BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

THIẾT KẾ MÔ HÌNH NHÀ THÔNG MINH SỬ DỤNG PHP MYSQL SERVER VÀ ESP32

Cán bộ hướng dẫn	**:**	**TS. Hà Thị Kim Duyên**
Sinh viên thực hiện	**:**	**Bùi Xuân Đặng**
Mã sinh viên	**:**	**2018604733**

Hà Nội – 2022

LỜI CẢM ƠN

Để hoàn thành đề tài đồ án này, lời đầu tiên em xin cảm ơn chân thành đến toàn thể thầy cô trong trường Đại học Công Nghiệp Hà Nội và các thầy cô trong khoa Điện Tử những người đã tận tình hướng dẫn, dạy dỗ và trang bị cho em những kiến thức bổ ích trong những năm vừa qua.

Đặc biệt em xin gửi lời cảm ơn chân thành đến cô **Hà Thị Kim Duyên,** người đã hướng dẫn cho em những kiến thức, kỹ năng cơ bản cần có để hoàn thành đề tài nghiên cứu này.

Sau cùng em xin gửi lời cảm ơn chân thành tới gia đình, bạn bè đã động viên, cổ vũ và đóng góp ý kiến trong quá trình học tập, nghiên cứu cũng như quá trình làm đồ án tốt nghiệp đại học, chuyên ngành điện tử viễn thông.

Tuy nhiên trong quá trình nghiên cứu đề tài, vì kiến thức chuyên ngành còn hạn chế nên em vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong nhận được sự quan tâm, góp ý của các thầy cô giảng viên để đề tài của em được đầy đủ và hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

Hà Nội, Ngày… tháng… năm 2022

Sinh viên thực hiện

Bùi Xuân Đặng

MỤC LỤC

DANH MỤC VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
IoT	Internet Of Things	Internet kết nối vạn vật
PHP	Hypertext Preprocessor	Bộ tiền xử lý siêu văn bản
AJAX	Asynchronous JavaScript And XML.	JavaScript không đồng bộ và XML
JSON	JavaScript Object Notation	Ký hiệu - đối tượng JavaScript
FreeRTOS	Free Real Time Operating System	Hệ điều hành thời gian thực
OTA	Over The Air	Cập nhật Firmware từ xa

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

MỞ ĐẦU

1. Đặt vấn đề

Thuật ngữ IoT (Internet of Things) hẳn không còn xa lạ với chúng ta, nó cũng là một trong những mũi nhọn, là xu thế phát triển hàng đầu của thời đại 4.0. Hiểu một các đơn giản IoT giúp chúng ta tương tác, điều khiển với mọi vật xung quanh thông qua các cảm biến, mạng truyền thông, v.v. Sản phẩm của IoT rất đa dạng và được ứng dụng rộng rãi trong cuộc sống như: Smart home, Hệ thống tưới cây thông minh, Các thiết bị thông minh v.v. Nhìn chung, công nghệ IoT giúp nâng cao và có ý nghĩa vô cùng quan trọng trong cuộc sống, cho sự phát triển của tương lai và là một xu thế tất yếu của thời đại.

Là một sinh viên nghành điện tử và có niềm yêu thích với công nghệ thông tin. Em nhận thấy bản thân có rất nhiều lợi thế khi tìm hiểu và làm việc về lĩnh vực IoT.

Nhận thấy sự quan trọng cũng như tính ứng dụng cao của IoT trong cuộc sống hiện tại, cũng như tương lai. Em quyết định chọn đề tài "Thiết kế mô hình nhà thông minh sử dụng PHP MySQL Server và ESP32". Ý tưởng cốt lõi của hệ thống này là các cảm biến nhiệt độ, độ ẩm, ánh sáng, khí gas sẽ được thu thập và xử lý, sau đó gửi lên server qua wifi và hiển thị giao diện người dùng. Xây dựng một bo mạch mở rộng mà ở đó người dùng có thể tùy chỉnh cấu hình, kết nối với các thiết bị và điều khiển chúng qua wifi.

2. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu

- Ngôn ngữ lập trình PHP, cơ sở dữ liệu MySQL
- Module ESP32 DEVKIT V1 Doit

Phạm vi nghiên cứu

Phần Server:

- Sử dụng ngôn ngữ lập trình PHP để xây dựng Server
- Xây dựng cơ sở dữ liệu với MySQL
- Sử dụng công nghệ AJAX và kiểu dữ liệu JSON

Phần vi điều khiển:

- Tìm hiểu về ESP32 wifi và code bằng platform Adruino.
- Sử dụng API của RTOS để tạo tác vụ và chạy đa luồng
- Cập nhật Firmware từ xa bằng OTA
- Làm việc với các cảm biến như DHT11, BH1750, MQ2.

3. Mục đích của đồ án

Trong đề tài này:

- Hệ thống sử dụng kit ESP32 sẽ thu thập dữ liệu từ các cảm biến sau đó gửi lên Server, sau đó Sever sẽ xử lý dữ liệu và lưu vào Database.
- Server truy xuất dữ liệu từ Database và hiển thị ra giao diện người dùng.
- Server lấy dữ liệu điều khiển lưu vào Database và gửi về ESP32 để xử lí và thực thi.
- Server đa người dùng và cập nhật phần mềm bằng OTA (Over The Air)

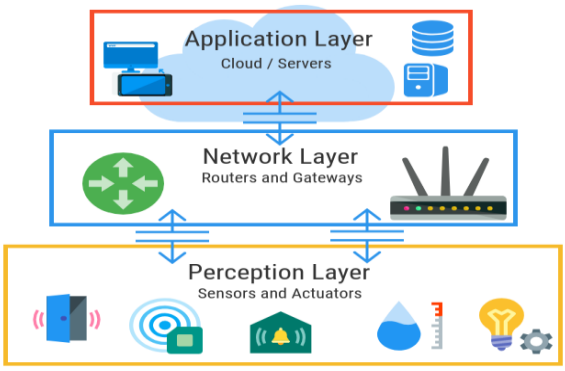
4. Ý nghĩa khoa học

Đề tài này giúp nghiên cứu về hệ thống IoT, lập trình Server và vi điều khiển. Được ứng dụng những công nghệ như AJAX, FreeRTOS, OTA vào đồ án. Giúp đưa IoT đến gần hơn với cuộc sống, được ứng dụng rộng rãi và thân thiện hơn với mọi người.

Sau khi tìm hiểu và đặt vấn đề em đã nhận thấy lợi ích cũng như sự phát triển và hướng đi lâu dài cùng với tính ứng dụng cao của lĩnh vực IoT. Từ đó em đã chọn được đề tài phù hợp, giúp em có thể vận dụng kiến thức đã học cũng như tìm hiểu thêm nhưng kiến thức mới. Đề tài này có tính ứng dụng cao và có thể phát triển trong tương lai.

CHƯƠNG 1. TỔNG QUAN VỀ HỆ THỐNG MẠNG

1.1. Tổng quan về IOT



Hình 1.1: Tổng quan về IOT

Mạng lưới vạn vật kết nối Internet hoặc là mạng lưới thiết bị kết nối Internet viết tắt là IoT là một kịch bản của thế giới, khi mà mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính. IoT đã phát triển từ sự hội tụ của công nghệ không dây, công nghệ vi cơ điện tử và Internet. Nói đơn giản là một tập hợp các thiết bị có khả năng kết nối với nhau, với Internet và với thế giới bên ngoài để thực hiện một công việc nào đó.

Internet of Things – IoT được đưa ra bởi các nhà sáng lập của MIT Auto-ID Center đầu tiên, năm 1999 Kevin Ashton đã đưa ra cụm từ Internet of Things nhằm để chỉ các đối tượng có thể được nhận biết cũng như sự tồn tại của chúng. Thuật ngữ Auto-ID chỉ tới bất kỳ một lớp rộng của các kỹ thuật xác minh sử dụng trong công nghiệp để tự động hóa, giảm các lỗi và tăng hiệu năng. Các kỹ thuật đó bao gồm các mã vạch, thẻ thông minh, cảm biến, nhận dạng tiếng nói, và sinh trắc học. Từ năm 2003 Kỹ thuật Auto-ID trong các hoạt động chính là Radio Frequency Identification – RFID.

Một số ứng dụng của IOT trong cuộc sống chúng ta:

+ Hệ thống tưới cây thông minh.
+ Nhà thông minh.

+ Các thiết bị đeo thông minh.
+ Transport & Logistic.
+ Thành phố, nhà máy thông minh…

Nhìn chung IoT có ảnh hưởng và tính ứng dụng vô cùng lớn trong mọi mặt của đời sống. Giúp nâng cao chất lượng cuộc sống và hiệu quả sản xuất.

1.2. Giao thức kết nối Wifi

1.2.1. Khái niệm Wifi

Wi-Fi là một họ các giao thức mạng không dây, dựa trên các tiêu chuẩn của họ IEEE 802.11, được sử dụng rộng rãi trong cho việc kết nối không dây của thiết bị trong mạng nội bộ và việc kết nối Internet, cho phép các thiết bị điện tử trong phạm vi ngắn chia sẻ dữ liệu thông qua sóng vô tuyến. Ngày nay, WiFi được sử dụng phổ biến trong các hệ thống mạng máy tính trên thế giới, như trong các hộ gia đình, văn phòng làm việc cho việc kết nối các máy tính bàn, laptop, tablet, điện thoại thông minh, máy in... mà không cần đến cáp mạng, cũng như việc kết nối Internet cho các thiết bị này. Các địa điểm công cộng như như sân bay, quán café, thư viện hoặc khách sạn cũng được bố trí WiFi để phục vụ nhu cầu kết nối Internet cho các thiết bị di động, khi các thiết bị đó nằm trong khu vực có sóng của những hệ thống WiFi này.

1.2.2. Nguyên tắc hoạt động của wifi

Truyền thông qua mạng không dây là truyền thông vô tuyến hai chiều. Cụ thể:

Thiết bị adapter không dây (hay bộ chuyển tín hiệu không dây) của máy tính chuyển đổi dữ liệu sang tín hiệu vô tuyến và phát những tín hiệu này đi bằng một ăngten.

Thiết bị router không dây nhận những tín hiệu này và giải mã chúng. Nó gởi thông tin tới Internet thông qua kết nối hữu tuyến Ethernet.

Quy trình này vẫn hoạt động với chiều ngược lại, router nhận thông tin từ Internet, chuyển chúng thành tín hiệu vô tuyến và gửi đến adapter không dây của máy tính.

1.2.3. Một số chuẩn Wifi nổi phổ biến

Bảng 1.1: Các chuẩn Wifi hiện nay

CÁC CHUẨN WIFI HIỆN NAY						
Chuẩn IEEE	802.11a	802.11b	802.11 g	802.11n	802.11ac	802.11ax
Năm phát hành	1999	1999	2003	2009	2014	2019
Tần số	5 GHz	2.4 GHz	2.4 GHz	2.4/5 GHz	5 GHz	6 GHz
Tốc độ tối đa	54 Mbps	11 Mbps	54 Mbps	600 Mbps	1 Gbps	10 Gbps
Phạm vi trong nhà	30 m	30m	38 m	68 m	73 m	Chưa công bố
Phạm vi ngoài trời	30 m	31 m	137 m	250 m	300 m	Chưa công bố

1.3. Tìm hiểu về ngôn ngữ lập trình PHP

1.3.1. Đặc điểm của ngôn ngữ lập trình PHP

PHP: Hypertext Preprocessor, thường được viết tắt thành PHP là một ngôn ngữ lập trình kịch bản hay một loại mã lệnh chủ yếu được dùng để phát triển các ứng dụng viết cho máy chủ, mã nguồn mở, dùng cho mục đích tổng quát. Nó rất thích hợp với web và có thể dễ dàng nhúng vào trang HTML. Do được tối ưu hóa cho các ứng dụng web, tốc độ nhanh, nhỏ gọn, cú pháp giống C và Java, dễ học và thời gian xây dựng sản phẩm tương đối ngắn hơn so với các ngôn ngữ khác nên PHP đã nhanh chóng trở thành một ngôn ngữ lập trình web phổ biến nhất thế giới. [7] [6]

1.3.2. Ưu điểm của ngôn ngữ lập trình PHP

✓ Cung cấp miễn phí hoàn toàn, người dùng chủ động trong việc thực hiện thử qua mọi tính năng được cộng đồng người dùng ngôn ngữ PHP hỗ trợ, đảm bảo việc tìm hiểu và sử dụng hiệu quả được thực hiện tốt nhất.

✓ Sở hữu cấu trúc đơn giản, không gây ra những khó khăn cho người dùng mới khi bắt đầu tham gia vào lĩnh vực lập trình.

✓ Sở hữu thư viện đồ sộ, đa dạng vừa là ưu điểm song cũng là hạn chế của

✓ Ngôn ngữ lập trình PHP khi cung cấp tới người dùng. [7] [12]

✓ Ngôn ngữ lập trình PHP được mở rộng, ngày càng có nhiều framework và extentions được cung cấp hỗ trợ cho quá trình lập trình của người dùng.

1.3.3. Nhược điểm của ngôn ngữ lập trình

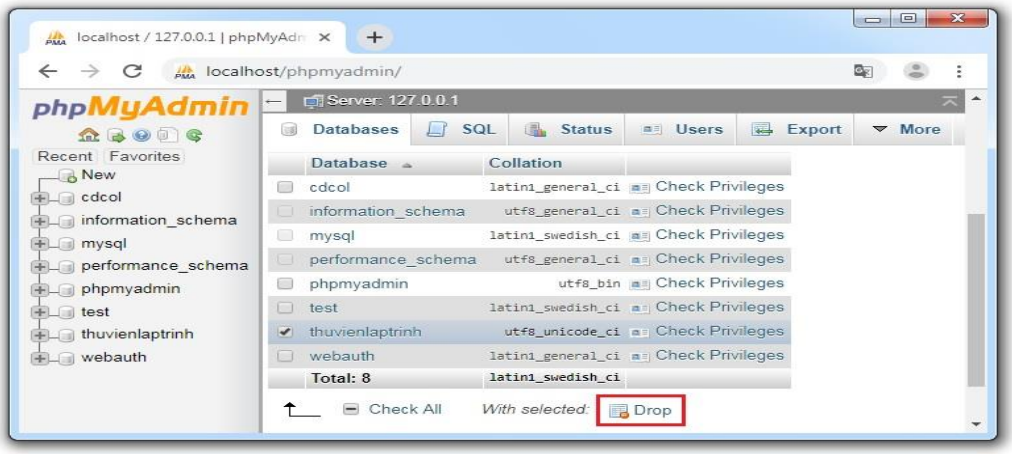
➢ PHP còn hạn chế về cấu trúc của ngữ pháp. Nó không được thiết kế gọn gàng và không được đẹp mắt như những ngôn ngữ lập trình khác.

➢ PHP chỉ có thể hoạt động và sử dụng được trên các ứng dụng trong web. Đó chính là lý do khiến cho ngôn ngữ này khó có thể cạnh tranh được với những ngôn ngữ lập trình khác. Nếu như muốn phát triển và nhân rộng hơn nữa trong lập trình. [7] [12]

1.4. Giới thiệu về cơ sở dữ liệu Mysql

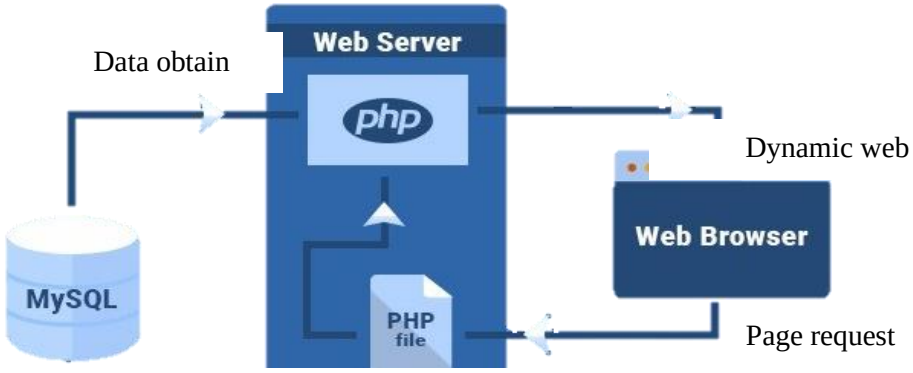
1.4.1. Đặc điểm của Mysql

Mysql là hệ quản trị cơ sở dữ liệu tự do nguồn mở phổ biến nhất thế giới và được các nhà phát triển rất ưa chuộng trong quá trình phát triển ứng dụng. Vì MySQL là hệ quản trị cơ sở dữ liệu tốc độ cao, ổn định và dễ sử dụng, có tính khả chuyển, hoạt động trên nhiều hệ điều hành cung cấp một hệ thống lớn các hàm tiện ích rất mạnh. Với tốc độ và tính bảo mật cao, MySQL rất thích hợp cho các ứng dụng có truy cập CSDL trên internet. Người dùng có thể tải về MySQL miễn phí từ trang chủ. MySQL có nhiều phiên bản cho các hệ điều hành khác nhau: phiên bản Win32 cho các hệ điều hành Windows, Linux, MacOSX, Unix, FreeBSD, NetBSD, Novell NetWare, SGI Irix, Solaris, SunOS... [12]



Hình 1.2 Cơ sở dữ liệu MySQL

1.4.2. MySQL hoạt động như thế nào?



Hình 1.3 Mô tả cách thức hoạt động của MySQL

Một máy client sẽ liên lạc với máy server trong một mạng nhất định. Mỗi client có thể gửi một request từ giao diện người dùng (Graphical user interface – GUI) trên màn hình, và server sẽ trả về kết quả như mong muốn. Miễn là cả hai hiểu nhau. Cách vận hành chính trong môi trường MySQL cũng như vậy:

- MySQL tạo ra bảng để lưu trữ dữ liệu, định nghĩa sự liên quan giữa các bảng đó.

- Client sẽ gửi yêu cầu SQL bằng một lệnh đặc biệt trên MySQL.

- Ứng dụng trên server sẽ phản hồi thông tin và trả về kết quả trên máy client.

1.4.3. Ưu điểm và nhược điểm của MySQL

Linh hoạt và dễ dùng: Quá trình cài đặt tương đối đơn giản và bạn có thể dễ dàng chỉnh sửa source code mà không phải thanh toán thêm tiền

Hiệu năng cao: Dù dữ liệu của bạn lớn như thế nào thì MySQL cũng đáp ứng với tốc độ cao, mượt mà kể cả big data của các trang thương mại điện tử hoặc những hoạt động kinh doanh nặng nề liên quan đến công nghệ thông tin.

Tiêu chuẩn trong ngành: Bất cứ ai đã dấn thân vào ngành công nghệ và dữ liệu thì đều đã sử dụng MySQL và người dùng cũng có thể triển khai dự án nhanh và thuê các chuyên gia dữ liệu.

An toàn: Vấn đề an toàn luôn là vấn đề cực kì quan trọng trong ngành dữ liệu và MySQL đảm bảo được tiêu chuẩn bảo mật rất cao. Bên cạnh nhưng ưu điểm nói trên Mysql vẫn còn tồn tại một số ngược điểm là: MySQL có thể bị

khai thác để chiếm quyền điều khiển. Dù có thể quản lí dữ liệu với số lượng lớn nhưng MySQL vẫn không đủ khả năng tích hợp quản lí dữ liệu khổng lồ và mang tính hệ thống cao như: hệ thống siêu thị trên toàn quốc, ngân hàng, quản lý thông tin dân số cả nước,…

1.5. Hệ điều hành thời gian thực – FreeRTOS

1.5.1. Đặc điểm của FreeRTOS

Hình 1.4: Logo FreeRTOS

FreeRTOS là một hệ điều hành nhúng thời gian thực (Real Time Operating System) mã nguồn mở được phát triển bởi Real Time Engineers Ltd, sáng lập và sở hữu bởi Richard Barry. FreeRTOS được thiết kế phù hợp cho nhiều hệ nhúng nhỏ gọn vì nó chỉ triển khai rất ít các chức năng như: cơ chế quản lý bộ nhớ và tác vụ cơ bản, các hàm API quan trọng cho cơ chế đồng bộ. Nó không cung cấp sẵn các giao tiếp mạng, drivers, hay hệ thống quản lý tệp (file system) như những hệ điều hành nhúng cao cấp khác. Tuy vậy, FreeRTOS có nhiều ưu điểm, hỗ trợ nhiều kiến trúc vi điều khiển khác nhau, kích thước nhỏ gọn (4.3 Kbytes sau khi biên dịch trên ARM7), được viết bằng ngôn ngữ C và có thể sử dụng, phát triển với nhiều trình biên dịch C khác nhau (GCC, OpenWatcom, Keil, IAR, Eclipse, …), cho phép không giới hạn các tác vụ chạy đồng thời, không hạn chế quyền ưu tiên thực thi, khả năng khai thác phần cứng. Ngoài ra, nó cũng cho phép triển khai các cơ chế điều độ giữa các tiến trình như: queues, counting semaphore, mutexes.

1.5.2. Ứng dụng của FreeRTOS

FreeRTOS [4] là một hệ điều hành nhúng rất phù hợp cho nghiên cứu, học tập về các kỹ thuật, công nghệ trong viết hệ điều hành nói chung và hệ điều hành nhúng thời gian thực nói riêng, cũng như việc phát triển mở rộng tiếp các thành phần cho hệ điều hành hành (bổ sung modules, driver, thực hiện porting).

1.6. Cập nhật Firmware từ xa OTA (Over The Air)

1.6.1. OTA là gì?

- Cập nhật firmware OTA (Over the Air) là tiến trình tải firmware mới vào ESP module thay vì sử dụng cổng Serial. Tính năng này thực sự rất hữu dụng trong nhiều trường hợp giới hạn về kết nối vật lý đến board mạch. [5]

- OTA có thể thực hiện với:

+Arduino IDE

+Web Browser

+HTTP Server

- Sử dụng OTA với tùy chọn dùng **Arduino IDE** trong quá trình phát triển, thử nghiệm, 2 tùy chọn còn lại phù hợp cho việc triển khai ứng dụng thực tế, cung cấp tính năng cập nhật OTA thông qua web hay sử dụng HTTP Server.

- Trong tất cả các trường hợp, thì Firmware hỗ trợ OTA phải được nạp lần đầu tiên qua cổng Serial, nếu mọi thứ hoạt động trơn tru, logic ứng dụng OTA hoạt động đúng thì có thể thực hiện việc cập nhật firmware thông qua OTA.

- Sẽ không có đảm bảo an ninh đối với quá trình cập nhật OTA bị hack. Nó phụ thuộc vào nhà phát triển đảm bảo việc cập nhật được phép từ nguồn hợp pháp, đáng tin cậy. Khi cập nhật hoàn tất, board mạch sẽ khởi động lại và thực thi code mới. Nhà phát triển phải đảm bảo ứng dụng thực trên module phải được tắt và khởi động lại 1 cách an toàn. Nội dung bên dưới cung cấp bổ sung các thông tin về an ninh, và an toàn cho tiến trình cập nhật OTA. [5]

1.6.2. Trình cập nhật OTA hoạt động như thế nào?

➢ Bản phác thảo đầu tiên nên được tải lên qua cổng nối tiếp. Bản phác thảo này phải chứa mã để tạo Trình cập nhật web OTA để bạn có thể tải lên mã sau này bằng trình duyệt của mình.

➢ Bản phác thảo của Trình cập nhật web OTA tạo ra một máy chủ web

mà bạn có thể truy cập để tải lên bản phác thảo mới thông qua trình duyệt web.

➢ Sau đó, bạn cần triển khai các quy trình OTA trong mỗi bản phác thảo mà bạn tải lên để có thể thực hiện các bản cập nhật / tải lên tiếp theo qua mạng.

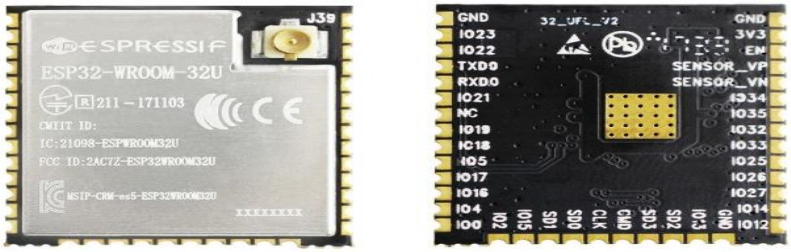
➢ Nếu bạn tải lên mã mà không có quy trình OTA, bạn sẽ không thể truy cập vào máy chủ web và tải lên bản phác thảo mới qua mạng. [5]

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về Module Esp32 DEVKIT V1 Doit

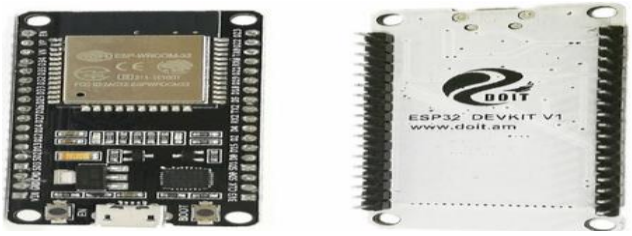
2.1.1. Tổng quan về ESP32 DEVKIT V1 Doit

Module NodeMCU ESP-32 DEVKIT V1 là một trong những bo mạch phát triển được tạo bởi NodeMcu để đánh giá module ESP-WROOM-32. Nó được dựa trên vi điều khiển ESP32 có Wifi, Bluetooth, Ethernet và Low Power hỗ trợ tất cả trong một chip duy nhất. [9]



Hình 2.1: ESP32-WROOM-32U

Cốt lõi của Module này là chip ESP32, được thiết kế để có thể mở rộng và thích ứng. Có 2 CPU có thể được điều khiển riêng hoặc cấp nguồn và tần số clock có thể điều chỉnh từ 80 MHz đến 240 MHz. Người dùng cũng có thể tắt nguồn CPU và sử dụng bộ xử lý công suất thấp để liên tục theo dõi các thiết bị ngoại vi để thay đổi hoặc vượt qua các ngưỡng. ESP32 tích hợp một tập hợp phong phú thiết bị ngoại vi, từ cảm biến điện dung cảm ứng, cảm biến Hall, bộ khuếch đại cảm biến tiếng ồn thấp, thẻ SD giao diện, Ethernet, SDIO / SPI tốc độ cao, UART, I2S và I2C.

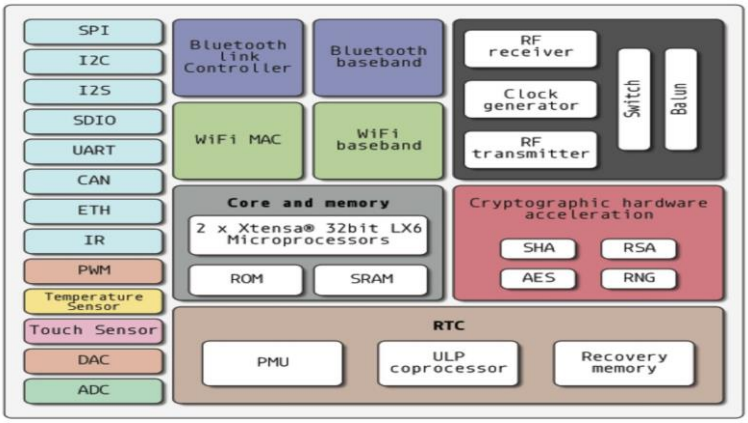


Hình 2.2: Kit ESP32 DIOT DEV KIT V1 Doit

Sự tích hợp Bluetooth, Bluetooth LE và Wi-Fi đảm bảo rằng một loạt các ứng dụng có thể được nhắm mục tiêu và đó là bằng chứng trong tương lai: Sử dụng Wi-Fi cho phép phạm vi vật lý lớn và kết nối trực tiếp tới internet thông qua bộ định tuyến WiFi, trong khi sử dụng Bluetooth cho phép người dùng kết

nối thuận tiện với điện thoại hoặc phát sóng đèn hiệu năng lượng thấp để phát hiện. Dòng ngủ của chip ESP32 ít hơn 5μA, làm cho nó phù hợp với các ứng dụng điện tử chạy bằng pin và đeo được. ESP-WROOM-32 hỗ trợ tốc độ dữ liệu lên đến 150 Mbps và công suất đầu ra 22 dBm tại PA để đảm bảo rộng nhất phạm vi vật lý. Như vậy, chip cung cấp thông số kỹ thuật hàng đầu trong ngành và tối ưu hóa tốt nhất hiệu suất cho tích hợp điện tử, phạm vi và mức tiêu thụ điện năng và kết nối. [9]

2.1.2. Thông số kĩ thuật



Hình 2.3: Sơ đồ khối chức năng của ESP32

a) Wifi

- 802.11 b / g / n / e / i.
- 802.11 n (2,4 GHz), với tốc độ lên tới 150 Mb / giây.
- 802.11 e: QoS để nhận ra kỹ thuật truyền thông không dây.
- WMM-PS, UAPSD.
- Kỹ thuật tập hợp A-MPDU và A-MSDU Frame.
- Phân mảnh và tái tổ hợp.
- Các tính năng an toàn 802.11 i: Xác thực trước và TSN.
- Hỗ trợ WPA /WPA2 /WPA2-Enterprise / Đã mã hóa WPS.
- Wi-Fi Direct (P2P), P2P phát hiện, chế độ P2P GO và quản lý điện năng P2P.
- Khả năng tương thích và chứng nhận UMA.
- Sự đa dạng và lựa chọn Antenna.

b) CPU và bộ nhớ

- Cung cấp điện áp: 2.2V đến 3.6V.

- Khả năng hoạt động của bộ xử lý lõi kép Xtensa® 32-bit LX6 có khả năng hoạt động cao tới 600 DMIPS.

- ROM 448 KByte /520 KByte SRAM SR 16 KByte SRAM trong RTC

- QSPI kết nối 4 Flash /SRAM nhiều nhất, và mọi Flash đều lớn nhất khoảng 16 MByte.

c) Thông số kỹ thuật nâng cao

Bộ ADC SAR 12 bit, hơn 18 truy cập.

- Hai bộ chuyển đổi DAC 8 bit.

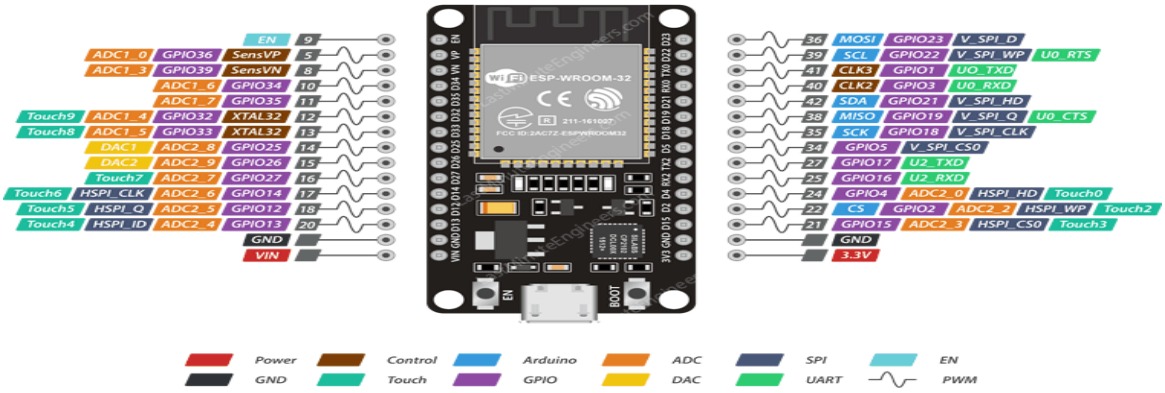
- Cảm biến 10 cảm ứng.

d) Board

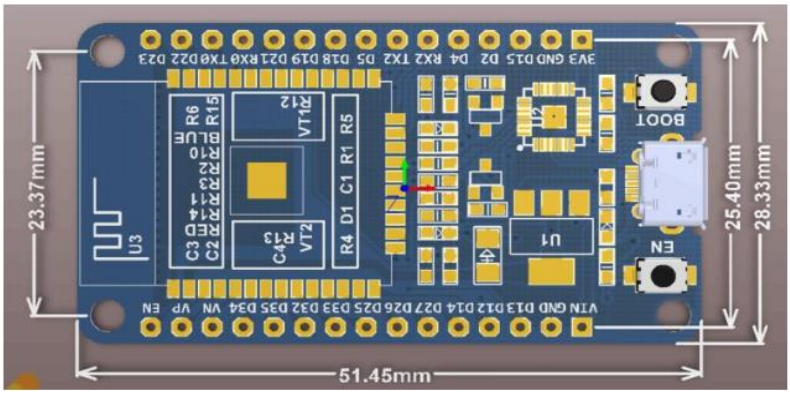
- USB .1x cổng microUSB để cấp nguồn và lập trình

- Các nút Misc.BOOT và EN, đèn LED màu đỏ (nguồn) và màu xanh (GPIO2)

- Nguồn điện: 5V qua USB hoặc 3V3 qua pin Vin [9]



Hình 2.4: Sơ đồ chân đầu ra của ESP32 DEVKIT V1 Doit



Hình 2.5: Kích thước board ESP32 DEVKIT V1 Doit

2.2. Module cảm biến đo nhiệt độ & độ ẩm DHT11

- **Khái quát**: Cảm biến nhiệt độ và độ ẩm DHT11 ra đời sau và được sử dụng thay thế cho dòng SHT1x ở những nơi không cần độ chính xác cao về nhiệt độ và độ ẩm. Cảm biến sử dụng giao tiếp số theo chuẩn 1 dây.

- **Ứng dụng**: Đo nhiệt độ và độ ẩm.

- **Thông số kỹ thuật**:

+ Nguồn: 3 -> 5 VDC.

+ Dòng sử dụng: 2.5mA max (khi truyền dữ liệu).

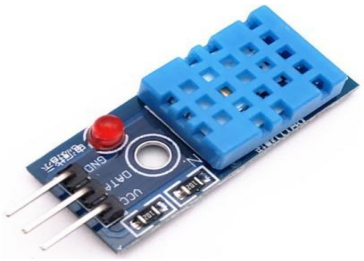
+ Đo tốt ở độ ẩm 20-80%RH với sai số 5%.

+ Đo tốt ở nhiệt độ 0 to 50°C sai số ±2°C.

+ Tần số lấy mẫu tối đa 1Hz (1 giây 1 lần)

+ Kích thước 15mm x 12mm x 5.5mm.

+ 4 chân, khoảng cách chân 0.1". [1]



Hình 2.6: Module cảm biến nhiệt độ & độ ẩm DHT11

- **Nguyên lí hoạt động**: Để có thể giao tiếp với DHT11 theo chuẩn 1 chân vi xử lý thực hiện theo 2 bước sau:

✓ Gửi tin hiệu muốn đo (Start) tới DHT11, sau đó DHT11 xác nhận lại.

✓ Khi đã giao tiếp được với DHT11, Cảm biến sẽ gửi lại 5 byte dữ liệu

và nhiệt độ đo được.

2.3. Module cảm biến ánh sáng BH1750

- Khái quát:

+Module cảm Biến cường độ ánh sáng BH1750 là cảm biến ánh sáng với bộ chuyển đổi AD 16bit tích hợp trong chip và có thể xuất ra trực tiếp dữ liệu theo dạng digital. cảm biến không cần bộ tính toán cường độ ánh sáng khác.

+BH1750 sử dụng đơn giản và chính xác hơn nhiều lần so với dùng cảm biến quang trở để đo cường độ ánh sáng với dữ liệu thay đổi trên điện áp dẫn đến việc sai số cao.Với cảm biến BH1750 cho dữ liệu đo ra trực tiếp với dạng đơn vị là LUX không cần phải tính toán chuyển đổi thông qua chuẩn truyền I2C.

- Ứng dụng: Đo cường độ ánh sáng.

- Thông số kỹ thuật:

+Chuẩn kết nối i2C

+Độ phân giải cao (1 - 65535 lx)

+Tiêu hao nguồn ít.

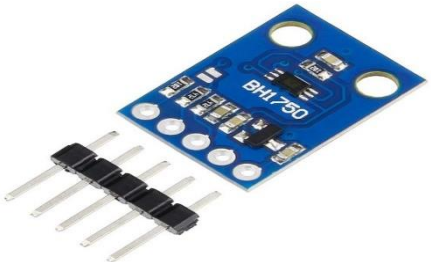
+Khả năng chống nhiễu sáng ở tần số 50 Hz/60 Hz

+Sự biến đổi ánh sáng nhỏ (+/- 20%)

+Độ ảnh hưởng bởi ánh sáng hồng ngoại rất nhỏ

+Nguồn cung cấp: 3.3V-5V

+Kích thước board: 0.85*0.63*0.13"(21*16*3.3mm) [2]



Hình 2.7: Module cảm biến đo cường độ ánh sáng BH1750

2.4. Module cảm biến đo khí gas MQ2

- Khái quát:

+**MQ2** là cảm biến khí, dùng để phát hiện các khí có thể gây cháy. Nó

được cấu tạo từ chất bán dẫn SnO2. Chất này có độ nhạy cảm thấp với không khí sạch. Nhưng khi trong môi trường có chất gây cháy, độ dẫn của nó thay đổi ngay. Cảm biến khí gas MQ2 đưa ra chân A0 điện áp từ 0V đến 5V tương ứng với nồng độ chất gây cháy trong không khí. Chúng ta có thể kết hợp với các bộ chuyển đổi để đọc tín hiệu điện áp này và đo lường chất lượng không khí.

+Ngoài ra, module được tích hợp bộ so sánh lm393 để đưa ra điện áp mức 0 hoặc mức 1. Biến trở điều chỉnh độ nhạy của cảm biến, khi tín hiệu từ chân A0 vượt quá giá trị mà ta cài đặt ở biến trở. Module sẽ đưa ra mức 1 ở chân DO.

- **Ứng dụng**: Được sử dụng để phát hiện các loại khí như sau

+LPG: là hỗn hợp hydrocarbon nhẹ, ở thể khí. LPG trong dân dụng và công nghiệp chủ yếu có thành phần gồm Propane

+Iso Butan (C4H10)

+Propan: C3H8

+Mêtan: CH4

+Rượu: ROH

+Hydrogen

+Khói

- **Thông số kỹ thuật:**

+Điện áp hoạt động: 3.3V-5V

+Kích thước PCB: 3cm * 1.6cm

+Led đỏ báo nguồn vào, Led xanh báo gas

+IC so sánh: LM393

+VCC: 3.3V-5V

+GND: 0V

+DO: Đầu ra tín hiệu số (0 và 1)

+AO: Đầu ra Analog (Tín hiệu tương tự)

+Cấu tạo từ chất bản dẫn Sno2

+Có 2 dạng tín hiệu: Analog (AO) và Digital (DO):

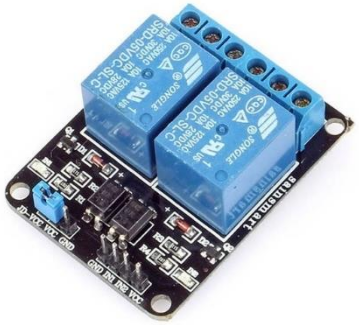
✓ Dạng tín hiệu: TTL đầu ra 100mA (Có thể sử dụng trực tiếp Relay, Còi công suất nhỏ...)

✓ Điều chỉnh độ nhạy bằng biến trở.

✓ Sử dụng LM393 để chuyển AO --> DO [3]

Hình 2.8: Module cảm biến đo khí gas MQ2

2.5. Module relay 2 kênh 5V-220V 10A



Hình 2.9: Module relay 2 kênh 5V-220V 10A

- **Khái quát**:

Module relay 2 kênh 5V-220V 10A thích hợp cho các ứng dụng đóng ngắt điện thế cao AC hoặc DC, các thiết bị tiêu thụ dòng lớn. Module thiết kế nhỏ gọn và có thể sử dụng nguồn ngoài giúp cho việc sử dụng trở nên linh động, dễ dàng hơn.

- **Ứng dụng**: Nhận tín hiệu từ chân vi điều khiển để đóng cắt relay

- **Thông số kỹ thuật**:

+ Sử dụng điện áp nuôi 5VDC.

+ 2 Relay đóng ngắt ở điện thế kích bằng 0V nên có thể sử dụng cho cả tín hiệu 5V hay 3v3 (cần cấp nguồn ngoài), mỗi Relay tiêu thụ dòng khoảng 80mA.

+ Điện thế đóng ngắt tối đa: AC250V - 10A hoặc DC30V - 10A.

2.6. Mạch giảm áp DC–DC Buck LM2596

Hình 2.10: Module giảm áp DC-DC Buck LM2596

- **Khái quát**:

Mạch giảm áp DC-DC Buck LM2596 có kích thước nhỏ gọn có khả năng giảm áp từ 30VDC xuống 1.5VDC mà vẫn đạt hiệu suất cao (92%), thích hợp cho các ứng dụng chia nguồn, hạ áp, cấp cho các thiết bị như camera, robot,...

- **Ứng dụng**: Được sử dụng để hạ áp, chia nguồn

- **Thông số kỹ thuật:**

✓ Điện áp đầu vào: Từ 3V đến 30V.

✓ Điện áp đầu ra: Điều chỉnh được trong khoảng 1.5V đến 30V.

✓ Dòng đáp ứng tối đa là 3A.

✓ Hiệu suất: 92%

✓ Công suất: 15W

✓ Kích thước: 45 (dài) * 20 (rộng) * 14 (cao) mm

2.7. Kết luận chương 2:

Chương 2 trình bày về nguyên lý hoạt động cũng như thông số kỹ thuật của các module cảm biến như DHT11, BH1750, MQ2, module relay 2 kênh 5V-220V 10A, Mạch giảm áp DC-DC Buck LM2596. Từ đó sẽ vận dụng tốt hơn các công nghệ, module trong đồ án này.

CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

3.1. Tính toán và thiết kế

3.1.1. Yêu cầu tính toán

a. Yêu cầu công nghệ

Bảng 3.1: Yêu cầu công nghệ

Yêu cầu	Mục tiêu, tiêu chí
Đối tượng sử dụng	Nhà thông minh, các đồ điện – điện tử gia dụng…
Server	✓ Server đa người dùng ✓ Chạy ổn định ✓ Người dùng tùy chỉnh cấu hình ✓ Giao diện thân thiện, đẹp mắt, responsive. ✓ Bảo mật
Vi điều khển	✓ Chạy ổn định ✓ Delay thấp ✓ Cập nhật firmware từ xa OTA ✓ Người dùng tùy chỉnh cấu hình, thông tin Wifi, ID, … ✓ Code tối ưu, chia functions rõ ràng
Mô hình sản phẩm	✓ Mô phỏng được thệ thống ✓ Hoạt động ổn định trong thời gian dài ✓ Thiết kế đẹp ✓ Chi phí tối ưu

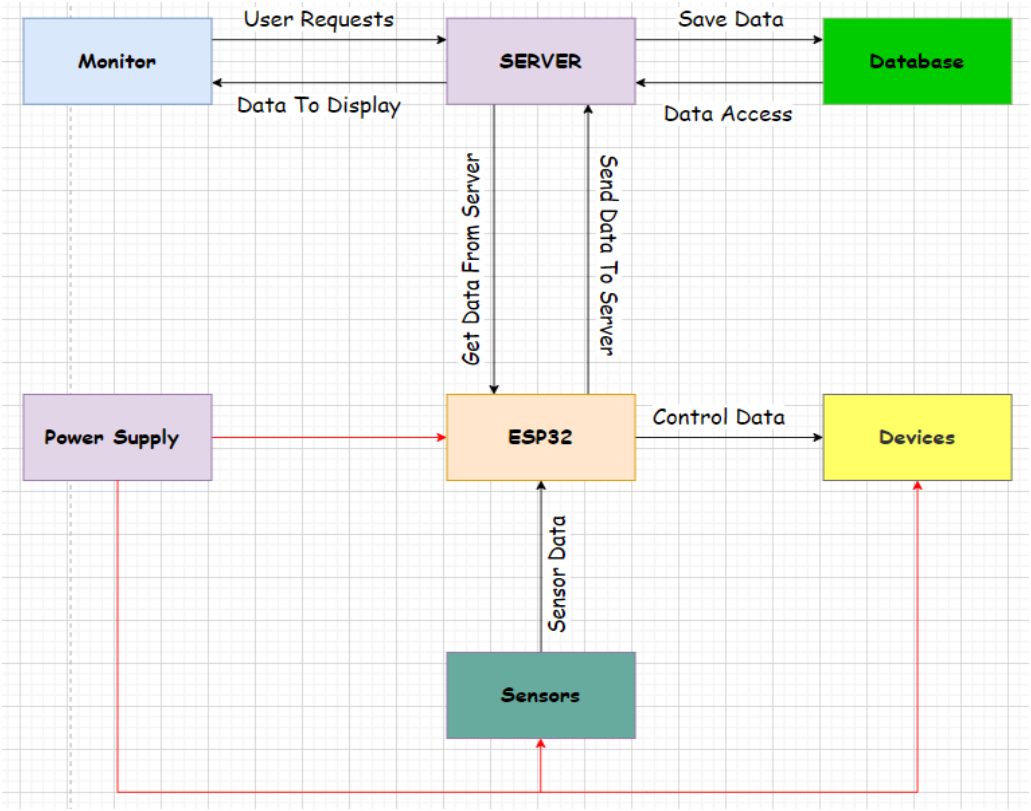
b. Xác định thành phần thực hiện yêu cầu thiết kế

Bảng 3.2: Thành phần thực hiện yêu cầu thiết kế

Thành Phần	Nội Dung
Yêu cầu bài toán	✓ Đo nhiệt độ, độ ẩm, cường độ ánh sáng, khí gas gửi lên Server và lưu vào Database. ✓ Server truy xuất dữ liệu từ Database để hiển thị ra giao diện người dùng và lưu lệnh từ màn hình điều khiển vào Database. ✓ Vi điều khiển nhận dữ liệu diều khiển từ Server

	sau đó giải mã, phân tích và thực thi.
Đầu vào hệ thống	✓ Các cảm biến (tín hiệu analog hoặc digital) ✓ Yêu cầu từ người dùng
Đầu ra Hệ thống	✓ Các thiết bị được điều khiển ✓ Giao diện người dùng

3.1.2. Sơ đồ khối tổng quan hệ thống



Hình 3.1: Sơ đồ hệ thống truyền nhận

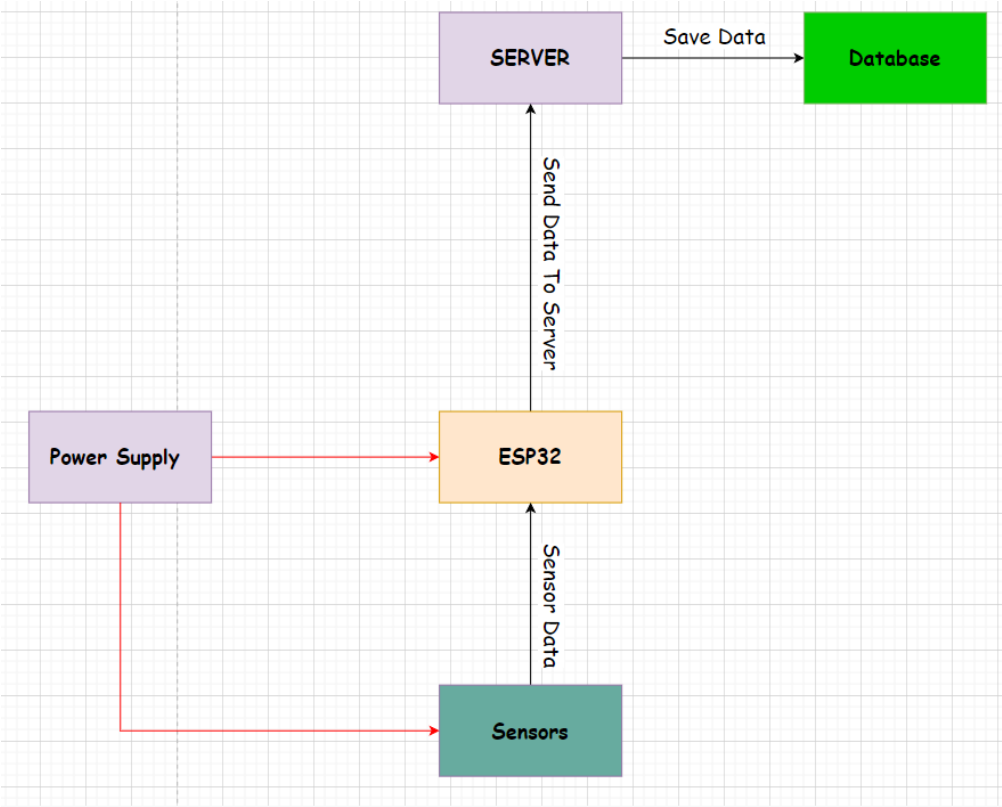
➢ **Để hệ thống hoạt động được ta cần giải quyết 3 bài toán lớn sau:**

✓ Đo nhiệt độ, độ ẩm, cường độ ánh sáng, khí gas gửi lên Server và lưu vào Database.

✓ Server truy xuất dữ liệu từ Database để hiển thị ra giao diện người dùng và lưu lệnh từ màn hình điều khiển vào Database.

✓ Vi điều khiển nhận dữ liệu diều khiển từ Server sau đó giải mã, phân tích và thực thi.

Bài toán 1: Đo nhiệt độ, độ ẩm, cường độ ánh sáng, khí gas gửi lên Server và lưu vào Database.

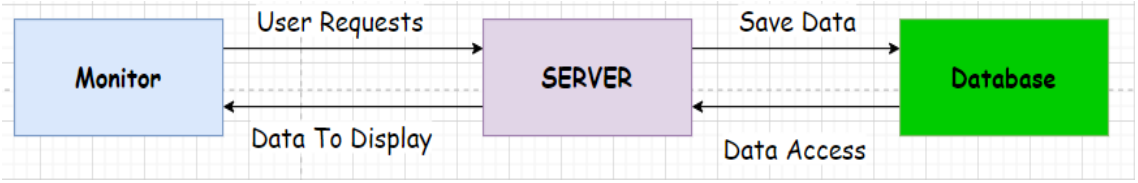


Hình 3.2: Sơ đồ khối bài toán 1

Bảng 3.3: Luồng code bài toán 1

Bước	Mô tả
1	Các cảm biến đo và gửi giá trị về cho ESP32.
2	ESP32 xử lý dữ liệu.
3	ESP32 kết nối đến Server và gửi dữ liệu theo kiểu JSON lên Server theo phương thức POST.
4	Server nhận và phân tích dữ liệu JSON được gửi lên.
5	Lưu dữ liệu vừa phân tích vào Database

Bài toán 2: Server truy xuất dữ liệu từ Database để hiển thị ra giao diện người dùng và lưu lệnh từ màn hình điều khiển vào Database**.**

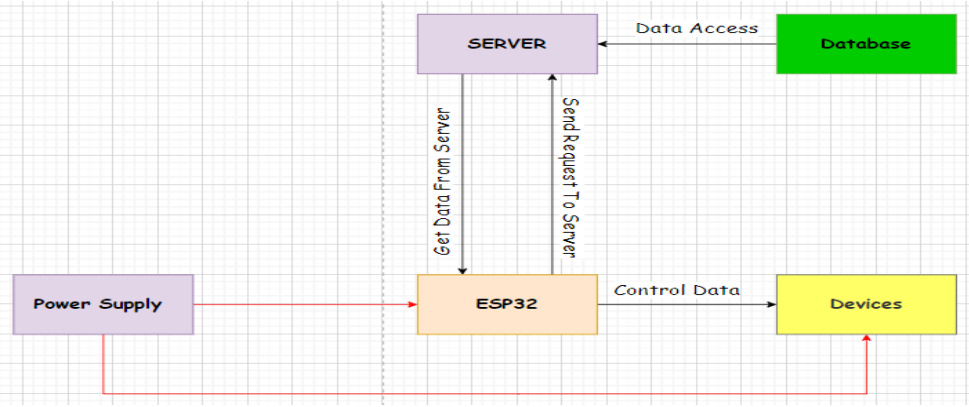


Hình 3.3: Sơ đồ khối bài toán 2

Bảng 3.4: Luồng code bài toán 2

Bước		**Mô tả**
Truy xuất dữ liệu từ Database	1	User tạo 1 request thông tin cần hiển thị đến Server.
	2	Server nhận và xử lí resquest.
	3	Server truy xuất dữ liệu cần hiển thị từ trong Database.
	4	Trả kết quả về màn hình hiển thị.
Lưu user request vào Database	1	User tạo 1 request đến Server.
	2	Server nhận và xử lí resquest.
	3	Lưu request đó vào bảng điều khiển trong Database

Bài Toán 3: Vi điều khiển nhận dữ liệu diều khiển từ Server sau đó giải mã, phân tích và thực thi.

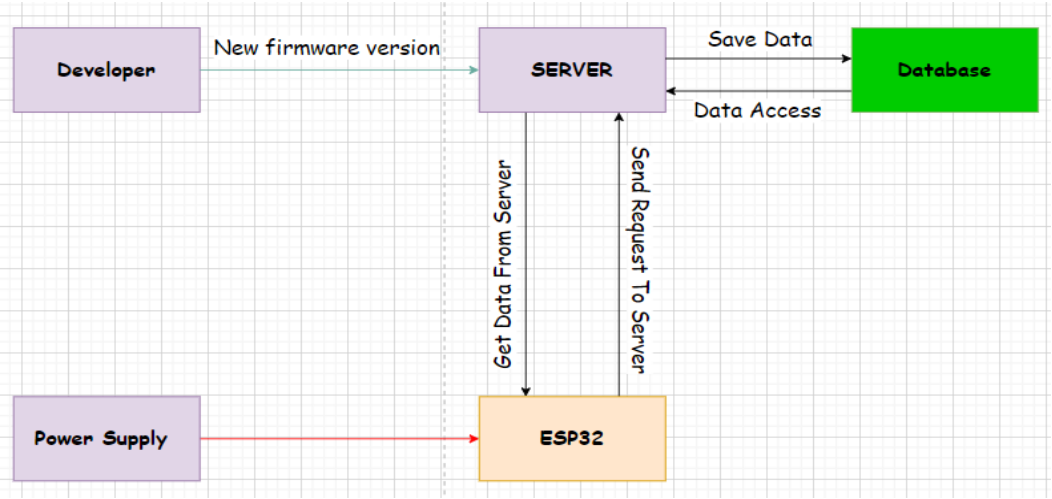


Hình 3.4: Sơ đồ khối bài toán 3

Bảng 3.5: Luồng code bài toán 3

Bước	**Mô tả**
1	ESP32 gửi request get data theo chu kỳ đặt trước lên Server
2	Server nhận request và xử lý
3	Truy xuất vào Database lấy dữ liệu cần trả về
4	Server trả về một chuỗi dữ liệu dưới dạng JSON
5	ESP32 nhận dữ liệu trả về theo phương thức GET
6	ESP32 giải mã và phân tích chuỗi JSON vừa nhận được
7	Thực thi hoặc xuất tín hiệu ra các chân tương ứng

3.1.3. Cập nhật firmware từ xa OTA



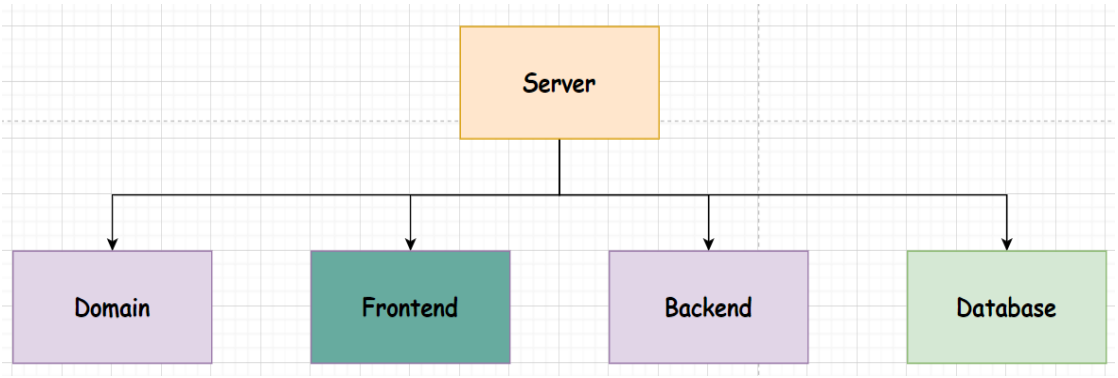
Hình 3.5: Sơ đồ khối Cập nhật OTA

Bảng 3.6: Luồng code cập nhật OTA

Bước	Mô tả
1	ESP32 gửi request get data theo chu kỳ đặt trước lên Server
2	Server nhận request và xử lý
3	Truy xuất vào Database lấy dữ liệu cần trả về
4	Server trả về một chuỗi dữ liệu dưới dạng JSON
5	ESP32 nhận dữ liệu trả về theo phương thức GET
6	ESP32 giải mã và phân tích chuỗi JSON vừa nhận được
7	Thực thi OTA function

3.2. Xây dựng Server, Database và lập trình vi điều khiển

3.2.1. Xây dựng Server



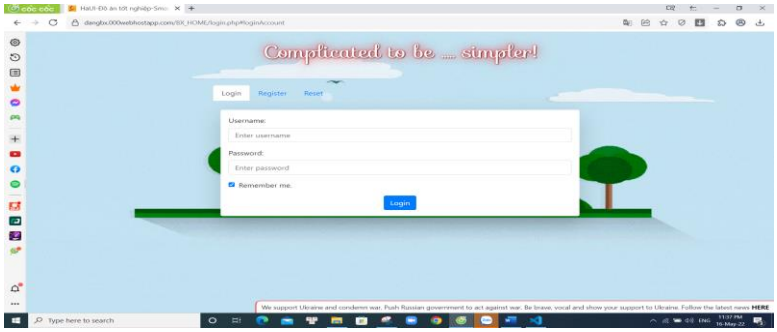
Hình 3.6:Thành phần cơ bản của Server

Bảng 3.7: Các thành phần của Server

Thành phần	**Mô tả**	**Sử dụng**
Host, Domain	Là nơi lưu trữ source code, các tài nguyên, các phiên bản cập nhật … Là địa chỉ truy cập của web server.	https://dangbx.000webhostapp.com
Frontend	Là phần giao diện sử dụng của Server.	Bootstrap 4, HTML, CSS, JavaScript.
Backend	Là phần xử lí dữ liệu, các thuật toán, cũng như tương tác với cơ sở dữ liệu.	PHP, JavaScript.
Database	Là nơi lưu trữ dữ liệu	MySQL Database.

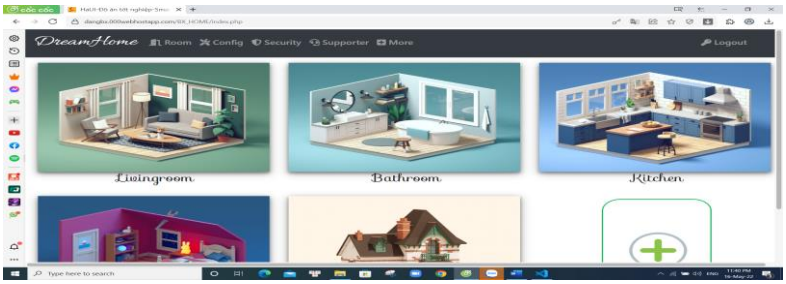
3.2.2. Các tính năng của Server

✓ Đăng nhập, đăng ký tài khoản và reset mật khẩu



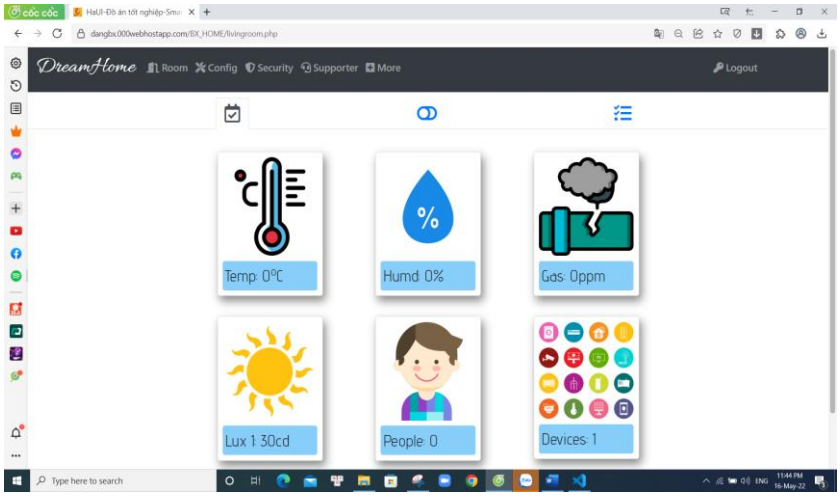
Hình 3.7: Trang đăng nhập, đăng ký và reset mật khẩu

✓ Giao diện chính: Danh sách các phòng



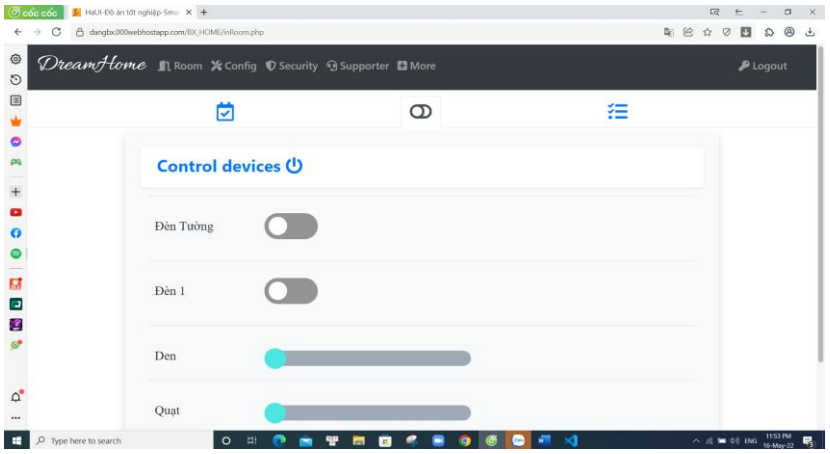
Hình 3.8: Giao diện hiển thị các phòng

✓ Giao diện giám sát trong phòng



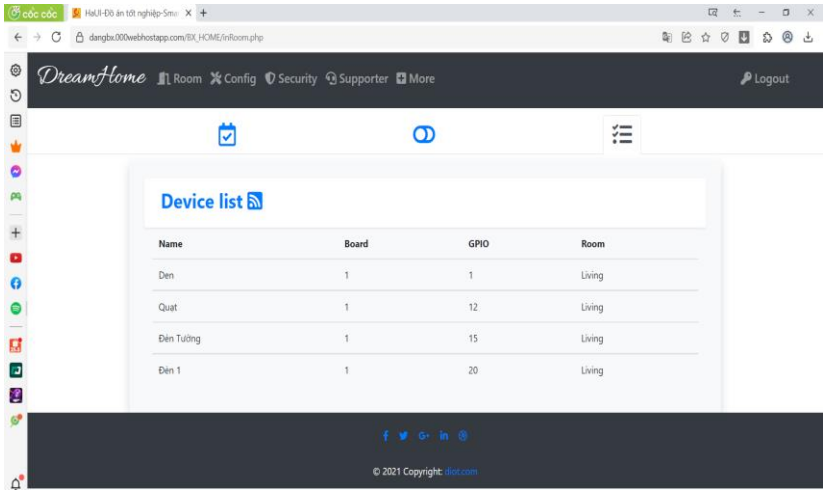
Hình 3.9: Giao diện giám sát

✓ Giao diện điều khiển thiết bị



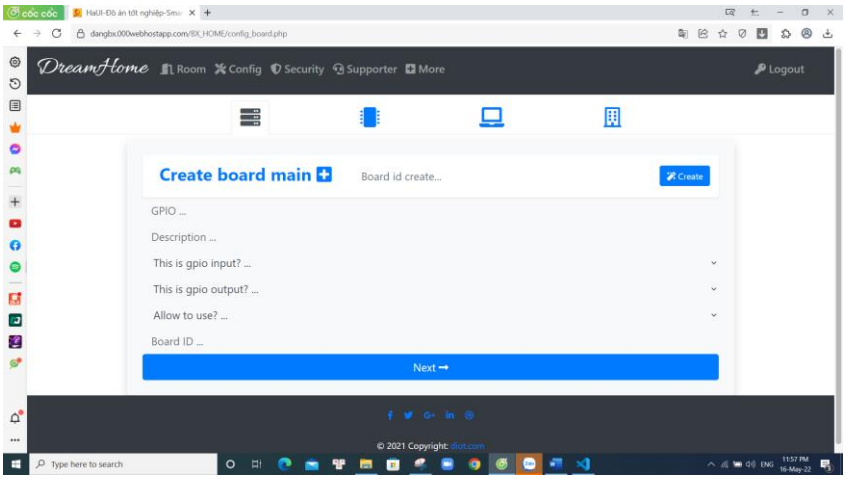
Hình 3.10: Giao diện điều khiển thiết bị

✓ Danh sách thông tin các thiết bị đang kết nối trong phòng



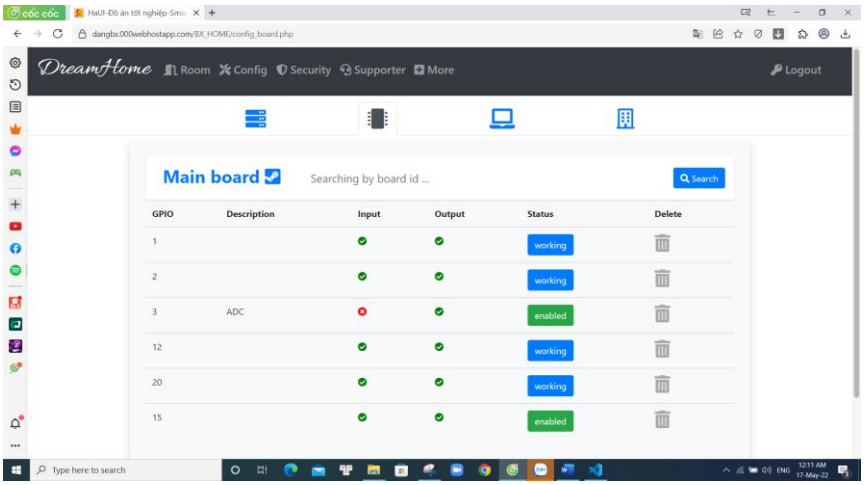
Hình 3.11: Danh sách các thiết bị kết nối

✓ Giao diện trang người dùng tự cấu hình



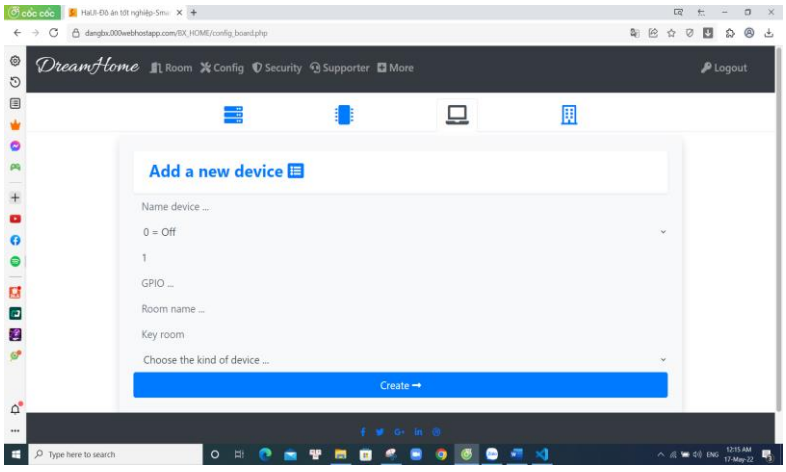
Hình 3.12: Thêm một Board mới

✓ Danh sách trạng thái các GPIO



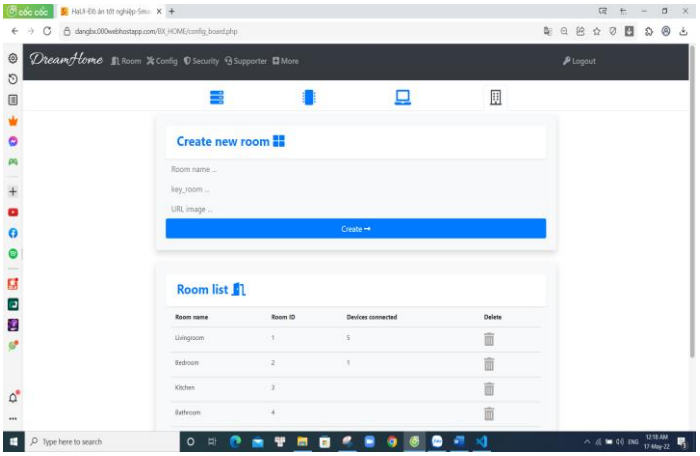
Hình 3.13: Danh sách GPIO

✓ Giao diện thêm các thiết bị



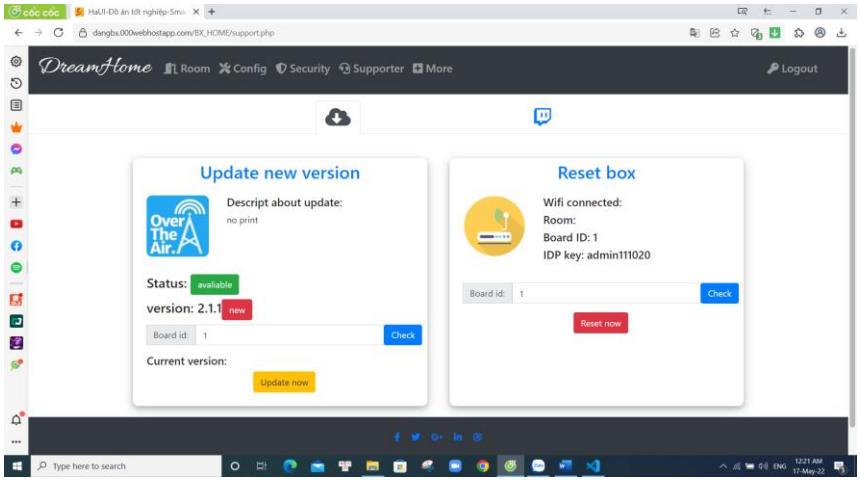
Hình 3.14: Thêm các thiết bị

✓ Giao diện thêm, xóa và danh sách các phòng



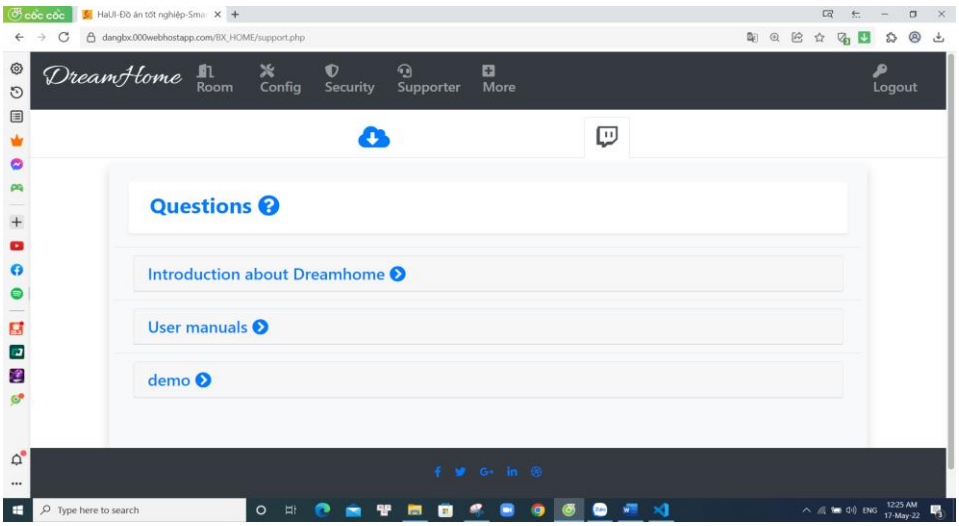
Hình 3.15: Thêm, xóa các phòng

✓ Giao diện cập nhật firmware và reset board



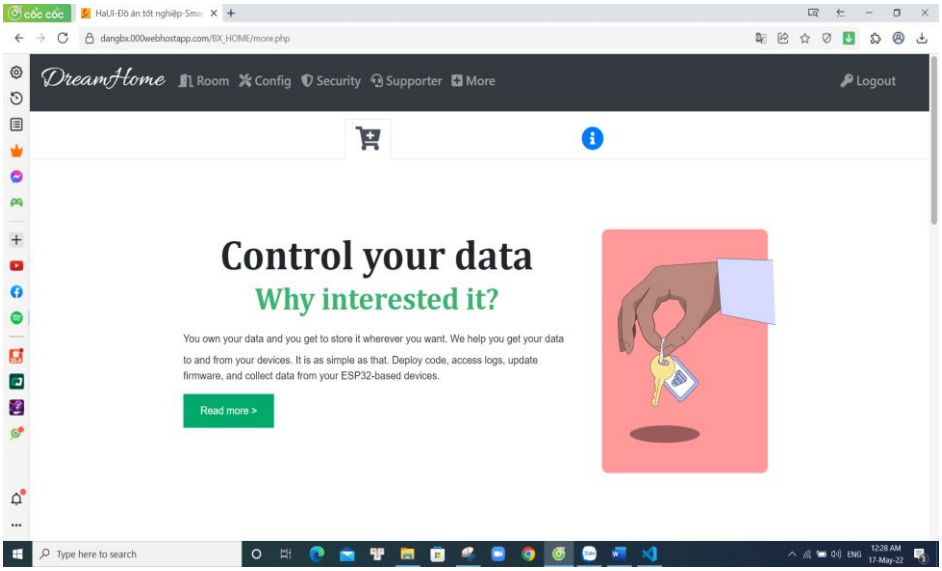
Hình 3.16: Cập nhật firmware và reset board

✓ Hướng dẫn sử dụng



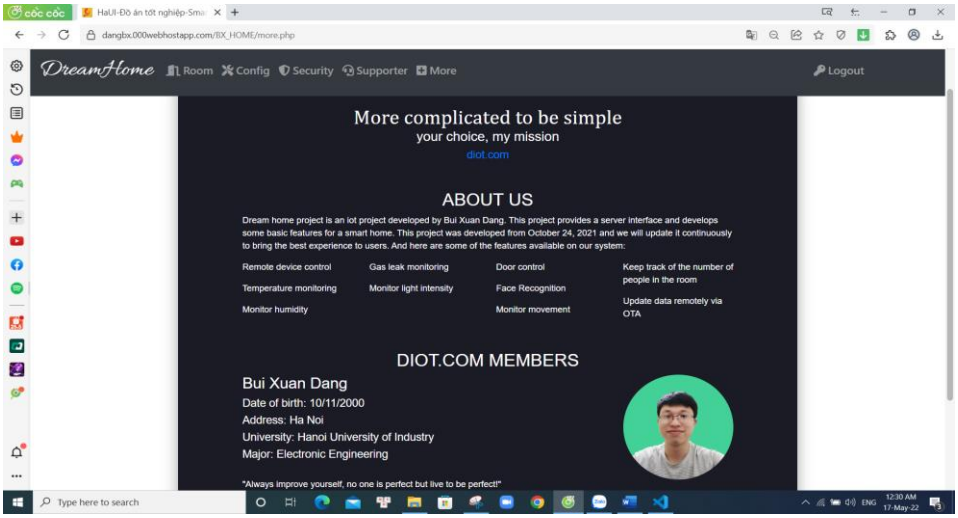
Hình 3.17: Hướng dẫn sử dụng

✓ Trang tiếp thị sản phẩm mới



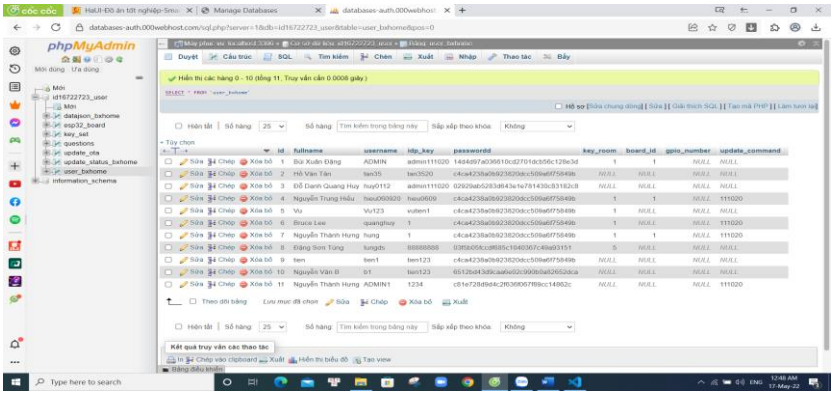
Hình 3.18: Tiếp thị sản phẩm

✓ Trang thông tin về nhà phát triển



Hình 3.19: Thông tin về nhà phát triển

3.2.3. Thiết kế Database

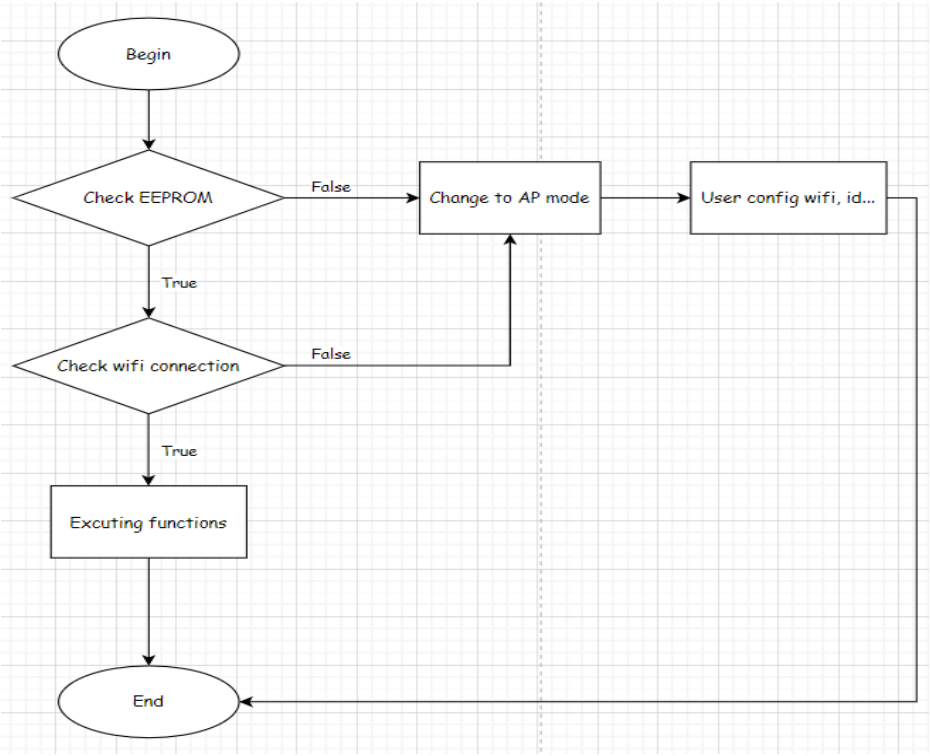


Hình 3.20: Giao diện MySQL Database

Bảng 3.8: Các bảng database sử dụng trong dự án

Bảng Data	**Mô tả**
User_bxhome	Lưu trữ account, và thông tin board …
Update_status_bxhome	Lưu trữ thông tin được đẩy lên từ vi điều khiển (Nhiệt độ, độ ẩm, ánh sáng …)
Update_ota	Lưu trữ thông tin phiên bản, trạng thái, mô tả của bản cập nhật firmware mới.
questions	Lưu trữ thông tin hướng dẫn sử dụng, …
Esp32_board	Lưu trữ thông tin về Board, GPIO
Datajson_bxhome	Lưu trữ thông tin, trạng thái các thiết bị.

3.3. Xây dựng code vi điều khiển



Hình 3.21: Lưu đồ thuật toán

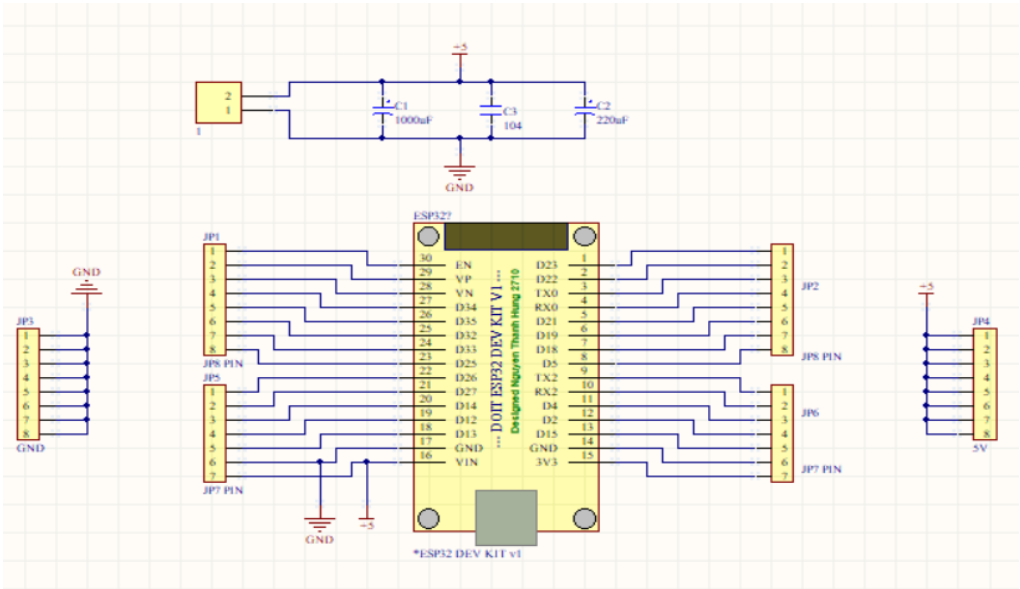
Bảng 3.9: Luồng code của vi điều khiển

Luồng code	**Mô tả**
Check EEPROME	➢ Đọc dữ liệu trong EEPROM để xem board đã được cấu hình các thông tin như Wifi, id … hay chưa.

	○ Nếu chưa: Chuyển sang chế độ Access Point. Người dùng sẽ bắt Wifi và truy cập vào web được nhúng trên board để cấu hình cho board mạch.
	✓ Nếu đã cấu hình thì chương trình sẽ kết nối với mạng đã cấu hình trong EEPROM.
Check wifi connection	➢ Kiểm tra xem đã kết nối thành công hay chưa.
	✓ Thành công: Chạy vào chương trình chính
	○ Không thành công: Thử kết nối lại trong một khoảng thời gian đặt trước. Nếu không kết nối được sẽ báo "time out". Xóa EEPROM và chuyển sang chế độ Access Point.
Core 0	➢ Gửi request lên Server, nhận dữ liệu trả về từ Server, giải mã, thực thi.
	➢ Kiểm tra kết nối wifi và tự động kết nối lại nếu đường truyền bị gián đoạn.
Core 1	➢ Đọc và xử lý các dữ liệu nhận từ cảm biến sau đó mã hóa theo kiểu JSON và gửi lên Server.

3.4. Thiết kế mạch nguyên lý

Sơ đồ nguyên lý



Hình 3.22: Sơ đồ nguyên lý

Thiết kế node cảm biến

Yêu cầu của khối cảm biến: khối này sẽ có nhiệm vụ thu thập các thông số của môi trường để cung cấp chúng cho khối xử lý trung tâm có thông số để từ đó có những xử lý, gửi lên Server.

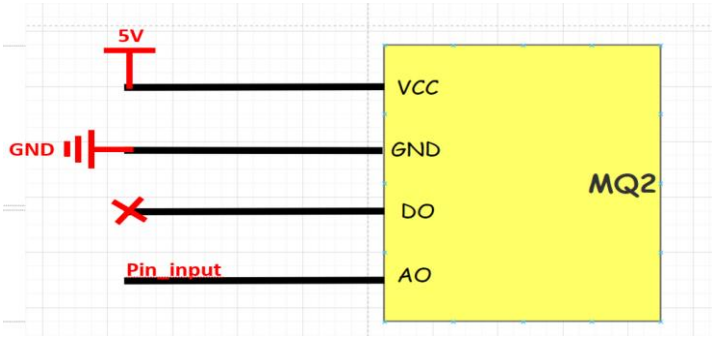
✓ **Cảm biến khí gas (MQ2)**

Thông số kỹ thuật:

- Điện áp hoạt động:5V.

- Data: Đầu ra Digital (Tín hiệu analog).

Mạch nguyên lý:



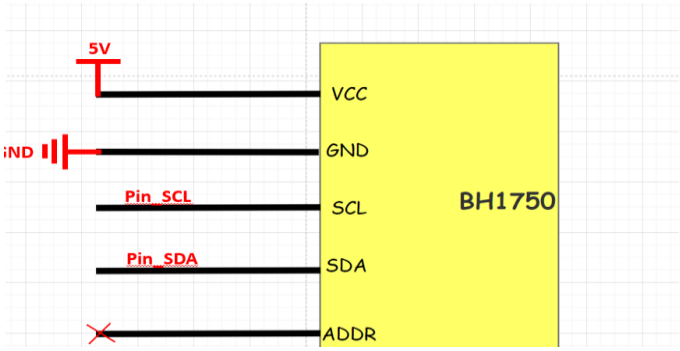
Hình 3.23: Mạch nguyên lý cảm biến khí gas MQ2

✓ **Cảm biến cường độ ánh sáng (BH1750)**

Thông số kỹ thuật:

- Điện áp hoạt động: 3.3V-5V.

- Chuẩn giao tiếp SPI.



Hình 3.24: Mạch nguyên lý cảm biến ánh sáng

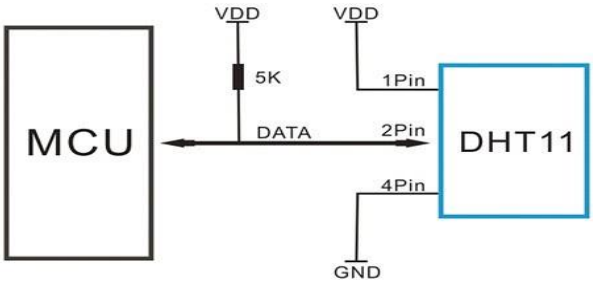
✓ **Cảm biến nhiệt độ, độ ẩm (DHT11)**

Thông số kỹ thuật:

- Điện áp hoạt động: 3.3V-5V.

- Data: Đầu ra Digital (Tín hiệu số).

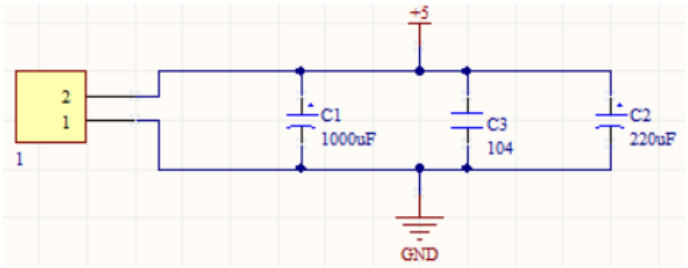
Mạch nguyên lý:



Hình 3.25: Mạch nguyên lý DHT11 và vi điều khiển

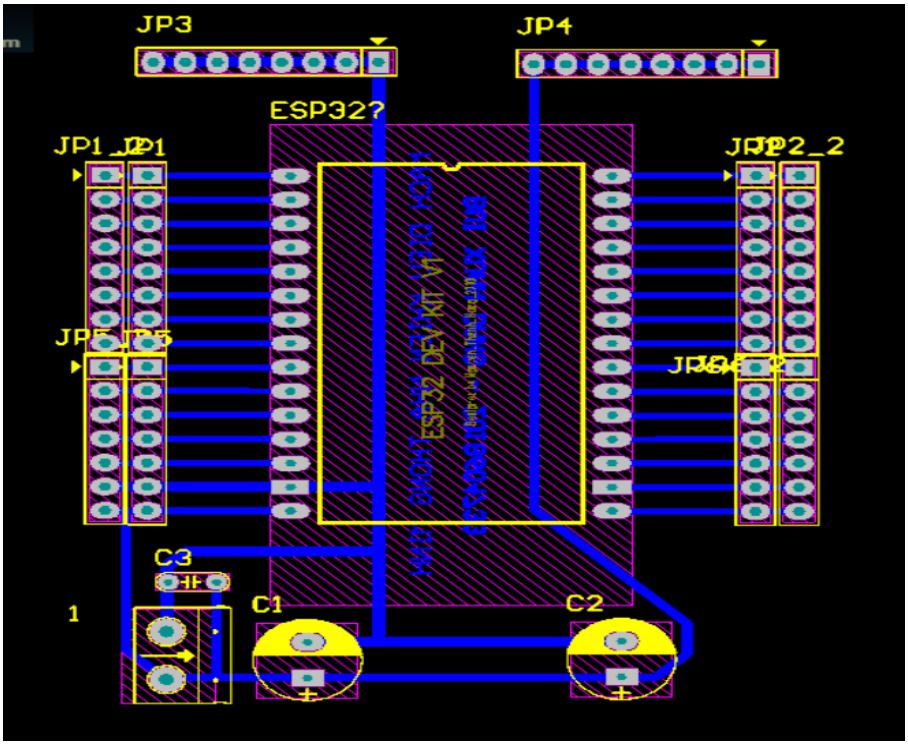
✓ **Khối nguồn:**

Khối nguồn cung cấp nguồn nuôi toàn bộ mạch. Nguồn được lấy vào là Adapter nguồn.

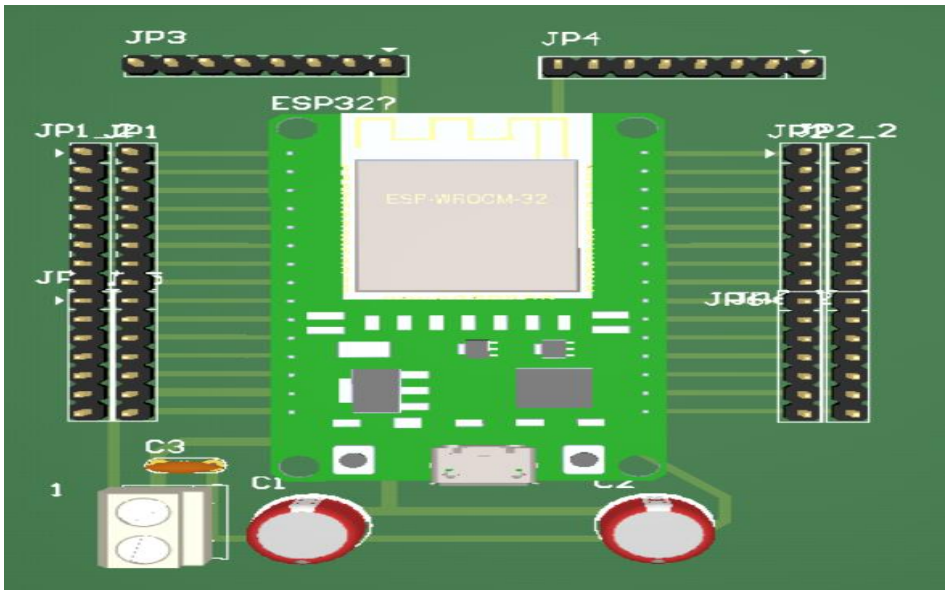


Hình 3.26: Khối nguồn

3.4.1. Mạch PCB



Hình 3.27: PCB 2D thiết kế trên phần mềm Altium Designer



Hình 3.28: PCB 3D thiết kế trên phần mềm Altium Designer

3.5. Xây dựng mô hình thực tế

3.5.1. Các linh kiện, vật liệu và module sử dụng trên board

Bảng 3.10: Các linh kiện sử dụng trên một board

Tên	**Số lượng**	**Giá thành (vnđ)**
ESP32 DEVKIT-V1 Doit	1	90.000
Jumper đực đơn 1x40P 2.54mm	3	5.000
Jumper cái đơn 1x40P 2.54mm	3	5.000
Tụ phân cực 1000uF	1	1.000
Tụ phân cực 220uF	1	1.000
Tụ gốm 104	1	1.000
Tổng giá thành		103.000

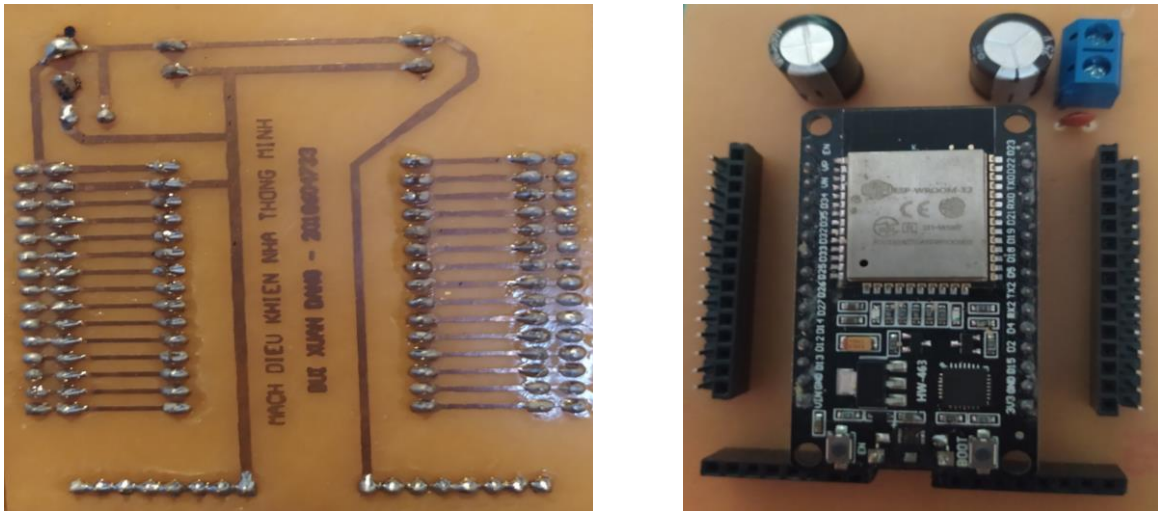
3.5.2. Các module cảm biến khác (tùy sử dụng)

Bảng 3.11: Các linh kiện và module khác

Tên	**Giá thành (vnđ / 1 sản phẩm)**
Cảm biến nhiệt độ, độ ẩm DHT11	18.000
Đo cường độ ánh sáng BH1750	36.000
Cảm biến khí gas MQ2	22.000

Module relay (1/2/4/8 kênh)	30.000 (2 kênh)
Mạch giảm áp DC-DC Buck LM2596	16.000
Còi 5V	4.000
Quạt 12V	15.000
Đèn led	6.000 (túi)
Bìa cứng Formex	9.000
Adapter 220V AC ~ 12V DC 2A	35.000
Điện trở 330 Ohm	2.000 (túi)
Điện trở 5K Ohm	2.000 (túi)

3.5.3. Hình ảnh sản phẩm thực tế



Hình 3.29: Mạch PBC thực tế

3.5.4. Mô hình:

Hình 3.30: Ảnh mô hình

3.6. Kiểm thử hệ thống

Sau khi đã thiết kế sơ đồ khối cũng như lập trình chương trình xong, em tiến hành kiểm tra thử nghiệm hệ thống để tiến hành tìm kiếm các lỗi cũng như kiểm tra hoạt động của hệ thống.

3.6.1. Kiểm tra khả năng hoạt động của Server:

Sau khi chạy test, Server hoạt động ổn định, đạt được toàn bộ các yêu cầy đề ra ở **bảng 3.1.** Tuy nhiên vẫn còn một vài hạn chế sau:

Bảng 3.12: Hạn chế của Server

Vấn đề	**Nguyên nhân**	**Giải quyết**	**Trạng thái**
Tốc độ Server thấp	- Do đường truyền mạng - Do sử dụng host có băng thông thấp	✓Cải thiện tốc độ đường truyền ✓Nâng cấp băng thông của host	Chưa được xử lý
Khả năng bảo mật chưa cao	- Do sử dụng host free - Tên miền đang là http:// - Mật khẩu chưa được mã hóa	✓Nâng cấp host ✓Đổi tên miền sang https:// ✓Mã hóa mật khẩu	Đã xử lý
Lỗi không kết nối đến Database	- Lỗi cấu hình khi đẩy từ local lên internet	✓Cấu hình lại Database trên internet	Đã xử lý

3.6.2. Kiểm tra khả năng hoạt động của vi điều khiển:

Sau khi chạy test, mô hình hoạt động ổn định, đạt được các yêu cầu chính như:

✓ Đọc được các dữ liệu cảm biến và đẩy lên Server

✓ Nhận dữ liệu từ Server và xử lý

✓ Chạy Multi Core và cập nhật Firmware từ xa bằng OTA

✓ Tùy chỉnh cấu hình chân và có thể phát triển xa trong tương lai

✓ Chi phí sản xuất thấp

Tuy nhiên vẫn còn một vài vấn đề sau

Bảng 3.13: Hạn chế của sản phẩm

Vấn đề	**Nguyên nhân**	**Giải quyết**	**Trạng thái**
Không hoạt động khi mất kết nối internet	- Chưa có nút nhấn để điều khiển local	✓ Thêm trường nút nhấn trên trang cấu hình ở Server và update code vi điều khiển. Tuy nhiên sẽ có khả năng tốn thêm tài nguyên GPIO	Chưa được xử lý
Tốn năng lượng	- Chạy Multi core - Chưa có deep sleep	✓ Chạy đơn luồng, tuy nhiên tốc độ xử lý sẽ bị chậm đi ✓ Tìm hiểu thêm về deep sleep	Chưa được xử lý
Đếm số người trong phòng	- Dùng cảm biến hồng ngoại chưa thực sự hợp lý	✓ Ứng dụng OpenCV	Chưa được xử lý
Wifi cố định	- Chưa có smart config	✓ Chạy web socket trên ESP32	Đã xử lý

3.6.3. Hướng dẫn sử dụng:

Bảng 3.14: Hướng dẫn sử dụng một số tính năng

Tính năng	**Mô tả các bước**	**Vai trò**
✓ Tạo tài khoản ✓ Đăng nhập ✓ Reset mật khẩu	B1: Vào trang đăng nhập B2: Chọn đăng ký / đăng nhập / reset mật khẩu B3: Nhập thông tin	Người dùng

	B4: Nhấn submit	
✓ Giám sát và điều khiển từng phòng	B1: Vào trang giao diện chính B2: Chọn phòng muốn giám sát và điều khiển B3: Giám sát và điều khiển	Người dùng
✓ Cấu hình cho Server	B1: Chọn tab config B2: Chọn các tab muốn cấu hình như: - Thêm board mạch mới, - Thêm/xóa: GPIO, thiết bị, phòng B3: Nhập thông tin tương ứng B4: Nhấn submit	Kỹ sư
✓ Cập nhật OTA	B1: Tạo file .bin của Firmware mới B2: Truy cập vào tab update OTA B3: Nhập thông tin tương ứng và đẩy file lên Server	Nhà phát triển
	B1: Vào tab supporter B1: Nhập id của board mạch muốn update firmware B4 PE: Nhấn kiểm tra phiên bản hiện tại B3: Nhấn update	Người dùng Kỹ sư
✓ Reset board mạch	B1: Vào tab supporter B2: Nhập id của board mạch muốn reset B3: Nhấn reset	Người dùng Kỹ sư

3.6.1. Một số lỗi thường gặp:

Vấn đề	**Khắc phục**
Cập nhật OTA không thành công	B1: Restart board mạch

	B2: Cập nhật lại
Cấu hình sai thông tin board mạch	B1: Vào trang supporter B2: Reset Board mạch B3: Truy cập vào websocket của board để cấu hình lại
Không truy cập được websocket	B1: Restart lại board B2: Kết nối lại wifi local B3: Nhập đúng địa chỉ của websocket và truy cập.

Nếu gặp lỗi không khắc phục được hãy gửi báo cáo về hòm thư của nhà phát triển.

Gmail: dangbuixuan10112000@gmail.com

Fanpage: https://www.facebook.com/diot.com

3.7. Kết luận chương 3:

Đã xây dựng thành công Server bằng PHP MySQL database. Sử dụng công nghệ AJAX cho việc gửi giữ liệu được mã hóa ở dạng JSON. Server đảm bảo hầu hết các tính năng yêu cầu đề ra và có khả năng tiến xa trong tương lai. Tuy nhiên vẫn bị hạn chế băng thông do sử dụng host free. Tính bảo mật chưa tối ưu. Server thiết kế đẹp mắt và thân thiện với người dùng.

Về phần vi điều khiển, đã đọc được các dữ liệu cảm biến và đẩy lên Server, hoàn thiện hầu hết các tính năng ngoại trừ đếm số người trong phòng. Ứng dụng RTOS và cập nhật Firmware từ xa OTA thành công. Mạch chạy ổn định.

Tuy còn những hạn chế, tuy nhiên với công nghệ OTA và ý tưởng thiết kế về một board mạch mở rộng, tùy chỉnh cấu hình thì tất cả vấn đề đều sẽ được giải quyết. Đó cũng chính là ý tưởng cốt lõi của đồ án này.

KẾT LUẬN

Sau khoảng thời gian làm đề tài dưới sự chỉ đạo hướng dẫn tận tình của TS. Hà Thị Kim Duyên, em đã hoàn thành được đồ án tốt nghiệp của mình. Nội dung của đồ án là xây dựng Server giám sát và điều khiển nhà thông minh. Em đã sử dụng kiến thức về lập trình web, PHP, MySQL, AJAX, JSON, OTA, FreeRTOS, ESP32… Trong quá trình tìm hiểu và thực hiện, đồ án đã đạt được một số kết quả sau:

- ✓ Xây dựng được Server đúng với mục tiêu đề ra.
- ✓ Xây dựng được mô hình nhà thông minh, hoạt động ổn định.
- ✓ Hoàn thiện được hết các tính năng yêu cầu.
- ✓ Xử lý được một số vấn đề xảy ra.
- ✓ Hiểu rõ hơn về hệ thống IoT.
- ✓ Nâng cao khả năng lập trình web, lập trình nhúng.

Hướng phát triển:

Đây là một dự án với tính ứng dụng cao, và đi theo thiên hướng mở, thân thiện với người dùng cũng như học tập …

Trong tương lai dự án sẽ được cập nhật code thường xuyên qua OTA và khắc phục những vấn đề chưa được giải quyết.

- ➢ Cải thiện giao diện Server sao cho thân thiện với người dùng hơn.
- ➢ Nâng cao tính năng bảo mật và phân quyền tài khoản.
- ➢ Hướng đến thương mại hóa sản phẩm.
- ➢ Ứng dụng công nghệ AI, thư viện OpenCV vào nhà thông minh.
- ➢ Phát triển tích hợp thêm nhiều board mạch khác.
- ➢ Điều khiển bằng giọng nói bằng Google Home, công nghệ IFTTT.

TÀI LIỆU THAM KHẢO

[1] "https://dientu360.com/cam-bien-nhiet-do-do-am-dht11,".

[2] "https://dientu360.com/cam-bien-cuong-do-anh-sang-bh1750,".

[3] "https://chotroihn.vn/module-cam-bien-khi-gas-mq-2,".

[4] "https://www.freertos.org,".

[5] "https://arduino.esp8266.vn/network/ota.html,".

[6] "https://www.w3schools.com/html,".

[7] "https://www.w3schools.com/php,".

[8] "https://www.w3schools.com/bootstrap4,".

[9] "https://randomnerdtutorials.com,".

[10] "https://www.youtube.com/c/digikey,".

[11] "https://www.youtube.com/channel/UCDp5eU-n6aaRcI42_uZz1Qw,".

[12] "https://www.youtube.com/c/PHPGURU,".

[13] "https://www.youtube.com/c/RuiSantosdotme,".

PHỤ LỤC

Code ESP32:

```
#include <Arduino.h>
#include <BH1750.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include<Arduino_JSON.h>
#include <Wire.h>
#include <ESP32httpUpdate.h>
#include "DHT.h"
#include <WebServer.h>
#include <EEPROM.h>

/* Open web server port 80 */
WebServer webServer(80);

/* Link of server */
const char* serverNameSendData = "http://dangbx.000webhostapp.com/BX_HOME/api.php";
const char* serverNameGetData;
String serverNameGetData1;

/* Timer's variables */
unsigned long previousMillisSendData = 0;
unsigned long previousMillisGetData =0;
unsigned long previousMillis = 0;
unsigned long interval = 1000;

/* PWM's variables*/
const int freq = 5000;
const int resolution = 8;
const int ledPin1 = 13;
const int ledPin2 = 14;
const int ledPin3 = 25;
```

```
int dutyCycle1;
int dutyCycle2;
int dutyCycle3;
const int ledChannel1 = 0;
const int ledChannel2 = 1;
const int ledChannel3 = 2;

/* Sensor's variable */
float temp,humd,gas,lux;
int people=0;
const int gasPin = 34;
const int speakerPin = 2;
const int fanOutPin = 4;
#define DHTPIN 15
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
BH1750 lightMeter;

/* Task names of multi core */
TaskHandle_t get_data_from_server_core0;
TaskHandle_t send_data_to_server_core1;

/* Json's variables */
String dataJson, idpJson;

/* Check value variable */
int value_check;

/* Version of firmware */
String version = "2.1";

/* Board ìnfomation's variables */
String ssid,pass,keyroom,boardDevice,idpkey,postData;
```

```
/* Web server infomations in AP mode*/
char* ssid_ap = "DREAM HOME";
char* pass_ap = "88888888";
IPAddress ip_ap(192,168,1,1);
IPAddress gateway_ap(192,168,1,1);
IPAddress subnet_ap(255,255,255,0);

/* Variable constain source of HTLM Website */
const char MainPage[] PROGMEM = R"=====(
<!DOCTYPE html>
 <html>
  <head>
    <title>DREAM HOME</title>
    <style>
      body {
        text-align:center;
     }
      h1
      {color:#003399;}
      h3{color: #003399;text-align: left;}
      h4{color: #003399;}
      a {text-decoration: none;color:#FFFFFF;}
      .contentBox{
        background-color: antiquewhite;
        width: 90%;
        margin: auto;
        margin-top: 50px;
        box-shadow: 5px 10px 8px #888888;
        border-radius: 15px;
        padding: 10px;
      }
      .contentInfo{
         clear: both;
         margin: 10px;
```

```
            }
            .inputInfo{
               float: right;
               border: none;
               border-radius: 10px;
               font-size: 1rem;
               padding: 5px;
               color: #003399;
               width: 60%;
            }
            .buttonSubmit{
              border-radius: 10px;
              font-size: 1rem;
              padding: 10px;
              border-width: 0.5px;
            }
          </style>
          <meta name="viewport" content="width=device-width,user-scalable=0" charset="UTF-8">
        </head>
        <body>
          <div class="contentBox">
            <div><h1>Dream Home</h1></div>
            <div class="contentInfo"><h3>Wifi name:<input class="inputInfo" id="ssid" placeholder="wifi name..."/></h3></div>
            <div class="contentInfo"><h3>Password:<input type="password" class="inputInfo" id="pass" placeholder="password..."/></h3></div>
            <div class="contentInfo"><h3> Key Room:<input class="inputInfo" id="keyroom" placeholder="key room..."/></h3></div>
            <div class="contentInfo"><h3> Board id:<input class="inputInfo" id="boardDevice" placeholder="Board id..."/></h3></div>
            <div class="contentInfo"><h3>IDP key:<input class="inputInfo" id="idpkey" placeholder="idp key..."/></h3></div>
            <div class="contentInfo" id="reponsetext"></div>
```

```
        <div>
          <button class="buttonSubmit bt_write" onclick="writeEEPROM()"
style="background-color: rgb(161, 255, 73);">CONFIG</button>
          <button class="buttonSubmit bt_clear" onclick="clearEEPROM()"
style="background-color: tomato;">DELETE</button>
        </div>
        <h4>Designed by: DangBX.iot.com</h4>
      </div>
      <script>
        var xhttp = new XMLHttpRequest();
        /* Initialized data and send requests ssid and password */
        function writeEEPROM(){
          var ssid = document.getElementById("ssid").value;
          var pass = document.getElementById("pass").value;
          var keyroom = document.getElementById("keyroom").value;
          var boardDevice = document.getElementById("boardDevice").value;
          var idpkey = document.getElementById("idpkey").value

xhttp.open("GET","/writeEEPROM?ssid="+ssid+"&pass="+pass+"&keyroom="+keyro
om+"&boardDevice="+boardDevice+"&idpkey="+idpkey,true);

          /* Get reponse */
          xhttp.onreadystatechange = process;
          xhttp.send();
        }
        function clearEEPROM(){
          xhttp.open("GET","/clearEEPROM",true);
          xhttp.onreadystatechange = process;//nhận reponse
          xhttp.send();
        }
        /* Check response */
        function process(){
          if(xhttp.readyState == 4 && xhttp.status == 200){
            /* Updata data using javascript */
```

```
          ketqua = xhttp.responseText;
          document.getElementById("reponsetext").innerHTML=ketqua;
         }
        }
      </script>
   </body>
  </html>
)=====";

void setup() {
  /* Setup pin mode */
  pinMode(speakerPin,OUTPUT);
  pinMode(fanOutPin,OUTPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
  pinMode(DHTPIN, INPUT);

  /* Open serial with bault rate is 115200 */
  Serial.begin(115200);

  /* Initialized eeprom memory with 512 bytes*/
  EEPROM.begin(512);
  delay(10);

  /* Read eeprom to check the empty of board infomations*/
  if(read_EEPROM()){
    checkConnection();
  }else{

    /* Change to AP mode */
    WiFi.disconnect();
    WiFi.mode(WIFI_AP);
    WiFi.softAPConfig(ip_ap, gateway_ap, subnet_ap);
```

```
    WiFi.softAP(ssid_ap,pass_ap);
    Serial.println("Soft Access Point mode!");
    Serial.print("Please connect to ");
    Serial.println(ssid_ap);
    Serial.print("Web Server IP Address: ");
    Serial.println(ip_ap);
   }
   webServer.on("/",mainpage);
   webServer.on("/writeEEPROM",write_EEPROM);
   webServer.on("/clearEEPROM",clear_EEPROM);
   webServer.begin();
   dht.begin();
   Wire.begin();// Enable I2C pins of Arduino
   lightMeter.begin();

   /* CORE 0 */
   xTaskCreatePinnedToCore(task_get_data_from_server_core0,  "Task1codeCore0",
5000, NULL, 1, &get_data_from_server_core0, 0);

   /* CORE 1 */
   xTaskCreatePinnedToCore(task_send_data_to_server_core1,    "Task1codeCore1",
5000, NULL, 1, &send_data_to_server_core1, 1);
  }

  void task_get_data_from_server_core0( void * pvParameters ){
   for(;;){
    unsigned long currentMillisSendData = millis();
    if(currentMillisSendData - previousMillisSendData >= interval){
     if(idpkey != 0){
      if(WiFi.status()== WL_CONNECTED){
       read_BH1750();
       read_DHT11();
       read_MQ2();
       send_data();
```

```
        Serial.println(postData);
        previousMillisSendData = currentMillisSendData;
        vTaskDelay(2000);
       }
       else{
        wifi_reconnect();
       }
      }
      else{
       webServer.handleClient();
      }
    }
  }
}
void task_send_data_to_server_core1( void * pvParameters ){
  for(;;){
    unsigned long currentMillisGetData = millis();
    if(currentMillisGetData - previousMillisGetData >= interval){
      if(idpkey != 0){
        if(WiFi.status()== WL_CONNECTED){
          json_decoder();
          pwm_process();
          Serial.println(serverNameGetData);
          previousMillisGetData = currentMillisGetData;
          vTaskDelay(2000);
        }
        else{
          wifi_reconnect();
        }
      }
      else{
        webServer.handleClient();
      }
    }
```

```
  }
}
void loop() {
  // put your main code here, to run repeatedly
}

/* Wifi config */
/* Functions is used to run web socket*/
void mainpage(){
  String s = FPSTR(MainPage);
  webServer.send(200,"text/html",s);
}

/* Functions is used to read data from eeprom */
boolean read_EEPROM(){
  Serial.println("Reading EEPROM...");
  if(EEPROM.read(0)!=0){
    ssid = pass = keyroom = idpkey = boardDevice="";
    for (int i=0; i<32; ++i){
      ssid += char(EEPROM.read(i));
    }
    Serial.print("SSID: ");
    Serial.println(ssid);
    for (int i=32; i<64; ++i){
      pass += char(EEPROM.read(i));
    }
    Serial.print("PASSWORD: ");
    Serial.println(pass);
    for (int i=64; i<96; ++i){
      keyroom += char(EEPROM.read(i));
    }
    Serial.print("Key room: ");
    Serial.println(keyroom);
    for (int i=96; i<128; ++i){
```

```
      boardDevice += char(EEPROM.read(i));
    }
    Serial.print("Board id: ");
    Serial.println(boardDevice);
    for (int i=128; i<192; ++i){
      idpkey += char(EEPROM.read(i));
    }
    Serial.print("IDP key: ");
    Serial.println(idpkey);
    ssid = ssid.c_str();
    pass = pass.c_str();
    keyroom = keyroom.c_str();
    boardDevice = boardDevice.c_str();
    idpkey = idpkey.c_str();
    serverNameGetData1 =
"http://dangbx.000webhostapp.com/BX_HOME/action.php?action=get_data_json&board
Device="+boardDevice+"&idp_key="+idpkey;
    serverNameGetData = (char*) serverNameGetData1.c_str();
    return true;
  }else{
    Serial.println("Data wifi not found!");
    return false;
  }
 }

 /* Function is used to check the connection */
 void checkConnection() {
  Serial.println();
  Serial.print("Check connecting to ");
  Serial.println(ssid);
  WiFi.disconnect();
  WiFi.begin(ssid.c_str(),pass.c_str());
  int count=0;
  while(count < 100){
```

```
    if(WiFi.status() == WL_CONNECTED){
      WiFi.mode(WIFI_STA);
      Serial.println();
      Serial.print("Connected to ");
      Serial.println(ssid);
      Serial.print("Web Server IP Address: ");
      Serial.println(WiFi.localIP());
      return;
    }
    delay(200);
    Serial.print(".");
    count++;
  }
  clear_EEPROM();
  Serial.println("Timed out.");
  WiFi.disconnect();
  WiFi.mode(WIFI_AP);
  WiFi.softAPConfig(ip_ap, gateway_ap, subnet_ap);
  WiFi.softAP(ssid_ap,pass_ap);
  Serial.println("Soft Access Point mode!");
  Serial.print("Please connect to ");
  Serial.println(ssid_ap);
  Serial.print("Web Server IP Address: ");
  Serial.println(ip_ap);
}

/* Function is used to wirte data to eeprom */
void write_EEPROM(){
  ssid = webServer.arg("ssid");
  pass = webServer.arg("pass");
  keyroom = webServer.arg("keyroom");
  boardDevice = webServer.arg("boardDevice");
  idpkey = webServer.arg("idpkey");
  Serial.println("Clear EEPROM!");
```

```
for (int i = 0; i < 192; ++i) {
  EEPROM.write(i, 0);
  delay(10);
}
for (int i = 0; i < ssid.length(); ++i) {
  EEPROM.write(i, ssid[i]);
}
for (int i = 0; i < pass.length(); ++i) {
  EEPROM.write(32 + i, pass[i]);
}
for (int i = 0; i < keyroom.length(); ++i) {
  EEPROM.write(64 + i, keyroom[i]);
}
for (int i = 0; i < boardDevice.length(); ++i) {
  EEPROM.write(96 + i, boardDevice[i]);
}
for (int i = 0; i < idpkey.length(); ++i) {
  EEPROM.write(128 + i, idpkey[i]);
}
EEPROM.commit();
Serial.println("Writed to EEPROM!");
Serial.print("SSID: ");
Serial.println(ssid);
Serial.print("PASS: ");
Serial.println(pass);
Serial.print("Key room: ");
Serial.println(keyroom);
Serial.print("Board id: ");
Serial.println(boardDevice);
Serial.print("IDP key: ");
Serial.println(idpkey);
String s = "Đã lưu thông tin wifi";
webServer.send(200, "text/html", s);
restart_ESP();
```

```
}

/* Function is used to restart ESP32 */
void restart_ESP(){
  ESP.restart();
  String s = "Đã khởi động lại ";
  webServer.send(200, "text/html", s);
}

/* Function is used to clear eeprom */
void clear_EEPROM(){
  Serial.println("Clear EEPROM!");
  for (int i = 0; i < 192; ++i) {
    EEPROM.write(i, 0);
    delay(10);
  }
  EEPROM.commit();
  String s = "Đã xóa bộ nhớ EEPROM";
  webServer.send(200,"text/html", s);
}

/* Function is used to auto reconnect wifi*/
void wifi_reconnect(){
  unsigned long currentMillis = millis();
  /* f WiFi is down, try reconnecting every CHECK_WIFI_TIME seconds */
  if ((WiFi.status() != WL_CONNECTED) && (currentMillis - previousMillis >=
interval)) {
    Serial.print(millis());
    Serial.println("Reconnecting to WiFi...");
    WiFi.disconnect();
    WiFi.reconnect();
    previousMillis = currentMillis;
  }
}
```

```
/* Function is used to update firmware by OTA */
void ota_update(){
 if (1)
 {
  t_httpUpdate_return ret = ESPhttpUpdate.update("https://dangbx.000webhostapp.com/BX_HOME/uploads/HTTPS_OTA_Update.ino.doitESP32devkitV1.bin"); // Bạn cần thay đúng địa chỉ web chứa fw của bạn
    switch(ret) {
     case HTTP_UPDATE_FAILED:
        Serial.println("[update] Update failed.");
        break;
     case HTTP_UPDATE_NO_UPDATES:
        Serial.println("[update] Update no Update.");
        break;
     case HTTP_UPDATE_OK:
        Serial.println("[update] Update ok."); // may not called we reboot the ESP
        restart_ESP();
        break;
    }
 }
}

/* Function is used to decoder data Json */
void json_decoder(){
 dataJson = httpGETRequestDevice(serverNameGetData);
 Serial.println(dataJson);
 JSONVar deviceObj = JSON.parse(dataJson);
 if (JSON.typeof(deviceObj)=="undefined")
 {
  Serial.print("Parsing input failed!");
  return;
 }
```

```
Serial.print("JSON Object Device = ");
Serial.println(deviceObj);
JSONVar keys = deviceObj.keys();
 for (int i = 0; i < keys.length(); i++)
 {
  JSONVar value = deviceObj[keys[i]];
  value_check = atoi(value);
  if (atoi(keys[i]) == 111020)
  {
   if(value_check == 1){
    Serial.print("cap nhat one");
    ota_update();
   }
   else if(value_check == 2){
    Serial.print("Reset board");
    clear_EEPROM();
   }
  }
  else if (atoi(keys[i])!=13 && atoi(keys[i])!=14 && atoi(keys[i])!=15 &&
atoi(keys[i]) != 111020)
  {
   Serial.print("GPIO: ");
   Serial.print(keys[i]);
   Serial.print(" - SET to: ");
   Serial.println(value);
   pinMode(atoi(keys[i]), OUTPUT);
   digitalWrite(atoi(keys[i]), value_check);
  }
  else{
   if (atoi(keys[i])==13)
   {
    dutyCycle1=value_check;
   }
   else if(atoi(keys[i])==14){
```

```
        dutyCycle2=value_check;
       }
       else{
        dutyCycle3=value_check;
       }
      }
    }
}

/* Functions is used to process pwm */
void pwm_process(){
   /* configure LED PWM functionalitites */
  ledcSetup(ledChannel1, freq, resolution);
  ledcSetup(ledChannel2, freq, resolution);
  ledcSetup(ledChannel3, freq, resolution);

  /* attach the channel to the GPIO to be controlled */
  ledcAttachPin(ledPin1, ledChannel1);
  ledcAttachPin(ledPin2, ledChannel2);
  ledcAttachPin(ledPin3, ledChannel3);

  ledcWrite(ledChannel1, dutyCycle1);
  ledcWrite(ledChannel2, dutyCycle2);
  ledcWrite(ledChannel3, dutyCycle3);
}

/* SENSOR'S FUNCTIONS */
/* Functions is used to read gas data from MQ2 module*/
void read_MQ2(){
  gas=analogRead(gasPin);
  gas=map(gas,0,4095,0,100);
  Serial.println(gas);
  if(gas>85)
  {
```

```
    digitalWrite(speakerPin,LOW);
    digitalWrite(fanOutPin,HIGH);
   }
   else{
    digitalWrite(speakerPin,HIGH);
    digitalWrite(fanOutPin,LOW);
   }
}

/* Functions is used to read temperature and humidity data from DHT11 module*/
void read_DHT11() {
  humd = dht.readHumidity();
  temp = dht.readTemperature();
  while (isnan(humd) || isnan(temp)) {
    dht.begin();
    humd = dht.readHumidity();
    temp = dht.readTemperature();
  }
  Serial.print(F("Humidity: "));
  Serial.print(humd);
  Serial.print(F("%  Temperature: "));
  Serial.print(temp);
  Serial.println(F("°C "));
}

/* Function is used to read light data using BH1750 */
void read_BH1750(){
  lux = lightMeter.readLightLevel();
}

/* Functions is used to send data to server */
void send_data(){
  postData = (String)"idpkey=" + idpkey + "&keyroom=" + keyroom +
"&boardDevice=" + boardDevice + "&ssid=" + ssid + "&version_now=" + version +
```

```
"&temp=" + temp + "&humd=" + humd + "&gas=" + gas + "&lux=" + lux + "&people="
+ people;
    HTTPClient http;
    http.begin(serverNameSendData);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    auto httpCode = http.POST(postData);
    String payload = http.getString();
    http.end();
  }

  /* Function is used to get data from server */
  String httpGETRequestDevice(const char* serverNameGetData){
    HTTPClient http;
    http.begin(serverNameGetData);
    int httpResponseCode = http.GET();
    String payload = "{}";
    while (httpResponseCode <= 0)
    {
      http.begin(serverNameGetData);
      httpResponseCode = http.GET();
    }
    if (httpResponseCode > 0)
    {
      payload = http.getString();
    }
    else{
      Serial.print("Erorr code: ");
      Serial.println(httpResponseCode);
    }
    http.end();
    return payload;
  }
```