BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC PHENIKAA

BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN HỆ NHÚNG

THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN ĐIỀU HÒA THÔNG MINH

Nhóm: 01

Danh sách thành viên:

Mã sinh viên	Họ và tên	Lớp
22010291	Nguyễn Việt Anh	CNTT5-K16
21010666	Đỗ Đăng Hoàn	CNTT2-K15
20010889	Nguyên Xuân Dinh	CNTT4-K14

Hà Nội, Năm 2024

MỤC LỤC

PHÂN CÔNG CÔNG VIỆC

Mã SV	Họ và tên	Nhiệm vụ
22010291	Nguyễn Việt Anh	Slide, Word, thuyết trình: - Giới thiệu đề tài - Thiết kế và triển khai phần cứng Mô phỏng: - Thao tác với mô phỏng LCD - Thao tác với LED Ứng dụng di động: - Thiết kế giao diện - Code giao diện nút bấm
21010666	Đỗ Đăng Hoàn	Slide, Word, thuyết trình: Thiết kế và triển khai phần mềm Mô phỏng: - Thao tác với mô phỏng GPIO - Điều khiển bằng âm thanh - Kết nối WebSocket Ứng dụng di động: - Kết nối WebSocket - Thao tác với các nút bấm
20010889	Nguyễn Xuân Dinh	Slide, Word, thuyết trình: - Kiến trúc hệ thống - Kết luận Mô phỏng: - Tìm hiểu và nghiền cứu về mô phỏng DHT22 - Đọc dữ liệu mô phỏng DHT22 - Thao tác với biến Ứng dụng di động: - Code giao diện hiển thị

DANH SÁCH HÌNH ẢNH

1. GIỚI THIỆU ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong thời đại công nghệ phát triển mạnh mẽ như hiện nay, các hệ thống nhúng đóng vai trò quan trọng trong nhiều lĩnh vực của cuộc sống, từ công nghiệp, y tế cho đến gia đình và đời sống hàng ngày. Hệ nhúng không chỉ giúp tăng cường tính tự động hóa, mà còn góp phần tối ưu hóa hiệu suất, tiết kiệm năng lượng và mang lại sự tiện lợi vượt trội.

Báo cáo học phần "Hệ Nhúng" mà chúng em thực hiện tập trung vào việc thiết kế hệ thống điều hòa điều khiển thông minh. Đây là một giải pháp ứng dụng công nghệ nhằm cải tiến khả năng kiểm soát và vận hành của hệ thống điều hòa không khí, giúp hệ thống có thể tự động điều chỉnh nhiệt độ, độ ẩm cũng như tiêu thụ năng lượng một cách hiệu quả. Bên cạnh đó, hệ thống còn có khả năng tương tác thông minh với người dùng thông qua các cảm biến và giao diện điều khiển linh hoạt.

Trong quá trình thực hiện đề tài này, chúng em đã áp dụng những kiến thức về hệ nhúng được học, kết hợp với các công nghệ như cảm biến nhiệt độ, độ ẩm, hệ thống vi điều khiển và các phương thức kết nối, điều khiển thông minh khác. Chúng em hy vọng rằng, với những kiến thức và kỹ năng tích lũy được, hệ thống này sẽ mang lại hiệu quả cao và đáp ứng được yêu cầu thực tiễn trong tương lai.

1.2. Mục tiêu nghiên cứu

Với sự phát triển nhanh chóng của công nghệ và nhu cầu nâng cao chất lượng cuộc sống, các thiết bị gia dụng thông minh đang dần trở thành xu hướng tất yếu. Hệ thống điều hòa không khí, một trong những thiết bị quan trọng giúp duy trì môi trường sống thoải mái, cũng đang chứng kiến sự chuyển đổi mạnh mẽ từ các hệ thống điều khiển thủ công truyền thống sang các hệ thống điều khiển thông minh và tự động hóa.

Mặc dù các hệ thống điều hòa thông minh đã xuất hiện trên thị trường, nhưng phần lớn vẫn gặp những hạn chế về tính năng tự động hóa, khả năng tối ưu năng lượng, và sự linh hoạt trong việc đáp ứng các điều kiện môi trường khác nhau. Do đó, việc phát triển một hệ thống điều hòa điều khiển thông minh với khả năng tự động điều chỉnh nhiệt độ và độ ẩm, tối ưu hóa việc tiêu thụ năng lượng, đồng thời tích hợp các công nghệ như IoT (Internet of Things) và học máy (AI) trở thành một nhu cầu cấp thiết.

1.3. Phạm vi nghiên cứu

1.3.1. Nội dung nghiên cứu

Dự án của chúng em tập trung vào việc phát triển một hệ thống điều khiển điều hòa thông minh, với Raspberry Pi 4 đóng vai trò là bộ điều khiển trung tâm. Hệ thống này được thiết kế nhằm mang lại trải nghiệm sử dụng tối ưu, cho phép người dùng không chỉ điều chỉnh nhiệt độ của điều hòa mà còn có thể dễ dàng chuyển đổi giữa các chế độ làm mát được cài đặt trước. Các chế độ này có thể bao gồm làm mát, sưởi ấm, thông gió, và chế độ tiết kiệm năng lượng, phù hợp với từng điều kiện môi trường và nhu cầu của người dùng.

Hơn nữa, hệ thống còn được tích hợp các cảm biến đo nhiệt độ và độ ẩm, cho phép theo dõi liên tục các thông số môi trường trong phòng. Dữ liệu từ cảm biến sẽ được cập nhật thường xuyên và hiển thị cho người dùng trên giao diện ứng dụng. Nhờ đó, người dùng có thể theo dõi và điều chỉnh hệ thống điều hòa một cách thông minh để duy trì mức độ thoải mái và tiết kiệm năng lượng hiệu quả hơn.

Một điểm nổi bật của dự án là khả năng điều khiển từ xa thông qua ứng dụng di động. Ứng dụng này sẽ được kết nối với Raspberry Pi thông qua mạng Wi-Fi hoặc internet, cho phép người dùng điều chỉnh hệ thống điều hòa ở bất cứ đâu và vào bất kỳ lúc nào. Tính năng điều khiển thời gian thực giúp người dùng không cần phải ở nhà mà vẫn có thể theo dõi và điều chỉnh nhiệt độ phòng theo mong muốn. Ví dụ, trước khi về nhà, người dùng có thể bật chế độ làm mát để khi về đến nhà, căn phòng đã sẵn sàng với nhiệt độ lý tưởng.

Ngoài ra, hệ thống còn hỗ trợ tự động hóa một số tác vụ dựa trên cảm biến, chẳng hạn khi nhiệt độ hoặc độ ẩm vượt ngưỡng cài đặt, hệ thống sẽ tự động điều chỉnh để giữ cho môi trường luôn thoải mái mà không cần can thiệp thủ công. Điều này giúp tối ưu hóa việc sử dụng điều hòa, giảm tiêu thụ điện năng, và góp phần bảo vệ môi trường.

Tổng thể, hệ thống điều khiển điều hòa thông minh này không chỉ mang lại sự tiện ích cho người sử dụng mà còn giúp quản lý năng lượng một cách hiệu quả, đồng thời đảm bảo duy trì môi trường sống thoải mái và hiện đại.

1.3.2. Giới hạn về tài nghiên và công cụ

Trong báo cáo này, chúng em tập trung vào việc sử dụng các thiết bị phần cứng như Raspberry Pi 4, cảm biến nhiệt độ và độ ẩm DHT22, màn hình LCD1602, nút bấm và đèn LED để phát triển hệ thống điều khiển thông minh. Tuy nhiên, để thuận tiện cho việc thử nghiệm và phát triển, chúng em đã lựa chọn phương pháp mô phỏng thay thế cho các thành phần phần cứng thực tế.

Cụ thể, chúng em sử dụng mô phỏng GPIO để thay thế cho các chân điều khiển vào/ra trên Raspberry Pi, giúp dễ dàng kiểm tra tín hiệu của các nút bấm và đèn LED. Mô phỏng LCD1602 được áp dụng để hiển thị các thông tin như nhiệt độ và độ ẩm mà không cần sử dụng màn hình thực tế. Đối với cảm biến DHT22, chúng em cũng sử dụng mô phỏng để lấy dữ liệu nhiệt độ và độ ẩm từ môi trường, thay thế cho các phép đo thực tế. Cuối cùng, mô phỏng môi trường Raspberry Pi 4 qua phần mềm QEMU cho phép chúng em kiểm tra toàn bộ hệ thống trên máy tính mà không cần đến Raspberry Pi thật.

Mặc dù việc sử dụng mô phỏng mang lại nhiều lợi ích như tiết kiệm chi phí, giảm thiểu lỗi phần cứng trong quá trình phát triển và giúp nhanh chóng kiểm tra chức năng của hệ thống, nhưng nó cũng đi kèm với nhiều hạn chế. Mô phỏng chỉ có thể phản ánh một phần các chức năng của phần cứng thực, dẫn đến việc không thể đạt được kết quả chính xác hoặc hiệu suất cao như khi sử dụng thiết bị thực tế. Ví dụ, độ chính xác của dữ liệu từ cảm biến nhiệt độ và độ ẩm DHT22 sẽ không hoàn toàn giống với dữ liệu thật, hoặc tốc độ phản hồi của mô phỏng GPIO có thể khác so với khi sử dụng Raspberry Pi thật.

Do đó, trong báo cáo này, chúng em không chỉ mô phỏng hệ thống để minh họa các chức năng cốt lõi mà còn nêu rõ những giới hạn mà việc sử dụng mô phỏng mang lại. Điều này nhằm làm rõ rằng, mặc dù các chức năng chính của hệ thống đã được mô phỏng thành công, nhưng để đạt được hiệu quả cao nhất, việc triển khai trên phần cứng thực tế vẫn là lựa chọn tối ưu. Những hạn chế này ảnh hưởng đến độ chính xác và độ tin cậy của hệ thống, nhưng vẫn cung cấp một nền tảng hữu ích để phát triển và thử nghiệm trước khi chuyển sang giai đoạn sử dụng phần cứng thực tế.

2. KIẾN TRÚC HỆ THỐNG

2.1. Tổng quan kiến trúc hệ thống

Hệ thống điều khiển thông minh sẽ được thiết kế với Raspberry Pi 4 làm bộ điều khiển trung tâm. Raspberry Pi 4 có nhiệm vụ nhận dữ liệu từ cảm biến DHT22, xử lý các logic điều khiển và hiển thị thông tin lên màn hình LCD1602. Cảm biến DHT22 đảm nhận vai trò đo nhiệt độ và độ ẩm trong phòng và gửi kết quả về cho Raspberry Pi 4. Các thông tin này sẽ được hiển thị trên màn hình LCD1602, bao gồm: nhiệt độ phòng hiện tại, độ ẩm, thời gian hẹn giờ và chế độ hoạt động của điều hòa.

Hệ thống sẽ được thiết kế như một bảng điều khiển gắn tường kết nối trực tiếp với điều hòa, cho phép người dùng điều chỉnh các thông số một cách dễ dàng. Trong quá trình nghiên cứu và phát triển, chúng em sẽ sử dụng mô phỏng phần cứng để thay thế cho các thiết bị vật lý thực tế, bao gồm mô phỏng GPIO, màn hình LCD và cảm biến DHT22, nhằm kiểm tra và đánh giá các chức năng của hệ thống trước khi triển khai trên phần cứng thực nếu có.

2.2. Các thành phần chính của hệ thống

Bộ điều khiển trung tâm – Raspberry Pi 4 đóng vai trò là bộ điều khiển chính trong hệ thống. Raspberry Pi 4 chịu trách nhiệm tiếp nhận và xử lý các lệnh điều khiển từ người dùng, đồng thời nhận dữ liệu từ các thiết bị ngoại vi như cảm biến và hiển thị thông tin lên màn hình. Dựa trên dữ liệu đầu vào và các logic đã lập trình, Raspberry Pi 4 sẽ điều chỉnh điều hòa, đảm bảo nhiệt độ và chế độ hoạt động phù hợp với môi trường và yêu cầu của người dùng.

Cảm biến DHT22 có nhiệm vụ đo lường các thông số môi trường, cụ thể là nhiệt độ và độ ẩm trong phòng. Sau khi thu thập dữ liệu, cảm biến sẽ gửi kết quả về cho Raspberry Pi 4 để xử lý. Dữ liệu này không chỉ được sử dụng để điều chỉnh hoạt động của điều hòa mà còn được hiển thị lên màn hình để người dùng có thể theo dõi tình trạng môi trường trong thời gian thực.

Màn hình LCD1602 đảm nhận việc hiển thị các thông tin quan trọng như nhiệt độ phòng hiện tại, độ ẩm, thời gian hẹn giờ và chế độ hoạt động của điều hòa. Màn hình này cung cấp giao diện trực quan giúp người dùng dễ dàng theo dõi và kiểm soát các chức năng của hệ thống điều hòa thông qua việc hiển thị thông tin rõ ràng và đầy đủ.

Nút bấm và đèn LED giúp người dùng tương tác với hệ thống một cách trực tiếp. Nút bấm cho phép người dùng thay đổi các thông số cài đặt như nhiệt độ, chế độ hoạt động, hoặc cài đặt thời gian hẹn giờ cho điều hòa. Trong khi đó, đèn LED sẽ thể hiện trạng thái hoạt động của hệ thống, báo hiệu các trạng thái như hệ thống đang bật, hẹn giờ hoặc các cảnh báo nếu cần thiết.

2.3. Luồng hoạt động của hệ thống

Đo lường thông số môi trường: Cảm biến DHT22 đóng vai trò liên tục giám sát điều kiện môi trường trong phòng bằng cách đo lường nhiệt độ và độ ẩm. Các giá trị này được cảm biến thu thập thường xuyên và gửi về Raspberry Pi 4 để xử lý, đảm bảo rằng hệ thống luôn nhận được thông tin cập nhật về môi trường để có thể điều chỉnh điều hòa một cách tối ưu.

Xử lý thông tin: Raspberry Pi 4 sẽ nhận dữ liệu từ cảm biến DHT22 và so sánh các giá trị nhiệt độ, độ ẩm này với các ngưỡng cài đặt trước đó. Dựa trên sự so sánh này, hệ thống sẽ quyết định hành động phù hợp, chẳng hạn như bật hoặc tắt điều hòa, chuyển đổi giữa các chế độ hoạt động (làm mát, sưởi ấm, hoặc tiết kiệm năng lượng) hoặc duy trì chế độ hiện tại để đảm bảo sự thoải mái và hiệu quả năng lượng.

Hiển thị thông tin: Màn hình LCD1602 đóng vai trò cung cấp giao diện trực quan cho người dùng, hiển thị các thông số quan trọng như nhiệt độ phòng hiện tại, độ ẩm, chế độ điều hòa đang hoạt động và thời gian hẹn giờ. Nhờ vậy, người dùng có thể dễ dàng theo dõi các điều kiện môi trường và trạng thái của hệ thống mà không cần phải kiểm tra thủ công.

Tương tác người dùng: Người dùng có thể thao tác trực tiếp với hệ thống thông qua các nút bấm để thay đổi chế độ hoạt động của điều hòa, điều chỉnh nhiệt độ mong muốn, hoặc cài đặt thời gian hẹn giờ. Mỗi khi người dùng thực hiện một hành động, Raspberry Pi 4 sẽ xử lý lệnh này và điều chỉnh điều hòa theo các thông số đã được thiết lập, đảm bảo rằng hệ thống luôn phản hồi nhanh chóng và chính xác theo yêu cầu.

2.4. Công nghệ và công cụ sử dụng

Về phần mềm, chúng em chọn Python làm ngôn ngữ lập trình chính để phát triển các chương trình điều khiển hoạt động của hệ thống. Python được triển khai trong môi trường mô phỏng QEMU nhằm giả lập đầy đủ các chức năng của Raspberry Pi 4, cho phép chúng em thử nghiệm mà không cần phải sử dụng phần cứng thật trong giai đoạn nghiên cứu. Để thay thế cho các phần cứng thực, chúng em sử dụng các mô phỏng tương ứng như mô phỏng GPIO, mô phỏng cảm biến nhiệt độ và độ ẩm DHT22, và màn hình LCD1602. Những mô phỏng này giúp tái hiện lại các chức năng cơ bản của hệ thống, đồng thời tạo điều kiện thuận lợi để chúng em kiểm tra và tối ưu hóa các logic điều khiển.

Bên cạnh đó, hệ thống còn sử dụng WebSocket để thực hiện giao tiếp thời gian thực giữa Python và ứng dụng di động được phát triển trên nền tảng Flutter. WebSocket đóng vai trò quan trọng trong việc duy trì sự kết nối liên tục giữa hệ thống điều khiển và ứng dụng di động, cho phép người dùng điều khiển và giám sát hệ thống một cách trực tiếp từ xa. Ứng dụng di động này được xây dựng bằng Flutter và ngôn ngữ lập trình Dart, cung cấp giao diện thân thiện và trực quan, giúp người dùng dễ dàng thao tác với hệ thống qua điện thoại di động.

Về các công nghệ kết nối, chúng em sử dụng giao thức I2C để kết nối giữa Raspberry Pi 4 và màn hình LCD1602. I2C là giao thức truyền thông hai dây, giúp giảm thiểu số lượng chân kết nối cần thiết trên Raspberry Pi và đồng thời đảm bảo tốc độ truyền tải dữ liệu nhanh, độ tin cậy cao. Nhờ I2C, hệ thống có thể hiển thị dữ liệu như nhiệt độ, độ ẩm, và trạng thái hoạt động của điều hòa một cách ổn định và chính xác.

Ngoài ra, giao tiếp GPIO được chúng em sử dụng để kết nối và điều khiển các nút bấm và đèn LED. Các chân GPIO trên Raspberry Pi giúp hệ thống nhận lệnh từ nút bấm và điều khiển đèn LED để báo hiệu trạng thái hoạt động của hệ thống. Ví dụ, khi người dùng nhấn nút để thay đổi chế độ hoặc điều chỉnh nhiệt độ, các tín hiệu sẽ được truyền về Raspberry Pi để xử lý và đèn LED sẽ phản hồi trạng thái tương ứng, chẳng hạn như bật/tắt điều hòa hoặc chuyển sang chế độ tiết kiệm năng lượng.

Tổng thể, hệ thống của chúng em không chỉ tập trung vào việc mô phỏng các chức năng phần cứng thực, mà còn chú trọng vào sự tương tác liên tục giữa người dùng và hệ

thống thông qua các công nghệ kết nối hiện đại như WebSocket, I2C và GPIO, giúp hệ thống hoạt động một cách hiệu quả và chính xác.

3. THIẾT KẾ VÀ TRIỂN KHAI PHẦN CỨNG

3.1. Kiến trúc phần cứng hệ thống

Hệ thống điều khiển điều hòa thông minh của chúng em được thiết kế với kiến trúc phần cứng tối ưu để mang lại trải nghiệm tương tác dễ dàng và trực quan cho người dùng. Raspberry Pi 4 là bộ điều khiển trung tâm, kết nối với nhiều thành phần phần cứng để thực hiện các chức năng điều khiển và giám sát. Hệ thống sử dụng 8 nút bấm, mỗi nút đại diện cho một chức năng cụ thể của hệ thống điều hòa, cùng với đèn LED hiển thị trạng thái, màn hình LCD để hiển thị thông tin, và cảm biến nhiệt độ để giám sát môi trường trong phòng.

Hệ thống có 8 nút bấm, mỗi nút tương ứng với một chức năng điều khiển riêng, tạo sự linh hoạt và dễ sử dụng cho người dùng. Nút bật/tắt điều hòa cho phép người dùng nhanh chóng điều khiển trạng thái hoạt động của điều hòa. Nút tăng nhiệt độ và giảm nhiệt độ giúp điều chỉnh nhiệt độ mong muốn. Nút chuyển trang trên màn hình LCD cho phép người dùng xem các trang thông tin khác nhau như nhiệt độ, độ ẩm, và chế độ hoạt động. Nút thay đổi chế độ (Mode) giúp chuyển đổi giữa các chế độ như làm mát, hẹn giờ, quạt gió, hoặc chế độ tiết kiệm năng lượng. Nút tốc độ quạt cho phép thay đổi tốc độ gió để phù hợp với yêu cầu của người dùng. Nút hẹn giờ tắt điều hòa cho phép thiết lập thời gian tắt tự động, giúp tiết kiệm năng lượng. Cuối cùng, nút ra lệnh bằng giọng nói kích hoạt chức năng điều khiển bằng giọng nói, giúp điều khiển hệ thống dễ dàng mà không cần thao tác thủ công.

Một đèn LED được tích hợp để báo hiệu trạng thái hiện tại của điều hòa. Khi điều hòa bật, đèn LED sẽ sáng, và khi điều hòa tắt, đèn sẽ tắt, giúp người dùng dễ dàng nhận biết trạng thái hoạt động mà không cần kiểm tra trực tiếp.

Màn hình LCD1602 sẽ hiển thị các thông tin quan trọng như nhiệt độ phòng, độ ẩm, thời gian hẹn giờ, và chế độ hoạt động của điều hòa. Màn hình này giúp cung cấp thông tin rõ ràng và trực quan cho người dùng để dễ dàng theo dõi và điều chỉnh hệ thống.

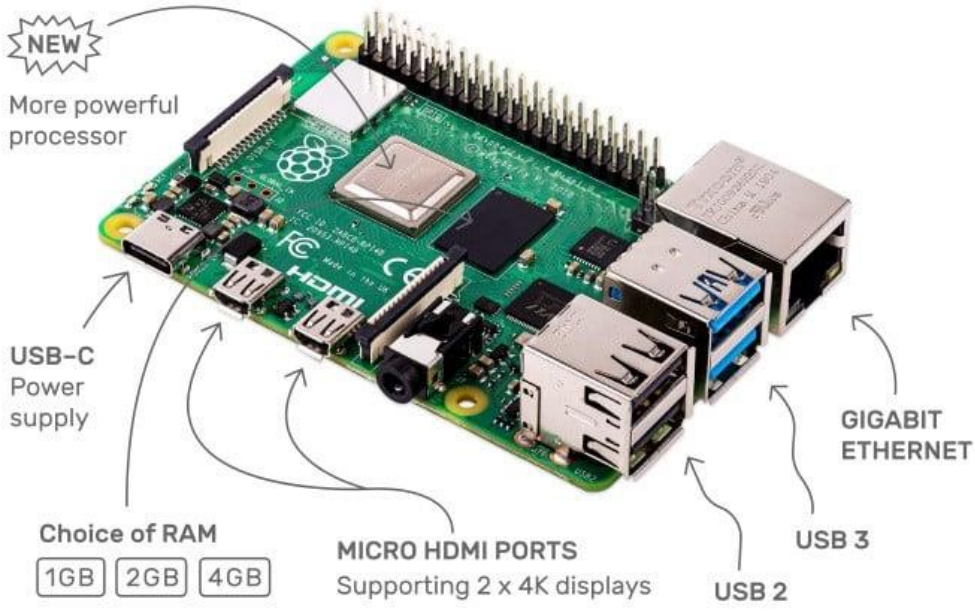
Cảm biến DHT22 đo lường nhiệt độ trong phòng và cung cấp dữ liệu chính xác về nhiệt độ môi trường. Thông tin từ cảm biến sẽ được gửi về Raspberry Pi để hệ thống điều chỉnh hoạt động của điều hòa nhằm đảm bảo điều kiện môi trường tối ưu.

Với kiến trúc phần cứng này, hệ thống đảm bảo khả năng điều khiển linh hoạt và chính xác, cung cấp cho người dùng sự tiện lợi và khả năng kiểm soát hoàn toàn hệ thống điều hòa một cách dễ dàng và thông minh.

3.2. Lựa chọn thiết bị phần cứng

3.2.1. Raspberry Pi 4

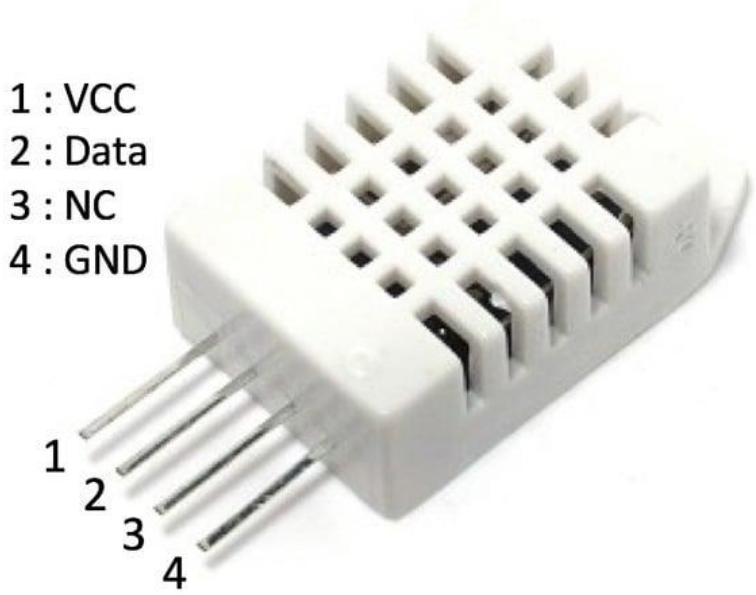
Raspberry Pi 4 là bộ điều khiển trung tâm của hệ thống, chịu trách nhiệm xử lý tất cả các dữ liệu nhận được từ cảm biến và điều khiển các thiết bị ngoại vi. Với CPU Quad-core ARM Cortex-A72 tốc độ 1.5GHz và RAM tùy chọn từ 2GB đến 8GB, Raspberry Pi 4 đủ mạnh để xử lý các tác vụ phức tạp và đảm bảo hiệu suất hệ thống. Nó cung cấp nhiều cổng kết nối như GPIO, USB, HDMI, Ethernet, và hỗ trợ Wi-Fi và Bluetooth để kết nối với các thiết bị ngoại vi và mạng không dây. Với vai trò là trung tâm điều khiển, Raspberry Pi 4 nhận và xử lý dữ liệu từ cảm biến nhiệt độ và độ ẩm và các phương thức khác.



Hình 1: Kit Raspberry Pi 4

3.2.2. Cảm biến nhiệt độ và độ ẩm DHT22

Cảm biến độ ẩm, nhiệt độ DHT22 Temperature Humidity Sensor ra chân là phiên bản ra chuẩn chân cắm thông dụng 2.54mm hàn sẵn trên mặt in với trở kéo dễ dàng sử dụng, ứng ụng đo độ ẩm, nhiệt độ môi trường với độ chính xác cao, cảm biến có chất lượng tốt, độ bền và độ ổn định cao.



Hình 2: Cảm biến DHT22

Thông số kỹ thuật:

- Nguồn sử dụng: 3~5 VDC.
- Dòng sử dụng: 2.5mA max (khi truyền dữ liệu).
- Đo tốt ở độ ẩm 0100%RH với sai số 2-5%.
- Đo tốt ở nhiệt độ -40 to 80°C sai số ±0.5°C.
- Tần số lấy mẫu tối đa 0.5Hz (2 giây 1 lần)
- Kích thước 27mm x 59mm x 13.5mm (1.05" x 2.32" x 0.53")
- Chân tín hiệu: 5VDC(+) | OUT | GND (-) (chân out nối trực tiếp với chân giao tiếp của VĐK không cần trở kéo vì đã có sẵn trên mạch).

3.2.3. Màn hình LCD1602

Mạch hiển thị MKE-M07 LCD1602 I2C module được sử dụng để hiển thị thông tin dưới dạng ký tự với khả năng hiển thị 2 dòng, mỗi dòng 16 ký tự, mạch được tích hợp sẵn bộ chuyển đổi giao tiếp I2C cho LCD nên có thể dễ dàng kết nối và sử dụng với chỉ 2 chân giao tiếp I2C là SDA (data) và SCL (clock).

Hình 3: Mạch hiển thị MKE-M07 LCD1602 I2C module

Thông số kỹ thuật:

- Điện áp hoạt động: 5VDC
- Chuẩn giao tiếp: Digital I2C
- Các chân giao tiếp: SDA (Serial Data) / SCL (Serial Clock)
- Điện áp giao tiếp: TTL 3.3/5VDC
- Loại LCD: LCD1602 (2 dòng, mỗi dòng 16 ký tự), datasheet.

- IC chuyển giao tiếp LCD sang I2C: PCF8574T, datasheet.
- Sử dụng trực tiếp an toàn với các board mạch giao tiếp ở cả hai mức điện áp 3.3VDC và 5VDC như: Arduino, Raspberry Pi, Jetson Nano, …
- Bổ sung thêm các thiết kế ổn định, chống nhiễu.
- Chuẩn kết nối: Conector XH2.54 4Pins

3.2.4. LED, nút bấm, điện trở

LED được sử dụng để báo hiệu trạng thái của hệ thống. Ví dụ, khi điều hòa đang bật hoặc khi hệ thống đang hoạt động bình thường, đèn LED sẽ sáng để thông báo cho người dùng biết hệ thống đang ở trạng thái nào. LED có thể phát sáng với màu sắc khác nhau để biểu thị các trạng thái khác nhau của hệ thống, như chế độ bật/tắt, chế độ làm mát hoặc sưởi ấm.

Các nút bấm cho phép người dùng thực hiện các thao tác điều khiển thủ công, chẳng hạn như bật/tắt điều hòa hoặc thay đổi chế độ hoạt động. Nút bấm sẽ gửi tín hiệu đến Raspberry Pi thông qua chân GPIO khi người dùng thực hiện một thao tác. Từ đó, hệ thống sẽ xử lý và thực hiện lệnh tương ứng, đảm bảo người dùng có thể dễ dàng tương tác với hệ thống mà không cần đến giao diện điều khiển từ xa.

Điện trở làm nhiệm vụ bảo vệ các thành phần điện tử như LED và nút bấm khỏi dòng điện quá mức. Điện trở đảm bảo dòng điện chạy qua LED và nút bấm luôn nằm trong ngưỡng an toàn, tránh hiện tượng cháy hỏng các thiết bị do quá tải điện. Việc sử dụng đúng giá trị điện trở là rất quan trọng để hệ thống hoạt động an toàn và ổn định.

3.2.5. Thẻ nhớ microSD

Thẻ nhớ microSD đóng vai trò lưu trữ hệ điều hành (Raspberry Pi OS) và các chương trình điều khiển của hệ thống. Với dung lượng tối thiểu 16GB và tốc độ Class 10 hoặc UHS-I, thẻ nhớ này đảm bảo hiệu suất tốt trong việc đọc/ghi dữ liệu, giúp hệ thống vận hành mượt mà. Thẻ nhớ microSD không chỉ chứa hệ điều hành mà còn lưu trữ các dữ liệu liên quan đến việc điều khiển hệ thống và giao tiếp với các thiết bị ngoại vi.

3.2.6. Nguồn điện cho Raspberry Pi 4

Nguồn điện là thành phần quan trọng giúp cung cấp năng lượng ổn định cho Raspberry Pi 4 và các thiết bị ngoại vi. Raspberry Pi 4 yêu cầu nguồn điện 5V DC với dòng điện tối thiểu 3A để đảm bảo hoạt động ổn định, đặc biệt khi kết nối với nhiều thiết bị ngoại vi như cảm biến và màn hình LCD. Việc cung cấp nguồn điện ổn định sẽ đảm bảo hệ thống vận hành một cách trơn tru và không bị gián đoạn do thiếu năng lượng. Nguồn điện là thành phần quan trọng giúp cung cấp năng lượng ổn định cho Raspberry Pi 4 và các thiết bị ngoại vi. Raspberry Pi 4 yêu cầu nguồn điện 5V DC với dòng điện tối thiểu 3A để đảm bảo hoạt động ổn định, đặc biệt khi kết nối với nhiều thiết bị ngoại

vi như cảm biến và màn hình LCD. Việc cung cấp nguồn điện ổn định sẽ đảm bảo hệ thống vận hành một cách trơn tru và không bị gián đoạn do thiếu năng lượng.

3.2.7. Cáp và kết nối

Cáp và kết nối đóng vai trò kết nối Raspberry Pi với các thiết bị ngoại vi như cảm biến DHT22, màn hình LCD1602, LED, và nút bấm. Cáp kết nối GPIO giúp Raspberry Pi giao tiếp với các thiết bị này, truyền dữ liệu và lệnh điều khiển một cách chính xác. Cáp I2C được sử dụng để kết nối với màn hình LCD1602, giúp giảm thiểu số lượng chân GPIO cần thiết và đảm bảo kết nối ổn định. Các loại cáp kết nối này là yếu tố cần thiết để duy trì hệ thống điều khiển hoạt động ổn định và hiệu quả.

3.3. Thiết kế mạch điện

Thiết kế mạch của hệ thống bao gồm việc kết nối các nút bấm, LED và các thiết bị ngoại vi khác với các chân GPIO của Raspberry Pi 4, đảm bảo hệ thống có thể thực hiện các chức năng điều khiển và phản hồi theo lệnh của người dùng.

Các nút bấm được kết nối với Raspberry Pi để thực hiện các chức năng điều khiển khác nhau:

- Nút bật/tắt điều hòa kết nối với GPIO 18, cho phép người dùng bật hoặc tắt điều hòa dễ dàng.
- Nút tăng nhiệt độ kết nối với GPIO 19 giúp điều chỉnh nhiệt độ lên cao hơn.
- Nút giảm nhiệt độ kết nối với GPIO 20 để giảm nhiệt độ khi cần thiết.
- Nút chuyển trang kết nối với GPIO 16, giúp người dùng chuyển đổi giữa các trang hiển thị trên màn hình LCD1602.
- Nút Mode (thay đổi chế độ hoạt động) kết nối với GPIO 21, cho phép chuyển đổi giữa các chế độ điều hòa như làm mát, sưởi ấm, hoặc tiết kiệm năng lượng.
- Nút điều chỉnh tốc độ quạt kết nối với GPIO 15, giúp thay đổi tốc độ quạt của điều hòa.
- Nút hẹn giờ kết nối với GPIO 14, cho phép người dùng đặt hẹn giờ tắt điều hòa tự động.
- Nút ra lệnh bằng giọng nói kết nối với GPIO 13, kích hoạt tính năng điều khiển bằng giọng nói để tăng tính tiện lợi khi điều khiển hệ thống.

Các nút bấm này được kết nối với GPIO của Raspberry Pi và một đầu nối với GND để hoàn thành mạch. Để đảm bảo tín hiệu ổn định và tránh tình trạng nhiễu hoặc tín hiệu không xác định, mỗi nút bấm cần được kết nối với một điện trở kéo xuống (pull-down resistor) 10kΩ. Điện trở này giúp giữ cho tín hiệu GPIO ở mức thấp khi nút bấm chưa được nhấn.

Đèn LED báo hiệu trạng thái bật/tắt của điều hòa kết nối với GPIO 22. Để bảo vệ LED khỏi hư hỏng do dòng điện quá cao, cần sử dụng điện trở hạn dòng 220Ω hoặc 330Ω. Điện trở này giúp giới hạn dòng điện qua LED, đảm bảo rằng LED hoạt động ổn định và không bị cháy.

Màn hình LCD1602 sử dụng giao thức I2C, giúp tiết kiệm số lượng chân GPIO sử dụng. LCD kết nối với SDA và SCL của Raspberry Pi để hiển thị thông tin như nhiệt độ, độ ẩm và trạng thái của hệ thống điều hòa.

Cảm biến DHT22 kết nối với GPIO 12 và chịu trách nhiệm đo nhiệt độ và độ ẩm trong phòng. Dữ liệu từ cảm biến được gửi về Raspberry Pi để xử lý và điều chỉnh hoạt động của điều hòa phù hợp với điều kiện môi trường.

Với việc bổ sung các điện trở cho nút bấm và LED, thiết kế mạch đảm bảo sự an toàn và ổn định khi hoạt động. Các điện trở giúp hạn chế dòng điện quá mức và đảm bảo tín hiệu từ các nút bấm luôn chính xác, tránh hiện tượng dao động tín hiệu. Toàn bộ mạch điện được thiết kế để đáp ứng các yêu cầu điều khiển và giám sát hệ thống một cách hiệu quả, đơn giản và an toàn.

4. THIẾT KẾ VÀ TRIỂN KHAI PHẦN MỀM

4.1. Yêu cầu phần mềm

Hệ thống phần mềm được phát triển với mục tiêu mô phỏng chính xác các thành phần vật lý của hệ thống điều khiển điều hòa, cho phép tương tác như khi sử dụng phần cứng thực tế. Python là ngôn ngữ chính để lập trình các mô phỏng này, giúp tái hiện lại hoạt động của các cảm biến, nút bấm, và các thiết bị ngoại vi khác trong môi trường giả lập. Các mô phỏng này cho phép hệ thống phản hồi giống như khi sử dụng phần cứng thực, đảm bảo rằng các hành động từ người dùng như nhấn nút hoặc thay đổi thông số sẽ được xử lý và phản hồi một cách chính xác.

Ngoài việc mô phỏng các thành phần vật lý, hệ thống cũng hỗ trợ người dùng tương tác trực tiếp thông qua các nút bấm trong môi trường mô phỏng. Các nút này được lập trình để hoạt động tương tự như các nút vật lý ngoài đời thật, cho phép người dùng thực hiện các thao tác như bật/tắt điều hòa, tăng giảm nhiệt độ, chuyển đổi chế độ, và nhiều chức năng khác ngay trên giao diện mô phỏng.

Đối với ứng dụng di động, em sử dụng Flutter để phát triển giao diện điều khiển trên thiết bị di động. Ứng dụng Flutter cung cấp một giao diện thân thiện, cho phép người dùng điều khiển hệ thống điều hòa từ xa thông qua các tính năng được tích hợp sẵn như thay đổi nhiệt độ, cài đặt hẹn giờ, hoặc thậm chí sử dụng giọng nói để điều khiển các chức năng của hệ thống.

Phần mềm cần phải đảm bảo tính linh hoạt và tương tác liền mạch giữa các thành phần mô phỏng và hệ thống điều khiển thực tế, đồng thời cung cấp trải nghiệm người dùng tương tự như khi thao tác trực tiếp với phần cứng.

4.2. Kiến trúc phần mềm

Hệ thống phần mềm được thiết kế với một cấu trúc đơn giản nhưng hiệu quả, sử dụng tệp config.py làm trung tâm lưu trữ các biến quan trọng của hệ thống. config.py chứa các biến toàn cục cần thiết để lưu trữ trạng thái hệ thống, chẳng hạn như nhiệt độ, chế độ điều hòa, thời gian hẹn giờ, và các thông số khác. Mỗi khi có sự thay đổi ở một chức năng nào đó trong hệ thống, các giá trị biến trong tệp config.py sẽ được tự động cập nhật và ghi đè. Điều này đảm bảo rằng tất cả các phần của hệ thống đều sử dụng các giá trị mới nhất, giúp duy trì tính đồng nhất và chính xác của dữ liệu.

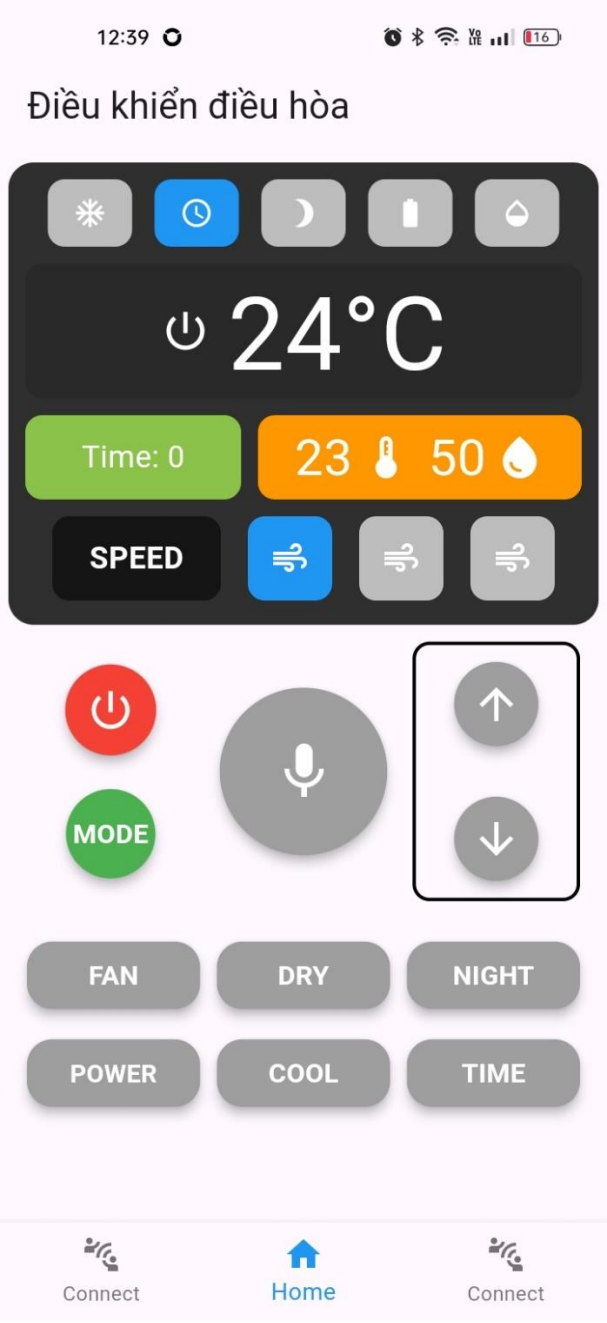
Ngoài việc sử dụng config.py để lưu trữ và cập nhật dữ liệu, hệ thống còn áp dụng giao thức WebSocket để giao tiếp dữ liệu giữa phần mềm Python và ứng dụng di động phát triển bằng Flutter. Dữ liệu được truyền dưới dạng JSON, cho phép gửi và nhận dữ liệu một cách hiệu quả giữa các thiết bị. Khi dữ liệu trên ứng dụng di động Flutter thay đổi (chẳng hạn như người dùng điều chỉnh nhiệt độ, thay đổi chế độ, hoặc cài đặt hẹn giờ), dữ liệu mới sẽ được gửi qua WebSocket về hệ thống Python, sau đó ghi đè lên các biến tương ứng trong tệp config.py. Tương tự, khi dữ liệu trong hệ thống thay đổi, nó sẽ được gửi đến ứng dụng Flutter để cập nhật giao diện người dùng.

Kiến trúc phần mềm này đảm bảo rằng cả hệ thống mô phỏng trên Python và ứng dụng di động đều hoạt động liền mạch, với sự đồng bộ liên tục giữa các thành phần, giúp người dùng có thể điều khiển hệ thống một cách trực quan và hiệu quả từ xa thông qua ứng dụng di động.

4.3. Thiết kế giao diện người dùng

Hình 5: Giao diện mô phỏng

Trong quá trình phát triển hệ thống, do không có các phần cứng vật lý thật để thao tác, chúng em đã sử dụng các thư viện giả lập để mô phỏng các chức năng tương ứng của phần cứng. Các thư viện này giúp hệ thống có thể hoạt động và kiểm tra đầy đủ các tính năng của nó trong môi trường mô phỏng. Khi có phần cứng thật, chúng em chỉ cần thay đổi các thư viện mô phỏng bằng các thư viện tương thích với phần cứng thực, và hệ thống sẽ hoạt động hoàn chỉnh mà không cần phải thay đổi kiến trúc hay logic điều khiển. Điều này giúp chúng em tiết kiệm thời gian và đảm bảo tính linh hoạt trong quá trình phát triển.



Hình 6: Giao diện người dùng trên thiết bị di động

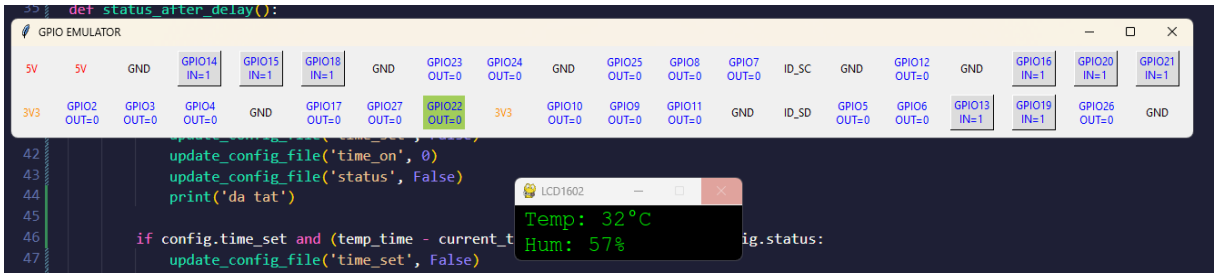
Giao diện trên ứng dụng di động được thiết kế trực quan với các chức năng điều khiển điều hòa dễ sử dụng. Ở giữa, nhiệt độ hiện tại được hiển thị lớn và rõ ràng cùng các biểu tượng điều chỉnh chế độ như làm mát, hẹn giờ, ban đêm, và tiết kiệm năng lượng. Dưới đó, các thông số như thời gian hẹn giờ, nhiệt độ phòng và độ ẩm được hiển thị rõ ràng. Người dùng có thể điều chỉnh tốc độ quạt và nhiệt độ thông qua các nút tăng giảm. Nút bật/tắt điều hòa và nút kích hoạt điều khiển giọng nói được bố trí nổi bật.

Ngoài ra, còn có các nút chức năng khác như Mode, Fan, Dry, Night, Cool, và Time để chuyển đổi các chế độ.

4.4. Triển khai phần mềm

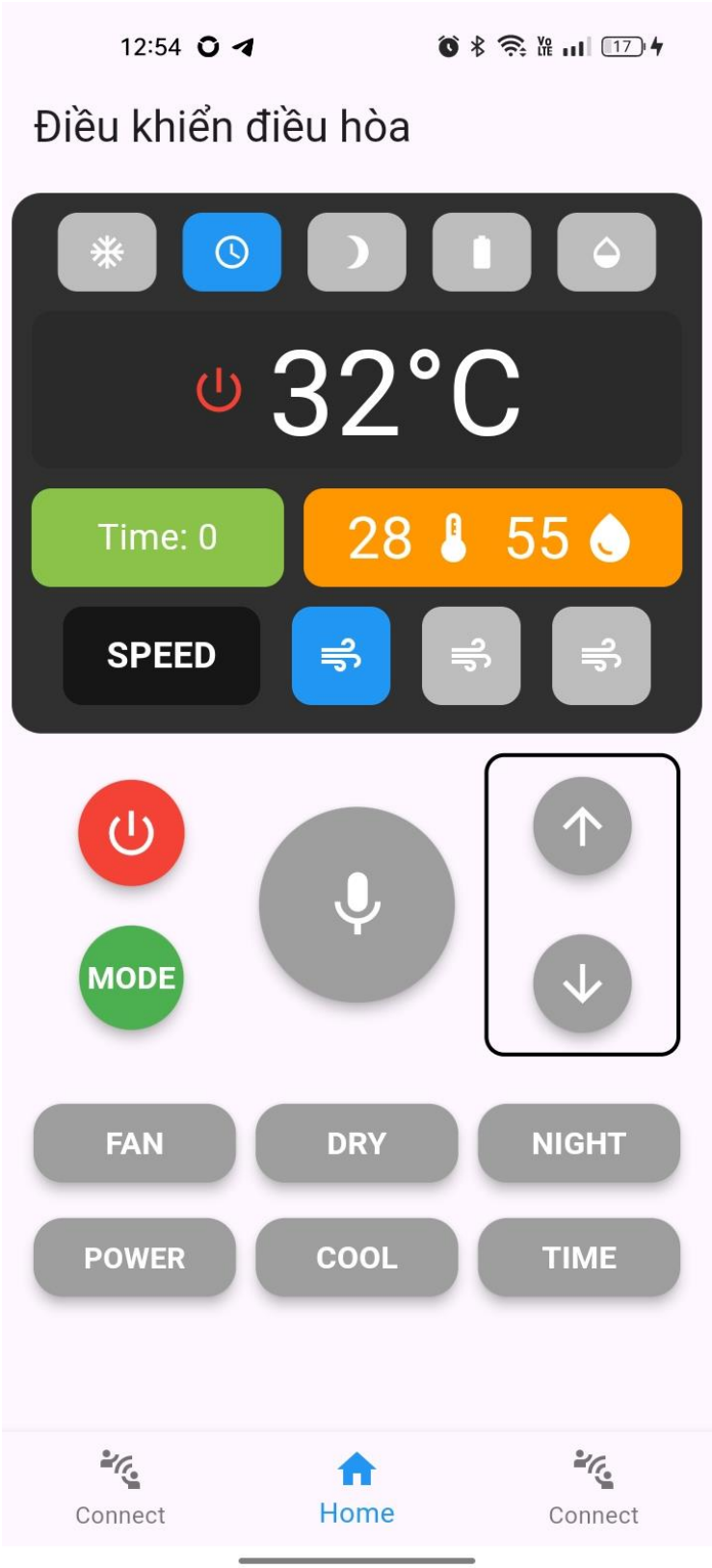
Để triển khai hệ thống, trước tiên cần chạy tệp connect.py để thiết lập kết nối và truyền dữ liệu giữa hệ thống mô phỏng và ứng dụng di động. Sau khi kết nối thành công, tiếp tục khởi chạy tệp main.py để bắt đầu hệ thống mô phỏng. Sau khi hệ thống hoạt động, người dùng có thể truy cập ứng dụng di động để điều khiển các tính năng của hệ thống. Dữ liệu từ hệ thống mô phỏng và ứng dụng di động sẽ được đồng bộ hóa và mọi thay đổi sẽ được ghi đè vào tệp config.py, đảm bảo trạng thái hệ thống luôn được cập nhật chính xác.

4.5. Kiểm thử phần mềm



Hình 7: Hệ thống đang mở

Hệ thống đang mở và có nhiệt độ 32 độ C và độ ẩm 57%. Led ở GPIO đang sáng hệ thống đã hoạt động ổn định.



Hình 8: Hệ thống đang bật trên ứng dụng di động

Hệ thống đang biểu thị ở chế độ bật bằng cách icon hình bặt/tắt có màu đỏ - bật/trắng - tắt.

5. KẾT LUẬN

5.1. Tóm tắt kết quả đạt được

Trong quá trình phát triển và triển khai hệ thống, chúng em đã hoàn thành các chức năng cơ bản với độ ổn định cao, đáp ứng được các yêu cầu ban đầu. Cụ thể, hệ thống đã thực hiện thành công các chức năng điều khiển chính bao gồm bật/tắt điều hòa, cho phép người dùng dễ dàng khởi động hoặc ngừng hoạt động của điều hòa từ cả hệ thống mô phỏng và ứng dụng di động. Chức năng tăng/giảm nhiệt độ cũng đã được tích hợp và hoạt động trơn tru, giúp người dùng điều chỉnh nhiệt độ phòng theo nhu cầu chỉ bằng một vài thao tác đơn giản.

Ngoài ra, tính năng hẹn giờ đã được thực hiện thành công, cho phép người dùng cài đặt khoảng thời gian sau đó điều hòa sẽ tự động tắt. Chức năng này giúp tiết kiệm năng lượng và tăng tính tiện lợi cho người dùng, đặc biệt là trong các trường hợp muốn điều hòa hoạt động trong một khoảng thời gian nhất định. Chức năng điều chỉnh tốc độ quạt cũng đã được tích hợp vào hệ thống, mang lại khả năng kiểm soát tốc độ gió trong phòng, từ đó giúp người dùng tạo ra một không gian thoải mái theo ý muốn.

Một trong những thành tựu lớn nhất của hệ thống là khả năng kết nối từ xa thông qua ứng dụng di động, giúp người dùng có thể điều khiển điều hòa mọi lúc, mọi nơi. Ứng dụng di động được phát triển bằng Flutter không chỉ cung cấp giao diện thân thiện và dễ sử dụng mà còn đồng bộ hóa dữ liệu một cách mượt mà với hệ thống thông qua kết nối WebSocket. Dữ liệu từ ứng dụng và hệ thống mô phỏng được đồng bộ và ghi đè vào file config.py, đảm bảo rằng mọi thay đổi đều được lưu trữ và phản ánh ngay lập tức trên cả hai hệ thống.

Toàn bộ các chức năng cơ bản đã được thử nghiệm và triển khai thành công, tạo ra một hệ thống điều hòa thông minh có thể điều khiển từ xa với nhiều tùy chọn cho người dùng, giúp tăng tính linh hoạt và tiện lợi trong quá trình sử dụng. Những kết quả này đã đặt nền tảng cho các bước phát triển tiếp theo, khi phần cứng thật có sẵn, hệ thống có thể hoạt động hoàn chỉnh mà không cần thay đổi lớn về kiến trúc hoặc logic điều khiển.

5.2. Kết quả chưa đạt được

Mặc dù chúng em đã đạt được nhiều kết quả đáng khích lệ, nhưng vẫn còn một số vấn đề chưa thể hoàn thiện hoàn toàn. Trước hết, hệ thống chưa hoạt động trơn tru như

mong muốn. Cần có sự tối ưu hóa hơn nữa để đảm bảo tính ổn định và mượt mà trong mọi tình huống vận hành. Ngoài ra, giao diện người dùng của ứng dụng di động vẫn chưa được tối ưu hóa về mặt thẩm mỹ, cần cải tiến để giao diện trở nên hấp dẫn và thân thiện hơn.

Một hạn chế lớn nữa là khả năng phân tích dữ liệu người dùng chưa được phát triển hoàn thiện. Mục tiêu của hệ thống là có thể thu thập và phân tích dữ liệu sử dụng từ người dùng để đưa ra các điều chỉnh thông minh, tự động điều chỉnh chế độ hoạt động của điều hòa theo thói quen và nhu cầu cá nhân của người dùng. Tuy nhiên, hiện tại, tính năng này vẫn chưa được tích hợp hoàn chỉnh và cần thêm thời gian phát triển.

Vấn đề truyền dữ liệu giữa hệ thống và ứng dụng di động cũng chưa được hoàn toàn mượt mà. Mặc dù hệ thống có thể gửi và nhận dữ liệu thông qua kết nối WebSocket, nhưng vẫn xuất hiện hiện tượng dữ liệu truyền không chính xác hoặc bị gián đoạn, dẫn đến các biến không được cập nhật đúng cách. Bên cạnh đó, hệ thống hiện vẫn yêu cầu các thiết bị phải kết nối trong cùng một mạng Wi-Fi để truyền dữ liệu, chưa thể hỗ trợ kết nối từ xa hoàn toàn, hoặc tự động tìm kiếm và kết nối với hệ thống điều hòa thông qua các mạng khác nhau.

Những hạn chế này vẫn cần được khắc phục trong các bước phát triển tiếp theo để đảm bảo hệ thống hoạt động hiệu quả và mang lại trải nghiệm tốt nhất cho người dùng.

5.3. Bài học kinh nhiệm

Trong quá trình phát triển hệ thống, chúng em đã rút ra được nhiều bài học quan trọng. Trước tiên, việc thiết kế và triển khai hệ thống đòi hỏi sự hiểu biết sâu sắc về cả phần cứng và phần mềm, đặc biệt là khi phải làm việc với môi trường mô phỏng thay vì phần cứng thực tế. Sử dụng các thư viện mô phỏng giúp tiết kiệm chi phí và thời gian, nhưng đồng thời cũng làm nổi bật những giới hạn khi chuyển từ môi trường mô phỏng sang thực tế, đòi hỏi phải cẩn thận trong từng bước triển khai để đảm bảo tính tương thích.

Chúng em cũng nhận thấy tầm quan trọng của giao tiếp giữa các thành phần trong hệ thống, đặc biệt là việc truyền dữ liệu giữa Python và Flutter thông qua WebSocket. Điều này đã dạy chúng em rằng việc truyền tải dữ liệu không chỉ đơn giản là gửi và nhận mà còn cần phải đảm bảo tính ổn định, chính xác và đồng bộ hóa dữ liệu để tránh các lỗi phát sinh. Vấn đề về kết nối mạng và yêu cầu các thiết bị phải ở cùng mạng cũng

là một thách thức lớn, giúp chúng em hiểu rõ hơn về các yếu tố ảnh hưởng đến khả năng kết nối từ xa và bảo mật dữ liệu khi làm việc với mạng lưới IoT.

Một bài học quan trọng khác là tính trực quan và thẩm mỹ của giao diện người dùng đóng vai trò quan trọng trong trải nghiệm người dùng. Giao diện cần không chỉ thân thiện mà còn phải tối ưu về mặt tương tác, dễ dàng sử dụng, và hấp dẫn để tạo cảm giác thoải mái khi điều khiển hệ thống.

Cuối cùng, bài học lớn nhất là việc lên kế hoạch và phân tích trước khi phát triển. Từ việc xây dựng kiến trúc hệ thống cho đến triển khai chức năng, nếu không có kế hoạch và phân tích kỹ lưỡng, sẽ dễ dẫn đến các vấn đề phát sinh không mong muốn. Quá trình phát triển dạy chúng em phải luôn linh hoạt, biết cách ứng phó với các tình huống bất ngờ và không ngừng cải tiến hệ thống để đạt được hiệu quả tối ưu.

5.4. Hướng phát triển trong tương lai

Trong tương lai, một trong những ưu tiên hàng đầu của chúng em là tối ưu hóa hiệu suất hệ thống để cải thiện sự ổn định và mượt mà khi vận hành. Việc tối ưu hóa này sẽ bao gồm cải tiến mã nguồn, giảm thiểu độ trễ trong quá trình xử lý, và nâng cấp thuật toán điều khiển để đảm bảo hệ thống phản hồi nhanh chóng và chính xác hơn. Mục tiêu là tạo ra một hệ thống hoạt động trơn tru hơn, mang lại trải nghiệm tốt nhất cho người dùng.

Chúng em cũng hướng đến việc cải tiến giao diện người dùng của ứng dụng di động. Một giao diện trực quan, thẩm mỹ và dễ sử dụng sẽ giúp người dùng tương tác với hệ thống dễ dàng hơn. Thiết kế giao diện sẽ được tối ưu hóa để tương thích với nhiều thiết bị di động, đồng thời cải thiện tính thẩm mỹ để tạo cảm giác thoải mái khi sử dụng.

Một bước tiến quan trọng khác trong tương lai là tích hợp trí tuệ nhân tạo (AI) để phân tích dữ liệu người dùng. Bằng cách thu thập và phân tích dữ liệu sử dụng từ người dùng, hệ thống có thể học hỏi và tự động điều chỉnh chế độ hoạt động của điều hòa dựa trên thói quen của người dùng và điều kiện môi trường. Điều này sẽ giúp hệ thống trở nên thông minh hơn, mang lại sự tiện ích cao và tối ưu hóa năng lượng tiêu thụ.

Ngoài ra, chúng em sẽ tập trung vào cải thiện khả năng truyền dữ liệu và đồng bộ hóa giữa hệ thống mô phỏng và ứng dụng di động. Điều này bao gồm việc khắc phục các vấn đề liên quan đến dữ liệu không chính xác và tăng cường độ tin cậy của kết nối

WebSocket. Mục tiêu là đảm bảo dữ liệu luôn được đồng bộ hóa mượt mà giữa hệ thống và ứng dụng, giúp người dùng kiểm soát hệ thống một cách chính xác.

Một tính năng quan trọng khác trong tương lai là phát triển khả năng kết nối từ xa toàn diện. Hiện tại, hệ thống yêu cầu thiết bị phải ở trong cùng một mạng để truyền dữ liệu. Chúng em sẽ hướng tới việc hỗ trợ kết nối từ xa, cho phép người dùng điều khiển hệ thống điều hòa từ bất cứ đâu, thông qua mạng internet mà không cần phải cùng một mạng Wi-Fi, mở rộng tính linh hoạt và tiện ích cho người dùng.

Hơn nữa, hệ thống sẽ cần phát triển tính năng tự động tìm kiếm và kết nối với điều hòa, giúp người dùng dễ dàng hơn trong việc kết nối ứng dụng với hệ thống điều hòa. Điều này sẽ giúp loại bỏ thao tác thủ công, cho phép ứng dụng tự động phát hiện và kết nối với điều hòa ngay cả khi các thiết bị không chia sẻ cùng một mạng.

Cuối cùng, một yếu tố không thể thiếu trong hướng phát triển tương lai là bảo mật. Khi kết nối từ xa và truyền dữ liệu qua mạng trở nên phổ biến, việc tích hợp các biện pháp bảo mật như mã hóa dữ liệu và xác thực người dùng là vô cùng cần thiết. Điều này sẽ giúp bảo vệ hệ thống khỏi các mối đe dọa tiềm tàng từ an ninh mạng, đảm bảo tính an toàn và bảo mật cho người dùng khi sử dụng hệ thống.

5.5. Kết luận chung

Sau quá trình nghiên cứu và phát triển, trong bài báo này, chúng em đã hoàn thành và triển khai thành công các chức năng cơ bản của hệ thống điều hòa thông minh. Các tính năng như bật/tắt, điều chỉnh nhiệt độ, hẹn giờ, chỉnh tốc độ quạt, và đặc biệt là kết nối và điều khiển từ xa qua ứng dụng di động đã hoạt động ổn định và đồng bộ giữa hệ thống mô phỏng và ứng dụng di động. Chúng em cũng đã thiết kế được giao diện người dùng dễ sử dụng, trực quan, và hệ thống đã cho thấy sự linh hoạt khi điều khiển từ xa, mang lại trải nghiệm tiện lợi cho người dùng.

Tuy nhiên, bên cạnh những thành tựu đạt được, vẫn còn một số hạn chế mà chúng em chưa hoàn thiện. Hệ thống chưa đạt được sự mượt mà hoàn toàn, cần có sự tối ưu hóa để hoạt động trơn tru hơn. Giao diện người dùng hiện tại vẫn cần cải tiến về mặt thẩm mỹ và trải nghiệm người dùng. Khả năng truyền dữ liệu giữa hệ thống và ứng dụng di động đôi lúc chưa ổn định, với một số biến truyền đi không chính xác. Hơn nữa, hệ thống hiện yêu cầu các thiết bị phải ở cùng một mạng Wi-Fi để kết nối, chưa hỗ trợ kết nối từ xa hoàn toàn thông qua mạng internet.

Những hạn chế này sẽ là mục tiêu cho các giai đoạn phát triển tiếp theo, nơi chúng em sẽ tiếp tục hoàn thiện và mở rộng hệ thống, hướng tới việc tạo ra một hệ thống điều hòa thông minh hoạt động ổn định, linh hoạt và đáp ứng nhu cầu người dùng một cách tối ưu.

TÀI LIỆU THAM KHẢO

[1]. Nguyễn, Tất Bảo Thiện (2019), Lập trình hệ thống nhúng với Raspberry /, Thanh niên, 9786049833175.

[2]. Achim Rettberg (2018), Embedded system design: topics, techniques, and trends: IFIP TC10 Working Conference--International Embedded Systems Symposium (IESS): May 30-June 1, 2007, Irvine (CA), USA /, Springer, 9783319858128.