

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

-----o0o-----



BÁO CÁO ĐỒ ÁN MÔN HỌC 2

**Mô Hình Đơn Giản Của Nhà Kính
Thông Minh Điều Khiển Qua WebSite**

GVHD: Th.S Trần Hoàng Quân

SVTH: Trần Anh Tân

MSSV: 4133561

TPHCM, 10/06/2017

GVHD: Th.S Trần Hoàng Quân

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

-----o0o-----



BÁO CÁO ĐỒ ÁN MÔN HỌC 2

**Mô Hình Đơn Giản Của Nhà Kính
Thông Minh Điều Khiển Qua WebSite**

GVHD: Th.S Trần Hoàng Quân

SVTH: Trần Anh Tân

MSSV: 4133561

TPHCM, 10/06/2017

LỜI CẢM ƠN

Đầu tiên, em xin chân thành cảm ơn thầy Trần Hoàng Quân, người đã giúp đỡ, hướng dẫn em tận tình trong suốt quá trình thực hiện đồ án môn học 2.

Để đạt được đến ngày hôm nay, không thể nhắc đến sự dạy bảo, hướng dẫn tận tình của các thầy, cô của Trường Đại học Bách khoa TP. HCM, đặc biệt là các thầy cô trong khoa Điện – Điện tử đã giúp em có được các kiến thức để thực hiện đồ án môn học 2 này.

Cuối cùng, em xin được cảm ơn tất cả bạn bè, những người đã bên cạnh động viên, giúp đỡ em trong suốt quá trình học tập tại trường và thời gian thực hiện đồ án môn học 2.

Tp. Hồ Chí Minh, ngày 10 tháng 06 năm 2017.

Trần Anh Tân

TÓM TẮT ĐỒ ÁN

Đồ án này trình bày về hệ thống điều khiển và giám sát từ xa qua mạng Internet ứng dụng công nghệ Wifi kết hợp internet. Hệ thống có thể giúp cho người sử dụng có thể điều khiển các thiết bị cũng như giám sát trạng thái các thiết bị từ xa trên các thiết bị có thể kết nối Internet như máy vi tính, điện thoại di động, ...

Nguyên lý hoạt động cụ thể của hệ thống này như sau:

- ESP8266v01 kết nối internet thông qua wifi đọc dữ liệu từ cơ sở dữ liệu trên webserver xử lý dữ liệu và gửi về dữ liệu về vi xử lý.
- Dữ liệu nhận được từ ESP dưới dạng text được vi điều khiển xử lý và thực hiện nhiệm vụ và gửi lại kết quả dưới dạng text cho ESP.
- ESP nhận dữ liệu từ vi xử lý rồi gửi lại kết quả lên cơ sở dữ liệu.
- Trang web sẽ có nhiệm vụ hiển thị trạng thái của các thiết bị và thời gian thay đổi trạng thái của các thiết bị. Người dùng cũng sẽ điều khiển các thiết bị thông qua các lựa chọn trên trang web này.

MỤC LỤC

1. GIỚI THIỆU	1
1.1. Tổng quan.....	1
1.1.1. Những xu hướng thống trị thế giới công nghệ thời gian qua	1
1.1.2. Đây sẽ là xu hướng công nghệ của tương lai?	1
1.2. Nhiệm vụ đồ án.....	2
2. LÝ THUYẾT.....	3
2.1. Giới thiệu wifi	3
2.1.1. Sự ra đời của wifi.....	3
2.1.1.1. Sự khởi đầu	3
2.1.1.2. Hợp nhất tiêu chí	3
2.1.1.3. Tìm một tên gọi phù hợp	4
2.1.1.4. Đi vào cuộc sống.....	4
2.2. Tìm hiểu WIFI	5
2.2.1. Khái niệm WIFI	5
2.2.2. Các thành phần của mạng wifi.....	5
2.2.2.1. Antenna.....	6
2.2.2.2. Wireless Access Point	6
2.2.2.3. Wireless End-user device (Wireless Adapter Card).....	7
2.2.3. Nguyên tắc hoạt động:.....	7
2.3. Mạng LAN	8
2.3.1. Các mô hình WLAN:	8
2.3.1.1. Mô hình mạng AD_HOC (Independent Basic Service sets (IBSSs)).....	8
2.3.1.2. Mô hình mạng cơ sở (Basic service sets (BSSs)).....	8
2.3.1.3. Mô hình mạng mở rộng (Extended Service Set (ESSs))	9
2.3.2. Bảo mật mạng không dây	9

2.4. Các chuẩn hóa.....	10
2.4.1. WEP (Wire Equivalent Privacy).....	10
2.4.2. WPA (Wi-fi Protected Access).....	10
2.4.3. WPA2	11
2.5. IP Tĩnh và IP Động	11
2.5.1. Địa chỉ IP là gì?.....	11
2.5.2. Cấu trúc một địa chỉ IP.....	11
2.5.3. Tìm hiểu về IP tĩnh và IP động.....	11
2.5.3.1. IP tĩnh là gì?	11
2.5.3.2. IP động là gì?	11
2.6. NAT (Network Address Translation)	12
2.6.1. NAT (Network Address Translation) là gì?	12
2.6.2. NAT (Network Address Translation) làm nhiệm vụ gì?.....	13
2.6.3. Giới thiệu chung về NAT	13
2.6.3.1. Static NAT (NAT tĩnh).....	13
2.6.3.2. Dynamic NAT (NAT động).....	13
2.6.4. Thay đổi cấu hình dynamic NAT.....	14
2.6.4.1. Overloading NAT.....	14
2.6.4.2. Overlapping NAT.....	15
2.7. Ưu – Nhược điểm của wifi.....	15
2.7.1. Ưu điểm:	15
2.7.2. Nhược điểm:	16
2.8. Giới thiệu ESP8266	16
2.8.1. Giới thiệu ESP8266.....	16
2.8.1.1. Thông số kỹ thuật:.....	17
2.8.1.2. Các lệnh AT chung.....	17
2.8.1.3. Các lệnh AT cấu hình Module Wifi	17

2.8.1.4. Các lệnh AT đối với Module Wifi cấu hình là Station / client .	18
2.8.1.5. Các lệnh AT với Module Wifi cấu hình là Access Point	18
2.8.1.6. Sơ đồ chân của ESP8266 V01	19
2.8.1.7. Chân kết nối:	19
2.8.2. Vấn đề lập trình trên ESP8266 V01	19
2.8.2.1. Lập Trình ESP8266 V1 trên IDE arduino	19
2.8.2.2. Thư viện ESP8266WiFi.....	19
2.8.2.3. WiFi Access Point.....	20
2.8.2.4. Thiết lập mạng.....	21
2.8.2.5. softAP	21
2.8.2.6. Thư Viện ESP8266HTTPClient.h.....	22
2.8.3. GIỚI THIỆU VỀ ARDUINO VÀ ATMEGA328	22
2.8.3.1. Một vài thông số của Arduino UNO R3	23
2.8.3.2. IC Atmega328	23
2.8.4. MỘT SỐ LINH KIỆN KHÁC	25
2.8.4.1. Rơ-le	25
2.8.4.2. Cảm biến độ ẩm đất:.....	26
2.8.4.3. Động cơ bước.....	27
2.8.4.4. Cảm biến ánh sáng	29
3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG	31
3.1. Yêu cầu thiết kế phần cứng	31
3.2. Sơ đồ tổng quát.....	32
3.3. Thiết kế khối Relay.....	32
3.4. Sơ đồ toàn mạch	34
4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM.....	35
4.1. Yêu cầu thiết kế	35
4.2. Phần mềm cho vi điều khiển và ESP8266	35

4.2.1. Phần mềm dành cho ATMEGA328.....	35
4.2.2. Phần mềm cho ESP8266.....	38
4.3. Trang Web và phía Server.....	42
4.3.1. Phần mềm Microsoft WebMatrix và ASP.NET framework	42
4.3.2. Tạo cơ sở dữ liệu.....	43
4.3.3. Trang hiển thị chính của trang Web.....	45
4.3.4. Trang hiển thị trạng thái cảm biến và thiết bị.....	47
4.3.5. Trang Web phía Server.....	48
4.3.6. Trang hiển thị yêu cầu cho thiết bị.....	50
5. KẾT QUẢ THỰC HIỆN	51
5.1. Đăng tải trang Web lên mạng Internet.....	51
5.2. Phần mềm của vi điều khiển.....	52
5.3. Hoạt động của toàn hệ thống.....	52
6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	55
6.1. Kết luận	55
6.2. Hướng phát triển.....	55
7. TÀI LIỆU THAM KHẢO.....	56
8. PHỤ LỤC.....	57
8.1. Code cho vi điều khiển ATMEGA328:	57
8.2. Code ESP8266.....	64
8.3. Mã lệnh của trang “Default.cshtml”	77
8.4. Mã lệnh của trang “Sensor.cshtml”	80
8.5. Mã lệnh trang SendRequest.cshtml	81
8.6. Mã lệnh trang GetStatus.cshtml.....	82
8.7. Mã lệnh trang DisplayStatus.cshtml	83
8.8. Mã lệnh trang DisplaySensor.cshtml	84

DANH SÁCH HÌNH MINH HỌA

Hình 1.1 Internet of Things là những vật dụng có khả năng kết nối với nhau thông qua Internet.	2
Hình 2.1 Mô hình hoạt động của Access Point.....	6
Hình 2.2. Mô hình hoạt động của Access Point.....	6
Hình 2.3. Các lớp thiết lập bảo mật.....	10
Hình 2.4. Mô hình IP.....	14
Hình 2.5. Các chân ESP.....	19
Hình 2.6. Mô hình kết nối wifi.....	21
Hình 2.7. Hình sơ đồ chân ATMEGA328.....	24
Hình 2.8. Hình thực tế ATMEGA328.....	25
Hình 2.9. Hình Một module relay kiểu mẫu.....	26
Hình 2.10. Relay.....	26
Hình 2.11. Cảm biến độ ẩm.	27
Hình 2.12. Động cơ bước.....	28
Hình 2.13. Sơ đồ kết nối dây.....	28
Hình 3.1. Cảm biến ánh sáng.....	29
Hình 4.1. Sơ đồ khối phần cứng.....	32
Hình 4.2. Sơ đồ khối Relay.....	33
Hình 4.3. Sơ đồ mạch chính.....	34
Hình 5.1. Sơ đồ hệ thống.....	36
Hình 5.2 Sơ đồ giải thuật.....	36
Hình 5.3. Sơ đồ giải thuật của hàm xử lý chuỗi.....	37
Hình 5.4. Chương trình Microsoft WebMatrix 3.....	42
Hình 5.5. Bảng Devices của cơ sở dữ liệu.....	44
Hình 5.6. Bảng Sensor của cơ sở dữ liệu.....	44

Hình 5.7. Bảng DLog của cơ sở dữ liệu	44
Hình 5.8. Bảng SLog của cơ sở dữ liệu	45
Hình 5.9. Trang hiển thị chính của trang Web.....	45
Hình 6.1. Trang Web sau khi được đăng tải lên mạng	51
Hình 6.2. Phần mềm IDE.....	52
Hình 6.3. Tên truy cập wifi.....	52
Hình 6.4. websever offline.....	53
Hình 6.5. Phần cứng khi hoạt động.	53
Hình 6.6. Dữ liệu web nhận được.	54
Hình 6.7. Đối chiếu nhiệt độ.	54

DANH SÁCH BẢNG TÍNH

Bảng 2.1. Tập lệnh AT chung	17
Bảng 2.2. Các lệnh AT cấu hình Module Wifi	17
Bảng 2.3. Các lệnh AT cấu hình Staition	18
Bảng 2.4. Các lệnh AT với Module Wifi cấu hình là Access Point.....	18
Bảng 2.5. Một vài thông số của Arduino UNO R3	23

1. GIỚI THIỆU

1.1. Tổng quan

1.1.1. Những xu hướng thống trị thế giới công nghệ thời gian qua

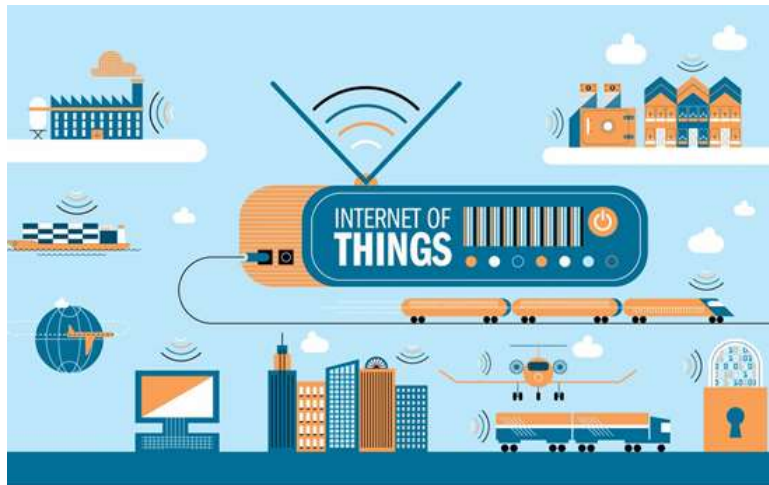
Microsoft, Google, Facebook là những cái tên không còn xa lạ với những người dùng Internet hiện nay. Tất cả những “ông trùm” về công nghệ này đều có một điểm chung: Họ có tầm nhìn rất xa để biến những ý tưởng dù đơn giản trở thành nền tảng mà hàng tỷ người dùng Internet phải sử dụng.

Với tầm nhìn xa trông rộng Bill Gates đã “thấy trước” tương lai rằng máy tính cá nhân sẽ bao phủ khắp mọi gia đình. Microsoft sau đó còn cho ra mắt hệ điều hành Windows, nền tảng thống trị hoàn toàn thị trường máy tính cá nhân kể từ khi ra đời đến nay. Google của anh em nhà Page cũng là một ví dụ tương tự. Chỉ với một ý tưởng tạo ra nền tảng giúp mọi người có thể tìm kiếm dễ dàng hơn, họ đã tạo ra cả một “đế chế” Google thống trị mạng Internet ngày nay. Hay gần đây nhất, Mark Zuckerberg, một trong những nhà sáng lập mạng xã hội Facebook, cũng tạo ra “đế chế” tỷ đô chỉ với một ý tưởng đó là làm sao để mọi người kết nối được với nhau tốt hơn trên Internet.

Họ đều là những người biết tạo nên nền tảng phù hợp nhu cầu thị trường và mở ra xu hướng mới cho người tiêu dùng.

1.1.2. Đâu sẽ là xu hướng công nghệ của tương lai?

Với sự phát triển của Internet, smartphone và đặc biệt là các thiết bị cảm biến, Internet of Things (IoT) đang trở thành xu hướng mới của thế giới. IoT được định nghĩa là những vật dụng có khả năng kết nối Internet. Bạn vào nhà, mở khóa cửa, đèn sẽ tự động sáng chỗ bạn đứng, điều hòa sẽ tự động điều chỉnh nhiệt độ, nhạc sẽ tự động bật để chào đón bạn... những điều chỉ có trong phim khoa học viễn tưởng, đang dần trở thành hiện thực với công nghệ IoT.



Hình 1.1 Internet of Things là những vật dụng có khả năng kết nối với nhau thông qua Internet.

Các thiết bị IoT được vận hành nhờ những bộ vi xử lý SOC bên trong. Không như những bộ vi xử lý thông thường, SOC giống như một máy tính trọn vẹn được thu gọn trong diện tích của một con chip điện tử, có kết nối không dây và đảm bảo tiết kiệm điện. Dù nhỏ gọn, sức mạnh của các vi xử lý SOC là không phải bàn cãi khi nó hoàn toàn có thể vận hành trơn tru những hệ điều hành nặng nề như Windows hay Linux. SOC rất phổ biến trong bên trong các linh kiện điện thoại.

Theo dự báo của IDC, thị trường IoT được dự báo sẽ tăng gấp 3 lần, đạt 1,7 nghìn tỉ USD vào năm 2020. Không ít các doanh nghiệp lớn đã nhìn thấy tiềm năng của IoT và mạnh dạn đầu tư vào đây. Tuy nhiên, cũng giống như bất kỳ một công nghệ mới nào, IoT sẽ cần một nền tảng để vận hành. Và các doanh nghiệp công nghệ hiểu rằng, ai tạo ra được nền tảng dẫn đầu, họ sẽ là người chiến thắng trong xu hướng mới này.

Do đó việc nghiên cứu và phát triển các ứng dụng kết nối wifi là hết sức quan trọng.

1.2. Nhiệm vụ đồ án

Nội dung 1: Tìm hiểu về mạng Wifi và internet

Nội dung 2: Tìm hiểu về Esp8266 v1 nguyên lý hoạt động, cách kết nối wifi và kết nối với mạng internet.

Nội dung 3: Tìm hiểu về Kit vi điều khiển Arduino.

Nội dung 5: Thiết kế và thực hiện mạch Relay dùng để điều khiển các thiết bị điện.

Nội dung 6: Thiết kế trang web và phần server để liên kết với cơ sở dữ liệu.

Nội dung 7: Viết phần mềm cho vi điều khiển và hoàn thành kết nối hệ thống.

2. LÝ THUYẾT

2.1. *Giới thiệu wifi*

2.1.1. *Sự ra đời của wifi*

2.1.1.1. *Sự khởi đầu*

Năm 1985, Ủy ban liên lạc liên bang Mỹ FCC (cơ quan quản lý viễn thông của nước này), quyết định “mở cửa” một số băng tần của dải sóng không dây, cho phép sử dụng chúng mà không cần giấy phép của chính phủ. Đây là một điều khá bất thường vào thời điểm đó. Song, trước sự thuyết phục của các chuyên viên kỹ thuật, FCC đã đồng ý “thả” 3 dải sóng công nghiệp, khoa học và y tế cho giới kinh doanh viễn thông.

Ba dải sóng này, gọi là các “băng tần rác” (900 MHz, 2,4 GHz, 5,8 GHz), được phân bổ cho các thiết bị sử dụng vào các mục đích ngoài liên lạc, chẳng hạn như lò nướng vi sóng sử dụng các sóng vô tuyến radio để đun nóng thức ăn. FCC đã đưa các băng tần này vào phục vụ mục đích liên lạc dựa trên cơ sở: bất cứ thiết bị nào sử dụng những dải sóng đó đều phải đi vòng để tránh ảnh hưởng của việc truy cập từ các thiết bị khác. Điều này được thực hiện bằng công nghệ gọi là phổ rộng (vốn được phát triển cho quân đội Mỹ sử dụng), có khả năng phát tín hiệu radio qua một vùng nhiều tần số, khác với phương pháp truyền thống là truyền trên một tần số đơn lẻ được xác định rõ.

2.1.1.2. *Hợp nhất tiêu chí*

Dấu mốc quan trọng cho Wi-Fi diễn ra vào năm 1985 khi tiến trình đi đến một chuẩn chung được khởi động. Trước đó, các nhà cung cấp thiết bị không dây dùng cho mạng LAN như Proxim và Symbol ở Mỹ đều phát triển những thiết sản phẩm độc quyền, tức là thiết bị của hãng này không thể liên lạc được với của hãng khác. Nhờ sự thành công của mạng hữu tuyến Ethernet, một số công ty bắt đầu nhận ra rằng việc xác lập một chuẩn không dây chung là rất quan trọng. Vì người tiêu dùng khi đó sẽ dễ dàng chấp nhận công nghệ mới nếu họ không còn bị bó hẹp trong sản phẩm và dịch vụ của một hãng cụ thể.

Năm 1988, công ty NCR, vì muốn sử dụng dải tần “rác” để liên thông các máy rút tiền qua kết nối không dây, đã yêu cầu một kỹ sư của họ có tên Victor Hayes tìm hiểu việc thiết lập chuẩn chung. Ông này cùng với chuyên gia Bruce Tuch của Trung tâm nghiên cứu Bell Labs đã tiếp cận với Tổ chức kỹ sư điện và điện tử IEEE, nơi mà một tiểu ban có tên 802.3 đã xác lập ra chuẩn mạng cục bộ Ethernet phổ biến hiện nay. Một tiểu ban mới có tên 802.11 đã ra đời và quá trình thương lượng hợp nhất các chuẩn bắt đầu.

Thị trường phân tán ở thời điểm đó đồng nghĩa với việc phải mất khá nhiều thời gian để các nhà cung cấp sản phẩm khác nhau đồng ý với những định nghĩa chuẩn và đề ra một tiêu chí mới với sự chấp thuận của ít nhất 75% thành viên tiểu ban. Cuối

cùng, năm 1997, tiêu ban này đã phê chuẩn một bộ tiêu chí cơ bản, cho phép mức truyền dữ liệu 2 Mb/giây, sử dụng một trong 2 công nghệ dải tần rộng là frequency hopping (tránh nhiễu bằng cách chuyển đổi liên tục giữa các tần số radio, còn gọi là truyền chéo) hoặc direct-sequence transmission (phát tín hiệu trên một dải gồm nhiều tần số, còn gọi là truyền thẳng).

Chuẩn mới chính thức được ban hành năm 1997 và các kỹ sư ngay lập tức bắt đầu nghiên cứu một thiết bị mẫu tương thích với nó. Sau đó có 2 phiên bản chuẩn, 802.11b (hoạt động trên băng tần 2,4 GHz) và 802.11a (hoạt động trên băng tần 5,8 GHz), lần lượt được phê duyệt tháng 12 năm 1999 và tháng 1 năm 2000. Sau khi có chuẩn 802.11b, các công ty bắt đầu phát triển những thiết bị tương thích với nó. Tuy nhiên, bộ tiêu chí này quá dài và phức tạp với 400 trang tài liệu và vấn đề tương thích vẫn nổi cộm. Vì thế, vào tháng 8/1999, có 6 công ty bao gồm Intersil, 3Com, Nokia, Aironet (về sau được Cisco sáp nhập), Symbol và Lucent liên kết với nhau để tạo ra Liên minh tương thích Ethernet không dây WECA.

2.1.1.3. Tìm một tên gọi phù hợp

Mục tiêu hoạt động của tổ chức WECA là xác nhận sản phẩm của những nhà cung cấp phải tương thích thực sự với nhau. Tuy nhiên, các thuật ngữ như “tương thích WECA” hay “tuân thủ IEEE 802.11b” vẫn gây bối rối đối với cả cộng đồng. Công nghệ mới cần một cách gọi thuận tiện đối với người tiêu dùng. Các chuyên gia tư vấn đề xuất một số cái tên như “FlankSpeed” hay “DragonFly”. Nhưng cuối cùng được chấp nhận lại là cách gọi “Wi-Fi” vì nghe vừa có vẻ công nghệ chất lượng cao (hi-fi) và hơn nữa người tiêu dùng vốn quen với kiểu khái niệm như đầu đĩa CD của công ty nào thì cũng đều tương thích với bộ khuếch đại amplifier của hãng khác. Thế là cái tên Wi-Fi ra đời. Cách giải thích “Wi-Fi có nghĩa là wireless fidelity” về sau này người ta mới nghĩ ra. Gần đây, nhiều chuyên gia cũng đã viết bài khẳng định lại Wi-Fi thực ra chỉ là một cái tên đặt ra cho dễ gọi chứ chả có nghĩa gì ban đầu.

2.1.1.4. Đi vào cuộc sống

Như vậy là công nghệ kết nối cục bộ không dây đã được chuẩn hóa, có tên thống nhất và đã đến lúc cần một nhà vô địch để thúc đẩy nó trên thị trường. Wi-Fi đã tìm được Apple, nhà sản xuất máy tính nổi tiếng với những phát minh cấp tiến. “Quả táo” tuyên bố nếu hãng Lucent có thể sản xuất một bộ điều hợp adapter với giá chưa đầy 100 USD thì họ có thể tích hợp một khe cắm Wi-Fi vào mọi chiếc máy tính xách tay. Lucent đáp ứng được điều này và vào tháng 7/1999, Apple công bố sự xuất hiện của Wi-Fi như một sự lựa chọn trên dòng máy iBook mới của họ, sử dụng thương hiệu AirPort. Điều này đã hoàn toàn làm thay đổi thị trường mạng không dây. Các nhà sản xuất máy tính khác lập tức ồ ạt làm theo. Wi-Fi nhanh chóng tiếp cận với người tiêu dùng gia đình trong bối cảnh chi tiêu cho công nghệ ở các doanh nghiệp đang bị hạn chế năm 2001.

Wi-Fi sau đó tiếp tục được thúc đẩy nhờ sự phổ biến mạnh mẽ của kết nối Internet băng rộng tốc độ cao trong các hộ gia đình và trở thành phương thức dễ nhất để cho phép nhiều máy tính chia sẻ một đường truy cập băng rộng. Khi công nghệ này phát triển rộng hơn, các điểm truy cập thu phí gọi là hotspot cũng bắt đầu xuất hiện ngày một nhiều ở nơi công cộng như cửa hàng, khách sạn, các quán café. Trong khi đó, Ủy ban liên lạc liên bang Mỹ FCC một lần nữa thay đổi các quy định của họ để cho phép một phiên bản mới của Wi-Fi có tên 802.11g ra đời, sử dụng kỹ thuật dải phổ rộng tiên tiến hơn gọi là truy cập đa phân tần trực giao OFDM (orthogonal frequency-division multiplexing - còn gọi là ghép kênh chia tần số trực giao) và có thể đạt tốc độ lên tới 54 Mb/giây ở băng tần 2,4 Ghz.

2.2. *Tìm hiểu WIFI*

2.2.1. **Khái niệm WIFI**

Wi-Fi được viết tắt từ Wireless Fidelity (không dây trung thực). Thực chất nó có tên là The Standard for Wireless Fidelity (chuẩn cho không dây trung thực).

WiFi là tên gọi phổ thông của mạng không dây theo công nghệ WLAN (Wireless Local Area Network) là một loại mạng máy tính nhưng việc kết nối giữa các thành phần trong mạng không sử dụng các loại cáp như một mạng thông thường, môi trường truyền thông của các thành phần trong mạng là không khí. Các thành phần trong mạng sử dụng sóng điện từ để truyền thông với nhau cho phép người sử dụng nối mạng trong phạm vi phủ sóng của các điểm kết nối trung tâm. Phương thức kết nối này từ khi ra đời đã mở ra cho người sử dụng sự lựa chọn tối ưu, bổ sung cho các phương thức kết nối truyền thống dùng dây.

WiFi hiện nay được sử dụng cho hàng loạt các dịch vụ như internet, điện thoại internet, máy chơi game và cả các đồ điện tử như TV, đầu đọc DVD và máy ảnh số. Ứng dụng phổ thông nhất của WiFi là kết nối Internet bằng các thiết bị cầm tay như máy tính xách tay, sổ tay điện tử PDA, các điện thoại tích hợp WiFi...

2.2.2. **Các thành phần của mạng wifi**

Một mạng wireless gồm các thành phần sau:

2.2.2.1. Antenna



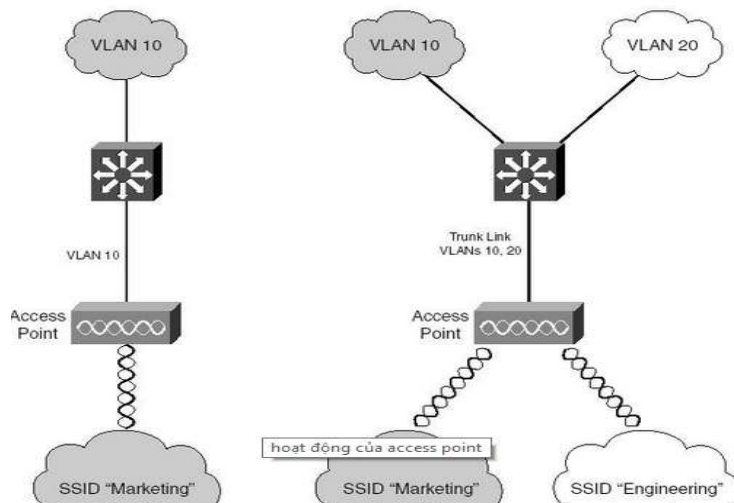
Hình 2.1 Mô hình hoạt động của Access Point

Antenna chính là thiết bị thu phát sóng điện từ và có các đặc điểm cơ bản sau:

- Antenna phát sẽ chuyển năng lượng điện thành sóng điện từ và phát ra ngoài, ngược lại anten sẽ chuyển sóng điện từ thu được thành năng lượng điện. Một anten về cơ bản bao gồm một bộ bức xạ và một nguyên tố anten (antenna element).
- Kích thước vật lý của anten (chẳng hạn như chiều dài của anten) liên quan trực tiếp đến tần số hoạt động của anten.

2.2.2.2. Wireless Access Point

Là 1 thiết bị ngoại vi dùng sóng để thu phát tín hiệu, truyền tải thông tin giữa các thiết bị wireless và mạng dùng dây. Trên thị trường phổ biến là các AP chuẩn B (11 Mb/s), và G (54Mb/s), gần đây xuất hiện Super G sử dụng công nghệ MIMO (Multi Input-Multi Output) có thể truyền file với tốc độ 108Mb/s. AP cung cấp cho client một



Hình 2.2. Mô hình hoạt động của Access Point

điểm truy cập vào mạng. Hình dưới đây mô tả AP và nơi sử dụng chúng trong mạng WLAN.

2.2.2.3. Wireless End-user device (Wireless Adapter Card)

Được hiểu như những thành phần mà AP coi là client trong mạng Wireless.

Gồm có:

- PCMCIA (Personal Computer Memory Card International Association) : Có thể sử dụng cho các máy tính xách tay (MTXT) thuộc đời cũ, không tích hợp sẵn chức năng wifi. Đây là chuẩn chân cắm có sẵn ở hầu hết MTXT. Card wireless PCMCIA sẽ được gắn trực tiếp lên mainboard nên nó phải nằm cố định ở trong đó nếu bạn không muốn phải mở máy ra. Card loại này bắt sóng khá tốt, có thể nói là không thua kém card mà nhà sản xuất tích hợp sẵn trong máy.

- Compact flash Cards (CF): Là một loại flash memory (bộ nhớ flash) thường được dùng cho máy ảnh số (digital camera), ĐTDĐ và các thiết bị di động kỹ thuật số khác. Được sản xuất lần đầu tiên vào năm 1994 từ SanDisk. Nó là một miếng plastic hình chữ nhật, bao gồm 2 loại: Compact Flash I dày 3mm, Compact Flash II dày 5 mm. Compact Flash I cũng có thể được sử dụng trong khe cắm Compact Flash II.

- Ethernet và Serial converters: được sử dụng với tất cả thiết bị ethernet hoặc có cổng serial 9 chân. Thường được sử dụng cho Print Server khi kết nối vào mạng wireless

- USB Adapter: Card loại này có hình dáng và kích thước giống như một chiếc USB flash mà chúng ta hay dùng để lưu dữ liệu, có thể dùng cho cả PC và MTXT do các máy tính hiện nay đều có sẵn cổng này. Ưu điểm của card cổng USB là dễ dàng cắm vào, tháo ra hoặc nối dài thông qua dây nối có đầu USB.

- PCI, PCI Express: phù hợp dùng cho máy tính để bàn (PC) do PC thường có sẵn các cổng này trên mainboard (có thể dùng để cắm card wireless, card VGA, sound card) và không gian đủ rộng. Chuẩn PCI Express được phát triển trên chuẩn PCI nên sẽ cho tốc độ xử lý tốt hơn.

2.2.3. Nguyên tắc hoạt động:

Sóng vô tuyến được truyền từ các anten và các router và sẽ được nhận bởi các bộ nhận như máy tính, điện thoại di động được trang bị card Wi-Fi... Khi các thiết bị này nhận được tín hiệu thì các card Wi-Fi sẽ đọc tín hiệu và tạo kết nối không dây. Một khi một kết nối được thiết lập giữa người dùng và mạng thì người dùng và mạng thì người dùng sẽ được nhắc nhở bằng một màn hình login và password nếu như đó là mạng thuê.

Vùng phủ sóng bởi 1 hay nhiều AP (access point). Một AP có phạm vi khoảng từ 1 căn phòng đến vài dặm, Trên thế giới thì các AP này được đặt ở các thành phố để

mọi người với laptop có thể truy cập Internet, AP có ở khắp nơi như trong nhà hàng, khách sạn, trường học, sân bay...

2.3. *Mạng LAN*

WLAN là một loại mạng máy tính nhưng việc kết nối giữa các thành phần trong mạng không sử dụng các loại cáp như một mạng thông thường, môi trường truyền thông của các thành phần trong mạng là không khí. Các thành phần trong mạng sử dụng sóng điện từ để truyền thông với nhau.

2.3.1. Các mô hình WLAN:

Các mô hình WLAN: Mạng 802.11 linh hoạt về thiết kế, gồm 3 mô hình mạng sau:

- Mô hình mạng độc lập (IBSSs) hay gọi là mạng Ad hoc
- Mô hình mạng cơ sở (BSSs)
- Mô hình mạng mở rộng (ESSs)

2.3.1.1. *Mô hình mạng AD_HOC (Independent Basic Service sets (IBSSs))*

Các nút di động (máy tính có hỗ trợ card mạng không dây) tập trung lại trong một không gian nhỏ để hình thành nên kết nối ngang cấp (peer-to-peer) giữa chúng. Các nút di động có card mạng wireless là chúng có thể trao đổi thông tin trực tiếp với nhau, không cần phải quản trị mạng. Vì các mạng ad-hoc này có thể thực hiện nhanh và dễ dàng nên chúng thường được thiết lập mà không cần một công cụ hay kỹ năng đặc biệt nào vì vậy nó rất thích hợp để sử dụng trong các hội nghị thương mại hoặc trong các nhóm làm việc tạm thời. Tuy nhiên chúng có thể có những nhược điểm về vùng phủ sóng bị giới hạn, mọi người sử dụng đều phải nghe được lẫn nhau.

2.3.1.2. *Mô hình mạng cơ sở (Basic service sets (BSSs))*

Bao gồm các điểm truy nhập AP (Access Point) gắn với mạng đường trục hữu tuyến và giao tiếp với các thiết bị di động trong vùng phủ sóng của một cell. AP đóng vai trò điều khiển cell và điều khiển lưu lượng tới mạng. Các thiết bị di động không giao tiếp trực tiếp với nhau mà giao tiếp với các AP. Các cell có thể chồng lấn lên nhau khoảng 10-15 % cho phép các trạm di động có thể di chuyển mà không bị mất kết nối vô tuyến và cung cấp vùng phủ sóng với chi phí thấp nhất. Các trạm di động sẽ chọn AP tốt nhất để kết nối. Một điểm truy nhập nằm ở trung tâm có thể điều khiển và phân phối truy nhập cho các nút tranh chấp, cung cấp truy nhập phù hợp với mạng đường trục, ấn định các địa chỉ và các mức ưu tiên, giám sát lưu lượng mạng, quản lý chuyển đi các gói và duy trì theo dõi cấu hình mạng. Tuy nhiên giao thức đa truy nhập tập trung không cho phép các nút di động truyền trực tiếp tới nút khác nằm trong cùng

vùng với điểm truy nhập như trong cấu hình mạng WLAN độc lập. Trong trường hợp này, mỗi gói sẽ phải được phát đi 2 lần (từ nút phát gốc và sau đó là điểm truy nhập) trước khi nó tới nút đích, quá trình này sẽ làm giảm hiệu quả truyền dẫn và tăng trễ truyền dẫn.

2.3.1.3. Mô hình mạng mở rộng (*Extended Service Set (ESSs)*)

Mạng 802.11 mở rộng phạm vi di động tới một phạm vi bất kì thông qua ESS. Một ESSs là một tập hợp các BSSs nơi mà các Access Point giao tiếp với nhau để chuyển lưu lượng từ một BSS này đến một BSS khác để làm cho việc di chuyển dễ dàng của các trạm giữa các BSS, Access Point thực hiện việc giao tiếp thông qua hệ thống phân phối. Hệ thống phân phối là một lớp mỏng trong mỗi Access Point mà nó xác định đích đến cho một lưu lượng được nhận từ một BSS. Hệ thống phân phối được tiếp sóng trở lại một đích trong cùng một BSS, chuyển tiếp trên hệ thống phân phối tới một Access Point khác, hoặc gởi tới một mạng có dây tới đích không nằm trong ESS. Các thông tin nhận bởi Access Point từ hệ thống phân phối được truyền tới BSS sẽ được nhận bởi trạm đích.

2.3.2. Bảo mật mạng không dây

Mạng không dây (hay vô tuyến) sử dụng sóng vô tuyến xuyên qua vật liệu của các tòa nhà và bao phủ không giới hạn ở bên trong một tòa nhà. Sóng vô tuyến có thể xuất hiện trên đường phố, từ các trạm phát từ các mạng LAN này, và như vậy ai cũng có thể truy cập nhờ thiết bị thích hợp. Do đó mạng không dây của một công ty cũng có thể bị truy cập từ bên ngoài tòa nhà công ty của họ.

Để cung cấp mức bảo mật tối thiểu cho mạng WLAN thì ta cần hai thành phần sau:

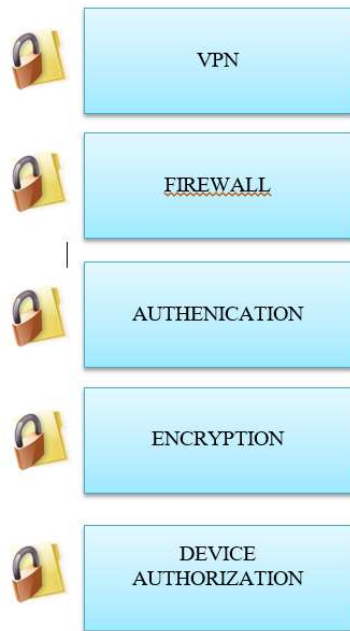
Cách thức để xác định ai có quyền sử dụng WLAN - yêu cầu này được thỏa mãn bằng cơ chế xác thực (authentication).

Một phương thức để cung cấp tính riêng tư cho các dữ liệu không dây – yêu cầu này được thỏa mãn bằng một thuật toán mã hóa (encryption).

Một WLAN gồm có 3 phần: Wireless Client, Access Points và Access Server.

- Wireless Client điển hình là một chiếc laptop với NIC (Network Interface Card) không dây được cài đặt để cho phép truy cập vào mạng không dây.
- Access Points (AP) cung cấp sự bao phủ của sóng vô tuyến trong một vùng nào đó (được biết đến như là các cell (tế bào)) và kết nối đến mạng không dây.
- Còn Access Server điều khiển việc truy cập. Một Access Server (như là Enterprise Access Server (EAS)) cung cấp sự điều khiển, quản lý, các đặc tính bảo mật tiên tiến cho mạng không dây Enterprise.

CÁC THIẾT LẬP BẢO MẬT TRONG MẠNG KHÔNG DÂY



VPN: EAS bao gồm một IPSec VPN server cho phép các Client không dây thiết lập các session VPN vững chắc trên mạng.

Firewall: EAS hợp nhất packet filtering và port blocking firewall dựa trên các chuỗi IP. Việc cấu hình từ trước cho phép các loại lưu lượng chung được enable hay disable.

Authentication hỗ trợ sự ủy quyền lẫn để bảo đảm chỉ có các Client không dây được ủy quyền mới được truy cập vào mạng... Điều này đã tăng tối đa sự bảo mật và giảm tối thiểu các thủ tục hành chính.

Encryption: WLAN cũng hỗ trợ WEP, 3DES và chuẩn TLS (Transport Layer Security) sử dụng mã hóa để tránh người truy cập trộm. Các khóa WEP có thể tạo trên một per-user, per session basis.

Device Authorization: Các Client không dây có thể bị ngăn chặn theo địa chỉ phần cứng của họ (ví dụ như địa chỉ MAC). EAS duy trì một cơ sở dữ liệu của các Client không dây được cho phép và các AP riêng biệt khóa hay lưu thông lưu lượng phù hợp.

Hình 2.3. Các lớp thiết lập bảo mật

2.4. Các chuẩn hóa

2.4.1. WEP (Wire Equivalent Privacy)

- » Có hai loại WEP key – 64 bit và 128 bit. Mỗi loại đều dùng một vector khởi tạo dài 24 bit, theo sau đó là một khóa bí mật dài 40 bit hay 104 bit để tương ứng với hai loại độ dài khác nhau
- » Có 2 dạng WEP Key tĩnh và WEP Key động.
- » So với WEP key tĩnh, cung cấp key tập trung thông qua máy chủ có tính thực tế cao hơn và an toàn hơn trong mạng LAN không dây.
- » WEP dùng giải thuật RC4 để mã hóa dữ liệu và CRC-32 checksum để kiểm tra sự toàn vẹn dữ liệu trên đường truyền. Những điểm yếu này để ngỏ nhiều lỗ hổng khiến cho WEP dễ bị khai thác tấn công.

2.4.2. WPA (Wi-fi Protected Access)

WPA được thiết kế để lấp những lỗ hổng bảo mật của WEP, đặc biệt là quá trình mã hóa dữ liệu và authenticate yếu. Cải tiến mã hóa dữ liệu thông qua TKIP.

TKIP hiện thực chức năng kiểm tra sự toàn vẹn thông điệp (Message Integrity Check – MIC) 64 bit với giải thuật MICHAEL.

2.4.3. WPA2

Thế hệ thứ hai của WPA, dựa trên bản sửa đổi 802.11i của IEEE dành cho chuẩn 802.11. Điểm khác biệt chính giữa hai thế hệ là WPA2 dùng kỹ thuật mã hóa phức tạp hơn là AES (Advanced Encryption Standard), còn gọi là Rijndael. Đây là một giải thuật mã hóa khối được công nhận như là chuẩn mã hóa cho chính phủ Mỹ.

2.5. IP Tĩnh và IP Động

2.5.1. Địa chỉ IP là gì?

Địa chỉ IP là thuật ngữ được viết tắt bởi từ Internet Protocol (giao thức Internet). Địa chỉ IP là địa chỉ đơn nhất, nó giúp các thiết bị điện tử kết nối với nhau trên mạng internet

Nói đơn giản là như thế này, địa chỉ IP nó như địa chỉ nhà của bạn (Số nhà, số ngõ, phường, thành phố...) để mọi người có thể tìm và đến được.

2.5.2. Cấu trúc một địa chỉ IP

Địa chỉ IPv4: IPv4 sử dụng 32 bit để mã hóa dữ liệu theo dạng: EFG.HIJ.KMN.OPQ (ví dụ địa chỉ IP của capquangviettel.vn là: 103.1.208.205). Các số sẽ được người thiết lập mạng hoặc thiết bị Modem liên quan gán cho. Tuy nhiên, do sự phát triển internet quá nhanh nên địa chỉ IPv4 đang dần cạn kiệt, thời gian chỉ tính bằng tháng. Để khắc phục được vấn đề này, các nhà cung cấp dịch vụ đã xây dựng liên giao thức IPv6

Địa chỉ IPv6: được mã hóa 128bit, số lượng địa chỉ IP mà IPv6 có thể cung cấp là con số rất lớn lên tới $(4 \times 10^4)^4$ (4 tỷ mũ 5). Với lượng khổng lồ như vậy, IPv6 có thể đảm bảo cung cấp lượng IP trong thời gian rất dài trên toàn thế giới. Nhưng quá trình triển khai IPv6 cũng đang gặp khó khăn do sai khác về cấu hình cho các thiết bị sử dụng IPv4 trước đây. Hiện tại, ở Việt Nam IPv6 đang thử nghiệm và đưa vào khai thác trong năm nay.

2.5.3. Tìm hiểu về IP tĩnh và IP động

2.5.3.1. IP tĩnh là gì?

IP tĩnh là địa chỉ IP cố định được dành riêng cho một người, hoặc nhóm người sử dụng mà thiết bị kết nối đến Internet của họ luôn luôn được đặt một địa chỉ IP. Thông thường IP tĩnh được cấp cho một máy chủ với một mục đích riêng (máy chủ web, mail...) để nhiều người có thể truy cập mà không làm gián đoạn các quá trình đó.

2.5.3.2. IP động là gì?

Trái với địa chỉ IP tĩnh, IP động sẽ thay đổi khi thiết bị Modem bị khởi động lại. IP động là giải pháp tạm thời khi IPv4 đang rất khan hiếm, nó sẽ luân phiên sử dụng

trên mạng Internet. Với người dùng là cá nhân, hộ gia đình thường sẽ được các nhà cung cấp dịch vụ Internet triển khai IP động.

2.6. NAT (*Network Address Translation*)

Kết nối **Internet** hiện đại ngày nay đều phải sử dụng đến kỹ thuật **NAT (Network Address Translation)**. **NAT (Network Address Translation)** cho phép một hay nhiều **địa chỉ IP** nội miền được ánh xạ với một hay nhiều **địa chỉ IP** ngoại miền. Để hiểu thêm về **NAT** cũng như cơ chế hoạt động của **NAT**, các bạn cùng tham khảo bài viết dưới đây.

Mạng **Internet** ngày càng phát triển hơn so với tưởng tượng của chúng ta. Mặc dù không thể thông kê con số chính xác là bao nhiêu nhưng chúng ta có thể ước tính được con số là hơn 100 triệu Host và hơn 350 triệu người truy cập **Internet** hàng ngày. Trên thực tế, tỉ lệ này tăng gấp đôi theo mỗi năm.

Kết nối **Internet** hiện đại ngày nay đều phải sử dụng đến kỹ thuật **NAT (Network Address Translation)**. **NAT** cho phép một (hay nhiều) **địa chỉ IP** nội miền được ánh xạ với một (hay nhiều) **địa chỉ IP** ngoại miền.

Địa chỉ IP (IP – Internet Protocol) là chuỗi số có chiều dài 32 bit (**IPv4**) hoặc 128 bit (**IPv6**) dùng để định danh một thiết bị mạng trên hệ thống mạng giúp chúng nhận diện và liên lạc với nhau. Trong một mô hình mạng, mỗi một thiết bị mạng chỉ có một **địa chỉ IP** duy nhất. Có thể hiểu nôm na **địa chỉ IP** giống như địa chỉ nơi bạn sinh sống. Người khác có thể tìm ra bạn và gửi thông tin cho bạn thông qua địa chỉ đó.

Cùng với sự bùng nổ **Internet** như hiện nay và nhu cầu sử dụng hệ thống mạng ngày càng gia tăng, không gian **địa chỉ IPv4** bắt đầu bị giới hạn. Giải pháp đưa ra là thiết kế lại định dạng **địa chỉ IP**, cho phép nhiều **địa chỉ IP** hơn nữa (cụ thể là **IPv6**). Tuy nhiên giải pháp này vẫn đang trong giai đoạn nghiên cứu và phát triển và phải mất nhiều năm để thực hiện.

Do đó giải pháp tốt nhất là sử dụng đến kỹ thuật **NAT (Network Address Translation)**. **NAT** hay **Network Address Translation** cho phép một thiết bị như **Router** hoạt động như một người đại diện trung gian giữa **Internet** (hoặc **Public Network**: hệ thống mạng công cộng) và **Local** (hoặc **Private**: hệ thống mạng nội bộ). Điều này có nghĩa là một máy tính chỉ có một **địa chỉ IP** duy nhất.

2.6.1. NAT (*Network Address Translation*) là gì?

Hiểu nôm na, **NAT** cũng giống như một nhân viên lễ tân tại một văn phòng lớn. Nếu bạn muốn gặp một ai đó trong công ty đều phải thông qua và làm theo hướng dẫn của nhân viên lễ tân. Hoặc nếu bạn muốn gọi điện nói chuyện với một ai đó nhưng người đó không có mặt ở công ty hoặc họ đang bận họp, ... bạn có thể để lại tin nhắn cho lễ tân sau đó họ sẽ chuyển tiếp tin nhắn tới người mà bạn cần nói chuyện để thông

báo. Trong một trường hợp khác bạn có thể nói chuyện với lễ tân và yêu cầu họ nối máy đến người bạn cần gặp.

Hay có thể hiểu khi một người muốn nói chuyện với bạn, nhưng họ chỉ biết số điện thoại văn phòng nơi bạn làm việc. Họ sẽ gọi điện đến văn phòng của bạn và yêu cầu nhân viên lễ tân chuyển tiếp cuộc gọi đến bạn. Lúc này nhân viên lễ tân sẽ tiến hành kiểm tra trên bảng tra cứu để tìm ra tên của bạn và các thông tin mở rộng khác. Và sau đó họ sẽ chuyển tiếp cuộc gọi đến cho bạn trên phần mở rộng của bạn.

2.6.2. NAT (Network Address Translation) làm nhiệm vụ gì?

NAT (Network Address Translation) giống như một **Router**, chuyển tiếp các gói tin giữa những lớp mạng khác nhau trên một mạng lớn. **NAT** dịch hay thay đổi một hoặc cả hai địa chỉ bên trong một gói tin khi gói tin đó đi qua một **Router**, hay một số thiết bị khác. Thông thường **NAT** thường thay đổi địa chỉ thường là địa chỉ riêng (**IP Private**) của một kết nối mạng thành địa chỉ công cộng (**IP Public**).

NAT cũng có thể coi như một **Firewall** (tường lửa) cơ bản. **NAT** duy trì một bảng thông tin về mỗi gói tin được gửi qua. Khi một máy tính trên mạng kết nối đến 1 website trên **Internet** header của **địa chỉ IP nguồn** được thay thế bằng địa chỉ **Public** đã được cấu hình sẵn trên **NAT sever**, sau khi có gói tin trở về **NAT** dựa vào bảng record mà nó đã lưu về các gói tin, thay đổi **địa chỉ IP đích** thành địa chỉ của **PC** trong mạng và chuyển tiếp đi. Thông qua cơ chế đó quản trị mạng có khả năng lọc các gói tin được gửi đến hay gửi từ một **địa chỉ IP** và cho phép hay ngăn truy cập đến một port cụ thể.

2.6.3. Giới thiệu chung về NAT

NAT (Network Address Translation) được phát triển bởi **Cisco**. **NAT** bao gồm một số loại cơ bản dưới đây:

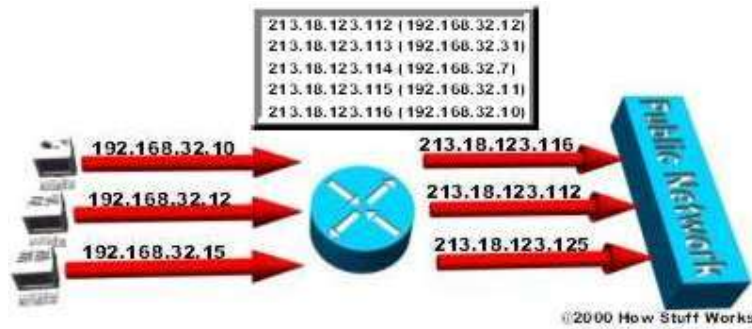
2.6.3.1. Static NAT (NAT tĩnh)

Static NAT (NAT tĩnh) là phương thức NAT một đôi một. Một địa chỉ IP Private sẽ được map với một địa chỉ IP Public. NAT tĩnh được sử dụng khi thiết bị cần truy cập từ bên ngoài mạng.

Trong **Static NAT (NAT tĩnh)**, **địa chỉ IP** của máy tính là 192.168.32.10 luôn được **Router** biên dịch đến **địa chỉ IP** 213.18.123.110.

2.6.3.2. Dynamic NAT (NAT động)

Một **địa chỉ IP Private** sẽ được map với một **địa chỉ IP Public** trong nhóm **địa chỉ IP Public**.



Hình 2.4. Mô hình IP

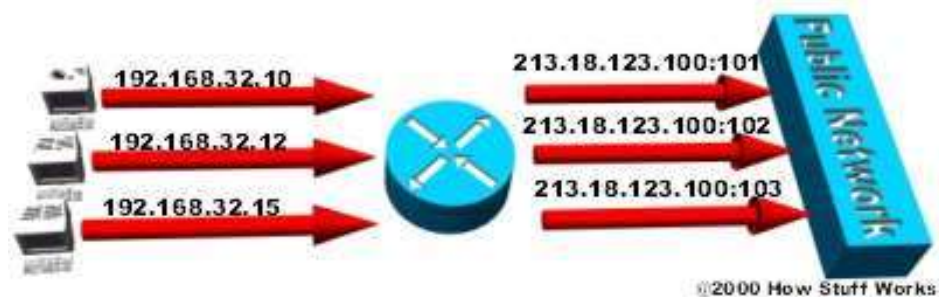
Trong **Dynamic NAT (NAT động)**, máy tính có **địa chỉ IP** 192.168.32.10 luôn được **Router** biên dịch đến địa chỉ đầu tiên 213.18.123.100 trong dãy **địa chỉ IP** từ 213.18.123.100 đến 213.18.123.150.

2.6.4. Thay đổi cấu hình dynamic NAT

2.6.4.1. *Overloading NAT*

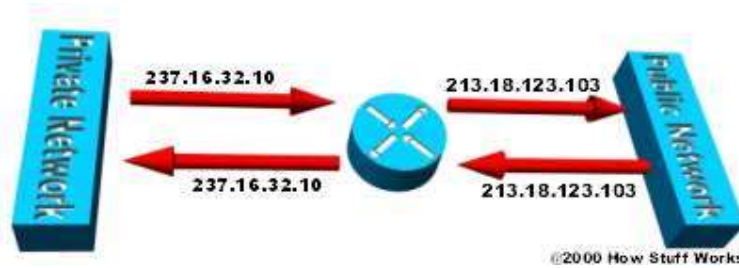
NAT Overloading là một dạng thức của NAT động (Dynamic Overload). Nhiều địa chỉ IP Private sẽ được map với một địa chỉ IP Public qua các Port (cổng) khác nhau.

Cũng giống như PAT (Port Address Translation), một địa chỉ NAT hoặc Port sẽ có nhiều mức độ NAT khác nhau.



Trong Overloading NAT, mỗi máy tính trong mạng nội bộ (Private Network) được Router biên dịch đến cùng một địa chỉ IP 213.18.123.100 nhưng trên các cổng giao tiếp khác nhau.

2.6.4.2. Overlapping NAT



Khi **địa chỉ IP** trong hệ thống mạng nội bộ là **IP Public** đang sử dụng trên một hệ thống mạng khác, **Router** phải duy trì một bảng tìm kiếm các địa chỉ này để ngăn và thay thế bằng một **IP Public** duy nhất.

Điều quan trọng cần lưu ý rằng **NAT router** phải biên dịch địa chỉ "nội bộ" thành một **địa chỉ IP Public** duy nhất cũng như biên dịch địa chỉ "ngoài" thành một địa chỉ **IP Private** duy nhất. Bạn có thể sử dụng **NAT tĩnh** hoặc sử dụng kết hợp **DNS và NAT động**.

Hệ thống mạng nội bộ thông thường là **LAN (Local Area Network)**, hay còn gọi là **Stub Domain**. Một **Stub Domain** là một LAN sử dụng **địa chỉ IP nội bộ**.

Hầu hết các **Network Traffic** (là lưu lượng mạng ổn định, không bị gián đoạn trong quá trình truyền) trong **Stub Domain** mang tính chất cục bộ, do đó hệ thống mạng nội bộ không bao giờ bị lộ ra bên ngoài.

Một **Stub Domain** có thể bao gồm cả **địa chỉ IP Public** và **IP Private**. Bất kỳ máy tính nào sử dụng **địa chỉ IP Private** đều phải dùng **NAT (Network Address Translation)** để trao đổi thông tin với các máy tính khác.

2.7. Ưu – Nhược điểm của wifi

2.7.1. Ưu điểm:

Mạng không dây cũng như hệ thống mạng thông thường. Nó cho phép người dùng truy xuất tài nguyên mạng ở bất kỳ nơi đâu trong khu vực được triển khai (nhà hay văn phòng). Với sự gia tăng số người sử dụng máy tính xách tay (laptop), đó là một điều rất thuận lợi.

Với sự phát triển của các mạng không dây công cộng, người dùng có thể truy cập Internet ở bất cứ đâu. Có thể triển khai ở những nơi không thuận tiện về địa hình, không ổn định, không triển khai mạng có dây được

Người dùng có thể duy trì kết nối mạng khi họ đi từ nơi này đến nơi khác

Mạng không dây có thể đáp ứng tức thì khi gia tăng số lượng người dùng. Với hệ thống mạng dùng cáp cần phải gắn thêm cáp

Mạng Internet không dây (Wi-Fi) đã đem lại nhiều lợi ích cho cộng đồng, nó giúp cho việc truyền tải, tiếp nhận thông tin cực kỳ nhanh chóng và tiện lợi, giúp người sử dụng công nghệ tiết kiệm thời gian nâng cao hiệu quả công việc, chúng ta cũng có thể truy cập Internet để theo dõi tin tức bằng sử dụng điện thoại có kết nối Wi-Fi. Như vậy công nghệ Wi-Fi cũng giúp cho việc thúc đẩy phát triển kinh tế - xã hội một cách rõ rệt.

2.7.2. Nhược điểm:

- Môi trường kết nối không dây là không khí nên khả năng bị tấn công của người dùng là rất cao.
- Chủ yếu là trong mô hình mạng nhỏ và trung bình, với những mô hình lớn phải kết hợp với mạng có dây. Một mạng chuẩn 802.11g với các thiết bị chuẩn chỉ có thể hoạt động tốt trong phạm vi vài chục mét. Nó phù hợp trong 1 căn nhà, nhưng với một tòa nhà lớn thì không đáp ứng được nhu cầu. Để đáp ứng cần phải sử dụng thêm Repeater hay access point, dẫn đến chi phí gia tăng.
- Bị ảnh hưởng bởi các yếu tố bên ngoài như môi trường truyền sóng, có thể nhiễu do thời tiết.
- Chịu nhiều cuộc tấn công đa dạng, phức tạp, nguy hiểm của những kẻ phá hoại vô tình và cố tình, nguy cơ cao hơn mạng có dây
- Tốc độ của mạng không dây (1- 125 Mbps) rất chậm so với mạng sử dụng cáp (100Mbps đến hàng Gbps)
- Sóng Wi-Fi không ảnh hưởng đến sức khỏe con người, nhưng cũng cần lưu ý một số khuyến cáo tương tự khuyến cáo về sử dụng điện thoại di động như: Không nên để máy tính xách tay có Wi-Fi lên đùi, không nên gắn bộ phát Wi-Fi vào đầu giường ngủ... mặc dù chưa có nghiên cứu nào khẳng định sóng Wi-Fi có ảnh hưởng tới sức khỏe. Tuy nhiên, sóng Wi-Fi lại ảnh hưởng không tốt đối với trẻ em, gây ra triệu chứng (đau đầu, hoa mắt, mất ngủ). Vì cơ thể các em nhạy cảm hơn so với người lớn khi tiếp xúc với một số tia bức xạ có hại.

2.8. Giới thiệu ESP8266

2.8.1. Giới thiệu ESP8266

ESP8266 là một chip tích hợp được thiết kế dùng cho chuẩn kết nối mới. Có thể dùng nó để đưa những dự án của bạn kết nối đến Internet. Đơn giản nó sử dụng ngõ giao thức nối tiếp với tốc độ Baud 115200 (mặc định). Kết nối mạng không dây, giống như một máy chủ hoặc một cầu nối trung gian và có thể download dữ liệu từ Internet.

Module ESP8266 V1 có bộ nhớ là 1M, được cài firmware 1.01 và có tốc độ mặc định là 115200 và bạn có thể thay đổi bằng lệnh AT.

2.8.1.1. Thông số kỹ thuật:

Hỗ trợ chuẩn 802.11 b/g/n.
 Wi-Fi 2.4 GHz, hỗ trợ WPA/WPA2.
 Chuẩn điện áp hoạt động: 3.3V.
 Chuẩn giao tiếp nối tiếp UART với tốc độ Baud lên đến 115200
 Có 3 chế độ hoạt động: Client, Access Point, Both Client and Access Point.
 Hỗ trợ các chuẩn bảo mật như: OPEN, WEP, WPA_PSK, WPA2_PSK, WPA_WPA2_PSK.
 Hỗ trợ cả 2 giao tiếp TCP và UDP
 Làm việc như các máy chủ có thể kết nối với 5 máy con.

2.8.1.2. Các lệnh AT chung

AT	Kiểm tra lệnh, luôn trả về "OK"		<i>AT</i>
AT+RST	Khởi động lại module		<i>AT+RST</i>
AT+GMR	Truy vấn phiên bản Firmware		<i>AT+GMR</i>

Bảng 2.1. Tập lệnh AT chung

2.8.1.3. Các lệnh AT cấu hình Module Wifi

AT+CWMODE=<mode>	Cài đặt chế độ	1 = Station 2 = Access Point 3 = Both	<i>AT+CWMODE=1</i>
AT+CWMODE?	Truy vấn chế độ đã cài đặt		<i>AT+CWMODE?</i>
AT+CWMODE=?	Truy vấn các chế độ có thể cài đặt		<i>AT+CWMODE=?</i>
AT+CIPMUX=<mode>	Cài đặt số lượng các kênh kết nối	0 = 1 kênh kết nối 1 = Nhiều kênh kết nối	<i>AT+CIPMUX=1</i>
AT+CIPMODE=<mode>	Cài đặt chế độ dữ liệu	0 = transparent 1 = Data	<i>AT+CIPMODE=1</i>
AT+CIPMODE?	Truy vấn chế độ dữ liệu cài đặt		<i>AT+CIPMODE?</i>

Bảng 2.2. Các lệnh AT cấu hình Module Wifi

2.8.1.4. Các lệnh AT đối với Module Wifi cấu hình là Station / client

AT+CWJAP = <ssid>, <password>	Kết nối với 1 mạng wifi	ssid "SSID" pass "password"	<i>AT+CWJAP = "MLAB", "1235678"</i>
AT+CWJAP?	Truy vấn mạng wifi đang kết nối		<i>AT+CWJAP?</i>
AT+CWLAP	Truy vấn các mạng wifi có thể kết nối		<i>AT+CWLAP</i>
AT+CWQAP	Đóng kết nối wifi với một Access Point		<i>AT+ CWQAP</i>
AT+CIFSR	Xem địa chỉ IP của module		<i>AT+CIFSR</i>

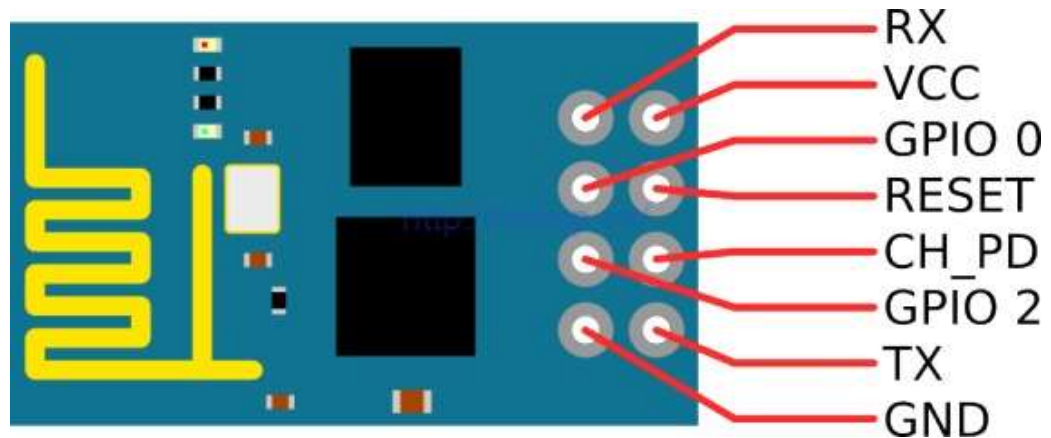
Bảng 2.3. Các lệnh AT cấu hình Station

2.8.1.5. Các lệnh AT với Module Wifi cấu hình là Access Point

AT+CWSAP=<ssid>,<password> , <chan>, <enc>	Cài đặt các thông số cho Access Point	ssid "SSID" pass "password" chan "channel" enc "Encryption" (0 = Open 1= WEP 2= WPA_PSK 3= WPA2_PSK 4=WPA_WPA2_PSK)	<i>AT+CWSAP="MLAB", "12345678", 5, 3</i>
AT+CWSAP?	Xem cài đặt hiện tại của Access Point		<i>AT + CWSAP?</i>
AT+CWLIF	Danh sách các station đang kết nối		<i>AT + CWLIF</i>

Bảng 2.4. Các lệnh AT với Module Wifi cấu hình là Access Point

2.8.1.6. Sơ đồ chân của ESP8266 V01



Hình 2.5. Các chân ESP

2.8.1.7. Chân kết nối:

- VCC: 3.3V lên đến 300mA
- GND: Mass
- Tx: Chân Tx của giao thức UART, kết nối đến chân Rx của vi điều khiển.
- Rx: Chân Rx của giao thức UART, kết nối đến chân Tx của vi điều khiển.
- RST: chân reset, kéo xuống mass để reset.
- CH_PD: Kích hoạt chip, sử dụng cho Flash Boot và updating lại module
- GPIO0: kéo xuống thấp cho chế độ update.
- GPIO2: không sử dụng.

2.8.2. Vấn đề lập trình trên ESP8266 V01

2.8.2.1. Lập Trình ESP8266 V1 trên IDE arduino

ESP8266 được tích hợp vi điều khiển Tenslica L106 32 bit (MCU) có tính năng tiêu thụ điện năng thấp và RSIX 16 bit, đạt tốc độ 160MHz. Với hệ thống hoạt động thời gian thực và chức năng ngăn xếp Wi-Fi, khoảng 80% sức mạnh của vi xử lý vẫn còn sẵn cho người dùng lập trình và phát triển.

Để lập trình được cho ESP ta cần thư viện cho ESP. Trên IDE ta có thể tải về qua Boards Manager.

2.8.2.2. Thư viện ESP8266WiFi

WiFi.mode(m): thiết lập chế độ WIFI_AP, WIFI_STA, WIFI_AP_STA hoặc WIFI_OFF.

Gọi WiFi.softAP(ssid) để thiết lập một open network

Gọi `WiFi.softAP(ssid, password)` để thiết lập WPA2-PSK (mật khẩu ít nhất 8 ký tự)

`WiFi.macAddress(mac)` cho STA, `WiFi.softAPmacAddress(mac)` cho AP.

`WiFi.localIP()` cho STA, `WiFi.softAPIP()` cho AP.

Để chuyển đổi sang chế độ station, ta dùng hàm `begin`. Các tham số cần thiết sẽ là SSID và password, để module có thể kết nối đến một Access Point (AP) cụ thể.

`WiFi.begin(ssid, password)`

Theo mặc định, ESP sẽ cố kết nối lại đến mạng WiFi sau khi bị disconnect. Do đó chúng ta không cần phải xử lý việc này trong code.

`WiFi.begin()`

Gọi hàm này module sẽ chuyển sang chế độ station và kết nối với điểm truy cập cuối cùng được sử dụng dựa trên cấu hình được lưu trong bộ nhớ flash. Để thiết lập tất cả các thông số, ta có thể dùng lệnh:

`WiFi.begin(ssid, password, channel, bssid, connect)`

Các thông số:

`ssid` - tên WiFi của điểm truy cập mà chúng ta muốn kết nối đến, có thể có tối đa lên đến 32 ký tự.

`password` - mật khẩu của điểm truy cập, có độ dài từ 8 đến 64 ký tự.

`channel` - thiết lập kênh cho WiFi, tham số này có thể bỏ qua.

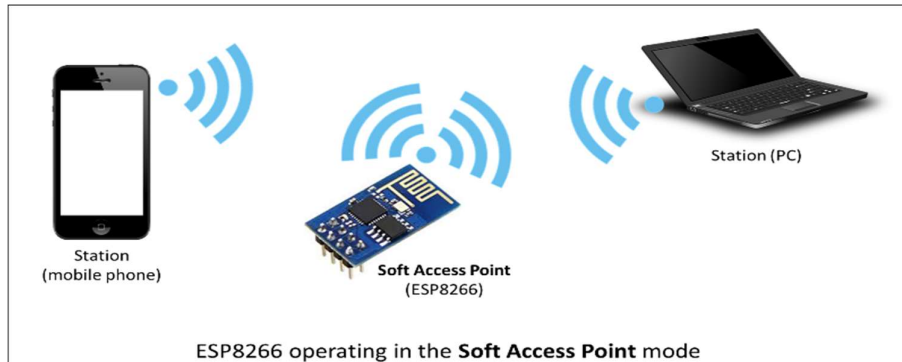
`bssid` - địa chỉ MAC của AP

`connect` - nếu giá trị là false, module sẽ lưu các tham số nhưng không thiết lập kết nối đến điểm truy cập.

2.8.2.3. WiFi Access Point

Access Point (AP - Điểm truy cập) cung cấp khả năng truy cập mạng WiFi cho các thiết bị khác (Station) và kết nối chúng với mạng có dây. ESP8266 có thể cung cấp

chức năng tương tự nhưng nó không kết nối có dây với một mạng. Chế độ hoạt động như vậy gọi là **soft-AP**. Số lượng trạm tối đa kết nối với soft-AP là 5.



Hình 2.6. Mô hình kết nối wifi

Phần mô tả API này gồm có 3 phần: cách thiết lập soft-AP, quản lý kết nối và lấy thông tin về cấu hình soft-AP.

2.8.2.4. Thiết lập mạng

Phần này mô tả các chức năng để thiết lập và cấu hình ESP8266 ở chế độ điểm truy cập mềm (soft-AP).

2.8.2.5. softAP

Cách thiết lập đơn giản nhất chỉ yêu cầu một tham số và được sử dụng để thiết lập một mạng Wi-Fi mở.

WiFi.softAP(ssid)

Để thiết lập mạng được bảo vệ bằng mật khẩu, hoặc để cấu hình các thông số mạng bổ sung, sử dụng quá tải sau đây:

WiFi.softAP(ssid, password, channel, hidden)

Tham số đầu tiên của hàm này là bắt buộc, còn lại ba tùy chọn.

ssid - chuỗi ký tự chứa SSID mạng (tối đa 63 ký tự)

password - chuỗi ký tự tùy chọn với mật khẩu. Đối với mạng WPA2-PSK, nó phải có ít nhất 8 ký tự. Nếu không được chỉ định, điểm truy cập sẽ mở ra cho bất kỳ ai kết nối.

channel - Tham số tùy chọn để thiết lập kênh Wi-Fi, từ 1 đến 13. Kênh mặc định = 1.

hidden - Tham số tùy chọn, thiết lập là true để ẩn SSID

2.8.2.6. Thư Viện ESP8266HTTPClient.h

Chưa các hàm:

http.begin("http://anhtan102.freeasphost.net/sendrequest?"). Để gửi kết yêu cầu truy cập đến một địa chỉ.

http.GET() nhận dữ liệu trả về từ trang web.

Thư viện ESP8266WebServer.h

```
server.on("/", []()
{
    IPAddress ip = WiFi.softAPIP();
    String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.'
+ String(ip[3]);

    content = "<!DOCTYPE HTML>\r\n<html><h2>XSwitch</h2>";
    content += "<form method=\"get\" action=\"setting\">";
    content += "<div>Wifi</div>";
    content += "<div><input name=\"ssid\" size=\"40\"></div>";
    content += "<div>Mat Khau</div>";
    content += "<div><input name=\"pass\" size=\"40\"></div>";
    content += "<div><input type='submit'></div>";
    content += "<p>";
    content += st;
    content += "</p>";
    content += "</html>";
    server.send(200, "text/html", content);
});
```

2.8.3. GIỚI THIỆU VỀ ARDUINO VÀ ATMEGA328

Arduino là một nền tảng điện tử nguồn mở dựa trên những phần cứng và phần mềm linh hoạt, dễ sử dụng. Arduino dành cho những nghệ sĩ, nhà thiết kế, người thích tìm tòi và tất cả những ai yêu thích sáng tạo các sản phẩm và môi trường có tính tương tác cao.

Bo mạch Arduino Uno R3 là là một hệ thống mở

Sử dụng chip AVR ATmega328 của ATmel.

Mạch arduino được lắp ráp từ các linh kiện dễ tìm và hướng đến đối tượng người dùng đa dạng. Đừng lo nếu bạn là dân nghiệp dư vì arduino có một hệ thống thư viện phong phú, cộng đồng người dùng arduino đông đảo sẵn sàng chia sẻ kiến thức và mã nguồn sẽ giúp bạn tạo nên những dự án thiết thực.

2.8.3.1. Một vài thông số của Arduino UNO R3

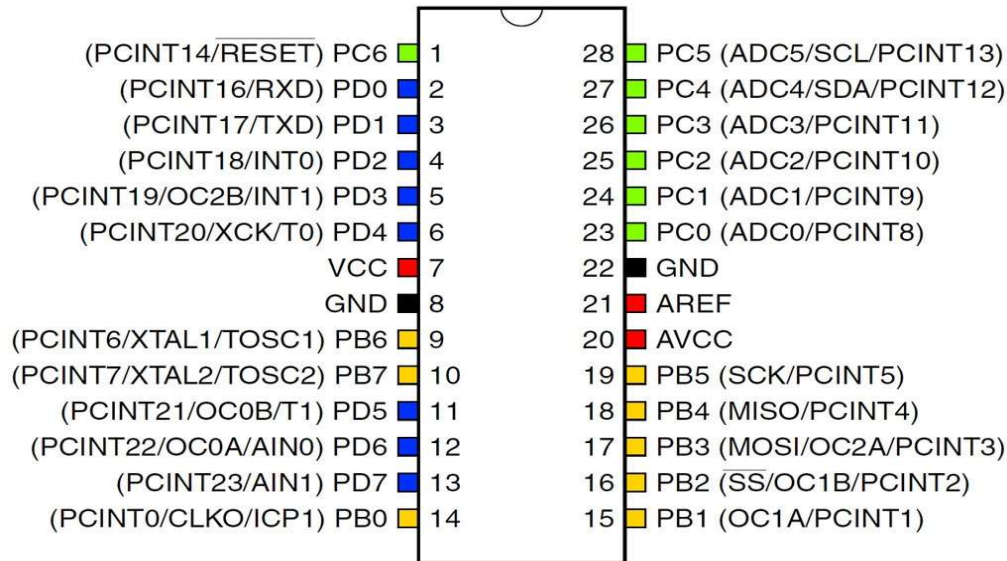
<i>Vi điều khiển</i>	<i>ATmega328 họ 8bit</i>
<i>Điện áp hoạt động</i>	<i>5V DC (chỉ được cấp qua cổng USB)</i>
<i>Tần số hoạt động</i>	<i>16 MHz</i>
<i>Dòng tiêu thụ</i>	<i>khoảng 30mA</i>
<i>Điện áp vào khuyến dùng</i>	<i>7-12V DC</i>
<i>Điện áp vào giới hạn</i>	<i>6-20V DC</i>
<i>Số chân Digital I/O</i>	<i>14 (6 chân hardware PWM)</i>
<i>Số chân Analog</i>	<i>6 (độ phân giải 10bit)</i>
<i>Dòng tối đa trên mỗi chân I/O</i>	<i>30 mA</i>
<i>Dòng ra tối đa (5V)</i>	<i>500 mA</i>
<i>Dòng ra tối đa (3.3V)</i>	<i>50 mA</i>
<i>Bộ nhớ flash</i>	<i>32 KB (ATmega328) với 0.5KB dùng bởi bootloader</i>
<i>SRAM</i>	<i>2 KB (ATmega328)</i>
<i>EEPROM</i>	<i>1 KB (ATmega328)</i>

*Bảng 2.5. Một vài thông số của Arduino UNO R3***2.8.3.2. IC Atmega328**

Atmega328 là một chip vi điều khiển được sản xuất bởi hãng [Atmel](#) thuộc họ MegaAVR có sức mạnh hơn hẳn [Atmega8](#). Atmega 328 là một bộ vi điều khiển 8 bit dựa trên kiến trúc RISC bộ nhớ chương trình 32KB ISP flash có thể ghi xóa hàng nghìn lần, 1KB EEPROM, một bộ nhớ RAM vô cùng lớn trong thế giới vi xử lý 8 bit (2KB SRAM)

Với 23 chân có thể sử dụng cho các kết nối vào hoặc ra i/O, 32 thanh ghi, 3 bộ timer/counter có thể lập trình, có các ngắt nội và ngoại (2 lệnh trên một vector ngắt),

giao thức truyền thông nối tiếp USART, SPI, I2C. Ngoài ra có thể sử dụng bộ biến đổi số tương tự 10 bit (ADC/DAC) mở rộng tới 8 kênh, khả năng lập trình được watchdog timer, hoạt động với 5 chế độ nguồn, có thể sử dụng tới 6 kênh điều chế độ rộng xung (PWM), hỗ trợ bootloader.



Hình 2.7. Hình sơ đồ chân ATMEGA328

Atmega328 có khả năng hoạt động trong một dải điện áp rộng (1.8V – 5.5V), tốc độ thực thi (thông lượng) 1MIPS trên 1MHz

Ngày nay vi điều khiển Atmega328 thực sự được sử dụng phổ biến từ các dự án nhỏ của sinh viên, học sinh với giá thành rẻ, xử lý mạnh mẽ, tiêu tốn ít năng lượng (chế độ hoạt động: 0.2 mA, chế độ ngủ: 0.1 μ A, chế độ tích kiệm: 0.75 μ A) và sự hỗ trợ nhiệt tình của cộng đồng người dùng AVR. Và không thể không nhắc tới sự thành công của Vi điều khiển Atmega328 trong dự án mã nguồn mở Arduino với các modul Arduino Uno (R3), Arduino Nano, Arduino Pro mini những sản phẩm dẫn dắt chúng ta vào thế giới mã nguồn mở để hoàn thành một chương trình trong “nháy mắt”.



Hình 2.8. Hình thực tế ATMEGA328

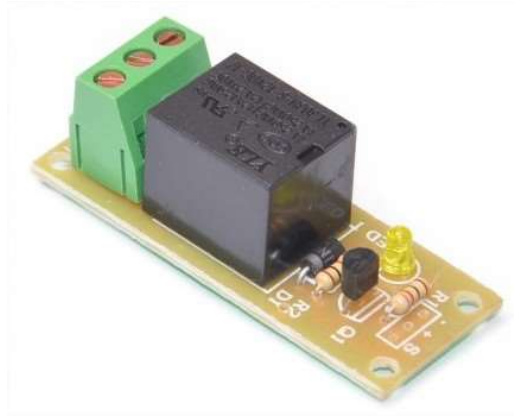
Thông số chính Atmega328P-PU:

- + Kiến trúc: AVR 8bit
- + Xung nhịp lớn nhất: 20Mhz
- + Bộ nhớ chương trình (FLASH): 32KB
- + Bộ nhớ EEPROM: 1KB
- + Bộ nhớ RAM: 2KB
- + Điện áp hoạt động rộng: 1.8V – 5.5V
- + Số timer: 3 timer gồm 2 timer 8-bit và 1 timer 16-bit
- + Số kênh xung PWM: 6 kênh (1timer 2 kênh)

2.8.4. MỘT SỐ LINH KIỆN KHÁC

2.8.4.1. Rơ-le

Rơ-le là một **công tắc** (khóa K). Nhưng khác với công tắc ở một chỗ cơ bản, rơ-le được kích hoạt bằng điện thay vì dùng tay người. Chính vì lẽ đó, rơ-le được dùng làm công tắc điện tử! Vì rơ-le là một công tắc nên nó có 2 trạng thái: **đóng** và **mở**. "*Khi nào nó đóng? Khi nào nó mở? và làm sao thay đổi được trạng thái của nó? ,...*" đó chính là những câu hỏi mà chúng ta cần tìm kiếm câu trả lời trong bài viết này.



Hình 2.9. Hình Một module relay kiểu mẫu

- Cấu tạo Relay gồm 2 phần:

+ Cuộn hút:

- Tạo ra năng lượng từ trường để hút tiếp điểm về phía mình.

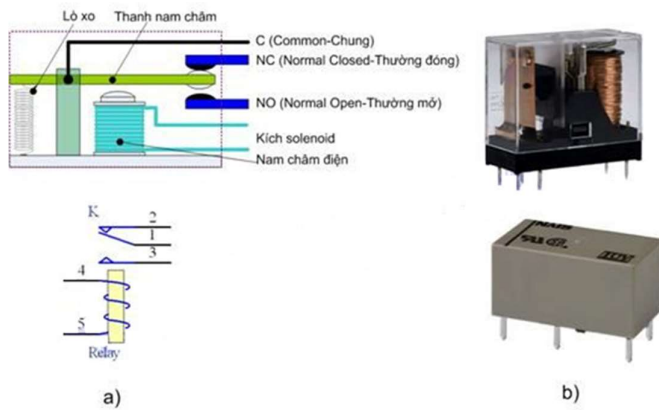
- Tùy vào điện áp làm việc người ta chia Relay ra DC: 5V, 12V, 24V - AC: 110V, 220V

+ Cặp tiếp điểm:

- Khi không có từ trường (không cấp điện cho cuộn dây). Tiếp điểm 1 được tiếp xúc với 2 nhờ lực của lò xo. Tiếp điểm thường đóng.

- Khi có năng lượng từ trường thì tiếp điểm 1 bị hút chuyển sang 3.

- Trong Relay có thể có 1 cặp tiếp điểm, 2 cặp tiếp điểm hoặc nhiều hơn.



Hình 2.10. Relay

2.8.4.2. Cảm biến độ ẩm đất:

- Điện áp làm việc: 3.3-5V
- Kích thước 3.2x1.4 Cm

- Sử dụng chip LM393 để so sánh
- D0 đầu ra digital
- A0 đầu ra tín hiệu tương tự.



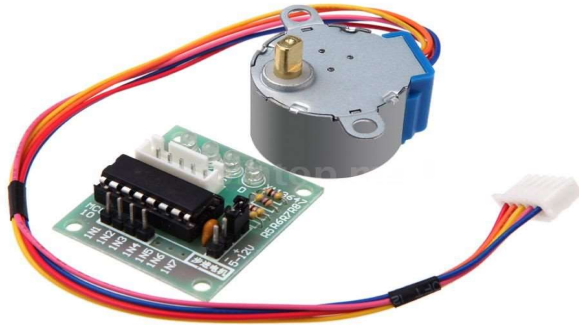
Hình 2.11. Cảm biến độ ẩm.

2.8.4.3. Động cơ bước

- Khái niệm:

Động cơ bước là một loại động cơ mà ở đó bạn sẽ có thể quy định chính xác số góc quay và động cơ bước sẽ phải quay. Không giống như Servo, động cơ bước có thể quay bao nhiêu độ tùy ý và mỗi lần quay nó sẽ quay được 1 step, 1 step ở đây là bao nhiêu còn phụ thuộc vào động cơ bước của bạn. Ví dụ, động cơ bước của bạn có 72 step thì nó sẽ cần quay 72 step để hoàn thành một vòng quay. Số step này là hằng số, nhưng bạn có thể dùng công nghệ micro step để "cải thiện" số vòng quay động cơ bước của bạn

- Thông số kỹ thuật:

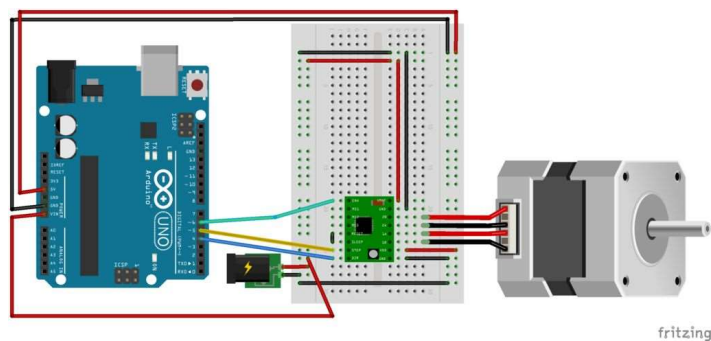


Hình 2.12. Động cơ bước

Rated Current/phase (dòng tiêu thụ tối đa mỗi pha)	2.0A
Phase Resistance (điện trở từng pha)	1.4ohms
Voltage (hiệu điện thế)	2.8V

Trong đó, tham số Rated Current/phase thể hiện dòng điện tối đa mà mỗi pha có thể nhận được, nếu driver điều khiển nào có dòng điện mỗi pha cao hơn thì sẽ hỏng. Nên lựa loại driver cho dòng ra xấp xỉ 80 - 90% thông số này. Điện trở mỗi pha là hằng số (bạn xem trong datasheet của động cơ bước mà bạn mua). Còn con số voltage là hiệu điện thế tối ưu để làm stepper hoạt động ổn định ($2.0 * 1.4 = 2.8V$)

- Cách nối dây:



Hình 2.13. Sơ đồ kết nối dây

2.8.4.4. Cảm biến ánh sáng

- Giới thiệu:

Cảm biến ánh sáng sử dụng quang trở có khả năng thay đổi điện trở theo cường độ ánh sáng chiếu vào. Tín hiệu xuất ra của cảm biến là digital HIGH (5V) và LOW tương trưng cho các trạng thái bật, tắt thiết bị điện tự động mà bạn không cần phải thao tác vào!



Hình 2.14. Cảm biến ánh sáng

- **Mạch cảm biến ánh sáng dùng quang trở có ưu điểm**
 - Nhỏ gọn.
 - Độ chính xác cao.
 - Các thành phần phụ như điện trở, tụ điện... cần thiết cho mạch đã được gắn đầy đủ. Bạn chỉ cần cấp nguồn, nối dây điều khiển vào là có thể tắt/mở bóng đèn hay các thiết bị điện khác theo cường độ ánh sáng chiếu vào cảm biến.
 - Giá thành thấp, khoảng 50.000 đồng.
 - Sử dụng điện áp chuẩn 5V tương thích với nền tảng Arduino.
 - **Cảm biến ánh sáng có thể điều chỉnh được độ nhạy**

Trên mạch có 1 biến trở 10K ohm dùng để điều chỉnh độ nhạy sáng:

- **Vặn về bên trái:** bạn sẽ tăng độ nhạy của cảm biến với ánh sáng: chỉ cần lượng ánh sáng nhỏ thì mạch sẽ tự ngắt.

- **Vấn về bên phải:** bạn sẽ giảm độ nhạy của cảm biến với ánh sáng, cần lượng ánh sáng với cường độ mạnh hơn để ngắt mạch.

- **Lắp mạch và lập trình**

Cảm biến này có thể sử dụng kết hợp với Arduino để lập trình bật tắt thay vì mạch Rơ-le nhé.

Cảm biến này là một dạng cảm biến Digital - tín hiệu xuất ra là giá trị Digital HIGH (5V) và LOW. Tại chân OUT, mạch trả về mức HIGH (5V) khi trời tối (cường độ ánh sáng chiếu vào thấp) và LOW nếu ngược lại.

3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

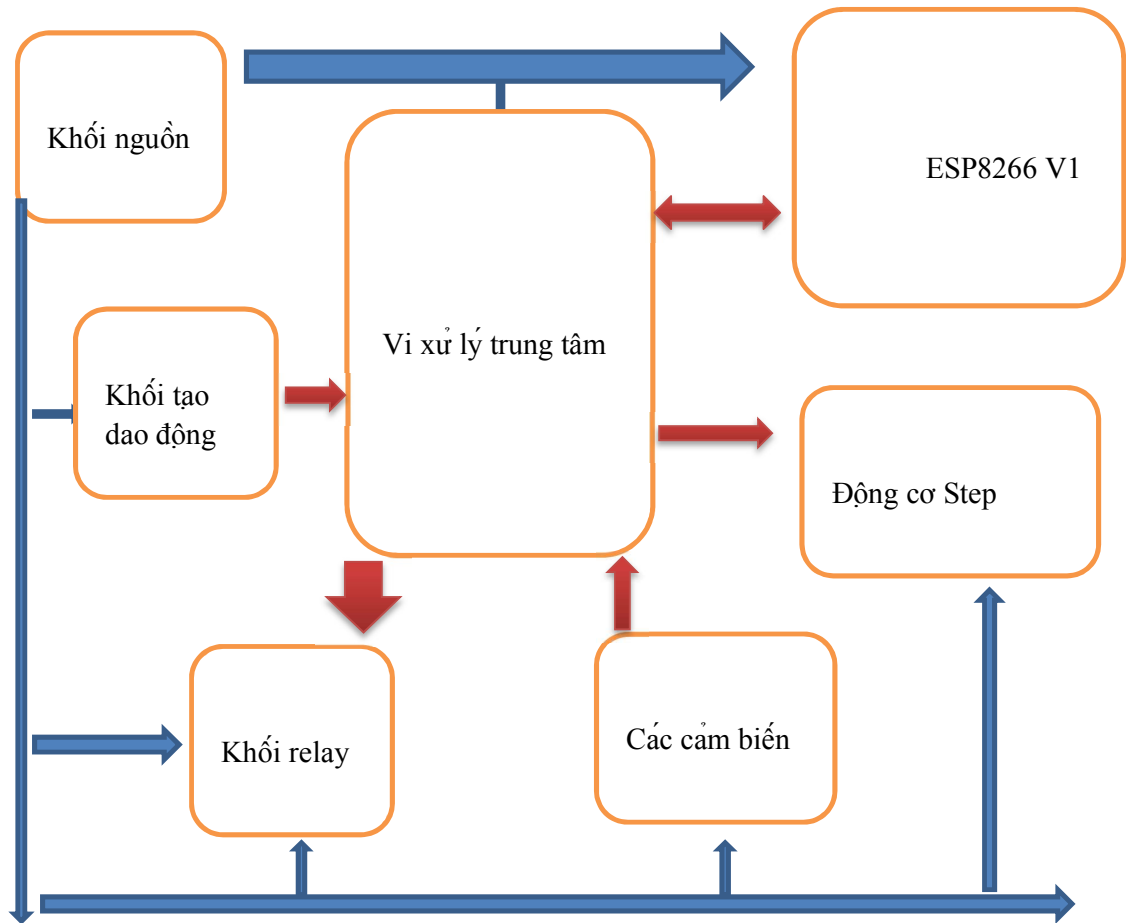
3.1. *Yêu cầu thiết kế phần cứng*

Phần thiết kế có chức năng nhận dữ liệu nhận được từ ESP8266 và xử lý dữ liệu. Chuyển thành lệnh điều khiển các Relay, Led và động cơ Step.

Với động cơ Step mình dùng sẵn phần cứng đi kèm và có sẵn driver. Với ngõ ra điều khiển là 4 chân tín hiệu và 2 chân nguồn.

Mình chọn IC ATMEGA328 để làm vi xử lý trung tâm vì: trong quá trình thực hiện đồ án và lên ý tưởng. Mình dùng mạch Arduino để nạp code và chạy thử nghiệm. Sau khi mình thử nghiệm thành công mình chuyển sang dùng luôn IC của mạch này cho tiện. Vì yêu cầu thời gian ngắn nên cách này là một cách hơi tốn kém nhưng mà hiệu quả.

3.2. Sơ đồ tổng quát.



Hình 3.1. Sơ đồ khối phân cứng

3.3. Thiết kế khối Relay.

Để điều khiển các thiết bị điện thông thường, ta không thể sử dụng trực tiếp các chân ngõ ra của vi điều khiển vì hạn chế của điện áp và dòng ra của vi điều khiển. Muốn điều khiển thông qua vi điều khiển, ta cần một mạch relay. Mạch này bao gồm một BJT đóng vai trò khóa điện tử, nhận tín hiệu điều khiển từ vi điều khiển. BJT sẽ đóng ngắt nguồn điện đi qua cuộn dây của relay, từ đó relay sẽ đóng ngắt các thiết bị điện.

Đèn LED được mắc song song với cuộn dây của relay để chỉ thị trạng thái của relay. Điện áp rơi trên LED là 2.47-3.7V và chọn dòng qua LED là 10 -20mA, ta tính được R10 như sau:

$$R10 = \frac{5 - 3.7}{0.02} \sim \frac{5 - 2.47}{0.01} = 65 \sim 223\Omega$$

Chọn R10 là 220Ω, khi đó dòng qua LED là 11.5 mA, đảm bảo để LED sáng bình thường.

Ở đây sử dụng relay SRD-05VDC-SL-C có điện áp hoạt động của cuộn dây là 5V, có thể đóng ngắt tải có điện áp xoay chiều tối đa là 250V, điện áp một chiều tối đa là 30V và dòng tối đa là 15A. Dòng đi qua cuộn dây lúc hoạt động là 71.4 mA.

BJT Q1815 được phân cực để hoạt động như khóa điện tử. Vì ngõ ra các chân của ATMEGA328 có điện áp là 5 V nên khi có tín hiệu mức cao tương ứng với 5 V thì BJT dẫn bão hòa. Để BJT dẫn bão hòa thì :

$$I_B \geq \frac{I_C}{h_{fe \min}} \text{ với } I_C = 80 \text{ mA và } h_{fe \min} = 30$$

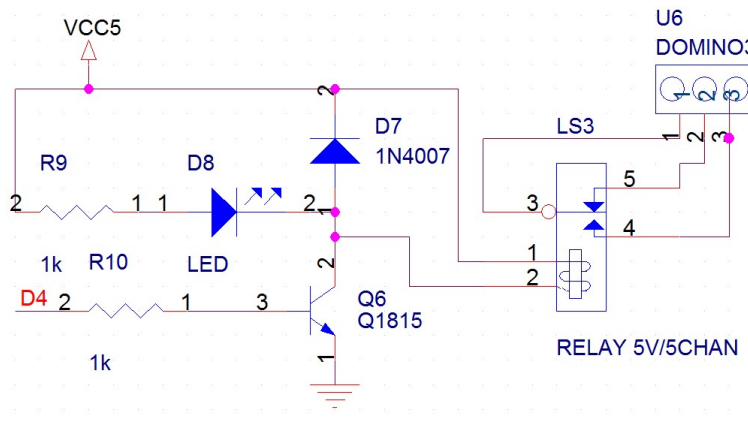
Khi đó $I_B \geq 2.67 \text{ mA}$. Từ đó tính được R1 :

$$R10 < \frac{V_{IN} - V_{BE}}{I_B} = \frac{5 - 0.7}{0.0026} = 1654\Omega. \text{ Chọn } R10 = 1k.$$

Khi áp ở ngõ vào cực B là 5V thì BJT sẽ dẫn bão hòa, có dòng đi qua cuộn dây và đèn LED. Relay sẽ chuyển sang vị trí thường hở, đèn LED sẽ sáng.

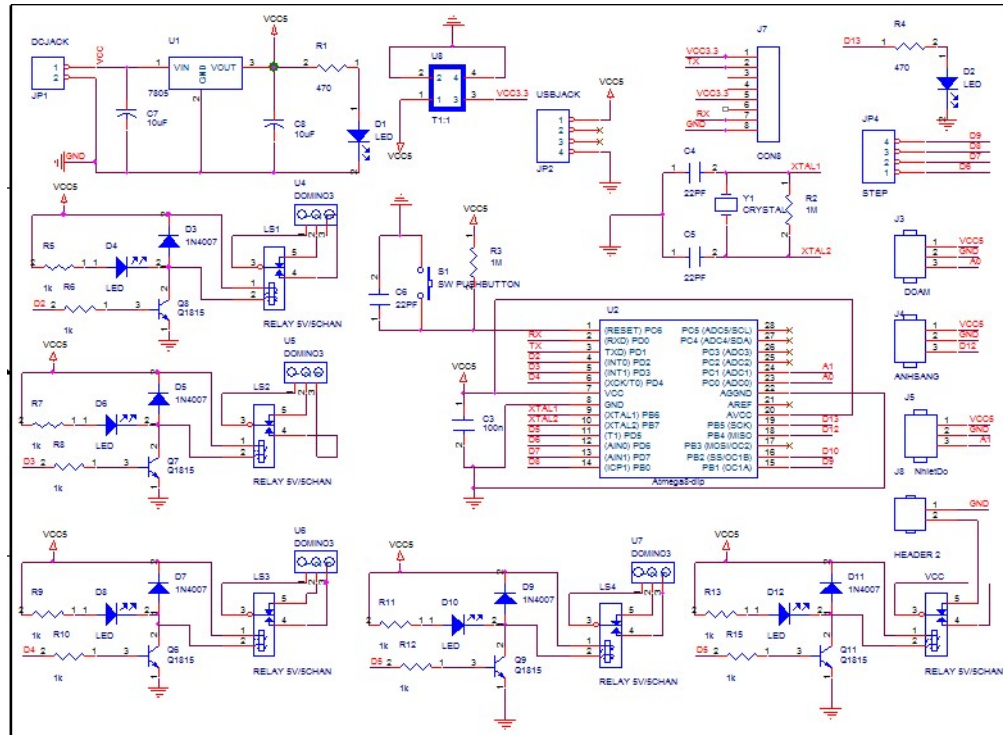
Khi áp ở ngõ vào cực B là 0 V, $V_{IN} < V_{BE}$ nên BJT tắt, không có dòng đi qua cuộn dây, relay ở vị trí thường đóng.

Diode mắc ngược D1 có nhiệm vụ bảo vệ BJT khỏi các gai xung điện áp khi BJT tắt.



Hình 3.2. Sơ đồ khối Relay

3.4. Sơ đồ toàn mạch



Hình 3.3. Sơ đồ mạch chính

4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

4.1. *Yêu cầu thiết kế*

Yêu cầu thiết kế phần mềm gồm có hai phần chính là: trang Web, phía Server và phần mềm cho vi điều khiển.

Phần trang Web cần hiển thị các lựa chọn để điều khiển các thiết bị, trạng thái của các thiết bị và trạng thái của cảm biến. Ngoài ra còn hiển thị các dữ liệu ghi lại thời gian thay đổi trạng thái của thiết bị và thời gian phát hiện chuyển động của cảm biến. Phía Server có nhiệm vụ nhận các giá trị biến do vi điều khiển gửi lên và lưu vào cơ sở dữ liệu

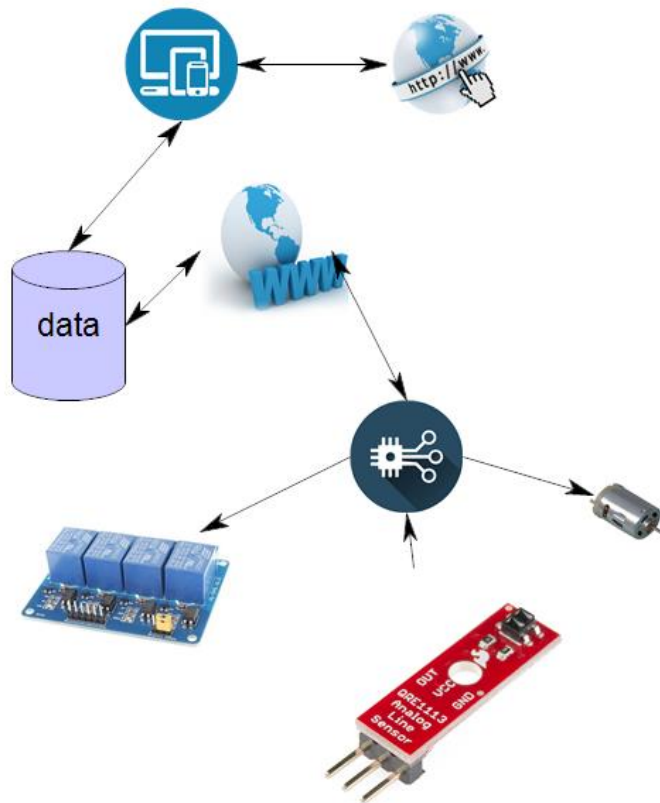
4.2. *Phần mềm cho vi điều khiển và ESP8266*

- Vi điều khiển có nhiệm vụ nhận lệnh từ ESP8266 và thực hiện điều khiển thiết bị rồi trả về kết quả cũng như trạng thái của các thiết bị.
- ESP8266 kết nối wifi gửi và nhận lệnh từ cơ sở dữ liệu.

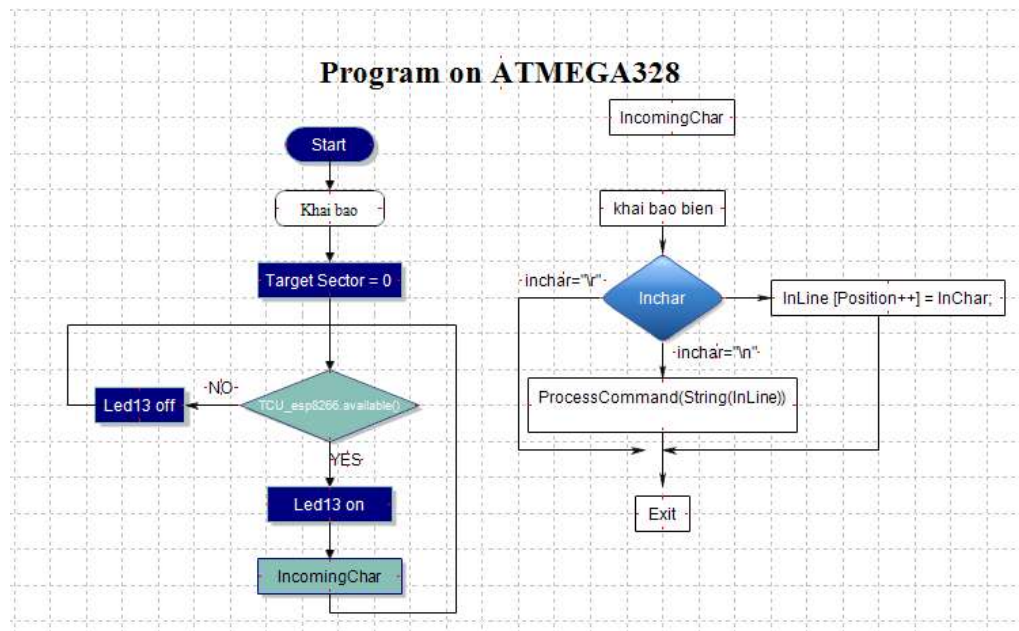
4.2.1. **Phần mềm dành cho ATMEGA328.**

Là nhân của Kit Arduino nên ta lập trình cho ATMEGA này cũng y như lập trình cho Kit Arduino uno R3. Ta cũng dùng phần mềm biên dịch IDE để viết.

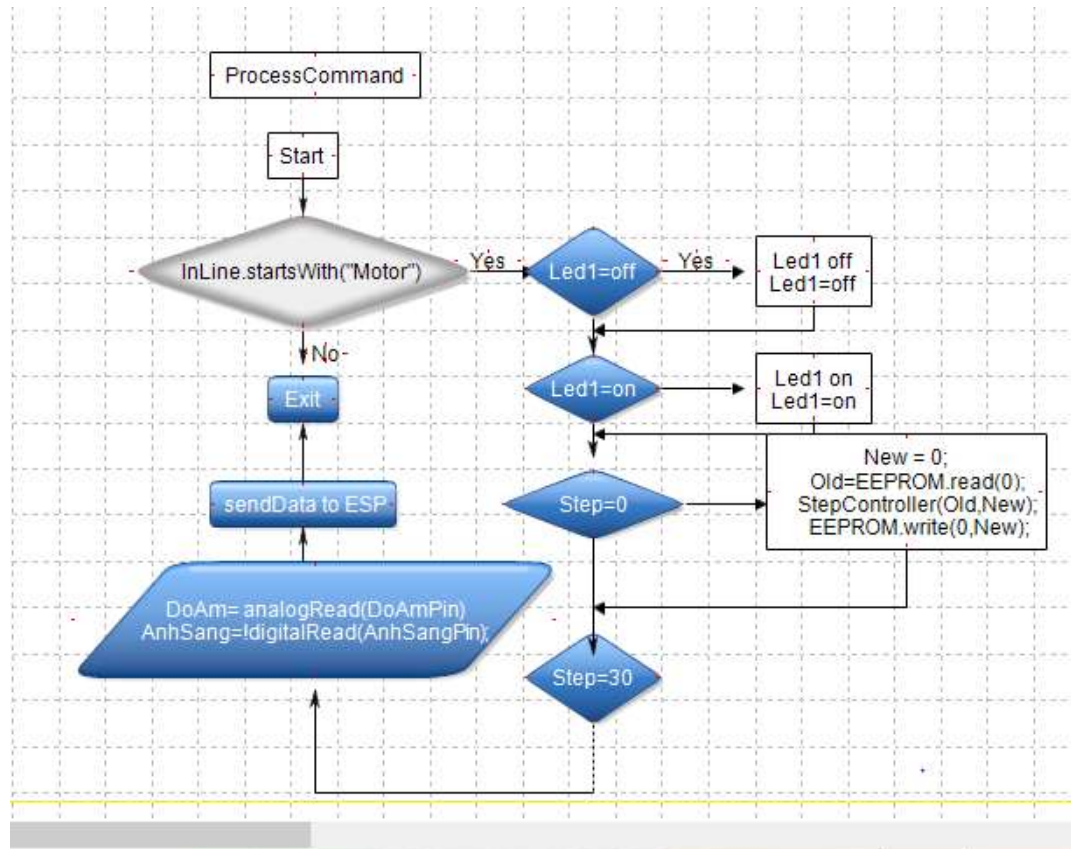
Ic này làm việc như sau: Ban đầu chương trình khai báo các biến và các chân tín hiệu ra điều khiển các thiết bị. Chương trình sẽ mở cổng seriport để nhận tín hiệu từ ESP. Khi nhận tín hiệu xong. Chương trình sẽ xử lý tín hiệu và điều khiển thiết bị theo nội dung nhận được. Sau khi thực hiện xong chương trình sẽ đọc các tín hiệu độ ẩm đất, nhiệt độ trong phòng, độ ẩm trong phòng và ánh sáng trong phòng để gửi dữ liệu này qua ESP.



Hình 4.1. Sơ đồ hệ thống



Hình 4.2 Sơ đồ giải thuật



Hình 4.3. Sơ đồ giải thuật của hàm xử lý chuỗi

```

if(analogRead(AnhSangPin)<200)
{ AnhSang=1;}
else
{AnhSang=0;}

long DoAm1=analogRead(DoAmPin);
DoAm=100-DoAm1*100/1023;
dht11.read(pinDHT11, &NhietDo, &DoAmKK,NULL);

String Getstatus ="DE,&Motor="
+Motor+"&Fan="
+Fan+"&Step="
+New+ "&Led1="
+Led1+"&Led2="
+Led2+"&DoAm="

```



```
+DoAm+"&AnhSang="
+AnhSang+"&NhietDo="
+(int)NhietDo+"&DoAmKK="+ (int)DoAmKK;
Myserial.println(Getstatus);.
```

4.2.2. Phần mềm cho ESP8266.

ESP8266 có thể được lập trình như một vi xử lý bình thường. Với đầy đủ chức năng của một MCU.

Ở đây mình dùng chính IDE của arduino để lập trình cho ESP. Với các thư viện hỗ trợ được cung cấp bởi nhà sản xuất của ESP.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WebServer.h>
#include "EEPROM.h"
```

Thư viện #include <ESP8266WiFi.h> được dùng để thiết lập các chế độ wifi.

- WiFi.mode(WIFI_STA). chế độ điểm truy cập
- WiFi.scanNetworks(). Hàm tìm kiếm các điểm truy cập xung quanh

Sơ lược cách thức hoạt động của chương trình:

Khi khởi động chương trình. ESP đọc tên điểm truy cập và mật khẩu đã lưu trong bộ nhớ EEPROM và thực hiện kết nối với điểm truy cập đó. Sau một khoảng thời gian. Nếu không truy cập được (có thể do tín hiệu yếu hoặc không còn tồn tại điểm truy cập đó nữa) thì ESP sẽ tự động chuyển sang chế độ phát wifi và là một webserver đơn giản để tương tác với người dùng. Khi người dùng truy cập và điểm phát wifi do ESP tạo ra. Và gõ vào trình duyệt địa chỉ 192.168.4.1 thì giao diện web sẽ mở ra và hiện các điểm truy cập hiện có gần đây. Người dùng có thể chọn và gõ mật khẩu vào để ESP tự lưu mật khẩu và tên truy cập vào EEPROM để cho những lần kết nối sau. Sau khi nhập điểm truy cập và mật khẩu. Ta tiến hành khởi động lại ESP. Sau lần khởi động. Nếu truy cập được điểm phát wifi. ESP sẽ chạy vào vòng lặp. Trong vòng lặp này, chương trình sẽ truy cập tới một địa chỉ web đã định sẵn và đọc nội dung từ trang web đó. Nội dung trang web đó như sau.

```
<!DOCTYPE
html>
```

```
<html lang="en">
```

```

<head>

<meta charset="utf-8" />

<title>Status</title>

</head>

<body>

<div id="statusupdate">

<ul>

<li>Motor = on.</li>

<li>Fan = on.</li>

<li>Step = 0.</li>

<li>Led1 = off.</li>

<li>Led2 = off.</li>

</ul>

</div>

</body>

</html>

```

Đây là trang web thể hiện các trạng thái mà người dùng muốn các thiết bị đáp ứng. ESP sẽ lọc dữ liệu và đọc các trạng thái đó bằng đoạn code.

```

if (InLine.indexOf("<li") {
    if (InLine.indexOf("Led1 = off") != -1)
    {
        Led1 = "off";
    }
    if (InLine.indexOf("Led1 = on") != -1)
    {

```

```
Led1 = "on";  
}  
if (InLine.indexOf("Led2 = off") != -1)  
{  
    Led2 = "off";  
}  
if (InLine.indexOf("Led2 = on") != -1)  
{  
    Led2 = "on";  
}  
if (InLine.indexOf("Motor = on") != -1)  
{  
    Motor = "on";  
}  
if (InLine.indexOf("Motor = off") != -1)  
{  
    Motor = "off";  
}  
if (InLine.indexOf("Fan = off") != -1)  
{  
    Fan = "off";  
}  
if (InLine.indexOf("Fan = on") != -1)  
{  
    Fan = "on";  
}  
if (InLine.indexOf("Step = 0") != -1)  
{
```

```

    New = 0;
}
if (InLine.indexOf("Step = 30") != -1)
{
    New = 30;
}

if (InLine.indexOf("Step = 60") != -1)
{
    New = 60;

}

if (InLine.indexOf("Step = 100") != -1)
{
    New = 100;
}

String Request="Motor="+ Motor + " Fan="+ Fan + " Step="+ New + "
Led1="+ Led1 + " Led2="+ Led2;

Serial.println(Request);

```

Sau khi có được các trạng thái thì ESP sẽ gửi dữ liệu cho ATMEGA328 qua seriport bằng lệnh:

```

String Request="Motor="+ Motor + " Fan="+ Fan + " Step="+ New + "
Led1="+ Led1 + " Led2="+ Led2;

Serial.println(Request);.

```

Sau đó ESP sẽ chờ phản hồi từ ATMEGA và nhận chuỗi dữ liệu để gửi lên web ở địa chỉ:

```

if (InLine.startsWith("DE")) {

    HTTPClient http;

    link = "http://anhtan102.freeasphost.net/GetStatus?" + InLine;

    http.begin(link);

```

```

int httpCode = http.GET();

http.end();

k=1;

}.

```

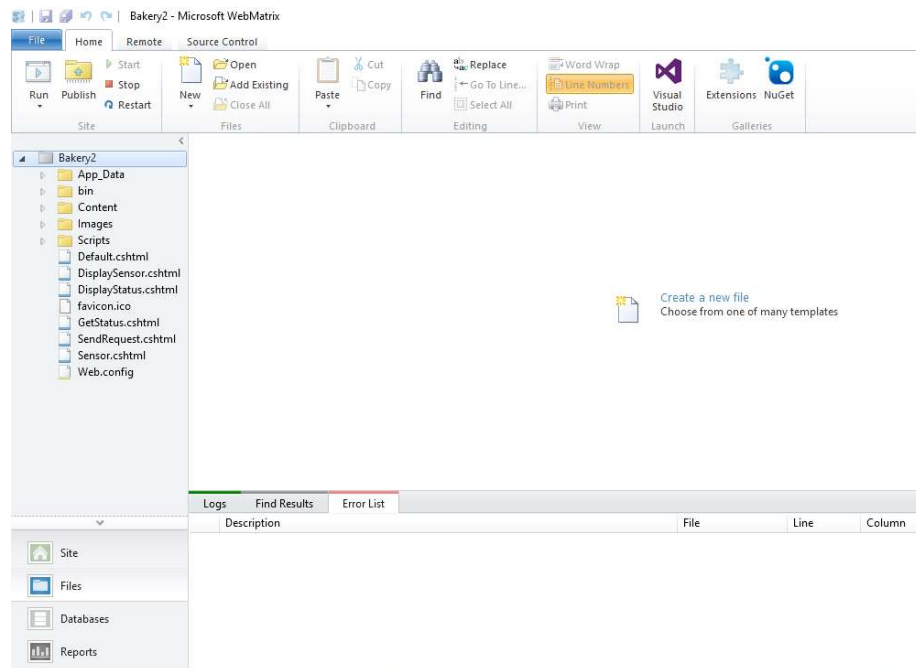
“Inline” là chuỗi dữ liệu mà ESP nhận được từ ATM.

Sau khi thực hiện xong ESP lại gửi lệnh truy cập lên web để lấy nội dung điều khiển của người dùng về và gửi cho ATMEGA328. Cứ như thế lặp đi lặp lại.

4.3. Trang Web và phía Server

4.3.1. Phần mềm Microsoft WebMatrix và ASP.NET framework

Giao diện trang Web, phía Server và cơ sở dữ liệu trong luận văn này đều được tạo trên phần mềm WebMatrix sử dụng ngôn ngữ HTML kết hợp với ASP.NET Razor.



Hình 4.4. Chương trình Microsoft WebMatrix 3

WebMatrix là một phần mềm ứng dụng phát triển Web miễn phí cho hệ điều hành Windows được phát hành bởi Microsoft. WebMatrix cho phép người phát triển xây dựng trang Web sử dụng các mẫu tích hợp có sẵn hoặc các ứng dụng mã nguồn mở thông dụng với sự hỗ trợ tuyệt đối với ASP.NET, PHP, Node.js và HTML5. Microsoft phát triển WebMatrix cho mục đích cung cấp cho người phát triển Web các tính năng

lập trình, chỉnh sửa và phát hành trang Web chỉ trên một phần mềm duy nhất. Phiên bản mới nhất hiện nay là Microsoft WebMatrix 3.

ASP.NET là một chương trình khung ứng dụng Web phía Server thiết kế cho việc phát triển Web để tạo các trang Web động. Được phát triển bởi Microsoft cho phép người phát triển Web xây dựng các trang Web động, ứng dụng Web và dịch vụ Web.

ASP.NET Web pages, được biết chính thức với tên Web Forms, là các khối xây dựng chính cho phát triển ứng dụng. Web forms được chứa trong các tập tin với phần mở rộng “.aspx”, các tập tin này thường chứa các đánh dấu động HTML cũng như các đánh dấu định nghĩa các điều khiển phía Server và các điều khiển của người dùng. Các mã động được chạy trên Server có thể được đặt trong một khối `<% -- mã động -- %>`.

Phần mềm WebMatrix hỗ trợ ASP.NET Razor. Razor là cú pháp lập trình của ASP.NET sử dụng để tạo trang Web động với ngôn ngữ lập trình C# hoặc VisualBasic.NET. Razor cho phép người dùng sử dụng quy trình xây dựng của HTML. Thay vì sử dụng cú pháp đánh dấu với các ký tự `<% -- %>` để chỉ ra khối mã lệnh, cú pháp Razor bắt đầu khối mã lệnh với một ký tự @ duy nhất và không cần thêm ký tự nào để đóng khối mã lệnh. Ưu điểm của Razor là cung cấp một cú pháp tối ưu cho HTML sử dụng một cách tiếp cận tập trung vào mã lệnh, với sự thay đổi nhỏ nhất giữa HTML và mã lệnh.

4.3.2. Tạo cơ sở dữ liệu

Cơ sở dữ liệu được tạo bằng Microsoft SQL Server 2012 trên phần mềm WebMatrix. Cơ sở dữ liệu gồm có các bảng dùng để lưu trữ dữ liệu được gửi lên từ vi điều khiển và Sim900A, đồng thời cũng lưu trữ các yêu cầu được chọn ở giao diện trang Web.

Cơ sở dữ liệu có tên iot, gồm có các bảng:

- Bảng Devices gồm có các cột: cột id chứa số thứ tự, cột Name chứa tên của các thiết bị, cột DStatus chứa trạng thái của các thiết bị, cột DRequest chứa yêu cầu từ dao diện trang Web.

id	Name	DStatus	DRequest
1	Motor	NULL	off
2	Fan	NULL	off
3	Step	on1	on2
4	Led1	NULL	off
5	Led2	NULL	off
6	Led3	NULL	off

Hình 4.5. Bảng Devices của cơ sở dữ liệu

- Bảng Sensor gồm có các cột: cột id chứa số thứ tự, cột Name chứa tên cảm biến, cột Status chứa trạng thái của cảm biến.

id1	Name	Status
1	AnhSang	NULL
2	DoAm	NULL
3	NhietDo	NULL

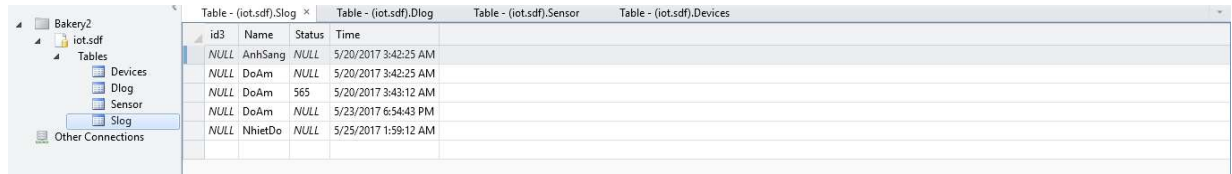
Hình 4.6. Bảng Sensor của cơ sở dữ liệu

- Bảng DLog gồm có các cột: cột id chứa số thứ tự, cột Name chứa tên của thiết bị, cột Status chứa trạng thái vừa thay đổi của thiết bị, cột ATime chứa thời gian thay đổi của thiết bị.

id	Name	Status	ATime
NULL	Step	on1	5/25/2017 4:52:00 AM

Hình 4.7. Bảng DLog của cơ sở dữ liệu

- Bảng SLog gồm các cột: cột id chứa số thứ tự, cột Time chứa thời gian khi có sự thay đổi giá trị của các cảm biến – giá trị cột Status trong bảng Sensor là 1.



id3	Name	Status	Time
NULL	AnhSang	NULL	5/20/2017 3:42:25 AM
NULL	DoAm	NULL	5/20/2017 3:42:25 AM
NULL	DoAm	565	5/20/2017 3:43:12 AM
NULL	DoAm	NULL	5/23/2017 6:54:43 PM
NULL	NhietDo	NULL	5/25/2017 1:59:12 AM

Hình 4.8. Bảng SLog của cơ sở dữ liệu

4.3.3. Trang hiển thị chính của trang Web

Trang hiển thị chính của trang Web được viết bằng HTML – Ngôn ngữ đánh dấu siêu văn bản, và các phần truy cập vào cơ sở dữ liệu phía Server sử dụng Razor.

Trang hiển thị chính của trang Web có 3 phần chính: phần hiển thị trạng thái của cảm biến, phần hiển thị trạng thái của thiết bị và phần hiển thị các lựa chọn để điều khiển thiết bị.



Hình 4.9. Trang hiển thị chính của trang Web

Với phần hiển thị trạng thái cảm biến và hiển thị trạng thái của thiết bị, trang chính sẽ tải từ hai trang phụ là “DisplaySensor.cshtml” và “DisplayStatus.cshtml”. Trang chính sẽ tải hai trang này liên tục với khoảng trễ 500 ms để hiển thị tức thời trạng thái của cảm biến và thiết bị. Để thực hiện việc này, ta sử dụng đoạn mã JavaScript như sau:

```
<script>
```

```
function Load_external_content(){
```

```
    $('#status').load('DisplayStatus.cshtml');
```

```
    $('#sensor').load('DisplaySensor.cshtml');
```



```

    }

    setInterval('Load_external_content()', 500);

</script>

```

Đoạn mã này sẽ tải trang “DisplaySensor.cshtml” vào phần được đặt trong phần đánh dấu có id="sensor" và tải trang “DisplayStatus.cshtml” vào phần được đặt trong phần đánh dấu có id="status". setInterval là phương pháp gọi hàm sau một khoảng thời gian tính bằng ms và thực thi hàm đó. Ở đây setInterval sẽ gọi hàm “Load_external_content()” sau khoảng thời gian 500 ms.

Với phần hiển thị các lựa chọn để điều khiển các thiết bị, sử dụng các Form để lựa chọn. Đoạn mã của một form như sau:

```

<form method="post">

    <select name="Choice1">

        <option value="on">On</option>

        <option value="off">Off</option>

    </select>

    <input id="button1" type="submit" value="Submit 1"
name="Button"/>

</form>

```

Người dùng có thể lựa chọn hai lựa chọn là On và Off, sau đó nhấn nút Submit để xác nhận lựa chọn. Khi nhấn nút Submit, Form sẽ tạo một yêu cầu POST đến phía Server để thực thi việc lưu giá trị value vào cơ sở dữ liệu.

Nếu nhấn nút Submit 1, phía Server sẽ thực hiện cập nhật cột DRequest của bảng Devices tương ứng với id là 1. Tương tự, sẽ cập nhật cột DRequest tương ứng với id là 2 hoặc 3 nếu nhấn nút Submit 2 hoặc Submit 3.

Để cập nhật giá trị cơ sở dữ liệu ta sử dụng cú pháp câu lệnh SQL:

```

UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;

```

Phía Server được viết bằng ASP.NET Razor. Để kết nối với cơ sở dữ liệu ta sử dụng phương pháp Database.Open(filename). Để thực thi các câu lệnh SQL, ta sử dụng Database.Execute(SQLstatement[,parameters]). Đoạn mã ASP.NET Razor để thực hiện cập nhật cột DRequest tương ứng với id là 1 khi có yêu cầu POS như sau:

```

@{

    var request1 = "";

```

```

var db = Database.Open("GPRS Demo");

var SQLUPDATE01 = "UPDATE Devices Set DRequest=@0 WHERE
id=1";

if(IsPost){
    if(Request["Button"].Equals("Submit 1")){
        request1 = Request["Choice1"];
        db.Execute(SQLUPDATE01, request1);
    }
}
}

```

4.3.4. Trang hiển thị trạng thái cảm biến và thiết bị

Để lựa chọn dữ liệu trong một bảng để hiển thị từ cơ sở dữ liệu ta sử dụng cú pháp câu lệnh SQL như sau:

```

SELECT column_name,column_name
FROM table_name;

```

Hoặc để lựa chọn toàn bộ dữ liệu trong một bảng ta sử dụng cú pháp câu lệnh SQL như sau :

```

SELECT * FROM table_name;

```

Với trang hiển thị trạng thái cảm biến “DisplaySensor.cshtml”, ta sẽ lấy dữ liệu từ bảng Sensor. Nếu như giá trị của cột Status là 1 thì sẽ hiển thị “Movement Detected”, nếu giá trị là 0 thì sẽ hiển thị “No movement”. Đoạn mã ASP.NET để lấy dữ liệu từ cơ sở dữ liệu như sau:

```

@{
var db = Database.Open("iot");
var sqlQ1 = "SELECT * FROM Sensor WHERE id1=1";
var sqlQ2 = "SELECT * FROM Sensor WHERE id1=2";
var sqlQ3 = "SELECT * FROM Sensor WHERE id1=3";
var selectQueryString = "SELECT * FROM SLog ORDER BY id3";
var data1 = db.QuerySingle(sqlQ1);
var sstt1 = data1.Status;
var Name1 = data1.Name;
var data2 = db.QuerySingle(sqlQ2);
var sstt2 = data2.Status;
var Name2 = data2.Name;
var data3 = db.QuerySingle(sqlQ3);
var sstt3 = data3.Status;

```

```

var Name3 = data3.Name;
var Display = "";

if(sstt1 == "0"){
    Display = "Không Có Ánh Sáng. ";
}
if(sstt1 == "1"){
    Display = "Có Ánh Sáng. ";
}
Display = Display + "Độ Ẩm Hiện Tại: "+sstt2+ ". Nhiệt Độ Hiện Tại: "
+sstt3+ ".";
}

```

Để hiển thị giá trị của biến Display trong đoạn mã HTML, chỉ cần gọi @Display.

Đối với trang hiển thị trạng thái thiết bị “DisplayStatus.cshtml”, ta sẽ lấy các giá trị của cột DStatus trong bảng Devices để hiển thị bằng các dòng lệnh như sau:

```

<ul>

@foreach(var row in data){

    <li>@row.Name is @row.DStatus.</li>

}

</ul>

```

Hiển thị thời gian thay đổi trạng thái của các thiết bị bằng phương pháp tương tự như trong trang hiển thị trạng thái cảm biến.

4.3.5. Trang Web phía Server

Gồm hai trang riêng biệt, giá trị của cảm biến sẽ được gọi lên trang “Sensor.cshtml” và giá trị của các thiết bị sẽ được gọi lên trang “GetStatus.cshtml”.

Các giá trị biến được gọi lên các trang này nhờ vào giao thức chuyển tải siêu văn bản (HTTP). Hai phương pháp thường được sử dụng là GET và POST. Ở đây sử dụng phương pháp GET. Chuỗi truy vấn (các cặp tên/giá trị) của phương pháp GET được gọi trong đường dẫn URL như sau:

/ GetStatus.cshtml?name1=value1&name2=value2

Trang GetStatus.cshtml” sẽ đọc giá trị từ đường dẫn và lưu vào cơ sở dữ liệu. Đồng thời sẽ ghi lại thời gian thay đổi của các giá trị vào cơ sở dữ liệu.

Để đọc giá trị biến từ chuỗi truy vấn HTTP, sử dụng cú pháp Request.QueryString(variable).

@{

```

var db = Database.Open("iot");
var ustt1 = Request.QueryString["Motor"];
var ustt2 = Request.QueryString["Fan"];
var ustt3 = Request.QueryString["Step"];
var ustt4 = Request.QueryString["Led1"];
var ustt5 = Request.QueryString["Led2"];
var ustt6 = Request.QueryString["AnhSang"];
var ustt7 = Request.QueryString["DoAm"];
var ustt8 = Request.QueryString["NhietDo"];

var SQLUPDATE1 = "UPDATE Devices Set DStatus=@0 WHERE id=1";
var SQLUPDATE2 = "UPDATE Devices Set DStatus=@0 WHERE id=2";
var SQLUPDATE3 = "UPDATE Devices Set DStatus=@0 WHERE id=3";
var SQLUPDATE4 = "UPDATE Devices Set DStatus=@0 WHERE id=4";
var SQLUPDATE5 = "UPDATE Devices Set DStatus=@0 WHERE id=5";
var SQLUPDATE6 = "UPDATE Sensor Set Status=@0 WHERE id1=1";
var SQLUPDATE7 = "UPDATE Sensor Set Status=@0 WHERE id1=2";
var SQLUPDATE8 = "UPDATE Sensor Set Status=@0 WHERE id1=3";
var sqlQ1 = "SELECT * FROM Devices WHERE id=1";
var sqlQ2 = "SELECT * FROM Devices WHERE id=2";
var sqlQ3 = "SELECT * FROM Devices WHERE id=3";
var sqlQ4 = "SELECT * FROM Devices WHERE id=4";
var sqlQ5 = "SELECT * FROM Devices WHERE id=5";
var sqlQ6 = "SELECT * FROM Sensor WHERE id1=1";
var sqlQ7 = "SELECT * FROM Sensor WHERE id1=2";
var sqlQ8 = "SELECT * FROM Sensor WHERE id1=3";
var SQLINSERTD = "INSERT INTO DLog (Name,Status,ATime) VALUES
(@0,@1,@2)";
var SQLINSERTS = "INSERT INTO SLog (Name,Status,Time) VALUES
(@0,@1,@2)";

string vnTimeZoneKey = "SE Asia Standard Time";
TimeZoneInfo vnTimeZone = TimeZoneInfo.FindSystemTimeZoneById(vnTimeZoneKey);
;
DateTime timelog =
TimeZoneInfo.ConvertTimeFromUtc(DateTime.UtcNow,vnTimeZone);

var data1 = db.QuerySingle(sqlQ1);
var data2 = db.QuerySingle(sqlQ2);
var data3 = db.QuerySingle(sqlQ3);
var data4 = db.QuerySingle(sqlQ4);
var data5 = db.QuerySingle(sqlQ5);
var data6 = db.QuerySingle(sqlQ6);
var data7 = db.QuerySingle(sqlQ7);
var data8 = db.QuerySingle(sqlQ8);
if(String.Compare(ustt1,data1.DStatus) != 0){
    db.Execute(SQLINSERTD,"Motor",ustt1,timelog);
    db.Execute(SQLUPDATE1, ustt1);
}
if(String.Compare(ustt2,data2.DStatus) != 0){
    db.Execute(SQLINSERTD,"Fan",ustt2,timelog);
    db.Execute(SQLUPDATE2, ustt2);
}
if(String.Compare(ustt3,data3.DStatus) != 0){
    db.Execute(SQLINSERTD,"Step",ustt3,timelog);
    db.Execute(SQLUPDATE3, ustt3);
}

```

```

    }
    if(String.Compare(ustt4,data4.DStatus) != 0){
        db.Execute(SQLINSERTD,"Led1",ustt4,timelog);
        db.Execute(SQLUPDATE4, ustt4);
    }
    if(String.Compare(ustt5,data5.DStatus) != 0){
        db.Execute(SQLINSERTD,"Led2",ustt5,timelog);
        db.Execute(SQLUPDATE5, ustt5);
    }
    if(String.Compare(ustt6,data6.Status) != 0){
        db.Execute(SQLINSERTS,"AnhSang",ustt6,timelog);
        db.Execute(SQLUPDATE6, ustt6);
    }
    if(String.Compare(ustt7,data7.Status) != 0){
        db.Execute(SQLINSERTS,"DoAm",ustt7,timelog);
        db.Execute(SQLUPDATE7, ustt7);
    }
    if(String.Compare(ustt8,data8.Status) != 0){
        db.Execute(SQLINSERTS,"NhietDo",ustt8,timelog);
        db.Execute(SQLUPDATE8, ustt8);
    }
}
}

```

4.3.6. Trang hiển thị yêu cầu cho thiết bị

Trang hiển thị yêu cầu “SendRequest.cshml”, có nhiệm vụ hiển thị các yêu cầu từ bảng Devices của cơ sở dữ liệu để SIM900A có thể đọc và gửi cho vi điều khiển. Yêu cầu của một thiết bị được bắt đầu bằng kí tự <#> và kết thúc bằng ký tự <.> để dễ dàng nhận biết.

Mã của trang này như sau :

```

@{
    var db= Database.Open("iot");

    var sqlQ = "SELECT * FROM Devices";

    var data = db.Query(sqlQ);
}

<html lang="en">

<body>

    @foreach(var row in data){

        <p>#@row.id @row.DRequest .</p>

    }

</body></html>

```

5. KẾT QUẢ THỰC HIỆN

5.1. Đăng tải trang Web lên mạng Internet

Phần trang Web sau khi được viết trên WebMatrix và chạy thử trên LocalHost được tải lên host để đăng tải lên mạng. Trong đồ án này lựa chọn sử dụng dịch vụ host của Freeasphost.net vì có cung cấp dịch vụ host miễn phí đồng thời hỗ trợ ASP.NET và cung cấp một tên miền con cho trang Web và không có quảng cáo.

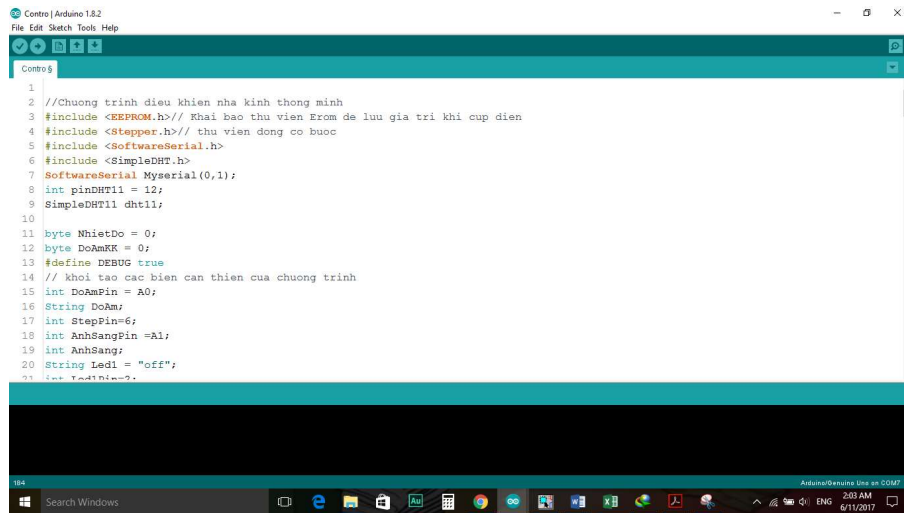
Dịch vụ host của Freeasphost cho phép đăng tải trang Web qua cách tải lên các tập tin một cách đơn giản. Trang Web đã được đưa lên host có tên miền “www.anhtan102.freeasphost.net”.



Hình 5.1. Trang Web sau khi được đăng tải lên mạng

5.2. Phần mềm của vi điều khiển

Phần mềm của ATMEGA328 và ESP8266 được viết trên IDE của Arduino. Trên này có các thư viện hỗ trợ lập trình cũng như tải code lên vi điều khiển một cách đơn giản và thuận tiện trong quá trình thử nghiệm.

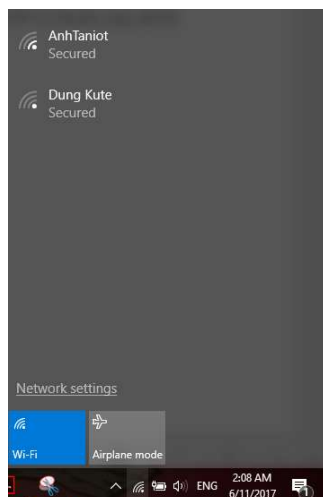


Hình 5.2. Phần mềm IDE

5.3. Hoạt động của toàn hệ thống

Sau khi thiết kế thi công và nạp code vào các vi điều khiển ta tiến hành chạy thử nghiệm.

Lúc mới khởi động do ERROM chưa lưu sẵn nội dụng nên ESP sẽ chuyển sang chế độ Websever và làm một điểm phát wifi có tên "AnhTaniot".



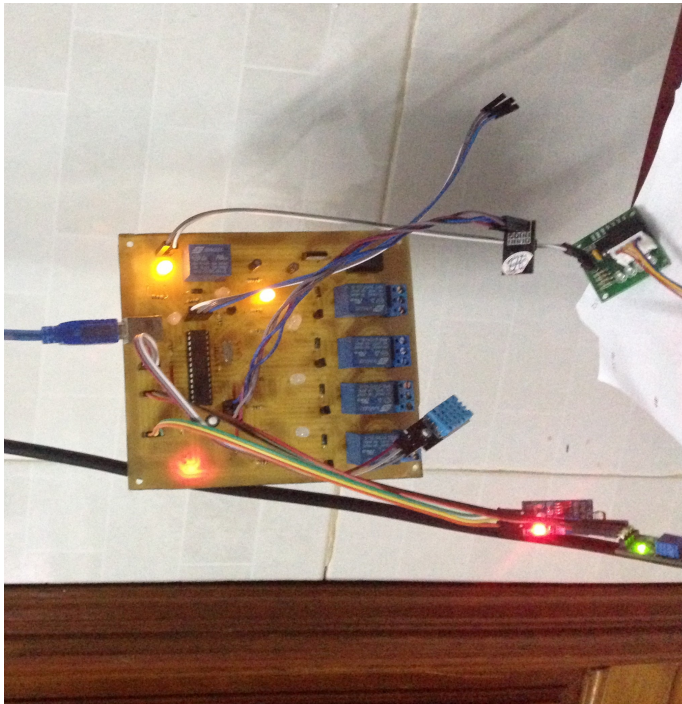
Hình 5.3. Tên truy cập wifi

Sau khi đăng nhập vào wifi. Ta mở trình duyệt web và truy cập vào địa chỉ 192.168.4.1 thì ta được giao diện sau. Và điền wifi mình muốn cho hệ thống đăng nhập. Sau khi điền chính xác và nhấn nút Submit. Trang web trả về thông tin :
 {"Success": "Luu vao he thong. Khoi dong lai ten wifi moi"}. Như vậy hệ thống đã cập nhật lại điểm truy cập Wifi và lưu vào ERROM.



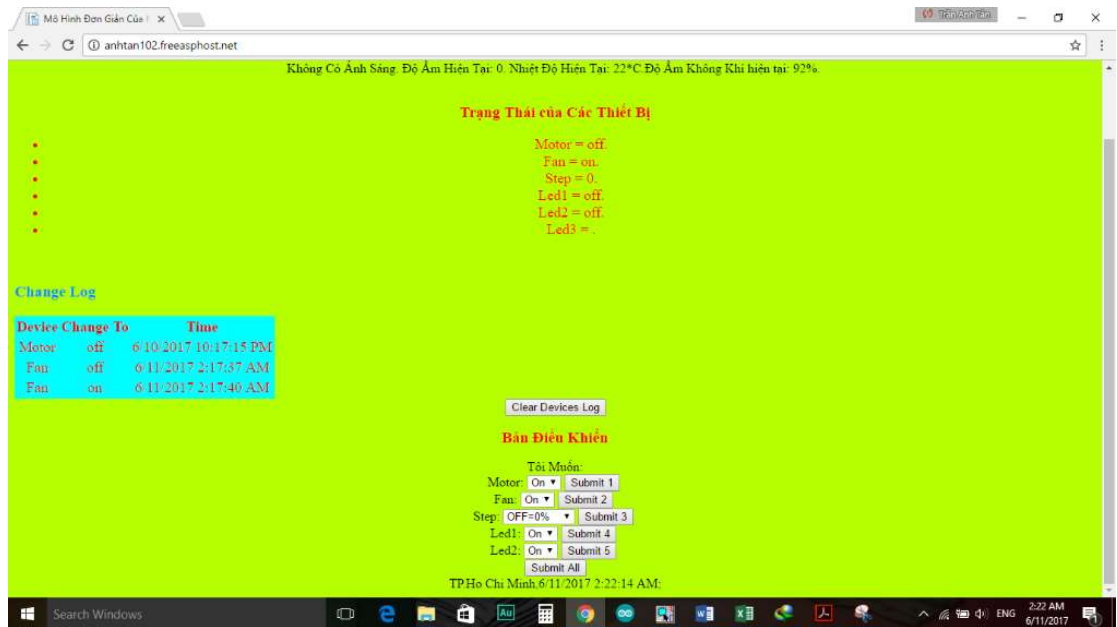
Hình 5.4. webserver offline.

Ta tiến hành khởi động lại hệ thống.



Hình 5.5. Phần cứng khi hoạt động.

Giao diện web. Ta thấy các thông số cảm biến: Không có ánh sáng, Độ ẩm hiện tại = 0. Nhiệt độ hiện tại 22. Độ ẩm không khí hiện tại là 92%. Khi tra dữ liệu từ ứng dụng của iphone thì nhiệt độ là 22.



Hình 5.6. Dữ liệu web nhận được.



Hình 5.7. Đối chiếu nhiệt độ.

Hệ thống hoạt động tốt. Có delay không đáng kể.

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. *Kết luận*

Trong quá trình thực hiện đề tài, với sự hạn chế về thời gian và tài liệu vì vậy đòi hỏi bản thân phải thật cố gắng tìm tòi và nghiên cứu để hoàn thành đề tài một cách trọn vẹn. Quá trình thực hiện đề tài đã giúp rèn luyện khả năng tự tìm hiểu và tổng hợp các nguồn tài liệu, đồng thời rèn luyện các kỹ năng đã được học trong suốt quá trình học tập tại trường như thiết kế, thi công mạch in, viết chương trình cho vi điều khiển, ... cũng như biết thêm được các kiến thức mới như thiết kế trang Web đơn giản.

Hệ thống còn khá thô và nhiều đây nên thiếu thẩm mỹ.

6.2. *Hướng phát triển*

Hệ thống này có thể phát triển phía phần mềm, xây dựng cơ sở dữ liệu dữ liệu chặt chẽ hơn, thêm chức năng đăng nhập để đảm bảo sự bảo mật của người dùng. Có thể kết hợp các chức năng hẹn giờ, tự xử lý các tác vụ...

Với hệ thống wifi ngày càng phát triển và rộng khắp thì đây là một ứng dụng dành cho tất cả mọi người mọi nhà có nhu cầu.

Hệ thống này có thể được sử dụng trong cuộc sống hàng ngày như trong các hệ thống nhà thông minh. Nó còn có thể ứng dụng trong các ứng dụng an ninh để theo dõi cơ quan, nhà xưởng từ xa hoặc ứng dụng trong công nghiệp để theo dõi trạng thái của các thiết bị từ xa, hoặc theo dõi ở những nơi khó tiếp cận, điều kiện khắc nghiệt.

Nhìn chung, đây là một đề tài có tính ứng dụng cao, có tính khả thi, phù hợp với khả năng kinh tế của hầu hết mọi người và góp phần nâng cao chất lượng đời sống của con người.

7. TÀI LIỆU THAM KHẢO

- [1] Nguyễn Như Anh, “Kỹ thuật xung”, Nhà Xuất Bản Đại Học Quốc Gia TP. Hồ Chí Minh, 2007.
- [2] Hồ Trung Mỹ, “Vi Xử Lý”, Nhà Xuất Bản Đại Học Quốc Gia TP. Hồ Chí Minh, 2006.
- [3] Lê Tiến Thường, “Mạch Điện Tử 1”, Nhà Xuất Bản Đại Học Quốc Gia TP. Hồ Chí Minh, 2000.
- [4] “ASP.NET Tutorial”, [Online] <http://www.w3schools.com/aspnet/default.asp>.
- [5] Lưu Trọng Nhân, “LVTN_Luu Trong Nhan_LVTN_GPRS”.

8. PHỤ LỤC

8.1. Code cho vi điều khiển ATMEGA328:

```
//Chương trình điều khiển nhà kính thông minh

#include <EEPROM.h> // Khai báo thư viện Erom để lưu giá trị khi cúp điện

#include <Stepper.h> // thư viện động cơ bước

#include <SoftwareSerial.h>

#include <SimpleDHT.h>

SoftwareSerial Myserial(0,1);

int pinDHT11 = 12;

SimpleDHT11 dht11;

byte NhietDo = 0;

byte DoAmKK = 0;

#define DEBUG true

// khởi tạo các biến cần thiết của chương trình

int DoAmPin = A0;

String DoAm;

int StepPin=6;

int AnhSangPin =A1;

int AnhSang;

String Led1 = "off";

int Led1Pin=2;

String Led2 = "off";

int Led2Pin=3;

String Motor ="off";

int MotorPin=4;
```

```

String Fan = "off";

int FanPin=5;

int StepSet =64;

Stepper MyStepper(StepSet,7,8,9,10);// cai dat cac chan ra cua don co step

char x;

int New=0;

int Old;

void setup()
{
    pinMode(13,OUTPUT);// cai dat chan 13 la chan ra
    MySerial.begin(115200); // chu y phai cung toc do voi ESP 8266
    pinMode(Led1Pin,OUTPUT);// cai dat chan 2 la chan ra
    digitalWrite(Led1Pin,LOW);// tinh cuc muc thap
    pinMode(Led2Pin,OUTPUT);// cai dat chan 2 la chan ra
    digitalWrite(Led2Pin,LOW);// tinh cuc muc thap
    pinMode(MotorPin,OUTPUT);// cai dat chan 2 la chan ra(motor)
    digitalWrite(MotorPin,LOW);// tinh cuc muc cao
    pinMode(FanPin,OUTPUT);// cai dat chan 2 la chan ra(motor)
    digitalWrite(FanPin,LOW);// tinh cuc muc cao
    pinMode(AnhSangPin,INPUT);
    pinMode(StepPin,OUTPUT);
    digitalWrite(StepPin,LOW);
    MyStepper.setSpeed(250);
    digitalWrite(13,LOW);
}

void loop()

```

```

    { digitalWrite(13,LOW);
      while(Myserial.available())
      {
        IncomingChar(Myserial.read ());
        digitalWrite(13,HIGH);
      }
    }

void IncomingChar (const byte InChar)//ham nhan gai tri vao tu seriport
{
  static char InLine [300];
  static unsigned int Position = 0;
  switch (InChar)
  {
    case 'r': // Don't care about carriage return so throw away.
      break;

    case 'n': // ket thuc mot ban tin
      InLine [Position] = 0;
      ProcessCommand(String(InLine));//xu ly tin hieu tu chuoai ky
tu nhan duoc tu seriport
      Position = 0;
      break;

    default:
      InLine [Position++] = InChar;
    }
  }

void ProcessCommand(String InLine)// ham so sanh va thuc hien lenh nhan
duoc tu seriort

```

```

{
  if(InLine.startsWith("Motor"))
  {
    if(InLine.indexOf("Led1=off")!= -1)
    {
      Led1="off";
      digitalWrite(Led1Pin,LOW);
    }
    if(InLine.indexOf("Led1=on")!= -1)
    {
      Led1="on";
      digitalWrite(Led1Pin,HIGH);
    }
    if(InLine.indexOf("Led2=off")!= -1)
    {
      Led2="off";

      digitalWrite(Led2Pin,LOW);
    }
    if(InLine.indexOf("Led2=on")!= -1)
    {
      Led2="on";
      digitalWrite(Led2Pin,HIGH);
    }
    if(InLine.indexOf("Motor=on")!= -1)
    {

```

```

    Motor = "on";

    digitalWrite(MotorPin,HIGH);
}

if(InLine.indexOf("Motor=off")!= -1)
{
    Motor = "off";

    digitalWrite(MotorPin,LOW);

}

if(InLine.indexOf("Fan=off")!= -1)
{
    Fan = "off";

    digitalWrite(FanPin,LOW);

}

if(InLine.indexOf("Fan=on")!= -1)
{
    Fan = "on";

    digitalWrite(FanPin,HIGH);

}

if(InLine.indexOf("Step=0")!= -1)
{
    New = 0;

    Old=EEPROM.read(0);

    if(New!=Old)
    {

        digitalWrite(StepPin,HIGH);

        StepController(Old,New);
    }
}

```



```

EEPROM.write(0,New);

}

digitalWrite(StepPin,LOW);

}

if(InLine.indexOf("Step=30")!=-1)
{
    New = 30;

    Old=EEPROM.read(0);
    if(New!=Old)
    {
        digitalWrite(StepPin,HIGH);

        StepController(Old,New);

        EEPROM.write(0,New);

    }

    digitalWrite(StepPin,LOW);

}

if(InLine.indexOf("Step=60")!=-1)
{
    New = 60;

    Old=EEPROM.read(0);
    if(New!=Old)
    {
        digitalWrite(StepPin,HIGH);

        StepController(Old,New);

        EEPROM.write(0,New);

    }
}

```

```

    digitalWrite(StepPin,LOW);
}
if(InLine.indexOf("Step=100")!= -1)
{
    New = 100;
    Old=EEPROM.read(0);
    if(New!=Old)
    {
        digitalWrite(StepPin,HIGH);
        StepController(Old,New);
        EEPROM.write(0,New);
    }
    digitalWrite(StepPin,LOW);
}
if(analogRead(AnhSangPin)<200)
{ AnhSang=1;}
else
{AnhSang=0;}
long DoAmI=analogRead(DoAmPin);
DoAm=100-DoAmI*100/1023;
dht11.read(pinDHT11, &NhietDo, &DoAmKK,NULL);

String Getstatus ="DE,&Motor="
+Motor+"&Fan="
+Fan+"&Step="
+New+ "&Led1="
+Led1+"&Led2="

```

```

+Led2+"&DoAm="
+DoAm+"&AnhSang="
+AnhSang+"&NhietDo="
+(int)NhietDo+"&DoAmKK="+ (int)DoAmKK;
Myserial.println(Getstatus);

}

}

void StepController(int ol,int ne)// ham chay dong co buoc
{
    int Step=(ne-ol)*2048;
    MyStepper.step(Step);
}

```

8.2. Code ESP8266

```

//chuong trinh gao tiep esp voi mang va may tinh
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WebServer.h>
#include "EEPROM.h"
ESP8266WebServer server(80);

const char* ssid = "AnhTaniot";//ten wifi o che do phat wifi
const char* passphrase = "123698745";

String st;

String content;

int statusCode;

String link;

```

```

static String Led1 = "off";
static String Led2 = "off";
static String Motor = "off";
static String Fan = "off";
static int New = 0;
static int k=1;

void setup()
{
  Serial.begin(115200);
  EEPROM.begin(512);
  delay(10);
  Serial.println("Startup");
  // read eeprom for ssid, pass and blynk
  Serial.println("Reading EEPROM ssid");
  String esid;
  for (int i = 0; i < 32; ++i)
  {
    esid += char(EEPROM.read(i)); // do noi dung tu bo nho erom
  }
  Serial.print("SSID: ");
  Serial.println(esid.c_str());
  esid.trim();
  Serial.println("Reading EEPROM pass");
  String epass = "";
  for (int i = 32; i < 96; ++i)
  {

```

```

    epass += char(EEPROM.read(i));
}
Serial.print("PASS: ");
Serial.println(epass.c_str());
epass.trim();
EEPROM.end();
if ( esid.length() > 1 )
{
    WiFi.begin(esid.c_str(), epass.c_str());
    if (!testWifi())
    {
        setupAP();
        int k=1;
        while(k)
        {
            server.handleClient();//khai tao sever
            if(WiFi.status() == WL_CONNECTED)
            {k=0;}
        }
        EEPROM.end();
    }
}
WiFi.softAPdisconnect(true);//tat cho do phat wifi
}
bool testWifi(void)
{

```

```

int c = 0;

Serial.println("Xin vui long doi ket noi WIFI");

while ( c < 20 )
{
    if (WiFi.status() == WL_CONNECTED)
    {
        Serial.println("");
        Serial.println("Connected");
        return true;
    }
    delay(1000);
    Serial.print(WiFi.status());

    c++;
}

Serial.println("");
Serial.println("Thoi gian ket noi cham, Mo AP");
return false;
}

void setupAP(void)
{
    WiFi.mode(WIFI_STA);//cai dat che do wifi
    WiFi.disconnect();
    delay(100);

    int n = WiFi.scanNetworks();// quet cac wifi gan day
    Serial.println("Tim hoan tat");

    if (n == 0)

```

```

{
    Serial.println("no networks found");
}
else
{
    Serial.print(n);

    Serial.println(" networks found");

    for (int i = 0; i < n; ++i)
    {
        // Print SSID and RSSI for each network found

        Serial.print(i + 1);

        Serial.print(": ");

        Serial.print(WiFi.SSID(i));

        Serial.print(" (");

        Serial.print(WiFi.RSSI(i));

        Serial.print(")");

        Serial.println((WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " " :
        "**");

        delay(10);
    }
}

Serial.println("");

st = "<ol>";

for (int i = 0; i < n; ++i)
{
    // Print SSID and RSSI for each network found

    st += "<li>";

```

```

    st += WiFi.SSID(i);

    st += " (";

    st += WiFi.RSSI(i);

    st += ")";

    st += (WiFi.encryptionType(i) == ENC_TYPE_NONE) ? " " : "**";

    st += "</li>";

}

st += "</ol>";

delay(100);

Serial.println("softap");

Serial.println(ssid);

Serial.println(passphrase);

WiFi.softAP(ssid, passphrase, 6);


launchWeb(1);

Serial.println("over");

}

void launchWeb(int webtype)//chay trang web sever khi co yeu cau
{

    Serial.println("");

    Serial.println("WiFi ket noi");

    Serial.print("Dia chi IP: ");

    Serial.println(WiFi.localIP());

    Serial.print("SoftAP IP: ");

    Serial.println(WiFi.softAPIP());

    createWebServer(webtype);

```



```

// Start the server

server.begin();

Serial.println("May chu bat dau");
}

void createWebServer(int webtype)
{
    if ( webtype == 1 )
    {
        server.on("/", []()
        {
            IPAddress ip = WiFi.softAPIP();

            String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.'
+ String(ip[3]);

            content = "<!DOCTYPE HTML>\r\n<html><h2>XSwitch</h2>";

            content += "<form method=\"get\" action=\"/setting\">";

            content += "<div>Wifi</div>";

            content += "<div><input name=\"ssid\" size=\"40\"></div>";

            content += "<div>Mat Khau</div>";

            content += "<div><input name=\"pass\" size=\"40\"></div>";

            content += "<div><input type='submit'></div>";

            content += "<p>";

            content += st;

            content += "</p>";

            content += "</html>";

            server.send(200, "text/html", content);

        });

        server.on("/setting", []()

```

```

{
    String qsid = server.arg("ssid");
    String qpass = server.arg("pass");
    if (qsid.length() > 0 && qpass.length() > 0)
    {
        EEPROM.begin(512);
        Serial.println("clearing eeprom");
        for (int i = 0; i < 128; ++i)
        {
            EEPROM.write(i, 0);
        }
        EEPROM.commit();
        Serial.println(qsid);
        Serial.println("");
        Serial.println(qpass);
        Serial.println("");
        Serial.println("writing eeprom ssid:");
        for (int i = 0; i < qsid.length(); ++i)
        {
            EEPROM.write(i, qsid[i]);
            Serial.print("Wrote: ");
            Serial.println(qsid[i]);
        }
        Serial.println("writing eeprom pass:");
        for (int i = 0; i < qpass.length(); ++i)
        {

```

```

        EEPROM.write(32 + i, qpass[i]);

        Serial.print("Wrote: ");

        Serial.println(qpass[i]);

    }

    EEPROM.commit();

    EEPROM.end();

    content = "{\"Success\":\"Luu vao he thong. Khoi dong lai ten wifi
moi\"}";

    statusCode = 200;

}

else

{

    content = "{\"Error\":\"404 not found\"}";

    statusCode = 404;

    Serial.println("Sending 404");

}

server.send(statusCode, "application/json", content);

});

}

}

void loop()

{

    while (Serial.available())

    {

        IncomingChar(Serial.read ());

    }

    if(k){

```

```

while (WiFi.status() == WL_CONNECTED) { //Check WiFi connection status

    HTTPClient http; //Declare an object of class HTTPClient

    http.begin("http://anhtan102.freeasphost.net/sendrequest?"); //Specify
request destination

    int httpCode = http.GET(); //Send
the request

    if (httpCode > 0)
    {
        String payload = http.getString();

        ProcessCommand(payload); //Print the response payload
    }

    http.end();

    break;
}

}

if(WiFi.status() != WL_CONNECTED)
{
    setup();
}

}

void IncomingChar (const byte InChar)
{
    static char InLine [300];

    static unsigned int Position = 0;

    switch (InChar)
    {

```

```

    case '\r':
        k=1; // Don't care about carriage return so throw away.
        break;
    case '\n': // ket thuc mot ban tin
        InLine [Position] = 0;
        ProcessCommand(String(InLine));
        Position = 0;
        break;
    default:
        k=0;
        InLine [Position++] = InChar;
    }
}

void ProcessCommand(String InLine)
{
    if (InLine.startsWith("DE")) {
        HTTPClient http;

        link = "http://anhtan102.freeasphost.net/GetStatus?" + InLine;
        http.begin(link);
        int httpCode = http.GET();
        http.end();
        k=1;
    }
    if (InLine.indexOf("<li") {
        if (InLine.indexOf("Led1 = off") != -1)

```

```
{  
    Led1 = "off";  
}  
  
if (InLine.indexOf("Led1 = on") != -1)  
  
    {  
        Led1 = "on";  
    }  
  
if (InLine.indexOf("Led2 = off") != -1)  
  
    {  
        Led2 = "off";  
    }  
  
if (InLine.indexOf("Led2 = on") != -1)  
  
    {  
        Led2 = "on";  
    }  
  
if (InLine.indexOf("Motor = on") != -1)  
  
    {  
        Motor = "on";  
    }  
  
if (InLine.indexOf("Motor = off") != -1)  
  
    {  
        Motor = "off";  
    }  
  
if (InLine.indexOf("Fan = off") != -1)  
  
    {  
        Fan = "off";  
    }
```

```

    }

    if (InLine.indexOf("Fan = on") != -1)
    {
        Fan = "on";
    }

    if (InLine.indexOf("Step = 0") != -1)
    {
        New = 0;
    }

    if (InLine.indexOf("Step = 30") != -1)
    {
        New = 30;
    }

    if (InLine.indexOf("Step = 60") != -1)
    {
        New = 60;
    }

    if (InLine.indexOf("Step = 100") != -1)
    {
        New = 100;
    }

    String Request="Motor=" + Motor + " Fan=" + Fan + " Step=" + New + "
    Led1=" + Led1 + " Led2=" + Led2;

    Serial.println(Request);

    k=1;
}

```

}

8.3. Mã lệnh của trang “Default.cshtml”

@{

```

var request1 = "";
var request2 = "";
var request3 = "";
var request4 = "";
var request5 = "";
var request6 = "";
var db = Database.Open("iot");
var SQLUPDATE01 = "UPDATE Devices Set DRequest=@0 WHERE id=1";
var SQLUPDATE02 = "UPDATE Devices Set DRequest=@0 WHERE id=2";
var SQLUPDATE03 = "UPDATE Devices Set DRequest=@0 WHERE id=3";
var SQLUPDATE04 = "UPDATE Devices Set DRequest=@0 WHERE id=4";
var SQLUPDATE05 = "UPDATE Devices Set DRequest=@0 WHERE id=5";
var SQLUPDATE06 = "UPDATE Devices Set DRequest=@0 WHERE id=6";
var SQLDELETE1 = "DELETE FROM SLog";
var SQLDELETE2 = "DELETE FROM DLog";

if(IsPost){
    if(Request["Button"].Equals("Submit 1")){
        request1 = Request["Choice1"];
        db.Execute(SQLUPDATE01, request1);
    }

    if(Request["Button"].Equals("Submit 2")){
        request2 = Request["Choice2"];
        db.Execute(SQLUPDATE02, request2);
    }

    if(Request["Button"].Equals("Submit 3")){
        request3 = Request["Choice3"];
        db.Execute(SQLUPDATE03, request3);
    }

    if(Request["Button"].Equals("Submit 5")){
        request5 = Request["Choice5"];
        db.Execute(SQLUPDATE05, request5);
    }

    if(Request["Button"].Equals("Submit 6")){
        request6 = Request["Choice6"];
        db.Execute(SQLUPDATE06, request6);
    }

    if(Request["Button"].Equals("Submit 4")){
        request4 = Request["Choice4"];
        db.Execute(SQLUPDATE04, request4);
    }

    if(Request["Button"].Equals("Submit All")){
        request1 = Request["Choice1"];
        request2 = Request["Choice2"];
        request3 = Request["Choice3"];
    }
}

```



```

        request4 = Request["Choice4"];
        request5 = Request["Choice5"];
        request6 = Request["Choice6"];

        db.Execute(SQLUPDATE01, request1);
        db.Execute(SQLUPDATE02, request2);
        db.Execute(SQLUPDATE03, request3);
        db.Execute(SQLUPDATE04, request4);
        db.Execute(SQLUPDATE05, request5);
        db.Execute(SQLUPDATE06, request6);
    }

    if(Request["Button"].Equals("Clear Sensor Log")){
        db.Execute(SQLDELETE1);
    }

    if(Request["Button"].Equals("Clear Devices Log")){
        db.Execute(SQLDELETE2);
    }
}

<!DOCTYPE html>
<html lang="en">
<head>
<style>
    body {
        background-color: #b6ff00;
        font-family: 'Times New Roman';
        text-align: center;
    }
div {
    font-family: 'Times New Roman';
    text-align: center;
}
h1 { font-size: x-large;
    background-color: #0ff;
    color: #0026ff;
}
h2 { font-size: larger;
    background-color: #b6ff00;
    color: #f00;
}
</style>

<meta charset="utf-8" />

<title>Mô Hình Đơn Giản Của Nhà Kính Thông Minh Điều Khiển Qua
WebSite</title>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></scri
pt>
<script>

```

```

function Load_external_content() {
    $('#status').load('DisplayStatus.cshtml');

    $('#sensor').load('DisplaySensor.cshtml');
}
setInterval('Load_external_content()', 500);
</script>
</head>

<body>
    <header>
        <h1 class="Title">Mô Hình Đơn Giản Của Nhà Kính Thông Minh Điều
Khiển Qua WebSite</h1>
    </header>

    <h2 id="h2st">Các Cảm Biến</h2>
    <div id="sensor">
        <br>
    </div>

    <h2 id="h2st">Trạng Thái của Các Thiết Bị</h2>
    <div id="status">
        <br>
    </div>
    <form method="post">
        <input id="buttondel" type="submit" value="Clear Devices Log"
name="Button"/>
    </form>

    <h2 id="h2st">Bản Điều Khiển</h2>
    <div id="control">
        Tôi Muốn:
        <form method="post">
            Motor:
            <select name="Choice1">
                <option value="on">On</option>
                <option value="off">Off</option>
            </select>
            <input id="button1" type="submit" value="Submit 1"
name="Button"/>
            <br>
            Fan:
            <select name="Choice2">
                <option value="on">On</option>
                <option value="off">Off</option>
            </select>
            <input id="button2" type="submit" value="Submit 2"
name="Button"/>
            <br>
            Step:
            <select name="Choice3">
                <option value="0">OFF=0%</option>
                <option value="30">ON1=30%</option>
                <option value="60">On2=60%</option>
                <option value="100">On3=100%</option>
            </select>

```

```

        <input id="button3" type="submit" value="Submit 3"
name="Button"/>
    </br>

    Led1:
    <select name="Choice4">
        <option value="on">On</option>
        <option value="off">Off</option>
    </select>
    <input id="button4" type="submit" value="Submit 4"
name="Button"/>
    </br>

    Led2:
    <select name="Choice5">
        <option value="on">On</option>
        <option value="off">Off</option>
    </select>
    <input id="button5" type="submit" value="Submit 5"
name="Button"/>
    </br>

    <input id="button7" type="submit" value="Submit All"
name="Button"/>
    </form>
</div>

<div>

</div>

<div id="footer">
    <footer>
        @{
            string vnTimeZoneKey = "SE Asia Standard Time";
            TimeZoneInfo vnTimeZone = TimeZoneInfo.FindSystemTimeZoneById(vnTimeZoneKey)
            ;
            DateTime time = TimeZoneInfo.ConvertTimeFromUtc(DateTime.UtcNow,vnTimeZone);

        }
        TP.Ho Chi Minh,@time;
    </footer>
</div>
</body>
</html>

```

8.4. Mã lệnh của trang “Sensor.cshtml”

```

@{
    var db = Database.Open("iot");
    var DoAm = Request.QueryString["DoAm"];
    var NhietDo = Request.QueryString["NhietDo"];
    var AnhSang = Request.QueryString["AnhSang"];
    var timeLog = DateTime.Now;
    var SQLUPDATE1 = "UPDATE Sensor Set Status=@0 WHERE id1=1";

```

```

var SQLINSERT1 = "INSERT INTO SLog (Name,Status,Time) VALUES
(@0,@1,@2)";
var SQLUPDATE2 = "UPDATE Sensor Set Status=@0 WHERE id1=2";
var SQLINSERT2 = "INSERT INTO SLog (Name,Status,Time) VALUES
(@0,@1,@2)";
var SQLUPDATE3 = "UPDATE Sensor Set Status=@0 WHERE id1=3";
var SQLINSERT3 = "INSERT INTO SLog (Name,Status,Time) VALUES
(@0,@1,@2)";
var sqlQ1 = "SELECT * FROM Sensor WHERE id1=1";
var sqlQ2 = "SELECT * FROM Sensor WHERE id1=2";
var sqlQ3 = "SELECT * FROM Sensor WHERE id1=3";

var data1 = db.QuerySingle(sqlQ1);
var data2 = db.QuerySingle(sqlQ2);
var data3 = db.QuerySingle(sqlQ3);
if(String.Compare(AnhSang,data1.Status) != 0){
    db.Execute(SQLINSERT1,"AnhSang",AnhSang,timelog);
    db.Execute(SQLUPDATE1, AnhSang);
}
if(String.Compare(NhietDo,data3.Status) != 0){
    db.Execute(SQLINSERT3,"NhietDo",NhietDo,timelog);
    db.Execute(SQLUPDATE3, NhietDo);
}
if(String.Compare(DoAm,data2.Status) != 0){
    db.Execute(SQLINSERT2,"DoAm",DoAm,timelog);
    db.Execute(SQLUPDATE2, DoAm);
}
}
}

```

8.5. Mã lệnh trang *SendRequest.cshtml*

```

@{
    var db = Database.Open("iot");
    var sqlQ = "SELECT * FROM Devices";
    var selectQueryString = "SELECT * FROM DLog ORDER BY id";
    var data = db.Query(sqlQ);
}

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>Status</title>
    </head>
    <body>
        <div id="statusupdate">
            <ul>
                @foreach(var row in data){
                    <li>@row.Name = @row.DRequest.</li>
                }
            </ul>
        </div>
    </body>
</html>

```

8.6. Mã lệnh trang *GetStatus.cshtml*

```
@{
    var db = Database.Open("iot");
    var ustt1 = Request.QueryString["Motor"];
    var ustt2 = Request.QueryString["Fan"];
    var ustt3 = Request.QueryString["Step"];
    var ustt4 = Request.QueryString["Led1"];
    var ustt5 = Request.QueryString["Led2"];
    var ustt6 = Request.QueryString["AnhSang"];
    var ustt7 = Request.QueryString["DoAm"];
    var ustt8 = Request.QueryString["NhietDo"];
    var ustt9 = Request.QueryString["DoAmKK"];
    var SQLUPDATE1 = "UPDATE Devices Set DStatus=@0 WHERE id=1";
    var SQLUPDATE2 = "UPDATE Devices Set DStatus=@0 WHERE id=2";
    var SQLUPDATE3 = "UPDATE Devices Set DStatus=@0 WHERE id=3";
    var SQLUPDATE4 = "UPDATE Devices Set DStatus=@0 WHERE id=4";
    var SQLUPDATE5 = "UPDATE Devices Set DStatus=@0 WHERE id=5";
    var SQLUPDATE6 = "UPDATE Sensor Set Status=@0 WHERE id1=1";
    var SQLUPDATE7 = "UPDATE Sensor Set Status=@0 WHERE id1=2";
    var SQLUPDATE8 = "UPDATE Sensor Set Status=@0 WHERE id1=3";
    var SQLUPDATE9 = "UPDATE Sensor Set Status=@0 WHERE id1=4";
    var sqlQ1 = "SELECT * FROM Devices WHERE id=1";
    var sqlQ2 = "SELECT * FROM Devices WHERE id=2";
    var sqlQ3 = "SELECT * FROM Devices WHERE id=3";
    var sqlQ4 = "SELECT * FROM Devices WHERE id=4";
    var sqlQ5 = "SELECT * FROM Devices WHERE id=5";
    var sqlQ6 = "SELECT * FROM Sensor WHERE id1=1";
    var sqlQ7 = "SELECT * FROM Sensor WHERE id1=2";
    var sqlQ8 = "SELECT * FROM Sensor WHERE id1=3";
    var sqlQ9 = "SELECT * FROM Sensor WHERE id1=4";
    var SQLINSERTD = "INSERT INTO DLog (Name,Status,ATime) VALUES (@0,@1,@2)";
    var SQLINSERTS = "INSERT INTO SLog (Name,Status,Time) VALUES (@0,@1,@2)";

    string vnTimeZoneKey = "SE Asia Standard Time";
    TimeZoneInfo vnTimeZone = TimeZoneInfo.FindSystemTimeZoneById(vnTimeZoneKey);
    ;
    DateTime timelog =
    TimeZoneInfo.ConvertTimeFromUtc(DateTime.UtcNow,vnTimeZone);

    var data1 = db.QuerySingle(sqlQ1);
    var data2 = db.QuerySingle(sqlQ2);
    var data3 = db.QuerySingle(sqlQ3);
    var data4 = db.QuerySingle(sqlQ4);
    var data5 = db.QuerySingle(sqlQ5);
    var data6 = db.QuerySingle(sqlQ6);
    var data7 = db.QuerySingle(sqlQ7);
    var data8 = db.QuerySingle(sqlQ8);
    var data9 = db.QuerySingle(sqlQ9);
    if(String.Compare(ustt1,data1.DStatus) != 0){
        db.Execute(SQLINSERTD,"Motor",ustt1,timelog);
        db.Execute(SQLUPDATE1, ustt1);
    }
    if(String.Compare(ustt2,data2.DStatus) != 0){
        db.Execute(SQLINSERTD,"Fan",ustt2,timelog);

```

```

        db.Execute(SQLUPDATE2, ustt2);
    }
    if(String.Compare(ustt3,data3.DStatus) != 0){
        db.Execute(SQLINSERTD,"Step",ustt3,timelog);
        db.Execute(SQLUPDATE3, ustt3);
    }
    if(String.Compare(ustt4,data4.DStatus) != 0){
        db.Execute(SQLINSERTD,"Led1",ustt4,timelog);
        db.Execute(SQLUPDATE4, ustt4);
    }
    if(String.Compare(ustt5,data5.DStatus) != 0){
        db.Execute(SQLINSERTD,"Led2",ustt5,timelog);
        db.Execute(SQLUPDATE5, ustt5);
    }
    if(String.Compare(ustt6,data6.Status) != 0){
        db.Execute(SQLINSERTS,"AnhSang",ustt6,timelog);
        db.Execute(SQLUPDATE6, ustt6);
    }
    if(String.Compare(ustt7,data7.Status) != 0){
        db.Execute(SQLINSERTS,"DoAm",ustt7,timelog);
        db.Execute(SQLUPDATE7, ustt7);
    }
    if(String.Compare(ustt8,data8.Status) != 0){
        db.Execute(SQLINSERTS,"NhietDo",ustt8,timelog);
        db.Execute(SQLUPDATE8, ustt8);
    }
    if(String.Compare(ustt9,data9.Status) != 0){
        db.Execute(SQLINSERTS,"DoAmKK",ustt9,timelog);
        db.Execute(SQLUPDATE9, ustt9);
    }
}
}

```

8.7. Mã lệnh trang *DisplayStatus.cshtml*

```

@{
    var db = Database.Open("iot");
    var sqlQ = "SELECT * FROM Devices";
    var selectQueryString = "SELECT * FROM DLog ORDER BY id";
    var data = db.Query(sqlQ);
}

<!DOCTYPE html>
<html lang="en">
    <head>
        <style>

table {
    font-size: large;

    color: #f00;
    text-align: center;
}
tr {
    font-size: large;

```

```

        color: #f00;
        text-align: center;
    }
    ul { font-size: large;
        color: #f00;
        text-align: center ;
    }
    th { font-size: large;

        color: #f00;
        text-align: center ;
    }
    h3{ font-size: larger;
        color: #0094ff;
        text-align: center;

    }
    th {
        text-align: center;
    }
</style>
</head>
<body>
    <div id="statusupdate">
        <ul>
            @foreach(var row in data){
                <li>@row.Name = @row.DStatus.</li>
            }
        </ul>
        <div id="log1">
            <br>
            <h3> Change Log</h3>
        </div>
        <table>
            <tr>
                <th>Device    </th>
                <th>Change To  </th>
                <th>Time    </th>
            </tr>
            @foreach(var row in db.Query(selectQueryString)){
                <tr>
                    <td>@row.Name</td>
                    <td>@row.Status</td>
                    <td>@row.ATime</td>
                </tr>
            }
        </table>
    </div>
</body>
</html>

```

8.8. Mã lệnh trang *DisplaySensor.cshtml*

```

@{
    var db = Database.Open("iot");
    var sqlQ1 = "SELECT * FROM Sensor WHERE id1=1";
}

```

```

var sqlQ2 = "SELECT * FROM Sensor WHERE id1=2";
var sqlQ3 = "SELECT * FROM Sensor WHERE id1=3";
var sqlQ4 = "SELECT * FROM Sensor WHERE id1=4";
var selectQueryString = "SELECT * FROM SLog ORDER BY id3";
var data1 = db.QuerySingle(sqlQ1);
var sstt1 = data1.Status;
var Name1 = data1.Name;
var data2 = db.QuerySingle(sqlQ2);
var sstt2 = data2.Status;
var Name2 = data2.Name;
var data3 = db.QuerySingle(sqlQ3);
var sstt3 = data3.Status;
var Name3 = data3.Name;
var data4 = db.QuerySingle(sqlQ4);
var sstt4 = data4.Status;
var Name4 = data4.Name;
var Display = "";

if(sstt1 == "0"){
    Display = "Không Có Ánh Sáng. ";
}
if(sstt1 != "0"){
    Display = "Có Ánh Sáng. ";
}
Display = Display + "Độ Ẩm Hiện Tại: "+sstt2+ ". Nhiệt Độ Hiện Tại: "
+sstt3+ "°C."+"Độ Ẩm Không Khí hiện tại: "+sstt4+"%.";
}

```

```

<!DOCTYPE html>
<html lang="en">
    <head>
    <style>

table { font-size: large;
        background-color: #0ff;
        color: #f00;
        text-align: center;
        }
h3{ font-size: larger;
    color: #0094ff;
    text-align: left

}
th {
    text-align: center;
}

    </style>

    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    <div id="log">
        <div id="log1">
            @Display

```



```
        </br></br>
    </div>
</div>
</body>
</html>
```