|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  BÁO CÁO  **KỸ THUẬT VI XỬ LÝ**  **Đề tài:**  **THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN THIẾT BỊ VÀ GIÁM SÁT NHIỆT ĐỘ VÀ ĐỘ ẨM CHO NHÀ THÔNG MINH**    **Giảng viên hướng dẫn**: TS. Hàn Huy Dũng  **Nhóm sinh viên :** Ngô Gia Hải 20183729  Đoàn Ngọc Phú 20180153  Vũ Ngọc Tú 20182852  Đỗ Văn Đại 20181865  Lê Hoàng Lân 20183940  ***Hà Nội, tháng 6 năm 2021*** |

**MỤC LỤC**

[DANH MỤC HÌNH VẼ iii](#_Toc78526335)

[DANH MỤC BẢNG BIỂU iv](#_Toc78526336)

[LỜI NÓI ĐẦU v](#_Toc78526337)

[CHƯƠNG 1. GIỚI THIỆU 1](#_Toc78526338)

[1.1 Đặt vấn đề 1](#_Toc78526339)

[1.2 Mục tiêu nghiên cứu 2](#_Toc78526340)

[1.3 Phương pháp nghiên cứu 2](#_Toc78526341)

[1.4 Phạm vi nghiên cứu 2](#_Toc78526342)

[1.5 Lập kế hoạch và phân công nhiệm vụ 2](#_Toc78526343)

[1.5.1 Phân tích nhân lực 2](#_Toc78526344)

[1.5.2 Lập kế hoạch và phân chia nhiệm vụ 3](#_Toc78526345)

[CHƯƠNG 2. YÊU CẦU HỆ THỐNG 5](#_Toc78526346)

[2.1 Mô tả hoạt động của hệ thống 5](#_Toc78526347)

[2.2 Yêu cầu chức năng 5](#_Toc78526348)

[2.3 Yêu cầu phi chức năng 6](#_Toc78526349)

[CHƯƠNG 3. THIẾT KẾ HỆ THỐNG 7](#_Toc78526350)

[3.1 Sơ đồ khối hệ thống 7](#_Toc78526351)

[3.2 Chi tiết từng khối hệ thống 8](#_Toc78526352)

[3.2.1 Khối cảm biến 8](#_Toc78526353)

[3.2.2 Khối điều khiển 9](#_Toc78526354)

[3.2.3 Khối đóng ngắt tải 11](#_Toc78526355)

[3.2.4 Khối nguồn 12](#_Toc78526356)

[3.2.5 Khối hiển thị 13](#_Toc78526357)

[3.3 Sơ đồ nguyên lý 15](#_Toc78526358)

[3.4 Thiết kế chương trình 16](#_Toc78526359)

[3.4.1 Khai báo thư viện và các hằng, các biến 16](#_Toc78526360)

[3.4.2 Hàm khởi tạo setup 18](#_Toc78526361)

[3.4.3 Vòng lặp loop 19](#_Toc78526362)

[3.4.4 Các hàm đọc, gửi dữ liệu và điều khiển 19](#_Toc78526363)

[3.4.5 Hàm điều khiển bật tắt thiết bị bằng phím bấm trên App Blynk 19](#_Toc78526364)

[3.4.6 Hàm điều khiển tự động bật tắt thiết bị 20](#_Toc78526365)

[3.5 Tạo giao diện App Blynk 21](#_Toc78526366)

[3.6 Thiết kế mạch in 23](#_Toc78526367)

[3.6.1 Danh sách linh kiện sử dụng 23](#_Toc78526368)

[3.6.2 Mạch in 25](#_Toc78526369)

[CHƯƠNG 4. KẾT QUẢ MÔ PHỎNG 27](#_Toc78526370)

[4.1 Thực hiện mô phỏng 27](#_Toc78526371)

[4.2 Kết quả mô phỏng 29](#_Toc78526372)

[CHƯƠNG 5. KẾT QUẢ THỰC NGHIỆM 30](#_Toc78526373)

[5.1 Test một số chức năng 30](#_Toc78526374)

[5.1.1 Test chức năng đọc giá trị nhiệt độ, độ ẩm và gửi lên Blynk 30](#_Toc78526375)

[5.1.2 Test chức đăng bật tắt thiết bị thiết bị trên App 31](#_Toc78526376)

[5.1.3 Test chế độ điều khiển tự động 32](#_Toc78526377)

[5.1.4 Test mạch kích relay trên Proteus 33](#_Toc78526378)

[5.2 Chạy thử hệ thống 34](#_Toc78526379)

[5.2.1 Cắm mạch trên Boardtest 34](#_Toc78526380)

[5.2.2 Kết quả 35](#_Toc78526381)

[5.2.3 Nhận xét 36](#_Toc78526382)

[KẾT LUẬN 37](#_Toc78526383)

[TÀI LIỆU THAM KHẢO 38](#_Toc78526384)

[PHỤ LỤC 39](#_Toc78526385)

[Phụ lục 1. Sơ đồ chân Module ESP8266 V1 và giải thích 39](#_Toc78526386)

[Phụ lục 2. Mã nguồn chương trình 43](#_Toc78526387)

# DANH MỤC HÌNH VẼ

[Hình 3.1 Sơ đồ khối của hệ thống 7](#_Toc78526388)

[Hình 3.2 Hai kiểu cảm biến DHT11 trên thị trường 9](#_Toc78526389)

[Hình 3.3 ESP32 Dev kit V1 10](#_Toc78526390)

[Hình 3.4 NodeMCU ESP8266 10](#_Toc78526391)

[Hình 3.5 Relay Songle SLA 6P 11](#_Toc78526392)

[Hình 3.6 Relay Songle SRD 5P 11](#_Toc78526393)

[Hình 3.7 Sơ đồ khối Relay 12](#_Toc78526394)

[Hình 3.8 Adapter nguồn 5V-1A 13](#_Toc78526395)

[Hình 3.9 Các thành phần của Blynk 14](#_Toc78526396)

[Hình 3.10 Sơ đồ nguyên lý mạch 15](#_Toc78526397)

[Hình 3.11 Giao diện người dùng trên App Blynk 23](#_Toc78526398)

[Hình 3.12 Mạch in dạng 2D 25](#_Toc78526399)

[Hình 3.13 Mạch in dạng 3D 26](#_Toc78526400)

[Hình 4.1 Sơ đồ nguyên lý mạch mô phỏng hệ thống 27](#_Toc78526401)

[Hình 4.2 App Blynk trong BlueStack 5.0 28](#_Toc78526402)

[Hình 4.3 Kết nối cổng COM1 với Blynk server 28](#_Toc78526403)

[Hình 4.4 Chạy mô phỏng hệ thống 29](#_Toc78526404)

[Hình 5.1 Dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT11 30](#_Toc78526405)

[Hình 5.2 Hiển thị nhiệt độ và độ ẩm trên App Blynk 31](#_Toc78526406)

[Hình 5.3 Bật tắt thiết bị trên App Blynk 31](#_Toc78526407)

[Hình 5.4 Chế độ điều khiển tự động 32](#_Toc78526408)

[Hình 5.5 Mô phỏng mạch kích relay trên Proteus 33](#_Toc78526409)

[Hình 5.6 Cắm mạch chạy thử trên Boardtest 34](#_Toc78526410)

[Hình 5.7 Nạp chương trình cho ESP8266 35](#_Toc78526411)

[Hình 5.8 Giao diện App Blynk hoàn chỉnh 36](#_Toc78526412)

[Hình 0.1 Sơ đồ chân ESP8266 V1 39](#_Toc78526413)

# DANH MỤC BẢNG BIỂU

[Bảng 1.1 Bảng phân tích nhân lực 2](#_Toc75952939)

[Bảng 1.2 Phân chia công việc theo thành viên 3](#_Toc75952940)

[Bảng 2.1 Các thông số yêu cầu chế độ điều khiển tự động 6](#_Toc75952941)

[Bảng 3.1 Bảng so sánh cảm biến DHT11 và DHT22 8](#_Toc75952942)

[Bảng 3.2 Bảng so sánh 2 vi điều khiển ESP32 và ESP8266 10](#_Toc75952943)

[Bảng 3.3 Bảng so sánh 2 loại Relay 11](#_Toc75952944)

[Bảng 3.4 Năng lượng tiêu thụ cho các linh kiện trong mạch 12](#_Toc75952945)

[Bảng 3.5 Danh sách và bảng giá linh kiện 23](#_Toc75952946)

[Bảng 0.1 Bảng số GPIO 36](#_Toc75952947)

# LỜI NÓI ĐẦU

Trong xu thế phát triển hiện nay, với sự bùng nổ của các ngành công nghệ thông tin, điện tử, tự động hóa, … Đã làm cho đời sống của con người ngày càng hoàn thiện. Các thiết bị tự động hóa đã ngày càng xâm lấn vào trong sản xuất và thậm chí là vào cuộc sống sinh hoạt hàng ngày của mỗi con người. Do đó một ngôi nhà thông minh có thể giám sát và điều khiển từ xa đã trở thành hiện thực. Là một sinh viên viện Điện tử viễn thông, bằng những kiến thức đã học và mong muốn thiết kế được một mô hình nhà tự động hóa đáp ứng được nhu cầu sinh hoạt hàng ngày. Nhóm em đã mạnh dạn chọn “Thiết kế mạch điều khiển thiết bị và giám sát nhiệt độ, độ ẩm qua internet” làm đề tài cho bài tập lớn môn học Kỹ thuật vi xử lý [1].

Trong quá trình thực hiện đồ án của mình, dưới sự hướng dẫn của thầy HÀN HUY DŨNG, chúng em đã cố gắng hết sức để hoàn thiện một cách tốt nhất. Nhưng với kiến thức và sự hiểu biết có hạn nên sẽ không tránh khỏi những thiếu sót mong thầy và các bạn đóng góp ý kiến đề tài của nhóm em có thể hoàn thiện hơn.

# GIỚI THIỆU

## Đặt vấn đề

Hiện nay cùng với sự phát triển của xã hội, cuộc sống ngày càng được nâng cao thì việc áp dụng công nghệ khoa học kỹ thuật vào đời sống công việc càng cần thiết hơn. Cùng với sự phát triển của các ngành khoa học kỹ thuật, công nghệ kỹ thuật điện tử mà trong đó đặc biệt là kỹ thuật điều khiển tự động đóng vai trò quan trọng trong mọi lĩnh vực đời sống như khoa học kỹ thuật, quản lý, công nghiêp, nông nghiệp, đời sống, quản lý thông tin...

Trong thời kỳ phát triển hiện nay, điện thoại di động là một thiết bị không thể thiếu trong cuộc sống. Ngoài chức năng nghe và nhận cuộc gọi, nhắn tin, xem phim, chơi game, chụp ảnh… điện thoại còn có khả năng giám sát và điều khiển thiết bị từ khoảng cách xa thông qua kết nối Internet/Wifi [2].

Từ nhu cầu thực tế cần có một thiết bị điều khiển, giám sát nhiệt độ độ ẩm trong gia đình, phòng máy chủ, trung tâm dữ liệu, kho dữ liệu, tủ lạnh vắc xin, tủ lạnh thực phẩm, trong nhà kính… bằng điện thoại di động, thông qua Module Wifi Esp8266. Từ đó, có tính linh động cao, cập nhật các dữ liệu 24/24h nhằm tăng hiệu quả quản lý [2].

Bên cạnh đó cùng với sự xuất hiện của Arduino vào năm 2005 tại Italia đã góp phần không nhỏ cho nền kinh tế tự động hóa. Sự xuất hiện của Arduino đã hỗ trợ cho con người rất nhiều trong việc lập trình và thiết kế, nhất là đối với những người mới bắt đầu tìm tòi về vi điều khiển mà không có quá nhiều kiến thức, hiểu biết sâu sắc về vật lý và điện tử. Phần cứng của thiết bị đã được tích hợp nhiều chức năng cơ bản và mã nguồn mở. Ngôn ngữ lập trình lại vô cùng dễ sử dụng và được chia sẻ miễn phí nên Arduino đang ngày càng phổ biến và được phát triển mạnh mẽ trên toàn thế giới [3].

Xuất phát từ những lý do trên em chọn đề tài: “Thiết kế hệ thống điều khiển thiết bị và giám sát nhiệt độ, độ ẩm qua internet” làm đề tài cho bài tập lớn môn học Kỹ thuật Vi xử lý.

## Mục tiêu nghiên cứu

Thiết kế hệ thống điều khiển thiết bị từ xa và hiển thị nhiệt độ, độ ẩm bằng điện thoại di động thông qua Module Wifi ESP8266.

## Phương pháp nghiên cứu

Trong đề tài này em đã sử dụng các phương pháp nghiên cứu:

Phương pháp nghiên cứu lí thuyết: tìm hiểu khái niệm, cấu tạo,nguyên lý hoạt động, các thông số kĩ thuật, sơ đồ nối chân của DHT11 và Esp8266. Tìm hiểu cách sử dụng App Blynk, phần mềm Proteus, Altium và Arduino IDE.

Phương pháp nghiên cứu thực nghiệm: thiết kế mạch điều khiển từ xa về nhiệt độ và độ ẩm trên điện thoại di động thông qua Module Wifi Esp8266.

## Phạm vi nghiên cứu

Thiết kế hệ thống điều khiển thiết bị từ xa và hiển thị nhiệt độ, độ ẩm bằng điện thoại di động thông qua Module Wifi ESP8266 bằng kiến thức được học trong môn kỹ thuật vi xử lý và kiến thức tìm hiểu trên mạng.

## Lập kế hoạch và phân công nhiệm vụ

### Phân tích nhân lực

Để có thể hoàn thành project hoàn chỉnh nhất, nhóm đã thực hiện phân tích các điểm mạnh, điểm yếu của từng thành viên nhóm làm căn cứ phân chia công việc 1 cách hiệu quả.

Bảng . Bảng phân tích nhân lực

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | Điểm mạnh | Điểm yếu |
| 1 | Ngô Gia Hải | * Điện tử tương tự * Lập trình Arduino * Thiết kế mạch * Layout PCB | * Điện tử số * Hạn chế về thời gian |
| 2 | Đoàn Ngọc Phú | * Lập trình C * Lập trình Arduino * Teamwork tốt | * Layout PCB |
| 3 | Vũ Ngọc Tú | * Điện tử tương tự * Từng làm việc với arduino * Layout PCB | * Điện tử số * Lập trình C++ |
| 4 | Đỗ Văn Đại | * Thiết kế mạch * Đọc tài liệu tiếng Anh tốt * Lập trình Arduino, C++ | * Layout PCB |
| 5 | Lê Hoàng Lân | * Đọc tài liệu tiếng Anh tốt * Lập trình Arduino, C++ * Teamwork tốt | * Layout PCB * Thiết kế mạch |

Căn cứ vào bảng 1.1 phân tích nhân lực ở trên, cùng với việc phân tích các công việc cần phải làm, nhóm sẽ phân chia cụ thể công việc cho từng thành viên.

### Lập kế hoạch và phân chia nhiệm vụ

Bảng . Phân chia công việc theo thành viên

|  |  |  |  |
| --- | --- | --- | --- |
| Nhóm công việc | Công việc cụ thể | Nhân lực | Thời gian |
| Phân tích đề tài | Tìm hiểu thị trường, các sản phẩm đã có | Hải, Phú, Tú, Lân, Đại | 3/5 - 9/5 |
| Định hướng sản phẩm, đưa ra mô hình giải quyết bài toán | Hải, Phú, Tú | 10/5 - 6/5 |
| Phân tích các yêu cầu | Hải, Lân, Đại | 10/5 - 16/5 |
| Thiết kế phần cứng | Đưa ra sơ đồ khối | Hải, Phú, Tú, Lân, Đại | 17/5 - 23/5 |
| Phân tích từng khối và lựa chọn giải pháp, linh kiện | Phú, Tú, Lân, Đại | 17/5 - 23/5 |
| Vẽ mạch nguyên lý và mô phỏng | Hải, Phú, Tú | 24/5 - 29/5 |
| Thiết kế thư viện của linh kiện và mạch in | Hải, Tú | 31/5 - 6/6 |
| Thiết kế phần mềm | Viết chương trình cho hệ thống | Phú, Tú, Lân, Đại | 24/5 - 6/6 |
| Thiết kế giao diện App Blynk | Phú, Tú, Lân, Đại | 6/6 - 13/6 |
| Hoàn thiện sản phẩm | Chạy thử chương trình cả hệ thống | Hải, Phú, Tú | 14/6 - 20/6 |
| Làm mạch in và chạy thử | Hải | 14/6 - 20/6 |
| Đo đạc, kiểm tra đánh giá sản phẩm | Hải, Lân, Đại | 20/6 - 27/6 |

Dựa trên kế hoạch lập trên bảng 1.2, nhóm sẽ họp hàng tuần và trao đổi những công việc cụ thể hơn để quá trình làm việc nhóm được hiệu quả.

# YÊU CẦU HỆ THỐNG

## Mô tả hoạt động của hệ thống

Hệ thống hoạt động dựa trên sự kết hợp của Module Nodemcu Esp8266 và app Blynk trên smartphone. Blynk sẽ lưu dữ liệu. App Blynk ngoài chức năng lưu trữ dữ liệu từ mạch đo gửi lên còn có chức năng hiển thị giao diện điều khiển thiết bị, dữ liệu về nhiệt độ, độ ẩm ra giao diện người dùng.

Khi nhận được tín hiệu nhiệt độ báo về từ cảm biến thì bộ vi xử lý của hệ thống module Nodemcu Wifi ESP8266 sẽ xử lý tín hiệu và sau đó truyền lên phần mềm blynk thông qua internet.

Khi module wifi ESP8266 gửi tín hiệu lên App blynk người dùng có thể truy cập vào hệ thống để điều khiển thiết bị trong gia đình và giám sát nhiệt độ, độ ẩm từ xa có mạng internet.

Tại Blynk, thông qua internet, nhà quản lý có thể:

* Điều khiển các thiết bị điện trong gia đình.
* Giám sát nhiệt độ liên tục tại các thời điểm trong ngày. Có thể xem nhiệt độ tại các điểm đo trong cùng một thời điểm.
* Hiển thị nhiệt độ cao nhất và thời điểm nhiệt độ cao nhất, thuận tiên cho việc theo dõi nhiệt độ vượt ngưỡng.

## Yêu cầu chức năng

Hệ thống có chức năng điều khiển bật tắt các thiết bị điện trong nhà thông qua relay, cần đảm bảo một số yêu cầu như sau:

* Bật tắt các thiết bị điện trong nhà qua giao diện trên app.
* Chỉ cần có kết nối wifi.
* Tại giao diện app, người dùng có thể biết được các thiết bị trong nhà đang bật hay tắt.

Ngoài ra, có thể hiển thị thông số nhiệt độ, độ ẩm trong nhà với các yêu cầu như sau:

* Dữ liệu được đo liên tục, 10s 1 lần.
* Có khả năng lưu trữ dữ liệu đã đo đạc trong quá khứ trên server.

Hệ thống còn có thêm chế độ điều khiển tự động, có khả năng điều khiển các thiết bị điện trong nhà như quạt, bóng đèn, điều hòa, … một cách tự động dựa trên các thông số về độ ẩm, nhiệt độ.

Bảng . Các thông số yêu cầu chế độ điều khiển tự động

|  |  |  |
| --- | --- | --- |
| Thông số | Giá trị | Hành động |
| Nhiệt độ | > 28­o | Bật quạt |
| < 19o | Bật máy sưởi |
| Độ ẩm | > 85% | Bật máy hút ẩm |
| < 40% | Bật máy phun sương |

Trên hình 2.1 thể hiện các thông số và hành động của chế độ điều khiển tự động.

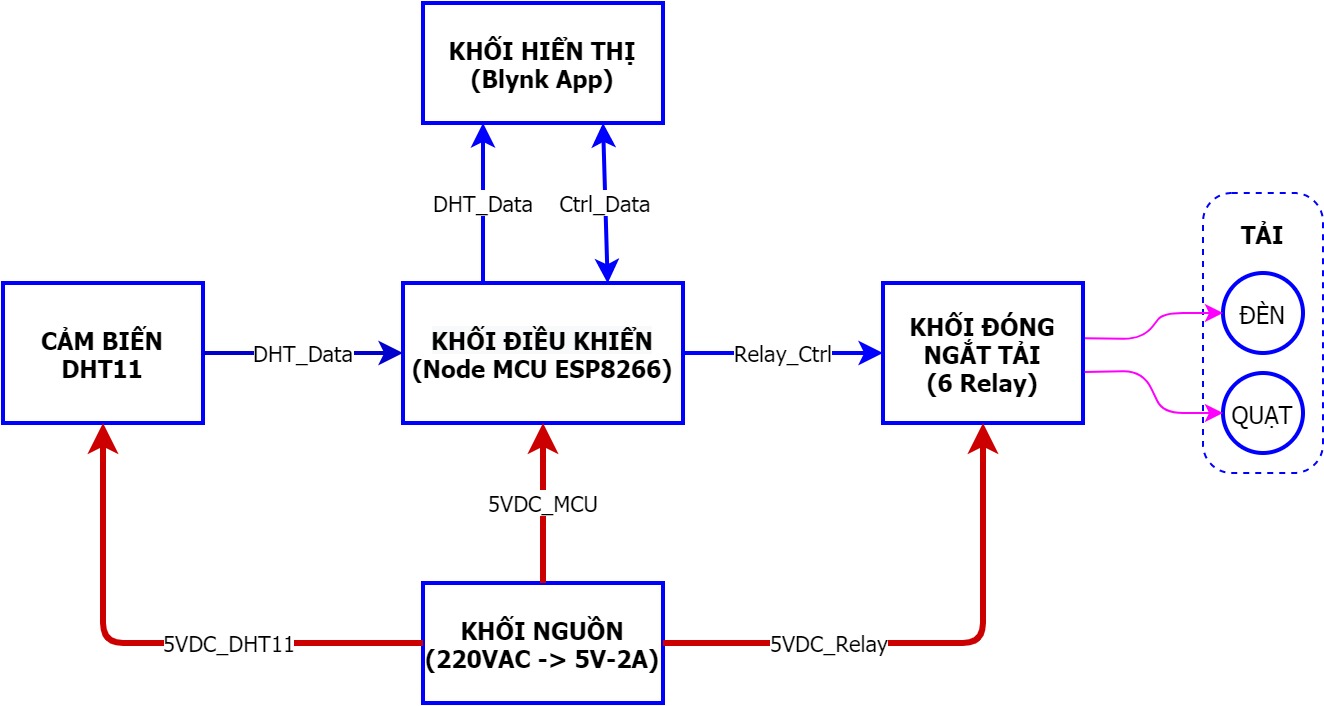
## Yêu cầu phi chức năng

Bên cạnh các yêu cầu chức năng ở trên, sản phẩm cuối cùng cần phải đảm bảo một số yêu cầu như:

* Mạch hoạt động ổn định, chính xác.
* Sản phẩm nhỏ gọn, kích thước mạch không quá 10 x 15cm, dễ dàng lắp đặt.
* Sản phẩm có giá thành phù hợp, dưới 250.000đ
* Thông tin độ ẩm, nhiệt độ được đồng bộ trên app, độ trễ không quá 10s.
* Điều khiển bật tắt trên app có độ trễ không quá 3s
* Giao diện trên app trực quan, dễ sử dụng.

# THIẾT KẾ HỆ THỐNG

## Sơ đồ khối hệ thống



Hình 3.1 Sơ đồ khối của hệ thống

Trên hình 3.1 là sơ đồ khối của hệ thống gồm các thành phần chính:

* Khối cảm biến: thực hiện đo giá trị nhiệt độ và độ ẩm từ bên ngoài môi trường, chuyển thông tin đó thành dạng tín hiệu điện đưa đến khối điều khiển để xử lý.
* Khối điều khiển : Xử lý thông tin, đọc dữ liệu nhiệt độ, độ ẩm từ cảm biến DHT11 gửi lên App Blynk qua Sever, thực hiện các lệnh xử lý thông tin để đưa ra tín hiệu điều khiển tới relay. Khối điều khiển còn có nhiệm vụ đọc và giải mã lệnh nhận được từ giao diện trên App Blynk mà người dùng đưa ra.
* Khối đóng ngắt tải: nhận tín hiệu điều khiển từ vi xử lý và thực hiện đóng ngắt tải. Tải ở đây là các thiết bị điện công suất nhỏ trong gia đình như đèn, quạt, …
* Khối hiển thị: Hiển thị dữ liệu nhiệt độ, độ ẩm được gửi từ khối điều khiển, đồng thời nhận lệnh điều khiển từ người dùng và gửi về khối điều khiển.
* Khối nguồn: Cung cấp nguồn điện cho khối điều khiển, cảm biến và khối đóng ngắt tải hoạt động ổn định với mức điện áp 5V.

## Chi tiết từng khối hệ thống

### Khối cảm biến

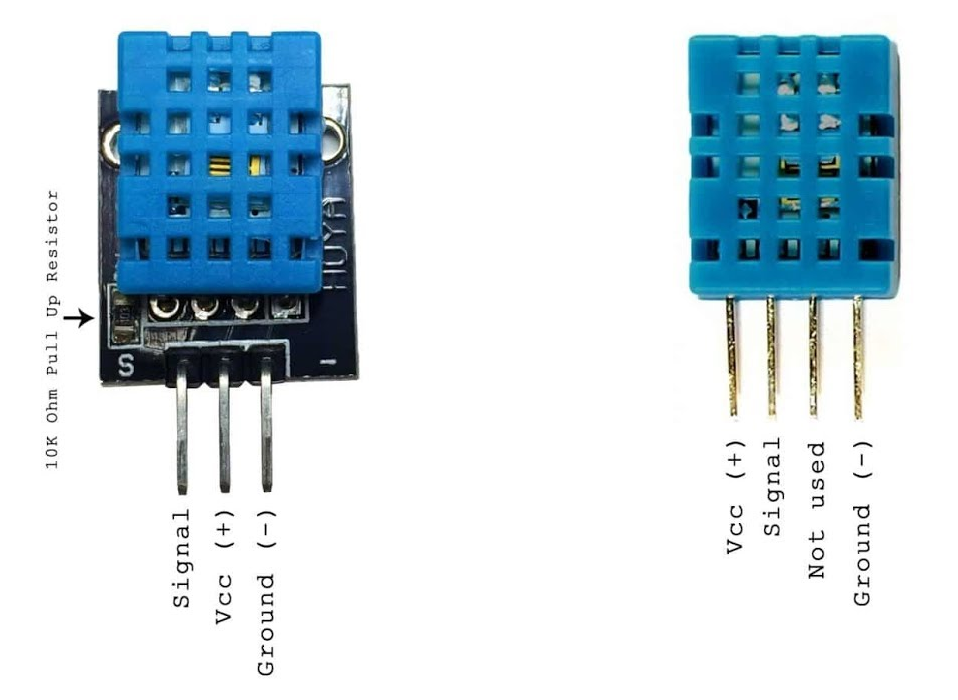
Với yêu cầu về dải đo và sai số đo nhiệt độ và độ ẩm, ta thấy trên thị trường hiện nay có 2 loại cảm biến là DHT11 và DHT22 khá phổ biến, dễ mua, và đáp ứng yêu cầu đề ra. Dưới đây là so sánh về các thông số của 2 loại cảm biến trên:

Bảng . Bảng so sánh cảm biến DHT11 và DHT22

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **DHT11** | **DHT22** |
| Nguồn | 3-5V | 3,3-6V |
| Khoảng độ ẩm | 20 – 80%RH với sai số 5%. | 100%RH với sai số 2-5%. |
| Khoảng nhiệt độ | 0 – 50°C sai số ±2°C. | -40 to 80°C sai số ±0.5°C. |
| Tần số lấy mẫu tối đa | 1Hz (1 giây 1 lần) | 0.5Hz (2 giây 1 lần) |
| Kích thước | 15mm x 12mm x 5.5mm. | 27mm x 59mm x 13.5mm |
| Giá tiền | 25.000đ | 95.000đ |

Từ so sánh 2 loại cảm biến trên bảng 3.1, ta lựa chọn cảm biến DHT11 do có giá thành rẻ hơn nhiều nhưng vẫn đáp ứng đủ yêu cầu đề tài.

DHT11 gửi và nhận dữ liệu với một dây tín hiệu DATA qua giao tiếp 1-wire. Dữ liệu truyền về của DHT11 gồm: 40bit dữ liệu theo thứ tự: 8 bit biểu thị phần nguyên của độ ẩm + 8 bit biểu thị phần thập phân của độ ẩm + 8 bit biểu thị phần nguyên của nhiệt độ + 8 bit biểu thị phần thập phân của nhiệt độ + 8 bit checksum. Phải đảm bảo sao cho ở chế độ chờ (idle) dây DATA có điện áp ở mức cao, nên trong mạch sử dụng DHT11, dây DATA phải được mắc với một trở kéo bên ngoài (thông thường giá trị là 4.7kΩ).



Hình 3.2 Hai kiểu cảm biến DHT11 trên thị trường

Như trên hình 3.2, hiện nay ngoài thị trường có hai loại đóng gói cho DHT11: 3 chân (dạng module cảm biến) và 4 chân. Ta chọn sử dụng loại module cảm biến đã có sẵn điện trở kéo.

### Khối điều khiển

Theo yêu cầu chức năng của sản phẩm, ta có thể xác định các yêu cầu cho khối xử lý trung tâm như sau:

* Có kết nối Wifi để gửi dữ liệu lên App Blynk
* Tối thiểu 7 GPIO (1 INPUT và 6 OUTPUT)
* Tiết kiệm năng lượng (dòng tiêu thụ < 300mA)
* Tích hợp sẵn antenna trên chip

Dựa trên các yêu cầu đặt ra, nhóm đã lựa chọn 2 phương án module Vi điều khiển:

* ESP32
* ESP8266

|  |  |
| --- | --- |
| Hình 3.3 ESP32 Dev kit V1 | Hình 3.4 NodeMCU ESP8266 |

Trên hình 3.3 và hình 3.4 là các vi điều khiển ESP32 và ESP8266 ở dạng module tích hợp sẵn thạch anh, chip flash, anten và chống nhiễu cho mạch nên cho ra sản phẩm có hoạt động ổng định.

Dưới đây là so sánh 2 loại vi điều khiển trên:

Bảng . Bảng so sánh 2 vi điều khiển ESP32 và ESP8266

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **ESP32 Dev kit V1** | **NodeMCU ESP8266 V1** |
| Giá thành | 175000 vnđ | 105000 vnđ |
| Số GPIO khả dụng | 34 | 12 |
| Dòng tiêu thụ | Dòng duy trì 40mA, dòng cho mỗi đầu ra 28mA | Tại công suất phát cao nhất 170mA |
| Băng tần | 2.4GHz | 2.4GHz |
| Tần số thạch anh | 40MHz | 26MHz |

Dựa vào bảng so sánh trên, sau khi tối ưu các yêu cầu cần thiết, nhóm lựa chọn MCU ESP8266 với 3 đặc điểm: giá thành rẻ, đủ số lượng GPIO yêu cầu, dòng tiêu thụ thấp.

### Khối đóng ngắt tải

Sản phẩm có chức năng chính là điều khiển bật tắt các thiết bị điện dân dụng sử dụng điện áp 220VAC. Hiện nay có 2 giải pháp phổ biến cho đóng ngắt điện áp 220VAC là sử dụng Thyristor và Relay. Trong đó đối với yêu cầu đầu ra chỉ có 2 trạng thái là bật và tắt, tần số thay đổi trạng thái thấp thì giải pháp sử dụng relay được ưu tiên sử dụng với ưu điểm giá thành rẻ, giải pháp điều khiển đơn giản.

Dưới đây là bảng so sánh 2 loại relay phổ biến trên thị trường:

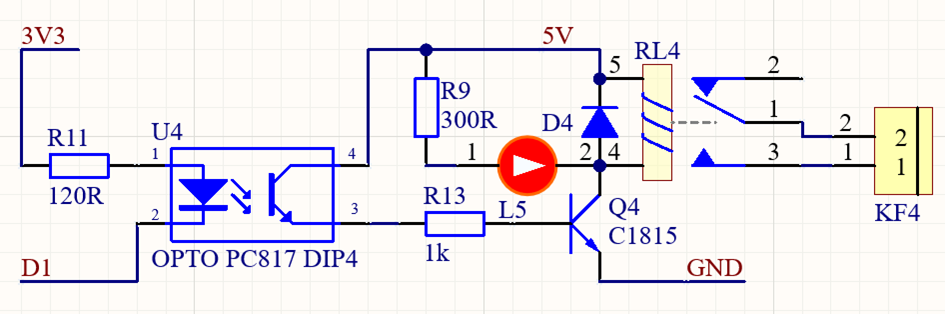
Bảng . Bảng so sánh 2 loại Relay

|  |  |  |
| --- | --- | --- |
| **Tên** | **Relay Songle SLA 6P** | **Relay Songle SRD 5P** |
| Hình ảnh | Hình 3.5 Relay Songle SLA 6P | Hình 3.6 Relay Songle SRD 5P |
| Dòng chịu tải | 30A | 10A |
| Điện áp đóng | 5V | 5V |
| Dòng điện đóng | 170mA | 50mA |
| Kích thước | 32 x 27 x 20 mm | 18x15x16mm |

Sau khi khảo sát về công suất của các thiết bị điện trong nhà, nhóm nhận thấy các thiết bị thường có công suất từ 100-500W tương đương 0.5-2A. Từ thông số khảo sát, nhóm quyết định lựa chọn Relay Songle SRD 5P do kích thước nhỏ gọn, tiết kiệm năng lượng đồng thời vẫn đáp ứng các yêu cầu về dòng chịu tải.

Để thực hiện đóng cắt relay, ta cần đặt 1 điện áp 5V với dòng điện 50mA vào 2 đầu cuộn dây của relay. Trong khi đó dòng điện tối đa của mỗi GPIO của vi điều khiển là 12mA. Để hỗ trợ việc đóng cắt cuộn dây và đảm bảo cho hoạt động của vi điều khiển, sản phẩm sử dụng 1 transistor NPN cho mỗi relay.

Để tránh nhiễu do hiện tượng tự cảm của cuộn dây gây nên, 1 diode được mắc song song với cuộn dây để triệt tiêu điện áp ngược của hiện tượng tự cảm. giữa cực Bazo của BJT và GPIO của vi điều khiển được mắc 1 trở hạn dòng cho vi điều khiển.



Hình 3.7 Sơ đồ khối Relay

Trên hình 3.7 là sơ đồ nguyên lý của 1 khối relay. Khi có tín hiệu mức 0 (LOW) ở ngõ ra GPIO, Opto PC817 sẽ mở, sẽ có điện áp phân cực ở cực B của BJT C1815, do đó sẽ có dòng điện qua cuộn dây, RELAY sẽ kích (hoạt động).

Để tránh nhiễu do hiện tượng tự cảm của cuộn dây gây nên, 1 diode được mắc song song với cuộn dây để triệt tiêu điện áp ngược của hiện tượng tự cảm. giữa cực Bazo của BJT và đầu ra cực E của Opto được mắc 1 trở hạn dòng.

Ở đây ta sử dụng Opto quang PC817 để cách ly điện giữa vi điều khiển và nguồn điện kích cho relay, tránh nhiễu và làm hỏng vi điều khiển.

### Khối nguồn

Đầu vào của khối nguồn là điện áp 220VAC-50Hz, đầu ra là điện áp 5VDC. Nhóm đã thống kê các yêu cầu cần thiết của khối nguồn để lựa chọn phương án khối nguồn sao cho phù hợp:

Bảng . Năng lượng tiêu thụ cho các linh kiện trong mạch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Thiết bị | Số lượng | Điện áp yêu cầu | Dòng tiêu thụ tối đa |
| 1 | Module DHT11 | 1 | 3 - 5.5 V | 20mA |
| 2 | Relay Songle SRD 5P | 6 | 4-5V | 50mA |
| 3 | Module ESP8266 V1 | 1 | 5V | 170mA |
| 4 | Led | 7 | 2-2.2V | 20mA |
| 5 | BJT | 6 | 5V | 2mA |

Từ bảng 3.4 ta tính được tổng dòng tiêu thụ tối đa: 650mA.

Để thiết bị có hoạt động ổn định, ta lựa chọn nguồn có công suất gấp 150% công suất tối đa. Suy ra dòng tải tối đa của nguồn bằng 650mA \* 150% = 975mA ~ 1 A



Hình 3.8 Adapter nguồn 5V-1A

Dựa trên dòng tiêu thụ tối đa, nhóm lựa chọn loại nguồn AC-DC có đầu ra 5V-1A phổ biến trên thị trường, giá thành rẻ như trên hình 3.8

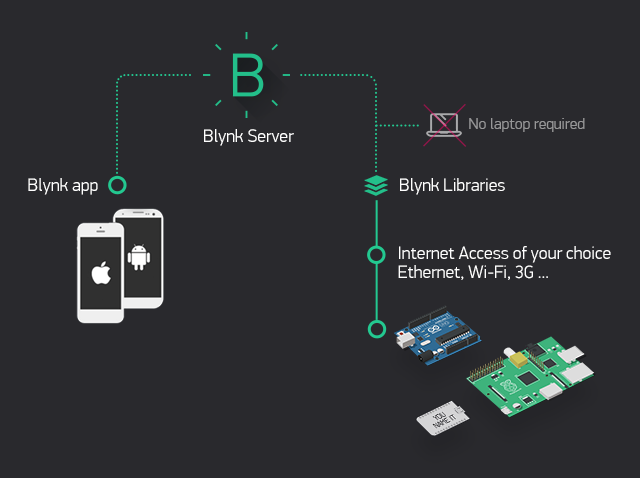
### Khối hiển thị

Blynk là một nền tảng có ứng dụng iOS, Android cho phép điều khiển Arduino, Raspberry Pi, ESP8266. Bạn có thể xây dựng ứng dụng điều khiển bằng cách kéo, thả các Widget [1].

Blynk được thiết kế cho IoT, nó có thể điều khiển phần cứng từ xa, hiển thị dữ liệu cảm biến, lưu trữ dữ liệu và có thể làm một vài điều khác khá thú vị.

Nhóm lựa chọn Blynk vì một số lý do sau:

* **Dễ sử dụng**: Quá đơn giản, chỉ việc vào store, cài đặt, sau đó đăng ký tài khoản và mất không quá 5 phút để làm quen.
* **Đẹp và đầy đủ**: Giao diện của Blynk quá tuyệt vời, sử dụng bằng cách kéo thả, bạn cần nút bấm, kéo thả nút bấm, bạn cần đồ thị, kéo thả đồ thị, bạn cần LCD, kéo thả LCD, tóm lại là bạn cần gì thì kéo thả cái đó.
* **Không phải lập trình android hay ios**: Nếu như không có kiên thức về làm app trên điện thoại thì việc điều khiển thiết bị từ chính smartphone của mình quả là điều vô cùng khó khăn và phức tạp. Nhờ blynk thì chúng ta có thể bỏ qua bước lập trình tạo app. Có thể thử nhanh chóng và ứng dụng được dự án của mình vào thực tế[1].
* **Thử nghiệm nhanh chóng, có thể điều khiển giám sát ở bất kỳ nơi nào có internet.**



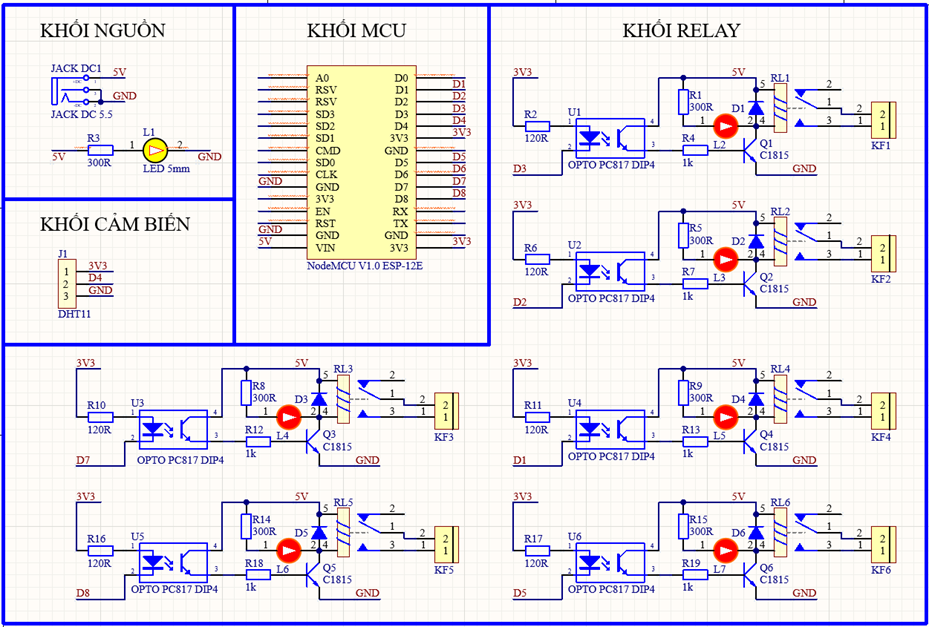
Hình 3.9 Các thành phần của Blynk

Trên hình 3.9 thể hiện các thành phần của Blynk, gồm 3 phần:

* Blynk App: cho phép tạo các giao diện từ Widget có sẵn
* Blynk Server: truyền tải thông tin giữa Smartphone và thiết bị. Blynk Server có thể là 1 đám mây của Blynk hoặc có thể cài đặt trên máy cá nhân. Có thể cài đặt trên Raspberry Pi.
* Blynk Libraries: thư viện cung cấp kết nối phần cứng đến server, xử lý các lệnh đến và đi.

## Sơ đồ nguyên lý

Dưới đây là sơ đồ nguyên lý mạch điều khiển đóng ngắt relay và đọc nhiệt độ, độ ẩm từ cảm biến DHT11.



Hình 3.10 Sơ đồ nguyên lý mạch

Như trên hình 3.9, mạch gồm 4 khối: khối nguồn, cảm biến, khối vi điều khiển MCU và khối relay.

Nguồn cấp cho toàn mạch qua jack 5VDC, có 1 led vàng 5mm để báo nguồn. Nguồn cấp cho cảm biến DHT11 qua chân GND và chân VDD của DHT11. Nguồn cấp cho Node MCU qua chân VIN và chân GND. Chân DATA của DHT11 nối với chân D4 của Node MCU để đọc dữ liệu. Các chân D1, D2, D3, D5, D7, D8 của Node MCU nối với ngõ vào của opto để làm tín hiệu điều khiển kích relay.

## Thiết kế chương trình

### Khai báo thư viện và các hằng, các biến

* Khai báo các thư viện sử dụng trong chương trình:

#include <ESP8266WiFi.h> //Thư viện dùng để kết nối WiFi của ESP8266

#include <BlynkSimpleEsp8266.h> //Thư viện kết nối với Blynk

#include <SimpleTimer.h> // Thư viện của timer

#include <DHT.h> //Thư viện cảm biến DHT

* Khai báo các hằng, biến và định nghĩa :

#define DHTPIN 13 //Nối chân data cảm biến với chân D7

#define DHTTYPE DHT11 //Loại cảm biến là DHT11

#define LAMP1\_PIN 5 //Pinout đèn 1 là chân D1

#define LAMP2\_PIN 4 //Pinout đèn 2 là chân D2

#define LAMP3\_PIN 0 //Pinout đèn 3 là chân D3

#define LAMP4\_PIN 2 //Pinout đèn 4 là chân D4

#define FAN\_PIN 14 //Pinout quạt là chân D5

#define HEATER\_PIN 12 //Pinout máy sưởi 1 là chân D6

int VIRTUAL\_PIN\_1; //VitualPin đèn 1

int VIRTUAL\_PIN\_2; //VitualPin đèn 2

int VIRTUAL\_PIN\_3; //VitualPin đèn 3

int VIRTUAL\_PIN\_4; //VitualPin đèn 4

int VIRTUAL\_PIN\_5; //VitualPin bật hết đèn

int VIRTUAL\_PIN\_6; //VitualPin tắt hết đèn

int VIRTUAL\_PIN\_7; //VitualPin quạt

int VIRTUAL\_PIN\_8; //VitualPin máy sưởi

int VIRTUAL\_PIN\_10; //VitualPin chế độ điều khiển tự động

//Trạng thái các nút nhấn trên AppBlynk

bool b1\_isPushOnApp = false;

bool b2\_isPushOnApp = false;

bool b3\_isPushOnApp = false;

bool b4\_isPushOnApp = false;

bool b5\_isPushOnApp = false;

bool b6\_isPushOnApp = false;

bool b7\_isPushOnApp = false;

bool b8\_isPushOnApp = false;

bool b10\_isPushOnApp = false;

char auth[] = "Auth"; // AuthToken kết nối với Blynk

char ssid[] = "Wifi\_Name"; // Tên WiFi

char pass[] = "Pass"; // Mật khẩu WiFi

int humDht = 0; //Biến lưu giá trị độ ẩm

int tempDht = 0; //Biến lưu giá trị nhiệt độ

DHT dht(DHTPIN, DHTTYPE); //Định nghĩa cảm biến DHT11

SimpleTimer timer; //Định nghĩa timer

* Gọi hàm khi nhấn nút trên AppBlynk:

//Hàm ghi giá trị VitualPin V1 vào biến VIRTUAL\_PIN\_1 với kiểu nguyên

BLYNK\_WRITE(V1)

{

VIRTUAL\_PIN\_1 = param.asInt();

n1\_isPushOnApp = true;

}

//Khi nhấn nút đèn 1 trên App, VitualPin V1 sẽ thay đổi, khi đó hàm trên sẽ được gọi ra

### Hàm khởi tạo setup

void setup()

{

//Khởi tạo các pin OUTPUT

pinMode(LAMP1\_PIN, OUTPUT);

pinMode(LAMP2\_PIN, OUTPUT);

pinMode(LAMP3\_PIN, OUTPUT);

pinMode(LAMP4\_PIN, OUTPUT);

pinMode(FAN\_PIN, OUTPUT);

pinMode(HEATER\_PIN, OUTPUT);

//Set các OUTPUT ban đầu ở mức cao

digitalWrite(LAMP1\_PIN,HIGH);

digitalWrite(LAMP2\_PIN,HIGH);

digitalWrite(LAMP3\_PIN,HIGH);

digitalWrite(LAMP4\_PIN,HIGH);

digitalWrite(FAN\_PIN,HIGH);

digitalWrite(HEATER\_PIN,HIGH);

dht.begin(); // Bắt đầu đọc dữ liệu từ cảm biến DHT11

Blynk.begin(auth, ssid, pass); //Khởi tạo kết nối với Blynk

//Set thời gian timer cho các hàm dọc dữ liệu và điều khiển

timer.setInterval(30000L, getDhtData);

timer.setInterval(30000L, sendDhtData);

timer.setInterval(100L, ctrlDevices);

timer.setInterval(100L, ctrlAllLamp);

timer.setInterval(5000L, autoCtrl);

}

### Vòng lặp loop

void loop()

{

Blynk.run(); //Khởi chạy Blynk

timer.run(); //Khởi chạy timer

}

### Các hàm đọc, gửi dữ liệu và điều khiển

* Hàm đọc và gửi dữ liệu lên Blynk:

void getDhtData(void)

{

tempDht = dht.readTemperature(); //Đọc giá thị nhiệt độ

humDht = dht.readHumidity(); //Đọc giá trị độ ẩm

}

void sendDhtData(void)

{

Blynk.virtualWrite(V11, tempDht); //Nhiệt độ với pin V10

Blynk.virtualWrite(V12, humDht); // Độ ẩm với pin V11

}

### Hàm điều khiển bật tắt thiết bị bằng phím bấm trên App Blynk

if(b1\_isPushOnApp) // Nếu nút đèn 1 trên App được nhấn

{

//Ghi trạng thái VitualPin V1 lên OutputPin đèn 1

digitalWrite(LAMP1\_PIN, !(VIRTUAL\_PIN\_1));

//Xoá biến trạng thái

b1\_isPushOnApp = false;

}

if(b5\_isPushOnApp) // Nếu nút Bật hết trên App được nhấn

{

//Thực hiện bật cả 4 đèn và ghi giá trị mức cao lên các VirtualPin V1, V2, V3, V4

digitalWrite(LAMP1\_PIN, 0);

Blynk.virtualWrite(V1, 1);

digitalWrite(LAMP2\_PIN, 0);

Blynk.virtualWrite(V2, 1);

digitalWrite(LAMP3\_PIN, 0);

Blynk.virtualWrite(V3, 1);

digitalWrite(LAMP4\_PIN, 0);

Blynk.virtualWrite(V4, 1);

b5\_isPushOnApp = false; //Xoá biến trạng thái

}

### Hàm điều khiển tự động bật tắt thiết bị

void autoCtrl(void)

{

if(VIRTUAL\_PIN\_10) //Khi nhấn nút Auto trên App thì sẽ thực hiện điều khiển tự động

{

//Nhiệt độ lớn hơn 30oC thì bật quạt

if(tempDht >= 30)

{

digitalWrite(FAN\_PIN, 0);

Blynk.virtualWrite(V7, 1);

}

else

{

digitalWrite(FAN\_PIN, 1);

Blynk.virtualWrite(V7, 0);

}

//Nhiệt độ nhỏ hơn 16oC thì bật máy sưởi.

if(tempDht <= 16)

{

digitalWrite(HEATER\_PIN, 0);

Blynk.virtualWrite(V8, 1);

}

else

{

digitalWrite(HEATER\_PIN, 1);

Blynk.virtualWrite(V8, 0);

}

}

}

## Tạo giao diện App Blynk

Để tạo giao diện trên App Blynk, đầu tiên ta cần tạo một tài khoản và đăng nhập vào App Blynk.

Sau đó nhấn Create New Project để tạo project mới, đặt tên cho project ở phần Project Name là SMART HOME. Chọn thiết bị ở phần Chose Device là ESP8266. Nhấn Create để tạo project.

Tiếp theo ta thêm các widget ở Widget Box:

* Thêm 6 Button để bật tắt cho 4 đèn, quạt và máy sưởi, đều chọn Mode là SWITCH, Output là VitualPin V1, V2, V3, V4 cho 4 đèn, V7 cho quạt, V8 cho máy sưởi.
* Thêm 2 Button để bật/tắt tất cả đèn, chọn Mode là PUSH, Output là V5 và V6
* Thêm Gauge để hiển thị nhiệt độ, Input là V11, dải hiển thị từ 0 đến 50, Label là oC, Reading rate là 1 sec.
* Thêm Gauge để hiển thị độ ẩm, Input là V12, dải hiển thị từ 0 đến 100, Label là %, Reading rate là 1 sec.
* Thêm Style Button để bật tắt chế độ điều khiển tự động, Output là V10, Mode là SWITCH.



Hình 3.11 Giao diện người dùng trên App Blynk

Sau khi thực hiện thêm các widget, ta được giao diện trực quan như trên hình 3.10

## Thiết kế mạch in

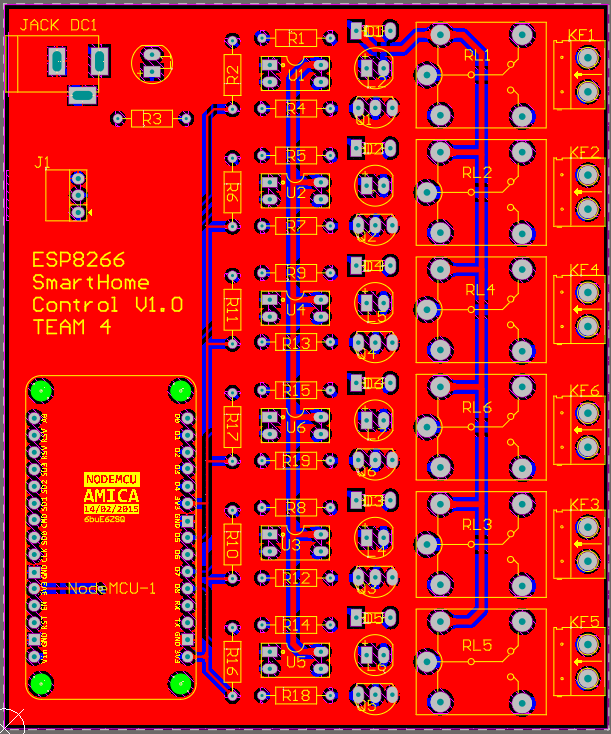
### Danh sách linh kiện sử dụng

Bảng . Danh sách và bảng giá linh kiện

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên linh kiện và thông số** | **Số lượng** | **Giá (VND)** |
| 1 | NodeMCU ESP8266 V1 | 1 | 105.000 |
| 2 | LED đỏ 5mm | 6 | 300 |
| 3 | LED vàng 5mm | 1 | 300 |
| 4 | Relay Songle SRD 5P 5V10A | 6 | 5.500 |
| 5 | Terminal KF301 | 6 | 1.500 |
| 6 | Jack 5VDC | 1 | 1.000 |
| 7 | Opto PC817 DIP4 | 6 | 2.500 |
| 8 | Module DHT11 | 1 | 25.000 |
| 9 | Diode 1N4148 450mA 100V (DIP) | 6 | 500 |
| 10 | Jump cái 3P 2.54mm | 1 | 500 |
| 11 | Điện trở 1/4W 300R | 7 | 40 |
| 12 | Điện trở 1/4W 120R | 6 | 40 |
| 13 | Điện trở 1/4W 1k | 6 | 40 |
| 14 | BJT NPN C1815 TO-92 | 6 | 400 |
| 15 | Adapter nguồn 5V1A | 1 | 25.000 |
| **Tổng (VND)** | 221.760 | | | |

Trên bảng 3.5 là danh sách với giá các linh kiện. Tổng giá thành các linh kiện trong mạch là khoảng 225.000 VND

### Mạch in

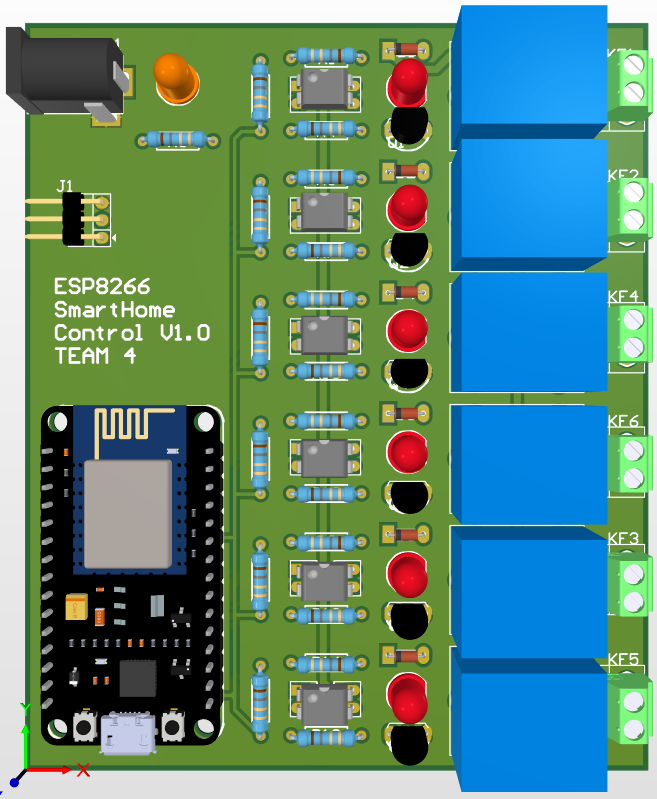


Hình 3.12 Mạch in dạng 2D

Trên hình 3.12 là mạch in 2 lớp ở dạng 2D được thiết kế bằng phần mềm Altium Designer 21, đảm bảo các đường dây được nối đầy đủ, sắp xếp linh kiện hợp lý, thuận lợi cho việc lắp ráp mạch và sản phẩm hoạt dộng ổn định, không gây ra chập cháy.

Một số luật cơ bản cho mạch in trong quá trình đi dây mà nhóm lựa chọn:

* Có 3 cỡ dây là 0.5mm, 1mm và 1.75mm
* Khoảng cách giữa đường dây và lỗ via tối thiểu là 0.5mm
* Độ rộng lỗ via từ 0.4mm đến 0.7mm



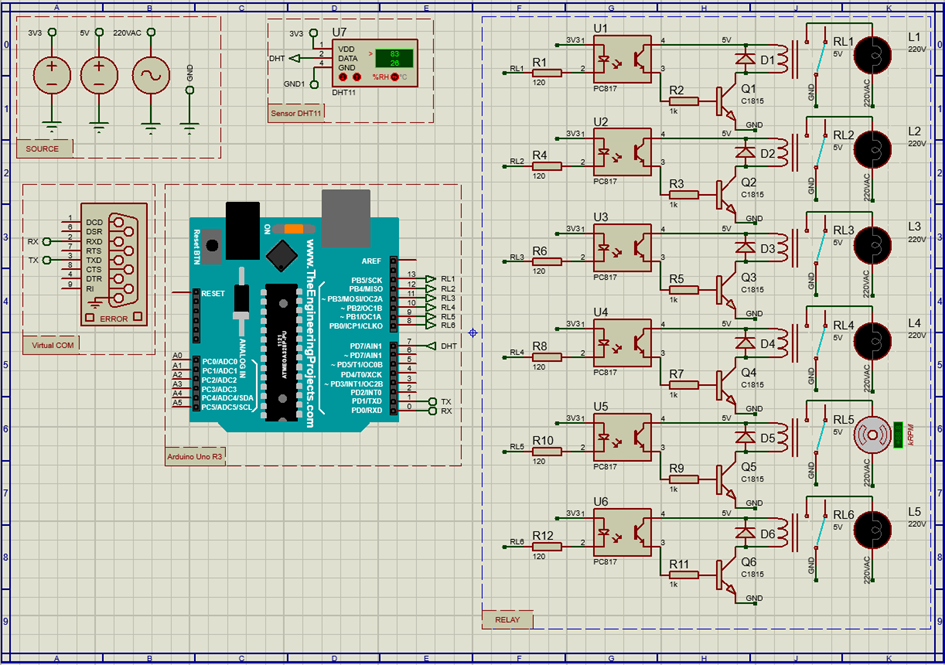
Hình 3.13 Mạch in dạng 3D

Trên hình 3.13 là mạch in dạng 3D, thể hiện trực quan việc bố trí các linh kiện lắp đặt trên mạch. Kích thước mạch là 9cm x 11cm x 1.5cm, đảm bảo yêu cầu mạch nhỏ gọn.

# KẾT QUẢ MÔ PHỎNG

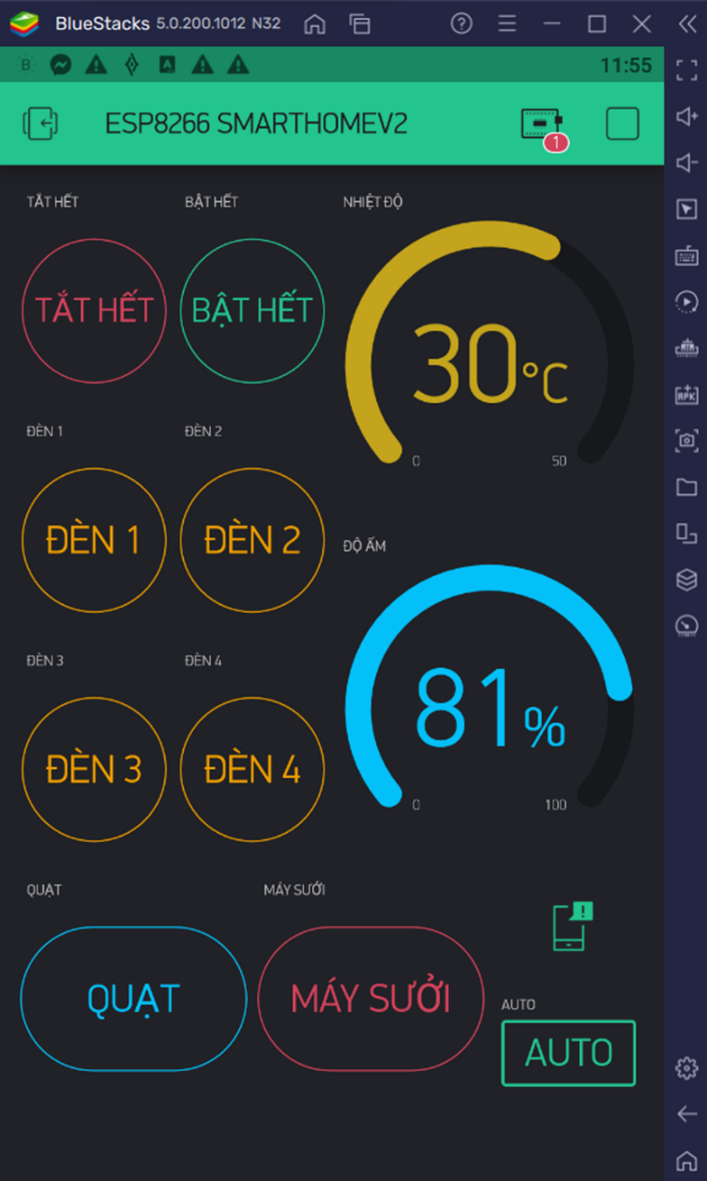
## Thực hiện mô phỏng

Để thực hiện mô phỏng hệ thống, ta sử dụng phần mềm Proteus 8.11. Do không có ESP8266 trên mô phỏng nên ta sử dụng Arduino Uno R3 để thay thế, vẫn đảm bảo được yêu cầu kiểm tra hoạt động của chương trình.



Hình 4.1 Sơ đồ nguyên lý mạch mô phỏng hệ thống

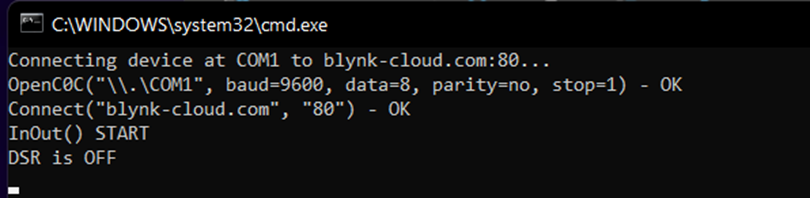
Sơ đồ nguyên lý mạch mô phỏng như trên hình 4.1. Trên sơ đồ nguyên lý bao gồm: khối nguồn; khối cảm biến DHT11 có thể thay đổi giá trị nhiệt, độ ẩm trong quá trình mô phỏng; khối relay với 6 mạch kích relay mức thấp, vi điều khiển là Uno R3 và một cổng COM có chức năng kết nối vi điều khiển với app Blynk qua Blynk sever.



Hình 4.2 App Blynk trong BlueStack 5.0

Để theo dõi kết quả mô phỏng trong app Blynk trên máy tính ta sử dụng phần mềm giả lập Android là BlueStacks 5.0 để cài app Blynk và đăng nhập tài khoản. Giao diện hiển thị như trên hình 4.2.

Cổng COM trong Proteus sẽ kết nối với Uno R3 bằng giao thức UART qua chân TX và RX. Ta sẽ kết nối cổng COM ảo này với Blynk server. Trong thư mục chứa thư viện Blynk của phần mềm Arduino IDE, ta mở file blynk-ser.bat để kết nối cổng COM ảo với Blynk server. Mặc định thì cổng COM1 sẽ được kết nối với Blynk server.



Hình 4.3 Kết nối cổng COM1 với Blynk server

Khi giao diện trên cửa sổ Command Prompt hiển thị như trên hình 4.3 thì đã kết đối thành công. Sau đó ta nhấn nút Run trên Proteus để chạy mô phỏng.

## Kết quả mô phỏng

Graphical user interface

Description automatically generated with medium confidence

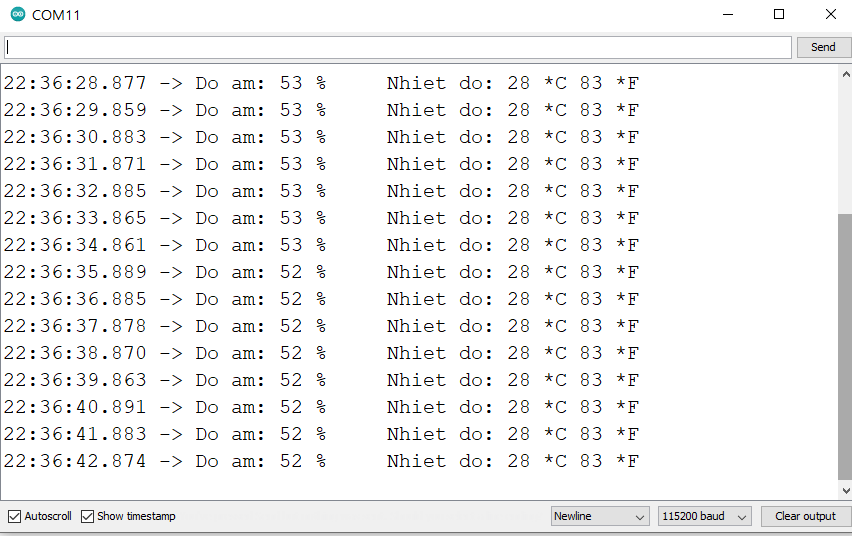
Hình 4.4 Chạy mô phỏng hệ thống

Sau khi chạy mô phỏng thì ta được kết quả như trên hình 4.4. Nhiệt độ và độ ẩm hiển thị trên app Blynk giống với giá trị mà ta đặt trên cảm biến DHT11. Ta nhấn các nút bật tắt thiết bị trên app thì thấy thiết bị bật tắt trên mạch đúng như đã thiết kế. Chức năng điều khiển tự động cũng hoạt động tốt khi ta thay đổi giá trị nhiệt độ và độ ẩm.

# KẾT QUẢ THỰC NGHIỆM

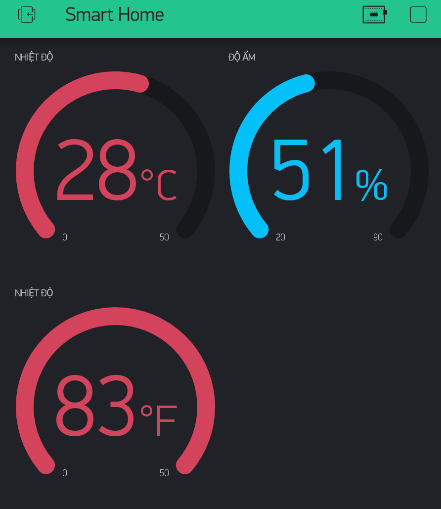
## Test một số chức năng

### Test chức năng đọc giá trị nhiệt độ, độ ẩm và gửi lên Blynk



Hình 5.1 Dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT11

Trên hình 4.1, dữ liệu độ ẩm và nhiệt độ theo đơn vị oC và oF được hiển thị trên Serial Monitor của Arduino IDE, tương đối chính xác với nhiệt độ và độ ẩm của môi trường.



Hình 5.2 Hiển thị nhiệt độ và độ ẩm trên App Blynk

Trên hình 4.2 là kết quả hiển thị dữ liệu nhiệt độ, độ ẩm trên App Blynk gửi từ NodeMcu qua Blynk server. Dữ liệu trùng khớp với dữ liệu gửi từ vi điều khiển.

### Test chức đăng bật tắt thiết bị thiết bị trên App



Hình 5.3 Bật tắt thiết bị trên App Blynk

Trên hình 4.3 là kết quả hiển thị trên app khi test chức năng điều khiển thiết bị qua App. Khi nhấn nút Bật hết, cả 4 đèn trên giao diện cùng sáng, đồng thời cả 4 relay đều kích gần như đồng thời. Độ trễ chỉ khoảng 0.5 đến 1 giây.

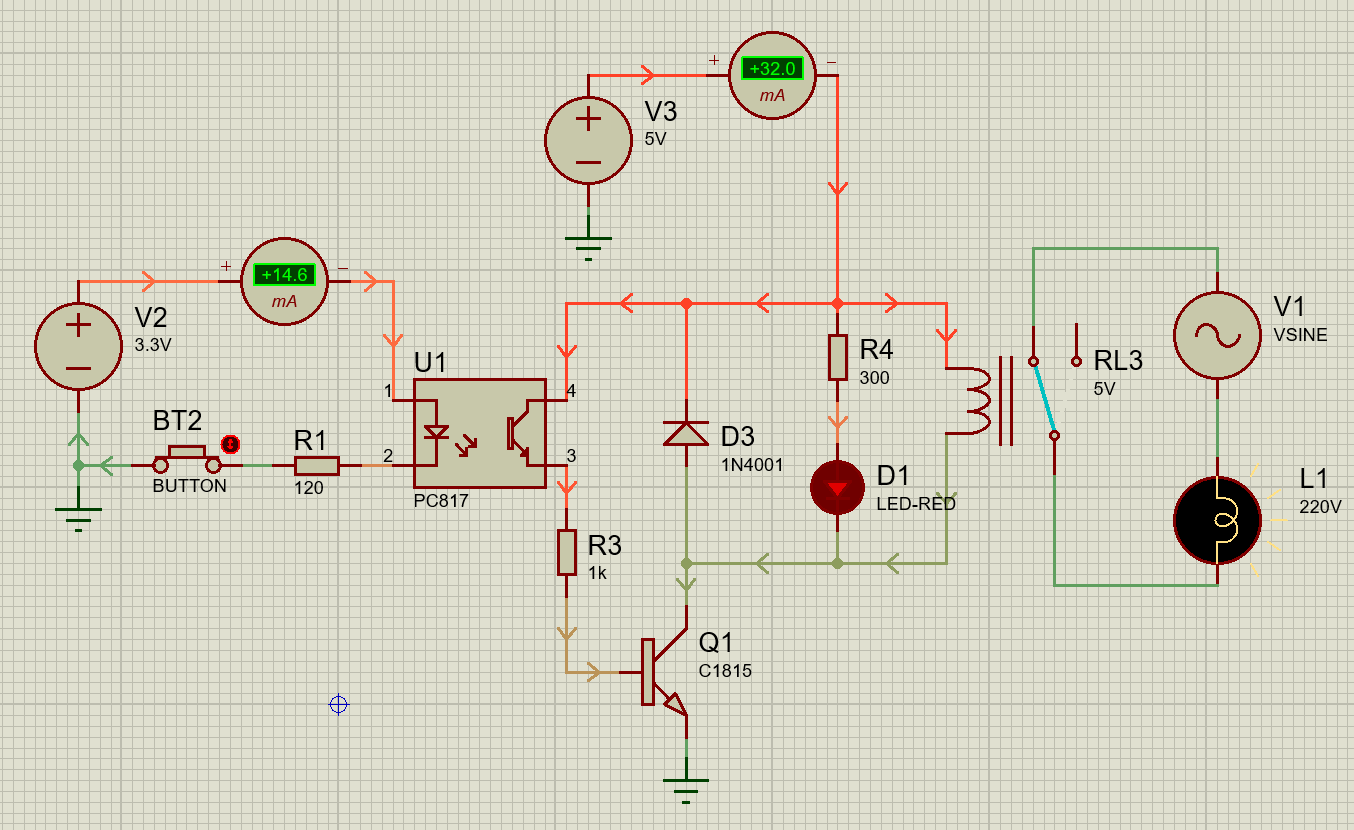
### Test chế độ điều khiển tự động



Hình 5.4 Chế độ điều khiển tự động

Trên hình 4.4 là giao diện chế độ điều khiển tự động. Chế độ điều khiển tự động được bật khi nút AUTO ở trạng thái On. Ở đây phần chương trình ta đặt giá trị nhiệt độ để bật quạt là >= 25oC nên khi nhiệt độ là 27oC thì quạt bật. Khi nhiệt độ xuống dưới 25oC thì quạt tắt. Kết quả chạy thử đã đúng với chương trình.

### Test mạch kích relay trên Proteus

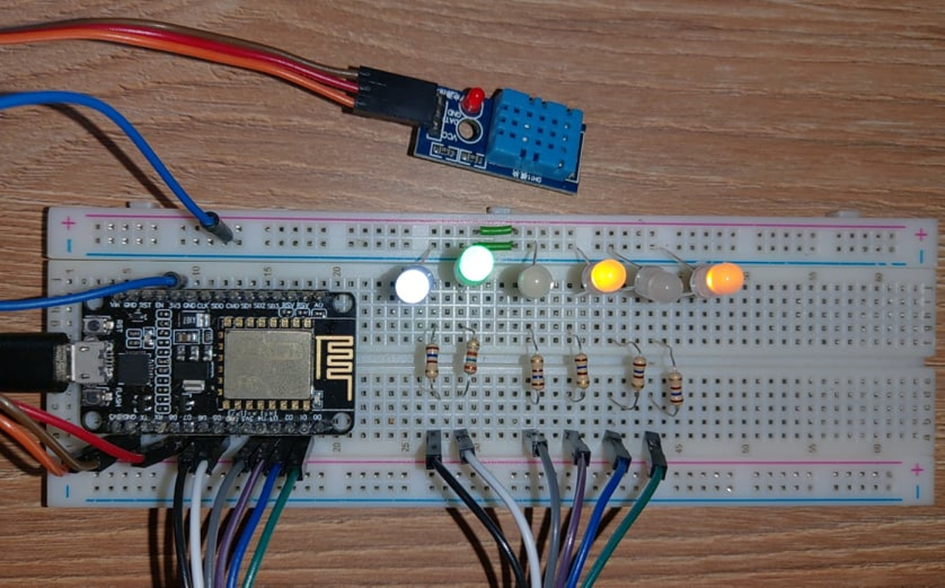


Hình 5.5 Mô phỏng mạch kích relay trên Proteus

Trên hình 4.4 là mạch kích relay ở mức 0 (LOW) với tín hiệu điều khiển từ vi điều khiển. Từ đó ta thấy để kích relay thì cần tốn dòng điện 14.6 mA với mức điện áp 3.3V và dòng 32mA với mức điện áp 5V, tổng dòng điện khoảng 47mA. Dòng điện cho 6 relay là khoảng 282 mA, giống với ước lượng khi chọn khối nguồn, do vậy nguồn 5V 1A đã đảm bảo cấp nguồn đủ cho mạch.

## Chạy thử hệ thống

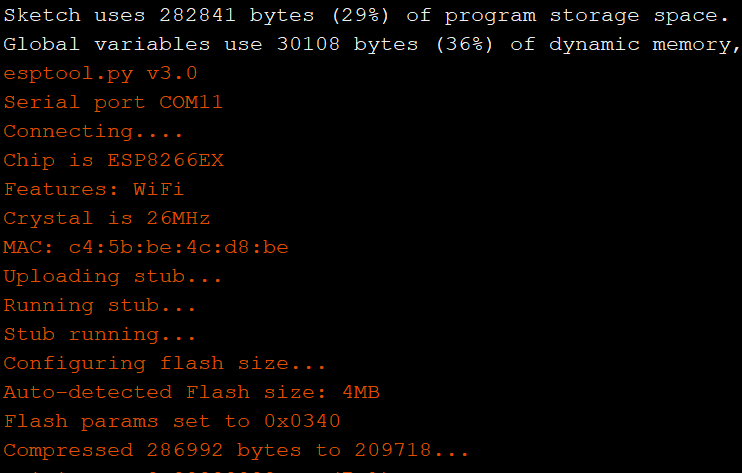
### Cắm mạch trên Boardtest



Hình 5.6 Cắm mạch chạy thử trên Boardtest

Trên hình 5.6 là mạch cắm trên Boardtest. Ta sử dụng 6 đèn led thay thế cho 6 relay. Sử dụng 6 điện trở 150 Ohm để hạn dòng cho led.

### Kết quả



Hình 5.7 Nạp chương trình cho ESP8266

Trên hình 4.6 là quá trình nạp chương trình cho ESP8266, chương trình chiếm 282841 bytes (276kB) – 29% của bộ nhớ Flash (không gian lưu trữ chương trình). Các biến toàn cục của chương trình chiếm 30108 bytes (29kB) – 36% của bộ nhớ động (SRAM).



Hình 5.8 Giao diện App Blynk hoàn chỉnh

Trên hình 4.7 là giao diện hoàn chỉnh App Blynk với các chức năng điều khiển chủ động từ người dùng, điều khiển tự động và hiển thị nhiệt độ, độ ẩm.

### Nhận xét

Qua quá trình chạy thử cho thấy các chức năng đều hoạt động ổn định, độ trễ không quá cao, chỉ khoảng 1 đến 2 giây. Tuy nhiên, khi cho mạch chạy trong thời gian dài thì chip có dấu hiệu nóng lên và điều khiển trên app có thể bị trễ nhiều hơn, điều này có thể khắc phục trong tương lai bằng cách đưa ESP8266 về chế độ Sleep khi không cần sử dụng. Ngoài ra mạch có thể hoạt động không ổn định khi đặt ở nơi có sóng Wifi yếu.

# KẾT LUẬN

Việc thiết kế hệ thống điều khiển thiết bị và giám sát nhiệt độ, độ ẩm từ xa qua internet có thể ứng dụng trong đời sống xã hội và trong công nghiệp. Ngoài ra, module wifi kết hợp với Arduino cũng mở ra rất nhiều hướng ứng dụng khác, phục vụ tốt cho việc học tập và nghiên cứu của sinh viên. Việc xây dựng hệ thống giám sát nhiệt độ, độ ẩm từ xa qua internet này cũng liên quan đến nhiều nền tảng kiến thức từ những kiến thức lý thuyết cho đến những kiến thức thực tiễn [1].

**Hướng phát triển của đề tài**

Trong thời gian tới, nhóm em sẽ tiếp tục nghiên cứu phát triển đề tài theo hướng sau đây:

* Tăng tính chính xác và ổn định hơn nữa.
* Giám sát nhiệt độ độ ẩm và điều khiển thiết bị qua web sever điều khiển bằng cả máy tính và điện thoại.
* Tích hợp nhiều cảm biến hơn nữa phù hợp với nhiều đối tượng người dùng để phục vụ cho cuộc sống và phục vụ trong công nghiệp.
* Thêm điều khiển bằng nút nhấn cảm ứng, sử dụng xử lý ngắt (Interupt)
* Hiển thị nhiệt độ, độ ẩm trên màn hình LCD

# TÀI LIỆU THAM KHẢO

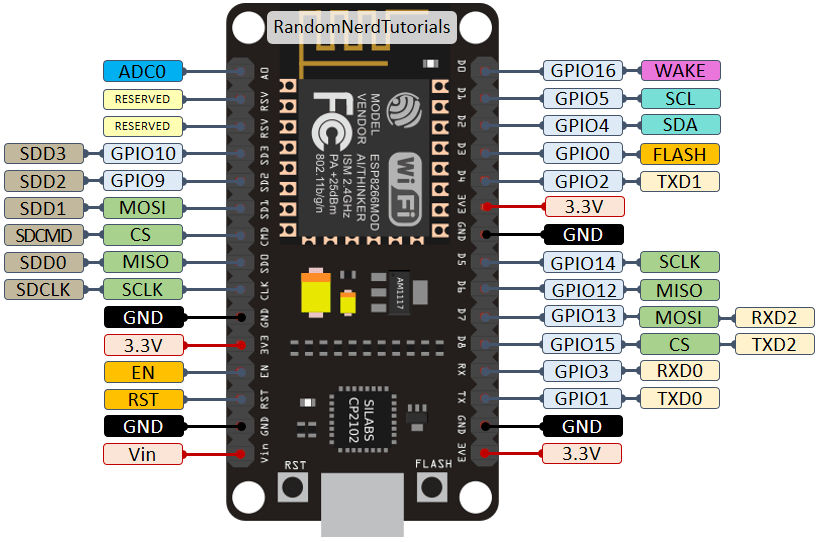
1. Nhóm sinh viên, *Thiết kế hệ thống giám sát nhiệt độ bà độ ẩm bằng ESP8266 NODEMCU.* Đại học Công Nghiệp Hà Nội, Hà Nội, 2018.
2. Nguyễn Thị Kim Anh, *Điều khiển thiết bị qua module ESP8266.* Đại học Huế, Đại học Sư Phạm khoa Vật lý, Huế, 2018
3. Mesquita, J., Guimaraes, D., Pereira, C., Santos, F., & Almeida, L. (2018). *Assessing the ESP8266 WiFi module for the Internet of Things.* IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), 2018.
4. Shakthidhar, S., Srikrishnan, P., Santhosh, S., & Sandhya, M. K. (2019). *Arduino and NodeMcu based Ingenious Household Objects Monitoring and Control Environment.* Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM), 2019.
5. *Esp8266 kết nối internet phần 1, truy cập ngày 12/6/2021,*<http://arduino.vn/bai-viet/1496-esp8266-ket-noi-internet-phan-1-cai-dat-esp8266-lam-mot-socket-client-ket-noi-toi>
6. *Arduino – nền tảng trong việc điều khiển tự động, truy cập ngày 12/6/2021,*

<http://blog.siliconstraits.vn/arduino-nen-tang-moi-trong-viec-dieu-khien-tu-dong/>

1. *Giới thiệu cảm biến nhiệt độ độ ẩm DHT11 Và Code Giao Tiếp,* <http://blog.siliconstraits.vn/arduino-nen-tang-moi-trong-viec-dieu-khien-tu-dong/>

# PHỤ LỤC

## Phụ lục 1. Sơ đồ chân Module ESP8266 V1 và giải thích



Hình 0.1 Sơ đồ chân ESP8266 V1

Một điều quan trọng cần lưu ý về ESP8266 là số GPIO không khớp với nhãn trên màn lụa của bo mạch. Ví dụ, D0 tương ứng với GPIO16 và D1 tương ứng với GPIO5.

Bảng bên dưới cho thấy sự tương ứng giữa các nhãn trên màn lụa và số GPIO cũng như những chân nào là tốt nhất để sử dụng trong mạch và những chân nào cần phải thận trọng.

Bảng . Bảng số GPIO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nhãn | GPIO | Đầu vào | Đầu ra | Ghi chú |
| D0 | GPIO16 | không gián đoạn | không hỗ trợ PWM hoặc I2C | MỨC CAO khi khởi động  Sử dụng để đánh thức khi ngủ sâu |
| D1 | GPIO5 | OK | OK | thường được sử dụng như SCL (I2C) |
| D2 | GPIO4 | OK | OK | thường được sử dụng như SDA (I2C) |
| D3 | GPIO0 | kéo lên | OK | kết nối với nút FLASH, khởi động không thành công nếu kéo MỨC THẤP |
| D4 | GPIO2 | Kéo lên | OK | MỨC CAO khi khởi động  kết nối với đèn LED trên bo mạch, khởi động không thành công nếu kéo MỨC THẤP |
| D5 | GPIO14 | OK | OK | SPI (SCLK) |
| D6 | GPIO12 | OK | OK | SPI (MISO) |
| D7 | GPIO13 | OK | OK | SPI (MOSI) |
| D8 | GPIO15 | kéo đến GND | OK | SPI (CS)  Khởi động không thành công nếu kéo MỨC CAO |
| RX | GPIO3 | OK | Chân RX | MỨC CAO khi khởi động |
| TX | GPIO1 | Chân TX | OK | MỨC CAO khi khởi động  đầu ra gỡ lỗi khi khởi động, khởi động không thành công nếu kéo MỨC THẤP |
| A0 | ADC0 | Đầu vào analog | X |  |

***GPIO được kết nối với Chip Flash***

GPIO6 đến GPIO11 thường được kết nối với chip flash trong bo mạch ESP8266. Vì vậy, những chân này không được khuyến khích sử dụng.

***Chân được sử dụng trong khi khởi động***

ESP8266 có thể bị ngăn không cho khởi động nếu một số chân được kéo MỨC THẤP hoặc MỨC CAO. Danh sách sau đây cho thấy trạng thái của các chân khi khởi động:

* GPIO16: chân ở mức cao khi khởi động
* GPIO0: lỗi khởi động nếu kéo mức thấp
* GPIO2: chân ở mức cao khi khởi động, không khởi động được nếu kéo mức thấp
* GPIO15: lỗi khởi động nếu kéo mức cao
* GPIO3: chân ở mức cao khi khởi động
* GPIO1: chân ở mức cao khi khởi động, không khởi động được nếu kéo mức thấp
* GPIO10: chân ở mức cao khi khởi động
* GPIO9: chân ở mức cao khi khởi động

***Chân mức cao khi khởi động***

Có một số chân xuất ra tín hiệu 3.3V khi ESP8266 khởi động. Điều này sẽ là vấn đề cần phải quan tâm nếu bạn có relay hoặc thiết bị ngoại vi khác được kết nối với các GPIO đó. Các GPIO sau xuất tín hiệu mức cao khi khởi động:

* GPIO16
* GPIO3
* GPIO1
* GPIO10
* GPIO9

Ngoài ra, các GPIO khác, ngoại trừ GPIO5 và GPIO4, có thể xuất ra tín hiệu điện áp thấp khi khởi động, có thể có vấn đề nếu chúng được kết nối với transistor hoặc relay.

***Đầu vào analog***

ESP8266 chỉ hỗ trợ đọc analog trong một GPIO. GPIO đó được gọi là ADC0 và nó thường được đánh dấu trên màn lụa là A0.

Điện áp đầu vào tối đa của chân ADC0 là 0 đến 1V nếu bạn đang sử dụng chip trần ESP8266. Nếu bạn đang sử dụng bo phát triển như bộ ESP8266 12-E NodeMCU, thì dải điện áp đầu vào là 0 đến 3,3V vì bo này có bộ chia điện áp bên trong.

***Đèn LED trên bo mạch***

Hầu hết các bo phát triển ESP8266 đều có đèn LED tích hợp. Đèn LED này thường được kết nối với GPIO2.

Đèn LED hoạt động với logic ngược. Gửi tín hiệu CAO để tắt và tín hiệu THẤP để bật.

***Chân RST***

Khi chân RST được kéo THẤP, ESP8266 sẽ reset. Thao tác này cũng giống như nhấn nút reset trên bo mạch.

***GPIO0***

Khi GPIO0 được kéo THẤP, nó sẽ đặt ESP8266 vào chế độ bộ nạp khởi động. Thao tác này cũng giống như nhấn nút FLASH / BOOT trên bo mạch.

***GPIO16***

GPIO16 có thể sử dụng để đánh thức ESP8266 khỏi chế độ ngủ sâu. Để đánh thức ESP8266 khỏi chế độ ngủ sâu, GPIO16 phải được kết nối với chân RST.

***I2C***

ESP8266 không có chân I2C phần cứng, nhưng nó có thể được triển khai trong phần mềm. Vì vậy, bạn có thể sử dụng bất kỳ GPIO nào làm I2C. Thông thường, các GPIO sau được sử dụng làm chân I2C:

* GPIO5: SCL
* GPIO4: SDA

***SPI***

Các chân được sử dụng làm SPI trong ESP8266 là:

* GPIO12: MISO
* GPIO13: MOSI
* GPIO14: SCLK
* GPIO15: CS

***Các chân PWM***

ESP8266 cho phép phần mềm PWM ở tất cả các chân I / O: GPIO0 đến GPIO16. Tín hiệu PWM trên ESP8266 có độ phân giải 10-bit.

***Chân ngắt***

ESP8266 hỗ trợ chân ngắt trong bất kỳ GPIO nào, ngoại trừ GPIO16.

## Phụ lục 2. Mã nguồn chương trình

|  |
| --- |
| #define BLYNK\_PRINT Serial  #include <ESP8266WiFi.h>  #include <BlynkSimpleEsp8266.h>  #include <SimpleTimer.h>  #include <DHT.h>  #define DHTPIN 13  #define DHTTYPE DHT11  #define LAMP1\_PIN 5  #define LAMP2\_PIN 4  #define LAMP3\_PIN 0  #define LAMP4\_PIN 2  #define FAN\_PIN 14  #define HEATER\_PIN 12  int VIRTUAL\_PIN\_1;  int VIRTUAL\_PIN\_2;  int VIRTUAL\_PIN\_3;  int VIRTUAL\_PIN\_4;  int VIRTUAL\_PIN\_5;  int VIRTUAL\_PIN\_6;  int VIRTUAL\_PIN\_7;  int VIRTUAL\_PIN\_8;  int VIRTUAL\_PIN\_10;  bool n1\_isPushOnApp = false;  bool n2\_isPushOnApp = false;  bool n3\_isPushOnApp = false;  bool n4\_isPushOnApp = false;  bool n5\_isPushOnApp = false;  bool n6\_isPushOnApp = false;  bool n7\_isPushOnApp = false;  bool n8\_isPushOnApp = false;  bool n10\_isPushOnApp = false;  char auth[] = "5rr5mu7i\_v59RIRcpCPvRjKV6tN21EZi"; // Nhập AuthToken  char ssid[] = "Wifi\_Name"; // Nhập tên WiFi  char pass[] = "Pass"; // Nhập password WiFi  int humDht = 0;  int tempDht = 0;  BLYNK\_WRITE(V1)  {  VIRTUAL\_PIN\_1 = param.asInt();  n1\_isPushOnApp = true;  }  BLYNK\_WRITE(V2)  {  VIRTUAL\_PIN\_2 = param.asInt();  n2\_isPushOnApp = true;  }  BLYNK\_WRITE(V3)  {  VIRTUAL\_PIN\_3 = param.asInt();  n3\_isPushOnApp = true;  }  BLYNK\_WRITE(V4)  {  VIRTUAL\_PIN\_4 = param.asInt();  n4\_isPushOnApp = true;  }  BLYNK\_WRITE(V5)  {  VIRTUAL\_PIN\_5 = param.asInt();  n5\_isPushOnApp = true;  }  BLYNK\_WRITE(V6)  {  VIRTUAL\_PIN\_6 = param.asInt();  n6\_isPushOnApp = true;  }  BLYNK\_WRITE(V7)  {  VIRTUAL\_PIN\_7 = param.asInt();  n7\_isPushOnApp = true;  }  BLYNK\_WRITE(V8)  {  VIRTUAL\_PIN\_8 = param.asInt();  n8\_isPushOnApp = true;  }  BLYNK\_WRITE(V10)  {  VIRTUAL\_PIN\_10 = param.asInt();  n10\_isPushOnApp = true;  }  DHT dht(DHTPIN, DHTTYPE);  SimpleTimer timer;  void setup()  {  // Debug console  Serial.begin(115200);  pinMode(LAMP1\_PIN, OUTPUT);  pinMode(LAMP2\_PIN, OUTPUT);  pinMode(LAMP3\_PIN, OUTPUT);  pinMode(LAMP4\_PIN, OUTPUT);  pinMode(FAN\_PIN, OUTPUT);  pinMode(HEATER\_PIN, OUTPUT);  digitalWrite(LAMP1\_PIN,1);  digitalWrite(LAMP2\_PIN,1);  digitalWrite(LAMP3\_PIN,1);  digitalWrite(LAMP4\_PIN,1);  digitalWrite(FAN\_PIN,1);  digitalWrite(HEATER\_PIN,1);  dht.begin(); // Bắt đầu đọc dữ liệu  Blynk.begin(auth, ssid, pass);  timer.setInterval(30000L, getDhtData);  timer.setInterval(30000L, sendDhtData);  timer.setInterval(100L, ctrlDevices);  timer.setInterval(100L, ctrlAllLamp);  timer.setInterval(5000L, autoCtrl);  }  void loop()  {  Blynk.run();  timer.run();  }  void getDhtData(void)  {  tempDht = dht.readTemperature();  humDht = dht.readHumidity();  }  void sendDhtData(void)  {  Blynk.virtualWrite(V11, tempDht); //Nhiệt độ với pin V10  Blynk.virtualWrite(V12, humDht); // Độ ẩm với pin V11  }  void ctrlDevices(void)  {  if(n1\_isPushOnApp) // neu Nut nhan 1 tren App duoc nhan  {  digitalWrite(LAMP1\_PIN, !(VIRTUAL\_PIN\_1)); //ghi trang thai V1 len LED  n1\_isPushOnApp = false; // xoa bien cap nhat  }  if(n2\_isPushOnApp)  {  digitalWrite(LAMP2\_PIN, !(VIRTUAL\_PIN\_2));  n2\_isPushOnApp = false;  }  if(n3\_isPushOnApp)  {  digitalWrite(LAMP3\_PIN, !(VIRTUAL\_PIN\_3));  n3\_isPushOnApp = false;  }  if(n4\_isPushOnApp)  {  digitalWrite(LAMP4\_PIN, !(VIRTUAL\_PIN\_4));  n4\_isPushOnApp = false;  }  if(n7\_isPushOnApp)  {  digitalWrite(FAN\_PIN, !(VIRTUAL\_PIN\_7));  n7\_isPushOnApp = false;  }  if(n8\_isPushOnApp)  {  digitalWrite(HEATER\_PIN, !(VIRTUAL\_PIN\_8));  n8\_isPushOnApp = false;  }  }  void ctrlAllLamp(void)  {  if(n5\_isPushOnApp)  {  digitalWrite(LAMP1\_PIN, 0);  Blynk.virtualWrite(V1, 1);  digitalWrite(LAMP2\_PIN, 0);  Blynk.virtualWrite(V2, 1);  digitalWrite(LAMP3\_PIN, 0);  Blynk.virtualWrite(V3, 1);  digitalWrite(LAMP4\_PIN, 0);  Blynk.virtualWrite(V4, 1);  n5\_isPushOnApp = false;  }  if(n6\_isPushOnApp)  {  digitalWrite(LAMP1\_PIN, 1);  Blynk.virtualWrite(V1, 0);  digitalWrite(LAMP2\_PIN, 1);  Blynk.virtualWrite(V2, 0);  digitalWrite(LAMP3\_PIN, 1);  Blynk.virtualWrite(V3, 0);  digitalWrite(LAMP4\_PIN, 1);  Blynk.virtualWrite(V4, 0);  n6\_isPushOnApp = false;  }  }  void autoCtrl(void)  {  if(VIRTUAL\_PIN\_10)  {  if(tempDht >= 25)  {  digitalWrite(FAN\_PIN, 0);  Blynk.virtualWrite(V7, 1);  }  else  {  digitalWrite(FAN\_PIN, 1);  Blynk.virtualWrite(V7, 0);  }    if(tempDht <= 16)  {  digitalWrite(HEATER\_PIN, 0);  Blynk.virtualWrite(V8, 1);  }  else  {  digitalWrite(HEATER\_PIN, 1);  Blynk.virtualWrite(V8, 0);  }  }  } |