

HANOI NATIONAL UNIVERSITY

VIETNAM JAPAN UNIVERSITY

-----oo-----



BÁO CÁO IOT

YEAR 2024

SMART HOME

Hanoi, April 2024

THÔNG TIN CHUNG VỀ ĐỀ TÀI

Tên đề tài: Smart Home

Thời gian thực hiện: 10/05/2024 - 12/06/2024

Nhóm trưởng: Bùi Thế Trung

Điện thoại: 0373.104.304

Email: 21110108@st.vju.ac.vn

Chương trình: Khoa học và kỹ thuật máy tính

DANH SÁCH THÀNH VIÊN THAM GIA THỰC HIỆN ĐỀ TÀI

| TT | Họ và tên | Nội dung công việc |
|----|-----------------|--|
| 1 | Bùi Thế Trung | Lên kế hoạch + Thiết kế chung + Tổng kết báo cáo |
| 2 | Phạm Minh Tuấn | Thiết kế mô hình + Viết báo cáo |
| 3 | Lê Hải Nam | Thiết kế app MIT + Viết báo cáo |
| 4 | Đỗ Trung Hiếu | Thiết kế app MIT + Viết báo cáo |
| 5 | Nguyễn Duy Tùng | Thiết kế mô hình + Viết báo cáo |

MỤC LỤC

| | |
|---|-----------|
| I. GIỚI THIỆU..... | 9 |
| 1.1. Tổng quan..... | 9 |
| 1.2 Tình hình nghiên cứu trong và ngoài nước..... | 10 |
| 1.3 Mục tiêu, phạm vi báo cáo..... | 11 |
| 1.4. Ý nghĩa thực tiễn..... | 12 |
| II. LÝ THUYẾT..... | 15 |
| 2.1. Tìm hiểu về vi điều khiển ESP32..... | 15 |
| 2.1.1. Giới thiệu về ESP32..... | 15 |
| 2.1.2. Cấu hình của ESP32..... | 16 |
| CPU..... | 16 |
| Hỗ trợ 2 giao tiếp không dây..... | 17 |
| Hỗ trợ tất cả các loại giao tiếp..... | 17 |
| Cảm biến tích hợp trên chip ESP32..... | 17 |
| Bảo mật..... | 17 |
| Nguồn điện hoạt động..... | 18 |
| Ứng dụng..... | 18 |
| 2.1.3. Sơ đồ chân của ESP32..... | 18 |
| Chân Input Only..... | 19 |
| Chân tích hợp Flash trên ESP32..... | 19 |
| Chân cảm biến điện dung..... | 20 |
| Bộ chuyển đổi tương tự sang số ADC (Analog to Digital Converter)..... | 21 |
| Bộ chuyển đổi số sang tương tự DAC (Digital to Analog Converter)..... | 22 |
| Các chân thời gian thực RTC..... | 22 |

| | |
|--|-----------|
| Chân PWM..... | 23 |
| Chân I2C..... | 23 |
| Chân SPI..... | 24 |
| Chân ngắt ngoài..... | 24 |
| 2.2. Tìm hiểu về MIT App Inventor 2..... | 24 |
| 2.2.1. Tìm hiểu tổng quan về MIT App Inventor 2..... | 24 |
| 2.2.2. Thiết kế giao diện..... | 27 |
| 2.2.3. Lập trình chức năng cho app..... | 29 |
| 2.3. Tìm hiểu về các linh kiện..... | 37 |
| 2.3.1. Các loại cảm biến..... | 37 |
| 2.3.2. Động cơ Relay..... | 42 |
| 2.3.3. Một số thiết bị khác..... | 43 |
| III. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG..... | 48 |
| 3.1. Yêu cầu thiết kế..... | 48 |
| 3.2. Phân tích..... | 48 |
| 3.2.1. Yêu cầu về nguồn điện..... | 48 |
| 3.2.2. Khả năng kết nối..... | 49 |
| 3.2.3. Tính khả thi trong lắp đặt..... | 49 |
| 3.4 Sơ đồ khối chi tiết..... | 51 |
| 3.5. Tổng thể mô hình..... | 53 |
| IV. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM..... | 54 |
| 4.1 Yêu cầu thiết kế..... | 54 |
| 4.2 Phân tích..... | 54 |
| 4.3 Sơ đồ khối tổng quát..... | 55 |
| 4.4 Sơ đồ khối chi tiết..... | 57 |
| 4.5 Giải thích code..... | 60 |

| | |
|--|-----------|
| V. KẾT QUẢ THỰC HIỆN..... | 67 |
| 5.1 Kết quả phần mềm..... | 67 |
| 5.2 Kết quả phần cứng..... | 67 |
| 5.3 Đánh giá hiệu suất..... | 68 |
| VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN..... | 69 |
| 6.1 Kết luận..... | 69 |
| 6.2 Hướng phát triển cho đề tài..... | 69 |
| VII. TÀI LIỆU THAM KHẢO..... | 71 |

Lời Cảm Ơn

Trong bối cảnh công nghệ ngày càng phát triển, Internet of Things (IoT) đã và đang biến đổi cuộc sống của chúng ta một cách sâu sắc, đặc biệt là trong lĩnh vực nhà thông minh (Smart Home). Nghiên cứu của chúng em về ứng dụng IoT trong Smart Home đã giúp chúng em hiểu rõ hơn về tiềm năng to lớn của việc kết nối các thiết bị thông minh và tầm quan trọng của việc xây dựng một môi trường sống tiện nghi, an toàn và tiết kiệm năng lượng.

Chúng em xin gửi lời cảm ơn chân thành đến thầy Bùi Huy Kiên, người đã tận tình hướng dẫn và truyền đạt những kiến thức quý báu trong suốt quá trình tôi học khóa Internet of Things (IoT). Sự tận tâm, kiến thức sâu rộng và sự hỗ trợ nhiệt tình của thầy đã giúp chúng em vượt qua thử thách và đồng hành trong quá trình nghiên cứu và phát triển đề tài về ứng dụng IoT trong Smart Home. Những bài học sâu sắc từ thầy không chỉ là nguồn động lực to lớn mà còn là nền tảng quan trọng cho sự phát triển cá nhân và nâng cao kiến thức của chúng em.

Xin chân thành cảm ơn!

Danh mục viết tắt

| Viết tắt | Giải thích |
|-----------------|---|
| ESP32 | Một vi điều khiển tích hợp WiFi và Bluetooth, được sử dụng trong các ứng dụng IoT. |
| MIT AI2 | MIT App Inventor 2, nền tảng phát triển ứng dụng di động dành cho hệ điều hành Android. |
| IoT | Internet of Things, mạng lưới các thiết bị kết nối Internet. |
| ADC | Analog to Digital Converter, bộ chuyển đổi từ tín hiệu tương tự sang số. |
| DAC | Digital to Analog Converter, bộ chuyển đổi từ tín hiệu số sang tương tự. |
| PWM | Pulse Width Modulation, điều chế độ rộng xung. |
| I2C | Inter-Integrated Circuit, một giao thức truyền thông nối tiếp. |
| SPI | Serial Peripheral Interface, giao diện ngoại vi nối tiếp. |
| GPIO | General Purpose Input/Output, các chân vào/ra đa dụng. |
| RTOS | Real-Time Operating System, hệ điều hành thời gian thực. |

Danh mục ảnh

| | |
|---|----|
| Hình 2.1.1: Vị điều khiển ESP32 (Nguồn https://ozlaboratory.tistory.com)..... | 16 |
| Hình 2.1.2: Cấu hình của ESP32..... | 17 |
| Hình 2.1.3: Sơ đồ chân ESP32 (Nguồn: https://docs.espressif.com)..... | 20 |
| Hình 2.2.1 : Giao diện tổng quan của MIT App Inventor 2..... | 26 |
| Hình 2.2.2.a. Giao diện chính của app..... | 28 |
| Hình 2.2.2.b : Giao diện app trên điện thoại..... | 30 |
| Hình 2.2.3.a : Thuật toán kết nối qua Bluetooth..... | 31 |
| Hình 2.2.3.b : Thuật toán kết nối thành công thì gọi đến các biến khác..... | 32 |
| Hình 2.2.3.c : Thuật toán lấy thông số cảm biến để hiển thị lên app..... | 33 |
| Hình 2.2.3.d : Thuật toán cảm biến người và cảnh báo đèn..... | 33 |
| Hình 2.2.3.e : Thuật toán cảm biến chạm và cảnh báo đèn..... | 34 |
| Hình 2.2.3.f : Thuật toán cảm biến khói và cảnh báo đèn..... | 35 |
| Hình 2.2.3.g : Thuật toán cho động cơ quạt..... | 35 |
| Hình 2.2.3.h : Thuật toán cho bóng đèn..... | 36 |
| Hình 2.2.3.x : Thuật toán cho điều khiển chức năng tự động..... | 37 |
| Hình 2.2.3.z : Thuật toán cho tắt toàn bộ thiết bị quạt và đèn..... | 37 |
| Hình 2.3.1.a : Cảm biến nhiệt độ và độ ẩm (nguồn: vn.szks-kuongshun)..... | 38 |
| Hình 2.3.1.b : Cảm biến khói, khí ga MQ-2 (nguồn: dientu360.com)..... | 39 |
| Hình 2.3.1.c : Cảm biến ánh sáng LDR (nguồn: nshopvn.com)..... | 40 |
| Hình 2.3.1.d : Cảm biến siêu âm HC-SR04 (nguồn: amazon.in)..... | 41 |
| Hình 2.3.1.e : Cảm biến chạm Touch SenSor (nguồn: nshopvn.com)..... | 42 |
| Hình 2.3.2 : Động cơ relay 2 module (nguồn: dientusti.com)..... | 43 |
| Hình 2.3.3.a : Buzzer chủ động 5V (nguồn: linhkienthuduc.com)..... | 44 |
| Hình 2.3.3.b : Đèn LED 5V (nguồn: linhkienthuduc.com)..... | 45 |
| Hình 2.3.3.c : Quạt tản nhiệt 5V (nguồn: cytrontech.vn)..... | 47 |

| | |
|--|----|
| Hình 2.3.3.d : Bóng đèn LED (nguồn: denphucloc.com.vn)..... | 48 |
| Hình 3.3 : Sơ đồ khói tổng quát phần cứng..... | 51 |
| Hình 3.4 : Sơ đồ khói cảm biến..... | 53 |
| Hình 3.5.a : Tổng thể mô hình..... | 54 |
| Hình 3.5.b : Sơ đồ mạch thiết kế..... | 54 |
| Hình 4.3 : Lưu đồ giải thuật tổng quát..... | 56 |
| Hình 4.4.a : Lưu đồ giải thuật phần điều khiển thiết bị..... | 58 |
| Hình 4.4.b : Lưu đồ giải thuật cảm biến..... | 60 |
| Hình 4.5.a : Đảm bảo Bluetooth và Bluedroid đã được bật trong cấu hình..... | 61 |
| Hình 4.5.b : Cấu hình, định nghĩa các chân..... | 62 |
| Hình 4.5.c : Đọc nhiệt độ, độ ẩm..... | 63 |
| Hình 4.5.d : Chương trình điều khiển quạt, đèn tự động..... | 64 |
| Hình 4.5.e : Chương trình cảnh báo khói, bật đèn và còi cảnh báo..... | 65 |
| Hình 4.5.g : Chương trình cảnh báo chạm, bật đèn và còi cảnh báo..... | 66 |
| Hình 4.5.h : Chương trình cảnh báo người, bật đèn..... | 66 |
| Hình 4.5.i : Chương trình thay đổi các chế độ tự động, thủ công, tắt tất cả..... | 67 |

I. GIỚI THIỆU

1.1. Tổng quan

Trong kỷ nguyên công nghệ 4.0, việc tích hợp các thiết bị thông minh vào cuộc sống hàng ngày đang trở thành xu hướng phổ biến, mang lại nhiều tiện ích và cải thiện chất lượng cuộc sống. Một trong những ứng dụng nổi bật của công nghệ này là hệ thống tự động hóa nhà thông minh (Smart Home Automation).

Hệ thống tự động hóa nhà thông minh cho phép người dùng kiểm soát và giám sát các thiết bị điện trong nhà một cách tự động và từ xa. Thông qua việc sử dụng các cảm biến và vi điều khiển, hệ thống có thể thực hiện các chức năng như điều chỉnh ánh sáng, nhiệt độ, an ninh, và cảnh báo cháy nổ. Tất cả các thiết bị này có thể được điều khiển và giám sát qua ứng dụng di động, mang lại sự tiện lợi và an toàn tối đa cho người dùng.

Báo cáo này tập trung vào việc thiết kế và triển khai một hệ thống tự động hóa nhà thông minh sử dụng vi điều khiển ESP32. ESP32 là một vi điều khiển mạnh mẽ, tích hợp WiFi và Bluetooth, hỗ trợ hệ điều hành thời gian thực (RTOS), giúp hệ thống có thể hoạt động một cách hiệu quả và đồng bộ. Ứng dụng di động đi kèm được phát triển trên nền tảng MIT App Inventor 2, cho phép người dùng dễ dàng điều khiển và giám sát hệ thống từ xa.

Hệ thống được thiết kế với mục tiêu chính là cải thiện sự tiện nghi, an toàn, và hiệu quả năng lượng cho ngôi nhà. Các chức năng chính bao gồm tự động điều chỉnh đèn và quạt dựa trên cảm biến nhiệt độ và ánh sáng, giám sát an ninh với cảm biến siêu âm và cảm biến chạm, cùng hệ thống cảnh báo khói và cháy nổ. Báo cáo sẽ đi sâu vào các khía cạnh kỹ thuật, từ thiết kế phần cứng và phần

mềm, quy trình lắp đặt và cài đặt, cho đến thử nghiệm và đánh giá hiệu suất của hệ thống.

Qua việc xây dựng và triển khai hệ thống này, chúng tôi hy vọng mang lại một giải pháp tự động hóa nhà thông minh toàn diện, nâng cao chất lượng cuộc sống và thúc đẩy sự phát triển của các công nghệ thông minh trong tương lai.

1.2 Tình hình nghiên cứu trong và ngoài nước

Đối với tình hình nước ngoài:

Trên phạm vi toàn cầu, Nhà Thông Minh được xem là một lĩnh vực có dư địa phát triển rất lớn. Các đại gia công nghệ như Microsoft, Apple, Google, Samsung... tỏ ra sôt sắng với xu hướng này bằng một loạt vụ thâu tóm. Google mua lại Nest (hãng sản xuất bộ điều khiển nhiệt độ thông minh và thiết bị báo khói), Samsung ra mắt hệ thống nhà thông minh khép kín trong các thiết bị của hãng, Apple giới thiệu nền tảng phát triển ứng dụng nhà thông minh HomeKit.[3]

Ngoài ra, khi nhắc đến những thương hiệu Nhà Thông Minh đến từ nước ngoài, không thể không kể đến: hãng Schneider của Pháp, Smartg4 của Mỹ, Gamma của Gamma JSC, Arteor của hãng Legrand (Pháp), My Home của hãng Bticino (Ý), WattStopper (Mỹ), Mhouse, Home access, Came với đại lý chính thức là NTMC, Hager (Pháp), Crestron (Mỹ)... Các sản phẩm thiết bị điện thông minh đến từ nước ngoài đều mang trong mình những ưu điểm vượt trội về thiết kế, tính năng với những giải pháp tiên tiến và hiện đại.[4]

Đối với tình hình trong nước:

Các doanh nghiệp Việt Nam vẫn không ngừng bám đuổi công nghệ trên thế giới, không ngừng nghiên cứu về Nhà Thông Minh. Thực tế, đã có nhiều doanh

nghiệp của chúng ta đã cho ra mắt các sản phẩm phẩm Nhà Thông Minh “Made in Vietnam”.

Ta có thể kể tới:

- Nhà Thông Minh BKAV của tập đoàn BKAV. Đây là doanh nghiệp đã sớm đón đầu xu thế công nghệ - IoT. Họ đã bắt đầu tham gia vào việc phát triển các thiết bị và phần mềm về Smart Home từ năm 2004. Cho đến nay, giải pháp Nhà Thông Minh BKAV đã hoàn chỉnh, sẵn sàng phục vụ mọi nhu cầu của người dân.[5]
- Nhà Thông Minh Lumi, tập trung mạnh vào phân khúc nhà thông minh trung và cao cấp trên thị trường Việt Nam. Ra mắt giải pháp Nhà Thông Minh vào năm 2012, cho đến nay, họ đã phát triển và hoàn chỉnh sản phẩm của mình.[6]
- Không thể không nhắc tới Nhà Thông Minh ASIC, một dạng start up thành công ở Việt Nam và đang ngày càng phát triển.

1.3 Mục tiêu, phạm vi báo cáo

Mục tiêu:

Báo cáo này nhằm thiết kế và triển khai một hệ thống tự động hóa nhà thông minh, sử dụng vi điều khiển ESP32 và ứng dụng di động phát triển trên nền tảng MIT App Inventor 2. Hệ thống được xây dựng với mục tiêu chính là:

- Tự động hóa các thiết bị điện trong nhà, bao gồm đèn, quạt, và các thiết bị khác, thông qua các chế độ điều khiển thủ công, tự động dựa trên cảm biến, và chế độ tắt hoàn toàn.
- Giám sát nhiệt độ và ánh sáng trong nhà thông qua cảm biến nhiệt độ DHT11 và cảm biến ánh sáng LDR, với khả năng hiển thị thông tin trên màn hình OLED và ứng dụng di động.
- Tăng cường an ninh nhà ở bằng cách phát hiện người tiếp cận cửa thông qua cảm biến siêu âm, và cảnh báo khi có chạm vào cửa bằng cảm biến chạm của ESP32.

- Đảm bảo an toàn phòng cháy chữa cháy bằng cách phát hiện khói với cảm biến MQ2 và kích hoạt báo động.

Phạm vi:

Phạm vi của báo cáo bao gồm các khía cạnh sau:

- **Thiết kế phần cứng:** Báo cáo sẽ mô tả chi tiết về các thành phần phần cứng cần thiết cho hệ thống, bao gồm vi điều khiển ESP32, các loại cảm biến (nhiệt độ, ánh sáng, khói, siêu âm, chạm), các thiết bị điều khiển (relay, LED, buzzers), và các thiết bị đầu ra như bóng đèn và quạt.
- **Phát triển phần mềm:** Bao gồm việc lập trình cho vi điều khiển ESP32 bằng Arduino IDE, tích hợp freeRTOS để đảm bảo hoạt động thời gian thực, và phát triển ứng dụng di động trên nền tảng MIT App Inventor 2 để điều khiển và giám sát hệ thống.
- **Quy trình lắp đặt và cài đặt:** Hướng dẫn chi tiết về cách lắp đặt các thành phần phần cứng, cài đặt và cấu hình phần mềm, kết nối và vận hành hệ thống.
- **Thử nghiệm và đánh giá:** Thực hiện các thử nghiệm để đánh giá hiệu suất và độ tin cậy của hệ thống trong các điều kiện khác nhau, phân tích kết quả và đưa ra nhận xét về khả năng ứng dụng thực tế.
- **Đề xuất cải tiến:** Dựa trên kết quả thử nghiệm và đánh giá, báo cáo sẽ đưa ra các đề xuất cải tiến để nâng cao hiệu suất và mở rộng tính năng của hệ thống trong tương lai.

1.4. Ý nghĩa thực tiễn

Nâng cao chất lượng cuộc sống

Hệ thống tự động hóa nhà thông minh mang lại sự tiện lợi và thoải mái cho người sử dụng bằng cách tự động điều chỉnh các thiết bị điện trong nhà theo các điều kiện môi trường và yêu cầu của người dùng. Việc này không chỉ giúp tiết kiệm thời gian và công sức mà còn nâng cao chất lượng cuộc sống.

Tiết kiệm năng lượng

Với khả năng tự động điều chỉnh đèn, quạt và các thiết bị khác dựa trên cảm biến nhiệt độ và ánh sáng, hệ thống giúp tối ưu hóa việc sử dụng năng lượng, giảm thiểu lãng phí và tiết kiệm chi phí điện năng. Điều này có ý nghĩa đặc biệt trong bối cảnh tài nguyên năng lượng ngày càng khan hiếm và giá điện tăng cao.

Tăng cường an ninh và an toàn

Hệ thống tích hợp các cảm biến an ninh như cảm biến siêu âm và cảm biến chạm để phát hiện người lạ tiếp cận và cảnh báo khi có xâm nhập bất hợp pháp, giúp bảo vệ ngôi nhà khỏi các nguy cơ trộm cắp. Ngoài ra, cảm biến khói và hệ thống báo động phòng cháy chữa cháy giúp phát hiện sớm và xử lý kịp thời các tình huống cháy nổ, đảm bảo an toàn cho người sử dụng.

Giám sát và điều khiển từ xa

Với ứng dụng di động phát triển trên nền tảng MIT App Inventor 2, người dùng có thể giám sát và điều khiển hệ thống từ bất kỳ đâu, bất kỳ lúc nào. Điều này mang lại sự linh hoạt và tiện lợi, đặc biệt hữu ích khi người dùng không có mặt tại nhà.

Thúc đẩy ứng dụng công nghệ 4.0

Việc triển khai hệ thống tự động hóa nhà thông minh này không chỉ dừng lại ở việc mang lại lợi ích trực tiếp cho người sử dụng mà còn góp phần thúc đẩy ứng dụng công nghệ 4.0 vào đời sống hàng ngày. Điều này khuyến khích sự phát triển của các sản phẩm và giải pháp công nghệ mới, tạo động lực cho các nghiên cứu và cải tiến trong lĩnh vực tự động hóa và IoT (Internet of Things).

Giáo dục và nghiên cứu

Hệ thống tự động hóa nhà thông minh còn có ý nghĩa lớn trong lĩnh vực giáo dục và nghiên cứu. Nó cung cấp một mô hình thực tế để sinh viên và các nhà nghiên cứu có thể học tập, thử nghiệm và phát triển các ứng dụng mới, góp phần đào tạo nguồn nhân lực chất lượng cao trong lĩnh vực công nghệ thông tin và tự động hóa.

II. LÝ THUYẾT

2.1. Tìm hiểu về vi điều khiển ESP32

2.1.1. Giới thiệu về ESP32

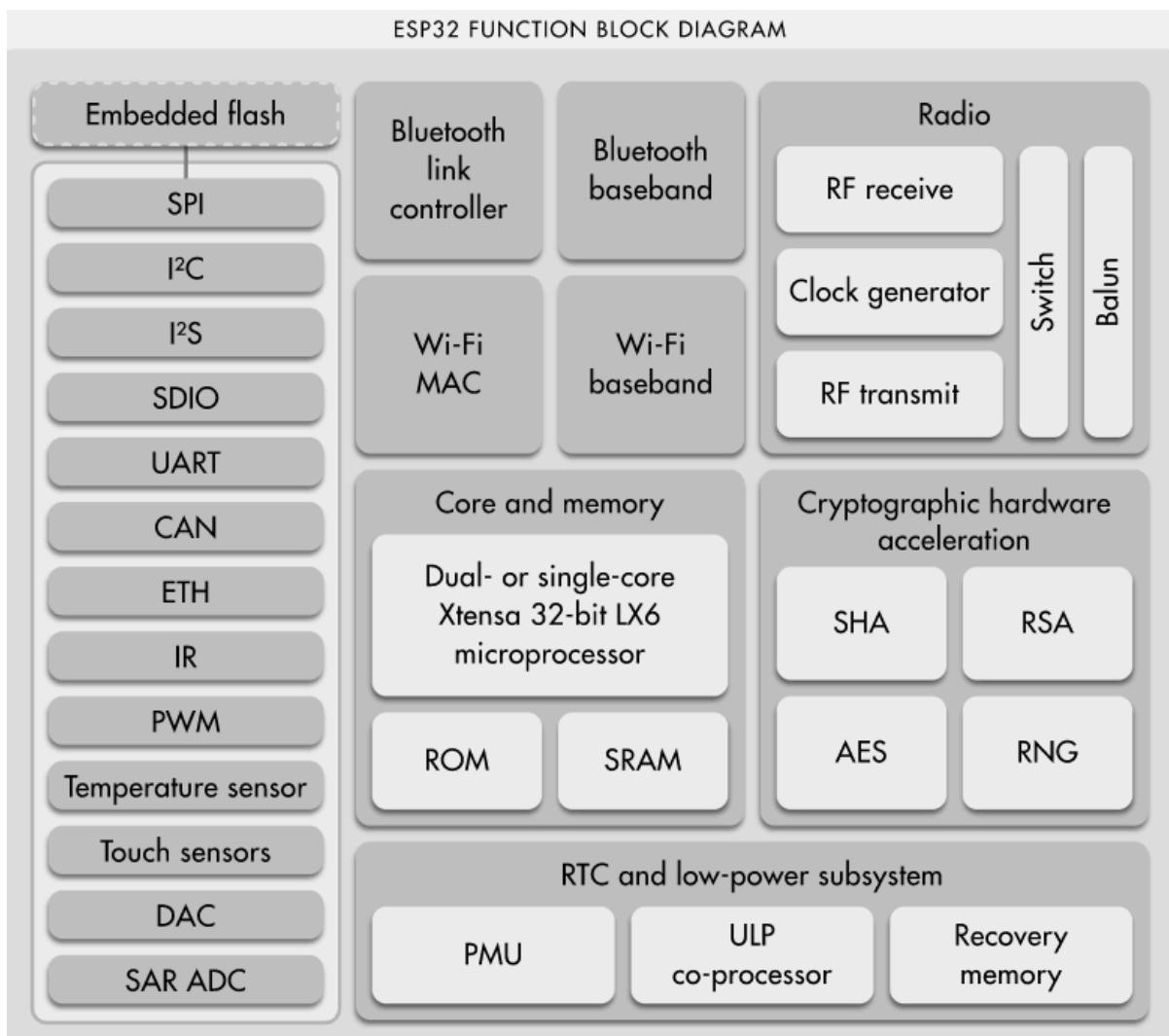


Ảnh 2.1.1: Vi điều khiển ESP32 (Nguồn <https://ozlaboratory.tistory.com>)

ESP32 là một series các vi điều khiển trên một vi mạch giá rẻ, năng lượng thấp có tích hợp WiFi và dual-mode Bluetooth (tạm dịch: Bluetooth chế độ kép). Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 có hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc antenna tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng.

ESP32 được chế tạo và phát triển bởi Espressif Systems, một công ty Trung Quốc có trụ sở tại Thượng Hải, và được sản xuất bởi TSMC bằng cách sử dụng công nghệ 40 nm. ESP32 là sản phẩm kế thừa từ vi điều khiển ESP8266.

2.1.2. Cấu hình của ESP32



Hình 2.1.2: Cấu hình của ESP32

CPU

- CPU: Xtensa Dual-Core LX6 microprocessor.
- Chạy hệ 32 bit
- Tốc độ xử lý từ 160 MHz đến 240 MHz
- ROM: 448 Kb
- Tốc độ xung nhịp từ 40 Mhz ÷ 80 Mhz (có thể tùy chỉnh khi lập trình)
- RAM: 520 Kb SRAM liền chip. Trong đó 8 Kb RAM RTC tốc độ cao
 - 8 Kb RAM RTC tốc độ thấp (dùng ở chế độ DeepSleep).

Hỗ trợ 2 giao tiếp không dây

- Wi-Fi: 802.11 b/g/n/e/i
- Bluetooth: v4.2 BR/EDR và BLE

Hỗ trợ tất cả các loại giao tiếp

- 2 bộ chuyển đổi số sang tương tự (DAC) 8 bit
- 18 kênh bộ chuyển đổi tương tự sang số (ADC) 12 bit.
- 2 cổng giao tiếp I²C
- 3 cổng giao tiếp UART
- 3 cổng giao tiếp SPI (1 cổng cho chip FLASH)
- 2 cổng giao tiếp I²S
- 10 kênh ngõ ra điều chế độ rộng xung (PWM)
- SD card/SDIO/MMC host
- Ethernet MAC hỗ trợ chuẩn: DMA và IEEE 1588
- CAN bus 2.0
- IR (TX/RX)

Cảm biến tích hợp trên chip ESP32

- 1 cảm biến Hall (cảm biến từ trường)
- 1 cảm biến đo nhiệt độ
- Cảm biến chạm (điện dung) với 10 đầu vào khác nhau.

Bảo mật

- Hỗ trợ tất cả các tính năng bảo mật chuẩn IEEE 802.11, bao gồm WFA, WPA/WPA2 và WAPI
- Khởi động an toàn (Secure boot)
- Mã hóa flash (Flash encryption)

- 1024-bit OTP, lên đến 768-bit cho khách hàng
- Tăng tốc phần cứng mật mã: AES, SHA-2, RSA, mật mã đường cong elliptic (ECC – elliptic curve cryptography), bộ tạo số ngẫu nhiên (RNG – random number generator)

Nguồn điện hoạt động

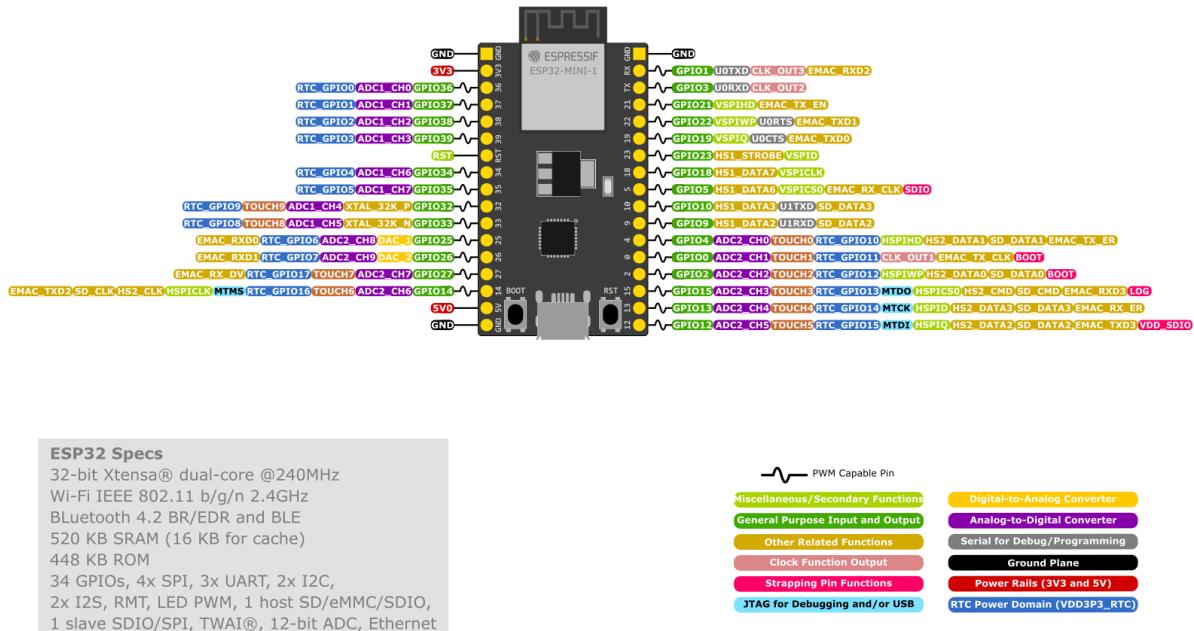
- Điện áp hoạt động: 2,2V ÷ 3,6V
- Nhiệt độ hoạt động: -40oC ÷ + 85oC
- Số cổng GPIO: 36

Ứng dụng

- Module được dùng nhiều trong các ứng dụng thu thập dữ liệu và điều khiển thiết bị qua WiFi, Bluetooth.
- Sử dụng cho các ứng dụng tiết kiệm năng lượng, điều khiển mạng lưới cảm biến, mã hóa hoặc xử lý tiếng nói, xử lý Analog-Digital trong các ứng dụng phát nhạc, hoặc với các file MP3...
- Module cũng có thể dùng cho các thiết bị điện tử đeo tay như đồng hồ thông minh...

2.1.3. Sơ đồ chân của ESP32

Chip ESP32 bao gồm 48 chân với nhiều chức năng khác nhau. Không phải tất cả các chân đều lộ ra trên các module ESP32 và một số chân không thể được sử dụng.



Hình 2.1.3: Sơ đồ chân ESP32 (Nguồn: <https://docs.espressif.com>)

Chân Input Only

GPIO từ 34 đến 39 là các chân chỉ đầu vào. Các chân này không có điện trở kéo lên hoặc kéo xuống bên trong. Chúng không thể được sử dụng làm đầu ra, vì vậy chỉ sử dụng các chân này làm đầu vào:

- GPIO34
- GPIO35
- GPIO36
- GPIO39

Chân tích hợp Flash trên ESP32

GPIO 6 đến GPIO 11 dùng để kết nối Flash SPI trên chip ESP-WROOM-32, không khuyến khích sử dụng cho các mục đích sử dụng khác.

- GPIO6 (SCK/CLK)
- GPIO7 (SDO/SD0)
- GPIO8 (SDI/SD1)
- GPIO9 (SHD/SD2)
- GPIO10 (SWP/SD3)
- GPIO11 (CSC/CMD)

Chân cảm biến điện dung

ESP32 có 10 cảm biến điện dung bên trong. Các cảm biến này có thể phát hiện được sự thay đổi về điện áp cảm ứng trên các chân GPIO. Các chân cảm ứng điện dung cũng có thể được sử dụng để đánh thức ESP32 khỏi chế độ ngủ sâu (deep sleep).

Các chân ESP32 này có chức năng như 1 nút nhấn cảm ứng, có thể phát hiện sự thay đổi về điện áp cảm ứng trên chân.

Các cảm biến cảm ứng bên trong đó được kết nối với các GPIO sau:

- TOUCH0 (GPIO4)
- TOUCH1 (GPIO0)
- TOUCH2 (GPIO2)
- TOUCH3 (GPIO15)
- TOUCH4 (GPIO13)
- TOUCH5 (GPIO12)
- TOUCH6 (GPIO14)
- TOUCH7 (GPIO27)
- TOUCH8 (GPIO33)
- TOUCH9 (GPIO32)

Bộ chuyển đổi tương tự sang số ADC (Analog to Digital Converter)

ESP32 có 18 kênh đầu vào ADC 12 bit (trong khi ESP8266 chỉ có 1 kênh ADC 10 bit). Đây là các GPIO có thể được sử dụng làm ADC và các kênh tương ứng:

- ADC1_CH0 (GPIO36)
- ADC1_CH1 (GPIO37)
- ADC1_CH2 (GPIO38)
- ADC1_CH3 (GPIO39)
- ADC1_CH4 (GPIO32)
- ADC1_CH5 (GPIO33)
- ADC1_CH6 (GPIO34)
- ADC1_CH7 (GPIO35)
- ADC2_CH0 (GPIO4)
- ADC2_CH1 (GPIO0)
- ADC2_CH2 (GPIO2)
- ADC2_CH3 (GPIO15)
- ADC2_CH4 (GPIO13)
- ADC2_CH5 (GPIO12)
- ADC2_CH6 (GPIO14)
- ADC2_CH7 (GPIO27)
- ADC2_CH8 (GPIO25)
- ADC2_CH9 (GPIO26)

Các kênh đầu vào ADC có độ phân giải 12 bit. Điều này có nghĩa là bạn có thể nhận được các giá trị tương tự từ 0 đến 4095, trong đó 0 tương ứng với 0V và 4095 đến 3,3V. Bạn cũng có thể thiết lập độ phân giải cho các kênh thông qua chương trình (code).

Bộ chuyển đổi số sang tương tự DAC (Digital to Analog Converter)

Có 2 kênh DAC 8 bit trên ESP32 để chuyển đổi tín hiệu số sang tương tự. Các kênh này chỉ có độ phân giải 8 bit, nghĩa là có giá trị từ $0 \div 255$ tương ứng với $0 \div 3.3V$

Đây là các kênh DAC:

- DAC1 (GPIO25)
- DAC2 (GPIO26)

Các chân thời gian thực RTC

Các chân này có tác dụng đánh thức ESP32 khi trong chế độ ngủ sâu (Low Power Mode). Sử dụng như 1 chân ngắt ngoài.

Các chân RTC:

- RTC_GPIO0 (GPIO36)
- RTC_GPIO3 (GPIO39)
- RTC_GPIO4 (GPIO34)
- RTC_GPIO5 (GPIO35)
- RTC_GPIO6 (GPIO25)
- RTC_GPIO7 (GPIO26)
- RTC_GPIO8 (GPIO33)
- RTC_GPIO9 (GPIO32)
- RTC_GPIO10 (GPIO4)
- RTC_GPIO11 (GPIO0)
- RTC_GPIO12 (GPIO2)
- RTC_GPIO13 (GPIO15)
- RTC_GPIO14 (GPIO13)

- RTC_GPIO15 (GPIO12)
- RTC_GPIO16 (GPIO14)
- RTC_GPIO17 (GPIO27)

Chân PWM

ESP32 LED PWM có 16 kênh độc lập có thể được cấu hình để tạo tín hiệu PWM với các thuộc tính khác nhau. Tất cả các chân có thể hoạt động như đầu ra đều có thể được sử dụng làm chân PWM (GPIO từ 34 đến 39 không thể tạo PWM).

Để xuất PWM, bạn cần định nghĩa các thông số này trong code:

- Tần số tín hiệu
- Chu kỳ làm việc
- Kênh PWM
- Chân GPIO xuất tín hiệu ra

Chân I2C

ESP32 có hai kênh I2C và bất kỳ chân nào cũng có thể được đặt làm SDA hoặc SCL. Khi sử dụng ESP32 với Arduino IDE, các chân I2C mặc định là:

- GPIO21 (SDA)
- GPIO22 (SCL)

Nếu các bạn muốn sử dụng chân khác cho việc điều khiển I2C có thể sử dụng câu lệnh:

```
Wire.begin(SDA, SCL);
```

Chân SPI

Theo mặc định, ánh xạ chân cho SPI là:

| SPI | MOSI | MISO | CLK | CS |
|-------------|---------|---------|---------|---------|
| VSPI | GPIO 23 | GPIO 19 | GPIO 18 | GPIO 5 |
| HSPI | GPIO 13 | GPIO 12 | GPIO 14 | GPIO 15 |

Chân ngắt ngoài

Tất cả các chân ESP32 đều có thể sử dụng ngắt ngoài.

2.2. Tìm hiểu về MIT App Inventor 2

2.2.1. Tìm hiểu tổng quan về MIT App Inventor 2

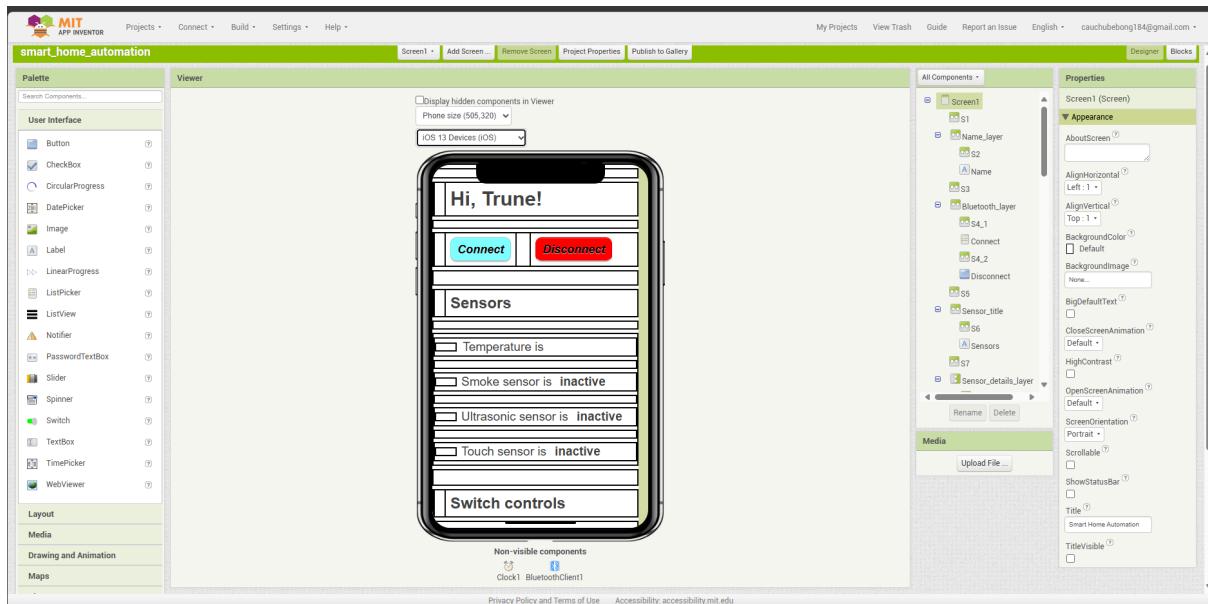
Giới thiệu MIT App Inventer:

MIT App Inventor là một ứng dụng web nguồn mở ban đầu được cung cấp bởi Google và hiện tại được duy trì bởi Viện Công nghệ Massachusetts (MIT). Nền tảng cho phép nhà lập trình tạo ra các ứng dụng phần mềm cho hệ điều hành Android (OS). Bằng cách sử dụng giao diện đồ họa, nền tảng cho phép người dùng kéo và thả các khối mã (blocks) để tạo ra các ứng dụng có thể chạy trên thiết bị Android. Đến thời điểm 07/2017, phiên bản iOS của nền tảng này đã bắt đầu được đưa vào thử nghiệm bởi Thunkable, là một trong các nhà cung cấp ứng dụng web cho ngôn ngữ này.

Mục tiêu cốt lõi của MIT App Inventor là giúp đỡ những người chưa có kiến thức về ngôn ngữ lập trình từ trước có thể tạo ra những ứng dụng có ích trên hệ điều hành Android. Phiên bản mới nhất là MIT App Inventor 2.

Ngày nay, MIT đã hoàn thiện App Inventor và nó được chia sẻ ngay trên tài khoản Google. Các lập trình viên mới bắt đầu hoặc bất kỳ ai muốn tạo ra ứng dụng Android chỉ cần vào địa chỉ web của MIT, nhập thông tin tài khoản Google, và từ những mảnh ghép nhỏ, xây dựng những ý tưởng của mình.

Sau đây là giao diện tổng quan của MIT App Inventor 2:



Hình 2.2.1 : Giao diện tổng quan của MIT App Inventor 2

Trên giao diện chính của App Inventor 2, nhìn từ trên xuống dưới, chúng ta có:

+ Hàng menu:

- **Projects:** chứa tất cả các dự án của chúng ta.
- **Connect:** mục này giúp kết nối với các device khác để xem demo app, các device này có thể là: giả lập có sẵn trong App Inventor 2, giả lập bên thứ ba (như Genymotion), hoặc là một chiếc smart phone.

- **Build:** sau khi xây dựng app xong, ta xuất ra file .apk để cài lên thiết bị android.
- **Help:** hỗ trợ người dùng với rất nhiều mục.
- Tiếp theo là các menu hỗ trợ khác: **My Projects, Gallery, Guide, Report an Issue, Ngôn ngữ...**

+ Phía dưới gồm 4 layout:

- Palette
- Viewer
- Components
- Properties

Palette: Chứa các thành phần có thể đặt lên trên Screen như: Button, Label, Image, Listview, Video player, Đến các thanh phần chức năng không nhìn thấy trên Screen như: BLE extension, Notifier, các sensors,

Viewer: Hiển thị giao diện screen. Kéo thả các thành phần từ khung Palette sang đây để thiết kế giao diện cho phần mềm.

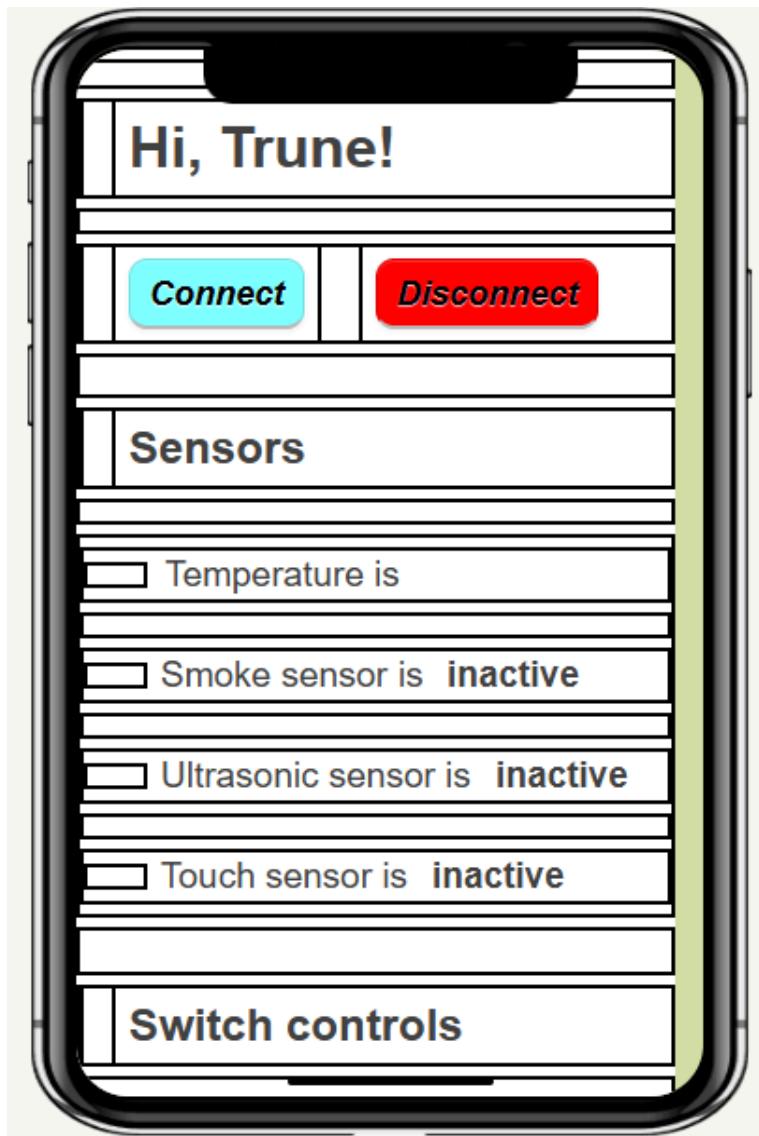
Components: Sơ đồ cây thể hiện cấu trúc các thành phần đã được bố trí trên Screen.

Properties: Hiển thị thuộc tính của component tương ứng được chọn.

Để xây dựng một ứng dụng android, ta có 2 phần cơ bản là **Design** và **Block**. Phần Design là để thiết kế giao diện cho ứng dụng. Phần Block là để lập trình cho ứng dụng. Nhưng đối với Mit App Inventer, ta sẽ lập trình bằng cách tạo ra các thuật toán hợp lý để thực hiện các chức năng, bằng việc kéo thả các lệnh trong phần Block.

2.2.2. Thiết kế giao diện

Phần giao diện của chúng ta sẽ gồm nhiều Screen: Screen 1, Screen 2, Screen 3. Screen nghĩa là màn hình, nó sẽ được hiển thị lên màn hình của thiết bị. Screen 1 là màn hình chính, màn hình đầu tiên xuất hiện khi chúng ta mở app lên.



Hình 2.2.2.a. Giao diện chính của app

Giao diện này được thiết kế để cung cấp thông tin trực quan và tương tác trực tiếp với các thành phần của hệ thống nhà thông minh, cho phép người dùng dễ dàng quản lý và điều khiển từ xa.

Nút kết nối:

- **Kết nối (màu xanh):** Cho phép người dùng kết nối với hệ thống.
- **Ngắt kết nối (màu đỏ):** Cho phép người dùng ngắt kết nối với hệ thống.

Bảng điều khiển cảm biến:

Mỗi cảm biến có một mục riêng trên giao diện và hiện trạng thái hoạt động của nó (hoạt động hoặc không hoạt động). Các cảm biến bao gồm:

- Nhiệt độ
- Cảm biến khói
- Cảm biến siêu âm
- Cảm biến chạm

Điều khiển công tắc:

Người dùng có thể chọn giữa ba chế độ để điều khiển các thiết bị như đèn và quạt:

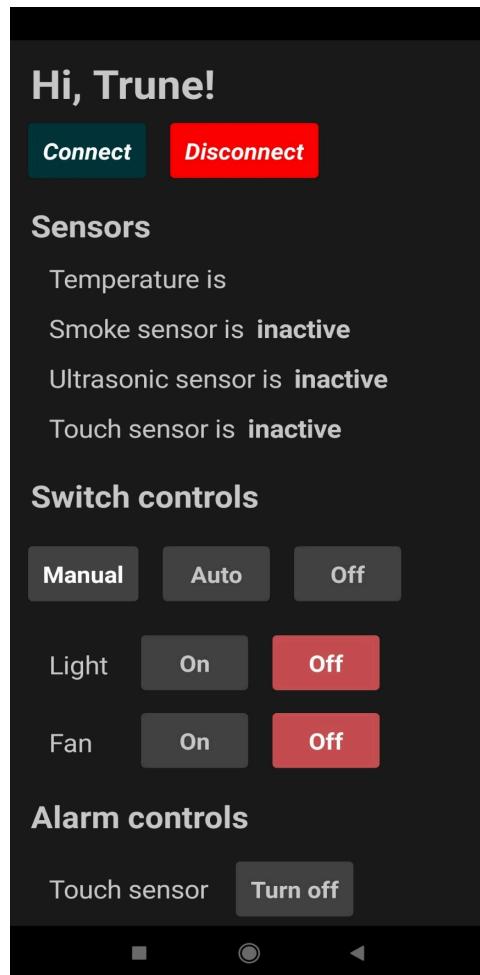
- Thủ công (màu xanh)
- Tự động (màu xanh lá cây)
- Tắt (màu đỏ)

Điều khiển báo động:

Cho phép người dùng kích hoạt hoặc hủy kích hoạt báo động cho các tính năng an ninh như cảm biến chạm ở cửa.

Có một hộp kiểm cho "Cảm biến chạm" và một nút "Tắt" để hủy kích hoạt báo động.

Giao diện app sau khi cài trên điện thoại

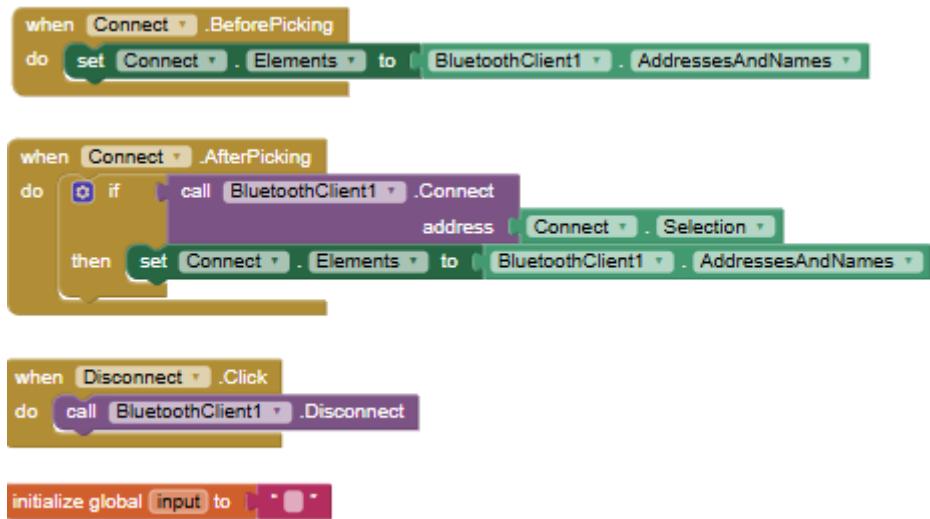


Hình 2.2.2.b : Giao diện app trên điện thoại

2.2.3. Lập trình chức năng cho app

Sau khi thiết kế xong giao diện, để các thành phần của giao diện có thể hoạt động, thì ta phải lập trình cho chúng.

Đầu tiên chúng ta cần thuật toán để kết nối Bluetooth:



Hình 2.2.3.a : Thuật toán kết nối qua Bluetooth

Khối: "when Connect .BeforePicking"

- Mục đích:** Khi người dùng bắt đầu thao tác kết nối, khối này sẽ thiết lập danh sách các thiết bị Bluetooth có sẵn để kết nối.
- Hoạt động:** Danh sách được lấy từ thuộc tính AddressesAndNames của BluetoothClient1, chứa địa chỉ và tên của các thiết bị.
- Ứng dụng:** Khi người dùng nhấp vào danh sách thả xuống để kết nối, họ sẽ được cung cấp một danh sách các thiết bị có sẵn được cập nhật.

Khối: "when Connect .AfterPicking"

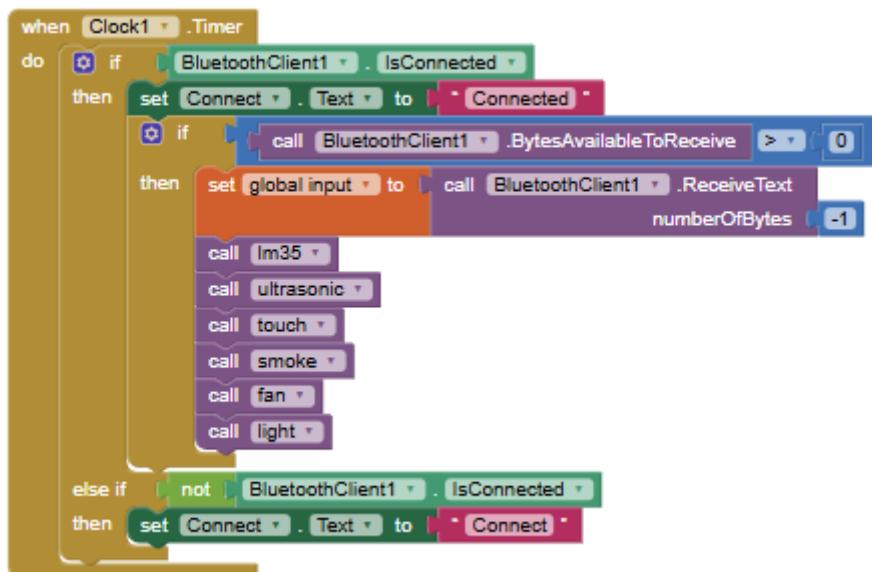
- Mục đích:** Sau khi người dùng chọn một thiết bị từ danh sách, khối này xử lý quá trình kết nối đến thiết bị đã chọn.
- Hoạt động:** Sử dụng phương thức Connect của BluetoothClient1 với địa chỉ của thiết bị được chọn (Connect .Selection), để thiết lập kết nối.
- Điều kiện:** Chỉ thực hiện kết nối nếu khối if đánh giá là true, có nghĩa là có thiết bị được chọn từ danh sách.

- **Cập nhật danh sách:** Sau khi thử kết nối, danh sách các thiết bị trong dropdown menu được cập nhật lại để phản ánh trạng thái mới nhất của các kết nối.

Khoi: "when Disconnect .Click"

- **Mục đích:** Cho phép người dùng ngắt kết nối Bluetooth hiện tại thông qua một nút "Disconnect".
- **Hoạt động:** Khi nút "Disconnect" được nhấn, phương thức Disconnect của `Bluetooth`

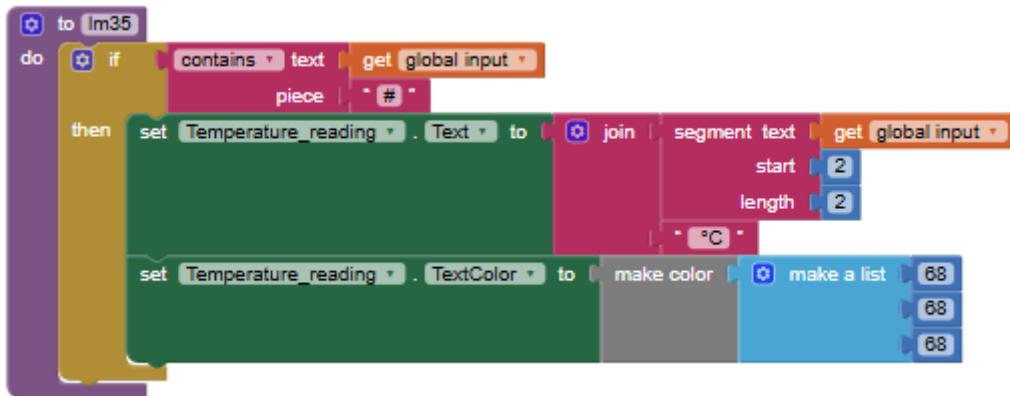
Tiếp theo sau khi kết nối thành công thì gọi đến các biến khác:



Hình 2.2.3.b : Thuật toán kết nối thành công thì gọi đến các biến khác

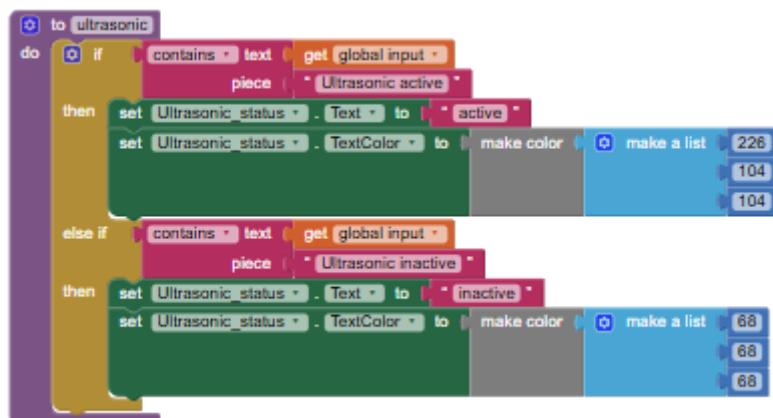
Chúng ta sẽ lần lượt xem qua các thuật toán có trong dự án

Thuật toán để lấy dữ liệu nhiệt độ được gửi lên app:



Hình 2.2.3.c : Thuật toán lấy thông số cảm biến để hiển thị lên app

Tương tự, chúng ta có thuật toán lấy dữ liệu phát hiện người, khói, chạm...



Hình 2.2.3.d : Thuật toán cảm biến người và cảnh báo đèn

Giải thích thuật toán:

Kiểm tra điều kiện trong biến global input:

Khỏi if kiểm tra xem chuỗi trong biến global input có chứa từ "Ultrasonic active" hay không.

Nếu có, thực hiện các thao tác trong khôi then:

- Thiết lập văn bản của biến Ultrasonic_status thành "active".
- Đặt màu của văn bản Ultrasonic_status thành màu xanh lá cây, được tạo từ RGB (228, 104, 104).

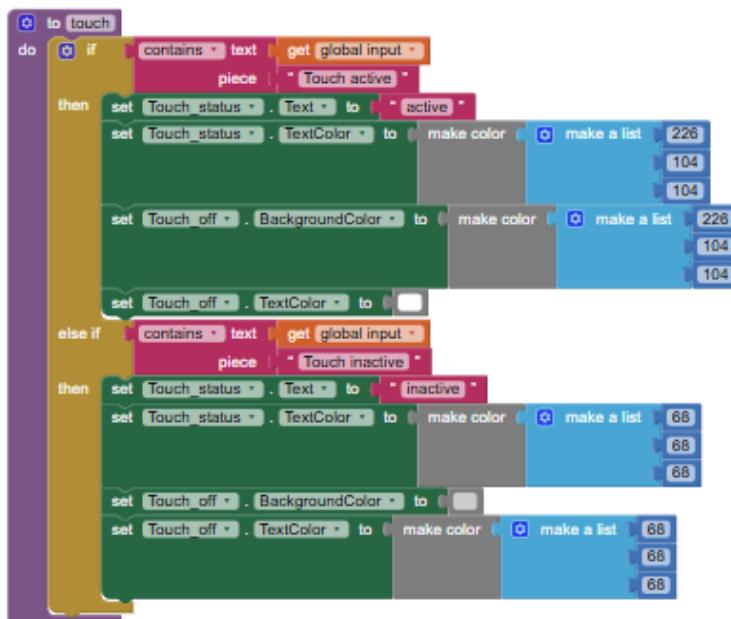
Kiểm tra điều kiện thay thế:

Nếu điều kiện đầu tiên không được thỏa mãn (tức là global input không chứa "Ultrasonic active"), khôi else if sẽ được kiểm tra tiếp theo. Khôi else if này kiểm tra xem chuỗi trong global input có chứa "Ultrasonic inactive" hay không.

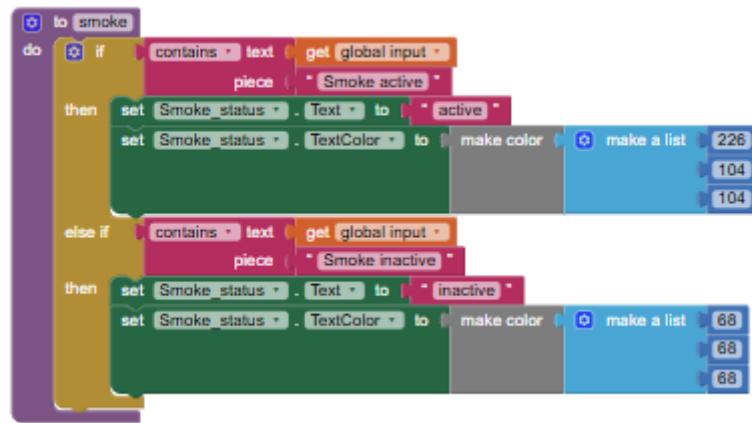
Nếu có, thực hiện các thao tác trong khôi else if:

- Thiết lập văn bản của Ultrasonic_status thành "inactive".
- Đặt màu của văn bản Ultrasonic_status thành màu xám, được tạo từ RGB (68, 68, 68).

Tương tự đối với các biến khác:



Hình 2.2.3.e : Thuật toán cảm biến chạm và cảnh báo đèn

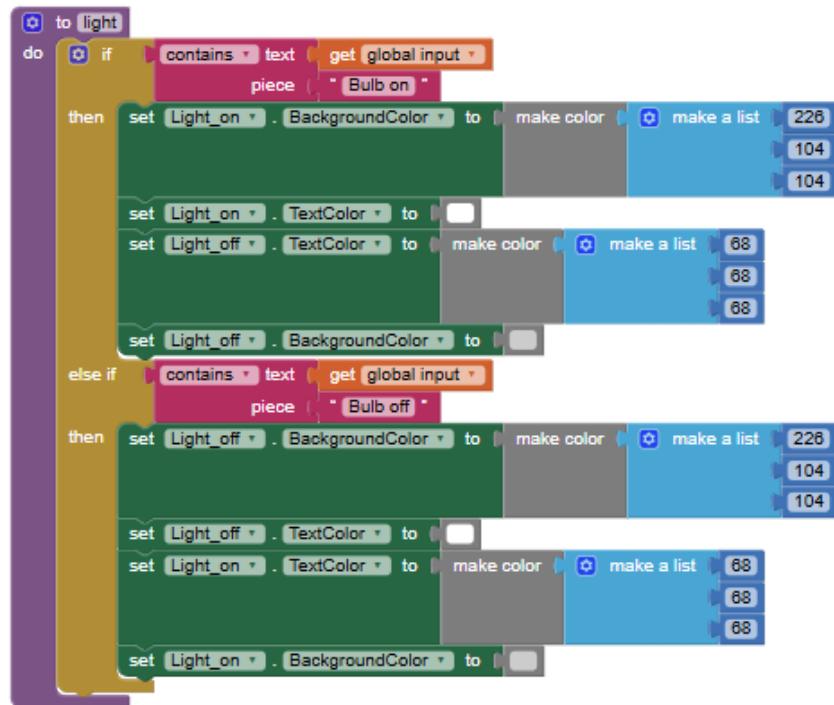


Hình 2.2.3.f : Thuật toán cảm biến khói và cảnh báo đèn

Ngoài ra thuật toán điều khiển cho động cơ quạt và đèn

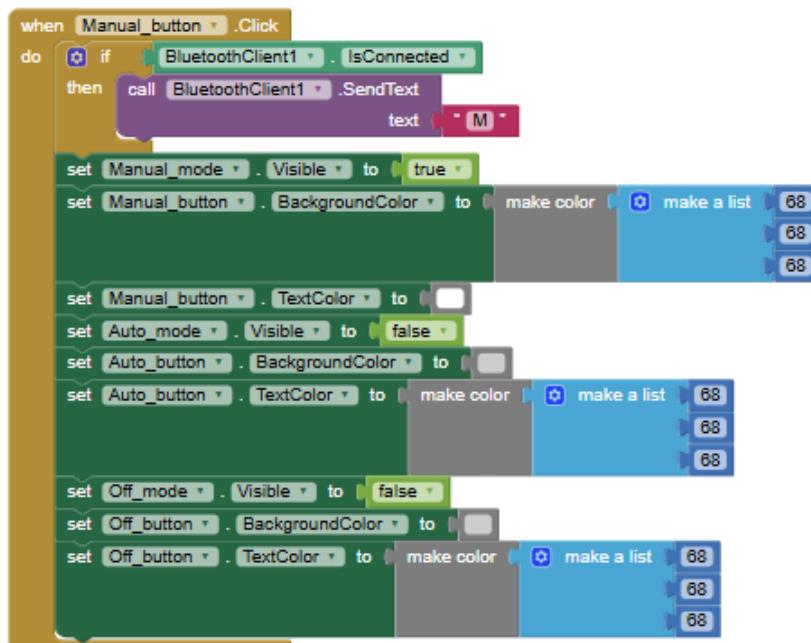


Hình 2.2.3.g : Thuật toán cho động cơ quạt

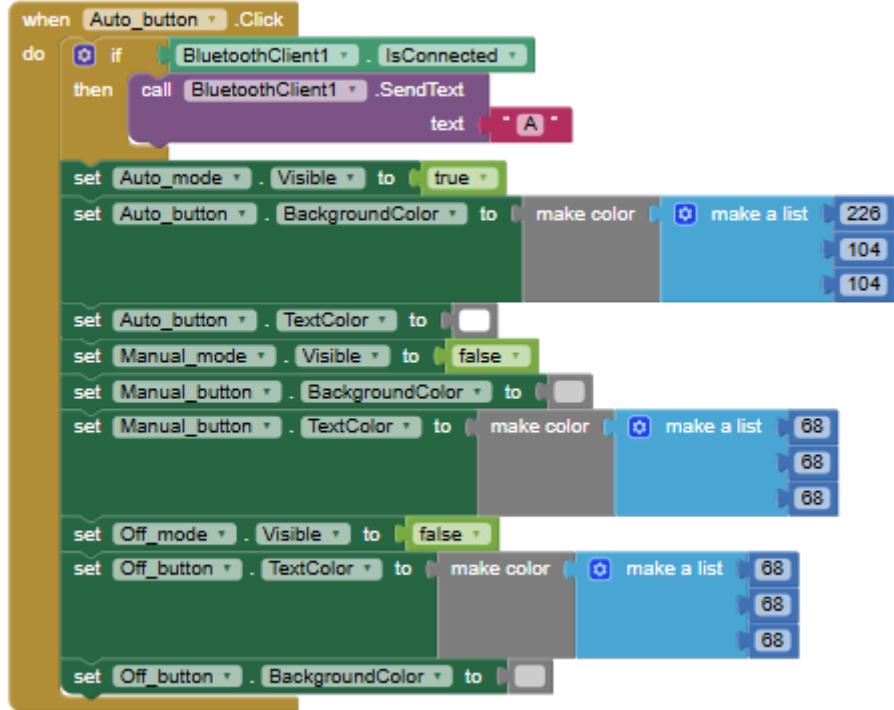


Hình 2.2.3.h : Thuật toán cho bóng đèn

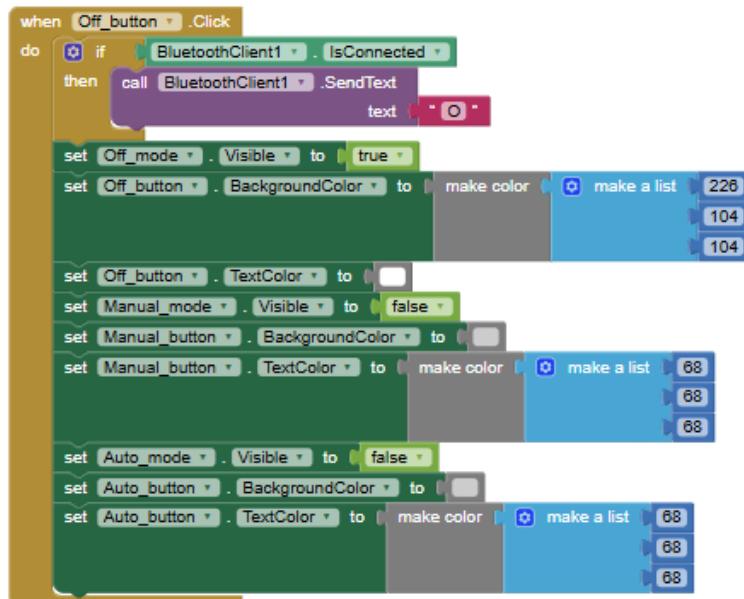
Tiếp theo, chúng ta sẽ đến phần thuật toán dành cho các nút điều khiển



Hình 2.2.3.i : Thuật toán cho điều khiển chức năng thủ công



Hình 2.2.3.x : Thuật toán cho điều khiển chức năng tự động

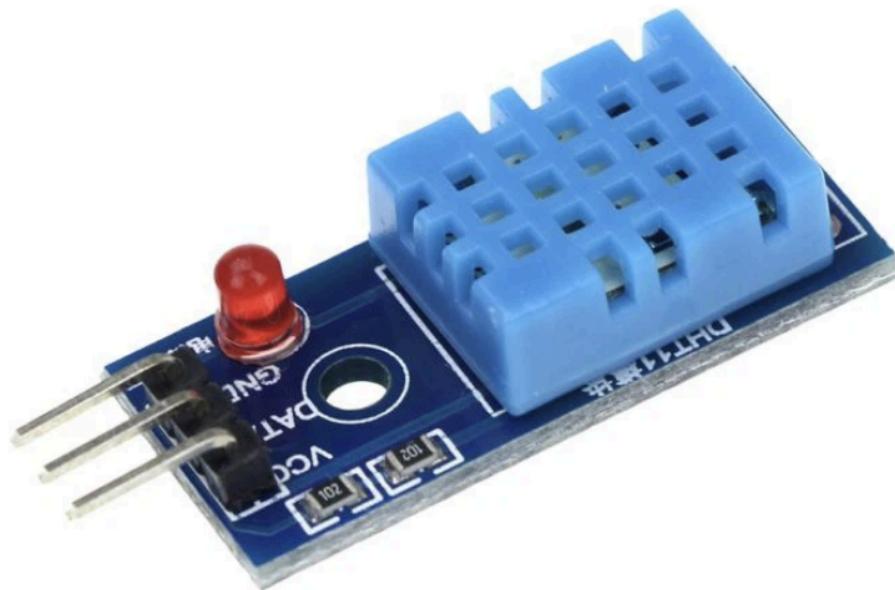


Hình 2.2.3.z : Thuật toán cho tắt toàn bộ thiết bị quạt và đèn

2.3. Tìm hiểu về các linh kiện

2.3.1. Các loại cảm biến

Cảm biến nhiệt độ và độ ẩm (DHT11)



Hình 2.3.1.a : Cảm biến nhiệt độ và độ ẩm (nguồn: vn.szks-kuongshun.com)

Mô tả

Cảm biến DHT11 là một cảm biến kỹ thuật số kết hợp đo nhiệt độ và độ ẩm tương đối. Nó sử dụng một lớp mỏng chất bán dẫn có điện trở thay đổi theo nhiệt độ và độ ẩm để đo các thông số này.

Thông số kỹ thuật

- Điện áp hoạt động: 3.3V - 5V
- Dòng điện tiêu thụ: < 2.5 mA
- Độ chính xác nhiệt độ: $\pm 2^{\circ}\text{C}$
- Phạm vi đo nhiệt độ: $0^{\circ}\text{C} - 50^{\circ}\text{C}$
- Độ chính xác độ ẩm: $\pm 5\%\text{RH}$

- Phạm vi đo độ ẩm: 20% - 80%RH
- Thời gian phản hồi: 1 giây
- Kết nối: 1 dây tín hiệu và 1 dây nguồn
- Giao tiếp: I2C

Cảm biến khói – khí ga (MQ-2)



Hình 2.3.1.b : Cảm biến khói, khí ga MQ-2 (nguồn: dientu360.com)

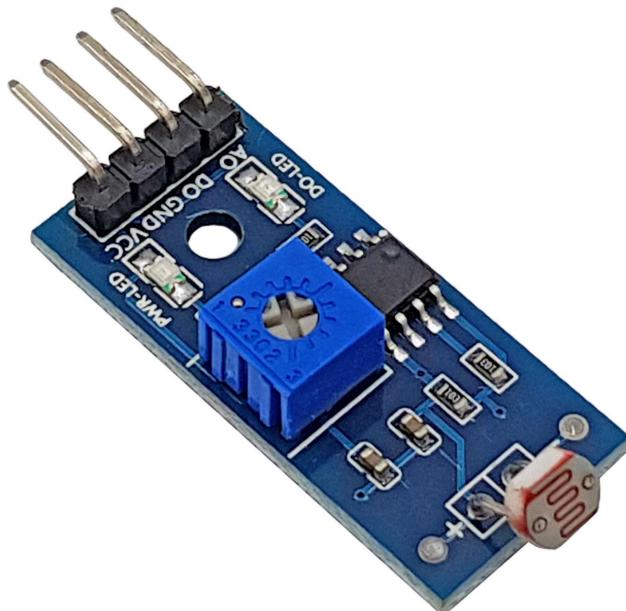
Mô tả:

- Cảm Biến Khí Gas MQ2 sử dụng để phát hiện khí gas trong môi trường.
- Cảm biến có độ nhạy cao khả năng phản hồi nhanh, độ nhạy có thể điều chỉnh được bằng biến trở.
- Cảm biến MQ2 có thể phát hiện khí gas, metan, butan, LPG, khói.

Thông số kỹ thuật:

- Nguồn hoạt động: 5V.
- Loại dữ liệu: Analog.
- Phạm vi phát hiện rộng.
- Tốc độ phản hồi nhanh và độ nhạy cao.
- Mạch đơn giản.
- Ổn định khi sử dụng trong thời gian dài.

Cảm biến ánh sáng (LDR)



Hình 2.3.1.c : Cảm biến ánh sáng LDR (nguồn: nshopvn.com)

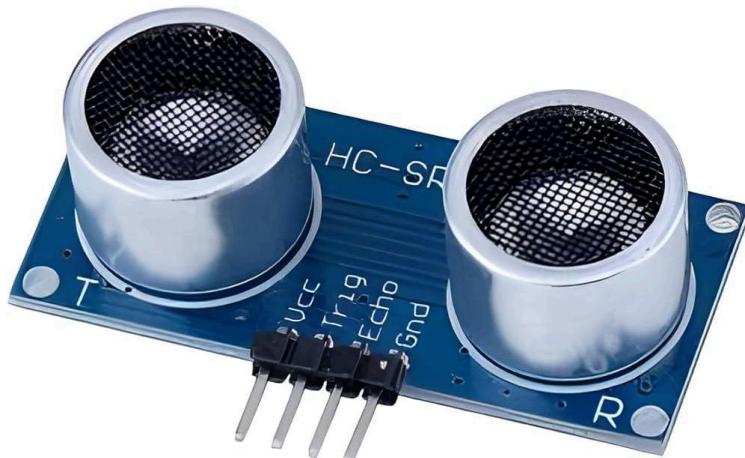
Mô tả

LDR là một loại điện trở có giá trị thay đổi phụ thuộc vào cường độ ánh sáng chiếu vào nó. Khi ánh sáng chiếu vào, điện trở giảm và dòng điện chạy qua nó tăng lên.

Thông số kỹ thuật

- Điện trở tối thiểu: $\sim 10 \text{ k}\Omega$ (trong bóng tối)
- Điện trở tối đa: $\sim 100 \Omega$ (dưới ánh sáng mạnh)
- Thời gian phản hồi: $\sim 100 \text{ ms}$
- Kết nối: 2 đầu nối (đầu vào và đầu ra)

Cảm biến siêu âm (HC-SR04)



Hình 2.3.1.d : Cảm biến siêu âm HC-SR04 (nguồn: [amazon.in](#))

Mô tả

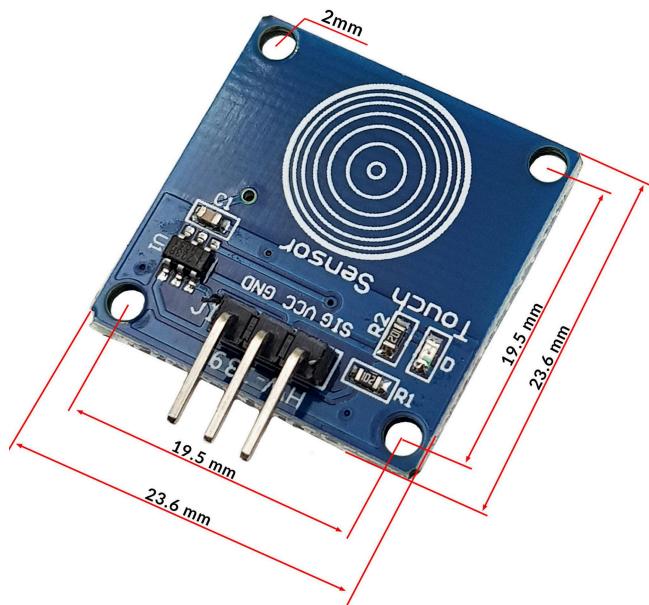
Cảm biến siêu âm phát ra sóng siêu âm và đo thời gian sóng phản xạ trở lại từ vật thể. Dựa vào thời gian này, cảm biến tính toán khoảng cách từ cảm biến đến vật thể.

Thông số kỹ thuật

- Điện áp hoạt động: 5V DC
- Dòng điện tiêu thụ: $\sim 15 \text{ mA}$
- Phạm vi đo: 2 cm - 400 cm
- Độ chính xác: $\sim 3 \text{ mm}$
- Góc quét: $\sim 15^\circ$

- Kết nối: 4 chân (VCC, GND, Trig, Echo)
- Giao tiếp: Tín hiệu kỹ thuật số (Echo)

Cảm biến chạm



Hình 2.3.1.e : Cảm biến chạm Touch SenSor (nguồn: nshopvn.com)

Mô tả

Cảm biến chạm hoạt động dựa trên sự thay đổi điện dung khi một vật thể tiếp xúc với cảm biến. Khi một vật thể chạm vào cảm biến, điện dung thay đổi, tín hiệu được xử lý để phát hiện sự chạm.

Thông số kỹ thuật

- Điện áp hoạt động: 3V - 5V DC
- Dòng điện tiêu thụ: < 1 mA
- Kết nối: 2 chân (NO - normally open và NC - normally closed)
- Tuổi thọ: ~1 triệu lần bấm

2.3.2. Động cơ Relay



Hình 2.3.2 : Động cơ relay 2 module (nguồn: dientusti.com)

Mô tả

Module Relay 2 kênh 5VDC là một bảng mạch điện tử nhỏ gọn, tích hợp 2 relay. Nó có thể được sử dụng để điều khiển các thiết bị điện có dòng điện cao bằng tín hiệu điều khiển từ vi điều khiển hoặc mạch điện tử khác. Mỗi relay được điều khiển độc lập bằng tín hiệu logic cao hoặc logic thấp.

Thông số kỹ thuật

- Điện áp hoạt động: 5VDC (DC 5V)
- Dòng điện tiêu thụ:
 - Cuộn dây: 70mA (tối đa)
 - Relay: 10A (tối đa)
- Điện áp tải:

- DC: 30VDC (tối đa)
- AC: 250VAC (tối đa)
- Dòng điện tải: 10A (tối đa)
- Loại Relay: JQC-3FF-S-Z
- Kích thước: 45mm x 30mm
- Kết nối:
 - VCC: Nguồn 5V DC
 - GND: Ground
 - IN1/IN2: Tín hiệu điều khiển relay 1/2 (logic cao/thấp)
 - NO1/NO2: Tiếp điểm thường mở relay 1/2
 - NC1/NC2: Tiếp điểm thường đóng relay 1/2
 - COM1/COM2: Tiếp điểm chung relay 1/2

2.3.3. Một số thiết bị khác

Buzzer 5V



Hình 2.3.3.a : Buzzer chủ động 5V (nguồn: linhkienthuduc.com)

Mô tả

Buzzer chủ động 5V là một loại loa nhỏ gọn, được thiết kế để phát ra âm thanh khi được cấp nguồn. Nó thường được sử dụng trong các dự án điện tử để báo hiệu, cảnh báo, hoặc tạo hiệu ứng âm thanh.

Thông số kỹ thuật

- Điện áp hoạt động: 5V DC
- Dòng điện tiêu thụ: Khoảng 20 mA (tùy thuộc vào nhà sản xuất)
- Âm thanh: Tiếng bíp (beep) liên tục hoặc có thể điều chỉnh tần số
- Kích thước: Khoảng 12 mm đường kính, 10 mm chiều cao
- Kết nối: 2 chân:
 - Chân dương (+): Nối với nguồn dương 5V
 - Chân âm (-): Nối với ground

Đèn LED 5V



Hình 2.3.3.b : Đèn LED 5V (nguồn: linhkienthuduc.com)

Mô tả

LED là viết tắt của cụm từ Light Emitting Diode - Nghĩa là Diode phát sáng, các Dilde phát ra ánh sáng hoặc tia hồng ngoại, tia tử ngoại. Đèn LED RGB là hệ thống chiếu sáng có sự kết hợp của 3 nguồn sáng gồm đỏ, xanh lá, xanh dương và khả năng đổi màu liên tục.

Thông số kỹ thuật

- Kích thước: Đường kính 5mm
- Điện áp hoạt động: Tùy thuộc vào màu sắc, thường từ 1.8V - 3.3V (ví dụ: Đỏ: 1.8V, Xanh lá: 2.0V, Xanh dương: 3.0V).
- Dòng điện hoạt động: Khoảng 20 mA (tùy thuộc vào màu sắc và nhà sản xuất)
- Màu sắc: Có nhiều màu sắc khác nhau: đỏ, xanh lá, xanh dương, vàng, trắng, tím, hồng, vv.
- Góc chiếu sáng: Tùy thuộc vào loại lens, thường từ 120° đến 160°
- Kết nối: 2 chân:
 - Chân dương (+): Nối với cực dương (anode)
 - Chân âm (-): Nối với cực âm (cathode)
- Tuổi thọ: Khoảng 100.000 giờ (tùy thuộc vào điều kiện hoạt động)

Quạt tản nhiệt 5V



Hình 2.3.3.c : Quạt tản nhiệt 5V (nguồn: cytrontech.vn)

Mô tả

Quạt 5V là một loại quạt làm mát nhỏ gọn, được thiết kế để hoạt động với nguồn điện 5V DC. Nó thường được sử dụng trong các dự án điện tử, máy tính, thiết bị điện tử gia dụng và các ứng dụng yêu cầu làm mát hiệu quả.

Thông số kỹ thuật

- Điện áp hoạt động: 5V DC
- Dòng điện tiêu thụ: Khoảng 0.2A (tùy thuộc vào nhà sản xuất và tốc độ quay)
- Kích thước: Thường là 40mm x 40mm (hình vuông) hoặc 30mm x 30mm (hình vuông)
- Tốc độ quay: Khoảng 8000 - 10000 vòng/phút (RPM)

- Lưu lượng khí: Khoảng 2.5 CFM (tùy thuộc vào kích thước và tốc độ quay)
- Kết nối: 2 dây:
 - Dây đỏ: Nối với nguồn dương (+5V)
 - Dây đen: Nối với ground
- Độ ồn: Khoảng 25 dBA (tùy thuộc vào tốc độ quay)

Bóng đèn LED



Hình 2.3.3.d : Bóng đèn LED (nguồn: denphucloc.com.vn)

Thông Số Kỹ Thuật Bóng Đèn LED:

- Hình dạng: Bóng đèn hình cầu, đuôi vặn E27.
- Màu sắc ánh sáng: Trắng ấm hoặc trắng ngày, tùy chọn.
- Công suất: 10W (tương đương bóng đèn sợi đốt 60W).
- Hiệu quả chiếu sáng: 800 lumen.
- Tuổi thọ trung bình: 25,000 giờ.
- Nhiệt độ màu: 2700K (trắng ấm) hoặc 5000K (trắng ngày).
- Điện áp: 220V-240V.

III. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

3.1. Yêu cầu thiết kế

Nhiệm vụ luận văn yêu cầu thiết kế một hệ thống vừa có thể điều khiển thiết bị, vừa cảm biến và cảnh báo và hiển thị lên app và vừa có thể tự động điều khiển theo hoàn cảnh.

Phần cứng gồm các phần quan trọng như sau: vi điều khiển ESP32, thiết bị để điều khiển (đèn, quạt, động cơ...) và linh kiện cảm biến. Chúng ta sẽ kết hợp các phần này lại với nhau để tạo nên một khối thống nhất. Phản ứng từ vi điều khiển ra các linh kiện, ta sẽ sử dụng Test Board.

3.2. Phân tích

Trong phần này, chúng ta sẽ phân tích các yếu tố ảnh hưởng đến thiết kế hệ thống tự động hóa nhà thông minh, nhằm đảm bảo hệ thống hoạt động hiệu quả, ổn định và đáp ứng được các yêu cầu đề ra.

3.2.1. Yêu cầu về nguồn điện

Nguồn điện là yếu tố quan trọng hàng đầu trong thiết kế hệ thống tự động hóa nhà thông minh. Hệ thống cần một nguồn điện ổn định và đủ mạnh để cung cấp cho tất cả các thành phần, bao gồm vi điều khiển ESP32, các cảm biến, thiết bị điều khiển và màn hình hiển thị. Một số điểm cần lưu ý:

- Điện áp cung cấp:** Vi điều khiển ESP32 thường hoạt động ở điện áp 3.3V, trong khi một số cảm biến và thiết bị điều khiển khác có thể yêu cầu điện áp 5V hoặc 12V. Do đó, cần thiết kế mạch chuyển đổi điện áp phù hợp.

- **Dòng điện:** Tổng dòng điện tiêu thụ của hệ thống cần được tính toán để chọn nguồn cấp điện có công suất phù hợp. Việc này đảm bảo tất cả các thiết bị hoạt động ổn định mà không gây quá tải cho nguồn cấp.
- **Pin dự phòng:** Để đảm bảo hệ thống hoạt động liên tục ngay cả khi mất điện lưới, có thể xem xét việc sử dụng pin dự phòng hoặc bộ lưu điện (UPS).

3.2.2. Khả năng kết nối

Khả năng kết nối là một yếu tố quan trọng khác cần được xem xét để đảm bảo các thành phần của hệ thống có thể giao tiếp với nhau một cách hiệu quả:

- **Kết nối không dây:** ESP32 tích hợp WiFi và Bluetooth, cho phép kết nối không dây với ứng dụng di động và các thiết bị khác. Việc sử dụng kết nối không dây giúp giảm bớt sự phức tạp trong việc đi dây và tăng tính linh hoạt cho hệ thống.
- **Giao tiếp giữa các cảm biến và vi điều khiển:** Các cảm biến và thiết bị điều khiển thường sử dụng các giao thức như I2C, SPI, UART để giao tiếp với vi điều khiển ESP32. Cần đảm bảo các giao thức này được triển khai và cấu hình đúng cách.

3.2.3. Tính khả thi trong lắp đặt

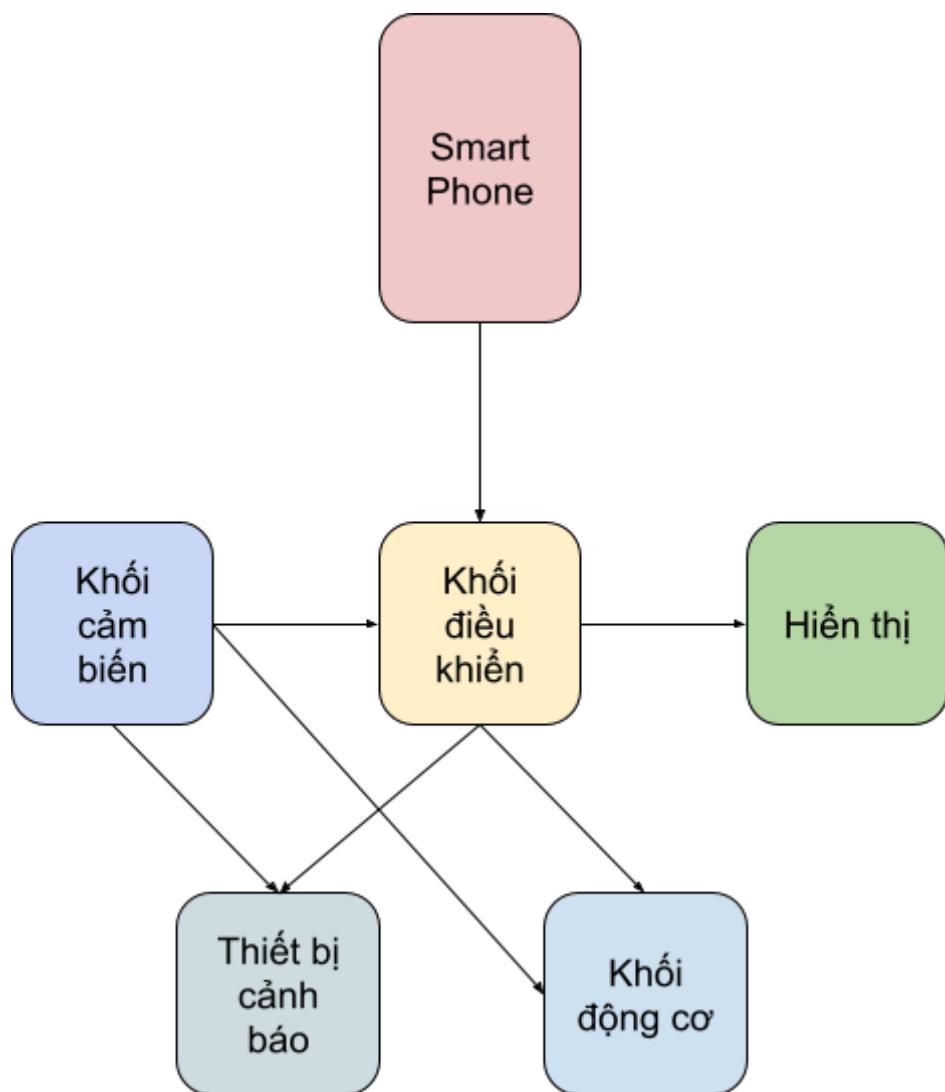
Tính khả thi trong lắp đặt ảnh hưởng trực tiếp đến việc triển khai hệ thống trong thực tế:

- **Không gian lắp đặt:** Cần xác định rõ không gian lắp đặt các cảm biến, thiết bị điều khiển và vi điều khiển. Các thiết bị này cần được đặt ở vị trí hợp lý để tối ưu hóa hiệu suất và đảm bảo tính thẩm mỹ cho ngôi nhà.
- **Đi dây và quản lý dây:** Việc đi dây từ vi điều khiển đến các cảm biến và thiết bị điều khiển cần được lên kế hoạch cẩn thận để tránh rối dây và dễ

dàng bảo trì. Sử dụng Test Board giúp việc kết nối trở nên đơn giản và gọn gàng hơn.

- **Khả năng mở rộng:** Hệ thống cần được thiết kế sao cho dễ dàng mở rộng trong tương lai, cho phép thêm các cảm biến và thiết bị điều khiển mới mà không cần thay đổi cấu trúc hệ thống quá nhiều.

3.3 Sơ đồ khói tổng quát



Hình 3.3 : Sơ đồ khói tổng quát phần cứng

Giải thích về sơ đồ khối tổng quát phần cứng:

Phần cứng của đè tài này gồm 6 khối chính:

- Khối Smartphone: gồm 1 chiếc smartphone để điều khiển
- Khối điều khiển: gồm vi điều khiển ESP32
- Khối thiết bị: gồm đèn, quạt, động cơ.
- Khối cảm biến: gồm các loại cảm biến thông dụng hiện nay (nhiệt độ, độ ẩm, khói, ánh sáng, chuyển động, chạm).
- Khối hiển thị: OLED 0.96” Display Module
- Khối thiết bị cảnh báo: gồm còi buzz và led đỏ, xanh, vàng.

3.4 Sơ đồ khối chi tiết

Ta đi vào cụ thể từng khối sau đây:

Khối Smart Phone

Đây là khối truyền lệnh từ người dùng dùng để điều khiển. Người dùng truyền lệnh điều khiển thông qua app được cài trên smartphone chạy hệ điều hành android.

Ngoài việc gửi lệnh điều khiển từ người dùng. Khối này còn có chức năng hiển thị một số thông số cho người dùng. Ví dụ thông số nhiệt độ, độ ẩm, tình trạng khói hay có phát hiện người hay không.

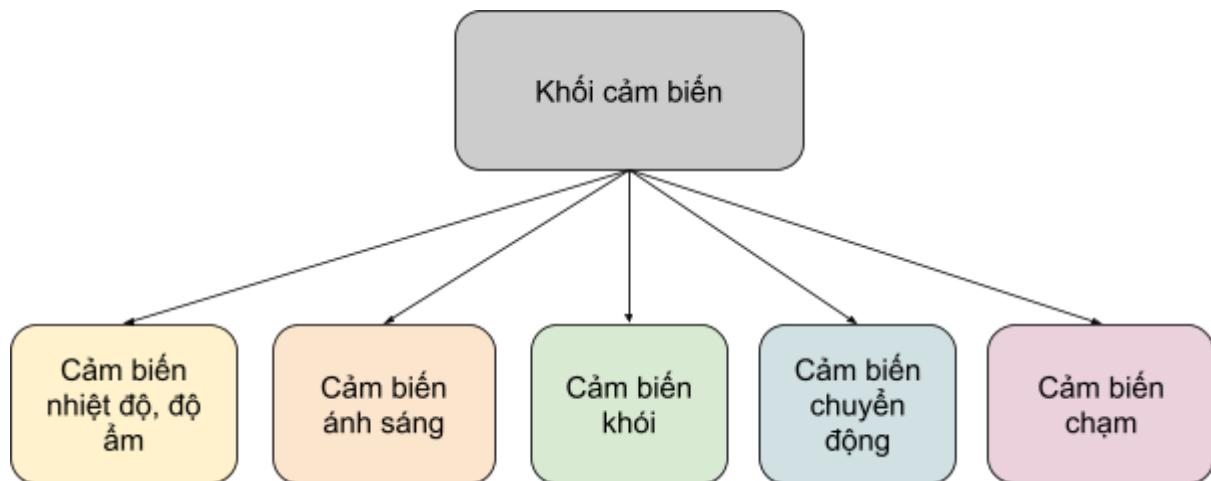
Khối điều khiển

Khối điều khiển ESP32 có chức năng như sau:

- Xử lý và điều khiển cho các cảm biến trong nhà: nhiệt độ, độ ẩm, khói, chuyển động,...
- Xử lý và điều khiển cho các thiết bị trong nhà: quạt các phòng. Đồng thời điều khiển đèn
- Xử lý dữ liệu cảm biến và các thiết bị cảnh báo (gồm còi buzz và đèn led).

Khối cảm biến

Khối cảm biến có nhiệm vụ tiếp nhận dữ liệu từ môi trường. Sau đó dữ liệu này được đưa tới vi điều khiển để xử lý. Sau khi xử lý dữ liệu thì sẽ hiển thị lên OLED Display và app để người dùng biết. Đồng thời nếu quá ngưỡng cho phép, thì sẽ cảnh báo ra loa và đèn cảnh báo.



Hình 3.4 : Sơ đồ khối cảm biến

Khối hiển thị

Khối hiển thị Oled 0.96" Display Module

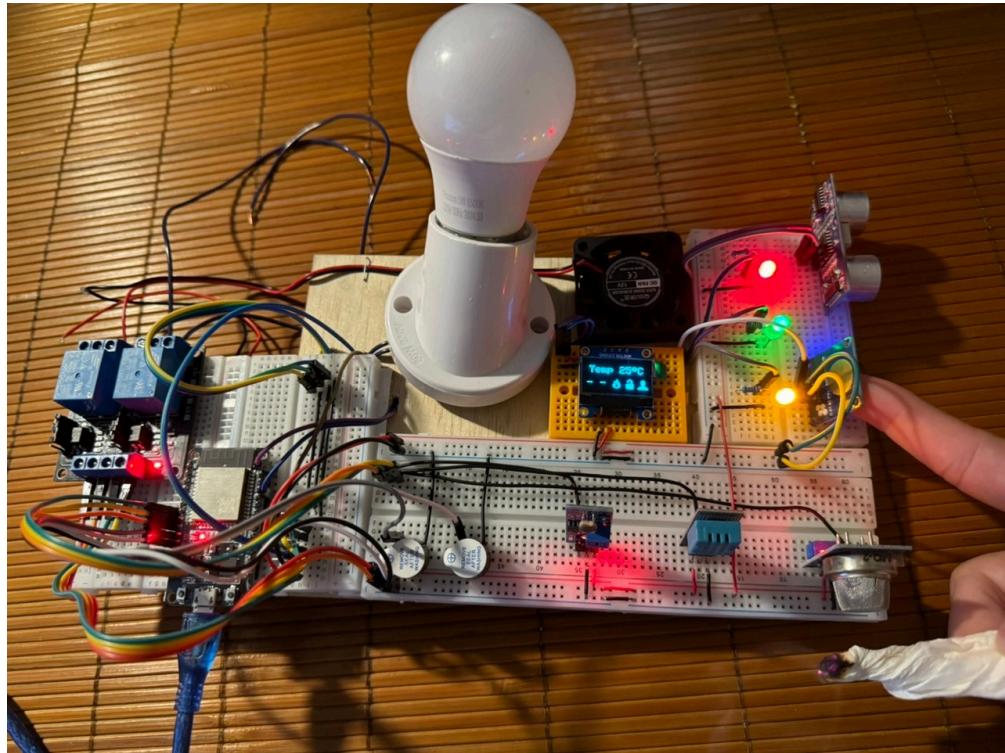
Khối cảnh báo

Khối cảnh báo gồm còi buzz và đèn led đỏ để cảnh báo.

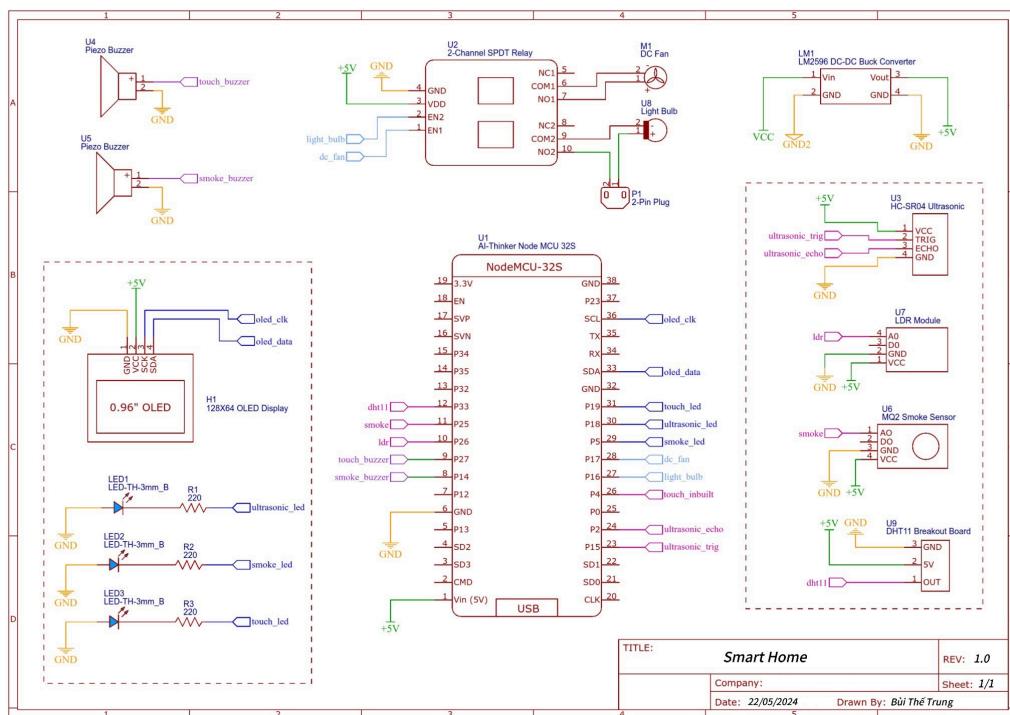
Khối thiết bị

Khối này gồm có đèn, quạt, động cơ relay.

3.5. Tông thể mô hình



Hình 3.5.a : Tông thể mô hình



Hình 3.5.b : Sơ đồ mạch thiết kế

IV. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

4.1 Yêu cầu thiết kế

Trong mục phần mềm này, chúng ta phân ra 2 phần:

- App
- Code cho vi điều khiển.

Chúng ta lần lượt đi vào từng phần:

App

- Tại trang chủ giao diện của app, sẽ có các vùng điều khiển chính và vùng để hiển thị thông số của các giá trị cảm biến.
- Tiếp theo vào các màn hình con, ta sẽ có vùng để điều khiển, hiển thị trạng thái của từng thiết bị, ví dụ thiết bị đang được bật hay tắt, hay là có đang được kích hoạt hay không
- App được thiết kế và lập trình bằng MIT App Inventor 2.

Code cho vi điều khiển

Chúng ta sẽ sử dụng trên môi trường lập trình Arduino IDE

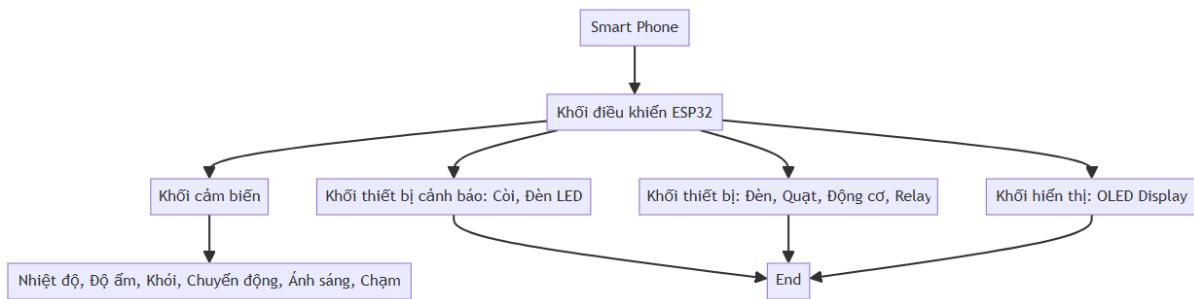
4.2 Phân tích

Về phần mềm để điều khiển:

Chúng ta sẽ có ba lựa chọn: app, web và software trên máy tính.

- Qua các phân tích ưu điểm và nhược điểm trên, dùng App (MIT App Inventor 2) là lựa chọn tốt nhất. Vì nó nhiều ưu điểm hơn so với 2 lựa chọn còn lại.
- **Ưu điểm như sau:** Tiện lợi cho người dùng, tốc độ thực thi nhanh, có thể sử dụng ở bất cứ nơi đâu. Các kỹ sư dễ dàng thiết kế và lập trình.

4.3 Sơ đồ khái tổng quát



Hình 4.3 : Lưu đồ giải thuật tổng quát

Sơ đồ khái này mô tả một hệ thống điều khiển thiết bị thông minh qua Bluetooth, có thể được áp dụng trong ngữ cảnh nhà thông minh (smart home). Hệ thống bao gồm ba thành phần chính: Ứng dụng trên điện thoại, Module Bluetooth và Vi điều khiển ESP32.

Ứng dụng trên điện thoại (App):

Đây là giao diện người dùng, nơi người dùng tương tác với hệ thống.

Ứng dụng cho phép người dùng gửi các lệnh điều khiển (bật/tắt thiết bị, điều chỉnh độ sáng, nhiệt độ,...) và nhận thông tin phản hồi từ hệ thống (trạng thái thiết bị, giá trị cảm biến,...).

Module Bluetooth:

- Module này đóng vai trò cầu nối giữa ứng dụng trên điện thoại và vi điều khiển ESP32.
- Nó nhận các lệnh điều khiển từ ứng dụng và chuyển đổi chúng thành tín hiệu mà ESP32 có thể hiểu được.
- Đồng thời, nó nhận dữ liệu từ ESP32 và gửi về ứng dụng để hiển thị cho người dùng.

Vị trí điều khiển ESP32:

- Đây là bộ não của hệ thống, chịu trách nhiệm xử lý các lệnh điều khiển từ ứng dụng và thực hiện các hành động tương ứng (bật/tắt thiết bị, điều chỉnh,...).
- ESP32 cũng nhận dữ liệu từ các cảm biến (nhiệt độ, độ ẩm, ánh sáng,...) và gửi dữ liệu này về ứng dụng thông qua module Bluetooth.

Luồng hoạt động:

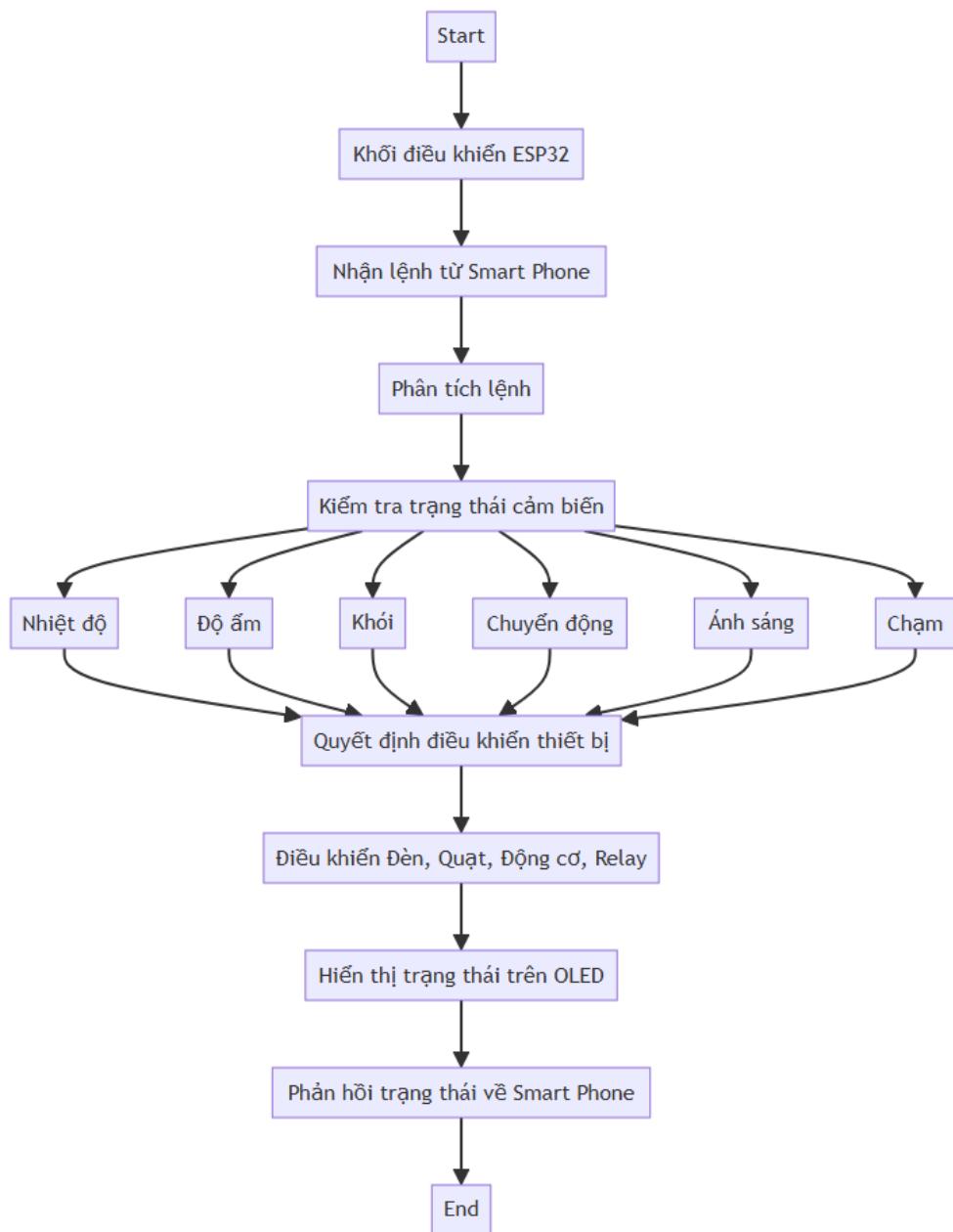
- Người dùng tương tác với ứng dụng trên điện thoại để gửi lệnh điều khiển hoặc yêu cầu thông tin.
- Ứng dụng gửi lệnh/yêu cầu đến module Bluetooth thông qua kết nối Bluetooth.
- Module Bluetooth chuyển đổi lệnh/yêu cầu thành tín hiệu và gửi đến vi điều khiển ESP32.
- ESP32 nhận tín hiệu, xử lý và thực hiện các hành động tương ứng (ví dụ: bật đèn, điều chỉnh tốc độ quạt,...).
- ESP32 cũng đọc dữ liệu từ các cảm biến và gửi dữ liệu này về module Bluetooth.
- Module Bluetooth chuyển đổi dữ liệu cảm biến thành tín hiệu và gửi về ứng dụng trên điện thoại.
- Ứng dụng hiển thị thông tin cảm biến hoặc trạng thái thiết bị cho người dùng.

Ví dụ:

1. Người dùng muốn bật đèn trong phòng khách. Họ nhấn nút "Bật đèn" trên ứng dụng.
2. Ứng dụng gửi lệnh "Bật đèn" đến module Bluetooth.
3. Module Bluetooth chuyển đổi lệnh thành tín hiệu và gửi đến ESP32.
4. ESP32 nhận tín hiệu và bật đèn trong phòng khách.

5. ESP32 gửi tín hiệu xác nhận "Đèn đã bật" về module Bluetooth.
6. Module Bluetooth gửi tín hiệu xác nhận về ứng dụng.
7. Ứng dụng hiển thị thông báo "Đèn đã bật" cho người dùng.

4.4 Sơ đồ khái chi tiết



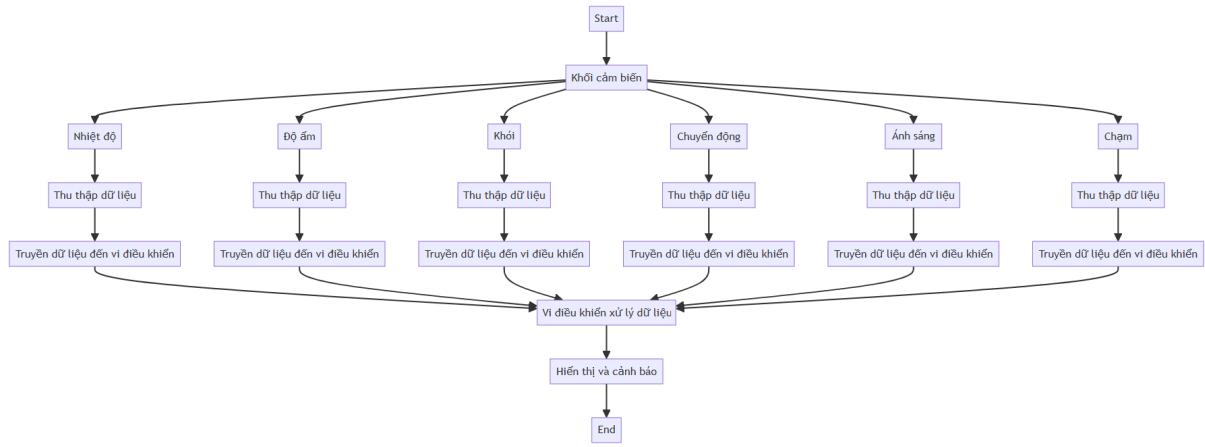
Hình 4.4.a : Lưu đồ giải thuật phần điều khiển thiết bị

Các bước chính trong quy trình:

- **Khởi động (Start):** Hệ thống bắt đầu hoạt động.
- **Khởi điều khiển ESP32:** ESP32 là bộ não của hệ thống, chịu trách nhiệm xử lý thông tin và đưa ra quyết định điều khiển.
- **Nhận lệnh từ Smart Phone:** Người dùng gửi lệnh điều khiển các thiết bị từ điện thoại thông minh.
- **Phân tích lệnh:** ESP32 phân tích lệnh nhận được để hiểu người dùng muốn thực hiện hành động gì (ví dụ: bật đèn, tắt quạt).

Kiểm tra trạng thái cảm biến:

- **Nhiệt độ, độ ẩm, khói, chuyển động, ánh sáng, chạm:** ESP32 thu thập dữ liệu từ các cảm biến khác nhau để nắm bắt tình trạng môi trường và các sự kiện xảy ra.
- **Quyết định điều khiển thiết bị:** Dựa trên lệnh từ người dùng và thông tin từ cảm biến, ESP32 đưa ra quyết định điều khiển các thiết bị.
- **Điều khiển Đèn, Quạt, Động cơ, Relay:** ESP32 gửi tín hiệu điều khiển đến các thiết bị như đèn, quạt, động cơ và relay để thực hiện các hành động tương ứng (bật/tắt, tăng/giảm tốc độ).
- **Hiển thị trạng thái trên OLED:** Trạng thái hiện tại của các thiết bị và thông tin khác được hiển thị trên màn hình OLED để người dùng theo dõi.
- **Phản hồi trạng thái về Smart Phone:** ESP32 gửi thông tin phản hồi về điện thoại của người dùng để thông báo kết quả của các lệnh điều khiển và cập nhật trạng thái hệ thống.
- **Kết thúc (End):** Quy trình kết thúc.



Hình 4.4.b : Lưu đồ giải thuật cảm biến

Các bước chính trong quy trình:

Khởi động (Start): Hệ thống bắt đầu hoạt động.

- **Khởi tạo cảm biến:** Các cảm biến khác nhau được khởi tạo và chuẩn bị sẵn sàng để thu thập dữ liệu. Các loại cảm biến bao gồm:
- **Nhiệt độ:** Đo nhiệt độ môi trường.
- **Độ ẩm:** Đo độ ẩm không khí.
- **Khói:** Phát hiện sự hiện diện của khói.
- **Chuyển động:** Phát hiện sự chuyển động trong khu vực.
- **Ánh sáng:** Đo cường độ ánh sáng.
- **Chạm:** Phát hiện sự tiếp xúc hoặc chạm vào bề mặt cảm biến.

Thu thập dữ liệu: Mỗi cảm biến thực hiện việc thu thập dữ liệu liên quan đến môi trường xung quanh.

Truyền dữ liệu đến vi điều khiển: Dữ liệu thu thập được từ các cảm biến được gửi đến vi điều khiển trung tâm để xử lý.

Vì điều khiển xử lý dữ liệu: Vì điều khiển nhận dữ liệu từ các cảm biến, phân tích và đưa ra quyết định dựa trên các thông số đã được thiết lập trước.

Hiển thị và cảnh báo: Dựa trên kết quả xử lý dữ liệu, vi điều khiển có thể hiển thị thông tin trên màn hình, gửi cảnh báo đến người dùng hoặc thực hiện các

hành động khác (ví dụ: bật đèn khi trời tối, kích hoạt báo động khi phát hiện khói).

Kết thúc (End): Quy trình kết thúc.

4.5 Giải thích code

Đoạn code này đảm bảo rằng:

- Bluetooth và Bluedroid đã được bật trong cấu hình của dự án.
- Serial Port Profile (SPP) đã được bật nếu ứng dụng cần sử dụng Bluetooth nối tiếp.
- Ứng dụng sẽ chạy trên lõi CPU được chỉ định bởi biến app_cpu, tùy thuộc vào cấu hình FreeRTOS (một lõi hoặc nhiều lõi).

```
9
10 #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
11 #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
12 #endif
13
14 #if !defined(CONFIG_BT_SPP_ENABLED)
15 #error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
16 #endif
17
```

Hình 4.5.a : Đảm bảo Bluetooth và Bluedroid đã được bật trong cấu hình

Đoạn mã dưới đây cung cấp một cấu hình ban đầu để giao tiếp với các thành phần phần cứng trong dự án. Các chân GPIO và thông số được định nghĩa ở đây sẽ được sử dụng trong mã nguồn chính của dự án để đọc dữ liệu từ cảm biến, điều khiển rơ le, đèn LED, còi báo động và hiển thị thông tin trên màn hình OLED.

```

24
25 /* Sensor pins */
26 #define DHTPIN 33 // DHT11 temperature sensor
27 #define DHTTYPE DHT11
28 #define lightSensor 26 // LDR sensor
29 #define smokeSensor 25 // MQ2 smoke and gas sensor
30 #define touchSensor 4 // Touch sensor (GPIO 4)
31 #define echo 2 // Ultrasonic sensor echo pin
32 #define trigger 15 // Ultrasonic sensor trigger pin
33
34 /* Relay pins */
35 #define fanRelay 17 // Relay for fan
36 #define lightRelay 16 // Relay for light
37
38 /* Buzzer pins */
39 #define smokeBuzzer 14 // Buzzer for alerting smoke or gas
40 #define touchBuzzer 27 // Buzzer for alerting touch
41
42 /* Led pins */
43 #define smokeLed 5 // Led for alerting smoke
44 #define touchLed 19 // Led for alerting touch
45 #define ultrasonicLed 18 // Led for alerting when someone in range
46
47 /* Setting OLED display parameters */
48 #define SCREEN_WIDTH 128 // OLED display width, in pixels
49 #define SCREEN_HEIGHT 64 // OLED display height, in pixels
50 #define SCREEN_ADDRESS 0x3C // i2c address for OLED display
51 #define OLED_RESET 4
52

```

Hình 4.5.b : Câu lệnh, định nghĩa các chân

Đoạn mã dưới đây thực hiện việc đọc giá trị nhiệt độ từ cảm biến DHT11, sau đó gửi giá trị này qua Bluetooth và hiển thị trên màn hình Serial. Quá trình này được lặp lại liên tục với chu kỳ 2 giây. Nếu đọc cảm biến bị lỗi, chương trình sẽ in thông báo lỗi và dừng lại.

```

149
150 /* Task for temperature sensing using DHT11 */
151 void tempRead(void *parameter) {
152     int t = 0;
153
154     while (true) {
155         t = dht.readTemperature();
156
157         if (isnan(t)) {
158             Serial.println(F("Failed to read from DHT sensor!"));
159             return;
160         }
161
162         /* Send temperature values via bluetooth */
163         SerialBT.print("#");
164         SerialBT.print(t);
165         SerialBT.print("?");
166
167         /* Print temperature and humidity values on serial monitor */
168         Serial.print("Temperature: ");
169         Serial.print(t);
170         Serial.println(" °C");
171
172         xQueueSend (tempReading, (void*)&t, 10);
173         vTaskDelay(2000 / portTICK_PERIOD_MS);
174     }
175 }
```

Hình 4.5.c : Đọc nhiệt độ, độ ẩm

Chương trình này tạo ra hệ thống điều khiển quạt và đèn tự động dựa trên các ngưỡng nhiệt độ và cường độ ánh sáng được cài đặt sẵn. Nó sử dụng hàng đợi để giao tiếp giữa các tác vụ và có thể gửi thông báo trạng thái qua Bluetooth.

```

178 void autoFan(void *parameter) {
179     int tempValue;
180
181     while (true) {
182         xQueueReceive(tempReading, (void *)&tempValue, portMAX_DELAY);
183
184         if (tempValue >= 32) {
185             SerialBT.print ("Fan on?");
186             digitalWrite(fanRelay,LOW);
187             fanStatus = true;
188         }
189         else if (tempValue < 30) {
190             SerialBT.print ("Fan off?");
191             digitalWrite(fanRelay,HIGH);
192             fanStatus = false;
193         }
194         vTaskDelay(200 / portTICK_PERIOD_MS);
195     }
196 }
197 /*
198 *
199 * -----
200 * LDR based bulb control
201 * -----
202 */
203
204 /* Task for light intensity sensing using LDR */
205 void lightRead(void *parameter) {
206     int lightValue;
207
208     while (true) {
209         lightValue = analogRead(lightSensor);
210         Serial.print("Light intensity: ");
211         Serial.println(lightValue);
212
213         xQueueSend (lightReading, (void*)&lightValue, 10);
214         vTaskDelay(2000 / portTICK_PERIOD_MS);
215     }
216 }
217
218 /* Task for bulb control in auto mode */
219 void autoLight(void *parameter) {
220     int lightValue;
221
222     while (true) {
223         xQueueReceive(lightReading, (void *)&lightValue, portMAX_DELAY);
224
225         if (lightValue >= 2200) {
226             SerialBT.print("Bulb on?");
227             digitalWrite(lightRelay,LOW);
228             lightStatus = true;
229         }
230         else if (lightValue < 2200) {
231             SerialBT.print("Bulb off?");
232             digitalWrite(lightRelay,HIGH);
233             lightStatus = false;
234         }
235         vTaskDelay(200 / portTICK_PERIOD_MS);
236     }
237 }
238

```

Hình 4.5.d : Chương trình điều khiển quạt, đèn tự động

Khi phát hiện khói/gas hoặc có sự kiện chạm, chương trình sẽ gửi thông báo qua Serial Bluetooth, bật đèn LED và còi báo động tương ứng. Các giá trị đọc được từ cảm biến cũng được in ra Serial để theo dõi và gỡ lỗi.

Nếu khoảng cách nhỏ hơn hoặc bằng 20 cm, nó sẽ gửi cảnh báo "Ultrasonic active?" qua Bluetooth và bật đèn LED. Nếu khoảng cách lớn hơn 20 cm, nó sẽ gửi cảnh báo "Ultrasonic inactive?" và tắt đèn LED.

```
245  /* Task for detecting smoke or gas using MQ2 sensor */
246  void smokeDetect(void *parameter) {
247      int smokeValue;
248
249      while (true) {
250          smokeValue = analogRead(smokeSensor);
251          Serial.print("Smoke: ");
252          Serial.println(smokeValue);
253
254          if (smokeValue >= 3200) {
255              SerialBT.print("Smoke active?");
256              digitalWrite(smokeLed, HIGH);
257              digitalWrite(smokeBuzzer, HIGH);
258              smokeStatus = true;
259          }
260          else if (smokeValue < 3200) {
261              SerialBT.print("Smoke inactive?");
262              digitalWrite(smokeLed, LOW);
263              digitalWrite(smokeBuzzer, LOW);
264              smokeStatus = false;
265          }
266
267          vTaskDelay(1000 / portTICK_PERIOD_MS);
268      }
269  }
```

Hình 4.5.e : Chương trình cảnh báo khói, bật đèn và còi cảnh báo

```

271  /* Task for detecting touch using inbuilt touch sensor */
272  void touchDetect(void *parameter) {
273      int touchValue;
274
275      while (true) {
276          touchValue = digitalRead(touchSensor); // Read the touch sensor value
277
278          if (touchValue == 1) { // If the touch sensor is touched (value is 1)
279              SerialBT.print("Touch active?");
280              Serial.println(touchValue);
281              digitalWrite(touchLed, HIGH); // Turn on the touch LED
282              digitalWrite(touchBuzzer, HIGH); // Turn on the touch Buzzer
283              touchStatus = true; // Set the touch status to true
284          }
285          else if (touchValue == 0) { // If the touch sensor is not touched (value is 0)
286              SerialBT.print("Touch inactive?");
287              Serial.println(touchValue);
288              digitalWrite(touchLed, LOW); // Turn off the touch LED
289              digitalWrite(touchBuzzer, LOW); // Turn off the touch Buzzer
290              touchStatus = false; // Set the touch status to false
291          }
292          vTaskDelay(1000 / portTICK_PERIOD_MS);
293      }
294  }
295

```

Hình 4.5.g : Chương trình cảnh báo chạm, bật đèn và còi cảnh báo

```

295
296  /* Task for finding distance using Ultrasonic sensor */
297  void ultrasonicDetect() {
298      int distance;
299      int duration;
300
301      digitalWrite(trigger, LOW);
302      delayMicroseconds(2);
303      digitalWrite(trigger, HIGH);
304      delayMicroseconds(10);
305      digitalWrite(trigger, LOW);
306
307      duration = pulseIn(echo, HIGH);
308      distance = (duration / 2) * 0.0343;
309
310      Serial.print("Distance: ");
311      Serial.println(distance);
312
313      if (distance > 20) {
314          SerialBT.print("Ultrasonic inactive?");
315          digitalWrite(ultrasonicled, LOW);
316          ultrasonicStatus = false;
317      }
318      else if (distance <= 20) {
319          SerialBT.print("Ultrasonic active?");
320          digitalWrite(ultrasonicled, HIGH);
321          ultrasonicStatus = true;
322      }
323  }

```

Hình 4.5.h : Chương trình cảnh báo người, bật đèn

Đoạn code này xử lý các chế độ thủ công, tự động, tắt tất cả và điều khiển báo động chạm

```
331  /* Task for controlling relays and alarms using app */
332  void switchControl(void *parameter) {
333      char input;
334
335      while (true) {
336          if (SerialBT.available() > 0) {
337              input = SerialBT.read();
338              switch (input) {
339                  /* Select manual mode */
340                  case 'M': {
341                      vTaskSuspend (autoFan_handle);
342                      vTaskSuspend (autoLight_handle);
343                      break;
344                  }
345                  /* Switch on fan */
346                  case 'F': {
347                      SerialBT.print("Fan on?");
348                      digitalWrite(fanRelay, LOW);
349                      fanStatus = true;
350                      break;
351                  }
352                  /* Switch off fan */
353                  case 'V': {
354                      SerialBT.print("Fan off?");
355                      digitalWrite(fanRelay, HIGH);
356                      fanStatus = false;
357                      break;
358                  }
359                  /* Switch on light */
360                  case 'L': {
361                      SerialBT.print("Bulb on?");
362                      digitalWrite(lightRelay, LOW);
363                      lightStatus = true;
364                      break;
365                  }
366                  /* Switch off light */
367                  case 'Z': {
368                      SerialBT.print("Bulb off?");
369                      digitalWrite(lightRelay, HIGH);
370                      lightStatus = false;
371                      break;
372                  }
373                  /* Select automatic mode */
374                  case 'A': {
375                      vTaskResume(autoFan_handle);
376                      vTaskResume(autoLight_handle);
377                      break;
378                  }
379                  /* Select off mode */
380                  case 'O': {
381                      vTaskSuspend(autoFan_handle);
382                      vTaskSuspend(autoLight_handle);
383                      SerialBT.print("Fan off?");
384                      SerialBT.print("Bulb off?");
385                      digitalWrite(fanRelay, HIGH);
386                      digitalWrite(lightRelay, HIGH);
387                      fanStatus = false;
388                      lightStatus = false;
389                      break;
390                  }
391                  /* Turn off touch alarm */
392                  case 'T': {
393                      SerialBT.print("Touch inactive?");
394                      digitalWrite(touchBuzzer, LOW);
395                      digitalWrite(touchLed, LOW);
396                      touchStatus = false;
397                      break;
398                  }
399              }
400          }
401      }
402  }
```

Hình 4.5.i : Chương trình thay đổi các chế độ tự động, thủ công, tắt tất cả

V. KẾT QUẢ THỰC HIỆN

5.1 Kết quả phần mềm

Trong quá trình phát triển và thử nghiệm, phần mềm điều khiển nhà thông minh đã đạt được những kết quả sau:

- **Giao diện người dùng:** Giao diện ứng dụng được thiết kế trực quan và dễ sử dụng, cho phép người dùng dễ dàng điều khiển các thiết bị trong nhà từ điện thoại di động. Các chức năng chính như bật/tắt đèn, điều chỉnh nhiệt độ và theo dõi trạng thái của các cảm biến đều hoạt động mượt mà.
- **Kết nối Bluetooth:** Hệ thống kết nối Bluetooth ổn định, cho phép giao tiếp nhanh chóng giữa điện thoại và vi điều khiển ESP32. Các lệnh điều khiển được truyền đi và phản hồi ngay lập tức, đảm bảo sự tương tác liên tục và chính xác.
- **Tích hợp cảm biến:** Phần mềm đã tích hợp thành công các dữ liệu từ cảm biến nhiệt độ, độ ẩm, khói và chuyển động. Các dữ liệu này được hiển thị rõ ràng trên ứng dụng, giúp người dùng dễ dàng theo dõi và quản lý môi trường trong nhà.
- **Chức năng cảnh báo:** Khi phát hiện các tình huống bất thường như khói, cảm biến chạm hoặc chuyển động, hệ thống sẽ gửi cảnh báo đến người dùng ngay lập tức. Chức năng này giúp tăng cường an ninh và an toàn cho ngôi nhà.

5.2 Kết quả phần cứng

Phần cứng của hệ thống nhà thông minh cũng đã được triển khai và thử nghiệm với những kết quả như sau:

- **Vi điều khiển ESP32:** Vi điều khiển ESP32 hoạt động ổn định và hiệu quả, thực hiện chính xác các lệnh điều khiển từ ứng dụng. Các chức năng điều khiển thiết bị như đèn, quạt và động cơ đều đáp ứng nhanh chóng.
- **Cảm biến:** Các cảm biến nhiệt độ, độ ẩm, khói và chuyển động đều hoạt động đúng chức năng, cung cấp dữ liệu chính xác và kịp thời cho hệ thống. Cảm biến khói và cảm biến chuyển động đã chứng minh khả năng phát hiện và cảnh báo hiệu quả.
- **Thiết bị điều khiển:** Các thiết bị điều khiển như relay, đèn LED và quạt đã được kết nối và điều khiển thành công bởi hệ thống. Các relay hoạt động chính xác, bật/tắt các thiết bị điện một cách ổn định.
- **Màn hình OLED:** Màn hình OLED hiển thị đầy đủ và rõ ràng các thông tin quan trọng từ các cảm biến, giúp người dùng dễ dàng theo dõi trạng thái hệ thống trực tiếp trên thiết bị.

5.3 Đánh giá hiệu suất

- **Hiệu suất hệ thống:** Hệ thống đã được thử nghiệm trong nhiều điều kiện khác nhau và cho thấy hiệu suất ổn định, độ tin cậy cao. Các thiết bị và cảm biến hoạt động một cách đồng bộ, đáp ứng tốt các yêu cầu của người dùng.
- **Tiết kiệm năng lượng:** Hệ thống tự động điều chỉnh các thiết bị điện dựa trên cảm biến nhiệt độ và ánh sáng, giúp tối ưu hóa việc sử dụng năng lượng và tiết kiệm chi phí điện năng.
- **An ninh và an toàn:** Hệ thống tích hợp các cảm biến an ninh như cảm biến khói và cảm biến chuyển động, cung cấp các cảnh báo kịp thời, đảm bảo an toàn cho ngôi nhà.

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Sau quá trình nghiên cứu, thiết kế và triển khai hệ thống nhà thông minh sử dụng vi điều khiển ESP32 và nền tảng MIT App Inventor 2, chúng em đã đạt được những kết quả quan trọng như sau:

- **Hoàn thiện hệ thống:** Hệ thống nhà thông minh đã được triển khai hoàn chỉnh, bao gồm các thành phần phần cứng và phần mềm, hoạt động ổn định và đáng tin cậy.
- **Chức năng điều khiển:** Hệ thống cho phép người dùng điều khiển các thiết bị trong nhà như đèn, quạt và các thiết bị khác thông qua ứng dụng di động. Các chức năng tự động hóa dựa trên cảm biến nhiệt độ và ánh sáng hoạt động hiệu quả, giúp tối ưu hóa việc sử dụng năng lượng.
- **Giám sát an ninh và an toàn:** Hệ thống tích hợp các cảm biến an ninh như cảm biến khói, cảm biến chạm và cảm biến siêu âm, cung cấp cảnh báo kịp thời khi có sự cố, giúp bảo vệ an toàn cho ngôi nhà.
- **Ứng dụng công nghệ 4.0:** Việc triển khai hệ thống này không chỉ mang lại lợi ích trực tiếp cho người sử dụng mà còn góp phần thúc đẩy ứng dụng công nghệ 4.0 vào đời sống hàng ngày, khuyến khích sự phát triển của các sản phẩm và giải pháp công nghệ mới.

6.2 Hướng phát triển cho đài tài

Mặc dù hệ thống nhà thông minh đã đạt được những kết quả quan trọng, chúng em nhận thấy vẫn còn nhiều cơ hội để cải tiến và phát triển hệ thống trong tương lai. Một số hướng phát triển chính bao gồm:

- **Mở rộng tính năng:** Nâng cấp hệ thống để hỗ trợ thêm nhiều loại thiết bị và cảm biến khác, mở rộng khả năng tự động hóa và giám sát. Ví dụ, tích hợp thêm cảm biến chất lượng không khí, cảm biến ánh sáng tự nhiên để điều chỉnh ánh sáng nhân tạo phù hợp.
- **Tối ưu hóa phần mềm:** Cải thiện hiệu suất của ứng dụng di động, giảm độ trễ khi điều khiển thiết bị và tăng cường tính năng bảo mật. Đồng thời, phát triển ứng dụng trên nhiều nền tảng khác nhau như iOS để mở rộng đối tượng người dùng.
- **Nâng cao an ninh:** Tích hợp các công nghệ bảo mật tiên tiến như mã hóa dữ liệu, xác thực hai yếu tố để bảo vệ hệ thống khỏi các mối đe dọa an ninh mạng.
- **Tích hợp với các hệ thống khác:** Nghiên cứu và phát triển khả năng tích hợp hệ thống nhà thông minh với các nền tảng và dịch vụ khác như hệ thống quản lý năng lượng, hệ thống điều khiển âm thanh và hình ảnh trong nhà.
- **Phát triển trí tuệ nhân tạo (AI):** Áp dụng các thuật toán AI để học hỏi thói quen của người dùng và tự động điều chỉnh các thiết bị trong nhà một cách thông minh hơn. Ví dụ, hệ thống có thể học và dự đoán thời gian bật/tắt đèn, điều chỉnh nhiệt độ phù hợp với từng thời điểm trong ngày.
- **Tăng cường trải nghiệm người dùng:** Tạo ra giao diện người dùng thân thiện hơn, cung cấp các hướng dẫn và hỗ trợ kịp thời thông qua ứng dụng. Đồng thời, phát triển các tính năng tương tác bằng giọng nói để người dùng có thể điều khiển hệ thống một cách tiện lợi hơn.

Với những hướng phát triển này, chúng em hy vọng hệ thống nhà thông minh sẽ ngày càng hoàn thiện và mang lại nhiều tiện ích hơn cho người sử dụng, góp phần nâng cao chất lượng cuộc sống và thúc đẩy sự phát triển của công nghệ thông minh trong tương lai.

VII. TÀI LIỆU THAM KHẢO

1. Espressif Systems. "ESP32 Technical Reference Manual."
<https://docs.espressif.com>.
2. MIT. "MIT App Inventor 2 Documentation." Massachusetts Institute of Technology. <https://appinventor.mit.edu>.
3. Các nhà sản xuất thiết bị thông minh như Google, Samsung, Apple, Schneider, Smartg4, Gamma, Arteor, My Home, WattStopper, Mhouse, Hager, Crestron.
4. BKAV Corporation. "Nhà Thông Minh BKAV." BKAV, 2024.
5. Lumi Vietnam. "Giải pháp Nhà Thông Minh Lumi." Lumi, 2024.
6. ASIC. "Nhà Thông Minh ASIC." ASIC, 2024.