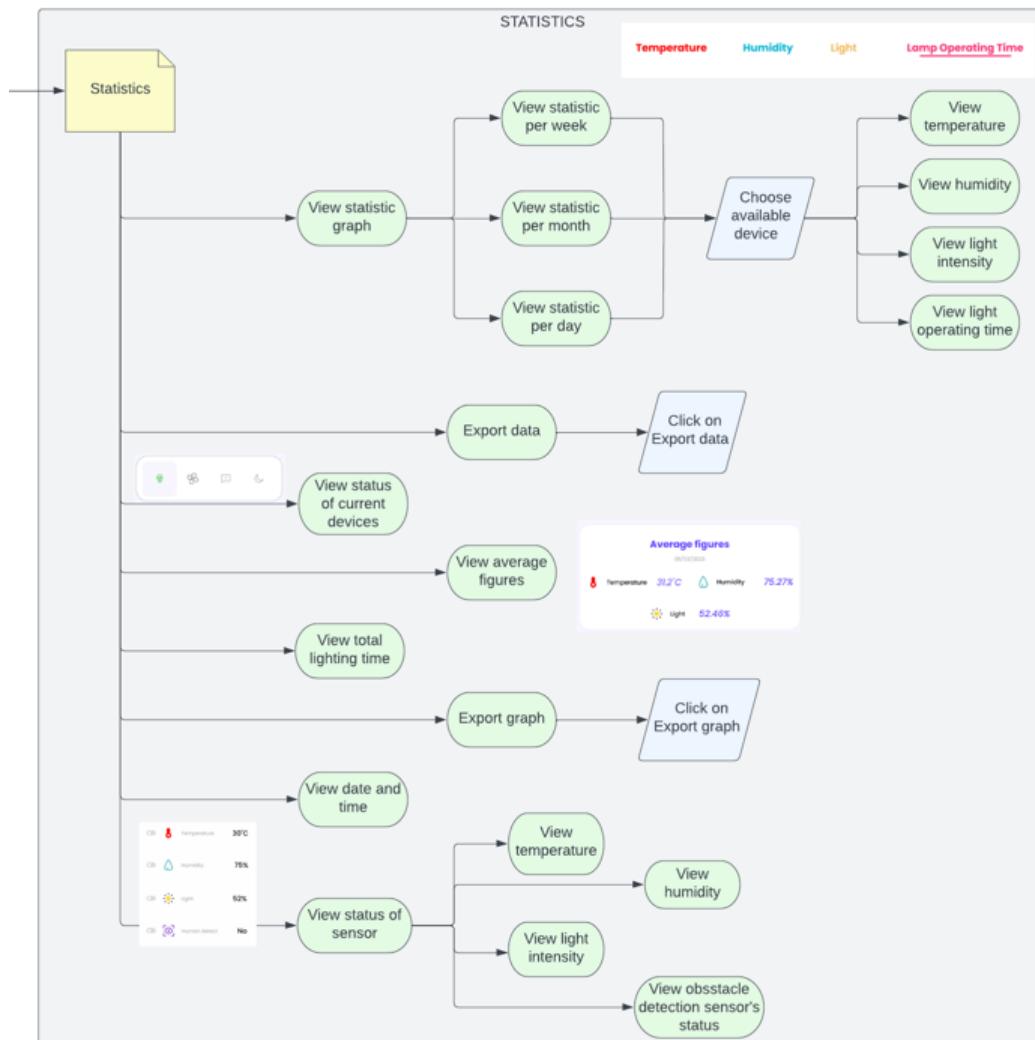
Hình 47: User Flow phần Container



Hình 48: User Flow phần Context Setup



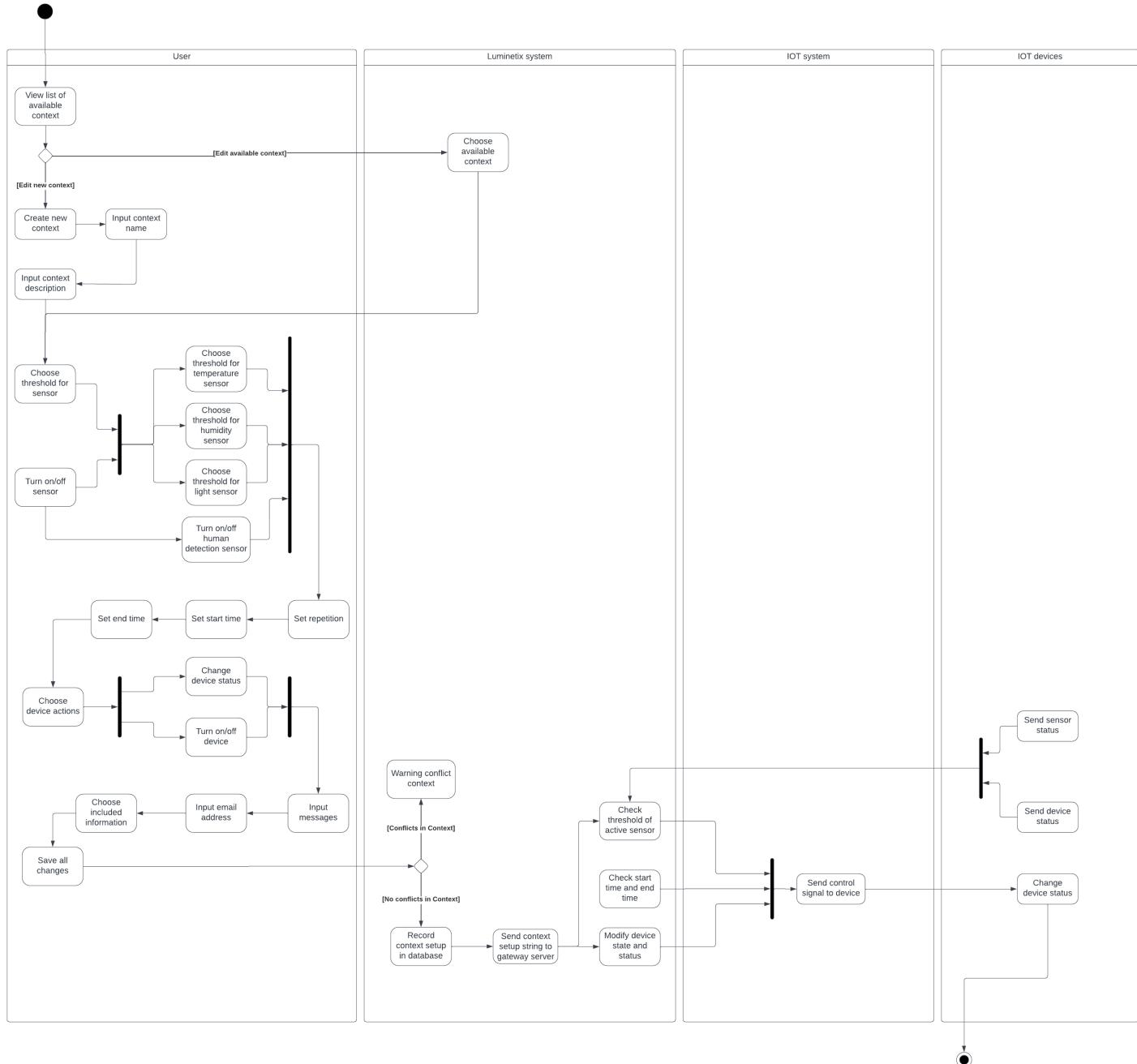
Hình 49: User Flow phần Manual Settings



Hình 50: User Flow phần Statistics

7 Activity Diagram: Context Setup

Trong phần Activity Diagram nhóm chỉ giới hạn lại ở việc vẽ diagram với tính năng Context Setup do các phần khác thường sẽ là những tính năng lựa chọn đơn giản hay quy trình chỉ bao gồm 1-3 bước.



Hình 51: Activity Diagram - Context Setup

Như đã nói ở phần User Flow, tính năng Context Setup là tính năng phức tạp nhất và yêu cầu quy trình thực hiện thao tác nhiều bước từ người dùng. Cho nên ngoài User Flow thì nhóm hiện thực thêm phần Activity Diagram cho thao tác này.

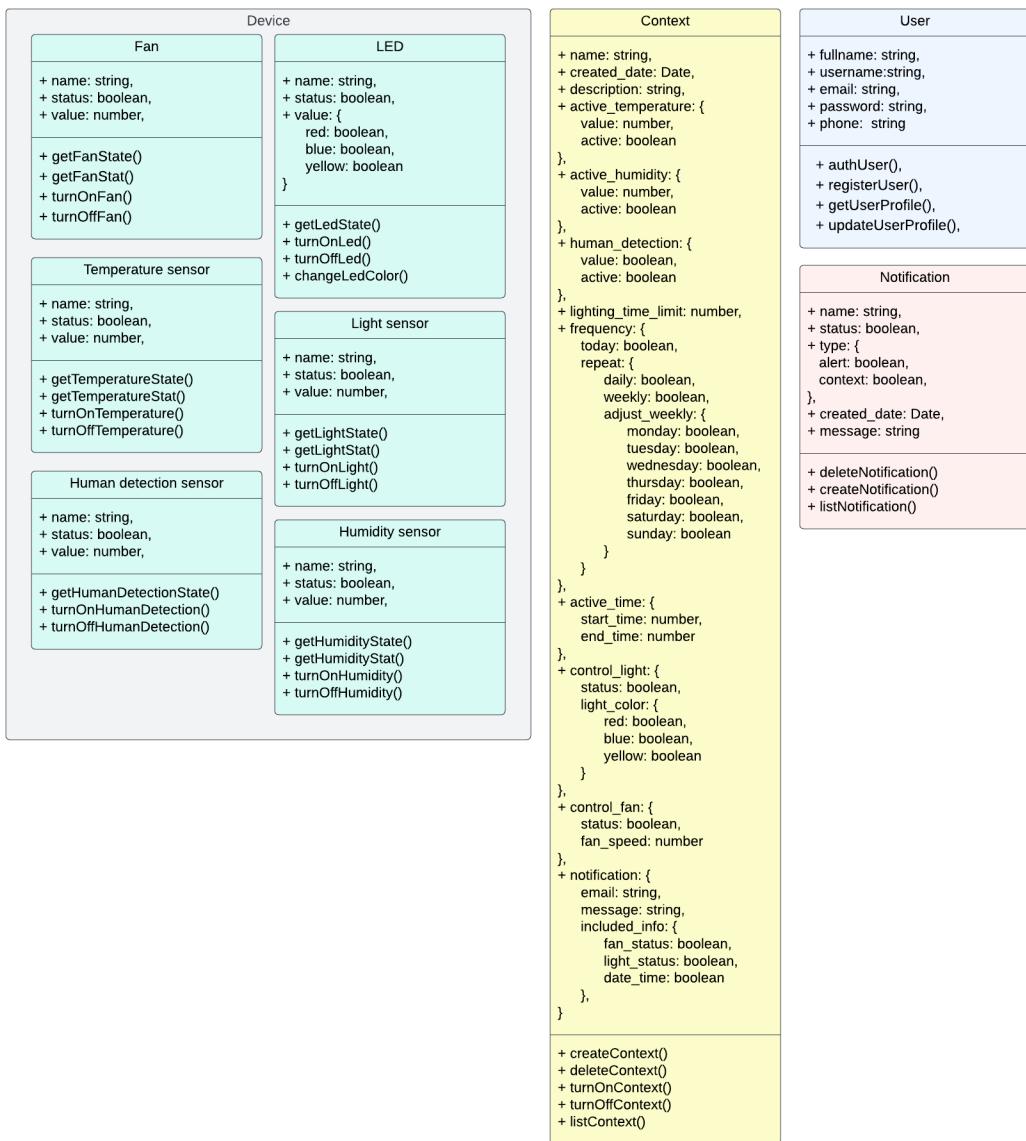
Cũng như đã trình bày, thì quy trình người dùng thêm một kịch bản context setup hay chỉnh sửa một context setup sẵn có sẽ tương tự nhau.

Sau khi người dùng đã thao tác tạo kịch bản Context Setup xong, thì ở hệ thống app sẽ xác định xem kịch bản có hợp lệ hay không.

Nếu như kịch bản không hợp lệ thì app sẽ cảnh báo người dùng kịch bản không hợp lệ và người dùng sẽ phải chỉnh sửa lại kịch bản. Còn nếu hợp lệ thì app sẽ trước hết kiểm tra nhận dữ liệu từ các thiết bị cảm biến bằng cách gửi các lệnh điều khiển bật nhận dữ liệu cảm biến xuống IoT Server. IoT Server nhận lệnh sẽ tiến hành điều khiển các thiết bị cảm biến.

Khi đến thời gian đã cài đặt trong Context setup, nếu xảy ra tình trạng báo động như vượt ngưỡng các giá trị cho phép hay phát hiện người trong thời gian giới nghiêm thì hệ thống app sẽ gửi các lệnh cảnh báo xuống hệ thống phần cứng IoT system. Ở IoT system sẽ tiếp nhận các lệnh và điều khiển các thiết bị ở Node thực hiện chức năng cảnh báo như bật đèn, bật quạt hay phát loa cảnh báo.

8 Thiết kế Database

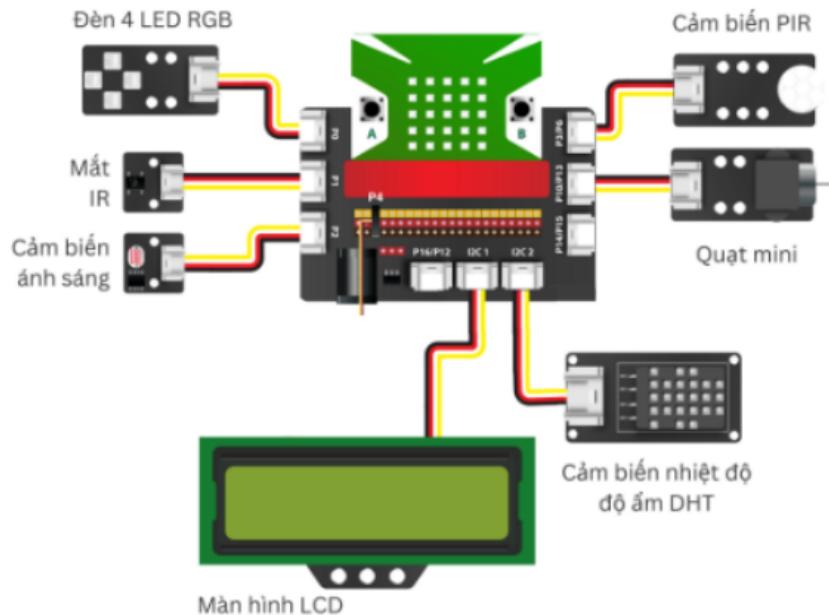


Đây là database schema của project, gồm có các models như: Context, User, Notification, Fan, LED, Temperature Sensor, Light sensor, Human detection sensor, Humidity sensor. Mỗi model quản lý các trường dữ liệu cần lưu trữ và các phương thức giao tiếp với model đó.

9 Hiệu thực phần cứng

9.1 Hiệu thực Node

9.1.1 Sơ đồ dự kiến kết nối các thiết bị phần cứng



Hình 52: Sơ đồ kết nối các thiết bị phần cứng

Chân	Tên các thiết bị kết nối
P0	Đèn 4 LED RGB
P1	Mắt IR đọc tín hiệu Remote
P2	Cảm biến ánh sáng
P16 (hoặc P3)	Cảm biến PIR hồng ngoại
P10.P13	Quạt mini
P19.P20 - I2C1	Màn hình LCD
P19.P20 - I2C2	Cảm biến DHT20

Vì có thể xảy ra các sự cố khách quan như mạch bị nhiễu hay chân cảm biến hư nên trừ các chân I2C nối với LED LCD, DHT20 thì các chân còn lại có thể thay đổi kết nối linh hoạt giữa các thiết bị với nhau. Tuy nhiên nếu có thay đổi thì tại chương trình chính Yolo:Bit ta cần phải thay đổi code sao cho đồng bộ quá trình kết nối.

9.1.2 Hiệu thực chương trình

Nhóm hiện thực chương trình chính để nạp vào Yolo:Bit để tiến hành hiện thực một số tính năng sau:

- Tiến hành kết nối Yolo:Bit với Wifi, Gateway, reset các giá trị ban đầu, reset LED LCD.
- Gửi các dữ liệu định kỳ (khoảng 20 giây) của các cảm biến nhiệt độ, độ ẩm DHT20, cảm biến ánh sáng lên Gateway.
- Hiển thị các thông tin dữ liệu cập nhật ra LED LCD. Các thông tin được hiển thị bao gồm: nhiệt độ, độ ẩm không khí, ánh sáng và ngày tháng hiện thời.
- Dùng loa được tích hợp sẵn trên Yolo:Bit báo hiệu nếu giá trị nhiệt độ vượt ngưỡng mức nguy hiểm. Dựa vào các dữ liệu nhiệt độ hoạt động của các thành phần phần cứng cùng với mức độ chịu nhiệt của con người, nhóm đặt cỗ định nhiệt độ hoạt động của hệ thống thuộc khoảng từ 5-45 độ C. Nếu quá ngưỡng này thì hệ thống sẽ tự động phát loa cảnh báo cho người dùng biết.
- Khi cảm biến hồng ngoại phát hiện người, gửi tín hiệu 1 hiển thị ở Yolo:Bit, ngược lại sẽ gửi tín hiệu 0. Đồng thời cứ mỗi 10 giây sẽ gửi dữ liệu phát hiện người lên Gateway.
- Xử lý dữ liệu nhận được từ Gateway. Các trường hợp dữ liệu nhận được:
 - Nhận dữ liệu bật tắt nhận dữ liệu từ các cảm biến nhiệt độ không khí, độ ẩm, cảm biến ánh sáng.
 - Nhận dữ liệu điều khiển tốc độ quạt.
 - Nhận dữ liệu điều khiển đèn.
 - Nhận dữ liệu cảnh báo để phát loa cảnh báo trên phần cứng.
- Nhận dữ liệu từ remote điều khiển từ xa, dữ liệu nhận được từ chương trình AI nhận diện giọng nói người dùng để thay đổi bật tắt đèn, thay đổi tốc độ quạt theo các mức tự định sẵn, tắt âm thanh cảnh báo.

Ngoài chương trình này ra, nhóm còn hiện thực chương trình AI nhận diện giọng nói để kết nối vào Yolo:Bit. Chương trình này sẽ xử lý giọng nói từ người dùng để gửi lệnh đến chương trình chính nạp vào Yolo:Bit thông qua kết nối Bluetooth. Các lệnh xử lý trong chương trình bao gồm:

- Nhận diện giọng nói bật tắt đèn.
- Nhận diện giọng nói bật tắt và điều khiển tốc độ quạt theo 4 mức: Mức 1 (25%), mức 2 (50%), mức 3 (75%), mức 4 (100% - mạnh nhất).
- Nhận diện giọng nói tắt loa cảnh báo.

9.2 Hiện thực Gateway

Nhóm sử dụng **Python** để hiện thực một Gateway đơn giản, đóng vai trò trung tâm kết nối giữa Yolobit với Server. Khi đó Yolobit sẽ được nhóm hiện thực lại để đóng vai trò chính là một node trung tâm tổng hợp dữ liệu từ các cảm biến và xử lý lệnh được nhận; kết nối với Gateway theo phương thức Serial.

9.2.1 Kết nối với Yolo:Bit

Gateway hiện thực bằng Python sẽ được tích hợp việc kết nối với Yolobit thông qua *cổng COM ảo USB*.

Như đã trình bày ở các phần trước, mạch Yolobit kết nối với máy tính sẽ được gọi là mạch Yolobit trung tâm.

Cách tiếp cận này có một số lợi ích được liệt kê như sau:

- Gateway đang được hiện thực trên một máy tính. Trong tương lai, quy trình này sẽ được mang xuống máy tính nhúng. Đặc điểm chung của mọi hệ thống máy tính, đó là hệ thống thông dụng nhưng không chuyên dụng. Việc kết nối với một thiết bị chuyên dụng sẽ phải thông qua kết nối USB.
- Việc kết nối thêm thiết bị chuyên dụng, cung cấp tính năng mở rộng gần như vô hạn cho hệ thống máy tính. Trong trường hợp này, chúng ta muốn mở rộng khả năng giao tiếp không dây của mạch Yolobit cho Gateway IoT trên máy tính.

Việc hiện thực lập trình cho Gateway sẽ bao gồm các phần sau:

- Thêm thư viện lập trình kết nối Serial.
- Hiện thực các hàm nhận dữ liệu kết nối cổng COM ảo. Cổng USB kết nối sẽ có tên theo dạng USB-SERIAL CH340.
- Nhận và xử lý dữ liệu từ Yolobit.



Hình 53: Cấu trúc dữ liệu từ Yolo:Bit gửi lên Gateway

Dữ liệu từ YoloBit gửi lên Gateway sẽ có cấu trúc như hình trên. Cấu trúc ở trên là một dạng kinh điển trong giao tiếp máy tính (Computer Communications). Chúng ta sẽ quy định ký tự bắt đầu (SOC = Start Of Character) là "!" và ký tự kết thúc (EOC = End Of Character) là "#". Trường thông tin ở giữa, chúng ta sẽ định nghĩa là ID:FIELD:VALUE, trong đó ID là định danh của một node, FIELD là thông tin định nghĩa và VALUE là giá trị cần gửi. Các trường này cách nhau bằng ":" - thuật ngữ chuyên ngành gọi là Escape Character.

Sau khi đã nhận được dữ liệu từ Yolobit, ta cần lập trình hàm đọc và phân tách gói dữ liệu nhận được để gửi vào đúng kênh dữ liệu trên Server.

Gửi gói dữ liệu lệnh từ Server xuống Yolobit. Gói dữ liệu lệnh này thường đơn giản và có thể xử lý ở phần lập trình hiện thực Yolobit để điều khiển các thiết bị.

9.2.2 Kết nối với Adafruit IO Server

Gateway kết nối với Adafruit Server thông qua cơ chế giao tiếp **MQTT Pub/Sub**. Trước hết, Gateway cần phải được kết nối với server thông qua các trường username và key. Để có thể nhận được lệnh từ các kênh dữ liệu của Adafruit Server, Gateway cần phải đăng ký (subscribe) các kênh dữ liệu đó thông qua các Feed ID. Còn khi Gateway muốn gửi dữ liệu lên server, ta chỉ cần hiện thực publish các kênh dữ liệu..

Lưu ý: Để đơn giản hóa việc lập trình từ các thiết bị IoT gửi lên feed, chúng ta sẽ cấu hình cho nó là dạng Public.

Việc hiện thực lập trình cho Gateway trong phần này sẽ bao gồm các bước sau:

- Thêm thư viện lập trình kết nối Adafruit
- Diền các thông tin để kết nối với Server
- Tạo một đối tượng client kết nối MQTT
- Thiết lập các hàm để kết nối truyền nhận dữ liệu với server theo cơ chế publish-subscribe

9.3 Hiện thực Adafruit IO Server

Nhóm quyết định sử dụng Adafruit IO Server là trung tâm nhận, xử lý dữ liệu và gửi các gói lệnh điều khiển. Việc kết nối với server này cần phụ thuộc vào các yếu tố:

- **Username:** Tên tài khoản sử dụng server.
- **Key:** Khóa riêng ứng với mỗi tài khoản. Điều này giúp tăng tính bảo mật của server cũng như đảm bảo khả năng truyền nhận dữ liệu an toàn và không bị xảy ra tình trạng nhiễu loạn.
- **Feed:** Cũng như key, mỗi kênh dữ liệu hay mỗi Dashboard được thiết lập sẽ có một mã ID để thiết lập kết nối trong quá trình lập trình hiện thực.

Một ưu điểm của Adafruit là người quản lý có thể xem các dữ liệu thô của từng kênh dữ liệu, từ đó có cái nhìn chi tiết hơn và đánh giá dựa trên dữ liệu được nhận chính xác hơn.

Ngoài ra Adafruit cũng có giao diện Dashboard thân thiện, dễ sử dụng và thiết lập chi tiết, có nhiều lựa chọn thiết kế cho người dùng.

Các Feeds hay Dashboards đều có cơ chế bảo mật riêng tư. Người dùng có thể thiết lập private hay public.

Giao diện Dashboard được nhóm thiết kế gồm các khối tương ứng với các kênh Feed sau:

9.3.1 Nhóm các khối điều khiển thiết bị, giám sát các trạng thái



Hình 54: Giao diện Dashboard Adafruit IO - Phần điều khiển thiết bị, giám sát các trạng thái

- *Khối điều khiển đèn LED - Tương ứng với kênh feed LED:* Có chế độ thông thường là bật và tắt. Ngoài ra còn có chế độ có thể đổi bất kì màu đèn nào. Dữ liệu được lưu vào kênh có dạng mã hex màu đèn (VD: #ffffff - màu trắng).
- *Khối điều khiển quạt - Tương ứng với kênh feed FAN:* Có chức năng điều khiển quạt với dữ liệu được lưu vào kênh là giá trị trong khoảng từ 0-100.
- *Khối điều khiển bật tắt nhận dữ liệu cảm biến nhiệt độ không khí - Tương ứng với kênh feed S-TEMP:* Có chức năng điều khiển bật tắt nhận dữ liệu nhiệt độ không khí từ Node với 2 loại giá trị được lưu vào kênh tương ứng với hai chế độ: T-ON (bật) và T-OFF (tắt).
- *Khối điều khiển bật tắt nhận dữ liệu cảm biến độ ẩm không khí - Tương ứng với kênh feed S-HUMI:* Hoạt động tương tự với phần nhiệt độ với 2 loại giá trị được lưu vào kênh tương ứng với hai chế độ: H-ON (bật) và H-OFF (tắt).
- *Khối điều khiển bật tắt nhận dữ liệu cảm biến nhiệt độ - Tương ứng với kênh feed S-LIGHT:* Hoạt động tương tự với 2 loại giá trị được lưu vào kênh tương ứng với hai chế độ: L-ON (bật) và L-OFF (tắt).
- *Khối điều khiển giám sát phát hiện người - Tương ứng với kênh feed HUMAN:* Giám sát tình trạng phát hiện người với 2 loại giá trị được lưu vào trong kênh: 1 (có người) và 0 (không có người).
- *Khối điều khiển giám sát cảnh báo - Tương ứng với kênh feed ALERT:* Giám sát việc phát cảnh báo cho hệ thống phần cứng với 2 loại giá trị được lưu: ALERT (bật cảnh báo) và ALERT-OFF (tắt cảnh báo).

9.3.2 Nhóm các khối quan sát các thông số dữ liệu

Ở nhóm này có 3 khối dữ liệu tương ứng với 3 kênh feed có chức năng tương đồng với nhau: Quan sát giá trị thực và xem thống kê dữ liệu thu thập được. Giá trị được lưu vào các kênh chính là giá trị dữ liệu thu thập được từ các cảm biến ở Node bao gồm: nhiệt độ, độ ẩm không khí, cường độ ánh sáng.

- *Khối quan sát giá trị dữ liệu nhiệt độ - Tương ứng với kênh feed TEMP.*
- *Khối quan sát giá trị dữ liệu độ ẩm không khí - Tương ứng với kênh feed HUMI.*
- *Khối quan sát giá trị dữ liệu cường độ ánh sáng - Tương ứng với kênh feed LIGHT.*



Hình 55: Giao diện Dashboard Adafruit IO - Phần quan sát các thông số dữ liệu

Default			
Feed Name	Key	Last value	Recorded
W_ALERT	w-alert	ALERT	about 9 hours ago
W_FAN	w-fan	0	28 minutes ago
W_HUMAN	w-human	0	about 8 hours ago
W_HUMI	w-humi	46.9	about 8 hours ago
W_LED	w-led	#000000	40 minutes ago
W_LIGHT	w-light	0	about 8 hours ago
W_S_HUMI	w-s-humi	H_ON	about 2 hours ago
W_S_LIGHT	w-s-light	L_ON	about 8 hours ago
W_S_TEMP	w-s-temp	T_ON	about 5 hours ago
W_TEMP	w-temp	26.7	about 8 hours ago

Hình 56: Danh sách các kênh Feeds trong server Adafruit IO

10 Hiện thực Web & App - Design Pattern

10.1 Kiến trúc MVC (MVC pattern)

Nhóm lựa chọn kiến trúc MVC (Model-View-Controller) để hiện thực riêng phần app và web cho hệ thống **Mô tả về kiến trúc MVC**

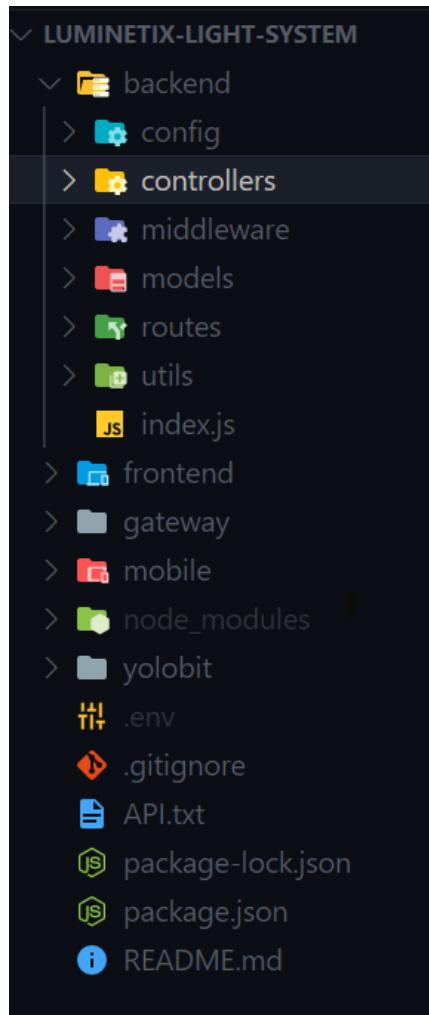
Model - View - Controller (MCV) là một kiến trúc thiết kế thường được ứng dụng trong phát triển website hoặc ứng dụng di động. Các hệ thống với kiến trúc MCV thường được chia thành 3 phần kết nối với nhau, mỗi phần có một nhiệm vụ riêng, độc lập. 3 phần trong kiến trúc MVC:

1. Model: bao gồm các dữ liệu và logic liên quan đến xử lý dữ liệu.
2. View: hiển thị dữ liệu đến người dùng hoặc xử lý tương tác người dùng.
3. Controller: Điều khiển sự tương tác của hai thành phần Model và View.

Chi tiết hơn về từng bộ phận, ta có:

1. Model: là bộ phận lưu trữ toàn bộ dữ liệu và cung cấp các logic, phương thức để tương tác với dữ liệu. Đối với hệ thống Luminetix, thành phần model:
 - Lưu trữ các dữ liệu về thông tin người dùng, chỉ số của các cảm biến, các thiết bị đèn và quạt, các ngữ cảnh (context)
 - Cung cấp các phương thức để xử lý dữ liệu và tương tác với hệ cơ sở dữ liệu như việc đăng ký, đăng nhập tài khoản người dùng, bật tắt cảm biến và điều khiển các thiết bị, tạo mới các ngữ cảnh và điều chỉnh các ngữ cảnh.
 - Thông báo các thay đổi về dữ liệu để view refresh và hiển thị bản cập nhật.
2. View: Giao diện tương tác với người dùng, trình bày các dữ liệu tới người dùng.
 - Trình bày các yếu tố giao diện của hệ thống: bar, form, button...
 - Ghi nhận và truyền tải các tương tác của người dùng:
 - Ví dụ người dùng click vào dropdown menu => hiển thị menu.
 - Gửi các sự kiện về cho controller: ví dụ khi người dùng click submit form, view gửi thông báo đến controller.
 - Truyền các dữ liệu người dùng nhập vào: email, mật khẩu, tên đăng nhập cho controller.
 - Khi được thông báo các thay đổi trạng thái từ model, thực hiện truy xuất dữ liệu từ model để thực hiện refresh lại view và update view mới.
3. Controller: Xử lý các yêu cầu người dùng nhận được thông qua view:
 - Hiển thị, trả về view các dữ liệu đã được định dạng, sắp xếp theo yêu cầu tương tác của người dùng.

- Gọi các hàm từ model hiện thực chỉnh sửa, thao tác trên dữ liệu:
 - Tạo mới ngữ cảnh (context) và chỉnh sửa ngữ cảnh
 - Bật tắt các thiết bị, cảm biến
- Xử lý xác thực dữ liệu: kiểm tra tính hợp lệ của dữ liệu nhập vào từ người dùng: email, tài khoản, mật khẩu.



Hình 57: Cấu trúc thư mục của dự án với 2 phần backend (models, controller) và frontend tuân theo MVC pattern

Lý do lựa chọn MVC

1. MVC tách biệt các phần Model, Controller và View

- Dễ dàng cho việc kiểm thử, chỉnh sửa và mở rộng hệ thống
- Nhiều lập trình viên làm việc đồng thời trên từng bộ phận khác nhau giúp quá trình phát triển hệ thống được nhanh chóng

2. Đồng thời cung cấp nhiều khung View cho Model
3. Sử dụng mô hình MVC chức năng Controller có vai trò quan trọng và tối ưu trên các nền tảng ngôn ngữ khác nhau.

Nhược điểm của MVC

1. Độ lỗi độ chính xác và phức tạp trong phần code hiện thực để 3 thành phần model - view - controller hoạt động, tương tác hiệu quả.

10.2 Observer Pattern

Định nghĩa mỗi phụ thuộc một - nhiều giữa các đối tượng để khi mà một đối tượng có sự thay đổi trạng thái, tất cả các thành phần phụ thuộc của nó sẽ được thông báo và cập nhật một cách tự động. Một đối tượng có thể thông báo đến một số lượng không giới hạn các đối tượng khác.

Đoạn code ứng dụng:

```
const checkContext = async (context) => {
  let satisfied = true;
  if (!context.active || !context.auto_active) {
    satisfied = false;
    return satisfied;
  }
  if (context.input.active_temperature.active) {
    const { min, max } = context.input.active_temperature;
    const temps = await Temperature.find({
      status: true,
      value: { $lte: max, $gte: min },
    });
    if (temps.length === 0) {
      satisfied = false;
      return satisfied;
    }
  }
  if (context.input.active_light.active) {
    const { min, max, active } = context.input.active_light;
    const lights = await Light.find({
      status: active,
      value: { $lte: max, $gte: min },
    });
    if (lights.length === 0) {
```

```
    satisfied = false;
    return satisfied;
}
}

if (context.input.active_humidity.active) {
    const { min, max } = context.input.active_humidity;
    const humis = await Humidity.find({
        status: true,
        value: { $lte: max, $gte: min },
    });
    if (humis.length === 0) {
        satisfied = false;
        return satisfied;
    }
}
if (context.input.human_detection.active) {
    // Add human detection logic here
    const humans = await HumanDetection.find({
        status: true,
        value: context.input.human_detection.value,
    });
    if (humis.length === 0) {
        satisfied = false;
        return satisfied;
    }
}

return satisfied;
};

const handleContext = async (context) => {
    console.log(`Evaluate context ${context.name}`);
    let satisfied = await checkContext(context);
    if (satisfied) {
        console.log(`Context ${context.name} is satisfied!`);
        const fans = context.output.control_fan;
        if (fans.length > 0) {
            fans.forEach(async (fan) => {
                controlDevice("w-fan", fan.name, fan.value);
            });
        }
    }
}
```

```
});  
}  
const leds = context.output.control_led;  
if (leds.length > 0) {  
    leds.forEach(async (led) => {  
        controlDevice("w-led", led.name, led.value);  
    });  
}  
  
//message  
let message = context.notification.message  
? context.notification.message  
: context.description;  
if (context.notification.included_info.fan_status) {  
    message += "\nFan status:";  
    fans.forEach((fan) => {  
        message += '\n${fan.name} value: ${fan.value}';  
    });  
}  
if (context.notification.included_info.light_status) {  
    message += "\nLED status:";  
    leds.forEach((led) => {  
        message += '\n${led.name} value: ${led.value}';  
    });  
}  
if (context.notification.included_info.date_time) {  
    message += "\nDate time: ";  
    message += `${new Date().getHours()}:${new Date().getMinutes()} ${new Date()  
        .getMonth() + 1  
    }/${new Date().getFullYear()}`;  
}  
if (context.notification.email) {  
    // Code for sending email notifications goes here  
    const transporter = nodemailer.createTransport({  
        host: "smtp.gmail.com",  
        port: 587,  
        secure: false, // true for 465, false for other ports  
        auth: {  
            user: "your-email@gmail.com",  
            pass: "your-password"  
        }  
    });  
    transporter.sendMail({  
        from: "your-email@gmail.com",  
        to: "recipient-email@example.com",  
        subject: "Smart Home Status Update",  
        text: message  
    }, (error, info) => {  
        if (error) {  
            console.error(error);  
        } else {  
            console.log(`Email sent: ${info.messageId}`);  
        }  
    });  
}  
});
```

```
        user: process.env.EMAIL_USER, // your email address
        pass: process.env.EMAIL_PASSWORD, // your email password
    },
});

// send mail with defined transport object
const emailSetup = {
    from: process.env.EMAIL_USER, // sender address
    to: context.notification.email, // list of receivers
    subject: '[Luminetix] Notification for Context: ${context.name}', // Sub
    text: message, // plain text body
};

transporter.sendMail(emailSetup, (err, info) => {
    if (err) {
        console.log(err);
    } else {
        console.log('Email sent: ${info.response}');
    }
});

// Send context notification to user
const users = await User.find({});

for (const user of users) {
    await addNoti({
        body: {
            name: 'Context: ${context.name}',
            type: "context",
            message: message,
        },
        user: user,
    });
}

if (
    context.output.frequency.no_repeat ||
    !context.output.active_time.endTime
)
    context.active = false;
```

```
    await context.save();
} else {
    console.log(`Context ${context.name} is not satisfied!`);
}
};

const trackingContext = async (deviceType, message) => {
try {
    let contexts = {};
    if (deviceType === "w-temp") {
        contexts = await Context.find({
            active: true,
            auto_active: true,
            "input.active_temperature.active": true,
        });
    }
    if (deviceType === "w-light") {
        contexts = await Context.find({
            active: true,
            auto_active: true,
            "input.active_light.active": true,
        });
    }
    if (deviceType === "w-humi") {
        contexts = await Context.find({
            active: true,
            auto_active: true,
            "input.active_humidity.active": true,
        });
    }
    if (deviceType === "w-human" && message === "1") {
        contexts = await Context.find({
            active: true,
            auto_active: true,
            "input.active_human.active": true,
        });
    }
    if (deviceType === "w-s-temp" && message === "T_OFF") {
        contexts = await Context.find({
```

```
    active: true ,
    auto_active: true ,
    "input.active_temperature.active": false ,
  });
}

if (deviceType == "w-s-light" && message == "L_OFF") {
  contexts = await Context.find({
    active: true ,
    auto_active: true ,
    "input.active_light.active": false ,
  });
}

if (deviceType == "w-s-humi" && message == "H_OFF") {
  contexts = await Context.find({
    active: true ,
    auto_active: true ,
    "input.active_humidity.active": false ,
  });
}

if (contexts.length > 0) {
  contexts.forEach(async (context) => {
    handleContext(context);
  });
}

} catch (err) {
  console.log("Context tracking error: ", err);
}

};

const controlDevice = async (deviceType, deviceName, message) => {
  try {
    let deviceModel;
    if (deviceType == "w-led") deviceModel = LED;
    else if (deviceType == "w-fan") deviceModel = Fan;
    else if (deviceType == "w-s-temp") deviceModel = Temperature;
    else if (deviceType == "w-s-light") deviceModel = Light;
    else if (deviceType == "w-s-humi") deviceModel = Humidity;
    else console.log(`Error: Invalid device type ${deviceType}`);
    const actualDevice = await deviceModel.findOne({ name: deviceName });
  }
}
```

```
if (!actualDevice) throw new Error("Device not found!");
if (
    actualDevice.value != message
) {
    client.publish(
        `${process.env.ADAFRUIT_USERNAME}/feeds/w-${deviceName}`,
        JSON.stringify({
            value: message,
        }),
        (err) => {
            if (err) {
                throw new Error(err);
            }
        }
    );
    console.log(`Publish message to w-${deviceName} : ${message}`);
    if (
        (deviceType === "w-led" && message === "#00000") ||
        (deviceType === "w-fan" && message === "0")
    )
        actualDevice.status = false;
    else actualDevice.status = true;
    await actualDevice.save();
}
} catch (err) {
    console.log(err);
}
};
```

Đoạn code trên sử dụng design pattern "Observer", trong đó hàm "checkContext" là subject và các hàm "handleContext" và "trackingContext" là observer.

Hàm "checkContext" kiểm tra xem một ngữ cảnh cụ thể có thỏa mãn hay không dựa trên các điều kiện nhất định. Nếu ngữ cảnh được thỏa mãn, thì hàm "handleContext" được gọi, hàm này sẽ thực hiện một số hành động nhất định dựa trên đầu ra của ngữ cảnh.

Hàm "trackingContext" cũng là một observer được gọi khi một số sự kiện nhất định xảy ra (ví dụ: khi nhiệt độ hoặc chỉ số ánh sáng thay đổi). Nó truy xuất danh sách các ngữ cảnh đang hoạt động được liên kết với sự kiện và xác định xem bất kỳ ngữ cảnh nào trong số chúng sẽ được kích hoạt hoặc hủy kích hoạt dựa trên dữ liệu của sự kiện.

10.3 React Hooks - State Pattern

useState Hàm useState làm gì khi được gọi? Nó khai báo một “state variable” (biến state). Đây là cách để “lưu giữ” các giá trị giữa các lần gọi hàm. Thông thường, các biến này “biến mất” khi hàm kết thúc nhưng các biến state này được React giữ lại. Hàm useState nhận tham số gì? Tham số duy nhất được truyền vào hook useState() là state ban đầu. Không giống như khai báo với Class, state không cần thiết phải là object mà có thể là số hoặc chuỗi. useState trả về gì? Nó trả về một cặp giá trị dưới dạng mảng: state hiện tại và một hàm để update nó. Ví dụ ta viết const [count, setCount] = useState(). Thì ở ví dụ này, count là state ban đầu truyền vào, và setCount là hàm thay đổi giá trị của count. Khi setCount được gọi, thì sẽ buộc React rerender lại component.

useEffect useEffect là một hook cho phép chúng ta làm việc với các life cycle ở functional component. Có thể hiểu đơn giản rằng useEffect Hook là của 3 phương thức componentDidMount, componentDidUpdate, và componentWillUnmount kết hợp lại với nhau. Lifecycle là một phần rất quan trọng trong một component. Trong một vài trường hợp chúng ta cần phải fetch data từ API khi component đã được render, hay thực hiện hành động nào đó khi một component được update. Nhưng trong một functional component không thể làm việc với các life cycle này bằng cách thông thường, bởi vậy useEffect Hooks sinh ra để làm điều này. useEffect nhận vào 2 tham số, tham số thứ nhất (1) là 1 arrow function chứa các hàm cần thực thi, tham số thứ 2 là 1 mảng các dependencies. Nếu mảng này trống, thì các hàm trong (1) chỉ được thực thi 1 lần và duy nhất sau khi component được render lần đầu tiên. Nếu mảng này có các phần tử, thì các hàm trong (1) sẽ được thực thi mỗi khi các phần tử này thay đổi, và buộc React phải rerender lại component đó ứng với mỗi thay đổi. Nếu không truyền tham số thứ 2, thì các hàm trong (1) sẽ được thực thi ứng với mọi thay đổi state bất kỳ trong component, khi đó component sẽ được rerender lại. Và ứng với 3 trường hợp trên, thì useEffect luôn được chạy ngay sau khi component render lần đầu tiên.

useNavigation useNavigation là 1 hook cho phép truy cập tới đối tượng navigation. Việc sử dụng useNavigation trở nên hữu ích trong trường hợp không thể truyền trực tiếp navigation prop vào trong component, hoặc chỉ đơn giản là không muốn truyền nó trong trường hợp của 1 component con ở quá sâu bên trong component cha.

useDispatch và useSelector useDispatch return về một tham chiếu đến dispatch function từ Redux store và được sử dụng để dispatch các action.

useSelector cho phép chúng ta lấy state từ Redux store bằng cách sử dụng một selector function làm tham số đầu vào. Khi dispatch một action, useSelector sẽ thực hiện so sánh tham chiếu với giá trị được return trước đó và giá trị hiện tại. Nếu chúng khác nhau, component sẽ bị re-render. Nếu chúng giống nhau, component sẽ không re-render.

11 Hoàn thành sản phẩm

Các chương trình của cả hệ thống IOT giám sát và điều khiển đèn chiếu sáng được tổng hợp bao gồm:

- Chương trình chính nạp vào Yolo:Bit
- Chương trình AI kết nối vào Yolo:Bit bằng Bluetooth
- Chương trình hiện thực Gateway bằng Python
- Chương trình hiện thực Web App
- Chương trình hiện thực Mobile App

Link github các chương trình + thông tin, hướng dẫn sử dụng:

[Link github Luminetix Light System](#)

Nhóm cũng đã hiện thực video demo toàn hệ thống có link dưới đây:

[Link video demo Luminetix Light System](#)

12 Đánh giá hệ thống

12.1 Ưu điểm

Nhóm đã xây dựng được hệ thống có các ưu điểm sau:

- Hiện thực đầy đủ các tính năng cơ bản xử lý lệnh điều khiển các thiết bị.
 - Hiện thực hoàn chỉnh thu nhận và hiển thị dữ liệu từ các thiết bị lên app.
 - Hệ thống ghi nhận, tính toán và đưa ra các dữ liệu thống kê chi tiết giúp người dùng quản lý hệ thống tốt hơn.
 - Hệ thống khi xảy ra sự cố mất mạng Internet vẫn có thể hoạt động ở một vài tính năng cơ bản như bật tắt đèn hay bật tắt quạt bằng remote - điều mà đèn hay quạt bình thường cần phải có.
 - Hệ thống có khả năng đồng bộ dữ liệu tốt giữa các thành phần kiến trúc hệ thống với nhau mà đặc biệt ở phần Server với hai thành phần nhỏ là Adafruit IO Server và MongoDB Server.
 - Mặc dù hệ thống chỉ demo được có 1 node, nhưng hệ thống có khả năng mở rộng thành nhiều nodes và có tính toàn vẹn dữ liệu cao nhờ vào khả năng nhận và xử lý dữ liệu riêng biệt cụ thể, không xảy ra tình trạng nhiễu loạn dữ liệu với nhau.
 - Ứng dụng đa dạng với việc hiện thực cả Web App và Mobile App cùng với giao diện được thiết kế đơn giản, dễ sử dụng, có hướng dẫn sử dụng đối với các tính năng khó chăng hạn như Context Setup.
 - Phần cứng có khả năng tự cảnh báo cho người dùng nếu nhiệt độ vượt ngưỡng khả năng làm việc của hệ thống hay vượt quá khả năng chịu đựng của con người (mặc định từ 5 - 45 độ C)
 - Nhờ vào tính năng Context Setup, người dùng có thể cài đặt các ngưỡng nguy hiểm về giá trị nhiệt độ, độ ẩm không khí, cường độ ánh sáng hay mở chế độ cảnh báo hiện người trong thời gian giới nghiêm,... Hệ thống sẽ cảnh báo kịp thời cho người dùng về các trường hợp xấu có thể xảy ra.
- ...

12.2 Nhược điểm

Tuy nhiên hệ thống nhóm hiện thực còn mắc phải một số nhược điểm như sau:

- Quá trình truyền dữ liệu vẫn còn xảy ra tình trạng delay. Các lý do có thể kể đến là vẫn đề kết nối mạng hay luồng truyền vẫn chưa thực sự tối ưu khi phải qua nhiều thành phần như Gateway, Adafruit IO, server MongoDB,... hay các thiết bị được sử dụng chỉ là các thiết bị mô phỏng tạm thời,...

- Vì một số giới hạn về cấu tạo ở Yolo:Bit, giới hạn truyền nhận dữ liệu ở Adafruit IO Server nên hệ thống chưa có khả năng xử lý dữ liệu ở tốc độ cao và gần như cùng lúc xử lý nhiều lệnh được. Vậy nên ở Context Setup sẽ có trường hợp mặc dù ở Gateway đã nhận được nhiều lệnh, tuy nhiên Yolo:Bit lại không xử lý cùng lúc nhiều lệnh được dẫn đến có thể đèn hoặc quạt sẽ không bật được.
- Server của hệ thống vẫn còn yếu và có tỉ lệ nhỏ bị sập một cách đột ngột khiến hệ thống phải khởi động lại từ đầu.
- App của nhóm hiện thực vẫn còn một vài lỗi nhỏ về hiển thị và giao diện. ...

12.3 Hướng phát triển

Hệ thống của nhóm sẽ có một số hướng phát triển sau đây:

- Nâng cấp phần cứng của hệ thống cùng với việc kết nối trở nên cứng cáp thông qua các kỹ thuật hàn mạch,... giúp giảm tình trạng nhiễu tín hiệu.
- Nâng cấp hệ thống Server để giúp cải thiện bảo mật, mở rộng bộ nhớ lưu trữ, giảm thời gian delay trong quá trình truyền, giảm thiểu tối đa tỉ lệ bị sập đột ngột, cải thiện khả năng xử lý dữ liệu ở tốc độ cực cao,...
- Nâng cấp ứng dụng trở nên hoàn thiện hơn,...

13 Tổng kết đề tài

Xây dựng hệ thống giám sát và điều khiển đèn chiếu sáng đã giúp cho người sử dụng tiết kiệm được chi phí điện năng và giảm thiểu tác động xấu của việc sử dụng năng lượng đến môi trường. Đồng thời, hệ thống này còn giúp tăng tính an toàn và tiện nghi cho khu vực sử dụng, giảm thiểu rủi ro tai nạn và giúp tạo ra một không gian sống và làm việc thông minh, tiện nghi hơn. Việc ứng dụng công nghệ IoT sẽ giúp cho hệ thống này hoạt động hiệu quả và ổn định hơn, đồng thời đảm bảo tính bền vững và phát triển trong tương lai.

Qua bài tập lớn này, nhóm chúng em đã tìm hiểu thêm về khái niệm, ứng dụng kiến thức IoT vào thực tiễn cuộc sống và cách để triển khai hiện thực một hệ thống IoT cơ bản đảm bảo những yêu cầu đặt ra. Ngoài ra nhóm cũng đã cung cấp những kiến thức về quy trình làm một sản phẩm phần mềm, những kiến thức về lập trình các ngôn ngữ,... Đề tài trên sẽ là cơ sở tham khảo và là động lực để các thành viên trong nhóm tiếp tục phát triển những dự án trong tương lai.

Sản phẩm đề tài của nhóm đã hoàn thành và cơ bản đã có kết quả tốt như nhóm mong đợi. Nhóm đã lên kế hoạch chi tiết, cụ thể nội dung công việc hàng tuần và trong suốt quá trình làm việc các thành viên trong nhóm phối hợp làm việc rất tốt, cùng nhau thảo luận, trao đổi để cùng học hỏi, đưa ra những ý tưởng và giải quyết những khúc mắc, vấn đề tồn tại. Tuy nhiên vì mỗi thành viên trong nhóm đều vẫn còn có những mặt hạn chế về kiến thức cũng như xung đột thời gian làm việc giữa các thành viên và các lí do khách quan như quá trình mượn thiết bị còn gặp nhiều sự cố, các thiết bị hư hay không kết nối được,... nên quá trình thực hiện đề tài cũng đã gặp phải những khó khăn.

14 Tài liệu tham khảo

- [1] Giáo trình Ohstem Yolo:Home - AI và IOT cho nhà thông minh

Truy cập tại:

https://drive.google.com/file/d/1VGmRnSNY3Jgfd4YHsT8Uxccm0_n6eYP_/view?usp=sharing

- [2] Giáo trình Ohstem YoloFarm Nông nghiệp công nghệ - Nông nghiệp dựa trên AI và IOT

Truy cập tại:

<https://drive.google.com/file/d/170soJDfErBaltggUBCd67zDM82w-YFcT/view?usp=sharing>

- [3] The Dariu Foundation - Phát triển Gateway IoT bằng Python

- [4] Kênh giáo dục Ohstem - Series Video Khám phá công nghệ mới AI IoT

Truy cập tại:

<https://www.youtube.com/watch?v=2ctToU455pM&list=PLtkN2G0bngmsQ8XMtAIZSKSr3pn75Glpind>

- [5] Trang chủ App Ohstem

Truy cập tại: <https://app.ohstem.vn/>

- [6] Adafruit IO Server

Truy cập tại: <https://www.adafruit.com/>