TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA ĐIỆN TỬ

----------o0o----------

BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH ĐIỆN TỬ VIỄN THÔNG

THIẾT KẾ HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ , ĐỘ ẨM VÀ ĐIỀU KHIỂN THIẾT BỊ ĐIỆN QUA INTERNET DÙNG ESP32

Cán bộ hướng dẫn : Th.S Dương Thị Hằng

Nhóm thực hiện:

1. **Nguyễn Đại Hồng** **MSV: 2020601376**
2. **Nguyễn Thị Thanh Huyền** **MSV: 2020601561**
3. **Nguyễn Thị Thu Hiền** **MSV: 2020601742**
4. **Tiêu Tiến Đức** **MSV: 2020601913**
5. **Hoàng Hồng Trà** **MSV: 2019600391**

Hà Nội, 2023

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Hà Nội, Ngày…Tháng…Năm

Người nhận xét

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Tiếng anh	Tiếng việt (tạm dịch)
1	IoT	Internet of Things	Kết nối vạn vật
2	IC	Integrated Circuit	Vi mạch
3	MCU	Microcontroller Unit	Bộ vi điều khiển
4	CPU	Central Processing Unit	Bộ xử lý trung tâm
5	JSON	JavaScript Object Notation	
6	WiFi	Wireless Fidelity	

MỞ ĐẦU

1. Lý do chọn đề tài

Chúng ta hiện nay đang sống trong một xã hội phát triển về mọi mặt. Vì vậy, chúng ta muốn thích nghi thì phải không ngừng học tập và nghiên cứu. Sinh viên cũng vậy phải luôn học tập và hoàn thiện mình hơn. Khi kinh tế phát triển, nhu cầu cuộc sống của con người không ngừng được nâng cấp, Nhận thấy được điều đó nên ngành kỹ thuật - công nghệ điện tử viễn thông cũng không ngừng phát triển để có thể phục vụ nhu cầu của con người. Có thể nói các thiết bị đo nhiệt độ, độ ẩm ngày càng phổ biến trong đời sống chúng ta. Ngày nay chúng ta bắt gặp trên thị trường rất nhiều các thiết bị đo nhiệt độ, độ ẩm. Từ những nhu cầu giải trí đó đồng thời góp phần làm sáng tỏ ứng dụng của ngành kỹ thuật công nghệ điện tử viễn thông em đưa ra đề tài " Thiết kế hệ thống giám sát nhiệt độ , độ ẩm và điều khiển thiết bị điện qua internet dùng ESP32".

2. Mục tiêu đề tài

Tiếp nhận tín hiện từ các cảm biến và điều khiển các thiết bị.

Có chức năng giám sát và điều khiển từ xa qua internet, sử dụng điện thoại hoặc máy tính.

Có thể thi công đồ án trên một ngôi nhà thực tế hoặc mô hình.

3. Nội dung nghiên cứu

Việc thực hiện thiết kế mạch ‘‘Giám sát nhiệt độ độ ẩm và điều khiển thiết bị điện qua internet dùng ESP32’’ sẽ cần phải thực hiện các nội dung như sau :• Nội dung 1: Nghiên cứu tài liệu về KIT ESP32, giao tiếp không dây và mạng Internet.

- Nội dung 2: Nghiên cứu các mô hình điều khiển.
- Nội dung 3: Thiết kế và tính toán thiết kế mạch phần cứng cho thiết bị.
- Nội dung 4: Thi công phần cứng, thử nghiệm và hiệu chỉnh phần cứng.
- Nội dung 5: Thử nghiệm và điều chỉnh hệ thống cũng như chương trình để hệ thống được tối ưu. Đánh giá các thông số của mô hình so với thực tế.

4. Giới hạn đề tài

- Kích thước mô hình.
- Sử dụng KIT ESP32
- Tập trung vào thiết bị điều khiển trung tâm.
- Sử dụng các nền tảng đã có sẵn và các thư viện mở để phát triển sản phẩm.

5. Phạm vi ứng dụng

Đề tài là mô hình thu nhỏ, tuy nhiên có thể được ứng dụng rộng rãi ở các môi trường khác nhau như nhà ở, nhà xưởng, nhà kính….Trong sản xuất cũng như sinh hoạ

6. Bố cục quyển báo cáo

Nội dung báo cáo:

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

CHƯƠNG 2: THIẾT KẾ HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ, ĐỘ ẨM VÀ ĐIỀU KHIỂN THIẾT BỊ ĐIỆN QUA INTERNET DÙNG ESP32

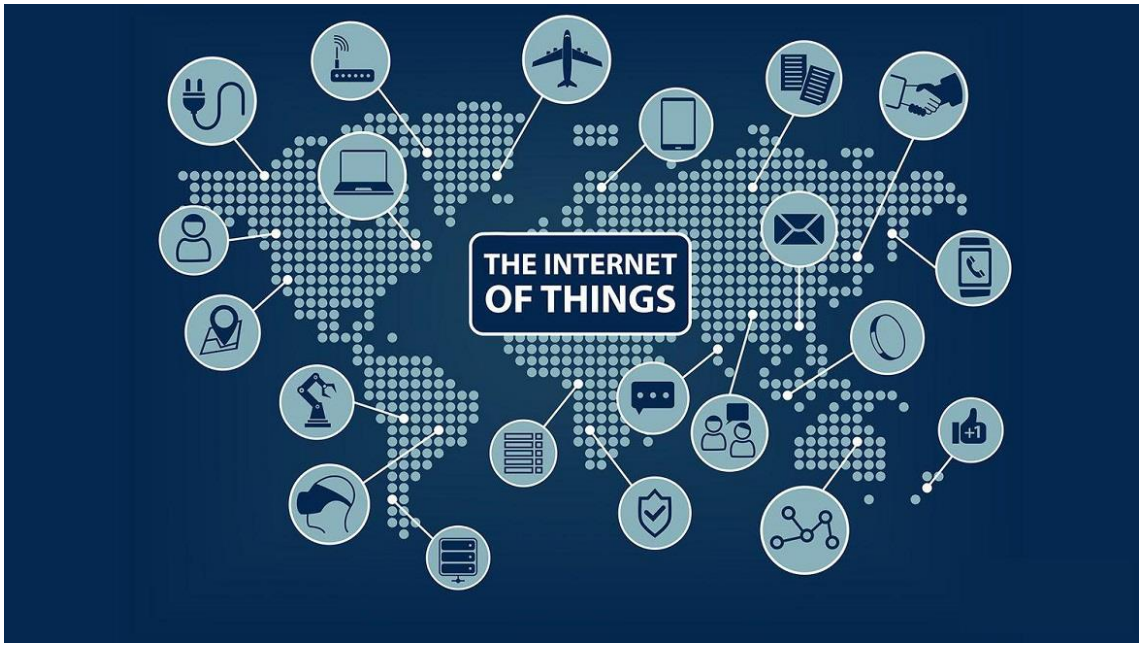
CHƯƠNG 3: KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1. Tổng quan về Iot

1.1.1. *Giới thiệu về Internet of Things (IoT)*

IoT là viết tắt của "Internet of Things", hay còn gọi là "mạng Internet của các vật thể". Đây là một khái niệm trong lĩnh vực công nghiệp và công nghệ thông tin mô tả mạng lưới các thiết bị vật lý (đối tượng, máy móc, cảm biến, thiết bị điện tử) được kết nối và có khả năng truyền thông thông qua Internet. Mục tiêu của IoT là tạo ra hệ thống thông tin toàn cầu mà mọi vật thể có thể tương tác với nhau thông qua Internet[2].



Hình 1. 1 Tổng quan về IoT

Các thành phần chính của một hệ thống IoT:

- Thiết bị IoT (IoT Devices): bao gồm thiết bị cảm biến và thiết bị thực thi.
- Mạng kết nối (Connectivity): bao gồm các giao thức truyền thông và mạng.
- Trung tâm xử lý (Processing Center): điều khiển, quản lí thiết bị và đưa ra các giải pháp bảo mật.
- Dịch vụ mạng (Network Services): quản lý địa chỉ IP, quản lý băng thông.
- Phần mềm ứng dụng: bao gồm ứng dụng và giao diện người dùng, phân tích và xử lý dữ liệu.
- Lưu trữ (Storage): Lưu trữ dữ liệu từ các thiết bị IoT để sử dụng sau này.
- Đám mây (Cloud): Cung cấp không gian lưu trữ, tính toán, và các dịch vụ khác thông qua đám mây.

1.1.2. *Lịch sử hình thành*

Khái niệm về một mạng lưới thiết bị được kết nối với nhau đã được thảo luận vào đầu năm 1982, với một máy bán hàng tự động Coke được thực hiện ở Đại học Carnegiie Mellon trở thành thiết bị kết nối Interner đầu tiên trêm thế giới. Thuật ngữ "Internet of Things" được sử dụng lần đầu tiên bởi Kevin Ashton vào năm 1999. Sau đó IoT trải qua nhiều giai đoạn và có bước phát triển nhảy vọt cho đến nay[2].

1.1.3. *Ứng dụng của IoT*

Nhà thông minh (Smart Home): những thiết bị thông minh trong gia đình được sử dụng cho các mục đích như sau:

- Tự động tắt các thiết bị khi gia chủ không có nhu cầu sử dụng.
- Quản lý cũng như bảo trì các bất động sản cho thuê.
- Tìm đồ thất lạc nhanh chóng như chìa khóa hoặc ví.
- Tự động hóa các công việc thường ngày cho gia chủ như hút bụi, pha cà phê,…

Hình 1. 2 Nhà thông minh

Xe hơi thông minh: Hiện nay, với sự phát triển mạnh mẽ của công nghệ các phương tiện di chuyển như ô tô có thể kết nối Internet bằng nhiều cách khác nhau. Cụ thể là có thẻ thông qua camera hành trình, hệ thống tin học giải trí, qua các cổng kết nối của phương tiện[3].

Hình 1. 3 Xe hơi thông minh

Quá trình này diễn ra thông qua việc thu thập dữ liệu từ chân ga, đồng hồ tốc độ, phanh, đồ hồ đo quãng đường, bình xăng, bánh xe,… để giám sát hiệu suất của người lái cũng như tình trạng phương tiện.

Chăm sóc sức khỏe: Miếng dán theo dõi sức khỏe cho bệnh nhân, bạn không cần đến bác sĩ, những thông số về nhịp tim, huyết áp, đều được thu thập từ xa được phân tích sau đó chuẩn đoán đưa ra tình trạng sức khỏe hiện tại của bệnh nhân và có thể dự đoán nguy cơ mắc bệnh nhằm có biện pháp phòng ngừa kịp thời.

Hình 1. 4 Chăm sóc sức khỏe ứng dụng IoT

Thành phố thông minh (Smart City): Có thể xxem đây là tập hợp tất cả các ứng dụng của IoT vào một hệ thống lớn. Một giải pháp đã và đang được nhiều quốc gia trên thế giới áp dụng ở các thành phố lớn nhằm giải quyết những vấn đề cấp bách như kẹt xe, gia tăng dân số, ô nhiễm môi trường, ngập lụt,... Mọi thứ trong thành phố thông minh này được kết nối, dữ liệu sẽ được giám sát bởi một loại các máy tính mà không cần bất kỳ sự tương tác nào cảu cong người[3].

Hình 1. 5 Thành phố thông minh

1.2. Công nghệ wifi

Wifi là một mạng không dây thay thế cho mạng có dây thông thường, thường được sử dụng để kết nối các thiết bị ở chế độ không dây bằng việc sử dụng công nghệ sóng vô tuyến. Dữ liệu được truyền qua sóng vô tuyến cho phép các thiết bị truyền nhận dữ liệu ở các tần số 2.4 GHz hoặc 5GHz.

Wifi (Wireless Fidelity) là thuật ngữ dùng chung dùng để chỉ tiêu chuẩn IEEE802.11 cho mạng cục bộ không dây (Wireless Local Networks) hoặc WLANs.

1.3. Giới thiệu về vi điều khiển ESP32

ESP32 là một module Wi-Fi và Bluetooth tích hợp mạnh mẽ, được phát triển bởi Espressif Systems, một công ty công nghệ có trụ sở tại Trung Quốc. Được giới thiệu lần đầu vào năm 2016, ESP32 đã nhanh chóng trở thành một trong những platform phổ biến cho phát triển các ứng dụng IoT (Internet of Things) và thiết bị kết nối không dây[4].

Hình 1. 6 HÌnh ảnh thực tế của ESP 32

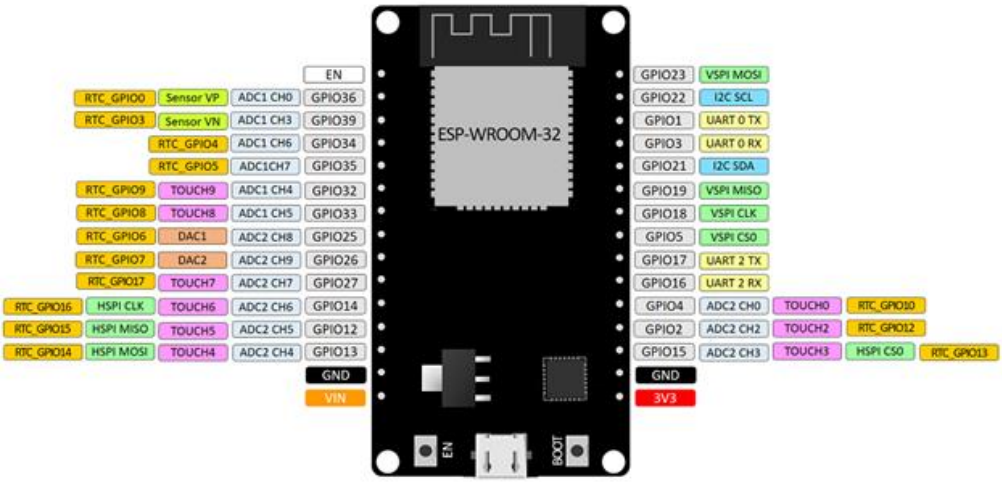
1.3.1. *Cấu tạo của ESP32*

- Chân GPIO: chân GPIO được sử dụng để kết nối với các thành phần ngoại vi như cảm biến, bảng mạch, LED, và nhiều thiết bị khác.
- Chân nguồn và GND:
  3V3 và 5V: chân này cung cấp nguồn 3.3V hoặc 5V cho việc nạp chương trình và cung cấp nguồn cho các thiết bị ngoại vi.
  GND: chân đất, kết nối với đất.
- Chân UART (Universal Asynchronous Receiver/Transmitter): chân TX (Transmit) và RX(Receive) sử dụng để truyền dữ liệu qua chuẩn giao tiếp UART.
- Chân SPI (Serial Peripheral Interface): MISO, MOSI, SCK, CS/SS: dùng cho giao tiếp SPI với các thiết bị ngoại vi như cảm biến, màn hình, bộ nhớ flash,...
- Chân I2C (Inter-Integrated Circuit): SDA và SDL dùng cho giao tiếp với các thiết bị như cảm biến, màn hình, EEPROM và nhiều thiết bị khác.
- Chân PWM (Pulse Width Modulation): chân được sử dụng để tạo xung PWM, thường được sử dụng để kiểm soát độ sáng của LED, servo motor, và các ứng dụng khác.
- Chân ADC (Analog-to-Digital Converter): được sử dụng để đọc giá trị Analog từ cảm biến và các thiết bị khác.
- Chân nguồn và kiểm soát nguồn:

  EN (Enable): chân kích hoạt, thường được sử dụng để bật hoặc tắt nguồn của ESP32.

VIN: chân này sử dụng để cấp nguồn từ bên ngoài.

- Chân GPIO đặc biệt: chân GPIO 34 đến GPIO 39: các chân GPIO có chức năng đặt biệt như : đầu vào ngắn mạch, nút nhấn (button), hoặc có thể được sử dụng cho các chức năng khác.



Hình 1. 7 Chức năng các chân của ESP32

1.3.2. *Tính năng của ESP32*

- Kết nối Wi-Fi và Bluetooth: ESP32 hỗ trợ Wi-Fi 802.11 b/g/n và Bluetooth Classic (Bluetooth BR/EDR) cùng với Bluetooth Low Energy (BLE). Điều này cho phép nó kết nối đến mạng Wi-Fi và giao tiếp với các thiết bị Bluetooth khác, cung cấp tính năng linh hoạt cho các ứng dụng IoT.
- Vi xử lý mạnh mẽ: ESP32 được trang bị một vi xử lý kép với hai lõi Tensilica Xtensa LX6 có tốc độ cao, cung cấp khả năng xử lý mạnh mẽ và đủ lý tưởng cho các ứng dụng đòi hỏi sự phức tạp, như xử lý dữ liệu sensor và truyền thông.
- Chế độ tiết kiệm năng lượng: ESP32 có các chế độ tiết kiệm năng lượng linh hoạt giúp gia tăng tuổi thọ pin trong các ứng dụng di động hoặc thiết bị chạy bằng pin.
- Hỗ trợ nhiều giao diện: Ngoài giao diện Wi-Fi và Bluetooth, ESP32 còn có nhiều giao diện khác như UART, SPI, I2C, GPIO, và nhiều giao diện khác, cho phép kết nối với nhiều loại thiết bị ngoại vi.
- Hệ thống phát triển phổ biến: Cộng đồng phát triển cho ESP32 rất lớn, và có sẵn nhiều tài liệu, thư viện và ví dụ để giúp các nhà phát triển dễ dàng bắt đầu. Arduino IDE và PlatformIO hỗ trợ việc phát triển ứng dụng cho ESP32.

- Giá trị chi phí tốt: ESP32 thường có giá trị chi phí tốt, là một giải pháp phù hợp cho các dự án IoT và nhúng.
- Hỗ trợ OTA (Over-the-Air) cập nhật: ESP32 cho phép cập nhật phần mềm từ xa thông qua kết nối Wi-Fi, giúp dễ dàng và hiệu quả trong việc duyệt nguồn thường xuyên.
- Hệ thống đa nhiệm:

– Hỗ trợ nhiều luồng xử lý đồng thời và có thể thực hiện các tác vụ đa nhiệm.

- Hệ điều hành tương thích:

– Hỗ trợ nhiều hệ điều hành nhúng, bao gồm FreeRTOS.

- Giao tiếp OTA (Over-the-Air):

– Hỗ trợ cập nhật phần mềm từ xa qua kết nối Wi-Fi.

- Nhiệt độ hoạt động:

– Hoạt động trong khoảng nhiệt độ -40°C đến 85°C.

- Chế độ điện tiêu chuẩn và chế độ ngủ sâu:

– Hỗ trợ chế độ tiết kiệm năng lượng trong thời gian không hoạt động.

1.4. Giới thiệu về cảm biến SHT 30

Cảm biến SHT30 là một loại cảm biến nhiệt độ và độ ẩm được sản xuất bởi Sensirion AG, một công ty chuyên về cảm biến và ứng dụng liên quan đến môi trường. Cảm biến SHT30 là một phiên bản nâng cấp của SHT3x, cung cấp độ chính xác và hiệu suất tốt hơn trong việc đo nhiệt độ và độ ẩm[5].

Hình 1. 8 Cảm biến SHT30

Một số đặc điểm nổi bật của cảm biến SHT30:

- Đo nhiệt độ và độ ẩm: Cảm biến SHT30 có khả năng đo cả nhiệt độ và độ ẩm trong môi trường xung quanh nó. Điều này rất hữu ích trong nhiều ứng dụng, bao gồm điều khiển điều hòa không khí, giám sát môi trường, và nhiều ứng dụng khác.
- Độ chính xác cao: Cảm biến này có độ chính xác cao trong việc đo nhiệt độ và độ ẩm, giúp đảm bảo dữ liệu thu thập được chính xác và đáng tin cậy.
- Giao tiếp dễ dàng: SHT30 thường được tích hợp với giao diện số I2C hoặc SPI, cho phép dễ dàng kết nối với các vi điều khiển, vi xử lý, hoặc các hệ thống nhúng.
- Kích thước nhỏ gọn: Cảm biến SHT30 có kích thước nhỏ, giúp dễ dàng tích hợp vào các thiết bị và ứng dụng có hạn chế về không gian.
- Tiết kiệm năng lượng: SHT30 được thiết kế để tiết kiệm năng lượng, giúp kéo dài thời gian sử dụng pin trong các ứng dụng di động hoặc sử dụng pin.
- Ứng dụng đa dạng: Cảm biến SHT30 có thể được sử dụng trong nhiều lĩnh vực, bao gồm trong các thiết bị IoT (Internet of Things), thiết bị y tế, hệ thống tự động hóa, thiết bị tiêu dùng và nhiều ứng dụng khác liên quan đến giám sát và điều khiển môi trường.

Cảm biến SHT30 là một trong những lựa chọn phổ biến để đo nhiệt độ và độ ẩm với độ chính xác trong nhiều ứng dụng khác nhau.

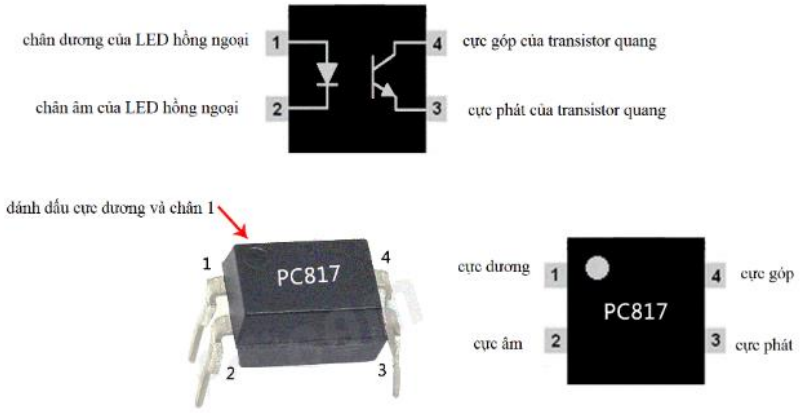
Thông số kỹ thuật[5]:

- Model: SHT30 V3 I2C
- Vỏ cảm biến bằng PVC chống va đập, thiết kế hở giúp cảm biến cảm nhận các thông số thay đổi của môi trường nhanh tuy nhiên sẽ không có khả năng chống nước, bụi.
- Cảm biến phía trong: SHT30
- Chuẩn giao tiếp: I2C
- Mức tín hiệu giao tiếp: TTL 3.3~5VDC
- Điện áp sử dụng: 2.15~5.5VDC
- Khoảng nhiệt độ đo được: -40 ~ 125 độ C, sai số 0.2 độ C
- Khoảng độ ẩm đo được: 0 ~100% RH, sai số 2% RH.
- Có sẵn trở treo 10K và tụ lọc nhiễu.
- Kích thước cảm biến: 15 x 66.5mm
- Chiều dài dây dẫn: 50cm.

1.5. Cách ly quang

Mục đích của việc cách ly quang để ngăn cách giữa Relay và vi điều khiển nhằm bảo vệ mạch và vi điều khiển khi có sự cố về điện xảy ra. IC cách ly quang được sử dụng trong mạch là Opto quang PC817.

Kí hiệu, sơ đồ mạch và hình ảnh thực tế của IC PC817:



Hình 1. 9 IC PC817

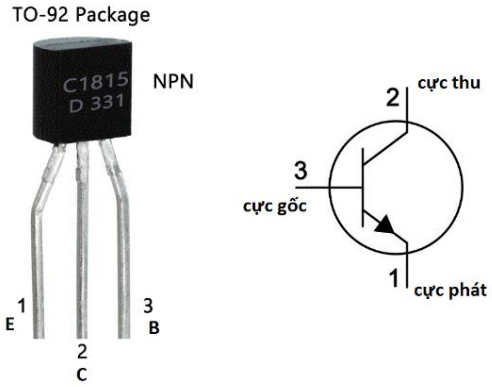
Thông số kĩ thuật:

Thông số	Ký hiệu	Giá trị
Điện áp ngược đầu vào lớn nhất	V_R	6V
Điện áp đầu ra C - E lớn nhất	V_{CEO}	35V
Dòng thuận lớn nhất	I_F	50
Dòng điện Collector lớn nhất	I_C	50

Bảng 1. 1 Thông số kĩ thuật của IC PC817

1.6. Khuếch đại và relay

Tín hiệu ra của vi điều khiển và Opto quang rất thấp, do đó cần phải khuếch đại nhằm đảm bảo đủ dòng để điều khiển cuộn hút của Relay. Mạch sử dụng Transistor C1815 mắc phân cực cố định để điều khiển các thiết bị điện xoay chiều 220V. Mạch sử dụng các Relay 5V với tiếp điểm 250V 10A[1].



Hình 1. 10 IC C1815

Thông số kĩ thuật:

Bảng 1.1: Thông số chính của IC C1815

Ký hiệu	Thông số	Điều kiện thử nghiệm	Min	Max
I_{CBO}	Dòng cắt Collector	V_{CB}=60V, I_E=0		0.1µA
I_{EBO}	Dòng cắt Emitter	V_{EB}=5V, I_C=0		0.1µA
h_{FE1} h_{FE2}	Hệ số khuyếch đại DC	V_{CE}=6V, I_C=2mA V_{CE}=6V, I_C=150mA	70 25	700
$V_{CE\ (sat)}$	Điện áp bão hòa Collector-Emitter	I_C=100mA, I_B=10mA		0.25V
$V_{BE\ (sat)}$	Điện áp bão hòa Base-Emitter	I_C=100mA, I_B=10mA		1V

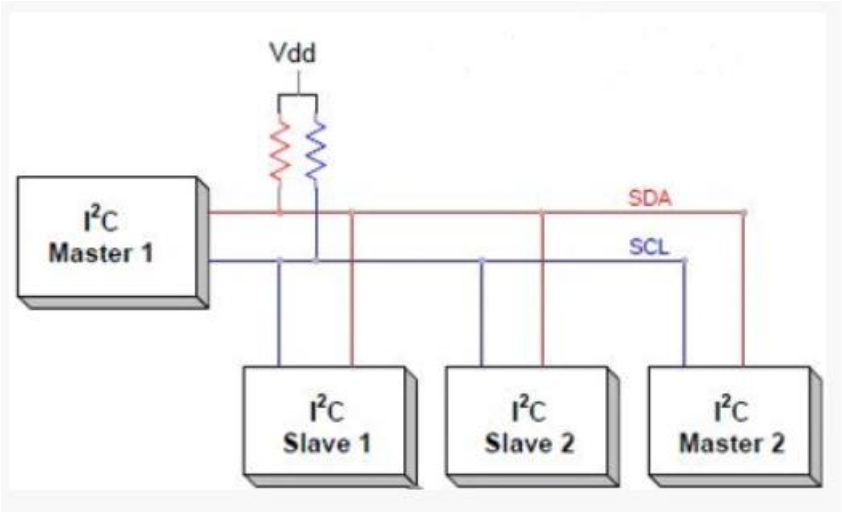
1.7. Chuẩn giao tiếp I2C

- Cấu tạo

I2C sử dụng 2 đường truyền tín hiệu:

SCL - Serial Clock Line : Tạo xung nhịp đồng hồ do Master phát đi

SDA - Serial Data Line : Đường truyền nhận dữ liệu



Hình 1. 11 Giao tiếp I2C

Giao tiếp I2C bao gồm quá trình truyền nhận dữ liệu giữa các thiết bị chủ tớ, hay Master - Slave.

Thiết bị Master là 1 vi điều khiển, nó có nhiệm vụ điều khiển đường tín hiệu SCL và gửi nhận dữ liệu hay lệnh thông qua đường SDA đến các thiết bị khác.

Các thiết bị nhận các dữ liệu lệnh và tín hiệu từ thiết bị Master được gọi là các thiết bị Slave. Các thiết bị Slave thường là các IC, hoặc thậm chí là vi điều khiển.

Master và Slave được kết nối với nhau như hình trên. Hai đường bus SCL và SDA đều hoạt động ở chế độ Open Drain, nghĩa là bất cứ thiết bị nào kết nối với mạng I2C này cũng chỉ có thể kéo 2 đường bus này xuống mức thấp (LOW), nhưng lại không thể kéo được lên mức cao. Vì để tránh trường hợp bus vừa bị 1 thiết bị kéo lên mức cao vừa bị 1 thiết bị khác kéo xuống mức thấp gây hiện tượng ngắn mạch. Do đó cần có 1 điện trờ (từ 1 – 4,7 kΩ) để giữ mặc định ở mức cao.

- **Các chế độ hoạt động của I2C:**

  Chế độ chuẩn (standard mode) với tốc độ 100 kBit/s.

  Chế độ tốc độ thấp (low speed mode) với tốc độ 10 kBit/s.

Ngoài ra, khác với giao tiếp SPI chỉ có thể có 1 Master, giao tiếp I2C cho phép chế độ truyền nhận dữ liệu giữa nhiều thiết bị Master khác nhau với thiết bị Slave. Tuy nhiên quá trình này có hơi phức tạp vì thiết bị Slave có thể nhận 1 lúc nhiều khung dữ liệu từ các thiết bị Master khác nhau, điều đó đôi khi dẫn đến xung đột hoặc sai sót dữ liệu nhận được.

Để tránh điều đó, khi làm việc ở chế độ này, mỗi thiết bị Master cần phát hiện xem đường SDA đang ở trạng thái nào. Nếu SDA ở mức 0, nghĩa là đang có 1 thiết bị Master khác đang có quyền điều khiển và phải chờ đến khi truyền xong. Ngược lại nếu SDA ở mức 1, nghĩa là đường truyền SDA đã an toàn và có sử dụng .

1.8. Hệ điều hành FreeRTOS

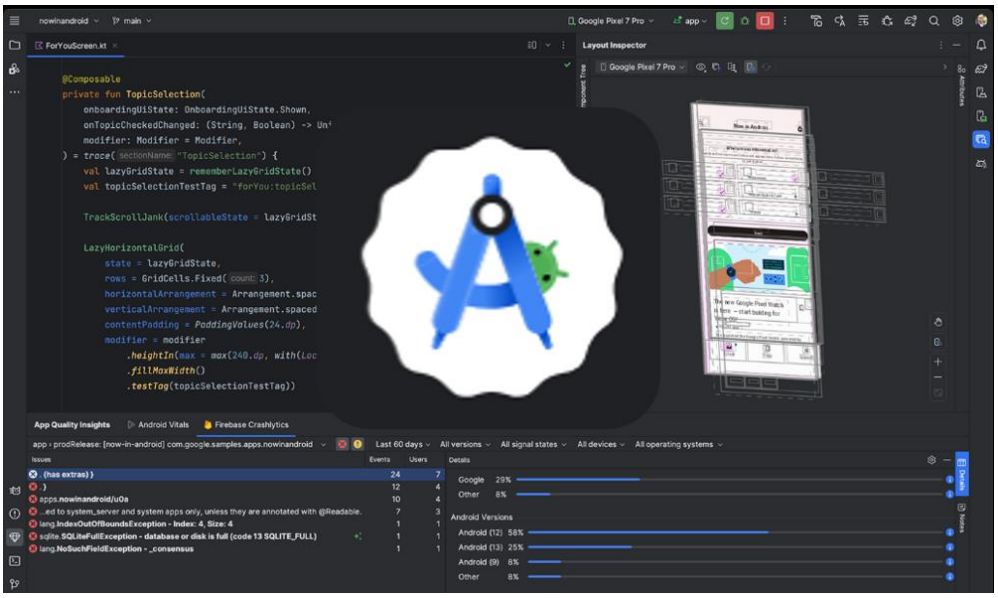
FreeRTOS là một hệ điều hành thời gian thực mã nguồn mở (RTOS - Real-Time Operating System) dành cho các ứng dụng nhúng (embedded applications). Nó được phát triển bởi Richard Barry và sử dụng giấy phép mã nguồn mở (open-source license) phù hợp với nhiều dự án nhúng.

FreeRTOS có các đặc điểm sau:

- Thời gian thực: FreeRTOS thiết kế đặc biệt để hoạt động trong các ứng dụng thời gian thực, nơi các tác vụ phải được thực hiện trong khoảng thời gian chính xác. Nó cung cấp các cơ chế lập lịch và ưu tiên để đảm bảo tác vụ quan trọng được thực hiện đúng thời điểm.
- Mã nguồn mở: FreeRTOS là một dự án mã nguồn mở, điều này có nghĩa là bạn có quyền sửa đổi mã nguồn và sử dụng nó trong các dự án thương mại mà không phải trả bất kỳ giấy phép nào.
- Xây dựng cho ứng dụng nhúng: FreeRTOS được tối ưu hóa để sử dụng ít tài nguyên hệ thống và có kích thước nhỏ, nên nó phù hợp cho các ứng dụng nhúng có tài nguyên hạn chế.
- Hỗ trợ đa nhiệm và nhiều tiến trình: FreeRTOS cho phép bạn chia tác vụ thành nhiều tiến trình và quản lý chúng bằng cơ chế lập lịch.
- Hỗ trợ nhiều kiến thức lập trình: FreeRTOS hỗ trợ nhiều ngôn ngữ lập trình như C, C++, Ada, và Rust.
- Thư viện bổ sung: Cộng đồng FreeRTOS đã phát triển nhiều thư viện bổ sung và phiên bản tùy chỉnh cho nhiều loại vi điều khiển và kiến thức.
- Hỗ trợ đa nền tảng: FreeRTOS có sẵn cho nhiều nền tảng vi điều khiển (microcontroller platforms), giúp bạn dễ dàng di chuyển mã nguồn giữa các nền tảng khác nhau.

1.9. Phần mềm điều khiển

Sử dụng công cụ lập trình là Android Studio để thiết kế phần mềm điều khiển. Android Studio là môi trường phát triển tích hợp chính thức dành cho phát triển nền tảng Android. Nó được ra mắt vào ngày 16 tháng 5 năm 2013 tại hội nghị Google I/O.



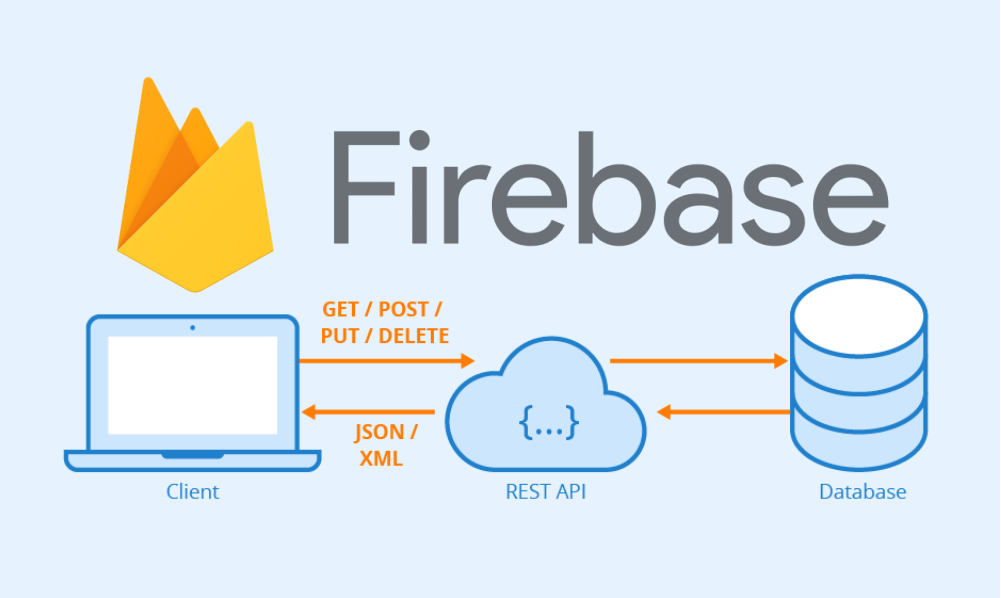
Hình 1. 12 Giao diện Android studio

Ngôn ngữ lập trình Kotlin được sử dụng để viết phần mềm điều khiển. Kotlin là một ngôn ngữ lập trình nguồn mở, kiểu tĩnh, hỗ trợ cả lập trình chức năng lẫn hướng đối tượng. Kotlin cung cấp cú pháp và khái niệm tương tự trong các ngôn ngữ khác, bao gồm cả C#, Java và Scala cùng nhiều ngôn ngữ khác. Kotlin không phải là độc nhất – mà Kotlin lấy cảm hứng từ nhiều thập kỷ để phát triển ngôn ngữ. Mã này tồn tại trong các biến thể nhắm đến JVM (Kotlin/VM), JavaScript (Kotlin/JS) và mã gốc (Kotlin/mã gốc).

Hình 1. 13 Ngôn ngữ lập trình Kotlin

1.10. Cơ sở dữ liệu Fire-base

Firebase là một nền tảng phát triển ứng dụng di động và web của Google, bao gồm một loạt dịch vụ và sản phẩm cho việc xây dựng ứng dụng di động và web. Firebase cung cấp một số tính năng quan trọng liên quan đến cơ sở dữ liệu và quản lý dữ liệu. Firebase Realtime Database là một trong các dịch vụ chính của Firebase liên quan đến cơ sở dữ liệu.



Hình 1. 14 Firebase

Một số điểm quan trọng về Firebase Realtime Database:

- Cơ sở dữ liệu thời gian thực: Firebase Realtime Database là một cơ sở dữ liệu thời gian thực (real-time database), có nghĩa rằng dữ liệu trong nó được cập nhật ngay lập tức khi có sự thay đổi. Điều này làm cho nó phù hợp cho các ứng dụng đòi hỏi sự đồng bộ và cập nhật nhanh chóng giữa các thiết bị.
- Dữ liệu dưới dạng JSON: Firebase Realtime Database lưu trữ dữ liệu dưới dạng JSON, điều này làm cho việc lưu trữ và truy xuất dữ liệu dễ dàng và trực quan.
- Thời gian thực và đồng bộ hóa: Firebase Realtime Database tự động đồng bộ hóa dữ liệu giữa các thiết bị và người dùng khác nhau, giúp tạo ra trải nghiệm người dùng liền mạch.
- Quyền truy cập và bảo mật: Firebase cung cấp các tùy chọn để kiểm soát quyền truy cập và bảo mật dữ liệu. Bạn có thể thiết lập quyền

truy cập dựa trên người dùng, phạm vi dữ liệu, và nhiều thông tin khác.

- Hỗ trợ đa nền tảng: Firebase Realtime Database có thư viện và SDK hỗ trợ cho nhiều nền tảng phát triển, bao gồm Android, iOS, web, và nhiều ngôn ngữ lập trình khác.
- Dịch vụ liên quan: Firebase cung cấp nhiều dịch vụ và sản phẩm khác, bao gồm xác thực người dùng, quảng cáo, thông báo đẩy, phân tích, lưu trữ tệp, và nhiều tính năng khác.

CHƯƠNG 2. THIẾT KẾ HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ , ĐỘ ẨM VÀ ĐIỀU KHIỂN THIẾT BỊ ĐIỆN QUA INTERNET DÙNG ESP32

2.1. Yêu cầu của hệ thống

2.1.1. *Chức năng*

- Quan trắc nhiệt độ độ ẩm.
- Dóng, cắt thiết bị điện .
- Theo dõi và điều khiển thông qua ứng dụng trên điện thoại thông minh.

2.1.2. *Yêu cầu kĩ thuật*

- Hệ thống sử dụng nguồn điện tử Adapter 12V
- Đo nhiệt độ từ 40 ~ 125 độ C, sai số 0.2 độ C
- Khoảng độ ẩm đo được: 0 ~100% RH, sai số 2% RH.
- Kích thước màn hình LCD: 16x2
- Số ngõ ra công suất:4
- Điện áp tải tối đa AC 250V-10A / DC 30V-10A

2.2. Đề xuất giải pháp thiết kế

Mô hình thiết kế ngoài việc đảm bảo các yêu cầu vận hành, chức năng cần có, còn cần đảm bảo việc ứng dụng tối ưu kiến thức được đào tạo tại trường vào giải quyết các vấn đề của bài toán đồng thời chọn ra được thiết kế tối ưu về mặt trang thiết bị, kinh phí. Theo đó nhóm xin đề xuất một số phương án như sau:

- Phương án 1: Sử dụng kết nối bluetooth của ESP 32 để truyền dữ liệu lên ứng dụng điều khiển, sử dụng cảm biến DHT11 để thu thập thông tin về nhiệt độ, độ ẩm môi trường. Sử dụng kết nối bluetooth để điều khiển các thiết bị trong hệ thống
- Phương án 2: Sử dụng kết nối Wifi với modul ESP8266 điều khiển các ngõ ra của hệ thống và gửi dữ liệu đến phần mềm điều khiển. Sử dụng cảm biến DHT 11 để thu thập thông tin về nhiệt độ và độ ẩm.
- Phương án 3: Sử dụng kết nối wifi của modul ESP32 để gửi dữ liệu đến phần mềm điều khiển và điều khiển các ngõ ra. Sử dụng cảm biến SHT30 để thu thập dữ liệu nhiệt độ, độ ẩm.

2.3. Lựa chọn giải pháp thiết kế

Phương án 1:

Ưu điểm: Chi phí thấp vì giá thành của vi điều khiển và cảm biến rẻ. Lập trình trên nền Arduino rất dễ ràng và có nhiều nguồn tham khảo lớn, uy tín.

Nhược điểm: Độ tin cậy của cảm biên không cao ,sai số 2 độ C - 5% RH. Sử dụng kết nối bluetooth sẽ bị hạn chế khoảng cánh điều khiển khi sử dụng phần mềm điều khiển.

Phương án 2:

Ưu điểm: Giá thành của sản phẩm rẻ mà vẫn đáp ứng đủ yêu câu của bài toán. Sử dụng kết nối wifi sẽ không giới hạn khoảng cách điều khiển thiết bị từ phần mềm điều khiển.

Nhược điểm: Độ tin cậy của cảm biến chưa cao.

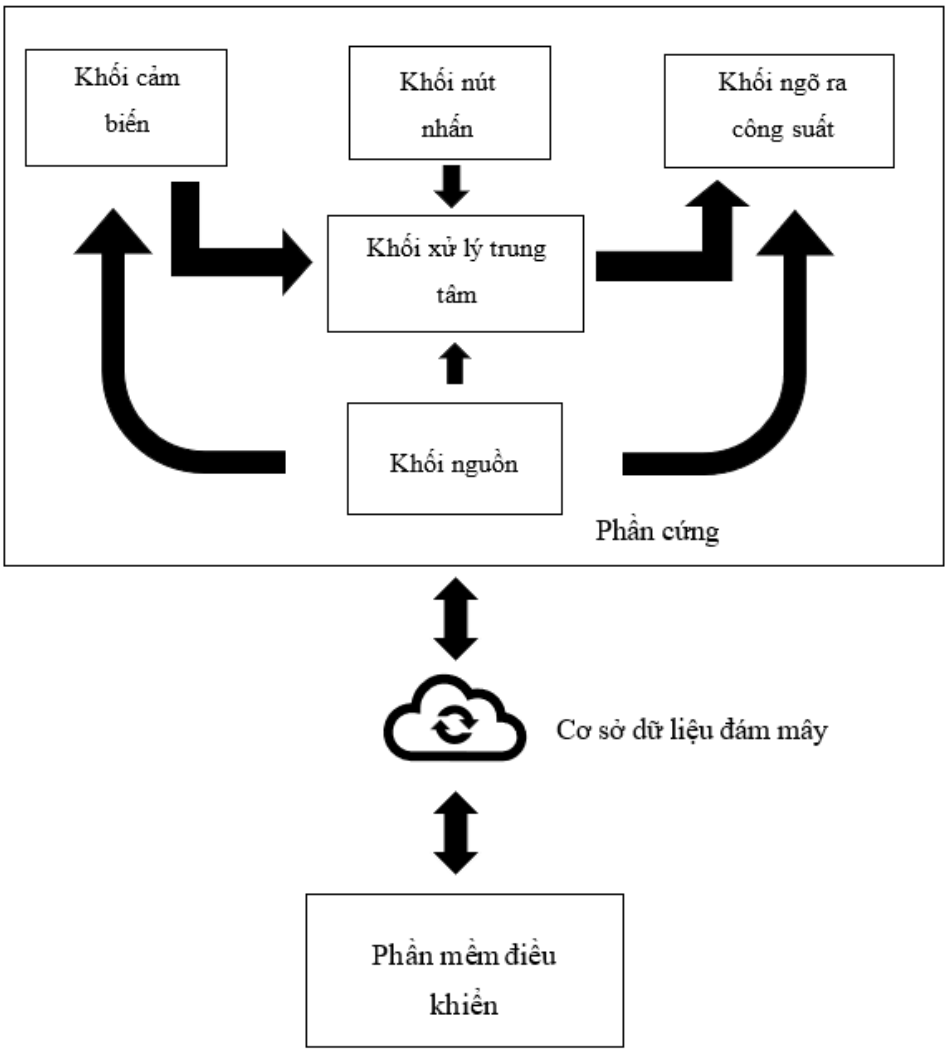
Phương án 3:

Ưu điểm: ESP32 là một module wifi tích hợp, vì vậy việc kết nối nó với internet rất đơn giản, có thể được sử dụng để điều khiển nhiều thiết bị khác nhau, giúp mở rộng khả năng của hệ thống. Nó còn là một module wifi giá rẻ, giúp giảm tổng chi phí của hệ thống.Độ tin cậy khi sử dụng cảm biến SHT30 là khá cao, sai số của cảm biến SHT 30 là 0.2 độ - 2% RH.

Nhược điểm: Giá thành sản phẩm cao do giá của cảm biến khá cao.

Nhằm cân đối các nội dung đồ án, vận dụng tối đa những kiến thức được đào tạo nhóm em lựa chọn phương án thứ 3 để thiết kế đồ án mô phỏng. Phương án này sẽ cân đối về mặt chi phí và độ tin cậy của sản phẩm.Nhóm lựa chọn phương án có chi phí cao hơn nhưng nó mang lại độ ổn định và chất lượng cho sản phẩm .

2.4. Sơ đồ khối của hệ thống



Hình 2. 1 Sơ đồ khối của hệ thống

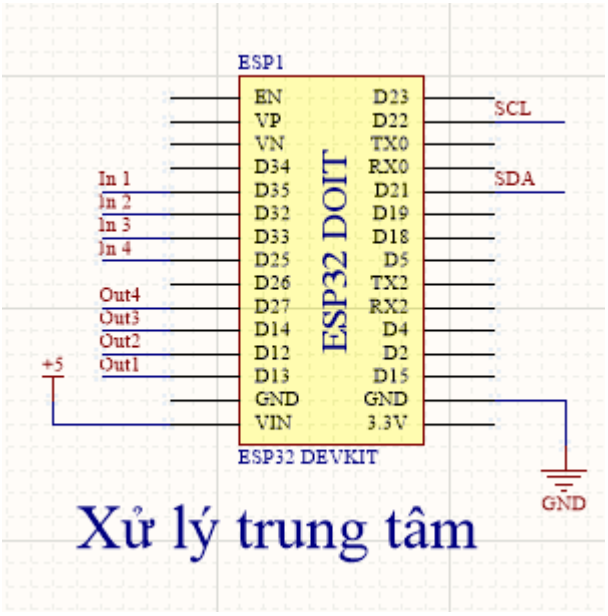
- Khối xử lý trung tâm: Trung tâm điều khiển hoạt động của toàn bộ hệ thống. Nhận tín hiệu từ ứng dụng Android hoặc nút nhấn, xử lý sau đó chuyển tín hiệu điều khiển đến khối công suất thực thi, tiếp theo dữ liệu được gửi lên khối Server. Khối cảm biến nhiệt độ sử dụng cảm biến SHT 30 có khả năng đo cả 2 thông số nhiệt độ và độ ẩm.
- Khối nguồn: Cấp nguồn cho toàn mạch, sử dụng nguồn 5VDC cấp cho khối xử lý trung tâm, mạch Relay, cảm biến và nguồn cho các thiết bị điện.
- Khối ngõ ra công suất:Đóng ngắt các tiếp điểm Relay theo sự điều khiển của ngõ ra vi điều khiển, từ đó điều khiển các thiết bị điện (220VAC). Đồng thời cách ly giữa mạch công suất và mạch điều khiển.

- Khối cảm biến: Có chức năng giám sát nhiệt độ và độ ẩm của môi trường và gửi dữ liệu nhiệt độ độ ẩm về khối xử lý trung tâm.
- Phần mềm điều khiển: Xử lý và gửi tín hiệu điều khiển đến vi điều khiển, điều khiển trực tiếp trên điện thoại thông minh. Hiển thị thông tin nhiệt độ, độ ẩm.
- Nút nhấn: Gửi tín hiệu đến vi điều khiển để điều khiển ngõ ra của vi điều khiển. Từ đó điều khiển trạng thái tắt bật của relay.

2.5. Tính toán thiết kế phần cứng

2.5.1. *Khối xử lý trung tâm*

Khối điều khiển sử dụng vi điều khiển ESP32 đáp ứng được yêu cầu đặt ra và có khả năng mở rộng cho nhiều ứng dụng.Vi điều được cấp nguồn 5VDC. Chi tiết kết nối giữa vi điều khiển và các khối khác được thể hiện trong hình bên dưới:

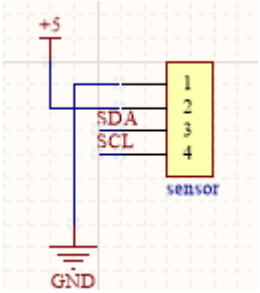


Hình 2. 2 Khối xử lý trung tâm

- Chân nguồn Vin và GND được nối với mạch nguồn cung cấp 5VDC.
- Các chân D27, D14, D12, D13 lần lượt nối với ngõ vào của các Relay tương ứng.
- Các chân D35, D32, D33, D25 được nối với các nút nhấn.
- Chân D22 và D21 lần lượt là chân SCL và SDA của chuẩn giao tiếp I2C trên ESP32 được nối với I2C của LCD và cảm biến SHT30.

2.5.2. *Khối cảm biến*

Đề tài này có giám sát nhiệt độ độ ẩm, hiển thị lên internet. Với yêu cầu đề ra thì SHT30 là cảm biến phù hợp để sử dụng trong đề tài này. SHT30 là cảm biến có mức điện áp hoạt động từ 2.15V-5.5V, dòng cung cấp 0.5mA - 2.5mA phù hợp với dòng và áp ra của bộ xử lý trung tâm để module hoạt động bình thường.



Hình 2. 3 Khối cảm biến

Cảm biến SHT30 gồm 4 chân được kết nối như sau:

- Chân VCC được nối với nguồn 5VDC.
- Chân GND nối với chân GND của nguồn.
- Chân SCL, SDA nối với chân SCL, SDA của vi điều khiển ESP32.

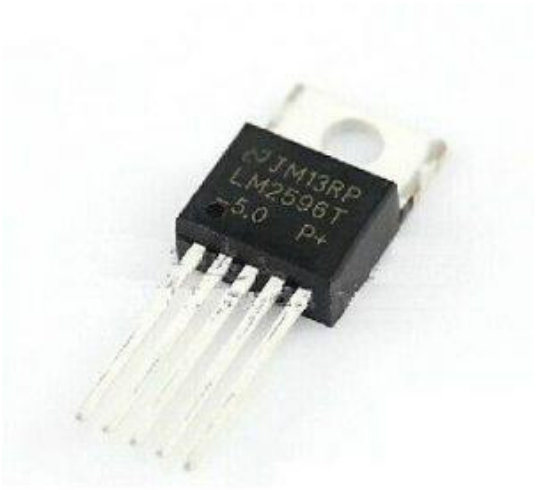
2.5.3. *Thiết kế khối nguồn*

Nguồn chính sử dụng trong mạch là nguồn 5VDC. Nguồn này được lấy từ nguồn Adapter 12V qua module hạ áp DC-DC về 5V để cấp cho các module: board ESP32, cảm biến SHT30, Relay,LCD.

STT	Tên	Số lượng	Dòng tiêu thụ(mA)	Tổng dòng tiêu thụ(A)
1	ESP32	1	250	0.25
2	SHT30	1	2.5	0.0025
3	Relay	4	80	320
4	OPTO PC817	4	70	280
5	LCD	1	10	0.01

Bảng 2. 1 Thống kê dòng điện của các linh kiện chính

Từ bảng trên tổng dòng tiêu thụ cho toàn bộ mạch điều khiển là 0.7625A vì vậy ta thiết kế mạch nguồn với IC LM2596 là hoàn toàn đủ để đáp ứng cho toàn mạch điều khiển.

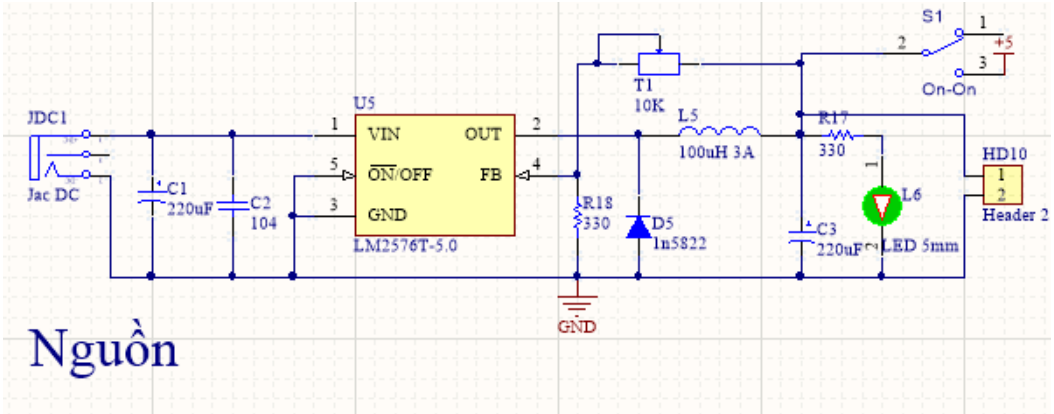


Hình 2. 4 IC Nguồn LM2596

Bộ điều chỉnh LM2596 là mạch tích hợp nguyên khối rất phù hợp lý tưởng cho thiết kế dễ dàng và thuận tiện của bộ điều chỉnh điện áp một chiều điện áp giảm (bộ chuyển đổi buck). Nó có khả năng cung cấp cho tải một dòng điện có giá trị lên đến 3.0 A. Nó được bù nội bộ để giảm thiểu số lượng các thành phần bên ngoài để đơn giản hóa thiết kế nguồn cung cấp điện.

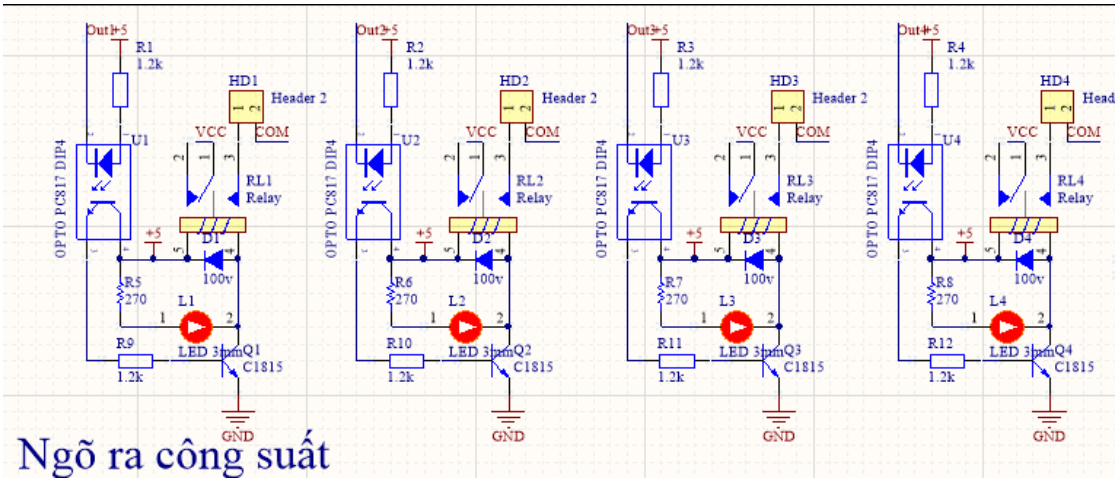
Do bộ chuyển đổi LM2596 là nguồn cung cấp năng lượng chuyển đổi, hiệu suất của nó cao hơn đáng kể so với các bộ điều chỉnh tuyến tính ba chân phổ biến, đặc biệt là với điện áp đầu vào cao hơn. LM2596 hoạt động ở tần số chuyển đổi 150 kHz, do đó cho phép các thành phần bộ lọc có kích thước nhỏ hơn mức cần thiết với các bộ điều chỉnh chuyển đổi tần số thấp hơn.

Sơ đồ nguyên lý của khối nguồn:



Hình 2. 5 Sơ đồ nguyên lý khối nguồn

2.5.4. *Thiết kế khối ngõ ra công suất*



Hình 2. 6 Sơ đồ nguyên lý khối ngõ ra công suất

Khối ngõ ra công suất được điều khiển bởi các chân GPIO của vi điều khiển ESP32. Mạch sử dụng 4 Relay để điều khiển 4 bóng đèn bật/tắt.

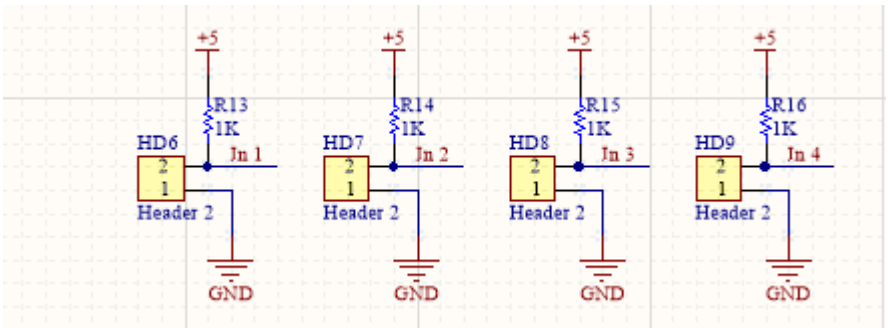
Khối Relay được cách ly với vi điều khiển Opto quang PC817, đồng thời nguồn điều khiển cuộn hút Relay được lấy trực tiếp từ khối nguồn, trong khi đó nguồn của vi điều khiển được lấy từ khối cách ly nguồn.

Mục đích dùng Opto quang kết hợp với dùng cách ly nguồn nhằm đảm bảo chống nhiễu tối đa và an toàn cho mạch điều khiển khi có sự cố từ thiết bị mà Relay điều khiển.

Mạch sử dụng thêm Diode (1N4148) mắc song song và mắc ngược với cuộn hút của Relay: nhằm ngăn chặn sự tăng đột biến điện áp lớn phát sinh khi nguồn điện bị ngắt (gọi là điện áp ngược) bảo vệ cho Relay và mạch điều khiển.

Tín hiệu lấy ra từ vi điều khiển và lấy ra từ Opto quang không đủ nên mạch sử dụng thêm mạch khuếch đại với Transistor C1815 mắc phân cực cố định nhằm tăng dòng điều khiển cho cuộn hút của Relay.

2.5.5. *Thiết kế khối nút nhấn*

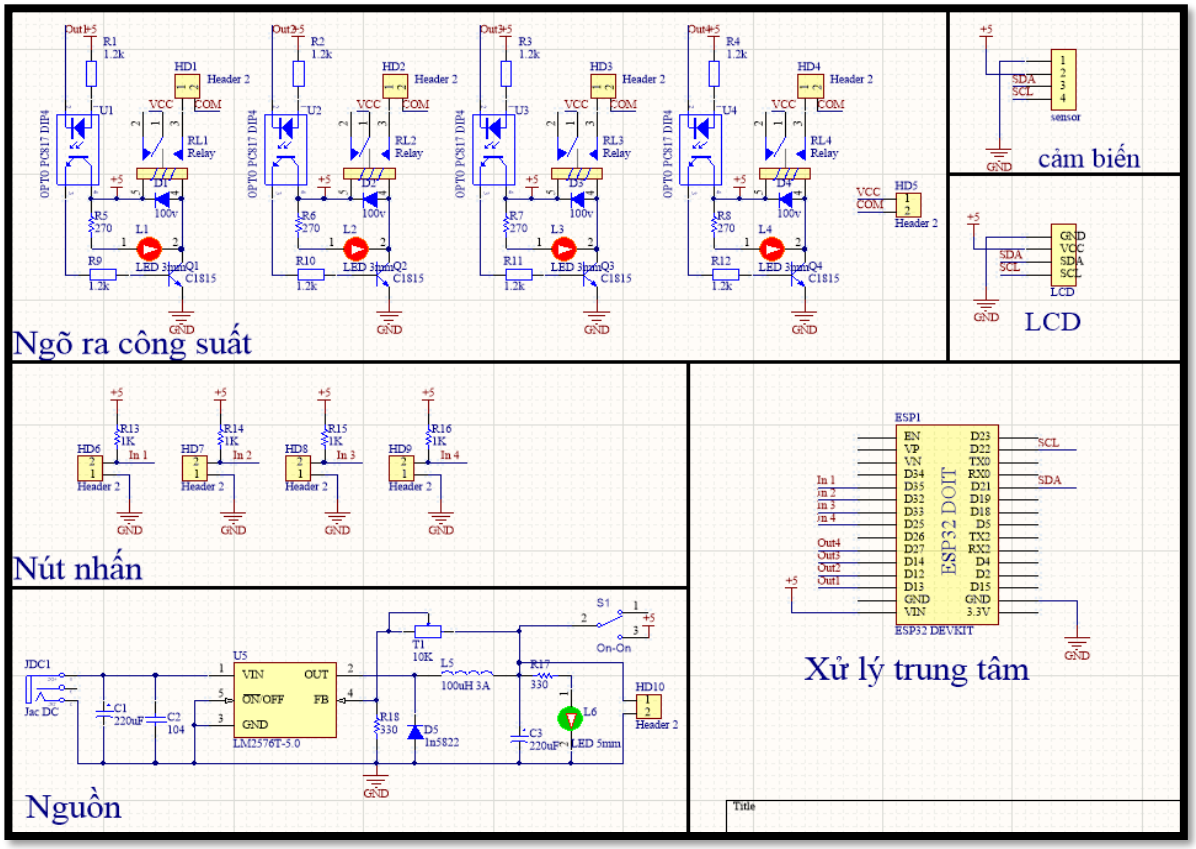


Hình 2. 7 Sơ đồ nguyên lý khối ngõ vào điều khiển

Khối ngõ vào điều khiển là 4 nút nhấn được nối với chân GPIO của vi điều khiển ESP 32 để điều khiển trạng thái của thiết bị. Nút nhấn được sử dụng ở đây là nut nhấn không giữ trạng thái và có đen báo tín hiệu cho nút nhấn.

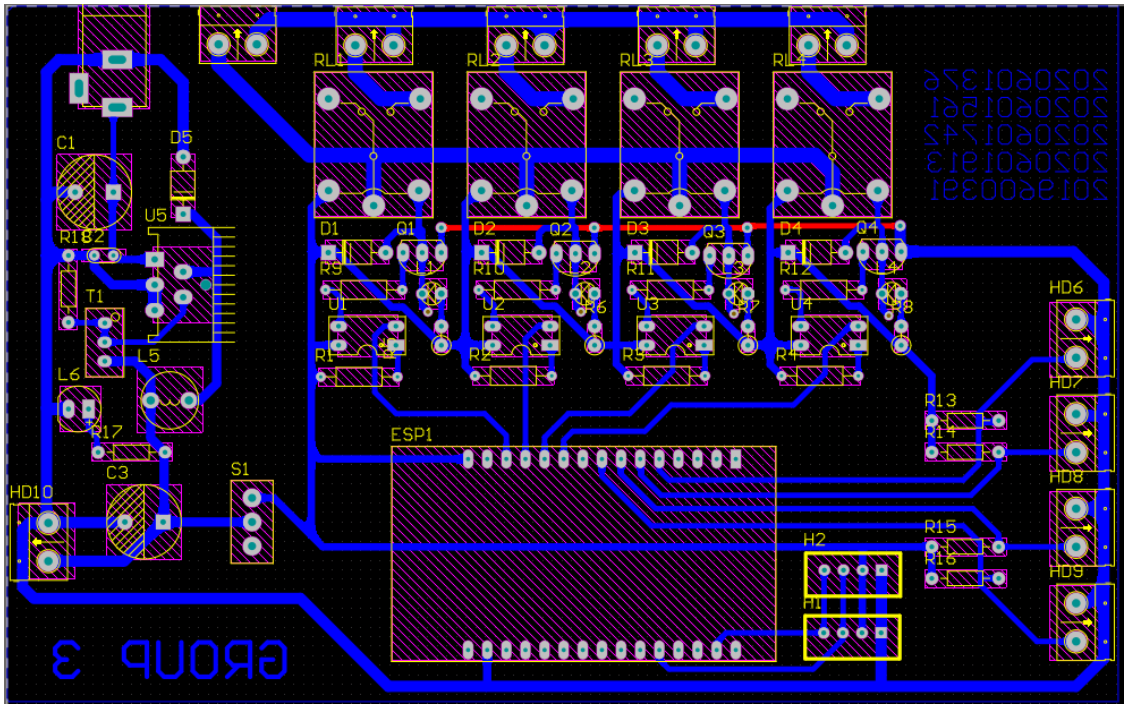
Hình 2. 8 Nút nhấn không giữ trạng thái

2.5.6. *Sơ đồ nguyên lý trên Altium designer*

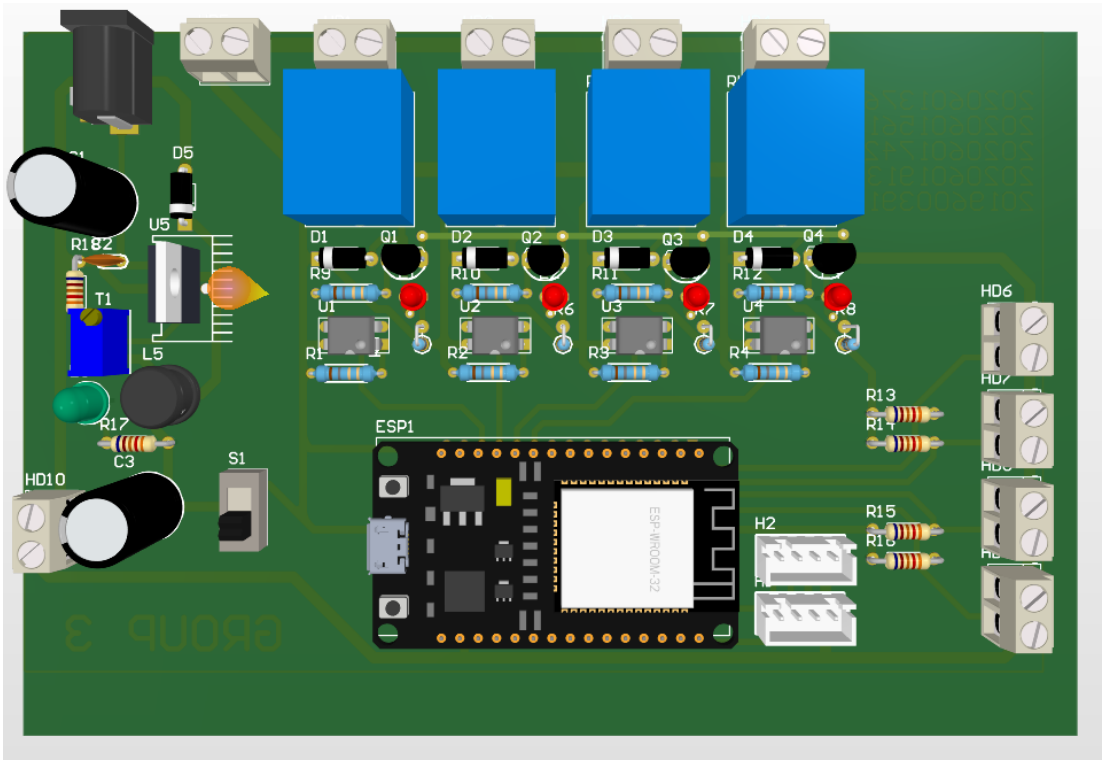


Hình 2. 9 Sơ đồ nguyên lý hệ thống

2.5.7. *Mạch in PCB trên Altium Designer*

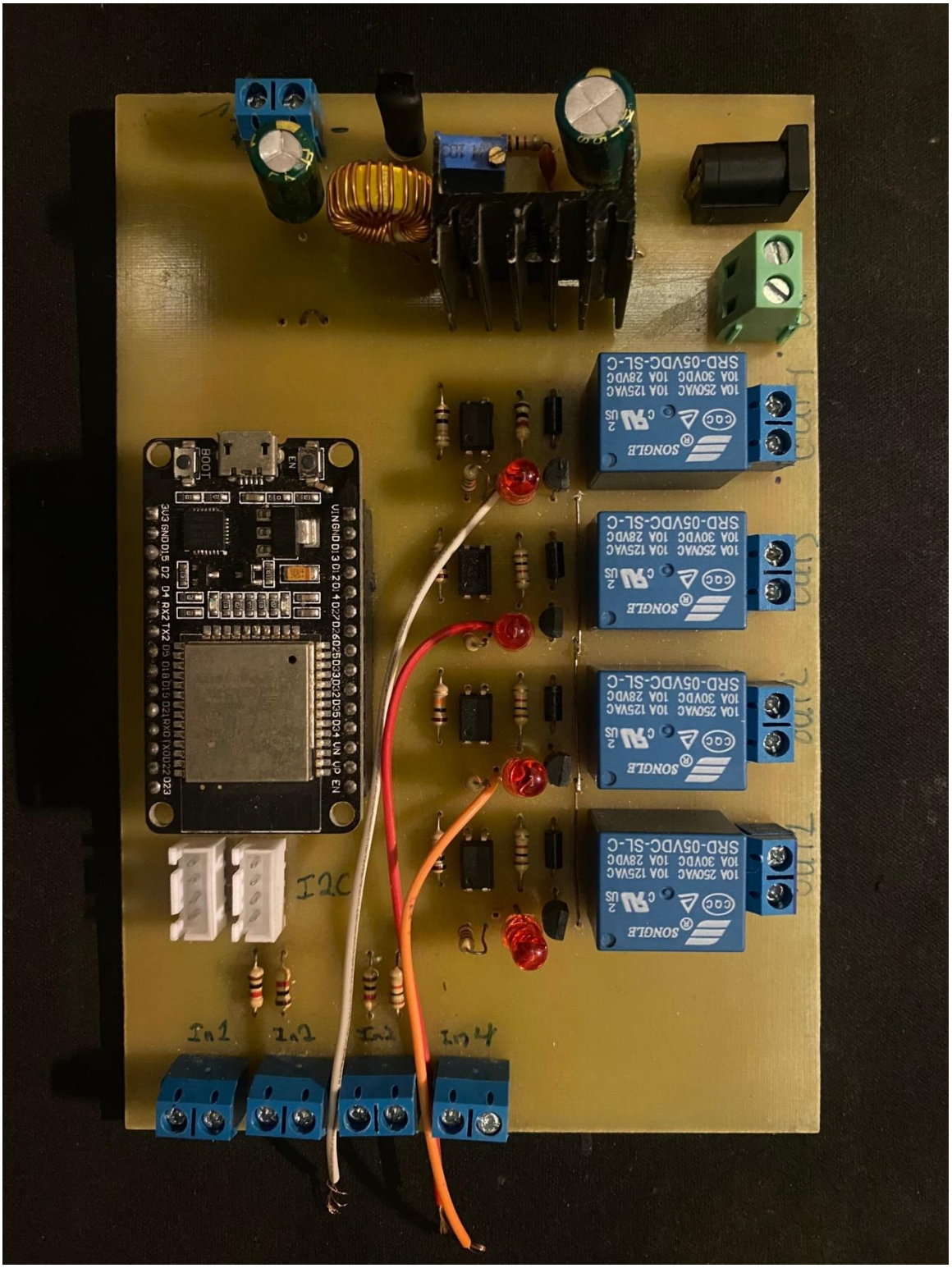


Hình 2. 10 Mạch in 2D



Hình 2. 11 Mạch in 3D

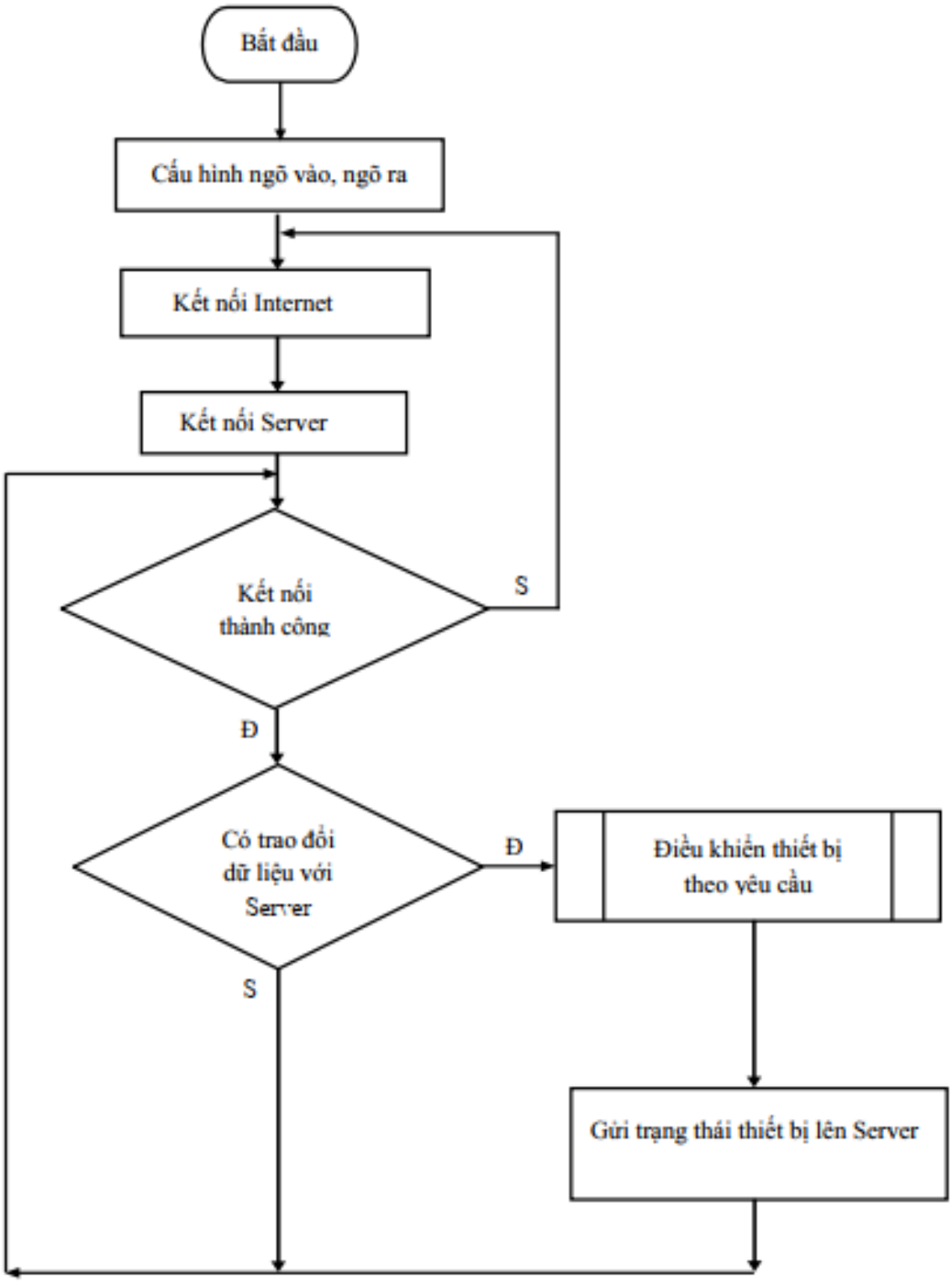
2.5.8. *Mạch thực tế*



Hình 2. 12 Mạch thực tế

2.6. Tính toán thiết kế phần mềm

2.6.1. *Lưu đồ giải thuật*



Hình 2. 13 Lưu đồ giải thuật

❖ Giải thích

Mạch điều khiển thực hiện được trong 3 trường hợp:

- Khi người dùng tác động vào giao diện trên ứng dụng Android, thì thiết bị tương ứng được bật tắt, đồng thời sẽ gửi trạng thái của thiết bị vừa được tác động lên Server.
- Khi người dùng tác động vào nút nhấn, thì ESP32 sẽ nhận tín hiệu từ nút nhấn, thiết bị tương ứng được bật tắt, đồng thời sẽ gửi trạng thái của thiết bị vừa được tác động qua Server.
- ESP32 sẽ tiến hành kết nối Internet (Wifi), và thiết lập kết nối với Server. Đợi khi kết nối thành công. Nếu có trao đổi dữ liệu với Server (người dùng tác động vào giao diện ứng dụng hoặc có tín hiệu từ Server hoặc nút nhấn gửi xuống), thì thiết bị sẽ được điều khiển theo yêu cầu người dùng. Ở bất kỳ trường hợp điều khiển nào thì trạng thái điều khiển của thiết bị cũng đều được đồng bộ trên điện thoại

2.6.2. *Lập trình cho vi điều khiển*

Arduino IDE là môi trường phát triển tích hợp với mã nguồn mở của Arduino.Đây là một ứng dụng đa nền tản và được viết trên nền tản bằng ngôn ngữ Java, và từ IDE này chương trình sẽ được sử dụng cho ngôn ngữ lập trình nguồn mở khác. Chương trình được thiết kế nhằm giúp cho những người dùng mới có thể làm quen dễ dàng với lĩnh vực phát triển phần mềm. Nó bao gồm đầy đủ các phần như các phần mềm lập trình khác nhưng với mức độ dễ sử dụng hơn như: đánh dấu cú pháp, tự động canh lề, biên dịch và nạp chương trình lên board. Chương trình của Arduino được gọi là Sketch.

Các chương trình khi lập trình trên phần mềm được viết bằng ngôn ngữ C hoặc C++. Trên Arduino IDE người dùng chỉ cần định nghĩa 2 hàm để tạo ra được một chương trình hoàn chỉnh có thể chạy được gồm:

- Setup(): hàm này chạy mỗi khi khởi động chương trình, dùng để thiết đặt các thông số cài đặt từ đầu.
- Loop(): hàm này được hiểu là vòng lặp cho đến khi không sử dụng nữa hay ngắt nguồn board điều khiển

2.6.3. *Thiết kế, lập trình giao diện hiển thị và điều khiển*

Sử dụng android studio để thiết kế giao diện điều khiển với các chức năng sau:

- Hiển thị thông tin nhiệt độ, độ ẩm
- Điều khiển 4 relay bằng nut nhấn trên màn hình giao diện

2.6.4. *Đăng kí và tạo cơ sở dữ liệu với Firebase*

Để tạo được project Firebase, điều tất nhiên ta cần là 1 tài khoản Google và thực hiện các bước sau:

1. Login vào tài khoản Google.
2. Truy cập: https://console.firebase.google.com
3. Nhấn nút “Add Project”
4. Một bảng popup sẽ hiện ra, ta sẽ nhập tên Project cũng như chọn nước phát hành.
5. Nhấn “Create project” để tiếp tục tạo project.

Trong đề tài này thì dịch vụ Realtime Database của Firebase được sử dụng để tạo cơ sở dữ liệu. Realtime Database là một cơ sở dữ liệu thời gian thực. Việc sử dụng thuật ngữ này phổ biến nhất là đề cập đến một hệ thống cơ sở dữ liệu sử dụng công nghệ phát trực tuyến để xử lý khối lượng công việc có trạng thái thay đổi liên tục.

Hình 2. 14 Cấu trúc cơ sở dữ liệu trên Firebase

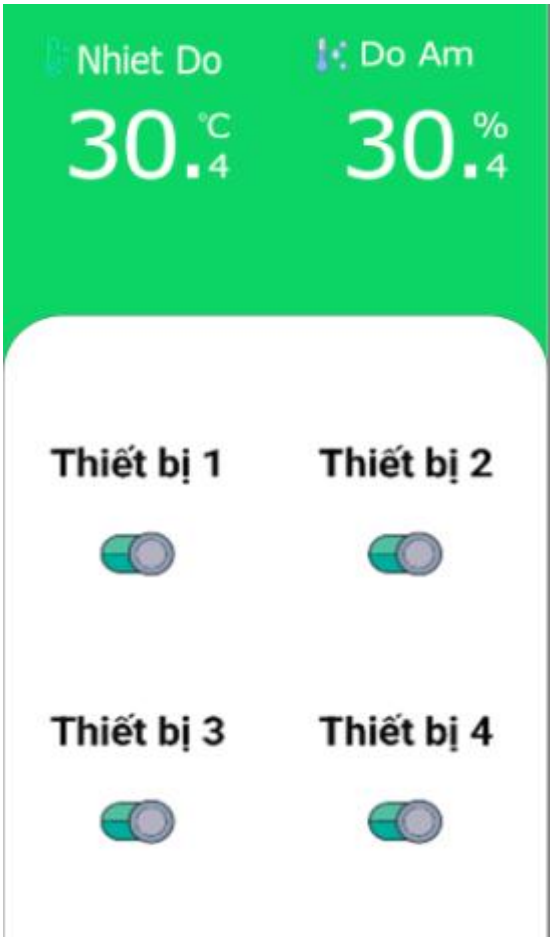
CHƯƠNG 3. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ

3.1. Kết quả mô hình hệ thống

-Mô hình phần cứng

Hình 3. 1 Mô hình hoàn thiện

-Giao diện giám sát và điều khiển



Hình 3. 2 Giao diện phần mềm

Sau khi thực hiện, chúng em đã hoàn thành hệ thống đáp ứng cơ bản những yêu cầu ban đầu đặt ra, dưới đây là một số nhận xét:

- Ưu điểm

+ Hệ thống hoạt động ổn định qua nhiều lần thử nghiệm.
+ Đồng bộ trạng thái điều khiển từ mô hình hệ thống, phần mềm ứng dụng và cơ sở dữ liệu.
+ Giám sát được nhiệt độ và độ ẩm của môi trường xung quanh.
+ Tốc độ điều khiển bằng tay tương đối nhanh.
+ Giao diện điều khiển trực quan, đẹp mắt.
+ Dễ dàng sử dụng, lắp đặt và bảo dưỡng.

- Nhược điểm

+ Hệ thống phụ thuộc vào tốc độ mạng Wifi, và sự ổn định của Firebase.
+ Chưa tích hợp nhiều tính năng thành một hệ thống IoT hoàn chỉnh.

3.2. Đánh giá thực nghiệm hệ thống

Trong quá trình vận hành hệ thống, chúng em đã ghi nhận lại kết quả được tổng hợp.

Công việc	Số lần thao tác	Số lần thành công	Thời gian đáp ứng	Đánh giá
Điều khiển thiết bị qua ứng dụng	50	48	1 – 2 giây	Đạt
Điều khiển bằng nút nhấn	50	50	1 giây	Đạt
Giám sát cảm biến	Ổn định	Ổn định	2 giây	Đạt
Đánh giá chung				Đạt

Bảng 3. 1 Đánh giá chức năng của thống

Qua những số liệu được thống kê ở bảng trên, chúng em đánh giá hệ thống về cơ bản đã đáp ứng được mục tiêu đặt ra. Hệ thống hoạt động ổn định sau nhiều lần chạy, kiểm tra thử trong nhiều trường hợp. Mô hình nhỏ gọn, thẩm mỹ, nhưng vẫn đảm bảo tính an toàn cao, dễ dàng lắp đặt và sử dụng. Nhưng để đưa hệ thống này áp dụng vào thực tế thì em cần phải hoàn thiện một số phần như sau: tăng tốc độ điều khiển cũng như phản hồi, tối ưu hóa mô hình, thêm một số chức năng như: giám sát nơi điều khiển, cảnh báo chống trộm, báo cháy.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Sau thời gian nghiên cứu, thi công thì đồ án tốt nghiệp của em với đề tài "Thiết kế hệ thống giám sát nhiệt độ , độ ẩm và điều khiển thiết bị điện qua internet dùng ESP32" đã hoàn thiện, đáp ứng được những yêu cầu ban đầu đặt ra.

- Ưu điểm

– Mạch điều khiển nhỏ gọn, hoạt động khá ổn định, thời gian đáp ứng khá nhanh.
– Giao diện điều khiển và giám sát dễ sử dụng, thân thiện người dùng.
– Mô hình hệ thống có độ chính xác, tính an toàn và dễ dàng thao tác với
– người dùng.
– Phù hợp cho các hệ thống điện trong phòng học, hộ gia đình.

Nhìn chung, mô hình đã hoạt động tương đối ổn định, có thể làm việc liên tục, đạt 100% yêu cầu đề ra ban đầu. Người dùng thao tác một cách đơn giản, dễ sử dụng.

- Khuyết điểm

Tuy nhiên, do sự hạn chế về kiến thức và thời gian thực hiện, nguồn tài liệu tham khảo chủ yếu thông qua internet nên đề tài không tránh khỏi sai sót và còn một số hạn chế:

– Hạn chế lớn nhất là tác động điều khiển còn chậm do giao thức hoạt động
– chính sử dụng dịch vụ Cloud.
– Hệ thống phụ thuộc vào nguồn điện 220VAC, và tốc độ truy cập mạng
– Internet.
– Hoạt động chủ yếu tại môi trường có phủ sóng wifi.
– Hộp mô hình còn mang tính tượng trưng.
– Kích thước sản phẩm còn thô, thiếu tính thẩm mỹ.
– Số lượng thiết bị còn hạn chế
– Giới hạn về thời gian, kiến thức nên hệ thống chưa được tối ưu.

2. Tác động của sản phẩm thiết kế tới môi trường/ kinh tế/ xã hội

Tác động tới môi trường:

+ Tiết kiệm năng lượng: Sản phẩm có thể được thiết kế để tối ưu hóa sử dụng năng lượng, giúp giảm lượng điện tiêu thụ không cần thiết và hỗ trợ trong việc tiết kiệm năng lượng.

+ Quản lý năng lượng hiệu quả: Có thể tận dụng chức năng tự động hóa để quản lý năng lượng hiệu quả hơn, chẳng hạn như tắt các thiết bị khi chúng không còn được sử dụng.

Tác động đến kinh tế:

+ Tiết Kiệm chi phí năng lượng: Việc quản lý năng lượng thông qua các thiết bị này có thể giúp người tiêu dùng tiết kiệm chi phí năng lượng trong thời gian dài.

+ Tạo ra cơ hội kinh doanh mới: Do sự linh hoạt của ESP8266 và khả năng kết nối qua WiFi, nó có thể tạo ra cơ hội kinh doanh mới cho các doanh nghiệp phát triển và bán các sản phẩm thông minh.

Tác động đến xã hội:

+ Thuận tiện và an toàn: Sản phẩm giúp tăng cường sự thuận tiện và an toàn trong cuộc sống hàng ngày của người tiêu dùng, ví dụ như kiểm soát đèn, quạt, và thiết bị gia đình từ xa.

+ Tạo ra cơ hội nghề nghiệp:Công nghiệp Internet of Things (IoT) liên quan đến việc phát triển và duy trì các sản phẩm này có thể tạo ra cơ hội nghề nghiệp mới trong lĩnh vực công nghiệp kỹ thuật điện tử và lập trình.

+ Thách thức về bảo mật: Tuy nhiên, cũng cần chú ý đến thách thức bảo mật khi sử dụng các thiết bị kết nối internet, để ngăn chặn các tình huống xâm nhập và bảo vệ thông tin cá nhân của người dùng.

3. Hướng phát triển

- Tạo giao diện phong phú và đa dạng hơn.
- Giảm độ trễ khi điều khiển.
- Phát triển đa nền tảng (Android, IOS…).
- Nâng cao tính bảo mật của thiết bị

TÀI LIỆU THAM KHẢO

[1] Bồ Quốc Bảo, Trần Quang Việt, Đề cương bài giảng Thiết kế mạch điện tử, Khoa Điện tử, ĐH Công nghiệp HN

[2] Mansaf Alam, Internet of things (IoT) concepts and applications, Springer Nature,2020.

[3] Lưu Văn Đại, Công nghệ Internet of Things, Nhà xuất bản Giáo dục Việt Nam, 2010.

[4] ESP 32

- https://www.espressif.com/en/products/socs/esp32

[5] SHT30

- https://pdf1.alldatasheet.vn/datasheet-pdf/view/897974/ETC2/SHT30.html

PHỤ LỤC

1. Chương trình cho vi điều khiển

```
#include <Arduino.h>
#include <SHT3x.h>
SHT3x Sensor;
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
#include <WiFiManager.h>
#include <ArduinoJson.h>
#include "FirebaseESP32.h"
#define FIREBASE_HOST"doanchuyennganh-
37767defaultrtdb.firebaseio.com/"
#define FIREBASE_AUTH
"9rUcE6LlTwmjlDkybL0e7bvUYzrMwDYVSI8OKKef"
FirebaseData fb;
unsigned long t1 = 0;
float temp, temp_old;
float humi, humi_old;
int stt_update;
int stt_dv[4] = {0, 0, 0, 0};
int output_dv[] = {13, 26, 14, 27};
int input_btn[] = {25, 33, 32, 35};
int stt_input[] = {1, 1, 1, 1};
String path = "/";
FirebaseJson json;
void task1(void *parameter);
void task2(void *parameter);
void task3(void *parameter);
void task4(void *parameter);
void setup() {
  pinMode(13, OUTPUT);
  pinMode(26, OUTPUT);
  pinMode(14, OUTPUT);
  pinMode(27, OUTPUT);
  pinMode(25, INPUT_PULLUP);
```

```
  pinMode(33, INPUT_PULLUP);
  pinMode(32, INPUT_PULLUP);
  pinMode(35, INPUT_PULLUP);
  for (int i = 0; i < 4; i++) {
    digitalWrite(output_dv[i], stt_dv[i]);
  }
  Serial.begin(115200); // Khởi tạo Serial để in ra Serial Monitor
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.print("ESP 32 ");
  Serial.begin(115200);
  // Khởi động WiFiManager
  WiFiManager wifiManager;
  // reset và erase cài đặt trước đó
  // wifiManager.resetSettings();
  // tên thiết bị, để tạo SSID mặc định
  wifiManager.autoConnect("ESP32_MH ");
  Serial.println("Kết nối WiFi thành công!");
  lcd.setCursor(0, 1);
  lcd.print("connect wifi ss!");
  //khởi tạo firebase
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  // lấy dữ liệu trên firebase về
  Firebase.getInt(fb, path + "/Data_device/user1/dv1");
  stt_dv[0] = fb.intData(); //lay gtri tu fb ve esp
  Firebase.getInt(fb, path + "/Data_device/user1/dv2");
  stt_dv[1] = fb.intData(); //lay gtri tu fb ve esp
  Firebase.getInt(fb, path + "/Data_device/user1/dv3");
  stt_dv[2] = fb.intData(); //lay gtri tu fb ve esp
  Firebase.getInt(fb, path + "/Data_device/user1/dv4");
  stt_dv[3] = fb.intData(); //lay gtri tu fb ve esp
  // Tạo các task với các ưu tiên khác nhau
  xTaskCreatePinnedToCore(task1, "Task 1", 10000, NULL, 3, NULL, 0);
  xTaskCreatePinnedToCore(task2, "Task 2", 10000, NULL, 3, NULL, 1);
  xTaskCreatePinnedToCore(task3, "Task 3", 50000, NULL, 2, NULL, 0);
  xTaskCreatePinnedToCore(task4, "Task 4", 10000, NULL, 3, NULL, 1);
}

void loop() {
  // Không cần có code trong loop khi sử dụng FreeRTOS
}

// Task 1
void task1(void *parameter) {
  for (;;) {
```

```
    Serial.println("Task 1 is running");
    get_data_Sensor();
    print_lcd();
    vTaskDelay(1000 / portTICK_PERIOD_MS); // Delay 1s
  }
}

// Task 2
void task2(void *parameter) {
  for (;;) {
    Serial.println("Task 2 is running");
    for (int i = 0; i < 4; i++) {
      digitalWrite(output_dv[i], stt_dv[i]);
    }
    vTaskDelay(1000 / portTICK_PERIOD_MS); // Delay 1s
  }
}

// Task 3
void task3(void *parameter) {
  for (;;) {
    Serial.println("Task 3 is running");
    if (millis() - t1 > 5000) {
      Firebase.setString(fb, path + "/Data/Data_sensor/hummid",
(String)(humi_old));
      Firebase.setString(fb, path + "/Data/Data_sensor/temp", (String)(temp_old))
;
      t1 = millis();
    } if (stt_update == 0) {
      Firebase.getInt(fb, path + "/Data_device/user1/dv1"); //lấy gtri từ Firebase về
gán vào biến x
      stt_dv[0] = fb.intData(); //lay gtri tu fb ve esp
      Firebase.getInt(fb, path + "/Data_device/user1/dv2"); //lấy gtri từ Firebase về
gán vào biến x
      stt_dv[1] = fb.intData(); //lay gtri tu fb ve esp
      Firebase.getInt(fb, path + "/Data_device/user1/dv3"); //lấy gtri từ Firebase về
gán vào biến x
      stt_dv[2] = fb.intData(); //lay gtri tu fb ve esp
      Firebase.getInt(fb, path + "/Data_device/user1/dv4"); //lấy gtri từ Firebase về
gán vào biến x
      stt_dv[3] = fb.intData(); //lay gtri tu fb ve esp
    }
    // gửi dữ liệu button lên Firebase
    if (stt_update == 1) {
      Firebase.setInt(fb, path + "/Data_device/user1/dv1", stt_dv[0]);
      Firebase.setInt(fb, path + "/Data_device/user1/dv2", stt_dv[1]);
      Firebase.setInt(fb, path + "/Data_device/user1/dv3", stt_dv[2]);
      Firebase.setInt(fb, path + "/Data_device/user1/dv4", stt_dv[3]);
```

```
      stt_update = 0;
    }
    vTaskDelay(3000 / portTICK_PERIOD_MS); // Delay 3s
  }
}

// Task 4
void task4(void *parameter) {
  for (;;) {
    Serial.println("Task 4 is running");
    read_button();
    vTaskDelay(500 / portTICK_PERIOD_MS); // Delay 4s
  }
}
void print_lcd() {
  //Hiển thị lên LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp  : ");
  lcd.print(temp_old);
  lcd.print((char)223);
  lcd.print("C ");
  lcd.setCursor(0, 1);
  lcd.print("Humid : ");
  lcd.print(humi_old);
  lcd.print("%H   ");
}
void get_data_Sensor()
{ Sensor.UpdateData();
  temp = Sensor.GetTemperature();
  temp = round(temp * 10) / 10;
  humi = Sensor.GetRelHumidity();
  humi = round(humi * 10) / 10;
  if(temp != 0) temp_old = temp;
  if (humi != 0) humi_old = humi;
}
void read_button() {
  for (int i = 0; i < 4; i++) {
    if (digitalRead(input_btn[i]) == LOW) {
      stt_input[i] = 1;
    }
    else {
      if (stt_input[i] == 1) {
        stt_dv[i] = !digitalRead(output_dv[i]);
        digitalWrite(output_dv[i], stt_dv[i]);
        stt_input[i] = 0;
        stt_update = 1;
      }
    }
```

```
  }
}
```

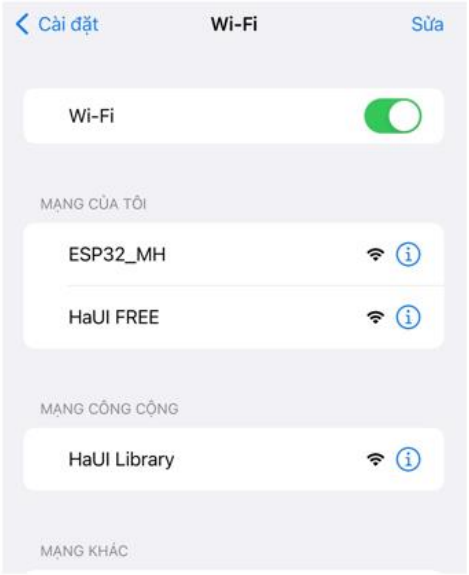
2. Hướng dẫn sử dụng

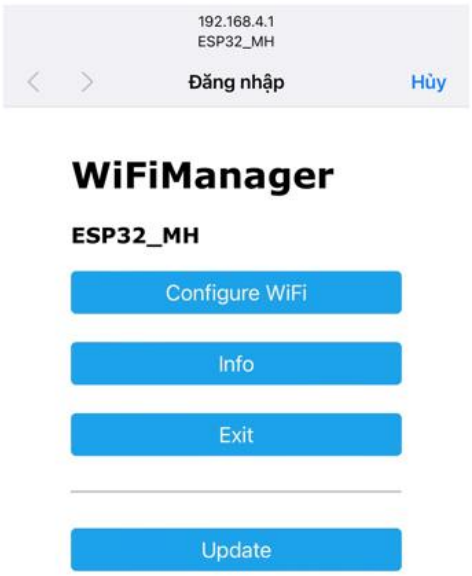
– Bước 1: Tiến hành cấp nguồn cho toàn bộ hệ thống bao gồm nguồn 220V cho ngõ ra mạch công suất, và nguồn 12V từ Adapter cho hệ thống để tạo nguồn DC 5V nuôi mạch điều khiển.

– Bước 2: Chờ đến khi Esp32 kết nối Wifi, mở ứng phần mềm giám sát và điều khiển trên điện thoại . Khi phần mềm đã được kết nối, tiến hành điều khiển, giám sát trực tiếp các thiết bị sử dụng trên giao diện.

Chờ để kết nối wifi cho phần cứng.

Thao tác kết nối Wifi cho hệ thống

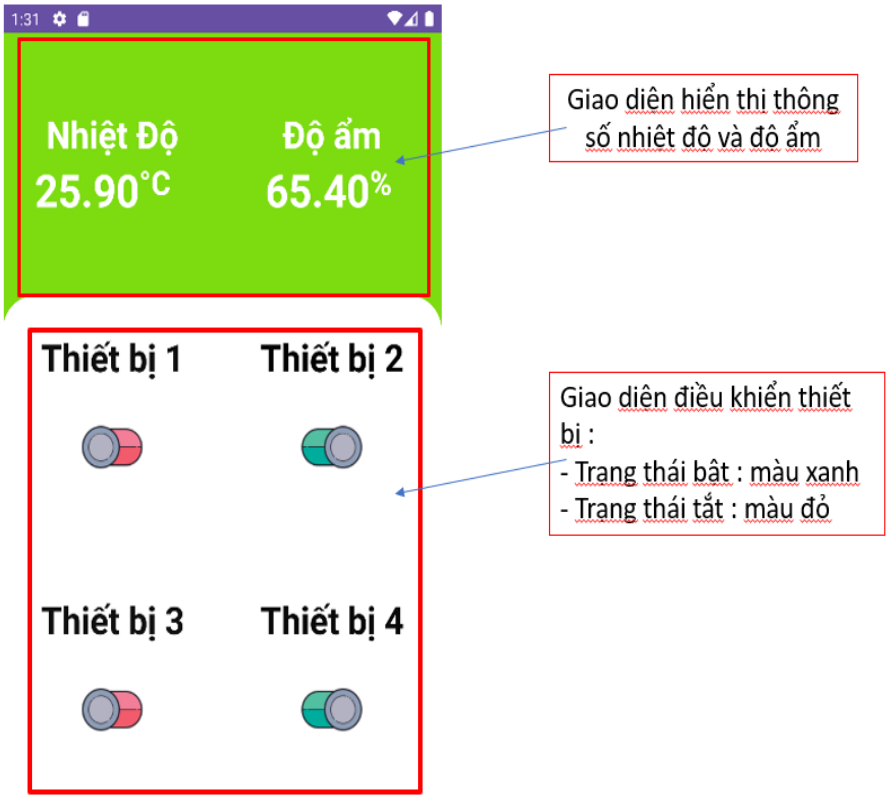




Sử dụng điện thoại thông minh để kết nối tới Wifi có tên **ESP32_MH.** Chọn configue Wifi để kết nối mạng cho thiết bị.

Sau khi kết nối Wifi thành công trên màn hình LCD sẽ hiển thị dòng chữ connect success ! . Sau đó hệ thống sẽ bắt đầu hoạt động

Giao diện điều khiển



3. **Quy trình thao tác**

Cấp nguồn => Kiểm tra kết nối wifi => Điềukhiển thiết bị

Dữ liệu về nhiệt độ, độ ẩm sẽ liên tục được hiển thị trên LCD và gửi lên Firebase. Ta có thế theo dõi thông số nhiệt độ, độ ấm trực tiếp trên mô hình thiết bị hoặc theo dõi gián tiếp qua ứng dụng di động.Và có thể điều khiển thiết bị điện thông qua 2 phương thức:

- **Điều khiển trực tiếp bằng điện thoại**

Người sử dụng mở ứng dụng Android, nhấn nút điều khiển trên màn hình. Trạng thái nút nhấn sẽ được gửi lên lên Firebase và khối xử lý chung tâm sẽ lấy dữ liệu từ đây về . Khối xử lý trung tâm sẽ căn cứ vào dữ liệu trả về để diều khiển ngõ ra công suất , dóng cắt thiết bị .

- **Điều khiển thông qua nút nhấn**

Ta trực tiếp nhấn nút điều khiển trên bộ điều khiển, nút nhấn sẽ gửi xung tín hiệu về khối xử lý trung tâm. Bộ xử lý trung tâm nhận được tín hiệu và tác động đến khối công suất để tiến hành bật tắt thiết bị. Đồng thời Server cũng cập nhật trạng thái của thiết bị vừa được điều khiển.

Bất kể khi sử dụng phương thức điều khiển nào thì trạng thái của thiết bị đều được cập nhật và đồng bộ trên cả phần cứng và phần mềm.