

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

-----o0o-----



**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**TIÊU ĐỀ LUẬN VĂN**

**GVHD: Lưu Phú**

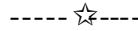
**SVTH: Lê Nhựt Hạ**

**MSSV: 1610936**

**TP. HỒ CHÍ MINH, THÁNG 7 NĂM 2020**

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  
TRƯỜNG ĐẠI HỌC BÁCH KHOA

Độc lập – Tự do – Hạnh phúc.



Số: \_\_\_\_\_ /BKĐT  
Khoa: Điện – Điện tử  
Bộ Môn: Điện Tử

## NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

1. HỌ VÀ TÊN : **LÊ NHỰT HẠ** MSSV: **1610936**  
 2. NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG** LỚP : **DD16DV2**  
 3. Đề tài: **Hệ thống cảnh báo giả định**  
 4. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):  
   - Thiết kế, lập trình và thực hiện các mạch điện tử: Mạch home server, Mạch camera chuông cửa, Mạch điều khiển các thiết bị bằng hồng ngoại, Mạch điều khiển relay bằng wifi  
   - Thiết kế mô hình mạng kết nối các thiết bị và streaming dữ liệu hình ảnh âm thanh  
   - Lập trình giao diện web điều khiển các thiết bị  
   - Lập trình một số ứng dụng xử lý ảnh  
 5. Ngày giao nhiệm vụ luận văn: .15/03/2020.....  
 6. Ngày hoàn thành nhiệm vụ: .... 15/07/2020.....  
 7. Họ và tên người hướng dẫn: Phàn hướng dẫn  
        ..... **LUU PHU** ..... Toàn bộ.....  
 Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.  
*Tp.HCM, ngày..... tháng..... năm 20*

**CHỦ NHIỆM BỘ MÔN**

**NGƯỜI HƯỚNG DẪN CHÍNH**

### PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):.....

Đơn vị:.....

Ngày bảo vệ : .....

Điểm tổng kết: .....

Nơi lưu trữ luận văn: .....

Danh sách hội đồng

## LỜI CẢM ƠN

Trải qua 4 năm học ở Đại học Bách Khoa Hồ Chí Minh là một trong những khoảng thời gian thử thách nhất em từng trải qua. Với những kiến thức kĩ thuật rộng lớn vào ngày càng phát triển để một sinh viên như em có thể nắm vững quả thực không dễ dàng. Việc trải qua nhiều môn học khác nhau để được thực hiện quyền Luận Văn Tốt Nghiệp là cả một quá trình dài. Chỉ riêng đề tài Luận Văn này thôi cũng đã tồn tại nhiều tháng chuẩn bị kiến thức và 4 tháng ráo riết thực hiện. Mặc dù thành quả còn nhiều hạn chế tuy nhiên với quỹ thời gian có giới hạn cùng lượng lớn kiến thức và công việc cần xử lý có thể xem đề tài là một thành công của riêng em trong nỗ lực tìm hiểu và áp dụng kiến thức học được. Trải qua tất cả các khâu từ tìm hiểu kiến thức, lập trình, dựng mô hình, mô phỏng, thiết kế mạch, gia công mạch đến sửa chữa lỗi và hoàn thiện tất cả đều đem đến những kinh nghiệm quý báu mà chỉ khi bắt tay thực hiện mới nhận thấy được. Thành quả của đề tài không chỉ là sản phẩm đã làm ra mà là vốn kiến thức học hỏi được trong suốt thời gian thực hiện đề tài.

Và trong suốt quá trình đó rất biết ơn trường đã tạo điều kiện thuận lợi cũng sự hướng dẫn tận tình của các thầy cô. Đồng thời cũng rất tự hào với tinh thần là Sinh viên Bách Khoa.

Xin gửi lời cảm ơn đến Trường Đại học Bách Khoa Hồ Chí Minh, các thầy, cô, cán bộ của Khoa Điện – Điện tử cùng các bạn bè đã hết lòng hỗ trợ trong suốt quá trình học tập ở trường.

Sau cùng xin gửi lời cảm ơn đặc biệt đến Thầy Lưu Phú đã tận tình giúp đỡ, giải đáp thắc mắc cùng những kinh nghiệm quý báu mà thầy truyền lại trong suốt quá trình thực hiện luận văn.

Em xin trân trọng cảm ơn!

Tp. Hồ Chí Minh, ngày tháng năm .

**Sinh viên**

Lê Nhựt Hợp

## **TÓM TẮT LUẬN VĂN**

Luận văn trình bày các nội dung chính sau đây:

- Lý thuyết về công nghệ Streaming hình ảnh âm thanh bằng WebRTC
- Mô hình thiết kế kết nối hệ thống
- Kết hợp với các thuật toán xử lý ảnh
- Thiết kế phần cứng và phần mềm các thiết bị sử dụng trong hệ thống:
  - + Thiết bị HomeServer
  - + Thiết bị Camera Chuông cửa
  - + Thiết bị điều khiển tín hiệu hồng ngoại
  - + Thiết bị điều khiển thiết bị dân dụng bằng Relay

**Mục lục**

Chương 1 GIỚI THIỆU ĐỀ TÀI .....	4
1.1 Giới thiệu tổng quan.....	7
1.2 Ý tưởng thiết kế đề tài.....	7
1.3 Giới thiệu các hệ thống gia dụng có trên thị trường.....	7
1.3.1 Smart Home Tuya.....	7
1.3.2 BKAV SmartHome.....	7
1.3.3 LUMI.....	9
1.4 Giới thiệu hệ thống.....	10
1.4.1 Home server.....	11
1.4.2 Camera chuông cửa .....	12
1.4.3 Điều khiển thiết bị hồng ngoại .....	13
1.4.4 Điều khiển Relay và LCD.....	13
Chương 2 CÁC VĂN ĐỀ LÝ THUYẾT LIÊN QUAN .....	15
2.1 Phần cứng .....	15
2.1.1 Module raspberry pi .....	15
2.1.2 Module esp8266 .....	17
2.1.3 Vi điều khiển PIC16F877A .....	18
2.1.4 Loa .....	19
2.1.5 Micro .....	21
2.1.6 Hiển thị .....	23
2.1.7 Nút nhấn .....	24
2.1.8 LED chiếu sáng .....	26
2.1.9 LED phát hồng ngoại .....	28

2.1.10 LED thu hồng ngoại .....	29
2.1.11 Giao tiếp UART .....	29
2.1.12 Giao tiếp I2C .....	30
2.1.13 Giao tiếp GPIO.....	30
2.2 Phân mềm .....	31
2.2.1 Lý thuyết về các giao thức mạng được sử dụng .....	31
2.2.2 Các ngôn ngữ lập trình, thư viện và framework được sử dụng.....	34
2.2.3 Kỹ thuật truyền streaming đa phương tiện .....	37
2.2.4 Các giải thuật xử lý ảnh .....	49
2.2.5 Google Cloud Platform .....	51
Chương 3 THIẾT KẾ .....	52
3.1 THIẾT KẾ PHẦN CỨNG .....	52
3.1.1 Home server.....	52
3.1.2 Camera chuông cửa .....	59
3.1.3 Điều khiển thiết bị hồng ngoại .....	60
3.1.4 Điều khiển Relay và LCD .....	67
3.2 THIẾT KẾ PHẦN MỀM .....	75
3.2.1 Giải thuật phần mềm thiết bị Home server.....	75
3.2.2 Giải thuật phần mềm Camera chuông cửa.....	80
3.2.3 Giải thuật phần mềm thiết bị điều khiển hồng ngoại .....	81
3.2.4 Giải thuật phần mềm thiết bị điều khiển relay .....	82
3.2.5 Giải thuật phần mềm của Cloud Server: .....	83
3.3 GIAO THÚC KẾT NỐI VÀ GIAO DIỆN NGƯỜI DÙNG.....	85
3.3.1 Sơ đồ khối .....	85

3.3.2	Kết nối giữa các thiết bị .....	85
3.3.3	Kết nối đến cloud .....	87
3.3.4	Thiết kế hệ thống Streaming trên nền tảng WebRTC .....	87
3.3.5	Giới thiệu cơ bản một quá trình cấu hình chuẩn bị cho việc streaming với FFmpeg, Janus Gateway và webRTC api .....	89
3.3.6	Giao diện người dùng .....	92
Chương 4	THI CÔNG VÀ MỞ RỘNG .....	93
4.1	Thi công thực tế .....	93
4.1.1	Các vấn đề khi thi công .....	93
4.1.2	Kết quả thi công và Layout .....	94
4.2	Ý tưởng mở rộng đề tài trên nền tảng thiết kế .....	97
Chương 5	TÀI LIỆU THAM KHẢO .....	99
Chương 6	Phụ lục .....	101
6.1	Sơ đồ nguyên lý mạch HomeServer và Camera chuông cửa .....	101
6.2	Sơ đồ nguyên lý mạch điều khiển hồng ngoại .....	102
6.3	Sơ đồ nguyên lý mạch điều khiển thiết bị bằng Relay .....	103
6.4	Sourcecode WebServer .....	104
6.5	Sourcecode Mạch Camera và mạch HomeServer .....	129
6.6	File cấu hình Janus .....	172
6.7	File cấu hình Coturn Server .....	181
6.8	SourceCode Mạch Điều khiển thiết bị hồng ngoại .....	181

## DANH SÁCH HÌNH MINH HỌA

Hình 1-1 Bộ điều khiển trung tâm của LUMI.....	9
Hình 1-2 Công tắc 3 nút của LUMI.....	9
Hình 1-3 Bộ điều khiển hồng ngoại của LUMI .....	10
Hình 1-4 Mô tả sơ lược kết nối các thiết bị .....	10
Hình 2-1 Ngoại vi module ESP8266 12E.....	18
Hình 2-2 Sơ đồ chân PIC16F877A.....	18
Hình 2-3 Ngoại vi của PIC16F877A.....	19
Hình 2-4 Loa 4Ohm 3W .....	21
Hình 2-5 Microphone USB .....	23
Hình 2-6 Màn hình LCD 1602 .....	24
Hình 2-7 Nút nhấn 6mm .....	24
Hình 2-8 Nút nhấn 12mm .....	24
Hình 2-9 Nút nhấn dán .....	25
Hình 2-10 Nút nhấn PLC .....	25
Hình 2-11 Led RGB 5050.....	27
Hình 2-12 Giới thiệu bộ thư viện Ffmpeg .....	37
Hình 2-13 Mô hình P2P đơn giản nhất .....	42
Hình 2-14 Mô hình bị chặn bởi NAT .....	42
Hình 2-15 Mô hình TURN Server và TURN Serve .....	43
Hình 2-16 Giới thiệu về công nghệ WebRTC.....	44
Hình 2-17 Giới thiệu về bộ thư viện JanusGateway .....	47
Hình 2-18 Minh họa mô hình của Janus Gateway.....	49
Hình 3-1 Sơ đồ mạch Home server .....	52

Hình 3-2 Sơ đồ nguyên lý mạch HomeServer .....	53
Hình 3-3 Sơ đồ nguyên lý khói đèn LED .....	54
Hình 3-4 Bảng gốc Datasheet LED 5050 .....	55
Hình 3-5 Sơ đồ nguyên lý mạch khuếch đại âm thanh .....	56
Hình 3-6 Sơ đồ chân PAM8403.....	57
Hình 3-7 Sơ đồ nguyên lý khói nút nhấn mạch HomeServer .....	58
Hình 3-8 Sơ đồ nguyên lý khói nút nhấn cảm ứng.....	58
Hình 3-9 Sơ đồ khói thiết bị camera chuông cửa .....	59
Hình 3-10 Module cảm biến PIR .....	60
Hình 3-11 Sơ đồ khói thiết bị điều khiển hồng ngoại .....	60
Hình 3-12 Khối LED phát hồng ngoại .....	61
Hình 3-13 Sơ đồ khói sử lý trung tâm và nút nhấn mạch điều khiển hồng ngoại ...	64
Hình 3-14 LED thu hồng ngoại TSOP1838 .....	65
Hình 3-15 Adapter 5V.....	65
Hình 3-16 Sơ đồ nguyên lý mạch ổn áp dùng AMS1117.....	66
Hình 3-17 Sơ đồ mạch điều khiển Relay và LCD .....	67
Hình 3-18 Biến áp 12V .....	69
Hình 3-19 Chỉnh lưu cầu.....	69
Hình 3-20 Sơ đồ chân LM2596 .....	70
Hình 3-21 Khối Opto và Relay .....	73
Hình 3-22 Khối LCD.....	74
Hình 3-23 Khối bàn phím .....	74
Hình 3-24 Mô tả kết nối các khối phần mềm .....	75
Hình 3-25 Máy trạng thái thiết bị Homeserver .....	76

Hình 3-26 Sơ đồ khối âm thanh.....	77
Hình 3-27 Sơ đồ khói đèn LED .....	78
Hình 3-28 Khối nhận gói tin.....	78
Hình 3-29 Khối gửi gói tin .....	79
Hình 3-30 Lưu đồ khói truy xuất File .....	79
Hình 3-31 Mô tả quá trình hoạt động của WebServer .....	80
Hình 3-32 Lưu đồ giải thuật khói phát hồng ngoại .....	81
Hình 3-33 Lưu đồ khói thu hồng ngoại.....	82
Hình 3-34 Mô tả hoạt động của Cloud Server .....	84
Hình 3-35 Giao diện nhập mật khẩu wifi .....	86
Hình 3-36 Giao diện cấu hình wifi cho các thiết bị.....	86
Hình 3-37 Mô tả quá trình kết nối giữa trình duyệt Web và Server.....	92
Hình 4-1 Layout mạch Camera chuông cửa và Home Server.....	94
Hình 4-2 Kết quả thi công mạch HomeServer và chuông cửa.....	94
Hình 4-3 Layout mạch điều khiển thiết bị hồng ngoại .....	96
Hình 4-4 Kết quả thi công thiết bị điều khiển hồng ngoại .....	96
Hình 4-5 Layout mạch điều khiển thiết bị bằng Relay .....	97
Hình 4-6 Kết quả thi công mạch điều khiển thiết bị bằng Relay.....	97
Hình 6-1 Sơ đồ nguyên lý mạch Home Server và camera chuông cửa .....	101
Hình 6-2 Sơ đồ nguyên lý mạch điều khiển hồng ngoại.....	102
Hình 6-3 Sơ đồ nguyên lý mạch điều khiển thiết bị bằng hồng ngoại.....	103

## Chương 1 GIỚI THIỆU ĐỀ TÀI

### 1.1 Giới thiệu tổng quan

### 1.2 Ý tưởng thiết kế đề tài

Hiện nay với sự bùng nổ của các thiết bị thông minh và hệ thống internet được bao phủ khắp mọi nơi kéo theo sự phát triển của các thiết bị ứng dụng thành quả của các nghiên cứu về Internet vạn vật, trong đó có hệ thống cảnh báo gia dụng thông minh

Đề tài hướng đến một giải pháp thiết kế một hệ thống cảnh báo gia dụng thông minh đa chức năng, vận hành ổn định và giá thành phù hợp, dễ lắp đặt như một thiết bị gia dụng thông thường

### 1.3 Giới thiệu các hệ thống gia dụng có trên thị trường

Hiện tại các hệ thống trên thị trường đang có nhiều nhà cung cấp với nhiều chủng loại đa dạng mẫu mã khác nhau. Sau đây là một số ví dụ tiêu biểu:

#### 1.3.1 Smart Home Tuya

##### Giới thiệu<sup>[1]</sup>

Tuya Smart là nền tảng IoT toàn cầu cho phép các sản phẩm điện trở lên thông minh hơn cho người tiêu dùng. Nó cung cấp giải pháp giúp các sản phẩm thông minh có thể làm việc với nhau trên cùng 1 APP chung. Dễ dàng điều khiển các thiết bị từ xa qua smartphone ở bất cứ đâu có mạng Internet.

##### Bộ sản phẩm tương tự<sup>[1]</sup>

- Bộ Báo Trộm Không Dây Wifi Kèm Camera SmartHomePlus SHP-CK2 Plus giá 2,350,000đ. Bộ sản phẩm bao gồm 1 camera IP, 1 loa báo động wifi, 1 cảm biến chuyển động PIR, 1 cảm biến cửa từ, 1 remote điều khiển.
- Công Tắc Wifi Và RF Công Suất Lớn 40A Tuya SHP-SW3 giá 750,000đ

#### 1.3.2 BKAV SmartHome

##### Giới thiệu<sup>[2]</sup>

Bkav SmartHome Security gồm thiết bị an ninh trung tâm, hàng rào điện tử, các cảm biến, hệ thống camera ghi hình... dựng nên hàng rào nhiều lớp, giám sát ngôi nhà theo thời gian thực. Khi phát hiện chủ nhân quên đóng cửa nhà hoặc nguy cơ cháy nổ, xâm nhập trái phép, hệ thống sẽ phát đi cảnh báo theo các cấp độ an ninh khác nhau như: bật đèn tại khu vực có đột nhập, hú còi báo động, gửi tin nhắn, gọi điện tới gia chủ... Đây là giải pháp an ninh đầu tiên trên thị trường tích hợp công nghệ AI, cho phép phân biệt người trong/ngoài nhà và có khả năng tự chuyển chế độ an ninh theo ngữ cảnh.

Các thiết bị an ninh giá rẻ trên thị trường hiện có rất nhiều nhưng cũng tiềm ẩn nguy cơ mang đến rắc rối cho gia chủ như phát hiện, cảnh báo nhầm, gây bất tiện khi nhà có khách, có người lớn tuổi sinh hoạt không theo giờ giấc thông thường... Bộ thiết bị Bkav SmartHome Security hoạt động "tự nhiên", tin cậy, xác định chính xác khi nào cần kích hoạt hệ thống an ninh. Giao diện giám sát ngôi nhà hiển thị trực quan, sinh động, cho phép xem hình ảnh trực tiếp từng khu vực có xâm nhập.

Với các thiết bị nhỏ gọn, kết nối bằng công nghệ truyền thông không dây ZigBee / WiFi, việc lắp đặt hệ thống an ninh Bkav SmartHome Security thuận tiện, dễ dàng triển khai ngay trên hạ tầng sẵn có.

### **Bộ sản phẩm tương tự [2]**

- **Thiết bị xử lý trung tâm.** Điều khiển và kiểm soát toàn bộ các thiết bị trong hệ thống an ninh Bkav SmartHome Security. Có khả năng quản lý và điều khiển tới 500 thiết bị trong nhà. Nhận và phân tích các tín hiệu cảnh báo được gửi về từ các cảm biến an ninh khu vực để đưa ra các kịch bản an ninh phù hợp.
- **Thiết bị phân vùng kiểm soát an ninh.** Phân vùng kiểm soát, trực tiếp các cảm biến an ninh trong nhà như Hàng rào điện tử, cảm biến vị trí, cảm biến khói, cảm biến mở cửa... Tín hiệu sau khi được thu thập sẽ được chuyển về thiết bị xử lý trung tâm để xử lý.
- **Hệ thống camera giám sát.** Hệ thống camera tích hợp hỗ trợ giám sát an ninh tự động qua hình ảnh. Qua camera hệ thống cho xem xem trực tiếp hình ảnh an ninh qua cảnh báo trên giao diện phần mềm.

### 1.3.3 LUMI

#### Bộ điều khiển trung tâm



Hình 1-1 Bộ điều khiển trung tâm của LUMI

Mô tả: Là bộ não của ngôi nhà thông minh, tích hợp công nghệ truyền thông không dây Zigbee cho phép: Kết nối và quản lý các thiết bị điện, lưu trữ các thông tin cấu hình cài đặt của người dùng, cập nhật trạng thái các thiết bị cho người sử dụng.

Điện áp: 5Vdc - 1A

Nhiệt độ hoạt động: 75°C max

Giá bán: 2.530.000 đ

#### Công tắc cảm ứng 3 nút



Hình 1-2 Công tắc 3 nút của LUMI

Mô tả: là loại công tắc cảm ứng thông minh sử dụng công nghệ cảm ứng điện dung, công nghệ không dây Zigbee, mặt kính cường lực, chống xước, chống va đập, kết hợp với vòng tròn tỏa sáng LED bao quanh tạo nên sự tinh tế, sang trọng, đẳng cấp.

Điện áp: 150 - 250 VAC

Nhiệt độ hoạt động: 0 - 40°C max

Công suất: 700w(Đèn sợi đốt)-150w(Đèn led)/1 nút

Kích thước: Hình vuông (95 x 95 mm) - Hình chữ nhật (121,5 x 80 mm)

Giá bán: 1.760.000 đ

### Bộ điều khiển hồng ngoại



**Hình 1-3 Bộ điều khiển hồng ngoại của LUMI**

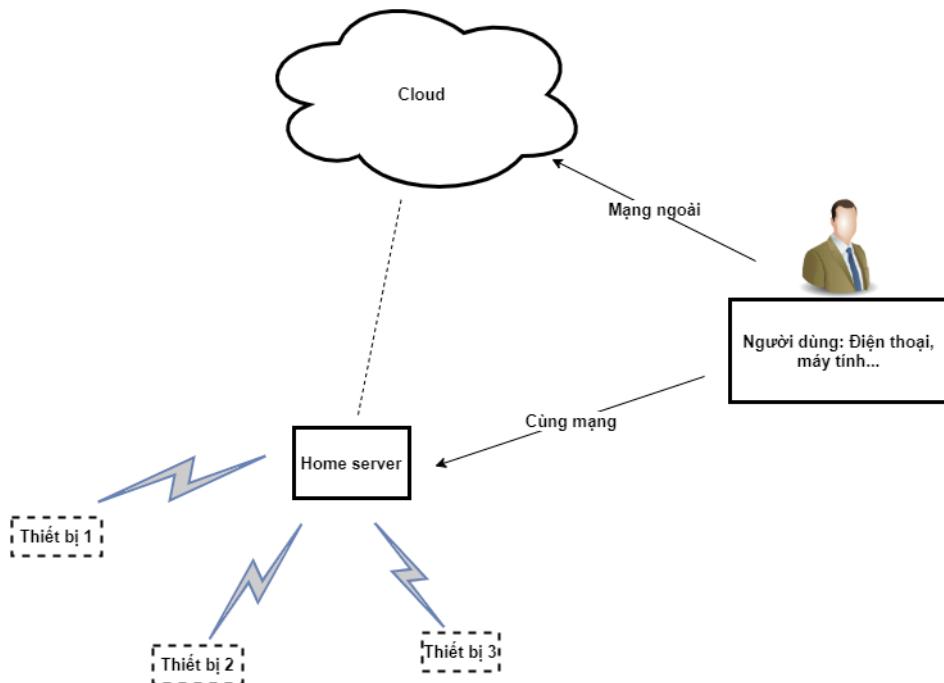
Mô tả: Học lệnh của điều khiển remote, có thể lưu tối 1000 lệnh. Tích hợp công nghệ không dây Zigbee tiên tiến.

Điện áp: 5VDC

Nhiệt độ hoạt động: 50°C Max

Giá bán: 1.650.000 đ

### 1.4 Giới thiệu hệ thống



**Hình 1-4 Mô tả sơ lược kết nối các thiết bị**

### 1.4.1 Home server

#### 1.4.1.1 Giới thiệu

Thiết bị hoạt động trong nhà. Giúp dễ dàng quản lý kết nối các thiết bị mà không cần quá nhiều cấu hình, đồng thời hỗ trợ kết nối đến cloud Server cho phép truy cập hệ thống qua kết nối Internet. Thiết bị còn hoạt động như một TV box cho phép kết nối đến màn hình và phát trực tiếp camera.

#### 1.4.1.2 Chức năng

- Quản lý các thiết bị chức năng với tốc độ đáp ứng cao
- Quản lý lớp giao diện giao tiếp người dùng với thiết bị, tương thích với nhiều loại thiết bị thông minh khác nhau
- Cầu nối quản lý các thiết bị với hệ thống điện toán đám mây, chủ động đóng ngắt kết nối theo yêu cầu. Đảm bảo tối ta tính riêng tư.
- Khả năng kết nối với màn hình hiển thị trực tiếp camera

#### 1.4.1.3 Chỉ tiêu thiết kế và yêu cầu kỹ thuật

- Điện áp hoạt động: Nguồn một chiều 5V - 2.5A. Cấp nguồn qua cổng micro USB
- Nút nhấn giao tiếp với thiết bị
- Wifi: 2.4Ghz IEEE 802.11b/g/n.
- Điều kiện hoạt động: trong nhà
- Tương tác với người dùng thông qua trình duyệt web có thể kết nối bằng các thiết bị có thể chạy được trình duyệt (máy tính, điện thoại thông minh,...)
- Tương tác được với các thiết bị camera an ninh có trên thị trường hỗ trợ giao thức RTSP
- Âm lượng loa điều chỉnh được. Công suất loa 3W. Độ nhạy: 100 dB/W
- Tích hợp đèn LED thông báo
- Tốc độ đáp ứng nhanh. Độ trễ <500ms

- Độ trễ khi truyền media thấp
- Kết nối với màn hình theo chuẩn HDMI

### **1.4.2 Camera chuông cửa**

#### **1.4.2.1 Giới thiệu**

Thiết bị camera an ninh kết hợp với nút nhấn cảm ứng dựa trên ý tưởng một camera tích hợp với chuông cửa. Mang đến khả năng quan sát như một camera an ninh kết hợp với hệ thống intercom để giao tiếp đến người đến nhà mà không cần phải bước ra cửa cũng như hệ thống nhận diện, quản lý an ninh cửa ra vào.

#### **1.4.2.2 Özellik**

- Camera an ninh
- Tích hợp nút nhấn chuông thông báo
- Cảm biến tiệm cận tự động phát hiện người đến
- Tích hợp hệ thống giao tiếp trong ngoài thông qua smartphone, máy tính

#### **1.4.2.3 Chỉ tiêu thiết kế và yêu cầu kỹ thuật**

- Điện áp hoạt động: Nguồn một chiều 5V - 2.5A. Cấp nguồn qua cổng micro USB
- Nút nhấn giao tiếp với thiết bị
- Wifi: 2.4Ghz IEEE 802.11b/g/n.
- Điều kiện hoạt động: trong nhà
- Tương tác với người dùng thông qua trình duyệt web có thể kết nối bằng các thiết bị có thể chạy được trình duyệt (máy tính, điện thoại thông minh,...)
- Âm lượng loa điều chỉnh được. Công suất loa 3W. Độ nhạy: 100 dB/W
- Tích hợp đèn LED thông báo
- Tốc độ đáp ứng nhanh. Độ trễ <500ms
- Độ trễ khi truyền media thấp

### 1.4.3 Điều khiển thiết bị hồng ngoại

#### 1.4.3.1 Giới thiệu

Thiết bị mở rộng tín năng điều khiển thiết bị hồng ngoại thông qua kết nối wifi. Hỗ trợ các thiết bị sử dụng hồng ngoại có trên thị trường: TV, máy lạnh, máy quạt,...

#### 1.4.3.2 Chức năng

- Kết nối vào mạng wifi và nhận tín hiệu điều khiển
- Điều khiển các thiết bị hồng ngoại

#### 1.4.3.3 Chỉ tiêu thiết kế và Yêu cầu kỹ thuật

- Điện áp hoạt động: Nguồn một chiều 5V - 2.5A. Cấp nguồn qua cổng micro USB
- Tầm hoạt động : Điều khiển các thiết bị trong nhà. Tầm hoạt động trong phạm vi 10m
- Tần số hồng ngoại: 38kHz
- Điều kiện hoạt động: Trong nhà
- Tương tác với người dùng thông qua trình duyệt web có thể kết nối bằng các thiết bị có thể chạy được trình duyệt (máy tính, điện thoại thông minh,...)
- Tốc độ đáp ứng nhanh. Độ trễ <500ms

### 1.4.4 Điều khiển Relay và LCD

#### 1.4.4.1 Giới thiệu

Thiết bị mở rộng tín năng đóng ngắt các thiết bị sử dụng điện dân dụng

Tích hợp tính năng điều khiển từ xa

#### 1.4.4.2 Chức năng

- Kết nối vào mạng wifi và nhận tín hiệu điều khiển từ xa qua smartphone, máy tính
- Điều khiển tắt/mở nguồn cho các thiết bị trong lưới điện. Nguồn điện hoạt động 220V xoay chiều, tần số 50Hz.

#### **1.4.4.3 Chỉ tiêu thiết kế Yêu cầu kỹ thuật**

- Điện áp hoạt động: nguồn điện xoay chiều 220V/50Hz cấp bằng lưới điện dân dụng
- Điều kiện hoạt động trong nhà
- Tương tác với người dùng thông qua trình duyệt web có thể kết nối bằng các thiết bị có thể chạy được trình duyệt (máy tính, điện thoại thông minh,...)
- Tốc độ đáp ứng nhanh. Độ trễ <500ms

## Chương 2 CÁC VẤN ĐỀ LÝ THUYẾT LIÊN QUAN

### 2.1 Phần cứng

#### 2.1.1 Module raspberry pi

##### 2.1.1.1 Giới thiệu

Raspberry Pi là chiếc máy tính kích thước nhỏ được tích hợp nhiều phần cứng mạnh mẽ đủ khả năng chạy hệ điều hành và cài đặt được nhiều ứng dụng trên nó. Với giá chỉ vài chục USD, Raspberry hiện đang là mini computer nổi bật nhất hiện nay. Ban đầu, tổ chức Raspberry Pi Foundation phát triển dự án Raspberry với mục tiêu chính là giảng dạy máy tính cho trẻ em và tạo ra một công cụ giá rẻ (chỉ vài chục USD) để sinh viên nghiên cứu học tập. Tuy nhiên, sau khi xuất hiện, Raspberry Pi được cộng đồng đánh giá cao về tính ứng dụng với phần cứng được hỗ trợ tốt, Pi đã nhanh chóng phát triển một cách rộng rãi. Pi phù hợp cho những ứng dụng cần khả năng xử lý mạnh mẽ, đa nhiệm hoặc giải trí và đặc biệt cần chi phí thấp. Hiện nay đã có hàng ngàn ứng dụng đa dạng được cài đặt trên Rasberry Pi.

Phiên bản Raspberry Pi đầu tiên được phát hành tháng 2 năm 2012, và tới nay đã có nhiều phiên bản khác nhau, với sự nâng cấp của phần cứng, cũng như hướng tới những mục tiêu khác nhau.

Thông số kỹ thuật module Raspberry Pi 2:

- Vi xử lý: Broadcom BCM2837B0, quad-core A53 (ARMv8) 64-bit SoC @1.4GHz
- RAM: 1GB LPDDR2 SDRAM
- Kết nối: Bluetooth 4.2, BLE, Gigabit Ethernet over USB 2.0 (Tối đa 300Mbps).
- Cổng USB: 4 x 2.0
- Mở rộng: 40-pin GPIO
- Video và âm thanh: 1 cổng full-sized HDMI, Cổng MIPI DSI Display, cổng MIPI CSI Camera, cổng stereo output và composite video 4 chân.

- Multimedia: H.264, MPEG-4 decode (1080p30), H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
- Lưu trữ: MicroSD
- Nguồn điện sử dụng: 5V/2.5A DC cổng microUSB, 5V DC trên chân GPIO, Power over Ethernet (PoE) (yêu cầu thêm PoE HAT).

Thông số kỹ thuật module Raspberry Pi 3

- Vi xử lý: Broadcom BCM2837B0, quad-core A53 (ARMv8) 64-bit SoC @1.4GHz
- RAM: 1GB LPDDR2 SDRAM
- Kết nối: 2.4GHz and 5GHz IEEE 802.11 b/g/n/ac wireless LAN, Bluetooth 4.2, BLE, Gigabit Ethernet over USB 2.0 (Tối đa 300Mbps).
- Cổng USB: 4 x 2.0
- Mở rộng: 40-pin GPIO
- Video và âm thanh: 1 cổng full-sized HDMI, Cổng MIPI DSI Display, cổng MIPI CSI Camera, cổng stereo output và composite video 4 chân.
- Multimedia: H.264, MPEG-4 decode (1080p30), H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
- Lưu trữ: MicroSD
- Nguồn điện sử dụng: 5V/2.5A DC cổng microUSB, 5V DC trên chân GPIO, Power over Ethernet (PoE) (yêu cầu thêm PoE HAT).

### **2.1.1.2 Mục đích sử dụng**

- Với việc streaming đa phương tiện và ứng dụng điều khiển thiết bị. Cần hệ thống máy tính vừa phải và hỗ trợ các ngoại vi phổ biến.
- Raspberry Pi với hệ điều hành Linux, hỗ trợ các giao tiếp phổ biến, cùng với nguồn cung cấp rộng rãi dễ thay thế là lựa chọn phù hợp để thực nghiệm trước khi ứng dụng rộng rãi.

- Cụ thể: Thiết bị Camera chuông cửa sử dụng mạch Raspberry Pi 2 và thiết bị HomeServer sử dụng mạch Raspberry Pi 3
- Module chạy hệ điều hành Linux và hỗ trợ kết nối mạng nên có thẻ dễ dàng lập trình thông qua cổng SSH port 22 và chuyển file hỗ trợ SFTP.

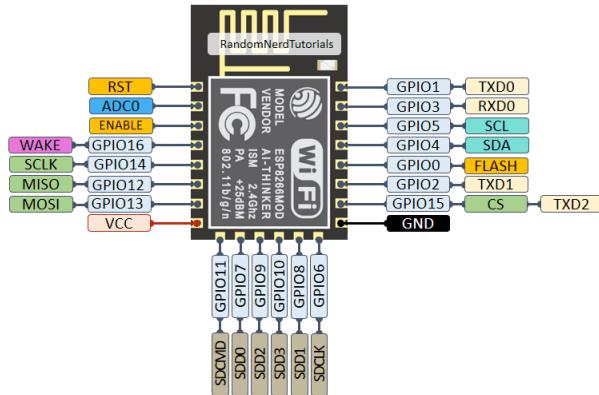
### 2.1.2 Module esp8266

#### 2.1.2.1 Giới thiệu

Chip ESP8266 được phát triển bởi Espressif để cung cấp giải pháp giao tiếp Wifi cho các thiết bị IoT. Điểm đặc biệt của dòng ESP8266 là nó được tích hợp các mạch RF như balun, antenna switches, TX power amplifier và RX filter ngay bên trong chip với kích thước rất nhỏ chỉ 5x5mm nên các board sử dụng ESP8266 không cần kích thước board lớn cũng như không cần nhiều linh kiện xung quanh. Ngoài ra, giá thành của ESP8266 cũng rất thấp đủ để hấp dẫn các nhà phát triển sản phẩm IoT

Ngoài vi:

- WiFi: 2.4 GHz hỗ trợ chuẩn 802.11 b/g/n
- Điện áp hoạt động: 3.3V
- Số chân I/O: 11 (tất cả các chân I/O đều có Interrupt/PWM/I2C/One-wire, trừ chân D0)
- Số chân Analog Input: 1 (điện áp vào tối đa 3.3V)
- Bộ nhớ Flash: 4MB
- Giao tiếp: Cable Micro USB (tương đương cáp sạc điện thoại)
- Hỗ trợ bảo mật: WPA/WPA2
- Tích hợp giao thức TCP/IP
- Lập trình trên các ngôn ngữ: C/C++, Micropython,...



Hình 2-1 Ngoại vi module ESP8266 12E

### 2.1.2.2 Mục đích sử dụng

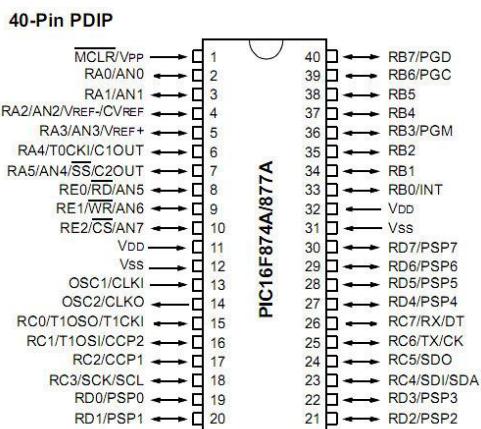
ESP8266 là module kết nối vào mạng và nhận tín hiệu. Với giá thành rẻ, nguồn cung cấp linh kiện lớn, phổ biến và lượng ngoại vi vừa đủ module là lựa chọn phù hợp.

Module ESP8266 được sử dụng trong đề tài là module ESP8266 12E với ngôn ngữ lập trình C.

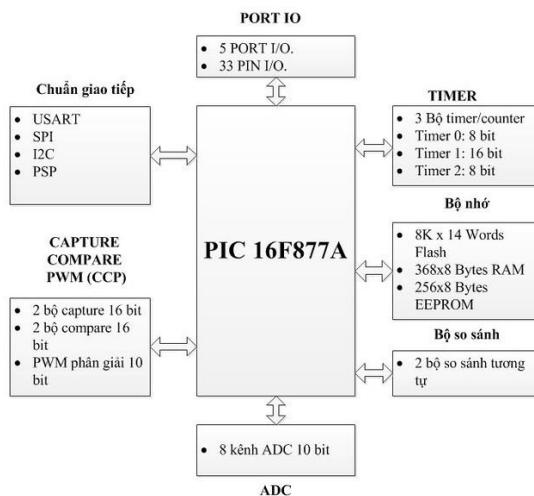
### 2.1.3 Vi điều khiển PIC16F877A

- PIC16F877A là loại vi điều khiển 8bit tầm trung của hãng Microchip
- PIC16F877A có kiến trúc Harvard, sử dụng tập lệnh kiểu RISC (Reduced Instruction Set Computer) với chỉ 35 lệnh cơ bản.
- Tất cả các lệnh được thực hiện trong một chu kỳ lệnh ngoại trừ các lệnh rẽ nhánh.

#### 2.1.3.1 Giới thiệu



Hình 2-2 Sơ đồ chân PIC16F877A



Hình 2-3 Ngoại vi của PIC16F877A

### 2.1.3.2 Mục đích sử dụng

PIC16F877A sẽ được sử dụng như một trong hai vi điều khiển trong điều khiển relay. Với yêu cầu về ngoại vi và số lượng chân GPIO cần thiết cùng độ ổn định cao. PIC16F877A là một lựa chọn hoàn toàn phù hợp.

### 2.1.4 Loa

#### Giới thiệu

Tất cả các loại loa đều hoạt động dựa trên nguyên tắc căn bản là làm không khí chuyển động theo sự điều khiển của tín hiệu điện để tạo nên các sóng âm lan truyền trong không khí, tác động tới tai người nghe và giúp chúng ta thưởng thức được âm nhạc...

Có nhiều cách thức vận dụng nguyên tắc này, nhưng chúng tôi phân loại thành 5 nhóm chính, tương ứng với năm loại loa khác nhau, đó là loa điện động, lao màng tĩnh điện, loa mành nam châm, loa kèn và một loại loa khá mới mẻ, có cấu tạo đặc biệt đó là loa plasma.

**- Loa điện động.** Loa điện động hoạt động rất linh hoạt, sử dụng tiện lợi, đặc biệt ở tần số thấp nhưng phiền toái ở chỗ nó cần phải có bộ phận phân tần và thùng phải lắp nhiều loa con. Sự cồng kềnh này không tránh khỏi việc suy hao tín hiệu. Ngoài ra, âm thanh ở gần điểm phân tần thường bị suy giảm khiến cho màn âm thanh tổng thể không mượt mà như khi chúng được tái hiện mà không có bộ phận tần. Do những nhược điểm trên, để phát ra âm thanh tốt, loa điện động cần tới một điện năng khá lớn, một phần để chuyển

động cuộn dây, phần lớn khác chỉ để làm nóng cuộn dây. Chúng cũng cần một nam châm to và một cấu trúc thùng nâng đỡ thật khoẻ, tương xứng với trọng lượng của các loa con. Tuy nhiên vì dễ chế tạo nên loại loa này rất phổ biến trong các thiết bị hi-fi.

- **Loa mành tĩnh điện.** Với ưu điểm có màng loa nhẹ và không cần dùng bộ phận phân tán, loa tĩnh điện đã ra đời và khắc phục những tồn tại trên của loa điện động. thay vì sử dụng loa con, các thiết kế loa tĩnh điện chỉ dùng một chiếc mành treo trong từ trường tĩnh điện. Chiếc mành này vừa rộng, vừa cao (khoảng 1m2), rất phẳng, mịn và cực kỳ nhẹ nên rất nhạy cảm với những dao động của tần số âm thanh. Chiếc mành này thay thế các loa con trong loa điện động, do vậy, loa tĩnh điện không cần đến bộ phận phân tán và tránh được các nhược điểm đi kèm bộ phận phân tán. Tuy nhiên, loa tĩnh điện cũng có mặt trái là chiếc mành quá mỏng và rộng bánh của nó không thể di chuyển được những khoảng cách lớn để có thể tái hiện các tần số thấp như loa điện động. Do vậy, nhiều loa mành phải dùng đến loa trầm điện động và một bộ phận phân tầng.

- **Loa mành nam châm.** Tương tự loa tĩnh điện, loa mành nam châm không hoạt động theo kiểu “điện động” như loa điện động nhưng nó có điểm khác biệt là không cần phải cắm vào ổ điện trên tường. Nếu loa tĩnh điện vận hành được nhờ có 1 chiếc mành nhẹ treo ở giữa 2 tấm kim loại tích điện thì loa mành nam châm thay thế tấm màng mỏng, và rộng này bằng 1 dải ruy băng kim loại mỏng, treo giữa 2 nam châm. Loa này hoặc điện động khi có dòng điện chạy qua ruy băng kim loại này. Khi đó, ruy băng sẽ bị các nam châm đẩy và hút. Sự chuyển động này sinh ra sóng âm trong không khí bao quanh ruy băng. Tấm ruy băng này mỏng nhẹ giúp loa mành nam châm. Phản ứng nhanh nhạy với tín hiệu tiếng và trình diễn âm thanh 1 cách trong trẻo nên tái hiện các tần số cao rất hợp. Loa mành nam châm có dải ruy băng dài khoảng vài inch, thường đóng vai trò làm loa treble và nó kết hợp rất tuyệt vời với loa trung/trầm điện động. Giống như loa tĩnh điện, loa mành nam châm thường đi liền với các loa trầm điện động. Thường giữ vị trí dành cho âm nhạc 2 kênh, tất nhiên loại loa này cũng có mặt trong các hệ thống rạp hát gia đình đa kênh. Có thể kể đến Apogee, Magnepan... là những làm loa mành nam châm có tiếng.

- **Loa plasma.** Đây là loại loa hết sức đặc biệt. Nó không cần thùng, không cần màng loa mà vẫn phát ra âm thanh rất trong trẻo, chính xác, độ méo cực thấp. Loa này

thường đóng vai trò của loa treble. Nguyên lý loa plasma rất đơn giản, nó có bộ phóng điện gắn với 1 biến thế cao áp ở 1 đầu ra của 1 ampli công suất. Ampli này điều khiển biến thế có điện áp ra lên đến hàng ngàn volt theo tín hiệu âm thanh. Kích cỡ của ngọn lửa phát ra từ que phóng điện đó cũng thay đổi theo, do vậy, áp suất không khí xung quanh ngọn lửa cũng biến đổi. Bạn sẽ nghe được những âm thanh phát ra trực tiếp trong không khí mà không hề thấy tám màng loa này rung động. Tiếng tuy hơi nhỏ nhưng bù lại rất trong trẻo, âm thanh lan toả đều trong phòng nhạc. Tuy nhiên, con đường đến với thiên đường hi-fi của loại loa này còn gập ghềnh. Trong quá trình hoạt động, loa plasma sinh ra khí ôzôn rất độc hại, có thể gây ung thư. Điện áp của loa lại quá cao, nguy hiểm cho tính mạng con người. Ngoài ra, bức xạ tàn số radio của nó giàn như không thể kiểm soát được, có thể phá hỏng quá trình thu nhận tín hiệu truyền hình của tivi, can thiệp vào hoạt động của các thiết bị khác, ví dụ: đầu CD. Hiện nay, một số nhà sản xuất loa này cũng đang cố gắng khắc phục những nhược điểm của nó...

Đề tài sử dụng loại loa mành nam châm phô biến trong các mạch điện tử. Mạch Camera cửa và mạch HomeServer sử dụng loại loa này, làm nguồn phát âm thanh cho hai mạch trên. Cụ thể loại loa được sử dụng có trở kháng 4Ohm và công suất 3W.



Hình 2-4 Loa 4Ohm 3W

### 2.1.5 Micro

Là một thiết bị hỗ trợ quá trình thu âm thanh. Nó đóng vai trò trung gian giữa nguồn âm và người nghe.

Microphone là một thiết bị biến năng, nó chuyển đổi năng lượng sóng cơ học từ một giọng nói hoặc từ các nhạc cụ thông qua việc sóng âm thanh giao động trong không khí đến màng chắn của micro, khi đó màng chắn dao động thông qua nam châm có bên trong micro và tạo ra năng lượng điện, năng lượng điện đại diện cho sóng âm thanh, sau đó được chuyển đến các thiết bị tiếp theo trong hệ thống các thiết bị âm thanh khác thông qua dây microphone hoặc thông qua một hệ thống thu phát sóng không dây.

Ngày nay micro được ứng dụng rộng rãi trong nhiều lĩnh vực. Nó là thiết bị hỗ trợ quan trọng đối với các đối tượng như diễn thuyết, diễn giả, ca sĩ hay những công việc liên quan đến tương tác xã hội...

### **Phân loại micro**

Việc phân loại micro được dựa trên nhiều tiêu chí. Nếu dựa trên nguyên lý hoạt động sẽ có rất nhiều loại micro khác nhau như: micro điện động, micro điện dung, micro áp điện, micro dải băng, micro than, micro sợi quang, micro laser, micro nước, micro mems. Trong đó hay sử dụng nhất là micro điện động, micro điện dung, micro áp điện.

#### **Micro điện động**

Có câu tạo giống loa điện động, trong đó màng của nó được làm mỏng, cuộn dây được cuốn nhiều vòng và thường có trở kháng tới 300 Ohm. Nó có độ nhạy thấp, dải tần có hạn (thường từ 50 Hz đến 16 KHz). Micro này có âm sắc ngọt và mềm thường dùng cho ca sĩ, trong các quán karaoke, hay tại gia đình.

Loại này sử dụng điện động sử dụng nam châm để thay đổi sóng điện. Đặc điểm của nó là có hoặc không có các nút on/off bật tắt trên thân micro. Micro dynamic có khả thu âm tốt tại các khoảng cách gần, không cần nguồn điện cung cấp để hoạt động và dùng tốt cho việc thu âm một người.

#### **Micro điện dung**

Loại này rất thông dụng, được sử dụng rộng rãi. Nó có độ nhạy cao với nhiều kích thước từ lớn cho đến nhỏ, dải tần âm thanh rộng (từ 20 Hz đến 20KHz). Loại này thường được dùng trong điện thoại, micro không dây.

Micro này hoạt động trên nguyên tắc dùng hiệu ứng thay đổi điện dung để thay đổi sóng điện. Nó có độ nhạy lớn hơn nhiều so với dynamic, tuy nhiên nó cần nguồn điện để hoạt động. Ưu điểm của nó là có thể thu âm từ khoảng cách xa, có thể thu âm với số người từ 2-6. Bởi thế nó thường được dùng trong giảng đường, trong một nhà hát lớn, hay các khu vực rộng lớn...

### **Micro áp điện**

Có trở kháng lớn và thường dùng trong khuếch đại âm thanh từ nhạc cụ, trống,...

Về thiết kế, cấu tạo người ta có thể chia làm : micro có dây, Micro không dây bao gồm mic cầm tay, mic cài áo hay mic cài đầu.

Đề tài sử dụng micro hỗ trợ kết nối USB thường được ứng dụng trong máy tính các loại máy tính để bàn, laptop sử dụng kết nối USB. Sử dụng trong mạch Camera Cửa và mạch HomeServer, làm phần thu âm thanh cho hai mạch trên. Thông số kỹ thuật của micro được sử dụng trong đề tài

Tần số đáp ứng: 100-16kHz

Độ nhạy: -47dBV/Pascal

Hỗ trợ kết nối USB 2.0



**Hình 2-5 Microphone USB**

#### **2.1.6 Hiển thị**

Hiện nay thiết bị hiện thị trên thị trường có rất nhiều loại tuy vào nhu cầu của người sử dụng và vẫn đang ngày một phát triển.

Trong nội dung đề tài sử dụng các loại thiết bị hiển thị:

- Màn hình LCD 1602: Dùng trong mạch điều khiển thiết bị bằng relay



Hình 2-6 Màn hình LCD 1602

- Màn hình Oled 0.9 inch: Dùng trong mạch HomeServer
- Màn hình máy tính 18.5 inch: Dùng kết nối hiển thị với mạch HomeServer

### 2.1.7 Nút nhấn

Đây là một loại ngoại vi cơ bản, phổ biến trong các thiết bị điện tử. Mục đích chính là để tắt/mở nguồn, giao tiếp cơ bản với thiết bị phần cứng. Một số loại nút nhấn phổ biến

- **Nút nhấn cắm loại 6mm/12mm:** Đây là loại button rất phổ biến, cũng như đèn LED, loại button này cũng có các kính thước cạnh 6mm hoặc 12mm. Cấu tạo gồm hai cặp chân nên khá chắc chắn ghi hàn và độ bền cao. Ngoài ra cũng có loại tương tự nhưng thiết kế gồm hai chân tùy thuộc vào yêu cầu thiết kế



Hình 2-7 Nút nhấn 6mm



Hình 2-8 Nút nhấn 12mm

- **Nút nhấn dán:** Kích thước nhỏ 2-3mm vì vậy rất phù hợp cho những mạch yêu cầu về kích thước.



**Hình 2-9 Nút nhấn dán**

- **Nút bấm PLC:** Những loại nút bấm này thường được dùng để chế tạo những đồ trong công nghiệp, hoặc những máy móc lớn cần bấm nhiều và cần đèn trạng thái. Nói một cách nôm na, nút bấm PLC là nút bấm lớn với một cái đèn bên dưới nút bấm. Loại này đôi khi có đèn, đôi khi lại không. Với loại không có đèn thì cũng có 2 chân như các loại ở trên, còn loại có đèn thì có đến 4 chân (2 chân của button, 1 chân dương và 1 chân âm của led).



**Hình 2-10 Nút nhấn PLC**

Loại nút nhấn sử dụng trong đè tài là loại nút nhấn dán 3mm và loại cắm 6mm

Cụ thể:

- Nút nhấn cắm loại 6mm: Mạch HomeServer, mạch camera chuông cửa, mạch điều khiển thiết bị hồng ngoại, mạch điều khiển thiết bị relay
- Nút nhấn dán: mạch điều khiển thiết bị relay

### 2.1.8 LED chiếu sáng

#### Giới thiệu

- Đèn led có tuổi thọ và hiệu suất lớn hơn nhiều lần đèn sợi đốt và hiệu quả hơn so với hầu hết các loại đèn huỳnh quang. Một số chip có khả năng phát ra hơn 300 lumen / watt . Năm 2014, thị trường LED đạt 2 tỷ USD và dự kiến có thể đạt mức 25 tỷ USD vào năm 2023. Theo thống kê trong năm 2016, các thiết bị chiếu sáng ứng dụng công nghệ led mới chỉ chiếm 10% thị phần so với các công nghệ chiếu sáng khác.

- Không giống như hầu hết các bóng đèn huỳnh quang (huỳnh quang compact hoặc đèn CFL), LED phát sáng hoàn toàn mà không cần thời gian khởi động. Do vậy tuổi thọ của chúng cao hơn đèn huỳnh quang. Chi phí ban đầu để mua đèn led thường cao hơn loại sợi đốt hay huỳnh quang, tuy nhiên xét về mức độ tiết kiệm điện năng và tuổi thọ thì chúng được đánh giá tiết kiệm chi phí hơn.

- Hiện nay hầu hết các sản phẩm được thiết kế theo tiêu chuẩn có thể thay thế trực tiếp cho bóng đèn sợi đốt hoặc huỳnh quang. Trên bao bì thường ghi rõ lumen, công suất watt, nhiệt độ màu, phạm vi nhiệt độ hoạt động. Chúng không phát ra ánh sáng theo mọi hướng, và các đặc tính hướng của chúng ảnh hưởng đến việc thiết kế, mặc dù ngày nay đã có không ít những thiết kế có góc chiếu sáng 360 độ.

- Giống như hầu hết các thiết bị chiếu sáng khác, led bị ảnh hưởng bởi nhiệt độ cao, mặc dù chúng phát ra rất ít nhiệt khi hoạt động. Nhưng phần nhiệt độ này cũng đủ để gây ra những tổn hại như giảm lumen, giảm tuổi thọ. Do đó, trong thiết kế thường có thêm bộ phận tản nhiệt, làm mát. Tuổi thọ của đèn led phụ thuộc lớn vào chất lượng của bộ tản nhiệt.

- Để hoạt động, chip led đòi hỏi phải chuyển đổi dòng điện từ xoay chiều sang dòng điện một chiều bằng một thiết bị biến áp và chuyển đổi gọi là driver. Đèn led đi kèm một driver chất lượng có thể đảm bảo tuổi thọ dài cho và cung cấp các tính năng điều khiển ánh sáng. Nó có thể được đặt bên trong bóng đèn (loại tích hợp) hoặc được đặt bên ngoài (loại độc lập). Tùy theo ứng dụng chiếu sáng mà được áp dụng driver khác nhau (ví dụ: như trình driver ngoài trời cho ánh sáng đường phố, driver điểm cho chiếu sáng trong nhà và driver tuyến tính cho các đèn quảng cáo).

### Ưu điểm

- Phát ra nhiều quang thông ánh sáng hơn các loại khác với cùng mức công suất.
- Kích thước nhỏ gọn hơn so với các loại đèn khác
- Có thể bật và tắt nhiều lần mà không ảnh hưởng tới tuổi thọ
- Có thể kết hợp với dimmer.
- Có tuổi thọ rất dài.
- Chiếu sáng theo hướng

### Khuyết điểm

- Giá khởi điểm cao
- Ánh sáng bị ảnh hưởng bởi nhiệt độ
- Rất nhạy với điện áp
- Chất lượng ánh sáng

### Loại Led sử dụng

Led RGB 5050



Hình 2-11 Led RGB 5050

### Thông số kỹ thuật

Loại LED	Điện áp thuận Vf	Dòng hoạt động If
Red	1.8V – 2.4V	20mA
Green	2.8V – 3.6V	20mA
Blue	2.8V – 3.6V	20mA

### 2.1.9 LED phát hồng ngoại

#### Giới thiệu

Đèn LED phát hồng ngoại (IR LED) hoạt động giống như đèn LED thông thường, nhưng có thể sử dụng các vật liệu khác nhau để tạo ra ánh sáng hồng ngoại.

Một đèn LED phát hồng ngoại, giống như tất cả các đèn LED, một loại diode, hoặc chất bán dẫn đơn giản.

Điốt được thiết kế sao cho dòng điện chỉ có thể chạy theo một hướng. Khi dòng điện chạy, các electron rời từ một phần của diode vào lỗ trên một phần khác. Để rời vào các lỗ này, các electron phải làm năng lượng dưới dạng photon tạo ra ánh sáng.

Bước sóng và màu sắc của ánh sáng được tạo ra phụ thuộc vào vật liệu được sử dụng trong diode.

Đèn LED phát hồng ngoại sử dụng vật liệu tạo ra ánh sáng trong phần hồng ngoại của quang phổ, tức là, ngay dưới những gì mắt người có thể nhìn thấy.

Đèn LED hồng ngoại khác nhau có thể tạo ra ánh sáng hồng ngoại của các bước sóng khác nhau, giống như các đèn LED khác nhau tạo ra ánh sáng có màu sắc khác nhau.

– Một nơi rất phổ biến để tìm đèn LED hồng ngoại là điều khiển từ xa cho TV hoặc thiết bị khác như màn hình quảng cáo, quạt điện, điều hòa....

– Một hoặc nhiều đèn LED bên trong điều khiển từ xa truyền các xung ánh sáng hồng ngoại nhanh đến một máy thu trên TV. Người nhận sau đó giải mã và diễn giải các xung này thành một lệnh và thực hiện thao tác mong muốn.

– Ánh sáng hồng ngoại cũng có thể được sử dụng để truyền dữ liệu giữa các thiết bị điện tử.

– Điện thoại di động, trợ lý kỹ thuật số cá nhân (PDA) và một số máy tính xách tay có thể có đèn LED và bộ thu hồng ngoại được thiết kế để truyền dữ liệu tầm ngắn.

– Một số bàn phím không dây và chuột máy tính cũng sử dụng đèn LED phát hồng ngoại và bộ thu để thay thế cáp.

– Mặc dù vô hình đối với mắt người, nhiều loại camera và cảm biến khác có thể nhìn thấy ánh sáng hồng ngoại. Điều này làm cho công nghệ LED hồng ngoại phù hợp với các ứng dụng như hệ thống an ninh và kính nhìn đêm. Nhiều camera an ninh và máy quay phim sử dụng đèn LED hồng ngoại để cung cấp chế độ nhìn ban đêm.

### 2.1.10 LED thu hồng ngoại

Tương ứng với LED phát hồng ngoại ở các thiết bị cần có thêm LED thu để nhận dữ liệu truyền từ LED phát.

### 2.1.11 Giao tiếp UART

**UART** (Universal Asynchronous Receiver – Transmitter). Nó là một mạch tích hợp được sử dụng trong việc truyền dẫn dữ liệu nối tiếp giữa máy tính và các thiết bị ngoại vi.

**UART** có chức năng chính là truyền dữ liệu nối tiếp. Trong UART, giao tiếp giữa hai thiết bị có thể được thực hiện theo hai phương thức là giao tiếp dữ liệu nối tiếp và giao tiếp dữ liệu song song.

Trong giao tiếp UART có các thông số chính:

Baud rate (tốc độ baud ): Khoảng thời gian để 1 bit được truyền đi. Phải được cài đặt giống nhau ở cả phần gửi và nhận

Frame (khung truyền): Khung truyền quy định về mỗi lần truyền bao nhiêu bit

Start bit: là bit đầu tiên được truyền trong 1 Frame. Báo hiệu cho thiết bị nhận có một gói dữ liệu sắp đc truyền đến. Đây là bit bắt buộc

Data: dữ liệu cần truyền. Bit có trọng số nhỏ nhất LSB được truyền trước sau đó đến bit MSB.

Parity bit: kiểm tra dữ liệu truyền có đúng không

Stop bit : là 1 hoặc các bit báo cho thiết bị rằng các bit đã được gửi xong. Thiết bị nhận sẽ tiến hành kiểm tra khung truyền nhằm đảm bảo tính đúng đắn của dữ liệu. Đây là bit bắt buộc

### 2.1.12 Giao tiếp I2C

**I2C** (Inter-Integrated Circuit). Nó là một giao thức giao tiếp được phát triển bởi Philips Semiconductors để truyền dữ liệu giữa một bộ xử lý trung tâm với nhiều IC trên cùng một board mạch chỉ sử dụng hai đường truyền tín hiệu.

Do tính đơn giản của nó nên loại giao thức này được sử dụng rộng rãi cho giao tiếp giữa vi điều khiển và mảng cảm biến, các thiết bị hiển thị, thiết bị IoT, EEPROMs, v.v ...

Đây là một loại giao thức giao tiếp nối tiếp đồng bộ. Nó có nghĩa là các bit dữ liệu được truyền từng bit một theo các khoảng thời gian đều đặn được thiết lập bởi một tín hiệu đồng hồ tham chiếu.

Một số đặc điểm quan trọng của giao thức giao tiếp I2C:

Chỉ cần có hai đường bus (dây) chung để điều khiển bất kỳ thiết bị / IC nào trên mạng I2C

Không cần thỏa thuận trước về tốc độ truyền dữ liệu như trong giao tiếp UART. Vì vậy, tốc độ truyền dữ liệu có thể được điều chỉnh bất cứ khi nào cần thiết

Cơ chế đơn giản để xác thực dữ liệu được truyền

Sử dụng hệ thống địa chỉ 7 bit để xác định một thiết bị / IC cụ thể trên bus I2C

Các mạng I2C dễ dàng mở rộng. Các thiết bị mới có thể được kết nối đơn giản với hai đường bus chung I2C

### 2.1.13 Giao tiếp GPIO

**GPIO** (General-purpose input/output – “cổng vào/ra vận năng”) không được xác định một mục đích đặc biệt nào. Ý tưởng là đôi khi người tích hợp hệ thống tạo ra một hệ thống đầy đủ dùng chip có thể cần dùng một vài đường điều khiển số bổ sung và việc có sẵn chúng trên chip có thể bớt công sức bố trí một mạch điện mới.

## 2.2 Phần mềm

### 2.2.1 Lý thuyết về các giao thức mạng được sử dụng

#### 2.2.1.1 Giao thức TCP, UDP

**TCP (Transmission Control Protocol - "Giao thức điều khiển truyền vận")** là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.

**UDP (User Datagram Protocol)** là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.

Trong nội dung đề tài các giao thức TCP, UDP được trực tiếp sử dụng làm để tạo các gói tin mạng chạy cục bộ trong mạng LAN giữa các thiết bị chức năng và Home server đồng thời cũng là giao thức tầng vận chuyển của tất cả các giao thức lớp trên

#### 2.2.1.2 Giao thức MQTT

##### **MQTT (Message Queue Telemetry Transport)**

Đây là một giao thức truyền thông điệp (message) theo mô hình publish/subscribe (xuất bản – theo dõi), sử dụng băng thông thấp, độ tin cậy cao và có khả năng hoạt động trong điều kiện đường truyền không ổn định.

MQTT là một giao thức nhắn tin gọn nhẹ được thiết kế để liên lạc nhẹ giữa các thiết bị và hệ thống máy tính. MQTT được thiết kế ban đầu cho các mạng SCADA, các kịch

bản sản xuất và băng thông thấp, MQTT đã trở nên phổ biến gần đây do sự phát triển của Internet-of-Things (IoT).

Kiến trúc mức cao (high-level) của MQTT gồm 2 phần chính là Broker và Clients.

Trong đó, broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ client. Nhiệm vụ chính của broker là nhận mesage từ publisher, xếp các message theo hàng đợi rồi chuyển chúng tới một địa chỉ cụ thể. Nhiệm vụ phụ của broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs,...

Client thì được chia thành 2 nhóm là publisher và subscriber. Client là các software components hoạt động tại edge device nên chúng được thiết kế để có thể hoạt động một cách linh hoạt (lightweight). Client chỉ làm ít nhất một trong 2 việc là publish các message lên một topic cụ thể hoặc subscribe một topic nào đó để nhận message từ topic này

#### **2.2.1.3 Giao thức RTP/ RTSP**

RTP (**Real time transfer protocol** – “**giao thức truyền tải thời gian thực**”) là một giao thức mạng để chuyển tập tin, video, âm thanh qua mạng IP. RTP được sử dụng rộng rãi trong các hệ thống truyền thông và giải trí liên quan đến các streaming media như gọi điện thoại, các ứng dụng hội nghị truyền hình, các dịch vụ truyền hình và các tính năng push-to-talk dựa trên nền web. RTP chạy trên giao thức UDP (User Diagram Protocol), RTP được sử dụng kết hợp với RTP Control Protocol (RTCP). Trong khi RTP mang các luồng truyền thông (ví dụ: âm thanh và video), RTCP được sử dụng để giám sát số liệu truyền tải và chất lượng dịch vụ (QoS) và đồng bộ hóa nhiều luồng. RTP là một trong những cơ sở kỹ thuật của Voice over IP và trong ngữ cảnh này thường được sử dụng kết hợp với một giao thức báo hiệu như Session Initiation Protocol (SIP) thiết lập kết nối qua mạng

RTSP (**Real Time Streaming Protocol** – “**Giao thức truyền tin thời gian thực**”) là một giao thức điều khiển truyền thông mạng ở tầng ứng dụng được thiết kế để sử dụng trong các hệ thống giải trí và truyền thông để điều khiển máy chủ chứa các dữ liệu truyền tin đa phương tiện (streaming media). Giao thức này được sử dụng để thiết lập và điều

khiến các phiên truyền thông giữa các trạm cuối. Các máy khách của các máy chủ truyền thông ban ra các lệnh kiểu VCR, chẳng hạn như chơi, thâu và tạm dừng, để điều khiển thời gian thực của các phương tiện truyền tin trực tuyến từ máy chủ tới máy khách (Video On Demand) hoặc từ máy khách đến máy chủ (Voice Recording).

#### **2.2.1.4 Giao thức Websocket**

WebSocket là một giao thức giúp truyền dữ liệu hai chiều giữa server-client qua một kết nối TCP duy nhất. Hơn nữa, webSocket là một giao thức được thiết kế để truyền dữ liệu bằng cách sử dụng cổng 80 và cổng 443 và nó là một phần của HTML5. Vì vậy, webSockets có thể hoạt động trên các cổng web tiêu chuẩn, nên không có rắc rối về việc mở cổng cho các ứng dụng, lo lắng về việc bị chặn bởi các tường lửa hay proxy server.

Không giống với giao thức HTTP là cần client chủ động gửi yêu cầu cho server, client sẽ chờ đợi để nhận được dữ liệu từ máy chủ. Hay nói cách khác với giao thức Websocket thì server có thể chủ động gửi thông tin đến client mà không cần phải có yêu cầu từ client.

Tất cả dữ liệu giao tiếp giữa client-server sẽ được gửi trực tiếp qua một kết nối cố định làm cho thông tin được gửi đi nhanh chóng và liên tục khi cần thiết. WebSocket làm giảm độ trễ bởi vì một khi kết nối WebSocket được thành lập, server không cần phải chờ đợi cho một yêu cầu từ client.

Tương tự như vậy, client có thể gửi tin nhắn đến server bất cứ lúc nào. Yêu cầu duy nhất này giúp làm giảm đáng kể độ trễ, mà sẽ gửi một yêu cầu trong khoảng thời gian, cho dù thông điệp có sẵn.

#### **2.2.1.5 Giao thức HTTP**

HTTP (HyperText Transfer protocol – “Giao thức truyền tải siêu văn bản). Được sử dụng trong World Wide Web dùng để truyền tải dữ liệu Web server đến các trình duyệt Web và ngược lại. Giao thức này sử dụng cổng 80 (port 80) là chủ yếu.

Đây là giao thức chủ yếu để truyền tải dữ liệu Web với người dùng.

### 2.2.1.6 Giao thức mDNS

Multicast Domain Name System. Giống như hệ thống phân giải tên miền DNS nhưng dùng trong mạng LAN với đuôi .local và không cần máy chủ DNS. Nếu máy tính cần gửi yêu cầu đến một tên miền có đuôi là **.local**, nó sẽ gửi một truy vấn multicast đến tất cả các thiết bị khác trên mạng LAN có hỗ trợ mDNS, thiết bị sẽ tự nhận dạng tên miền của chính mình. Sau đó thiết bị có tên phù hợp sẽ phản hồi bằng cách gửi một multicast khác kèm địa chỉ IP của nó. Đến lúc này máy tính của bạn biết địa chỉ IP của thiết bị và có thể gửi các yêu cầu bình thường. Kỹ thuật này giúp các thiết bị trong mạng LAN có thể tìm thấy địa chỉ IP và giao tiếp với nhau khi biết được tên miền.

Đây là giải pháp hữu hiệu thay thế cho IP tĩnh vốn cần nhiều cấu hình. Trên window. Sử dụng Apple's Bonjour Service để sử dụng dịch vụ mDNS. Trên máy tính Linux sử dụng package Avahi.

## 2.2.2 Các ngôn ngữ lập trình, thư viện và framework được sử dụng

### 2.2.2.1 Ngôn ngữ lập trình Python

Trong nội dung đề tài ngôn ngữ Python được dùng để lập trình các chức năng sau:

- Tạo Webserver đáp ứng các gói tin truy vấn đến từ trình duyệt và giao tiếp với người dùng. Webserver được chạy trên nền Flask framework. Do Flask là một framework gọn nhẹ và khá đơn giản nên khi chạy thực tế cần được kết hợp với bộ server giúp đáp ứng các yêu cầu cao hơn như cân bằng tải, reverse proxy cụ thể trong đề tài sử dụng cặp đôi: Nginx và WSGI.
- Đảm nhiệm các chức năng xử lý ảnh
- Viết chương trình hoạt động cho mạch raspberry Pi
- Viết bộ chuyển đổi giữa WebSocket và MQTT hoạt động trên máy chủ đám mây

### **2.2.2.2 Ngôn ngữ lập trình C/C++**

Trong nội dung đề tài ngôn ngữ Ngôn ngữ lập trình C/C++ được dùng để lập trình các chức năng sau:

- Viết firmware cho các mạch sử dụng ESP8266 và PIC
- Ngôn ngữ C được nhúng lên vi điều khiển PIC16F877A bằng trình biên dịch PICC
- Ngôn ngữ C++ được nhúng lên ESP8266 bằng trình biên dịch Adruino IDE

### **2.2.2.3 Ngôn ngữ lập trình Javascript**

Đây là ngôn ngữ lập trình cơ bản nhất có thể chạy trên trình duyệt và được tất cả các trình duyệt thông dụng sử dụng. Ngôn ngữ cho phép lập trình viên tạo các đối tượng động trên trang web, bắt các sự kiện mà người dùng tạo ra và liên kết giữa trình duyệt người dùng với webserver.

Bộ thư viện SocketIO: Với những trang web bình thường chỉ cần người dùng gửi yêu cầu để server truy xuất dữ liệu. Tuy nhiên để đảm bảo tính thời gian thực cần thêm chiều ngược lại từ Server gửi ngược lại đến trình duyệt. Giả sử một người nhấn thiết bị chuông cửa. Trình duyệt không thể biết bao giờ có tín hiệu chuông để báo đến người dùng. Do đó Server có nhiệm vụ thông báo lại đến trình duyệt sự kiện nhấn chuông. Đó là khi bộ thư viện SocketIO phát huy tác dụng:

### **2.2.2.4 HTML và CSS**

**HTML** (viết tắt cho *HyperText Markup Language*, hay là "Ngôn ngữ Đánh dấu Siêu văn bản") là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web với các mẩu thông tin được trình bày trên World Wide Web. Cùng với CSS và JavaScript, HTML tạo ra bộ ba nền tảng kỹ thuật cho World Wide Web. HTML được định nghĩa như là một ứng dụng đơn giản của SGML và được sử dụng trong các tổ chức cần đến các yêu cầu xuất bản phức tạp. HTML đã trở thành một chuẩn Internet do tổ chức World Wide Web Consortium (W3C) duy trì. Phiên bản chính thức mới nhất của HTML là HTML 4.01 (1999). Sau đó, các nhà phát triển đã thay thế nó bằng XHTML. Hiện nay, HTML đang được phát triển tiếp với phiên bản HTML5 hứa hẹn mang lại diện mạo mới

cho Web. Bằng cách dùng HTML động hoặc Ajax, lập trình viên có thể được tạo ra và xử lý bởi số lượng lớn các công cụ, từ một chương trình soạn thảo văn bản đơn giản – có thể gõ vào ngay từ những dòng đầu tiên – cho đến những công cụ xuất bản WYSIWYG phức tạp.**Hypertext** là cách mà các trang Web (các tài liệu HTML) được kết nối với nhau. Và như thế, đường link có trên trang Web được gọi là Hypertext. Như tên gọi đã gợi ý, **HTML** là ngôn ngữ đánh dấu bằng thẻ (**Markup Language**), nghĩa là bạn sử dụng HTML để đánh dấu một tài liệu text bằng các **thẻ (tag)** để nói cho trình duyệt Web cách để cấu trúc nó để hiển thị ra màn hình.

**CSS** (viết tắt cho *Cascading Style Sheets*, hay là "tập tin định kiểu theo tầng") được dùng để miêu tả cách trình bày các tài liệu viết bằng ngôn ngữ HTML và XHTML. Ngoài ra ngôn ngữ định kiểu theo tầng cũng có thể dùng cho XML, SVG, XUL. Các đặc điểm kỹ thuật của CSS được duy trì bởi World Wide Web Consortium (W3C). Thay vì đặt các thẻ quy định kiểu dáng cho văn bản HTML (hoặc XHTML) ngay trong nội dung của nó, bạn nên sử dụng CSS. Dùng để thiết kế giao diện web điều chỉnh cấu trúc của HTML sao cho phù hợp với người dùng. Hạn chế tối thiểu việc làm rối mã HTML của trang Web bằng các thẻ quy định kiểu dáng (chữ đậm, chữ in nghiêng, chữ có gạch chân, chữ màu), khiến mã nguồn của trang Web được gọn gàng hơn, tách nội dung của trang Web và định dạng hiển thị, dễ dàng cho việc cập nhật nội dung. Tạo ra các kiểu dáng có thể áp dụng cho nhiều trang Web, giúp tránh phải lặp lại việc định dạng cho các trang Web giống nhau

### **2.2.2.5 Bộ thu vien Gstreamer**

### **2.2.2.6 Bộ thu vien Ffmpeg**



**Hình 2-12 Giới thiệu bộ thu vien Ffmpeg**

FFmpeg là một bộ mã nguồn mở và miễn phí, bao gồm các thư viện và chương trình để xử lý video, âm thanh, các file và luồng media.

Trên thực tế, người dùng sử dụng một ffmpeg thực thi duy nhất, thông qua rất nhiều kiểu chuyển đổi (hợp lý và nhất quán) để thực hiện các thao tác trên các file media.

## **2.2.3 Kỹ thuật truyền streaming đa phương tiện**

### **2.2.3.1 Giới thiệu**

### **2.2.3.2 Codec**

Các tập tin phim ảnh, âm nhạc thường rất lớn nên rất khó chia sẻ qua mạng. Để tăng tốc độ tải xuống, các codec toán học được dùng để mã hóa, hoặc rút gọn tín hiệu truyền tải, sau đó giải mã để xem và chỉnh sửa. Nếu không có codec, việc tải các file video và audio sẽ lâu hơn bấy giờ từ 3 tới 5 lần.

Trên Internet hiện tại có hàng trăm codec và bạn thường sẽ phải sử dụng kết hợp để phát tập tin. Có các codec để nén audio, video, tập tin đa phương tiện stream qua mạng, bài diễn thuyết, cuộc họp qua video, phát MP3, chụp ảnh màn hình.

Nhiều người khi chia sẻ tập tin trên mạng còn sử dụng các codec lạ để nén tập tin. Điều đó khiến người dùng khi tải sẽ không biết được họ phải dùng codec nào. Thông thường, bạn sẽ cần khoảng 10 tới 12 codec để phát các loại nhạc, phim.

Trong phạm vi đề tài sử dụng các loại Codec sau:

### **H264:**

H.264/MPEG-4 Part 10 hay AVC (Advanced Video Coding - Mã hóa video cao cấp), thường được gọi tắt là H.264, là một chuẩn mã hóa/giải mã video và định dạng video đang được sử dụng rộng rãi nhất hiện nay để ghi, nén và chia sẻ video phân giải cao, dựa trên việc bù trừ chuyển động (motion-compensation) trên từng block (block oriented).

H.264 được chấp thuận bởi tổ chức truyền thông quốc tế ITU-T với tên gọi Recommendation H.264 và bởi tổ chức chuẩn hóa quốc tế (ISO/IEC) với tên gọi International Standard 14496-10 (MPEG-4 part 10) Advanced Video Coding. Lần đầu tiên được đề xuất vào năm 1998, nhóm chuyên gia nén video (VCEG – ITU-T SG16 Q.6) kêu gọi đưa ra ý tưởng cho dự án gọi là H.26L, với mục đích tăng gấp đôi độ hiệu quả nén video so với các chuẩn nén video hiện có áp dụng cho nhiều loại ứng dụng, thiết bị đa dạng. Thiết kế dự thảo (drafting) đầu tiên được phê chuẩn vào tháng 10 năm 1999. Vào tháng 12 năm 2001, VCEG và nhóm chuyên gia về ảnh động (MPEG - ISO/IEC JTC 1/SC 29/WG 11) hợp tác thành nhóm Joint Video Team (JVT), được lập ra để hoàn thành bản dự thảo về chuẩn nén video mới để trình chấp thuận với tên H.264/AVC vào tháng 3 năm 2003. Chuẩn ITU-T H.264 cùng với ISO/IEC MPEG-4 AVC được hợp tác duy trì nên nội dung của chúng về kỹ thuật là tương đương.

H.264 ngày nay được biết đến nhiều trong các ứng dụng như đĩa Blu-ray: mọi đĩa Blu-ray đều có khả năng giải mã H.264, và trong các dịch vụ stream video như Youtube, Vimeo, iTunes Store...; các phần mềm web như Adobe Flash, Microsoft Silverlight, HTML5...; Các dịch vụ truyền hình HDTV mặt đất (ATSC, ISDB-T, DVB-T, DVB-T2), cáp (DVB-C) và vệ tinh (DVB-S và DVB-S2).

### 2.2.3.3 Giao thức NAT

**Biên dịch địa chỉ mạng** (*Network address translation - NAT*) trong mạng máy tính là quá trình thay đổi thông tin địa chỉ IP trong gói tin đang được truyền qua một thiết bị định tuyến.

Loại NAT đơn giản nhất cung cấp việc biên dịch một - một cho một địa chỉ IP. Chuẩn RFC 2663 gọi loại NAT này là NAT cơ bản. Nó đôi khi cũng được gọi là NAT một-một. Trong kiểu phiên dịch này chỉ có địa chỉ IP, IP header, phần kiểm lỗi cần phải thay đổi (đối với TCP/UDP thì như vậy là đủ, nhưng một số giao thức tầng cao hơn có thể yêu cầu biên dịch nhiều hơn). NAT cơ bản có thể sử dụng để kết nối hai mạng máy tính có địa chỉ không tương thích.

Thông thường trong một mạng máy tính, các máy trong mạng được gán địa chỉ IP riêng, mạng chỉ có một hoặc một vài địa chỉ công cộng được gán cho thiết bị trực tiếp kết nối với mạng toàn cầu. Để các máy trong mạng có thể kết nối với bên ngoài, người ta sử dụng loại NAT một-nhiều. Trong kiểu NAT này, thông tin ở bậc cao trong các gói gửi đi hơn như cổng TCP/UDP được thay đổi và lưu trữ một bảng biên dịch, các gói tin trả về sẽ được tham chiếu với bảng này để thay đổi số cổng trước khi gửi tới máy cần nhận. Trong chuẩn RFC 2663, người ta còn gọi loại NAT này là NAPT (network address and port translation: biên dịch địa chỉ mạng và cổng). Các tên gọi khác là PAT (port address translation: biên dịch địa chỉ cổng), giả dạng IP, quá tải NAT. Do đây là loại thường gặp nhất của NAT nên khi gọi NAT, ta ngầm hiểu đó là NAT một-nhiều.

Như đã nói, phương pháp này cho phép kết nối hoạt động chỉ khi yêu cầu kết nối bắt nguồn từ bên trong mạng. Ví dụ một trình duyệt web từ bên trong có thể kết nối tới một trang mạng bên ngoài, nhưng một trình duyệt từ bên ngoài không thể truy cập vào một trang web bên trong. Tuy nhiên, hiện nay hầu hết các thiết bị NAT đều cho phép cấu hình các giá trị trong bảng biên dịch sử dụng trong thời gian dài, từ đó ta có thể thiết lập NAT tĩnh hay chuyển tiếp cổng để cho phép các truy cập từ mạng "bên ngoài" vào các dịch vụ trong mạng nội bộ.

Trong những năm giữa thập niên 90, NAT trở thành một công cụ phổ biến để giải quyết vấn đề cạn kiệt địa chỉ IPv4. Nó trở thành một chức năng thiết yếu trong thiết bị

định tuyến cho các kết nối tại nhà và công ty nhỏ. Hầu hết các hệ thống sử dụng thiết lập NAT để cho phép nhiều máy chủ trong mạng cục bộ có thể kết nối mạng toàn cầu thông qua chỉ một địa chỉ IP công cộng.

Biên dịch địa chỉ mạng bị hạn chế bởi chất lượng của kết nối mạng và yêu cầu nghiêm ngặt trong việc cài đặt. Ngoài ra tất cả các kiểu NAT đều bẻ gãy chuẩn kết nối khi truyền thông với mạng toàn cầu. NAT-T (NAT traversal) được đề xuất để xử lý vấn đề này.

#### **2.2.3.4 Các loại NAT**

##### **2.2.3.4.1 Static Nat**

Static NAT được dùng để chuyển đổi một địa chỉ IP này sang một địa chỉ khác một cách cố định, thông thường là từ một địa chỉ cục bộ sang một địa chỉ công cộng và quá trình này được cài đặt thủ công, nghĩa là địa chỉ ánh xạ và địa chỉ ánh xạ chỉ định rõ ràng tương ứng duy nhất.

Static NAT rất hữu ích trong trường hợp những thiết bị cần phải có địa chỉ cố định để có thể truy cập từ bên ngoài Internet. Những thiết bị này phổ biến là những Server như Web, Mail,...

##### **2.2.3.4.2 Dynamic NAT**

Dynamic NAT được dùng để ánh xạ một địa chỉ IP này sang một địa chỉ khác một cách tự động, thông thường là ánh xạ từ một địa chỉ cục bộ sang một địa chỉ được đăng ký. Bất kỳ một địa chỉ IP nào nằm trong dải địa chỉ IP công cộng đã được định trước đều có thể được gán một thiết bị bên trong mạng

##### **2.2.3.4.3 NAT Overload**

Nat Overload là một dạng của Dynamic NAT, nó thực hiện ánh xạ nhiều địa chỉ IP thành một địa chỉ (many - to - one) và sử dụng các địa chỉ số cổng khác nhau để phân biệt cho từng chuyển đổi. NAT Overload còn có tên gọi là PAT (Port Address Translation).

Chỉ số cổng được mã hóa 16 bit, do đó có tới 65536 địa chỉ nội bộ có thể được chuyển đổi sang một địa chỉ công cộng.

Việc phân loại và xác định loại NAT sử dụng có ý nghĩa rất lớn trong việc triển khai hệ thống vốn phụ thuộc vào các kết nối P2P.

### 2.2.3.5 Mạng P2P

[<sup>13]</sup>Mạng ngang hàng (tiếng Anh: peer-to-peer network), còn gọi là mạng đồng đẳng, là một mạng máy tính trong đó hoạt động của mạng chủ yếu dựa vào khả năng tính toán và băng thông của các máy tham gia chứ không tập trung vào một số nhỏ các máy chủ trung tâm như các mạng thông thường. Mạng đồng đẳng thường được sử dụng để kết nối các máy thông qua một lượng kết nối dạng ad hoc. Mạng đồng đẳng có nhiều ứng dụng. Ứng dụng thường xuyên gấp nhất là chia sẻ tệp tin, tất cả các dạng như âm thanh, hình ảnh, dữ liệu,... hoặc để truyền dữ liệu thời gian thực như điện thoại VoIP.

Một mạng đồng đẳng đúng nghĩa không có khái niệm máy chủ và máy khách, nói cách khác, tất cả các máy tham gia đều bình đẳng và được gọi là peer, là một nút mạng đóng vai trò đồng thời là máy khách và máy chủ đối với các máy khác trong mạng.

**Ưu thế của mạng đồng đẳng:** Một mục đích quan trọng của mạng đồng đẳng là trong mạng tất cả các máy tham gia đều đóng góp tài nguyên, bao gồm băng thông, lưu trữ, và khả năng tính toán. Do đó khi càng có nhiều máy tham gia mạng thì khả năng tổng thể của hệ thống mạng càng lớn. Ngược lại, trong cấu trúc máy chủ-máy khách, nếu số lượng máy chủ là cố định, thì khi số lượng máy khách tăng lên khả năng chuyển dữ liệu cho mỗi máy khách sẽ giảm xuống.

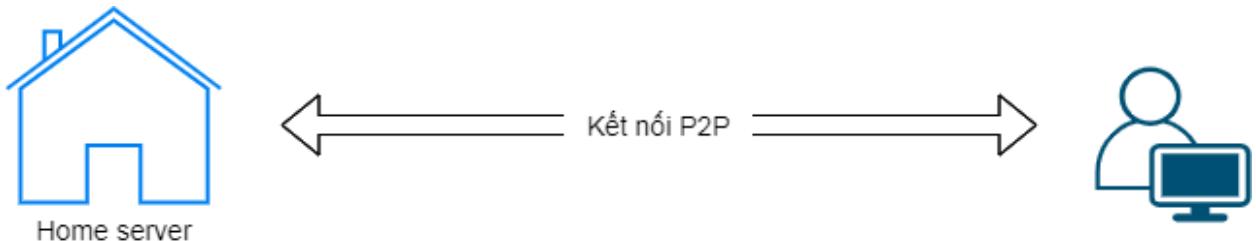
Tính chất phân tán của mạng đồng đẳng cũng giúp cho mạng hoạt động tốt khi một số máy gặp sự cố. Đối với cấu trúc tập trung, chỉ cần máy chủ gặp sự cố thì cả hệ thống sẽ ngưng trệ.

Các vấn đề liên quan đến Streaming hình ảnh, âm thanh trong đè tài đều dựa trên việc thiết lập kết nối đồng đẳng P2P giữa hai thiết bị. Việc kết nối này có hoàn thành được hay không phụ thuộc rất nhiều vào mô hình gồm các Signaling server và Server vận hành kỹ thuật Nat Hole Punching và rộng hơn là giao thức ICE sẽ được trình bày trong phần tới đây.

### 2.2.3.6 Kỹ thuật NAT Hole Punching và ICE

#### Interactive Connectivity Establishment (ICE)

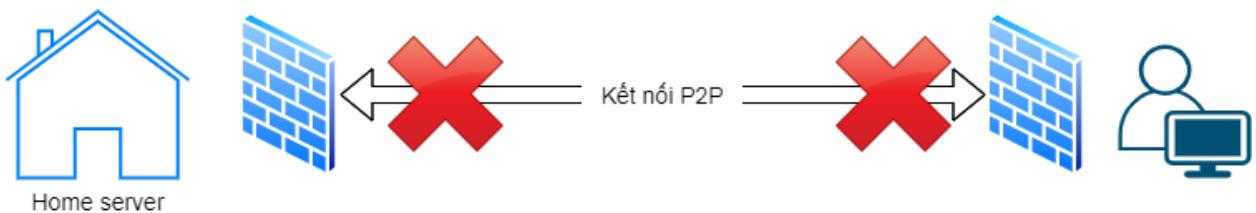
Một kết nối P2P đơn giản được hiểu như hình sau:



**Hình 2-13 Mô hình P2P đơn giản nhất**

Khi đó người dùng chỉ việc kết nối trực tiếp đến thiết bị và trao đổi dữ liệu. Yêu cầu duy nhất là biết được địa chỉ IP và cổng Port và kết nối với giao thức qui định trước mà không có ràng buộc gì. Tuy nhiên mô hình này chỉ hoạt động trong một mạng nội bộ, hoặc mở rộng hơn là cấu hình Router cho việc cố định Port đến thiết bị (Kỹ thuật Port Forwarding) hoặc là giữa các router hỗ trợ Zero Config (Giao thức uPnP,...). Trên thực tế chưa nói đến việc biên dịch địa chỉ mạng, khi đã biết địa chỉ và cổng Port cũng không thể nào kết nối được. Đó là do việc sử dụng công nghệ NAT rộng rãi ở thời điểm hiện tại.

Khi tính đến việc hai máy ở hai mạng khác nhau và kết nối đến Internet sử dụng Router dùng công nghệ NAT, ta sẽ gặp trường hợp như sau: (hình minh họa)



**Hình 2-14 Mô hình bị chặn bởi NAT**

Khi người dùng muốn kết nối với thiết bị, đã biết địa chỉ IP và cổng Port của thiết bị nhưng cũng không thể kết nối được do bên trong mạng còn rất nhiều thiết bị, router không thể biết gói tin gửi đến đâu và hủy gói tin. Có 2 phương pháp gửi để vượt qua được chướng ngại này:

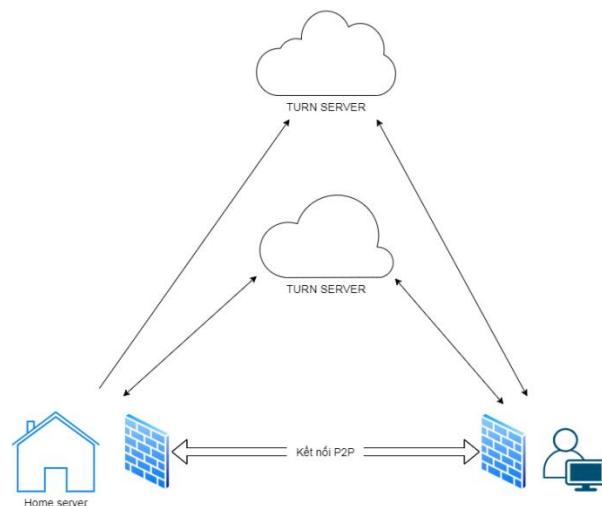
1/ Cấu hình tĩnh Port, Port Forwarding hay NAT port. Việc này giúp cho Router hiểu được gói tin gửi đến địa chỉ ở Port A sẽ được gửi đúng đến thiết bị A ở port nội bộ

qui định. Tuy nhiên do tất cả gói tin từ bên ngoài gửi đến đều có thể vào được bên trong dẫn đến những rủi ro về bảo mật.

2/ Sử dụng kỹ thuật Hold Punching. Phương pháp này hoạt động cần có sự điều hướng của một Server tín hiệu gửi thông tin IP, Port đến các đơn vị trong kết nối. Muốn bên ngoài kết nối được với thiết bị thì cần trải qua các giai đoạn sau:

- + Lấy mỗi bên lấy thông tin về IP và Port (Public của nhau)
- + Bên A và bên B mỗi bên gửi các gói tin (Cùng giao thức vận chuyển UDP/TCP)
 

Các gói tin đầu tiên chắc chắn bị hủy do hai router không hề biết gửi đến ai.
- + Các gói tin sau Router bên A đã biết A gửi dữ liệu đến đâu và cũng đang đợi phản hồi và trong lúc đó các gói tin bên B cũng gửi qua dẫn đến kết nối được thiết lập bên B gửi dữ liệu được đến A và phía bên B cũng tương tự. Kết quả là Hole Punching thành công.
- + Tuy nhiên kỹ thuật trên chỉ áp dụng được khi cổng Port và IP của Router không đổi khi thực hiện mỗi gói tin. Và việc rất tiếc là việc thay đổi không dự đoán được của các Router là rất phổ biến vì lý do bảo mật và lý do quản lý nhiều thiết bị mạng cùng lúc. Bên cạnh đó cũng không thể thực hiện gửi dữ liệu đến tất cả các Port cùng một lúc được vì mỗi bên có hơn 60000 port và khi tính tổng hợp ra lượng trường hợp sẽ là rất lớn. Do đó trong trường hợp thực tế người ta sử dụng thêm một Server để chuyển giao dữ liệu giữa hai bên khi cần thiết (STUN Server)



Hình 2-15 Mô hình TURN Server và TURN Serve

### 2.2.3.7 Công nghệ webRTC



Hình 2-16 Giới thiệu về công nghệ WebRTC

**WebRTC (Web Real-Time Communication)** là một web API được phát triển bởi World Wide Web Consortium (W3C), khả năng hỗ trợ trình duyệt (browser) giao tiếp với nhau thông qua VideoCall, VoiceCall hay transfer data "Peer-to-Peer" (P2P) mà không cần browser phải cài thêm plugins hay phần mềm hỗ trợ nào từ bên ngoài.

Lịch sử của WebRTC là lịch sử của mỗi công ty làm việc ở nhiều mảng khác nhau, ảnh hưởng lẫn nhau. Các công ty đó trở thành các công ty con cho một dự án lớn.

Tháng năm 2011, Google sử dụng nguồn tài chính, công nghệ, kỹ thuật sẵn có của mình đổ vào WebRTC, phát hành dự án mã nguồn mở cho trình duyệt giúp hỗ trợ tương tác với nhau thời gian thực (real-time communications), gọi là WebRTC. Công nghệ này nhằm chuẩn hóa các giao thức IETF vào các API của trình duyệt trong W3C. Đây cũng là thời điểm thúc đẩy việc sáp nhập các công ty con vào dự án, cụ thể là Google mua lại On2 (công ty sáng tạo ra chuẩn video codec VP8), Global IP solutions (công ty được cấp phép phát triển các thành phần cốt lõi nằm trong ngôn ngữ lập trình bậc thấp của WebRTC).

Dự án Chrome WebRTC được Google tạo ra khởi đầu cho các dự án mã nguồn mở mà Google muốn thực hiện, các nhà phát triển lúc này cũng dần bắt đầu chọn G.711, OPUS, và VP8 như một chuẩn video codec mới, nếu muốn hoạt động tốt trên trình duyệt sử dụng các API của WebRTC.

Công ty đầu tiên ứng dụng WebRTC là Ericsson (2011), tháng 12 / 2012, Chrome 23 là phiên bản đầu tiên hỗ trợ các API của WebRTC. Hiện nay (2015), WebRTC hỗ trợ tốt trên trình duyệt Chrome, Opera và Firefox.

Cốt lõi của WebRTC API nằm ở PeerConnection. Nhóm công tác dự án Real-Time Communications mô tả đặc điểm kỹ thuật để phát triển:

Hoàn thành thảo luận về WebRTC tại IETF, xác định các bộ giao thức, xác định thông tin liên lạc thời gian thực trong các trình duyệt web trong khi không có một giao thức nào để thực hiện tốt nhiệm vụ ủy thác (delegation). SIP trên Websockets (RFC 7118) được sử dụng một phần là do các ứng dụng của SIP hầu hết giao tiếp sẵn có của phần mềm mã nguồn mở như JsSIP.

Vấn đề bảo mật phát sinh trong việc truyền tải các streams giữa các client với nhau.

Thảo luận kỹ thuật về việc thực hiện các kênh dữ liệu.

Kinh nghiệm thu được qua các thử nghiệm đầu tiên.

Nhận thông tin phản hồi từ các nhóm và cá nhân khác.

Các phần chính của WebRTC bao gồm:

getUserMedia, cho phép trình duyệt web truy cập vào camera và/hoặc microphone để lấy dữ liệu hình ảnh âm thanh cho việc truyền tải.

RTCPeerConnection dùng để cài đặt videocall/voicecall dùng cho việc truyền tải.

RTCDataChannel cho phép trình duyệt chia sẻ dữ liệu peer-to-peer.

WebRTC API bao gồm chức năng:

getStats cho phép ứng dụng web lấy tập hợp các số liệu thống kê về các session WebRTC. Những dữ liệu thống kê được mô tả trong một tài liệu W3C riêng biệt.

Tính đến tháng 3 năm 2015, IETF WebRTC Codec và Truyền thông xử lý yêu cầu dự thảo đòi hỏi phải triển khai để cung cấp PCMA / PCMU (RFC 3551), tổ chức sự kiện như điện thoại DTMF (RFC 4733), và Opus (RFC 6716) codec âm thanh, tối thiểu là H.264 và video codec VP8. Các kênh PeerConnection, dữ liệu và các phương tiện truyền thông API được nói chi tiết trong W3C.

W3C phát triển ORTC (Object thời gian thực Truyền thông) cho WebRTC. Nên WebRTC thường gọi chính xác là WebRTC1.1.

#### **2.2.3.8 Các khái niệm/API chính của WebRTC**

**Signaling Server.** Trước khi các Peer có thể giao tiếp với nhau, chúng ta phải có một tầng trung gian nào đó để cài đặt kết nối giữa 2 máy đó hay nói cách khác chúng ta cần 1 server ở trung gian, server đó thường gọi là Signaling Server.

**RTCPeerConnection.** Là interface trong bộ WebRTC API được cung cấp bởi Browser, nó thể hiện kết nối WebRTC từ một máy local đến một máy remote, interface này cung cấp các phương thức để cho phép ứng dụng web tạo kết nối, duy trì kết nối, quan sát kết nối và đóng kết nối khi quá trình hội thoại kết thúc.

**MediaStream.** Là interface trong bộ WebRTC API được cung cấp bởi Browser, là interface thể hiện stream truyền tải dữ liệu giữa hai máy sau khi kết nối đã được thiết lập. Dữ liệu audio và video của cuộc hội thoại sẽ được truyền đi trong mỗi stream, đầu vào của máy local này chính là output của máy remote kia.

**RTCDataChannel.** Là interface trong bộ WebRTC API được cung cấp bởi Browser, sau khi kết nối được thiết lập, các kênh truyền dữ liệu sẽ được gán kết vào kết nối đó cho phép các Peer có thể thực hiện việc truyền dữ liệu với nhau thông qua các P2P protocol, theo lý thuyết thì mỗi kết nối có thể có đến 65,534 kênh dữ liệu được gắn kết vào. Tuy nhiên thực tế con số bao nhiêu thì nó phụ thuộc vào Browser.

#### **2.2.3.9 Lợi ích của việc sử dụng WebRTC**

Mã nguồn mở miễn phí: WebRTC là một dự án mã nguồn mở miễn phí. Google cho biết đây là một công cụ truyền thông thời gian thực hoàn toàn miễn phí và có sẵn trên mọi trình duyệt.

Hỗ trợ đa nền tảng: Mặc dù WebRTC vẫn trong giai đoạn phát triển nhưng nó đã hoạt động tốt trên hầu hết mọi trình duyệt của các hệ điều hành bất kỳ. Cho phép lập trình viên viết các đoạn mã HTML làm việc với máy tính hoặc thiết bị di động.

Bảo mật voice và video: Giao thức SRTP (Secure Real-Time Transport Protocol) được dùng để mã hóa và xác thực dữ liệu media. Chống lại các khả năng bị nghe trộm trong quá trình thực hiện tác vụ video hay voice.

Không cần plugin hay phần mềm hỗ trợ: Yếu tố quan trọng giúp WebRTC được đánh giá rất cao chính là khả năng hoạt động không cần đến plugin bên thứ ba mang đến sự tiện lợi, tối ưu tốc độ, tiết kiệm chi phí,...

Tương đối dễ sử dụng: WebRTC có thể được tích hợp trong các dịch vụ web bằng cách dùng JavaScript APIs, các Framework có sẵn.

Sử dụng băng thông hiệu quả: Hỗ trợ nhiều kiểu media và các thiết bị đầu cuối khác nhau, WebRTC sử dụng băng thông hiệu quả hơn, hoạt động tốt trong mọi điều kiện đường truyền mạng.

#### 2.2.3.10 *Bộ thư viện Janus Gateway*



Hình 2-17 Giới thiệu về bộ thư viện JanusGateway

Để thực hiện streaming đa phương tiện từ các giao thức thông dụng như RTP/RTSP, VoIP cần một bộ chuyển đổi để hoạt động tương thích với WebRTC. Cần một công cụ giúp chuyển đổi các giao thức truyền cơ bản để chuyển thành các ứng dụng trên nền webRTC. Một số gateway đang được sử dụng khá phổ biến hiện tại có thể kể tên: Jitsi, Kurento, Janus..

Janus là một máy chủ WebRTC mã nguồn mở, mục đích chung được thiết kế và phát triển bởi Meetecho. Janus là một máy chủ nhẹ đa năng thực hiện các phương tiện để thiết lập truyền thông phương tiện WebRTC giữa các peer-to-peer Phiên bản máy chủ này được thiết kế riêng cho các hệ thống Linux, mặc dù nó có thể được biên dịch và cài đặt trên các máy MacOS. Windows không được hỗ trợ, nhưng nếu đó là một yêu cầu,

Janus được biết là hoạt động trong "Hệ thống con Windows cho Linux" trên Windows 10.

Janus cực kỳ mạnh mẽ ở chỗ nó có khả năng tùy biến cao và cung cấp một số tính năng độc đáo thông qua các plugin như

EchoTest plugin

Streaming plugin

VideoCall plugin

SIP plugin

SIPre plugin

NoSIP plugin

AudioBridge plugin

VideoRoom plugin

Janus TextRoom

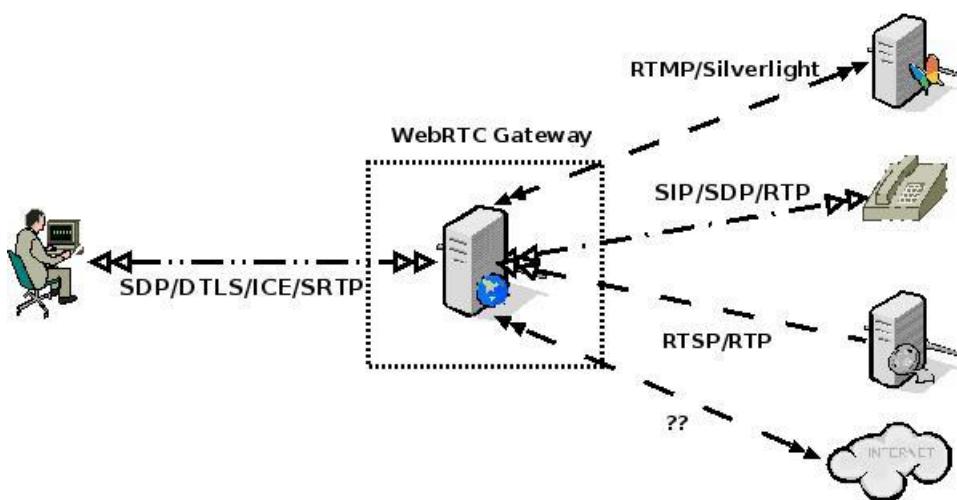
Record&Play plugin

VoiceMail plugin

Lua plugin

JavaScript (Duktape) plugin

Để làm việc được với Janus server thì janus cung cấp các service dưới dạng Restfull API Ngoài ra cung cấp các cách làm việc bằng WebSockets, RabbitMQ, MQTT, Nanomsg and UnixSockets API



Hình 2-18 Minh họa mô hình của Janus Gateway

## 2.2.4 Các giải thuật xử lý ảnh

### 2.2.4.1 Nhận dạng mã QR

#### 2.2.4.1.1 Giới thiệu

**Mã QR** hay **mã hai chiều** là một mã vạch ma trận (hay mã vạch hai chiều, mã phản hồi nhanh) được phát triển bởi công ty Denso Wave (Nhật Bản) vào năm 1994. Chữ "QR" xuất phát từ "Quick Response", trong tiếng Anh có nghĩa là *đáp ứng nhanh*, vì người tạo ra nó có ý định cho phép mã được giải mã ở tốc độ cao. Các mã QR được sử dụng phổ biến nhất ở Nhật Bản, Trung Quốc, và hiện là loại mã hai chiều thông dụng nhất ở Nhật Bản.

Mặc dù lúc đầu mã QR được dùng để theo dõi các bộ phận trong sản xuất xe hơi, hiện nay nó được dùng trong quản lý kiểm kê ở nhiều ngành khác nhau. Gần đây hơn, phần mềm đọc mã QR đã được cài vào điện thoại di động có gắn camera (camera phone) ở Nhật. Điều này đưa đến các ứng dụng mới và đa dạng hướng về người tiêu dùng, nhằm làm nhẹ nhàng việc nhập dữ liệu vào điện thoại di động, vốn không hấp dẫn mấy. Mã QR cũng được thêm vào danh thiếp, làm đơn giản đi rất nhiều việc nhập dữ kiện cá nhân của người mới quen vào sổ địa chỉ trên điện thoại di động.

Người dùng có chương trình thu tín hiệu (capture program) và máy tính có giao diện RS-232C có thể dùng máy quét ảnh (scanner) để thu dữ liệu.

Tiêu chuẩn Nhật Bản cho các mã QR, JIS X 0510, được công bố vào tháng 1 năm 1999, và Tiêu chuẩn Quốc tế ISO tương ứng, ISO/IEC18004, được chấp thuận vào tháng 6 năm 2000

#### **2.2.4.2 Nhận dạng vật thể và mô hình Yolo**

YOLO - YOU ONLY LOOK ONCE là một trong những thuật toán nhận diện vật thể nhanh nhất thời điểm hiện tại. Mặc dù không phải là phương pháp có độ chính xác cao nhất tuy nhiên Yolo vẫn là được ứng dụng rất nhiều trong những dự án thực tế khi mà độ chính xác không phải là ưu tiên hàng đầu.

YOLO chia bức ảnh thành SxS grid, boundary box có 5 phần tử: x, y, w, h, confidence score.

YOLO có 3 phiên bản v1, v2, v3.

YOLO v1: sử dụng framework Darknet được train trên tập ImageNet-1000. Nó không thể tìm thấy các object nhỏ nếu chúng xuất hiện dưới dạng một cụm. Phiên bản này gặp khó khăn trong việc phát hiện các đối tượng nếu hình ảnh có kích thước khác với hình ảnh được train.

YOLO v2 đặt tên là YOLO9000 đã được Joseph Redmon và Ali Farhadi công bố vào cuối năm 2016. Cải tiến chính của phiên bản này tốt hơn, nhanh hơn, tiên tiến hơn để bắt kịp faster R-CNN (phương pháp sử dụng Region Proposal Network), xử lý được những vấn đề gặp phải của YOLO v1.

Sự thay đổi của YOLO v2 với YOLO v1:

- Batch Normalization: giảm sự thay đổi giá trị unit trong hidden layer, do đó sẽ cải thiện được tính ổn định của neural network.
- Higher Resolution Classifier: Kích thước đầu vào trong YOLO v2 được tăng từ 224\*224 lên 448\*448.
- Anchor boxes: dự đoán bounding box và được thiết kế cho tập dữ liệu đã cho sử dụng clustering.

- Fine-Grained Features: YOLO v2 chia ảnh thành 13\*13 grid cells, do đó có thể phát hiện được những object nhỏ hơn, đồng thời cũng hiệu quả với các object lớn.
- Multi-Scale Training: YOLO v1 có điểm yếu là phát hiện các đối tượng với các kích cỡ đầu vào khác nhau. Điều này được giải quyết bằng YOLO v2, nó được train với kích thước ảnh ngẫu nhiên trong khoảng 320\*320 đến 608\*608.
- Darknet 19: YOLO v2 sử dụng Darknet 19 với 19 convolutional layers, 5 max pooling layers và 1 softmax layer.

YOLO v3: phát hiện, phân loại chính xác đối tượng, và được xử lý thời gian thực.

Cải tiến của YOLO v3:

- Bounding Box Predictions: cung cấp score mỗi bounding boxes sử dụng logistic regression.
- Class Predictions: sử dụng logistic classifiers cho mọi class thay vì softmax.
- Feature Pyramid Networks(FPN)
- Darknet-53

### 2.2.5 Google Cloud Platform

**Google Cloud Platform**, được cung cấp bởi Google, là một bộ dịch vụ điện toán đám mây chạy trên cùng một cơ sở hạ tầng mà Google sử dụng nội bộ cho các sản phẩm của người dùng cuối, như Google Search và YouTube. Bên cạnh một bộ công cụ quản lý, nó cung cấp một loạt các dịch vụ đám mây mô-đun bao gồm điện toán, lưu trữ dữ liệu, phân tích dữ liệu và học máy.

Để tài sử dụng google cloud plateform chạy một máy ảo linux để thiết lập kết nối với thiết bị.

Nền tảng chính chạy trên cloud:

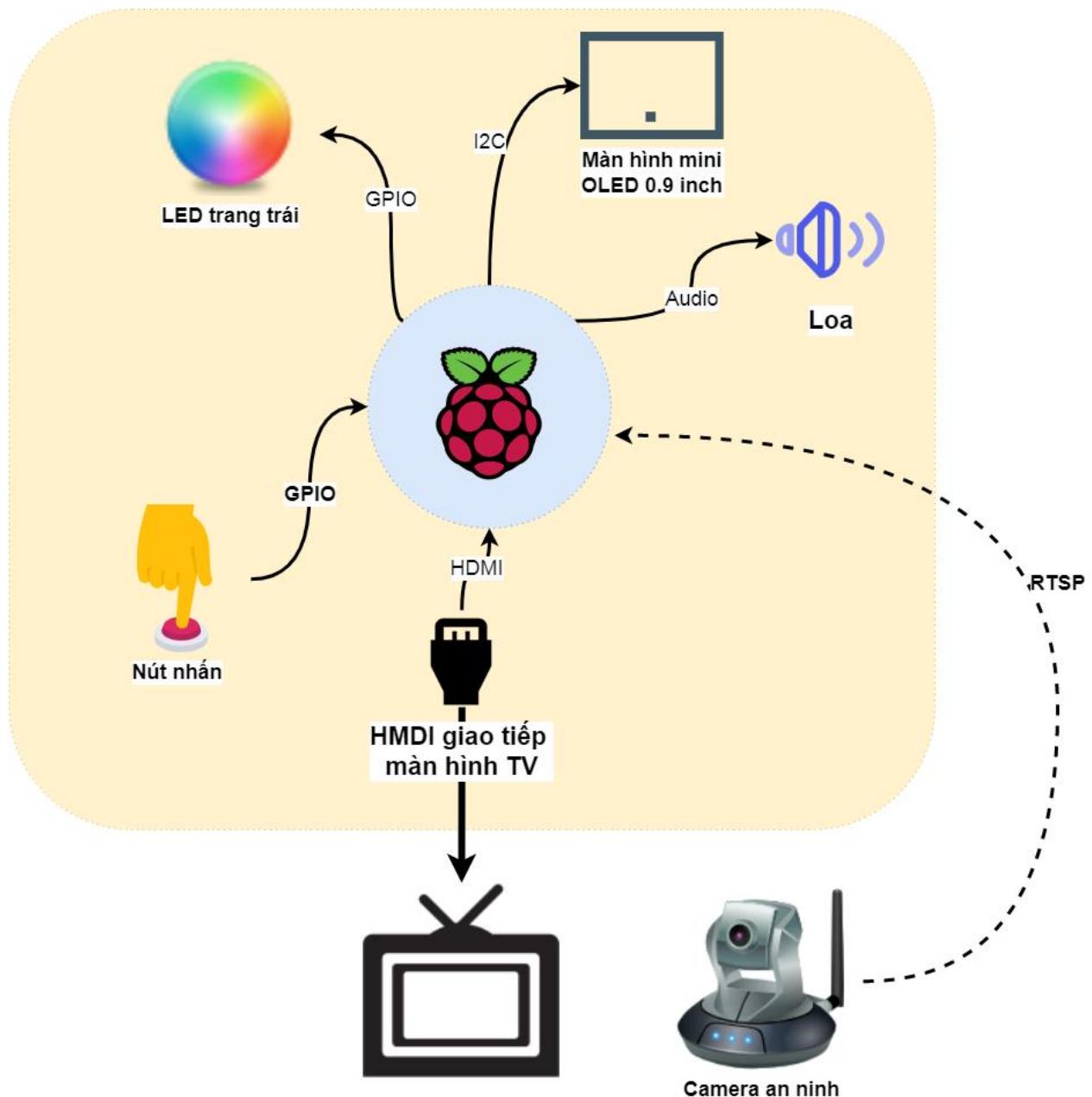
- Một máy ảo linux: Truy vấn trình duyệt web, thiết lập kết nối MQTT đến máy các thiết bị Homeserver
- Một hệ cơ sở dữ liệu MySQL lưu trữ các thông tin: đăng nhập, cấu hình mỗi nhà.

## Chương 3 THIẾT KẾ

### 3.1 THIẾT KẾ PHẦN CỨNG

#### 3.1.1 Home server

##### 3.1.1.1 Sơ đồ khối

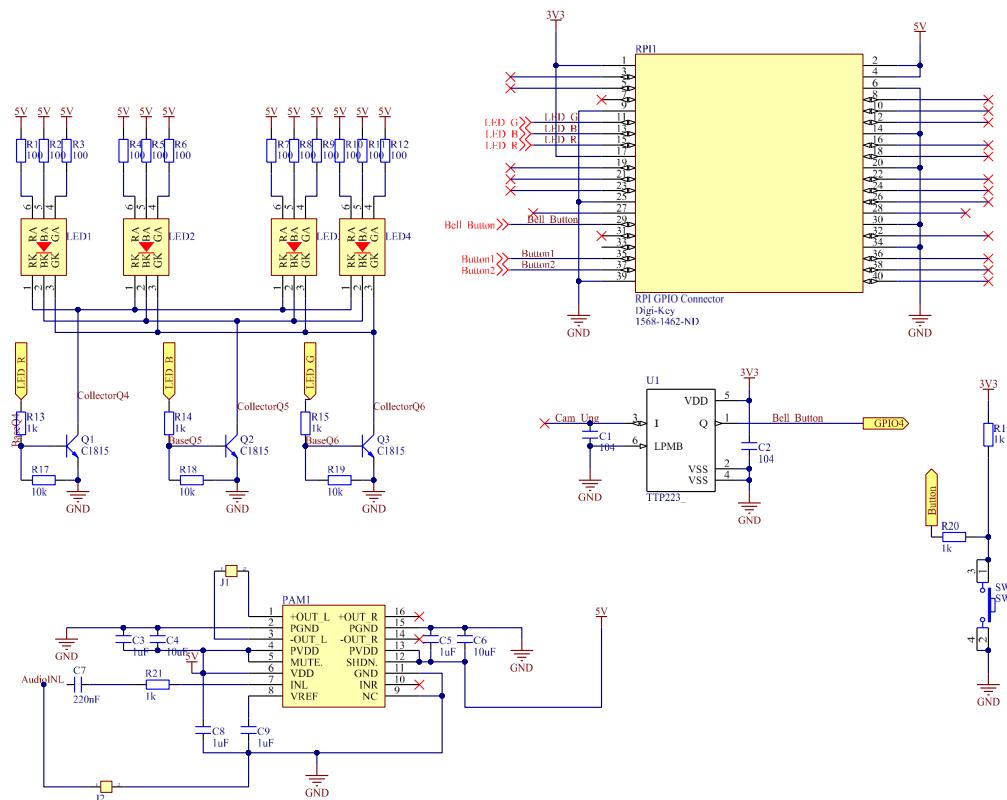


Hình 3-1 Sơ đồ mạch Home server

### **3.1.1.2 Danh sách linh kiện và module chính**

STT	Tên	Số lượng	Ghi chú
1	Module Raspberry pi 3	1	
2	Module Oled 0.9 inch	1	
3	Loa 4Ohm 3W	1	
4	IC khuếch đại âm thanh PAM8403	1	
5	IC cảm ứng điện dung TP223	1	
6	LED RGB 5050	4	
7	Transistor C1815 dán	3	
8	Nút nhấn	2	4 chân, 6mm

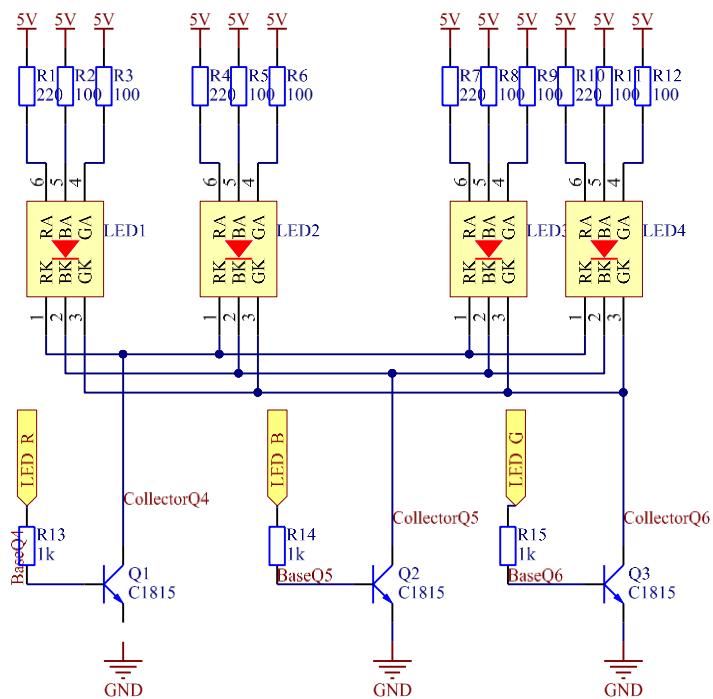
### **3.1.1.3 Sơ đồ nguyên lý**



**Hình 3-2 Sơ đồ nguyên lý mạch HomeServer**

### 3.1.1.4 Chi tiết các khối chức năng

#### 3.1.1.4.1 Khối đèn LED



Hình 3-3 Sơ đồ nguyên lý khôi đèn LED

**Nhiệm vụ:** Đèn tín hiệu và trang trí

Sử dụng LED 5050. LED được cấp nguồn 5V. Điều khiển gián tiếp qua mạch đệm bằng transistor C1815. Các LED RED, GREEN, BLUE dù nằm trên cùng linh kiện nhưng có thể được cấp nguồn độc lập mà không ảnh hưởng đến các LED còn lại.

Dựa theo datasheet LED5050 của hãng: [5]

Ta có các thông số:

Loại LED	Áp roi trên LED Vf	Dòng hoạt động If
Red	1.8V – 2.4V	20mA
Green	2.8V – 3.6V	20mA
Blue	2.8V – 3.6V	20mA

Parameter	Symbol	Color	Min	Typ	Max	Unit	Tolerance	Test Condition
Forward Voltage	Vf	R	1.80	---	2.40	V	$\pm 0.05V$	IF forward current=20mA Test Temperature=25°C
		G	2.80	---	3.60			
		B	2.80	---	3.60			
Luminous Intensity	IV	R	100	---	---	mcd	$\pm 10 mcd$	
		G	400	---	---			
		B	100	---	---			
Dominant Wavelength	$\lambda_d$	R	620	---	630	nm	$\pm 2nm$	
		G	515	---	530			
		B	460	---	475			
Lighting Angle	$\theta$	/	115	120	125	deg	$\pm 2$	
Reverse Current	IR	/	---	---	10	$\mu A$	$\pm 0.1 \mu A$	Vr=5V

Hình 3-4 Bảng gốc Datasheet LED 5050

Sử dụng Q1,Q2,Q3=C1815. Theo datasheet  $\beta=100$ ,  $VCE=0.2V$ ,  $VBE=0.7V$ .

Với nguồn cấp 5V. Tính toán phân cực LED ta được

LED Green và Blue:

$$R = (Vcc - Vf - VCE) / If = (5V - Vf - 0.2) / 20mA$$

Với điện áp  $Vf$  từ 2.8V đến 3.6V, theo công thức trên giá trị điện trở trong khoảng từ 70 Ohm đến 110 Ohm. Chọn giá trị 100 Ohm để phân cực cho LED khi đó dòng qua LED có giá trị trong khoảng từ 14mA đến 22mA.

Tức là các điện trở R2, R3, R5, R6, R8, R9, R11, R12 có giá trị 100Ohm.

LED Red:

$$R = (Vcc - Vf - VCE) / If = (5V - Vf - 0.2) / 20mA$$

Với điện áp  $Vf$  từ 1.8V đến 2.4V, theo công thức trên giá trị điện trở trong khoảng từ 130 Ohm đến 160 Ohm. Tuy nhiên do độ sáng LED Red lớn khá lớn chênh lệnh với 2 LED còn lại nên thiết kế chọn giá trị điện trở 220 Ohm khi đó dòng qua LED vào khoảng 11mA đến 14mA.

Tức là các điện trở R1, R4, R7, R10 có giá trị 220Ohm.

Sử dụng Q1,Q2,Q3=C1815. Theo datasheet  $\beta=100$ ,  $VCE=0.2V$ ,  $VBE=0.7V$ .

$I_c$  của mỗi transistor sẽ là:  $I_c = 20\text{mA} \times 4 = 80\text{mA}$ .

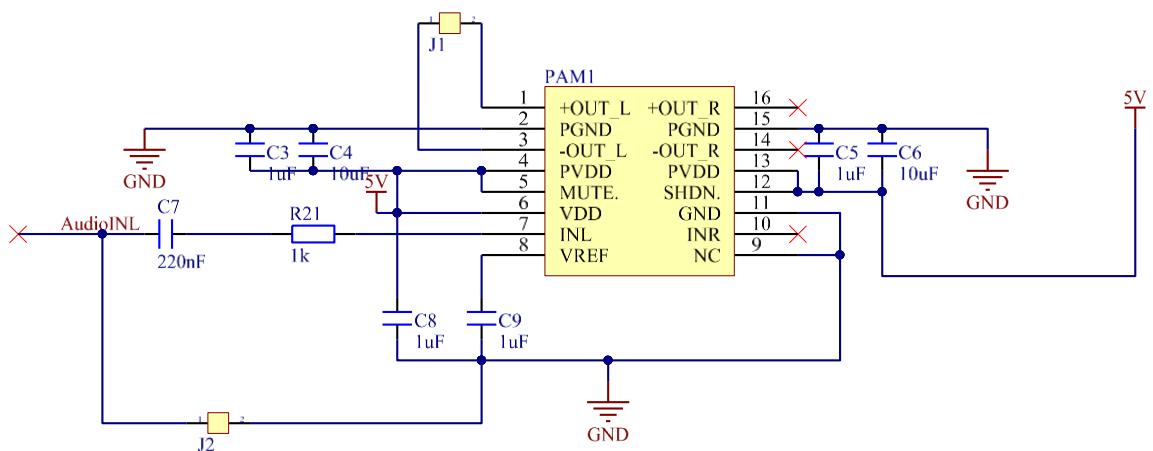
Để transistor bão hòa  $I_b > I_c/100 = 0.8\text{mA}$ . Chọn  $I_b = 2\text{mA}$

$$R = (3.3\text{V} - 0.7\text{V})/2\text{mA} = 1.3\text{kOhm}$$

Chọn điện trở  $1\text{kOhm}$  thì  $I_b = 2.6\text{mA}$  đạt yêu cầu.

### 3.1.1.4.2 Khối âm thanh

**Nhiệm vụ:** Khuếch đại âm thanh từ cổng Audio out của Raspberry Pi 3.



Hình 3-5 Sơ đồ nguyên lý mạch khuếch đại âm thanh

IC PAM8403 đảm nhiệm chức năng chính để khuếch đại âm thanh từ Raspberry. Phần Audio IN được cấp trực tiếp từ Jack 3.5mm ở phần Audio OUT của Raspberry Pi.

Sơ đồ mắc tương tự với sơ đồ mắc tham khảo của nhà sản xuất gồm các tụ lọc tín hiệu nguồn vào ( $C_3, C_4, C_5, C_6$ ), các tụ ngăn điện áp DC ( $C_7$ ). Sử dụng nguồn 5V để cấp cho IC. Mạch được kết nối với một loa 4Ohm 3W ở kênh L. Kênh R không được sử dụng.

Pin Number	Pin Name	Description
1	+OUT_L	Left Channel Positive Output
2	P GND	Power GND
3	-OUT_L	Left Channel Negative Output
4	P VDD	Power VDD
5	MUTE	Mute Control Input (active low)
6	V D D	Analog VDD
7	I N L	Left Channel Input
8	V R E F	Internal analog reference, connect a bypass capacitor from VREF to GND
9	N C	No connect
10	I N R	Right Channel Input
11	G N D	Analog GND
12	S H D N	Shutdown Control Input (active low)
13	P VDD	Power VDD
14	-OUT_R	Right Channel Negative Output
15	P G ND	Power GND
16	+OUT_R	Right Channel Positive Output

**Hình 3-6 Sơ đồ chân PAM8403****Các thông số quan trọng:**

Điện áp hoạt động: 2.5V-5.5V

Độ lợi: Gv=24dB

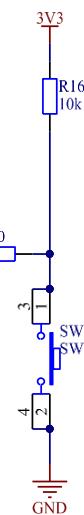
Công suất ngõ ra: 2.5W với điều kiện. trở kháng Loa  $R_L=4\text{Ohm}$ , tần số 1kHz, THD+N=1%.

Hiệu suất: 83%

Hệ số tín hiệu trên nhiễu SNR: 80dB

**3.1.1.4.3 Khối nút nhấn****Nhiệm vụ:** Giao tiếp nút nhấn với thiết bị

Loại nút sử dụng. Nút dạng cắm, 4 chân, 6mm. Mạch gồm 2 nút nhấn chức năng.

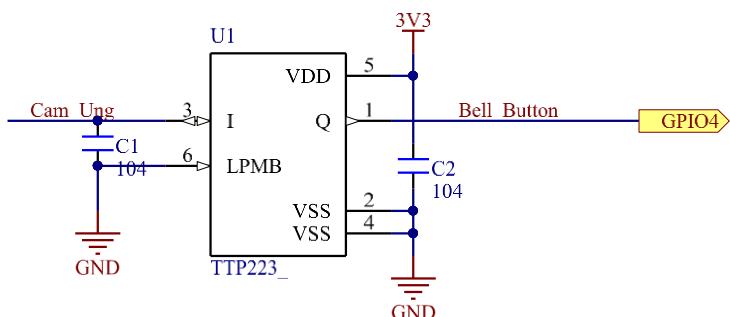


Hình 3.7 Sơ đồ nguyên lý khôi nút nhấn mạch HomeServer

Sử dụng điện trở kéo lên 10k. Điện áp hoạt động 3.3V.

Theo nguyên lý trên, chân I/O sẽ ở mức cao ở trạng thái bình thường và ở mức thấp khi nhấn nút. Khi đó ở mức cao dòng đổ vào chân I/O sẽ là  $3.3V/(11k) = 0.3mA$  đủ để mạch nhận tín hiệu mức cao.

#### 3.1.1.4.4 Khối nút nhấn cảm ứng



Hình 3.8 Sơ đồ nguyên lý khôi nút nhấn cảm ứng

Sử dụng IC chức năng TTP223. Dựa vào Datasheet TTP223:

Sơ đồ chân IC tham khảo bảng sau:

Chân 1: Ngõ ra
Chân 2: Đất
Chân 3: Ngõ vào touchpad
Chân 4: Chân cấu hình AHLB (mức cao/thấp)
Chân 5: Chân nguồn 3.3V
Chân 6: Chân cấu hình TOG (mode direct/toggle)

Tùy vào cách thiết kế có thể cấu hình IC theo các mode:

TOG	AHLB	Q
0	0	1 → khi nút nhấn mức cao
0	1	1 → nút nhấn mức thấp
1	0	1 → Kích cạch xuống
0	1	1 → Kích cạch lên

Dựa vào thiết kế trên chân I/O sẽ ở mức cao khi chạm vào phần cảm ứng. Dòng cung cấp của mạch đến chân I/O vào khoảng 8mA

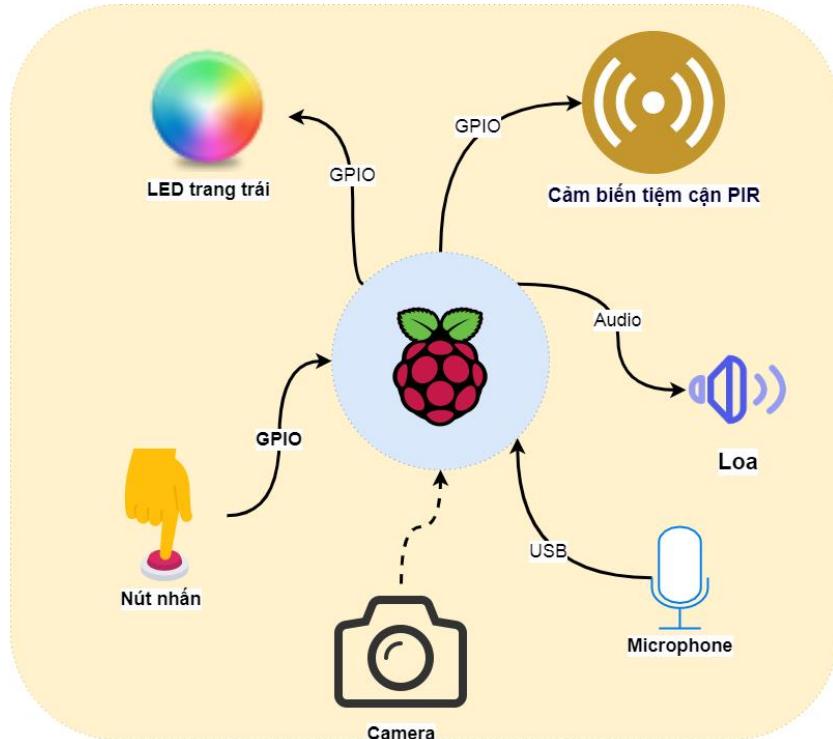
### 3.1.1.4.5 Mở rộng kết nối với module

Ngoài ra ở chân 3-5-7 và GND của module được mở rộng kết nối với Header để giao tiếp mới một màn hình mini OLED 0.9inch

Phản giao tiếp với màn hình được thực hiện qua cổng HDMI tích hợp sẵn trên module

## 3.1.2 Camera chuông cửa

### 3.1.2.1 Sơ đồ khái niệm



Hình 3-9 Sơ đồ khái niệm thiết bị camera chuông cửa

### 3.1.2.2 Danh sách linh kiện và module chính

STT	Tên	Số lượng	Ghi chú
1	Module Raspberry pi 2	1	
2	Module cảm biến tiệm cận PIR	1	
3	Loa 4Ohm 3W	1	
4	IC khuếch đại âm thanh PAM8403	1	
5	IC cảm ứng điện dung TP223	1	
6	LED RGB 5050	4	
7	Transistor C1815 dán	3	
8	Nút nhấn	2	4 chân, 6mm
9	Thiết bị Microphone kết nối USB	1	

### 3.1.2.3 Chi tiết các khối chức năng:

Các khối chức năng hoàn toàn giống mạch HomeServer.

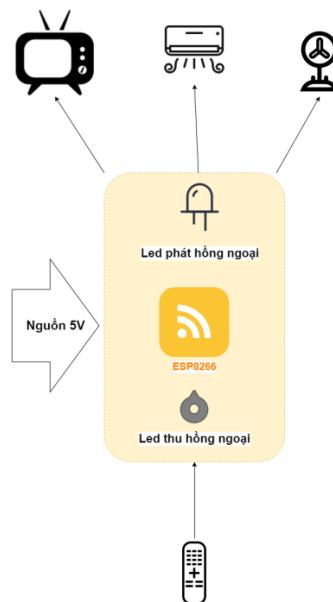
Phần mở rộng kết nối có thêm header nối với chân GPIO mở rộng tín năng kết nối với module cảm biến tiệm cận PIR. Đồng thời cũng được tích hợp thêm USB Microphone để thu âm thanh.



Hình 3-10 Module cảm biến PIR

### 3.1.3 Điều khiển thiết bị hồng ngoại

#### 3.1.3.1 Sơ đồ khối



Hình 3-11 Sơ đồ khối thiết bị điều khiển hồng ngoại

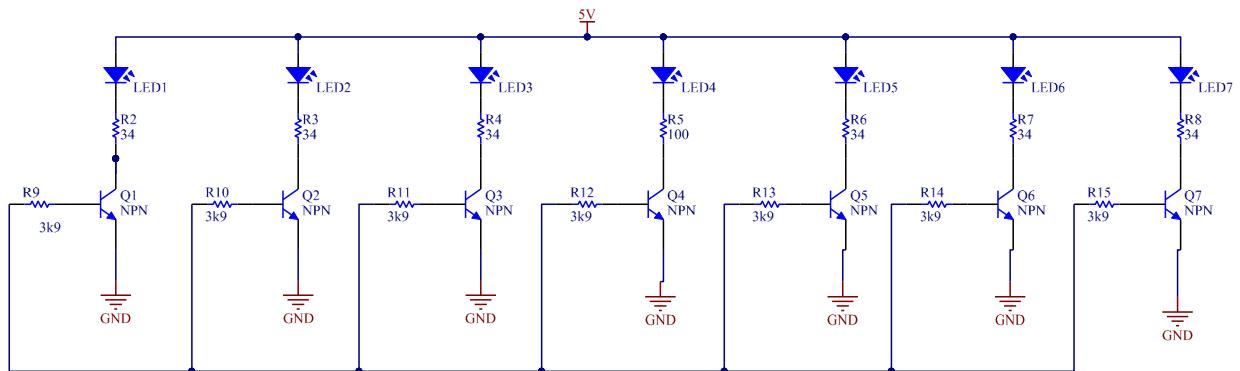
### 3.1.3.2 Danh sách linh kiện và module

STT	Tên	Số lượng	Thông số	Ghi chú
1	Module ESP8266	1	ESP8266 12E	
2	Transistor C1815 dán	6		
3	LED phát hồng ngoại	6	5mm	
4	LED thu hồng ngoại	1	TSOP1838	
5	Nút nhấn	1	2 chân, 6mm	
6	IC ổn áp AMS1117	1	3.3V	
7	Điện trở dán 1k			
8	Điện trở dán			
9	Điện trở dán			
10	Tụ điện			
11	Tụ điện			

### 3.1.3.3 Sơ đồ nguyên lý

#### 3.1.3.3.1 Khối phát hồng ngoại

Gồm 7 LED phát hồng ngoại kích thước 5mm mắc qua bộ đệm bằng transistor C1815 và được nối chung đến chân GPIO output của module ESP8266. Chân GPIO này khi hoạt động sẽ tạo ra một chùm xung điện áp bật/tắt các LED ở tần số 38kHz để tạo ra tín hiệu hồng ngoại phát đến các thiết bị điện tử.



Hình 3-12 Khối LED phát hồng ngoại

Tham khảo data sheet IR led IR333-A:

Điện áp thuận  $V_f = 1.6V$

Dòng định  $I_p = 100mA$

Dòng trung bình khi hoạt động  $I_{tb} = 30mA$

Cường độ dòng bức xạ  $I_e=10\text{mW/sr}$   $IF=10\text{mA}$  và  $I_e=100\text{ mW/sr}$   $IF=100\text{mA}$ .

Tham khảo một số dòng Remote trên thị trường (Trên source fource LIRC)

### **LG-5988**

```

name  Euroconsumers_LG-5988
bits      16
flags SPACE_ENC|CONST_LENGTH
eps       30
aeps      100

header     8991  4538
one        541   1713
zero       541   585
ptrail     543
pre_data_bits 16
pre_data    0x11EE
gap         108238
toggle_bit_mask 0x0

```

### **Panasonic DVD**

```

name  DVD
bits      17
flags SPACE_ENC
eps       20
aeps      200

header     4000  1600
one        400   1200
zero       400   400
ptrail     400
pre_data_bits 31
pre_data    0x20020680
gap         76000
min_repeat 4
toggle_bit  0

```

### **Sanyo**

```

name  Sanyo
bits      16
flags SPACE_ENC|CONST_LENGTH
eps       30
aeps      100
header     9000  4500
one        563   1687
zero       563   562
ptrail     563
pre_data_bits 26
gap         108000

repeat     9000  4500

```

Rút ra được nhận xét chung về chu kỳ hoạt động

- Tùy vào lượng bit truyền đi mà thời gian LED phải ở mức cao khác nhau.
- Ngoài khói Header cần thời gian truyền mức cao lâu còn lại hầu như là thời gian truyền mức cao luôn bé hơn mức thấp.

Chẳng hạn:

- + Bit 1, mức cao ít hơn mức thấp 3 lần.
- + Bit 2, mức thấp bằng với mức cao.

Ước lượng chu kỳ nhiệm vụ tối đa  $\frac{1}{2}$  chu kỳ truyền.

Với sóng mang đối xứng thì dòng trung bình và dòng đỉnh  $I_p=100mA$

**$I_{tb} = \frac{1}{2} * \frac{1}{2} * 100mA = 25mA$ . (Với sóng mang đối xứng  $\frac{1}{2}$  và chu kỳ nhiệm vụ ước lượng  $\frac{1}{2}$ )**

Với  $I_p = 100mA$ . Transistor Q là C1815.  $h_{fe}=100$ ,  $V_{CEQ}=0.7$ ,  $V_{BEQ}=0.2$

Điện trở phân cực LED:

$$R = (5 - V_{ce} - V_f) / I_p = (5 - 0.7 - 1.6) / 100mA = 27\Omega. Chọn điện trở 34\Omega.$$

Với  $I_{tb}=30mA=I_c$ . Để transistor bão hòa  $I_b > 30/100=0.3mA$ . Chọn  $I_b = 0.8mA$

Do đó điện trở phân cực transistor:

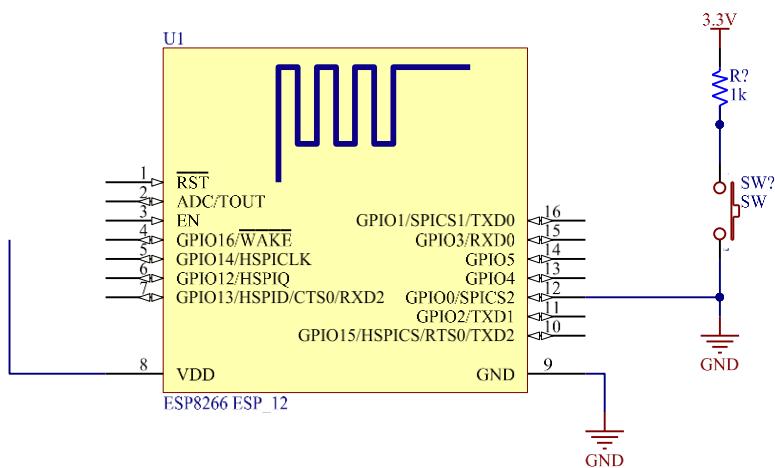
$$R = (3.3V - V_{be}) / 0.3mA = (3.3 - 0.2) / 0.8mA = 3.875K\Omega$$

Chọn điện trở 3k9.

Khi đó chân GPIO của ESP8266 phải chịu dòng  $0.8*7 = 5.6mA$ .

### 3.1.3.3.2 Khối xử lý trung tâm và nút nhấn

Khối xử lý dùng ESP8266 để giao tiếp mạng Wifi và sử dụng giao tiếp GPIO để thực hiện các chức năng khác của mạch



**Hình 3-13 Sơ đồ khối xử lý trung tâm và nút nhấn mạch điều khiển hồng ngoại**

ESP8266 có các chế độ hoạt động và boot khác nhau. Tham khảo bảng sau đây

<b>IO0</b>	<b>IO15</b>	<b>IO2</b>	<b>Chế độ</b>
Thấp	Thấp	Cao	Lập trình bằng cổng Serial
Cao	Thấp	Cao	Boot chương trình từ bộ nhớ Flash
X	Cao	X	Boot từ thẻ SD

Mạch sử dụng duy nhất một nút nhấn cơ (dạng cảm hai chân) mắc với điện trở pull up 10k. Nhiệm vụ chính của nút nhấn này chuyển trạng mạch và nạp chương cho mạch. Với việc kết nối vào chân IO0 chỉ cần nhấn nút để chân IO0 xuống mức thấp, để treo hai chân IO15 và IO2 để mặc định mức thấp và cao của từng chân. Khởi động nguồn và kết nối UART là có thể lập trình lại được module.

### 3.1.3.3.3 Khối thu hồng ngoại

Sử dụng duy nhất một LED thu hồng ngoại TSOP 1838 để thu tín hiệu hồng ngoại phát đến mạch. Việc giải mã thông tin truyền đến sẽ được thực hiện bằng giải thuật phần mềm. Chân tín hiệu của LED thu sẽ thay đổi điện áp khi có tín hiệu hồng ngoại đến. Điện áp sẽ càng giảm khi cường độ của chùm hồng ngoại càng tăng từ đó có thể thu được xung tín hiệu chuyển từ dạng truyền bằng hồng ngoại sang tín hiệu điện áp để giải mã.

LED thu được cấp nguồn 3.3V để kết nối với chân GPIO input của module ESP8266



Hình 3-14 LED thu hồng ngoại TSOP1838

Dựa vào datasheet TSOP 1838:

Với điện áp hoạt động bình thường là 5V thì dòng tiêu thụ tối đa 1.5mA

### 3.1.3.3.4 Khối nguồn

Mạch sử dụng 2 mức điện áp chính: 5V và 3.3V.

Mức điện áp 5V: Cấp nguồn cho các bóng đèn LED hồng ngoại hoạt động phát tín hiệu hồng ngoại. Nguồn 5V được cấp từ Adapter 5V/2.5A.

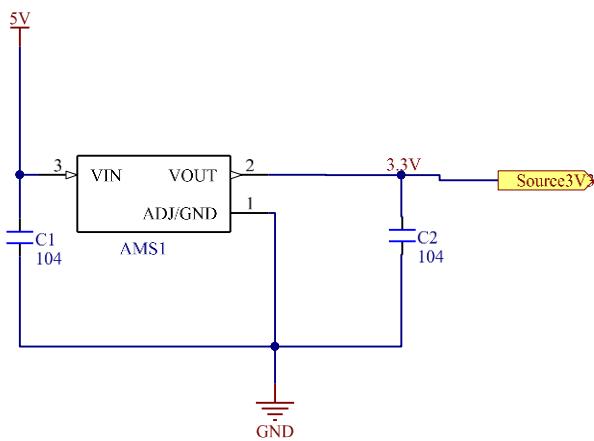


Hình 3-15 Adapter 5V

Mức điện áp 3.3V: Cấp nguồn hoạt động cho ESP8266, khói led thu hồng ngoại và khói nút nhấn. Nguồn 3.3V được cấp từ nguồn 5V đi qua mạch ổn áp dùng AMS1117.

Sử dụng IC ổn áp AMS1117 – 3.3 đảm nhiệm chức năng chỉnh để biến đổi điện áp 5V thành 3.3V. AMS1117-3.3 là IC chuyên dụng để ổn định điện áp tuyến tính với sụt áp thấp, điện áp đầu ra 3.3V, dòng điện đầu ra 1A. Nguồn cấp đầu vào của IC có thể lên

đến 11V. Mạch có sơ đồ mắc đơn giản chỉ gồm các tụ lọc đầu vào và đầu ra. Cụ thể theo sơ đồ sau:



Hình 3-16 Sơ đồ nguyên lý mạch ổn áp dùng AMS1117

Đầu ra của khối nguồn này được cấp trực tiếp đến chân cấp nguồn của module ESP8266. IC AMS1117 cũng là IC ổn áp được sử dụng rộng rãi ở các bo mạch Wemos, Node MCU,.. và các thiết bị trên thị trường như của Tuya SmartHome để cấp nguồn cho ESP8266.

Dựa vào tính toán công suất các khối có thể tính toán tương tối nguồn Adapter cần cấp cho thiết bị là:

**Khối LED phát:** Cấp nguồn cho LED với dòng trung bình tối đa:

$$30\text{mA} \times 7 = 210\text{mA}$$

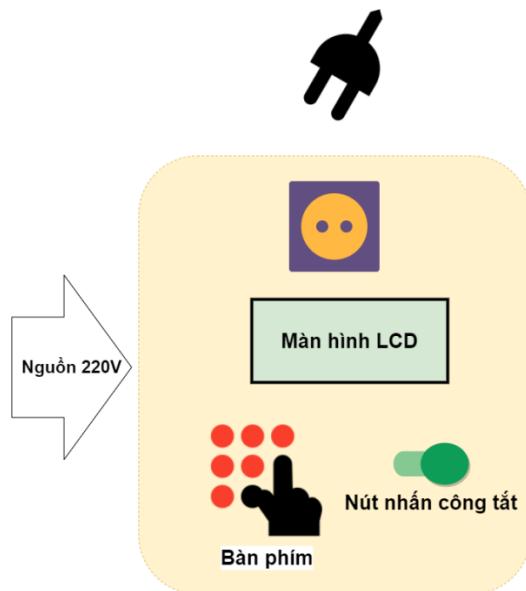
**Khối LED thu:** 1.5mA

**Khối ổn áp cấp nguồn tiêu thụ cho ESP8266:** tối đa 1A

Do đó sử dụng Adapter nguồn 5V/ 2.5A là phù hợp

### 3.1.4 Điều khiển Relay và LCD

#### 3.1.4.1 Sơ đồ khối



Hình 3-17 Sơ đồ mạch điều khiển Relay và LCD

#### 3.1.4.2 Danh sách linh kiện và module

STT	Tên	Số lượng	Thông số	Ghi chú
1	Module ESP8266	1	ESP8266 12E	
2	Transistor C1815	6		
3	PIC16F877A	1		
4	Opto PC817	3		
5	Nút nhấn cảm	3	4 chân, 6mm	
6	IC ống áp AMS1117	1	3.3V	
7	IC buck LM2596	1	5V	
8	Relay 12V	3		
9	Cầu chì	1		
10	Chỉnh lưu cầu 8A	1		
11	Biến áp 220V -12V	1		

#### 3.1.4.3 Sơ đồ nguyên lý

##### 3.1.4.3.1 Khối nguồn

**Chức năng:**

Từ nguồn cấp 220V của lưới điện. Chuyển thành các mức điện áp:

+ Nguồn một chiều 12VDC: Nguồn điều khiển Relay

+ Nguồn một chiều 5VDC: Nguồn cấp cho vi điều khiển PIC16F877A, các thiết bị LCD

+ Nguồn một chiều 3.3V: Nguồn cấp cho vi điều khiển ESP8266

### Cấu tạo chính của khối:

+ Mạch biến áp

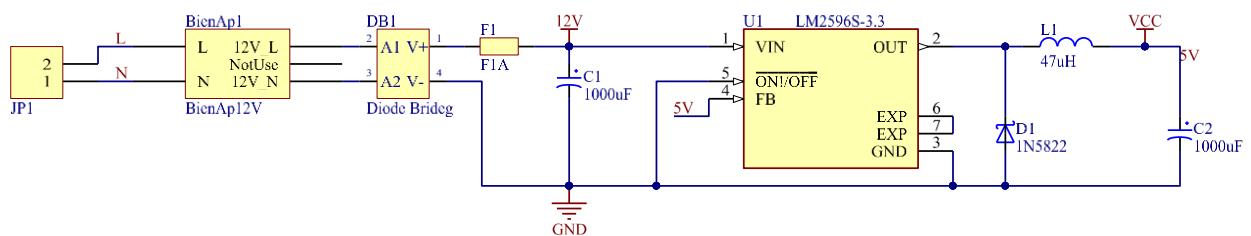
+ Mạch chỉnh lưu cầu

+ Các mạch lọc

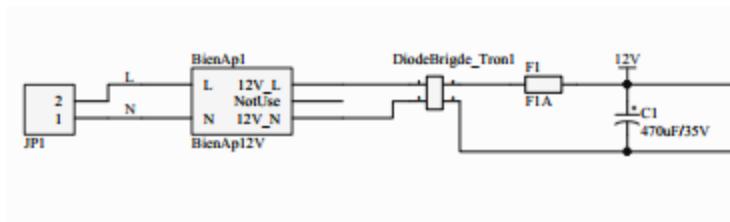
+ Mạch bảo vệ dùng cầu chì

+ Mạch Buck 12VDC xuống 5VDC

+ Mạch ổn áp 3.3V



### Chi tiết:

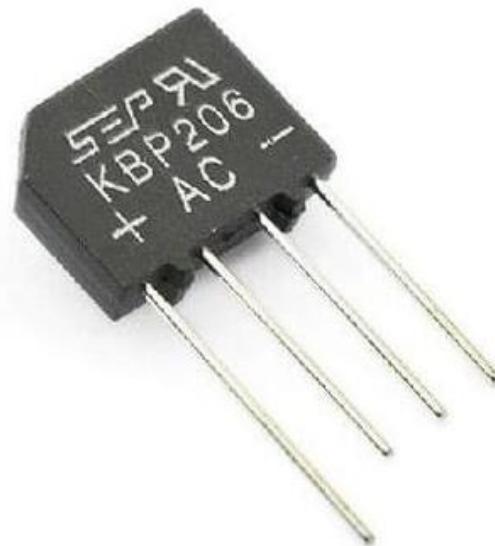


Mạch hạ áp qua biến áp 220VAC – 12VAC.



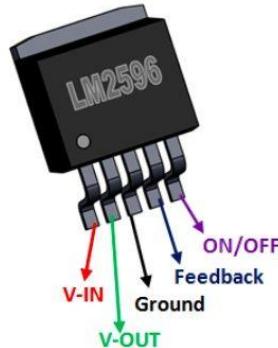
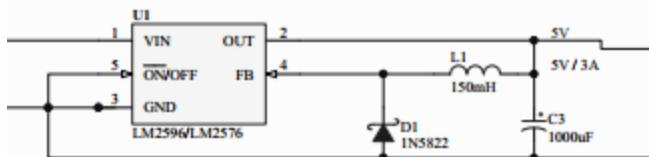
**Hình 3-18 Biến áp 12V**

Nguồn 12VAC sau đó được chuyển qua bộ chỉnh lưu cầu 8A cho ra nguồn 12VDC sau đó qua cầu chì bảo vệ và tụ lọc phẳng điện áp



**Hình 3-19 Chỉnh lưu cầu**

Nguồn 12VDC sau đó qua mạch Buck Converter sử dụng IC LM2596 5.0V cho ra điện áp nguồn 5V

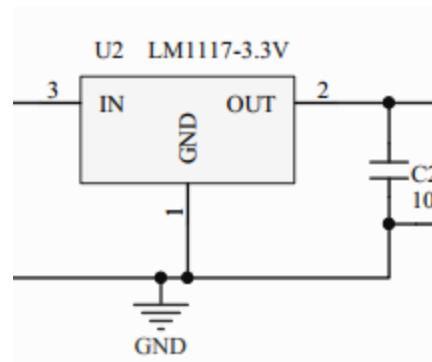


Hình 3-20 Sơ đồ chân LM2596

Cuộn cảm được chọn là cuộn xuyến có giá trị 47uH. Cho phép điện áp đầu vào đến 40V. Tụ 1000uF lọc phẳng dạng sóng

Mạch LM1117 3V3 làm hạ điện áp xuống 3.3V cấp nguồn cho ESP8266.

Do chỉ có mỗi mạch ESP8266 sử dụng nguồn 3.3V nên với công suất đầu ra của LM1117 thì phù hợp



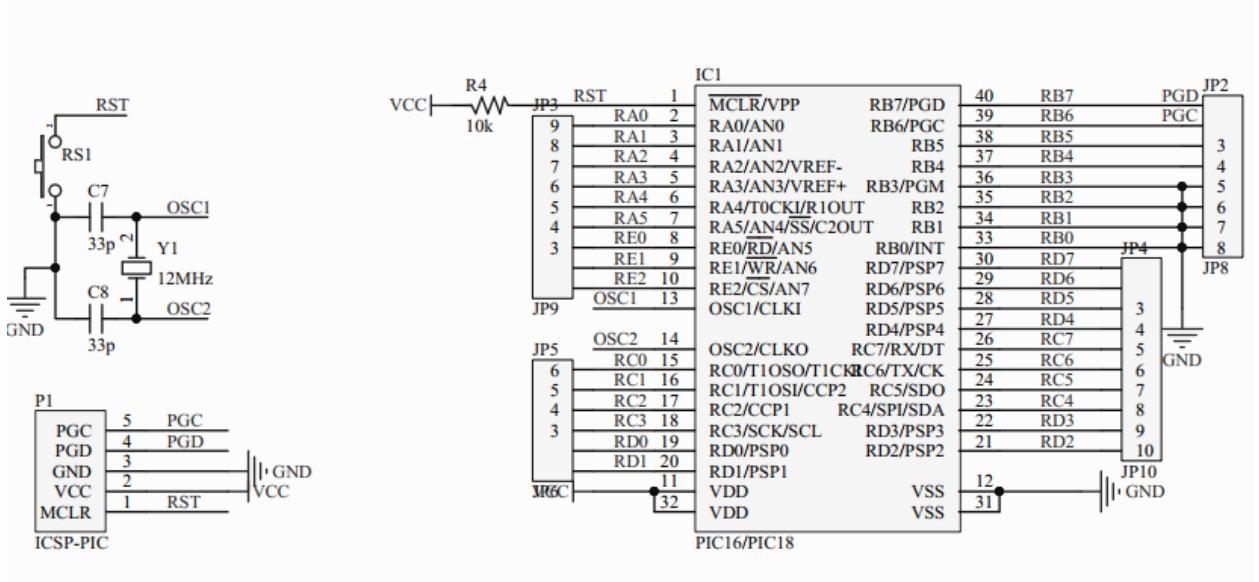
### 3.1.4.3.2 Khối vi điều khiển PIC16F877A

#### Chức năng:

- + Mạch điều khiển các ngoại vi LCD, nút nhấn, giao tiếp với ESP8266
- + Các giao tiếp sử dụng

### Cấu tạo chính:

- + Mạch biến áp
- + Mạch chỉnh lưu cầu
- + Các mạch lọc
- + Mạch bảo vệ dùng cầu chì
- + Mạch Buck 12VDC xuống 5VDC
- + Mạch ổn áp 3.3V



Mạch cấp nguồn 5V.

Các chân nạp ICSP được nối với header để nạp chương trình cho vi điều khiển

Dùng thạch anh 12Mhz kết hợp tụ 33pF tạo dao động mẫu

Nút nhấn reset

Các chân Port D được dùng kết nối đến LCD 1602

Port C dùng giao tiếp UART với ESP8266 nên chỉ dùng được 2 chân TX và RX

Port B sử dụng giao tiếp với Keypad 4x3 với điện trở pulldown 10kOhm

Port A sử dụng kết nối với Relay và giao tiếp với nút nhấn

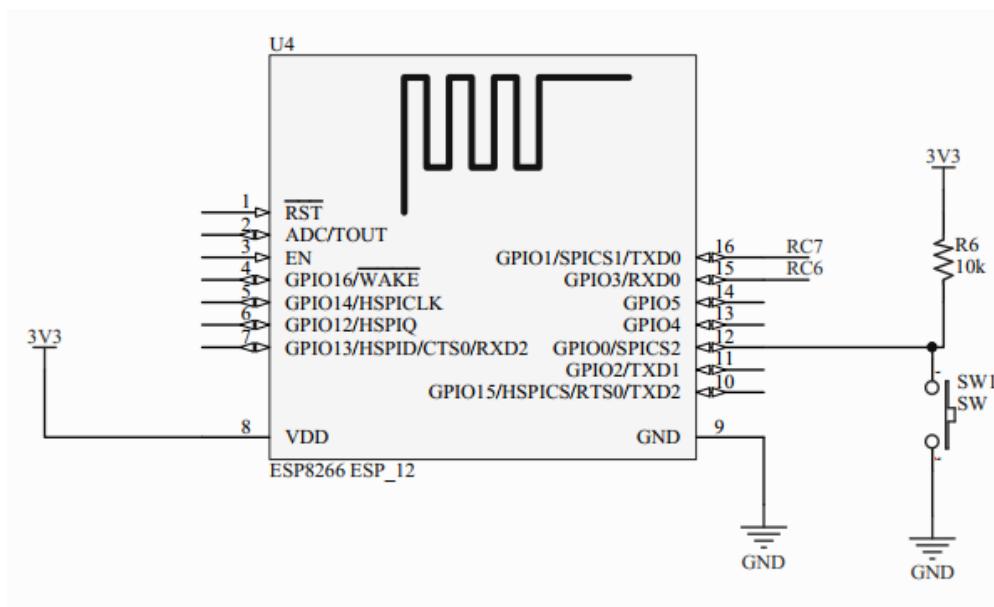
### 3.1.4.3.3 Khối module ESP8266

Cáp nguồn 3.3V giao tiếp UART với PIC. Chân GPIO 0 được kết nối với nút nhấn với điện trở 10kOhm pullup. Chân GPIO0 ở mức cao ở trạng thái bình thường và sẽ ở mức thấp khi nối đất. Mục đích của nút nhấn là thiết lập một số cấu hình cần thiết cho ESP8266 và vị trí nút nhấn ở chân GPIO0 giúp tiện lợi hơn khi nạp chương trình cho ESP8266.

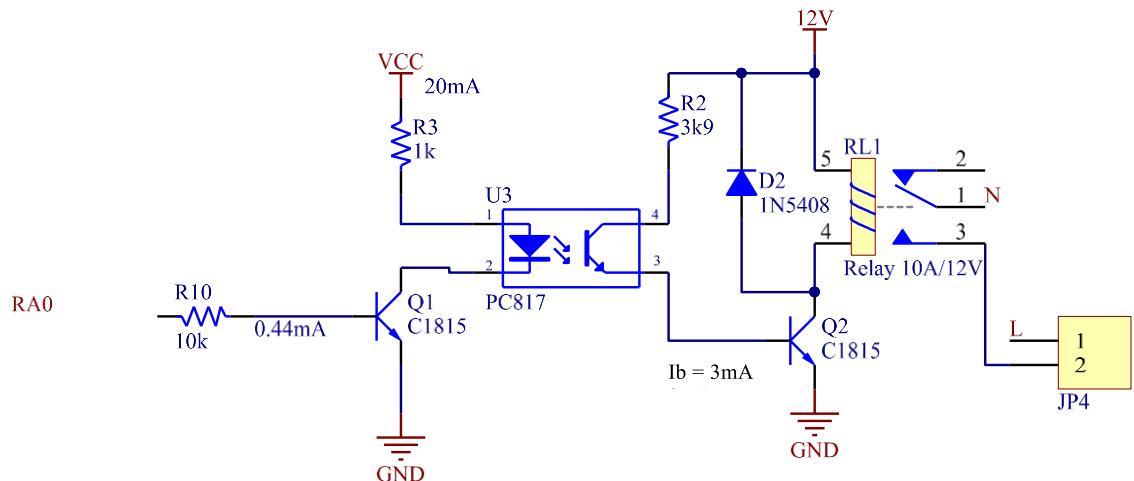
Chế độ boot của ESP8266 được thể hiện qua bảng sau:

<b>IO0</b>	<b>IO15</b>	<b>IO2</b>	<b>Chế độ</b>
Thấp	Thấp	Cao	Lập trình bằng cổng Serial
Cao	Thấp	Cao	Boot chương trình từ bộ nhớ Flash
X	Cao	X	Boot từ thẻ SD

Chân RC6 và RC7 của PIC được nối trực tiếp với TX và RX của ESP8266 để ‘giao tiếp’ UART.



### 3.1.4.3.4 Khối opto và relay



Hình 3-21 Khối Opto và Relay

Khối sử dụng relay 12V. Được nối trung gian qua Opto với 2 bộ khuếch đại sử dụng transistor NPN C1815.

Chọn relay 12VDC/80mA

IC2=80mA

Q2=C1815  $\beta=100$ ,  $VCE2=0.2V$ ,  $VBE2=0.7V$

Theo data sheet optocouple PC817 :

Hệ số truyền đạt :  $IC/IF=0.5 - 6$

Áp rơi trên diode(VF)=1.2 – 1.4V

Áp bão hòa BJT( $VCE$ )=0.1 – 0.2V

Chọn  $IC/IF=1$ ,  $VF=1.4V$ ,  $VCE=0.2V$

Q2 bão hòa:  $IB2 > IC2/\beta = 80/100 = 0.8mA$

Chọn  $IB2=3mA \rightarrow R2=(12-VCE_{opto}-VBE2)/IB2=(12-0.2-0.7)/3=3.7K$

Chọn **R2=3.9K** →  $IB2=(12-0.2-0.7)/3.9=2.85mA$  đạt yêu cầu

→  $IC1= IB2 (IC=IF opto.)=3mA$

$Q1=C1815 \beta=100$ ,  $VCE1=0.2V$ ,  $VBE1=0.7V$

$$R3 = (Vcc - VF - VCE1) / IC1 = (5 - 1.4 - 0.2) / 3 = 1.13K$$

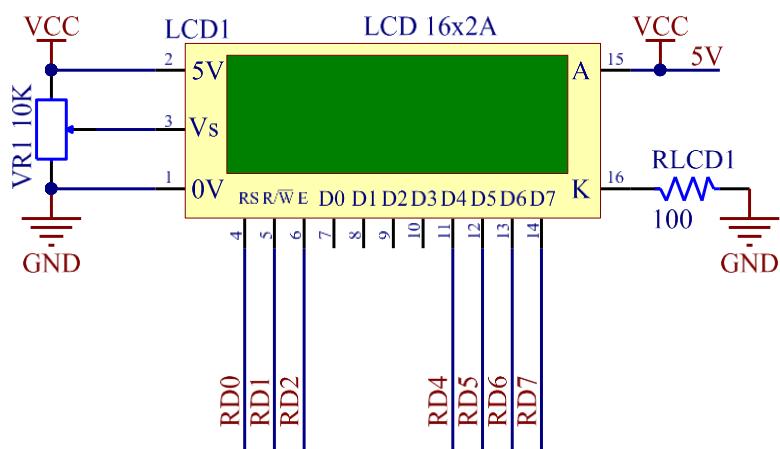
Chọn **R3=1K**

Q1 bão hòa:  $IB1 > IC1 / \beta = 3/100 = 0.03mA$

Chọn  $R5=10K \rightarrow IB1 = (Vi - VBE1) / R5 = (5 - 0.7) / 10 = 0.43mA$  đạt yêu cầu

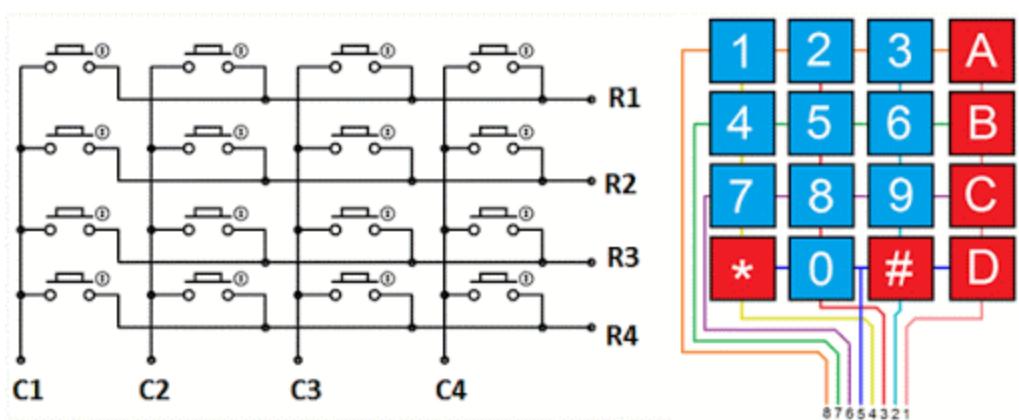
### 3.1.4.3.5 Khối LCD

Khối LCD kết nối trực tiếp đến Port D của PIC



Hình 3-22 Khối LCD

### 3.1.4.3.6 Khối keypad



Hình 3-23 Khối bàn phím

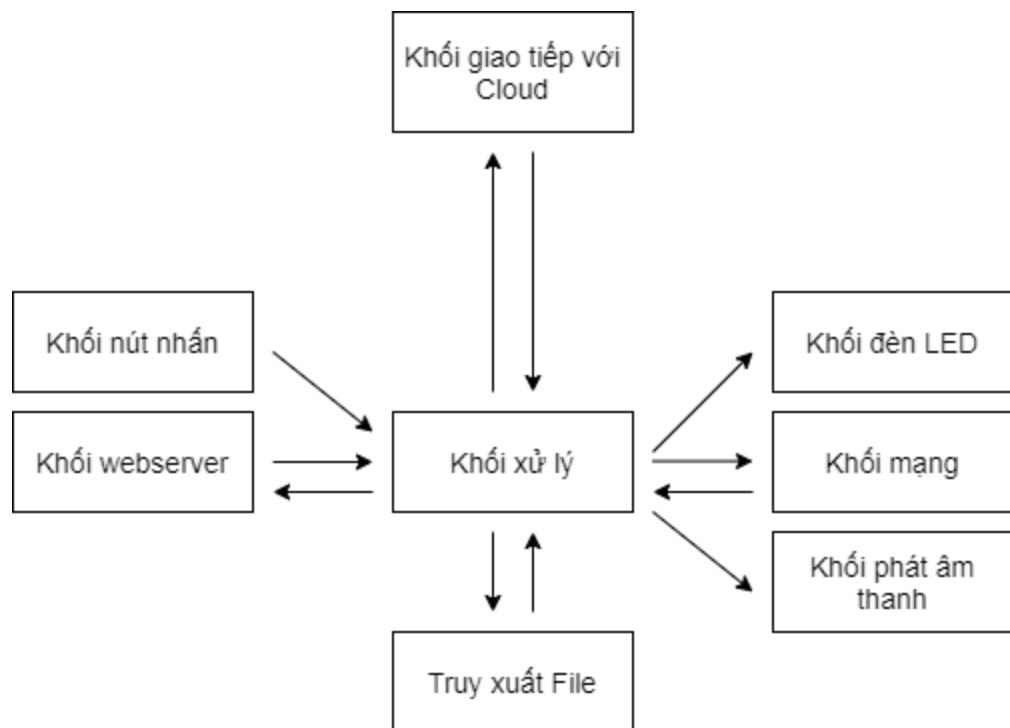
Kết nối với PortB của PIC. Các chân C1, C2, C3, C4 nối với điện trở pulldown

## 3.2 THIẾT KẾ PHẦN MỀM

### 3.2.1 Giải thuật phần mềm thiết bị Home server

#### 3.2.1.1 Các khối giải thuật

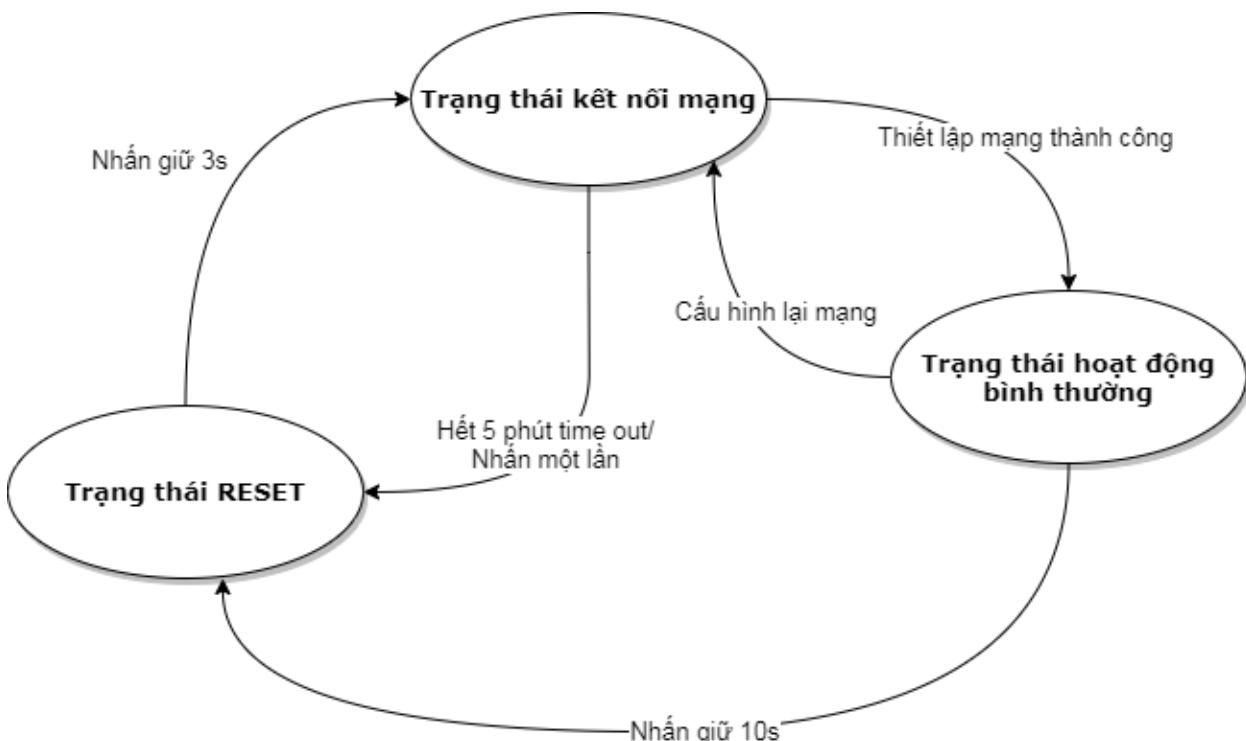
- Khối nút nhấn
- Khối âm thanh
- Khối điều khiển đèn LED trang trí
- Khối giao tiếp mạng
- Khối truy xuất File
- Khối Webserver
- Các khối chức năng hoạt động bằng các luồng Thread song song chạy trên nền ngôn ngữ Python



Hình 3-24 Mô tả kết nối các khối phần mềm

### 3.2.1.2 Các trạng thái hoạt động

- Trạng thái reset: Không có kết nối mạng. Đèn tắt. Chỉ có khói nút nhấn hoạt động để đổi trạng thái
- Trạng thái kết nối mạng: Chớp đèn đều đặn khi đang tạo accesspoint hoạt kết nối đến cloud
- Trạng thái hoạt động bình thường: Đèn sáng, thiết lập kết mạng, có thể hoạt động các chức năng được cấu hình



Hình 3-25 Máy trạng thái thiết bị Homeserver

### 3.2.1.3 Chi tiết các khối giải thuật

#### 3.2.1.3.1 Khối nút nhấn

Mạch có gồm hai nút nhấn.

Chức năng nút 1: Chuyển đổi trạng các thiết bị

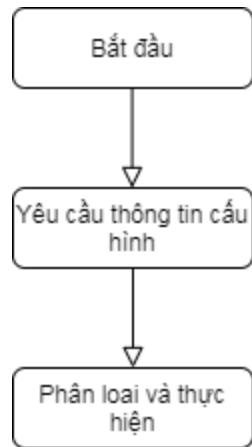
Chức năng nút 2: Quản lý kết thiết bị

Hoạt động dựa trên bộ thư viện RPi.GPIO của ngôn ngữ Python

### 3.2.1.3.2 Khối âm thanh

Chức năng: Quản lý phát âm thanh trên thiết bị.

Hoạt động dựa trên các bộ thư viện aplay, mplayer, ffmpeg, gstreamer, vlc. Đây là các bộ thư viện chạy trên hệ thống Linux.



Hình 3-26 Sơ đồ khối âm thanh

Khối hoạt động đơn giản. Khi có tín hiệu yêu cầu khôi sẽ bắt đầu thực thi lệnh, phân loại tín hiệu phát ra những file âm thanh đã được lưu sẵn. Tùy theo định dạng file mà thực hiện các lệnh Linux để chạy các bộ thư viện khác nhau.

### 3.2.1.3.3 Khối đèn LED

Đèn LED thể hiện trạng thái hoạt động của mạch và chức năng trang trí.

Mạch gồm 4 đèn LED RGB. Kết hợp tạo ra 7 bảy màu và trạng thái tắt.

Hoạt động dựa trên bộ thư viện RPi.GPIO của ngôn ngữ Python.

Khối hoạt động thường trực trong suốt quá trình hoạt động của thiết bị.

Bao gồm các giai đoạn đọc thiết lập, đọc File cấu hình, hoạt động.

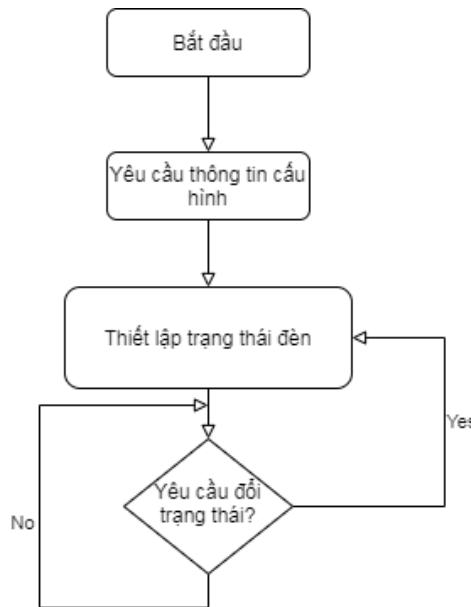
Các chế độ hoạt động bao gồm:

+ Tắt

+ Sáng bình thường một trong bảy màu (tổ hợp của 3 màu LED)

+ Chớp với tần số qui ước

Bình thường thiết bị sẽ hoạt động theo chế độ của File cấu hình. Khi có tín hiệu từ hệ thống hoặc người dùng khởi động thực hiện thay đổi chế độ tùy vào yêu cầu.



Hình 3-27 Sơ đồ khối đèn LED

### 3.2.1.3.4 Khối giao tiếp mạng

Các hình thức giao tiếp mạng:

- + Giao tiếp bằng các gói tin TCP/UDP. Tạo một Server nhận các gói tin gửi đến và một khối xử lý các gói tin gửi đi.



Hình 3-28 Khối nhận gói tin



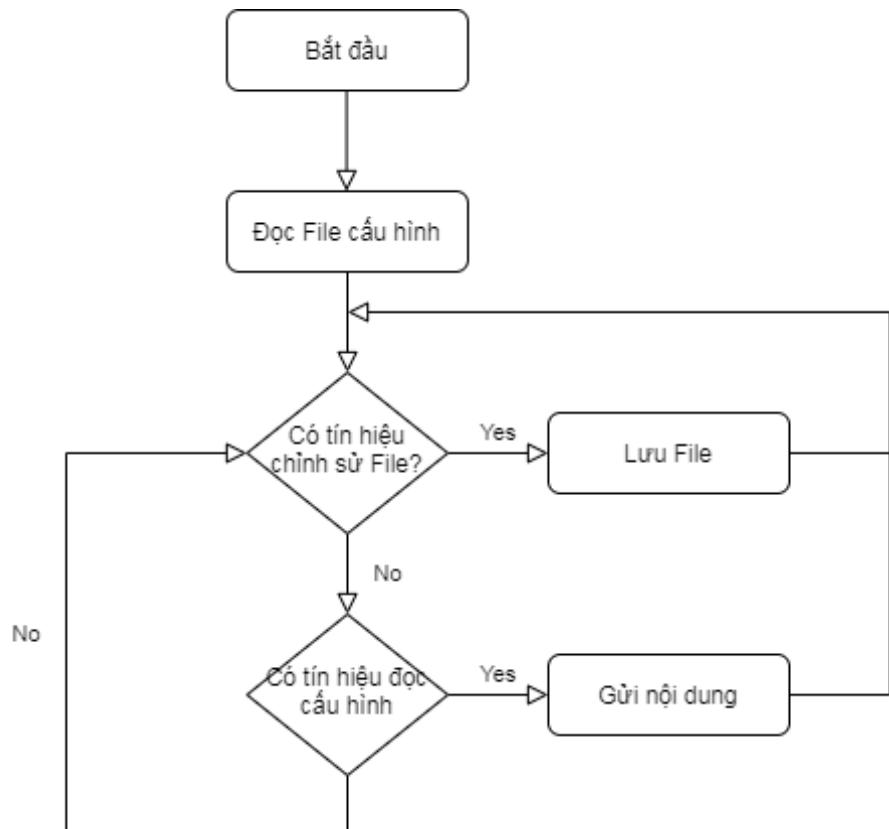
Hình 3-29 Khối gửi gói tin

+ Giao tiếp bằng giao thức MQTT. Pub/sub các lệnh gửi đến topic đã được quy định trước trên borker

### 3.2.1.3.5 Khối truy xuất File

Lưu trữ cấu hình người dùng, cấu hình thiết bị và các cài đặt cần thiết.

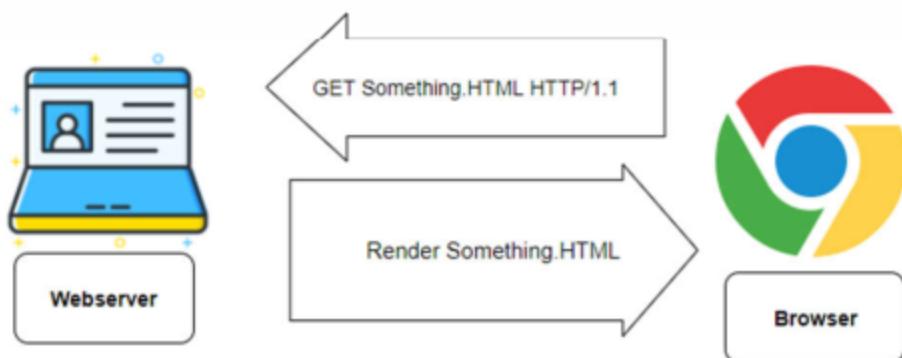
Hoạt động dựa trên thư viện ConfigParser của python.



Hình 3-30 Lưu đồ khối truy xuất File

### 3.2.1.3.6 Khối Webserver

Dựa trên nền tảng Flask Framework được viết trên nền Python Flask Framework sẽ giúp truy vấn các gói tin HTTP để hiển thị trên trình duyệt. Khi người nhấn vào đường link hoặc nút nhấn trên trình duyệt. Trình duyệt sẽ gửi một truy vấn đến server và server sẽ gửi nội dung lên trình duyệt bằng HTML. Ngoài ra để tạo thêm hiệu ứng và làm đẹp trang web cũng như gửi thực hiện một số tác vụ cần dùng đến CSS và Javascript. Khi đã nhận đủ dữ liệu nhiệm vụ còn lại là của trình duyệt tự động biên dịch và biến các gói tin thành giao diện trang web. Tuy nhiên nếu chỉ đơn thuần sử dụng web thì không thể cập nhật các thông tin theo hướng ngược lại từ server lên trình duyệt nếu không có truy vấn từ trình duyệt. Sử dụng SocketIO để tạo nên Web page động. Các nội dung trên trang Web đều sẽ được cập nhật thời gian thực với các tác động lên thiết bị.



Hình 3-31 Mô tả quá trình hoạt động của WebServer

### 3.2.2 Giải thuật phần mềm Camera chuông cửa

#### 3.2.2.1 Các khối giải thuật

- Khối nút nhấn
- Khối âm thanh
- Khối điều khiển đèn LED trang trí
- Khối giao tiếp mạng
- Khối truy xuất File

### **3.2.2.2 Chi tiết các khối giải thuật**

Các khối giải thuật hoạt động tương tự thiết bị Home Server đã được giới thiệu phía trên.

### **3.2.3 Giải thuật phần mềm thiết bị điều khiển hồng ngoại**

#### **3.2.3.1 Các khối giải thuật**

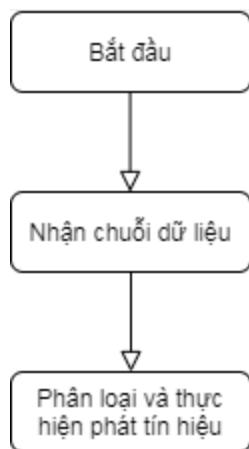
- Khối nút nhấn
- Khối thu hồng ngoại
- Khối phát hồng ngoại
- Khối giao tiếp mạng
- Khối đọc ghi EPPROM

#### **3.2.3.2 Chi tiết các khối giải thuật**

Các khối giải thuật về nút nhấn, giao tiếp mạng hoạt động tương tự với nguyên lý của khối giải thuật phần HomeServer.

Khối đọc ghi EPPROM cũng có chức năng và nguyên lý tương tự khối quản lý File nhưng với cách thức đọc ghi khác và nội dung ghi cũng hạn chế hơn do không lưu trữ.

##### **3.2.3.2.1 Khối phát hồng ngoại:**



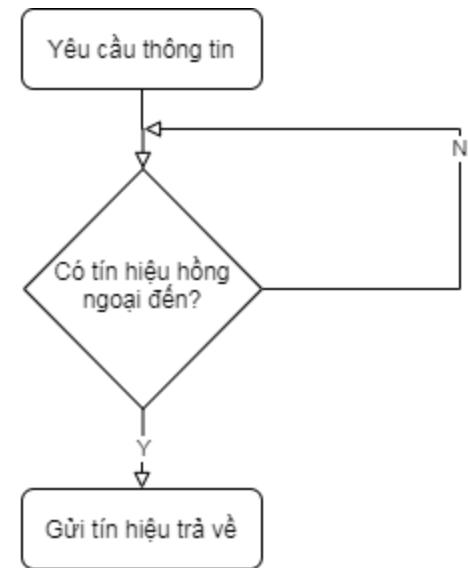
Hình 3-32 Lưu đồ giải thuật khói phát hồng ngoại

Thiết bị sẽ nhận tín hiệu gửi từ Server. Có 2 cơ chế gửi:

+ Cơ chế 1: Các thiết bị điều khiển có trong thư viện cứng của chương trình. Server chỉ cần gửi thông tin câu lệnh

+ Cơ chế 2: Server gửi các tín hiệu có liên quan đến mã code, chu kỳ, độ dài code,... đến thiết bị. Các thông tin này được lưu trong cơ sở dữ liệu của server thông qua quá trình thu thập từ các nguồn và học lệnh.

### 3.2.3.2.2 Khối thu hồng ngoại



Hình 3-33 Lưu đồ khối thu hồng ngoại

Sau khi nhận tín hiệu yêu cầu thu tín hiệu hồng ngoại để nhận dạng hoặc học lệnh. Thiết bị sẽ đợi đến khi có tín hiệu hồng ngoại gửi đến và trả về Server các mã nhận dạng nếu tìm thấy hoặc mã học lệnh và lưu trữ ở phía Server.

### 3.2.4 Giải thuật phần mềm thiết bị điều khiển relay

#### 3.2.4.1 Các khối giải thuật

- Khối nút nhấn
- Khối điều khiển relay
- Khối giao tiếp mạng
- Khối đọc ghi EEPROM

- Khối nhận Keypad
- Khối điều khiển LCD

### ***3.2.4.2 Chi tiết các khối giải thuật***

Các khối hoạt động đều giống thiết bị điều khiển hồng ngoại và tín hiệu sẽ được truyền trung gian đến ESP8266 rồi truyền đến PIC thông qua cổng truyền thông UART. PIC sẽ đảm nhiệm tác vụ thực thi các lệnh này như hiển thị LCD hoặc điều khiển Relay.

Một số khối độc lập với ESP8266:

#### **3.2.4.2.1 Khối nhận Keypad**

Quét tín hiệu ở 4 chân nối với các cột của Keypad và thu tín hiệu ở bốn chân thu kết nối với 4 hàng của Keypad. Khi nhấn nút tín hiệu thu được ở chân thu cộng với tín hiệu đang quét sẽ biết tọa độ của hàng và cột của nút Keypad vừa nhấn.

#### **3.2.4.2.2 Khối điều khiển LCD**

Các chân LCD sẽ kết nối trực tiếp với các chân I/O của PIC. Tín hiệu sẽ được truyền thông qua các chân I/O đến LCD rồi hiển thị.

### **3.2.5 Giải thuật phần mềm của Cloud Server:**

#### ***3.2.5.1 Các khối giải thuật***

- Khối giao tiếp mạng MQTT
- Khối chạy ICE server
- Khối Webserver

#### ***3.2.5.2 Chi tiết các khối giải thuật***

##### **3.2.5.2.1 Khối giao tiếp mạng MQTT**

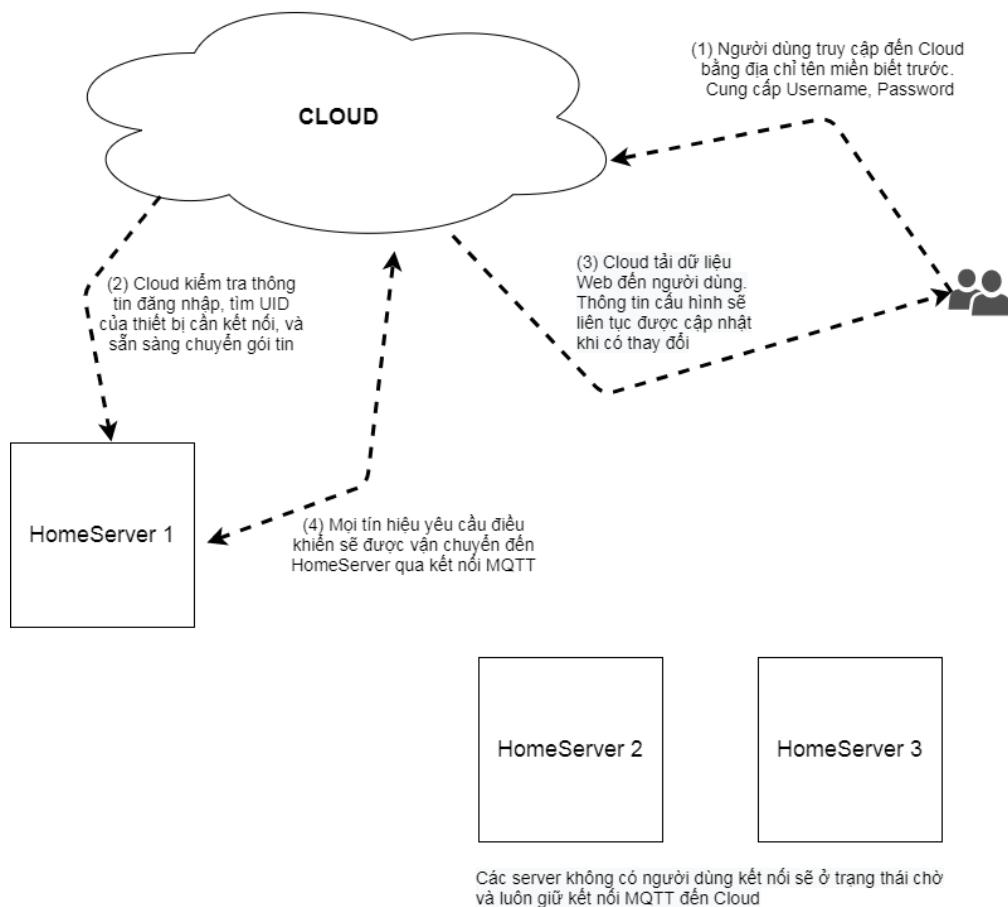
Giao tiếp MQTT với các thiết HomeServer. Broker sẽ được chạy trên Cloud mỗi thiết bị HomeServer sẽ được cấp một UID riêng mỗi UID này sẽ được kết nối vào một topic riêng, Cloud và HomeServer sẽ trao đổi dữ liệu thông qua việc pub/sub các gói tin trên topic này.

Chương trình được viết dựa trên bộ thư viện Paho-MQTT trên nền tảng ngôn ngữ Python

### 3.2.5.2.2 Khối Webserver

Tương tự với khối Webserver của HomeServer với một số chỉnh sửa:

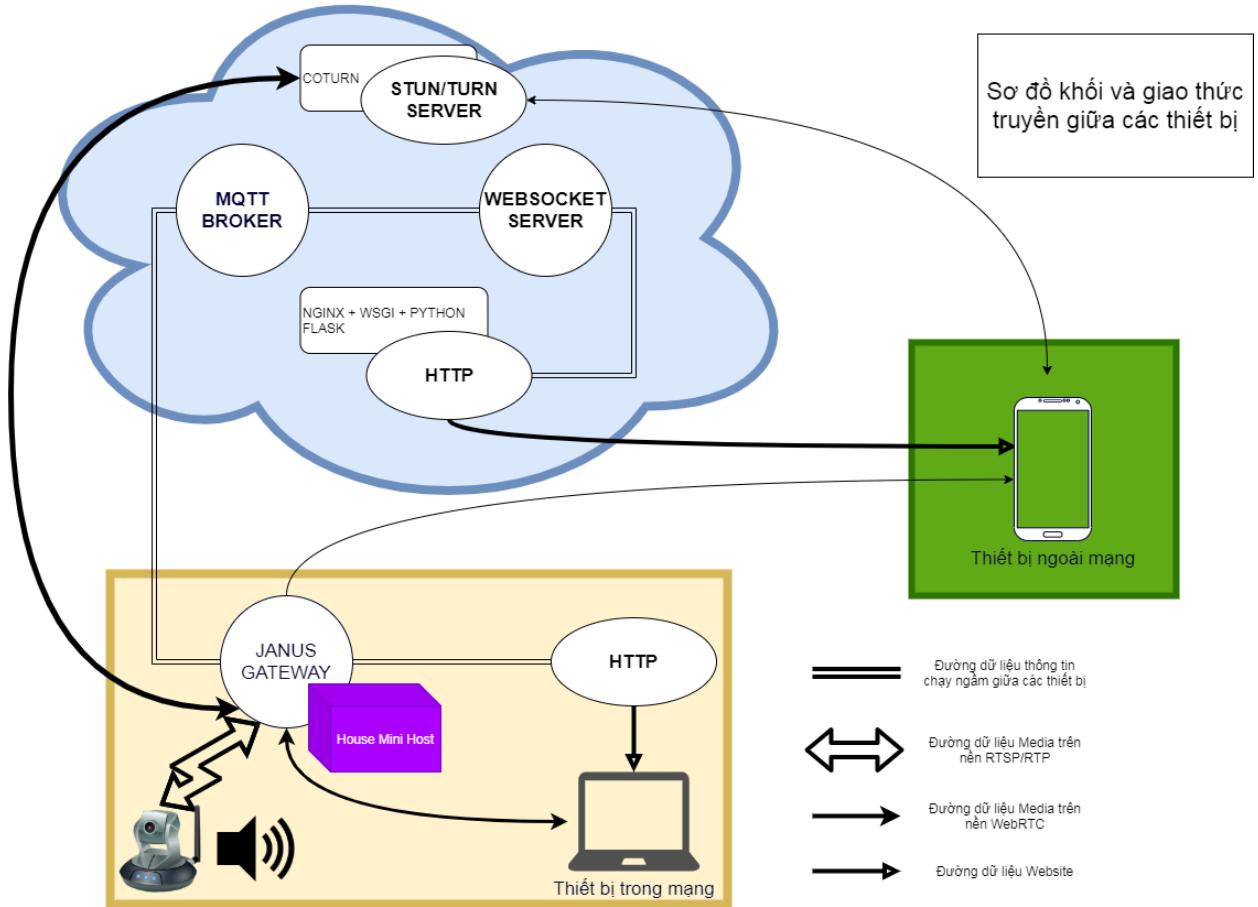
- + Thiết bị thay vì gửi Socket trực tiếp thì sẽ gửi tín hiệu thông qua đường kết nối MQTT đến thiết bị HomeServer.
- + Hệ thống cơ sở dữ liệu mySQL được đồng bộ với các File cấu hình của mỗi thiết HomeServer



**Hình 3-34 Mô tả hoạt động của Cloud Server**

### 3.3 GIAO THỨC KẾT NỐI VÀ GIAO DIỆN NGƯỜI DÙNG

#### 3.3.1 Sơ đồ khái quát



#### 3.3.2 Kết nối giữa các thiết bị

Kết nối các thiết bị trong mạng LAN

- Đường kết nối các gói tin nhỏ tín hiệu: Sử dụng giao thức vận chuyển cơ bản nhất là các gói tin UDP và TCP. Các giao đoạn kết nối:

+ Giai đoạn 1: Nhập cấu hình Wifi. Homeserver được kết nối vào mạng bằng 2 cách. Cách 1: dùng dây mạng LAN kết nối trực tiếp đến Router. Cách 2: Homeserver tạo thành Access point để người dùng thiết lập wifi.

## Cài đặt mạng

*Bạn đang ngoại tuyến. Hãy kết nối cảm biến bằng dây mạng hoặc thiết lập mật khẩu WiFi*

### Wireless

SSID

Mật khẩu

mDNS

### Cấu hình Cloud

Tên miền

Hình 3-35 Giao diện nhập mật khẩu wifi

- + Giai đoạn 2: Cấu hình wifi cho các thiết bị. Với phương pháp tương tự. Homeserver sẽ tạo một access point để các thiết bị liên kết đến và gửi thông tin qua các gói tin TCP để đăng nhập wifi.

The screenshot shows the 'Device manager' interface. At the top, it says 'Device manager'. Below that, there's a 'Choose Device' dropdown set to 'IR001'. Under 'More Info', there's a 'More detail' button. In the 'Network' section, there are fields for 'Ssid' and 'Password', both currently empty. There are also checkboxes for 'Smart config', 'Static IP', and 'Enter code'. A note field below says 'Write something..'. At the bottom right, there's a 'Connect to device' button.

Hình 3-36 Giao diện cấu hình wifi cho các thiết bị

Trong giai đoạn các gói tin cũng sẽ được trao đổi để thiết bị biết được cách kết nối đến server như địa chỉ IP/tên miền mDNS của server bên trong mạng

+ Giao đoạn 3: Hoạt động. Các gói tin TCP được dùng để kết nối tín hiệu điều khiển giữa các thiết bị.

Trong suốt quá trình hoạt động. Gói tin được định dạng bằng chuẩn JSON để giao tiếp. Các gói tin TCP đảm bảo cho việc hoạt động chính xác đảm bảo rằng gói tin đến đúng và đủ đến thiết bị ở cả hai chiều. Các gói tin UDP chỉ được dùng trong khâu kết nối khi cần một lượng lớn gói tin được gửi đi và bên nhận chỉ cần nhận đúng một gói tin là đủ.

### 3.3.3 Kết nối đến cloud

Trong toàn mạng chỉ có duy nhất một thiết bị được kết nối với Cloud đó là Homeserver. Kết nối này được thực hiện bằng giao thức MQTT với broker nằm ở phía Cloud và các Homeserver hoạt động là một Client. Mỗi một HomeServer sẽ được cấp UID duy nhất. UID này được cấp khi người dùng đăng ký đến Cloud và không đổi đến khi reset server.

### 3.3.4 Thiết kế hệ thống Streaming trên nền tảng WebRTC

#### Giới thiệu

Thực tế về việc truyền tải dữ liệu trên thị trường có rất nhiều phương án. Cụ thể:

- Port Forwarding (Cấu hình Router mạng để có định cổng Port trả đến thiết bị Camera) đồng thời kết hợp với việc mua tên miền để truy cập đến mạng gia đình do IP cấp cho cá nhân sử dụng đều là các IP động. Khi hết phiên đều thay đổi IP. Sử dụng phương pháp này tuy đơn giản về mặt kỹ thuật, giá thành cực thấp do không cần máy chủ để điều hành truyền dữ liệu nhưng lại gây khó khăn cho người dùng khi không phải ai cũng hiểu rõ về hệ thống mạng để cấu hình. Do đó cần tốn nhân lực kỹ thuật viên để hỗ trợ. Hiện nay một số router cũng hỗ trợ cơ chế uPnP giúp người dùng ít khó khăn hơn trong việc cấu hình mạng nhưng việc triển khai là không đồng bộ chưa nói đến các vấn đề về bảo mật do phương pháp này gây nên.

- Dùng hệ thống mạng ảo VPN. Hệ thống mạng trong nhà và người dùng sẽ cùng kết nối đến hệ thống máy chủ VPN từ đó hệ thống mạng trong gia đình và người dùng có

thể xem như nằm trong cùng một mạng ảo và có thể dễ dàng trao đổi dữ liệu trong mạng. Đây cũng là một phương pháp ổn định. Tuy nhiên chi phí điều hành một hệ thống mạng VPN khá cao. Trên thị trường các hệ thống sử dụng sẽ được tính phí theo tháng và tính phí cả dung lượng sử dụng

- Hoạt động cơ chế Cloud Server. Đây là một phương pháp khá phổ biến và tối ưu. Hệ thống sẽ gửi dữ liệu lên Cloud Server người dùng sẽ kết nối đến Cloud Server để nhận nhận dữ liệu. Tuy nhiên cũng sẽ tốn chi phí vận hành Cloud Server và tất cả băng thông truyền đều đi qua Cloud mới đến được với người dùng.

Để tài sẽ sử dụng một phương pháp khác. Dựa trên một nền tảng mới WebRTC và cơ chế P2P.

Với việc sử dụng nền tảng WebRTC để tài nhằm tối ưu việc sử dụng băng thông mạng và ưu tiên kết nối P2P một cách giữa người dùng và hệ nhằm tối ưu quá trình hoạt động của server vốn đã tồn tại nguyên.

Tuy nhiên với thiết kế hệ thống mạng được triển khai rộng rãi hiện nay cũng với lượng Ipv4 gần cạn kiệt và giao thức NAT hệ thống không thể đảm bảo 100% kết nối là P2P. Việc kết nối được thực hiện của các giai đoạn được thực hiện tuần tự theo giao thức ICE đã được trình bày ở phần lý thuyết.

## Chi tiết

Để việc Streaming hoạt động được bao quát các trường hợp trong các điều kiện kết nối khác nhau cần các thành phần cấu thành sau.

- Hệ thống cần kết nối
- Người dùng
- Hệ thống TURN Server và STUN Server.

Việc kết nối sẽ không phải lúc nào cùng dùng đến server. Việc quyết định kết nối sẽ dựa vào cơ chế ICE (chi tiết ở phần lý thuyết).

Hệ thống sẽ tự động nhận diện khi nào sử dụng Server trung gian và khi nào không cần thiết theo thứ tự ưu tiên.

Thứ tự 1: Nếu như có thể kết nối trực tiếp với nhau. Giả sử như trong cùng một mạng LAN thiết bị người dùng và hệ thống sẽ được kết nối trực tiếp

Thứ tự 2: Không thể thiết lập kết nối trực tiếp. Bắt đầu cơ chế Hole Punching.

Dựa vào STUN Server hệ thống và người dùng sẽ tìm được địa chỉ IP và số Port kết nối. Việc kết nối được hay không sẽ phụ thuộc vào cơ chế NAT mà các bên sử dụng.

Nếu như đây là Static NAT ở cả hai bên kết nối sẽ được thực hiện mà không cần đến TURN Server

Thứ tự 3: Không thể trực tiếp thực hiện kết nối P2P. Toàn bộ dữ liệu media sẽ được chuyển tiếp qua một server bên thứ 3: TURN Server.

Do đó việc kết nối sẽ giúp tối ưu băng thông gửi cho việc truyền dữ liệu đến Cloud. Giảm tối đa chi phí và độ trễ cũng như không cần cấu hình khó khăn ở phía người dùng

### **3.3.5 Giới thiệu cơ bản một quá trình cấu hình chuẩn bị cho việc streaming với FFmpeg, Janus Gateway và webRTC api**

Mục này sẽ trình bày một ví dụ thực hiện cơ bản một kết nối từ thiết bị camera và hiện trên trình duyệt web bằng các công nghệ nêu trên. Đây chỉ là một ví dụ cơ bản. Việc kết nối hệ cần nhiều bước cấu hình và lập trình phức tạp hơn nhưng cũng dựa trên những nền tảng cơ bản. Việc cơ nối cần ba thiết bị cơ bản

1/ Camera:

Camera có thể là bất cứ loại nào có hỗ trợ phát triển bằng RTP hoặc RTSP sử dụng các loại codec phổ biến (H263/H264, VP8/VP9,...).

Trong mục này sẽ sử dụng module Raspberry Pi 2 tích hợp camera phát tín hiệu RTP bằng FFmpeg.

Một câu lệnh Ffmpeg cơ bản sẽ gồm 3 thành phần. Đầu vào, Codec + Các thiết lập lên nội dung và đầu ra.

Sau đây là một ví dụ một câu lệnh cơ bản:

**Câu lệnh 1:**

```
ffmpeg -f v4l2 -channel 0 -s 1280x720 -i /dev/video0 -f rtp -vcodec libx264 -r 30 -b:v 64k rtp://ip_cua_janus.com:8081
```

Phân tích câu lệnh trên:

- + Đầu vào được đánh dấu bằng `-i`. Nội dung `/dev/video0` tức là camera 0 do hệ điều hành Linux qui định
- + Phần codec được sử dụng là h264 với thư viện libx264 nêu trên. Phần Codec này sẽ chạy trên CPU nên khá tốn tài nguyên của Raspi. Tuy nhiên có thể thay đổi cấu hình để chạy được trên GPU tuy nhiên việc thiết lập bên nhận sẽ khó khăn hơn do có sự khác biệt về các gói tin khi gửi SDP rtp. Việc xử lý sẽ không trình bày ở đây.
- + Phần đầu ra `rtp://ip_cua_janus.com:8081` chính là đường dẫn để truyền tín hiệu rtp đến bên thu với địa chỉ của máy thu và port.

Ngoài ffmpaq còn có một số bộ thư viện khác như Gstreamer cũng có chức năng tương tự. Một số câu lệnh Gstreamer được sử dụng trong hệ thống:

## 2/ Server chạy Janus Gateway

Do Janus hiện tại hỗ trợ hệ điều hành Linux nên server có thể là thiết bị chạy được hệ điều hành Linux.

Trong mục này sẽ sử dụng module Raspberry Pi 3 làm server

Bộ thư viện Janus gồm nhiều plugin khác nhau. Trong mục này sẽ giới thiệu về plugin Streaming.

Để thiết lập Server chạy Janus có 2 cách. Sử dụng các File cấu hình hoặc cấu hình bằng API. Ví dụ này sẽ cấu hình đơn giản bằng File cấu hình.

## 3/ Thiết bị cuối cùng là máy tính hoặc điện thoại thông minh chạy được trình duyệt.

Hiện tại các phiên bản mới Chrome, Opera, Mozilla Firefox đều hỗ trợ WebRTC (đã kiểm chứng).

Để giao tiếp với Server Janus. Nhà phát triển đã cung cấp một bộ thư viện Javascript. Chi tiết tham ở link đính kèm ở tài liệu tham khảo mục

Một số đoạn lập trình quan trọng

### Đoạn khởi tạo server

```
var janus = new Janus(
{
    server: 'http://dia_chi_server_chay_janus:8088/janus',
    success: function() {
    },
    error: function(cause) {
    },
    destroyed: function() {
    }
});
```

**server:** địa chỉ server kết nối đến.

**success:** thực hiện kết nối thành công, bắt đầu khởi tạo các plugin và các đoạn custom cần thiết

**error:** trả về khi có lỗi xảy ra

**destroyed:** trả về khi chủ động ngắt kết nối

### Đoạn khởi tạo plugin

```
janus.attach(
{
    plugin: "janus.plugin.streaming",
    success: function(pluginHandle) {
    },
    error: function(cause) {
    },
    consentDialog: function(on) {
    },
    onmessage: function(msg, jsep) {
    },
    onlocalstream: function(stream) {
    },
    onremotestream: function(stream) {
    },
    oncleanup: function() {
    },
    detached: function() {
    }
});
```

**plugin:** tên plugin kết nối đến Janus. Janus chạy trên hệ thống các plugin. Lập trình viên có thể viết thêm các plugin phù hợp với yêu cầu hệ thống hoặc tự do chỉnh sửa các plugin đã có.

**success:** trả về khi thiết lập thành công cùng với Object pluginHandle để tương tác với plugin

**error:** trả về do gặp lỗi kết nối

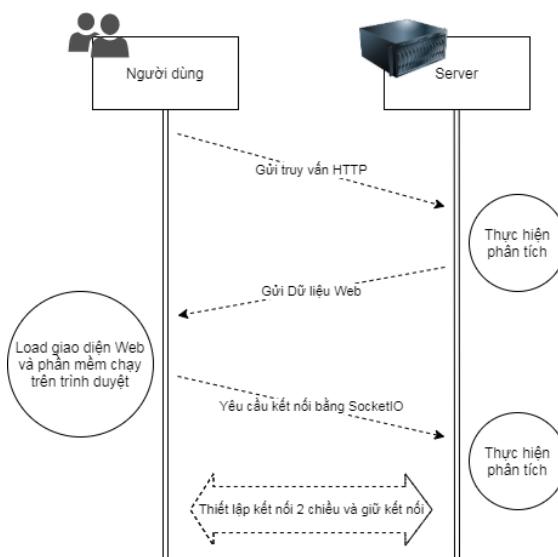
**onmessage:** quản lý thông báo từ plugin

**detached:** trả về khi ngắt kết nối

### 3.3.6 Giao diện người dùng

Phần giao diện người dùng sẽ được truy cập thông qua trình duyệt web. Thiết bị HomeServer và Cloud sẽ đảm nhiệm chức năng làm WebServer để thực hiện truy vấn các yêu cầu HTTP từ trình duyệt và tải giao diện bằng các đoạn File HTML, CSS và JavaScript để hình thành trang Web.

Hệ thống còn mở rộng thêm kỹ thuật sử dụng SocketIO để kết nối “Realtime” với dùng. Trong quá trình hoạt động luôn có những trường hợp phải gửi dữ liệu từ Server ngược lại phía Client. Quá trình này để hoạt động được phải sử dụng một kết nối liên tục để Server có thể gửi thông tin hoạt động ngược về người dùng. Quá trình làm việc có thể mô tả một cách cơ bản nhất qua hình ảnh sau:



Hình 3-37 Mô tả quá trình kết nối giữa trình duyệt Web và Server

## Chương 4 THI CÔNG VÀ MỞ RỘNG

### 4.1 Thi công thực tế

#### 4.1.1 Các vấn đề khi thi công

Khi thực hiện thi công có một số vấn đề. Tuy không quá lớn nhưng cũng đáng kể gây mất thời gian như:

- Thi công lỗi: Gắn ngược LED, gắn ngược Diode, gắn ngược IC, ngâm thêu hóa chất, cùng nhiều hàn dính chân linh kiện và đứt/bong tróc đường mạch

- Thiết kế lỗi: Một số lỗi đáng chú ý:

+ Thiết kế nhầm giá trị tụ ở mạch nút nhấn cảm ứng.

+ Thiết kế hai chân Relay nối L với N gây chập mạch điện

+ Thiết kế vị trí các LED hồng ngoại không hợp lý dẫn đến việc không phát đèn tín hiệu hồng ngoại ra các hướng

Tổng số mạch phải thi công:

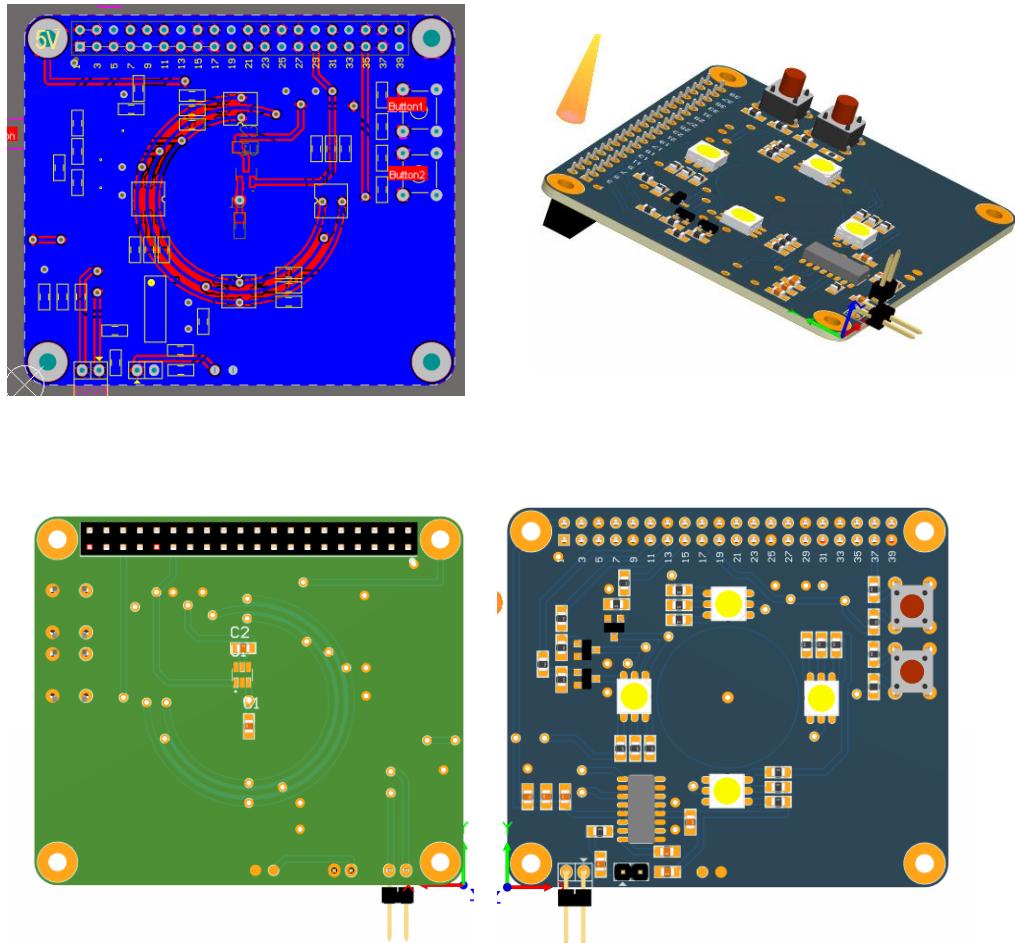
+ Mạch Camera và HomeServer: 4 mạch (Mạch đặt)

+ Mạch điều khiển thiết bị hồng ngoại: PCB 1 (3 mạch), PCB 2 (6 mạch), PCB 3 (1 mạch)

+ Mạch điều khiển thiết bị bằng relay: PCB 1 (2 mạch), PCB 2 (2 mạch)

#### 4.1.2 Kết quả thi công và Layout

##### 4.1.2.1 Mạch Homeserver và camera chuông cửa:

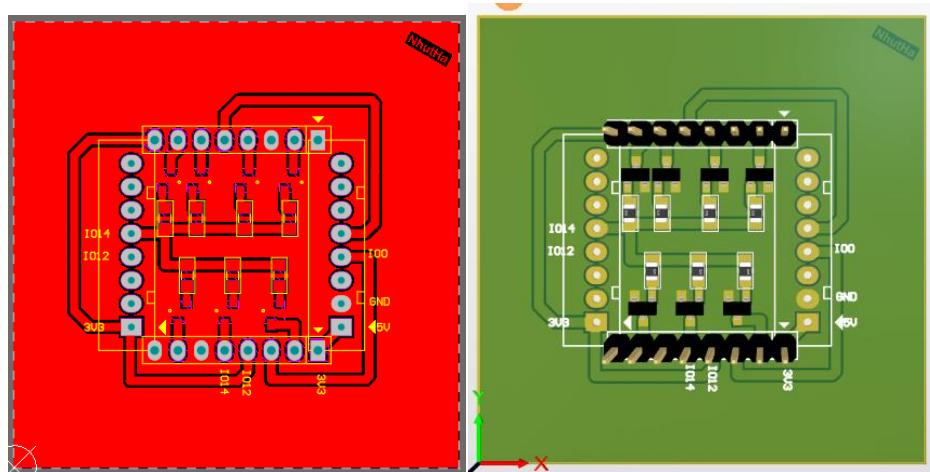
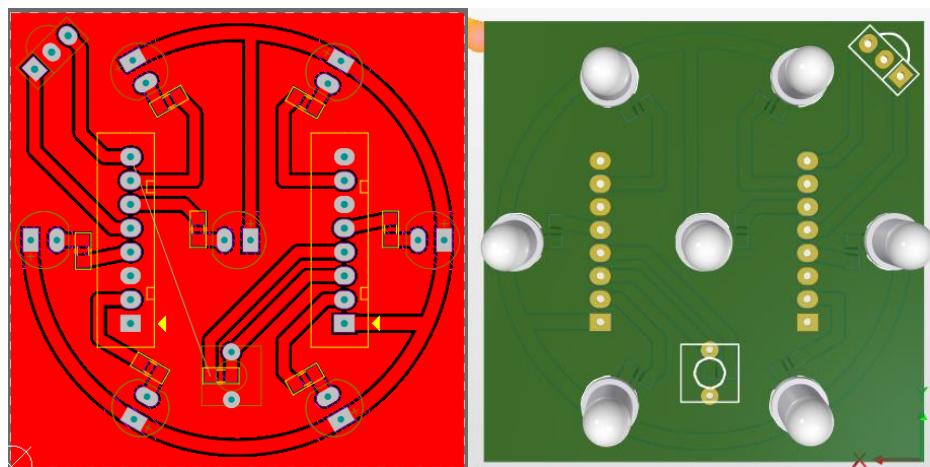
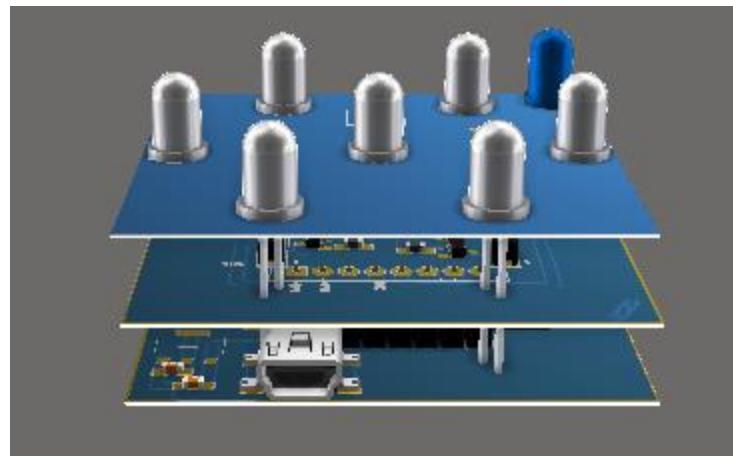


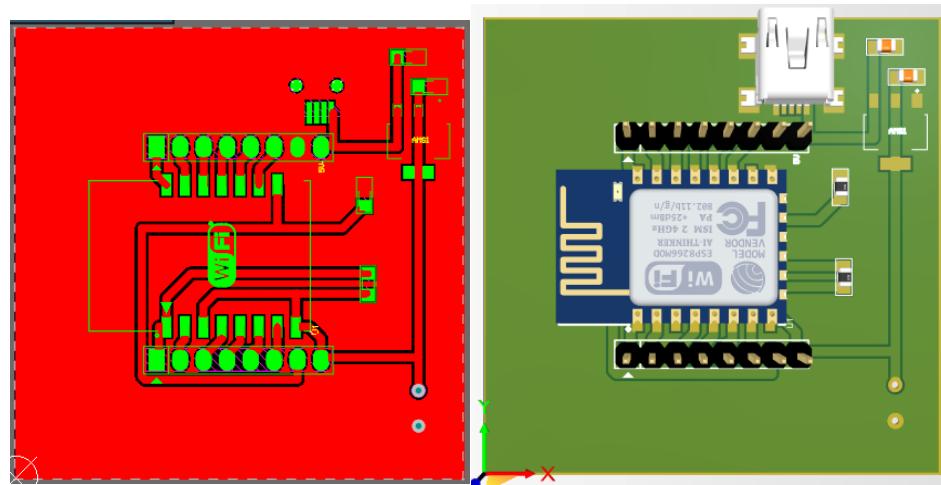
Hình 4-1 Layout mạch Camera chuông cửa và HomeServer



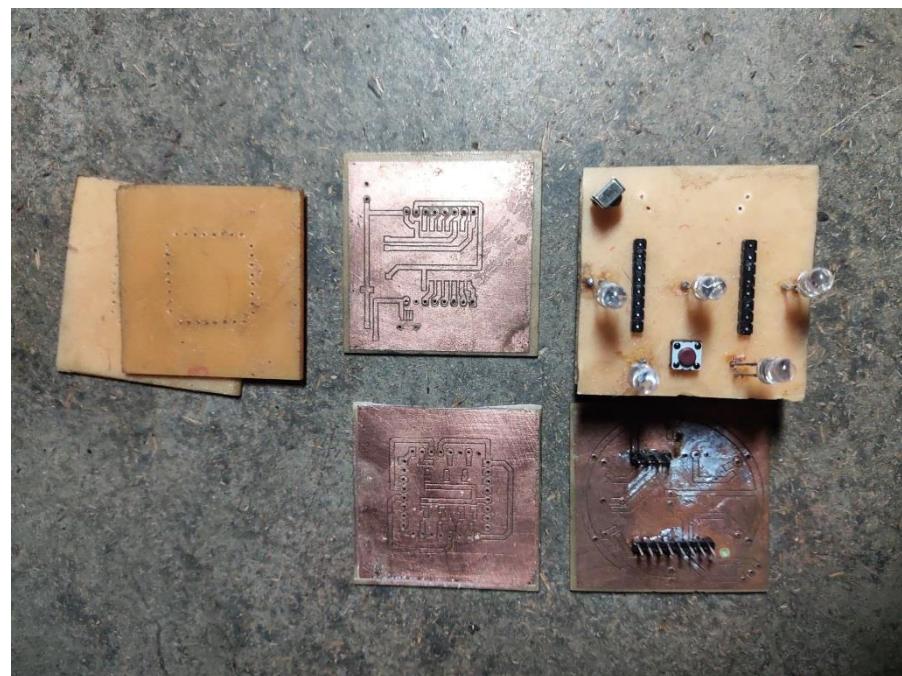
Hình 4-2 Kết quả thi công mạch HomeServer và chuông cửa

#### 4.1.2.2 Mạch điều khiển thiết bị hồng ngoại



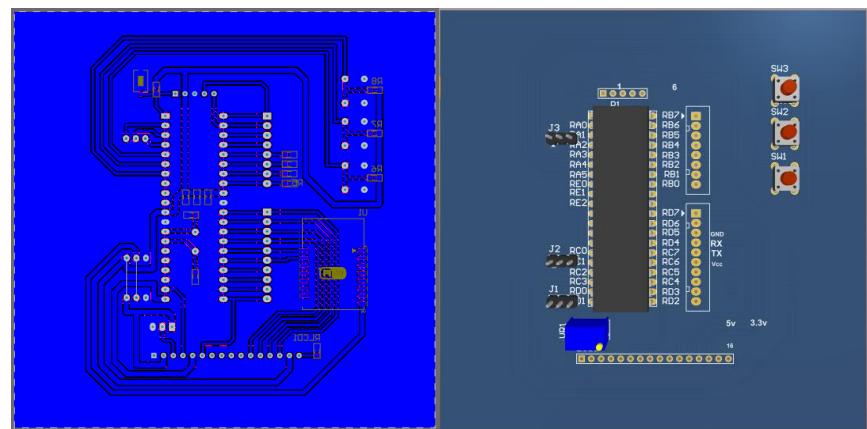
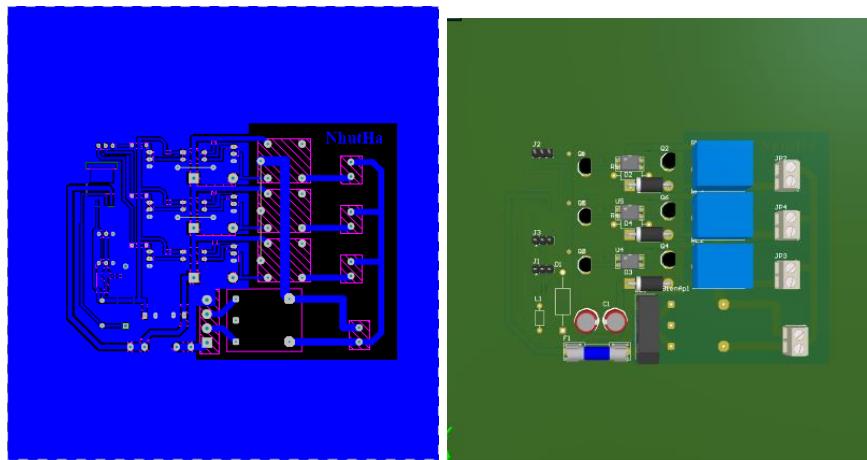


Hình 4-3 Layout mạch điều khiển thiết bị hồng ngoại



Hình 4-4 Kết quả thi công thiết bị điều khiển hồng ngoại

#### 4.1.2.3 Mạch điều khiển thiết bị bằng Relay



Hình 4-5 Layout mạch điều khiển thiết bị bằng Relay



Hình 4-6 Kết quả thi công mạch điều khiển thiết bị bằng Relay

#### 4.2 Ý tưởng mở rộng đề tài trên nền tảng thiết kế

- Tăng thêm một số các thiết bị chức năng vào hệ thống:

- + Các hệ thống cảm biến: nhiệt độ, độ ẩm, khí gas, lượng mưa,..
  - + Các thiết bị tự động: Đèn chuyển màu tự động
  - Kết hợp một số tính năng vào thiết bị:
    - + Nhận dạng giọng nói
    - + Hoàn thiện các chức năng thị giác máy
  - Mở rộng hỗ trợ tiếp cận bằng điện thoại thông minh, smart TV
  - Phiên bản phần mềm hỗ trợ chạy server trên hệ thống máy tính mạnh mẽ hơn
  - Chuyên dụng hóa hệ thống vào các lĩnh vực y tế, giáo dục,..
- Cụ thể:
- Lĩnh vực y tế:
    - + Số khám bệnh điện tử: Quản lý lịch khám bệnh, tình trạng bệnh, thanh toán viện phí.
    - + Hệ thống quản lý và hướng dẫn bệnh nhân lắp đặt trong bệnh viện
  - Lĩnh vực giáo dục:
    - + Hệ thống quản lý học sinh, thời khóa biểu, lịch thi lịch làm việc, học phí

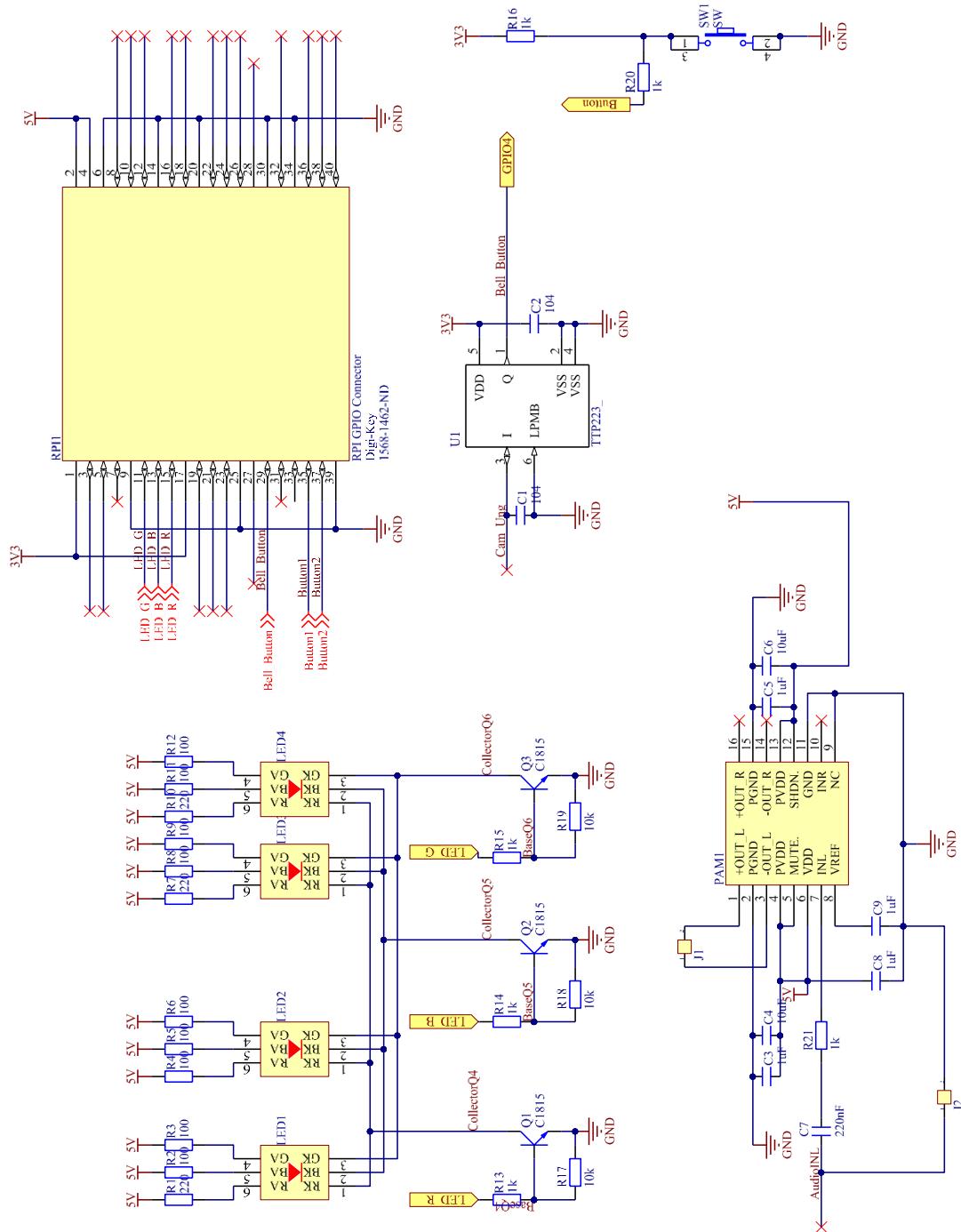
## Chương 5 TÀI LIỆU THAM KHẢO

- [1] Giới thiệu về hệ thống của Tuya <https://tuya.vn/>
- [2] Giới thiệu về hệ thống của Samsung <https://www.samsung.com/vn/apps/smart-switch/>
- [3] Giới thiệu về hệ thống của BKAV <https://smarthome.com.vn/>
- [4] Giới thiệu về hệ thống của LUMI  
[http://lumi.vn/?utm\\_source=google&utm\\_medium=cpc&utm\\_campaign=search\\_lumivn](http://lumi.vn/?utm_source=google&utm_medium=cpc&utm_campaign=search_lumivn)
- [5] Datasheet LED 5050 <https://e-radionica.com/productdata/RGB5050LED.pdf>
- [6] Datasheet C1815 <https://www.alldatasheet.com/datasheet-pdf/pdf/30084/TOSHIBA/C1815.html>
- [7] Datasheet TSOP1838 <https://www.alldatasheet.com/datasheet-pdf/pdf/26604/VISHAY/TSOP1838.html>
- [8] Datasheet ESP8266 [https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)
- [9] Datasheet PIC16F877A <https://www.alldatasheet.com/datasheet-pdf/pdf/115039/MICROCHIP/PIC16F877A.html>
- [10] Datasheet Raspberry 3 model B [https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi\\_DATA\\_CM3plus\\_1p0.pdf](https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM3plus_1p0.pdf)
- [11] Datasheet Raspberry 2 model B <https://cdn-shop.adafruit.com/pdfs/raspberrypi2modelb.pdf>
- [12] Lý thuyết về WebRTC <https://vi.wikipedia.org/wiki/WebRTC>
- [13] Lý thuyết về P2P [https://vi.wikipedia.org/wiki/M%E1%BA%A1ng\\_ngang\\_h%C3%A0ng](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_ngang_h%C3%A0ng)
- [14] Lý thuyết về NAT <https://www.totolink.vn/article/90-3-loai-nat-network-address-translation-ban-can-biet.html>
- [15] Lý thuyết về H264 [https://vi.wikipedia.org/wiki/H.264/MPEG-4\\_AVC](https://vi.wikipedia.org/wiki/H.264/MPEG-4_AVC)

- [16] Giới thiệu về Codec <https://quantrimang.com/codec-la-gi-sao-phai-can-codec-lam-gi-158152>
- [17] Giới thiệu về FFMPEG <https://bizflycloud.vn/tin-tuc/ffmpeg-la-gi-cach-xem-video-tren-trinh-phat-media-cu-bang-ffmpeg-20200521161944536.htm>
- [18] Datasheet PAM8204 <https://html.alldatasheet.com/html-pdf/246505/PAM/PAM8403/234/4/PAM8403.html>
- [19] Thư viện hồng ngoại LIRC <http://lirc-remotes.sourceforge.net/remotes-table.html>
- [20] Janus API <https://janus.conf.meetecho.com/docs/JS.html>

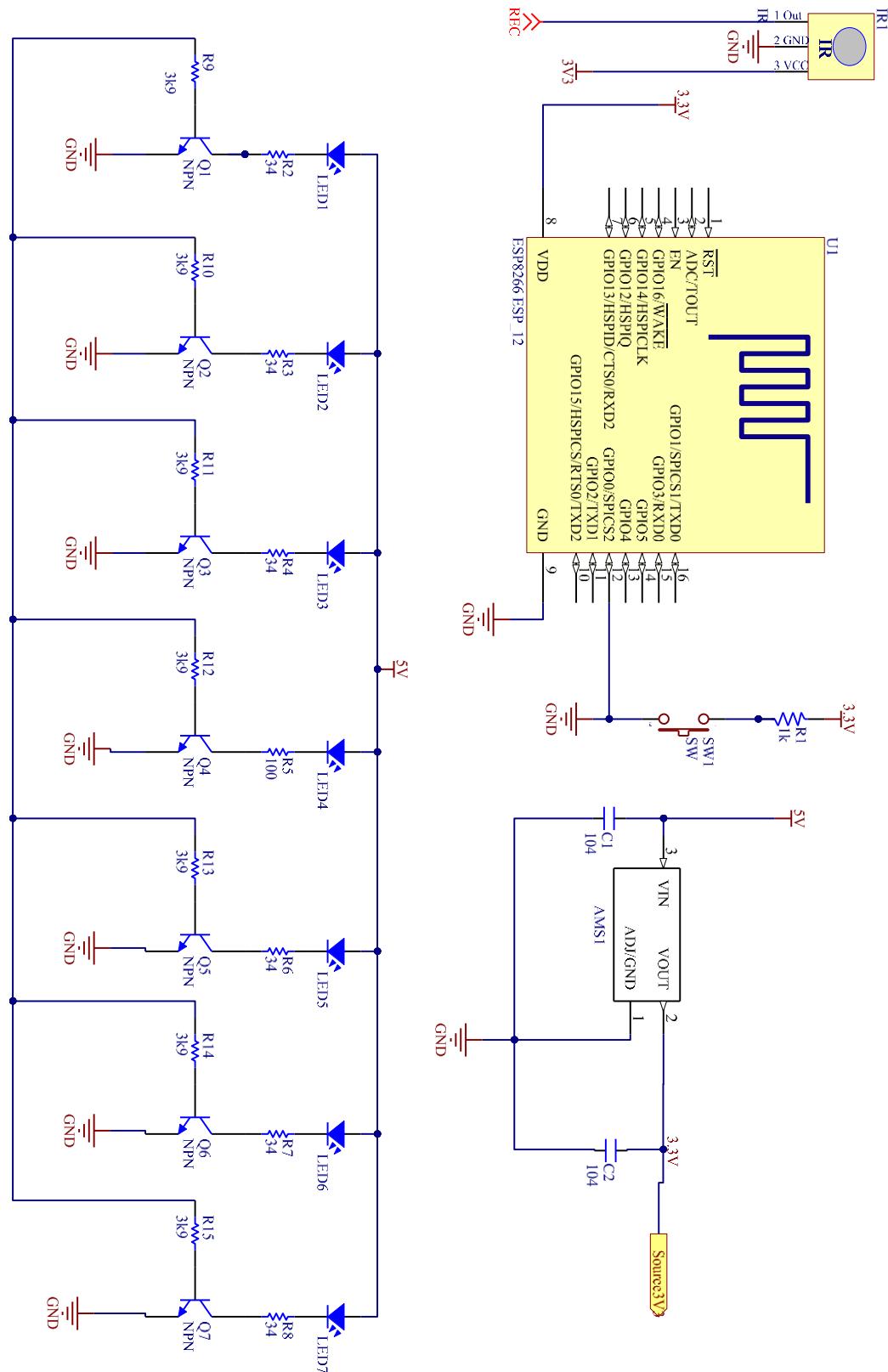
## Chương 6 Phụ lục

### 6.1 Sơ đồ nguyên lý mạch HomeServer và Camera chuông cửa



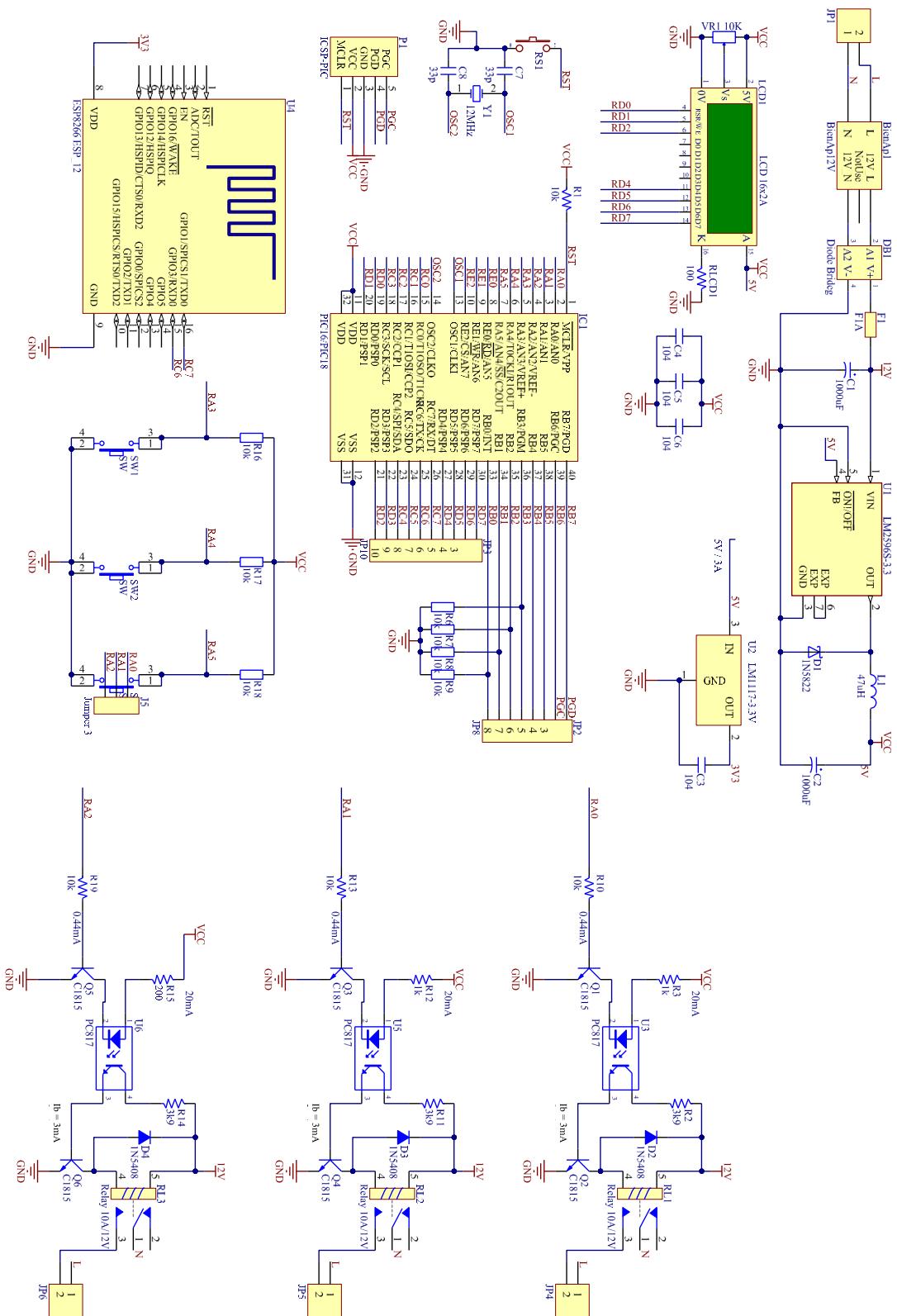
Hình 6-1 Sơ đồ nguyên lý mạch HomeServer và camera chuông cửa

## 6.2 Sơ đồ nguyên lý mạch điều khiển hồng ngoại



Hình 6-2 Sơ đồ nguyên lý mạch điều khiển hồng ngoại

### 6.3 Sơ đồ nguyên lý mạch điều khiển thiết bị bằng Relay



**Hình 6-3 Sơ đồ nguyên lý mạch điều khiển thiết bị bằng hồng ngoại**

## 6.4 Sourecode WebServer

Code được chia nhiều File khác nhau

### main.py

```
import threading
import Socket_Listener
from flask import Flask, render_template, send_from_directory
from flask_socketio import SocketIO
import json as JSON
import JsonCMD
import time
JsonCmd = JsonCMD.JsonCMD()
app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret!'
socketio = SocketIO(app)
SocketListener = Socket_Listener.socket_server_threading(1, "Thread-local")#@SOCKET_LISTEN@#
socket =Socket_Listener.SocketSender()
def ESP_temp.asking():
    while True:
        temperature=Socket_Listener.tcpClientResponse("{ \"req\": \"temp\" }\r\n".encode())
        print(temperature)
        # temperature = temperature.decode()
        # temperature = JSON.loads(temperature)['data']
        # socketio.emit('Client_Listen_Socket',{'server':'temperature', 'data':temperature})
        time.sleep(3)

def ListenSocket():
    while True:
        data = SocketListener.webQueue.get(block=True)
        print(f'ListenSocket working on: {data}')
        socketio.emit('Client_Listen_Socket',{'server':'test', 'data':'test_socket_listen'})
        SocketListener.webQueue.task_done()

@socketio.on('Client_Listen_Socket')
def testSocketIO(json, methods=['GET', 'POST']):
    print("[Client_Listen_Socket]", json)
    jres = {'req':'hello'}
    socketio.emit('Client_Listen_Socket', jres)

@socketio.on('Server_Listen_Socket')
def Server_Socket_IO(json, methods=['GET', 'POST']):
    ...
```

```
{'client': 'click', 'web_id': 'li_hex_2'}
- parse
- load device data
- send
{client: change_structure, data:[]}
'''

print("[Server_Listen_Socket]", json)
JsonCmd.load(json)
JsonCmd.execute()
feedback = JsonCmd.get_feedback()
print(feedback)
try:
    if feedback['jsoncmd'] == "tcp":
        print(feedback['data'])
        Socket_Listener.tcpClient((feedback['data']+ '\r').encode())
except:
    pass
if feedback != "":
    socketio.emit('Client_Listen_Socket', feedback)

@app.route('/NetworkConfig')
def NetworkConfig():
    return render_template('NetworkConfig.html')

@app.route('/JanusCamera')
def JanusCamera():
    DATA = {'username': 'admin', 'password': '123456', 'uid': '000'} #Debug
    return render_template('MyCamera_01.html', data=DATA)

@app.route('/remoteTV')
def remoteTV():
    return render_template('TV_remote.html')
@app.route('/sample')
def sample():
    return render_template('sample.html')
@app.route('/advance')
def advance():
    return render_template('advance.html')
@app.route('/vision_test')
def vision_test():
    return render_template('vision_test_02.html')
@app.route('/models/<path:filename>', methods=['GET', 'POST'])
def models(filename):
    path = "./static/javascript/vision/models"
    return send_from_directory(directory=path, filename=filename)

@app.route('/')

```

```

def hello_world():
    file = open("./file/page.json", 'r')
    file_data = file.read()
    PAGE_JSON = JSON.loads(file_data)
    return render_template(' MainPage_04.html', data=PAGE_JSON)

if __name__ == '__main__':
    # With SOCKET_LISTENER
    # SocketListener.start()#@SOCKET_LISTEN@#
    # threading.Thread(target=ListenSocket, daemon=True).start()#@SOCKET_LISTEN@#
    # socketio.run(app, host="0.0.0.0")#@SOCKET_LISTEN@#

    # Front-End
    socketio.run(app, debug=True)
    #socketio.run(app, host="0.0.0.0")

    # Others
    # threading.Thread(target=ESP_temp.asking, daemon=True).start()
    # app.run(host="0.0.0.0", port=7000)
    # app.run(debug=True)

```

## MainPage\_04.html

```

<!doctype html>
<html>

<head>
    <meta charset="utf-8">
    <title>hexagon</title>
    <!-- Data from server and constant -->
    <script type="text/javascript">
        var page_json = {{ data| tojson }};
        console.log(page_json);
    </script>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="Cache-Control" content="no-cache, no-store, must-
revalidate" />
    <meta http-equiv="Pragma" content="no-cache" />
    <meta http-equiv="Expires" content="0" />
    <!-- <script>
        var page_init_json =
    </script> -->
    <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='cs
s/rotate_hexagon_01.css')}}">
    <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='cs
s/hexagons_01.css')}}">

```

```
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/button_01.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/navigation_menu.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/power_button.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/general.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/menu.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/top_title.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/pop_up.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/responsive.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/scroll_menu.css')}}">
<link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/global.css')}}">
<link rel='stylesheet' href="{{ url_for('static', filename='webfonts/css/all.css')}}">
<link rel='stylesheet' href="{{ url_for('static', filename='webfonts/css/fontawesome.css')}}">
<link rel='stylesheet' href="{{ url_for('static', filename='webfonts/css/fontawesome.min.css')}}">

<link href='https://fonts.googleapis.com/css?family=Raleway:300' rel='stylesheet' type='text/css'>
<script type="text/javascript" src="{{ url_for('static', filename = 'javascript/jquery-3.4.1.js')}}"></script>
<script type="text/javascript" src="{{ url_for('static', filename = 'javascript/control_button.js')}}"></script>
<script type="text/javascript" src="{{ url_for('static', filename = 'javascript/control_hexagons.js')}}"></script>
<script type="text/javascript" src="{{ url_for('static', filename = 'javascript/power_button.js')}}"></script>
<script type="text/javascript" src="{{ url_for('static', filename = 'javascript/drag_drop.js')}}"></script>
<script type="text/javascript" src="{{ url_for('static', filename = 'javascript/scroll_to_appear_menu.js')}}"></script>
<script type="text/javascript" src="{{ url_for('static', filename = 'javascript/weather_popup.js')}}"></script>
<script type="text/javascript"
src="{{ url_for('static', filename = 'javascript/init_page_from_json.js')}}"></script>
```

```
<!-- SocketIO Library -->
<script src="{{ url_for('static', filename='websrc/jquery.min.js') }}"></script>
<script src="{{ url_for('static', filename='websrc/socket.io.min.js') }}"></script>
<script type="text/javascript" src="{{ url_for('static', filename = 'javascript/socket_io_handler.js') }}"></script>
</head>

<body class="hex-body">
<!-- ScrollMenu
<div id="scroll-menu">
    <a><i class="fa fa-home" style="text-align: center; "></i></a>
    <a><i class="fa fa-home" style="text-align: center; "></i></a>
    <a><i class="fa fa-home" style="text-align: center; "></i></a>
</div> -->
<div class="modal" onclick="close_choose_device()"></div>
<div class="menu-create-device">
    <h1 style="margin-left: 40%;">Create Device Menu</h1>
    <form action="javascript:void(0);">
        <div class="row">
            <div class="col-25">
                <label for="fname">Device Name</label>
            </div>
            <div class="col-75">
                <textarea type="text" id="fname" name="firstname" placeholder="Name of device"></textarea>
            </div>
        </div>
        <div class="row">
            <div class="col-25">
                <label for="lname">More Info</label>
            </div>
            <div class="col-75">
                <textarea type="text" id="lname" name="lastname" placeholder="More detail"></textarea>
            </div>
        </div>
        <div class="row">
            <div class="col-25">
                <label for="choose_device_btn">Choose Device</label>
            </div>
            <div class="col-75">
                <select id="choose_device_btn" name="choose_device_btn">
                    <option value="IR001">IR001</option>
                    <option value="SW123">SW123</option>
                    <option value="SW002">SW002</option>
                </select>
            </div>
        </div>
    </form>
</div>
```

```
        </div>
    </div>
    <div class="row">
        <div class="col-25">
            <label for="background">Background</label>
        </div>
        <div class="col-75">
            <textarea id="background" name="background"
                placeholder="Background color code (#) or image($)"></tex
tarea>
        </div>
    </div>
    <div class="row">
        <div class="col-25">
            <label for="subject">Note</label>
        </div>
        <div class="col-75">
            <textarea id="subject" name="subject" placeholder="Write some
thing.." style="height:200px"></textarea>
        </div>
    </div>
    <div class="row">
        <input type="submit" value="Create" onclick="click_submit_create_
device(event)">
    </div>
</form>
</div>

<div class="device-manager">
    <h1 style="margin-left: 40%;">Device manager</h1>
    <form action="javascript:void(0);">
        <div class="row">
            <div class="col-25">
                <label for="choose_device_btn">Choose Device</label>
            </div>
            <div class="col-75">
                <select id="devMa-selectDevice" name="devMa-selectDevice">
                    <option value="IR001">IR001</option>
                    <option value="SW123">SW123</option>
                    <option value="SW002">SW002</option>
                    <option value="NEW">New device</option>
                </select>
            </div>
        </div>
        <div class="row">
            <div class="col-25">
                <label for="lname">More Info</label>
            </div>
```

```
</div>
<div class="col-75">
    <textarea type="text" id="lname" name="lastname" placeholder="More detail"></textarea>
    </div>
</div>
<div class="row">
    <div class="col-25">
        <label for="network">Network</label>
    </div>
    <div class="col-371">
        <textarea id="network" name="network" placeholder="Ssid"></textarea>
    </div>
    <div class="col-37r">
        <textarea id="network" name="network" placeholder="Password"></textarea>
    </div>
    <div class="small">
        <small>Advance</small>
    </div>
</div>
<div class="row">
    <div class="col-25">
    </div>
    <div class="col-25">
        <input type="checkbox"><label for="smart-config">Smart config</label>
    </div>
    <div class="col-25">
        <input type="checkbox"><label for="static-ip">Static IP</label>
    </div>
    <div class="col-25">
        <input type="checkbox"><label for="domain">Enter code</label>
    </div>
</div>
<div class="row">
    <div class="col-25">
        <label for="subject">Note</label>
    </div>
    <div class="col-75">
        <textarea id="subject" name="subject" placeholder="Write something.." style="height:200px"></textarea>
    </div>
</div>
<div class="row">
```

```
        <input type="submit" onclick="submit_device_manger()" value="Connect to device" onclick="">
    </div>
</form>
</div>

<div class="top-panel">
    <!-- <div class="top-logo"></div>
    <div class="top-navi-icon-1 top-login-icon">
        <h1><i class="fa fa-home"></i></h1>
    </div>
    <div class="top-navi-icon-2 top-login-icon">
        <h1><i class="fa fa-search"></i></h1>
    </div>
    <h1 class="top-title">Device Web Page</h1>
    <div class="top-login">
        <h1><i class="fa fa-paper-plane top-login-icon"></i></h1>
    </div> -->
    <div style="cursor: pointer;" onclick="location.reload()"></div>
    <!-- <div>
        <h1><i class="fa fa-home" style="text-align: center; margin-top: 15px;"></i></h1>
    </div>
    <div>
        <h1><i class="fa fa-phone" style="text-align: center; margin-top: 15px;"></i></h1>
    </div>
    <div>
        <h1><i class="fa fa-search" style="text-align: center; margin-top: 15px;"></i></h1>
    </div> -->
    <div>
        <h1 class="top-title" style="text-align: center; margin-top: 15px;">Dissertation</h1>
    </div>
    <div style="text-align: center;">
        <div style="float: right; padding-left: 5px; padding-right: 5px; margin-top: 15px;">
            <h1><i class="fa fa-paper-plane top-login-icon"></i></h1>
        </div>
        <div style="float: right; margin-top: 30px; text-align: center;">
            <a style="border: 3px solid black;"><b>The Beatles</b></a>
        </div>
    </div>
</div>
```

```
<div class="hexDiv">
    <ul id="hexGrid">

        </ul>
    </div>
    <div id="nav-myNav" class="nav-overlay">
        <a href="javascript:void(0)" class="nav-
closebtn" onclick="closeNav()">&times;</a>
        <div class="nav-overlay-content">
            <a onclick="closeNav();button_adding_bottom()">Web view manager</a>
            <a onclick="closeNav();device_manager()">Device manager</a>
            <a href="#">Security</a>
            <a href="/advance">Advance</a>
        </div>
    </div>

    <div class="rotateHexagon-wrapper custom-rotateHexagon" id="scroll-
symbol" style="cursor: pointer;" ondblclick="weather_popup(event)">
        <div class="rotateHexagon-hexagon">
            <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 28 32" fill="cur-
rentcolor">
                <path d="M0 8 L0 24 L14 32 L28 24 L28 8 L14 0 Z" />
            </svg>
        </div>
    </div>
    <div class="weather-popup" id="weather-popup" style="display: none;">
        <i class="fas fa-cloud-rain"></i><a> Ho Chi Minh city</a>
        <br>
        <h1>29°C</h1>
    </div>

    <div class="bottom-
menu" style="position: fixed; bottom: 5px; right:5px; padding: 10px 6px">
        <!-- <div style="display: grid; grid-template-columns: repeat(2, 1fr);">
            <div>
                <button class="button-adding-
bottom" onclick="button_adding_bottom()"><i
                    class="fa fa-plus"></i></button>
            </div>
            <div>
                <button class="button-minus-
bottom" onclick="button_minus_bottom()"><i class="fa fa-minus"></i></button>
            </div>
        </div>
    <div>
```

```

        <span style="font-
size:30px;cursor:pointer" onclick="openNav()">#9776; open</span>
    </div> -->
    <div style="font-size:30px;cursor:pointer" onclick="openNav()">
        <i class="fa fa-cog" style="font-size:48px;"></i>
    </div>
</div>
<div class="popup-apply">
    <div>
        <p><b>Do you want to apply?</b></p>
    </div>
    <div class="popup-icon-devide">
        <div>
            <button class="button-adding-bottom" onclick="apply_popup()"><i
                class="fa fa-check"></i></button>
        </div>
        <div>
            <button class="button-minus-
bottom" onclick="close_popup()"><i class="fa fa-trash-alt "></i></button>
        </div>
    </div>
</div>

<div id="images-template" style="display:none">
    
    
    
    
    
    
    
    
</div>
<div id="sample">
    <!-- The following object for sample drag and drop to copy -
no id added -->
    <li class="hex" id="empty-sample-hexagon">
        <div class="hexIn" id='empty-and-cant-be-
used' ondrop="drop(event)" ondragover="allowDrop(event)">

```

```

        draggable="true" ondragstart="drag(event)">
        <a class="hexLink">
        </a>
    </div>
</li>
</div>
<div id="ram" style="display:none">
    <div id="ram_icon_1"></div>
    <div id="ram_icon_2"></div>
</div>
<script>
    init_page();
    ($('window').width() < 600)?$('li#skip').remove():"";

    dragElement(document.getElementById("scroll-symbol"));
</script>
<script>//init
    edit_web_object()
</script>
</body>

</html>

```

## Network\_Config.html

```

<head>
    <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/general.css')}}">
    <link rel="stylesheet" type="text/css" href="{{url_for('static', filename='css/global.css')}}">
    <style>
        h1,h2,h3{
            color: darkblue;
            margin-bottom: 10px;
            margin-top: 15px;
        }
        input {
            margin-bottom: 10px;
        }
    </style>
</head>
<body style="margin-left: 50px;">
    <h1>Cài đặt mạng</h1>
    <i>Bạn đang ngoại tuyến. Hãy kết nối cảm biến bằng dây mạng hoặc thiết lập mật khẩu Wifi</i>
    <div style="border-style: solid; width: 50%; padding-left: 20px;">
        <h2>Wireless</h2>
        <div style="display: grid; grid-template-columns: 1fr 3fr 0.5fr;">

```

```

        <label for="SSID" style="">SSID</label>
        <input type="text" name="ssid" placeholder="Nhập SSID">
    </div>
    <br>
    <div style="display: grid; grid-template-columns: 1fr 3fr 0.5fr;">
        <label for="PSK" style="">Mật khẩu</label>
        <input type="text" name="psk" placeholder="Nhập mật khẩu">
    </div>
    <br>

    <div style="display: grid; grid-template-columns: 1fr 3fr 0.5fr;">
        <label for="hostname" style="">mDNS</label>
        <input type="text" name="hostname" placeholder="Nhập tên miền mDNS">
    </div>
</div>
</div>
<div style="border-style: solid; width: 50%; margin-top: 20px; padding-left: 20px;">
    <h2>Cấu hình Cloud</h2>
    <div style="display: grid; grid-template-columns: 1fr 3fr 0.5fr;">
        <label for="cloud_address">Tên miền</label>
        <input type="text" name="cloud_address" placeholder="Nhập tên miền cloud hoặc địa chỉ IP">
    </div>

</div>
<div>

</div>
</body>
```

**TV\_remote.html**

```

<!DOCTYPE html>
<html>

<head>
    <title>RemoteTV</title>
    <link rel='stylesheet' href="{{ url_for('static', filename='css/tv_remote.css') }}">
</head>

<body>
    <a style="cursor: pointer; float: left; position: fixed; href="/"></a>
    <div style="padding-top: 20%;"></div>
    <div class="TV-remote">
        <div class="TV-btn-div"><button class="TV-btn">mute</button></div>
```

```
<div class="TV-btn-div"></div>
<div class="TV-btn-div"><button class="TV-btn">power</button></div>

<div class="TV-btn-div"><button class="TV-btn">1</button></div>
<div class="TV-btn-div"><button class="TV-btn">2</button></div>
<div class="TV-btn-div"><button class="TV-btn">3</button></div>
<div class="TV-btn-div"><button class="TV-btn">4</button></div>
<div class="TV-btn-div"><button class="TV-btn">5</button></div>
<div class="TV-btn-div"><button class="TV-btn">6</button></div>
<div class="TV-btn-div"><button class="TV-btn">7</button></div>
<div class="TV-btn-div"><button class="TV-btn">8</button></div>
<div class="TV-btn-div"><button class="TV-btn">9</button></div>
<div class="TV-btn-div"></div>
<div class="TV-btn-div"><button class="TV-btn">0</button></div>
<div class="TV-btn-div"></div>
<!-- <div class="TV-direction-A1"></div>
<div class="TV-direction-A2"></div>
<div class="TV-direction-A3"></div>

<div class="TV-direction-B1"></div>
<div class="TV-direction-B1"></div>
<div class="TV-direction-B2"></div>

<div class="TV-direction-C1"></div>
<div class="TV-direction-C1"></div>
<div class="TV-direction-C2"></div> -->

</div>
<div class="TV-remote-circle">
    <div class="TV-direction"><button class="TV-direction-A-btn">CH+</button></div>
    <div class="TV-direction"><button class="TV-direction-A-btn">CH+</button></div>

        <div class="TV-direction"><button class="TV-direction-B-btn">CH+</button></div>
        <div class="TV-direction"><button class="TV-direction-B-btn">CH+</button></div>

    </div>
</div>

</body>

</html>
```

**control\_button.js**

```
function edit_web_object(){
    $('i.fa-edit').on("click", function(){
        alert($(this).closest('li').html())
    })
}

function button_adding_bottom() {
    var modal = document.getElementsByClassName("modal");
    modal[0].style.display = 'block';
    var menu = document.getElementsByClassName("menu-create-device");
    menu[0].style.display = 'block';
};

function device_manager() {
    var modal = document.getElementsByClassName("modal");
    modal[0].style.display = 'block';
    var menu = document.getElementsByClassName("device-manager");
    menu[0].style.display = 'block';
};

function close_choose_device() {
    console.log("Close choose device")
    var modal = document.getElementsByClassName("modal");
    modal[0].style.display = 'none';
    var menu = document.getElementsByClassName("menu-create-device");
    menu[0].style.display = 'none';
    var menu2 = document.getElementsByClassName("device-manager");
    menu2[0].style.display = 'none';
};

function create_blank_hexagon() {
    var get_grids = document.getElementById("hexGrid");
    var number_of_grid = get_grids.getElementsByTagName("div").length;
    console.log(number_of_grid);
    console.log("button_adding_bottom");
    var hexGrid = document.getElementById("hexGrid");
}
```

```

var _hex = document.createElement("li");
_hex.className = "hex";
var hexIn = document.createElement("div");
hexIn.className = "hexIn"
var hexLink = document.createElement("a");
hexLink.className = "hexLink"
hexLink.style.backgroundColor = "red";
hexIn.append(hexLink);
_hex.append(hexIn);
hexGrid.append(_hex);

}

function openNav() {
    document.getElementById("nav-myNav").style.width = "100%";
}

function closeNav() {
    document.getElementById("nav-myNav").style.width = "0%";
}

```

**control\_hexagon.js**

```

function click_on_hexagon(event) {
    if (event.target.className=="fa fa-edit"){
        return;
    }
    var get_grid = event.target.parentNode;
    var get_img_object = get_grid.getElementsByTagName("img")[0]
    if (get_img_object.style.opacity === "0.2") {
        get_img_object.style.opacity = "1";
    }
    else {
        get_img_object.style.opacity = "0.2";
    }
}

```

**drag\_drop.js**

```

function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {

```

```
if (ev.target === null || ev.target.className != 'hexIn') {
    return;
}
ev.target.id = 'moving_element';
ev.dataTransfer.setData("text", ev.target.id);
// create dragable object

}

function drop(ev) {
// if (ev.target.id === 'empty-and-cant-be-used') {
//   console.log('empty-and-cant-be-used');
//   return;
// }
ev.preventDefault();

var data = ev.dataTransfer.getData("text");
// ev.target.appendChild(document.getElementById(data));
//Some config:
var incoming_element = document.getElementById(data);
var outgoing_element = ev.target.parentNode;

var old_2nd_id = outgoing_element.parentNode.id;
// if (incoming_element== null || incoming_element.className != 'hexIn' || outgoing_element.className != 'hexIn'){
//   return;
// }
if (incoming_element === null || incoming_element.className != 'hexIn' || outgoing_element.className != 'hexIn') {
    return;
}
var old_1st_id = incoming_element.parentNode.id;
var cp_outgoing_element = outgoing_element.cloneNode(true);
$("#moving_element").removeAttr("id");
ev.target.parentNode.append(incoming_element);
document.getElementById(old_1st_id).appendChild(cp_outgoing_element);
outgoing_element.remove()

incoming_element.parentNode.id = "tmp_id_1st";
cp_outgoing_element.parentNode.id = "tmp_id_2nd";
$('#tmp_id_1st')[0].id = old_1st_id;
$('#tmp_id_2nd')[0].id = old_2nd_id;
show_apply_popup();
}

function show_apply_popup(){
var popup = document.getElementsByClassName('popup-apply')[0];
if (popup==undefined){
    return;
}
```

```
}

popup.classList.remove('popup-apply');
popup.classList.add('popup-apply-active');

}

function apply_popup(){
    var popup = document.getElementsByClassName('popup-apply-active')[0];
    if (popup.className=="popup-apply-active")
    {
        popup.classList.remove('popup-apply-active');
        popup.classList.add('popup-apply');
    }
    var list = $('#hexGrid').children();
    var ret = [];
    for (var a=0; a<list.length; a++){
        ret[a] = list[a].id;
    }
    emitSocket("change_structure", ret);
    //location.reload()
}

function close_popup(){
    var popup = document.getElementsByClassName('popup-apply-active')[0];
    if (popup.className=="popup-apply-active")
    {
        popup.classList.remove('popup-apply-active');
        popup.classList.add('popup-apply');
    }
    location.reload()
}

function dragElement(elmnt) {
    var pos1 = 0, pos2 = 0, pos3 = 0, pos4 = 0;
    if (document.getElementById(elmnt.id + "header")) {
        /* if present, the header is where you move the DIV from:*/
        document.getElementById(elmnt.id + "header").onmousedown = dragMouseDown;
    } else {
        /* otherwise, move the DIV from anywhere inside the DIV:*/
        elmnt.onmousedown = dragMouseDown;
    }
}

function dragMouseDown(e) {
    e = e || window.event;
    e.preventDefault();
    // get the mouse cursor position at startup:
    pos3 = e.clientX;
    pos4 = e.clientY;
    document.onmouseup = closeDragElement;
    // call a function whenever the cursor moves:
    document.onmousemove = elementDrag;
}
```

```

}

function elementDrag(e) {
    e = e || window.event;
    e.preventDefault();
    // calculate the new cursor position:
    pos1 = pos3 - e.clientX;
    pos2 = pos4 - e.clientY;
    pos3 = e.clientX;
    pos4 = e.clientY;
    // set the element's new position:
    elmnt.style.top = (elmnt.offsetTop - pos2) + "px";
    elmnt.style.left = (elmnt.offsetLeft - pos1) + "px";
}

function closeDragElement() {
    /* stop moving when mouse button is released:*/
    document.onmouseup = null;
    document.onmousemove = null;
}
}

```

**init\_page\_from\_json.js**

```

function init_page(data) {
    // var object = get_empty_object("hex_li_5");
    // // object = modify_object(object, 'image', 'Charizard_01.png', '', '', '');
    ;
    // object = modify_object(object, 'empty', '', 'yellow', '', '');
    // place_object(object);
    // // clean templates;
    // // delete_images_template();

    // var json_obj = '{"number_of_hex": 2,"data": [{"hexagon-li-0": {"type": "image/button/empty","background_color": "", "image_name": "", "hover": {"h_text": "", "p_text": ""}}, "button": {"type": "power/add"}]}, {"hexagon-li-1": {"type": "image", "image_name": "Gyarados_01.png", "hover": {"is_hover": true, "h_text": "", "p_text": ""}}, {"hexagon-li-2": {"type": "empty", "background_color": ""}}]}';
    // var json_obj = '{"number_of_hex": 1,"data": [{"hexagon-li-0": {"type": "empty", "background_color": "red"}}]}';
    // console.log(json_obj);
    // json_obj = JSON.parse(json_obj);

    var json_obj = window.page_json;
    console.log(json_obj)
    ExeceilJsonData(json_obj);
}

```

```

function get_empty_object(id_name) {
    // Version 1: get in own page _Work well
    var output_object = document.getElementById("empty-sample-
hexagon").cloneNode(true);
    output_object.id = id_name;
    // // Version 2: get in external Page
    // var output_object_0 = get_external_src('#empty-sample-hexagon')
    // var output_object = document.getElementById("external_src");
    // console.log("External Src 0:");
    // console.log(output_object);

    return output_object;
}
function modify_object(object, type, image = '', bg_color = '', p_text = '', h_te
xt = '', href) {
    console.log('      Create: ', type, image, bg_color, p_text, h_text, href);
    var href_data="";
    if (href==undefined || href==""){
        }else{
            href_data=' location.replace("'" +href+ "'");';
        }

    if (type === "image") {
        //image object
        // Version 1: Fail when create object due to Flask didn't load
        // Solution: Should be another reachable link to image
        // var image_object = document.createElement('img');

        // var src = document.createAttribute('src');
        // // src.value = "{{url_for('static', filename='images/" +image+ "')}}"
        // src.value = "/static/images/" +image+ ")"
        // image_object.setAttributeNode(src);

        // var alt = document.createAttribute('alt');
        // alt.value = "";
        // image_object.setAttributeNode(alt);

        //Version 2: Get from photo template
        var image_id = ['Charizard_01.png', 'Gyarados_01.png', 'Living_Room_01.jpg
', 'Bed_Room_01.jpg', 'TV_remote_01.jpg', 'Chat_01.png', 'Camera_01.jpeg', 'Lock_
01.png'];
        var image_index = image_id.indexOf(image)
        var image_object = document.getElementById('images-
template').getElementsByTagName('img')[image_index].cloneNode();
        object.getElementsByTagName('a')[0].appendChild(image_object)
    }
}

```

```

//click event
var onclick = document.createAttribute('onclick');
onclick.value = "click_on_hexagon(event); hex_click_event(event);"+href_data;
object.getElementsByTagName('a')[0].setAttributeNode(onclick);
}
else if (type=="skip"){}
else{
    var onclick = document.createAttribute('onclick');
    onclick.value = "hex_click_event(event);"+href_data;
    object.getElementsByTagName('a')[0].setAttributeNode(onclick);
}

if (p_text != '') {
    var p_object = document.createElement('p');
    p_object.innerHTML = p_text;
    object.getElementsByTagName('a')[0].appendChild(p_object);
}
if (h_text != '') {
    var h_object = document.createElement('h1');
    h_object.innerHTML = h_text +"></i>";
    object.getElementsByTagName('a')[0].appendChild(h_object);
    addDragDropAttr(object.getElementsByTagName('div')[0], "drag");
}
else {
    addDragDropAttr(object.getElementsByTagName('div')[0], "drop");
}
if (bg_color != '') {
    object.getElementsByTagName('a')[0].style.backgroundColor = bg_color;
}
return object;
}
function place_object(object) {
    var place_object = document.getElementById('hexGrid');
    object.getElementsByClassName('hexIn')[0].removeAttribute('id');
    place_object.appendChild(object);

}
function delete_images_template() {
    document.getElementById('images-template').remove();
}

function get_external_src(object_id) {
    // #images-template #empty-sample-hexagon
    var ex_src = document.createElement('div');
    ex_src.id = 'external_src';

```

```
var sample = document.getElementById('sample');
sample.appendChild(ex_src);

jQuery('#external_src').load('sample #empty-sample-hexagon');
var external_src = document.getElementById("external_src");
// external_src.remove();
return external_src;
}

function addDragDropAttr(object, type) {
    if (type == "drag") {
        object.removeAttribute("ondrop");
        object.removeAttribute("ondragover");
        // var new_drag = document.createAttribute("draggable");
        // var new_on_drag = document.createAttribute("ondragstart");
        // new_drag.value = "true";
        // new_on_drag.value = "drag(event)";
        // object.setAttributeNode(new_drag);
        // object.setAttributeNode(new_on_drag);
    } else {
        object.removeAttribute("draggable");
        object.removeAttribute("ondragstart");
    }
}

function ExecuteJsonData(json_obj) {
    // console.log("Get Json")
    // console.log(json_obj)
    var data = json_obj.data;
    var structure = json_obj.structure;
    var _length = structure.length;
    for (var x = 0; x < _length; x++) {
        id = structure[x];
        console.log('>>> ', id, ' <<<');
        // console.log(data[i]);
        var content = data[id];
        // console.log("Content");
        // console.log(content);
        // console.log([content.type, content.image_name], content.background_color, content.hover.p_text, content.hover.h_text]);
        var object = get_empty_object(id);
        try {
            var type = content.type;
        }
        catch (err) {
            var type = '';
        }
    }
}
```

```
        }
        try {
            var image_name = content.image_name;
        }
        catch (err) {
            var image_name = '';
        }
        try {
            var background_color = content.background_color;
        }
        catch (err) {
            var background_color = '';
        }
        try {
            var p_text = content.hover.p_text;
        }
        catch (err) {
            var p_text = '';
        }
        try {
            var h_text = content.hover.h_text;
        }
        catch (err) {
            var h_text = '';
        }
        href = content.href;
        modify_object(object, type, image_name, background_color, p_text, h_text,
        href);
        place_object(object);
    }
}

/* This is an empty object with dragable
<li class="hex" id="empty-sample-
hexagon" ondrop="drop(event)" ondragover="allowDrop(event)">
<div class="hexIn" id='empty-and-cant-be-used'>
    <a class="hexLink" style="background-color: aqua;">
    </a>
</div>
</li> */

/* <li class="hex" id='hexagon-li-0'>
<div class="hexIn" >
    <a class="hexLink" onclick="click_on_hexagon(event)">
        
        <h1>Device 1</h1>
        <p>Living room light<br></p>
```

```
</a>
</div>
</li> */}
```

### socket\_io\_handler.js

```
var socket = io.connect('http://' + document.domain + ':' + location.port);
socket.on('connect', function () {
    // socket.emit('client_connect', {
    //     req: 'LED'
    // })
    socket.emit('Client_Listen_Socket', {
        "client": "Hello"
    })
    console.log("[SocketIO] New Connection...")
})
socket.on('Client_Listen_Socket', function (_msg) {
    page_list={
        "TV_remote":"/remoteTV",
        "Janus_camera":"/JanusCamera"
    }
    console.log(_msg);
    // _msg = JSON.parse(_msg);
    var server = _msg.server;
    if (server=="reload_page"){
        location.reload()
    }else if(server == "load_page"){
        location.replace(page_list[_msg.page]);
    }else if(server == "test"){
        alert(_msg.data);
    }else if (server == "temperature"){
        alert(_msg.data)
    }
})
function hex_click_event(event) {
    // socket.emit('Socket_Listen_Socket',{
    //     "client":"ClickHex"
    // })
    var parent_node_id = event.target;
    if (parent_node_id.className == "fa fa-edit"){
        return
    }
    var x = 0;
    var is_err = true;
    for (x; x < 8; x++) {
        if (parent_node_id.tagName == 'LI') {
            parent_node_id = parent_node_id.id;
```

```

        is_err = false;
        break;
    } else {
        parent_node_id = parent_node_id.parentNode;
        continue;
    }
}
if (is_err) {
    console.error("[Click Err] No LI tagname Found!")
} else {
    var status;
    var get_grid = event.target.parentNode;
    var get_img_object = get_grid.getElementsByTagName("img")[0]
    if (get_img_object.style.opacity === "0.2") {
        status="OFF";
    }
    else {
        status="ON";
    }
    console.log("ClickHex --> ", parent_node_id, " status -->", status);
    socket.emit('Server_Listen_Socket', {
        "client": "click",
        "web_id": parent_node_id,
        "status": status
    })
}
}

function supply_id(){
    var date = new Date();
    var time_id = date.getTime();
    return time_id;
}
function click_submit_create_device(e) {
    e.preventDefault()
    console.log('Click Submit');
    var h_text = $('#fname').val();
    var p_text = $('#lname').val();
    var more = $('#subject').val();
    var device_id = $('#choose_device_btn').val();
    var web_id = document.getElementsByClassName("hex");
    var background = $('#background').val();
    var _type = "empty";
    var image_name = '';
    if (background.indexOf("$") == 0) {
        _type = "image";
        image_name = background.slice(1);
        background = '';
    }
}

```

```

else {
    background = background.toLowerCase();
}

id = supply_id();
web_id = "li_hex_" + id;
console.log("hexagon", web_id);
socket.emit('Server_Listen_Socket', {
    client: "new_web_object",
    web_id: web_id,
    data: {
        type: _type,
        background_color: background,
        image_name : image_name,
        hover: {
            h_text: h_text,
            p_text: p_text
        },
        device_id: device_id,
        more: more
    }
})
}

function emitSocket(key, data){
    // '''structure'''
    socket.emit('Server_Listen_Socket', {
        client: "change_structure",
        data: data
    })
}

function submit_device_manger(e){
    var device_type;
    var device_name;
    var more;
    var note;
    var network;
}

```

**weather\_popup.js**

```

function weather_popup(e){
    var default_top = 100;
    var default_left=100;
    var top = e.target.offsetTop;
    var left = e.target.offsetLeft;

```

```

// $('#ram_icon_1').html(top);
// $('#ram_icon_2').html(left);
// $('#scroll-
symbol').css({"top":default_top+"px", "left":default_left+"px"})
$('#weather-
popup').css({"top":e.target.offsetTop+"px", "left":e.target.offsetLeft+"px"})
$('#weather-popup').show()
// setTimeout(function(){ $('#weather-popup').hide()},5000)
}

```

**web\_socket.js**

```

var socket = new WebSocket('ws://localhost:8080/');
socket.onopen = function () {
    console.log('[Websocket] Connected!');
};
socket.onmessage = function (event) {
    console.log('[Websocket] Received data: ' + event.data);
    socket.close();
};
socket.onclose = function () {
    console.log('[Websocket] Lost connection!');
};
socket.onerror = function () {
    console.log('[Websocket] Error!');
};
socket.send('NEW CLIENT CONNECT');

```

**Socket\_Listener.py****socketIO\_01.py****TV\_remote.py****6.5 Sourcecode Mạch Camera và mạch HomeServer****DoorCam.py**

```

import signal
import File
import sys
import RPi.GPIO as GPIO
import Constant
import threading
import Key
import subprocess
import Timer
import SocketClient
import Network

```

```
import time
from SocketClient import socket as socket
import json
import Calendar
import myGPIO
import Timer

Calendar = Calendar.Calendar()
GPIO.setmode(GPIO.BCM)
GPIO.setup(Constant.LEDBLUE_PIN,GPIO.OUT)
GPIO.setup(Constant.LEDRED_PIN,GPIO.OUT)
GPIO.setup(Constant.LEDGREEN_PIN,GPIO.OUT)
##HIDDEN##GPIO.setup(PIN_FINGERPRINT,GPIO.IN)
# GPIO.setup(BELL_PIN,GPIO.IN,GPIO.PUD_DOWN)
GPIO.setup(Constant.BELL_PIN,GPIO.IN)
GPIO.setup(Constant.RESET_PIN,GPIO.IN,GPIO.PUD_UP)
GPIO.output(Constant.LEDBLUE_PIN, False)
GPIO.output(Constant.LEDRED_PIN, False)
GPIO.output(Constant.LEDGREEN_PIN, False)
Key = Key.Key()
myFile = File.myFile()
myFile.readFile()
myGPIO = myGPIO.myGPIO(GPIO, myFile)
myTimer = Timer.myTimer(myFile)
notBreakReset=True

def ledRGB(value):
    myGPIO.ledRGB(value)

def ledReboot() :
    DEFAULT_LED=Constant.DEFAULT_LED
    ledRGB("000")
    while True:
        GPIO.output(DEFAULT_LED, True)
        time.sleep(0.03)
        GPIO.output(DEFAULT_LED, False)
        time.sleep(0.03)
        GPIO.output(DEFAULT_LED, True)
        time.sleep(0.03)
        GPIO.output(DEFAULT_LED, False)
        time.sleep(0.03)

def ledUninstall() :
    DEFAULT_LED=Constant.DEFAULT_LED
    while GPIO.input(Constant.BELL_PIN) :
        GPIO.output(DEFAULT_LED, True)
        time.sleep(0.05)
        GPIO.output(DEFAULT_LED, False)
```

```

        time.sleep(0.1)
        ledRGB("000")

def ledAccesspoint() :
    DEFAULT_LED=Constant.DEFAULT_LED
    while _mode == 'ACCESS_POINT' :
        if Key.LED_key.is_set():
            time.sleep(0.1) # Khong sang den khi bi set
        else:
            GPIO.output(DEFAULT_LED,True)
            time.sleep(0.2)
            GPIO.output(DEFAULT_LED,False)
            time.sleep(0.2)
            GPIO.output(DEFAULT_LED,True)
            time.sleep(0.2)
            GPIO.output(DEFAULT_LED,False)
            time.sleep(1.5)

def ledReset() :
    global notBreakReset
    DEFAULT_LED=Constant.DEFAULT_LED
    while (GPIO.input(Constant.RESET_PIN) == 0) and notBreakReset:
        GPIO.output(DEFAULT_LED, True)
        time.sleep(0.1)
        GPIO.output(DEFAULT_LED, False)
        time.sleep(0.1)
    print('[LED] Stop led reset')

def discovery() :
    try:
        sock=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        sock.bind(("0.0.0.0", 9091))
        sock.setblocking(0)
    except:
        print("[UDP] Already open UDP 9091!")
    while True :
        data = None
        try:
            data, addr = sock.recvfrom(256)      # buffer size is 1024 bytes
        except:
            pass
        if data:
            print("received message:", data)
            decoded=json.loads(data.decode('utf-8'))
            if((decoded['req']=="IOT_DISCOVERY") and (decoded['type']=="DOORCAMERA")):
                _ssid = myFile.getssid()

```

```

        msg=json.dumps({'res':'IOT_DISCOVERY', 'type':'DOORCAMERA', 'status
': _mode,'ssid':_ssid, 'serial': serial_neo2}).encode()
            sock.sendto(msg, addr)
            print("DISCOVERY")
            # if _mode != 'PAIR' and _mode != 'STATION' :
            #     break
        if _mode != 'PAIR':
            print('[UDP] Break')
            break
        time.sleep(0.1)
    sock.close()

def changeDefaultSignalLed(jdata) :
    '''Parse JSON and link to setupOneLed'''
    print ("[DBG]-----> changeDefaultSignalLed")
    try:
        myFile._default_signal_led = str(jdata['value'])
        myGPIO.setupOneLed()
        myFile.writeFile('CONFIG', 'default_signal_led', myFile._default_signal_l
ed)
    except KeyError:
        myFile._default_signal_led = str(jdata['value'])
        myFile._default_signal_led = ""

def ledStarting() :
    station=myFile.station
    _default_signal_led=myFile._default_signal_led
    myGPIO.setupOneLed()
    while myFile.station == 'ERROR':
        ledRGB(myFile.LED_PIN)
        time.sleep(0.5)
        ledRGB("000")
        time.sleep(0.5)
        ledRGB(myFile.LED_PIN)

def ledPair() :
    global _mode
    DEFAULT_LED=Constant.DEFAULT_LED
    # GPIO.output(LEDBLUE_PIN, True)
    ledRGB('000')
    while (_mode == 'PAIR') :
        if Key.LED_key.is_set():
            time.sleep(0.1) # Khong sang den khi bi set
        else:
            GPIO.output(DEFAULT_LED, True)
            time.sleep(0.2)
            GPIO.output(DEFAULT_LED, False)
            time.sleep(0.2)

```

```
        GPIO.output(DEFAULT_LED, True)
        time.sleep(0.2)
        GPIO.output(DEFAULT_LED, False)
        time.sleep(0.2)
        GPIO.output(DEFAULT_LED, True)
        time.sleep(0.2)
        GPIO.output(DEFAULT_LED, False)
        time.sleep(1.5)

def uninstall():
    _mode=myFile._mode
    DEFAULT_LED=Constant.DEFAULT_LED
    ledRGB("000")
    i=0
    while True:
        GPIO.output(DEFAULT_LED,True)
        time.sleep(0.1)
        GPIO.output(DEFAULT_LED,False)
        time.sleep(0.1)
        i+=1
        if(i==15):
            #os.system("killall bip.py")
            break
        GPIO.output(DEFAULT_LED,False)
    print('-----')
    #os.system("ifconfig eth0 down");
    #os.system("ifconfig wlan0 down");
    subprocess.call("python3 ./boot_network.py lan -inet dhcp", shell=True)
    myFile.writeFile('CONFIG', 'mode', 'UNINSTALL')
    myFile._mode = 'UNINSTALL'
    #writeFile('CONFIG', 'alarm', 'INACTIVED')
    #_alarm = 'INACTIVED'
    #writeFile('CONFIG', 'doorcamera_id', '')
    #_doorcamera_id = ''
    #writeFile('CONFIG', 'server_ip', 'localhost')
    #_server_ip = 'localhost'
    ledUninstall()
    myFile.UNINSTALL_config()

def ledbell() :
    _default_signal_led=myFile._default_signal_led
    _bell_sound=myFile._bell_sound
    _interactive_mode=myFile._interactive_mode
    LED_PIN=myFile.LED_PIN
    LEDBLUE_PIN=Constant.LEDBLUE_PIN
    LEDGREEN_PIN=Constant.LEDGREEN_PIN
    LEDRED_PIN=Constant.LEDRD_PIN
    myGPIO.setupOneLed()
```

```
bt = time.time()
if _interactive_mode == 'READY':
    while True:
        if(GPIO.input(Constant.BELL_PIN)):
            bt = time.time()
            ledRGB(LED_PIN)
            time.sleep(.3)
            print("[GPIO] Hold")
            ledRGB("000")
            time.sleep(0.1)
        else:
            print("[GPIO] Realease")
            ledRGB(LED_PIN)
            time.sleep(0.1)
            break
elif _interactive_mode == 'DO_NOT_DISTURB':
    ledRGB("000")
    while True:
        if(GPIO.input(Constant.BELL_PIN)):
            bt = time.time()
            ledRGB(LED_PIN)
            time.sleep(.3)
            ledRGB("000")
            time.sleep(0.1)
        else:
            # ledRGB("000")
            ledRGB(LED_PIN)
            time.sleep(0.1)
            break
    ledRGB(LED_PIN)
elif _interactive_mode == 'GO_AWAY':
    ledRGB("000")
    while True:
        if(GPIO.input(Constant.BELL_PIN)):
            bt = time.time()
            ledRGB(LED_PIN)
            time.sleep(.3)
            ledRGB("000")
            time.sleep(0.1)
        else: #if(time.time() >= (bt + 1))
            # # GPIO.output(BIP_PIN, False)
            # GPIO.output(LEDGREEN_PIN, False)
            # #GPIO.output(LED_PIN2, False)
            # time.sleep(0.1)
            print("[GPIO] Realease")
            ledRGB(LED_PIN)
            time.sleep(0.1)
            break
```

```

    ledRGB(LED_PIN)
    Key.LED_bell_key.set()
    print("[BELL] Realease!!")

def aplay_timer(timer):
    time.sleep(timer)
    Key.APLAY_key.clear()

def actionbell() :
    station=myFile.station
    _bell_sound=myFile._bell_sound
    _auto_voice=myFile._auto_voice
    _rfprotocol=myFile._rfprotocol
    _rfpulse=myFile._rfpulse
    _rfpulse=myFile._rfpulse
    _rfbit=myFile._rfbit
    _rfcode=myFile._rfcode
    _rfenable=myFile._rfenable
    if(_rfenable == 'True'):
        # subprocess.call("./ringbell 1 200 24 13916568", shell=True)
        temp_cmd = './ringbell ' + _rfprotocol + ' ' + _rfpulse + ' ' + _rfbit +
        ' ' + _rfcode
        print(temp_cmd)
        try:
            subprocess.call(temp_cmd, shell=True)
        except:
            print('err')

    if not Key.APLAY_key.is_set():
        Key.APLAY_key.set()
        threading.Thread(target=aplay_timer, args=(4,)).start()
        subprocess.call('aplay ./DoorVillaSounds/DoorBell/'+'_bell_sound', shell =
True, stdout=Constant.FNULL); ## Used as BIP_PIN


def interactive_GOAWAY():
    _auto_voice=myFile._auto_voice
    _interactive_language=myFile._interactive_language
    print("GO AWAY-----")
    print("[DBG] Auto Voice: ", _auto_voice)
    if _auto_voice == "False":
        return
    Key.APLAY_key.set()
    if _interactive_language == 'vi':
        subprocess.call('aplay ' + Constant.VOICE_GOAWAY_VI, shell=True)
    else:
        subprocess.call('aplay ' + Constant.VOICE_GOAWAY_EN, shell=True)
    Key.APLAY_key.clear()

```

```
def interactive_DONOT():
    _interactive_language=myFile._interactive_language
    _auto_voice=myFile._auto_voice
    Key.APLAY_key.set()
    print("DO NOT DISTURB-----")
    print("[DBG] Auto Voice: ", _auto_voice)
    if _auto_voice == "False":
        return
    if _interactive_language == 'vi':
        subprocess.call('aplay '+ Constant.VOICE_NOTDISTURB_VI, shell=True)
    else:
        subprocess.call('aplay '+ Constant.VOICE_NOTDISTURB_EN, shell=True)
    Key.APLAY_key.clear()

def bell(channel):
    station=myFile.station
    _interactive_mode=myFile._interactive_mode
    auto_schedule=myFile.auto_schedule
    _server_ip=myFile._server_ip
    _server_domain=myFile._server_domain
    # _ftp_domain_dns=myFile._ftp_domain_dns
    # _ftp_user=myFile._ftp_user
    # _ftp_password=myFile._ftp_password
    _snapshot_if_press=myFile._snapshot_if_press
    Key.LED_bell_key.clear()
    print('Bell action mode: %s'%(station))
    if(_mode == 'STATION') :
        threading.Thread(target = ledbell, args = ()).start()
        ##### GET SCHEDULE
        if auto_schedule == "True":
            _interactive = Calendar.CALENDAR_checkFile()
            if _interactive == False:
                _interactive = _interactive_mode
        else:
            _interactive = _interactive_mode
    print("[INTERACTIVE] mode: ", _interactive)
    #####
    if _interactive == "READY":
        threading.Thread(target = actionbell, args = ()).start()
    elif _interactive == "GO_AWAY":
        if not Key.APLAY_key.is_set():
            threading.Thread(target= interactive_GOAWAY, args=()).start()
        else:
            print("[APLAY] is busy")
    elif _interactive == "DO_NOT_DISTURB":
        if not Key.APLAY_key.is_set():
```

```

        threading.Thread(target= interactive_DONOT, args=()).start()
    else:
        print("[APLAY] is busy")
    else:
        print("[INTERACTIVE] Not Match")
#Streaming & Sockket send
if(station == 'SUCCESS'):
    # subprocess.call('aplay ./DoorVillaSounds/DoorBell/'+_bell_sound, shell = True, stdout=NULL); ## Used as BIP_PIN
    time.sleep(0.1)
    print('[BELL] Belling...')
    msg=json.dumps({'req':'DOOR_VIDEOCALL','id':myFile._doorcamera_id, 'status':_interactive})
    msg=msg.encode()
    # threading.Thread(target = sslClient, args = (msg,ip_login,)).start()
)
    threading.Thread(target = SocketClient.sslClient, args = (msg,myFile._server_ip,)).start()
    time.sleep(0.1)
    # Old version
    ######
    # i=datetime.now()
    # name=str(i)
    # name=name.replace(" ", "")
    # name=name.replace(":", "")
    # name=name.replace("-", "")
    # name=name.replace(".", "")
    # name=name+"_door_press.png"
    # os.system("fswebcam /var/www/pictures/"+name)
    ######
    #New version --> Move to bell
line = subprocess.check_output('ps -ef | grep gst', shell=True)
line = line.decode()

# if 'gst-launch-1.0' in line.split(' '):
#     print("Catch gst-launch-1.0 --> reset timer ")
#     # Streaming_Timer.reset()
# else:
#     if(_snapshot_if_press=="True"):
#         print('[GST] Stream and snap shot')
#         threading.Thread(target=apache_handler, args=("call_snapshot", "ftp://%s:%s/%s"%( _ftp_user, _ftp_password, _ftp_domain_dns, serial_neo2),)).start()
#     else:
#         print('[GST] Streaming')
#         threading.Thread(target=apache_handler, args=("call",'')).start()

```

```

        #      threading.Thread(target=apache_handler, args=("getAudio",'',)).start()
        #      Streaming_Timer.reset()

        # Temp comment
        # if _setup_media_store == 'True' :
        #     store_path = "%s/doorcamera/%s/"%(_media_store_path,_doorca
mera_id)
        #     threading.Thread(target = sync_server, args = (store_path,i
p_login,)).start()

    elif(_mode == 'UNINSTALL') :
        ledUninstall()

def exithandler(signal, frame):
    # rfdevice.cleanup()
    ledRGB('000')
    GPIO.cleanup()
    # apache_end_period()
    sys.exit(0)

#####
#Control
#####
signal.signal(signal.SIGINT, exithandler)
# setupOneLed()
print(myFile._interactive_mode)
_mode=myFile._mode
f = open("/proc/device-tree/serial-number", "r")
serial_neo2 = f.read().strip()[0:16]
f.close()
print('Startup by mode: %s'%'_mode')
print('serial: %s'%'(serial_neo2)')
Key.OPEN_key.set()
GPIO.add_event_detect(Constant.BELL_PIN, GPIO.RISING,callback = bell,bouncetime =
300)
# GPIO.add_event_detect(BUTTON_01,GPIO.RISING,callback = openaction,bouncetime =
300)
# GPIO.add_event_detect(BUTTON_02,GPIO.FALLING,callback = reset,bouncetime = 300)
try:
    # Streaming_Timer = Timer.PeriodTimer(Constant.STREAMING_PERIOD, apache_end_p
eriod)
    # Streaming_Timer.start()
    # LockOpen_Timer = Timer.PeriodTimer(int(myFile.lock_open_time), Lock_close_e
vent)
    # LockOpen_Timer.start()
    thread_socket_server = Network.socket_server_threading(1, "Thread-
local", myFile, Key, myGPIO, myTimer) #Calls thread 1
    thread_socket_server.start()

```

```

print("hello")
try:
    threading.Thread(target = ledStarting, args = ()).start()
except:
    print("Unexpected error:", sys.exc_info()[0])
    pass
if _mode == 'PAIR':
    threading.Thread(target = discovery, args = ()).start()
if _mode == 'PAIR' :
    myFile.station = 'SUCCESS'
    threading.Thread(target = ledPair, args = ()).start()
elif _mode == 'ACCESS_POINT':
    print("[DBG] _reboot_due_to_ap", myFile._reboot_due_to_ap)
    if myFile._reboot_due_to_ap == 'True':
        myFile._reboot_due_to_ap = "False"
        myFile.writeFile("LOG","reboot_due_to_ap", "False")
        print("-----ACCESS_POINT-----")
        myFile.station = 'SUCCESS'
        threading.Thread(target = ledAccesspoint, args = ()).start()
        myFile._interface = 'WIFI'
        myFile.writeFile('CONFIG', 'interface', 'WIFI')
        # os.system("ifconfig "+_wifi_card_name+" up");
        # os.system("ifconfig eth0 down");
        subprocess.call('dnsmasq', shell=True)
    else:
        uninstall()
elif _mode == 'UNINSTALL':
    myFile.station = 'SUCCESS'
    GPIO.output(Constant.LEDRED_PIN, False)
elif _mode == 'STATION':
    SocketClient.startWhoIAm(myFile)

# GPIO.output(WARN_PIN, False)
Key.LED_key.clear()
Key.WRN_key.clear()
Key.RF_key.clear()
Key.APLAY_key.clear()
Key.OPEN_key.clear()
Key.LOCK_key.clear()
Key.LED_bell_key.set()
except KeyboardInterrupt:
    GPIO.cleanup()
    pass

```

**Constant.py**

```
import os
JANUS_AUDIOBRIDGE_CONFIG_PATH='/usr/local/etc/janus.plugin.audiobridge.jcfg'
'

JANUS_STREAMING_CONFIG_PATH='/usr/local/etc/janus.plugin.streaming.jcfg'
config_path='./config'
UDP_PORT=9091
PORT_AP=8092
PORT_SOCKET=8093
PORT=8091
WSS_PORT = 8099
PORT_SOCKET=8093
PORT_HTTP = 8080
PORT_HTTP_REALTIME = 8081 #Port real-time
HOST_CLI = ''
PASSWORD='123456'
BUFFER_SIZE=1024
FNULL = open(os.devnull, 'w')
STREAMING_PERIOD = 65 #seconds
FIN_MODE="NONE"
LEDRED_PIN=22
LEDGREEN_PIN=17
LEDBLUE_PIN=27
BELL_PIN=5
DEFAULT_LED=LEDBLUE_PIN
RESET_PIN=26
```

### File.py

```
import configparser as ConfigParser
import subprocess
import os
import Constant
import time
class myFile:
    def __init__(self):
        self.timeout_alarm = time.time()
        self.LED_PIN="111"
        self.config = ConfigParser.ConfigParser()
        self.config_path="./config"

        self._pin = ""
        self._mode = ""
        self.station = "ERROR"
        self._gcms = ""
        self._server_domain = ""

        self._setup_media_store = ""
```

```
self._doorcamera_name = ""
self._doorcamera_id = ""
self._auto_voice = ""
self._alarm = ""
self._alarm_auto_switch_off = ""
self._alarm_auto_switch_off_s = ""

self._snapshot_if_press = ""
self._snapshot_manual = ""
self._snapshot_if_attacked = ""
self._interface = ""
self.ip_login = ""

self._default_signal_led = ""
self._snapshot_if_rfid = ""

self._rfenable = ""
self._rfprotocol = ""
self._rfcode = ""
self._rfpulse = ""
self._rfbit = ""

self._wifi_card_name = ""
self._auto_set_static_ip = ""
self.interface_card_map = ""

self._rfidenable = ""
self._rfidnumber = ""
self._master = ""

self._interactive_mode = ""
self._pin_interactive_language = ""

self._bell_sound = ""
self._rfid_sound = ""
self._janus = ""
self._reboot_due_to_ap = ""
auto_schedule = ""

self.lock_firm = ""
self.lock_IP = ""
self.lock_seri = ""
self.lock_ssid = ""
self.lock_type = ""
self.lock_open_time = ""
self.lock_en = ""
```

```
f = open("/proc/device-tree/serial-number", "r")
self.serial_neo2 = f.read().strip()[0:16]

def readFile(self) :
    config = self.config
    config_path = self.config_path
    config.read(config_path)

    try:
        self.auto_schedule = config.get('CALENDAR', 'auto_schedule')
    except ConfigParser.NoOptionError:
        pass

    try:
        self._janus = config.get('CONFIG', 'janus')
    except ConfigParser.NoOptionError:
        pass

    try:
        self._mode = config.get('CONFIG', 'mode')
    except ConfigParser.NoOptionError:
        pass

    try:
        self._pin = config.get('SECURITY', 'pin')
    except ConfigParser.NoOptionError:
        pass

    try:
        self._gcms = config.get('SECURITY', 'gcms')
    except ConfigParser.NoOptionError:
        pass

    try:
        self._server_ip = config.get('CONFIG', 'server_ip')
        ip_login = self._server_ip
    except ConfigParser.NoOptionError:
        pass

    try:
        self._server_domain = config.get('CONFIG', 'server_domain')
        ip_login = self._server_domain
    except ConfigParser.NoOptionError:
        pass

    try:
        self._setup_media_store = config.get('CONFIG', 'setup_media_store')
    except ConfigParser.NoOptionError:
        pass
```

```
try:  
    self._doorcamera_name = config.get('CONFIG', 'doorcamera_name')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._doorcamera_id = config.get('CONFIG', 'doorcamera_id')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._auto_voice = config.get('CONFIG', 'auto_voice')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._alarm = config.get('CONFIG', 'alarm')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._alarm_auto_switch_off = config.get('CONFIG', 'alarm_auto_switch_off')  
    if self._alarm_auto_switch_off == 'True' :  
        self._alarm_auto_switch_off_s = config.get('CONFIG', 'alarm_auto_switch_off_s')  
    except ConfigParser.NoOptionError:  
        pass  
  
try:  
    self._snapshot_if_press = config.get('CONFIG', 'snapshot_if_press')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._snapshot_manual = config.get('CONFIG', 'snapshot_manual')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._snapshot_if_attacked = config.get('CONFIG', 'snapshot_if_attacked')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._snapshot_if_rfid = config.get('CONFIG', 'auto_take_snapshot_if_rfid')
```

```
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._interface = config.get('CONFIG', 'interface')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._default_signal_led = config.get('CONFIG', 'default_signal_led')  
    print("---> LED: ",self._default_signal_led)  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._rfenable = config.get('RF', 'enable')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._rfprotocol = config.get('RF', 'protocol')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._rfpulse = config.get('RF', 'pulse')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._rfbit = config.get('RF', 'bit')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._rfcode = config.get('RF', 'rfcode')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._rfidenable = config.get('RFID', 'enable')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._master = config.get('RFID', 'master')  
except ConfigParser.NoOptionError:  
    pass
```

```
try:  
    self._rfidnumber = config.get('RFID', 'number')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._interactive_mode = config.get('INTERACT', 'mode')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._interactive_language = config.get('INTERACT', 'language')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._rfid_sound = config.get('SOUND', 'rfid')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._bell_sound = config.get('SOUND', 'bell')  
except ConfigParser.NoOptionError:  
    pass  
  
if self.getInterfaceCard() == 2:  
    self._wifi_card_name = self.interface_card_map[1]  
    print(self._wifi_card_name)  
  
try:  
    self._auto_set_static_ip = config.get('LOG', 'auto_set_static_ip')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self._reboot_due_to_ap = config.get('LOG', 'reboot_due_to_ap')  
except ConfigParser.NoOptionError:  
    pass  
  
try:  
    self.lock_firm = config.get('LOCK', 'lock_firm')  
except ConfigParser.NoOptionError:  
    pass  
try:  
    self.lock_IP = config.get('LOCK', 'lock_ip')  
except ConfigParser.NoOptionError:  
    pass
```

```
try:  
    self.lock_seri = config.get('LOCK', 'lock_seri')  
except ConfigParser.NoOptionError:  
    pass  
try:  
    self.lock_ssid = config.get('LOCK', 'lock_ssid')  
except ConfigParser.NoOptionError:  
    pass  
try:  
    self.lock_type = config.get('LOCK', 'lock_type')  
except ConfigParser.NoOptionError:  
    pass  
try:  
    self.lock_open_time = config.get('LOCK', 'lock_open_time')  
except ConfigParser.NoOptionError:  
    pass  
try:  
    self.lock_en = config.get('LOCK', 'lock_en')  
except ConfigParser.NoOptionError:  
    pass  
try:  
    self.lock_secretkey = config.get('LOCK', 'lock_secretkey')  
except ConfigParser.NoOptionError:  
    pass  
  
def writeFile(self, section, key, value) :  
    config.set(section, key, value)  
    with open(self.config_path, 'w') as configfile:  
        config.write(configfile)  
  
def clearFile(self, section, key):  
    config.remove_option(section, key)  
    with open(self.config_path, 'w') as configfile:  
        config.write(configfile)  
  
def replace_line(self, file_name, line_num, text):  
    lines = open(file_name, 'r').readlines()  
    lines[line_num] = text  
    out = open(file_name, 'w')  
    out.writelines(lines)  
    out.close()  
  
def replace_key(self, file_name, key, text):  
    lines = []  
    isReplace = True  
    with open(file_name) as infile:  
        for line in infile.readlines():  
            if (key in line) and (' lo' not in line):
```

```
        line = text
        isReplace = False
        lines.append(line)
        if( isReplace == True ):
            lines.append(key)
    with open(file_name, 'w') as outfile:
        for line in lines:
            outfile.write(line)

def getInterfaceCard(self):
    line = subprocess.check_output('iwconfig', shell=True)
    line = line.decode()
    for paragraph in line.split('\n'):
        for interface in paragraph.split(' '):
            if interface == '':
                self.interface_card_map = ['eth0']
                return 1
            else:
                self.interface_card_map = ['eth0',interface]
                return 2
def authentication(self, jdata) :
    _mode=self._mode
    _pin=self._pin
    login_fail_map=self.login_fail_map
    interface_card_map=self.interface_card_map
    ip_login=self.ip_login
    login_fail_counter = 0
    try :
        password_ = jdata['password']
        serial_ = jdata['serial']
        if password_ == _pin and serial_neo2 == serial_:
            isContinue = False
            try:
                if login_fail_map[ip_login] < time.time() :
                    isContinue = True
            except KeyError:
                isContinue = True
                pass # do nothing!
        if isContinue :
            login_fail_counter = 0
            return True
        else :
            return False
    else :
        login_fail_counter = login_fail_counter + 1
        if login_fail_counter > 3 :
            login_fail_map.update({ip_login: (time.time() + 300)}));
        return False
```

```

        except KeyError:
            pass
        return False

    def changeRFIDInfo(self, jdata):
        self._rfidenable = str(jdata['integrated_rfid'])
        print("[DBG] -----> RFID enable: ", self._rfidenable)
        self.writeFile('RFID', 'enable', self._rfidenable)

    def changeLockInFo(self, jdata, LockOpen_Timer):
        self.lock_en = str(jdata['door_lock'])
        self.lock_open_time = str(jdata['open_time'])
        LockOpen_Timer.update_counter_time(int(self.lock_open_time))
        self.writeFile('LOCK', 'lock_en', self.lock_en)
        self.writeFile('LOCK', 'lock_open_time', self.lock_open_time)

    def changeCameraInfo(self, jdata) :
        self._pin = jdata['password']
        self._doortcamera_name = jdata['doortcamera_name']
        self.writeFile('SECURITY', 'pin', str(self._pin))
        self.writeFile('CONFIG', 'doortcamera_name', self._doortcamera_name)
        replace_key(Constant.JANUS_STREAMING_CONFIG_PATH, "pin =", "\tpin = \"%s\""
\n%(self._pin))
        # replace_key(JANUS_AUDIOBRIDGE_CONFIG_PATH, "pin =", "\tpin = \"%s\"\n%"(new_password_))
        os.system("systemctl enable janus.service")
        os.system("systemctl restart janus.service")

    def changeSnapshot(self, jdata) :
        self._snapshot_if_press = str(jdata['snapshot_if_press'])
        self._snapshot_if_attacked = str(jdata['snapshot_if_attacked'])
        self._snapshot_manual = str(jdata['snapshot_manual'])
        self._snapshot_if_rfid = str(jdata['auto_take_snapshot_if_rfid'])
        self._setup_media_store = str(jdata['setup_media_store'])
        # self._ftp_domain = str(jdata['ftp_domain'])
        # self._ftp_user = str(jdata['ftp_user'])
        # self._ftp_password = str(jdata['ftp_password'])
        self.writeFile('CONFIG', 'snapshot_if_press', self._snapshot_if_press)
        self.writeFile('CONFIG', 'snapshot_if_attacked', self._snapshot_if_attacked)
        self.writeFile('CONFIG', 'snapshot_manual', self._snapshot_manual)
        self.writeFile('CONFIG', 'auto_take_snapshot_if_rfid', self._snapshot_if_rfid)
        self.writeFile('CONFIG', 'setup_media_store', self._setup_media_store)
        # self.writeFile('CONFIG', 'ftp_domain', self._ftp_domain)
        # self.writeFile('CONFIG', 'ftp_user', self._ftp_user)
        # self.writeFile('CONFIG', 'ftp_password', self._ftp_password)

```

```

#socket.gethostbyname(_ftp_domain)
#Cap nhat cac file /usr/src/apache-tomcat-7.0.81/conf/streaming/*

def changeAlarm(self, jdata) :
    self._auto_voice = str(jdata['auto_voice'])
    self._alarm = str(jdata['alarm'])
    self._alarm_auto_switch_off = str(jdata['alarm_auto_switch_off'])
    self._alarm_auto_switch_off_s = str(jdata['alarm_auto_switch_off_s'])
    self.writeFile('CONFIG', 'auto_voice', self._auto_voice)
    self.writeFile('CONFIG', 'alarm', self._alarm)
    self.writeFile('CONFIG', 'alarm_auto_switch_off', self._alarm_auto_switch
_off)
    self.writeFile('CONFIG', 'alarm_auto_switch_off_s', self._alarm_auto_swit
ch_off_s)

def UNINSTALL_config(self):
    rfidenable=self.rfidenable
    _rfidnumber=self._rfidnumber
    _rfenable=self._rfenable
    _rfcode=self._rfcode
    _rfprotocol=self._rfprotocol
    _rfpulse=self._rfpulse
    _rfbit=self._rfbit
    auto_schedule=self.auto_schedule
    _snapshot_if_press=self._snapshot_if_press
    _snapshot_manual=self._snapshot_manual
    _snapshot_if_rfid=self._snapshot_if_rfid
    _alarm_auto_switch_off=self._alarm_auto_switch_off
    _setup_media_store=self._setup_media_store
    _snapshot_if_attacked=self._snapshot_if_attacked
    _default_signal_led=self._default_signal_led
    _interactive_mode=self._interactive_mode
    _interactive_language=self._interactive_language
    _pin=self._pin
    lock_firm=self.lock_firm
    lock_IP=self.lock_IP
    lock_seri=self.lock_seri
    lock_ssid=self.lock_ssid
    lock_type=self.lock_type
    ##### RFID #####
    config.read(config_path)
    number = config.get('RFID', 'Number')
    _rfidenable = "False"
    self.writeFile("RFID", "enable", "False")
    for x in range(int(number)):
        self.writeFile("RFID", "No"+str(x), "DELETE")
    self.writeFile("RFID", "number", "0")
    _rfidnumber = "0"

```

```

##### RF      #####
_rfenable = "False"
_rfenable = "UNSET"
_rfcode = "UNSET"
_rfprotocol = "UNSET"
_rfpulse = "UNSET"
_rfbit = "UNSET"
self.writeFile("RF", "enable", "False")
self.writeFile("RF", "protocol", "UNSET")
self.writeFile("RF", "rfcode", "UNSET")
self.writeFile("RF", "pulse", "UNSET")
self.writeFile("RF", "bit", "UNSET")
##### CALENDAR #####
self.writeFile("CALENDAR", "schedule", "[]")
self.writeFile("CALENDAR", "auto_schedule", "False")
auto_schedule = "False"
##### CONFIG #####
_snapshot_if_press = "False"
_snapshot_manual = "False"
_snapshot_if_rfid = "False"
_alarm_auto_switch_off = "False"
_setup_media_store = "False"
_snapshot_if_attacked = "False"
_default_signal_led = '4'
self.writeFile("CONFIG", "alarm_auto_switch_off", "False")
self.writeFile("CONFIG", "snapshot_if_press", "False")
self.writeFile("CONFIG", "snapshot_manual", "False")
self.writeFile("CONFIG", "setup_media_store", "False")
self.writeFile("CONFIG", "auto_take_snapshot_if_rfid", "False")
self.writeFile("CONFIG", "snapshot_if_attacked", "False")
self.writeFile("CONFIG", "default_signal_led", "4")
##### INTERACT #####
_interactive_mode = "READY"
_interactive_language = "vi"
self.writeFile("INTERACT", "mode", "READY")
self.writeFile("INTERACT", "language", "vi")
##### SECURITY #####
config.read(config_path)
_pin = config.get('DEFAULT', 'pin')
self.writeFile("SECURITY", "pin", _pin)
##### LOCK #####
lock_firm = "NOT_PAIR"
lock_IP = "NOT_PAIR"
lock_seri = "NOT_PAIR"
lock_ssid = "NOT_PAIR"
lock_type = "NOT_PAIR"
self.writeFile("LOCK", "lock_ip", "NOT_PAIR")
self.writeFile("LOCK", "lock_seri", "NOT_PAIR")

```

```

        self.writeFile("LOCK", "lock_ssid", "NOT_PAIR")
        self.writeFile("LOCK", "lock_type", "NOT_PAIR")
        self.writeFile("LOCK", "lock_firm", "NOT_PAIR")
        if number!="0":
            replace_key(config_path, 'DELETE', ' ')
        self.readFile()

def getssid(self):
    try:
        ssid=os.popen("iwconfig").readlines()
        ssid=ssid[0]
        ssid=ssid[ssid.index("")+1:ssid.index("\n")-3]
        print(ssid)
        return ssid
    except Exception:
        print("SSID null!")
        return ""

```

**JSON\_station.py**

```

import subprocess
import threading
import json
import os
import time
import SocketClient
import Lock
import RF
import Sound
class JSONStation:
    def __init__(self, myFile, Key, myGPIO, myTimer):
        self.myFile = myFile
        self.LockOpen_Timer = myTimer.LockOpen_Timer
        self.Key = Key
        self.Streaming_Timer = myTimer.Streaming_Timer
        self.Sound = Sound.Sound(Key, myFile)
        self.myGPIO = myGPIO

    def JSON_station(self, conn, jdata):
        config = self.myFile.config
        _mode=self.myFile._mode
        _pin=self.myFile._pin
        login_fail_map=self.myFile.login_fail_map
        _alarm=self.myFile._alarm
        _alarm_auto_switch_off=self.myFile._alarm_auto_switch_off
        _alarm_auto_switch_off_s=self.myFile._alarm_auto_switch_off_s
        timeout_alarm=self.myFile.timeout_alarm
        _snapshot_if_press=self.myFile._snapshot_if_press

```

```

_snapshot_manual=self.myFile._snapshot_manual
_snapshot_if_attacked=self.myFile._snapshot_if_attacked
ip_login=self.myFile.ip_login
_setup_media_store=self.myFile._setup_media_store
_ftp_domain=self.myFile._ftp_domain
_ftp_domain_dns=self.myFile._ftp_domain_dns
_ftp_user=self.myFile._ftp_user
_ftp_password=self.myFile._ftp_password
interface_card_map=self.myFile.interface_card_map
_default_signal_led=self.myFile._default_signal_led
_interactive_mode=self.myFile._interactive_mode
auto_schedule=self.myFile.auto_schedule
lock_firm=self.myFile.lock_firm
lock_IP=self.myFile.lock_IP
lock_seri=self.myFile.lock_seri
lock(ssid=self.myFile.lock(ssid)
lock_type=self.myFile.lock_type
_finenable=self.myFile._finenable
_finnumber=self.myFile._finnumber
_finpass=self.myFile._finpass
_finblock=self.myFile._finblock
_fin_search_number=self.myFile._fin_search_number
_server_domain=self.myFile._server_domain
ip_connect=self.myFile.ip_connect
_server_ip=self.myFile._server_ip
_doorcamera_id=self.myFile._doorcamera_id
if jdata['req'] == 'UPDATE_CAMERA' :
    dataLogin = jdata['login']
    isReboot = False
    if self.myFile.authentication(dataLogin) :
        isRFID = jdata['change_integrated_rfid']
        if isRFID == True:
            self.myFile.changeRFIDInfo(jdata['integrated_rfid'])
        isRFID = jdata['change_integrated_fingerprint']
        if isRFID == True:
            self.myFile.changeRFIDInfo(jdata['integrated_fingerprint'])
        isRF = jdata['change_door_bell']
        if isRF == True:
            self.myFile.changeRFInfo(jdata['door_bell'])
        isLock = jdata['change_door_lock']
        if isLock == True:
            self.myFile.changeLockInfo(jdata['door_lock'], self.LockOpen_
Timer)
        isInfo = jdata['change_info'];
        if isInfo == True :
            isReboot = True
            dataInfo = jdata['info']
            self.myFile.changeCameraInfo(dataInfo)

```

```

isSnapshot = jdata['change_snapshot']
if isSnapshot == True :
    dataSnapshot = jdata['snapshot']
    self.myFile.changeSnapshot(dataSnapshot)
isAlarm = jdata['change_alarm']
if isAlarm == True :
    dataAlarm = jdata['alarm']
    self.myFile.changeAlarm(dataAlarm)
jsonRes = {}
jsonRes['res'] = 'UPDATE_CAMERA'
response = '%s\n'%(json.dumps(jsonRes))
conn.write(response.encode())

isNetwork = jdata['change_network']
if isNetwork == True :
    isReboot = True
    dataNetwork = jdata['network']
    self.myFile.changeNetwork(dataNetwork)
if isReboot == True :
    print('rebooting >>>>>>>>>>>>>>>>>>>>> ')
    # thread.start_new_thread(reboot, (3,))
    threading.Thread(target=reboot, args = (3,)).start()
#calendar
isSchedule = jdata['change_status_by_schedule']
if isSchedule == True:
    schedules = jdata['schedules']
    self.writeFile("CALENDAR", "schedule", str(schedules).replace
('\'', '\"'))
    auto_schedule = jdata['auto_schedule']
    auto_schedule = str(auto_schedule)
    self.writeFile("CALENDAR", "auto_schedule", auto_schedule)
else :
    jsonRes = {}
    jsonRes['res'] = 'FAIL'
    response = '%s\n'%(json.dumps(jsonRes))
return response
elif jdata['req'] == 'NETWORK':
    # changeNetwork(jdata)
    # Old: Dung de change Network
    # New: Dung de lay thong tin interface
    self.getInterfaceCard()
    jsonRes = {}
    jsonRes['res'] = 'NETWORK'
    jsonInterface = {}
    proc = subprocess.check_output("timedatectl | grep 'Time zone'", shel
L=True)
    proc = proc.decode()
    timezone = proc.split(": ")[1]

```

```

        jsonRes['timezone'] = timezone
        for i in interface_card_map:
            if(i=='eth0') :
                jsonInterface[i] = 'lan'
            else:
                jsonInterface[i] = 'wifi'
        jsonRes['interface'] = jsonInterface
        # response = '%s\n'%(json.dumps(jsonRes))
        response = '%s\n'%(json.dumps(jsonRes))
    elif jdata[ 'req' ] == 'ALARM_ON' :
        if(_alarm=="ACTIVED"):
            temp_msg = "{\"res\":\"\\\"ALARM_ON\\\",\\\"id\\\":\\\""+ _doorcamera_id +"\\\"}"
        temp_msg = temp_msg.encode()
        threading.Thread(target = SocketClient.sslClient, args = (temp_ms
g,ip_login,)).start()
        # GPIO.output(WARN_PIN, True)
        threading.Thread(target = self.Sound.alarm_sound, args = ()).star
t()
        if _alarm_auto_switch_off == 'True' :
            time_ = int(_alarm_auto_switch_off_s)
            if timeout_alarm <= time.time() :
                timeout_alarm = time_ + time.time()
                # thread.start_new_thread(attackBell, ()).start()
                threading.Thread(target = self.Sound.attackBell, args = (
)).start()
            else :
                timeout_alarm = time_ + time.time()
        else :
            print('_alarm_auto_switch_off=False')
    elif jdata[ 'req' ] == 'ALARM_OFF' :
        temp_msg = "{\"res\":\"\\\"ALARM_OFF\\\",\\\"id\\\":\\\""+ _doorcamera_id +"\\\"}"
        temp_msg = temp_msg.encode()
        threading.Thread(target = SocketClient.sslClient, args = (temp_msg,ip
_login,)).start()
        # GPIO.output(WARN_PIN, False)
        Key.WRN_key.clear()
        subprocess.call("killall -9 aplay", shell=True)
        timeout_alarm = time.time()
    elif jdata[ 'req' ]=="ALARM_ACTIVE" :
        _alarm = 'ACTIVED'
        self.writeFile('CONFIG', 'alarm', 'ACTIVED')
        _alarm_auto_switch_off = str(jdata['alarm_auto_switch_off'])
        self.writeFile('CONFIG', 'alarm_auto_switch_off', _alarm_auto_switch_
off)
        _alarm_auto_switch_off_s = str(jdata['alarm_auto_switch_off_s'])
        self.writeFile('CONFIG', 'alarm_auto_switch_off_s', _alarm_auto_switc
h_off_s)

```

```

    elif jdata['req']=="ALARM_INACTIVE" :
        _alarm = 'INACTIVED'
        self.writeFile('CONFIG', 'alarm', 'INACTIVED')
        _alarm_auto_switch_off = str(jdata['alarm_auto_switch_off'])
        self.writeFile('CONFIG', 'alarm_auto_switch_off', _alarm_auto_switch_
off)
        _alarm_auto_switch_off_s = str(jdata['alarm_auto_switch_off_s'])
        self.writeFile('CONFIG', 'alarm_auto_switch_off_s', _alarm_auto_switc
h_off_s)
    elif(jdata['req']=="SNAPSHOT_CONFIG") :
        try:
            snapshot_if_press = jdata['snapshot_if_press']
        except KeyError:
            snapshot_if_press = ""
        try:
            snapshot_manual = jdata['snapshot_manual']
        except KeyError:
            snapshot_manual = ""
        try:
            snapshot_if_attacked = jdata['snapshot_if_attacked']
        except KeyError:
            snapshot_if_attacked = ""
        if(snapshot_if_press=="ON"):
            _snapshot_if_press = 'True'
            self.writeFile('CONFIG', 'snapshot_if_press', _snapshot_if_press)
        elif(snapshot_if_press=="OFF"):
            _snapshot_if_press = 'False'
            self.writeFile('CONFIG', 'snapshot_if_press', _snapshot_if_press)
        if(snapshot_manual=="ON"):
            _snapshot_manual = 'True'
            self.writeFile('CONFIG', 'snapshot_manual', _snapshot_manual)
        elif(snapshot_manual=="OFF"):
            _snapshot_manual = 'False'
            self.writeFile('CONFIG', 'snapshot_manual', _snapshot_manual)
        if(snapshot_if_attacked=="ON"):
            _snapshot_if_attacked = 'True'
            self.writeFile('CONFIG', 'snapshot_if_attacked', _snapshot_if_att
acked)
        elif(snapshot_if_attacked=="OFF"):
            _snapshot_if_attacked = 'False'
            self.writeFile('CONFIG', 'snapshot_if_attacked', _snapshot_if_att
acked)
    elif(jdata['req']=="SIGNAL_LED_COLOR") :
        try:
            _default_signal_led = str(jdata['value'])
            self.myGPIO.setupOneLed()
            self.writeFile('CONFIG', 'default_signal_led', _default_signal_le
d)

```

```

        except KeyError:
            _default_signal_led = ""

##LOCK##
elif(jdata['req'] == "UPDATE_LOCK"):
    lock_firm = jdata['firmware']
    lock_ssid = jdata['ssid']
    lock_seri = jdata['serial']
    lock_type = jdata['type']
    lock_IP = jdata['ip_address']
    self.writeFile("LOCK", "lock_ip", lock_IP)
    self.writeFile("LOCK", "lock_seri", lock_seri)
    self.writeFile("LOCK", "lock_ssid", lock_ssid)
    self.writeFile("LOCK", "lock_type", lock_type)
    self.writeFile("LOCK", "lock_firm", lock_firm)
    jsonRes = {}
    jsonRes['res'] = 'UPDATE_LOCK'
    response = '%s\n'%(json.dumps(jsonRes))

##RF#####
elif(jdata['req']=='ADD_RF'):
    _pul = jdata['pul']
    _rfbit = jdata['bit']
    _code = jdata['code']
    _pro = jdata['pro']
    self.writeFile('RF','protocol',_pro)
    self.writeFile('RF','rfcode',_code)
    self.writeFile('RF','pulse',_pul)
    self.writeFile('RF','bit',_rfbit)

elif(jdata['req']=='DETECT_RF'):
    print('Detect RF')
    jsonRes = {}
    response = '%s\n'%(json.dumps(jsonRes))
    threading.Thread(target = RF.RF_listen_handler, args = (30, self.Key.RF_key, self.myFile)).start()

elif(jdata['req']=='REMOVE_RF'):
    print('Remove RF')
    self.writeFile('RFID','protocol','1')
    self.writeFile('RFID','rfcode','0')
    self.writeFile('RFID','pulse','0')
    self.writeFile('RFID','bit','0')
    jsonRes = {}
    jsonRes['res'] = 'SUCCESS'
    response = '%s\n'%(json.dumps(jsonRes))

#####INTERACT
elif(jdata['req']=='LED_STATUS'):
    auto_schedule = jdata['auto_schedule']

```

```

        self.writeFile("CALENDAR","auto_schedule", str(auto_schedule))
        _interactive_mode = jdata['status']
        self.writeFile('INTERACT','mode',_interactive_mode)
        response = '\n'

    elif(jdata[ 'req ']== 'STREAMING_STATUS '):
        # Kiem tra gst-launch-1.0 co dang chay khong
        # Khong --> Chay
        # Co thi thoai
        if jdata['status']=='PLAY':
            line = subprocess.check_output('ps -ef | grep gst', shell=True)
            line = line.decode()
            if 'gst-launch-1.0' in line.split(' '):
                print('[GST] catch gst-launch-1.0 --> reset timer')
                self.Streaming_Timer.reset()
            else:
                if(_snapshot_if_press=="True"):
                    print('[GST] stream and snap shot')
                    threading.Thread(target=apache_handler, args=("call_snaps
hot", "ftp://%s:%s@%s/%s"%( _ftp_user, _ftp_password, _ftp_domain_dns, serial_neo2
),)).start()
                else:
                    print('[GST] stream')
                    threading.Thread(target=apache_handler, args=("call",''))
        ).start()
        threading.Thread(target=apache_handler, args=("getAudio",''))
    ).start()
        self.Streaming_Timer.reset()

    elif jdata[ 'status ']== 'STOP ':
        subprocess.call('pkill gst-launch-1.0',shell=True)
        jsonRes = {}
        jsonRes[ 'res ']= 'STREAMING_STATUS '
        response = '%s\n'%(json.dumps(jsonRes))

# Request Paring lock
    elif jdata[ 'req ']== 'SIGNAL_IOT_DISCOVERY ':
        threading.Thread(target=Lock.Lock_Discovery_Send, args=()).start()
        # threading.Thread(target=Lock_Discovery_Rcv, args=()).start()
        response = Lock.Lock_Discovery_Rcv()
    elif jdata[ 'req ']== 'SIGNAL_PAIR ':
        Lock.Lock_Discovery_TCP(jdata)
##### LOCK control
    elif jdata[ 'req ']== 'LOCK_CONTROL ':
        _action = jdata["action"]
        Lock.Lock_Control(_action, self.myFile)
##### mDNS
    elif jdata[ 'req ']== 'whoiam ':
        _host = jdata[ 'hostname ']

```

```

        if _server_domain == _host:
            _server_ip = ip_connect
            self.writeFile("CONFIG", "server_ip", _server_ip)
    try:
        return response
    except:
        return '\n'

    self.myFile.readFile()

def reboot(self,delay) :
    if delay!=0:
        time.sleep(delay)
        os.system('sudo reboot')

```

**Key.py**

```

import threading
class Key:
    def __init__(self):
        self.LED_key = threading.Event()
        self.RF_key = threading.Event()
        self.WRN_key = threading.Event()
        self.APLAY_key = threading.Event()
        self.OPEN_key = threading.Event()
        self.LOCK_key = threading.Event() #set: open, #clear: close
        self.LED_bell_key = threading.Event() #set: free #clear: belling
        self.FIN_key = threading.Event() #set: free #clear: belling

    def Wait_FIN_key(self):
        self.FIN_key.wait() #Pass if FIN_Key is set --- Free
        self.FIN_key.clear()

```

**myGPIO.py**

```

import Constant
class myGPIO:
    def __init__(self, GPIO, myFile):
        self.GPIO = GPIO
        self.myFile = myFile

    def setupOneLed(self) :
        _mode = self.myFile._mode
        if _mode == 'STATION':
            _default_signal_led=self.myFile._default_signal_led

            if _default_signal_led == '1':
                self.myFile.LED_PIN = '100'

```

```

        elif _default_signal_led == '2':
            self.myFile.LED_PIN = '110'
        elif _default_signal_led == '3':
            self.myFile.LED_PIN = '010'
        elif _default_signal_led == '4':
            self.myFile.LED_PIN = '001'
        elif _default_signal_led == '5':
            self.myFile.LED_PIN = '011'
        elif _default_signal_led == '6':
            self.myFile.LED_PIN = '101'
        elif _default_signal_led == '7':
            self.myFile.LED_PIN = '111'
        self.ledRGB(self.myFile.LED_PIN)
        print("[DBG] -----> setupOneLed: ", _default_signal_led)

def ledRGB(self, value):
    """Eg: ledRGB('101')"""
    _r = int(value[-3:-2])
    _g = int(value[-2:-1])
    _b = int(value[-1:])
    self.GPIO.output(Constant.LEDRD_PIN, _r)
    self.GPIO.output(Constant.LEDGREEN_PIN, _g)
    self.GPIO.output(Constant.LEDBLUE_PIN, _b)

```

## Network.py

```

import threading
import File
import ssl
import socket
import Constant
import json
import subprocess
import os
import time
import sys
import JSON_station
JANUS_STREAMING_CONFIG_PATH=Constant.JANUS_STREAMING_CONFIG_PATH
JANUS_AUDIOBRIDGE_CONFIG_PATH=Constant.JANUS_AUDIOBRIDGE_CONFIG_PATH
FNULL=Constant.FNULL
BUFFER_SIZE=Constant.BUFFER_SIZE
FIN_MODE=Constant.FIN_MODE
class socket_server threading (threading.Thread):
    #Create variable to be passed
    def __init__(self, threadID, name, myFile, Key, myGPIO, myTimer):
        #Decode variable into local variables
        threading.Thread.__init__(self)
        self.threadID = threadID

```

```
self.name = name
self.myFile = myFile
self.Key = Key
self.myGPIO = myGPIO
self.myTimer = myTimer

def socketServer(self) :
    station = self.myFile.station
    _mode = self.myFile.station
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.bind(('', Constant.PORT_AP))
        sock.listen(2)

        self.myFile.station = 'SUCCESS'
        print('Station Success')
        while True:
            conn = None
            ssock, addr = sock.accept()
            try:
                # if _mode=='ACCESS_POINT':
                #     handle(ssock, addr)
                # else:
                #     conn = context.wrap_socket(ssock, server_side=True)
                #     handle(conn, addr)
                handle(ssock, addr)
            except ssl.SSLError as e:
                print(e)
            finally:

                # if _mode=='ACCESS_POINT':
                #     if ssock:
                #         ssock.close()
                # else:
                #     if conn:
                #         conn.close()
            if conn:
                conn.close()
    except:
        print("Unexpected error:", sys.exc_info()[0])
        raise
def writeFile(self, section, key, value):
    self.myFile.writeFile(section, key, value)
def getInterfaceCard(self):
    self.myFile.getInterfaceCard()
def authentication(self):
    self.myFile.authentication()
def replace_key(self):
```

```

        self.myFile.replace_key()
def handle(self, conn, addr):
    _mode=self.myFile._mode
    _pin=self.myFile._pin
    login_fail_map=self.myFile.login_fail_map
    _alarm=self.myFile._alarm
    _alarm_auto_switch_off=self.myFile._alarm_auto_switch_off
    _alarm_auto_switch_off_s=self.myFile._alarm_auto_switch_off_s
    timeout_alarm=self.myFile.timeout_alarm
    _snapshot_if_press=self.myFile._snapshot_if_press
    _snapshot_manual=self.myFile._snapshot_manual
    _snapshot_if_attacked=self.myFile._snapshot_if_attacked
    ip_login=self.myFile.ip_login
    interface_card_map=self.myFile.interface_card_map
    _default_signal_led=self.myFile._default_signal_led
    _setup_media_store=self.myFile._setup_media_store
    _ftp_domain=self.myFile._ftp_domain
    _ftp_user=self.myFile._ftp_user
    _ftp_password=self.myFile._ftp_password
    _interactive_mode=self.myFile._interactive_mode

FNULL=Constant.FNULL
BUFFER_SIZE=Constant.BUFFER_SIZE
_server_ip=self.myFile._server_ip
#{"req" : "ALARM", 'value':'TRUE'}
if _mode=='ACCESS_POINT':
    data_rec = conn.recv(BUFFER_SIZE)
else:
    data_rec = conn.recv()
print(data_rec)
json_data_rec = data_rec.decode('utf-8')
try:
    jdata = json.loads(json_data_rec)
except:
    print('[EXCEPTION] Wrong JSON format')
    return
address = str(addr)
ip_login = address[address.index("\'")+1:address.index(",")-1]
response = '\n'
if _mode == 'PAIR' :
    self.writeFile('CONFIG','server_ip',ip_login)
    _server_ip = ip_login
    if jdata['req'] == 'AUTHENTICATION_CAMERA' :
        self.getInterfaceCard()
        if self.authentication(jdata):
            jsonRes = {}
            jsonRes['res'] = 'SUCCESS'

```

```

        jsonInterface = {}
        for i in interface_card_map:
            if(i=='eth0') :
                jsonInterface[i] = 'lan'
            else:
                jsonInterface[i] = 'wifi'
        jsonRes['interface'] = jsonInterface
        response = '%s\n'%(json.dumps(jsonRes))
    else :
        jsonRes = {}
        jsonRes['res'] = 'FAIL'
        response = '%s\n'%(json.dumps(jsonRes))
    elif jdata[ 'req' ] == "PAIR" :
        (response, _change_ip)= pair(jdata)
        print('[LOG] Pairing')
        conn.write(response.encode())
        if _change_ip == True:
            self.changeIP(jdata)
        return
        #os.system("reboot")
        conn.write(response.encode())
        return
    elif _mode == 'ACCESS_POINT': #ACCESS POINT + SMARTPHONE --> IP
        if jdata['req'] == 'DEVICE':
            jsonRes = {}
            jsonRes['res'] = 'DEVICE'
            jsonRes[ 'type' ] = 'DOOR_CAMERA_WF'
            response = '%s\n'%(json.dumps(jsonRes))
            conn.send(response.encode())
        elif jdata[ 'req' ] == 'NETWORK':
            self.changeNetwork(jdata)
            subprocess.call('python3 ./boot_network.py wf -
inet dhcp',shell=True)
            jsonRes = {}
            jsonRes[ 'res' ] = 'NETWORK'
            response = '%s\n'%(json.dumps(jsonRes))
            conn.send(response.encode())
            self.writeFile('CONFIG', 'mode', 'PAIR')
            print("----Reboot----")
            self.reboot(0)
        return
    elif _mode == 'STATION' :
        response = JSON_station.JSON_station(self.myFile, self.Key, self.myGP
IO, self.myTimer).JSON_station(conn,jdata)
        conn.write(response.encode())
        return

def reboot(self,delay) :

```

```

if delay!=0:
    time.sleep(delay)
os.system('sudo reboot')

def changeIP(self, jdata) :
    '''Mini version of change Network'''
    print('-----Change IP-----')
    self.getInterfaceCard()
    _wifi_card_name=self.myFile._wifi_card_name
    _interface=self.myFile._interface
    network_interfaces_path=self.myFile.network_interfaces_path
    _ip_address = jdata['ip_address']
    _is_staic = jdata['is_static']
    if _is_staic == True:
        _inet = "static"
    else:
        _inet = "dhcp"
    if _interface == "WIRE":
        call_out_cmd = "python3 ./boot_network.py lan -inet "+_inet+" -wip "+_ip_address
        print("[CALL] callout cmd: ",call_out_cmd)
        subprocess.call(call_out_cmd, shell=True)
        os.system('ifconfig eth0 %s'%(_ip_address))
    else:
        call_out_cmd = "python3 ./boot_network.py wf -inet "+_inet+" -wip "+_ip_address
        print("[CALL] callout cmd: ",call_out_cmd)
        subprocess.call(call_out_cmd, shell=True)
        os.system('ifconfig %s %s'%(_wifi_card_name, _ip_address))

def changeNetwork(self, jdata) :
    self.getInterfaceCard()
    _mode=self.myFile._mode
    _pin=self.myFile._pin
    _server_ip=self.myFile._server_ip
    login_fail_map=self.myFile.login_fail_map
    ip_login=self.myFile.ip_login
    _wifi_card_name=self.myFile._wifi_card_name
    network_interfaces_path=self.myFile.network_interfaces_path
# try :
#     door_dhcp_ip = jdata['camera_host_door']
    if _mode == 'ACCESS_POINT':
        interface_card = jdata['type']
        # door_static_ip = jdata['ip_static']
        ip_configuration = jdata['ip']
        ip_configuration = ip_configuration.lower()
        door_default_gateway = jdata['gateway']
        door_wireles_network_name = jdata['ssid']

```

```

        door_wireles_network_key = jdata['password']
        door_network_mark_len = jdata['network_prefix_len']
        if door_network_mark_len == '24':
            door_network_mark = '255.255.255.0'
        elif door_network_mark_len == '16':
            door_network_mark = '255.255.0.0'
        elif door_network_mark_len == '8':
            door_network_mark = '255.0.0.0'
        else:
            interface_card = jdata['interface_card']
            # door_static_ip = jdata['door_static_ip']
            ip_configuration = jdata['ip_configuration']
            door_default_gateway = jdata['door_default_gateway']
            door_wireles_network_name = jdata['door_wireles_network_name']
            door_wireles_network_key = jdata['door_wireles_network_key']
            door_network_mark = jdata['door_network_mark']

            print('-----setup-----')
            print(interface_card)
            print(door_wireles_network_name)
            print(door_wireles_network_key)
            # print(door_static_ip)
            print(ip_configuration)
            print(door_network_mark)
            print(door_default_gateway)

            if (interface_card.find('eth') > -1) :
                print("Change network to ----> LAN")
                self.replace_key(network_interfaces_path,"allow-hotplug ","allow-
hotplug %s\n"%(interface_card))
                self.replace_key(network_interfaces_path,"iface ","iface %s inet
dhcp\n"%(interface_card))
                self.replace_key(network_interfaces_path, "netmask ","netmask %s\
n"%(door_network_mark))
                self.replace_key(network_interfaces_path,"gateway ","gateway %s\n
"%(door_default_gateway))
                self.replace_key(network_interfaces_path,"wpa-conf ","#wpa-
conf /etc/wpa_supplicant/wpa_supplicant.conf\n")
                self.writeFile('CONFIG', 'interface', 'WIRE')
                # os.system('ifconfig %s %s'%(interface_card, door_static_ip))
                os.system("ifconfig "+_wifi_card_name+" down");
                os.system("ifconfig eth0 up");
            else: #wifi
                print("Change network to ----> WIFI")
                self.replace_key(network_interfaces_path,"allow-hotplug ","allow-
hotplug %s\n%(_wifi_card_name)")
                file = open('/etc/wpa_supplicant/wpa_supplicant.conf', 'w');

```

```

        file.write('ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netd
ev\n');
        file.write('\n');
        file.write('update_config=1\n');
        file.write('network={\n');
        file.write('\tssid="%s"\n'%(door_wireles_network_name));
        file.write('\tpsk="%s"\n'%(door_wireles_network_key));
        file.write('\tkey_mgmt=WPA-PSK\n');
        file.write('}');
        file.close()
        self.replace_key(network_interfaces_path,"allow-hotplug ","allow-
hotplug %s\n"%(interface_card))
        self.replace_key(network_interfaces_path,"iface ","iface %s inet
dhcp\n"%(interface_card))
        self.replace_key(network_interfaces_path, "netmask ","netmask %s\
n"%(door_network_mark))
        self.replace_key(network_interfaces_path,"gateway ","gateway %s\n
"%(door_default_gateway))
        self.writeFile('CONFIG', 'interface', 'WIFI')
        self.replace_key(network_interfaces_path,"wpa-conf ","wpa-
conf /etc/wpa_supplicant/wpa_supplicant.conf\n")
        if _mode == 'ACCESS_POINT':
            os.system("killall -9 hostapd");
        else:
            os.system("ifconfig "+_wifi_card_name+" up");
            os.system("ifconfig eth0 down");

def pair(self, jdata) :
    _mode=self.myFile._mode
    _pin=self.myFile._pin
    _server_ip=self.myFile._server_ip
    _server_domain=self.myFile._server_domain
    login_fail_map=self.myFile.login_fail_map
    ip_login=self.myFile.ip_login
    _default_signal_led=self.myFile._default_signal_led
    _bell_sound=self.myFile._bell_sound
    lock_firm=self.myFile.lock_firm
    lock_IP=self.myFile.lock_IP
    lock_seri=self.myFile.lock_seri
    lock_ssid=self.myFile.lock_ssid
    lock_type=self.myFile.lock_type
    serial_neo2=self.myFile.serial_neo2
    response = '\n'
    _change_ip = False
    try :
        password_ = jdata['password']
        serial_ = jdata['serial']

```

```

        new_password_ = jdata['new_password']
        isNetwork_ = jdata['isNetwork']
        _server_domain = jdata['hostname']
        # ip_address_ = jdata['ip_address']
        # is_static_ = jdata['is_static']
        _doorcamera_id = jdata['id']
        _doorcamera_name = jdata['doorcamera_name']
        print("[DBG] Parse json done") #DBG
        print(password_)
        print(_pin)
        print(serial_neo2)
        print(serial_)
        if password_ == _pin and serial_neo2 == serial_:
            isContinue = False
            try:
                if login_fail_map[ip_login] < time.time() :
                    isContinue = True
            except KeyError:
                isContinue = True
                pass # do nothing!
        if isContinue : # True
            print("[DBG] Auth ---> True")
            if isNetwork_ == True:
                print("[DBG] Is Network")
                _change_ip = True
            if _pin != new_password_ :
                writeFile('SECURITY', 'pin', new_password_)
            # changeNetwork(jdata)
            # try:
            #     new_password = jdata['new_password']
            #     if _pin != new_password :
            #         writeFile('SECURITY', 'pin', pin)
            #         try :
            #             config_janus_room.set(str(room_id), 'pin', pin)
            #             with open(config_janus_room, 'wb') as configfil
e_:
            #             config_janus_room.write(configfile_)
            #             _pin = pin
            #             print('Change pin and restart janus')
            #             # os.system("pkill janus")## se co THREAD Auto
ON janus
            #         except NoOptionError:
            #             pass
            #     except KeyError:
            #         isContinue = True
            #         print("Key ERR") #DBG
            #         pass # do nothing!
            writeFile('CONFIG', 'doorcamera_id', str(_doorcamera_id))

```

```

        writeFile('CONFIG', 'server_ip', _server_ip)
        writeFile('CONFIG', 'doorcamera_name', _doorcamera_name)
        writeFile('CONFIG', 'mode', 'STATION')
        writeFile('CONFIG', 'server_domain', _server_domain)
        replace_key(JANUS_STREAMING_CONFIG_PATH, "rtsp-test-", "rtsp-
test-%s: {\n%" + (_doorcamera_id))
        replace_key(JANUS_STREAMING_CONFIG_PATH, "id =", "\tid = \\" + %
" + "\n%" + (_doorcamera_id))
        replace_key(JANUS_STREAMING_CONFIG_PATH, "pin =", "\tpin = \\" +
" + "\n%" + (new_password_))
        replace_key(JANUS_AUDIOBRIDGE_CONFIG_PATH, "pin =", "\tpin = \\" +
" + "\n%" + (new_password_))
        os.system("systemctl enable janus.service")
        os.system("systemctl restart janus.service")
        _mode = 'STATION'
        jsonRes = {}
        jsonRes['res'] = 'PAIR'
        if lock_IP != "NOT_PAIR":
            jsonRes['isLock'] = True
            jsonRes['ip_address'] = lock_IP
            jsonRes['firmware'] = lock_firm
            jsonRes['type'] = lock_type
            jsonRes['ssid'] = lock_ssid
            jsonRes['serial'] = lock_seri
            response = '%s\n' % (json.dumps(jsonRes))
            # GPIO.output(BIP_PIN, False)
            # subprocess.call('aplay ./DoorVillaSounds/DoorBell/' + _bell_s
ound, shell = True, stdout=DEVNULL); ## Used as BIP_PIN
            # GPIO.output(LEDBLUE_PIN, True)
            # GPIO.output(LEDRED_PIN, False)
            self.myGPIO.setupOneLed()
            print("Write Done")
            # thread.start_new_thread(reboot, (3,))
            # threading.Thread(target = reboot, args = (3,)).start()
            # print("Reboot")
            self.myFile.readFile();
        else :
            jsonRes = {}
            jsonRes['res'] = 'FAIL'
            response = '%s\n' % (json.dumps(jsonRes))
            print('authentication1')
        else :
            jsonRes = {}
            jsonRes['res'] = 'FAIL'
            response = '%s\n' % (json.dumps(jsonRes))
            print('authentication2')
    except KeyError as e: print(e)
    return (response, _change_ip)

```

```
def run(self):
    self.socketServer()
```

### SocketClient.py

```
import ssl
import socket
import Constant
import File
import threading
CERT=Constant.CERT
PASSWORD=Constant.PASSWORD
PORT=Constant.PORT
PORT_AP=Constant.PORT_AP
import subprocess
def startWhoIAm(myFile):
    _server_ip=myFile._server_ip
    _server_domain=myFile._server_domain
    output = subprocess.check_output("avahi-resolve-host-
name " + _server_domain, shell=True)
    try:
        output = output.decode()
        output = output.split('\t')
        output = output[1][:-1]
        print("IP detect for domain ", _server_domain, " is ", output)
        myFile._server_ip = output
        myFile.writeFile("CONFIG", "server_ip", _server_ip)
    except:
        print("NO SERVER WITH THAT HOST NAME DETECTED")
        return
    _my_domain = subprocess.check_output("cat /etc/hostname", shell=True)
    _my_domain = _my_domain.decode().replace('\n','') +".local"
    _start_up_msg = "{\"req\":\"whoiam\", \"hostname\":\"" + _my_domain+"\"}"
    _start_up_msg = _start_up_msg.encode()
    threading.Thread(target=sslClient, args=(_start_up_msg, _server_domain, False
,)).start()

def sslClient(message, ip_connect, startWhoIam=False):
    tcpClient(message, ip_connect)
    return

def tcpClient(message, ip_connect):
    tcp = socket.socket(socket.AF_INET)
    tcp.connect((ip_connect, PORT_AP))
    tcp.send(message + b'\n')
    data = tcp.recv(1024)
    print(data)
    if tcp:
```

```
tcp.close()  
print("[LOG] send TCP msg: ",str(message)," to ",ip_connect)
```

## Sound.py

```

import subprocess
import time
import threading
import SocketClient
class Sound:
    def __init__(self, Key, myFile):
        self.Key = Key
        self.myFile=File

    def alarm_sound(self):
        self.Key.WRN_key.set()
        while self.Key.WRN_key.is_set():
            subprocess.call('aplay ./temp/alarm.wav', shell=True)

    def attackBell(self) :
        WRN_key = self.Key.WRN_key
        timeout_alarm=self.myFile.timeout_alarm
        _doorcamera_id=self.myFile._doorcamera_id
        ip_login=self.myFile.ip_login
        while timeout_alarm > time.time() :
            time.sleep(1)
        # GPIO.output(WARN_PIN, False)
        if WRN_key.is_set():
            WRN_key.clear()
            subprocess.call("killall -9 aplay", shell=True)
            temp_msg="{\"res\": \"ALARM_OFF\", \"id\": \"{}\"}".format(_doorcamera_id)
            temp_msg=temp_msg.encode()
            threading.Thread(target = SocketClient.sslClient, args = (temp_msg,ip_login,)).start()

```

## Timer.py

```

import threading
import Constant
##### Period Timer #####
class PeriodTimer(threading.Thread):
    """Call a function after a specified number of seconds:

        t = PeriodTimer(30.0, f, args=None, kwargs=None)
        t.start()
        t.reset()      # count from beginning
    """
    def __init__(self, interval, function, args=None, kwargs=None):
        threading.Thread.__init__(self)
        self.interval = interval

```

```
        self.function = function
        self.args = args if args is not None else []
        self.kwargs = kwargs if kwargs is not None else {}
        self.finished = threading.Event()

    def reset(self):
        """Stop the timer if it hasn't finished yet."""
        self.finished.set()

    def run(self):
        while True:
            self._sleep()
            self.finished.clear()
            self.finished.wait(self.interval)
            if self.finished.is_set():
                print('[TIMER] Reset')
                self.run()
            else:
                self.function(*self.args, **self.kwargs)
                self.finished.clear()

    def _sleep(self):
        """sleep until set()"""
        while not self.finished.is_set():
            self.finished.wait(1000)

    def update_counter_time(self, new_time):
        """Change time setting"""
        self.interval = new_time

class myTimer:
    def __init__(self, myFile):
        self.Streaming_Timer = PeriodTimer(Constant.STREAMING_PERIOD, self.apache_end_period)
        self.LockOpen_Timer = PeriodTimer(int(myFile.lock_open_time), self.Lock_close_event)
        self.Streaming_Timer.start()
        self.LockOpen_Timer.start()
    def Lock_close_event(self):
        print("[Lock_close_event]--> Chua sua xong")
    def apache_end_period(self):
        print("[apache_end_period]--> Chua sua xong")
```

## 6.6 File cấu hình Janus

### Event handler

```

# This configures the MQTT event handler. Events are sent either on
# one topic or on a topic per event type.
#
# By default, configuration topics for handle and webrtc event types
# with the base topic are configured to /janus/events, e.g.:
#     /janus/events/handle
#     /janus/events/webrtc

general: {
    #enabled = true                      # By default the module is not
    enabled                                # Comma separated list of the events
    events = "all"                         # in. valid values
    mask you're interested                  # media, plugins,
                                              # default we
    subscribe to everything (all)          # whether the JSON messages should be
    json = "indented"                     # plain (no
    indented (default),                   # indentation) or compact (no indentation and no spaces)

    url = "tcp://localhost:1883"           # The URL of the MQTT server. Only
    tcp supported at this time.
    client_id = "janus.example.com"       # Janus client id. You have to
    configure a unique ID (default: guest).
    #keep_alive_interval = 20             # Keep connection for N
    seconds (default: 30)                 # Clean session flag
    #cleansession = 0                    # Default
    (default: off)
    #retain = 0                          # Default
    MQTT retain flag for published events
    #qos = 1                            # Default
    MQTT QoS for published events
    #disconnect_timeout = 100            # Seconds to wait before
    destroying client
    #username = "guest"                 # Username for
    authentication (default: no authentication)
    #password = "guest"                 # Password for
    authentication (default: no authentication)
    topic = "/janus/events"            # Base topic (default: /janus/events)
    #addevent = true                   # Whether we should
    add the event type to the base topic

    # Initial message sent to status topic
    #connect_status = "{\"event\": \"connected\", \"eventhandler\":"
    \"janus.eventhandler.mqttevh\"}"
    # Message sent after disconnect or as LWT
    #disconnect_status = "{\"event\": \"disconnected\"}"

    #will_enabled = false               # When
    #will_retain = 1
    #will_qos = 0

    # Additional parameters if "mqtts://" schema is used
    #tls_verify_peer = true            # Whether peer verification
must be enabled
    #tls_verify_hostname = true        # Whether hostname
verification must be enabled

```

```

# Certificates to use when SSL support is enabled, if needed
#tls_cacert = "/path/to/cacert.pem"
#tls_client_cert = "/path/to/cert.pem"
#tls_client_key = "/path/to/key.pem"
#tls_ciphers
#tls_version
}

```

## Janus

```

# General configuration: folders where the configuration and the plugins
# can be found, how output should be logged, whether Janus should run as
# a daemon or in foreground, default interface to use, debug/logging level
# and, if needed, shared apisecret and/or token authentication mechanism
# between application(s) and Janus.
general: {
    configs_folder = "/opt/janus/etc/janus"                      #
Configuration files folder
    plugins_folder = "/opt/janus/lib/janus/plugins"                #
Plugins folder
    transports_folder = "/opt/janus/lib/janus/transports"      # Transports
folder
    events_folder = "/opt/janus/lib/janus/events"                 #
Event handlers folder
    loggers_folder = "/opt/janus/lib/janus/loggers"              #
External loggers folder

        # The next settings configure logging
        #log_to_stdout = false                                     # whether the
Janus output should be written

        #log_to_file = "/path/to/janus.log"                      # whether to use a
log file or not
        debug_level = 4                                         #
Debug/logging level, valid values are 0-7
        #debug_timestamps = true                                # whether to
show a timestamp for each log line
        #debug_colors = false                                 # whether
colors should be disabled in the log
        #debug_locks = true                                  #
whether to enable debugging of locks (very verbose!)

        # This is what you configure if you want to launch Janus as a
daemon
        #daemonize = true                                    #
whether Janus should run as a daemon

        #pid_file = "/path/to/janus.pid"                      # PID file to create
when Janus has been

        # There are different ways you can authenticate the Janus and
Admin APIs
        #api_secret = "janusrocks"                         # String that all Janus
requests must contain
                                                               #
be accepted/authorized by the Janus core.                    #
                                                               #
useful if you're wrapping all Janus API requests          #
                                                               #
your servers (that is, not in the browser,                  #
                                                               #
where you do the things your way) and you                 #
                                                               #
don't want other application to mess with               #
                                                               #
this Janus instance.                                     #

```

```

        #token_auth = true                                # Enable a token
based authentication                                #
mechanism to force users to always provide        #
valid token in all requests. Useful if           #
want to authenticate requests from web          #
users.                                              #
#token_auth_secret = "janus"      # Use HMAC-SHA1 signed tokens (with
token_auth). Note that                           #
without this, the Admin API MUST                 #
enabled, as tokens are added and removed         #
through messages sent there.                      #
admin_secret = "janusoverlord" # String that all Janus requests must
contain                                         #
be accepted/authorized by the admin/monitor.    #
only needed if you enabled the admin API        #
any of the available transports.                 #

        # Generic settings
#interface = "1.2.3.4"                          # Interface to use (will be
used in SDP)
#server_name = "MyJanusInstance" # Public name of this Janus instance
# as
it will appear in an info request
#session_timeout = 60                            # How long (in seconds) we
should wait before
deciding a Janus session has timed out. A
session times out when no request is received
# for
# session_timeout seconds (default=60s).
Setting this to 0 will disable the timeout
mechanism, which is NOT suggested as it may
risk having orphaned sessions (sessions not
controlled by any transport and never freed).
# To
# avoid timeouts, keep-alives can be used.
#candidates_timeout = 45                         # How long (in seconds) we
should keep hold of
pending (trickle) candidates before discarding
them (default=45s). Notice that setting this
0 will NOT disable the timeout, but will
# be
considered an invalid value and ignored.
#reclaim_session_timeout = 0      # How long (in seconds) we should
wait for a
janus session to be reclaimed after the transport
# is
gone. After the transport is gone, a session

```

```

times out when no request is received for                                #
reclaim_session_timeout seconds (default=0s).                            #
Setting this to 0 will disable the timeout                                #
mechanism, and sessions will be destroyed immediately                   #
the transport is gone.                                                    # if
#recordings_tmp_ext = "tmp"          # The extension for
recordings, in Janus, is
.mjr, a custom format we devised ourselves.                               #
default, we save to .mjr directly. If you'd                            #
rather the recording filename have a temporary                         #
extension while it's being saved, and only                           #
have the .mjr extension when the recording                          #
over (e.g., to automatically trigger some                         # is
external scripts), then uncomment and set the                         #
recordings_tmp_ext property to the extension                         #
# to
add to the base (e.g., tmp --> .mjr.tmp).                            # By default, Janus
#event_loops = 8
handles each have their own
event loop and related thread for all the media
routing and management. If for some reason you'd
rather limit the number of loop/threads, and
want handles to share those, you can do that
configuring the event_loops property: this will
spawn the specified amount of threads at startup,
a separate event loop on each of them, and
new handles to one of them when attaching.
Notice that, while cutting the number of threads
possibly reducing context switching, this
might have an impact on the media delivery,
especially if the available loops can't take
care of all the handles and their media in time.
such, if you want to use this you should
provision the correct value according to the
available resources (e.g., CPUs available).
#opaqueid_in_api = true          # Opaque IDs set by
applications are typically
only passed to event handlers for correlation
#

```

```

purposes, but not sent back to the user or
application in the related Janus API responses
events; in case you need them to be in the
Janus API too, set this property to 'true'.
    #hide_dependencies = true                      # By default, a call to the
"info" endpoint of
either the Janus or Admin API now also returns
versions of the main dependencies (e.g.,
libnice, libsrtp, which crypto library is in
and so on). Should you want that info not
be disclose, set 'hide_dependencies' to true.

    # The following is ONLY useful when debugging RTP/RTCP
packets,
    # e.g., to look at unencrypted live traffic with a browser.
By
    # default it is obviously disabled, as WebRTC mandates
encryption.
    #no_webRTC_encryption = true

    # Janus provides ways via its API to specify custom paths to
save
    # files to (e.g., recordings, pcap captures and the like). In
order
    # to avoid people can mess with folders they're not supposed
to,
    # you can configure an array of folders that Janus should
prevent
    # creating files in. By default no folder is protected,
unless the
    # 'protected_folders' property below is uncommented and
configured.
    # Notice that at the moment this only covers attempts to
start
    # an .mjr recording and pcap/text2pcap packet captures.
protected_folders = [
    "/bin",
    "/boot",
    "/dev",
    "/etc",
    "/initrd",
    "/lib",
    "/lib32",
    "/lib64",
    "/proc",
    "/root",
    "/sbin",
    "/sys",
    "/usr",
    "/var",
    # we add what are usually the folders Janus is
installed to
    # as well: we don't just put "/opt/janus" because
that would
    # include folders like "/opt/janus/share" that is
where
    # recordings might be saved to by some plugins
    "/opt/janus/bin",
    "/opt/janus/etc",

```

```

        "/opt/janus/include",
        "/opt/janus/lib",
        "/opt/janus/lib32",
        "/opt/janus/lib64",
        "/opt/janus/sbin"
    ]
}

# Certificate and key to use for DTLS (and passphrase if needed). If missing,
# Janus will autogenerate a self-signed certificate to use. Notice that
# self-signed certificates are fine for the purpose of WebRTC DTLS
# connectivity, for the time being, at least until Identity Providers
# are standardized and implemented in browsers.
certificates: {
    #cert_pem = "/path/to/certificate.pem"
    #cert_key = "/path/to/key.pem"
    #cert_pwd = "secretpassphrase"
}

# Media-related stuff: you can configure whether if you want
# to enable IPv6 support, the minimum size of the NACK queue (in ms,
# defaults to 200ms) for retransmissions no matter the RTT, the range of
# ports to use for RTP and RTCP (by default, no range is envisaged), the
# starting MTU for DTLS (1200 by default, it adapts automatically),
# how much time, in seconds, should pass with no media (audio or
# video) being received before Janus notifies you about this (default=1s,
# 0 disables these events entirely), how many lost packets should trigger
# a 'slowlink' event to users (default=4), and how often, in milliseconds,
# to send the Transport Wide Congestion Control feedback information back
# to senders, if negotiated (default=200ms). Finally, if you're using
BoringSSL
# you can customize the frequency of retransmissions: OpenSSL has a fixed
# value of 1 second (the default), while BoringSSL can override that. Notice
# that lower values (e.g., 100ms) will typically get you faster connection
# times, but may not work in case the RTT of the user is high: as such,
# you should pick a reasonable trade-off (usually 2*max expected RTT).
media: {
    #ipv6 = true
    #min_nack_queue = 500
    #rtp_port_range = "20000-40000"
    rtp_port_range = "60000-61000"
    #dtls_mtu = 1200
    #no_media_timer = 1
    #slowlink_threshold = 4
    #twcc_period = 100
    #dtls_timeout = 500
}

# NAT-related stuff: specifically, you can configure the STUN/TURN
# servers to use to gather candidates if the gateway is behind a NAT,
# and srflx/relay candidates are needed. In case STUN is not enough and
# this is needed (it shouldn't), you can also configure Janus to use a
# TURN server# please notice that this does NOT refer to TURN usage in
# browsers, but in the gathering of relay candidates by Janus itself,
# e.g., if you want to limit the ports used by a Janus instance on a
# private machine. Furthermore, you can choose whether Janus should be
# configured to do full-trickle (Janus also trickles its candidates to
# users) rather than the default half-trickle (Janus supports trickle
# candidates from users, but sends its own within the SDP), and whether
# it should work in ICE-Lite mode (by default it doesn't). Finally,
# you can also enable ICE-TCP support (beware that it currently *only*
# works if you enable ICE Lite as well), choose which interfaces should
# be used for gathering candidates, and enable or disable the
# internal libnice debugging, if needed.
nat: {
#stun_server = "stun.1.google.com"
#stun_port = 19302
    stun_server = "stun.voip.eutelia.it"
}

```

```

stun_port = 3478
nice_debug = false
#full_trickle = true
#ice_lite = true
#ice_tcp = true

# In case you're deploying Janus on a server which is configured with
# a 1:1 NAT (e.g., Amazon EC2), you might want to also specify the
public
# address of the machine using the setting below. This will result in
# all host candidates (which normally have a private IP address) to
# be rewritten with the public address provided in the settings. As
# such, use the option with caution and only if you know what you're
doing.
# Make sure you keep ICE Lite disabled, though, as it's not strictly
# speaking a publicly reachable server, and a NAT is still involved.
#nat_1_1_mapping = "1.2.3.4"

# You can configure a TURN server in two different ways: specifying a
# statically configured TURN server, and thus provide the address of
the
# TURN server, the transport (udp/tcp/tls) to use, and a set of valid
# credentials to authenticate...
#turn_server = "192.168.1.103"
#turn_port = 3478
#turn_type = "udp"
#turn_user = "123"
#turn_pwd = "123"

# ... or you can make use of the TURN REST API to get info on one or
more
# TURN services dynamically. This makes use of the proposed standard
of
# such an API (https://tools.ietf.org/html/draft-uberti-behave-turn-
rest-00>)
# which is currently available in both rfc5766-turn-server and
coturn.
# You enable this by specifying the address of your TURN REST API
backend,
# the HTTP method to use (GET or POST) and, if required, the API key
Janus
# must provide.
#turn_rest_api = "http://yourbackend.com/path/to/api"
#turn_rest_api_key = "anyapikeyyoumayhaveset"
#turn_rest_api_method = "GET"

# You can also choose which interfaces should be explicitly used by
the
# gateway for the purpose of ICE candidates gathering, thus excluding
# others that may be available. To do so, use the 'ice_enforce_list'
# setting and pass it a comma-separated list of interfaces or IP
addresses
# to enforce. This is especially useful if the server hosting the
gateway
# has several interfaces, and you only want a subset to be used. Any
of
# the following examples are valid:
# ice_enforce_list = "eth0"
# ice_enforce_list = "eth0,eth1"
# ice_enforce_list = "eth0,192.168."
# ice_enforce_list = "eth0,192.168.0.1"
# By default, no interface is enforced, meaning Janus will try to use
them all.
#ice_enforce_list = "eth0"

# In case you don't want to specify specific interfaces to use, but
would

```

```

        # rather tell Janus to use all the available interfaces except some
that
        # you don't want to involve, you can also choose which interfaces or
IP
        # addresses should be excluded and ignored by the gateway for the
purpose
        # of ICE candidates gathering. To do so, use the 'ice_ignore_list'
setting
        # and pass it a comma-separated list of interfaces or IP addresses to
        # ignore. This is especially useful if the server hosting the gateway
        # has several interfaces you already know will not be used or will
simply
        # always slow down ICE (e.g., virtual interfaces created by VMware).
        # Partial strings are supported, which means that any of the
following
        # examples are valid:
        #     ice_ignore_list = "vmnet8,192.168.0.1,10.0.0.1"
        #     ice_ignore_list = "vmnet,192.168."
        # Just beware that the ICE ignore list is not used if an enforce list
        # has been configured. By default, Janus ignores all interfaces whose
        # name starts with 'vmnet', to skip VMware interfaces:
ice_ignore_list = "vmnet"

        # In case you want to allow Janus to start even if the configured
STUN or TURN
        # server is unreachable, you can set 'ignore_unreachable_ice_server'
to true.
        # WARNING: We do not recommend to ignore reachability problems,
particularly
        # if you run Janus in the cloud. Before enabling this flag, make sure
your
        # system is correctly configured and Janus starts after the network
layer of
        # your machine is ready. Note that Linux distributions offer such
directives.
        # You could use the following directive in systemd: 'After=network-
online.target'
        #
https://www.freedesktop.org/software/systemd/man/systemd.unit.html#Before=
        # ignore_unreachable_ice_server = true
}

# You can choose which of the available plugins should be
# enabled or not. Use the 'disable' directive to prevent Janus from
# loading one or more plugins: use a comma separated list of plugin file
# names to identify the plugins to disable. By default all available
# plugins are enabled and loaded at startup.
plugins: {
    disable = "libjanus_voicemail.so,libjanus_recordplay.so"
}

# You can choose which of the available transports should be enabled or
# not. Use the 'disable' directive to prevent Janus from loading one
# or more transport: use a comma separated list of transport file names
# to identify the transports to disable. By default all available
# transports are enabled and loaded at startup.
transports: {
    disable = "libjanus_rabbitmq.so"
}

# As a core feature, Janus can log either on the standard output, or to
# a local file. Should you need more advanced logging functionality, you
# can make use of one of the custom loggers, or write one yourself. Use the
# 'disable' directive to prevent Janus from loading one or more loggers:
# use a comma separated list of logger file names to identify the loggers
# to disable. By default all available loggers are enabled and loaded at
startup.
loggers: {

```

```

        #disable = "libjanus_jsonlog.so"
}

# Event handlers allow you to receive live events from Janus happening
# in core and/or plugins. Since this can require some more resources,
# the feature is disabled by default. Setting broadcast to yes will
# enable them. You can then choose which of the available event handlers
# should be loaded or not. Use the 'disable' directive to prevent Janus
# from loading one or more event handlers: use a comma separated list of
# file names to identify the event handlers to disable. By default, if
# broadcast is set to yes all available event handlers are enabled and
# loaded at startup. Finally, you can choose how often media statistics
# (packets sent/received, losses, etc.) should be sent: by default it's
# once per second (audio and video statistics sent separately), but may
# be considered too verbose, or you may want to limit the number of events,
# especially if you have many PeerConnections active. To change this,
# just set 'stats_period' to the number of seconds that should pass in
# between statistics for each handle. Setting it to 0 disables them (but
# not other media-related events).
events: {
    #broadcast = true
    #disable = "libjanus_sampleevh.so"
    stats_period = 5
}

```

## Websocket Plugin

```

# WebSockets stuff: whether they should be enabled, which ports they
# should use, and so on.
general: {
    json = "indented"                                # Whether the JSON
    messages should be indented (default),
    plain (no indentation) or compact (no indentation and no spaces)
        #pingpong_trigger = 30                         # After how many seconds of
    idle, a PING should be sent
        #pingpong_timeout = 10                          # After how many seconds of
    not getting a PONG, a timeout should be detected

    ws = true                                         # Whether to
enable the WebSockets API
    ws_port = 8188                                     # WebSockets server
port
    #ws_interface = "eth0"                            # whether we should bind this
server to a specific interface only
    #ws_ip = "192.168.0.1"                           # whether we should bind this
server to a specific IP address only
    wss = false                                       # whether to
enable secure WebSockets
    #wss_port = 8989                                 # WebSockets server
secure port, if enabled
    #wss_interface = "eth0"                            # whether we should bind this
server to a specific interface only
    #wss_ip = "192.168.0.1"                           # whether we should bind this
server to a specific IP address only
    #ws_logging = "err, warn"                        # libwebsockets debugging
level as a comma separated list of things
    debug, supported values: err, warn, notice, info, debug, parser,
header, ext, client, latency, user, count (plus 'none' and 'all')
        #ws_acl = "127.,192.168.0."                  # Only allow requests coming
from this comma separated list of addresses
}

# If you want to expose the Admin API via WebSockets as well, you need to
# specify a different server instance, as you cannot mix Janus API and

```

```

# Admin API messaging. Notice that by default the Admin API support via
# WebSockets is disabled.
admin: {
    admin_ws = false                                # Whether to
    enable the Admin API WebSockets API
    admin_ws_port = 7188                            # Admin API
    webSockets server port, if enabled
        #admin_ws_interface = "eth0"
    server to a specific interface only
        #admin_ws_ip = "192.168.0.1"
    server to a specific IP address only
        admin_wss = false
    enable the Admin API secure WebSockets
        #admin_wss_port = 7989
    WebSockets server secure port, if enabled
        #admin_wss_interface = "eth0"
    server to a specific interface only
        #admin_wss_ip = "192.168.0.1"
    server to a specific IP address only
        #admin_ws_acl = "127.,192.168.0."
    from this comma separated list of addresses
}

# Certificate and key to use for any secure WebSocket server, if enabled (and
# passphrase if needed).
certificates: {
    #cert_pem = "/path/to/cert.pem"
    #cert_key = "/path/to/key.pem"
    #cert_pwd = "secretpassphrase"
}

```

## 6.7 File cấu hình Coturn Server

```

realm=coturn.meetrix.io
fingerprint
listening-ip=0.0.0.0
external-ip = 115.74.31.238/192.168.1.103
listening-port=3478
min-port=50000
max-port=59999
log-file=/var/log/turnserver.log
verbose

```

## 6.8 SourceCode Mạch Điều khiển thiết bị hồng ngoại

```

#include <time.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <sys/time.h>
#include <coredecls.h>
#include <EEPROM.h>
#include <WiFiUdp.h>
#include <Arduino.h>
#include <Hash.h>
#include <stdio.h>
#include "configure.h"
#include <ESP8266mDNS.h>
#include <string.h>

```

```
#include <ESP8266HTTPClient.h>
#include <ESP8266httpUpdate.h>
#include "IR_remote_support.h"
String VERSION_STR = "IRSTA_IR_V_LUANVAN_01";
String DEVICE_TYPE = "IRSTATION";
String myDomain = "";
String serDomain = "";

//Timer timeDelay(SYS_100HZ); // khai tao timer delay 0.1s

boolean Mode_Station = false;
char *Ssid = "";
char *Password = "";
char *Ssid_old = "";
char *Password_old = "";
String HomeServer_IPStr = "";

IPAddress ClientConnectAddr;
IPAddress DiaChi_HomeServer;
IPAddress dns_;
IPAddress staticDNS(8, 8, 8, 8);
IPAddress HomeServer_IP;
IPAddress DiaChi_ThietBi;
IPAddress default_gateway;
IPAddress subnet_mask;

uint8_t countReconnectWifi = 0;
String soSeri = "";
String GoiTin_HomeServer = "";
String GoiTin_ThietBi = "";

bool flagUpdateFw = false;
bool flagUpdated = false;

boolean isStatic = false;
boolean connectFirst = false;
boolean wifiDisConnect = false;

unsigned Long timeStartLearnWF = 0;
unsigned Long timeStartServerLearnWF = 0;
unsigned Long timeStartCheckConnect = 0;
unsigned Long timeStartCheckConnectWifi = 0;

unsigned Long dbg_tempo_timer = 0;

unsigned Long timeStartReConnect = 0;
unsigned Long timeStartActivePairing = 0;
```

```
unsigned Long timeStartRSTFac = 0;
unsigned Long timePreLedOn = 0;
unsigned Long timePreLedOff = 0;
unsigned Long timeCounterLed = 0;
unsigned Long timeToCheckIPChange = 0;
uint8_t counterStopFeedback = 0;

String ir_remote_ctrl = "";
boolean res_version_to_server;

WiFiUDP Udp;
WiFiServer server(TCP_PORT_SERVER);
WiFiClient client;
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wdeprecated-declarations"
#pragma GCC diagnostic pop
void buttonInterrupt(void);
void dieu_khien_nut_nhan(void);

void dieu_khien_Led(void);
void storeIP_tmp(String ip, int offset);
void storeIP(String StrIP, String StrGateway, String StrSubnet, String ipState);

void ClearEeprom();
void IRStation_Discovery();
void ReadEeprom();
void tien_hanh_ket_noi_HomeServer(char *ssid, char *password);
String gui_du_lieu_den_HomeServer_IP(String data, boolean fb);
String gui_du_lieu_den_IP_gan_nhat(String data, boolean fb);
void nhan_du_lieu_tu_HomeServer(void);
void startupStatusProcess();
boolean ResetFactory();
void checkCmd(void);
void accessUpdate(String host, String ver);
String splitStringJson(String strSignal, String scr);
void printStr(PGM_P s);
void restartSys();
void tien_hanh_dieu_khien_IR();
//For serial testing
//##DETECT_REMOTE_SERIAL## void IR_Detect_Serial(bool enable);
//##DETECT_REMOTE_SERIAL## bool flag_IR_Detect_Serial = false;
void kiem_tra_he_thong(void);
void resVerAfterUpdate();
void thiet_lap_mDns(void);
void SaveHost(String host);
String GetHost(void);
String gui_du_lieu_den_HomeServer_Domain(String data, boolean fb);
void isIPChanged();
```

```
char sbuf[MAX_BUF_REC_SERIAL];

void setup()
{
    Serial.begin(115200);
    while (!Serial)
    {
        delay(10);
    }
    ESP.wdtDisable();
    ESP.wdtEnable(TIME_WTD);
    EEPROM.begin(EEPROM_SIZE);
    delay(10);
    String mac = WiFi.macAddress();
    int j, i = 0;
    int len = mac.length();
    for (i = 0; i < len; i++)
    {
        if (mac[i] != ':')
        {
            soSeri += mac[i];
        }
    }
    myDomain = soSeri;
    pinMode(Led, OUTPUT);
    digitalWrite(Led, LED_ST_OFF);
    pinMode(Button, INPUT);
    attachInterrupt(Button, buttonInterrupt, CHANGE);
    delay(10);
    ReadEeprom();
    startupStatusProcess();

    if (Mode_Station && (IRdevice.state == INACTIVE_PAIRING || IRdevice.state == WORKING))
    {
        WiFi.mode(WIFI_STA);
        // WiFi.config(DiaChi_ThietBi, default_gateway, subnet_mask);
        WiFi.begin(Ssid, Password);
        countReconnectWifi = 0;
        int timeout = 0;
        DBG("Connecting to wifi: ");
        DBGT(Ssid);
        delay(1000);
        if (WiFi.status() == WL_CONNECTED)
        {
            DBG("Connected to wifi : ");
            DBGT(Ssid);
            DBG("My IP : ");
        }
    }
}
```

```
        DBGT(WiFi.localIP());
        WiFi.config(DiaChi_ThietBi, default_gateway, subnet_mask, staticDNS);
        connectFirst = true;
    }
    else
    {
        wifiDisConnect = true;
        DBG("Can not connect to wifi : ");
        DBGT(Ssid);
    }
    server.begin();
    delay(100);
}
else
    WiFi.mode(WIFI_STA);
DBG("Setup DONE\n");
delay(10);
ESP.wdtFeed();
//Nh.Config
IRRemoteSetupForMain();
if (flagDebug == true)
    DBGT(VERSION_STR);
if (!MDNS.begin(myDomain))
{
    DBG("ERR: mDNS thiet lap that bai!");
}
DBG("mDNS thiet lap thanh cong ");
DBGT(myDomain);
IRdevice.isWakeUp = true;
}

void loop()
{
    kiem_tra_he_thong();
    ESP.wdtFeed();
    dieu_khien_nut_nhan();
    ESP.wdtFeed();
    checkCmd(); //for test by serial cmd
    ESP.wdtFeed();
    //dieu_khien_Led();
    ESP.wdtFeed();
    nhan_du_lieu_tu_HomeServer();
    ESP.wdtFeed();
    tien_hanh_dieu_khien_IR();
    //##DETECT_REMOTE_SERIAL## IR_Detect_Serial(flag_IR_Detect_Serial); //For serial testing
    ESP.wdtFeed();
}
```

```
void ICACHE_RAM_ATTR buttonInterrupt()
{
    if (digitalRead(Button) == PRESS)
    {
        bt1.pressed = PRESS;
        bt1.timestamp = millis();
        bt1.change = true;
    }
    else
    {
        bt1.pressed = RELEASE;
        bt1.timestamp = millis();
        bt1.change = true;
    }
}

void dieu_khien_nut_nhan()
{
    if (IRdevice.state == UNINSTALL)
    {
        if (bt1.change)
        {
            if (bt1.pressed == PRESS)
            {
                bt1.change = false;
            }
            else
            {
                bt1.change = false;
                unsigned long time_t = bt1.timestamp - bt1.timestamp;
                if (time_t < TIME_TO_LEARN_WF)
                {
                    bt1.led = true;
                }
                else
                {
                    IRdevice.preState = IRdevice.state;
                    IRdevice.state = LEARN_WF;
                    timeStartLearnWF = millis();
                    DBG("LEARN WIFI ...\\n");
                    tien_hanh_ket_noi_HomeServer(AP_SSID, AP_PW);
                }
            }
        }
    }
    else if (IRdevice.state == LEARN_WF)
    {
        if (millis() - timeStartLearnWF > TIME_WAIT_LEARN_WF)
```

```
{  
    //Nothing to do  
}  
else  
{  
    if (bt1.change)  
    {  
        if (bt1.pressed == PRESS)  
        {  
            bt1.change = false;  
        }  
        else  
        {  
            bt1.change = false;  
            IRdevice.state = IRdevice.preState;  
            timeCounterLed = 0;  
            // WiFi.softAPdisconnect();  
            WiFi.disconnect();  
            DBG("Break change mode to previous mode\n");  
        }  
    }  
}  
}  
else if (IRdevice.state == RST_FAC)  
{  
    DBG("OK RESET FACTORY\n");  
    ResetFactory();  
}  
else if (IRdevice.state == INACTIVE_PAIRING)  
{  
    if (bt1.change)  
    {  
        if (bt1.pressed == PRESS)  
        {  
            bt1.change = false;  
        }  
        else  
        {  
            bt1.change = false;  
            unsigned long time_t = bt1.timestop - bt1.timestamp;  
            if (time_t < TIME_TO_RST_FAC)  
            {  
                Udp.begin(UDP_PORT); // khai tao udp  
                DBG("Start UDP service\n");  
                IRdevice.preState = IRdevice.state;  
                IRdevice.state = ACTIVE_PAIRING;  
                timeStartActivePairing = millis();  
                DBG("Mode : ACTIVE_PAIRING\n");  
            }  
        }  
    }  
}
```

```
        }
    }
}
else if (IRdevice.state == ACTIVE_PAIRING)
{
    if (millis() - timeStartActivePairing > TIME_WAIT_ACTIVE_PAIRING)
    {
    }
    else
    {
        if (bt1.change)
        {
            if (bt1.pressed == PRESS)
            {
                bt1.change = false;
            }
            else
            {
                bt1.change = false;
                IRdevice.state = IRdevice.preState;
                timeCounterLed = 0;
                DBG("Break change mode to previous mode\n");

                unsigned Long time_t = bt1.timestop - bt1.timestart;
                if (time_t > TIME_TO_RST_FAC)
                {
                    DBG("WORKING: Wait for RST Factory ...\\n");
                    timeStartRSTFac = millis();
                    bt1.counter = 0;
                    IRdevice.preState = IRdevice.state;
                    IRdevice.state = RST_FAC;
                }
            }
        }
    }
}
else if (IRdevice.state == WORKING)
{
    if (bt1.change)
    {
        if (bt1.pressed == PRESS)
        {
            bt1.change = false;
        }
        else
        {
            bt1.change = false;
```

```
        unsigned Long time_t = bt1.timestop - bt1.timestamp;
        if (time_t < TIME_TO_RST_FAC)
        {
            // if (time_t < TIME_TO_UNWORKING) {
            bt1.led = false; // Bat den
            // } else if (time_t < TIME_TO_RST_FAC){
            // IRdevice.state = INACTIVE_PAIRING;
            // DBG("Break change mode to INACTIVE_PAIRING mode");
            }
        else
        {
            DBG("WORKING: Wait for RST Factory ...\\n");
            timeStartRSTFac = millis();
            bt1.counter = 0;
            IRdevice.preState = IRdevice.state;
            IRdevice.state = RST_FAC;
        }
    }
}

//for test
void checkCmd(void)
{
    if (Serial.available())
    {
        memset(sbuf, '\0', MAX_BUF_REC_SERIAL);
        Serial.readBytes(sbuf, MAX_BUF_REC_SERIAL);
        String cmdbk = "";
        cmdbk = sbuf;
        DBG("Received cmd: ");
        DBGT(cmdbk);
        String cmdbk_req = splitStringJson("req", cmdbk);

        if (cmdbk == "endbg")
        {
            flagDebug = true;
            DBG("Start debug\\n");
            EEPROM.write(ADD_DBG, 1);
            EEPROM.commit();
            delay(10);
        }
        else if (cmdbk == "disdbg")
        {
            flagDebug = false;
            DBG("Stop debug\\n");
            EEPROM.write(ADD_DBG, 0);
        }
    }
}
```

```

        EEPROM.commit();
        delay(10);
    }
    else if (cmdbk == "myip")
    {
        DBG("\tIP store : ");
        DBGT(DiaChi_ThietBi);
        DBG("\tIP now : ");
        DBGT(WiFi.localIP());
    }
    else if (cmdbk == "rship")
    {
        DBG("\tHomeServer_IP : ");
        DBGT(HomeServer_IP);
    }
    else if (cmdbk == "restart")
    {
        restartSys();
    }
    else if (cmdbk == "update")
    {
        flagUpdateFw = true;
        EEPROM.write(RES_VERS, NEWLY_UPDATE); //Bao Version Khi co lenh. Bao
        sau khi restart
        EEPROM.commit();
        res_version_to_server = NEWLY_UPDATE;
        accessUpdate("", "");
    }
    else if (cmdbk == "readrom")
    {
        ReadEeprom();
    }
    else if (cmdbk == "forcewifi")
    {
        String ssid_ser = "HLSoft 2";
        String password_ser = "mothaiba";
        String ipState_ser = "STATIC";
        String ipAdd_ser = "192.168.1.120";
        String network_prefix_len_ser = "24";
        String gateway_ser = "192.168.1.1";
        //Nh.Config=====
=====

        /*----- */ String dns_ser = "8.8.8.8";
        Ssid = (char *)malloc((ssid_ser.length() * sizeof(char)));
        Password = (char *)malloc((password_ser.length() * sizeof(char)));
        ssid_ser.toCharArray(Ssid, ssid_ser.length() + 1);
        password_ser.toCharArray(Password, password_ser.length() + 1);
        String subnet_mask_t;
    }
}

```

```

        if (ipState_ser == "STATIC")
        {
            if (network_prefix_len_ser == "8")
            {
                subnet_mask_t = "255.0.0.0";
            }
            else if (network_prefix_len_ser == "16")
            {
                subnet_mask_t = "255.255.0.0";
            }
            else if (network_prefix_len_ser == "24")
            {
                subnet_mask_t = "255.255.255.0";
            }
            storeIP(ipAdd_ser, gateway_ser, subnet_mask_t, ipState_ser);
        }
        Mode_Station = true;
        EEPROM.write(ADD_MODE, 1);
        EEPROM.write(ADD_LEN_SSID, strlen(Ssid));
        for (int i = 0; i < strlen(Ssid); i++)
        {
            int charNumber = Ssid[i];
            EEPROM.write(ADD_SSID_F + i, charNumber);
            delay(1);
        }
        EEPROM.write(ADD_LEN_PW, strlen(Password));
        for (int i = 0; i < strlen(Password); i++)
        {
            int charNumber = Password[i];
            EEPROM.write(ADD_PW_F + i, charNumber);
            delay(1);
        }
        // IPAddress HomeServer_IPRemote = clientConnect.remoteIP();
        IPAddress HomeServer_IPRemote = (192, 168, 1, 19);
        String rshIpStr = String(HomeServer_IPRemote[0]) + String(HomeServer_IPRemote[1]) + String(HomeServer_IPRemote[2]) + String(HomeServer_IPRemote[3]);
        storeIP_tmp(rshIpStr, ADD_RSH_IP_F);
        ReadEeprom();
        DBG("Close server\n");
        server.close();
        WiFi.softAPdisconnect();
        DBG("disconnect Wifi\n");
        WiFi.disconnect();
        DBG("Setup wifi and server\n");
        // WiFi.config(DiaChi_ThietBi, default_gateway, subnet_mask, staticDNS);
    } //-- nh --

```

```

//Nh.Config=====
=====
// WiFi.config(DiaChi_ThietBi, default_gateway, subnet_mas
k, dns_);
// WiFi.config(DiaChi_ThietBi, default_gateway, subnet_mask);
//-----

-----
WiFi.mode(WIFI_STA);
WiFi.begin(Ssid, Password);

IRdevice.state = INACTIVE_PAIRING;
EEPROM.write(ADD_STATE1, INACTIVE_PAIRING);

DBG("device.state = INACTIVE_PAIRING\n");
EEPROM.commit();
delay(5);
DBG("Configure DONE\n");
restartSys();
}

// else if (cmdbk == "onfan")
//{
//     String temp = "{\"req\":\"SEND\", \"remote\":\"TV\", \"code
type\": \"RAW\", \"data\": \" 626 1852 628 1830 1890 608 1870 610 630 1824 650 183
0 1892 586 1892 588 646 1832 646 1832 1892 608 1870 586 1894 584 648 1854 1870 58
6 650 1850 628 1852 650 1806 1892 588 1892 586 1890 588 648 1830 1892 608 626 216
66 646 1830 648 1830 1892 586 1892 586 650 1828 648 1832 1894 584 1892 588 648 18
30 672 1806 1892 588 1890 586 1892 588 646 1830 1894 586 648 1830 648 1830 670 18
08 1890 590 1886 590 1890 616 644 1830 1866 614 646 \"\"};"
//     readCommandJSON(temp);
//     for (int i = 0; i < 2; i++)
//     {
//         digitalWrite(Led, LED_ST_ON); //On off bi nguoc
//         delay(100);
//         digitalWrite(Led, LED_ST_OFF);
//         delay(100);
//     }
// }
// else if (cmdbk == "offfan")
//{
//     String temp = "{\"req\":\"SEND\", \"remote\":\"TV\", \"code
type\": \"RAW\", \"data\": \" 648 1828 648 1830 1892 590 1888 588 646 1830 648 183
0 1894 586 1892 588 1892 588 1890 586 648 1830 1892 586 648 1830 648 1830 1890 58
6 648 1834 1890 584 1892 588 646 1832 1892 586 672 1832 648 1808 1886 614 646 \"\"}";
//     readCommandJSON(temp);
// }
else if (cmdbk == "tvpow")
{
}

```

```

    {
        String temp = "{\"req\":\"SEND\", \"remote\":\"TV\", \"codeType\": \"RAW\", \"data\": \" 9060 4450 610 524 584 524 610 526 580 550 584 524 610 524 582 554 580 526 632 1632 584 1676 566 1656 636 1632 584 1660 580 1660 634 570 512 1658 610 1656 582 526 610 522 584 1658 634 1632 584 526 610 1656 586 524 608 522 610 1630 610 1660 584 524 608 524 584 1658 610 522 584 1656 610 41008 9058 2 208 588 \"}";  

        // String temp = "{\"req\":\"SEND\", \"remote\":\"TV\", \"codeType\": \"HEX\", \"manufacturer\":\"NEC\", \"data\": \"12EE50AF \", \"repeat\": \"\", \"bits\": \"32\"}";  

        readCommandJSON(temp);  

    }  

    // else if (cmdbk == "lg2")  

    // {  

    //     String temp = "{\"req\":\"SEND\", \"remote\":\"AC\", \"device\":\"Device-1\", \"manufacturer\":\"LG2\", \"power\":\"P-On\", \"swing\":\"S-Manual\", \"mode\":\"M-Cool\", \"fan_speed\":\"F-Low\", \"temp\":\"T-21\" }";  

    //     readCommandJSON(temp);  

    // }  

    // else if (cmdbk == "acpow")  

    // {  

    //     String temp = "{\"req\":\"SEND\", \"remote\":\"AC\", \"codeType\": \"RAW\", \"data\": \" 8912 4136 492 1550 544 566 478 498 544 532 502 1592 476 540 502 524 518 542 500 558 478 564 476 556 478 1598 478 566 478 556 476 1580 498 1602 474 562 472 522 520 544 498 1558 520 496 546 1574 500 568 476 1538 546 1580 496 564 450 1590 522 558 476 \"}";  

    //     readCommandJSON(temp);  

    // }  

    // else if (cmdbk == "gcpow")  

    // {  

    //     sendGlobalCache();  

    // }  

    else if (cmdbk == "wifistatus")  

    {  

        Serial.print("\nIP address: ");  

        Serial.println(WiFi.localIP());  

        Serial.printf("Gateway IP: %s\n", WiFi.gatewayIP().toString().c_str());  

;  

        Serial.print("\nSubnet mask: ");  

        Serial.println(WiFi.subnetMask());  

        Serial.println("\nDNS 1:");  

        WiFi.dnsIP().printTo(Serial);  

        Serial.println("\nDNS 2:");  

        WiFi.dnsIP(1).printTo(Serial);  

    }  

    else if (cmdbk == "working")

```

```

{
    EEPROM.write(ADD_STATE1, WORKING);
    IRdevice.state = WORKING;
    DBG("State WORKING\n");
    EEPROM.commit();
    Udp.stop();
    //          restartSys();
}
else if (cmdbk == "pairing")
{
    //  EEPROM.write(ADD_STATE1, ACTIVE_PAIRING);
    IRdevice.state = ACTIVE_PAIRING;
    DBG("State PAIRING\n");
    //          restartSys();
}
else if (cmdbk == "rstfac")
{
    DBG("\nResetFactory Done!\n");
    ResetFactory();
}
else if (cmdbk_req == "SERVER")
{
    String tempo = splitStringJson("ip", cmdbk);
    DiaChi_HomeServer[3] = tempo.toInt();
}
else if (cmdbk_req == "SEND")
{
    String tempo = readCommandJSON(cmdbk);
    DBGT(tempo);
    DBG("\n");
}
else if (cmdbk_req == "UPDATE_FW")
{
    String cmdbk_version = splitStringJson("version", cmdbk);
    flagUpdateFw = true;
    accessUpdate(UPDATE_HOST, cmdbk_version);
}
else if (cmdbk_req == "VERSION")
{
    DBG("\n");
    String res = "{\"res\":{\"VERSION\":\\" + VERSION_STR + "}";
    DBGT(res);
}
##DETECT_REMOTE_SERIAL##else if (cmdbk_req == "DETECT_REMOTE")
##DETECT_REMOTE_SERIAL##{
##DETECT_REMOTE_SERIAL##    DBG("\n");
##DETECT_REMOTE_SERIAL##    flag_IR_Detect_Serial = true;

```

```
//##DETECT_REMOTE_SERIAL##    Enable_IR_receive();
//##DETECT_REMOTE_SERIAL##}
else
{
    DBG("NOT FOUND\n");
}
}

void dieu_khien_Led(void)
{
    if (IRdevice.state == RST_FAC || ((millis() -
bt1.timestart > TIME_TO_RST_FAC + TIME_ADJUST_FOR_CHANGE_STATE) && (bt1.timestop
< bt1.timestart)) || wifiDisConnect)
    {
        if (timeCounterLed <= 0)
        {
            digitalWrite(Led, LED_ST_OFF);
            if (millis() - timePreLedOff > TIME_DELAY_LED_ERROR)
            {
                timeCounterLed = 2;
            }
        }
        else
        {
            if (digitalRead(Led) == LED_ST_ON)
            {
                if (millis() - timePreLedOn > TIME_DELAY_LED_ERROR)
                {
                    timeCounterLed--;
                    digitalWrite(Led, LED_ST_OFF);
                    timePreLedOff = millis();
                }
            }
            else
            {
                if (millis() - timePreLedOff > TIME_DELAY_LED_ERROR)
                {
                    digitalWrite(Led, LED_ST_ON);
                    timePreLedOn = millis();
                }
            }
        }
    }
    else if (
        IRdevice.state == ACTIVE_PAIRING || ((millis() -
bt1.timestart > TIME_TO_ACTIVE_PAIRING + TIME_ADJUST_FOR_CHANGE_STATE) && (IRdevice.state == INACTIVE_PAIRING) && (bt1.timestop < bt1.timestart)))
}
```

```
{  
    if (timeCounterLed <= 0)  
    {  
        digitalWrite(Led, LED_ST_OFF);  
        if (millis() - timePreLedOff > TIME_DELAY_LED_SOFF)  
        {  
            timeCounterLed = 3;  
        }  
    }  
    else  
    {  
        if (digitalRead(Led) == LED_ST_ON)  
        {  
            if (millis() - timePreLedOn > TIME_DELAY_LED_SIGNAL_ON)  
            {  
                timeCounterLed--;  
                digitalWrite(Led, LED_ST_OFF);  
  
                timePreLedOff = millis();  
            }  
        }  
        else  
        {  
            if (millis() - timePreLedOff > TIME_DELAY_LED_SIGNAL_OFF)  
            {  
                digitalWrite(Led, LED_ST_ON);  
                timePreLedOn = millis();  
            }  
        }  
    }  
}  
else if (  
    IRdevice.state == LEARN_WF || ((millis() -  
    bt1.timestart > TIME_TO_LEARN_WF + TIME_ADJUST_FOR_CHANGE_STATE) && (bt1.timesto  
p < bt1.timestart) && (IRdevice.state == UNINSTALL)))  
{  
    if (timeCounterLed <= 0)  
    {  
        digitalWrite(Led, LED_ST_OFF);  
        if (millis() - timePreLedOff > TIME_DELAY_LED_SOFF)  
        {  
            timeCounterLed = 2;  
        }  
    }  
    else  
    {  
        if (digitalRead(Led) == LED_ST_ON)  
        {
```

```
        if (millis() - timePreLedOn > TIME_DELAY_LED_SIGNAL_ON)
        {
            timeCounterLed--;
            digitalWrite(Led, LED_ST_OFF);
            timePreLedOff = millis();
        }
    }
    else
    {
        if (millis() - timePreLedOff > TIME_DELAY_LED_SIGNAL_OFF)
        {
            digitalWrite(Led, LED_ST_ON);
            timePreLedOn = millis();
        }
    }
}
else if (IRdevice.state == WORKING)
{
    digitalWrite(Led, LED_ST_OFF);
}
else if (IRdevice.state == UNINSTALL)
{
    digitalWrite(Led, LED_ST_ON);
}

// }

}

//*****READ EEPROM*****
void ReadEeprom()
{
    if (EEPROM.read(ADD_DBG) == 1)
    {
        flagDebug = true;
        DBG("Debug is enabled\n");
    }
    else
    {
        flagDebug = false;
    }
    // flagDebug = true;
    DBG(">>>> This test ");
    DBGT(DEVICE_TYPE);
    DBG(">>>> Version: ");
    DBGT(VERSION_STR);
    DBG(">>>> Series: ");
    DBGT(soSeri);
```

```
DBG("Read Eeprom:\n");
String tempStr = "";
Mode_Station = false;
boolean swrst = false;
if (EEPROM.read(ADD_SW_RST) == 1)
{
    DBG("SWRST nek\n");
    swrst = true;
    EEPROM.write(ADD_SW_RST, 0);
    EEPROM.commit();
    delay(5);
}
if (EEPROM.read(ADD_UPDATE_FW) == 1)
{
    swrst = true;
    EEPROM.write(ADD_UPDATE_FW, 0);
    EEPROM.commit();
    delay(5);
    DBG("Restart by new update installed\n");
}

IRdevice.secretKey = "";
IRdevice.state = EEPROM.read(ADD_STATE1);
for (int i = ADD_KEY1_F; i <= ADD_KEY1_E; i++)
{
    IRdevice.secretKey += String(EEPROM.read(i) - 0x30);
}
DBG("\tSecretkey gang 1 : \t");
DBGT(IRdevice.secretKey);
if (IRdevice.state == WORKING)
{
    tempStr = "WORKING";
}
else if (IRdevice.state == INACTIVE_PAIRING)
{
    tempStr = "INACTIVE_PAIRING";
}
else if (IRdevice.state == UNINSTALL)
{
    tempStr = "UNINSTALL";
}
else
{
    IRdevice.state = UNINSTALL;
    tempStr = "Not matched => change to : UNINSTALL";
}
DBG("\tState gang 1 : \t");
DBGT(tempStr);
```

```
// DOC STATION MODE
if (EEPROM.read(ADD_MODE) == 1)
{
    Mode_Station = true;
    tempStr = "STA";
}
else
{
    Mode_Station = false;
    tempStr = "AP";
}
DBG("\tMode : ");
DBGT(tempStr);
// IP STATIC OR DHCP
if (EEPROM.read(ADD_STATUS) == 1)
{
    isStatic = true;
    tempStr = "STATIC";
}
else
{
    isStatic = false;
    tempStr = "DHCP";
}
DBG("\tIP Status : ");
DBGT(tempStr);
if (Mode_Station && isStatic)
{
    IPAddress ip_(EEPROM.read(ADD_IP_STA_F), EEPROM.read((ADD_IP_STA_F + 1)),
EEPROM.read((ADD_IP_STA_F + 2)), EEPROM.read(ADD_IP_STA_E));
    DiaChi_ThietBi = ip_;
    DBG("\tDevice IP : ");
    DBGT(DiaChi_ThietBi);
    IPAddress gateway(EEPROM.read(ADD_GETWAY_F), EEPROM.read((ADD_GETWAY_F +
1)), EEPROM.read((ADD_GETWAY_F + 2)), EEPROM.read(ADD_GETWAY_E));
    default_gateway = gateway;
    DBG("\tDefault Gateway : ");
    DBGT(default_gateway);
    IPAddress subnet(EEPROM.read(ADD_SUBNETMARK_F), EEPROM.read(ADD_SUBNETMAR
K_F + 1), EEPROM.read(ADD_SUBNETMARK_F + 2), EEPROM.read(ADD_SUBNETMARK_E));
    subnet_mask = subnet;
    DBG("\tSubnet mask : ");
    DBGT(subnet_mask);
//Nh.config=====
=====
```

```

    IPAddress local_dns(EEPROM.read(ADD_DNS_1), EEPROM.read(ADD_DNS_2), EEPROM.read(ADD_DNS_3), EEPROM.read(ADD_DNS_4));
    dns_ = local_dns;
    DBG("\tDNS : ");
    DBGT(dns_);
    //-----
    -----
}

// READ SSID AND PASSWORD
if (
    IRdevice.state == INACTIVE_PAIRING || IRdevice.state == WORKING || IRdevice.state == ACTIVE_PAIRING)
{
    if (EEPROM.read(ADD_LEN_SSID) > 0)
    {
        String Ssid_ = "";
        String Password_ = "";
        for (int i = 0; i < EEPROM.read(ADD_LEN_SSID); i++)
        {
            Ssid_ = Ssid_ + "" + char(EEPROM.read(ADD_SSID_F + i));
        }
        if (EEPROM.read(ADD_LEN_PW) > 0)
        {
            for (int i = 0; i < EEPROM.read(ADD_LEN_PW); i++)
            {
                Password_ = Password_ + "" + char(EEPROM.read(ADD_PW_F + i));
            }
        }
        Ssid = (char *)malloc((Ssid_.length()) * sizeof(char));
        Password = (char *)malloc((Password_.length()) * sizeof(char));
        Ssid_.toCharArray(Ssid, Ssid_.length() + 1);
        Password_.toCharArray(Password, Password_.length() + 1);
        DBG("\tSSID : ");
        DBGT(Ssid);
        DBG("\tPassword : ");
        DBGT(Password);
    }
    else
    {
        IRdevice.state = UNINSTALL;
        DBG("Lose SSID and PW\n");
        DBG("state all = UNINSTALL\n");
    }
    uint8_t rship1 = EEPROM.read(ADD_RSH_IP_F);
    uint8_t rship2 = EEPROM.read((ADD_RSH_IP_F + 1));
    uint8_t rship3 = EEPROM.read((ADD_RSH_IP_F + 2));
    uint8_t rship4 = EEPROM.read(ADD_RSH_IP_E);
}

```

```
IPAddress HomeServer_IP_(rship1, rship2, rship3, rship4);
HomeServer_IP = HomeServer_IP_;
//Nh.Config: bookmark4
DiaChi_HomeServer = HomeServer_IP;
ClientConnectAddr = HomeServer_IP; //temporary: for first time send.
// -----
HomeServer_IPStr = String(rship1) + "." + String(rship2) + "." + String(r
ship3) + "." + String(rship4);
DBG("\tHomeServer_IP : ");
DBGT(HomeServer_IP);
}
//Return version to server
res_version_to_server = EEPROM.read(RES_VERS); //NEWLY_UPDATE: updated ; else
: No update
DBG("[DBG] Response version to server flag:");
DBGT(res_version_to_server);
// *** Get server host name
// SaveHost("dev1.local");
// EEPROM.commit();
serDomain = GetHost();
DBG(serDomain);
}

boolean ResetFactory()
{
    Mode_Station = false;
    ClearEeprom();
    ReadEeprom();
    IRdevice.state = UNINSTALL;
    delay(10);
    server.close();
    WiFi.softAPdisconnect();
    WiFi.disconnect();
    WiFi.mode(WIFI_OFF);
    delay(10);
    restartSys();
    return false;
}

void startupStatusProcess()
{
    if (flagUpdated)
    {
        EEPROM.write(ADD_UPDATE_FW, 0);
        EEPROM.commit();
    }
}
```

```
String gui_du_lieu_den_HomeServer_IP(String data, boolean fb)
{
    client.setTimeout(TIMEOUT_FB_CONNECT);
    //    if (!client.connect(HomeServer_IP, TCP_TLS_PORT_CLIENT))
    if (!client.connect(DiaChi_HomeServer, TCP_TLS_PORT_CLIENT))
    {
        client.stop();
        DBG("HomeServer_IP: ");
        DBGT(HomeServer_IP);
        DBG("Port: ");
        DBGT(TCP_TLS_PORT_CLIENT);
        DBG("Connect TCP TLS to feedback failed\n");
        // Send Who I am to server --> to update IP if fail
        thiet_lap_mDns();
        return "";
    }

    client.print(data);
    client.println("\r");
    String response = "";
    if (fb)
    {
        while (client.connected())
        {
            response = client.readStringUntil('\r');
            break;
        }
    }
    else
    {
        response = "ACK";
    }
    client.stop();
    return response;
}
String gui_du_lieu_den_IP_gan_nhat(String data, boolean fb)
{
    client.setTimeout(TIMEOUT_FB_CONNECT);
    if (!client.connect(ClientConnectAddr, TCP_TLS_PORT_CLIENT))
    {
        client.stop();
        DBG("IP Rashome: ");
        DBGT(HomeServer_IP);
        DBG("Port: ");
        DBGT(TCP_TLS_PORT_CLIENT);
        DBG("Connect TCP TLS to feedback failed\n");
        return "";
    }
```

```
client.print(data);
client.println("\r");
String response = "";
if (fb)
{
    while (client.connected())
    {
        response = client.readStringUntil('\r');
        break;
    }
}
else
{
    response = "ACK";
}
client.stop();
return response;
}
String gui_du_lieu_den_HomeServer_Domain(String data, boolean fb)
{
    client.setTimeout(TIMEOUT_FB_CONNECT);
//    if (!client.connect(HomeServer_IP, TCP_TLS_PORT_CLIENT))
//        if (strstr(serDomain, "local") == NULL)
    if (serDomain.length() < 5)
    {
        DBG("No host");
        return "Fail ser";
    }
    if (!client.connect(serDomain, TCP_TLS_PORT_CLIENT))
    {
        client.stop();
        DBG("Port: ");
        DBGT(TCP_TLS_PORT_CLIENT);
        DBG("Connect TCP TLS to feedback failed\n");
        return "";
    }
    DBG("IP Rashome -- get from Domain: ");
    ClientConnectAddr = client.remoteIP();
    DBGT(ClientConnectAddr);
    if (ClientConnectAddr != DiaChi_HomeServer)
    { //update new IP
        DiaChi_HomeServer = ClientConnectAddr;
        EEPROM.write(ADD_RSH_IP_F, DiaChi_HomeServer[0]);
        EEPROM.write(ADD_RSH_IP_F + 1, DiaChi_HomeServer[1]);
        EEPROM.write(ADD_RSH_IP_F + 2, DiaChi_HomeServer[2]);
        EEPROM.write(ADD_RSH_IP_F + 3, DiaChi_HomeServer[3]);
        EEPROM.commit();
    }
}
```

```
client.print(data);
client.println("\r");
String response = "";
if (fb)
{
    while (client.connected())
    {
        response = client.readStringUntil('\r');
        break;
    }
}
else
{
    response = "ACK";
}
client.stop();
return response;
}

void nhan_du_lieu_tu_HomeServer(void)
{
    if (IRdevice.state == ACTIVE_PAIRING)
    {
        IRStation_Discovery();
    }
    // else
    // {
    //     Udp.stop();
    // }
    if (IRdevice.state == LEARN_WF ||| IRdevice.state == ACTIVE_PAIRING)
    )
    {
        WiFiClient clientConnect = server.available();
        clientConnect.setNoDelay(true);
        if (!clientConnect)
        {
            return;
        }
        // Wait until the client sends some data
        DBG("New client connect: ");
        DBGT(clientConnect.remoteIP());
        unsigned Long timeout = millis() + TIMEOUT_SV_CONNECT;
        while (!clientConnect.available() && millis() < timeout)
        {
            delay(10);
        }
        if (millis() > timeout)
        {
```

```
    DBG("timeout\n");
    clientConnect.flush();
    clientConnect.stop();
    return;
}
GoiTin_HomeServer = clientConnect.readStringUntil('\r');
clientConnect.flush();
DBGT(GoiTin_HomeServer);
if (GoiTin_HomeServer == "")
{
    DBG("Receive nothing!");
}

if (GoiTin_HomeServer != "")
{
    String req = splitStringJson("req", GoiTin_HomeServer); // "DEVICE"
    DBGT(req);
    if (req == "DEVICE")
    {
        GoiTin_ThietBi = "{\"res\":\"DEVICE\", \"type\":\"" + DEVICE_TYPE
+ "\"}\r";
        clientConnect.print(Goitin_ThietBi);
        DBG("GoiTin_ThietBi: ");
        DBGT(Goitin_ThietBi);
    }
    else if (req == "NETWORK")
    {
        String ssid_ser = splitStringJson("ssid", GoiTin_HomeServer);
        String password_ser = splitStringJson("password", GoiTin_HomeServer);
        String ipState_ser = splitStringJson("ip", GoiTin_HomeServer);
        String ipAdd_ser = splitStringJson("ip_static", GoiTin_HomeServer);
        String network_prefix_len_ser = splitStringJson("network_prefix_1
en", GoiTin_HomeServer);
        String gateway_ser = splitStringJson("gateway", GoiTin_HomeServer);
        String dns_ser = splitStringJson("dns", GoiTin_HomeServer);
        clientConnect.print("{\"res\":\"NETWORK\"}");
        Ssid = (char *)malloc((ssid_ser.length()) * sizeof(char));
        Password = (char *)malloc((password_ser.length()) * sizeof(char));
        ssid_ser.toCharArray(Ssid, ssid_ser.length() + 1);
        password_ser.toCharArray(Password, password_ser.length() + 1);
        String subnet_mask_t;
        if (ipState_ser == "STATIC")
        {
            if (network_prefix_len_ser == "8")
```

```
{  
    subnet_mask_t = "255.0.0.0";  
}  
else if (network_prefix_len_ser == "16")  
{  
    subnet_mask_t = "255.255.0.0";  
}  
else if (network_prefix_len_ser == "24")  
{  
    subnet_mask_t = "255.255.255.0";  
}  
storeIP(ipAdd_ser, gateway_ser, subnet_mask_t, ipState_ser);  
}  
Mode_Station = true;  
EEPROM.write(ADD_MODE, 1);  
EEPROM.write(ADD_LEN_SSID, strlen(Ssid));  
for (int i = 0; i < strlen(Ssid); i++)  
{  
    int charNumber = Ssid[i];  
    EEPROM.write(ADD_SSID_F + i, charNumber);  
    delay(1);  
}  
EEPROM.write(ADD_LEN_PW, strlen(Password));  
for (int i = 0; i < strlen(Password); i++)  
{  
    int charNumber = Password[i];  
    EEPROM.write(ADD_PW_F + i, charNumber);  
    delay(1);  
}  
IPAddress HomeServer_IPRemote = clientConnect.remoteIP();  
String rshIpStr = String(HomeServer_IPRemote[0]) + String(HomeServer_IPRemote[1]) + String(HomeServer_IPRemote[2]) + String(HomeServer_IPRemote[3]);  
storeIP_tmp(rshIpStr, ADD_RSH_IP_F);  
ReadEeprom();  
DBG("Close server\n");  
server.close();  
WiFi.softAPdisconnect();  
DBG("disconnect Wifi\n");  
WiFi.disconnect();  
DBG("Setup wifi and server\n");  
  
WiFi.mode(WIFI_STA);  
WiFi.begin(Ssid, Password);  
  
IRdevice.state = INACTIVE_PAIRING;  
EEPROM.write(ADD_STATE1, INACTIVE_PAIRING);
```

```
    DBG("device.state = INATIVE PAIRING\n");
    EEPROM.commit();
    delay(5);
    DBG("Configure DONE\n");
    restartSys();
    return;
}
}
}
/***** ---PROCESS--- *****/
else if (IRdevice.state == WORKING)
{
    WiFiClient clientConnect = server.available();
    clientConnect.setNoDelay(true);
    if (!clientConnect)
    {
        return;
    }

    DBG("New client connect tls: ");
    ClientConnectAddr = clientConnect.remoteIP();
    DBGT(ClientConnectAddr);
    boolean _flag_KeepAlive = false; // true: HI, false: timeout and BYE.
    do
    {
        unsigned Long timeout = millis() + TIMEOUT_SV_CONNECT;
        while (!clientConnect.available() && millis() < timeout)
        {
            delay(10);
        }
        if (millis() > timeout)
        {
            DBG("timeout\n");
            static uint8_t counter_timeout = 0;
            counter_timeout++;
            if (counter_timeout > 3)
            {
                restartSys();
            }
            clientConnect.flush();
            clientConnect.stop();
            for (int i = 0; i < 2; i++)
            {
                digitalWrite(Led, LED_ST_ON); //On off bi nguoc
                delay(500);
                digitalWrite(Led, LED_ST_OFF);
                delay(500);
            }
        }
    }
}
```

```

        }
        return;
    }
    // Read the first line of the request
    GoiTin_HomeServer = clientConnect.readStringUntil('\r');
    DBGT(Go iTin_HomeServer);
    // if (ClientConnectAddr != DiaChi_HomeServer)
    // {
    //     GoiTin_HomeServer = "";
    //     DBG("Wrong server!");
    //     return;
    // }
    if (GoiTin_HomeServer != "")
    {
        // String server_key = splitStringJson("secretkey", GoiTin_HomeSe
rver);
        // if (!_flag_KeepAlive) // Tat xac nhan Key khi da Keep Alive
        // {
        //     if (server_key != IRdevice.secretKey)
        //     {
        //         DBG("Wrong Key\n");
        //         DBG("---Server Key ");
        //         DBGT(server_key);
        //         DBG(" --- My Key ");
        //         DBGT(IRdevice.secretKey);
        //         clientConnect.flush();
        //         clientConnect.stop();
        //         return;
        //     }
        // }
        String req = splitStringJson("req", GoiTin_HomeServer);

        if (req == "SEND" || req == "READY")
        {
            clientConnect.println("{\"res\":\"SEND\"\}\r"); // Khong can t
hiet. Server can tat lenh rcv
            ir_remote_ctrl = GoiTin_HomeServer;
            GoiTin_HomeServer = "";
            tien_hanh_dieu_khien_IR();
            if (_flag_KeepAlive)
                continue;
        }
        else if (req == "DETECT_REMOTE")
        {
            clientConnect.println("\r");
            Enable_IR_receive();
            ir_remote_ctrl = GoiTin_HomeServer;
            GoiTin_HomeServer = "";
        }
    }
}

```

```

    }
    else if (req == "HI")
    {
        clientConnect.println("{\"res\":\"HI\"\r");
        _flag_KeepAlive = true;
        continue;
    }
    else if (req == "BYE")
    {
        break;
    }
    else if (req == "UPDATE_FW")
    {
        String version = splitStringJson("version", GoiTin_HomeServer
);
        String host = splitStringJson("host", GoiTin_HomeServer);
        clientConnect.println("{\"res\":\"Rec CMD UPDATE\", \"ver\":\""
" + version + "\", \"host\":\"" + host + "\">\r");
        EEPROM.write(RES_VERS, NEWLY_UPDATE); //Bao Version Khi co le
nh. Bao sau khi restart
        EEPROM.commit();
        res_version_to_server = NEWLY_UPDATE;
        if (version != "")
        {
            flagUpdateFw = true;
            accessUpdate(host, version);
        }
        else
        {
            DBG("\nWrong Version\n");
        }
    }
    //asdf
    else if (req == "whoiam")
    {
        String _client_host_name = splitStringJson("hostname", GoiTin
_HomeServer);
        if (_client_host_name == serDomain)
        {
            if (ClientConnectAddr != DiaChi_HomeServer)
            { //update new IP
                DiaChi_HomeServer = ClientConnectAddr;
                EEPROM.write(ADD_RSH_IP_F, DiaChi_HomeServer[0]);
                EEPROM.write(ADD_RSH_IP_F + 1, DiaChi_HomeServer[1]);
                EEPROM.write(ADD_RSH_IP_F + 2, DiaChi_HomeServer[2]);
                EEPROM.write(ADD_RSH_IP_F + 3, DiaChi_HomeServer[3]);
                EEPROM.commit();
            }
        }
    }
}

```

```

        }
    }
    else if (req == "BACKUP")
    {
        clientConnect.println("{\"res\": \"BACKUP\"\r");
        flagUpdateFw = true;
        accessUpdate("", "");
    }
    else if (req == "VERSION")
    {
        String res = "{\"res\": \"VERSION\", \"version\": \""
        + VERSION
        _STR + "\"}";
        clientConnect.println(res);
    }

    else if (req == "ENDBG")
    {
        clientConnect.println("{\"res\": \"ENDBG\"\r");
        flagDebug = true;
        DBG("Start debug");
        EEPROM.write(ADD_DBG, 1);
        EEPROM.commit();
        delay(5);
    }

    else if (req == "DISDBG")
    {
        clientConnect.println("{\"res\": \"DISDBG\"\r");
        DBG("Stop debug");
        flagDebug = false;
        EEPROM.write(ADD_DBG, 0);
        EEPROM.commit();
        delay(5);
    }
    else if (req == "CHANGE_WIFI")
    {
        String Id = splitStringJson("id", GoiTin_HomeServer);
        String ssid_ser = splitStringJson("ssid", GoiTin_HomeServer);
        String password_ser = splitStringJson("password", GoiTin_Home
Server);
        clientConnect.println("{\"res\": \"CHANGE_WIFI\", \"ssid\": \""
        + ssid_ser + "\", \"pass\": \""
        + password_ser + "\r");
        DBG("Close server\n");
        server.close();
        WiFi.softAPdisconnect();
        DBG("disconnect old Wifi\n");
        DBGT("Old SSID: ");
        DBGT(Ssid_old);
    }
}

```

```
        DBG("\n");
        DBGT("Old Password: ");
        DBGT>Password_old);
        DBG("\n");
        WiFi.disconnect();
        delay(10);
        Ssid = (char *)malloc((ssid_ser.length()) * sizeof(char));
        Password = (char *)malloc((password_ser.length()) * sizeof(ch
ar));
        ssid_ser.toCharArray(Ssid, ssid_ser.length() + 1);
        password_ser.toCharArray(Password, password_ser.length() + 1)
;
        EEPROM.write(ADD_LEN_SSID, strlen(Ssid));
        for (int i = 0; i < strlen(Ssid); i++)
        {
            int charNumber = Ssid[i];
            EEPROM.write(ADD_SSID_F + i, charNumber);
            delay(5);
        }
        EEPROM.write(ADD_LEN_PW, strlen(Password));
        for (int i = 0; i < strlen(Password); i++)
        {
            int charNumber = Password[i];
            EEPROM.write(ADD_PW_F + i, charNumber);
            delay(5);
        }
        EEPROM.commit();
        delay(5);
        ReadEeprom();
        // WiFi.config(DiaChi_ThietBi, default_gateway, subnet_mark);
        // WiFi.mode(WIFI_STA);
        delay(10);
        server.begin();
        DBG("Changed Wifi OK\n");
        delay(10);
        restartSys();
    }
    else
    {
        continue;
    }
    break;
}
} while (true);
clientConnect.flush();
clientConnect.stop();
}
```

}

```
*****FUNCTION SETUP WIFI ACCESS POINT*****/  
void tien_hanh_ket_noi_HomeServer(char *ssid, char *password)  
{  
    // IPAddress ip_(192, 168, 4, 1);  
    // IPAddress gateway(192, 168, 1, 1);  
    // IPAddress subnet(255, 255, 255, 0);  
    // WiFi.config(ip_, gateway, subnet);  
    // WiFi.mode(WIFI_AP);  
    // WiFi.setOutputPower(OUTPUT_POWER_FOR_WF);  
    // WiFi.softAP(ssid, password);  
    // delay(10);  
    // server.begin();  
    // delay(10);  
  
    WiFi.mode(WIFI_STA);  
    WiFi.begin(LEARN_WF_HOST_SSID, LEARN_WF_HOST_PASS);  
    while (WiFi.status() != WL_CONNECTED)  
    {  
        delay(500);  
    }  
    delay(10);  
    server.begin();  
    delay(10);  
}  
  
unsigned long timetest = 0;  
void IRStation_Discovery()  
{  
    char incomingPacket[255]; // buffer for incoming packets  
    int packetSize = Udp.parsePacket();  
    if (packetSize)  
    {  
        if (flagDebug)  
            printf("Received %d bytes from %s, port %d\n", packetSize, Udp.remoteIP().toString().c_str(), Udp.remotePort());  
        int len = Udp.read(incomingPacket, 255);  
        if (len > 0)  
            incomingPacket[len] = 0;  
        DBGT(incomingPacket);  
        Udp.beginPacket(Udp.remoteIP(), UDP_PORT);  
        DiaChi_HomeServer = Udp.remoteIP();  
        char *type_ = (char *)malloc((DEVICE_TYPE.length()) * sizeof(char));  
        DEVICE_TYPE.toCharArray(type_, DEVICE_TYPE.length() + 1);  
        if (strstr(incomingPacket, "IRSTATION_DISCOVERY") != NULL)  
        {  
            String ss = Ssid;  
            DBG("Detected IRSTATION_DISCOVERY\n");  
        }  
    }  
}
```

```

        String secretkey = String(random(1000, 9999));
        String res = "{\"res\":\"\\\"IRSTATION_DISCOVERY\\\",\\\"seriesNum\\\":\\\" + so
Seri + "\",\\\"secretkey\\\":\\\" + secretkey + \"\\\", \\\"ssid\\\":\\\" + ss + \"\\\",\\\"versio
n\\\":\\\" + VERSION_STR + \"\\\",\\\"hostname\\\":\\\" + myDomain + \"\\\"}\\r\\n";
        EEPROM.write(ADD_KEY1_F, secretkey[0]);
        EEPROM.write(ADD_KEY1_F + 1, secretkey[1]);
        EEPROM.write(ADD_KEY1_F + 2, secretkey[2]);
        EEPROM.write(ADD_KEY1_E, secretkey[3]);
        IRdevice.secretKey = secretkey;
        for (int i = 0; i < 5; i++)
        {
            Udp.print(res);
            Udp.endPacket();
            delay(250);
        }
        //Ket thuc UDP-----
        //Chuyen tu trang thai ACTIVE_PAIRING sang trang thai working
        EEPROM.write(ADD_STATE1, WORKING);
        IRdevice.state = WORKING;
        DBG("Change to State WORKING\\n");
        //asdf
        serDomain = splitStringJson("hostname", incomingPacket);
        SaveHost(serDomain);
        EEPROM.commit();
        Udp.stop();
        EEPROM.write(ADD_RSH_IP_F, DiaChi_HomeServer[0]);
        EEPROM.write(ADD_RSH_IP_F + 1, DiaChi_HomeServer[1]);
        EEPROM.write(ADD_RSH_IP_F + 2, DiaChi_HomeServer[2]);
        EEPROM.write(ADD_RSH_IP_F + 3, DiaChi_HomeServer[3]);
        EEPROM.commit();
        delay(5);
        ESP.restart();
    }
}

void ClearEeprom()
{
    EEPROM.begin(512);
    for (int i = 0; i < 512; i++)
    {
        EEPROM.write(i, 0);
        delay(5);
    }
    EEPROM.commit();
    delay(10);
    DBG("Eeprom cleared\\n");
}

```

```
void storeIP(String StrIP, String StrGateway, String StrSubnet, String ipState)
{
    storeIP_tmp(StrIP, ADD_IP_STA_F);
    storeIP_tmp(StrGateway, ADD_GETWAY_F);
    storeIP_tmp(StrSubnet, ADD_SUBNETMARK_F);
    if (ipState == "STATIC")
    {
        EEPROM.write(ADD_STATUS, 1);
    }
    else if (ipState == "DHCP")
    {
        EEPROM.write(ADD_STATUS, 0);
    }
    else
    {
        EEPROM.write(ADD_STATUS, 3);
    }
}

void storeIP_tmp(String ip, int offset)
{
    int arr[4] = {0, 0, 0, 0};
    for (int i = 0; i < 4; i++)
    {
        if (i == 3)
        {
            arr[i] = ip.toInt();
        }
        else
        {
            int mid = (int)(ip.indexOf('.'));
            int tmp = (ip.substring(0, ip.indexOf('.'))).toInt();
            arr[i] = tmp;
            ip = ip.substring(mid + 1);
        }
    }
    EEPROM.write(offset, arr[0]);
    EEPROM.write(offset + 1, arr[1]);
    EEPROM.write(offset + 2, arr[2]);
    EEPROM.write(offset + 3, arr[3]);
    EEPROM.commit();
    delay(5);
}

void accessUpdate(String host, String ver)
{
    //bookmark6
```

```
// if (flagUpdateFw && (IRdevice.status_val == "OFF"))
DBG("\nStart Update\n");
if (flagUpdateFw)
{
    if (ver != VERSION_STR)
    {
        EEPROM.write(ADD_UPDATE_FW, 1);
        EEPROM.commit();
        delay(10);
        if (WiFi.status() == WL_CONNECTED)
        {
            String path = "";
            if (host != "")
                path = "http://" + HomeServer_IPStr + host + "/" + ver + ".bi
n";
            else
                path = "http://www.hlsoft.com.vn:8888/IR_BACKUP.bin";
            DBG("\nHost: ")
            DBGT(path);
            t_httpUpdate_return ret = ESPhttpUpdate.update(path);
            switch (ret)
            {
                case HTTP_UPDATE_FAILED:
                    DBG("HTTP_UPDATE_FAILED Error :");
                    DBGT(ESPhttpUpdate.getLastErrorCode());
                    DBGT(ESPhttpUpdate.getErrorString());
                    DBG("\n");
                    break;

                case HTTP_UPDATE_NO_UPDATES:
                    DBG("HTTP_UPDATE_NO_UPDATES\n");
                    break;

                case HTTP_UPDATE_OK:
                    DBG("HTTP_UPDATE_OK\n");
                    flagUpdateFw = false;
                    break;
            }
        }
    }
    else
    {
        DBG("Can not update. This lasted version\n");
        flagUpdateFw = false;
    }
}
```

```

String splitStringJson(String strSignal, String scr)
{
    String result = "";
    if (scr.indexOf("{") >= 0 && scr.indexOf("}") > 1 && scr.indexOf(strSignal) >
0)
    {
        String json = scr.substring(scr.indexOf("{"), scr.indexOf("}") + 1);
        String temp = scr.substring(scr.indexOf(strSignal) + strSignal.length() +
2, scr.indexOf("}") + 1);
        // DBG("temp: "); DBGT(temp);
        String temp2 = temp.substring(temp.indexOf("\\"") + 1, temp.indexOf("}") +
1);
        // DBG("temp2: "); DBGT(temp2);
        result = temp2.substring(0, temp2.indexOf("\\""));
        // DBG("result: "); DBGT(result);
    }
    return result;
}

//#include <avr/pgmspace.h>
void printStr(PGM_P s)
{ //use printStr(PSTR("NHAT"));
    char c;
    while ((c = pgm_read_byte(s++)) != 0)
        Serial.print(c);
}
void restartSys()
{
    EEPROM.write(ADD_SW_RST, 1);
    EEPROM.commit();
    delay(10);
    ESP.restart();
}
void kiem_tra_he_thong(void)
{
    if (millis() - timeStartCheckConnectWifi > TIME_TO_CHECK_CONNECT_WIFI)
    {
        timeStartCheckConnectWifi = millis();
        if (IRdevice.state == WORKING || IRdevice.state == INACTIVE_PAIRING || IR
device.state == ACTIVE_PAIRING)
        {
            if (WiFi.status() != WL_CONNECTED)
            {
                wifiDisConnect = true;
                if (millis() - timeStartReConnect > TIME_TO_RECONNECT_WIFI)
                {
                    countReconnectWifi++;
                }
            }
        }
    }
}

```

```
        if (countReconnectWifi > 3) //10
        {
            restartSys();
        }

        timeStartReConnect = millis();
        // WiFi.disconnect(true);
        WiFi.mode(WIFI_STA);
        // WiFi.config(DiaChi_ThietBi, default_gateway, subnet_mask);
        WiFi.begin(Ssid, Password);
        delay(100);
        int timeout = 0;
        DBG("Reconnect to wifi :");
        DBGT(Ssid);
    }
}

else if (wifiDisConnect)
{
    countReconnectWifi = 0;
    wifiDisConnect = false;
    if (IRdevice.state == WORKING || IRdevice.state == INACTIVE_PAIRING
        || IRdevice.state == ACTIVE_PAIRING)
    {
        server.begin();
    }
    else
    {
        server.begin();
    }
    connectFirst = true;
}
else // Working properly
{
    /*_____MDNS_____*/
    MDNS.update();
    if (IRdevice.state == WORKING)
    {
        resVerAfterUpdate();
        isIPChanged();
        if (IRdevice.isWakeUp)
        {
            IRdevice.isWakeUp = false;
            thiet_lap_mDns();
        }
    }
}
}
```

```

if (connectFirst)
{
    // WiFi.config(DiaChi_ThietBi, default_gateway, subnet_mask, staticDNS);
    DBG("Connected to wifi : ");
    DBGT(Ssid);
    DBG("My IP : ");
    DBGT(WiFi.localIP());
    uint8_t updt = EEPROM.read(ADD_UPDATE_FW);
    if (updt == 1 || flagUpdateFw)
    {
        DBG("Update di\n");
    }
    connectFirst = false;
}
}

void tien_hanh_dieu_khien_IR()
{
    String msg_res = "";
    if (ir_remote_ctrl != "")
        msg_res = readCommandJSON(ir_remote_ctrl);

    if (msg_res != "")
    {
        String req_local = splitStringJson("req", ir_remote_ctrl);
        ir_remote_ctrl = "";
        if (req_local == "DETECT_REMOTE")
        {
            for (int i = 0; i < 2; i++)
            {
                digitalWrite(Led, LED_ST_ON); //On off bi nguoc
                delay(100);
                digitalWrite(Led, LED_ST_OFF);
                delay(100);
            }
            // if (gui_du_lieu_den_HomeServer_IP(msg_res, false) == "ACK")
            if (gui_du_lieu_den_IP_gan_nhat(msg_res, false) == "ACK")
            {
                DBG("\nSend Done\n");
            }
            else
            {
                DBG("\nSend Fail\n");
                for (int i = 0; i < 4; i++)
                {
                    digitalWrite(Led, LED_ST_ON); //On off bi nguoc
                    delay(200);
                }
            }
        }
    }
}

```

```

        digitalWrite(Led, LED_ST_OFF);
        delay(300);
    }
}
DBG("\nIR response: \n");
DBGT(msg_res);
DBG("Listening Done!\n");
Disable_IR_receive();
}
else
{
    DBG("\nIR response: \n");
    DBGT(msg_res);
}
}
}

void resVerAfterUpdate()
{
    if (res_version_to_server == NEWLY_UPDATE) // sau khi co lenh up date se bao
version
    {
        String msg = "{\"res\":\"VERSION\", \"version\":\"" + VERSION_STR + "\"}";
;
        DBG("Response update to server")
        if (gui_du_lieu_den_HomeServer_IP(msg, false) == "ACK")
        {
            DBG("Send to server Success")
            res_version_to_server = 0;
            EEPROM.write(RES_VERS, 0);
            EEPROM.commit();
            DBG("\n\n\nThis is version:\n");
            DBGT(VERSION_STR);
            return;
        }
        else
        {
            //             res_version_to_server = 1;
            counterStopFeedback++;
            if (counterStopFeedback > COUNTER_STOP_FEEDBACK) // Ngung Feedback
            {
                res_version_to_server = 0;
                EEPROM.write(RES_VERS, res_version_to_server);
                EEPROM.commit();
            }
            DBGT(counterStopFeedback);
            return;
        }
    }
}

```

```
        }
    }
    void thiet_lap_mDns()
{
    //IPChanged
    DiaChi_ThietBi = WiFi.localIP();
    storeIP_tmp(DiaChi_ThietBi.toString(), ADD_IP_STA_F);
    if (IRdevice.state == WORKING)
    {
        DBG("Wake up: Send Who I Am to server\n");
        String msg = "{\"req\":\"whoiam\",\"hostname\":\"" + myDomain + ".local\"}";
    };
    if (gui_du_lieu_den_HomeServer_Domain(msg, false) == "ACK")
    {
        DBG("Send Done: Server is alive\n");
    }
    else
    {
        DBG("Send Fail: Server is dead\n");
        res_version_to_server = 0;
    }
}
void SaveHost(String host)
{
    EEPROM.write(ADD_HOST_LENGTH, host.length());
    for (int i = 0; i < host.length(); ++i)
    {
        EEPROM.write(ADD_HOST + i, host[i]);
        DBG("Wrote HOST: ");
        DBGT(host[i]);
    }
    // EEPROM.commit();
}
String GetHost()
{
    DBG("Reading EEPROM to get Host\n");
    String _host;
    int _host_length = 0;
    _host_length = EEPROM.read(ADD_HOST_LENGTH);
    for (uint16_t i = ADD_HOST; i < (ADD_HOST + _host_length); ++i)
    {
        _host += char(EEPROM.read(i));
    }
    //_host.trim();
    DBG("Host Length: ");
    DBGT(_host_length);
    DBGT(_host.length());
```

```
DBG("\nHost name: ");
DBG(_host);
return _host;
}
void isIPChanged()
{
    if (millis() - timeToCheckIPChange > 5000)
    {
        timeToCheckIPChange = millis();
        if (DiaChi_ThietBi != WiFi.localIP())
        {
            thiet_lap_mDns();
        }
    }
}
```