

TRƯỜNG ĐẠI HỌC XÂY DỰNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC
NGÀNH CÔNG NGHỆ THÔNG TIN

**XÂY DỰNG HỆ THỐNG NHÀ THÔNG
MINH TRONG GIA ĐÌNH**

Sinh viên thực hiện: **Nguyễn Trọng Thắng**
Mã sinh viên: 1545760
Lớp: **60PM1**
Giảng viên hướng dẫn: KS. Đoàn Như Tùng

HÀ NỘI 03/2020

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên: Nguyễn Trọng Thắng

Điện thoại liên lạc: 0332269266 Email: nsngochang63@gmail.com

Lớp: 60PM1 Hệ đào tạo: Đại học Năm tốt nghiệp: 2020

Đồ án tốt nghiệp được thực hiện tại: Đại học Xây Dựng

Thời gian làm ĐATN: Từ ngày 11/11/2019 đến 07/03/2020

2. Mục đích nội dung của ĐATN

- Xây dựng hệ thống nhà thông minh trong một gia đình.

3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu các yêu cầu của một hệ thống nhà thông minh, đưa ra các tính năng cần thiết.
- Phân tích yêu cầu.
- Thực hiện yêu cầu.
- Cài đặt.

4. Lời cam đoan của sinh viên:

Tôi, Nguyễn Trọng Thắng cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của KS. Đoàn Như Tùng.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày tháng năm
Tác giả ĐATN

Nguyễn Trọng Thắng

5. Xác nhận của giảng viên hướng dẫn về mức độ hoàn thành ĐATN và cho phép bảo vệ:

Hà Nội, ngày tháng năm
Giảng viên hướng dẫn

KS. Đoàn Như Tùng

TÓM TẮT NỘI DUNG ĐÒ ÁN TỐT NGHIỆP

Hệ thống nhà thông minh trong gia đình là một hệ thống giúp các thiết bị điện tử gia dụng đều được liên kết với thiết bị điều khiển trung tâm và có thể phối hợp với nhau để cùng thực hiện một chức năng. Các thiết bị này có thể tự đưa ra cách xử lý tình huống được lập trình trước, hoặc là được điều khiển và giám sát từ xa. Nó giúp điều phối các hoạt động trong ngôi nhà một cách thông minh và gần gũi với người dùng hơn.

Trong dự án này ngoài các sản phẩm thiết kế độc lập em còn tạo ra các thiết bị điện tử hỗ trợ như *Hộp điều khiển thiết bị điện*, *Hộp điều khiển hồng ngoại*, giúp biến thiết bị thông thường thành thiết bị điều khiển thông minh, tận dụng những thiết bị sẵn có trong ngôi nhà, do đó dễ dàng lắp đặt, tăng sự tiện dụng, đơn giản và dễ sửa chữa cùng với giá cả phải chăng.

Những sản phẩm này có khả năng thương mại hóa, sử dụng rộng rãi trong cộng đồng do sự tiện lợi và giá thành rẻ mà nó đem lại.

Việc xây dựng hệ thống gồm 3 phần:

1. Thiết lập bộ điều khiển trung tâm nhà thông minh
2. Thiết lập các thiết bị điện tử sử dụng trong gia đình, bao gồm:
 - Camera quan sát
 - Hộp điều khiển thiết bị điện
 - Hộp điều khiển hồng ngoại
 - Cảm biến nhiệt độ, độ ẩm
 - Cảm biến chất lượng không khí
 - Cảm biến cửa ra vào
 - Chuông báo động
3. Cấu hình hệ thống để có thể nhận thông báo, truy cập, kết nối thiết bị và điều khiển từ bên ngoài Internet gồm:
 - DuckDNS cùng với Let's Encrypt
 - Google Assistant
 - Thông báo với HTML5
 - PiVPN

LỜI MỞ ĐẦU

Để hoàn thành đồ án tốt nghiệp, lời đầu tiên em xin chân thành cảm ơn đến toàn thể thầy cô trong trường Đại Học Xây Dựng, đặc biệt hơn là các thầy cô trong ngành Công Nghệ Thông Tin, bộ môn Công Nghệ Phần Mềm, những người đã tận tình hướng dẫn dạy dỗ và trang bị cho em những kiến thức bổ ích trong những năm được học tập tại trường.

Đặc biệt em xin chân thành cảm ơn thầy Đoàn Như Tùng đã giúp đỡ em trong quá trình học tập, làm đồ án tốt nghiệp. Nhất là trong quá trình thực hiện, thầy đã hướng dẫn tận tình để em hoàn thành tốt đồ án “Xây dựng hệ thống nhà thông minh trong gia đình” này.

Trong quá trình làm đồ án do thiếu kinh nghiệm và thời gian có hạn nên không tránh khỏi có sự sai sót, vì vậy em mong thầy cô thông cảm chỉ bảo để em có thể rút kinh nghiệm giúp hoàn thiện kỹ năng bản thân hơn.

Em xin chân thành cảm ơn thầy cô.

Hà nội, ngày 12 tháng 03 năm 2020

Sinh viên thực hiện đồ án

Nguyễn Trọng Thắng

MỤC LỤC

CHƯƠNG I. ĐẶT VẤN ĐỀ VÀ ĐỊNH HƯỚNG GIẢI PHÁP.....	1
1. Bài toán	1
2. Mục tiêu đề tài	2
3. Định hướng giải quyết vấn đề.....	2
4. Giới thiệu về phần cứng.....	3
4.1. Raspberry Pi 4 Model B	3
4.2. NodeMcu ESP8266 CP2102	4
4.3. Module Relay 1 Kênh 5V10A.....	5
4.4. Raspberry Pi NoIR Camera V2	6
4.5. LED hồng ngoại cho Camera NoIR Raspberry Pi	7
4.6. Bóng đèn LED	7
4.7. Cảm biến DHT11	8
4.8. Cảm biến MQ135	9
4.9. Cảm biến cửa MC-38	10
4.10. Chuông điện	10
4.11. Mắt Thu Hồng Ngoại HS0038B.....	11
4.12. Điện trở 30 Ohm.....	12
4.13. LED hồng ngoại	13
4.14. Nguồn, dây cáp.....	14
4.15. Thẻ nhớ.....	16
5. Giới thiệu về cơ sở lý thuyết và công cụ sử dụng.....	17
5.1. Secure Shell	17
5.2. MQTT	17
5.3. Infrared Remote Control	19
5.4. Raspbian	22
5.5. MotionEyeOS	23
5.6. Home Assistant.....	23
5.7. Docker	24

5.8. PiVPN	24
5.9. Google Home và Google Assistant	26
5.10. DuckDNS và Let's Encrypt.....	27
5.11. Google Authenticator	28
CHƯƠNG II. XÂY DỰNG HỆ THỐNG	29
1. Sơ đồ hệ thống	29
1.1. Sơ đồ hệ thống phần cứng	29
1.2. Sơ đồ hệ thống phần mềm	30
2. Xây dựng bộ xử lý trung tâm.....	32
2.1. Cài đặt hệ điều hành Raspbian	32
2.2. Cài đặt Docker	35
2.3. Cài đặt Home Assistant	36
2.4. Cài đặt MQTT - Mosquitto	37
2.5. Cấu hình IP tĩnh.....	38
3. Camera	39
3.1. Cấu tạo phần cứng	39
3.2. Cài đặt hệ điều hành và cấu hình IP tĩnh.....	41
3.3. Cài đặt và cấu hình Motioneye.....	41
3.4. Thiết lập điều khiển LED hồng ngoại	43
3.5. Thiết lập Camera trên Home Assistant	46
4. Hộp điều khiển thiết bị điện.....	51
4.1. Cấu tạo phần cứng	52
4.2. Thiết lập điều khiển hộp điều khiển thiết bị điện	54
4.3. Thiết lập hộp điều khiển thiết bị điện trên Home Assistant	57
5. Hộp điều khiển hồng ngoại	59
5.1. Cấu tạo phần cứng thiết bị thu hồng ngoại.....	60
5.2. Thiết lập sử dụng thiết bị thu hồng ngoại.....	61
5.3. Cấu tạo phần cứng hộp điều khiển hồng ngoại	68
5.4. Thiết lập điều khiển hộp điều khiển hồng ngoại	69
5.5. Thiết lập hộp điều khiển hồng ngoại trên Home Assistant	78

6. Cảm biến nhiệt độ, độ ẩm và chất lượng không khí	97
6.1. Cấu tạo phần cứng	97
6.2. Thiết lập điều khiển cảm biến	99
6.3. Thiết lập cảm biến trên Home Assistant	103
7. Cảm biến cửa	107
7.1. Cấu tạo phần cứng	107
7.2. Thiết lập điều khiển cảm biến cửa.....	109
7.3. Thiết lập cảm biến cửa trên Home Assistant.....	112
8. Chuông báo động	114
8.1. Cấu tạo phần cứng	114
8.2. Thiết lập điều khiển chuông báo động	116
8.3. Thiết lập chuông báo động trên Home Assistant	122
9. Tự động hóa các thiết bị	128
9.1. Tự động bật đèn hồng ngoại của Camera.....	128
9.2. Bật chuông khi cửa mở.....	128
9.3. Tự động bật còi báo cháy và thông báo khi có hỏa hoạn	129
9.4. Bật đèn khi cửa mở.....	130
9.5. Tưới cây tự động	131
9.6. Bật máy lọc không khí tự động	133
9.7. Đóng, mở cửa sổ tự động	135
9.8. Bật tắt điều hòa tự động	136
9.9. Cảnh báo khi cửa mở	139
9.10. Cảnh báo khi cửa mở quá 5 phút.....	140
9.11. Cảnh báo khi chất lượng không khí xấu.....	141
9.12. Cảnh báo khi trời sắp mưa.....	142
10. Phân quyền và bảo mật tài khoản	143
10.1. Phân quyền tài khoản truy cập	143
10.2. Bảo mật tài khoản.....	145
CHƯƠNG III: CẤU HÌNH HỆ THỐNG CHO PHÉP TRUY CẬP TỪ BÊN NGOÀI INTERNET	147

1. Cấu hình Hass để truy cập từ bên ngoài Internet.....	147
2. Điều khiển bằng giọng nói với Google Assistant.....	150
3. Nhận thông báo thiết bị từ xa bằng HTML5	152
4. Kết nối thiết bị bên ngoài Internet vào Hass với PiVPN.....	154
CHƯƠNG IV: CÁC KẾT QUẢ ĐẠT ĐƯỢC.....	156
1. Kết quả thực hiện	156
2.1. Ưu điểm	156
2.2 Nhược điểm	157
3. Khả năng sử dụng, so sánh các sản phẩm cùng loại.....	157
4. Kết luận.....	158
PHỤ LỤC.....	158
TÀI LIỆU THAM KHẢO.....	160

Danh mục bảng

Bảng 1: Mã IR của điều khiển TV..... 67

Danh mục hình ảnh

Hình 1: Raspberry Pi 4 Model B	3
Hình 2: NodeMcu ESP8266 CP2102	4
Hình 3: Module Relay 1 Kênh 5V10A	5
Hình 4: Raspberry Pi NoIR Camera V2.....	6
Hình 5: LED hồng ngoại cho Camera NoIR Raspberry Pi	7
Hình 6: Bóng đèn LED	7
Hình 7: Cảm biến DHT11	8
Hình 8: Cảm biến MQ135.....	9
Hình 9: Cảm biến cửa MC-38.....	10
Hình 10: Chuông điện	10
Hình 11: Mắt Thu Hồng Ngoại HS0038B	11
Hình 12: Điện trở 30 Ohm	12
Hình 13: LED hồng ngoại	13
Hình 14: Nguồn Raspberry Pi 4	14
Hình 15: Nguồn NodeMCU	15
Hình 16: Dây USB A-Micro	15
Hình 17: Thẻ nhớ SanDisk.....	16
Hình 18: Mô hình giao thức MQTT.....	18
Hình 19: Logo Eclipse Mosquitto	19
Hình 20: Biểu đồ tần số và bước sóng của các bức xạ.....	19
Hình 21: Xử lý tín hiệu hồng ngoại	20
Hình 22: Giao thức NEC	21
Hình 23: Logo của MotionEyeOS.....	23
Hình 24: Logo của Home Assistant	23
Hình 25: Logo của Docker	24
Hình 26: Logo của PiVPN	24
Hình 27: Cách hoạt động của VPN	25
Hình 28: Logo của Google Home	26
Hình 29: Logo của Google Assistant	26
Hình 30: Logo của DuckDNS	27
Hình 31: Logo của Let's Ecrypt.....	27
Hình 32: Logo của Google Authenticator.....	28
Hình 33: Sơ đồ hệ thống phần cứng.....	29
Hình 34: Sơ đồ hệ thống phần mềm.....	30
Hình 35: Bộ xử lý trung tâm	32
Hình 36: Phần mềm Win32DiskImager	32

Hình 37: Phần mềm PuTTY	33
Hình 38: Terminal PuTTY	34
Hình 39: Màn hình Raspbian	34
Hình 40: Màn hình tạo tài khoản Hass	36
Hình 41: Camera được chế tạo và lắp đặt hoàn thiện	39
Hình 42: Sơ đồ đấu nối Camera	40
Hình 43: Đầu nối Camera thực tế	40
Hình 44: Phần mềm Geany	43
Hình 45: Giao thức MQTT của Đèn camera	43
Hình 46: Bảng điều khiển Camera	50
Hình 47: Hộp điều khiển thiết bị điện được chế tạo và lắp hoàn thiện	51
Hình 48: Sơ đồ đấu nối hộp điều khiển thiết bị điện	52
Hình 49: Đầu nối hộp điều khiển thiết bị điện thực tế	53
Hình 50: Đầu nối hộp điều khiển thiết bị điện với thiết bị thông thường	53
Hình 51: Sản phẩm đèn cửa thông minh hoàn thiện nhìn từ phía ngoài	53
Hình 52: Arduino IDE	54
Hình 53: Giao thức MQTT của hộp điều khiển thiết bị điện	54
Hình 54: Công tắc điều khiển đèn cửa	58
Hình 55: Hộp điều khiển hồng ngoại chế tạo và lắp hoàn thiện	59
Hình 56: Sơ đồ đấu nối thiết bị thu hồng ngoại	60
Hình 57: Đầu nối thiết bị thu hồng ngoại thực tế	60
Hình 58: Lấy mã IR của điều khiển	66
Hình 59: Sơ đồ đấu nối hộp điều khiển hồng ngoại	68
Hình 60: Đầu nối hộp điều khiển hồng ngoại thực tế	68
Hình 61: Giao thức MQTT của hộp điều khiển hồng ngoại	69
Hình 62: Bảng điều khiển TV	96
Hình 63: Cảm biến không khí được chế tạo và lắp hoàn thiện	97
Hình 64: Sơ đồ đấu nối cảm biến không khí	98
Hình 65: Đầu nối cảm biến không khí thực tế	98
Hình 66: Giao thức MQTT của cảm biến không khí	99
Hình 67: Bảng nhiệt độ, độ ẩm và chất lượng không khí	106
Hình 68: Cảm biến cửa được chế tạo và lắp đặt hoàn thiện	107
Hình 69: Sơ đồ đấu nối cảm biến cửa	108
Hình 70: Đầu nối cảm biến cửa thực tế	108
Hình 71: Giao thức MQTT của cảm biến cửa	109
Hình 72: Bảng cảm biến cửa	113
Hình 73: Chuông báo động được chế tạo và lắp hoàn thiện	114
Hình 74: Sơ đồ đấu nối chuông báo động	115

Hình 75: Đáu nối chuông báo động thực tế	115
Hình 76: Giao thức MQTT của chuông báo động	117
Hình 77: Bảng cảm biến cửa và chuông báo động	127
Hình 78: Bảng điều khiển kịch bản còi báo cháy	130
Hình 79: Bảng điều khiển kịch bản đèn khi mở cửa.....	131
Hình 80: Bảng điều khiển kịch bản tưới cây tự động	132
Hình 81: Bảng điều khiển kịch bản máy lọc không khí.....	134
Hình 82: Bảng điều khiển kịch bản đóng mở cửa sổ	136
Hình 83: Bảng điều khiển kịch bản bật tắt điều hòa	138
Hình 84: Thông báo khi cửa mở	139
Hình 85: Thông báo khi cửa mở quá 7 phút	140
Hình 86: Thông báo khi chất lượng không khí xấu	141
Hình 87: Thông báo khi trời mưa.....	142
Hình 88: Giao diện Home Assistant của tài khoản được phân quyền	144
Hình 89: Khóa tự động được tạo ra.....	145
Hình 90: Mã xác thực được tạo ra.....	146
Hình 91: Yêu cầu xác thực hai lớp khi đăng nhập.....	146
Hình 92: Website DuckDNS	147
Hình 93: Port Forwarding	149
Hình 94: Sử dụng Google Assistant và Google Home	151
Hình 95: Ví dụ về thông báo HTML5.....	153
Hình 96: Kết nối VPN trên điện thoại.....	155
Hình 97: Giao diện trong nhà.....	158
Hình 98: Giao diện điều khiển TV	159
Hình 99: Giao diện điều khiển tự động hóa	159

Danh mục các từ viết tắt và thuật ngữ:

IOT	Internet of Things
MQTT	MQ Telemetry Transport
SSH	Secure Shell
HASS	Home Assistant
LAN	Local Area Network
VPN	Virtual Private Network
DNS	Domain Name System
IP	Internet Protocol
IDE	Integrated Development Environment
HTTPS	Hypertext Transfer Protocol Secure
HTML5	Hypertext Markup Language Revision 5
QR	Quick Response
ID	Identifier
IR	Infrared
TV	Television

CHƯƠNG I. ĐẶT VẤN ĐỀ VÀ ĐỊNH HƯỚNG GIẢI PHÁP

1. Bài toán

Ngày nay, các cụm từ Internet vạn vật kết nối", "Nhà thông minh" hay "Smart Home" thường xuyên được nhắc đến như là một xu hướng tiên tiến, hướng con người đến với cuộc sống tiện nghi và thoái mái do công nghệ mang lại. Đó không còn là những thứ trong tương lai xa mà đã hiện diện ở khắp nơi trên thế giới bao gồm cả ở Việt Nam.

Tuy nhiên Smart Home cho đến nay hầu như chỉ là những sản phẩm xa xỉ và đắt đỏ chỉ dành cho các gia đình giàu có, ngay như hãng Smart Home rẻ nhất hiện tại cũng phải mất vài chục đến vài trăm triệu. Vậy nên bài toán được đặt ra là làm cách nào triển khai một nhà thông minh, không còn dành riêng cho các gia đình giàu có mà nó sẽ là một sản phẩm mang tiện ích thiết thực cho mọi gia đình trong xã hội. Giải pháp của em ngoài tạo ra các thiết bị thông minh giá rẻ mà quan trọng còn tận dụng các thiết bị sẵn có, giúp biến những món đồ điện tử bình thường trong ngôi nhà trở nên thông minh bằng cách gắn thêm vào nó một "Hộp điều khiển thiết bị điện", đây là giải pháp đơn giản, giá rẻ, không loại bỏ các thiết bị cũ, chỉ cần mất khoảng 5 triệu cho sự khởi đầu, sau này có thể bỏ thêm chi phí để phát triển thêm.

Đây còn là giải pháp đem đến sự riêng tư và gần gũi với người dùng, hệ thống sẽ vận hành trong mạng nội bộ của gia đình và các thiết bị thông minh được kiểm soát thông qua các thiết bị truyền thông như điện thoại di động, máy tính bảng,... ở bất cứ nơi đâu. Với giải pháp này, nó đạt được các tiêu chí : thông minh, giá rẻ, tiện dụng, đơn giản, an toàn và dễ sửa chữa.

2. Mục tiêu đề tài

Mục tiêu đề tài nhằm tạo ra các thiết bị điện tử thông minh, tiện dụng, giá cả phải chăng, dễ lắp đặt cùng với một mạng điều khiển liên kết chúng trong ngôi nhà và chúng được điều khiển thông qua điện thoại hay máy tính bảng ở bất cứ đâu đồng thời giúp chúng tự động hóa hoàn toàn hoặc bán tự động, thay thế con người trong việc thực hiện một hoặc một số thao tác quản lý, điều khiển giúp giảm bớt đi sự bất tiện khi phải điều khiển bằng tay, đem lại sự tiện nghi, thoải mái hơn cùng với sự an toàn và an ninh của ngôi nhà.

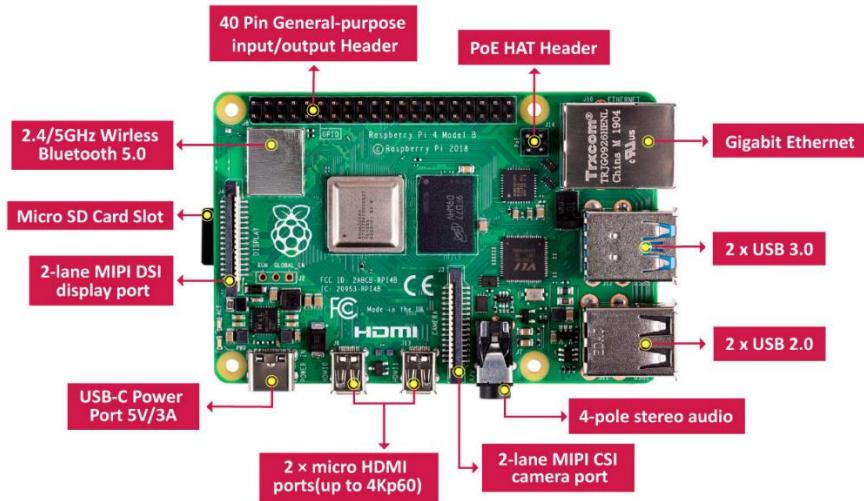
3. Định hướng giải quyết vấn đề

Qua quá trình nghiên cứu, em thấy có nhiều cách để triển khai một hệ thống nhà thông minh, mỗi cách đều có ưu và nhược điểm riêng. Ở đây em lựa chọn cho mình phương pháp tiếp cận từ các giải pháp mã nguồn mở, các thiết bị phần cứng và các ngôn ngữ lập trình có tính mở cao.

4. Giới thiệu về phần cứng

4.1. Raspberry Pi 4 Model B

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. Raspberry hiện đang là Mini Computer nổi tiếng bậc nhất hiện nay.



Hình 1: Raspberry Pi 4 Model B

Thông số kỹ thuật:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- Có 3 lựa chọn RAM: 1GB, 2GB hoặc 4GB LPDDR4-2400 SDRAM
- Wifi chuẩn 2.4 GHz và 5.0 GHz IEEE 802.11ac. Bluetooth 5.0, BLE
- Cổng mạng Gigabit Ethernet
- 2 cổng USB 3.0 và 2 cổng USB 2.0
- Chuẩn 40 chân GPIO, tương thích với các phiên bản trước
- Hỗ trợ 2 cổng ra màn hình chuẩn Micro HDMI với độ phân giải lên tới 4K
- Cổng MIPI DSI
- Cổng MIPI CSI
- Cổng AV 4 chân
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Khe cắm Micro-SD cho hệ điều hành và lưu trữ
- Nguồn điện DC 5V – 3A DC chuẩn USB-C
- 5V DC via GPIO header (minimum 3A*)
- Hỗ trợ Power over Ethernet (PoE) (yêu cầu có PoE HAT)
- Nhiệt độ hoạt động: 0 – 50 độ C

4.2. NodeMcu ESP8266 CP2102

Kít ESP8266 là kít phát triển dựa trên nền chip Wifi SoC ESP8266 với thiết kế dễ dàng sử dụng vì tích hợp sẵn mạch nạp sử dụng chip CP2102 trên borad. Bên trong ESP8266 có sẵn một lõi vi xử lý có thể trực tiếp lập trình cho ESP8266 mà không cần thêm bất kì con vi xử lý nào nữa. Hiện tại có hai ngôn ngữ có thể lập trình cho ESP8266, sử dụng trực tiếp phần mềm IDE của Arduino để lập trình với bộ thư viện riêng hoặc sử dụng phần mềm node MCU.



Hình 2: NodeMcu ESP8266 CP2102

Thông số kỹ thuật

- WiFi: 2.4 GHz hỗ trợ chuẩn 802.11 b/g/n
- Điện áp hoạt động: 3.3V
- Điện áp vào: 5V thông qua cổng USB
- 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

4.3. Module Relay 1 Kênh 5V10A

Relay là một công tắc chuyển đổi, dùng để đóng cắt mạch điều khiển, được hoạt động bằng điện. Relay là một công tắc vì có 2 trạng thái ON và OFF. Relay ở trạng thái ON hay OFF phụ thuộc vào có dòng điện chạy qua nó hay không.



Hình 3: Module Relay 1 Kênh 5V10A

Thông số kỹ thuật

- Điện áp tải tối đa: AC 250V-10A / DC 30V-10A
- Điện áp điều khiển: 5 VDC
- Dòng kích Relay: 5mA
- Trạng thái kích: Mức thấp (0V)
- Đường kính lỗ ốc: 3.1 mm
- Kích thước: 50 * 26 * 18.5 mm

4.4. Raspberry Pi NoIR Camera V2

Raspberry Pi NoIR Camera V2 kết nối với bất kỳ Raspberry Pi hoặc Single Board Compute Module nào qua cổng CSI, cho phép tạo ra các video HD và các bức ảnh tĩnh. Pi NoIR (Không hồng ngoại) giống như mô-đun máy ảnh tiêu chuẩn nhưng không có bộ lọc IR. Vì vậy nó là rất tốt cho nhiếp ảnh và video trong bóng tối. Bạn có thể quan sát được tất cả trong vật trong bóng tối với Pi NoIR Camera.



Hình 4: Raspberry Pi NoIR Camera V2

Thông số kỹ thuật:

- Ống kính tiêu cự cố định
- Cảm biến độ phân giải 8 megapixel cho khả năng chụp ảnh kích thước 3280 x 2464
- Hỗ trợ video 1080p30, 720p60 và 640x480p90
- Kích thước 25mm x 23mm x 9mm
- Trọng lượng chỉ hơn 3g
- Kết nối với Raspberry Pi thông qua cáp ribbon đi kèm dài 15 cm
- Camera Module được hỗ trợ với phiên bản mới nhất của Raspbian

4.5. LED hồng ngoại cho Camera NoIR Raspberry Pi

LED hồng ngoại cung cấp ánh sáng hồng ngoại cho Camera NoIR, cho phép Camera có thể nhìn thấy được vào ban đêm.



Hình 5: LED hồng ngoại cho Camera NoIR Raspberry Pi

Thông số kỹ thuật

- Công suất: 3 W
- Loại LED: IR LED
- Hoạt động tốt trong phạm vi của 1-2 m

4.6. Bóng đèn LED



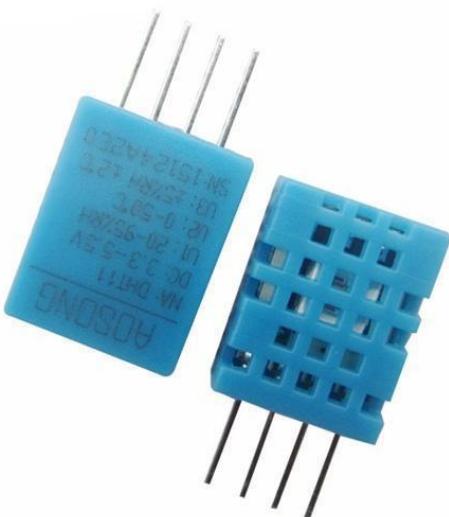
Hình 6: Bóng đèn LED

Thông số kỹ thuật

- Nguồn điện sử dụng: 220V
- Công xuất: 20W

4.7. Cảm biến DHT11

Cảm biến độ ẩm và nhiệt độ DHT11 Temperature Humidity Sensor là cảm biến rất thông dụng hiện nay vì chi phí rẻ và lấy dữ liệu thông qua giao tiếp 1 wire (giao tiếp digital 1 dây truyền dữ liệu duy nhất). Bộ tiền xử lý tín hiệu tích hợp trong cảm biến giúp bạn có được dữ liệu chính xác mà không phải qua bất kỳ tính toán nào.



Hình 7: Cảm biến DHT11

Thông số kỹ thuật:

- Nguồn: 3 -> 5 VDC.
- Dòng sử dụng: 2.5mA max (khi truyền dữ liệu).
- Đo tốt ở độ ẩm 2080%RH với sai số 5%.
- Đo tốt ở nhiệt độ 0 to 50°C sai số $\pm 2^\circ\text{C}$.
- Tần số lấy mẫu tối đa 1Hz (1 giây 1 lần)
- Kích thước 15mm x 12mm x 5.5mm.
- 4 chân, khoảng cách chân 0.1".

4.8. Cảm biến MQ135

Thường được dùng trong các thiết bị kiểm tra chất lượng không khí bên trong cao ốc, văn phòng, thích hợp để phát hiện NH₃, NOx, Ancol, Benzen, khói, CO₂,...



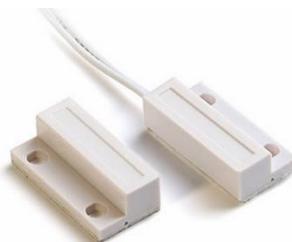
Hình 8: Cảm biến MQ135

Thông số kỹ thuật:

- Điện áp nguồn: 5V DC
- Điện áp của heater: 5V±0.1 AC/DC
- Điện trở tải: thay đổi được (2kΩ-47kΩ)
- Điện trở của heater: 33Ω±5%
- Công suất tiêu thụ của heater: ít hơn 800mW
- Khoảng phát hiện: 10 - 300 ppm NH₃, 10 - 1000 ppm Benzene, 10 - 300 Alcol
- Kích thước: 32mm*20mm

4.9. Cảm biến cửa MC-38

Cảm biến từ MC-38 là công tắc từ có dây, gắn trên cửa, tủ. Cảm biến sẽ đóng khi 2 miếng đặt gần nhau, thường được dùng trong ứng dụng chống trộm ở cửa, tủ, két sắt,...



Hình 9: Cảm biến cửa MC-38

Thông số kỹ thuật

- Kích thước: 27x 14 x 10mm
- Khoảng cách hoạt động: 18mm ± 6mm
- Điện áp giữa 2 tiếp điểm: tối đa 100 VDC
- Dòng tiêu thụ: 300mA (Max)
- Dạng ngõ ra: thường mở
- Tuổi thọ: 100 triệu lần

4.10. Chuông điện



Hình 10: Chuông điện

Điện áp: 220V AC

Công dụng: dùng làm chuông báo động, báo cháy,...

4.11. Mắt Thu Hồng Ngoại HS0038B

HS0038B là thiết bị thu loại nhỏ dùng cho hệ thống điều khiển từ xa bằng hồng ngoại. Khi nhận được tín hiệu hồng ngoại nó có thể trực tiếp giải mã tín hiệu bằng một bộ vi xử lý bên trong nó. Thiết bị này tương thích với tất cả các định dạng tín hiệu điều khiển từ xa IR phổ biến hiện nay.



Hình 11: Mắt Thu Hồng Ngoại HS0038B

Đặc trưng

- Điện áp : 2.5V đến 5.5V
- Có khả năng chống lại ánh sáng xung quanh
- Không nhạy cảm với việc cung cấp điện áp gợn và nhiễu
- Tần số sóng mang : 38 kHz

4.12. Điện trở 30 Ohm

Điện trở là linh kiện được dùng nhiều nhất trong các mạch điện tử. Công dụng chính của nó là hạn chế hoặc điều chỉnh dòng điện và phân chia điện áp trong mạch điện. Dùng để hạn chế dòng điện, phân áp, phân cực transistor, tham gia vào các mạch lọc, mạch tạo dao động, mạch khuếch đại, ...



Hình 12: Điện trở 30 Ohm

Thông số kỹ thuật

- Trị số điện trở: 30 Ohm
- Công suất định mức: 1/4W
- Sai số: 5%

4.13. LED hồng ngoại

LED dùng để phát tín hiệu hồng ngoại được sử dụng trong điều khiển , remote hồng ngoại. Ứng dụng trong mạch mô phỏng remote, mạch chống trộm bằng hồng ngoại, mạch dò đường, ...



Hình 13: LED hồng ngoại

Thông số kỹ thuật

- Điện Áp: 1.2 - 1.6V DC
- Dòng: 10 - 20mA
- Bước Sóng: 940 nm
- Kích Thước: 5 mm
- Ánh sáng phát ra từ LED hồng ngoại không nhìn thấy được

4.14. Nguồn, dây cáp

- Nguồn giúp cung cấp năng lượng cho thiết bị hoạt động.
- Dây cáp giúp kết nối nguồn với thiết bị.



Hình 14: Nguồn Raspberry Pi 4

Nguồn Raspberry Pi 4

- Output:
 - + Điện áp đầu ra : 5.1V
 - + Dòng điện đầu ra : 3.0A
 - + Công suất : 15.3W
 - + Cáp kết nối : 1.5m 18AWG
 - + Chuẩn kết nối : USB Type-C
- Input:
 - + Điện áp : 100-240 Vac(rated); 96-264 Vac(operating)
 - + Tần số : 50/60Hz ±3Hz
 - + Tiêu thụ điện năng : 0.075W maximum



Hình 15: Nguồn NodeMCU

Nguồn NodeMCU

- Điện áp vào : AC 100 - 240V / 50 / 60Hz. / 0.14A.
- Điện áp ra : 5V 1A.
- Cổng đầu ra USB.



Hình 16: Dây USB A-Micro

Dây USB A-Micro chiều dài 30CM

4.15. Thẻ nhớ

Được sử dụng làm bộ nhớ cho Raspberry Pi 4



Hình 17: Thẻ nhớ SanDisk

Thông số kỹ thuật:

- Loại SD: SDHC Class 10
- Thương hiệu: SanDisk
- Dung lượng: 64GB
- Tốc độ ghi: 48 MB/s
- Tốc độ đọc: 15 MB/s
- Màu sắc: Đen

5. Giới thiệu về cơ sở lý thuyết và công cụ sử dụng

5.1. Secure Shell

Secure Shell, hoặc được gọi là SSH, là một giao thức điều khiển từ xa cho phép người dùng kiểm soát và chỉnh sửa server từ xa qua Internet. Dịch vụ được tạo ra nhằm thay thế cho trình Telnet vốn không có mã hóa và sử dụng kỹ thuật cryptographic để đảm bảo tất cả giao tiếp gửi tới và gửi từ server từ xa diễn ra trong tình trạng mã hóa.

Trong đồ án này, em sử dụng phần mềm PuTTY để kết nối SSH với Raspberry Pi, giúp cài đặt và cấu hình thiết bị.

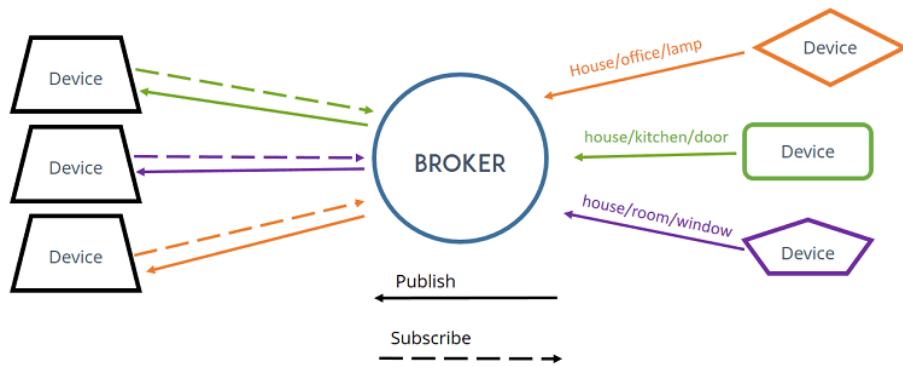
5.2. MQTT

Message Queue Telemetry Transport, hoặc được gọi tắt là MQTT, 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.

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 message 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 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.



Hình 18: Mô hình giao thức MQTT

Ưu điểm của MQTT:

- Chuyển thông tin hiệu quả hơn
- Tăng khả năng mở rộng
- Giảm đáng kể tiêu thụ băng thông mạng
- Giảm tốc độ cập nhật xuống giây
- Rất phù hợp cho điều khiển và đo thám
- Tối đa hóa băng thông có sẵn
- Chi phí cực nhẹ
- Rất an toàn với bảo mật dựa trên sự cho phép
- Được sử dụng bởi ngành công nghiệp dầu khí, Amazon, Facebook và các doanh nghiệp lớn khác
- Tiết kiệm thời gian phát triển
- Giao thức publish/subscribe thu thập nhiều dữ liệu hơn với ít băng thông hơn so với giao thức cũ.

Ứng dụng MQTT trong dự án này:

- Đề án chủ yếu sử dụng giao thức này cho việc truyền dữ liệu từ cảm biến và thiết bị về server và gửi lệnh điều khiển từ server hoặc người dùng đến các thiết bị.
- Để triển khai giao thức này, em sử dụng phần mềm Eclipse Mosquitto

Eclipse Mosquitto:



Hình 19: Logo Eclipse Mosquitto

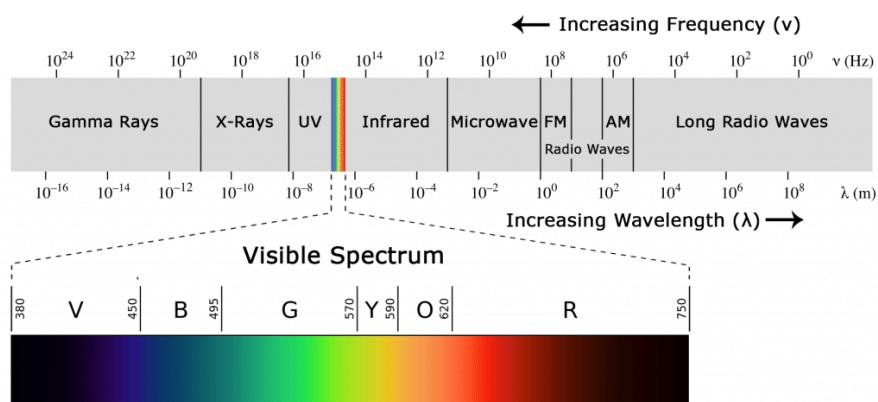
Eclipse Mosquitto là một nhà môi giới thông báo mã nguồn mở (được cấp phép EPL / EDL) sử dụng các giao thức MQTT phiên bản 5.0, 3.1.1 và 3.1. Mosquitto rất nhẹ và phù hợp để sử dụng trên tất cả các thiết bị từ máy tính bảng đơn năng lượng thấp đến máy chủ đầy đủ.

5.3. Infrared Remote Control

Điều khiển thiết bị bằng hồng ngoại là một công nghệ không dây được sử dụng rộng rãi và dễ thực hiện, có nhiều ứng dụng hữu ích. Các ví dụ nổi bật nhất trong cuộc sống hàng ngày là điều khiển từ xa TV, điều hòa,...

Hồng ngoại là gì ?

Bức xạ hồng ngoại là một dạng ánh sáng tương tự như ánh sáng chúng ta nhìn thấy xung quanh. Sự khác biệt duy nhất giữa ánh sáng hồng ngoại và ánh sáng khả kiến là tần số và bước sóng. Bức xạ hồng ngoại nằm ngoài phạm vi ánh sáng khả kiến, vì vậy con người không thể nhìn thấy nó:



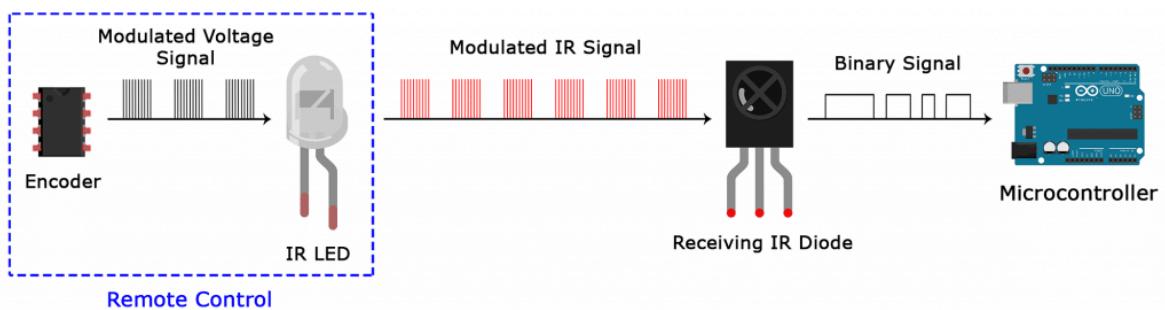
Hình 20: Biểu đồ tần số và bước sóng của các bức xạ

Bởi vì IR là một loại ánh sáng, giao tiếp IR đòi hỏi một đường ngắm trực tiếp từ máy thu đến máy phát. Nó không thể truyền qua tường hoặc các vật liệu khác như WiFi hoặc Bluetooth.

Một hệ thống thông tin hồng ngoại thông thường đòi hỏi một máy phát hồng ngoại và máy thu hồng ngoại. Máy phát trông giống như một đèn LED tiêu chuẩn, ngoại trừ nó tạo ra ánh sáng trong phổ IR thay vì phổ khả kiến. Bộ thu hồng ngoại gồm một đốt quang và tiền khuếch đại chuyển đổi ánh sáng hồng ngoại thành tín hiệu điện.

Cách điều chế tín hiệu hồng ngoại:

Ánh sáng hồng ngoại được phát ra từ mặt trời, bóng đèn và bát cứ thứ gì khác tạo ra nhiệt, điều đó có nghĩa là có rất nhiều ánh sáng hồng ngoại xung quanh chúng ta. Để ngăn các ánh sáng này gây nhiễu tín hiệu IR, kỹ thuật điều chế tín hiệu được sử dụng. Trong điều chế tín hiệu IR, bộ mã hóa trên điều khiển từ xa IR chuyển đổi tín hiệu nhị phân thành tín hiệu điện được điều chế. Tín hiệu điện này được gửi đến đèn LED để truyền. Đèn LED sẽ chuyển đổi tín hiệu điện thành tín hiệu ánh sáng hồng ngoại. Sau đó, bộ thu IR giải điều chế tín hiệu ánh sáng hồng ngoại và chuyển đổi trở lại thành nhị phân trước khi truyền thông tin đến vi điều khiển:



Hình 21: Xử lý tín hiệu hồng ngoại

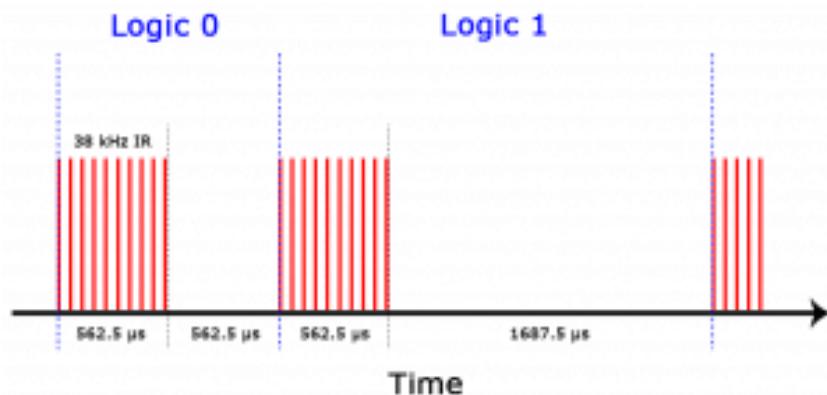
Tín hiệu IR được điều chế là một chuỗi các xung ánh sáng hồng ngoại được bật và tắt ở tần số cao được gọi là tần số sóng mang. Tần số sóng mang được sử dụng bởi hầu hết các máy phát là 38 kHz, vì nó rất hiếm trong tự nhiên và do đó có thể được phân biệt với nhiều xung quanh. Bằng cách này, bộ thu IR sẽ biết rằng tín hiệu 38 kHz được gửi từ máy phát và không được nhận từ môi trường xung quanh. Diode thu phát hiện tất cả các tần số của ánh sáng hồng ngoại, nhưng nó có bộ lọc thông dải và chỉ cho phép đi qua IR ở 38 kHz. Sau đó, nó khuếch đại tín hiệu điều chế bằng bộ

khuếch đại trước và chuyển đổi nó thành tín hiệu nhị phân trước khi gửi nó đến một vi điều khiển.

Giao thức truyền dẫn IR:

Mẫu tín hiệu là tín hiệu IR được điều chế chuyển đổi thành chuỗi nhị phân được xác định bởi một giao thức truyền nào đó. Có nhiều giao thức truyền IR như Sony, Samsung, NEC và RC5 là một số giao thức phổ biến, trong đó giao thức NEC được sử dụng phổ biến nhất, cách NEC chuyển đổi tín hiệu IR thành tín hiệu nhị phân như sau:

- Logic '1' bắt đầu với xung HIGH 562,5 µs có tần số 38 kHz, sau đó là xung LOW dài 1.687,5 µs.
- Logic '0' được truyền với xung HIGH dài 562,5 µs và sau là xung LOW dài 562,5 µs



Hình 22: Giao thức NEC

Đây là cách giao thức NEC mã hóa và giải mã dữ liệu nhị phân thành tín hiệu điều chế. Các giao thức khác chỉ khác nhau về thời lượng của các xung HIGH và LOW.

Mã IR:

Mỗi lần ta nhấn một nút trên điều khiển từ xa, một mã duy nhất sẽ được tạo. Đây là thông tin được điều chế và gửi qua IR đến người nhận. Để giải mã phím nào được nhấn, vi điều khiển nhận tín hiệu cần biết mã nào tương ứng với từng phím trên điều khiển từ xa. Các điều khiển từ xa khác nhau gửi các mã khác nhau cho các phím bấm, vì vậy ta sẽ cần xác định mã được tạo cho mỗi phím trên từng điều khiển từ xa cụ thể.

Để làm được điều đó, ta cần nhấn từng phím trên điều khiển từ xa đến máy thu để nhấn chuỗi mã cho từng phím, sau đó ta sao chép chuỗi mã cho máy phát để gửi đi điều khiển thiết bị sau này.

Trong dự án này, em sử dụng tín hiệu hồng ngoại để tạo ra *Hộp điều khiển hồng ngoại* giúp điều khiển các thiết bị trong nhà có chức năng điều khiển bằng hồng ngoại được.

5.4. Raspbian

Raspbian là một hệ điều hành dựa trên Debian cho Raspberry Pi. Có một số phiên bản của Raspbian bao gồm Raspbian Stretch và Raspbian Jessie. Từ năm 2015, nó đã được Raspberry Pi Foundation chính thức cung cấp như là hệ điều hành chính cho gia đình máy tính bảng đơn Raspberry Pi. Raspbian được tạo ra bởi Mike Thompson và Peter Green như một dự án độc lập. Hệ điều hành vẫn đang được phát triển tích cực. Raspbian được tối ưu hóa cao cho các CPU ARM hiệu suất thấp của dòng Raspberry Pi.

Trong dự án này, em sử dụng phiên bản Raspbian Stretch cho Raspberry Pi.

5.5. MotionEyeOS



Hình 23: Logo của MotionEyeOS

MotionEyeOS là một bản phân phối Linux biến máy tính single-board thành hệ thống giám sát video. Hệ điều hành dựa trên BuildRoot và sử dụng motion cho backend và motionEye cho frontend.

Bản thân MotionEyeOS là một hệ điều hành, nhưng vẫn có thể cài đặt lên hệ điều hành Raspbian trên Pi, đây là cách mà trong dự án này em áp dụng để tạo nên Camera giám sát.

5.6. Home Assistant



Hình 24: Logo của Home Assistant

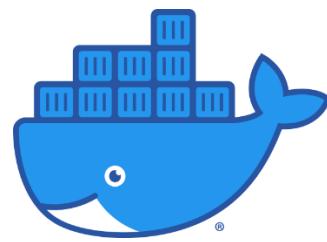
Home Assistant là một nền tảng tự động hóa mã nguồn mở chạy trên Python 3. Nó cho phép theo dõi và kiểm soát tất cả các thiết bị trong nhà và tự động kiểm soát. Home Assistant được thiết kế để dễ dàng triển khai trên bất kỳ máy tính nào từ Raspberry Pi đến các thiết bị lưu trữ trên mạng (NAS) và thậm chí là một container Docker để triển khai trên các hệ thống khác một cách dễ dàng.

Một số đặc điểm của Home Assistant

- Giống như hầu hết các hệ thống tự động, Home Assistant cung cấp bản client trên điện thoại và máy tính để điều khiển các thiết bị nhà thông minh từ xa.
- Home Assistant không có các thành phần điện toán đám mây.
- Vì Home Assistant không hoàn toàn khác biệt so với các framework IoT khác nên nó dễ dàng kết nối với nhiều nền tảng khác nhau từ Nest đến Arduino hay Kodi.
- Có một điểm mạnh của Home Assistant do Python mang tới đó là: Việc mở rộng hệ thống rất dễ dàng.

Trong dự án này, em sử dụng Home Assistant làm bộ điều khiển trung tâm cho nhà thông minh.

5.7. Docker



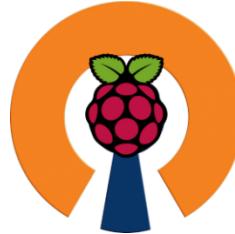
Hình 25: Logo của Docker

Docker là một dự án mã nguồn mở giúp tự động triển khai các ứng dụng Linux và Windows vào trong các container ảo hóa.

Docker cung cấp một lớp trừu tượng và tự động ảo hóa dựa trên Linux. Docker sử dụng những tài nguyên cô lập của Linux như cgroups, kernel, quản lý tệp để cho phép các container chạy độc lập bên trong một thực thể Linux.

Trong dự án này, em sử dụng Docker để cài đặt Home Assistant trên Raspbian

5.8. PiVPN



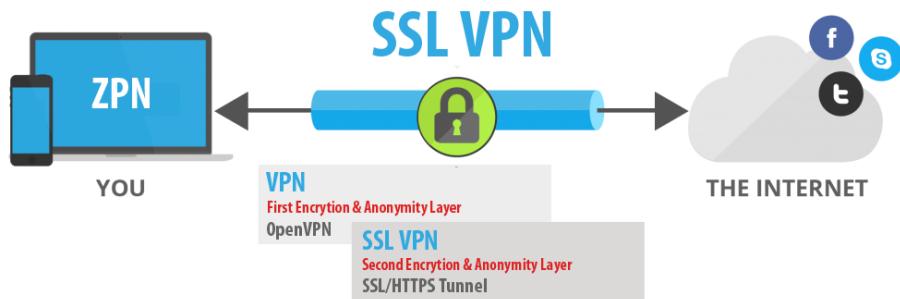
Hình 26: Logo của PiVPN

PiVPN cho phép thiết lập và quản lý VPN, được thiết kế cho Raspberry Pi.

VPN là gì ?

VPN hay còn gọi là Virtual Private Network (mạng riêng ảo), cho phép người dùng thiết lập mạng riêng ảo với một mạng khác trên Internet. VPN có thể được sử dụng để truy cập các trang web bị hạn chế truy cập về mặt vị trí địa lý, bảo vệ hoạt động duyệt web của bạn khỏi “sự tò mò” trên mạng Wifi công cộng bằng cách thiết lập mạng riêng ảo cho bạn.

Cách hoạt động của VPN:



Hình 27: Cách hoạt động của VPN

Khi kết nối máy tính của bạn (hoặc các thiết bị khác như điện thoại thông minh hoặc máy tính bảng) với VPN, máy tính sẽ hoạt động như thể một kết nối cục bộ như VPN. Tất cả lưu lượng mạng sẽ được gửi thông qua một kết nối an toàn đến VPN.

Và bởi vì máy tính của bạn hoạt động trên hệ thống mạng này, điều này cho phép bạn truy cập nguồn tài nguyên mạng cục bộ ngay cả khi bạn đang ở đầu bên kia của thế giới.

Trong dự án này, em sử dụng piVPN để kết nối các thiết bị điện tử bên ngoài Internet vào bộ điều khiển trung tâm nhà thông minh, vốn chỉ hoạt động trong phạm vi mạng LAN của gia đình.

5.9. Google Home và Google Assistant



Hình 28: Logo của Google Home

Google Home là phần mềm quản lý nhà thông minh của Google giúp ta có thể gán những thiết bị thông minh trong gia đình vào phần mềm để điều khiển thuận tiện.

Trong dự án này, em chỉ gán các thiết bị vào Google Home với mục đích để Google Assistant có thể nhận diện thiết bị.

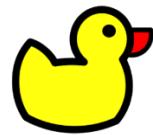


Hình 29: Logo của Google Assistant

Google Assistant là một trợ lý cá nhân ảo được phát triển bởi Google cho thiết bị di động và nhà thông minh, được giới thiệu lần đầu tại hội nghị nhà phát triển của hãng vào tháng 5 năm 2016. Là sự nâng cấp của Google Now, Google Assistant có thể tham gia các cuộc trò chuyện hai chiều.

Trong dự án này, em sử dụng Google Assistant để điều khiển bật tắt các thiết bị trong gia đình.

5.10. DuckDNS và Let's Encrypt



Hình 30: Logo của DuckDNS

Dynamic DNS là gì ?

Đa phần ở Việt Nam các nhà mạng đều cung cấp cho người dùng IP động, điều này dẫn đến sự thay đổi IP sau một khoảng thời gian, khi đó Dynamic DNS, phương thức ánh xạ tên miền tới địa chỉ IP có lần xuất thay đổi cao cung cấp một chương trình đặc biệt chạy trên máy tính của người sử dụng dịch vụ dynamic DNS gọi là Dynamic Dns Client. Chương trình này giám sát sự thay đổi địa chỉ IP tại host và liên hệ với hệ thống DNS mỗi khi địa chỉ IP của host thay đổi và sau đó update thông tin vào cơ sở dữ liệu DNS về sự thay đổi địa chỉ đó. Bằng cách này, cho dù máy chủ có thường xuyên bị thay đổi địa chỉ thì tên miền vẫn được hệ thống máy chủ DNS trả về đúng địa chỉ được cấp IP mới đó.

Trong dự án này, em sử dụng dịch vụ DNS động của DuckDNS, với mục đích có thể truy cập, điều khiển nhà thông minh bên ngoài Internet.



Hình 31: Logo của Let's Encrypt

Let's Encrypt là nhà cung cấp chứng chỉ SSL miễn phí, tự động, hoạt động vì lợi ích của cộng đồng. Nó được quản lý bởi Internet Security Research Group (ISRG).

Let's Encrypt cung cấp cho những người quản trị website một chứng nhận số cần thiết để kích hoạt HTTPS (SSL hoặc TLS) cho website của mình, hoàn toàn miễn phí, và theo cách thân thiện nhất có thể. Mục tiêu để tạo ra một môi trường Web an toàn, riêng tư và tôn trọng người dùng hơn.

5.11. Google Authenticator



Hình 32: Logo của Google Authenticator

Google Authenticator giúp tạo mã xác minh 2 bước trên điện thoại.

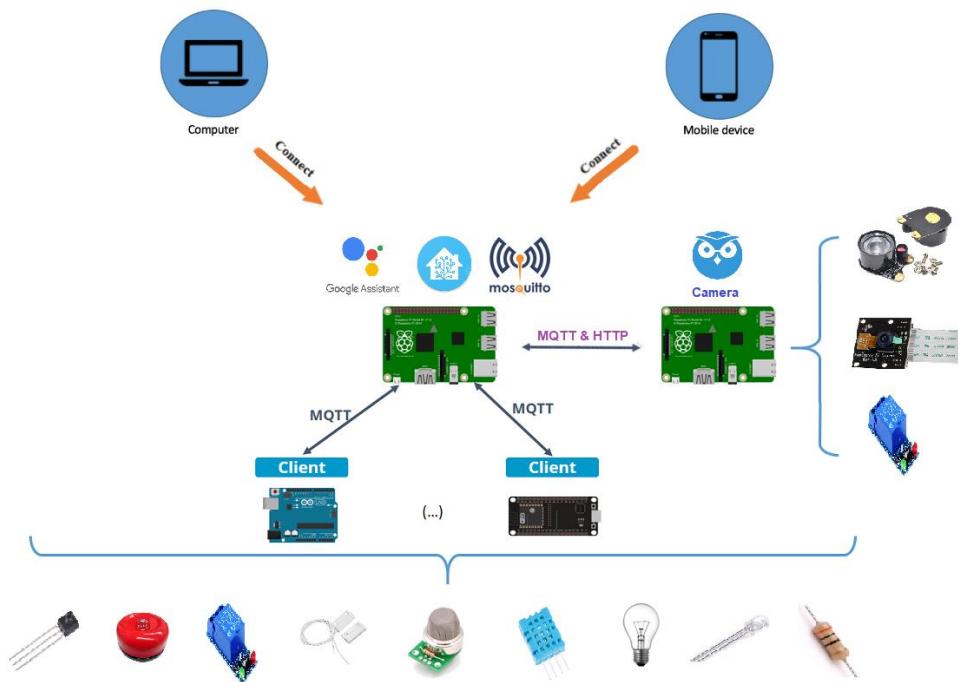
Xác minh 2 bước cung cấp bảo mật tốt hơn cho tài khoản bằng cách yêu cầu bước xác minh thứ hai khi đăng nhập. Ngoài mật khẩu, chúng ta sẽ cần mã được tạo bởi ứng dụng Google Authenticator trên điện thoại.

Trong dự án, em sử dụng Google Authenticator để tạo mã xác minh khi đăng nhập, giúp tăng cường bảo mật tài khoản khi truy cập từ bên ngoài.

CHƯƠNG II. XÂY DỰNG HỆ THỐNG

1. Sơ đồ hệ thống

1.1. Sơ đồ hệ thống phần cứng



Hình 33: Sơ đồ hệ thống phần cứng

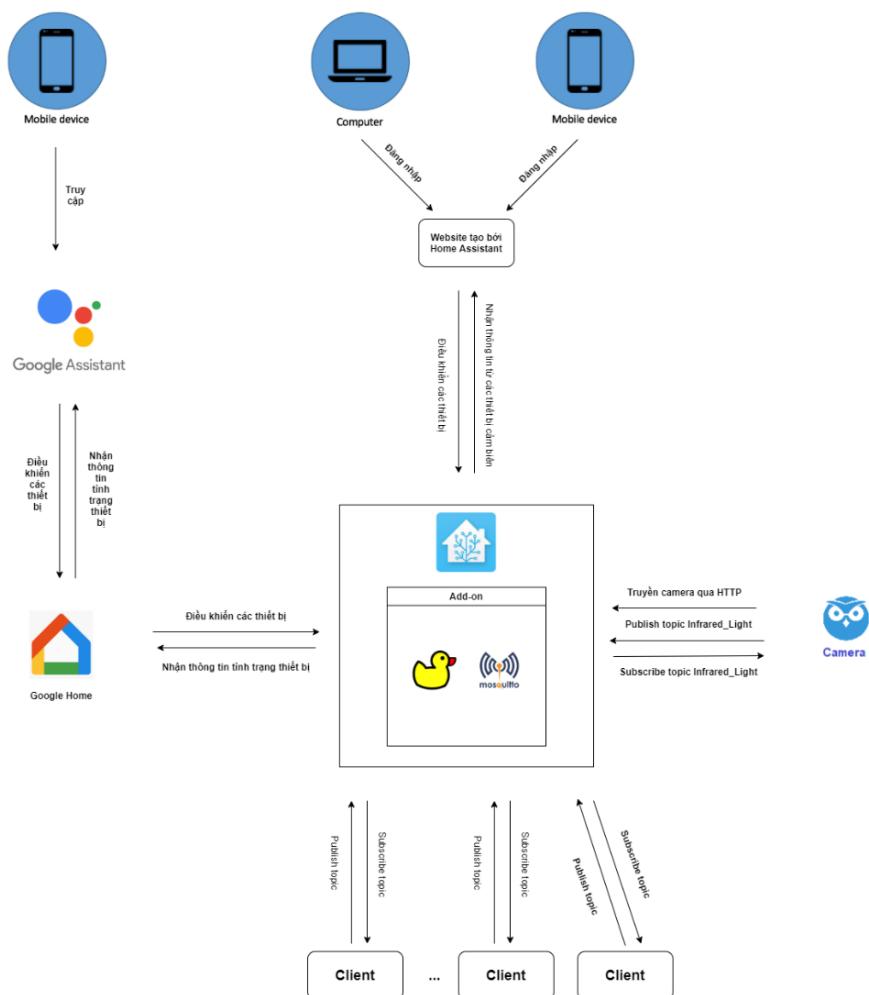
Hệ thống nhà thông minh xoay quanh một mạch Raspberry Pi là trung tâm điều khiển. Các thiết bị điện tử sẽ kết nối với nó qua giao thức MQTT hoặc HTTP, các thiết bị gồm:

- **Camera:** Được tạo bởi một mạch Pi, cùng với một module Camera quan sát và kết hợp với Module Relay và LED hồng ngoại cung cấp ánh sáng hồng ngoại cho camera nhìn thấy được vào ban đêm.
- **Hộp điều khiển thiết bị điện:** Được tạo bởi mạch NodeMCU kết hợp với Module Relay.
 - **Đèn cửa:** Được tạo bởi Hộp điều khiển thiết bị điện và đèn 220V.
- **Hộp điều khiển hồng ngoại:** Được tạo bởi mạch NodeMCU kết hợp với điện trở và LED hồng ngoại.

- **Điều khiển TV:** Được tạo bởi Hộp điều khiển hồng ngoại và lấy mẫu mã từ điều khiển TV.
- **Cảm biến nhiệt độ, độ ẩm và chất lượng không khí:** Được tạo bởi mạch NodeMCU kết hợp với cảm biến DHT11 và cảm biến MQ135.
- **Cảm biến cửa:** Được tạo bởi mạch NodeMCU kết hợp với cảm biến MC-38.
- **Chuông báo động:** Được tạo bởi mạch NodeMCU kết hợp với Module Relay và chuông điện.

Qua bộ xử lý trung tâm, nó sẽ tự động điều khiển các thiết bị kết nối hoặc thông qua các thiết bị thông minh như điện thoại, máy tính bảng,... ta có thể kết nối và điều khiển thiết bị.

1.2. Sơ đồ hệ thống phần mềm



Hình 34: Sơ đồ hệ thống phần mềm

Bộ xử lý trung tâm được quản lý bởi Home Assistant, bên trong nó sẽ được cài đặt:

- DuckDNS và Let's Encrypt: giúp truy cập vào hệ thống từ bên ngoài Internet
- MQTT: giúp kết nối các thiết bị điện tử với Home Assistant

Camera sẽ được cài đặt MotionEyeOS giúp điều khiển camera, và ta sử dụng ngôn ngữ Python để điều khiển đèn hồng ngoại.

Các thiết bị điện tử khác được lập trình bằng ngôn ngữ C.

Tất cả các thiết bị điện tử (client) đều được kết nối đến Home Assistant bằng MQTT. Mỗi client sẽ đăng ký các topic, quá trình này gọi là “Subscribe”. Mỗi client sẽ nhận được dữ liệu khi bộ xử lý trung tâm (broker) gửi dữ liệu ở topic đã đăng ký. Khi một client gửi dữ liệu tới topic đó, gọi là “Publish”.

Đặc biệt với Camera, ngoài giao thức MQTT ta sử dụng giao thức HTTP để truyền hình ảnh quay được từ camera đến Home Assistant.

Các thiết bị thông minh có thể nhận thông tin từ cảm biến hay điều khiển thiết bị qua giao diện của Home Assistant hoặc điều khiển bằng giọng nói bằng Google Assistant.

Việc điều khiển qua Google Assistant sẽ gửi lệnh trung gian qua Google Home, từ đó gửi lệnh về Home Assistant.

2. Xây dựng bộ xử lý trung tâm



Hình 35: Bộ xử lý trung tâm

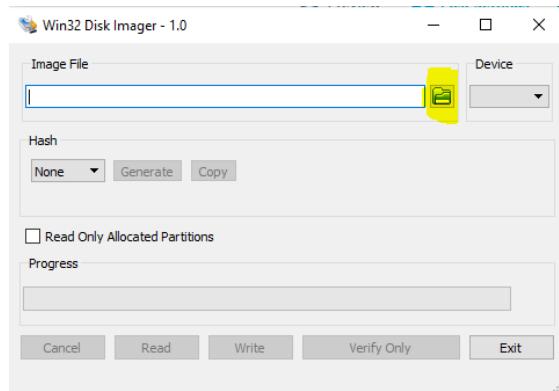
Bộ xử lý trung tâm là bộ não của toàn hệ thống nhà thông minh, giúp điều khiển, kiểm soát cũng như cài đặt chế độ tự hoạt động cho hệ thống thiết bị điện trong ngôi nhà của mình trên điện thoại, máy tính bảng ...

2.1. Cài đặt hệ điều hành Raspbian

Đầu tiên ta truy cập: <https://www.raspberrypi.org/downloads/raspbian/>
Ta tiến hành tải xuống phiên bản Raspbian Buster with desktop

Tiếp đó, ta định dạng lại thẻ nhớ dùng để làm bộ nhớ cho Pi thành định dạng SDHC

Sử dụng phần mềm Win32DiskImager để ghi hệ điều hành vừa tải về vào thẻ nhớ:



Hình 36: Phần mềm Win32DiskImager

Sau đó, ta kích hoạt SSH lần đầu cho Pi bằng cách thêm một file ssh (không có đuôi, và không có thông tin gì bên trong) vào phân vùng boot của thẻ nhớ.

Tiếp đến, cũng thêm vào một file có tên gọi *wpa_supplicant.conf* trong thẻ nhớ.

Sao khi tạo xong file, mở file này lên bằng bất cứ phần mềm Text Editor nào,

Trong file thêm đoạn code sau:

```
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

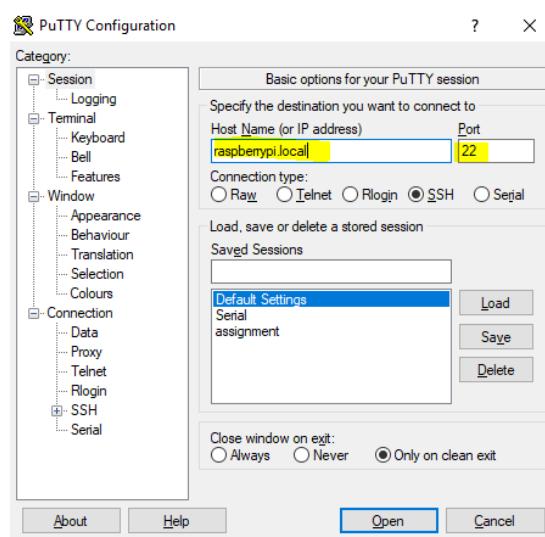
network={
    ssid="NETWORK-NAME"
    psk="NETWORK-PASSWORD"
}
```

Trong đó YOUR_NETWORK_NAME là tên mạng Wifi và YOUR_PASSWORD là mật khẩu Wifi nếu có.

Cuối cùng, chỉ cần save file này lại, tháo thẻ nhớ ra cắm vào Raspberry Pi và khởi động. Nếu có file này nó sẽ copy vào đúng nơi lưu trữ cấu hình Wifi rồi tự động kết nối Wifi đó.

Sử dụng phần mềm Putty và nhập vào như sau:

- Hostname : IP của pi
- Port 22



Hình 37: Phần mềm PuTTY

Bấm Open. Màn hình terminal đến RPi sẽ hiện lên như sau nếu không có lỗi :

```
Starting OpenBSD Secure Shell server...
[ OK ] Started OpenBSD Secure Shell server.
      Starting /etc/rc.local Compatibility...
      Starting Permit User Sessions...
[ OK ] Reached target Network is Online.
      Starting LSB: Start NTP daemon...
[ OK ] Started Turn on SSH if /boot/ssh is present.
[ OK ] Started /etc/rc.local Compatibility.
[ OK ] Started Permit User Sessions.
      Starting Terminate Plymouth Boot Screen...
      Starting Hold until boot process finishes up...

Raspbian GNU/Linux 8 rpi3 ttyS0

rpi3 login: [REDACTED]
```

Hình 38: Terminal PuTTY

Nhập vào thông tin đăng nhập như sau :

- login : pi
- password : raspberry

Để mở service SSH ta nhập lệnh:

```
sudo raspi-config
```

Sau đó chọn Interfacing Option, chọn SSH , bấm Tab để chọn Yes và bấm Enter.

Cuối cùng, ta khởi động lại:

```
sudo reboot
```



Hình 39: Màn hình Raspbian

2.2. Cài đặt Docker

Đầu tiên, ta update Pi:

```
sudo apt-get update
```

Chúng ta cài đặt các gói cần thiết trước theo lệnh sau:

```
sudo -i  
apt-get install software-properties-common  
apt-get install bash jq curl avahi-daemon dbus apparmor-utils network-manager
```

Tiếp theo, cài Docker:

```
curl -fsSL https://get.docker.com -o get-docker.sh  
sh get-docker.sh
```

Sau khi cài xong thì ta cấp phép cho user **pi** chạy docker.

```
sudo usermod -aG docker pi
```

Sau đó chúng ta sẽ cài Home Assistant.

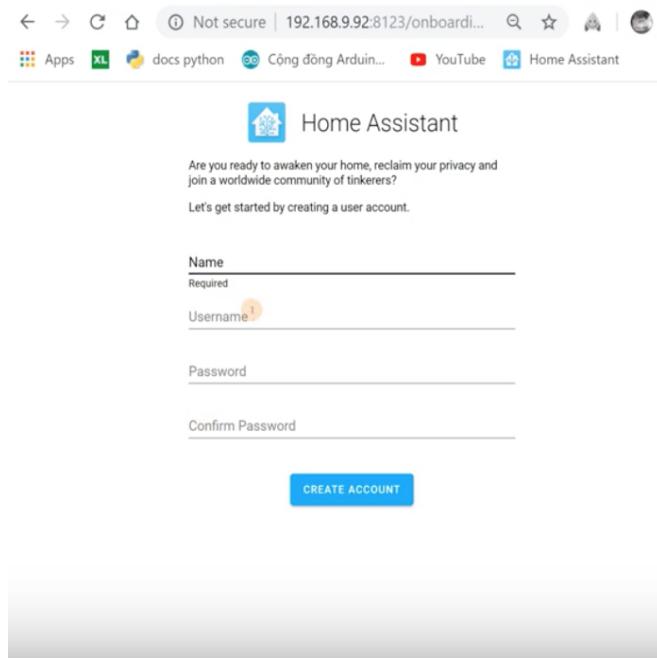
2.3. Cài đặt Home Assistant

Đầu tiên tải về file sh cài đặt:

```
curl -sL https://raw.githubusercontent.com/home-assistant/hassio-installer/master/hassio_install.sh | bash -s -- -m raspberrypi4
```

Sau khi thấy dòng Run Hass.io thì ta vào trình duyệt trên máy tính gõ IP:8123.

Chờ một lúc thì sẽ thấy logo Home Assistant hiện lên. Chờ thêm lúc nữa thì sẽ vào được Hass.



Hình 40: Màn hình tạo tài khoản Hass

Cuối cùng ta tạo một tài khoản cho Hass, vậy là hoàn thành.

2.4. Cài đặt MQTT - Mosquitto

Đầu tiên chúng ta truy cập vào HASS. Ở thanh Menu, chúng ta chọn Hass.io, nhấn Add-on Store. Tìm Add-on có tên là Mosquitto Broker rồi nhấn Install.

Sau khi tiến trình cài đặt thành công, thì chúng ta nhấn vào nút Start để khởi động MQTT.

Tiếp đó trong phần Config, thay thế dòng "**logins": []**", thành:

```
"logins": [  
    {  
        "username": "xxx",  
        "password": "xxx"  
    }  
,
```

Với username và password, là tài khoản và mật khẩu muốn đặt cho MQTT.

Sau khi khai báo xong thì nhấn Save.

Ở Terminal, ta nhập lệnh:

```
sudo nano /usr/share/hassio/homeassistant/configuration.yaml
```

Thêm đoạn sau:

```
#MQTT Broker  
  
mqtt:  
    broker: 192.168.10.86  
    username: xxx  
    password: xxx  
    discovery: True
```

Chúng ta lưu lại, Trở lại HASS, vào phần Configuration > Server Control > Check Config. Nếu hệ thống báo Configuration Valid! thì đã thành công.

Sau đó nhấn vào Reload Core và Restart để nạp lại cấu hình.

2.5. Cấu hình IP tĩnh

Đầu tiên ta nhập:

```
sudo nano /etc/dhcpcd.conf
```

Xuống dưới cùng của file, ta thêm:

```
interface wlan0
ip_address=192.168.0.200/24
routers=192.168.0.1
domain_name_servers=192.168.0.1
```

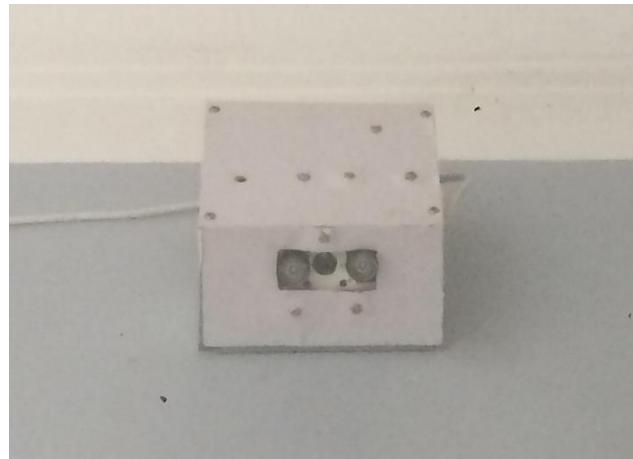
Với:

- interface : wlan0 tương ứng với mạch dùng wifi, eth0 là dùng dây cáp
- ip_address: Là IP muốn muốn đặt với đuôi /24
- routers: Địa chỉ IP của router
- domain_name_servers: Địa chỉ IP của router

Cuối cùng ta lưu file, rồi khởi động lại:

```
sudo reboot
```

3. Camera



Hình 41: Camera được chế tạo và lắp đặt hoàn thiện

Camera giúp quan sát an ninh cho ngôi nhà, nó ghi lại những hình ảnh đang diễn ra hàng ngày. Khi bạn không ở nhà camera quan sát giúp bạn bảo vệ an ninh cho ngôi nhà và kết hợp với các thiết bị thông minh khác sẽ tạo ra sự bảo vệ hết sức an toàn cho ngôi nhà của mình.

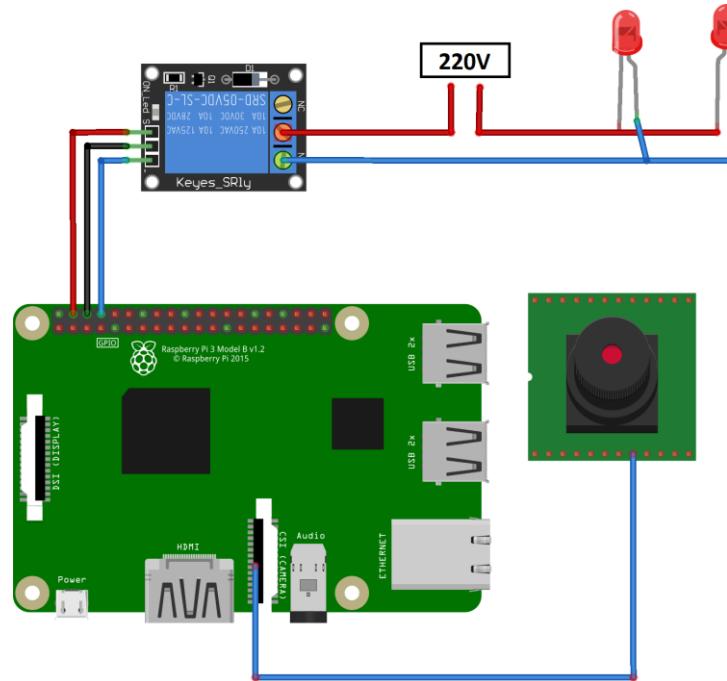
Việc tự tạo ra Camera theo cách dưới đây giúp kiểm soát thông tin quay được ở trong nội bộ, thay vì sử dụng các Camera nguyên bộ mua từ các hãng bên ngoài, thường gửi thông tin quay được lên trên server của họ trước khi gửi lại xuống thiết bị điều khiển của bạn, điều này có thể dẫn đến sự xâm phạm quyền riêng tư.

3.1. Cấu tạo phần cứng

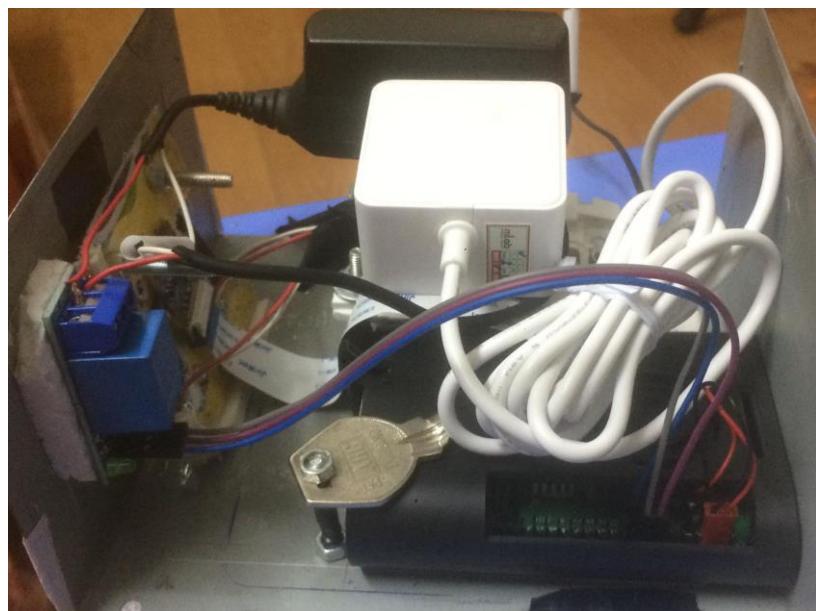
Để tạo ra Camera ta dùng những linh kiện sau:

- 1 Raspberry Pi 4 Model B 2GB
- 1 Module Relay 1 Kênh 5V10A
- 1 Raspberry Pi NoIR Camera V2
- 2 LED hồng ngoại
- 2 Nguồn điện 5V để dùng cho Raspberry và LED hồng ngoại

Từ những linh kiện đó ta đấu nối mạch như sau:



Hình 42: Sơ đồ đấu nối Camera



Hình 43: Đấu nối Camera thực tế

3.2. Cài đặt hệ điều hành và cấu hình IP tĩnh

Đầu tiên ta phải cài đặt hệ điều hành Raspbian và cấu hình IP tĩnh cho Raspberry Pi.

- Cài đặt hệ điều hành Raspbian đã được nêu ở trang 32
- Cấu hình IP tĩnh đã được nêu ở trang 38

3.3. Cài đặt và cấu hình Motioneye

Đầu tiên truy cập Terminal trên Raspbian của Camera, nhập:

```
sudo raspi-config
```

Sau đó chọn Interfacing Option, chọn Camera , bấm Tab để chọn Yes và bấm Enter. Điều này cho phép kích hoạt module Camera trên Raspberry Pi.

Tiếp đó, ta chuyển sang tài khoản “root”:

```
sudo -i
```

Ta cần cài đặt *ffmpeg* và các thành phần cần thiết cho *motion* :

```
apt-get install ffmpeg libmariadb3 libpq5 libmicrohttpd12
```

Cài đặt *motion*:

```
wget https://github.com/Motion-Project/motion/releases/download/release-4.2.2/pi_buster_motion_4.2.2-1_armhf.deb  
dpkg -i pi_buster_motion_4.2.2-1_armhf.deb
```

Cài đặt các thành phần cần thiết khác:

```
apt-get install python-pip python-dev libssl-dev libcurl4-openssl-dev libjpeg-dev libbz-dev
```

Cài đặt *motioneye*:

```
pip install motioneye
```

Chuẩn bị thư mục cấu hình:

```
mkdir -p /etc/motioneye  
cp /usr/local/share/motioneye/extra/motioneye.conf.sample  
/etc/motioneye/motioneye.conf
```

Chuẩn bị thư mục media:

```
mkdir -p /var/lib/motioneye
```

Thêm một tập lệnh khởi tạo, cấu hình nó để chạy khi Pi khởi động và khi khởi động lại máy chủ Motioneye:

```
cp /usr/local/share/motioneye/extramotioneye.systemd-unit-local  
/etc/systemd/system/motioneye.service  
systemctl daemon-reload  
systemctl enable motioneye  
systemctl start motioneye
```

Cuối cùng là nâng cấp lên phiên bản mới nhất của MotionEye:

```
pip install motioneye --upgrade  
systemctl restart motioneye
```

Tiếp đến, ta cấu hình Motioneye, sử dụng máy tính hoặc điện thoại truy cập :

```
<Địa chỉ IP Camera>:8765
```

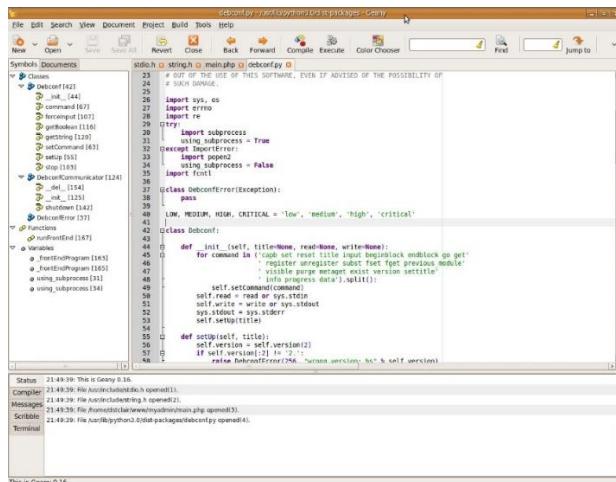
Đang nhập với tên tài khoản là “admin” và password bỏ trống.

Tiếp theo chọn “add camera...” ở góc trên cùng bên trái, chọn “*Camera type would be MMAL*” rồi nhấn OK.

Vậy cơ bản Camera đã hoàn thành. Cuối cùng ta cần lấy mã URL để truy cập camera từ Home Assistant. Ta vào cấu hình trong Motioneye, kích hoạt “*Video Streaming*”, sau đó copy hai URL là “*Snapshot URL*” và “*Streaming URL*”.

3.4. Thiết lập điều khiển LED hồng ngoại

Để lập trình trên Raspbian, ta sẽ sử dụng IDE Geany, phần mềm đã được cài sẵn trong Raspbian, giúp biên dịch ngôn ngữ Python thành chương trình.



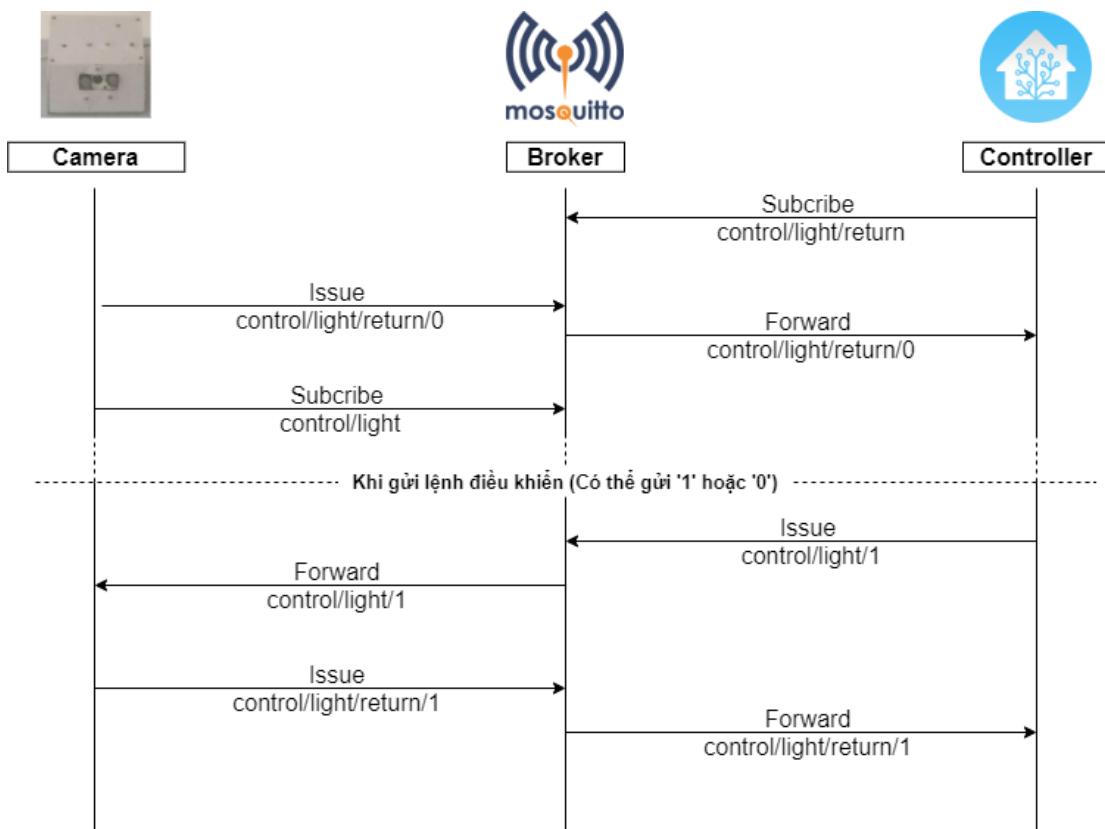
```

ledcontrol.py 100% 1/100 0:00:00
File Edit Search View Document Project Build Tools Help
New Open Save All Revert Close Back Forward Compile Execute Color Chooser Find Jump to
Symbol Documents
D Documents
D Delircntr(42)
D __init__(44)
D command(107)
D command(107)
D getlock(116)
D getstring(120)
D getvalue(120)
D setup(155)
D stop(183)
D Delircntr(124)
D __init__(114)
D __init__(125)
D shutdown(142)
D Delircntr(37)
D runFrontEnd(187)
Variables
Variables
D runInfraredProgram(185)
D __init__(185)
D using_subprocess(31)
D using_subprocess(34)
Status 21:49:39: This is Geany 0.16.
Compiler 21:49:39: File /usr/bin/python3 is executable.
Messages 21:49:39: File /usr/bin/python3 is executable.
Scrolled Terminal 21:49:39: File /usr/lib/python3.6/dist-packages/debcontrol.py opened(4).
This is Geany 0.16.

```

Hình 44: Phần mềm Geany

Để điều khiển LED hồng ngoại, ta sử dụng giao thức MQTT để gửi tin nhắn điều khiển thiết bị từ bộ điều khiển trung tâm đến Camera. Ta có sơ đồ như sau:



Hình 45: Giao thức MQTT của Đèn camera

Từ sơ đồ trên, ta lập trình để điều khiển đèn hồng ngoại trên Camera bằng ngôn ngữ Python trên IDE Geany như sau:

Code control_light.py

```
#!/usr/bin/python3

import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setup(23,GPIO.OUT)
msg=0

def on_message(client, userdata, message): #ham callback nhan tin nhan tu broker
    global msg
    msg=int(message.payload.decode("utf-8"))
    print("msg: ",str(msg))
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)
    #print("msg: ",str(msg))
    if (msg==0):
        GPIO.output(23,1) #TAT
        client.publish("control/light/return","0",retain=True) #tra ket qua ve cho
server mqtt
        print("return 0")
    else:
        GPIO.output(23,0)    #BAT
        client.publish("control/light/return","1",retain=True) #rerain dc kich hoat
        print("return 1")

def on_disconnect(client, userdata, rc): #neu disconnect?
    if rc != 0:
        print("Disconnect khong mong muon")

def on_connect(client, userdata, flags, rc):
    if rc==0:
        print("connected OK Returned code=",rc)
        client.subscribe("control/light")
        GPIO.output(23,1) #TAT
        client.publish("control/light/return","0",retain=True)
    else:
        print("Bad connection Returned code=",rc)

broker_address="192.168.1.167"
```

```

client = mqtt.Client("camera_control_light")
client.on_connect=on_connect
client.on_disconnect=on_disconnect
client.on_message=on_message
client.username_pw_set(username="mymqtt", password="mymqtt")
client.connect(broker_address)

try:
    client.loop_forever() #can thiet de ham callback co tac dung, cx co tac dung
    khi disconnect se tu dong reconnect
except KeyboardInterrupt:
    pass
except:
    pass
finally:
    client.disconnect() # disconnect
    GPIO.cleanup()

```

Về tổng quan, chương trình sẽ hoạt động như sau:

1. Khi bắt đầu, chương trình sẽ kết nối Camera đến Broker MQTT đồng thời khởi tạo trạng thái ban đầu tắt cho đèn hồng ngoại, gửi thông tin trạng thái đó cho Broker và Subscribe topic nhận lệnh điều khiển đèn.
2. Khởi tạo một vòng lặp vô hạn nhằm duy trì kết nối tới Broker đồng thời lắng nghe bản tin để điều khiển từ topic đã được subscribe trước đó.
3. Khi nhận được bản tin gửi đến, nếu bản tin là “1” thì bật đèn, nếu là “0” thì tắt đèn, đồng thời gửi trả lại trạng thái của đèn về Broker để thông báo thiết bị đã có sự thay đổi.
4. Nếu xảy ra sự mất kết nối tới Broker, thông báo lỗi sẽ được đưa ra, đồng thời thiết bị sẽ tự động kết nối lại.

3.5. Thiết lập Camera trên Home Assistant

Đầu tiên ta kết nối camera đến bộ điều khiển Pi trung tâm. Ta truy cập Terminal trên Pi trung tâm , làm như sau:

```
sudo nano /usr/share/hassio/homeassistant/configuration.yaml
```

Thêm đoạn sau:

```
camera:  
  - platform: mjpeg # camera setup with motioneye  
    name: Camera  
    mjpeg_url: <Streaming URL lấy từ trước>  
    still_image_url: <Snapshot URL lấy từ trước>  
    username: admin  
    password: !secret camera_password  
  
switch: !include switch.yaml
```

Tiếp đến ta khởi tạo file switch.yaml và truy cập nó:

```
touch /usr/share/hassio/homeassistant/switch.yaml  
sudo nano /usr/share/hassio/homeassistant/switch.yaml
```

Thêm:

```
- platform: mqtt  
  name: "Camera Light Manual"  
  state_topic: "control/light/return"  
  command_topic: "control/light"  
  payload_on: "1"  
  payload_off: "0"  
  state_on: "1"  
  state_off: "0"  
  optimistic: false  
  qos: 0  
  retain: true  
  
- platform: mqtt  
  name: "Camera Light Auto"  
  command_topic: "null"
```

```
payload_on: "1"
payload_off: "0"
```

Để điều khiển đèn hồng ngoại của camera một cách tự động, trong file automation.yaml ta có đoạn code sau:

```
- id: '1580970718836'
  alias: Bat_light_camera_tu_6h_toi
  description: ''
  trigger:
    - hours: '*'
      minutes: '*'
      platform: time_pattern
      seconds: '*'
  condition:
    - after: '18:00:00'
      before: 06:00:00
      condition: time
    - condition: and
      conditions:
        - condition: state
          entity_id: switch.camera_light_manual
          state: 'off'
  action:
    - entity_id: switch.camera_light_manual
      service: switch.turn_on
- id: '1580971084056'
  alias: Tat_light_camera_tu_6h_sang
  description: ''
  trigger:
    - hours: '*'
      minutes: '*'
      platform: time_pattern
      seconds: '*'
  condition:
    - after: 06:00:00
      before: '18:00:00'
      condition: time
    - condition: and
      conditions:
        - condition: state
          entity_id: switch.camera_light_manual
          state: 'on'

  action:
```

```

- entity_id: switch.camera_light_manual
  service: switch.turn_off
- id: '1580977058341'
  alias: Tat toan bo 2 automation Light Camera
  description: ''
  trigger:
    - entity_id: switch.camera_light_auto
      from: 'on'
      platform: state
      to: 'off'
  condition: []
  action:
    - entity_id:
        - automation.tat_light_camera_tu_6h_sang
        - automation.bat_light_camera_tu_6h_toi
      service: automation.turn_off
    - entity_id: switch.camera_light_manual
      service: switch.turn_off
- id: '1580977143929'
  alias: Bat toan bo 2 automation Light Camera
  description: ''
  trigger:
    - entity_id: switch.camera_light_auto
      from: 'off'
      platform: state
      to: 'on'
  condition: []
  action:
    - entity_id: switch.camera_light_manual
      service: switch.turn_off
    - entity_id:
        automation.bat_light_camera_tu_6h_toi,automation.tat_light_camera_tu_6h_sang
      service: automation.turn_on

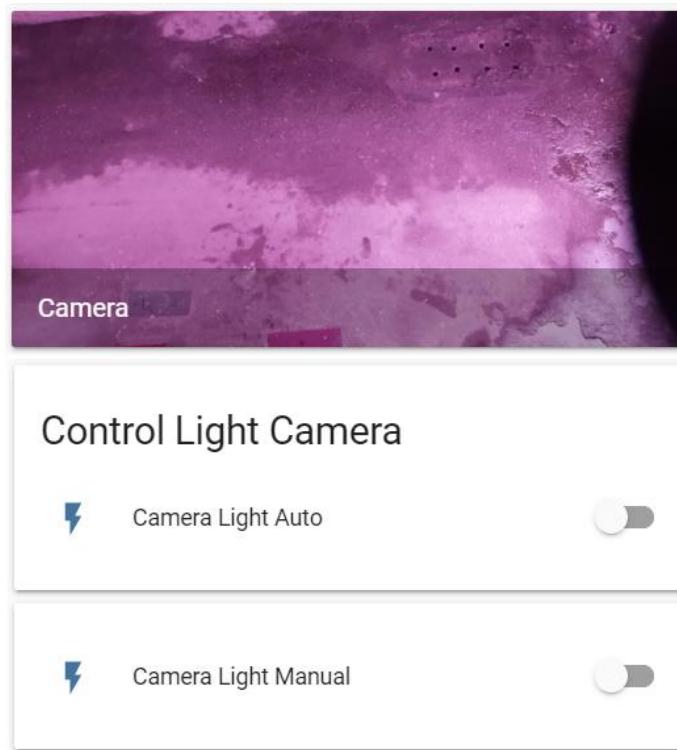
```

Đoạn code này cho phép đèn hồng ngoại tự động bật lúc 6 giờ sáng, và tắt lúc 6 giờ tối.

Để có thể người dùng điều khiển được các thiết bị trong Home Assistant, ta cần tạo ra các giao diện điều khiển các thiết bị đó trong *Lovelace*. Ta truy cập *Configure UI* và chọn *Raw config editor*, thêm đoạn config sau:

```
{  
    "aspect_ratio": "50%",  
    "camera_view": "live",  
    "entity": "camera.camera",  
    "name": "Camera",  
    "show_name": true,  
    "show_state": false,  
    "type": "picture-entity"  
},  
{  
    "entities": [  
        {  
            "entity": "switch.camera_light_auto"  
        }  
    ],  
    "show_header_toggle": false,  
    "theme": "Backend-selected",  
    "title": "Control Light Camera",  
    "type": "entities"  
},  
{  
    "card": {  
        "entities": [  
            "switch.camera_light_manual"  
        ],  
        "type": "entities"  
    },  
    "conditions": [  
        {  
            "entity": "switch.camera_light_auto",  
            "state": "off"  
        }  
    ],  
    "type": "conditional"  
},
```

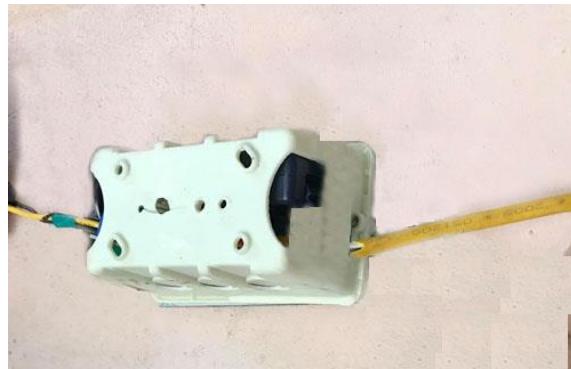
Thành quả đạt được:



Hình 46: Bảng điều khiển Camera

Thông qua bảng điều khiển camera, giúp ta quan sát bên ngoài, đồng thời cho phép ta điều khiển đèn hồng ngoại một cách tự động hoặc bằng tay, Nếu ta để chế độ tự động thì chế bộ điều khiển bằng tay sẽ tự động được ẩn đi, đèn hồng ngoại sẽ được bật lúc 6 giờ sáng và tắt lúc 6 giờ tối. Nếu muốn sử dụng chế độ bằng tay, chỉ cần tắt chế độ tự động là giao diện điều khiển bằng tay sẽ xuất hiện trở lại.

4. Hộp điều khiển thiết bị điện



Hình 47: Hộp điều khiển thiết bị điện được chế tạo và lắp hoàn thiện

Hộp điều khiển thiết bị điện giúp chúng ta biến các thiết bị bật tắt thông thường như đèn, động cơ điện, quạt thông gió, van điện từ, ... thành các thiết bị thông minh bằng cách gắn thêm thiết bị này vào các thiết bị thông thường mà không cần phải đục tường đi lại dây điện ..., đấu nối dễ dàng, bảo đảm thẩm mỹ, vì kích thước cực kỳ nhỏ gọn, giá thành lại rẻ dẫn đến chi phí cài tạo rất thấp.

Nó có thể bật tắt tự động, hẹn giờ, hoặc có thể điều khiển dễ dàng chỉ qua chiếc điện thoại ở bất cứ đâu. Đồng thời việc lắp đặt cũng rất dễ dàng, không quan trọng việc thiết bị bật tắt đã có công tắc sẵn hay chưa.

Ta có thể giới thiệu qua một số ví dụ như:

- Hộp điều khiển này lắp vào các van điện từ bình thường thì ta sẽ có được các van nước điện thông minh, từ đó ta sẽ có một hệ thống tưới tiêu, cấp nước thông minh, và sẽ rất hiệu quả khi dùng cho những khu vườn rộng lớn, hay sân gôn, ... càng thông minh khi ta kết hợp thêm cảm biến độ ẩm.
- Hộp điều khiển này lắp vào các động cơ điện ta cũng biến nó thành các động cơ điện thông minh được dùng nhiều trong cuộc sống: hệ thống bơm nước tiểu cảnh, bể cá, quạt thông gió ...
- Hộp điều khiển này lắp vào bóng đèn ở cửa ra vào, nó sẽ giúp ích rất nhiều cho chúng ta khi trời tối có ánh sáng đèn để quan sát, để mở khóa cửa, hay đi lên cầu thang.

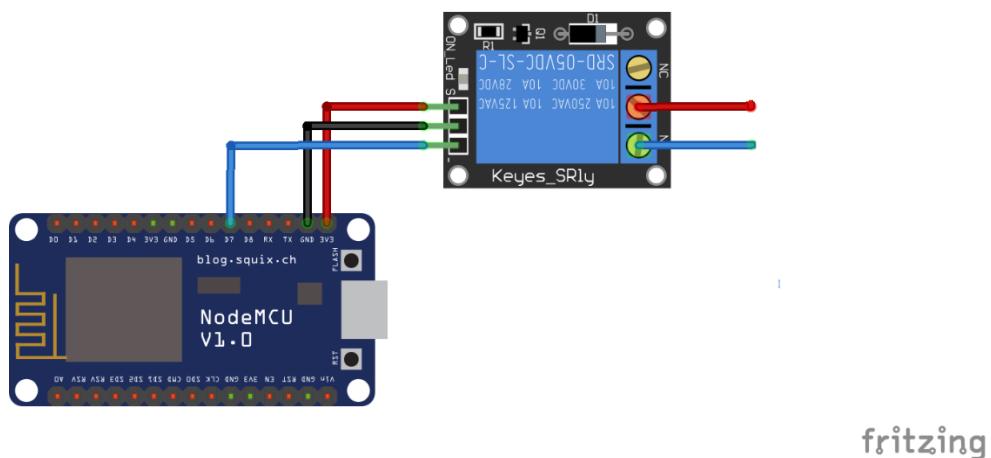
Trong dự án này em sẽ tạo ra hộp điều khiển thiết bị điện và lắp đặt nó với đèn cửa, giúp bật tắt tự động hoặc có thể điều khiển dễ dàng chỉ qua chiếc điện thoại, có thể bật tắt ở bất cứ đâu. Việc này khác với đèn thông minh mua ở bên ngoài, thường bóng đèn hỏng là không thể tái sử dụng mạch điều khiển, bóng đèn tự tạo này có thể dễ dàng thay thế bằng các bóng đèn thông thường khác nhau giúp giảm tối đa chi phí sửa chữa đồng thời việc tạo ra nó cũng rẻ hơn nhiều so với mua ngoài.

4.1. Cấu tạo phần cứng

Để tạo ra hộp điều khiển thiết bị điện ta dùng những linh kiện sau:

- 1 NodeMCU ESP8266 CP2102
- 1 Module Relay 1 Kênh 5V10A

Từ những linh kiện đó ta đấu nối mạch như sau:

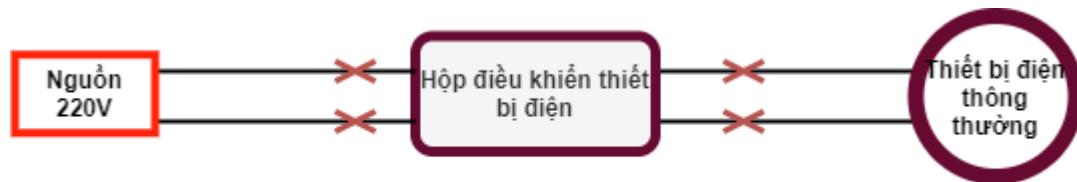


Hình 48: Sơ đồ đấu nối hộp điều khiển thiết bị điện

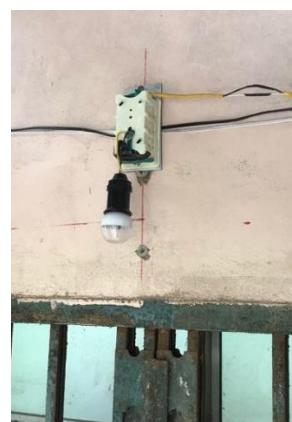


Hình 49: Đáu nối hộp điều khiển thiết bị điện thực tế

Sơ đồ lắp hộp điều khiển thiết bị điện và đèn 220V như sau:



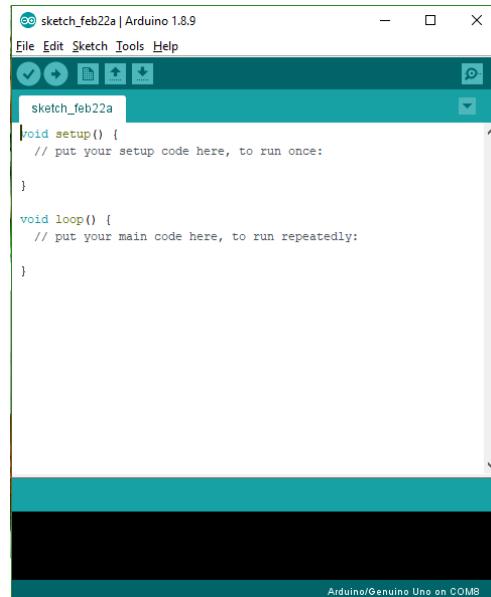
Hình 50: Đáu nối hộp điều khiển thiết bị điện với thiết bị thông thường



Hình 51: Sản phẩm đèn cửa thông minh hoàn thiện nhìn từ phía ngoài

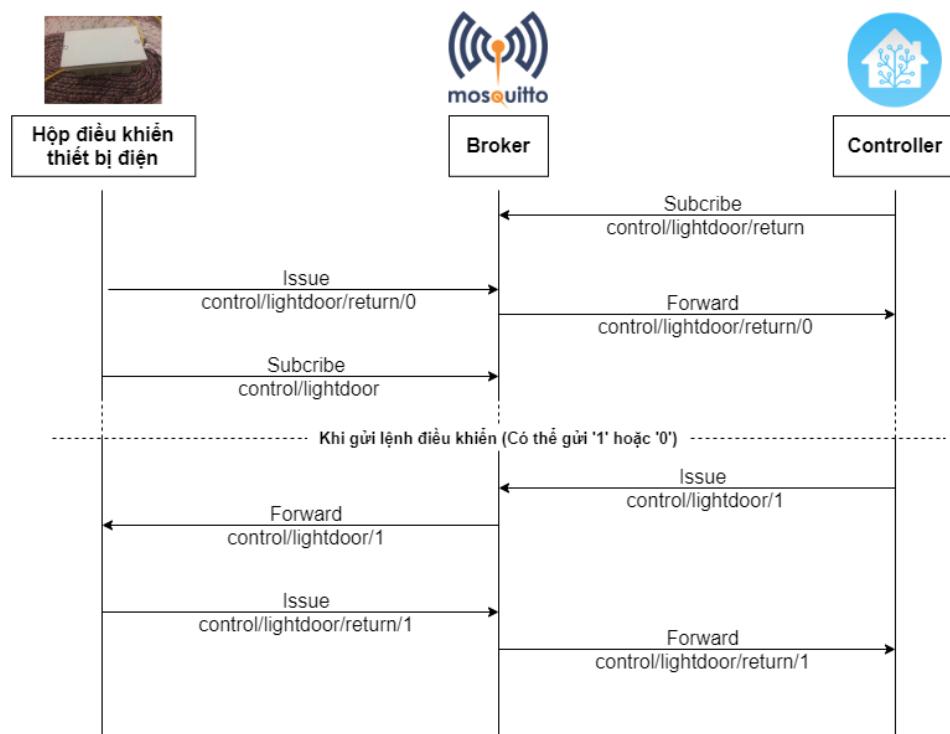
4.2. Thiết lập điều khiển hộp điều khiển thiết bị điện

Để lập trình trên NodeMCU, ta sẽ sử dụng IDE Arduino, giúp biên dịch ngôn ngữ C và nạp code cho vi điều khiển.



Hình 52: Arduino IDE

Để điều khiển hộp điều khiển thiết bị điện, ta sử dụng giao thức MQTT để gửi tin nhắn điều khiển thiết bị từ bộ điều khiển trung tâm. Ta có sơ đồ như sau:



Hình 53: Giao thức MQTT của hộp điều khiển thiết bị điện

Từ sơ đồ trên, ta lập trình để điều khiển hộp điều khiển bằng ngôn ngữ C trên IDE Arduino như sau:

Code control_lightdoor.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#define wifi_ssid "Manh"
#define wifi_password "manhvan1012"
#define mqtt_server "192.168.1.167"
#define mqtt_user "mymqtt"
#define mqtt_password "mymqtt"
const uint16_t mqtt_port = 1883;
WiFiClient espClient;
PubSubClient client(espClient);
#define pub_topic "control/lightdoor/return"
#define sub_topic "control/lightdoor"
#define RELAYPIN D7
long last = 0;
char message[100]; // lay gia tri tu mqtt server
void setup() {
    Serial.begin(9600);
    setup_wifi(); //Connect to Wifi network
    client.setServer(mqtt_server,mqtt_port);
    client.setCallback(callback);
    pinMode(RELAYPIN,OUTPUT);
}
void loop() {
    // Kiểm tra kết nối
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}
void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(wifi_ssid);
    WiFi.begin(wifi_ssid,wifi_password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
```

```

Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
    int i=0;
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (i = 0; i < length; i++) {
        message[i] = payload[i];
    }
    message[i] = '\0';
    String msgString = String(message);
    Serial.println("Message: " + msgString);
    Serial.println();
    if(msgString == "0")
    {
        digitalWrite(RELAYPIN,HIGH); //tat
        client.publish(pub_topic,"0", true);
    }
    else
    {
        digitalWrite(RELAYPIN,LOW); //bat
        client.publish(pub_topic,"1", true);
    }
}
void reconnect() {
    while (!client.connected()) {
        Serial.print("Connecting to MQTT broker ...");
        if (client.connect("ESP8266DoorLight",mqtt_user, mqtt_password)) {
            Serial.println("connected");
            // Can sub topic nao thi viet tiep duoi day:
            digitalWrite(RELAYPIN,HIGH); //tat
            client.publish(pub_topic,"0", true);
            client.subscribe(sub_topic);
        } else {
            Serial.print("failed, error = ");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}
}

```

Về tổng quan, chương trình sẽ hoạt động như sau:

1. Khi bắt đầu, vi điều khiển sẽ kết nối đến Broker MQTT đồng thời khởi tạo trạng thái ban đầu tắt cho hộp điều khiển, gửi thông tin trạng thái đó cho Broker và Subscribe topic nhận lệnh điều khiển đèn.
2. Khởi tạo một vòng lặp vô hạn nhằm duy trì kết nối tới Broker đồng thời lắng nghe bản tin để điều khiển từ topic đã được subscribe trước đó.
3. Khi nhận được bản tin gửi đến, nếu bản tin là “1” thì bật đèn, nếu là “0” thì tắt đèn, đồng thời gửi trả lại trạng thái của đèn về Broker để thông báo thiết bị đã có sự thay đổi.
4. Nếu xảy ra sự mất kết nối tới Broker thiết bị sẽ tự động kết nối lại.

4.3. Thiết lập hộp điều khiển thiết bị điện trên Home Assistant

Đầu tiên ta kết nối đèn đến bộ điều khiển Pi trung tâm. Ta thêm code trong file switch.yaml:

```
- platform: mqtt
  name: "Light Door"
  state_topic: "control/lightdoor/return"
  command_topic: "control/lightdoor"
  payload_on: "1"
  payload_off: "0"
  state_on: "1"
  state_off: "0"
  optimistic: false
  qos: 0
  retain: true
```

Trong file giao diện Lovelace ta thêm config sau:

```
{
  "entity": "switch.light_door",
  "hold_action": {
    "action": "more-info"
  },
  "icon": "mdi:track-light",
  "icon_height": "77px",
  "name": "Đèn cầu thang ngoài",
  "show_icon": true,
  "show_name": true,
```

```
"tap_action": {  
    "action": "toggle"  
},  
"type": "entity-button"  
},
```

Thành quả đạt được:



Hình 54: Công tắc điều khiển đèn cửa

Thông qua công tắc đèn cửa, ta có thể bật tắt ở bất cứ đâu một cách dễ dàng, hoặc có thể ứng dụng vào tự động hóa sẽ được đề cập ở phần sau.

5. Hộp điều khiển hồng ngoại



Hình 55: Hộp điều khiển hồng ngoại chế tạo và lắp hoàn thiện

Hộp điều khiển hồng ngoại giúp chúng ta biến các thiết bị được điều khiển từ xa thông thường như TV, điều hòa, quạt,... thành các thiết bị thông minh bằng cách điều khiển chúng thông qua hộp điều khiển hồng ngoại mà không cần thiết bị điều khiển từ xa thông thường. Việc lắp đặt cực kỳ dễ dàng, bảo đảm thẩm mỹ, vì kích thước cực kỳ nhỏ gọn, giá thành lại rẻ dẫn đến chi phí rất thấp.

Nó có thể bật tắt tự động, hẹn giờ, hoặc có thể điều khiển dễ dàng chỉ qua chiếc điện thoại ở bất cứ đâu, điều mà chiếc điều khiển từ xa thông thường khó làm được. Đồng thời có khả năng điều khiển nhiều thiết bị cùng lúc chỉ cần một hộp điều khiển hồng ngoại duy nhất giúp loại bỏ phiền phức mỗi khi đi tìm từng chiếc điều khiển cho từng loại thiết bị.

Trong dự án này em sẽ tạo ra hộp điều khiển hồng ngoại và sử dụng nó để điều khiển TV, giúp điều khiển tự động hoặc có thể điều khiển dễ dàng chỉ qua chiếc điện thoại, có thể điều khiển ở bất cứ đâu. Việc này khác với điều khiển từ xa có sẵn, vốn không thể làm được.

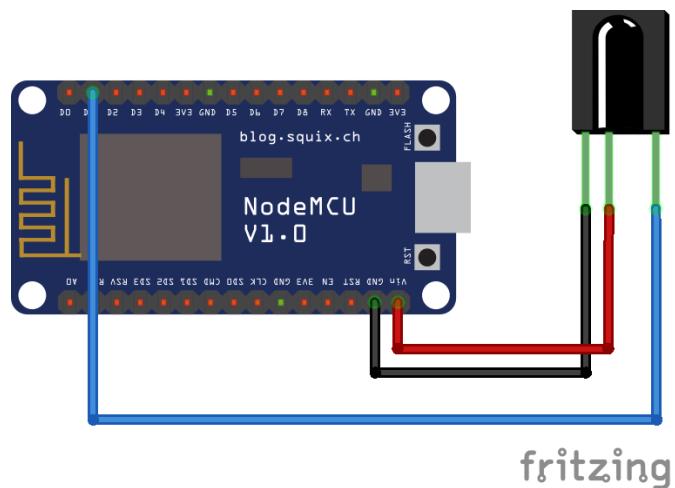
Đầu tiên, trước khi ta tạo hộp điều khiển hồng ngoại, ta cần tạo ra thiết bị thu hồng ngoại để lấy mẫu mã của từng phím bấm trên điều khiển từ xa của TV.

5.1. Cấu tạo phần cứng thiết bị thu hồng ngoại

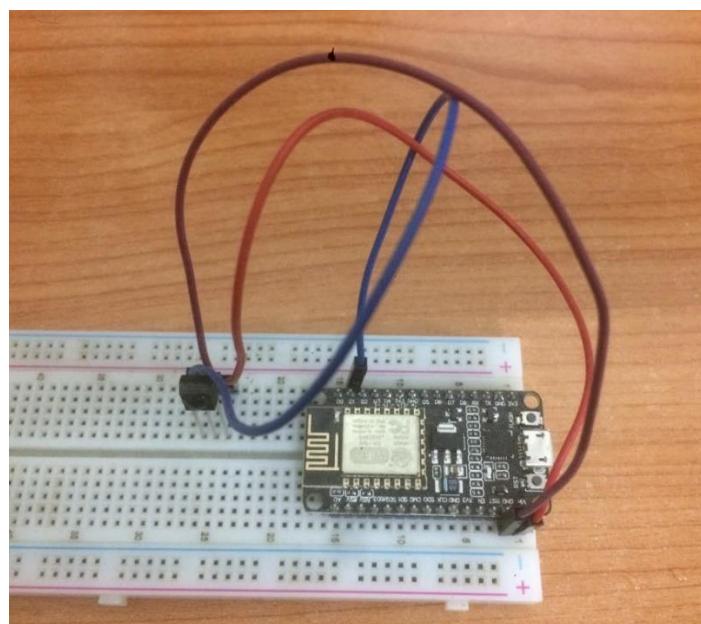
Để tạo ra cảm biến này ta dùng những linh kiện sau:

- 1 Node MCU ESP8266 CP2102
- 1 Mắt Thu Hồng Ngoại HS0038B

Từ những linh kiện đó ta đấu nối mạch như sau:



Hình 56: Sơ đồ đấu nối thiết bị thu hồng ngoại



Hình 57: Đầu nối thiết bị thu hồng ngoại thực tế

5.2. Thiết lập sử dụng thiết bị thu hồng ngoại

Ta lập trình để điều khiển thiết bị bằng ngôn ngữ C trên IDE Arduino dựa vào code của tác giả **crankyoldgit** như sau:

Nguồn source code của tác giả :

<https://github.com/crankyoldgit/IRremoteESP8266/blob/master/examples/IRrecvDumpV2/IRrecvDumpV2.ino>

Code IRrecvDumpV2.ino

```
/*
 * IRremoteESP8266: IRrecvDumpV2 - dump details of IR codes with IRrecv
 *
 * An IR detector/demodulator must be connected to the input kRecvPin.
 *
 * Copyright 2009 Ken Shirriff, http://arcfn.com
 * Copyright 2017-2019 David Conran
 *
 * Example circuit diagram:
 * https://github.com/crankyoldgit/IRremoteESP8266/wiki#ir-receiving
 *
 *
 * Changes:
 * Version 1.0 October, 2019
 *   - Internationalisation (i18n) support.
 *   - Stop displaying the legacy raw timing info.
 * Version 0.5 June, 2019
 *   - Move A/C description to IRac.cpp.
 * Version 0.4 July, 2018
 *   - Minor improvements and more A/C unit support.
 * Version 0.3 November, 2017
 *   - Support for A/C decoding for some protocols.
 * Version 0.2 April, 2017
 *   - Decode from a copy of the data so we can start capturing faster thus
 *     reduce the likelihood of miscaptures.
 * Based on Ken Shirriff's IrsendDemo Version 0.1 July, 2009,
 */

#include <Arduino.h>
#include <IRrecv.h>
```

```

#include <IRremoteESP8266.h>
#include <IRac.h>
#include <IRtext.h>
#include <IRutils.h>

// ===== start of TUNEABLE PARAMETERS =====

// An IR detector/demodulator is connected to GPIO pin 14

// e.g. D5 on a NodeMCU board.
// Note: GPIO 16 won't work on the ESP8266 as it does not have interrupts.

const uint16_t kRecvPin = D1;

// The Serial connection baud rate.
// i.e. Status message will be sent to the PC at this baud rate.

// Try to avoid slow speeds like 9600, as you will miss messages and
// cause other problems. 115200 (or faster) is recommended.

// NOTE: Make sure you set your Serial Monitor to the same speed.

const uint32_t kBaudRate = 115200;

// As this program is a special purpose capture/decoder, let us use a larger
// than normal buffer so we can handle Air Conditioner remote codes.

const uint16_t kCaptureBufferSize = 1024;

// kTimeout is the Nr. of milli-Seconds of no-more-data before we consider a
// message ended.
// This parameter is an interesting trade-off. The longer the timeout, the more
// complex a message it can capture. e.g. Some device protocols will send
// multiple message packets in quick succession, like Air Conditioner remotes.

// Air Conditioner protocols often have a considerable gap (20-40+ms) between
// packets.
// The downside of a large timeout value is a lot of less complex protocols

```

```

// send multiple messages when the remote's button is held down. The gap between
// them is often also around 20+ms. This can result in the raw data be 2-3+
// times larger than needed as it has captured 2-3+ messages in a single
// capture. Setting a low timeout value can resolve this.

// So, choosing the best kTimeout value for your use particular case is
// quite nuanced. Good luck and happy hunting.

// NOTE: Don't exceed kMaxTimeoutMs. Typically 130ms.

#ifndef DECODE_AC
// Some A/C units have gaps in their protocols of ~40ms. e.g. Kelvinator

// A value this large may swallow repeats of some protocols

const uint8_t kTimeout = 50;
#else // DECODE_AC
// Suits most messages, while not swallowing many repeats.

const uint8_t kTimeout = 15;
#endif // DECODE_AC
// Alternatives:
// const uint8_t kTimeout = 90;
// Suits messages with big gaps like XMP-1 & some aircon units, but can
// accidentally swallow repeated messages in the rawData[] output.

//
// const uint8_t kTimeout = kMaxTimeoutMs;
// This will set it to our currently allowed maximum.
// Values this high are problematic because it is roughly the typical boundary
// where most messages repeat.

// e.g. It will stop decoding a message and start sending it to serial at
// precisely the time when the next message is likely to be transmitted,
// and may miss it.

// Set the smallest sized "UNKNOWN" message packets we actually care about.

// This value helps reduce the false-positive detection rate of IR background

```

```

// noise as real messages. The chances of background IR noise getting detected

// as a message increases with the length of the kTimeout value. (See above)

// The downside of setting this message too large is you can miss some valid

// short messages for protocols that this library doesn't yet decode.

//

// Set higher if you get lots of random short UNKNOWN messages when nothing

// should be sending a message.

// Set lower if you are sure your setup is working, but it doesn't see messages

// from your device. (e.g. Other IR remotes work.)

// NOTE: Set this value very high to effectively turn off UNKNOWN detection.

const uint16_t kMinUnknownSize = 12;

// Legacy (No longer supported!)

//

// Change to `true` if you miss/need the old "Raw Timing[]" display.

#define LEGACY_TIMING_INFO false
// ===== end of TUNEABLE PARAMETERS =====

// Use turn on the save buffer feature for more complete capture coverage.

IRrecv irrecv(kRecvPin, kCaptureBufferSize, kTimeout, true);

decode_results results; // Somewhere to store the results

// This section of code runs only once at start-up.

void setup() {
#if defined(ESP8266)
  Serial.begin(kBaudRate, SERIAL_8N1, SERIAL_TX_ONLY);
#else // ESP8266
  Serial.begin(kBaudRate, SERIAL_8N1);
#endif // ESP8266
  while (!Serial) // Wait for the serial connection to be established.

  delay(50);
  Serial.printf("\n" D_STR_IRRECVDUMP_STARTUP "\n", kRecvPin);
}

```

```

#ifndef DECODE_HASH
    // Ignore messages with less than minimum on or off pulses.

    irrecv.setUnknownThreshold(kMinUnknownSize);
#endif // DECODE_HASH
    irrecv.enableIRIn(); // Start the receiver
}

// The repeating section of the code
void loop() {
    // Check if the IR code has been received.
    if (irrecv.decode(&results)) {
        // Display a crude timestamp.
        uint32_t now = millis();
        Serial.printf(D_STR_TIMESTAMP " : %06u.%03u\n", now / 1000, now % 1000);

        // Check if we got an IR message that was to big for our capture buffer.

        if (results.overflow)
            Serial.printf(D_WARN_BUFFERFULL "\n", kCaptureBufferSize);

        // Display the library version the message was captured with.

        Serial.println(D_STR_LIBRARY " : v" _IRREMOTEESP8266_VERSION_ "\n");

        // Display the basic output of what we found.
        Serial.print(resultToHumanReadableBasic(&results));
        // Display any extra A/C info if we have it.
        String description = IRAcUtils::resultAcToString(&results);

        if (description.length()) Serial.println(D_STR_MSGDESC ": " + description);

        yield(); // Feed the WDT as the text output can take a while to print.

#if LEGACY_TIMING_INFO
    // Output legacy RAW timing info of the result.
    Serial.println(resultToTimingInfo(&results));
    yield(); // Feed the WDT (again)
#endif // LEGACY_TIMING_INFO
    // Output the results as source code
    Serial.println(resultToSourceCode(&results));
    Serial.println(); // Blank line between entries
    yield(); // Feed the WDT (again)
}
}

```

Về tổng quan, chương trình sẽ hoạt động như sau:

1. Khi bắt đầu chạy, vi điều khiển sẽ đợi bất kì tia hồng ngoại điều khiển nào gửi đến mắt thu hồng ngoại.
 2. Khi ta hướng điều khiển từ xa của thiết bị nào đó đến mắt thu hồng ngoại rồi bấm một phím bất kì, mã IR của phím đó và các thông số khác của nó sẽ được lưu lại.

Hình 58: Lấy mã IR của điều khiển

Ta bấm từng phím của điều khiển từ xa đến mắt thu, ta được bảng mã IR của các phím như sau:

Protocol: NEC	
Phím điều khiển	Mã IR
Power	0xFF02FD
Mute	0xFF48B7
Audio	0xFFB847
Sub	0xFF8877
F1	0xFF728D
F2	0xFFF20D
F3	0xFF0AF5
F4	0xFF8A75
Home	0xFFB24D
Menu	0xFF6A95
Back	0xFF7887
OK	0xFF926D
Lên	0FFE21D
Xuống	0xFFD22D
Trái	0xFF12ED
Phải	0xFF52AD
<<	0xFFC23D
>>	0xFF629D
>	0xFF22DD
Vuong	0xFFA25D
Vol+	0xFF30CF
Vol-	0xFF18E7
Ch+	0xFFD02F
Ch-	0xFFA857
1	0xFF807F
2	0xFF40BF
3	0xFFC03F
4	0xFF20DF
5	0xFFA05F
6	0xFF609F
7	0FFE01F
8	0xFF10EF
9	0xFF906F
0	0xFF00FF
Del	0xFF50AF
Caps/num	0xFF6897

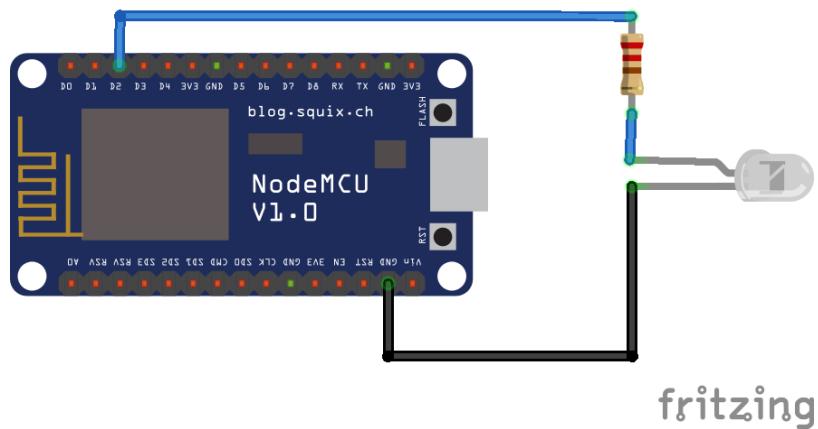
Bảng 1: Mã IR của điều khiển TV

5.3. Cấu tạo phần cứng hộp điều khiển hồng ngoại

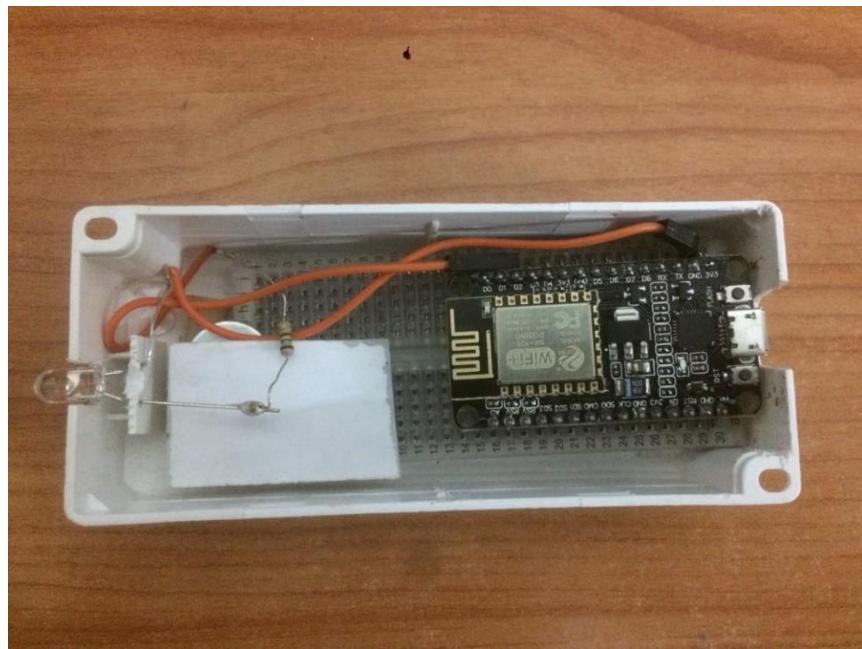
Để tạo ra hộp điều khiển hồng ngoại ta dùng những linh kiện sau:

- 1 Node MCU ESP8266 CP2102
- 1 Điện trở 30 Ohm
- 1 Led hồng ngoại

Từ những linh kiện đó ta đấu nối mạch như sau:



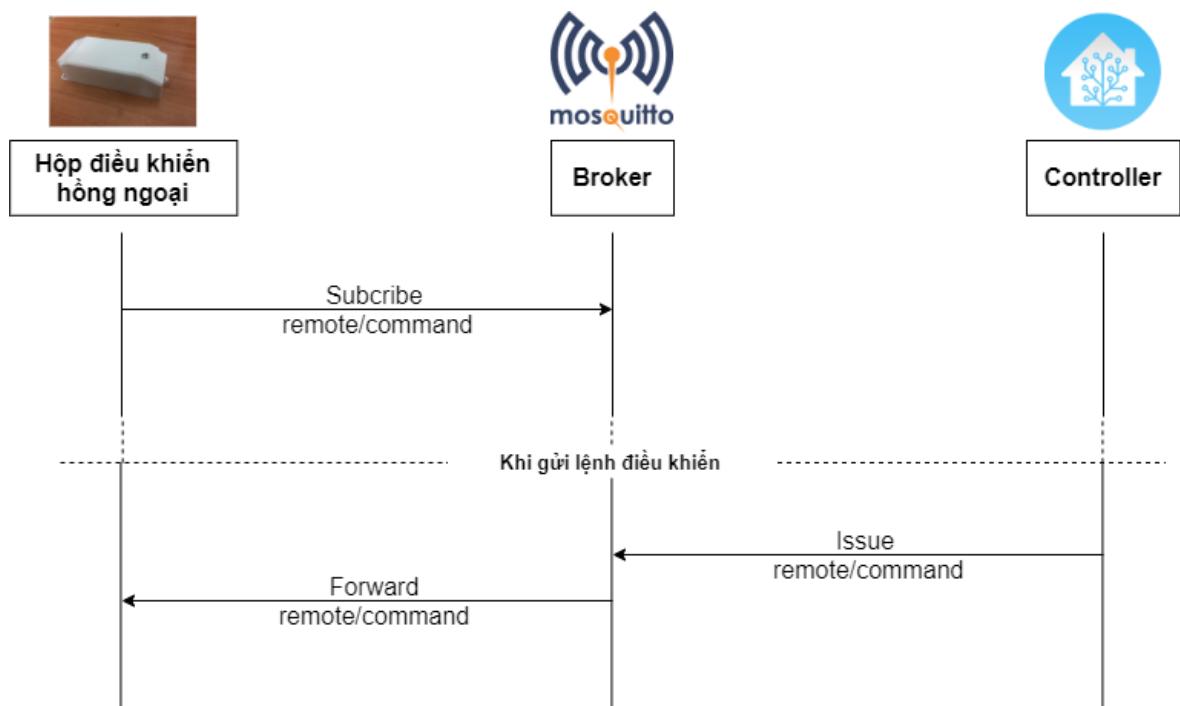
Hình 59: Sơ đồ đấu nối hộp điều khiển hồng ngoại



Hình 60: Đáu nối hộp điều khiển hồng ngoại thực tế

5.4. Thiết lập điều khiển hộp điều khiển không ngoại

Để điều khiển hộp điều khiển, ta sử dụng giao thức MQTT để gửi tin nhắn điều khiển thiết bị từ bộ điều khiển trung tâm. Ta có sơ đồ như sau:



Hình 61: Giao thức MQTT của hộp điều khiển không ngoại

Từ sơ đồ trên, ta lập trình để điều khiển cảm biến bằng ngôn ngữ C trên IDE Arduino như sau:

Code `ir_remote_send_final_version.ino`

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRsend.h>

#define wifi_ssid "Manh"
#define wifi_password "manhvan1012"
```

```

#define mqtt_server "192.168.1.167"
#define mqtt_user "mymqtt"
#define mqtt_password "mymqtt"
const uint16_t mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);

#define sub_topic "remote/command"

char message[100]; // lay gia tri tu mqtt server

const uint16_t kIrLed = 4;

IRsend irsend(kIrLed);

void setup() {
    Serial.begin(9600);
    setup_wifi();           //Connect to Wifi network
    client.setServer(mqtt_server,mqtt_port);
    client.setCallback(callback);
    irsend.begin();
}

void loop() {
    // Kiểm tra kết nối
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(wifi_ssid);
    WiFi.begin(wifi_ssid,wifi_password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
}

```

```

    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
    int i=0;
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (i = 0; i < length; i++) {
        message[i] = payload[i];
    }
    message[i] = '\0';
    String msgString = String(message);
    Serial.println("Message: " + msgString);
    Serial.println();

    if(msgString=="power")
    {
        Serial.println("SEND power");
        irsend.sendNEC(0xFF02FD,32);
        delay(50);
    }
    else
    if(msgString=="tattieng")
    {
        Serial.println("SEND tattieng");
        irsend.sendNEC(0xFF48B7,32);
        delay(50);
    }
    else
    if(msgString=="audio")
    {
        Serial.println("SEND audio");
        irsend.sendNEC(0xFFB847,32);
        delay(50);
    }
    else
    if(msgString=="sub")
    {
        Serial.println("SEND sub");
        irsend.sendNEC(0xFF8877,32);
        delay(50);
    }
    else

```

```
if(msgString=="f1")
{
    Serial.println("SEND f1");
    irsend.sendNEC(0xFF728D,32);
    delay(50);
}
else
if(msgString=="f2")
{
    Serial.println("SEND f2");
    irsend.sendNEC(0xFFFF20D,32);
    delay(50);
}
else
if(msgString=="f3")
{
    Serial.println("SEND f3");
    irsend.sendNEC(0xFF0AF5,32);
    delay(50);
}
else
if(msgString=="f4")
{
    Serial.println("SEND f4");
    irsend.sendNEC(0xFF8A75,32);
    delay(50);
}
else
if(msgString=="home")
{
    Serial.println("SEND home");
    irsend.sendNEC(0xFFB24D,32);
    delay(50);
}
else
if(msgString=="menu")
{
    Serial.println("SEND menu");
    irsend.sendNEC(0xFF6A95,32);
    delay(50);
}
else
if(msgString=="back")
{
    Serial.println("SEND back");
    irsend.sendNEC(0xFF7887,32);
```

```
    delay(50);
}
else
if(msgString=="ok")
{
    Serial.println("SEND ok");
    irsend.sendNEC(0xFF926D,32);
    delay(50);
}
else
if(msgString=="len")
{
    Serial.println("SEND len");
    irsend.sendNEC(0xFFE21D,32);
    delay(50);
}
else
if(msgString=="xuong")
{
    Serial.println("SEND xuong");
    irsend.sendNEC(0xFFD22D,32);
    delay(50);
}
else
if(msgString=="trai")
{
    Serial.println("SEND trai");
    irsend.sendNEC(0xFF12ED,32);
    delay(50);
}
else
if(msgString=="phai")
{
    Serial.println("SEND phai");
    irsend.sendNEC(0xFF52AD,32);
    delay(50);
}
else
if(msgString=="<<")
{
    Serial.println("SEND <<");
    irsend.sendNEC(0xFFC23D,32);
    delay(50);
}
else
if(msgString==">>")
```

```
{  
    Serial.println("SEND >>");  
    irsend.sendNEC(0xFF629D,32);  
    delay(50);  
}  
  
else  
if(msgString==">| |")  
{  
    Serial.println("SEND >| |");  
    irsend.sendNEC(0xFF22DD,32);  
    delay(50);  
}  
  
else  
if(msgString=="vuong")  
{  
    Serial.println("SEND vuong");  
    irsend.sendNEC(0xFFA25D,32);  
    delay(50);  
}  
  
else  
if(msgString=="vol+")  
{  
    Serial.println("SEND vol+");  
    irsend.sendNEC(0xFF30CF,32);  
    delay(50);  
}  
  
else  
if(msgString=="vol-")  
{  
    Serial.println("SEND vol-");  
    irsend.sendNEC(0xFF18E7,32);  
    delay(50);  
}  
  
else  
if(msgString=="ch+")  
{  
    Serial.println("SEND ch+");  
    irsend.sendNEC(0xFFD02F,32);  
    delay(50);  
}  
  
else  
if(msgString=="ch-")  
{  
    Serial.println("SEND ch-");  
    irsend.sendNEC(0xFFA857,32);  
    delay(50);  
}
```

```
    }
    else
    if(msgString=="mot")
    {
        Serial.println("SEND mot");
        irsend.sendNEC(0xFF807F,32);
        delay(50);
    }
    else
    if(msgString=="hai")
    {
        Serial.println("SEND hai");
        irsend.sendNEC(0xFF40BF,32);
        delay(50);
    }
    else
    if(msgString=="ba")
    {
        Serial.println("SEND ba");
        irsend.sendNEC(0xFFC03F,32);
        delay(50);
    }
    else
    if(msgString=="bon")
    {
        Serial.println("SEND bon");
        irsend.sendNEC(0xFF20DF,32);
        delay(50);
    }
    else
    if(msgString=="nam")
    {
        Serial.println("SEND nam");
        irsend.sendNEC(0xFFA05F,32);
        delay(50);
    }
    else
    if(msgString=="sau")
    {
        Serial.println("SEND sau");
        irsend.sendNEC(0xFF609F,32);
        delay(50);
    }
    else
    if(msgString=="bay")
    {
```

```

        Serial.println("SEND bay");
        irsend.sendNEC(0xFFE01F,32);
        delay(50);
    }
    else
    if(msgString=="tam")
    {
        Serial.println("SEND tam");
        irsend.sendNEC(0xFF10EF,32);
        delay(50);
    }
    else
    if(msgString=="chin")
    {
        Serial.println("SEND chin");
        irsend.sendNEC(0xFF906F,32);
        delay(50);
    }
    else
    if(msgString=="khong")
    {
        Serial.println("SEND khong");
        irsend.sendNEC(0xFF00FF,32);
        delay(50);
    }
    else
    if(msgString=="del")
    {
        Serial.println("SEND del");
        irsend.sendNEC(0xFF50AF,32);
        delay(50);
    }
    else
    if(msgString=="caps/num")
    {
        Serial.println("SEND caps/num");
        irsend.sendNEC(0xFF6897,32);
        delay(50);
    }
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Connecting to MQTT broker ...");

```

```
if (client.connect("ESP8266RemoteDo", mqtt_user, mqtt_password)) {
    Serial.println("connected");
    // Can sub topic nao thi viet tiep duoi day:
    client.subscribe(sub_topic);
} else {
    Serial.print("failed, error = ");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    delay(5000);
}
}
```

Về tổng quan, chương trình sẽ hoạt động như sau:

1. Khi bắt đầu, vi điều khiển sẽ kết nối đến Broker MQTT subscribe topic nhận phím điều khiển gửi đến từ Home Assistant.
2. Khởi tạo một vòng lặp vô hạn nhằm duy trì kết nối tới Broker đồng thời lắng nghe bản tin điều khiển từ topic đã được subscribe trước đó.
3. Khi nhận được bản tin gửi đến, dựa vào phím bấm nào, vi điều khiển sẽ gửi mã IR phím đó qua LED hồng ngoại đến thiết bị được điều khiển.
4. Nếu xảy ra sự mất kết nối tới Broker thiết bị sẽ tự động kết nối lại.

5.5. Thiết lập hộp điều khiển hồng ngoại trên Home Assistant

Đầu tiên ta thêm vào file scripts.yaml:

```
# REMOTE
'1583920006993':
  alias: power
  sequence:
  - data:
    payload: "power"
    topic: remote/command
    service: mqtt.publish
'1583920006994':
  alias: tattieng
  sequence:
  - data:
    payload: "tattieng"
    topic: remote/command
    service: mqtt.publish
'1583920006995':
  alias: audio
  sequence:
  - data:
    payload: "audio"
    topic: remote/command
    service: mqtt.publish
'1583920006996':
  alias: sub
  sequence:
  - data:
    payload: "sub"
    topic: remote/command
    service: mqtt.publish
'1583920006997':
  alias: f1
  sequence:
  - data:
    payload: "f1"
    topic: remote/command
    service: mqtt.publish
'1583920006998':
  alias: f2
  sequence:
  - data:
    payload: "f2"
```

```
topic: remote/command
service: mqtt.publish
'1583920006999':
alias: f3
sequence:
- data:
  payload: "f3"
  topic: remote/command
  service: mqtt.publish
'1583920007000':
alias: f4
sequence:
- data:
  payload: "f4"
  topic: remote/command
  service: mqtt.publish
'1583920007001':
alias: home
sequence:
- data:
  payload: "home"
  topic: remote/command
  service: mqtt.publish
'1583920007002':
alias: menu
sequence:
- data:
  payload: "menu"
  topic: remote/command
  service: mqtt.publish
'1583920007003':
alias: back
sequence:
- data:
  payload: "back"
  topic: remote/command
  service: mqtt.publish
'1583920007004':
alias: ok
sequence:
- data:
  payload: "ok"
  topic: remote/command
  service: mqtt.publish
'1583920007005':
alias: len
```

```
sequence:  
- data:  
    payload: "len"  
    topic: remote/command  
    service: mqtt.publish  
'1583920007006':  
    alias: xuong  
    sequence:  
- data:  
    payload: "xuong"  
    topic: remote/command  
    service: mqtt.publish  
'1583920007007':  
    alias: trai  
    sequence:  
- data:  
    payload: "trai"  
    topic: remote/command  
    service: mqtt.publish  
'1583920007008':  
    alias: phai  
    sequence:  
- data:  
    payload: "phai"  
    topic: remote/command  
    service: mqtt.publish  
'1583920007009':  
    alias: vvv  
    sequence:  
- data:  
    payload: "<<"  
    topic: remote/command  
    service: mqtt.publish  
'1583920007010':  
    alias: vvv2  
    sequence:  
- data:  
    payload: ">>"  
    topic: remote/command  
    service: mqtt.publish  
'1583920007011':  
    alias: vvv3  
    sequence:  
- data:  
    payload: ">||"  
    topic: remote/command
```

```
    service: mqtt.publish
'1583920007012':
  alias: vuong
  sequence:
  - data:
    payload: "vuong"
    topic: remote/command
    service: mqtt.publish
'1583920007013':
  alias: vol+
  sequence:
  - data:
    payload: "vol+"
    topic: remote/command
    service: mqtt.publish
'1583920007014':
  alias: vol-
  sequence:
  - data:
    payload: "vol-"
    topic: remote/command
    service: mqtt.publish
'1583920007015':
  alias: ch+
  sequence:
  - data:
    payload: "ch+"
    topic: remote/command
    service: mqtt.publish
'1583920007016':
  alias: ch-
  sequence:
  - data:
    payload: "ch-"
    topic: remote/command
    service: mqtt.publish
'1583920007017':
  alias: mot
  sequence:
  - data:
    payload: "mot"
    topic: remote/command
    service: mqtt.publish
'1583920007018':
  alias: hai
  sequence:
```

```
- data:  
  payload: "hai"  
  topic: remote/command  
  service: mqtt.publish  
'1583920007019':  
  alias: ba  
  sequence:  
- data:  
  payload: "ba"  
  topic: remote/command  
  service: mqtt.publish  
'1583920007020':  
  alias: bon  
  sequence:  
- data:  
  payload: "bon"  
  topic: remote/command  
  service: mqtt.publish  
'1583920007021':  
  alias: nam  
  sequence:  
- data:  
  payload: "nam"  
  topic: remote/command  
  service: mqtt.publish  
'1583920007022':  
  alias: sau  
  sequence:  
- data:  
  payload: "sau"  
  topic: remote/command  
  service: mqtt.publish  
'1583920007023':  
  alias: bay  
  sequence:  
- data:  
  payload: "bay"  
  topic: remote/command  
  service: mqtt.publish  
'1583920007024':  
  alias: tam  
  sequence:  
- data:  
  payload: "tam"  
  topic: remote/command  
  service: mqtt.publish
```

```

'1583920007025':
  alias: chin
  sequence:
  - data:
    payload: "chin"
    topic: remote/command
    service: mqtt.publish
'1583920007026':
  alias: khong
  sequence:
  - data:
    payload: "khong"
    topic: remote/command
    service: mqtt.publish
'1583920007027':
  alias: del
  sequence:
  - data:
    payload: "del"
    topic: remote/command
    service: mqtt.publish
'1583920007028':
  alias: caps/num
  sequence:
  - data:
    payload: "caps/num"
    topic: remote/command
    service: mqtt.publish

```

Trong file giao diện Lovelace ta thêm config sau:

```

cards:
  - cards:
    - entity: script.1583920006993
      hold_action:
        action: more-info
        icon: 'mdi:power'
        icon_height: 30px
        name: Power
        show_icon: true
        show_name: true

```

```
tap_action:
    action: toggle
    type: button
- entity: script.1583920006994
hold_action:
    action: more-info
icon: 'mdi:volume-variant-off'
icon_height: 30px
name: Mute
show_icon: true
show_name: true
tap_action:
    action: toggle
    type: button
- entity: script.1583920006995
hold_action:
    action: more-info
icon: 'mdi:audio-video'
icon_height: 30px
name: Audio
show_icon: true
show_name: true
tap_action:
    action: toggle
    type: button
- entity: script.1583920006996
hold_action:
    action: more-info
icon: 'mdi:subtitles'
icon_height: 30px
name: Sub
show_icon: true
show_name: true
tap_action:
```

```
action: toggle
type: button
type: horizontal-stack
- cards:
  - entity: script.1583920006997
    hold_action:
      action: more-info
      icon: 'mdi:keyboard-f1'
      icon_height: 30px
      name: ''
      show_icon: true
      show_name: true
    tap_action:
      action: toggle
      type: button
    - entity: script.1583920006998
      hold_action:
        action: more-info
        icon: 'mdi:keyboard-f2'
        icon_height: 30px
        name: ''
        show_icon: true
        show_name: true
      tap_action:
        action: toggle
        type: button
    - entity: script.1583920006999
      hold_action:
        action: more-info
        icon: 'mdi:keyboard-f3'
        icon_height: 30px
        name: ''
        show_icon: true
        show_name: true
```

```
tap_action:
    action: toggle
    type: button
- entity: script.1583920007000
hold_action:
    action: more-info
icon: 'mdi:keyboard-f4'
icon_height: 30px
name: ''
show_icon: true
show_name: true
tap_action:
    action: toggle
    type: button
type: horizontal-stack
- cards:
    - entity: script.1583920007001
hold_action:
    action: more-info
icon: 'mdi:home-circle'
icon_height: 30px
name: Home
show_icon: true
show_name: true
tap_action:
    action: toggle
    type: button
- entity: script.1583920007005
hold_action:
    action: more-info
icon: 'mdi:arrow-up-bold'
icon_height: 30px
name: ''
show_icon: true
```

```
show_name: true
tap_action:
  action: toggle
  type: button
- entity: script.1583920007002
hold_action:
  action: more-info
  icon: 'mdi:menu'
  icon_height: 30px
  name: Menu
  show_icon: true
  show_name: true
  tap_action:
    action: toggle
    type: button
  type: horizontal-stack
- cards:
  - entity: script.1583920007007
hold_action:
  action: more-info
  icon: 'mdi:arrow-left-bold'
  icon_height: 30px
  name: ''
  show_icon: true
  show_name: true
  tap_action:
    action: toggle
    type: button
- entity: script.1583920007004
hold_action:
  action: more-info
  icon: 'mdi:chess-rook'
  icon_height: 30px
  name: 'OK'
```

```
        show_icon: true
        show_name: true
        tap_action:
            action: toggle
        type: button
    - entity: script.1583920007008
        hold_action:
            action: more-info
        icon: 'mdi:arrow-right-bold'
        icon_height: 30px
        name: ''
        show_icon: true
        show_name: true
        tap_action:
            action: toggle
        type: button
    type: horizontal-stack
- cards:
    - entity: script.1583920007012
        hold_action:
            action: more-info
        icon: 'mdi:square'
        icon_height: 30px
        name: 'Off'
        show_icon: true
        show_name: true
        tap_action:
            action: toggle
        type: button
    - entity: script.1583920007006
        hold_action:
            action: more-info
        icon: 'mdi:arrow-down-bold'
        icon_height: 30px
```

```
name: ''  
show_icon: true  
show_name: true  
tap_action:  
    action: toggle  
    type: button  
- entity: script.1583920007003  
hold_action:  
    action: more-info  
icon: 'mdi:rotate-left'  
icon_height: 30px  
name: Back  
show_icon: true  
show_name: true  
tap_action:  
    action: toggle  
    type: button  
type: horizontal-stack  
- cards:  
    - entity: script.1583920007009  
hold_action:  
    action: more-info  
icon: 'mdi:skip-previous'  
icon_height: 30px  
name: ''  
show_icon: true  
show_name: true  
tap_action:  
    action: toggle  
    type: button  
- entity: script.1583920007010  
hold_action:  
    action: more-info  
icon: 'mdi:skip-next'
```

```
icon_height: 30px
name: ''
show_icon: true
show_name: true
tap_action:
  action: toggle
type: button
- entity: script.1583920007011
hold_action:
  action: more-info
icon: 'mdi:page-last'
icon_height: 30px
name: ''
show_icon: true
show_name: true
tap_action:
  action: toggle
type: button
type: horizontal-stack
type: vertical-stack
cards:
- cards:
  - entity: script.1583920007013
hold_action:
  action: more-info
icon: 'mdi:volume-plus'
icon_height: 30px
name: Volume Up
show_icon: true
show_name: true
tap_action:
  action: toggle
type: button
- entity: script.1583920007015
```

```
hold_action:
    action: more-info
    icon: 'mdi:arrow-up-bold-box'
    icon_height: 30px
    name: Channel Up
    show_icon: true
    show_name: true
    tap_action:
        action: toggle
        type: button
    type: horizontal-stack
- cards:
  - entity: script.1583920007014
    hold_action:
        action: more-info
        icon: 'mdi:volume-minus'
        icon_height: 30px
        name: Volume Down
        show_icon: true
        show_name: true
    tap_action:
        action: toggle
        type: button
  - entity: script.1583920007016
    hold_action:
        action: more-info
        icon: 'mdi:arrow-down-bold-box'
        icon_height: 30px
        name: Channel Down
        show_icon: true
        show_name: true
    tap_action:
        action: toggle
        type: button
```

```
type: horizontal-stack

- cards:

  - entity: script.1583920007017

    hold_action:

      action: more-info

      icon: 'mdi:numeric-1'

      icon_height: 30px

      name: ''

      show_icon: true

      show_name: true

      tap_action:

        action: toggle

        type: button

    - entity: script.1583920007018

      hold_action:

        action: more-info

        icon: 'mdi:numeric-2'

        icon_height: 30px

      name: ''

      show_icon: true

      show_name: true

      tap_action:

        action: toggle

        type: button

    - entity: script.1583920007019

      hold_action:

        action: more-info

        icon: 'mdi:numeric-3'

        icon_height: 30px

      name: ''

      show_icon: true

      show_name: true

      tap_action:

        action: toggle
```

```
type: button

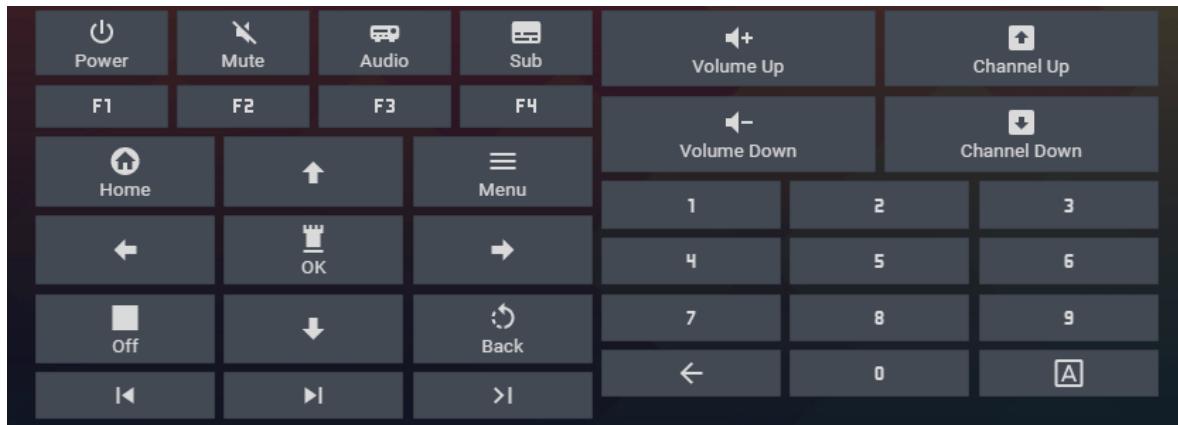
type: horizontal-stack
- cards:
  - entity: script.1583920007020
    hold_action:
      action: more-info
      icon: 'mdi:numeric-4'
      icon_height: 30px
      name: ''
      show_icon: true
      show_name: true
      tap_action:
        action: toggle
    type: button
  - entity: script.1583920007021
    hold_action:
      action: more-info
      icon: 'mdi:numeric-5'
      icon_height: 30px
      name: ''
      show_icon: true
      show_name: true
      tap_action:
        action: toggle
    type: button
  - entity: script.1583920007022
    hold_action:
      action: more-info
      icon: 'mdi:numeric-6'
      icon_height: 30px
      name: ''
      show_icon: true
      show_name: true
      tap_action:
```

```
        action: toggle
        type: button
        type: horizontal-stack
      - cards:
        - entity: script.1583920007023
          hold_action:
            action: more-info
            icon: 'mdi:numeric-7'
            icon_height: 30px
            name: ''
            show_icon: true
            show_name: true
          tap_action:
            action: toggle
            type: button
          - entity: script.1583920007024
            hold_action:
              action: more-info
              icon: 'mdi:numeric-8'
              icon_height: 30px
              name: ''
              show_icon: true
              show_name: true
            tap_action:
              action: toggle
              type: button
          - entity: script.1583920007025
            hold_action:
              action: more-info
              icon: 'mdi:numeric-9'
              icon_height: 30px
              name: ''
              show_icon: true
              show_name: true
```

```
tap_action:
    action: toggle
    type: button
    type: horizontal-stack
- cards:
  - entity: script.1583920007027
    hold_action:
        action: more-info
        icon: 'mdi:arrow-left'
        icon_height: 30px
        name: ''
        show_icon: true
        show_name: true
    tap_action:
        action: toggle
        type: button
    - entity: script.1583920007026
      hold_action:
          action: more-info
          icon: 'mdi:numeric-0'
          icon_height: 30px
          name: ''
          show_icon: true
          show_name: true
      tap_action:
          action: toggle
          type: button
    - entity: script.1583920007028
      hold_action:
          action: more-info
          icon: 'mdi:caps-lock'
          icon_height: 30px
          name: ''
          show_icon: true
```

```
show_name: true  
  
tap_action:  
    action: toggle  
    type: button  
  
type: horizontal-stack  
  
type: vertical-stack
```

Thành quả đạt được:



Hình 62: Bảng điều khiển TV

Thông qua bảng điều khiển trên, ta có thể điều khiển TV ở bất cứ đâu.

6. Cảm biến nhiệt độ, độ ẩm và chất lượng không khí



Hình 63: Cảm biến không khí được chế tạo và lắp hoàn thiện

Cảm biến giúp đo nhiệt độ, độ ẩm và chất lượng không khí ở môi trường xung quanh, giúp ích cho chúng ta quan sát độ sạch của không khí cũng như nhiệt độ, độ ẩm trong nhà.

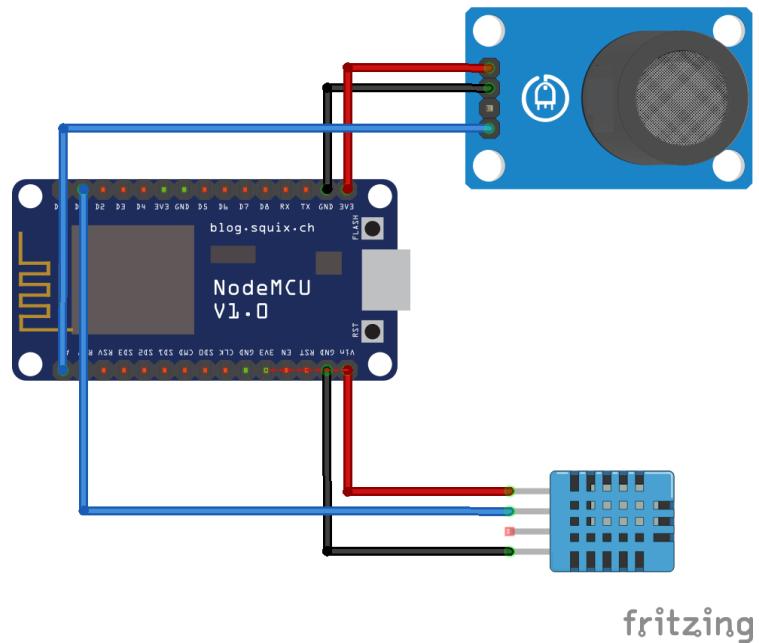
Việc tự tạo ra cảm biến theo cách dưới đây giúp dễ dàng có thể thay thế linh kiện cảm biến, giảm tối đa chi phí sửa chữa đồng thời việc tạo ra nó cũng rẻ hơn nhiều so với mua ngoài.

6.1. Cấu tạo phần cứng

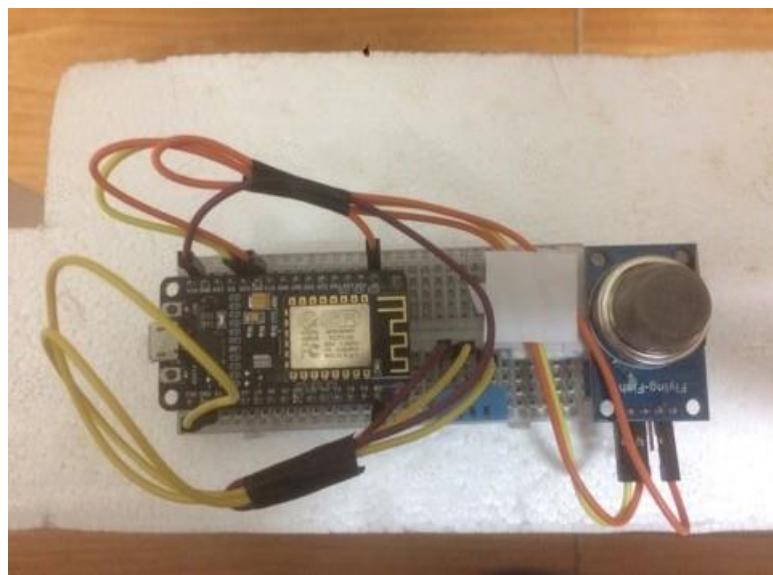
Để tạo ra cảm biến này ta dùng những linh kiện sau:

- 1 Node MCU ESP8266 CP2102
- 1 Cảm biến MQ135
- 1 Cảm biến DHT11

Từ những linh kiện đó ta đấu nối mạch như sau:



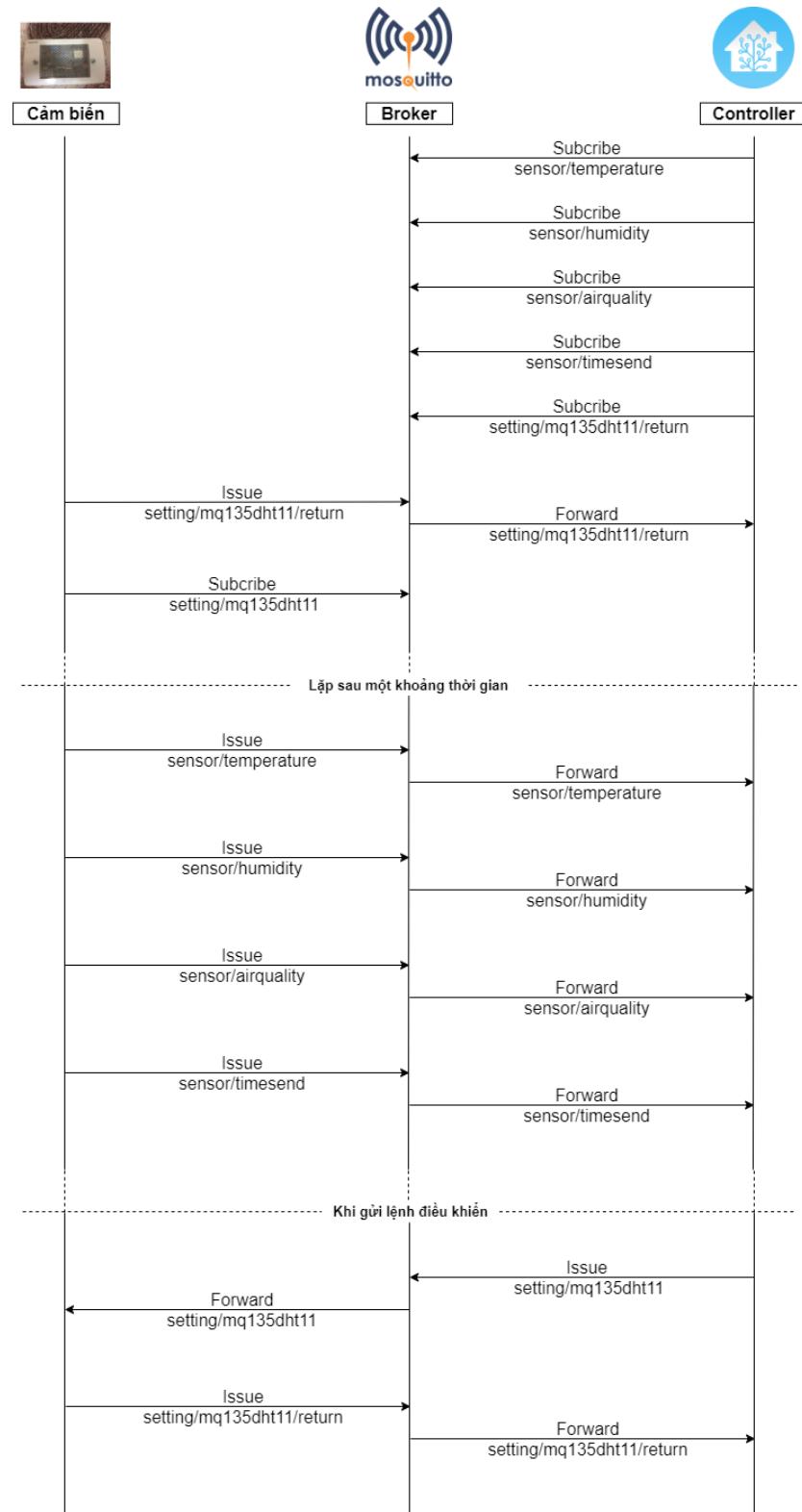
Hình 64: Sơ đồ đấu nối cảm biến không khí



Hình 65: Đáu nỗi cảm biến không khí thực tế

6.2. Thiết lập điều khiển cảm biến

Để điều khiển cảm biến, ta sử dụng giao thức MQTT để gửi tin nhắn điều khiển thiết bị từ bộ điều khiển trung tâm. Ta có sơ đồ như sau:



Hình 66: Giao thức MQTT của cảm biến không khí

Từ sơ đồ trên, ta lập trình để điều khiển cảm biến bằng ngôn ngữ C trên IDE Arduino như sau:

Code MQ135_DHT11.ino

```
#include <PubSubClient.h>
#include <NTPtimeESP.h>
#include "DHT.h"
#include "MQ135.h"
#include <ESP8266WiFi.h>
NTPtime NTPch("ch.pool.ntp.org");
#define wifi_ssid "Manh"
#define wifi_password "manhvan1012"
#define mqtt_server "192.168.1.167"
#define mqtt_user "mymqtt"
#define mqtt_password "mymqtt"
const uint16_t mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);
#define temperature_topic "sensor/temperature"
#define humidity_topic "sensor/humidity"
#define air_quality_topic "sensor/airquality"
#define time_send_topic "sensor/timesend"
#define delay_topic "setting/mq135dht11"
#define delay_topic_return "setting/mq135dht11/return"

int delaytime=5;
unsigned long last = 0;
char message[100]; // lay gia tri tu mqtt server
int check =1;

#define DHTTYPE DHT11
#define DHTPIN D1
DHT dht(DHTPIN, DHTTYPE);
#define PIN_MQ135 A0 //Khai báo pin nối với chân A0
MQ135 mq135_sensor = MQ135(PIN_MQ135); //Khai báo đối tượng thư viện

strDateTime dateTime;
void setup() {
  Serial.begin(9600);
  setup_wifi(); //Connect to Wifi network
  client.setServer(mqtt_server,mqtt_port);
  client.setCallback(callback);
  dht.begin();
}
```

```

void loop() {
    // Kiểm tra kết nối
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    if(check == 1){
        delaytime = delaytime*1000;
        check=0;
    }
    // Serial.print(delaytime);
    if((unsigned long)(millis()-last) > delaytime)
    {
        dateTIme = NTPch.getNTPtime(7.0, 0);
        if(dateTIme.valid){
            byte Hour = dateTIme.hour;           // Giờ
            byte Minute = dateTIme.minute;      // Phút
            byte Second = dateTIme.second;      // Giây
            int Year = dateTIme.year;          // Năm
            byte Month = dateTIme.month;       // Tháng
            byte Day =dateTIme.day;            // Ngày

            String lolol = String(Hour)+"h/"+String(Minute)+"p/"+String(Second)+"s
"+String(Day)+"/"+String(Month)+"/"+String(Year);
            float humidity = dht.readHumidity();
            float temperature = dht.readTemperature();
            Serial.print("Temperature : ");
            Serial.print(temperature);
            Serial.print(" | Humidity : ");
            Serial.println(humidity);
            float correctedPPM = mq135_sensor.getCorrectedPPM(temperature, humidity);
            Serial.print("PPM: ");
            Serial.print(correctedPPM);
            Serial.println("ppm");
            float resistance = mq135_sensor.getResistance();
            Serial.print("\t Resistance: ");
            Serial.print(resistance);
            client.publish(temperature_topic, String(temperature).c_str(), true);    //false: ko
cho tin nhan giu lai tren mqtt server
            client.publish(humidity_topic, String(humidity).c_str(), true);
            client.publish(air_quality_topic, String(correctedPPM).c_str(), true);
            client.publish(time_send_topic,lolol.c_str() , true);
            last = millis();
        }
    }
}

```

```

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(wifi_ssid);
    WiFi.begin(wifi_ssid,wifi_password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
    int i=0;
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (i = 0; i < length; i++) {
        message[i] = payload[i];
    }
    message[i] = '\0';
    String msgString = String(message);
    Serial.println("Message: " + msgString);
    Serial.println();
    delaytime = msgString.toInt();
    client.publish(delay_topic_return,String(delaytime).c_str(), true);
    check=1;
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Connecting to MQTT broker ...");
        if (client.connect("ESP8266Client",mqtt_user, mqtt_password)) {
            Serial.println("connected");
            // Can sub topic nao thi viet tiep duoi day:
            client.publish(delay_topic_return,String(delaytime).c_str(), true);
            client.subscribe(delay_topic);
        } else {
            Serial.print("failed, error = ");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}

```

```
}
```

Về tổng quan, chương trình sẽ hoạt động như sau:

5. Khi bắt đầu, vi điều khiển sẽ kết nối đến Broker MQTT đồng thời khởi tạo thời gian gửi lại thông tin của cảm biến lên server là 5 giây, thông tin thời gian đó được gửi cho Broker và vi điều khiển subscribe topic nhận lệnh sửa thời gian từ Home Assistant gửi lại của cảm biến.
6. Khởi tạo một vòng lặp vô hạn nhằm duy trì kết nối tới Broker đồng thời lắng nghe bản tin thời gian từ topic đã được subscribe trước đó.
7. Khi nhận được bản tin gửi đến, nó chính là thời gian chờ để thông tin cảm biến tiếp theo được gửi lên broker.
8. Cứ sau một khoảng thời gian chờ đó, thông tin của cảm biến bao gồm: nhiệt độ, độ ẩm, chất lượng không khí và thời điểm gửi thông tin sẽ được gửi lên server.
9. Nếu xảy ra sự mất kết nối tới Broker thiết bị sẽ tự động kết nối lại.

6.3. Thiết lập cảm biến trên Home Assistant

Đầu tiên ta kết nối cảm biến đến bộ điều khiển Pi trung tâm. Ta thêm code trong file configuration.yaml:

```
sensor: !include sensor.yaml
input_text: !include input_text.yaml
```

Tiếp đó ta tạo file sensor.yaml và file input_text.yaml:

```
touch /usr/share/hassio/homeassistant/sensor.yaml
touch /usr/share/hassio/homeassistant/input_text.yaml
```

Thêm vào file sensor.yaml:

```
- platform: mqtt
  name: "Temperature"
  state_topic: "sensor/temperature"
  unit_of_measurement: '°C'
```

```
- platform: mqtt
  name: "Humidity"
  expire_after: 0
  state_topic: "sensor/humidity"
  unit_of_measurement: '%'
- platform: mqtt
  name: "Airquality"
  expire_after: 0
  state_topic: "sensor/airquality"
  unit_of_measurement: "ppm"
- platform: mqtt
  name: "Delay time MQ135 DHT11"
  expire_after: 0
  state_topic: "setting/mq135dht11/return"
  unit_of_measurement: "s"
- platform: mqtt
  name: "Time send MQ135 DHT11"
  expire_after: 0
  state_topic: "sensor/timesend"
```

Thêm vào file input_text.yaml:

```
time_delay_mq135_dht11:
  name: Time Delay MQ135 DHT11 (s)
  pattern: "[1-9][0-9]{1,}"
```

Thêm vào file scripts.yaml:

```
'1582191205172':
  alias: Set time delay MQ135 and DHT11
  sequence:
    - data:
        payload_template: '{{ states.input_text.time_delay_mq135_dht11.state }}'
        retain: true
        topic: setting/mq135dht11
        service: mqtt.publish
```

Trong file giao diện Lovelace ta thêm config sau:

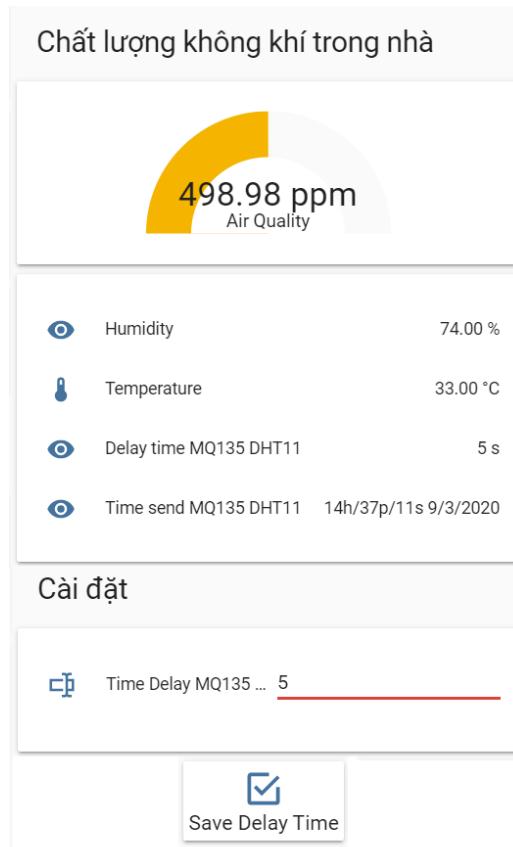
```
{
  "cards": [
    {
      "entity": "sensor.airquality",
      "max": 1000,
      "min": 0,
      "name": "Air Quality",
      "severity": {
        "green": 0,
        "red": 600,
        "yellow": 450
      },
      "theme": "default",
      "type": "gauge",
      "unit": "ppm"
    },
    {
      "entities": [
        {
          "entity": "sensor.humidity"
        },
        {
          "entity": "sensor.temperature"
        },
        {
          "entity": "sensor.delay_time_mq135_dht11"
        },
        {
          "entity": "sensor.time_send_mq135_dht11"
        }
      ],
      "show_header_toggle": false,
      "type": "entities"
    }
  ],
  "title": "Chất lượng không khí trong nhà",
  "type": "vertical-stack"
},
{
  "cards": [
    {
      "entities": [
        {
          "entity": "input_text.time_delay_mq135_dht11"
        }
      ],
      "show_header_toggle": false,
      "type": "entities"
    },
    {
      "cards": [
```

```

    {
        "entity": "script.1582191205172",
        "hold_action": {
            "action": "more-info"
        },
        "icon": "mdi:checkbox-marked-outline",
        "icon_height": "35px",
        "name": "Save Delay Time",
        "show_icon": true,
        "show_name": true,
        "tap_action": {
            "action": "toggle"
        },
        "type": "entity-button"
    }
],
"type": "horizontal-stack"
}
],
"title": "Cài đặt",
"type": "vertical-stack"
}

```

Thành quả đạt được:



Hình 67: Bảng nhiệt độ, độ ẩm và chất lượng không khí

Thông qua bảng cảm biến trên, giúp ta theo dõi được nhiệt độ, độ ẩm, chất lượng không khí, thời gian chờ cập nhật thông tin kế tiếp và thời điểm gửi thông tin lần cuối cùng. Riêng với chất lượng không khí được hiển thị bằng biểu đồ, nếu chất lượng không khí dưới 450 ppm thì là màu xanh lá cây, từ 450 ppm đến 600ppm là màu vàng và cuối cùng trên mức đó là màu đỏ biểu thị mức độ ô nhiễm đáng báo động.

Phía dưới là bảng cài đặt, cho phép ta chỉnh lại thời gian chờ cập nhật thông tin kế tiếp, với đơn vị giây.

7. Cảm biến cửa



Hình 68: Cảm biến cửa được chế tạo và lắp đặt hoàn thiện

Cảm biến được gắn ở cánh cửa sẽ phát hiện khi nào cửa đóng hay mở, góp phần bảo vệ an toàn trong ngôi nhà. Để tăng cường độ an toàn, cảm biến cửa được kết hợp với chuông báo động sẽ được đề cập ở phần sau.

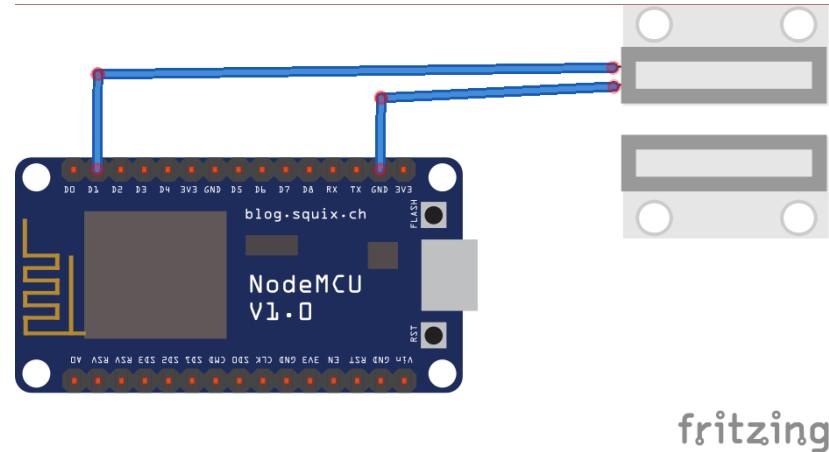
Việc tự tạo ra cảm biến cửa theo cách dưới đây giúp dễ dàng có thể thay thế linh kiện cảm biến, giảm tối đa chi phí sửa chữa đồng thời việc tạo ra nó cũng rẻ hơn nhiều so với mua ngoài.

7.1. Cấu tạo phần cứng

Để tạo ra cảm biến này ta dùng những linh kiện sau:

- 1 Node MCU ESP8266 CP2102
- 1 Cảm biến cửa MC-38

Từ những linh kiện đó ta đấu nối mạch như sau:



fritzing

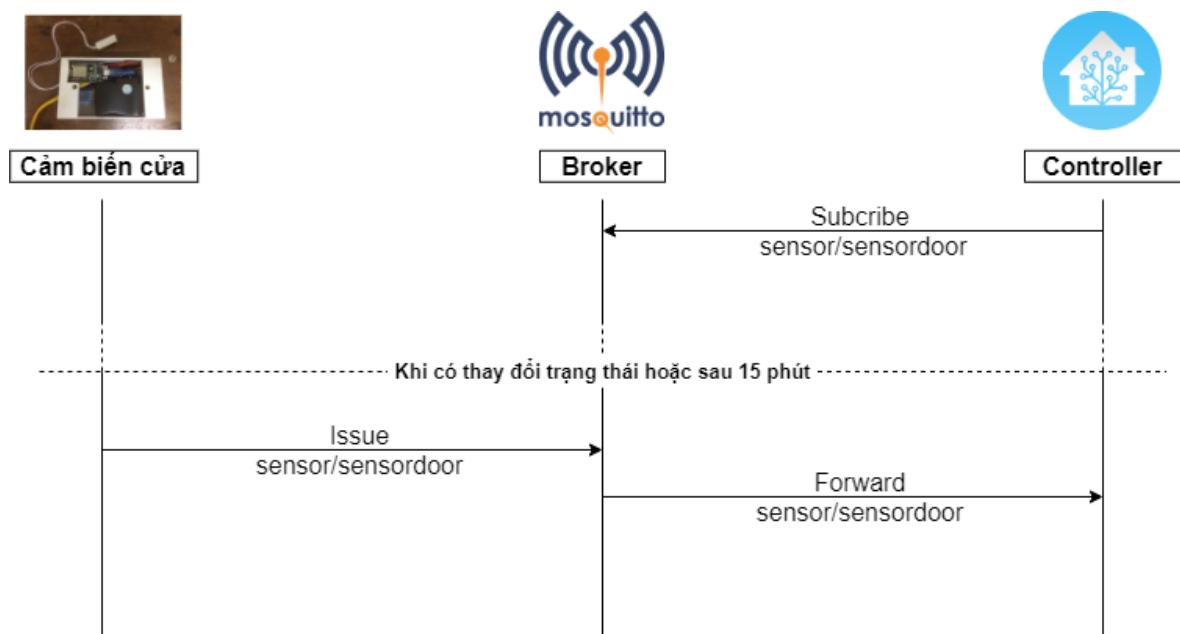
Hình 69: Sơ đồ đấu nối cảm biến cửa



Hình 70: Đầu nối cảm biến cửa thực tế

7.2. Thiết lập điều khiển cảm biến cửa

Để điều khiển cảm biến cửa, ta sử dụng giao thức MQTT để gửi tin nhắn điều khiển thiết bị từ bộ điều khiển trung tâm. Ta có sơ đồ như sau:



Hình 71: Giao thức MQTT của cảm biến cửa

Từ sơ đồ trên, ta lập trình để điều khiển cảm biến cửa bằng ngôn ngữ C trên IDE Arduino như sau:

Code door_sensor.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

#define wifi_ssid "Manh"
#define wifi_password "manhvan1012"
#define mqtt_server "192.168.1.167"
#define mqtt_user "mymqtt"
#define mqtt_password "mymqtt"
const uint16_t mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);
```

```

#define pub_topic "sensor/sensordoor"

unsigned long last = 0;
unsigned long last2 = 0;
char message[100]; // lay gia tri tu mqtt server


const int sensor = D1;
int state;
int before_state = 3;
int delaytime = 900000; //15p


void setup() {
    Serial.begin(9600);
    setup_wifi();           //Connect to Wifi network
    client.setServer(mqtt_server,mqtt_port);
    client.setCallback(callback);
    pinMode(sensor, INPUT_PULLUP);
}

void loop() {
    // Kiểm tra kết nối
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    if((unsigned long)(millis()-last2) > 200)
    {
        state = digitalRead(sensor);

        if((state != before_state) || ((unsigned long)(millis()-last) > delaytime) )
        {

            if (state == HIGH){
                client.publish(pub_topic,"Mở cửa", true);
                Serial.println("mo cua");
            }
            else
            {
                client.publish(pub_topic,"Đóng cửa", true);

                Serial.println("dong cua");
            }
        }
    }
}

```

```

        }
        before_state = state;
        last = millis();
    }
    last2 = millis();
}
}

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(wifi_ssid);
    WiFi.begin(wifi_ssid,wifi_password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {

    int i=0;
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (i = 0; i < length; i++) {
        message[i] = payload[i];
    }
    message[i] = '\0';
    String msgString = String(message);
    Serial.println("Message: " + msgString);
    Serial.println();
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Connecting to MQTT broker ...");
        if (client.connect("ESP8266DoorSensor",mqtt_user, mqtt_password)) {
            Serial.println("connected");
            // Can sub topic nao thi viet tiep duoi day:
        } else {

```

```

        Serial.print("failed, error = ");
        Serial.print(client.state());
        Serial.println(" try again in 5 seconds");
        delay(5000);
    }
}

```

Về tổng quan, chương trình sẽ hoạt động như sau:

1. Khi bắt đầu, vi điều khiển sẽ kết nối đến Broker MQTT.
2. Khởi tạo một vòng lặp vô hạn nhằm duy trì kết nối tới Broker đồng thời nếu có sự thay đổi trạng thái của cảm biến hoặc quá 15 phút, vi điều khiển sẽ gửi bản tin trạng thái của cảm biến lên broker.
3. Nếu xảy ra sự mất kết nối tới Broker thiết bị sẽ tự động kết nối lại.

7.3. Thiết lập cảm biến cửa trên Home Assistant

Đầu tiên ta kết nối cảm biến đến bộ điều khiển Pi trung tâm. Ta thêm code trong file sensor.yaml:

```

- platform: mqtt
  name: "Sensor door"
  expire_after: 960 # 16phut
  state_topic: "sensor/sensordoor"
  icon: mdi:home-lock
- platform: time_date
  display_options:
    - 'time'

- platform: template
  sensors:

    time_open_door_sensor:
      friendly_name: "Time Door Open"

```

```

    value_template: '{% if is_state("sensor.sensor_door","Mở cửa") %} {{ relative_time(states.sensor.sensor_door.last_updated) }} {% endif %} {% if is_state("sensor.sensor_door","Đóng cửa") %} None {% endif %}'
    entity_id: sensor.time

```

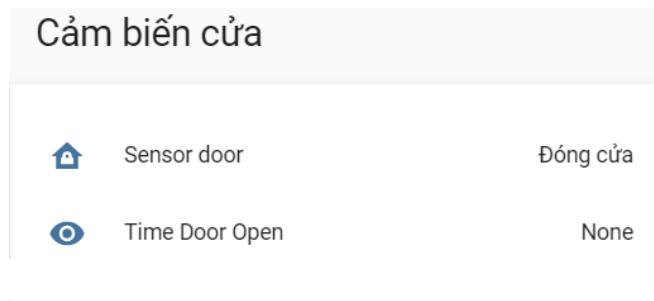
Trong file giao diện Lovelace ta thêm config sau:

```

{
  "cards": [
    {
      "entities": [
        {
          "entity": "sensor.sensor_door"
        },
        {
          "entity": "sensor.time_open_door_sensor"
        }
      ],
      "show_header_toggle": false,
      "type": "entities"
    }
  ],
  "title": "Cảm biến cửa",
  "type": "vertical-stack"
}

```

Thành quả đạt được:



Hình 72: Bảng cảm biến cửa

Thông qua bảng cảm biến trên, giúp ta theo dõi được cửa hiện tại đang đóng hay mở, và nếu cửa mở thì cửa đã mở được bao lâu. Cảm biến cửa sẽ được kết hợp với chuông báo động một cách tự động hóa sẽ được đề cập ở phần sau.

8. Chuông báo động



Hình 73: Chuông báo động được chế tạo và lắp hoàn thiện

Chuông báo động được đặt dễ dàng ở bất cứ đâu trong ngôi nhà, kết hợp với cảm biến cửa, khi cửa mở chuông báo động sẽ kêu lên, góp phần bảo vệ an toàn trong ngôi nhà.

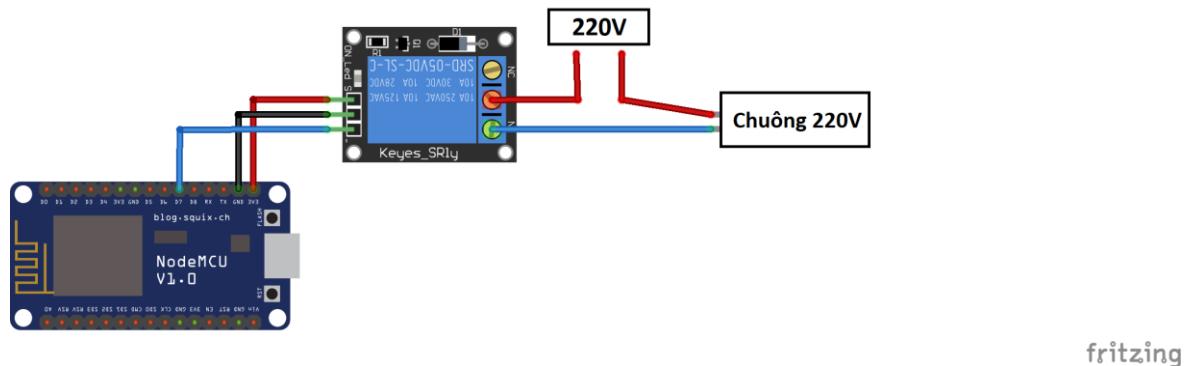
Việc tự tạo ra chuông báo động theo cách dưới đây giúp dễ dàng có thể thay thế linh kiện, giảm tối đa chi phí sửa chữa đồng thời việc tạo ra nó cũng rẻ hơn nhiều so với mua ngoài.

8.1. Cấu tạo phần cứng

Để tạo ra chuông báo động ta dùng những linh kiện sau:

- 1 Node MCU ESP8266 CP2102
- 1 Module Relay 1 Kênh 5V10A
- 1 Chuông điện

Từ những linh kiện đó ta đấu nối mạch như sau:



fritzing

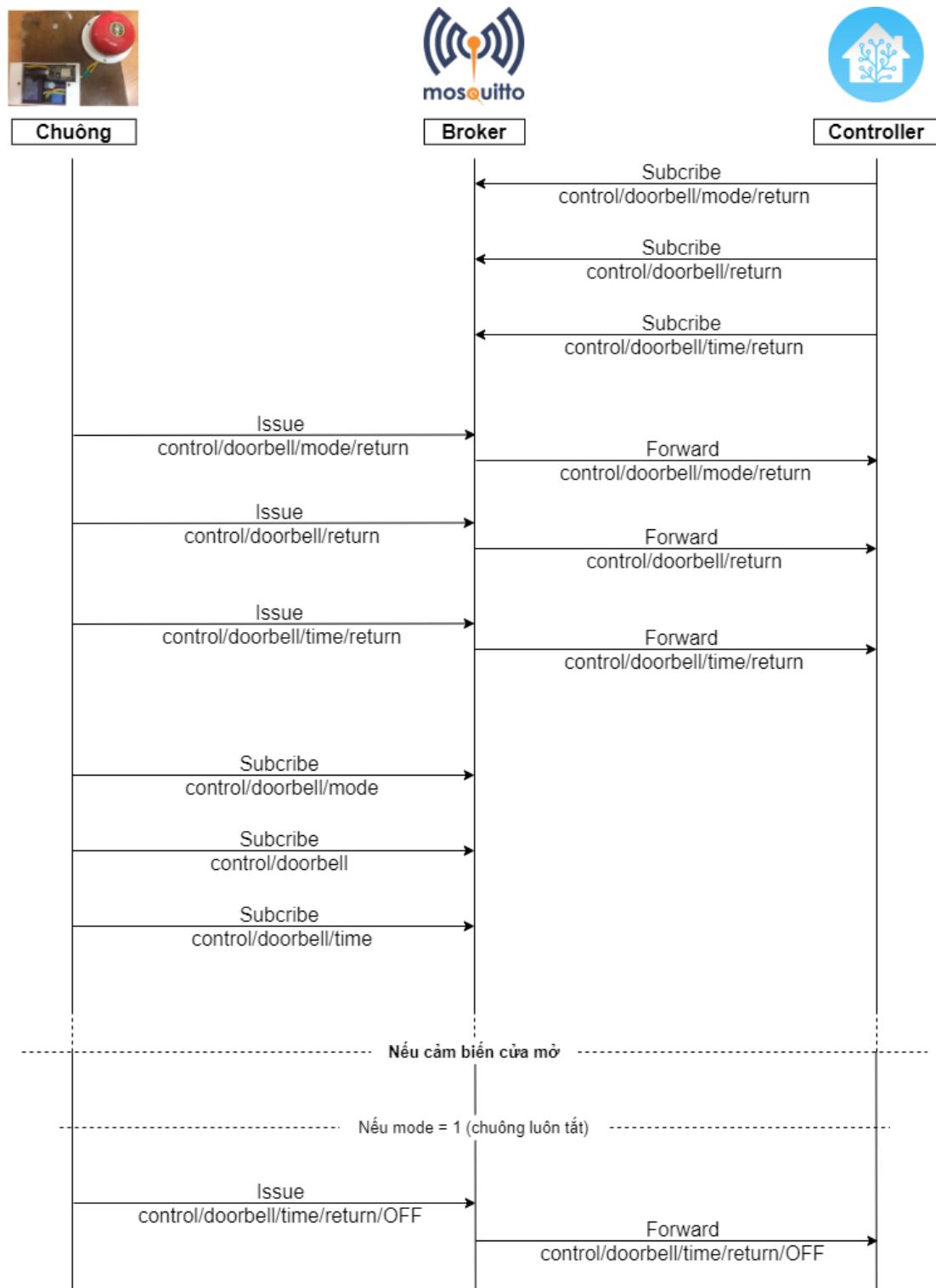
Hình 74: Sơ đồ đấu nối chuông báo động

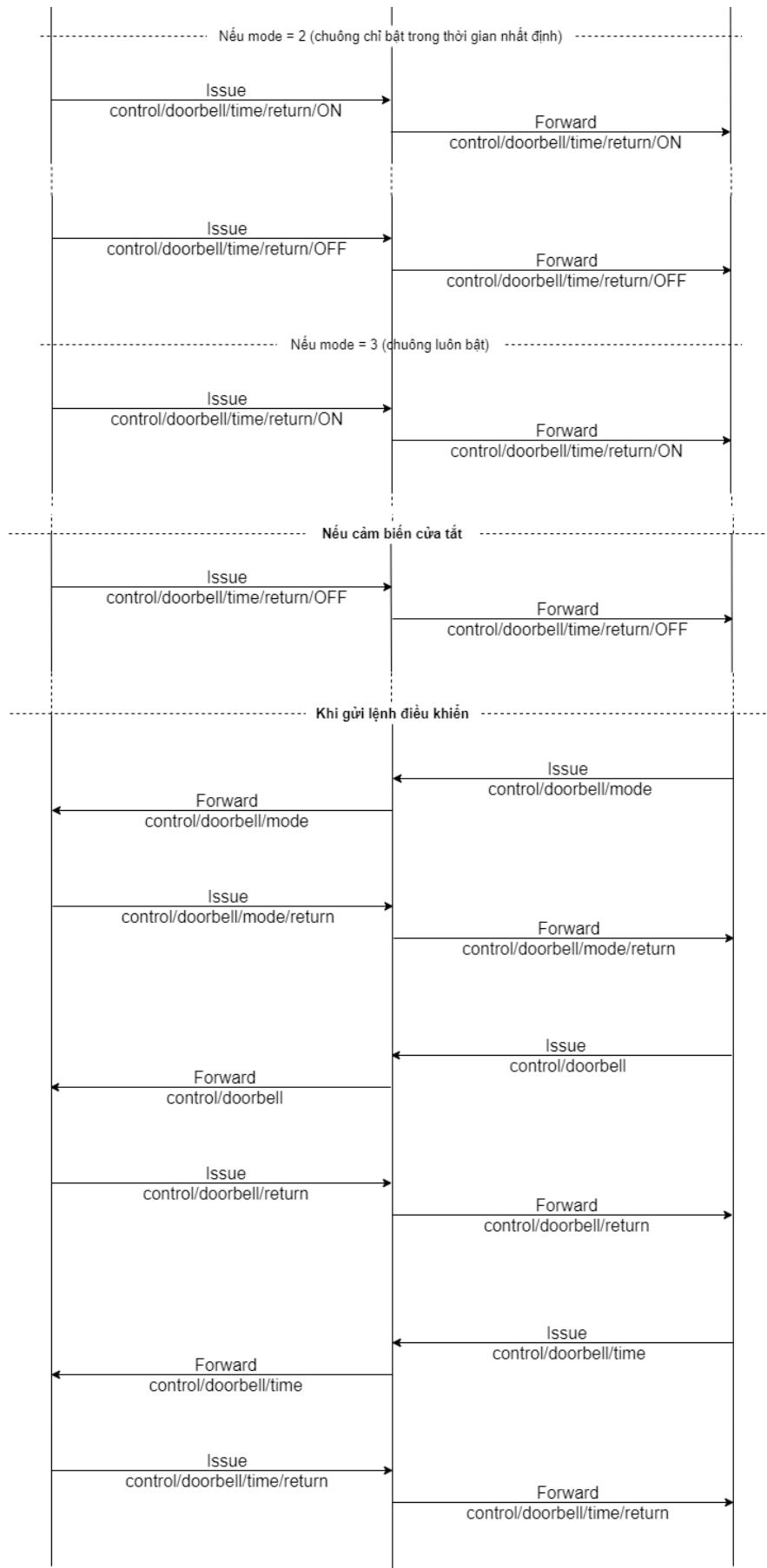


Hình 75: Đấu nối chuông báo động thực tế

8.2. Thiết lập điều khiển chuông báo động

Để điều khiển chuông báo động, ta sử dụng giao thức MQTT để gửi tin nhắn điều khiển thiết bị từ bộ điều khiển trung tâm. Ta có sơ đồ như sau:





Hình 76: Giao thức MQTT của chuông báo động

Từ sơ đồ trên, ta lập trình để điều khiển chuông báo động bằng ngôn ngữ C trên IDE Arduino như sau:

Code door_bell.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#define wifi_ssid "Manh"
#define wifi_password "manhvan1012"
#define mqtt_server "192.168.1.167"
#define mqtt_user "mymqtt"
#define mqtt_password "mymqtt"
const uint16_t mqtt_port = 1883;
WiFiClient espClient;
PubSubClient client(espClient);
#define mode_topic "control/doorbell/mode"
#define mode_topic_return "control/doorbell/mode/return"
#define control_topic "control/doorbell"
#define control_topic_return "control/doorbell/return"
#define time_run_topic "control/doorbell/time"
#define time_run_topic_return "control/doorbell/time/return"
#define RELAYPIN D7
unsigned long last = 0;
char message[100]; // lay gia tri tu mqtt server
int mode = 3;
int timerun = 10;
int check = 1;
int check2 = 0;
int state = 0;
int congtac = 0;
void setup() {
    Serial.begin(9600);
    setup_wifi(); //Connect to Wifi network
    client.setServer(mqtt_server,mqtt_port);
    client.setCallback(callback);
    pinMode(RELAYPIN,OUTPUT);
}

void loop() {
    // Kiểm tra kết nối
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    if(check == 1){
        timerun = timerun*1000;
    }
}
```

```

check=0;
}
if(check2 == 1){
last = millis();
check2 = 0;
}
if(state == 1)
{
if(mode == 1) // luon tat
{
if(congtac != 0)
{
digitalWrite(RELAYPIN,HIGH); //tat
congtac = 0;
client.publish(control_topic_return,String("OFF").c_str(), true);
}
}
if(mode == 3) // luon bat
{
if(congtac != 1)
{
digitalWrite(RELAYPIN,LOW); //bat
congtac = 1;
client.publish(control_topic_return,String("ON").c_str(), true);
}
}
if(mode == 2) //bat trong 1 khoang thoi gian nhat dinh
{
if((unsigned long)(millis()-last) < timerun)
{
if(congtac != 1)
{
digitalWrite(RELAYPIN,LOW); //bat
congtac = 1;
client.publish(control_topic_return,String("ON").c_str(), true);
}
}
else
{
if(congtac != 0)
{
digitalWrite(RELAYPIN,HIGH); //tat
congtac = 0;
client.publish(control_topic_return,String("OFF").c_str(), true);
}
}
}
}

```

```

    }
}

if(state == 0)
{
    if(congtac != 0)
    {
        digitalWrite(RELAYPIN,HIGH); //tat
        congtac = 0;
        client.publish(control_topic_return,String("OFF").c_str(), true);
    }
}
}

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(wifi_ssid);
    WiFi.begin(wifi_ssid,wifi_password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
    int i=0;
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (i = 0; i < length; i++) {
        message[i] = payload[i];
    }
    message[i] = '\0';
    String msgString = String(message);
    Serial.println("Message: " + msgString);
    Serial.println();
    if(strcmp(topic,mode_topic)==0)
    {
        mode = msgString.toInt(); // 1 2 3
        if(mode==1)
        {
            client.publish(mode_topic_return,"Luôn tắt", true);
        }
    }
}

```

```

    if(mode==3)
    {
        client.publish(mode_topic_return,"Bật liên tục", true);
    }
    if(mode==2)
    {
        client.publish(mode_topic_return,"Theo thời gian", true);
    }
}
if(strcmp(topic,time_run_topic)==0)
{
    timerun = msgString.toInt(); // s
    client.publish(time_run_topic_return,msgString.c_str(), true);
    check=1;
}
if(strcmp(topic,control_topic)==0)
{
    state = msgString.toInt(); //1 bat, 0 tat
    if(state == 1)
    {
        check2 = 1;
    }
}
void reconnect() {
    while (!client.connected()) {
        Serial.print("Connecting to MQTT broker ...");
        if (client.connect("ESP8266DoorBell",mqtt_user, mqtt_password)) {
            Serial.println("connected");
            // Can sub topic nao thi viet tiep duoi day:
            client.publish(mode_topic_return,String(mode).c_str(), true);
            client.publish(time_run_topic_return,String(timerun).c_str(), true);
            client.publish(control_topic_return,String("OFF").c_str(), true);
            digitalWrite(RELAYPIN,HIGH); //tat
            client.subscribe(mode_topic);
            client.subscribe(control_topic);
            client.subscribe(time_run_topic);
        } else {
            Serial.print("failed, error = ");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}

```

Về tổng quan, chương trình sẽ hoạt động như sau:

1. Khi bắt đầu, vi điều khiển sẽ kết nối đến Broker MQTT đồng thời khởi tạo chế độ mode là 3 (luôn bật cho đến khi cửa đóng), thời gian duy trì dùng cho mode 2 là 10 giây và trạng thái chuông ban đầu tắt, các thông tin đó được gửi cho Broker. Vi điều khiển subscribe các topic nhận lệnh sửa chế độ mode, thời gian duy trì, và điều khiển chuông từ Home Assistant gửi lại của cảm biến.
2. Khởi tạo một vòng lặp vô hạn nhằm duy trì kết nối tới Broker đồng thời lắng nghe các bản tin điều khiển từ các topic đã được subscribe trước đó.
3. Khi nhận được bản tin gửi đến, tùy theo lệnh điều khiển là gì mà nó thực thi công việc:
 - Nếu topic là mode:
 - Mode 3: Khi cửa mở, chuông sẽ bật cho đến khi đóng cửa.
 - Mode 2: Khi cửa mở, chuông chỉ bật trong một khoảng thời gian rồi tắt.
 - Mode 1: Chuông luôn tắt dù có mở cửa hay không.
 - Nếu topic là time_run: cài đặt khoảng thời gian bật chuông dùng cho mode 2.
 - Nếu topic là control: Báo hiệu cửa mở hay đóng.
4. Nếu xảy ra sự mất kết nối tới Broker thiết bị sẽ tự động kết nối lại.

8.3. Thiết lập chuông báo động trên Home Assistant

Đầu tiên ta kết nối chuông đến bộ điều khiển Pi trung tâm. Ta thêm code trong file configuration.yaml:

```
input_select: !include input_select.yaml
```

Ta thêm code trong file sensor.yaml:

```
- platform: mqtt
  name: "Bell Mode"
  expire_after: 0
  state_topic: "control/doorbell/mode/return"
- platform: mqtt
```

```
name: "Bell Time Run"
expire_after: 0
state_topic: "control/doorbell/time/return"
unit_of_measurement: "s"
icon: mdi:timer
- platform: mqtt
  name: "Bell State"
  expire_after: 0
  state_topic: "control/doorbell/return"
```

Tiếp đó ta tạo file input_select.yaml:

```
touch /usr/share/hassio/homeassistant/input_select.yaml
```

Thêm vào file input_select.yaml:

```
list_mode:
  name: List Mode Bell
  options:
    - "Luôn Tắt"
    - "Bật trong khoảng thời gian"
    - "Bật liên tục"
  icon: mdi:clipboard-list-outline
```

Thêm vào file input_text.yaml:

```
time_run_bell:
  name: Time Run Bell Door (s)
```

Thêm vào file script.yaml:

```
'1583220554014':
  alias: Set time run door bell
  sequence:
    - data:
        payload_template: '{{ states.input_text.time_run_bell.state }}'
        retain: true
        topic: control/doorbell/time
        service: mqtt.publish
'1583220846393':
  alias: Set mode bell
  sequence:
    - data:
```

```

    payload_template: '{% if is_state(''input_select.list_mode'', ''Luôn Tắt'')
    %} 1 {% endif %} {% if is_state(''input_select.list_mode'', ''Bật trong
    khoảng
    thời gian'') %} 2 {% endif %} {% if is_state(''input_select.list_mode'', ''Bật
    liên tục'') %} 3 {% endif %}
    retain: true
    topic: control/doorbell/mode
    service: mqtt.publish

```

Để tự động hóa bật chuông khi cửa mở, ta thêm vào file automation.yaml:

```

- id: '1583223412808'
  alias: Bat_Bell_Theo_Door_Sensor
  description: ''
  trigger:
    - entity_id: sensor.sensor_door
      from: Đóng cửa
      platform: state
      to: Mở cửa
  condition: []
  action:
    - data:
        payload_template: 1
        retain: true
        topic: control/doorbell
        service: mqtt.publish
- id: '1583223412919'
  alias: Tat_Bell_Theo_Door_Sensor
  description: ''
  trigger:
    - entity_id: sensor.sensor_door
      from: Mở cửa
      platform: state
      to: Đóng cửa
  condition: []
  action:
    - data:
        payload_template: 0
        retain: true
        topic: control/doorbell
        service: mqtt.publish

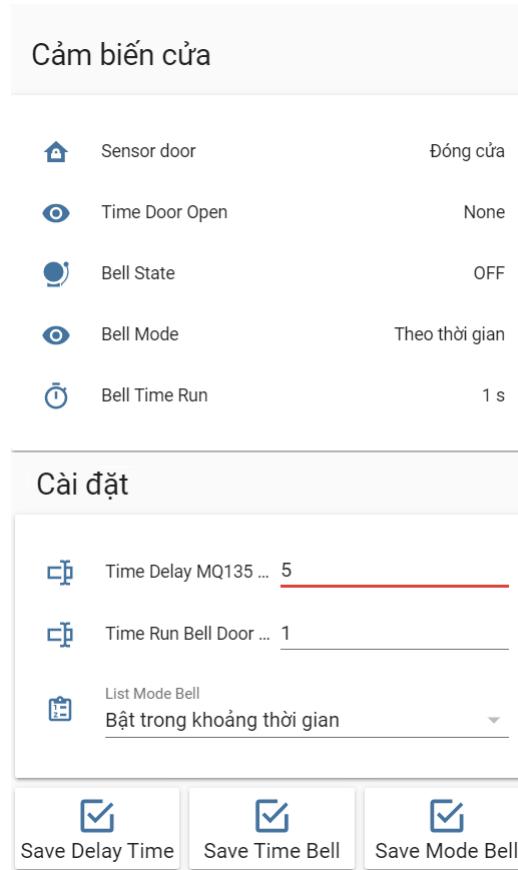
```

Trong file giao diện Lovelace ta thêm và sửa config sau:

```
{  
  "cards": [  
    {  
      "entities": [  
        {  
          "entity": "input_text.time_delay_mq135_dht11"  
        },  
        {  
          "entity": "input_text.time_run_bell"  
        },  
        {  
          "entity": "input_select.list_mode"  
        }  
      ],  
      "show_header_toggle": false,  
      "type": "entities"  
    },  
    {  
      "cards": [  
        {  
          "entity": "script.1582191205172",  
          "hold_action": {  
            "action": "more-info"  
          },  
          "icon": "mdi:checkbox-marked-outline",  
          "icon_height": "35px",  
          "name": "Save Delay Time",  
          "show_icon": true,  
          "show_name": true,  
          "tap_action": {  
            "action": "toggle"  
          },  
          "type": "entity-button"  
        },  
        {  
          "entity": "script.1583220554014",  
          "hold_action": {  
            "action": "more-info"  
          },  
          "icon": "mdi:checkbox-marked-outline",  
          "icon_height": "35px",  
          "name": "Save Time Bell",  
          "show_icon": true,  
          "show_name": true,  
          "tap_action": {  
            "action": "toggle"  
          },  
          "type": "entity-button"  
        },  
        {  
          "entity": "script.1583220846393",  
          "hold_action": {  
            "action": "more-info"  
          },  
          "icon": "mdi:checkbox-marked-outline",  
          "icon_height": "35px",  
          "name": "Run Bell",  
          "show_icon": true,  
          "show_name": true,  
          "tap_action": {  
            "action": "toggle"  
          },  
          "type": "entity-button"  
        }  
      ]  
    }  
  ]  
}
```

```
        "action": "more-info"
    },
    "icon": "mdi:checkbox-marked-outline",
    "icon_height": "35px",
    "name": "Save Mode Bell",
    "show_icon": true,
    "show_name": true,
    "tap_action": {
        "action": "toggle"
    },
    "type": "entity-button"
}
],
"type": "horizontal-stack"
}
],
"title": "Cài đặt",
"type": "vertical-stack"
},
{
"cards": [
{
"entities": [
{
"entity": "sensor.sensor_door"
},
{
"entity": "sensor.time_open_door_sensor"
},
{
"entity": "sensor.bell_state"
},
{
"entity": "sensor.bell_mode"
},
{
"entity": "sensor.bell_time_run"
}
],
"show_header_toggle": false,
"type": "entities"
}
]
},
"title": "Cảm biến cửa",
"type": "vertical-stack"
}
```

Thành quả đạt được:



Hình 77: Bảng cảm biến cửa và chuông báo động

Qua bảng trên, ngoài những chức năng được giới thiệu ở các phần trước, ta có trạng thái chuông hoạt động hay không, chế độ của chuông và khoảng thời gian bật chuông (dành cho mode 2).

Bảng cài đặt có thêm chức năng điều khiển chuông, gồm thời gian ta muốn chuông kêu, và các chế độ cho chuông được chọn theo danh sách đã được giới thiệu phần trên.

9. Tự động hóa các thiết bị

Tự động hóa các thiết bị là chức năng không thể thiếu trong nhà thông minh, nó giúp tự động hóa các chức năng thường nhật trong ngôi nhà và khiến mọi thứ trở nên dễ dàng hơn đem lại sự tiện nghi cho người dùng, đồng thời nó đem lại sự an ninh nhờ sự tự động của các thiết bị như chuông báo động, hay cảm biến báo cháy, các cảnh báo đèn điện thoại, ...

Với các sản phẩm Hộp điều khiển thiết bị điện và Hộp điều khiển hồng ngoại thì việc tự động hóa các thiết bị trong nhà trở nên dễ dàng.

Việc tự động hóa các thiết bị là vô cùng đa dạng, do trong dự án này do sự hạn chế về mặt thiết bị nên các thiết bị bật / tắt hoặc điều khiển bằng hồng ngoại được ví dụ trong các kịch bản tự động hóa dưới đây sẽ được thay thế bằng đèn cửa đã được giới thiệu phần trên, về mặt bản chất các thiết bị điện tử bật / tắt hoặc điều khiển hồng ngoại đều hoạt động có phần tương tự như bật / tắt đèn, nên ta có thể áp dụng hoàn toàn tương tự.

Dưới đây sẽ là một số kịch bản được nêu trong dự án này.

9.1. Tự động bật đèn hồng ngoại của Camera

Tự động hóa đèn hồng ngoại của Camera đã được đề cập ở trang 47.

9.2. Bật chuông khi cửa mở

Tự động hóa bật chuông khi cảm biến cửa mở đã được đề cập ở trang 124.

9.3. Tự động bật còi báo cháy và thông báo khi có hỏa hoạn

Khi nhiệt độ lớn hơn 45 độ C, tự động bật còi báo cháy và thông báo đến các thiết bị trong gia đình có quyền truy cập trong Home Assistant. (còi báo cháy sẽ được thay bằng đèn).

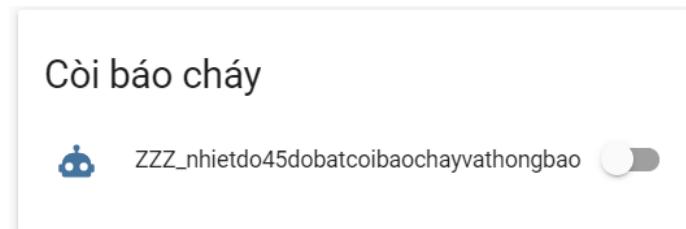
Trong file automations.yaml ta thêm:

```
- id: '1583239761643'
  alias: ZZZ_nhiетdo45dobatcoibaochayvathongbao
  description: Nếu nhiệt độ > 45 độ, bật còi báo cháy và thông báo (còi thay bằng đèn)
  trigger:
    - platform: template
      value_template: '{{ states.sensor.temperature.state | float > 45 }}'
  condition:
    - condition: state
      entity_id: switch.light_door
      state: 'off'
  action:
    - data: {}
      entity_id: switch.light_door
      service: switch.turn_on
    - data:
        data:
          tag: Thong-bao-chay
      data_template:
        message: 'Có cháy, nhiệt độ là : {{ states(''sensor.temperature'') }}.'
      service: notify.thong_bao
```

Để thuận tiện cho việc bật tắt kịch bản tự động, ta thêm trong giao diện Lovelace:

```
{
  "entities": [
    {
      "entity": "automation.zzz_nhiệtdo45dobatcoibaochay"
    }
  ],
  "show_header_toggle": false,
  "title": "Còi báo cháy",
  "type": "entities"
}
```

Thành quả:



Hình 78: Bảng điều khiển kịch bản còi báo cháy

9.4. Bật đèn khi cửa mở

Tự động bật đèn trong nhà khi cửa mở và tắt đèn khi cửa đóng sau 10 giây.

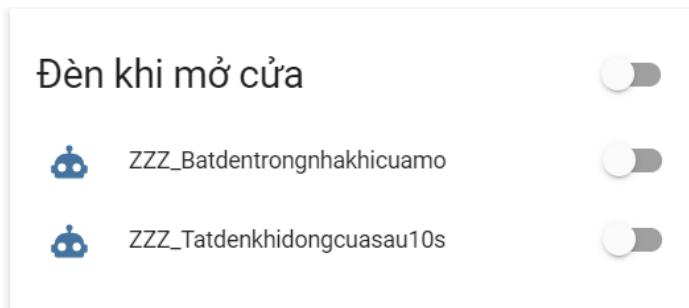
Trong file automations.yaml ta thêm:

```
- id: '1583241688840'
  alias: ZZZ_Batdentrongnhakhicuamo
  description: Bật đèn trong nhà khi cửa mở
  trigger:
    - entity_id: sensor.sensor_door
      from: Đóng cửa
      platform: state
      to: Mở cửa
    condition: []
  action:
    - data: {}
      entity_id: switch.light_door
      service: switch.turn_on
- id: '1583241852036'
  alias: ZZZ_Tatdenkhidongcuasau10s
  description: tắt đèn khi cửa đóng sau 10s
  trigger:
    - entity_id: sensor.sensor_door
      from: Mở cửa
      platform: state
      to: Đóng cửa
    condition: []
  action:
    - delay: 00:00:10
    - data: {}
      entity_id: switch.light_door
      service: switch.turn_off
```

Giao diện Lovelace:

```
{  
    "entities": [  
        {  
            "entity": "automation.zzz_batdentrongnhakhicuamo"  
        },  
        {  
            "entity": "automation.zzz_tatdenkhidongcuasau10s"  
        }  
  
    ],  
    "title": "Đèn khi mở cửa",  
    "type": "entities"  
}
```

Thành quả:



Hình 79: Bảng điều khiển kích bản đèn khi mở cửa

9.5. Tưới cây tự động

Từ 6 giờ sáng đến 6 giờ tối hàng ngày, cứ cách 2 giờ lại bật tưới cây trong 5 phút rồi tắt. (Bật tắt tưới cây thay bằng đèn cửa).

Trong file automations.yaml ta thêm:

```
- id: '1583242504065'  
  alias: ZZZ_6htoi6hsang_cach2htuoicaytrong5proitat  
  description: Từ 6h sáng đến 6h tối, cứ cách 2h lại bật tưới cây trong 5 phút rồi  
    tắt  
  trigger:  
    - hours: /2
```

```
minutes: '0'
platform: time_pattern
seconds: '0'

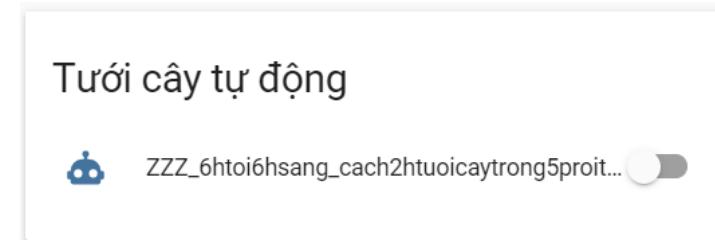
condition:
- after: 06:00:00
  before: '18:00:00'
  condition: time

action:
- data: {}
  entity_id: switch.light_door
  service: switch.turn_on
- delay: 00:05:00
- data: {}
  entity_id: switch.light_door
  service: switch.turn_off
```

Giao diện Lovelace:

```
{
  "entities": [
    {
      "entity": "automation.zzz_6htoi6hsang_cach2htuoicaytrong5proitat"
    }
  ],
  "show_header_toggle": false,
  "title": "Tưới cây tự động",
  "type": "entities"
}
```

Thành quả:



Hình 80: Bảng điều khiển kịch bản tưới cây tự động

9.6. Bật máy lọc không khí tự động

Nếu chất lượng không khí phòng lớn hơn 500ppm và máy lọc không khí chưa được bật thì bật máy lọc không khí, tắt máy lọc không khí khi máy lọc không khí đang bật và chất lượng không khí phòng nhỏ hơn 450ppm (Tắt bật máy lọc thay bằng đèn cửa).

Trong file automations.yaml ta thêm:

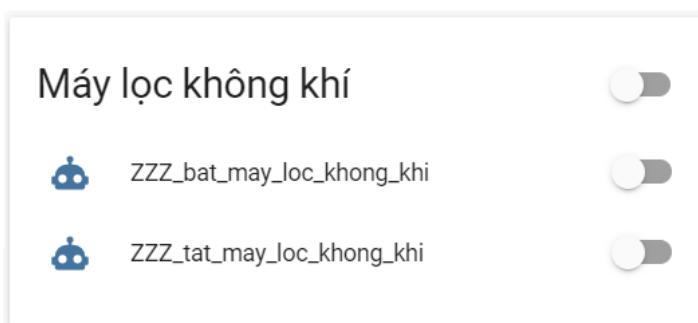
```
- id: '1583242985649'
  alias: ZZZ_bat_may_loc_khong_khi
  description: Nếu chất lượng không khí phòng > 500ppm và máy lọc không khí chưa được bật thì bật máy lọc không khí
  trigger:
    - hours: '*'
      minutes: '*'
      platform: time_pattern
      seconds: '*'
  condition:
    - condition: state
      entity_id: switch.light_door
      state: 'off'
    - condition: template
      value_template: '{{ states.sensor.airquality.state | float > 500 }}'
  action:
    - data: {}
      entity_id: switch.light_door
      service: switch.turn_on
- id: '1583243174588'
  alias: ZZZ_tat_may_loc_khong_khi
  description: tắt máy lọc không khí khi máy lọc không khí đang bật và chất lượng không khí phòng < 450ppm
  trigger:
    - hours: '*'
      minutes: '*'
      platform: time_pattern
      seconds: '*'
  condition:
    - condition: state
      entity_id: switch.light_door
      state: 'on'
    - condition: template
      value_template: '{{ states.sensor.airquality.state | float < 450 }}'
  action:
    - data: {}
```

```
entity_id: switch.light_door  
service: switch.turn_off
```

Giao diện Lovelace:

```
{  
    "entities": [  
        {  
            "entity": "automation.zzz_bat_may_loc_khong_khi"  
        },  
        {  
            "entity": "automation.zzz_tat_may_loc_khong_khi"  
        }  
  
    ],  
    "title": "Máy lọc không khí",  
    "type": "entities"  
}
```

Thành quả:



Hình 81: Bảng điều khiển kịch bản máy lọc không khí

9.7. Đóng, mở cửa sổ tự động

Nếu trời bắt đầu sáng (sunrise) sau 2 giờ và chất lượng không khí nhỏ hơn 450 ppm thì mở cửa sổ, khi trời bắt đầu tối (sunset) hoặc chất lượng không khí lớn hơn 600ppm thì đóng cửa sổ (đóng, mở cửa sổ thay bằng đèn cửa).

Trong file automations.yaml ta thêm:

```
- id: '1583243736267'
  alias: ZZZ_mo_cua_so
  description: Nếu trời bắt đầu sáng (sunrise) sau 2h và chất lượng không khí < 450
    ppm thì mở cửa sổ
  trigger:
    - event: sunrise
      offset: 02:00:00
      platform: sun
  condition:
    - condition: template
      value_template: '{{ states.sensor.airquality.state | float < 450 }}'
    - condition: state
      entity_id: switch.light_door
      state: 'off'
  action:
    - data: {}
      entity_id: switch.light_door
      service: switch.turn_on
- id: '1583243823625'
  alias: ZZZ_dong_cua_so
  description: 'trời bắt đầu tối (sunset) hoặc chất lượng không khí >600ppm thì đóng
    cửa sổ '
  trigger:
    - event: sunset
      platform: sun
    - platform: template
      value_template: '{{ states.sensor.airquality.state | float > 600 }}'
  condition:
    - condition: state
      entity_id: switch.light_door
      state: 'on'
  action:
    - data: {}
      entity_id: switch.light_door
      service: switch.turn_off
```

Giao diện Lovelace:

```
{  
  "entities": [  
    {  
      "entity": "automation.zzz_mo_cua_so"  
    },  
    {  
      "entity": "automation.zzz_dong_cua_so"  
    }  
  
  ],  
  "title": "Cửa sổ",  
  "type": "entities"  
}
```

Thành quả:



Hình 82: Bảng điều khiển kịch bản đóng mở cửa sổ

9.8. Bật tắt điều hòa tự động

Trong thời gian từ 0 giờ đến 6 giờ sáng điều hòa hoạt động nếu cảm biến cửa phòng đóng và nhiệt độ phòng lớn hơn 30 độ, tắt nếu hết thời gian hoặc nhiệt độ phòng nhỏ hơn 25 độ hoặc cảm biến cửa mở (Tắt bật điều hòa thay bằng đèn cửa)

Trong file automations.yaml ta thêm:

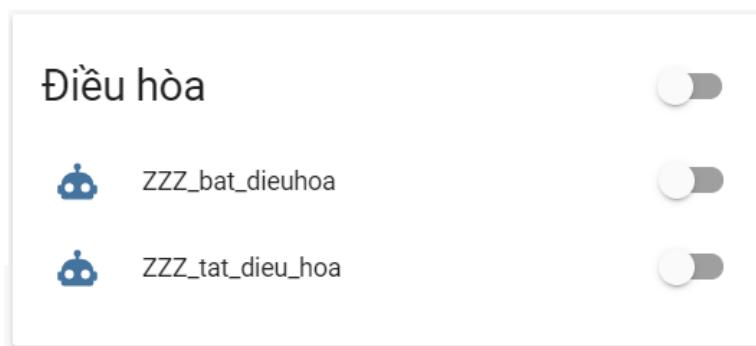
```
- id: '1583244382529'
  alias: ZZZ_bat_dieuhua
  description: Trong thời gian từ 0h đến 6h sáng điều hòa hoạt động nếu cảm biến cửa
    phòng đóng và nhiệt độ phòng > 30 độ
  trigger:
    - hours: '*'
      minutes: '*'
      platform: time_pattern
      seconds: '*'
  condition:
    - after: 00:00:00
      before: 06:00:00
      condition: time
    - condition: state
      entity_id: sensor.sensor_door
      state: Đóng cửa
    - condition: state
      entity_id: switch.light_door
      state: 'off'
    - condition: template
      value_template: '{{ states.sensor.temperature.state | float > 30 }}'
  action:
    - data: {}
      entity_id: switch.light_door
      service: switch.turn_on
- id: '1583244878479'
  alias: ZZZ_tat_dieu_hoa
  description: tắt nếu hết thời gian hoặc nhiệt độ phòng < 25 độ hoặc cảm biến cửa
    mở
  trigger:
    - hours: '*'
      minutes: '*'
      platform: time_pattern
      seconds: '*'
  condition:
    - condition: and
      conditions:
        - condition: state
          entity_id: switch.light_door
          state: 'on'
        - condition: or
          conditions:
            - after: 06:00:00
```

```
before: '23:59:59'
condition: time
- condition: template
  value_template: '{{ states.sensor.temperature.state | float < 25 }}'
- condition: state
  entity_id: sensor.sensor_door
  state: Mở cửa
action:
- data: {}
  entity_id: switch.light_door
  service: switch.turn_off
```

Giao diện Lovelace:

```
{
  "entities": [
    {
      "entity": "automation.zzz_dieuhhoa"
    },
    {
      "entity": "automation.zzz_tat_dieu_hoa"
    }
  ],
  "title": "Điều hòa",
  "type": "entities"
}
```

Thành quả:



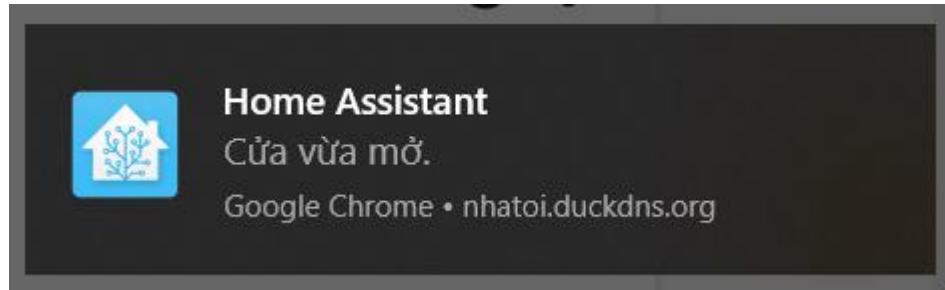
Hình 83: Bảng điều khiển kịch bản bật tắt điều hòa

9.9. Cảnh báo khi cửa mở

Trong file automations.yaml ta thêm:

```
- id: '1583831000237'
  alias: Canh_bao_khi_cua_vua_mo
  description: ''
  trigger:
    - entity_id: sensor.sensor_door
      from: Đóng cửa
      platform: state
      to: Mở cửa
  condition: []
  action:
    - data:
        data:
          tag: notification-cua-vua-dc-mo
        data_template:
          message: Cửa vừa mở.
        service: notify.thong_bao
```

Thành quả khi chạy:



Hình 84: Thông báo khi cửa mở

9.10. Cảnh báo khi cửa mở quá 5 phút

Trong file automations.yaml ta thêm:

```
- id: '1583823415456'
  alias: Thong_bao_khi_cua_mo_qua_5p
  description: ''
  trigger:
    - platform: template
      value_template: '{{states.sensor.time_open_door_sensor.state == "5 minutes"}}'
  condition: []
  action:
    - data:
        data:
          tag: mo-cua-quá-lau
        data_template:
          message: Cửa đã mở được {{ states('sensor.time_open_door_sensor') }}.
        service: notify.thong_bao
```

Thành quả khi chạy:



Hình 85: Thông báo khi cửa mở quá 7 phút

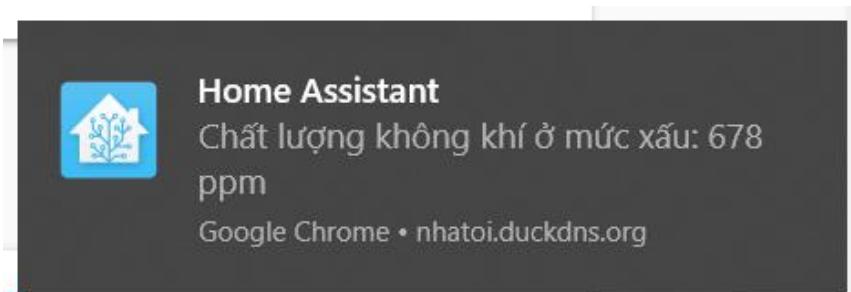
9.11. Cảnh báo khi chất lượng không khí xấu

Khi chất lượng không khí vượt qua mức 500 ppm, sẽ có cảnh báo gửi đến các thiết bị.

Trong file automations.yaml ta thêm:

```
- id: '1583831246917'
  alias: ZZZ_thongbaokhichatluongkhongkhihon500
  description: ''
  trigger:
    - platform: template
      value_template: '{{ states.sensor.airquality.state | float > 500 }}'
  condition: []
  action:
    - data:
        data:
          tag: notification-canhbao-clkk
        data_template:
          message: 'Chất lượng không khí ở mức xấu: {{ states("sensor.sensor") }} ppm'
        service: notify.thong_bao
```

Thành quả khi chạy:



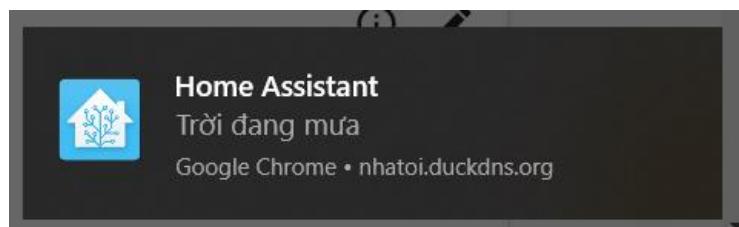
Hình 86: Thông báo khi chất lượng không khí xấu

9.12. Cảnh báo khi trời sắp mưa

Trong file automations.yaml ta thêm:

```
- id: '1583826219348'
  alias: ZZZ_Thongbaokhitroimua
  description: Thông báo khi trời mưa (thay cảm biến mưa bằng đèn)
  trigger:
    - entity_id: switch.light_door
      from: 'off'
      platform: state
      to: 'on'
  condition: []
  action:
    - data:
        data:
          tag: notification-troi-mua
        data_template:
          message: Trời đang mưa
        service: notify.thong_bao
```

Thành quả khi chạy:



Hình 87: Thông báo khi trời mưa

10. Phân quyền và bảo mật tài khoản

Phân quyền là một tính năng cần thiết cho hệ thống nhà thông minh, giúp cho ta quản lý được những thiết bị mà người dùng nào có thể truy cập. Cùng với nó là bảo mật tài khoản, giúp hạn chế phần nào kẻ xấu truy cập vào được hệ thống nhà thông minh ảnh hưởng đến hoạt động của các thiết bị trong gia đình và đặc biệt là quyền riêng tư của người dùng.

10.1. Phân quyền tài khoản truy cập

Home Assistant đã có hệ thống phân quyền người dùng kể từ Home Assistant bản 0.82. Quyền được gắn vào các nhóm, một người dùng có thể là một phần của nhiều nhóm. Ta có ba nhóm mặc định của hệ thống: "admin", "users" và "read-only". Cả ba đều có quyền truy cập vào tất cả các thực thể, nhưng "read-only" không thể điều khiển bất kỳ thực thể nào trong số chúng. Chỉ "user" và "admin" mới có thể điều khiển được. Riêng nhóm "admin" có thể cài đặt được hệ thống.

Vấn đề đặt ra việc tạo ra các nhóm tùy chỉnh ngoài ba nhóm có sẵn trên, chỉ cho phép người dùng truy cập vào một số thực thể nhất định. Ở thời điểm làm đồ án này, tính năng này của Home Assistant chưa được hoàn thiện và chưa được hỗ trợ, nhưng vẫn có thể làm được.

Để thêm một nhóm quyền mới, ta nhập lệnh Terminal:

```
sudo nano /usr/share/hassio/homeassistant/.storage/auth
```

Tìm khóa "groups" và thêm một nhóm mới:

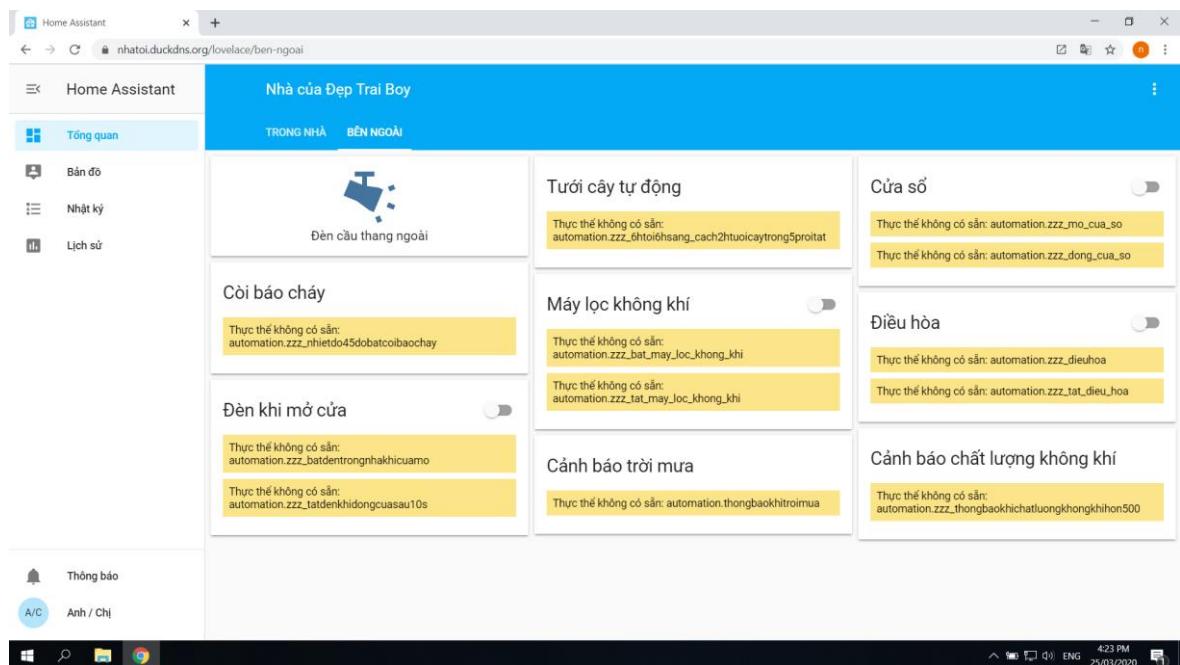
```
{
  "id": "limit-for-anhchi",
  "name": "Group cho Anh Chi",
  "policy": {
    "entities": {
      "entity_ids": {
        "switch.light_door": true
      }
    }
  }
}
```

Điều này sẽ tạo ra một nhóm chỉ cho phép xem và điều khiển duy nhất thiết bị đèn cửa.

Bây giờ tìm khóa "users", sửa "group_ids" của người dùng muốn phân quyền:

```
{  
    "group_ids": [  
        "limit-for-anhchi"  
    ],  
    "id": "f70ee1eff26d48518f9a25475febaf15",  
    "is_active": true,  
    "is_owner": false,  
    "name": "Anh Chị",  
    "system_generated": false  
},
```

Cuối cùng ta có thành quả sau:



Hình 88: Giao diện Home Assistant của tài khoản được phân quyền

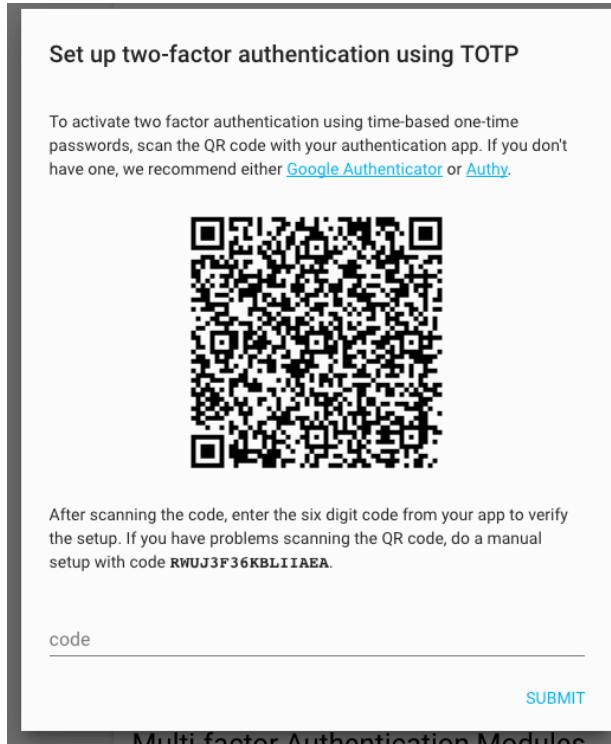
10.2. Bảo mật tài khoản

Để kích hoạt bảo mật hai lớp, ta thêm vào file configuration.yaml:

```
homeassistant:  
  auth_mfa_modules:  
    - type: totp
```

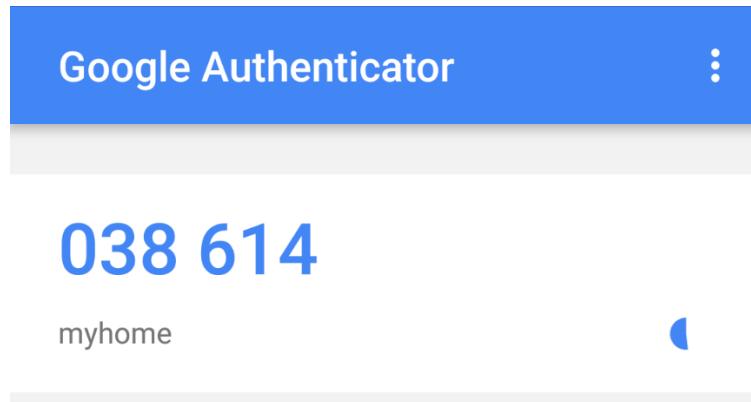
Sau đó ta truy cập vào Profile cửa tài khoản và chọn “Multi-factor Authentication Modules”

Chọn *Enable* và khóa tự động sẽ tự tạo ra.



Hình 89: Khóa tự động được tạo ra

Sử dụng ứng dụng Google Authenticator trên điện thoại và quét mã QR, khi đó phần mềm sẽ bắt đầu tạo mã sáu chữ số khác nhau cứ sau ba mươi giây, nhập một trong số này vào vùng code rồi nhấn SUBMIT. Từ đây Home Assistant và ứng dụng điện thoại sẽ được đồng bộ hóa và có thể sử dụng mã được hiển thị trong ứng dụng để xác nhận tài khoản ở mỗi lần đăng nhập.



Hình 90: Mã xác thực được tạo ra



Home Assistant

Bạn sắp cấp quyền cho <https://nhatoi.duckdns.org/> truy cập vào Home Assistant của bạn.

Đăng nhập bằng **Home Assistant Local**.

Mở **Time-based One Time Password** trên thiết bị của bạn để xem mã xác thực hai lớp và xác minh danh tính của bạn:

Mã xác thực hai lớp

NEXT

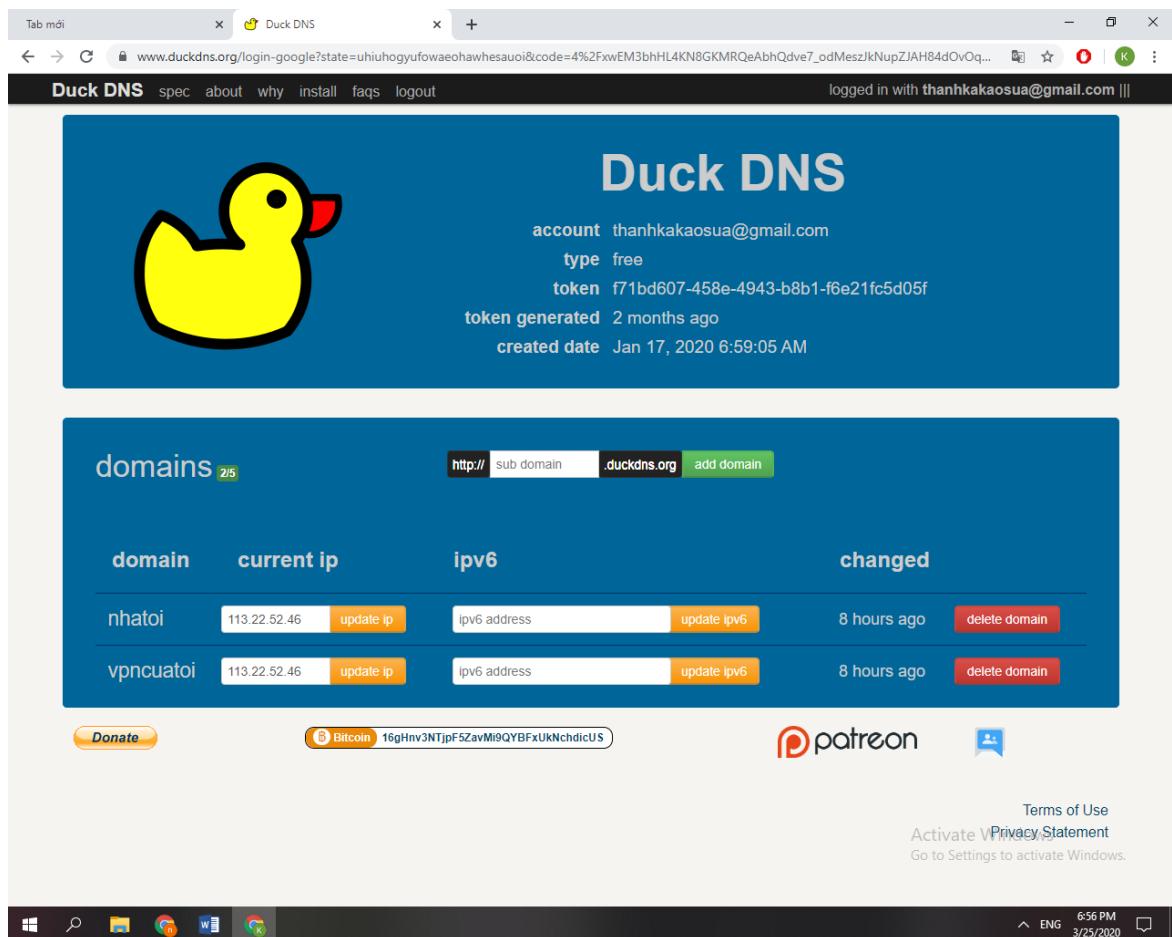
Hình 91: Yêu cầu xác thực hai lớp khi đăng nhập

CHƯƠNG III: CÁU HÌNH HỆ THỐNG CHO PHÉP TRUY CẬP TỪ BÊN NGOÀI INTERNET

1. Cấu hình Hass để truy cập từ bên ngoài Internet

Việc truy cập được Home Assistant từ bên ngoài Internet giúp ta có thể điều khiển và kiểm soát ngôi nhà thông minh ở bất cứ đâu. Đồng thời việc tạo theo cách dưới đây giúp giảm nguy cơ giả danh, đảm bảo an toàn cho người dùng do ta sử dụng giao thức HTTPS tích hợp chứng chỉ SSL.

Để bắt đầu cấu hình ta vào trang <https://duckdns.org/> để đăng ký một tài khoản. Sau khi đăng ký, DuckDNS sẽ cấp cho ta 5 domain miễn phí, ta sẽ nhập tên domain là: “*nhatoi*”, rồi bấm add domain.



Hình 92: Website DuckDNS

Đầu tiên chúng ta truy cập vào HASS. Ở thanh Menu, chúng ta chọn Hass.io, nhấn Add-on Store. Tìm Add-on có tên là DuckDNS rồi nhấn Install.

Sau khi tiến trình cài đặt thành công, thì chúng ta nhấn vào nút Start để khởi động DuckDNS.

Tiếp đó trong phần Config, ta nhập:

```
{  
    "lets_encrypt": {  
        "accept_terms": true,  
        "certfile": "fullchain.pem",  
        "keyfile": "privkey.pem"  
    },  
    "token": "f71bd607-458e-4943-b8b1-f6e21fc5d05f",  
    "domains": [  
        "nhatoi.duckdns.org"  
    ],  
    "seconds": 300  
}
```

Sau khi nhập xong ta nhấn Save.

Tiếp đến ta cần cấp quyền cho SSL ở hai file *fullchain.pem* và *privkey.pem*. Ở Terminal, ta nhập lệnh:

```
chmod -R 775 /ssl
```

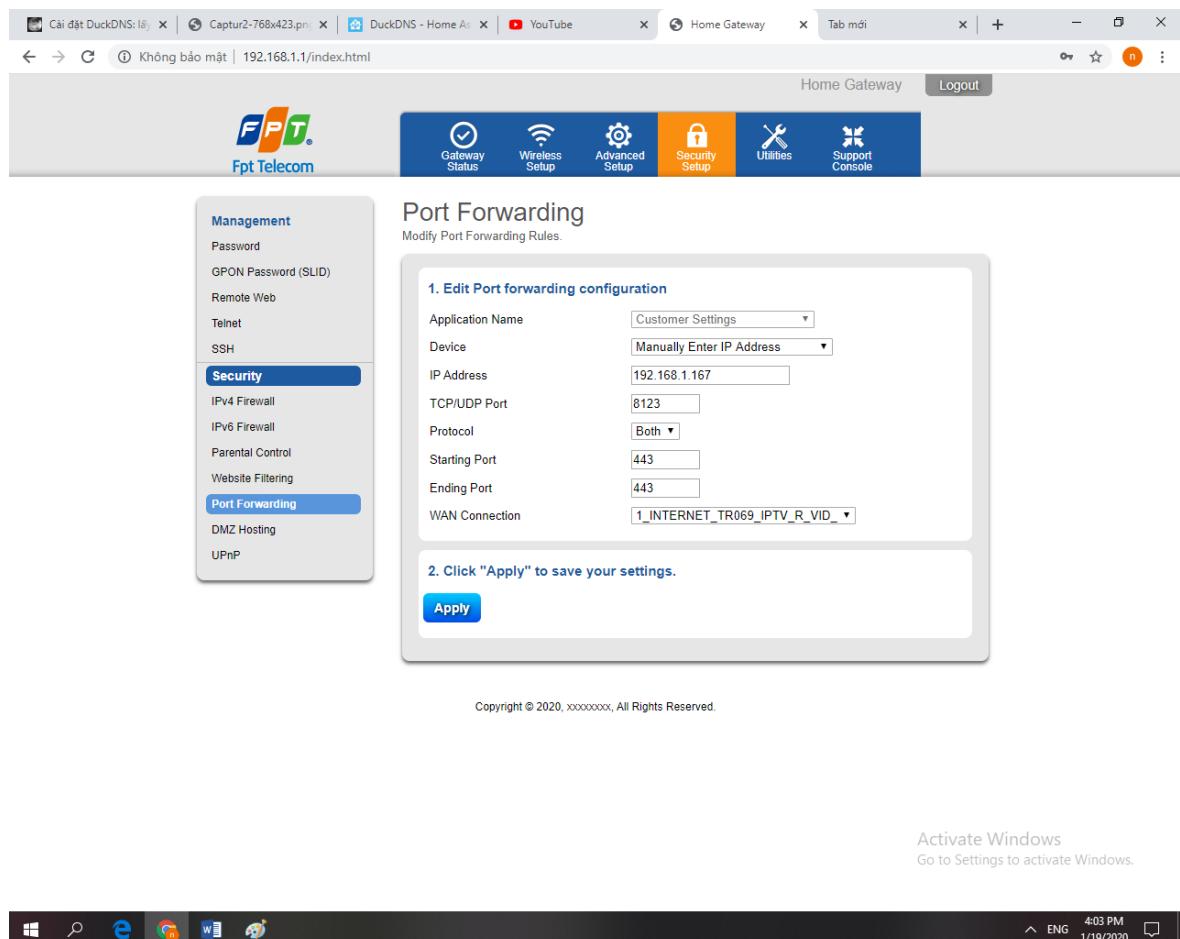
Sau đó, ta vào file configuration.yaml thêm dòng code sau:

```
http:  
  base_url: nhatoi.duckdns.org:8123  
  ssl_certificate: /ssl/fullchain.pem  
  ssl_key: /ssl/privkey.pem
```

Ta lưu lại và restart Home Assistant.

Cuối cùng ta cần mở Port để có thể truy cập từ bên ngoài. Ta vào trang quản trị của modem: 192.168.1.1.

Sau khi đăng nhập xong ta vào mục *Port Forwarding*. Ta cần mở cổng 443 vào cổng 8123, nên các bạn nhập lần lượt *Starting Port* là 443 và *Ending Port* là 8123. IP Address là IP của Pi. Chỗ Protocol là Both. Cuối cùng ta bấm Apply.



Hình 93: Port Forwarding

Từ bây giờ ta có thể vào Home Assistant từ bất cứ đâu qua tên miền :

<https://nhatoi.duckdns.org>

2. Điều khiển bằng giọng nói với Google Assistant

Việc tích hợp Google Assistant cho phép ta điều khiển thiết bị thông qua giọng nói ở bất cứ nơi đâu chỉ bằng thiết bị di động hoặc máy tính bảng.

Đầu tiên ta tạo dự án trên <https://console.actions.google.com/>

1. Chọn *Add/Import project* và đặt tên là “hasspi”
2. Chọn thẻ *Home control*, và chọn *Smart Home*
3. Chọn *Build your Action -> Add Action(s)*, nhập https://nhatoi.duckdns.org/api/google_assistant
4. Nhấn Save và chọn *Overview*.

Tiếp đến dưới mục *Quick Setup*, chọn *Account linking*.

1. Ta chọn *No, I only want to allow account creation on my website* rồi nhấn Next.
2. Ở phần *Linking type* ta chọn *Oauth* và *Authorization Code* rồi nhấn Next.
3. Ở *Client id* : <https://oauth-redirect.googleusercontent.com/>
4. Ở *Client secret* : “abc”
5. Ở *Authorization URL*: <https://nhatoi.duckdns.org/auth/authorize>
6. Ở *Token URL*: <https://nhatoi.duckdns.org/auth/token>
7. Ở *Configure your client* : nhập “email” và “name”
8. Không chọn *Google to transmit clientID and secret via HTTP basic auth header* rồi nhấn Next
9. Ở *Testing instructions* : nhập gì cũng được rồi chọn Save

Chọn *Develop* ở đầu trang, sau đó ở góc trên bên phải, chọn *Test* để tạo phiên bản thử nghiệm của ứng dụng.

Ta truy cập vào *Project setting*, ở phần *Project ID* ta lấy được khóa : “hasspi-b837f”

Tiếp đến ta cần thông tin về *service_account*.

1. Vào <https://console.cloud.google.com/apis/credentials/serviceaccountkey>
2. Ở *Service account list*, chọn *New service account*
3. Nhập tên ở *Service account name* và nhập ID ở *Service account ID*
4. Từ danh sách *Role*, chọn *Service Accounts > Service Account Token Creator*
5. Ở phần *Key type* chọn *JSON*
6. Nhấn *Create*, file JSON sẽ được download về máy, vậy là ta có *service_account*.

Ta thêm trong file configuration.yaml:

```
google_assistant: !include google_assistant.yaml
```

Tiếp đến ta khởi tạo file google_assistant.yaml và truy cập nó:

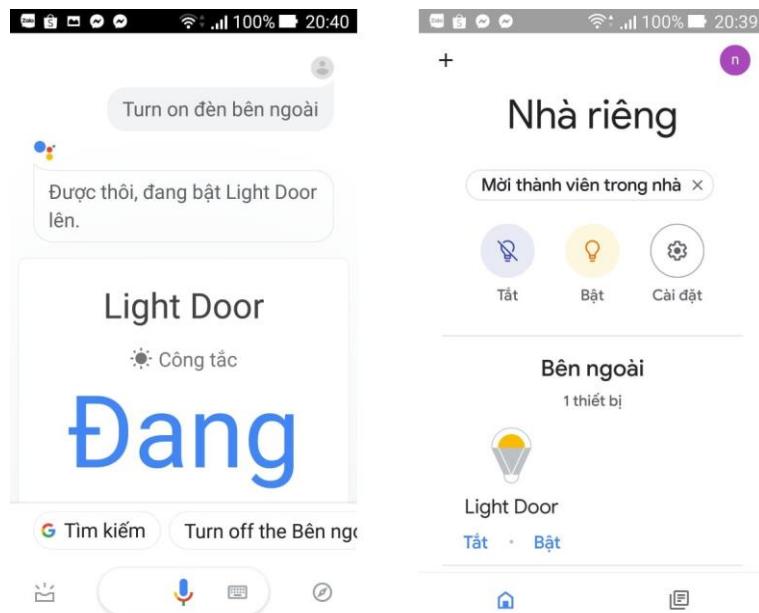
```
touch /usr/share/hassio/homeassistant/google_assistant.yaml  
sudo nano /usr/share/hassio/homeassistant/google_assistant.yaml
```

Ta thêm:

```
project_id: hasspi-b837f  
service_account: !include hasspi-ggasist.json  
report_state: true  
exposed_domains:  
  - switch  
  - light
```

Ở trên điện thoại ta mở ứng dụng Google Home, truy cập *Settings* bấm *Add...*, chọn *Have something already setup?* Và cuối cùng chọn *[test] your app name*. Màn hình sẽ chuyển đến trang đăng nhập của Home Assistant, sau khi đăng nhập ta có thể đặt phòng và đặt biệt hiệu cho thiết bị nếu muốn.

Ta có thành quả:



Hình 94: Sử dụng Google Assistant và Google Home

3. Nhận thông báo thiết bị từ xa bằng HTML5

Nền tảng thông báo HTML5 cho phép ta nhận được thông báo đầy đủ tới Chrome hoặc Firefox, ở bất cứ đâu. HTML5 cũng hỗ trợ Chrome và Firefox trên Android giúp đẩy thông báo lên thiết bị thông minh.

Để tạo thông báo HTML5 ta làm các bước sau:

1. Tạo một dự án mới tại <https://console.cloud.google.com/home/dashboard>, dự án này sẽ được liên kết với *Firebase*.
2. Vào <https://console.cloud.google.com/apis/credentials/domainverification> và xác minh tên miền thông qua *Google Webmaster Central / Search Console*:
 - Nhập domain là <https://nhatoi.duckdns.org/local>. Chọn *HTML file verification* và download file *.html.
 - Tạo thư mục *www* trong thư mục cấu hình Home Assistant , rồi cho file vừa download về vào đó, ta restart Home Assistant.
 - Ta truy cập <https://nhatoi.duckdns.org/local/google123456789.html>. Khi đó ta thấy “google-site-verification: ...” là thành công.
3. Với tên miền được xác minh, hãy truy cập <https://console.firebaseio.google.com> , chọn *Import Google project* và chọn dự án đã tạo.
4. Sau đó, nhấp vào bánh răng ở trên cùng bên trái và chọn *Project settings* rồi chọn *Cloud Messaging*.
5. Tạo một cặp khóa mới trong danh sách *Web configuration* ở cuối trang. Để xem khóa riêng, nhấp vào ba dấu chấm ở bên phải và chọn *Show private key*.

Ta thêm trong file configuration.yaml:

```
notify: !include notify.yaml
```

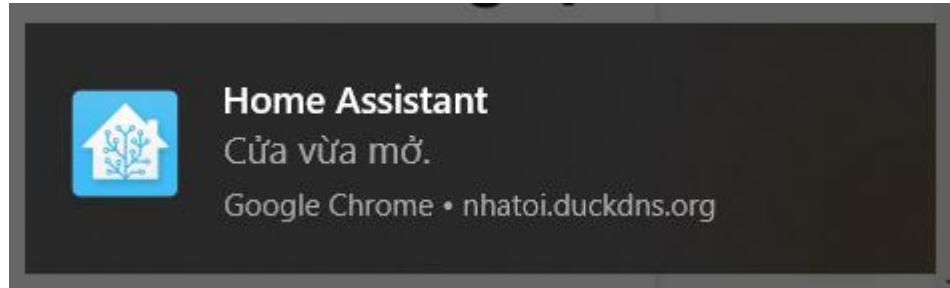
Tiếp đến ta khởi tạo file notify.yaml và truy cập nó:

```
touch /usr/share/hassio/homeassistant/notify.yaml  
sudo nano /usr/share/hassio/homeassistant/notify.yaml
```

Ta thêm:

```
platform: html5  
name: THONG_BAO  
vapid_pub_key:  
BKerHX9zxje7ud305oYkWhtkzs70bF9g4W1fUvfw5gFEZ64LHuH10SUF5876hyFvz1ZKQPq5KShWP0rilaas1  
Gs  
vapid_prv_key: U5pTA0mjQLJg64jLUv_-qaSUGaLcN4YiE20xdCnXIKY  
vapid_email: nsngochang63@gmail.com
```

Thành quả:



Hình 95: Ví dụ về thông báo HTML5

4. Kết nối thiết bị bên ngoài Internet vào Hass với PiVPN

Để Home Assistant có thể quản lý các thiết bị không nằm trong mạng cục bộ của gia đình, ta sử dụng mạng riêng ảo VPN. Điều này làm tăng phạm vi quản lý thiết bị rộng hơn cho người dùng.

Đầu tiên ta tạo một domain cho VPN, truy cập DuckDNS tạo một domain nữa là : vpncuatoi.duckdns.org

Truy cập Terminal của bộ điều khiển trung tâm, ta tạo thư mục duckdns và file duck.sh trong nó:

```
sudo -i  
mkdir duckdns  
cd duckdns  
vi duck.sh
```

Nhập đoạn sau rồi lưu file:

```
echo url="https://www.duckdns.org/update?domains=vpncuatoi&token=f71bd607-458e-4943-b8b1-f6e21fc5d05f&ip=" | curl -k -o ~/duckdns/duck.log -K -
```

Ta phân quyền thực thi file :

```
chmod 700 duck.sh
```

Cuối cùng ta cài vào thiết lập file tự động chạy giúp cập nhật lại DNS của gia đình sau mỗi 5 phút:

```
crontab -e
```

Thêm đoạn sau rồi lưu file:

```
*/5 * * * * ~/duckdns/duck.sh >/dev/null 2>&1
```

Vậy là ta tạo xong tên miền cho PiVPN.

Tiếp đến ta cài đặt PiVPN:

```
curl -L https://install.pivpn.io | bash
```

Trong quá trình cài đặt ta chọn giao thức UDP, Domain là vpncuatoi.duckdns.org và Port là 1197.

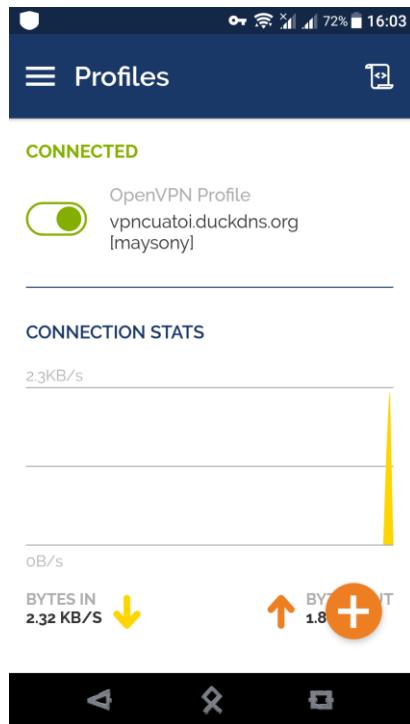
Sau khi cài đặt ta cần mở Port 1197 để truy cập bên ngoài, cách mở port đã được đề cập trang 149.

Vậy là ta cài đặt xong, tiếp đến ta cần tạo file Certificate để truy cập VPN, ta nhập:

```
pivpn add
```

Sau đó ta nhập tên, mật khẩu và thời hạn sử dụng của file Certificate rồi nhấn Enter. Khi đó PiVPN tạo cho ta một file dạng *.opvn, ta đưa nó vào điện thoại thông minh.

Ở điện thoại, ta sử dụng ứng dụng OpenVPN để mở file *.opvn và truy cập VPN ở nhà thông minh, đồng thời sử dụng điện thoại để phát wifi cho các thiết bị, nhờ vậy các thiết bị có thể truy cập được bởi bộ điều khiển trung tâm Home Assistant.



Hình 96: Kết nối VPN trên điện thoại

CHƯƠNG IV: CÁC KẾT QUẢ ĐẠT ĐƯỢC

1. Kết quả thực hiện

Đây là một đề tài nhằm nắm bắt xu hướng của thời đại 4.0, nơi Internet kết nối vạn vật, một ngôi nhà thông minh trong gia đình sẽ là xu thế phát triển trong tương lai, bằng sự tìm tòi và học hỏi từ các thầy cô thì sau thời gian thực hiện, em đã hoàn thành thiết kế và cài đặt hệ thống với dự án thực tế, bước đầu cho kết quả tốt:

- Hoàn thành tất cả các hạng mục của hệ thống đã đề ra
- Các thành phần làm việc với nhau trơn tru, không xảy ra xung đột
- Nói chung, hệ thống làm việc tốt và hiệu quả, các lỗi phát sinh đang được phát hiện và sửa chữa, sản phẩm sẽ ngày một hoàn thiện hơn.

Hiện tại hệ thống đang được lắp đặt và sử dụng tại chính ngôi nhà của em, cho đến thời điểm hiện tại, hệ thống vẫn đang vận hành rất tốt.

2. Ưu và nhược điểm của hệ thống

2.1. Ưu điểm

- Hệ thống hoạt động đúng theo yêu cầu và thiết kế đặt ra ban đầu.
- Hoạt động ổn định hàng ngày, hiếm khi xảy ra lỗi.
- Các lỗi phát sinh đều có thể kiểm soát được.
- Tất cả các thiết bị được tập trung điều khiển trên cùng một giao diện.
- Khả năng mở rộng hệ thống cao, có tính ứng dụng cao trong cuộc sống.
- Giá thành vừa phải.
- Tận dụng các thiết bị điện tử sẵn có.
- Lắp đặt thiết bị dễ dàng, gọn gàng, tương thích với mọi ngôi nhà dù xây mới hay đang sử dụng.
- Không can thiệp vào hệ thống điện sẵn có không phải đục sửa với ngôi nhà đang sử dụng.
- Quản lý đơn giản, không phức tạp.
- Dễ dàng sửa chữa thiết bị.

2.2 Nhược điểm

- Đa phần các thiết bị đều kết nối qua mạng không dây nên đôi khi còn thiếu sự ổn định.
- Camera hiện tại chỉ có thể kết nối trong mạng nội bộ hệ thống mà không thể từ mạng bên ngoài, do máy chủ không tìm được địa chỉ IP của Camera khi nó kết nối vào máy chủ từ bên ngoài bằng VPN.
- Cấu hình ban đầu cho hệ thống để triển khai ở các hộ gia đình vẫn còn phức tạp, chưa thể đóng gói các cấu hình để triển khai dễ dàng.

3. Khả năng sử dụng, so sánh các sản phẩm cùng loại

Hệ thống có khả năng sử dụng tốt trong gia đình hàng ngày do có tính tiện dụng, không quá cầu kì, lắp đặt đơn giản và dễ sửa chữa thay thế linh kiện.

Tất cả các thiết bị thông minh được điều khiển trên cùng một giao diện, đem đến sự tiện dụng, đơn giản và tối ưu hơn so với các thiết bị thông minh bán ngoài thị trường phải điều khiển riêng lẻ.

Tổng giá thành làm ra sản phẩm có giá cả phải chăng, khi so sánh các sản phẩm cùng loại có phần rẻ hơn đồng thời dễ dàng tự sửa chữa và thay thế linh kiện. Ví dụ như đèn thông minh, trên thị trường giá giao động từ 300.000đ tới 2.000.000đ, nhưng nếu tự làm, chi phí thành phẩm chỉ gần 200.000đ. Nếu ta sản xuất hàng loạt thì chi phí giảm xuống chỉ khoảng 60.000đ cho hộp điều khiển thiết bị điện với đèn thường sẵn có. Còn hộp điều khiển hồng ngoại giá thành chỉ khoảng 50.000đ, do vậy có khả năng thương mại hóa và phổ biến rộng trong cộng đồng.

Đặc biệt với Camera, ngoài giá thành rẻ và dễ thay thế linh kiện nó còn tăng tính bảo mật so với sản phẩm bên ngoài, do hình ảnh quay được không bị truyền lên server của nhà sản xuất khi sử dụng.

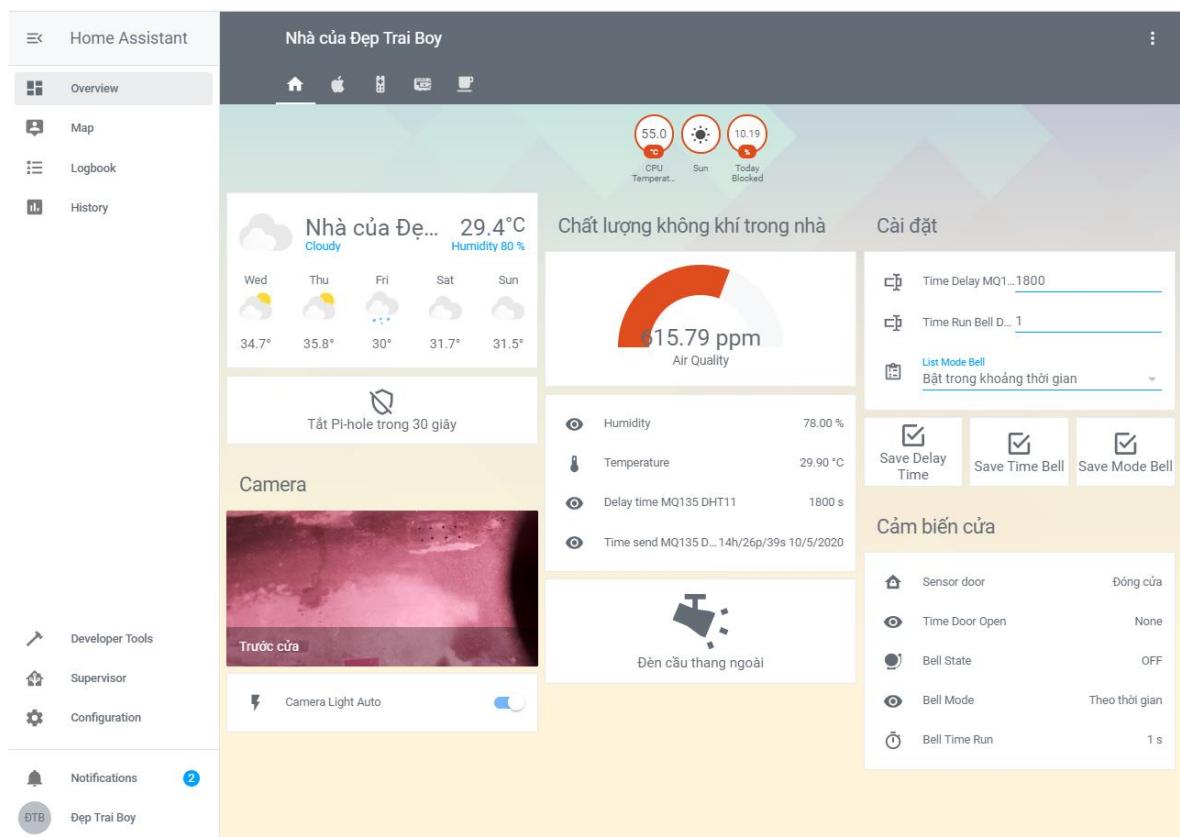
4. Kết luận

Nói chung, hệ thống hoạt động ổn định và hiệu quả, hiếm khi phát sinh lỗi, tất cả các hạng mục của hệ thống đều ra ban đầu đã được hoàn thành. Tuy nhiên, do thời gian có hạn và không đủ chi phí nên em chưa thể phát triển thêm nhiều các thiết bị để kết nối với nhà thông minh hơn, đồng thời hệ thống vẫn có một số nhược điểm còn tồn đọng.

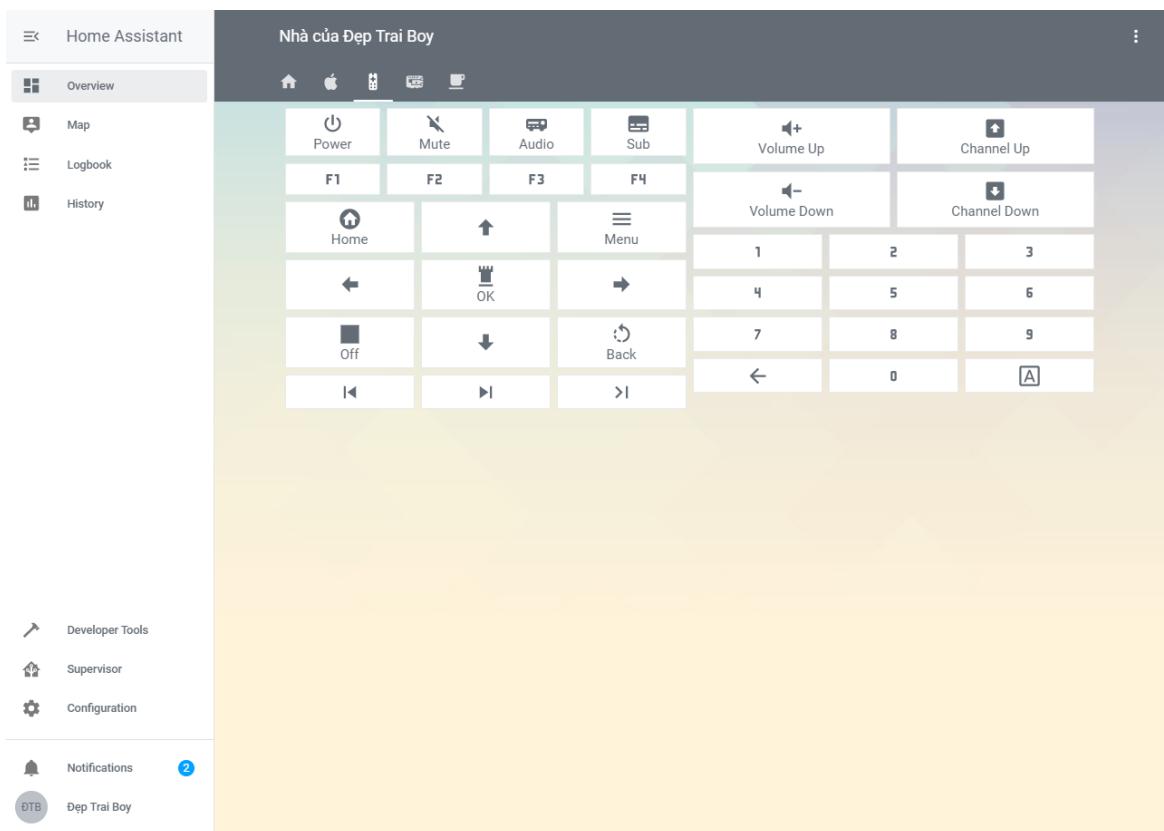
Để nhằm phát triển dự án sau này, trong tương lai em sẽ cố gắng tìm hiểu để khắc phục nhược điểm còn sót lại đồng thời phát triển thêm nhiều thiết bị và các tính năng mới nhằm giúp cho dự án nhà thông minh được tốt và hoàn thiện hơn.

PHỤ LỤC

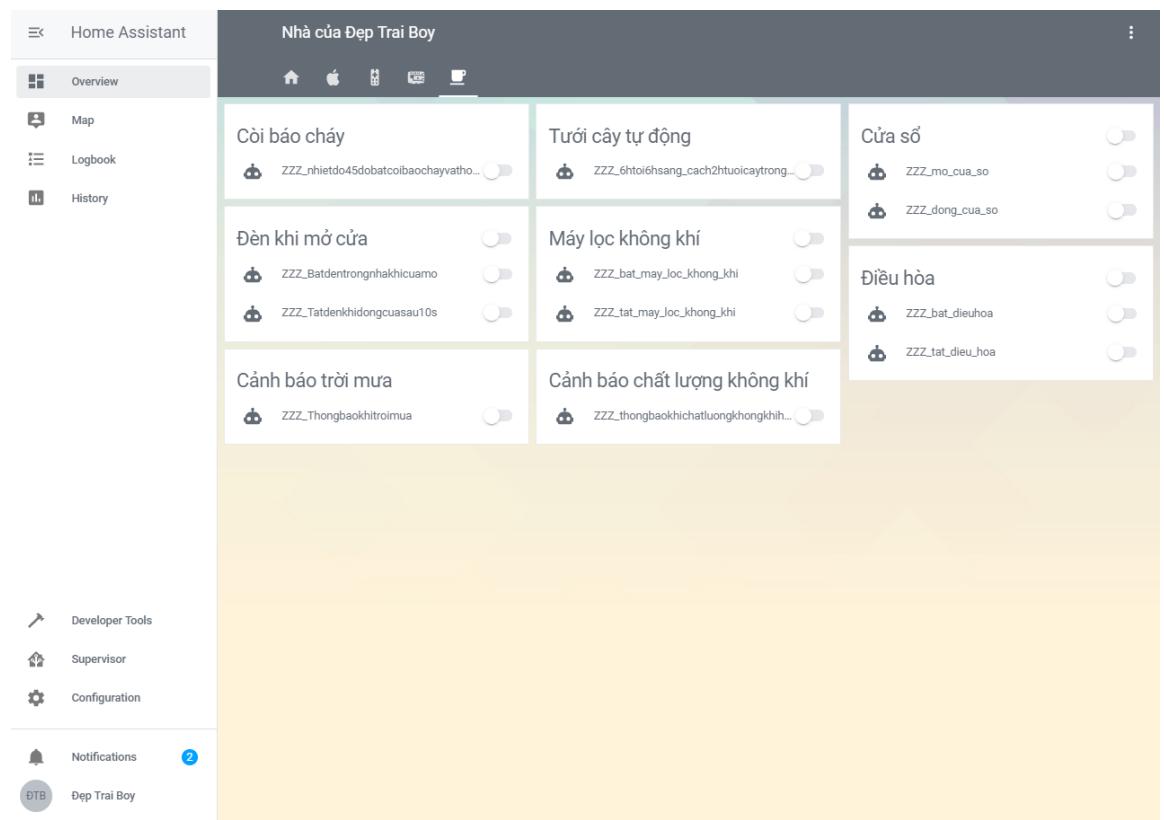
Giao diện điều khiển nhà thông minh:



Hình 97: Giao diện trong nhà



Hình 98: Giao diện điều khiển TV



Hình 99: Giao diện điều khiển tự động hóa

TÀI LIỆU THAM KHẢO

1. Gareth Halfacree, The Official Raspberry Pi Beginner's Guide, 3rd Edition, MagPi, 2019
2. Richard Smedley, Command Line 2, 2nd Edition, MagPi, 2019
3. Phil King, The Camera Module Guide, MagPi, 2017
4. Calin Crisan, MotionEye On Raspbian
URL: <https://github.com/ccrisan/motioneye/wiki/Install-On-Raspbian>
5. Eclipse Paho, MQTT Python client library
URL: <https://pypi.org/project/paho-mqtt/>
6. Home Assistant Documentation
URL: <https://www.home-assistant.io/docs/>
7. Home Assistant Integrations
URL: <https://www.home-assistant.io/integrations/>
8. Arduino Client for MQTT
URL: <https://github.com/knolleary/pubsubclient/>
9. NTPtime library for ESP8266
URL: <https://github.com/SensorsIot/NTPtimeESP/>
10. IR Remote for ESP8266
URL: <https://github.com/crankyoldgit/IRremoteESP8266>
11. IR Remote and Receiver on an Arduino
URL: <https://www.circuitbasics.com/arduino-ir-remote-receiver-tutorial/>
12. DuckDNS for Home Assistant
URL: <https://github.com/home-assistant/hassio-addons/tree/master/duckdns/>

13. Mosquitto for Home Assistant

URL: <https://github.com/home-assistant/hassio-addons/tree/master/mosquitto>

14. Domain DuckDNS for PiVPN

URL: http://www.hagensieker.com/blog/page/?post_id=51&title=how-to-make-your-own-vpn-server